Journal of Computational Physics 229 (2010) 2314–2338

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

SVD–GFD scheme to simulate complex moving body problems in 3D space

X.Y. Wang, P. Yu, K.S. Yeo *, B.C. Khoo

Department of Mechanical Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260, Singapore

ARTICLE INFO

Article history:
Received 19 December 2008
Received in revised form 22 September 2009
Accepted 30 November 2009
Available online 4 December 2009

MSC:
65M06
76D05

Keywords:
Incompressible Navier–Stokes
Moving body
Hybrid meshfree-and-Cartesian grid
Generalized finite difference
Flapping wings

ABSTRACT

The present paper presents a hybrid meshfree-and-Cartesian grid method for simulating moving body incompressible viscous flow problems in 3D space. The method combines the merits of cost-efficient and accurate conventional finite difference approximations on Cartesian grids with the geometric freedom of generalized finite difference (GFD) approximations on meshfree grids. Error minimization in GFD is carried out by singular value decomposition (SVD). The Arbitrary Lagrangian–Eulerian (ALE) form of the Navier–Stokes equations on convecting nodes is integrated by a fractional-step projection method. The present hybrid grid method employs a relatively simple mode of nodal administration. Nevertheless, it has the geometrical flexibility of unstructured mesh-based finite-volume and finite element methods. Boundary conditions are precisely implemented on boundary nodes without interpolation. The present scheme is validated by a moving patch consistency test as well as against published results for 3D moving body problems. Finally, the method is applied on low-Reynolds number flapping wing applications, where large boundary motions are involved. The present study demonstrates the potential of the present hybrid meshfree-and-Cartesian grid scheme for solving complex moving body problems in 3D.

1. Introduction

Flows involving moving bodies/boundaries are frequently encountered in engineering applications and everyday living. Examples are flows in turbomachines, artificial heart valves and extrusion processes. Further examples include flows associated with store separation problems during flight and self-propulsion problems in fish swimming and bird/insect flight etc. These are unsteady flows in which the fluid boundaries are constantly changing, either in a prescribed manner or in a manner determined by forces of interaction between the phases. The subject has attracted considerable attention over the years, and important contributions have been made both in methods and applications. Three main groups of methods are currently available for dealing with moving boundary problems. They are the mesh-based finite-element and finite-volume methods, Cartesian grid methods and the overset or chimera grid methods.

Mesh-based finite element methods (FEM) and finite-volume methods (FVM) have become highly refined and well developed over the years into powerful and popular numerical tools for the solution of a wide range of engineering problems. They have also been variously adapted for the solution of moving boundary flow problems. The main limitation for conventional FEM and FVM in moving body/boundary flow problems is the need to constantly regenerate the mesh to accommodate the changing solution domain. This inevitably leads to increased costs in mesh administration and data interpolation, and a possible increase in numerical errors. A number of numerical approaches have been proposed to improve the efficiency and

* Corresponding author. Tel.: +65 6874 2246; fax: +65 6779 1495.
E-mail address: mpeyeoks@nus.edu.sg (K.S. Yeo).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.11.037

accuracy, and only a few will be mentioned here. To avoid regenerating the mesh, Lilek et al. [1,2] have developed a multi-block FV method based on sliding mesh to solve moving boundary problem. A FE formulation based on the P2P1 Galerkin method and ALE technique has also been developed by Hu and his co-workers [3,4] and recently been improved by Choi [5] and Hu et al. [6]. The method has also been used in combination with an adaptive mesh refinement technique and applied to simulate the motion of a large number of particles in fluids [4–6]. Another recent innovation of conventional FEM to deal with problems with moving boundaries is the particle finite element method (PFEM) proposed by Onate et al. [7].

The best known Cartesian grid method for moving boundary problem is the immersed boundary method (IBM) of Peskin [8]. The IBM treats the boundary of an immersed object as a set of Lagrangian fluid particles amid a background of fixed Cartesian grid nodes. The key merits of Peskin's IBM are the simplicity of the grid system and the resultant efficiency in computation. But the main drawback is that the piecewise continuous solution across the immersed boundary is smeared by the distribution of the singular forces over several grid nodes, leading to reduced spatial resolution and accuracy near the boundary (frequently of first order). To overcome the limitation of the IBM, the immersed interface method (IIM) was proposed by Leveque and Li [9]. This method modifies the governing equations at the immersed boundary nodes by adding forcing functions constructed to enforce a set of appropriate jump conditions at the interface. The method has been claimed to maintain sharp-interfaces and second-order accuracy. Li and Lai [10], Lai and Peskin [11], and Xu et al. [12] have implemented the IIM for 2D flows and achieved second-order spatial accuracy. Jump conditions for interfaces in 3D have also been derived by Xu and Wang [13,14]. New immersed boundary methods that can avoid the problem of smeared interfaces have been developed by several authors. A typical example is the sharp interface immersed boundary method proposed by Mittal et al. [15]. This method applies a ghost cell technique to satisfy the boundary conditions on the immersed boundary. The method has been developed to simulate the compressible flows with complex shaped stationary immersed boundary [16] and the fluid-structure interaction problems [17]. In other newly developed immersed boundary methods, the interface is kept sharp by handling the boundary body force with projection-based method which satisfies the non-slip boundary condition [18] or exactly guarantees conservation of momentum [19].

Other Cartesian grid methods include the cut-cell methods. In this method, the fluxes or field data at intersection between the sharp boundaries and the Cartesian base mesh are determined by interpolating functions. The cut-cell method has been integrated into both finite difference (FD) [20] and FV [21,22] methods to solve both stationary and moving boundary problems. A similar method that uses interpolating functions to reconstruct solution near the boundaries is the Hybrid Cartesian/immersed boundary method [23,24]. It is worth noting that Marella et al. [25] has developed a Cartesian grid method in a finite difference framework that is much simpler than the cut-cell approach. In this method, the immersed boundary is represented by level-sets and no source term is applied to include boundary effect. This method is thus naturally suitable for moving boundary problems and can be easily apply to other types of flow problems including multiphase flows [26] and phase change [27]. More recently, this method has been further enhanced by adopting adaptive mesh technique as well as parallelization technique [28].

The overset-grid method, pioneered by Benek et al. [29], is based on a collection of structured component grids. Each immersed body is enveloped within its own separate structured grid component. Component grids are connected by interpolation conditions. The use of structured components contributes to achieving good computational efficiency. However, a major problem encountered in the simulations, especially for 3D problems, is that global conservation is not easy to maintain [30]. To ensure global mass conservation in each subdomain, Zang and Street [31] implemented a mass imbalance correction on the interpolated velocity field in their overset-grid method for solving the 3D, unsteady, Navier–Stokes equations. A mass-flux based interpolation (MFBI) algorithm has also been introduced by Tang et al. [32] to ensure global mass conservation in their overset-grid scheme. The MFBI algorithm avoids explicit correction to the interpolated velocity field, and thus simplifies the implementation of the method in 3D.

Three-dimensional (3D) moving body/boundary flow problems are solved by a hybrid meshfree-and-Cartesian grid method in the present paper. In this approach, an enveloping cloud of meshfree nodes discretizes the moving body/boundary and its immediate fluid neighbourhood. The moving body/boundary and its enveloping meshfree nodal cloud are superposed on a background space of Cartesian grid points. Spatial derivatives at Cartesian grids are carried out by conventional finite difference (FD) wherever applicable, while a generalized finite difference (GFD) method is applied at all meshfree nodes. The meshfree method on irregular nodal grids was originally proposed by Liszka and Orkisz [33] in applied mechanics, and further developed by Liszka [34], Liszka et al. [35], Duarte and Oden [36] and others. Ding et al. [37,38] subsequently extended the method to solve incompressible fluid flow problems in the streamfunction-vorticity formulation. The method was more recently developed by Chew et al. [39] to simulate moving body flow problems on hybrid meshfree-and-Cartesian grids, whereby flow at moving meshfree nodes is governed by a convective form of the Navier–Stokes equations.

Considerably more grid points or nodes are usually needed to define the spatial derivatives on an irregular grid using GFD; 9 nodes (the minimal set) are needed to define second-order partial space derivatives to second-order accuracy in 2D space. Least square approximation on an over-determined set of nodal data (>9) is usually used to overcome potential ill-conditioning arising from poor nodal configuration. GFD operations are more costly than conventional FD ones, and their application are thus confined only to complex boundary regions by employing a hybrid meshfree-and-Cartesian grid system. The hybrid grid exploits the geometric freedom of meshfree nodes to accurately resolve complex surfaces, while simultaneously allows one to take advantage of the high computational efficiency and accuracy of FD approximations in the bulk of the flow domain. Since only nodal indices or position data are needed in difference approximations, the hybrid grid difference scheme needs to maintain only a relatively simple geometric data base.

In the current study, Chew et al. [39]'s hybrid grid scheme is further developed for moving body/boundary problems in 3D space. A singular value decomposition (SVD) form of the GFD approximation [40,41] is used here. While somewhat more expensive than the regular least square approximation, SVD–GFD has been shown to be more robust in applications [40], especially those involving close interactions between moving bodies. As GFD method is based on the Taylor series expansion, it could be rendered to higher order if necessary by the inclusion of more support nodes. The present SVD–GFD method offers the following advantages by combining the meshfree and Cartesian approaches: (1) good efficiency as the bulk computation is based on standard 7-point finite difference scheme, (2) precise implementation of boundary conditions, (3) grid flexibility including the ability to accommodate complex geometry with good grid resolution and grid orthogonality at boundaries, (4) grid administration is relatively simple, and (5) grid/mesh regeneration frequently used in mesh-based methods is avoided and data interpolation is kept at a very minimal level.

The present study focuses on flow driven by solid body/boundary with prescribed motion. The case of fluid-fluid boundary, which is interactive in character, is not considered here. A moving nodal patch test shows that the mixed Lagrangian–Eulerian formulation for convecting nodes in flow produces results that are accurate and fully consistent with those from stationary Cartesian grid computation. The method is validated against published results for 3D moving body problems. Finally, the method is demonstrated on low-Reynolds number ($Re < 200$) flapping wing applications, where large and complex boundary motions are involved.

2. Spatial discretization on hybrid grid

The discretization of fluid space comprises a background of Cartesian grid nodes and clouds or patches of meshfree grids/nodes around immersed bodies/boundaries (Fig. 1). Cartesian nodes that are overlapped by a solid body are not involved in the flow computation. Cartesian nodes that are overlapped by a meshfree nodal cloud may either be included or excluded from flow computation. It is customary to exclude them to reduce unnecessary computations as well as to maintain good nodal quality. Further discussion on nodal grid organization in the 3D context is given in Section 4.

The standard 7-point central finite difference scheme is applied at all Cartesian nodes that do not have meshfree node(s) within its closed $[-\Delta x, \Delta x] \times [-\Delta y, \Delta y] \times [-\Delta z, \Delta z]$ neighbourhood. These constitute the bulk of the computational nodes in many applications. All other nodes are subject to the SVD-based generalized finite difference (GFD) treatment [41], which is briefly reviewed below for completeness.

The generalized finite difference (GFD) method is based on the Taylor series expansion and singular value decomposition (SVD) is used to obtain the pseudo-inverse of the resulting approximation matrix. For a function $f(\mathbf{x})$ of the 3D coordinate variable $\mathbf{x} = (x, y, z)$, the value of the function at $\mathbf{x}_1 = \mathbf{x}_0 + \Delta\mathbf{x}_1$ is given in terms of the derivative values of the function at the reference grid or node point $\mathbf{x}_0$ to order m by:

$$f(\mathbf{x}_1) = f(\mathbf{x}_0) + \sum_{j=1}^{m-1} \frac{1}{j!} (\Delta\mathbf{x}_1 \cdot \nabla_{\mathbf{x}_0})^j f(\mathbf{x}_0) + O(|\Delta\mathbf{x}_1|^m), \tag{1}$$

If the values $f(\mathbf{x}_i) = f_i (i = 0, 1, \ldots, n)$ of the function are known at a number of neighbouring points $\mathbf{x}_i = \mathbf{x}_0 + \Delta\mathbf{x}_i$, one may truncate the Taylor series Eq. (1) appropriately and approximate the derivatives $\nabla^j_{\mathbf{x}_0} f(\mathbf{x}_0)$ of the function at the reference node $\mathbf{x}_0$ by solving a system of linear equations. For such an approximation, one will typically select the points

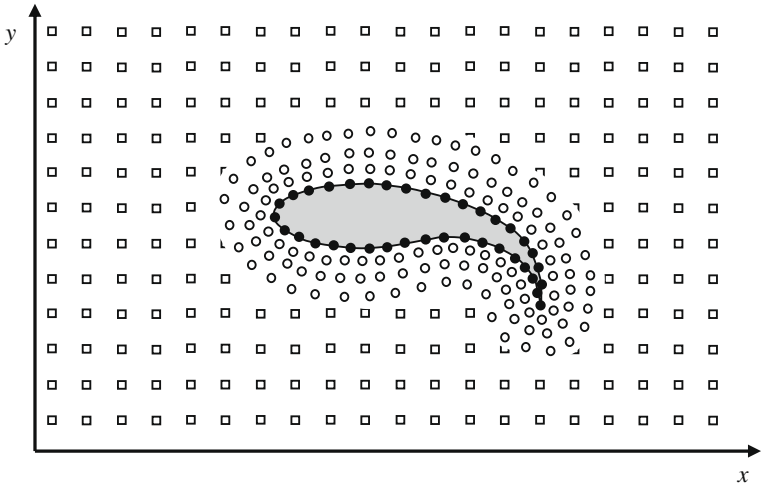


Fig. 1. Schematics of the hybrid meshfree-and-Cartesian grid: ● meshfree boundary nodes defining an immersed body; ○ meshfree cloud of nodes enveloping the body; □ Cartesian background nodes.

$\mathbf{x}_i (i = 1, \ldots, n)$, which are in the immediate or near-neighbourhood of $\mathbf{x}_0$. These *neighbours* or *support nodes* $\mathbf{x}_i$ are usually taken to be points within a prescribed distance d_0 from the reference node, i.e. $|\mathbf{x}_i - \mathbf{x}_0| = |\Delta\mathbf{x}_i| \leqslant d_0$. When the Taylor series Eq. (1) is truncated after the third-order derivative (i.e. $m = 4$), we obtain the following linear system,

$$\Delta\mathbf{f}_{n\times 1} = [S]_{n\times 19}\partial\mathbf{f}_{19\times 1}, \tag{2}$$

which relates the derivatives $\partial\mathbf{f}_{19\times 1} = \left(\partial_x, \partial_y, \partial_z, \partial_x^2, \partial_x\partial_y, \ldots, \partial_y^3, \partial_z^3\right)^T f|_{\mathbf{x}_0}$ of f at $\mathbf{x}_0$ to the functional values of f at the reference node $\mathbf{x}_0$ and the n support nodes. Here $\Delta\mathbf{f}_{n\times 1} = (f_1 - f_0, f_2 - f_0, \ldots, f_n - f_0)^T$, and

$$[S]_{n\times 19} = \begin{bmatrix} \Delta x_1 & \Delta y_1 & \Delta z_1 & 0.5\Delta x_1^2 & 0.5\Delta y_1^2 & 0.5\Delta z_1^2 & \ldots \\ \Delta x_2 & \Delta y_2 & \Delta z_2 & 0.5\Delta x_2^2 & 0.5\Delta y_2^2 & 0.5\Delta z_2^2 & \\ \Delta x_3 & \Delta y_3 & \Delta z_3 & 0.5\Delta x_3^2 & 0.5\Delta y_3^2 & 0.5\Delta z_3^2 & \\ \ldots\ldots & & & & & & \\ \Delta x_n & \Delta y_n & \Delta z_n & 0.5\Delta x_n^2 & 0.5\Delta y_n^2 & 0.5\Delta z_n^2 & \end{bmatrix}. \tag{3}$$

The matrix $[S]_{n\times 19}$ contains only information about the positions of the n support nodes $\mathbf{x}_i (i = 1, 2, \ldots, n)$ relative to the reference or central node $\mathbf{x}_0 = (x_0, y_0, z_0)$. Eq. (2) is exactly closed for $n = 19$. As with most unstructured grid/mesh schemes for fluid flow computation, the matrices obtained via the application of GFD are not symmetric. The linear system for $n = 19$ tends to be ill-conditioned. Ill-conditioning arises from poor spatial arrangement of the support nodes for causes ranging from extremely close separation between some nodes to highly irregular spread of nodes around the reference node. This difficulty can be overcome by including additional support nodes to yield an over-determined algebraic ($n > 19$) system in (2); and a best-approximation solution for (2) based on the minimization of the L_2-norm $\|\mathbf{E}\|_2$ of the residual error vector $\mathbf{E} = \Delta\mathbf{f}_{n\times 1} - [S]_{n\times 19}\partial\mathbf{f}_{19\times 1}$ is then sought. The Singular Value Decomposition (SVD) approach to minimization of the L_2 error norm is adopted here. The SVD scheme is more robust than the conventional normal equation approach in being more stable and accurate and will yield a solution (one with minimum norm) even when the system is under-determined ($n < 19$). Moreover, it also permits regularization in term of omission of contributions from the very small eigenvalues [42], and formal accuracy is not compromised if $n \geqslant 19$.

The components of the residual vector $\mathbf{E}$ represent the errors of approximation at the various support nodes. A distance-based weighting that gives greater weight to the errors at nodes nearer to the reference node is usually applied to enhance the accuracy of derivative approximation. This takes the form a diagonal matrix $[W_n]$ applied to $\mathbf{E}$; where the ith component of $[W_n]$ is a function of the normalized distance (i.e. $|\mathbf{x}_i - \mathbf{x}_0|/d_0$) from the reference node. The SVD determined solution for derivatives at the reference node thus has the form of

$$\partial\mathbf{f}_{19\times 1} = [W_n S]^{p-}_{n\times 19}[W_n]\Delta\mathbf{f}_{n\times 1}, \tag{4}$$

where $[W_n]$ is the weighting matrix and $[W_n S]^{p-}_{n\times 19}$ denotes the pseudo-inverse of $[W_n S]_{n\times 19}$ solved by an SVD algorithm. Some of the applicable weighting functions have been presented in Wang et al. [41].

3. ALE form of Navier–Stokes equations and projection method

Incompressible three-dimensional fluid flow is governed here by the Navier Stokes equation in a mixed Lagrangian–Eulerian form [43]. The Navier–Stokes and continuity equation at a convecting node, say b, are given by

$$\left.\frac{\partial\mathbf{u}}{\partial t}\right|_b = \mathbf{S}(\mathbf{x}_b, t) \equiv -(\mathbf{u} - \mathbf{u}_b)\cdot\nabla\mathbf{u} - \nabla p + \frac{1}{Re}\nabla^2\mathbf{u}, \tag{5}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{6}$$

where $\mathbf{u}(\mathbf{x}, t)$ is the non-dimensional Eulerian velocity field and $\mathbf{u}_b$ is the convection velocity of the node b at its current position $\mathbf{x}_b(t)$. $(\partial\mathbf{u}/\partial t)_b$ denotes time derivative of $\mathbf{u}$ following the convecting node. For a stationary node, $(\partial\mathbf{u}/\partial t)_b$ reduces to the customary partial time derivative of $\mathbf{u}(\mathbf{x}, t)$. $Re = UL/\nu$ is the Reynolds number where U and L are the velocity and length scales, respectively appropriate to the problem, and υ is the kinematic viscosity of the fluid.

As mentioned above, spatial derivatives of the present form of Navier–Stokes equations are carried out by a conventional FD at Cartesian grids while a GFD at meshfree nodes. GFD, as well as conventional FD, is not intrinsically conservative. However, for incompressible viscous flow, where the solution is necessarily smooth, conservation errors are part of the formal discretization errors, which may be kept as small as desirable through grid refinement.

ALE implementation must satisfy certain generalized conservation conditions [44] when it is applied to mesh/volume based schemes such as FE and FV. This is because mesh deformation with time may cause mass and momentum to be convected in and out of a cell. These conservation conditions are not applicable to nodal-base schemes such as the present – in which the solution at the node represents the changing Eulerian field as perceived by the observer on the moving node. The meshfree nodes represent moving sensors of the underlying Eulerian flow field.

The above issues on conservation have been discussed at length in Ang et al. [40], where conservation errors in mass and momentum were examined for problems involving the very close interaction of moving bodies. The results demonstrated

the conservation errors for mass and momentum are quite small (less than 0.005% and 0.01%, respectively for the 2D problem investigated) despite the non-conservative nature of the GFD discretization scheme.

The projection method and its variants are widely used in the literatures [45–48]. A second-order implicit projection method, based on a fractional-step Crank–Nicolson scheme is applied here:

$$\left.\frac{\mathbf{u}^* - u^n}{\Delta t}\right|_b = -\alpha\nabla q - \frac{1}{2}\left[((\mathbf{u}-\mathbf{u}_b)\cdot\nabla\mathbf{u})^n + ((\mathbf{u}-\mathbf{u}_b)\cdot\nabla\mathbf{u})^{n+1}\right] + \frac{1}{2Re}\nabla^2(\mathbf{u}^n + \mathbf{u}^*), \tag{7}$$

$$\left.\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t}\right|_b = -\nabla\Phi^{n+1}, \tag{8}$$

where q is a suitable approximation for the pressure field $p^{n+1/2} = (p^n + p^{n+1})/2$ and Φ^{n+1} is an auxiliary field variable. The requirement that $\mathbf{u}^{n+1}$ be divergence-free leads to the following Poisson equation for Φ^{n+1}

$$\nabla^2\Phi^{n+1} = \frac{1}{\Delta t}\nabla\cdot\mathbf{u}^*, \tag{9}$$

which is solved subject to the Neumann boundary condition $(\mathbf{n}\cdot\nabla\Phi^{n+1})_{\partial\Omega} = 0$, where $\mathbf{n}$ denotes the normal to the boundary. Eq. (8) suggests that a suitable boundary condition for the intermediate velocity field is,

$$\mathbf{u}^*|_{\partial\Omega} = \mathbf{u}^{n+1}|_{\partial\Omega}. \tag{10}$$

This boundary condition is easy to apply, especially for flow problems with prescribed boundary velocity. It is also consistent with the interpretation of $\mathbf{u}^*$ as an intermediate approximation to the velocity field $\mathbf{u}^{n+1}$ at time level $n+1$. The updating of the velocity and pressure fields follows

$$\mathbf{u}^{n+1} = \mathbf{u}^* - (\Delta t)\nabla\Phi^{n+1}, \tag{11}$$

$$p^{n+1/2} = \alpha q + \Phi^{n+1} - \frac{\Delta t}{2Re}\nabla^2\Phi^{n+1}. \tag{12}$$

It is important to interpret the terms in the above Eqs. (7)–(12) correctly for convecting nodes. In particular, the intermediate velocity $\mathbf{u}^*$ and all terms denoted to be at time level $n+1$ should be evaluated based on the position of the nodes at that time.

When $\alpha = 0$, we have a pressure-free formulation in that no pressure is involved in the computation of the intermediate velocity field. This formulation is equivalent to *Pm* III in Brown et al. [48] and similar to the scheme of Kim and Moin [47]. Intermediate schemes can be obtained for value of α between 0 and 1. These schemes are second-order when appropriately implemented – this generally requires careful implementation of the velocity boundary condition Eq. (10) to ensure that $\mathbf{u}^*$ is indeed a good estimation of the velocity field $\mathbf{u}^{n+1}$. When $\alpha = 1$, we recover the *Pm* II formulation described by Brown et al. [48], which is second-order accurate for both velocity and pressure fields.

An implementation with $\alpha = 1$ (*Pm* II formulation) is adopted here in which Eqs. (7), (9), (11) and (12) are solved together *iteratively* at every time level – this differs from the usual semi-implicit implementation. The function q in Eqs. (7) and (12) is initialized with $p^{n-1/2}$ at the start of iteration at time level $n+1$; thereafter the latest estimate for $p^{n+1/2}$ from Eq. (12) is used during iteration. The intermediate velocity field $\mathbf{u}^*$ is similarly updated with the latest estimate of $\mathbf{u}^{n+1}$ from Eq. (11) during iteration. At convergence $\Phi^{n+1} \to 0$, $\mathbf{u}^* \to \mathbf{u}^{n+1}$, and $q \to p^{n+1/2}$.

4. Aspects of numerical implementation

Since GFD allows spatial discretization to be performed on arbitrary nodal configurations, all flow computations are carried out within a single global Cartesian reference frame irrespective of the complexity of computational domain. For overset/chimera grid methods, it customary to use a different coordinate representation for each component grids. This section explains key aspects of nodal administration employed in this work.

4.1. Grid generation – motion of convecting nodes

Domains with complex immersed bodies and/or boundaries that move with prescribed motion may be easily handled by the present GFD method. The first step of grid generation is to define the immersed bodies or fluid-solid interfaces by a set of meshfree boundary-fitted nodes. This is best accomplished by *triangulating* the bodies/surfaces using a conventional finite element mesh generator; although for simple analytically-defined body/surface alternative methods of generating the boundary nodes may be devised. The triangulation of the surface also facilitates the computation of the surface normal $\mathbf{n}$ at the boundary nodes. This is needed for the computation of the normal derivative $\mathbf{n}\cdot\nabla$ during flow simulation (Section 3) and in the next step of grid generation described below. The normal at a boundary node is simply approximated here by taking the *simple* average of the unit normals at the triangular faces surrounding the node, following the studies of Gouraud [49], and Thürmer and Wüthrich [50]. The mesh information derived from surface triangulation is not used beyond this point.

Having thus discretized and defined the immersed objects or surfaces by their sets of meshfree boundary nodes, it remains to discretize the surrounding fluid domain. In theory, the GFD scheme can deal with any kind of unstructured nodal

distribution. For simplicity, two types of grid are described here. The first is the *hybrid* grid shown in Fig. 1. Here an immersed body/surface is enveloped over by several layers of meshfree nodes, which are typically generated along the outward normals **n** emanating from the surface nodes. Typically four to six enveloping layers of meshfree nodes are generated around an immersed body. The boundary and enveloping meshfree nodes are then superposed on a coarser background of Cartesian nodes. Moving away from the surface, the meshfree nodes are usually seeded with increasing nodal intervals, so as to give good resolution to viscous effects close to the surface and to offer compatible interval matching with the background Cartesian nodes for the outermost layer of meshfree nodes. Special attention is needed for node seeding at sharp convex and concave corners in this gridding approach to ensure uniform seeding. It is worth mentioning that this method also can be used to generate the meshfree nodes around a deformable body [51]. When a body deforms, the existing meshfree nodes automatically move/re-adjust with the changing orientation of the local surface normal along which they had been spawn. The numerical process remains the same as that for a moving rigid non-deforming body. The grid remains orthogonal or nearly so on smooth surfaces despite deformation. The second type of hybrid grid is shown in Fig. 2. In this case, the body is enveloped by a Cartesian-type nodal patch/cloud, which may be derived from a refinement of Cartesian background grid prior to the start of simulation. Note that we can only apply one layer of meshfree nodes, i.e., the surface nodes that define or specify the body geometry. This type of grid can be regarded as a special case of the first or second type of grid. In this situation, the present scheme can be defaulted to a purely Cartesian scheme.

For both types of grid distribution, a Cartesian background grid is present where inexpensive and accurate conventional finite difference scheme may be suitably applied. For many flow problems, especially those involving an external flow, the GFD-treated meshfree nodes will typically form only a very small fraction (just a few %) of the entire nodal population. Computational efficiency is thereby promoted. The second type of grid may be more efficient than the first because some of the nodes in patch/cloud may also to amendable to conventional finite difference treatment.

When the first type of grid is applied, one may suspect that the present method is similar to the Chimera scheme where the grid is also boundary-fitted. However, as mentioned above, the GFD scheme is very flexible and can deal with any kind of unstructured nodal distribution in theory. In this sense, the present method is different from a Chimera scheme. Besides this, for a Chimera scheme, each grid is associated with its own special reference frame, which is frequently non-Cartesian in order to take advantage of the particular geometry of the enclosed object (such as polar coordinate system for a cylinder). Chimera schemes based on the use of generalized (smooth) coordinate systems are typically not good at handling highly complex geometry. Chimera schemes then employ an interpolation technique to meld the solutions in different grid domains. In the present method, all flow computations are based on a *single global* computational frame. Cartesian nodes that have meshfree node(s) within its closed $[-\Delta x, \Delta x] \times [-\Delta y, \Delta y] \times [-\Delta z, \Delta z]$ neighbourhood are treated by SVD–GFD method. All the nodes are linked together and the solution procedure is performed on Cartesian and meshfree nodes simultaneously. Thus, no interpolation is required.

Hybrid meshfree-and-Cartesian grids are employed in the present study. The enveloping cloud of meshfree nodes convects synchronously with the boundary nodes of the moving body or surface. The motion of the nodes is tracked in Lagrangian manner in accordance with

$$\mathbf{x}_b^{n+1} = \mathbf{x}_b^n + \mathbf{u}_b^{n+1/2}\Delta t, \tag{13}$$

where $\mathbf{x}_b^n$ denotes the spatial position of a boundary or cloud node b at time level n and $\mathbf{u}_b^{n+1/2}$ is the prescribed convecting velocity of the node b at time $t^{n+1/2} = (n + 1/2)\Delta t$. The fluid dynamics at these convecting nodes are governed by the mixed Lagrangian–Eulerian form of the Navier–Stokes Eqs. (5) and (6).

4.2. Organization of nodal search

The implementation of GFD (Section 2) involves search for neighbouring support nodes. The nearest $N \geqslant 19$ nodes are usually selected for this purpose. To facilitate and expedite the search for nearest neighbours, we assign each meshfree node

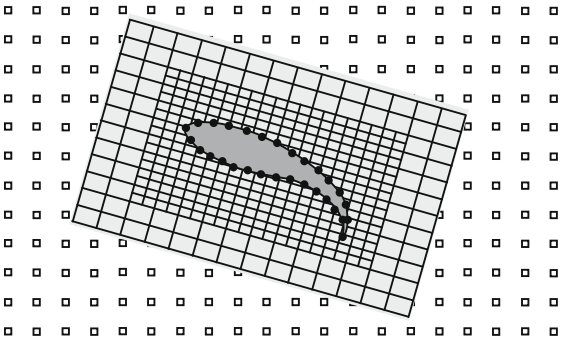


Fig. 2. A Cartesian-based nodal cloud/patch around an immersed body.

(including the boundary nodes) to the Cartesian background cell $(\Delta x, \Delta y, \Delta z)$ that contains it. A meshfree node B at the position $\mathbf{r}_B = (x, y, z)_B$ is contained in a Cartesian cell identified by its nodal label:

$$(i, j, k)_B = (\mathrm{int}(x_B/\Delta x), \mathrm{int}(y_B/\Delta y), \mathrm{int}(z_B/\Delta z)), \tag{14}$$

where 'int' denotes the *integer* part. A list of non-empty Cartesian cells, and the meshfree nodes that they contain, is maintained. The non-empty Cartesian cells are very small in number compared to the total number of Cartesian cells in the computational field. The list of non-empty Cartesian cells and their contents of cloud nodes are updated at every time step in a moving body problem.

Thus to find one or more meshfree nodes (of the nodal cloud) that are closest in distance to a particular Cartesian background node (i, j, k), we need only to restrict the search to meshfree nodes that are contained within the (i, j, k)-Cartesian cell and its neighbouring cells from the maintained list. Similarly, to find a meshfree node that is nearest to a given meshfree node B, we first locate the Cartesian cell $(i, j, k)_B$ (using (14)) that contains B, and then search for the nearest meshfree nodes that are contained within $(i, j, k)_B$-Cartesian cell and its neighbouring cells from the list.

By the above and related considerations, one can greatly reduce the cost needed to find the near neighbours of a given node. This is especially significant for moving body problems, where the nodal configuration is constantly changing.

4.3. Determination of overlapped Cartesian base nodes

In the present grid setup, nodes belonging to the Cartesian background grid are not involved in computation when they are covered by a solid body or medium. These are termed here *overlapped* nodes and assigned the computational status 0. The set of overlapped nodes, also termed exterior nodes in the studies of Chew et al. [39], changes with time because of the motion of the body/boundary. We need to identify these nodes in the course of simulation.

To determine the set of overlapped Cartesian nodes, we only need to consider Cartesian base nodes in the vicinity of the body or boundary; such as the Cartesian nodes within a certain cubic sub-domain containing the body. For a given Cartesian node P, let B denotes the boundary node that is closest in distance to P (see Fig. 3) – the nodal search procedure given in the preceding section may be used to find node B here. Node P is an *overlapped* node if

$$\mathbf{r}_{BP} \cdot \mathbf{n}_B > 0, \tag{15a}$$

$$\mathbf{r}_{BP} = \mathbf{r}_P - \mathbf{r}_B, \tag{15b}$$

where $\mathbf{r}_B$ and $\mathbf{r}_P$ are the position vectors of node B and P respectively, and $\mathbf{n}_B$ is the unit outward normal at node B. Its nodal status is then set to 0.

A thorough search of overlapped nodes with status 0 is carried out during the setup phase of the problem. During simulation, a more limited status-updating procedure may be used however. This is because only Cartesian base nodes that are in the immediate vicinity of the moving boundary are likely to undergo a change of computational status in the course of one time step Δt. Hence we really only need to restrict the checking of the overlap status using criterion (15) to the much smaller set of Cartesian base nodes that are in the vicinity of the *set of boundary nodes*. Again the nodal search scheme described in the preceding section is useful here.

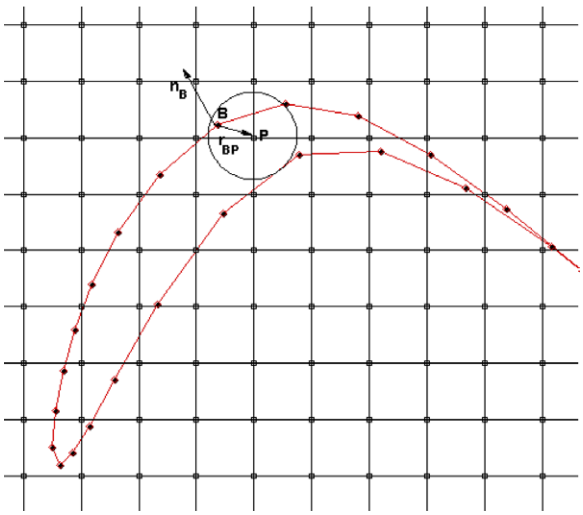


Fig. 3. Identification of overlapped nodes.

Poor meshfree nodal arrangement may arise when two bodies approach each other so closely that their meshfree clouds overlap. This problem has been resolved in Ang et al. [40] by a nodal selection applied at meshfree nodes that simply rejects GFD support nodes that are too close to other support nodes.

4.4. Nodal addition and deletion

One of the strengths of GFD method is the ease with which one may introduce new and delete existing nodes during computational simulation. As a moving body travels across the domain, it will overlap new Cartesian base grid nodes in its path, whose computational status is then changed from active to 0 (overlapped), and uncover Cartesian base nodes in its rear, whose computational status is then updated from 0 to active. The status-updating process during simulation has been explained in the last section. Cartesian base nodes that are thus uncovered by the motion of a body/boundary (see Fig. 4) and thus re-introduced back into computation are termed *fresh* nodes in the current time level.

Nodes that are set to status 0 are excluded from flow computation at the current time level. When a new node B is added or introduced into computation at the current time level n, it cannot immediately partake in the flow computation because the discretization of the dynamic flow Eqs. (7) and (8) requires field data at the preceding time level. The first computational action when introducing a new node is thus to determine the Eulerian field values $(\mathbf{u}, p)_B^n$ at the node at the current time level. This typically involves numerically interpolating the data from neighbouring nodes. Interpolation could make use of the second-order GFD templates already available at this time. Indeed this is the only occasion when data interpolation is invoked in the present numerical scheme. With $(\mathbf{u}, p)_B^n$ thus determined, the new node B can participate fully in the flow computation at the next time level $n+1$.

Fresh nodes are nothing more than newly added nodes in terms of numerical treatment. Their numbers are usually very small in each time level due to the limited distance that the body can travel through in each time step. Consequently, very minimal data interpolation is involved in the application of the present numerical scheme for moving body problems.

The present GFD-based scheme permits easy deletion and insertion of meshfree nodes, which can be exploited for adaptive grid refinement. As a novel possibility, ALE can also be contrived to conglomerate meshfree nodes in areas where finer spatial resolution is desired. Thus in a region of thinning boundary layer, the meshfree nodal layers could be nudged closer towards the solid boundary even as the nodes are convecting with the body.

5. Results and discussion

The present SVD–GFD–ALE numerical scheme is applied to a number of three-dimensional problems with moving boundaries in this section. These include flows within a box, flows induced by the motion of spheres, and flows driven by flapping wings. Hybrid grid consisting of meshfree grid around the body amidst a background of Cartesian grid is used in all cases.

5.1. Order of accuracy: three-dimensional flows in a cubic cavity containing a sphere

In the first example, we consider the steady flow in a cubic cavity driven by the top wall sliding in its own plane. The cavity contains a stationary sphere of diameter D that is 0.5 times the side of the cavity. Simulation of the flow is carried

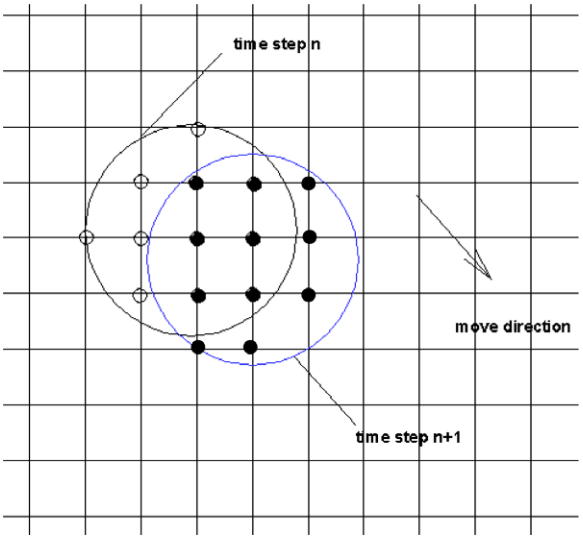


Fig. 4. Open circles ○ denote fresh nodes. They were overlapped by the body at time level n, but are uncovered by the motion of body at time level $n+1$. Filled circles • denotes overlapped or exterior nodes at time level $n+1$.

out at a Reynolds number of 20 (based on D and the sliding wall speed) on six uniform Cartesian grids, 21^3, 31^3, 51^3, 81^3,101^3 and 161^3 to validate the spatial accuracy of the numerical scheme. The same problem was considered by Gilmanov et al. [52]. For all grids, the surface of the sphere is discretized by 3302 surface nodes (6604 triangulating elements on surface). Five layers of meshfree nodes are generated around the sphere along the normals emanating from the surface nodes, with increasing intervals from the surface, see Fig. 5(a). All Cartesian background nodes exterior to the solid sphere are included in the computation of the present example (although it is frequently desirable to exclude Cartesian nodes below the enveloping nodal clouds in the interest of maintaining good nodal conditioning). Thus the refinement of the Cartesian background here automatically refines the grid around the sphere to create a denser meshfree nodal cloud around the sphere. Fig. 5(b) shows a sample of the flow field in the central $y = 0.5$ plane. The solution obtained on the finest grid 161^3 is employed as the reference solution here, with which errors of solutions on the five coarser grids are computed. The $L2$ norm of errors of the five coarser solutions are calculated based on all the computational nodes within the fluid domain, both Cartesian and meshfree, and shown in Fig. 5(c). The log–log error curve reflects an average gradient of approximately 2, which shows that the numerical scheme is effectively second-order.

5.2. Three-dimensional lid-driven flow with a spherical convecting meshfree nodal patch (moving patch test)

In the present study, the prescribed motion of solid bodies or boundaries against the background Cartesian nodes is emulated by the convecting meshfree nodes. It could be imagined that in the course of motion of the meshfree nodes, some

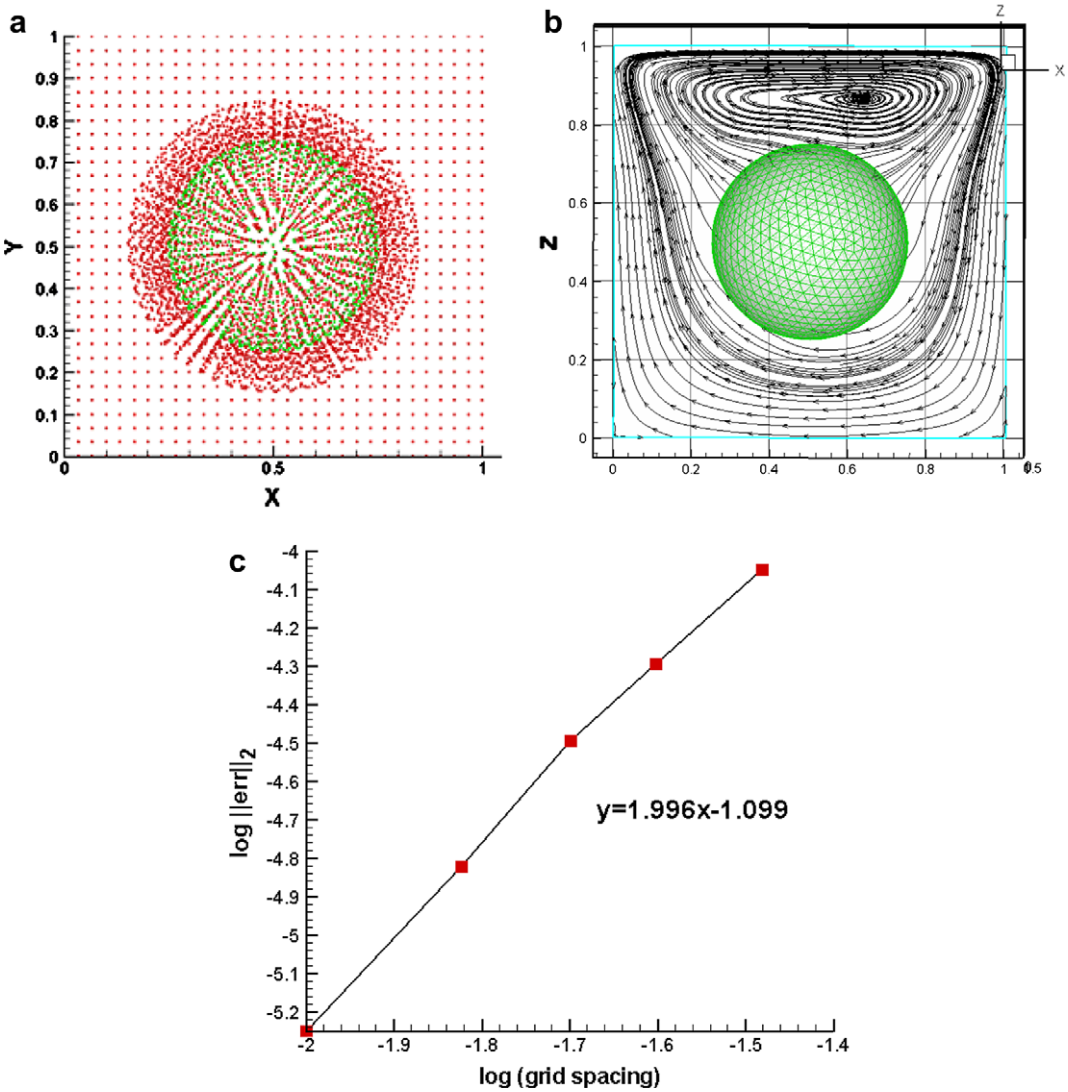
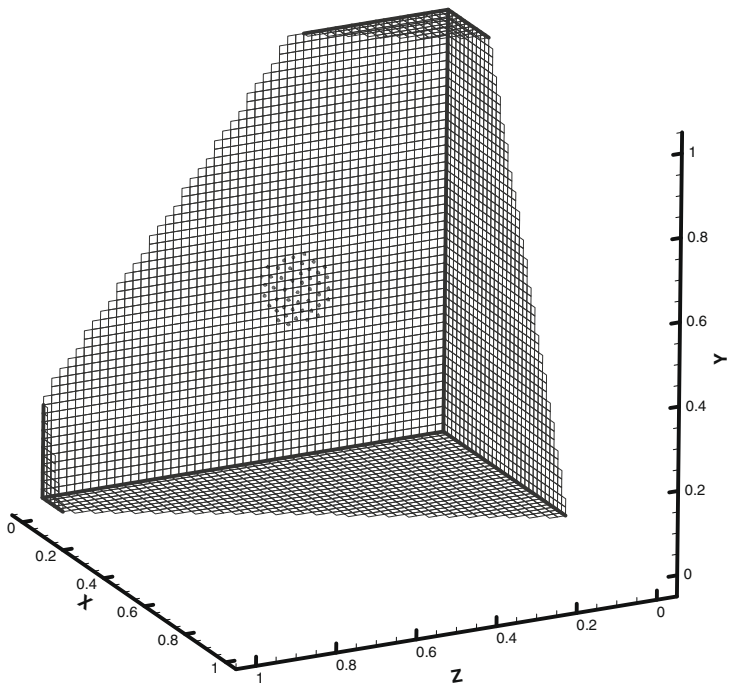


Fig. 5. Flow in a lid-driven cavity with a sphere at $Re = 20$: (a) Plan view of node distribution, (b) streamtraces in the $y = 0.5$ plane, and (c) variation of L_2 error norm of velocity with grid spacing.

Cartesian nodes would be covered over by the meshfree nodal grid and be removed from computation, while some new ones would emerge from under the cover of the meshfree grids and be enrolled into computation. The present test problem is employed to study whether a convecting patch of meshfree nodes will introduce any additional computational errors. It is a classic 3D lid-driven flow, in which the discretized computational domain comprises a uniform cubic Cartesian grid, which is further overlaid with a spherical patch of convecting meshfree nodes (see Fig. 6), and Cartesian base nodes below the spherical cloud is excluded from computation. However, as there is no assumption of a physically moving body within the cloud, the effects of computation on moving meshfree nodes (with deletion and addition of Cartesian base nodes) on solution accuracy can be isolated.

The Reynolds number is defined as $Re = UL/v$, where U is the lid velocity and L is the side length of the chamber. The spatial accuracy is investigated by varying the densities of both Cartesian nodes and meshfree nodes. Three sets of meshes were used for the simulations: (1) 51^3 Cartesian nodes with 943 meshfree nodes, (2) 101^3 Cartesian nodes with 5700 meshfree nodes, and (3) 151^3 Cartesian nodes with 10,950 meshfree nodes. The computation time step is set at 0.001 for these cases.



The Reynolds number for the grid-independent study is set at $Re = 400$. Fig. 7 provides a comparison of the velocity component distributions along the vertical centerline of cubic cavity (u–y) for different mesh densities. The results indicate that the mesh with 101^3 Cartesian nodes and 5700 meshfree nodes is sufficiently fine to obtain the grid-independent solution. However, to further eliminate the effect of grid density, the finest mesh with 151^3 Cartesian nodes and 10,950 meshfree nodes is chosen for the final simulations.

Numerical simulations were conducted with three Reynolds numbers, $Re = 100$, 400 and 1000. The u velocity profile along the vertical centerline in the plane of $z = 0.5$ is computed and compared with the numerical solutions of Shu et al. [53], Wong and Baker [54], and Jiang et al. [55]. It can be seen from Fig. 8 that the velocity profiles agree very well with those of Shu et al. [53], Wong and Baker [54], and Jiang et al. [55]. It indicates that the present method can also achieve good accuracy compared with the mesh-based method [53–55]. It also indicates that the effect of the moving meshfree nodes on the accuracy of the solution is negligible.

5.3. Flow due to a rotating sphere in a box and in infinite space

This case considers the flow driven by a sphere of radius R rotating at a constant angular velocity Ω about a diameter directed along the central z- or azimuthal axis of a box cavity, which is filled with an incompressible viscous fluid of density ρ and kinematic viscosity υ. The Reynolds number is defined as $Re = \Omega R^2/\upsilon$. The steady-state flow is computed in two different ways here. In the first or more conventional approach (Method I), the surface nodes defining the sphere and the enveloping layers of meshfree nodes are held stationary. The rotation of sphere is then prescribed by customary velocity boundary conditions. In the second approach (Method II), the surface and surrounding meshfree nodes are convected in accordance with the prescribed rotation of the sphere against the background of stationary Cartesian nodes. Computation at these moving meshfree nodes follows the ALE form of the NS Eq. (5) and the moving-node methodology of Section 3. The solutions by

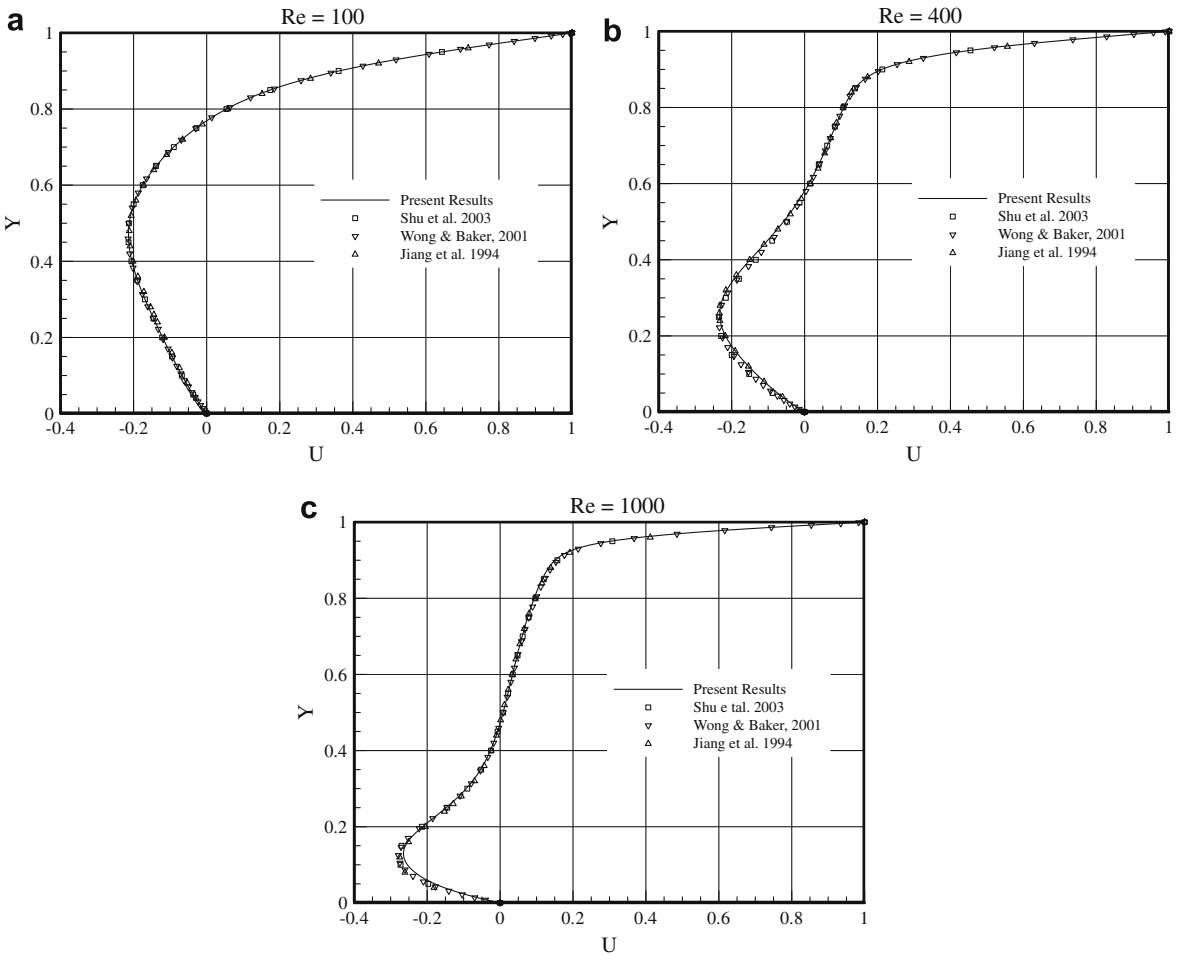


Fig. 8. Comparison of velocity component distribution along the vertical centerline of cubic cavity (u–y): (a) $Re = 100$, (b) $Re = 400$, and (c) $Re = 1000$; the present results based on 151^3 Cartesian nodes and 10,950 meshfree nodes.

the two different methods offer a g[illegible] conv[illegible] [illegible]s the conventional stationary-node implementations.

The main feature of the flow is t[illegible] ex[illegible]ed in[illegible] a[illegible] an outflow at the equator as shown in Fig. 9(b), which is similar to that o[illegible] the [illegible]w in[illegible] [illegible]nbounded fluid domain [56]. A critical angle θ_c, which is the angle betwe[illegible]n th[illegible]-a[illegible]n[illegible] t[illegible] the center of the toroidal vortex ring,

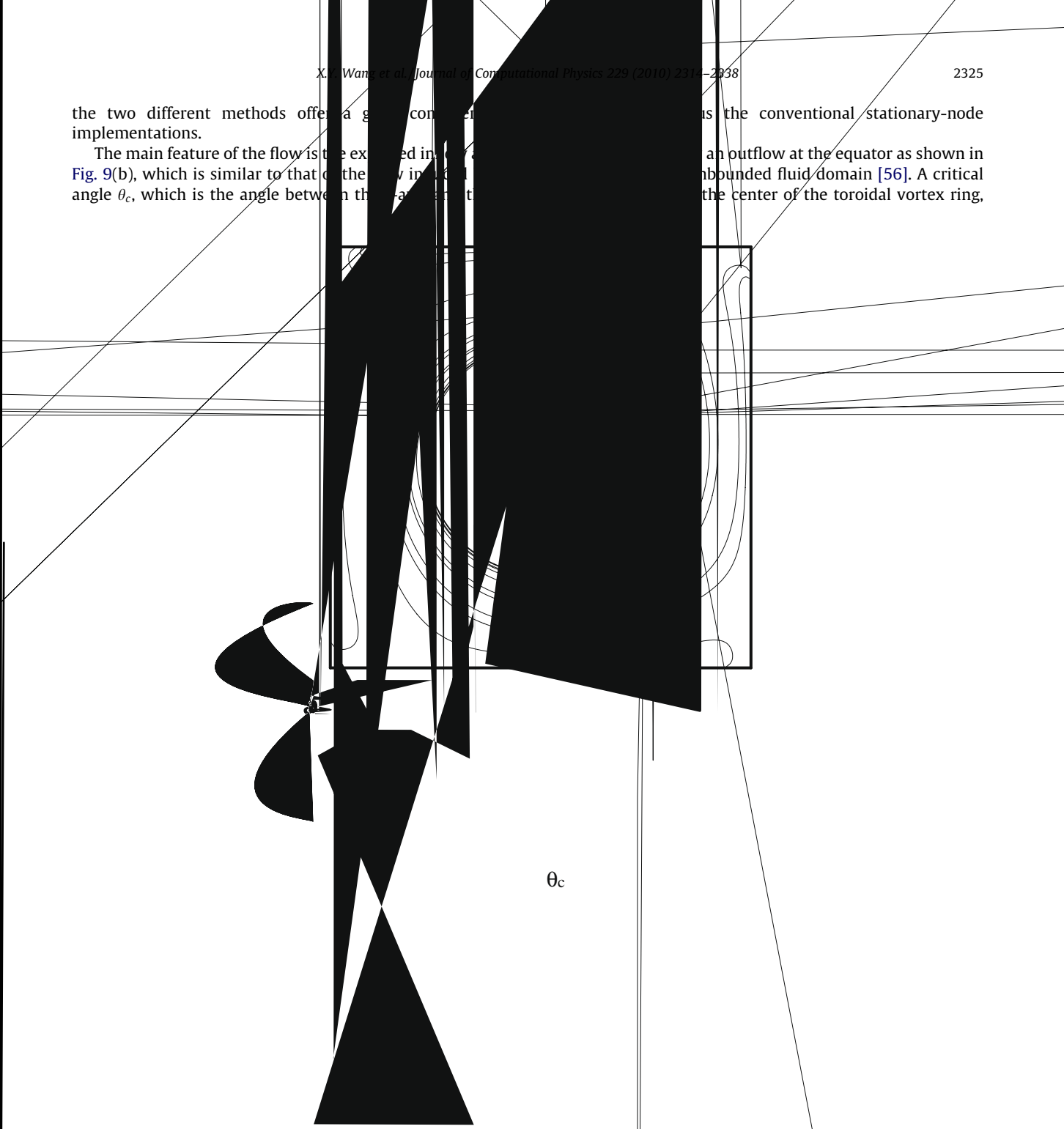


Fig. 9. Streamtraces of steady-state flow fields around a rotating sphere in a cube at $Re = 20$, computed by Method II (with ALE): (a) *xy* central plane, and (b) *yz* central plane.

Table 1
Comparison of the critical angles and the torque coefficients at different mesh sizes, computed by MethodS I and II (with ALE), $Re = 20$.

	Method I		Method II	
	θ_c	C_T	θ_c	C_T
Mesh 1	49.43	2.791	49.42	2.803
Mesh 2	49.43	2.859	49.40	2.863
Mesh 3	49.44	2.881	49.41	2.885

is defined to quantitatively characterize the transition from inflow to outflow (see Fig. 9(b)). An important quantity for this problem is the magnitude of the torque required to rotate the sphere at uniform angular velocity. The torque is computed by integrating the contribution from the stresses over the surface of sphere:

$$T = R\int_S \vec{e_r}\times\tau dS \tag{16}$$

where τ is the surface traction tensor and $\vec{e_r}$ is the unit vector along radial direction. The dimensionless torque coefficient C_T is defined as:

$$C_T = -\frac{T}{0.5\rho R^5\Omega^2}. \tag{17}$$

Five layers of meshfree points with increasing nodal interval from the surface are again used around the discretized sphere. Three sets of meshes were used for the simulations: (1) 41^3 Cartesian nodes with 7250 meshfree nodes, (2) 81^3 Cartesian nodes with 14,830 meshfree nodes, and (3) 121^3 Cartesian nodes with 37,505 meshfree nodes. The Reynolds number for the grid-independent study is set at $Re = 20$. Table 1 compares the critical angle and the torque coefficient based on different mesh sizes, obtained by the two methods. Table 1 shows that there is not much difference in the critical angle for all three mesh sizes. The error in the torque coefficient between Mesh 2 and Mesh 3 is within 1% for both two methods. The comparison indicates that the results based on Mesh 2 are accurate enough. However, the final simulations for this case are performed on Mesh 3.

Fig. 9 shows the streamtraces of the flow field in the x–y and y–z central planes at $Re = 20$. Only results for the ALE/convecting node case (Method II) are presented as the flow fields by stationary-node case (Method I) are visually identical for all practical purposes, and are thus not shown. Tables 2 and 3 compare the critical angle θ_c and the torque coefficients at

Table 2
Comparison of the critical angles at different Reynolds numbers, computed by Methods I and II (with ALE).

Re	10	20	50	100
Method I	48.69	49.44	53.22	55.20
Method II	48.66	49.41	53.22	55.25
Difference (%)	0.06	0.06	<0.01	0.09

Table 3
Comparison of the torque coefficients at different Reynolds numbers, computed by Methods I and II (with ALE).

Re	10	20	50	100
Method I	5.549	2.880	1.439	0.940
Method II	5.551	2.885	1.444	0.948
Difference (%)	0.04	0.17	0.35	0.85

Table 4
Comparison of computation times and torques for single and 5-layer meshfree nodes in rotating sphere problem.

Grid type	Case	Number of mesh nodes		Torque ($Re = 20$)	Time (50 Steps)
		Meshfree	Cartesian		
1 Layer of meshfree nodes	1	1447	41^3	2.760	35
	2	8352	81^3	2.853	235
5 Layers of meshfree nodes	3	1447×5	41^3	2.791	67
	4	1447×5	81^3	2.823	260

Table 5
Comparison of computation times for different number of meshfree nodes in rotating sphere problem.

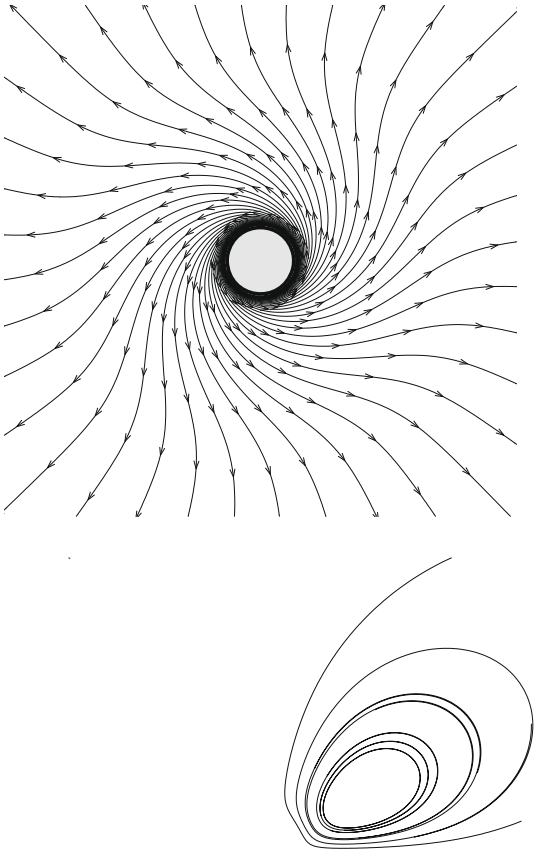
Number of layers	1	5	10
Number of meshfree nodes	1450	7250	14500
R (%)	0.42	2.07	4.06
Computational time (s)	743	939	1164

different Reynolds numbers, obtained by the two methods. The maximum difference in the critical angle between the two solutions is about 0.09% while the maximum difference in the torque coefficient is less than 1%. The consistency or agreement of the two sets of results up to a small numerical difference shows that the ALE nodal convection scheme can work accurately if properly implemented.

As the computational domain is confined, the present case also offers a good platform to investigate some issues related to the hybrid meshfree-and-Cartesian grid. It has already been noted earlier that the present scheme has the flexibility to default to a purely Cartesian scheme, with just one layer of meshfree nodes, i.e. the surface nodes. An investigation has been carried out to compare the default Cartesian case (with just the single layer of nodes defining the boundary) with the case

Table 6
Comparison of computation times for different number of support nodes in rotating sphere problem.

Number of support nodes	30	40	50	80	70	80
Computational time for 50 time steps (s)	801	824	856	894	904	939



when 5 layers of meshfree nodes are used for the problem of rotating sphere in a box (computed by Method I). In the single-layer case, the computational grid is not orthogonal at the boundary, whereas the grid is orthogonal for the 5-layer case. The results are summarized in Table 4. In general, computational stability is enhanced by the use of multiple meshfree layers as compared to just a single layer. For a background Cartesian grid of 81^3, around 8000 or more surface nodes (single-layer case) are needed to obtain a stable solution, whereas stable solution are obtained for 5 layers of 1447 surface meshfree nodes (total of 7235). The computational times for 50 time steps are also fairly comparable for the two cases. The average grid spacing based on $\sqrt{A_s/N}$ are 0.0079 and 0.019, respectively, where A_s is the surface area of sphere and N is the number of surface nodes used. The background grid spacing is 0.0125. This result suggests the importance of grid orthogonality and that with a number of surrounding meshfree layers, surface grid interval/size in excess of background grid could be used, whereas for single-layer surface meshfree grid (default Cartesian), the nodal interval has to be kept somewhat less than that of the background grid. If only a single-layer of 1447 meshfree nodes are used as surface nodes (surface grid size $\sqrt{A_s/N} \approx 0.019$), then computation is stable when a coarser background grid of 41^3 is used (grid interval 0.025, which is greater than 0.019 for the surface grid).

A subsequent investigation is carried out to examine the effect of the number of mesh nodes on the computational time. Again, the problem of rotating sphere in a box is computed by the Method I. The surface of the sphere is discretized by 1450 meshfree nodes while the whole computational domain is discretized by 121^3 Cartesian nodes. The number of the meshfree nodes is changed by varying the number of layer of meshfree nodes. As the present scheme can be regarded as a purely Cartesian scheme if only a single layer of meshfee nodes is applied, the present investigation can also provide information on comparison of computational efficiencies between the present approach (with 5 layers of meshfree nodes) and a purely Cartesian approach. The computational time for 50 time steps is recorded and summarized in Table 5. In the Table, R% denotes the fraction of meshfree nodes in the nodal population. The data showed that the computational time increases by around 26% when the number of meshfree layers is increased from one to five (R% from 0.42% to 2.07%) and by around 57% if ten layers are used (R% from 0.42% to 4.06%). Conventionally, we used only 5 layers of meshfree nodes. As discussed above, if the cloud of nodes is placed around the object, the simulations can be performed on coarser surface and boundary grids and thus the computational time could be still competitive with the single-layer case (or a purely Cartesian approach). It is pertinent to note that present single-layer case is actually unstable and fail not longer after 50 time steps.

Another investigation is carried out to examine the effect of the number of support nodes on the computational time by computing the problem of rotating sphere in a box. The mesh used for test case is 121^3 Cartesian nodes with 7250 meshfree nodes. The computational time of Method I for 50 time steps is recorded and summarized in Table 6. The results demonstrated that the computational time may increases around 17% if the support node number increases from $n = 30$ to $n = 80$. Although less support nodes lead to a high computational efficiency, it may result in the ill-conditioning of the matrix, which arises from poor spatial arrangement of the support nodes as mentioned before.

We next consider the unbounded domain case. This problem has been numerically studied by Dennis et al. [56] who solved the steady, axisymmetric Navier–Stokes equations using a vorticity-streamfuction formulation. The unsteady form of the problem in 3D was also solved by Gilmanov and Sotiropoulos [24] using a hybrid Cartesian/immersed boundary method and by Lv et al. [57] using an immersed membrane method. However, although the previous studies [24,56,57] have shown that the flow is steady as well as axisymmetric within the range of the Reynolds number investigated, full 3D unsteady simulations are applied in the present study. The surface and surrounding meshfree nodes are convected in accordance with the prescribed rotation of the sphere against the background of stationary Cartesian nodes. Computation at these moving meshfree nodes follows the ALE form of the Navier–Stokes Eq. (5) and the moving-node methodology of Section 3.

The sphere is now placed at the centre of a $(32R)^3$ Cartesian computational domain. The surface of the sphere is discretized with 2966 meshfree nodes. Four more layers of meshfree nodes were generated radially from the surface of the sphere, with smaller radial intervals for nodes nearer to the surface. The Cartesian computational domain is discretized with 181^3

Table 7
Comparison of the critical angle θ_c.

Re	Present results	Ref. [45]	Ref. [19]	Ref. [46]
20	62.1	62.6	61.6	/
50	69.7	69.4	68.2	/
100	74.2	73.8	72.1	74.3

Table 8
Comparison of the torque coefficient.

Re	Present results	Ref. [45]
10	5.312	5.399
20	2.990	3.048
50	1.535	1.544
100	0.956	0.966

nodes. The Cartesian grid is uniform in all three directions within a $(4R)^3$ rectangle box containing the sphere, with grid spacing of $\Delta x = \Delta y = \Delta z = 0.05R$. The Cartesian grid is stretched in the remaining of the computational domain. Note that the grid density around the sphere is the same as Mesh 2 for sphere in the box, which was fine enough to provide grid-independent results. The sphere starts to rotate impulsively at $t = 0$ with constant velocity Ω. The simulation is performed from $t = 0$ until a steady axisymmetric state is obtained.

Simulations are carried out at four Reynolds number, $Re = 10$, 20, 50 and 100. Fig. 10 shows the streamtraces of the flow field in the x–y and y–z central planes at $Re = 20$. Fig. 10(b) shows the expected inflow at the poles and outflow at the equator, such as has been observed in previous studies [24,56,57]. The critical angles θ_c obtained show very good agreement with the published results of [24,56,57] in Table 7. Note that in the study of Dennis et al. [56], the critical angle θ_c at which the inflow changes from inflow to outflow is measured at a definite radial distance where the radial velocity component along the equatorial radius $\theta = \pi/2$ reaches its maximum value. Dennis et al. pointed out that at a fixed Re the angle at which the inflow changes to an outflow does not vary greatly with radial distance. Thus, the good agreement between the present solutions and those of Dennis et al. [56] is also convincing. Table 8 further compares the torque coefficients obtained by the present method with the steady-axisymmetric streamfunction-vorticity results of Dennis et al. [56]. Again the agreement is very good.

5.4. Flow due to an oscillating sphere in an unbounded fluid domain

The next validation test case considers the unsteady flow induced by an oscillating sphere with x-velocity $U = U_0 \cos(\omega t)$ in an unbounded fluid domain that is assumed to be at rest at a large distance from the sphere (see Fig. 11). U_0 and ω are the

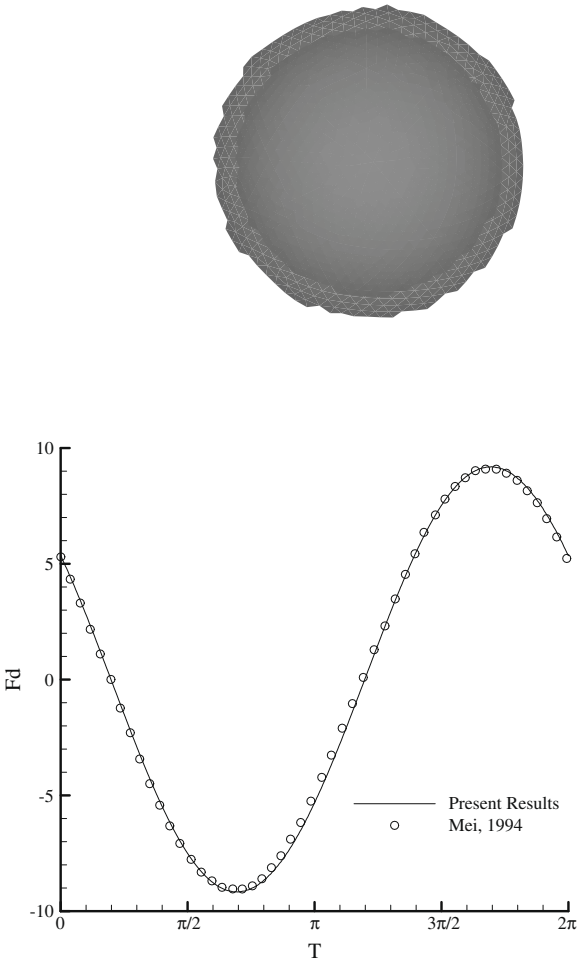


Fig. 12. Comparison of the unsteady dimensionless drag in a period when the periodic state is obtained; the drag F_d and the time T is dimensionalized by $6\pi\rho\nu U_0 R$ and $1/\omega$, respectively.

amplitude and frequency of the oscillation, respectively and t is time. The Reynolds number and the Stokes number are defined as $Re = 2RU_0/\upsilon$ and $\varepsilon = \sqrt{(\omega R/2\upsilon)}$, respectively, where R is the radius of the sphere and υ is the kinematic fluid viscosity. The computational domain is a $34R \times 32R \times 32R$ rectangular box discretized by $221 \times 181 \times 181$ Cartesian nodes. The Cartesian grid is uniform in all three directions within a $6R \times 4R \times 4R$ rectangular box containing the sphere, with grid spacing of $\Delta x = \Delta y = \Delta z = 0.05R$. The Cartesian grid is stretched in the remaining of the computational domain. The meshfree nodal discretization of the sphere remains the same as that in Section 5.3.

The flow has been studied by Mei [58] using different methods: (1) Fourier mode expansion in the frequency domain; (2) a time-dependent finite difference technique in the time domain; and (3) a matched asymptotic expansion for high-frequency oscillation. In the present study, the Reynolds number and the Stokes number are set at $Re = 40$ and $\varepsilon = 4$, respectively. The simulation is performed from $t = 0$ with $\Delta t = 0.0025$. The unsteady drag force is computed at each time step. The computed force history, presented in Fig. 12, shows very good agreement with the solution of Mei [58] obtained with the Fourier mode expansion method.

5.5. Simulation of flows around 3D flapping wings

In this section, we demonstrate the applications of the present numerical scheme to model unsteady flows around flapping wings at low Reynolds number. Such flows have attracted a good deal of interest in recent years as scientists and engi-

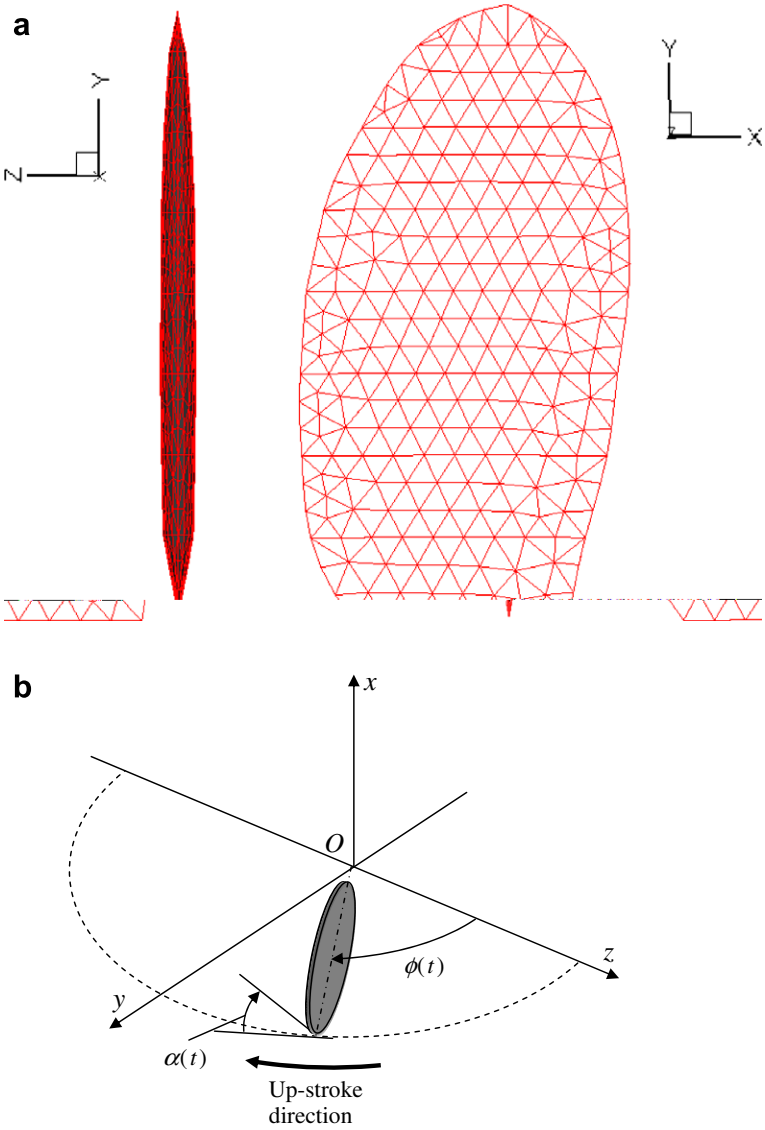


Fig. 13. (a) Side and plan views of the model wing. (b) Overall kinematics of wing motion: azimuthal/stroke angle ϕ and angle of attack α.

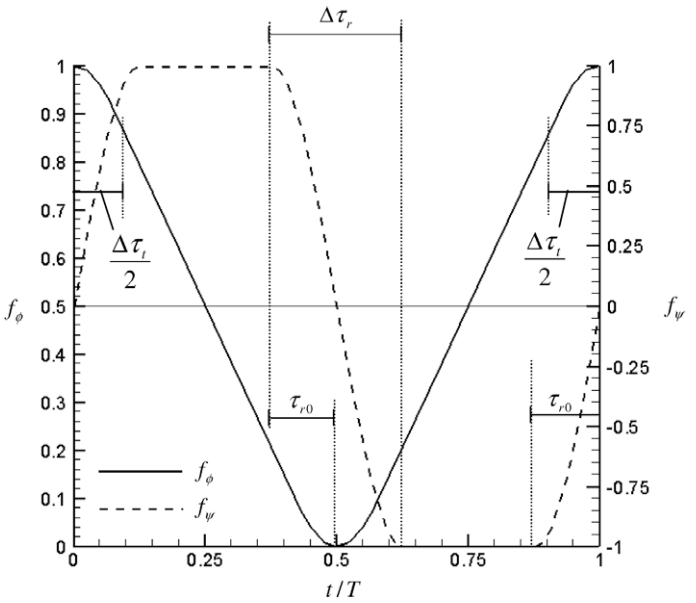


Fig. 14. The T-Periodic functions of azimuthal/stroke angle ϕ and twist angle ψ of flapping wing. τ_{r0} refers to the start time of wing twist/rotation measured *relative* to stroke reversal. $\Delta\tau_r$ is the duration of wing twist/rotation. $\Delta\tau_t$ is the duration of wing deceleration/acceleration during stroke reversal.

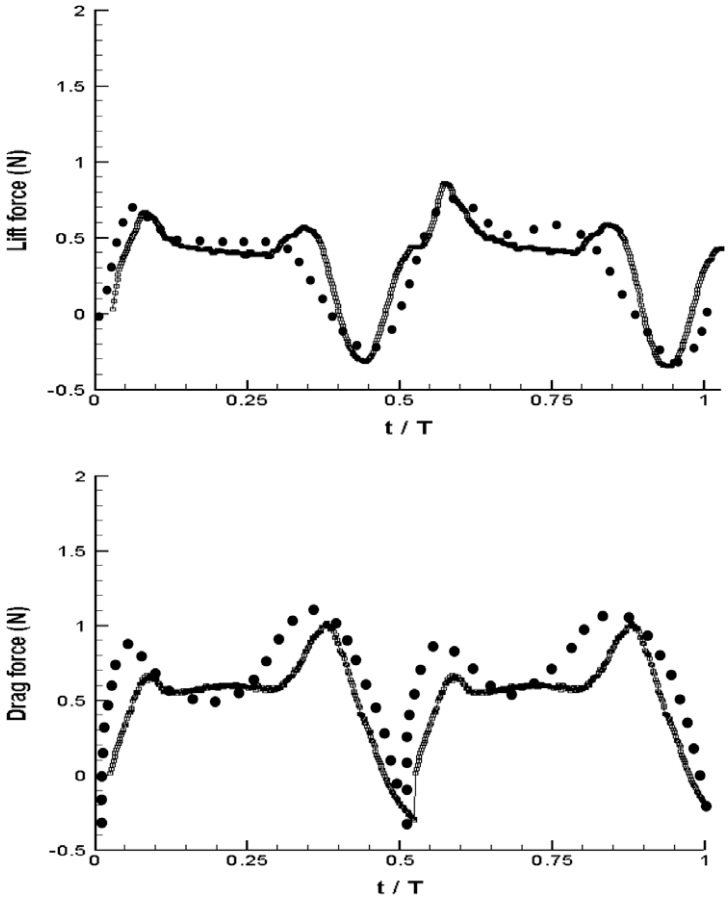


Fig. 15. Drag and lift forces acting on the wing.

neers strive to understand the complex and often novel aerodynamics underlying the flight of insect (see review of Sane [59]). Liu and Kawachi [60] was probably the first to simulate viscous flow around flapping insect wings. They used a finite-volume method (FVM), in which the complex sweeping cum twisting action of the wing is accommodated by dynamically rotating the grid about the wing base. This approach is suited for single-wing simulation. Sun and Yu [61] and Sun and Tang [62], on the other hand, had employed the structured overset/chimera grid approach. The scheme is able to treat multiple moving wings with relative ease by utilizing separate component grids around each wing. More recently, Aono et al. [63] applied an overset-grid FVM to model flows around full kinematic models of insects with wings and body. By encasing body and flapping wings in separate overlapping volume grids, re-meshing of the computational domain normally performed in FVM is avoided. Component FVM grids are also able to treat arbitrary wing and body geometry more easily than the structured grid components of conventional overset grid approaches. The flow around a flapping wing has also been simulated by Gilmanov and Sotiropoulos [24] using a Cartesian grid method. The method could handle complex body geometry and is also computationally efficient in using only a single Cartesian grid system over the entire computational domain. A fine Cartesian grid may be needed, however, to achieve good resolution of fine surface features and fluid boundary layer. The present hybrid meshfree-and-Cartesian scheme contains aspects of these various methods: (1) the domain is covered by a Cartesian base grid similar to Cartesian grid methods, (2) individual moving bodies are covered by its own convecting grid systems as in traditional overset-grid methods, and (3) computation is carried out in a single global frame as for Cartesian grid methods.

As the first example, we carry out the Navier–Stokes simulation for a single-wing following the kinematical scenario presented in the experiments of Sane and Dickinson [64] and Birch and Dickinson [65]. The wing has the form given in Fig. 13(a) with a thickness to maximum chord of about 12%, following [24]. It is discretized by a triangular mesh with 1332 elements. Fig. 13(b) shows the overall kinematics of wing motion. A complete wing flapping cycle comprises a downstroke ($\dot{\phi} < 0$) and an upstroke ($\dot{\phi} > 0$) in the reverse direction, with spanwise axis in the *yz*-plane. Simultaneously, the wing twists/rotates about its spanwise axis as it approaches stroke reversal to maintain the same leading edge in both strokes. Between the two twisting stages, the wing maintains a constant angle of attack α_0 as it sweeps forward. The simultaneous sweeping and twisting actions of the wing are described by the following time-dependent functions of stroke angle $\phi(t)$ (azimuthal angle) and angle of attack $\alpha(t)$ (measured *relative* to instantaneous direction of travel):

$$\phi(t) = \phi_0 + \Phi f_\phi(t/T), \tag{18}$$

$$\alpha(t) = \frac{\pi}{2} \mp \psi_0 f_\alpha(t/T + \tau_{\psi/\phi})(\pm\dot{\phi} \geqslant 0), \tag{19}$$

where $\psi_0 = \frac{\pi}{2} - \alpha_0$ and Φ is stroke amplitude of the wing, which is the azimuthal angle swept through by the wing in each stroke (upstroke and downstroke). The functions $f_\phi(t/T)$ and $f_\alpha(t/T)$, which govern the sweep and rotation of the wing respectively with respect to time, are presented in Fig. 14. Wing twist/rotation is coordinated with the stroking action of the wing in terms of a start time τ_{r0} and the duration of wing rotation $\Delta\tau_r$, both measured as a fraction of flapping period *T*. More specifically, τ_{r0} refers to the time at which the wing rotation begins, measured relative to the point of stroke reversal;

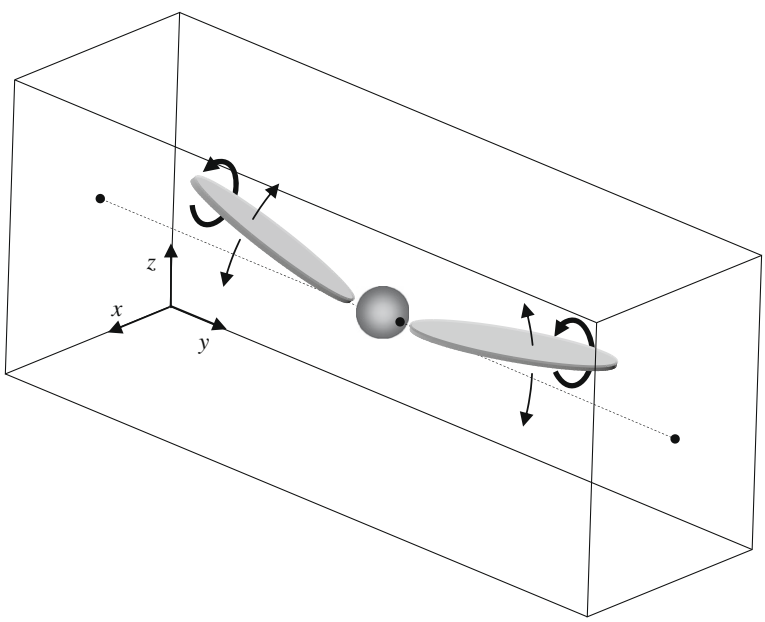


Fig. 16. A schematic view of two wings and body.

the relative phase/time shift $\tau_{\psi/\phi}$ between the two actions in (19) is then given by $\tau_{\psi/\phi} = -\tau_{r0} - \Delta\tau_r/2$. The $\Delta\tau_t$ in Fig. 14 denotes the duration of wing deceleration/acceleration during stroke reversal.

The present computed case is based a wing stroke amplitude of $\Phi = 160^\circ (\phi_0 = 10^\circ)$ and an attack of angle $\alpha_0 = 45^\circ$. The rotation of wing begins at $\tau_{r0} = -0.24$ and is sustained for a duration of $\Delta\tau_r = 0.24$, by the end of which time the wing at poised at $\alpha_0 = 45^\circ$ for the return stroke. Wing deceleration/acceleration occurs over the duration of $\Delta\tau_t = 0.16$ during stroke reversal. The Reynolds number is $Re = (\Phi n \bar{R})R/\upsilon \approx 160$, where n is the wingbeat frequency, R is the wing length and $\bar{R} = 0.65R$ in accordance with the experimental case. Simulation was carried by the present method $(\Delta t = 0.005)$ for 6 flapping cycles. The results for the drag and lift forces in the last cycle are compared against the closest corresponding experimental results of Sane and Dickinson [64] in Fig. 15. The experimental results were digitally scaled from their Fig. 6(B).

The present numerical force histories reflect in each stroke the double peak feature that is quite common for flapping wings. The computed lift force is in good agreement with the experimental measurement. The computed drag force also agrees well with experimental drag force over the mid-translational stage, but display somewhat lower and narrow peaks at the start and end of each stroke (down- and up-). The agreement with the experiment can be regarded to be rather good given the technical complexity of the experiment, and potential sources for experimental inaccuracies. The experimentally measured forces are quite small; from which extraneous gravitational and inertial forces acting on the wings must be subtracted out to yield the aerodynamic forces. Gearing slippage and wing deformation under load can contribute to deviation in the angle of attack – lift and drag are particularly sensitive to angle of attack changes according to [64]. A small increase in the angle of attack as the wing approaches stroke reversal at speed could lead to reduction in lift and increase in drag. The

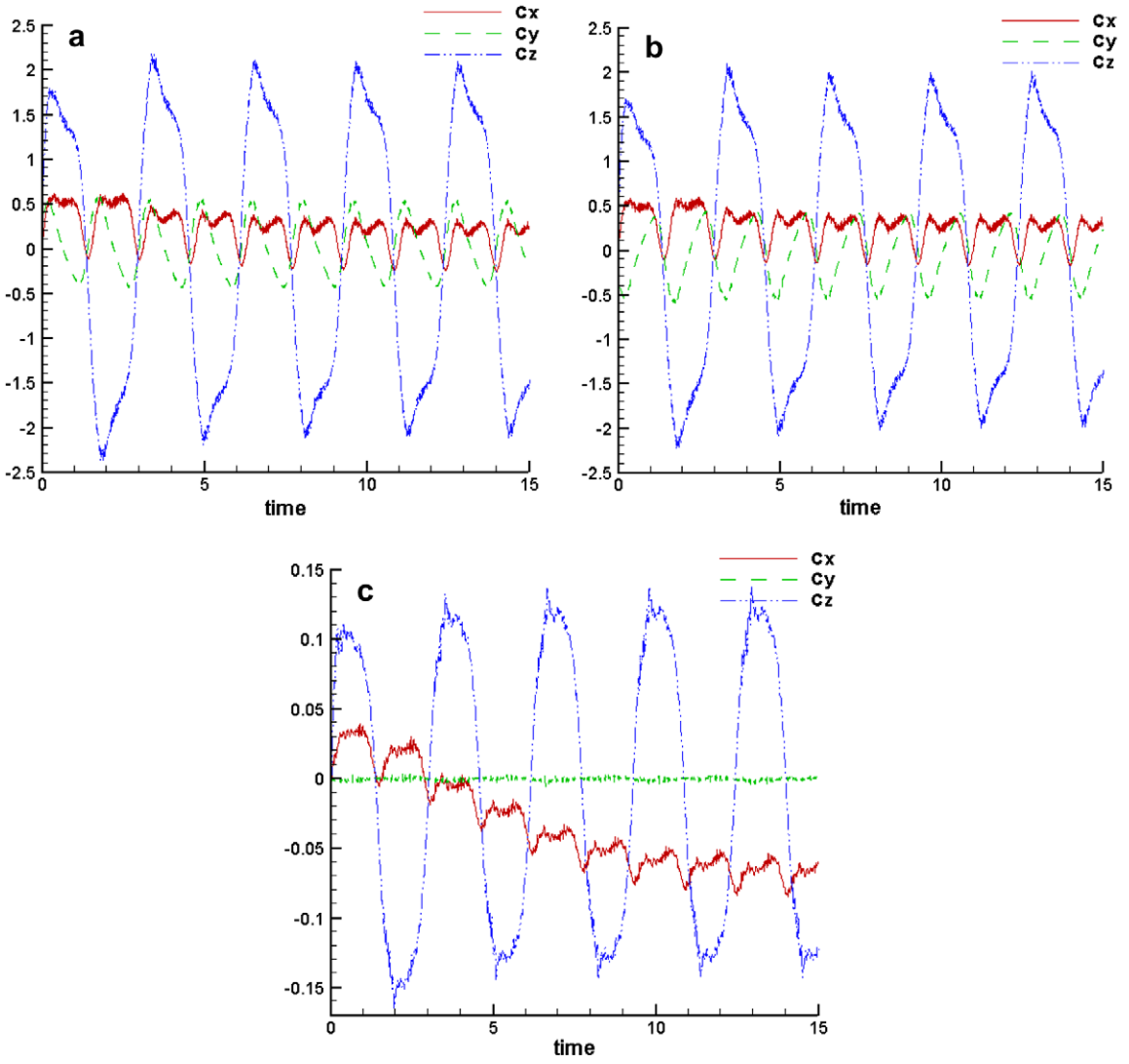


Fig. 17. Forces acting on the: (a) left wing, (b) right wing, and (c) spherical body in the three spatial directions.

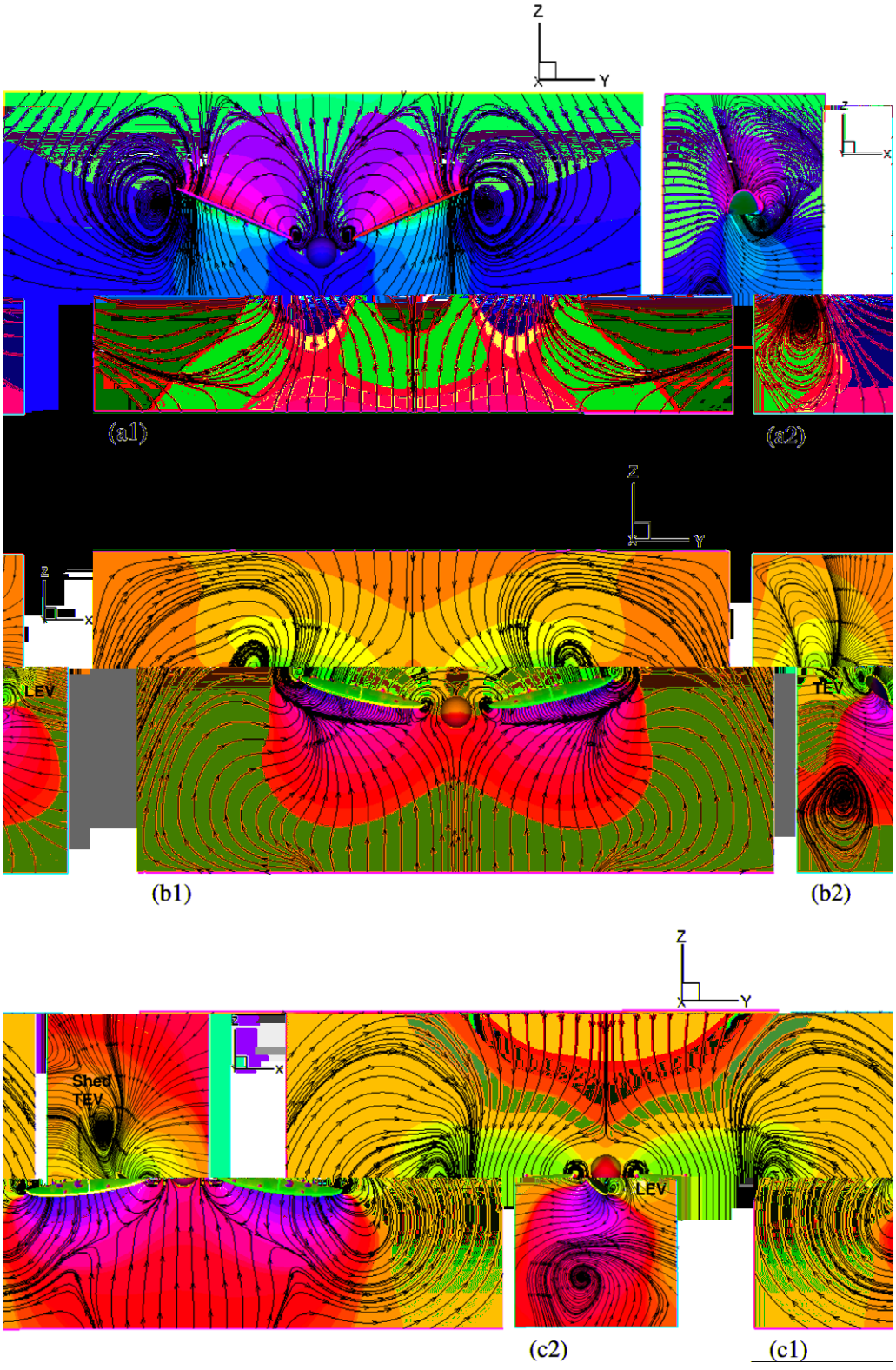


Fig. 18. Flow field and pressure distribution around a model flyer at time: (a) $t/T = 1/7$, (b) $t/T = 2/7$, (c) $t/T = 3/7$, (d) $t/T = 4/7$, (e) $t/T = 5/7$, (f) $t/T = 6/7$, and (g) $t/T = 7/7$). Left and right figures show flow sections at $x = 1.0$ and $y = 1.4$, respectively. LEV and TEV denote *leading* and *trailing* edge vortices, respectively. Red and blue denote high and low pressure regions respectively (see also web-version of this article).

differences may also possibly be due to some geometric and kinematic mismatches between the simulation and the experiment. A slightly larger stroke amplitude of $\Phi = 180^\circ$ (and hence $Re \approx 180$) was used in the experiment compared to 160° used in the present simulation. Finally, the width and magnitude of experimental force peaks and troughs may also be influenced by the acceleration and deceleration rates ($\ddot{\phi}$) of the wing during stroke reversal, for which details are not available in their papers. The above simulation demonstrates that the present computational scheme is able to capture the dynamical features of flapping wing flows with reasonable fidelity.

The next example illustrates the application of the present numerical scheme to a model problem comprising a multitude of independent bodies. The model comprises two wings flapping about a body, which is assumed here to be spherical for simplicity. The wings are highly compressed ellipsoids $\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}=1$ with non-dimensional semi-axes a, b and c equal to 0.4, 0.1 and 0.02, respectively. The spherical body has a diameter of 0.2. The wing tip is 1.0 from the centre of sphere (centre of rotation). A schematic view of the wings and body is given in Fig. 16. The wings flap symmetrically about the body. The flapping motion of the wings follows that of the preceding example with azimuthal sweep angle $\Phi = 45^\circ, \alpha_0 = 45^\circ$. The other parameters are $\Delta\tau_r = 0.5, \tau_{r0} = -0.25, \Delta\tau_t = 0.5$ and $\tau_{\psi/\phi} = 0$. The Reynolds number is about 160 based on radius of gyration $\overline{R} = 0.632R$ or Re = 250 for based on wing tip. Owing to resource limitation, a relatively small computational domain has been used in this example; with Neumann type velocity conditions $(\mathbf{n}\cdot\nabla)\mathbf{u} = \mathbf{0}$ applied at the domain boundary. Simulation is carried out without the assumption of flow symmetry.

Fig. 17 show the forces acting on the two wings and the spherical body in the three spatial directions as a function of time over about 5 cycles from the commencement of flapping. The force coefficients are defined by $C = 2F/A\rho(\Phi n\overline{R})^2$ where A is

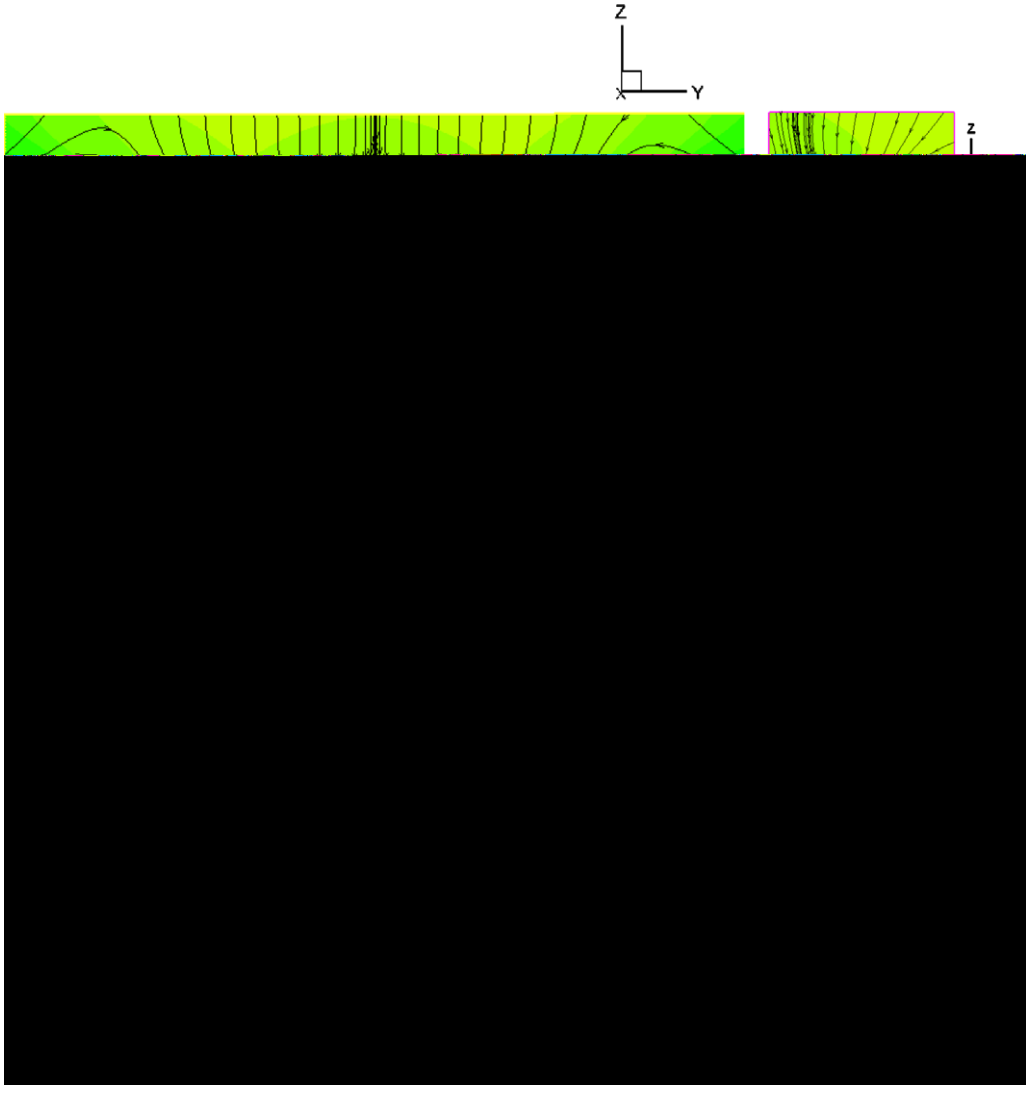


Fig. 18 (*continued*)

the planform area of wing. The wings experience a net thrust in the positive x-direction. The force histories are more or less identical on the two wings. The C_y coefficients of the two wings are in exact anti-phase, resulting in net zero lateral/side y-force. This may be expected from the physical symmetry of the problem. Fig. 17(c) shows a drag force developing on the spherical body in the negative x-direction. The development of the drag force suggests that the flow has acquired a roughly quasi-periodic state by the fifth flapping cycle. Its lateral force coefficient $C_y \approx 0$, while the variation of C_z is in phase with the same on the wings but of much smaller magnitude. These are in line with the general physical expectations of the flow.

Fig. 18 shows the flows around the wings at various points in the 5th flapping cycle. The right hand figures offer sectional views of the flow at plane $x = 1.0$ passing through the centre of the spherical body, viewed in the $-x$ direction. The left figures shows corresponding sections of the flow field around the left wing in the plane $y = -1.4$, viewed in the $+y$ direction. The pressure distributions in the planes are given by flooded contours (provided in colour in the softcopy).

The wings rotate rapidly about its spanwise axis near the end of upstroke in Fig. 18(a) (note that down- and up- strokes refer to the forward and backward sweeps of the wings, and are not to be associated with the direction of earth gravity). By this time, the translational (sweep) speed of the wings has already slowed down to near zero as the wings go into stroke reversal. Flow induced by the upstroke now impacts squarely upon the lower face of the wings, as they turn over; resulting in higher pressure there, compared with the pressure on the upper surface. The higher pressure on the lower face is maintained as the wings sweep/translate forward in the downstroke with nearly constant attack angle α in Fig. 18(b) and (c). The wings counter-rotate (about the spanwise axis) as they are reaching the end of the downstroke in Fig. 18(d). At this point the higher pressure is switched over to the upper surface of the wings. The higher pressure is now maintained on the upper sur-

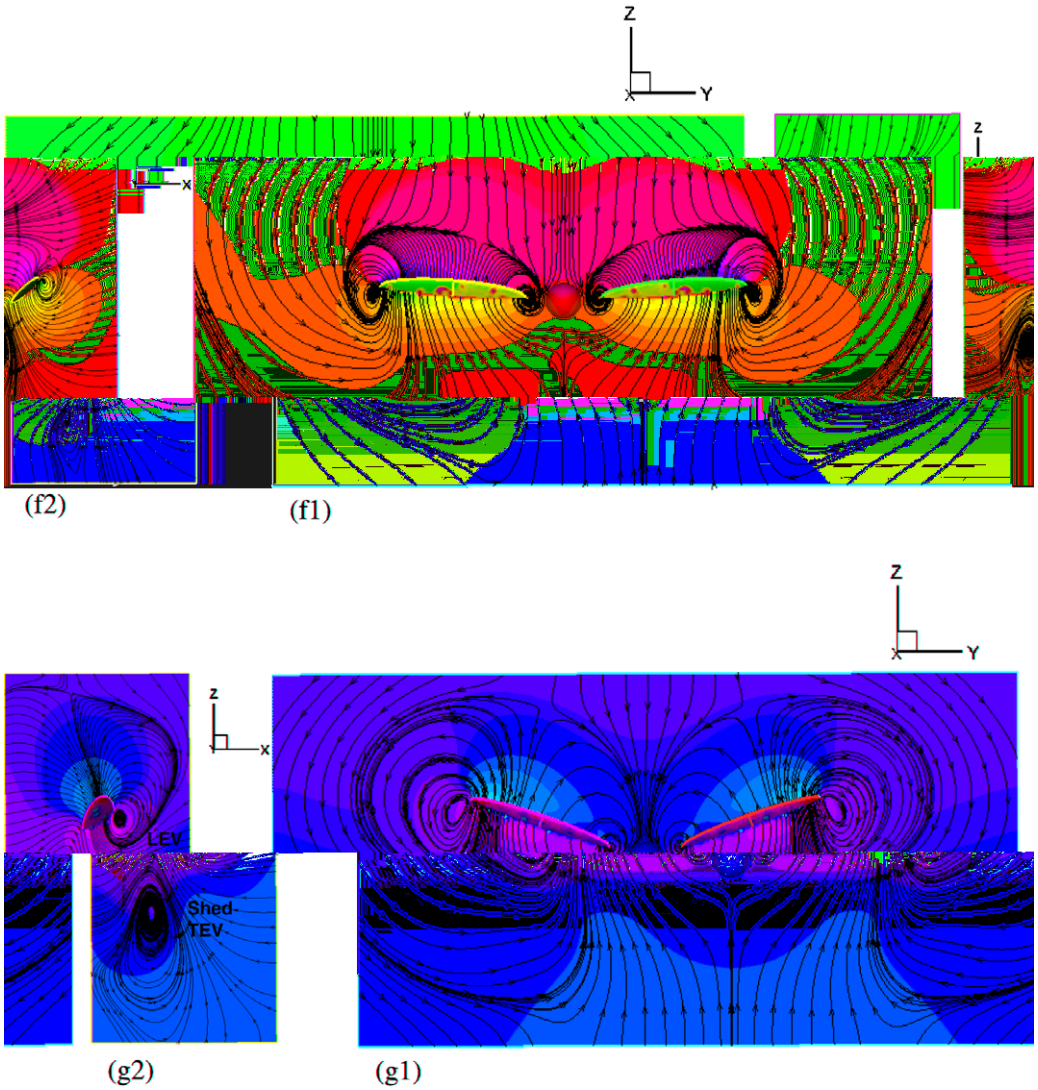


Fig. 18 (*continued*)

face as the wings rotate over and sweep/translate forward in the upstroke, Fig. 18(d)–(f). Fig. 18(f-g-b) show the rotation of the wings during stroke reversal from down- to up-stroke to complete a flapping cycle. The relative pressure difference on the two faces of the wings generates a predominantly positive thrust on the wings in $+x$ direction during both the up- and down-strokes (see frames on the right); which in turn drives a continuous flow of fluid in the $-x$ direction.

Delving a little into the flow structures, Fig. 18(b2) and (c2) show a leading edge vortex (LEV) attached above the leading edge of the wing during the translational (constant-α) stage of the downstroke. The LEV induces a low pressure region over the wings and has been identified as a leading unsteady lift enhancement mechanism in insect flight [59]. One can also discern a trailing edge vortex (TEV) being shed during this stage. The corresponding attached LEV and shed TEV also occur during the upstroke in Fig. 18(e2) and (f2). No shedding of the LEVs has been observed in this simulation due to the relatively small stroke amplitude used. Instead, the attached LEVs are compressed and 'sliced' through by the rotating wings during stroke reversal to form the TEVs, frames Fig. 18(a2) and (b2), which are then shed as the wing sweeps forward in frame Fig. 18(c2). Fig. 18(d2) and (f2) show the corresponding processes during the upstroke.

6. Conclusions

The present paper extends the hybrid meshfree-and-Cartesian grid method of Chew et al. [39] for simulating moving body flow problems in 3D space. A mixed Lagrangian–Eulerian form of the incompressible Navier–Stokes equations governs the flows at meshfree nodes, which convect with body. Discretization at meshfree nodes is performed with a singular value decomposition based generalized finite difference (SVD–GFD) scheme, while cost-efficient accurate standard finite difference is applied in the bulk of the Cartesian-discretized computational domain. Meshfree nodes allow complex boundary geometry to be precisely represented. The present hybrid grid scheme combines the merits of conventional finite difference and meshfree methods. It has the benefit of low geometric administration cost, whilst possessing a high degree of geometric flexibility comparable to that of unstructured mesh-based FV and FE methods. The requirement for explicit data interpolation is greatly minimized (at node creation) and boundary conditions are precisely implemented on the boundary nodes. The flow equations are integrated by a fractional-step projection method. The present scheme is validated by a moving patch test (for consistency against Cartesian grid computation) and against published results for 3D moving body problems. The agreement has been very good. Finally, the method is applied on low-Reynolds number flapping wing applications, where large boundary motions are involved. The present study demonstrates the potential of the present hybrid meshfree-and-Cartesian grid scheme for solving complex moving body problems in 3D.

References

[1] Ž. Lilek, S. Muzaferjia, M. Perić, V. Seidl, An implicit finite-volume method using nonmatching blocks of structured grid, Numer. Heat Transfer B 32 (1997) 385.
[2] Ž. Lilek, S. Muzaferjia, M. Perić, V. Seidl, Computation of unsteady flows using nonmatching blocks of structured grid, Numer. Heat Transfer B 32 (1997) 403.
[3] H.H. Hu, D.D. Joseph, M.J. Crochet, Direct simulation of fluid particle motion, Theor. Comput. Fluid Dyn. 3 (1992) 285.
[4] H.H. Hu, Direct simulation of flows of solid–liquid mixtures, Int. J. Multiphase Flow 22 (1996) 335.
[5] H.G. Choi, Splitting method for the combined formulation of the fluid-particle problem, Comput. Meth. Appl. Mech. Eng. 190 (2000) 1367.
[6] H.H. Hu, N.A. Patankar, M.Y. Zhu, Direct numerical simulations of fluid–solid systems using the arbitrary Lagrangian–Eulerian technique, J. Comput. Phys. 169 (2001) 427.
[7] E. Onate, S.R. Idelsohn, F. Del Pin, R. Aubry, The particle finite element method: an overview, Int. J. Comput. Meth. 1 (2004) 267.
[8] C.S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (1977) 220.
[9] 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.
[10] Z. Li, M.C. Lai, The immersed interface method for the Navier–Stokes equations with singular forces, J. Comput. Phys. 171 (2001) 822.
[11] M.C. Lai, C.S. Peskin, An immersed boundary method with formal second-order accuracy and reduced numerical viscosity, J. Comput. Phys. 160 (2000) 705.
[12] J.J. Xu, Z. Li, J. Lowengrub, H. Zhao, A level-set method for interfacial flows with surfactant, J. Comput. Phys. 212 (2006) 590.
[13] S. Xu, Z.J. Wang, An immersed interface method for simulating the interaction of a fluid with moving boundaries, J. Comput. Phys. 216 (2006) 454.
[14] S. Xu, Z.J. Wang, Systematic derivation of jump conditions for the immersed interface method in three dimensional flow simulation, SIAM J. Sci. Comput. 27 (2006) 1948.
[15] R. Mittal, H. Dong, M. Bozkurttas, F.M. Najjar, A. Vargas, A. von Loebbecke, A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries, J. Comput. Phys. 227 (2008) 4825.
[16] R. Ghias, R. Mittal, H. Dong, A sharp interface immersed boundary method for compressible viscous flows, J. Comput. Phys. 225 (2007) 528.
[17] H. Luo, R. Mittal, X. Zheng, S.A. Bielamowicz, R.J. Walsh, J.K. Hahn, An immersed-boundary method for flow–structure interaction in biological systems with application to phonation, J. Comput. Phys. 227 (2008) 9303.
[18] H. Zhao, J.B. Freund, R.D. Moser, A fixed-mesh method for incompressible flow–structure systems with finite solid deformations, J. Comput. Phys. 227 (2008) 3114.
[19] K. Taira, T. Colonius, The immersed boundary method: a projection approach, J. Comput. Phys. 225 (2007) 2118.
[20] H.S. Udaykumar, R. Mittal, W. Shyy, Computation of solid–liquid phase fronts in the sharp interface limit on fixed grids, J. Comput. Phys. 153 (1999) 535.
[21] 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.
[22] 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.
[23] 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.

[24] A. Gilmanov, F. Sotiropoulos, A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies, J. Comput. Phys. 207 (2005) 57.
[25] S. Marella, S. Krishnan, H. Liu, H.S. Udaykumar, Sharp interface Cartesian grid method I: an easily implemented technique for 3D moving boundary computations, J. Comput. Phys. 210 (2005) 1.
[26] H. Liu, S. Krishnan, S. Marella, H.S. Udaykumar, Sharp interface Cartesian grid method II: a technique for simulating droplet interactions with surfaces of arbitrary shape, J. Comput. Phys. 210 (2005) 32.
[27] Y. Yang, H.S. Udaykumar, Sharp interface Cartesian grid method III: solidification of pure materials and binary solutions, J. Comput. Phys. 210 (2005) 55.
[28] H.S. Udaykumar, S. Krishnan, S.V. Marella, Adaptively refined parallelised sharp interface Cartesian grid method for three-dimensional moving boundary problems, Int. J. Comput. Fluid Dyn. 23 (2009) 1.
[29] J.A. Benek, J.L. Steger, F.C. Dougherly, A flexible grid embedding technique with application to the Euler equations, AIAA Paper 83-1944, 1983.
[30] Z.J. Wang, A fully conservative interface algorithm for overlapped grids, J. Comput. Phys. 122 (1995) 96.
[31] Y. Zang, R.L. Street, A composite multigrid method for calculating unsteady incompressible flows in geometrically complex domains, Int. J. Numer. Meth. Fluid 20 (1995) 341.
[32] H.S. Tang, S.C. Jones, F. Sotiropoulos, An overset-grid method for 3D unsteady incompressible flows, J. Comput. Phys. 191 (2003) 567.
[33] T. Liszka, J. Orkisz, The finite difference method at arbitrary irregular grids and its application in applied mechanics, Comput. Struct. 11 (1980) 83.
[34] T. Liszka, An Interpolation method for an irregular net of nodes, Int. J. Numer. Meth. Eng. 20 (1984) 1599.
[35] T.J. Liszka, C.A.M. Duarte, W.W. Tworzydlo, hp-Meshless cloud method, Comput. Meth. Appl. Mech. Eng. 39 (1996) 263.
[36] C.A.M. Duarte, J.T. Oden, Hp clouds – a meshless method to solve boundary value problems, Technical Report 95–05, TICAM, University of Texas at Austin, 1995.
[37] H. Ding, C. Shu, K.S. Yeo, D. Xu, Development of least-square-based two-dimensional finite-difference schemes and their application to simulate natural convection in a cavity, Comput. Fluids 33 (2004) 137.
[38] H. Ding, C. Shu, K.S. Yeo, D. Xu, Simulation of incompressible viscous flows past a circular cylinder by hybrid FD scheme and meshless least square-based finite difference method, Comput. Meth. Appl. Mech. Eng. 193 (2004) 727.
[39] C.S. Chew, K.S. Yeo, C. Shu, A generalized finite-difference (GFD) ALE scheme for incompressible flows around moving solid bodies on hybrid meshfree – Cartesian grids, J. Comput. Phys. 218 (2006) 510.
[40] S.J. Ang, K.S. Yeo, C.S. Chew, C. Shu, A singular-value decomposition (SVD)-based generalized finite difference (GFD) for close-interaction moving boundary flow problems, Int. J. Numer. Meth. Eng. 76 (2008) 1892.
[41] X.Y. Wang, K.S. Yeo, C.S. Chew, B.C. Khoo, A SVD–GFD scheme for computing 3D incompressible viscous fluid flows, Comput. Fluids 37 (2008) 733.
[42] G. Hämmerlin, K.H. Hoffmann, Numerical Mathematics, Springer-Verlag, New York, 1991. translated by L. Schumaker.
[43] C.W. Hirt, A.A. Amsden, J.L. Cook, An arbitrary Lagrangian–Eulerian computing method for all flow speeds, J. Comput. Phys. 14 (1974) 227.
[44] H. Guillard, C. Farhatm, On the significance of the geometric conservation law for flow computations on moving meshes, Comput. Meth. Appl. Mech. Eng. 190 (2000) 1467.
[45] A.J. Chorin, Numerical solution of the Navier–Stokes equations, Math. Comput. 22 (1968) 745.
[46] A.J. Chorin, On the convergence of the discrete approximations to the Navier–Stokes equations, Math. Comput. 23 (1969) 341.
[47] J. Kim, P. Moin, Application of a fractional step method to incompressible Navier–Stokes equations, J. Comput. Phys. 59 (1985) 308.
[48] D.L. Brown, R. Cortez, M.L. Minion, Accurate projection methods for the incompressible Navier–Stokes equations, J. Comput. Phys. 168 (2001) 464.
[49] H. Gouraud, Continuous shading of curved surfaces, IEEE Trans. Comput. C-20 (1971) 623.
[50] G. Thürmer, C.A. Wüthrich, Normal computation for discrete surfaces in 3D space, Comput. Graph. Forum 16 (1997) C15.
[51] K.S. Yeo, S.J. Ang, C. Shu, Simulation of fish swimming and manoeuvring by an SVD–GFD method on a hybrid meshfree-Cartesian grid, Comput. Fluids, in press. doi:10.1016/j.compfluid.2009.08.002.
[52] 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.
[53] C. Shu, L. Wang, Y.T. Chew, Numerical computation of three dimensional incompressible Navier–Stokes equations in primitive variable form by DQ method, Int. J. Numer. Meth. Fluids 43 (2003) 345.
[54] K.L. Wong, A.J. Baker, A 3D incompressible Navier–Stokes velocity-vorticity weak form finite element algorithm, Int. J. Numer. Meth. Fluids 38 (2001) 99.
[55] B. Jiang, T.L. Lin, L.A. Povinelli, Large-scale computation of incompressible viscous flow by least-squares finite element method, Comput. Meth. Appl. Mech. Eng. 114 (1994) 213.
[56] S.C.R. Dennis, S.N. Singh, D.B. Ingham, The steady flow due to a rotating sphere at low and moderate Reynolds numbers, J. Fluid Mech. 101 (1980) 257.
[57] X. Lv, Y. Zhao, X.Y. Huang, G.H. Xia, Z.J. Wang, An efficient parallel/unstructured-multigrid preconditioned implicit method for simulating 3D unsteady compressible flows with moving objects, J. Comput. Phys. 215 (2006) 661.
[58] R. Mei, Flow due to an oscillating sphere and an expression for unsteady drag on the sphere at finite Reynolds number, J. Fluid Mech. 270 (1994) 133.
[59] S.P. Sane, The aerodynamics of insect flight, J. Exp. Biol. 206 (2003) 4191.
[60] H. Liu, K. Kawachi, A numerical study of insect flight, J. Comput. Phys. 146 (1998) 124.
[61] M. Sun, X. Yu, Flows of two airfoils performing fling and subsequent translation and subsequent clap, Acta Mech. Sin. 19 (2003) 103.
[62] M. Sun, J. Tang, Unsteady aerodynamic force generation by a model fruit fly wing in flapping flight, J. Exp. Biol. 205 (2002) 55.
[63] H. Aono, F. Liang, H. Liu, Near- and far-field aerodynamics in insect hovering flight: an integrated computational study, J. Exp. Biol. 211 (2008) 239.
[64] S.P. Sane, M.H. Dickinson, The control of flight force by a flapping wing: lift and drag production, J. Exp. Biol. 204 (2001) 2607.
[65] J.M. Birch, M.H. Dickinson, The influence of wing–wake interactions on the production of aerodynamic forces in flapping flight, J. Exp. Biol. 206 (2003) 2257.