Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 207 (2005) 151–172
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Parallel computation of unsteady incompressible viscous flows around moving rigid bodies using an immersed object method with overlapping grids

C.H. Tai [a,b], Y. Zhao [a,*], K.M. Liew [a,b]

[a] *School of Mechanical and Production Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798, Republic of Singapore*
[b] *Nanyang Centre for Supercomputing and Visualisation, Nanyang Technological University, Nanyang Avenue, Singapore 639798, Republic of Singapore*

Received 11 February 2004; received in revised form 12 January 2005; accepted 13 January 2005
Available online 3 March 2005

Abstract

A novel immersed object method is developed for simulating two-dimensional unsteady incompressible viscous flows around arbitrarily moving rigid bodies. It has been implemented in a parallel unstructured finite volume incompressible Navier–Stokes solver, based on the artificial compressibility (AC) approach using a higher-order characteristics-based upwind scheme and matrix-free implicit dual time-stepping. In the immersed object method, an object is immersed in the flow field, and it is supposed to contain frozen fluid, which moves like a solid body. This is realized by introducing source terms in the momentum equations during the AC sub-iterations. An internal mesh within the object is employed to search and locate all the Eulerian nodes within the object in every time step for imposing the source terms. Unlike many existing methods, this method does not require complex searching, extrapolation and interpolation to find the intersections of the object boundary with the unstructured background mesh and assign flow condition onto the object boundary. If it is necessary to capture the boundary layer accurately, then a dense overlapping grid can then be constructed around the object for further refined calculation. The immersed object method has been used to simulate steady and unsteady incompressible viscous flows over a stationary circular cylinder, rotating square cylinder and moving disk in cavity. The results agree well with published numerical solutions and experimental measurements.

Keywords: Immersed object method; Overlapping grids; Characteristics-based method; Matrix-free implicit method; Parallel computation; Unstructured grid; Unsteady incompressible viscous flow

* Corresponding author. Tel.: +65 6790 4545.
E-mail address: myzhao@ntu.edu.sg (Y. Zhao).

0021-9991/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2005.01.011

1. Introduction

The simulation of fluid flow with arbitrarily moving solid bodies is one of the challenges in computational fluid dynamics (CFD). The development of accurate, robust and efficient methods that can tackle this problem would be very useful for many practical applications. There are a few existing methods designed for this purpose. One approach that has been demonstrated to be effective is the immersed boundary method [1], which was originally developed for the simulation of blood flow in the human heart. In this method, a fixed or Eulerian mesh is used and the moving bodies (muscular heart walls and valves) are approximated by a series of control points on which tension forces are imposed pointwise and distributed in the neighbouring elements. Here, these forces are not known a priori and are calculated using theoretical models. It should be noted that the original method is suitable for calculating the moving boundary as one-dimensional curves (i.e., the thickness is negligible), and one cannot include the inertia effect and jumps in flow properties cross the boundary. The immersed boundary method has since been extended to a number of problems, for examples, flows over a circular cylinder [2], motion of flexible pulp fibres [3], and modelling arteriolar flow and mass transport [4]. In the extended method, the object boundary has to be well defined and its intersections with the mesh have to be calculated in every time step. In addition, elaborate interpolation is used to impose interfacial conditions directly onto the object boundary or a source term is distributed to its neighbouring nodes to achieve these conditions. Goldstein et al. [5,6] formulated a technique related to the immersed boundary method to introduce solid surfaces into a simulated flow field. This technique is known as virtual boundary method. In the simulation of no-slip boundary condition using this method [5], the embedded boundary in the fluid is to be treated by applying a force field to the fluid and this force is chosen to lie along a desired surface and to have a magnitude and direction opposing the local flow such that the flow is brought to rest on an element of the surface. Since the force field is not known initially, it is calculated based on a feedback-forcing scheme so that the velocity on the boundary is used to determine the desired force distribution. This method needs to specify two large negative constants, α and β, so that the imposed fluid velocity is close to the interface velocity. It is noted that this method may introduce spurious oscillations and restricts the physical time step size. Saiki and Biringen [7] have applied this virtual boundary method to compute a two-dimensional flow around stationary and moving cylinders. Tseng and Ferziger [8] also extended the immersed boundary method to achieve higher-order extrapolation/interpolation for the interfacial boundary condition for complex geometries by employing the ghost fluid method [9,10]. The combination resulted in the so-called ghost-cell immersed boundary method, which was then applied to study flow past a circular cylinder and large eddy simulation of turbulent flows.

Glowinski et al. [11,12] proposed a fictitious domain method for the numerical solutions of three-dimensional elliptic problems with Dirichlet boundary conditions for modelling incompressible viscous flow. The method combines finite element approximations, time discretization by operator splitting and Lagrangian multiplier for imposing the interfacial velocity condition. The advantage of this method is that it uses Eulerian mesh, therefore allowing the use of existing fast solvers. Elaborate interpolation is also needed to calculate the boundary condition based on the Lagrangian multiplier, and iteration as well as a feedback mechanism is required to achieve the interfacial condition. This method has been employed to simulate Stokes flow past moving rigid bodies [11], steady external three-dimensional Stokes flow and vortex shedding behind a cylinder in two-dimensional unsteady incompressible viscous flow [12].

The overlapping grid method has been one popular hybrid Eulerian/Lagrangian technique in CFD, allowing flows with arbitrarily shaped object geometries to be computed. Among them the Chimera mesh method [13] can deal with large-displacement problems by using two sets of meshes, one is used to cover the local area surrounding the objects and the other is a background mesh which covers the rest of the flow field. Flow field variables must be interpolated back and forth between these two sets of overlapping meshes. The Chimera method is difficult to implement in computer codes, as the method requires cutting

holes in the background mesh before superimposing the overlapping mesh into the flow field and building the overlapping region in every time step.

This work attempts to develop a so-called immersed object method for simulating unsteady incompressible viscous flows around moving rigid bodies. It is assumed that fluid within an object is frozen and moves like a solid body. This condition is enforced by adding source terms to the momentum equations for those Eulerian nodes lying inside the object. This is different from the direct forcing method [14] in that the direct forcing method is very much like the immersed boundary method and requires object–mesh intersection calculation, extrapolation and interpolation to impose boundary condition onto object surfaces. The Eulerian nodes inside the object can be efficiently searched and identified by using an internal mesh within the object based on a quatree search in every time step. Unlike the existing methods as surveyed above, this method does not require any calculation to determine object–mesh intersections, which can be very complicated for 2D/3D surfaces and unstructured meshes. In addition, no interpolation or extrapolation of flow properties for imposing interfacial conditions is needed. For high Reynolds-number flow, where its boundary layer is thin and has to be accurately captured, an overlapping mesh, that surrounds the object, can be added and the Eulerian solution can be transferred to the boundary of the overlapping mesh to perform further calculation to obtain refined solution, as well as accurate lift and drag coefficients. Compared with the Chimera method, this approach is relatively simpler to implement in computer codes because no hole-cutting is required in the Eulerian mesh, and this is especially advantageous when the solid body is moving arbitrarily or when there are multiple moving objects. A parallel strategy has also been developed to parallelize the method. The proposed immersed object method has been applied to simulate viscous flow over a stationary circular cylinder at both low and high Reynolds numbers, rotating square cylinder and moving circular disk in order to demonstrate and examine the performance, accuracy and robustness of the method.

2. Mathematical formulation

The non-dimensional Navier–Stokes equations in two-dimensions for incompressible unsteady flows and moving grid, modified by the artificial compressibility (AC) method, in vector form:

$$\mathbf{C}\frac{\partial \mathbf{W}}{\partial \tau} + \mathbf{K}\frac{\partial \mathbf{W}}{\partial t} + \nabla \cdot \vec{\mathbf{F}}_{\mathrm{c}} = \nabla \vec{\mathbf{F}}_{\mathrm{v}}, \tag{2.1}$$

where

$$\mathbf{W} = \begin{bmatrix} p \\ u \\ v \end{bmatrix}, \quad \vec{\mathbf{F}}_{\mathrm{c}} = \begin{bmatrix} \mathbf{U} \\ u\mathbf{U} + (p/\rho)\vec{i} \\ v\mathbf{U} + (p/\rho)\vec{j} \end{bmatrix}, \quad \vec{\mathbf{F}}_{\mathrm{v}} = \begin{bmatrix} 0 \\ \frac{1}{R}\left(\frac{\partial}{\partial x}\vec{i} + \frac{\partial}{\partial y}\vec{j}\right) \\ \frac{1}{R}\left(\frac{\partial}{\partial x}\vec{i} + \frac{\partial}{\partial y}\vec{j}\right) \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1/\beta & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.2}$$

where $\mathbf{W}$ is the flow field variable vector, $\mathbf{U} = \mathbf{U_f} - \mathbf{U_g}$ is the velocity vector, $\mathbf{U_f}$ and $\mathbf{U_g}$ are the fluid velocity and grid velocity respectively, and $\vec{\mathbf{F}}_{\mathrm{c}}$ and $\vec{\mathbf{F}}_{\mathrm{v}}$ are the convective and viscous flux vectors, respectively, β being a constant called AC, $\mathbf{K}$ is the unit matrix, except that the first element is zero and $\mathbf{C}$ is a preconditioning matrix.

The non-dimensional variables used in above equations are defined as follows:

$$(x, y) = \left(\frac{x^*}{L^*}, \frac{y^*}{L^*}\right); \quad (u, v) = \left(\frac{u^*}{U^*_\infty}, \frac{v^*}{U^*_\infty}\right); \quad t = \frac{t^*}{L^*/U^*_\infty}; \quad p = \frac{p^* - p_0}{\rho(U^*_\infty)^2}; \quad R = \frac{\rho_\infty U^*_\infty L^*}{\mu^*_\infty}.$$

3. Numerical methods

A high-order characteristics-based upwind finite-volume method on an unstructured grid [15] is used to discretize the governing equations. The 2D equation in Eq. (2.1) is transformed into an integral form and discretized on an unstructured grid. A cell-vertex scheme is adopted here. For every vertex, as shown in Fig. 1, a control volume is constructed using the median dual of the triangular grid. Spatial discretization is performed by using the integral form of the conservation equations over the control volume surrounding node P,

$$\mathbf{C}\frac{\partial}{\partial\tau}\int\int_{S_{cv}}\mathbf{W}_n\,\mathrm{d}S+\mathbf{K}\frac{\partial}{\partial}\int\int_{S_{cv}}\mathbf{W}_n\,\mathrm{d}S+\int\int_{S_{cv}}\nabla\cdot\vec{\mathbf{F}}_c\,\mathrm{d}S=\int\int_{S_{cv}}\nabla\cdot\vec{\mathbf{F}}_v\,\mathrm{d}S. \tag{3.1}$$

In order to introduce the upwind scheme using an edge-based procedure, the convective term is transformed into a summation,

$$\int\int_{S_{cv}}\nabla\cdot\vec{\mathbf{F}}_c\,\mathrm{d}S=\oint_{L_{cv}}\vec{\mathbf{F}}_c\cdot\vec{\mathbf{n}}\,\mathrm{d}\;=\sum_{=1}^{\text{bseg}}\left[\left(\vec{\mathbf{F}}_c\right)\;\cdot\vec{\mathbf{n}}\Delta\;\right], \tag{3.2}$$

where n_{bseg} is the number of the edges connected to node P, $(\vec{\mathbf{F}}_c)$ is the convection flux through the part of control volume boundary (similar to 1–C_{ij}–2 in Fig. 1). The length of the boundary is Δl_k and its effective outward normal unit vector is $\vec{\mathbf{n}}$. Therefore, all the fluxes are calculated for the edges and then collected at the two ends of each edge for updating of flow variables using the time-marching scheme. The viscous term is calculated using a cell-based method,

$$\int\int_{S_{cv}}\nabla\cdot\vec{\mathbf{F}}_v\,\mathrm{d}S=\oint_{L_{cv}}\vec{\mathbf{F}}_v\cdot\mathrm{d}\;=\sum_{=1}^{\text{cell}}(\vec{\mathbf{F}}_v\cdot\vec{\Delta}_c)\;=\frac{1}{2}\sum_{=1}^{\text{cell}}(\vec{\mathbf{F}}_v\cdot\vec{\Delta}_n)\,, \tag{3.3}$$

where $\vec{\Delta}_c$ is the part of control volume boundary in cell C_i, and $\vec{\Delta}_n$ is the edge vector of the cell edge opposite to node P of the triangle under consideration. Here, $(\vec{\mathbf{F}}_v)$ is calculated at the centre of the triangular cell, and can be obtained by using the Green's Theorem and the variables at the three vertices of the triangle. Here, n_{cell} is the number of cells surrounding node P. The viscous flux in Eq. (3.3) is actually calculated in a cell-by-cell manner and then collected at the nodes of the cells for the calculation of the residuals at all the nodes. The gradient of a flow variable Φ at the center of a cell is evaluated,

$$\text{grad}\;\Phi_c=-\frac{1}{2}\frac{\sum_{=1}^{3}\vec{\Phi}}{A}, \tag{3.4}$$

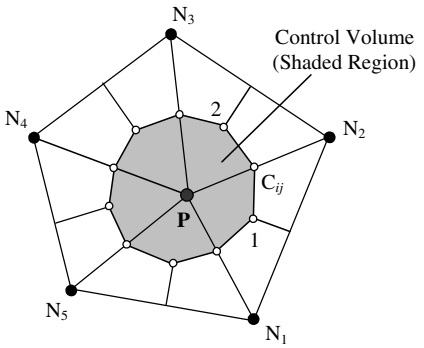


Fig. 1. Construction of control volume for node P.

where Φ_i is the flow variable at a vertex i of the cell and $\vec{}$ is the edge vector opposite to vertex i, A is the area of the triangular cell. Gradients at the vertices are obtained by a volume averaging of the gradients at the center of cells associated with the vertex under consideration.

The solution of the Navier–Stokes equations on a moving mesh requires a new grid velocities at each time step. In our work, a high-order characteristic-based scheme for incompressible flow for arbitrary unstructured 2D/3D grids proposed by Zhao and Tai [15] has been adopted. Such a scheme automatically introduces upwinding along the characteristic directions at both the differential equation and discretized-equation levels, without relying on any artificial viscosity. Combined with the third-order MUSCL interpolation, it can produce accurate and stable solution on unstructured grids. The left and right state vectors $\mathbf{W}_L$ and $\mathbf{W}_R$ on both sides of a control volume surface for the characteristics-based scheme are evaluated using an upwind-biased MUSCL interpolation scheme:

$$\mathbf{W}_L = \mathbf{W} + \frac{1}{2}[(1-\kappa)\vec{} \cdot \nabla \mathbf{W} + \kappa\Delta^+], \tag{3.5}$$

$$\mathbf{W}_R = \mathbf{W} - \frac{1}{2}[(1-\kappa)\vec{} \cdot \nabla \mathbf{W} + \kappa\Delta^-], \tag{3.6}$$

where $\Delta^+ = \Delta^- = \mathbf{W} - \mathbf{W}$, subscripts i and j represent indices for the two end nodes defining the edge, κ is set to 1/3, which corresponds to a third-order accuracy. $\vec{}$ is the vector representing the edge, which points from node P to its neighbouring node under consideration. The gradients of $\mathbf{W}$ at i and j are calculated by volume-averaging the gradients of the cells that surround i and j.

Finally, for a given node P, the spatially discretized equations form a system of coupled ordinary differential equations, which can be reformulated as

$$\mathbf{C}\frac{\partial}{\partial\tau}(\Delta S_{cv}\mathbf{W}_n) + K\frac{\partial}{\partial}(\Delta S_{cv}\mathbf{W}_n) = -\left\{\sum_{=1}^{bseg}[(\vec{\mathbf{F}}_c) \cdot \vec{n}\Delta] - \frac{1}{2}\sum_{=1}^{cell}(\vec{\mathbf{F}}_v \cdot \vec{n}\Delta_P)\right\} = -R(\mathbf{W}_P), \tag{3.7}$$

where $R(\mathbf{W}_P)$ represents the residual error which includes the convective and diffusive fluxes and ΔS_{cv} is the control volume of node P. An implicit scheme is adopted for Eq. (3.7) and the time dependent term is discretized using a second-order-accurate backward differencing scheme,

$$\mathbf{C}\frac{\partial}{\partial\tau}(\Delta S_{cv}\mathbf{W}_n) = -R(\mathbf{W}_n^{+1}) - \mathbf{K}\left(\frac{1.5\Delta S_{cv}^{+1}\mathbf{W}_n^{+1} - 2.0\Delta S_{cv}\mathbf{W}_n + 0.5\Delta S_{cv}^{-1}\mathbf{W}_n^{-1}}{\Delta}\right) = \tilde{R}(\mathbf{W}_n^{+1}), \tag{3.8}$$

where the superscript $(n+1)$ denotes the physical time level $(n+1)\Delta t$ and all the variables are evaluated at this time level, $\tilde{R}(\mathbf{W}_n^{+1})$ is the new modified residual which contains both the time derivative and flux vectors. The derivative with respect to a pseudo time τ is discretized as

$$\mathbf{C}\Delta S_{cv}^{+1}\frac{\mathbf{W}_n^{+1,\ +1} - \mathbf{W}_n^{+1,}}{d\tau} = \tilde{R}(\mathbf{W}_n^{+1,\ }), \tag{3.9}$$

whose solution is sought by marching to a pseudo steady state in τ. Here, m and $(m+1)$ denote the initial and final pseudo time levels. Once the artificial steady state is reached, the derivative of $\mathbf{W}_p$ with respect to τ becomes zero, and the solution satisfies $\tilde{R}(\mathbf{W}_n^{+1}) = 0$. Hence, the original unsteady Navier–Stokes equations are fully recovered. Therefore, instead of solving each time step in the physical time domain (t), the problem is transformed into a sequence of steady-state computations in the artificial time domain (τ). Eq. (3.9) is integrated in pseudo time by an explicit five-stage Runge–Kutta scheme. However, the pseudo time step size may be severely restricted if the physical time step size is very small. Hence, an implicit dual time stepping is adopted here.

An approximate flux function is introduced here to simplify the implicit time stepping calculation. Following [16], the total flux (including both inviscid and viscous fluxes) across a control volume surface associated with a certain edge ij can be approximated as

$$\mathbf{F} \approx \frac{1}{2}\left[\vec{\mathbf{F}}^{c}\cdot\vec{\mathbf{n}} + \vec{\mathbf{F}}^{c}\cdot\vec{\boldsymbol{n}} - |\lambda|(\mathbf{W} - \mathbf{W})\right],$$

where λ_{ij} is the spectral radius associated with edge ij, is given as

$$\lambda = \vec{\mathbf{U}}\cdot\vec{\boldsymbol{n}} + \sqrt{\left(\vec{\mathbf{U}}\cdot\vec{\boldsymbol{n}}\right)^{2} + \beta^{2}}.$$

A Taylor series expansion is performed for the residual in Eq. (3.9) with respect to the pseudo time for node i,

$$\tilde{R}(\mathbf{W}^{+1}) = \tilde{R}(\mathbf{W}) + \frac{\partial\tilde{R}}{\partial\mathbf{W}} + \sum_{=1}\frac{\partial\tilde{R}}{\partial\mathbf{W}}\Delta\mathbf{W}.$$

And in all the R terms, the Taylor series expansions of the fluxes are,

$$\frac{\partial\mathbf{F}}{\partial\mathbf{W}} = \frac{1}{2}\left[\frac{\partial\mathbf{F}^{c}}{\partial\mathbf{W}} - |\lambda|\right] \quad \text{and} \quad \frac{\partial\mathbf{F}}{\partial\mathbf{W}} = \frac{1}{2}\left[\frac{\partial\mathbf{F}^{c}}{\partial\mathbf{W}} + |\lambda|\right].$$

And for the physical time-dependent terms, we have

$$\mathbf{W}^{+1} = \mathbf{W} + \Delta\mathbf{W}.$$

After combining all the residuals at every node in the flow field into a vector, we have

$$R(\mathbf{W}^{+1,+1}) = R(\mathbf{W}^{+1,}) + A\Delta\mathbf{W},$$

where $A = \left\{\frac{\partial R}{\partial \mathbf{W}}\right\}$.

And the whole-field equivalent of Eq. (3.9) can then be re-written as

$$\mathbf{C}\Delta S_{cv}^{+1}\left(\frac{\Delta + 1.5\Delta\tau}{\Delta} - \frac{A\Delta\tau}{\Delta S_{cv}^{+1}}\right)\frac{\Delta\mathbf{W}}{\Delta\tau} = R^{+1,} - \mathbf{K}\left(\frac{1.5\Delta S_{cv}^{+1}\mathbf{W}^{+1,} - 2.0\Delta S_{cv}\mathbf{W} + 0.5\Delta S_{cv}^{-1}\mathbf{W}^{-1}}{\Delta}\right),$$

that is,

$$\mathbf{C}\Delta S_{cv}^{+1}\tilde{A}\frac{\Delta\mathbf{W}}{\Delta\tau} = \tilde{R}^{+1,}, \tag{3.10}$$

thus,

$$\mathbf{C}\Delta S_{cv}^{+1}\frac{\Delta\mathbf{W}}{\Delta\tau} = \tilde{\tilde{R}}^{,+1}, \tag{3.11}$$

where

$$\tilde{\tilde{R}}^{,+1} = \tilde{A}^{-1}\tilde{R}^{+1,} \quad \text{and} \quad \tilde{A} = \frac{\Delta + 1.5\Delta\tau}{\Delta} - \frac{A\Delta\tau}{\Delta S_{cv}}.$$

Therefore,

$$\tilde{R}^{+1,} = R^{+1,} - \mathbf{K}\left(\frac{1.5\Delta S_{cv}^{+1}\mathbf{W}^{+1,} - 2.0\Delta S_{cv}\mathbf{W} + 0.5\Delta S_{cv}^{-1}\mathbf{W}^{-1}}{\Delta}\right).$$

Further approximation can be introduced in order to achieve matrix-free computation. If we employ point implicit treatment to the preceding equations, then only the diagonal terms in $\tilde{A}$ are used in the pseudo time stepping. As a result, the equation for every node can now be written as

$$C\Delta S_{cv}^{+1}\frac{\Delta \mathbf{W}}{\Delta\tau} = \tilde{R}^{,+1}, \tag{3.12}$$

where

$$\tilde{R}^{,+1} = \tilde{A}^{-1}\tilde{R}^{,+1} \quad \text{and} \quad \tilde{A}^{-1} = \left(\frac{\Delta + 1.5\Delta\tau}{\Delta} - \frac{A\ \Delta\tau}{\Delta S_{cv}}\right)^{-1}.$$

Pseudo time stepping is then performed on Eq. (3.12). For a five-stage scheme, the stage coefficients are

$$\alpha_1 = 1/4, \quad \alpha_2 = 1/6, \quad \alpha_3 = 3/8, \quad \alpha_4 = 1/2, \quad \alpha_5 = 1.$$

4. Parallel implementation

This work adopts a parallelization strategy for the numerical simulations by using geometric domain decomposition and the single program multiple data (SPMD) programming paradigm [17]. Here, data communication between domains is based on the message-passing interface (MPI) [18]. Domain decomposition of a mesh into a set S of sub-domains that may be allocated to a set of processors involves finding a partition of the mesh so that equal amount of computational time on each processor is achieved. METIS [19] is employed to produce the partitions of grids (see Fig. 2).

The nodes and elements that are allocated uniquely to a processor are referred to as core mesh components in this work and each processor calculates the flow field variables and nodal gradients for them. Each sub-domain is enclosed with a layer of *ghost* nodes and overlapping elements, which overlap the sub-domains along the inter-processor boundaries as depicted in Figs. 3 and 8. These *ghost* nodes store flow field variables and nodal gradients which are transferred from neighbouring sub-domains for the solution of variables within the sub-domain. Therefore no computation is performed on the *ghost* nodes in the sub-domain under consideration. Communication between these core and *ghost* nodes is based on the MPI. The data flow direction is always from the core nodes to the *ghost* nodes.

The main concept of the algorithm to identify the *ghost* nodes and overlapping elements is that those elements along the inter-processor boundaries with nodes having different partition numbers and they are considered as overlapping elements which are cut through by the partition lines. And it should be noted

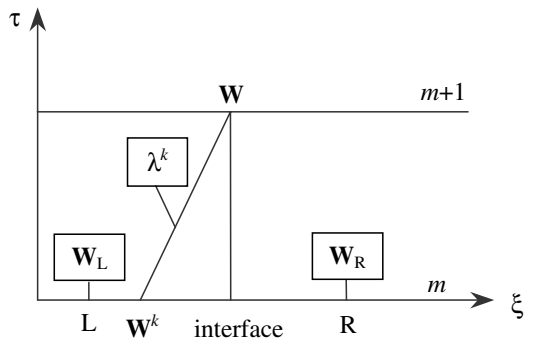


Fig. 2. τ–ξ co-ordinate.

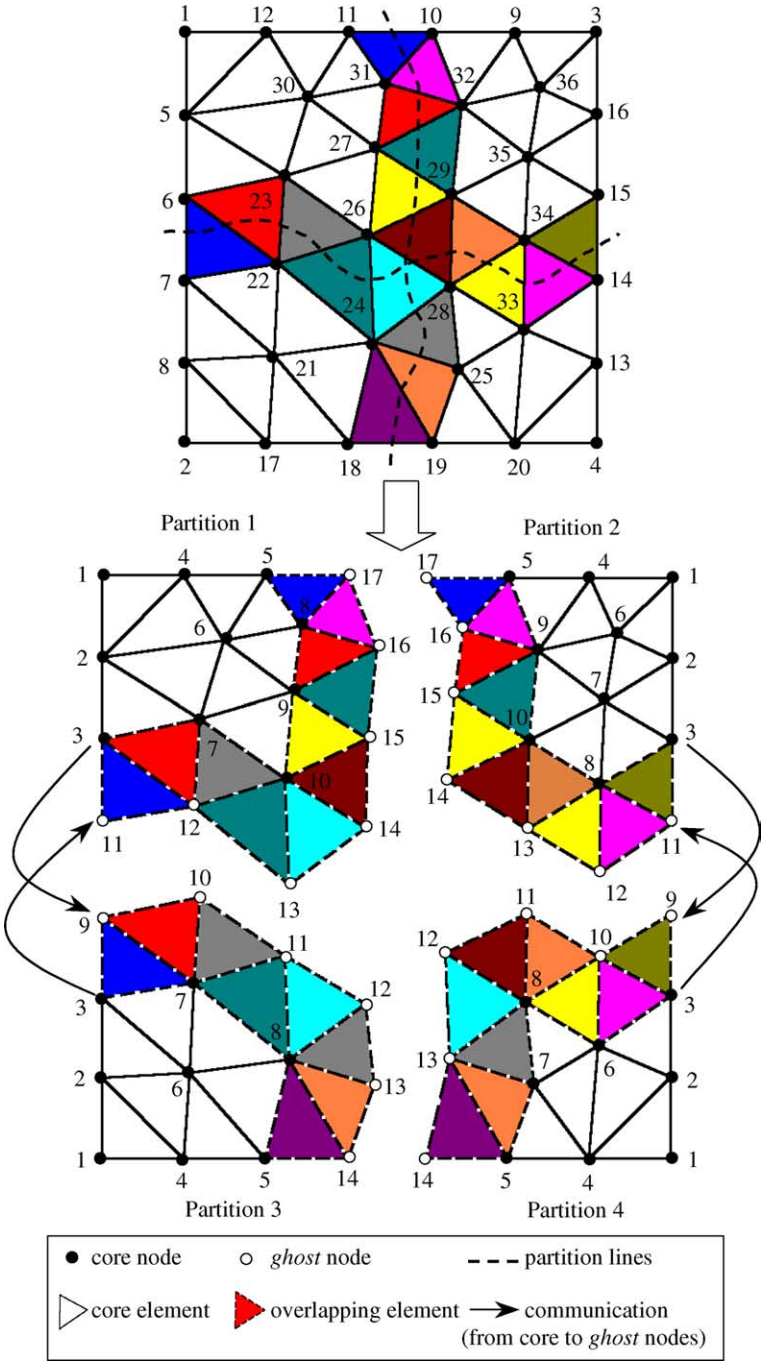


Fig. 3. Mesh decomposed into 4 partitions showing each sub-domain extended with a layer of *ghost* nodes and overlapping elements.

that the nodes forming these elements are a mixture of core and *ghost* nodes [17]. Basically, the *ghost* node of a partition is the mirror image of the core node of its neighbouring partitions. For example, as shown in Fig. 3, the *ghost* node number 11 of partition 3 is the mirror image of core node number 10 of partition 1

Table 1
Communication between core and *ghost* nodes for partition 1 depicted in Fig. 3

Global node number	Send		Receive	
	Partition number	Local node number (core node)	Partition number	Local node number (*ghost* node)
11	1	5	2	17
31		8	2	16
27		9	2	15
26		10	2	14
26		10	3	11
23		7	3	10
6		3	3	9
26		10	4	12
10	2	5	1	17
32		9		16
29		10		15
24	3	8		13
22		7		12
7		3		11
28	4	8		14

and vice-versa. Therefore, making use of this correlation between the core and *ghost* nodes, the rest of the *ghost* nodes can be identified so as to build the control volume for the core node.

The communication between the core and *ghost* nodes requires a data structure for each sub-domain, which holds the nodes and processor numbers to be sent to and received from. The mesh in each sub-domain is renumbered as a new mesh consisting of n_e elements and n_p nodes, where n_e and n_p are the local numbers of elements and nodes, respectively. In this work, communication and writing of variables to files are based on local numbering rather than global numbering and therefore, no translation back from local to global numbering is necessary. With reference to the grid decomposed in Fig. 3, the data structure established for communication between the core and *ghost* nodes is shown in Table 1 and it only shows the data structure for partition one and the data structure for the rest of the partitions will follow exactly the same fashion.

5. The immersed object method

The immersed object method adds source terms to the momentum equations for nodes inside solid objects in the flow field to force the fluid inside the objects to behave like solid ones. The advantage of this is that bodies of almost arbitrary shapes can be added without grid restructuring, a procedure which is often time-consuming and computationally expensive. Furthermore, multiple bodies may be simulated, and relative motion of those bodies may be accomplished at reasonable computational cost. The immersed object method developed in this study does not require specifying any boundary conditions to the stationary or moving object and thus no object–mesh intersection calculation and no elaborate interpolation at the interface are needed in each time step. And the source terms are only used in the AC sub-iterations and they will disappear when solid body motion is achieved.

A source term $\mathbf{F}$ is introduced into the momentum equations for the nodes inside an object such that a desired velocity distribution $\mathbf{V}_o$ can be imposed within the object,

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{U}\cdot\nabla\mathbf{U} = -\frac{\nabla p}{\rho} + \mu\cdot\nabla^2\cdot\mathbf{U} + \mathbf{F} \tag{5.1}$$

and the continuity equation remains unchanged,

$$\nabla \cdot \mathbf{U} = 0. \tag{5.2}$$

In principle, there are no restrictions for the object velocity distribution $\mathbf{V}_o$ and for the shape and motion of Ω; therefore a wide variety of boundary conditions can be imposed. The main advantage of this approach is that $\mathbf{F}$ can be prescribed on a regular mesh so that accuracy and efficiency of the solution procedure on simple grids can be maintained. Since no interpolation or extrapolation is required to impose the interfacial condition, the order of accuracy of the method remains the same as the baseline finite volume scheme.

The time-discretized equation with the introduction of the pseudo-time τ of Eq. (5.1) can be written as

$$\frac{\Delta V_{cv}(\mathbf{U}^{+1} - \mathbf{U})}{\Delta\tau} = \tilde{\tilde{R}} + \mathbf{F}. \tag{5.3}$$

In order to drive the velocity $\mathbf{U}^{m+1}$ to the desired value $\mathbf{V}_o$, within every physical time step, a number of subiteration is performed on the discrete Navier–Stokes equation (5.3) using a five-stage Runge–Kutta scheme:

$$\mathbf{U}^{+1} = \mathbf{U} + \alpha \frac{\Delta\tau}{\Delta V_{cv}}\left(\tilde{\tilde{R}} + \mathbf{F}\right), \tag{5.4}$$

where $\tilde{\tilde{R}}$ is defined in Eq. (3.12) for the matrix-free implicit dual time-stepping computation. α_m is the stage coefficients for the five-stage Runge–Kutta scheme. At the end of the subiteration, the original Navier–Stokes equations are recovered:

$$\tilde{\tilde{R}} = 0.$$

To force the fluid velocity within the object to be the object velocity (i.e., $\mathbf{U}^{n+1} = \mathbf{V}_o$), the source term $\mathbf{F}$ is given as follows,

$$\mathbf{F} = \begin{cases} -\tilde{\tilde{R}} + \frac{1}{\alpha}\frac{\Delta V_{cv}}{\Delta\tau}(\mathbf{V}_o - \mathbf{U}), & \text{on and within } \Omega; \\ 0 \quad \text{elsewhere.} \end{cases} \tag{5.5}$$

It should be noted that the source term is only used in the AC sub-iterations and it becomes zero when the solution becomes convergent.

In this work, we use two methods to define the physical boundary of the immersed object. The first method is to specify the geometric equation of the immersed object within the computational domain. The second method is to generate an internal grid inside the object. For complex immersed objects, the latter is found to be more robust and accurate than the former. The source term $\mathbf{F}$ of Eq. (5.5) is imposed on those fluid nodes that fall on and within the immersed object. Fig. 4 shows a circular disk grid immersed into a square cavity domain where it is used to classify those fluid nodes in the computational domain as solid nodes. From the figure, it can be seen that the computational mesh that is covered by the immersed objects needs to be refined so that the physical boundary can be defined as accurate as possible. The search algorithm for fluid nodes that are covered by the immersed object is based on the concept of dot product of two vectors: the unit normal vector oriented *inward* from an edge, $\vec{e}$, and the vector $\vec{r}$ which points from the centre of the edge to a node under consideration. The dot product of these two vectors, $(p_n)^{n_{edg}}$, for the three edges,

$$(p_n)^{n_{edg}} = \{\vec{n}\}\cdot\{\vec{e}\} = \{n_x\ n_y\}\cdot\begin{Bmatrix} e_x \\ e_y \end{Bmatrix} = (n_x e_x + n_y e_y)^{n_{edg}} \geqslant 0, \quad n_{edg} = 1\text{–}3, \tag{5.6}$$

where the superscript n_{edg} is the edge number of the cell. And $(p_n)^{n_{edg}}$ must be positive for all the three edges if the fluid node under consideration falls within the cell of the immersed object as depicted in Fig. 5 and

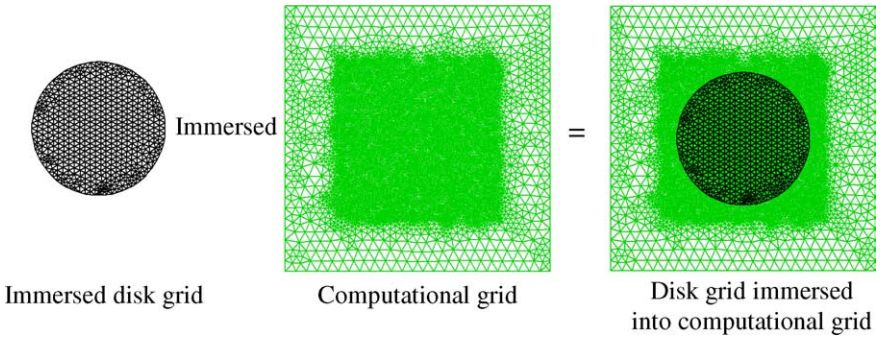


Fig. 4. Immersed object grid into computational domain.

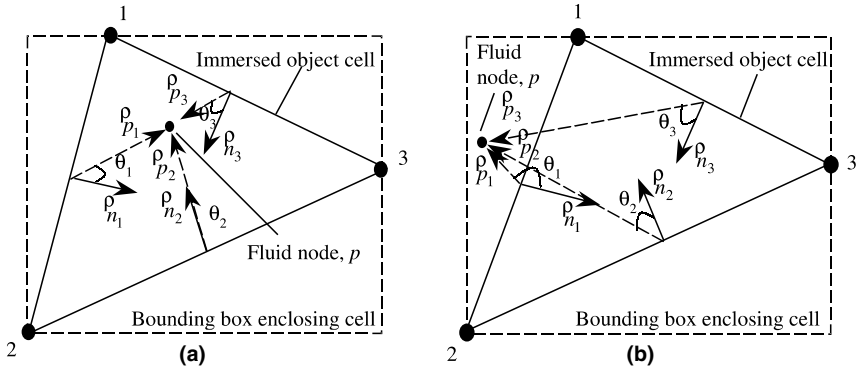


Fig. 5. (a) Fluid node falls within a immersed object cell using dot product; (b) fluid node does not fall within a immersed object cell.

this fluid node is classified as a solid node. To reduce searching time, a quadtree search method is employed. Before the search, both the computational domain covered by the immersed object and the internal grid inside the object are decomposed into a number of square zones, in which searching for a particular fluid node is only done within the related square zones, instead of searching the entire flow field. For parallel implementation, every processor has a copy of the internal object grid, as shown in Fig. 6, and a search

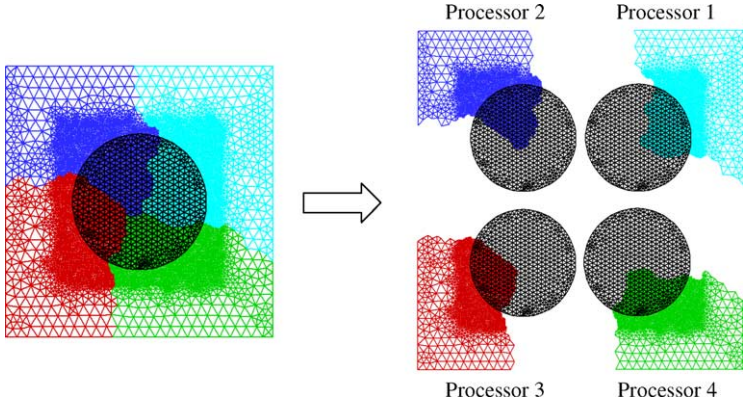


Fig. 6. Immersed object grid embedded in every processor.

for solid nodes (i.e., nodes inside the solid body) is done within its own processor using the search algorithm described above. For moving body problems, the search algorithm is performed at the beginning of each new time step once the immersed object has been moved to a new position and the speed of searching is found to have significant impact on the overall performance.

6. Overlapping grid

There are three kinds of meshes being constructed in the immersed object method. The first mesh is the Eulerian or background mesh which remains stationary throughout the computation. The second mesh is a Largrangian or overlapping mesh which wrap around the object and moves with it. The third is the internal grid inside the object, which is only used to define the boundary of the object to help identify nodes covered by the object at any time instant. The overlapping grid, which does not need to cover the entire flow field, is constructed with high mesh density near the object surface. This helps to fully resolve the physical boundary and enables accurate resolution of the boundary layer. To determine the solution on the overlapping grid, the solution on the background mesh is interpolated onto the overlapping grid as boundary conditions and this requires some form of interpolation in a transfer operator. Before the interpolation of solution takes place, it is necessary to determine the relative positional relations between the nodes and cells of the overlapping mesh and their counterparts in the background mesh. The search algorithm developed for identifying solid nodes covered by solid bodies is adopted here, as shown in Eq. (5.6) and Fig. 5. For moving body problems, this search is performed at the beginning of very time step and its efficiency also has significant impact on the overall performance of flow simulation and the search algorithm used here is found to be quite efficient in this work. Following the interpolation algorithm developed for multigrid in [17], the flow field variables on the background grid are interpolated onto the overlapping boundary according to the following equation:

$$\mathbf{W}_{\mathrm{OG}} = \frac{A_{\mathrm{BG1}}\mathbf{W}_{\mathrm{BG1}} + A_{\mathrm{BG2}}\mathbf{W}_{\mathrm{BG2}} + A_{\mathrm{BG3}}\mathbf{W}_{\mathrm{BG3}}}{A_{\mathrm{BG1}} + A_{\mathrm{BG2}} + A_{\mathrm{BG3}}}, \tag{6.1}$$

where $\mathbf{W}_{\mathrm{OG}}$ is the solution for the overlapping boundary node, $\mathbf{W}_{\mathrm{BG1}}$, $\mathbf{W}_{\mathrm{BG2}}$ and $\mathbf{W}_{\mathrm{BG3}}$ are the solutions for the background mesh. A_{BG1}, A_{BG2} and A_{BG3} are the areas of the corresponding triangles opposite to the background cell nodes. The flow field values at the overlapping boundary node, which is contained in the background cell formed by nodes BG1, BG2 and BG3, is a weighted average of the values at those nodes as shown in Fig. 7.

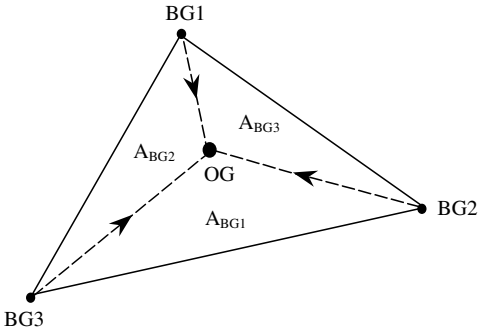


Fig. 7. Transfer of flow field values from the background mesh to the overlapping grid.

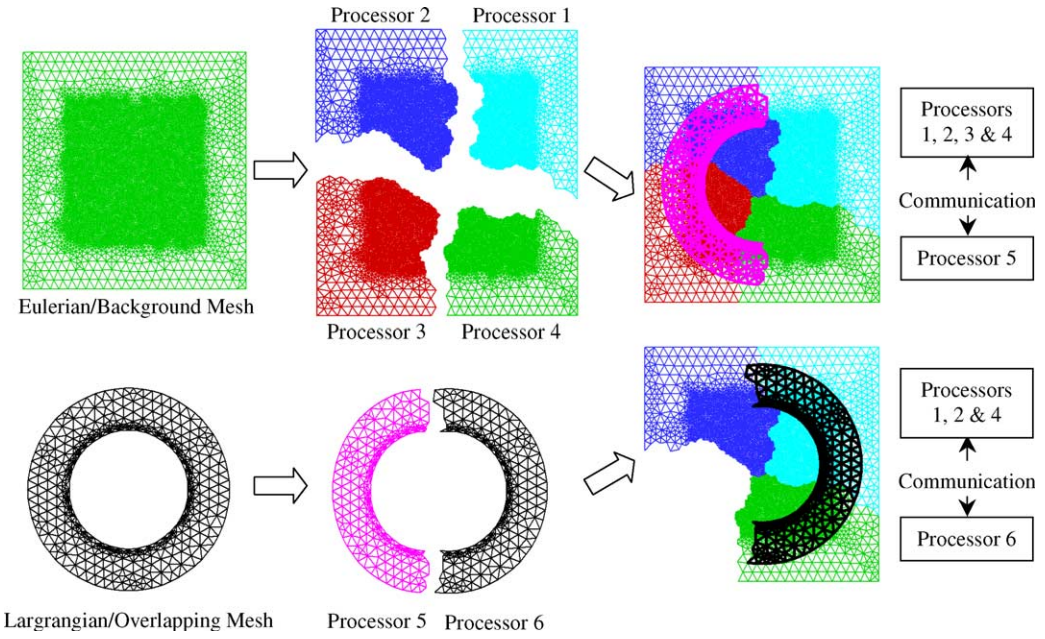


Fig. 8. Mesh partitions and communications between background and overlapping grids.

For parallel computation, the overlapping grid is decomposed into a number of sub-domains for different processors, which are independent of the computation for the background mesh. For example, as shown in Fig. 8, the background and overlapping grids are decomposed into four and two sub-domains, respectively. The sub-domains of the background mesh are distributed to processors 1, 2, 3 and 4, whereas the sub-domains of the overlapping grid are distributed to processors 5 and 6. From the figure, it can be seen that the overlapping grid for processor 5 overlaps with those for processors 1, 2, 3 and 4, and communications will take place between these processors. Likewise for the overlapping grid for processor 6, communications will take place between processor 6 and processors 1, 2 and 4. The parallel computation on the overlapping grid is exactly the same as that employed for the background mesh, except that interpolation is needed at the beginning of each new time step.

7. Initial and boundary conditions

At the solid wall, a *no-slip* condition is imposed for viscous flow by setting the flow velocity equal to that of the body. For the Eulerian mesh, an uniform velocity profile is given as free stream boundary condition, while the velocity at down stream boundary is calculated. Pressure at the free stream is calculated and pressure at the down stream boundary is fixed at a constant value. The flow field values are set to the free-stream values at the start of the computation.

8. Computational results

To demonstrate and examine the performance, accuracy and robustness of the immersed object method with overlapping grids, several test cases are computed. The test cases are viscous flow over a stationary

circular cylinder at both low and high Reynolds numbers, flow over a rotating square cylinder, as well as flow in a cavity with a disk undergoing both rotation and translation. Parallel computations are performed on an SGI Origin 3400 machine with 32 processors based on Silicon Graphics Scalable Distributed-Shared-Memory (DSM) Multi-processing architecture. It has 8 nodes and each node has four MIPs 64 Bit R12000 400 MHz/8 MB RISC CPUs.

8.1. Viscous flow over a stationary circular cylinder

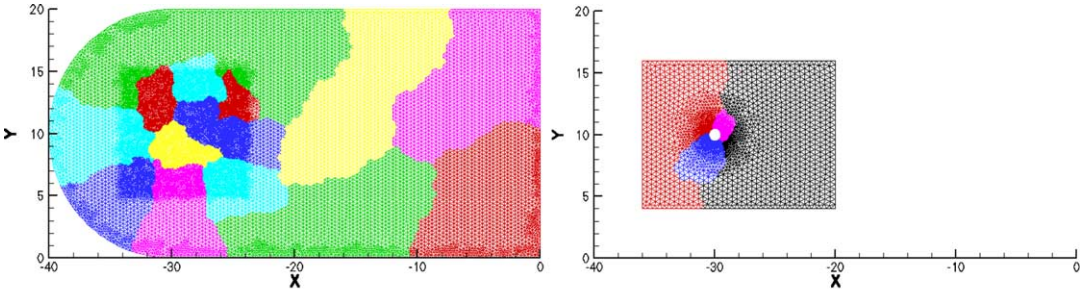
The first test case considered is viscous flow over a stationary circular cylinder at Reynolds numbers (*Re*) of 41 and 200. The third-order characteristics-based scheme is used in the computations together with the matrix-free implicit dual-time stepping scheme and the second-order temporal discretization. Since the cylinder is stationary, fluid velocity within the cylinder is set to an object velocity, $\mathbf{V}_o = \mathbf{0}$.

8.1.1. Steady viscous flow (*Re* = 41)

The background mesh consists of 20,869 nodes and 41,355 elements and it is partitioned into 16 sub-domains for parallel computation, and the overlapping grid consists of 4380 nodes and 8494 elements and is partitioned into 4 sub-domains, as shown in Fig. 9. The number of pseudo sub-iterations per time step is set to 100, CFL is set to 3.5 and the non-dimensional physical time step is set at 0.5. The computational results obtained using the immersed object method are compared with the numerical results of Tai and Zhao [17] and the experimental measurements of Dyke et al. [20]. The wake formed behind the cylinder agrees well with both numerical and experimental solutions as depicted in Fig. 10, where the separation point occurs at the same location. A comparison of the aspect ratio (separation bubble length, *S* over cylinder diameter, *d*) with the experimental results obtained by Nishioka and Sato [21] is given. An aspect ratio of around 2.3 is obtained using the immersed object method with an overlapping grid and 2.4 from the experimental result as shown in Fig. 11. Fig. 12 shows the convergence history plot using the present method and the normalized residual is reduced to an order of 1×10^{-5}.

8.1.2. Unsteady viscous flow (*Re* = 200)

The background mesh consists of 27,489 nodes and 54,595 elements and it is partitioned into 4 sub-domains for parallel computation. The overlapping grid consists of 29,580 nodes and 58,558 elements and it is also partitioned into 4 sub-domains, as shown in Fig. 13. In the background mesh, the wake region is further refined in order to accurately capture the fine details of the vortex shedding phenomenon as



Background mesh
(16 sub-domains)
(20,869 nodes and 41,355 elements)

Overlapping grid
(4 sub-domains)
(4,380 nodes and 8,494 elements)

Fig. 9. Partitioned background and overlapping grids for stationary circular cylinder flow at *Re* = 41.

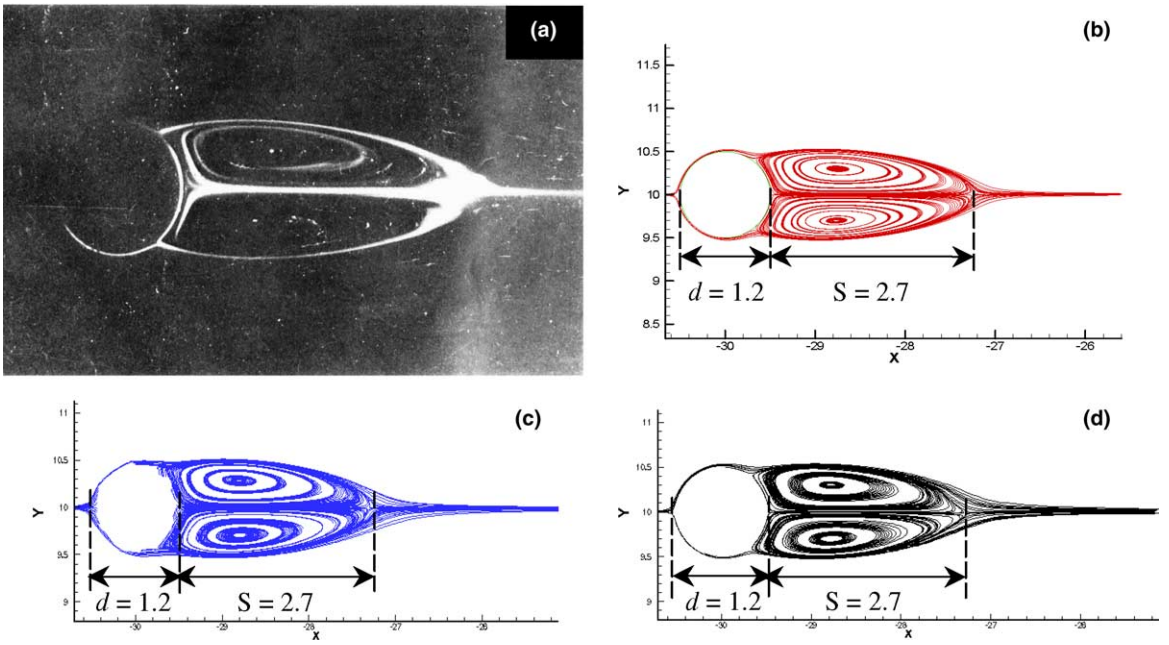


Fig. 10. Streamlines plot for flow over a stationary circular cylinder for (a) experimental measurements [17], (b) Tai and Zhao [14], (c) background mesh and (d) overlapping grids.

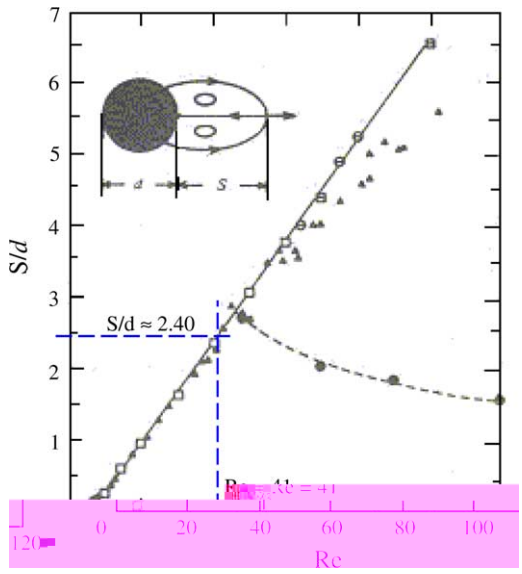


Fig. 11. Length of seperation bubbles behind cylinder vs. Reynolds number [18].

shown in Fig. 13. The number of pseudo sub-iterations per time step is set to 200, CFL is set to 0.5 and the non-dimensional physical time step is set to 0.09 for better temporal resolution. The flow is started from stationary conditions and the simulation is run until periodic shedding of vortices occurred.

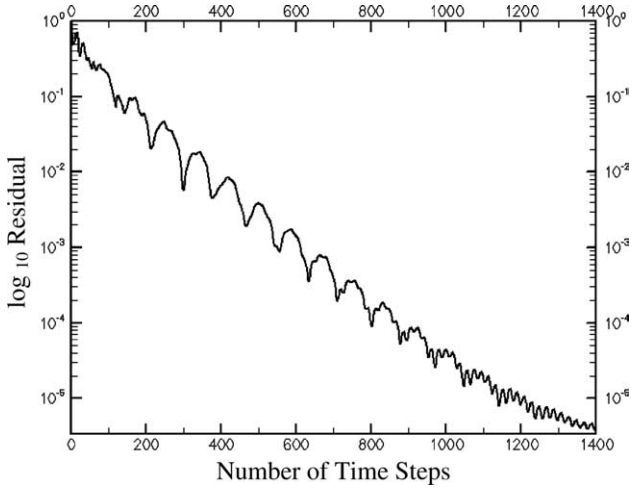


Fig. 12. Convergence history plot for flow over cylinder using immersed object method.

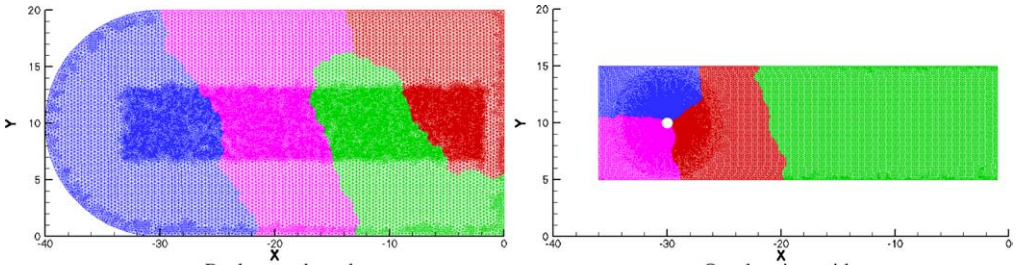


Fig. 13. Partitioned background and overlapping grids for circular cylinder flow at $Re = 200$.

Fig. 14 presents the computed lift and drag coefficients on the cylinder versus non-dimensional time. A pronounced asymmetric wake begins to appear at non-dimensional time of 20 and the flow becomes completely periodic at a time instant of 55. The lift coefficient, C_l, drag coefficient, C_d, and Strouhal number, St, obtained using the immersed object method are ± 0.69, 1.33 ± 0.05 and 0.198, respectively, and they agree well with other numerical and experimental results [22–24], all of which are presented in Table 2. Fig. 15 shows the contours of vorticity obtained by the immersed object method on both the background and overlapping grids. The figure shows that the vortices with opposite signs are shed from upper and lower surfaces alternately, thus forming the Kármán vortex street.

8.2. Rotating square cylinder

The second test case considered is the viscous flow over a rotating square cylinder of unity at $Re = 41$. The number of pseudo sub-iterations per time step is set to 200, CFL is set to 0.4 and the non-dimensional physical time step is set to 0.02. An overlapping mesh of square shape is immersed into the background mesh to define the physical boundary of the cylinder. The angle of rotation per time step for the cylinder

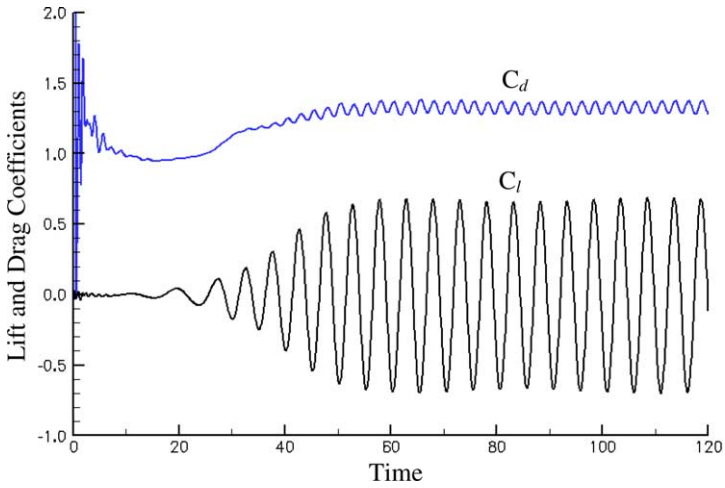


Fig. 14. Lift and drag coefficients versus time for flow over a stationary circular cylinder computed using the overlapping grid ($Re = 200$).

Table 2
Lift coefficient, drag coefficient and Strouhal number for flow over a stationary cylinder ($Re = 200$)

Reference	C_l	C_d	St
Present (immersed object method)	±0.69	1.33 ± 0.05	0.198
Liu et al. [19]	±0.69	1.31 ± 0.049	0.192
Roshko (Expt.) [20]	–	–	0.19
Wille (Expt.) [21]	–	1.30	–

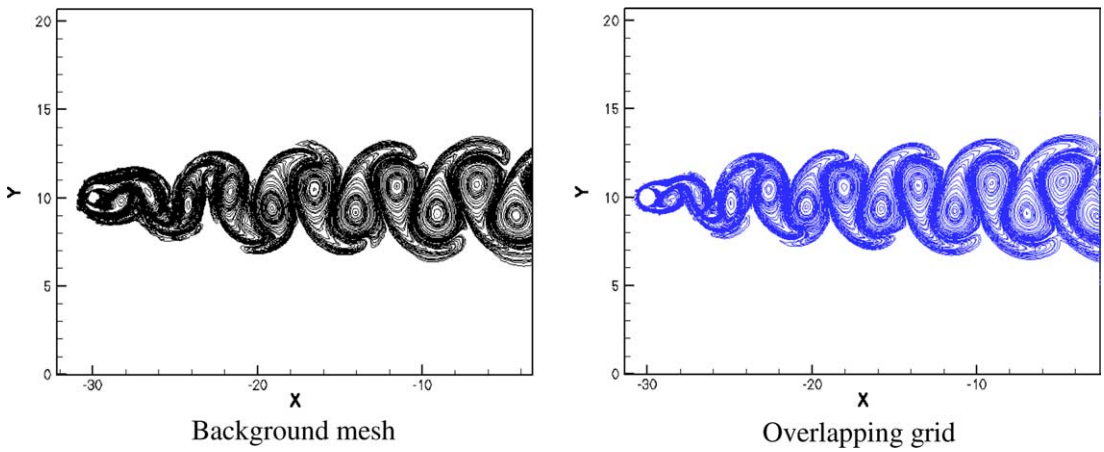


Fig. 15. Vorticity contours plot for background and overlapping grids ($Re = 200$).

is set to 0.2° or 3.5×10^{-3} radians. The square cylinder rotates counter-clockwise at an angular velocity of 0.175 rad/s for each time step. The flow field is started from rest with the square cylinder rotates at the set angular velocity and the simulation is run until periodic shedding of vortices occurred. The background

mesh is the same mesh used in the first test case for unsteady viscous flow over a stationary circular cylinder. And in this computation the background mesh is partitioned into 8 sub-domains for parallel computation, whist the overlapping grid consists of 15,129 nodes and 29,658 elements and is partitioned into 4 sub-domains, as shown in Fig. 16.

The computed lift and drag coefficients on the rotating square cylinder versus non-dimensional time are shown in Fig. 17. The period for the periodic motion is 36 and thus it takes a non-dimensional time of 36 for the square cylinder to complete one cycle. From Fig. 17, it can be seen that there are two peaks and two troughs within a cycle, corresponding to the four corners of the square. Fig. 18 shows the vorticity contour plots for the rotating square cylinder at different non-dimensional physical time and angle of rotation, which outlines the vortex shedding phenomenon. These phenomena are different from the Kármán vortex street phenomenon, because periodic shedding of vortices occurs at very low Reynolds numbers.

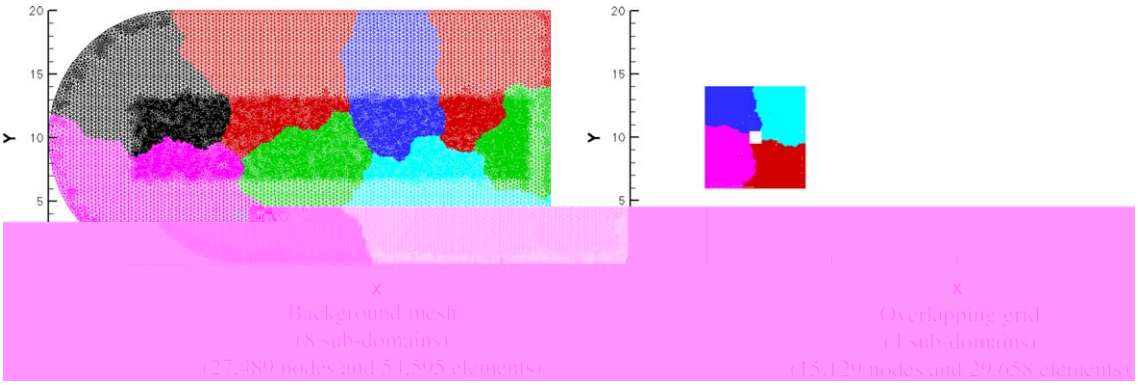


Fig. 16. Partitioned background and overlapping grids for rotating square cylinder flow.

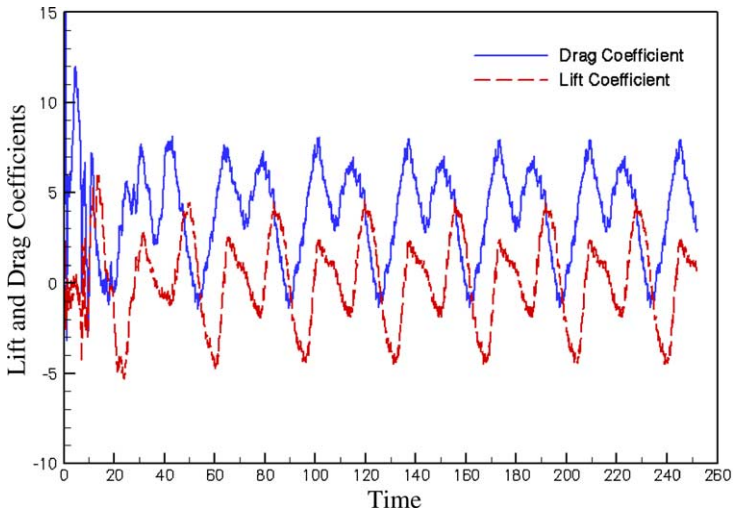


Fig. 17. Lift and drag coefficients versus time for flow over a rotating square cylinder computed using the overlapping grid.

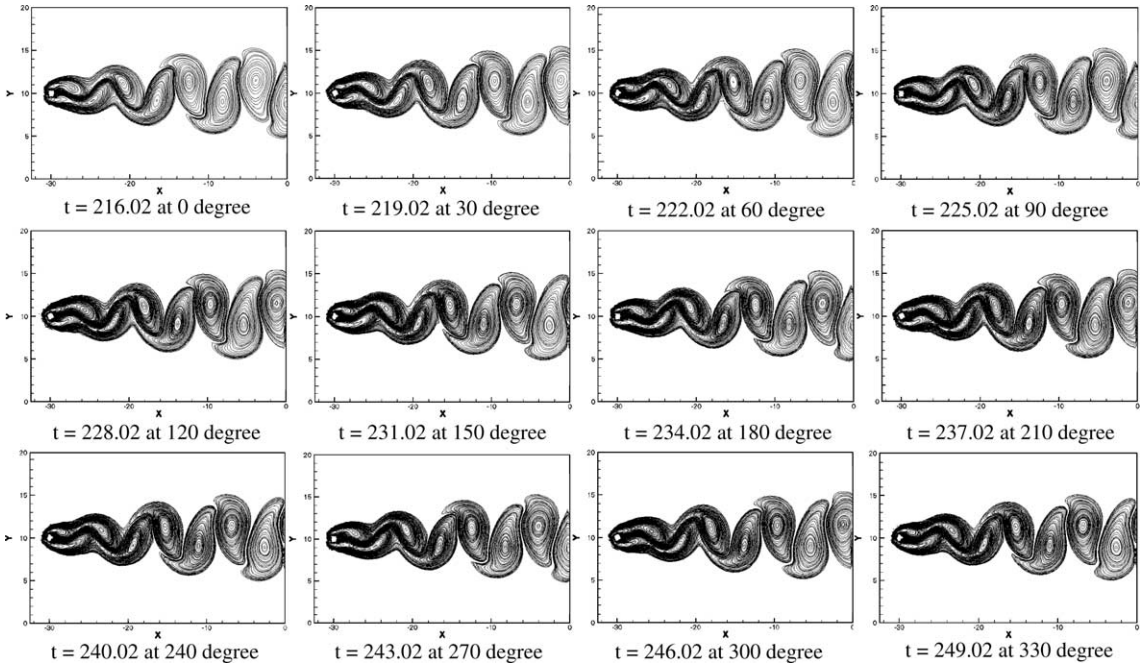


Fig. 18. Vorticity plots for rotating square cylinder at different time and angle of rotation.

8.3. Rotating and translating circular disk

The third test case considered is viscous flow over a rotating and translating circular disk of diameter $d = 0.25$ at $Re = 25$, based on kinematics viscosity $v = 1 \times 0^{-2}$. The disk moves within a cavity of $x = -0.35$ to 0.9 and $y = -0.5$ to 0.5 and the centre of the disk translates along a prescribed trajectory as follows,

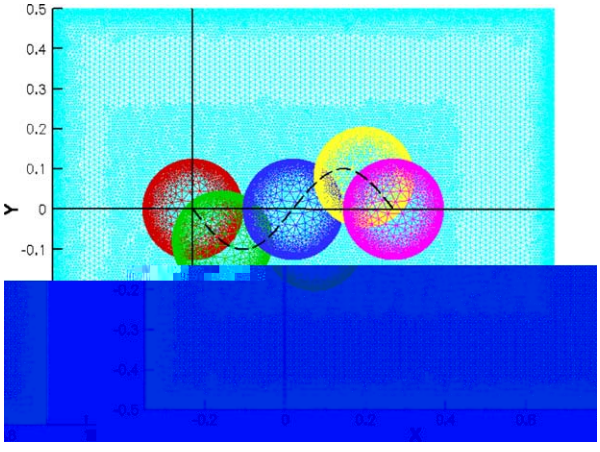


Fig. 19. Moving disk along a prescribed trajectory.

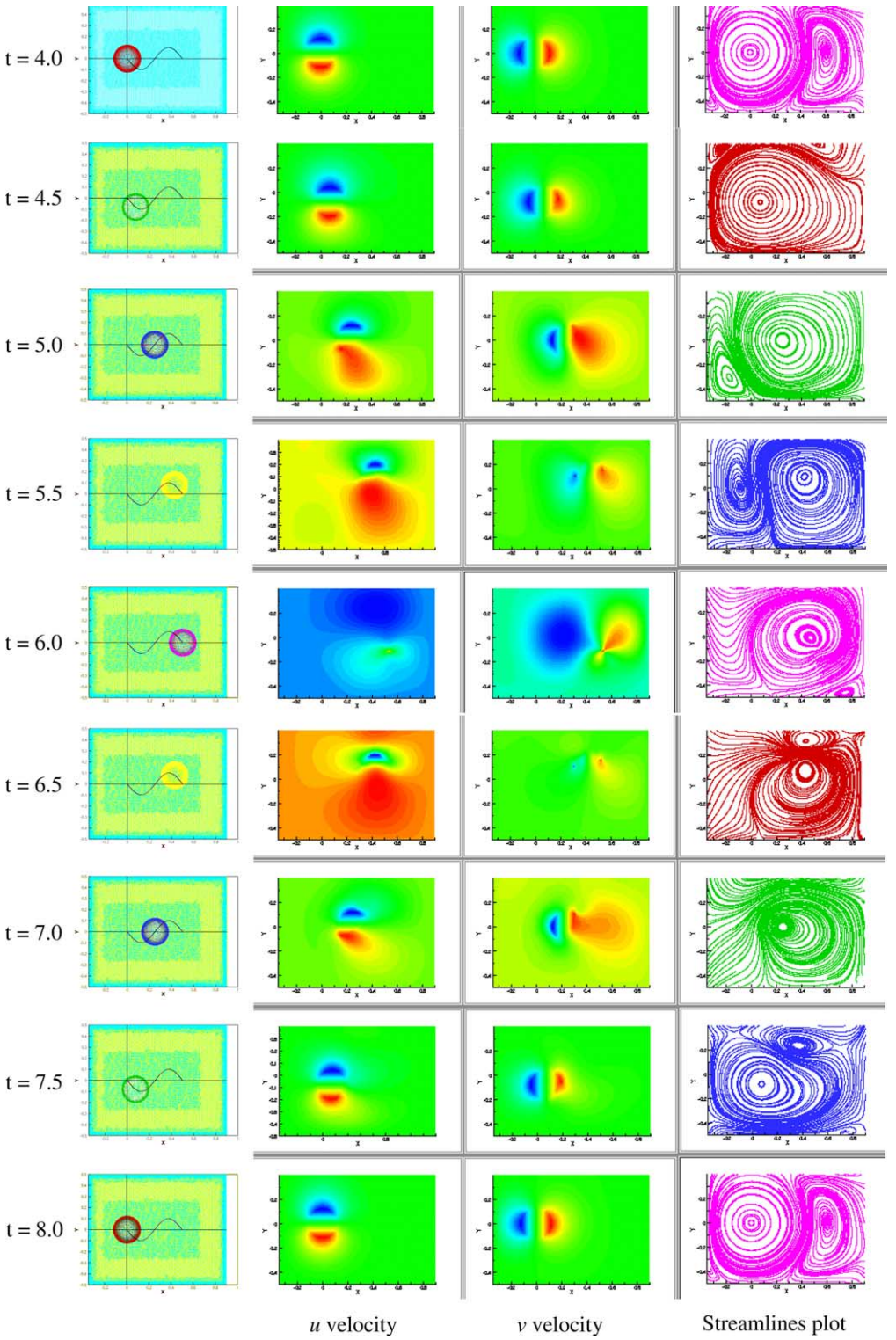


Fig. 20. Streamlines and velocities contour plots for the one period of the moving disk.

$$x(t) = 0.25\left(1 - \cos\left(\frac{t}{2}\right)\right), \quad y(t) = -0.1\sin\left(\pi\left(1 - \cos\left(\frac{t}{2}\right)\right)\right).$$

The computational domain, prescribed trajectory and several different positions of the disk are shown in Fig. 19. The four sides of the computational domain are solid wall with zero velocity. While its centre is moving along the trajectory, the disk is rotating counter-clockwise at an angular velocity of and the fluid velocity within the disk is set to an object velocity, $\mathbf{V}_o$, of π. The mesh shown in Fig. 19 consists of 27,323 nodes and 54,284 elements and is partitioned into 8 sub-domains for parallel computation. The number of pseudo sub-iterations per time step is set to 300, CFL is set to 1.5 and the non-dimensional physical time step is set to 0.01.

The flow is started with the fluid at rest and the disk rotates at an angular velocity of π and the simulation is run until the flow becomes periodic. The period of the periodic motion is 4. Fig. 20 shows the contour plots for both u and v velocities and streamline plots at different time instances between $t = 4.0$ and 8.0. From the figure, it can be seen that the rotating disk causes the flow field within the confined domain to rotate in counter-clockwise direction. At the same position of the rotating disk, the flow phenomenon is completely different at different time instances. For example, flow patterns at $t = 5.5$ and 6.5 are different from each other. This is mainly due to the rotating disk at $t = 5.5$ is translating in the forward direction, whereas at $t = 6.5$, it is translating back to the initial position at (0,0).

9. Conclusions

In this paper, a novel immersed object method has been developed and implemented in a two-dimensional parallel unstructured-mesh finite volume Navier–Stokes solver for the study of unsteady incompressible viscous flows around moving rigid objects. The advantage of this method lies in the fact that it does not require object–mesh intersection calculation and interpolation/extrapolation to impose boundary condition directly onto the object surfaces. The inclusion of overlapping grids in the immersed moving object method improves the accuracy and robustness of the method for high-Reynolds-number flows. Numerical results obtained with the immersed moving object method are found to agree well with published numerical solutions and experimental measurements. It is found that the method is capable of calculating complex flows with arbitrarily moving objects efficiently and accurately.

Acknowledgement

This research work was supported by a research scholarship provided by Nanyang Technological University (NTU) and Sun Microsystems Pte. Ltd. The provision of computing facilities by Nanyang Centre for Supercomputing and Visualisation (NCSV), NTU is acknowledged.

References

[1] C.S. Peskin, Flow patterns around heart valves: a numerical method, Journal of Computational Physics 10 (1972) 252–271.
[2] A.L.F. Lima, E. Silva, A. Silveira-Neto, J.J.R. Damasceno, Numerical simulation of two-dimensional flows over a circular cylinder using the immersed boundary method, Journal of Computational Physics 189 (2003) 351–370.
[3] J.M. Stockie, S.I. Green, Simulating the motion of flexible pulp fibres using the immersed boundary method, Journal of Computational Physics 147 (1998) 147–165.
[4] K.M. Arthurs, L.C. Moore, C.S. Peskin, E.B. Pitman, H.E. Layton, Modeling arteriolar flow and mass transport using the immersed boundary method, Journal of Computational Physics 147 (1998) 402–440.

[5] D. Goldstein, R. Handler, L. Sirovich, Modeling a no-slip flow boundary with an external force field, Journal of Computational Physics 105 (1993) 354–366.
[6] D. Goldstein, R. Handler, L. Sirovich, Direct numerical simulation of turbulent flow over a modelled riblet covered surface, Journal of Fluid Mechanics 302 (1995) 333–376.
[7] E.M. Saiki, S. Biringen, Spatial numerical simulation of boundary layer transition: effects of a spherical particle, Journal of Fluid Mechanics 345 (1997) 133–164.
[8] Y.H. Tseng, J.H. Ferziger, A ghost-cell immersed boundary method for flow in complex geometry, Journal of Computational Physics 192 (2003) 593–623.
[9] R.P. Fedkiw, T. Aslam, B. Merriman, S. Osher, A Non-oscillatory Eulerian approach to interfaces in multimaterial flows (the Ghost Fluid Method), Journal of Computational Physics 152 (1999) 457–492.
[10] R.P. Fedkiw, Coupling an Eulerian fluid calculation to a Lagrangian solid calculation with the Ghost Fluid method, Journal of Computational Physics 175 (2002) 200–224.
[11] R. Glowinski, T.W. Pan, J. Periaux, A fictitious domain method for dirichlet problems and applications, Computer Methods in Applied Mechanics and Engineering 111 (1994) 283–303.
[12] R. Glowinski, T.W. Pan, J. Periaux, A fictitious domain method for external incompressible viscous flow modeled by Navier–Stokes equations, Computer Methods in Applied Mechanics and Engineering 112 (1994) 133–148.
[13] C. Kiris, D. Kwak, S. Rogers, I.-D. Chang, Computational approach for probing the flow through artificial heart devices, Journal of Biomechanical Engineering 119 (1997) 452–460.
[14] J. Mohd-Yusof, Combined immersed-boundary/B-spline methods for simulations of flow in complex geometries, CTR Annual Research Briefs, NASA Ames/Standford University, 1997.
[15] Y. Zhao, C.H. Tai, Higher-order characteristics based methods for incompressible flow computation on unstructured grids, AIAA Journal 39 (7) (2001) 1280–1287.
[16] H. Lou, J.D. Baum, R. Löhner, An accurate fast, matrix-free implicit method for computing unsteady flows on unstructured grids, Computer & Fluids 30 (2001) 137–159.
[17] C.H. Tai, Y. Zhao, Parallel unsteady incompressible viscous flow computations using an unstructured multigrid method, Journal of Computational Physics 192 (1) (2003) 277–311.
[18] W. Gropp, E. Lusk, A. Skjellum, Using MPI: Portable Parallel Programming with the Message-passing Interface, The MIT Press, Cambridge, Massachusetts, 1994.
[19] G. Karypis, V. Kumar, Metis: A software package for partitioning unstructured graphs, partitioning meshes, computing fill-reducing orderings of sparse matrices, version 4.0, University of Minnesota, Department of Computer Science, 1998.
[20] V. Dyke, D. Milton, An Album of Fluid Motion, Parabolic Press, Stanford, CA, USA, 1982.
[21] M. Nishioka, H. Sato, Mechanism of determination of the shedding frequency of vortices behind a cylinder at low Reynolds numbers, Journal of Fluid Mechanics 89 (1978) 49–60.
[22] C. Liu, X. Zheng, C.H. Sung, Preconditioned multigrid methods for unsteady incompressible flows, Journal of Computational Physics 139 (1998) 35–57.
[23] A. Roshko, On the development of turbulent wakes from vortex streets, NACA Report, 1191, 1954.
[24] R. Wille, Karman vortex streets, Advances in Applied Mechanics, vol. 6, Academic, New York, 1960, pp. 273–287.