Available online at www.sciencedirect.com
SCIENCE DIRECT
Journal of Computational Physics 215 (2006) 661–690
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

An efficient parallel/unstructured-multigrid preconditioned implicit method for simulating 3D unsteady compressible flows with moving objects

X. Lv [a], Y. Zhao [a,*], X.Y. Huang [a], G.H. Xia [a], Z.J. Wang [b]

[a] School of Mechanical and Aerospace Engineering, College of Engineering, Nanyang Technological University, Singapore 639798, Singapore
[b] Department of Aerospace Engineering, College of Engineering, Iowa State University, Ames, IA 50011, USA

Received 13 July 2005; received in revised form 7 November 2005; accepted 8 November 2005
Available online 27 December 2005

Abstract

This paper presents the development and validation of a new parallel unstructured multi-grid preconditioned implicit method for the simulation of three-dimensional (3D) unsteady compressible flows using a so-called immersed membrane method (IMM) [G.H. Xia, Y. Zhao, J.H. Yeo, An immersed membrane method for simulation of fluid–structure interaction in bio-fluid flows, in: 1st International BioEngineering Conference (IBEC 2004), 08–10 September 2004]. The novel feature of the method presented is that it employs a unique combination of a parallel multi-grid scheme with low-Mach-number preconditioning and the IMM so that 3D unsteady low-Mach-number flows with arbitrarily moving objects can be efficiently simulated.

Keywords: Fluid–structure interaction; Parallel; Multigrid; Preconditioning; Immersed membrane method; Finite volume scheme; Implicit dual time stepping scheme; Higher-order TVD scheme

1. Introduction

The simulation of fluid flows 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. In recent years significant research efforts have been devoted to the development of numerical models for studying moving boundary problems based on the finite volume and finite element methods. Luo and Pedley [1–3] performed a time-dependent simulation of a coupled flow-membrane problem, using the Arbitrary Lagrangian Eulerian (ALE) method together with a spine scheme to treat a compliant wall moving in its wall normal direction in a channel. Zhao et al. [4,5] have also proposed a new dynamic mesh scheme to simulate an arbitrarily moving elastic wall in a similar channel based

* Corresponding author. Tel.: +65 6790 4545; fax: +792 4062.
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.2005.11.012

on the ALE. Gaitonde [6,7] developed a moving mesh method for the computation of compressible viscous flow past moving aerofoils. A sequence of body conforming grids and the corresponding grid speeds were required, where the inner and outer boundaries of the grids moved independently. The required grids and their velocities were found by using a transfinite interpolation technique. Mendes and Branco [8] analyzed the interaction of fluid-rigid body by a finite element procedure that incorporates the ALE method into a two-step projection scheme and the flow was assumed to be 2D, incompressible and viscous. Anju et al. [9] presented a finite element analysis of a interaction problem by the ALE method and a fractional step Navier–Stokes solver. The method was applied to analyze flow around an oscillating rectangular cylinder. It is interesting to note that in this study the whole calculation domain was divided into several sub-domains to improve computational efficiency. In the inner domain the mesh was moved and the calculation was based on the ALE method. The outer-most domain was an Eulerian domain with fixed grid. The intermediate domain was designated as a transition region. Heil [10] studied a three-dimensional steady Stokes flow in an elastic tube using the ALE, which was described by non-linear shell equations. Only the final equilibrium state was presented because the flow was steady. Lefrancois et al. [11] developed a finite element model for studying fluid–structure interaction and an ALE formulation was used to model the compressible inviscid flow with moving boundaries with large deformation. It is observed that all the works reported have mostly relied on the costly grid regeneration method to capture large movements of the boundary in the flow field, whist the less costly dynamic grid method is believed to be unable to handle large boundary and mesh deformations.

An alternative to the ALE is the Eulerian method, where the computational mesh is fixed without deformation or movement, thus fluid motion can be conveniently described with respect to this Eulerian frame. The group of Eulerian methods includes Immersed Boundary (IB) method and Fictitious Domain (FD) method etc. Peskin et al. [12] proposed the IB method to simulate the motion of human hearts and heart valves. At its early stage, the IB method could not consider the inertial effect of the structure because the dynamic equation of the structure was not used to calculate its movement. In their recent work [13], Zhu and Peskin did consider the inertial effect by taking the structure's (a soap film) mass into account. They accomplished this by employing an inhomogeneous density field in the unified momentum equation with different densities for the fluid domain and the structure positions. The philosophy of the IB method is to treat immersed elastic structures as parts of the fluid domain on which additional forces, arising from elastic stresses of the immersed structures, are applied. The governing equations for the fluid domain are solved on a Cartesian mesh, which is not modified by the presence of the immersed elastic structure. The immersed boundary is tracked in a Lagrangian manner, by following a collection of representative material points. The spatial configuration of these points is used to compute the elastic forces, which are then passed to nearby mesh points of the fluid domain through a delta function. Fluid velocity is updated under the influence of these forces, and the latest updated velocity is used to calculate the movement of the structure. The IB method has been applied to a wide range of problems, mostly in biofluid dynamics, including blood flow in the human heart [14], platelet aggregation during blood clotting [15,16] and the motion of flexible pulp fibers [17]. Glowinski et al. [18,19] studied a Lagrange multiplier based Fictitious Domain (FD) method for numerical solutions of three-dimensional elliptic problems with Dirichlet boundary conditions and also for the Navier–Stokes equations modeling incompressible viscous flows. This method is based on the imposition of velocity constraints associated with moving internal boundaries by means of a Lagrange multiplier. The FD method allows the coupling of domains with dissimilar element distributions and/or interpolation order by applying Lagrange multipliers on a fictitious boundary representing the immersed structure [20,21]. The main advantage of the FD method is that different mathematical descriptions for the fluid and structure can be maintained, allowing convenient classical formulations (Lagrangian or Eulerian formulations) for each of these subsystems. Moreover, the fluid mesh is not altered or interrupted by the presence of the immersed domain, therefore preserving its original quality [22–26]. Generally speaking, Eulerian methods are relatively less-complicated techniques by using fixed fluid meshes, which reduce the computational costs for mesh treatment. However, the above Eulerian methods do not allow for 'jump' conditions between two sides of immersed thin structures, because the flow conditions on structural boundaries are smoothed over several mesh cells across or near the immersed structures due to the fact that the structures are considered as internal conditions in the flow field and source terms are distributed to nearby fluid nodes for constraining the flow field. Recently Zhao et al. [27] has also developed a so-called Immersed Object Method (IOM) with overlapping unstructured grids for general Fluid–Structure-Interaction (FSI) simulation. The main idea of the method is

that the fluid covered by immersed objects is assumed to be frozen and moves like a solid, whose kinematics is enforced by adding source terms to the momentum equations. Overlapping grids are wrapped around the objects and the boundary conditions for the overlapping grids are transferred from the Eulerian grid to the overlapping grids for further computation on the overlapping grids, in order to capture the fine details of boundary lays over the moving objects. This method is found to be efficient to simulate moving objects, but it cannot handle thin structures, such as membranes.

Liu et al. [28] have developed a multigrid method using the dual time stepping scheme on structured grids for the computation of unsteady incompressible viscous flows, while Lin [29] has attempted the unstructured multigrid (MG) method for calculating unsteady inviscid flows. In Tai and Zhao's study [30], they successfully extended the higher-order characteristics-based finite-volume scheme for unstructured grids [31] to simulate unsteady incompressible viscous flows by introducing an unstructured multigrid method, which we will extend for unsteady compressible flow simulation in our current study.

One difficulty with compressible Navier–Stokes solvers is slow convergence rates and even unstable solutions for low Mach number flows. This difficulty can be traced to a disparity between the acoustic and convective speeds [32–41], and can be addressed by a preconditioning algorithm. Previous work in this area has been reported by Venkateswaran and Merkle [35,38], Turkel [39], Van Leer et al. [40], Weiss and Smith [41]. The applications of the preconditioning methods have been found in the computation of steady flows without considering arbitrarily moving objects in the flow field.

Tai and Zhao [42] parallelized an incompressible Navier–Stokes solver based on the artificial compressibility approach and higher-order characteristics-based finite volume scheme for unstructured non-moving grids using a multigrid domain decomposition approach (MG-DD) and the single program multiple data (SPMD) parallel strategy with message passing interface (MPI). They developed a communication scheme for an overlapping partition structure in order to obtain continuity of results in the whole domain. Singh and Zhao [43] developed a parallel dynamic unstructured moving grid Direct Monte Carlo Simulation (DMCS) of molecular gas dynamics and the associated thin film deposition. It is envisaged that the combination of the ALE approach and the parallel unstructured MG, as well as the preconditioning, may not be efficient enough and easy to be implemented into computer codes, compared with the combination with an Eulerian method for handling moving objects and the resulting unsteady 3D flows, not to mention the difficulty for the ALE to simulate large mesh deformations.

Therefore in this study, we aim to develop a parallel unstructured multi-grid preconditioned compressible Navier–Stokes solver implementing the IMM [46] to calculate 3D unsteady low-Mach-number flows with arbitrarily moving objects. The developed solver makes it possible to include moving objects in the flow fields with complexities that existing methods cannot easily handle because it does not requires complicated interpolation of boundary conditions along surface normal direction. Unlike the aforementioned Eulerian methods, the new method does not smooth fluid forces across the immersed structure and discontinuities in pressure and gradients of flow properties can be accurately accounted for as a result.

The paper is structured as follows. In Section 2 which is the next section, the finite-volume preconditioned compressible flow solver on unstructured grids is described. In Section 3, a brief description of our parallel and multigrid implementation is presented. This is followed by a detailed description of the IMM algorithm in Section 4. In Section 5, a grid convergence study is carried out for flows induced by both a steadily rotating rigid sphere and an oscillating sphere to establish the order of accuracy of the method and demonstrate the capability of the method for handling moving objects. The method is then validated by applying it to two test cases. The first one is viscous flow past a circular cylinder. The second case is viscous flow past an immersed fixed membrane in a 3D square channel. Finally, several concluding remarks are drawn in Section 6.

2. Governing equations and numerical methods

2.1. Governing equations

The Navier–Stokes equations for three-dimensional compressible unsteady flows can be given in vector form explicitly expressing the conservation laws of mass, momentum and energy. We also introduce, in the equations, pseudo-time terms to provide pseudo-time marching for their numerical solutions:

$$\Gamma_1 \frac{\partial W_p}{\partial \tau} + \frac{\partial W_c}{\partial t} + \nabla \cdot \vec{F}_i = \nabla \cdot \vec{F}_v \tag{2.1}$$

where

$$W_c = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e_t \end{bmatrix} \tag{2.2}$$

$$W_p = \begin{bmatrix} p \\ u \\ v \\ w \\ T' \end{bmatrix} \tag{2.3}$$

$$\vec{F}_i = \begin{bmatrix} \rho\vec{U} \\ \rho u\vec{U} + p\vec{i} \\ \rho v\vec{U} + p\vec{j} \\ \rho w\vec{U} + p\vec{k} \\ \rho H\vec{U} \end{bmatrix} \tag{2.4}$$

$$\vec{F}_v = \begin{bmatrix} 0 \\ \vec{\tau}_x \\ \vec{\tau}_y \\ \vec{\tau}_z \\ (\bar{\bar{\tau}} \cdot \vec{U} - \vec{q}) \end{bmatrix} \tag{2.5}$$

τ is the pseudo-time and Γ_1 is the preconditioning matrix in the pseudo-time terms for low-Mach-number flows which is defined in the appendix. W_c and W_p are the vectors of conservative and primitive dependent variables, respectively; $\vec{F}_i$ and $\vec{F}_v$ are the inviscid convective flux and viscous flux vectors. Furthermore we have the following formulas:

$$\vec{U} = u\vec{i} + v\vec{j} + w\vec{k}$$
$$\bar{\bar{\tau}} = \vec{\tau}_x\vec{i} + \vec{\tau}_y\vec{j} + \vec{\tau}_z\vec{k}$$
$$\vec{\tau}_i = \tau_{ix}\vec{i} + \tau_{iy}\vec{j} + \tau_{iz}\vec{k}$$
$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\delta_{ij}\nabla \cdot \vec{U}\right)$$
$$\vec{q} = q_x\vec{i} + q_y\vec{j} + q_z\vec{k}$$
$$T' = p/\rho = c^2/\gamma$$

$\vec{i}$, $\vec{j}$ and $\vec{k}$ are the three unit vectors in three Cartesian directions, τ_{ix},τ_{iy} and τ_{iz} the viscous stresses and q the heat transfer flux vector, defined by

$$\vec{q} = -\kappa\nabla T = -\frac{\mu C_p}{Pr}\nabla T$$

here T the temperature and Pr the Prandtl number:

$$Pr = \frac{\mu C_p}{\kappa}$$

The above equations are non-dimensionalised and the non-dimensional variables used are defined as follows:

$$(x^*, y^*, z^*) = \left(\frac{x}{L}, \frac{y}{L}, \frac{z}{L}\right); \quad (u^*, v^*, w^*) = \left(\frac{u}{U_\infty}, \frac{v}{U_\infty}, \frac{w}{U_\infty}\right)$$

$$t^* = \frac{t}{L/U_\infty}; \quad p^* = \frac{p - p_{\text{in}}}{\rho_\infty (U_\infty)^2}; \quad e_t^* = \frac{e_t}{(U_\infty)^2}; \quad H^* = \frac{H}{(U_\infty)^2}$$

$$\rho^* = \frac{\rho}{\rho_\infty}; \quad T^* = \frac{T}{(U_\infty)^2 / C_v}; \quad Re = \frac{\rho_\infty U_\infty L}{\mu}$$

$$q^* = -\frac{\gamma}{Pr\, Re} \nabla^* T^*; \quad \gamma = \frac{C_p}{C_v}; \quad T^* = \frac{p^*}{\rho^* (\gamma - 1)} = T'/(\gamma - 1)$$

where p_{in} is the inlet or reference pressure. L is the characteristic length of the computed model, U_∞ is the inflow velocity and μ is the dynamic viscosity of the flow. The variables with a superscript $*$ here are non-dimensional parameters and the asterisk sign will be dropped in subsequent equations for sake of convenience. It should be noted that after non-dimensionalization, we will use T' in W_p and both T^* and T' in other calculations. We subtract a constant value (the reference pressure) from the pressure term to control the round-off errors for low speed flows, which is found to be critical in controlling computational errors in the momentum equations for low-speed compressible flows. The use of gauge pressure is a common treatment for incompressible solvers because only pressure gradients are needed for all calculations. For compressible solvers, the absolute value of pressure must be used when dealing with the energy equation and state equation of gas. But when we use non-dimensional absolute pressure at low Mach numbers, it becomes extremely large although the pressure gradients in momentum equations are small. The use of gauge pressure can avoid performing the addition and subtraction operations between two extraordinarily large values of non-dimensional absolute pressures. In our experience, the result obtained with gauge pressure is more accurate than without using it. And the relative error can be up to 4–5% of the result.

2.2. Numerical methods

The 3D equations (2.1) are transformed into an integral form and discretized on an unstructured grid. A cell-vertex finite volume scheme is adopted here. For every vertex, as shown in Fig. 1, a control volume is constructed using the median duals of the tetrahedral cells. Spatial discretization is performed by using the integral form of the conservation equations over the control volume surrounding node P:

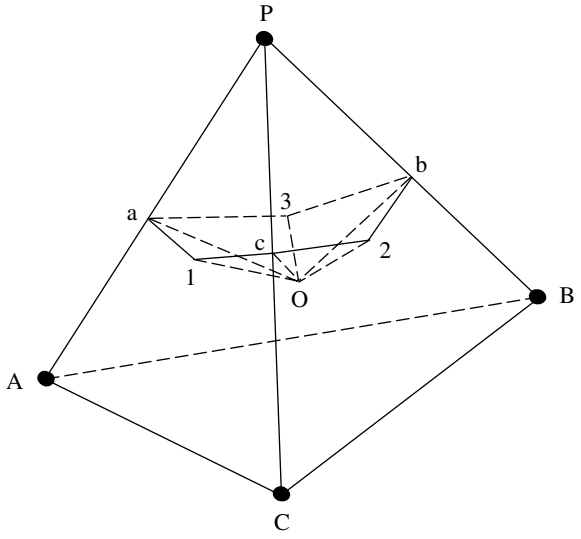


Fig. 1. Construction of control volume within a tetrahedron for a node P.

$$\int\int\int_{cv}\frac{\partial Q_1'}{\partial\tau}\,dV+\int\int\int_{cv}\frac{\partial W_c}{\partial t}\,dV+\int\int\int_{cv}\nabla\cdot\vec{F}_i\,dV-\int\int\int_{cv}\nabla\cdot\vec{F}_v\,dV=0 \tag{2.6}$$

Noted that a new variable Q_1' has arisen as $\frac{\partial Q_1'}{\partial\tau}=\Gamma_1\frac{\partial W_p}{\partial\tau}$, and the Jacobian $\Gamma_1=\frac{\partial Q_1'}{\partial W_p}$. So that, $\frac{\partial Q_1'}{\partial\tau}=\frac{\partial Q_1'}{\partial W_p}\frac{\partial W_p}{\partial\tau}=\Gamma_1\frac{\partial W_p}{\partial\tau}$.

The convective term is transformed into a summation:

$$\int\int\int_{cv}\nabla\cdot\vec{F}_i\,dV=\int_{S_{cv}}\vec{F}_i\cdot\vec{n}\,dS=\sum_{k=1}^{nbseg}[(\vec{F}_i)_{ij}\cdot\vec{n}\Delta S]_k \tag{2.7}$$

where 'nbseg' is the number of the edges associated with node P, $(\vec{F}_i)_{ij}^k$ is the inviscid flux through the part of control volume surface associated with edge k, and $\vec{n}$ is the unit normal vector of the control volume surface. Finally, ΔS_k is a part of the control volume surface associated with edge k. Therefore, all the fluxes are calculated for the edges and then collected at the two end of each edge for updating of flow variables in time marching. The viscous term is calculated using a cell-based method:

$$\int\int\int_{cv}\nabla\cdot\vec{F}_v\,dV=\int_{S_{cv}}\vec{F}_v\cdot\vec{n}\,dS=\sum_{i=1}^{ncell}[\vec{F}_v\cdot\vec{n}\Delta S_c]_i \tag{2.8}$$

where 'ncell' is the number of elements associated with node P and ΔS_{ci} is the part of control volume surface in cell i. By using the following relation:

$$\int_{S_{cv}}d\vec{S}=0$$

the total vector surface of the control volume in a cell i becomes

$$\vec{n}\Delta S_{ci}=\frac{1}{3}(\vec{n}\Delta S_{pi})$$

Thus, the calculation of viscous terms can be simplified as

$$\sum_{i=1}^{ncell}[\vec{F}_v\cdot\vec{n}\Delta S_c]_i=\frac{1}{3}\sum_{i=1}^{ncell}[\vec{F}_v\cdot\vec{n}\Delta S_p]_i \tag{2.9}$$

where $\vec{n}\Delta S_{pi}$ is the surface vector of the face opposite node P of the tetrahedron under consideration. Here the $(\vec{F}_v)_i$ is calculated at the center of the tetrahedron with a node P, and can be obtained by using Green's Theorem based on the variables at the four vertices of the tetrahedron. Similar to the Galerkin type of formulation, the gradient of a flow variable ϕ at the center of a tetrahedron is evaluated as follows:

$$\text{grad}\,\phi_c=-\frac{\sum_{i=1}^4\phi_i 9S_i}{27V}=-\frac{1}{3}\frac{\sum_{i=1}^4\phi_i S_i}{V} \tag{2.10}$$

where ϕ_i is the flow variable at a vertex i of the tetrahedron and S_i is the surface area that is opposite to node i, V is the volume of the tetrahedron. Gradients at the vertices are obtained by a volume averaging of the gradients at the center of cells associated with the vertex under consideration.

In this work, a high-order Roe's TVD scheme for compressible flow for arbitrary unstructured 3D grids has been adopted. Because of the preconditioning matrix Γ_1, the inviscid fluxes, $(\vec{F}_i)_{ij}^k$, through the face k is now reformulated as

$$\begin{aligned}(\vec{F}_i)_{ij}^k&\equiv\frac{1}{2}((\vec{F}_i)_i+(\vec{F}_i)_j)_k-\frac{1}{2}\left|\frac{\partial\vec{F}_i}{\partial Q_1'}\right|_k(\delta Q_1')_k\\&=\frac{1}{2}((\vec{F}_i)_i+(\vec{F}_i)_j)_k-\frac{1}{2}\left|\frac{\partial\vec{F}_i}{\partial W_p}\frac{\partial W_p}{\partial Q_1'}\right|_k\left(\frac{\partial Q_1'}{\partial W_p}\right)_k((W_p)_j-(W_p)_i)_k\end{aligned} \tag{2.11}$$

Note that we have retained the variable, Q'_1, in computing this flux. Defining the Jacobian in the normal direction as

$$(H_p)_k = \left(\frac{\partial \vec{F}_i}{\partial W_p}\right)_k$$

And using the previously defined Jacobian $\Gamma_1 = \frac{\partial Q'_1}{\partial W_p}$, then the above expression becomes

$$(\vec{F}_i)^k_{ij} \equiv \frac{1}{2}((\vec{F}_i)_i + (\vec{F}_i)_j)_k - \frac{1}{2}\left|H_p\Gamma_1^{-1}\right|_k \Gamma_{1k}((W_p)_j - (W_p)_i)_k$$

Drop the subscript k on the flux vector and the Jacobian with the assumption that the fluxes and Jacobians all correspond to conditions in the normal direction on the given control volume surface. And after some simple algebraic derivations we have

$$(\vec{F}_i)_{ij} \overset{\equiv}{} \frac{1}{2}((\vec{F}_i)_i + (\vec{F}_i)_j) - \frac{1}{2}\Gamma_1\left|\Gamma_1^{-1}H_p\right|((W_p)_j - (W_p)_i) \tag{2.12}$$

Combined with the third-order MUSCL interpolation, it can produce accurate and stable solution on unstructured grids. The left and right state vectors W_L and W_R at a control volume surface are evaluated using a nominally third-order upwind-biased interpolation scheme. If the left and right state vectors are set to W_i and W_j (i and j corresponding to the two end nodes of an edge), it is a first-order upwind scheme, which are shown as follows:

$$W_L = W_i + \frac{1}{4}\left[(1-\kappa)\Delta_i^- + (1+\kappa)\Delta_i^+\right] \tag{2.13.a}$$

$$W_R = W_j - \frac{1}{4}\left[(1-\kappa)\Delta_j^+ + (1+\kappa)\Delta_j^-\right] \tag{2.13.b}$$

where

$$\begin{aligned}
\Delta_i^+ &= \Delta_j^- = W_j - W_i \\
\Delta_i^- &= W_i - W_{i-1} = 2\vec{ij}\cdot\nabla W_i - (W_j - W_i) = 2\vec{ij}\cdot\nabla W_i - \Delta_i^+ \\
\Delta_j^+ &= W_{j+1} - W_j = 2\vec{ij}\cdot\nabla W_j - (W_j - W_i) = 2\vec{ij}\cdot\nabla W_j - \Delta_j^-
\end{aligned}$$

Therefore, substituting the above equations into Eqs. (2.13.a) and (2.13.b), the final equations based on upwind-biased interpolation scheme is shown as follows:

$$W_L = W_i + \frac{1}{2}\left[(1-\kappa)\vec{ij}\cdot\nabla W_i + \kappa\Delta_i^+\right] \tag{2.14.a}$$

$$W_R = W_j - \frac{1}{2}\left[(1-\kappa)\vec{ij}\cdot\nabla W_j + \kappa\Delta_j^-\right] \tag{2.14.b}$$

where κ is set to 1/3, which corresponds to a nominally third-order accuracy. $\vec{ij}$ is the vector representing the edge, which points from node P to its neighbouring node under consideration. The gradients of 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 Eq. (2.6) form a system of coupled ordinary differential equations, which can be reformulated as

$$\begin{aligned}
\frac{\partial \overline{Q'}_1}{\partial\tau}\Delta V_{cv} + \frac{\partial \overline{W}_c}{\partial t}\Delta V_{cv} &= -\left\{\frac{1}{2}\sum_{k=1}^{nbseg}\left[((\vec{F}_i)_i + (\vec{F}_i)_j) - \Gamma_1\left|\Gamma_1^{-1}H_p\right|((W_p)_j - (W_p)_i)\right]_k \cdot \vec{n}\Delta S - \frac{1}{3}\sum_{i=1}^{ncell}\left[\vec{F}_v\cdot\vec{n}\Delta S_P\right]_i\right\} \\
&= -R
\end{aligned} \tag{2.15}$$

where R represents the residual which includes the convective and diffusive fluxes and ΔV_{cv} is the control volume of node P. the over-bar in Eq. (2.15) denotes the cell-averaging value.

An implicit scheme is adopted for Eq. (2.15) and the time-dependent term is discretized using a second-order-accurate backward differencing scheme,

$$\frac{\partial \overline{Q}_1}{\partial \tau}\Delta V_{\mathrm{cv}} = -R^{n+1} - \left(\frac{1.5\Delta V_{\mathrm{cv}}^{n+1}W_c^{n+1} - 2.0\Delta V_{\mathrm{cv}}^{n}W_c^{n} + 0.5\Delta V_{\mathrm{cv}}^{n-1}W_c^{n-1}}{\Delta t}\right) = \widetilde{R}^{n+1} \tag{2.16}$$

where the superscript $(n+1)$ denotes the physical time level $(n+1)\,\Delta t$ and all the variables are evaluated at this time level, $\widetilde{R}(W_p^{n+1})$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

$$\Delta V_{\mathrm{cv}}^{n+1}\Gamma_1\frac{W_p^{n+1,m+1} - W_p^{n+1,m}}{d\tau} = \widetilde{R}^{n+1,m} \tag{2.17}$$

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 W_p with respect to τ becomes zero, and the solution satisfies $\widetilde{R}^{n+1} = 0$. Hence, the original unsteady Navier–Stokes equations are fully recovered. Therefore, instead of solving the equations in 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. (2.17) can be 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, a fully implicit dual time stepping is adopted here.

A Taylor series expansion is performed for the residual in Eq. (2.17) with respect to the pseudo-time for node i,

$$\widetilde{R}_i^{m+1} = \widetilde{R}_i^{m} + \frac{\partial \widetilde{R}_i}{\partial (W_p)_i}\Delta(W_p)_i + \sum_{j=1}^{\mathrm{nbseg}}\frac{\partial \widetilde{R}_i}{\partial (W_p)_j}\Delta(W_p)_j \tag{2.18}$$

where 'nbseg' is the number of edges connected to i, which is also equal to the number of neighboring points connected to point i through the edges. An approximate flux function is introduced here to simplify the implicit time stepping calculation. The total flux (including both convective and viscous fluxes) across a control volume surface associated with a certain edge ij can be approximated as

$$F_{ij} \approx \frac{1}{2}[(\vec{F}_i)_i \cdot \vec{n} + (\vec{F}_i)_j \cdot \vec{n} - |\lambda_{ij}|((W_p)_j - (W_p)_i)]$$

where λ_{ij} is the spectral radius based on the preconditioned system which is associated with edge ij, and is given in the appendix. Then in all of the $R(W)$ terms (Eq. (2.15)), the Taylor series expansions of the fluxes are

$$\frac{\partial R_i}{\partial (W_p)_i} = \sum_{j=1}^{\mathrm{nbseg}}\frac{1}{2}\left[\frac{\partial (\vec{F}_i)_{ij}}{\partial (W_p)_i} + |\lambda_{ij}|\right] \quad \text{and} \quad \frac{\partial R_i}{\partial (W_p)_j} = \sum_{i=1}^{\mathrm{nbseg}}\frac{1}{2}\left[\frac{\partial (\vec{F}_i)_{ij}}{\partial (W_p)_j} - |\lambda_{ij}|\right]$$

And for the physical time-dependent terms, we have

$$(W_c)_i^{m+1} = (W_c)_i^{m} + \frac{\partial W_c}{\partial W_p}\Delta(W_p)_i$$

After combining all the residuals terms at every node in the flow field into a vector and dropping the third term of the right-hand side of Eq. (2.18), we have

$$\begin{aligned}\widetilde{R}_i^{n+1,m+1} &= \widetilde{R}_i^{n+1,m} - \frac{\partial R_i}{\partial (W_p)_i}\Delta(W_p)_i - 1.5\frac{\Delta V_{\mathrm{cv}}^{n+1}}{\Delta t}\frac{\partial W_c}{\partial W_p}\Delta(W_p)_i \\ &= \widetilde{R}_i^{n+1,m} - \sum_{j=1}^{\mathrm{nbseg}}(H_{p,j})\Delta(W_p)_i - 1.5\frac{\Delta V_{\mathrm{cv}}^{n+1}}{\Delta t}M\Delta(W_p)_i\end{aligned}$$

where $H_{p,j} = \frac{1}{2}\left[\frac{\partial (\vec{F}_i)_{ij}}{\partial (W_p)_i} + |\lambda_{ij}|\right], M = \frac{\partial W_c}{\partial W_p}$. And the whole-field equivalent of Eq. (2.16) can then be re-written as

$$\left(I+\frac{1.5\Delta\tau}{\Delta t}\Gamma_1^{-1}M+\sum_{j=1}^{\text{nbseg}}\left(\frac{\Delta\tau}{\Delta V_{\text{cv}}^{n+1}}\Gamma_1^{-1}H_{p,j}\right)\right)\frac{W_p^{n+1,m+1}-W_p^{n+1,m}}{\Delta\tau}\Delta V_{\text{cv}}^{n+1}$$
$$=\Gamma_1^{-1}\left(-\frac{1.5W_c^{n+1,m}\Delta V_{\text{cv}}^{n+1}-2.0W_c^{n}\Delta V_{\text{cv}}^{n}+0.5W_c^{n-1}\Delta V_{\text{cv}}^{n-1}}{\Delta t}-R^{n+1,m}\right) \tag{2.19}$$

that is,

$$\widetilde{A}\,\frac{W_p^{n+1,m+1}-W_p^{n+1,m}}{\Delta\tau}\Delta V_{\text{cv}}^{n+1}=\Gamma_1^{-1}\widetilde{R}^{n+1,m}$$

thus,

$$\frac{W_p^{n+1,m+1}-W_p^{n+1,m}}{\Delta\tau}\Delta V_{\text{cv}}^{n+1}=\widetilde{\widetilde{R}}^{n+1,m}$$

where

$$\widetilde{\widetilde{R}}^{n+1,m}=\widetilde{A}^{-1}\Gamma_1^{-1}\widetilde{R}^{n+1,m}\quad\text{and}\quad\widetilde{A}=I+\frac{1.5\Delta\tau}{\Delta t}\Gamma_1^{-1}M+\sum_{j=1}^{\text{nbseg}}\left(\frac{\Delta\tau}{\Delta V_{\text{cv}}^{n+1}}\Gamma_1^{-1}H_{p,j}\right)$$

Therefore,

$$\widetilde{R}^{n+1,m}=-R^{n+1,m}-K\left(\frac{1.5\Delta S_{\text{cv}}^{n+1}W^{n+1,m}-2.0\Delta S_{\text{cv}}^{n}W^{n}+0.5\Delta S_{\text{cv}}^{n-1}W^{n-1}}{\Delta t}\right)$$

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 $\widetilde{A}$ are used in the pseudo-time stepping. As a result, the equation for every node can now be written as

$$\frac{W_p^{n+1,m+1}-W_p^{n+1,m}}{\Delta\tau}\Delta V_{\text{cv}}^{n+1}=\widetilde{\widetilde{R}}^{n+1,m} \tag{2.20}$$

where

$$\widetilde{\widetilde{R}}_i^{n+1,m}=\widetilde{A}_{ii}^{-1}\Gamma_1^{-1}\widetilde{R}_i^{n+1,m}$$

and

$$\widetilde{A}_{ii}^{-1}=\text{diag}\left[\left(I+\frac{1.5\Delta\tau}{\Delta t}\Gamma_1^{-1}M+\sum_{j=1}^{\text{nbseg}}\left(\frac{\Delta\tau}{\Delta V_{\text{cv}}^{n+1}}\Gamma_1^{-1}H_{p,j}\right)\right)^{-1}\right]$$

Pseudo-time stepping is then performed on Eq. (2.20). For a five-stage scheme, the stage coefficients are

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

To speed up the convergence rate, an implicit residual smoothing scheme developed for unstructured grids is employed. The smoothing equation for a vertex k can be expressed as follows:

$$\overline{R}_k=R_k+\varepsilon\nabla^2\overline{R}_k \tag{2.21}$$

where R is the original residual, $\overline{R}$ is smoothed residual and ε is the smoothing coefficient, which can be defined as

$$\varepsilon=\max\left\{\frac{1}{4}\left[\left(\frac{\text{CFL}}{\text{CFL}^*}\right)^2-1\right],0\right\} \tag{2.22}$$

where CFL^* is the maximum CFL number of the basic scheme. The solution to the above equations can be obtained on an unstructured grid by using the Jacobi iterative method as follows:

$$\overline{R}_k^{(m)} = \frac{R_k^{(0)} + \varepsilon \sum_{i=1}^{\mathrm{numnod}(k)} \overline{R}_i^{(m-1,m)}}{1 + \varepsilon \cdot \mathrm{numnod}(k)} \tag{2.23}$$

where numnod(k) is the number of neighboring nodes of vertex k.

2.3. Boundary treatment

The far field calculations are based on characteristic variables (Reimann invariants). Thus at inflow the incoming variables corresponding to positive eigenvalues are specified while the outgoing variables corresponding to negative eigenvalues are extrapolated. Once we change the time-dependent equations we also change the characteristics of the system (though not the signs of the eigenvalues). Hence, it is also necessary to modify the boundary conditions for the preconditioned system. Here the flux at a boundary is defined as

$$\vec{F}_i \cdot \vec{n} S_k = \left(\vec{F}_i^{+} + \vec{F}_i^{-}\right) \cdot \Delta S_k$$

Here, $\vec{F}_i^{\pm}$ has been redefined as $\vec{F}_i^{\pm} = X_{H_p,R} \Lambda^{\pm} X_{H_p,L} W_p$.

Where $\Lambda^{\pm} = \frac{\lambda_i \pm |\lambda_i|}{2}$, and λ_i represents the eigenvalue of H_p (see Appendix for details of λ_i). $X_{H_p,R}$ and $X_{H_p,L}$ are the right and left eigenvectors of H_p (see Appendix).

We first calculate

$$z = X_{H_p,L} W_p = \begin{pmatrix} (1-\gamma) n_x p/(\gamma\rho) + n_z v + n_x T' - n_y w \\ (1-\gamma) n_y p/(\gamma\rho) - n_z u + n_x w + n_y T' \\ (1-\gamma) n_z p/(\gamma\rho) + n_y u - n_x v + n_z T' \\ p + n_x \rho(\lambda_1 - \beta U) u + n_y \rho(\lambda_1 - \beta U) v + n_z \rho(\lambda_1 - \beta U) w \\ p + n_x \rho(\lambda_2 - \beta U) u + n_y \rho(\lambda_2 - \beta U) v + n_z \rho(\lambda_2 - \beta U) w \end{pmatrix} \tag{2.24}$$

And then

$$\begin{aligned} \vec{F}_i^{\pm} &= X_{H_p,R} \Lambda^{\pm} X_{H_p,L} W_p = (X_{H_p,R} \Lambda^{\pm}) z \\ &= \begin{pmatrix} \left((\beta U - \lambda_2)\lambda_1^{\pm} z_4 + (\lambda_1 - \beta U)\lambda_2^{\pm} z_5\right)/S \\ n_y \lambda_0^{\pm} z_3 - n_z \lambda_0^{\pm} z_2 + \left(n_x \lambda_1^{\pm} z_4 - n_x \lambda_2^{\pm} z_5\right)/(\rho S) \\ n_z \lambda_0^{\pm} z_1 - n_x \lambda_0^{\pm} z_3 + \left(n_y \lambda_1^{\pm} z_4 - n_y^{\pm} \lambda_2^{\pm} z_5\right)/(\rho S) \\ n_x \lambda_0^{\pm} z_2 - n_y \lambda_0^{\pm} z_1 + \left(n_z \lambda_1^{\pm} z_4 - n_z \lambda_2^{\pm} z_5\right)/(\rho S) \\ n_x \lambda_0^{\pm} z_1 + n_y \lambda_0^{\pm} z_2 + n_z \lambda_0^{\pm} z_3 + (\gamma - 1)\left((\beta U - \lambda_2)\lambda_1^{\pm} z_4 + (\lambda_1 - \beta U)\lambda_2^{\pm} z_5\right)/(\gamma\rho S) \end{pmatrix} \end{aligned} \tag{2.25}$$

where $S = \sqrt{U^2(\beta-1)^2 + 4\beta c^2}$.

3. The parallel unstructured multigrid method

This work focuses on examining the one-level method of parallelization strategies in the geometric domain decomposition technique, which employs MPI [44] as the communication library. And the multigrid domain decomposition (MG-DD) approach [42] is adopted for the multigrid parallelization. METIS [45] is used to decompose the flow domain into a set of S sub-domains that may be allocated to a set of P processors. 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 it. Nodes and elements are separately renumbered as a result of the use of the SPMD (Single Programme Multiple Data) approach, i.e., each partition is treated as a separate flow domain and copies of the same code are used for all these domains for calculations. Each sub-domain is enclosed by a layer of nodes and elements, which overlap the neighboring sub-domains along the inter-partition boundaries and provide the necessary boundary conditions obtained from its neighbors. These outer most nodes in the layer are called *ghost* nodes because

they lie in the neighboring domains and their flow variables are obtained by transferring the flow conditions from their corresponding images (*core* nodes in the neighbors) to them. Communication between these *core* and *ghost* nodes is based on MPI and proper synchronization between the computations in neighboring partitions ensures that the necessary boundary conditions are correctly exchanged between them. An algorithm developed in [42] is employed to identify the ghost nodes, overlapping elements and to write the individual grid files with local numbering for each partition. Basically, it consists of four essential steps:

(1) identify ghost nodes and overlapping elements;
(2) generate individual grid files with local numbering for every partition;
(3) establish data structures for communication between core and ghost nodes;
(4) generate a script file to integrate all the result files after the parallel simulation has been performed.

The main concept of this algorithm is that those elements along the inter-processor boundaries with nodes having different partition numbers are considered as overlapping elements which are cut through by partition lines. And those nodes that formed these elements are a mixture of *core* and *ghost* nodes with the outer most nodes being ghost nodes and inner ones core nodes. Basically, a *ghost* node of a partition is the mirror image of a corresponding *core* node in a neighboring partition.

Both *speed-up* and *efficiency* are commonly used to measure the performance of a parallel code. The run time of the original serial code is used as a measure of run time on one processor. In this context, run time can be defined as the time that has elapsed from the moment when the first process actually begins execution of the program to the moment when the last process executes its last statement. In this study, run time or total simulation time starts from the moment before mesh partitioning for either single grid or MG, identifying *ghost* nodes and writing local grid files for different partitions to the time after writing all the results to the respective files. Both CPU time and wall-clock time can be used to record the total simulation time. The main difference is that CPU time is the recorded time when only the processor performs a calculation, whereas clock time is almost similar to CPU time but it includes idling time when the processor idles while waiting for other processors to communicate with. Thus wall-clock time is used to represent the total simulation time in this work since it includes the idling time, computation time and communication time, which is the true representation of the total simulation time.

The basic idea of the multigrid method is to carry out early iterations on a fine grid and then progressively transfer these flow field variables and residuals to a series of coarser grids. On the coarser grids, the low frequency errors become high frequency ones and they can be easily eliminated by a time stepping scheme. The flow equations are then solved on the coarser grids and the corrections are then interpolated back to the fine grid. The process is repeated over a sufficient number of times until satisfactory convergence on the fine grid is achieved. For ease of implementation, the non-nested mesh method using independently generated non-nested (or overset) coarse meshes is adopted [42]. Two different cycle strategies have been investigated in the present work, which are V-cycle and W-cycle. The initial solution and residuals on the coarse grid ($h+1$) are transferred from the fine grid (h) using volume-weighted transfer operators [42].

$$W_{h+1}^{(0)} = T_h^{h+1} W_h \tag{3.1}$$

$$\widetilde{\widetilde{R}}_{h+1}^{(0)} = Q_h^{h+1} \widetilde{\widetilde{R}}(W_h) \tag{3.2}$$

where $W_{h+1}^{(0)}$ and $\widetilde{\widetilde{R}}_{h+1}^{(0)}$ are the initial coarse-grid flow field values and residuals, respectively, which are transferred from the fine grid. W_h and $\widetilde{R}(W_h)$ are the fine-grid flow field variables and residual. T_h^{h+1} and Q_h^{h+1} are the solution and residual transfer operators, based on volume-weighted averaging, the later having the property of conserving the fine-grid residuals during transfer.

In order to drive the coarser grid solution using the fine grid residual, a forcing function is calculated at the first stage of the implicit Runge–Kutta time stepping scheme and subsequently added to the residual on the coarse grid. The forcing function on the coarse grid is

$$P_{h+1} = \widetilde{\widetilde{R}}_{h+1}^{(0)} - \widetilde{\widetilde{R}}(W_{h+1}^{(0)}) \tag{3.3}$$

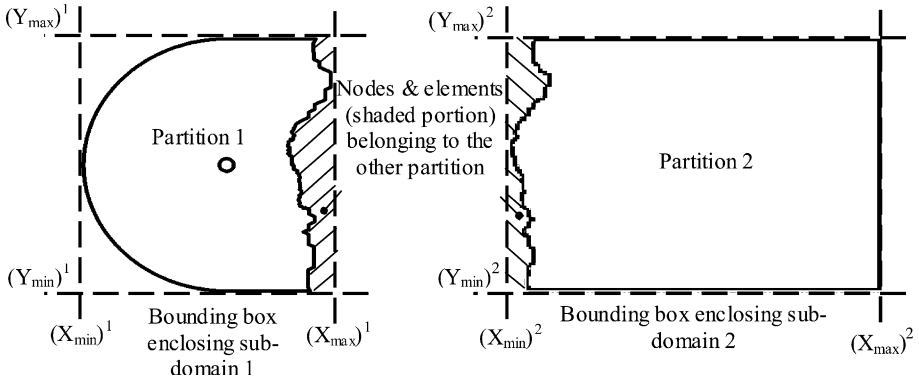


Fig. 2. Fine grid partitions to infer the coarse grid partitions [42].

It should be noted that on coarse grids, time-dependent terms in the residual containing W at n and $n-1$ time levels are not included for ease of calculation. Instead, they are only included in the fine-grid residual and directly transferred to the coarse grids.

After calculating the variables on the coarsest grid, the corrections are evaluated and interpolated back level-by-level to the finest grid. The correction is the difference between the newly computed value on the coarser grid, and the initial value that was transferred from the finer grid. This correction is transferred to the finer grid and added to the solution on that grid, i.e.,

$$W_h^+ = W_h + I_{h+1}^h\left(W_{h+1}^+ - W_{h+1}^{(0)}\right) \tag{3.4}$$

where I_{h+1}^h is an *interpolation* operator from the coarse grid to the fine grid and W_h^+ is the updated solution. To improve efficiency for the simulation of viscous flows, the viscous terms are only evaluated on the fine grid but not evaluated on the coarser grids. Since the coarser grids are only used to cancel the dominating low frequency errors, this treatment does not affect the accuracy of the solution. The upwind-biased interpolation scheme is also set to first-order at the coarser levels.

The MG-DD approach is adopted in this study. This means that the non-nested multi-grids are independently generated first. Then domain decomposition of the finest grid is performed, which is followed by decomposition of the various coarse levels of grids guided by the finer grid partitions. This is achieved by using the fine grid partitions to infer the coarse level partitions (i.e. the coarse grid is to inherit its partition from that of its corresponding finest grid) and load balancing in the coarse mesh is reasonably well ensured. A two-level multi-grid and two sub-domains are used to demonstrate the procedure of partitioning the coarse grid using the fine grid. The main idea about this algorithm is that the fine grid is partitioned into two sub-domains according to the algorithm developed for single grids. And both the maximum and minimum values in the x and y-directions ($X_{\min}$, $X_{\max}$, $Y_{\min}$ and $Y_{\max}$) of each partition for the fine grid are found. With these dimensions, an imaginary bounding box enclosing the sub-domains is formed as shown in Fig. 2. The main purpose of these bounding boxes is to identify the coarse nodes that fall within these boxes according to the fine grid partitions including those nodes beyond the sub-domain boundaries. The algorithms depicted in [42] are used to search for those actual coarse nodes that fall within a fine grid sub-domain and those nodes that fall beyond the sub-domain boundary (i.e. shaded portion as depicted in Fig. 2) are ignored. After classifying the respective coarse nodes according to which partition they belong to, then the *ghost* nodes and overlapping elements are identified using the algorithm depicted in Section 4. Individual grid files for the partitioned coarse grids and data structures for communication are then generated.

4. The immersed membrane method (IMM)

When the immersed body in a flow field is a thin structure, it will cause discontinuous fluid conditions across itself. Although velocity is continuous, the gradients of velocity, pressure, gradient of pressure and fluid stresses are quite different on both sides of the thin structure. In this work, the IMM [46] is adopted, which treats thin

structures or the boundaries of normal structures (fluid occupies one side of the membrane only) as membranes. This method uses an Eulerian background mesh for the fluid domain. When the membrane is present in the flow field, it will intersect with the Eulerian mesh. Taking a 2D mesh cell N for example, it is cut by the membrane as illustrated in Fig. 3. Fluid conditions on its nodes 1, 2 and 3 are discontinuous across the membrane. A set of imaginary ghost nodes are introduced here to replace the original nodes when they are on different sides of the membrane, i.e., nodes having discontinuous flow conditions. Considering node 1 on the left side of the membrane, node 3 stores discontinuous flow variables since it is on the other side of the membrane. And a ghost node g13 will be introduced to replace node 3 when the computation for node 1 involves conditions at node 3. In such a case, node 1 is called the real node and node 3 is called its corresponding ghost node g13. The naming convention for the ghost node is that it is prefixed with g for ghost and the first number denotes the real node and the second number the corresponding ghost node. Under this naming convention, all the ghost nodes are listed in Fig. 3. As described, the computations of convective fluxes are based on mesh edges. In the computation of convective flux along edge 23, for example, it involves the flow conditions at node 2 and 3. When the convective flux is computed for node 2, ghost node g23 is introduced to replace node 3 in the computation. Likewise, when the convective flux is calculated for node 3, ghost node g32 is introduced to replace node 2. Computations of viscous fluxes and gradients are based on mesh cells. For example, in cell N, when the viscous flux is computed for node 1, it involves fluid conditions of node 1, node 2 and node 3, and ghost node g13 is introduced to replace node 3 in the computation. The fluid variables at ghost nodes are extrapolated linearly from their corresponding real nodes based on the mesh edges which are intersected by the membrane. The extrapolated values are called ghost-node values. The novel feature of this IMM is that extrapolation of flow variables to ghost nodes is always along cell edges instead of the traditional membrane-normal direction [47,48]. As a result, every node can hold multiple ghost nodes and thus multiple ghost values since a node can be connected by multiple edges, which is especially true for 3D unstructured grids. The selection of a particular ghost value depends on which edge and node the computation is for. This feature is extremely efficient compared with the wall/membrane-normal approach, because one does not need to: (1) construct wall normal lines; (2) find out what cell faces they intersect with; (3) locate the exact positions of the interaction points; (4) interpolate the flow conditions from nearby nodes to the intersection. These calculations are very complex for a 3D surface/membrane intersecting with a 3D unstructured mesh.

Extrapolations of ghost-node velocities are illustrated in Fig. 4 (a). Taking node 2 as the real node, velocity at ghost node g23 is extrapolated as follows:

$$\begin{aligned} \frac{u_{g23} - u_I}{u_2 - u_I} &= -\frac{|r_{3I}|}{|r_{2I}|} \\ \frac{v_{g23} - v_I}{v_2 - v_I} &= -\frac{|r_{3I}|}{|r_{2I}|} \end{aligned} \tag{4.1}$$

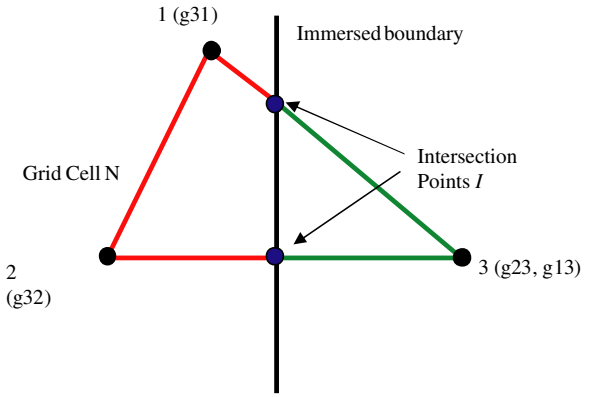


Fig. 3. Real nodes and ghost nodes of a 2D grid cell.

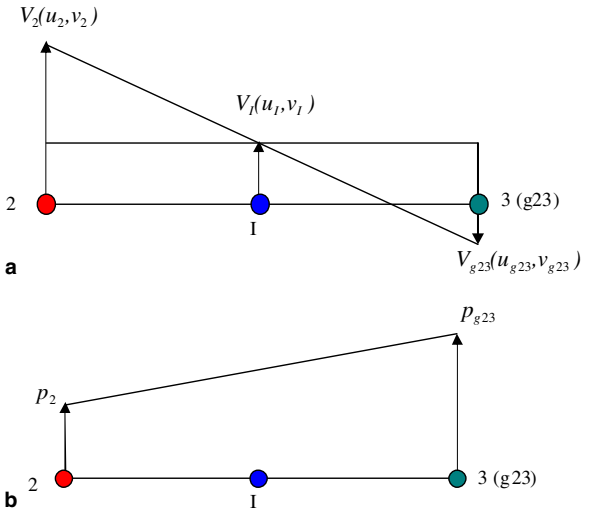


Fig. 4. (a) Velocity extrapolation for ghost node g23; (b) pressure extrapolation for ghost node g23.

Therefore,

$$u_{g23} = -\frac{|r_{3I}|}{|r_{2I}|}\cdot(u_2 - u_I) + u_I$$
$$v_{g23} = -\frac{|r_{3I}|}{|r_{2I}|}\cdot(v_2 - v_I) + v_I \tag{4.2}$$

where u_I and v_I are velocity components of intersection point I, u_2 and v_2 are velocity components of real node 2, u_{g23} and v_{g23} are the velocity components of ghost node g23. $\vec{V}_I = u_I\vec{i} + v_I\vec{j}$ used in the above extrapolations is the velocity of the immersed membrane. In this work, the structure and fluid domains are coupled by enforcing the velocity continuity condition as follows:

$$\vec{V}_s = \vec{V}_f \tag{4.3}$$

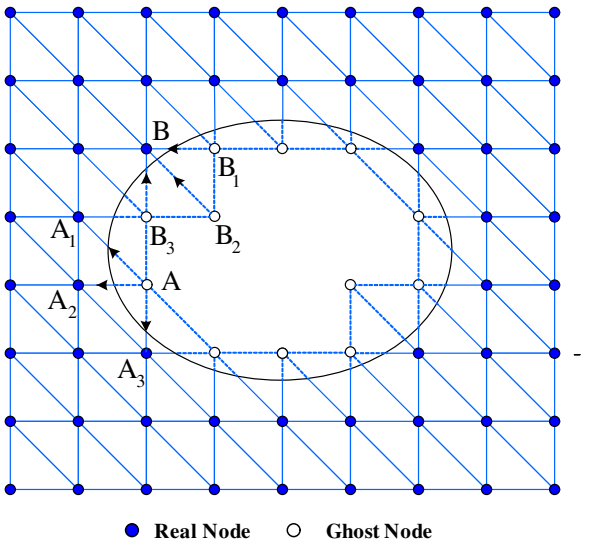


Fig. 5. The treatment of the fluid nodes inside the immersed body. For dummy node A, it can possess up to 3 different ghost values corresponding to fluid nodes A1, A2 and A3; for fluid node B, 3 dummy nodes B1, B2, B3 contribute to its flux computations.

And it is used to extrapolate fluid velocity to its corresponding ghost nodes. Therefore velocity continuity is enforced through these extrapolations to couple the fluid and structure domains. The extrapolation of ghost-node pressure is illustrated in Fig. 4 (b). Taking node 2 as real node, the ghost-node pressure of node g23 is obtained by

$$p_{g23} = p_2 + \vec{r}_{23} \bullet \nabla p_2 \tag{4.4}$$

where p_{g23} is the ghost-node pressure at node g23, ∇p_2 is the pressure gradient at node 2, $\vec{r}_{23}$ is the distance vector pointing from node 2 to node 3.

This linear interpolation results in a second-order accurate scheme. It is noted that higher-order MUSCL interpolation can also be applied here if higher order accuracy is required. On the other hand, when the immersed body is an arbitrary object with a finite volume, the given concept and interpolation method still apply, except that only the flow variables of the ghost nodes within the immersed body need to be calculated because physically there are no fluid nodes inside the body. As depicted in Fig. 5, the Eulerian fluid nodes inside the object are just bypassed in flow calculations. These inner nodes can be efficiently searched for and identified by using an internal volume mesh within the object based on a quatree search in every time step, while the surface mesh of the object is used to perform the interpolation and extrapolation.

5. Results and discussion

5.1. Order of accuracy

To determine the overall accuracy of the method, we carry out a grid convergence study for a test problem, which is a three-dimensional analogue of the problem used by Gilmanov et al. [48]. In this case, we simulate flow induced by a sphere of radius R_0, rotating at constant angular velocity Ω about the z-axis in a nearly incompressible, viscous fluid with kinematic viscosity v. The Reynolds number for this flow is defined as $\mathrm{Re} = \Omega R_0^2/v$. For Reynolds numbers in the range $Re = 1$–100, benchmark solutions for this problem have been reported by Dennis et al. [49] who solved numerically the steady, axisymmetric Navier–Stokes equations in polar coordinates using a vorticity-streamfunction formulation. In our studies, the Reynolds number is set to$Re = 100$, and we solve the full three-dimensional and unsteady flow problem with the sphere starting to rotate impulsively from quiescence relative to the stationary Cartesian grid.

The computational domain is a $(10R_0)^3$ cube with its center located at $x = 5.0R_0$, $y = 5.0R_0$ and $z = 5.0R_0$. Four uniformly spaced and successively finer mesh sizes with 20^3, 40^3, 80^3, and 160^3 grid points, respectively, are used for error analysis, and the finest-mesh solution is considered to be the 'exact' solution. The surface of the sphere is discretized with an unstructured triangular mesh consisting of 14,612 elements. The sphere is placed at the center of the cubical domain and starts to rotate impulsively at $t = 0$ with constant rotational velocity Ω about the z-axis. On all the grids the same physical time step ($\Delta t = 0.01T$) is employed in order to concentrate on the spatial resolution of the method, as done in [48]). For all the grids, the simulation is continued for one complete period, at the end of which the L_∞ and L_q norms of the u-velocity errors are calculated as follows:

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

where ε_N^∞ and ε_N^q are the infinity and qth error norms, $u_i^{(N)}$ is the u-velocity component at the ith node of the N^3 mesh, and u_i^e is the 'exact' velocity field calculated on the 160^3 grid. The results of the grid convergence study are summarized in Fig. 6, which shows the variation of the L_∞, L_1 and L_2 norms of errors with grid spacing in logarithmic coordinates. The lines with slope one and two are also given as reference. It is evident from Fig. 6 that the method is second order accurate. To further demonstrate the accuracy of our method, we also use the Richardson estimation procedure to study the accuracy of the solver as in [48]. Let f^N denote the numerical solution on the N^3 mesh. Assume that the discrete solution is a γ-order approximation to its value f^{exact}, and the flow field is continuous and has no singularity points, then we have

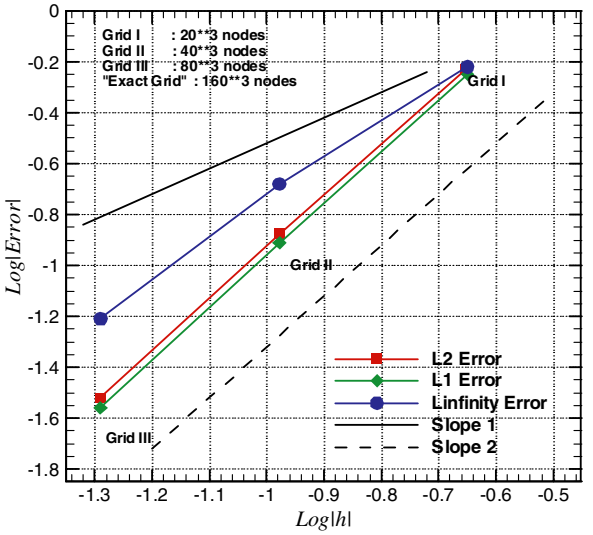


Fig. 6. Convergence of the $L_\infty L_\infty$, L_1 and L_2 error norms for the velocity field induced by a rotating sphere. Slope 1 and Slope 2 are the reference lines for 1-order and 2-order accuracy, respectively.

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

where $\|\ \|$ denotes an error norm (L_∞, L_1 or L_2). If $\gamma \approx 2$ the solution is second-order accurate. We apply the above procedure for $N = 160$ (using solutions obtained on meshes 40^3, 80^3, and 160^3) to calculate γ for successively refined meshes. And we use all three norms to compute the error and the results are summarized in Table 7.1, which strongly supports our assertion about the second-order accuracy of our method.

Fig. 7 shows several snapshots of instantaneous streamlines on the plane at $y = 2.5$. To compare the result quantitatively with those in [48,49], we compute the angle θ_s between the z-axis and the line that originates from the center of the sphere and passes through the center of the toroidal vortex ring (see Fig. 8 for definition). In Table 7.2, we compare our θ_s with those reported in [48,49] and the agreement is very good.

5.2. Flow induced by an oscillating sphere in a closed cavity

The grid convergence study is also performed for the case where a solid oscillating sphere is immersed in a fluid enclosed within a cube with solid walls. This is to demonstrate the capability of the method in handling objects moving with large displacements. The sphere of diameter $D = 1.0$ units is placed initially at the center of the cube of dimension $2 * 2$ units and oscillates horizontally with a non-dimensional time period of 1.0 and an amplitude of $0.25D$. The oscillation is effected by moving the sphere as a rigid body with velocity given by

$$u = 0.25\pi \sin(2\pi t), v = w = 0$$

The Reynolds number (based on the sphere diameter and maximum velocity) has been set to 20. The following sequence of grid sizes is employed in performing the error analysis: 20^3, 40^3, 80^3 and 160^3. And the result on

Table 7.1
Rate of convergence γ calculated for different error norms

Norm	Grids
	40^3, 80^3, 160^3
L_∞	1.82
L_1	2.84
L_2	2.15

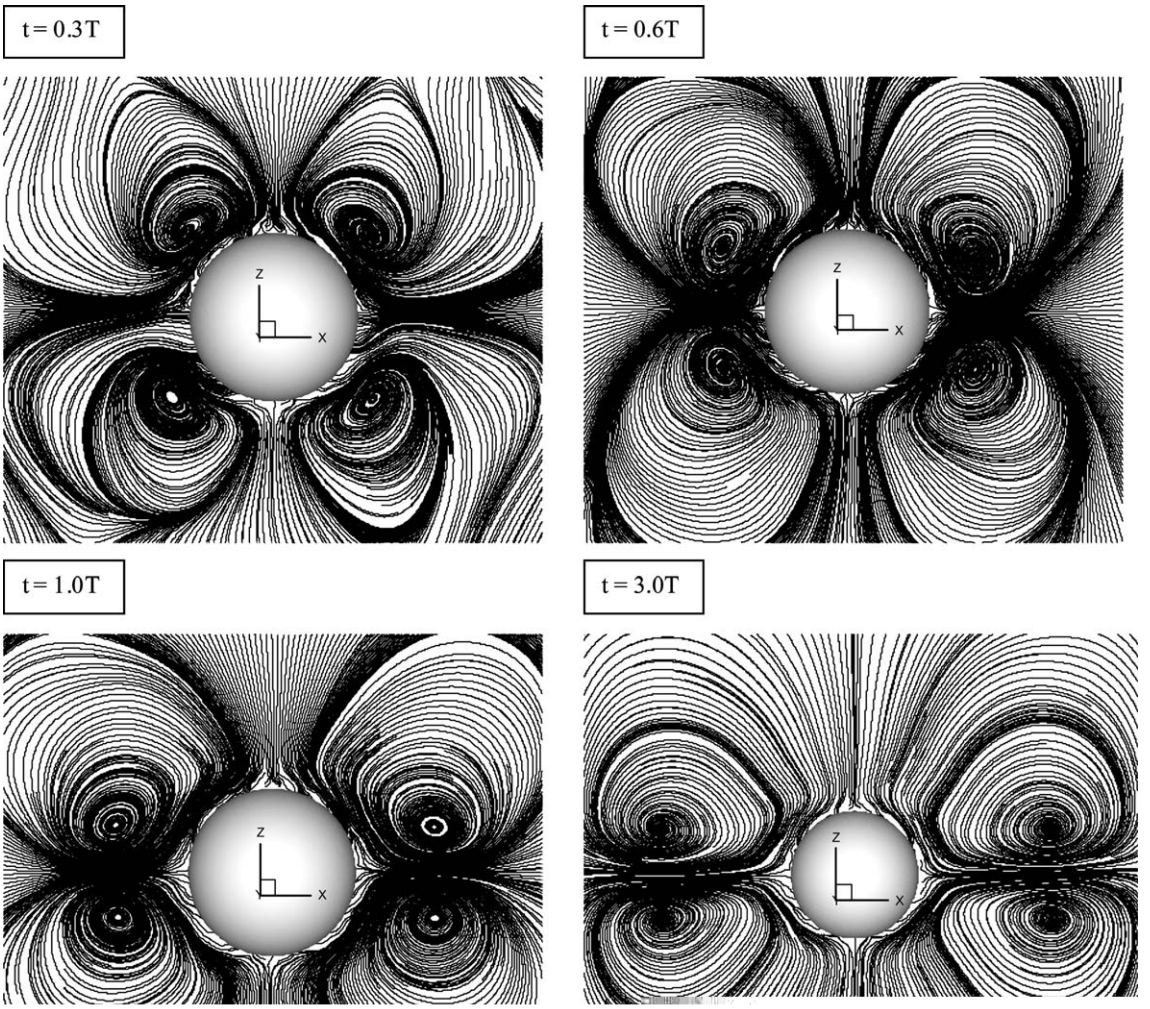


Fig. 7. Instantaneous snapshots of streamlines at the $y = 2.5$ plane depicting the early stages of flow evolution toward steady state for $Re = 100$. Time is measured from the start of the impulsive rotation and T is the rotation period of the sphere.

the 160^3 mesh is taken to be the "exact" solution for this case. A small time step of $\Delta t = 0.005$ is chosen for all these simulations in order to minimize the effect of temporal errors on the solution. The simulations are carried out for one oscillation period and the velocity components at each grid point are recorded for all the meshes under consideration at the end of the period. The instantaneous streamlines at the end of an oscillation cycle ($t = 1.0$ T) are shown in Fig. 9. Fig. 10 presents the error norms for the three meshes (20^3, 40^3 and 80^3). As can be noted, the convergence rates of errors in the simulations are close to the Slope 2 reference line, indicating second-order-accuracy.

5.3. Parallel computation of flow past a circular cylinder

Flow past a circular cylinder is a classical benchmark problem, which has been the subject of many theoretical, experimental and computational investigations due to its simple geometry and its representative behavior of general bluff body flows. Both low and high Reynolds numbers are used to demonstrate and examine the performance, accuracy and robustness of the parallel-MG compressible solver with the implementation of

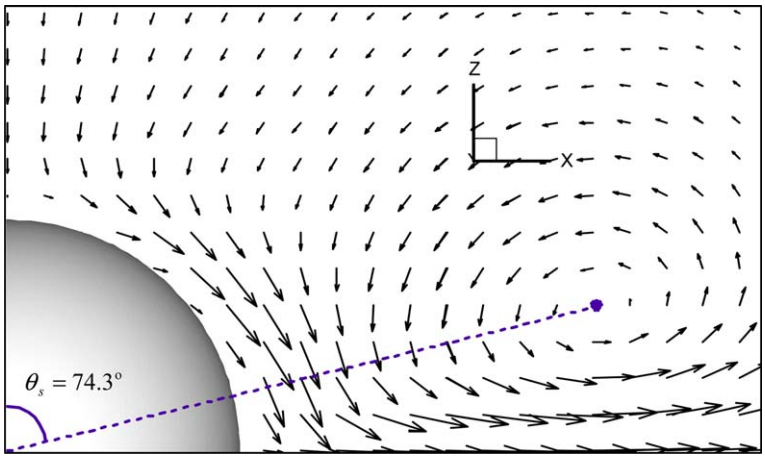


Fig. 8. The angle θ_s between the z-axis and the radius that passes through the centre of the toroidal vortex ring.

Table 7.2
Comparison of the measured [48,49] and calculated angle θ_s

Re	θ_{Dennis}	θ_{Gilmanov}	θ_{calc}	$\varepsilon(\%)$
100	73.8	72.1	74.3	0.68 (with [49]), 3.05 (with [48])

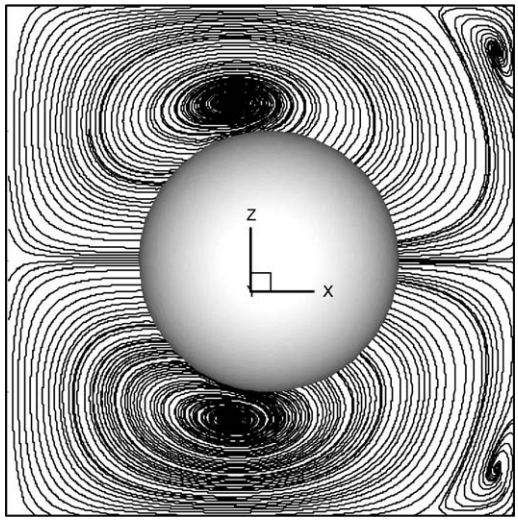


Fig. 9. Instantaneous streamlines at $t = 1.0T$ for flow induced by an oscillating sphere in a closed cube filled with compressible and viscous fluid.

IMM algorithm for steady and unsteady flows. The computational domain and the immersed cylinder are shown in Fig. 11. The parallel calculations are performed on the SGI Origin 3400 machine.

Three different grids are generated, which have 84,054, 122,202 and 182,536 nodes, respectively, and the immersed cylinder surface is discretized using 33,578 triangular elements. All the parameters for the three simulation runs are kept exactly the same and the simulation is run until non-dimensional time, $t = 150.0$ for $\mathrm{Re} = 200$ with an inflow Mach number of 0.2. The computed value of lift coefficient, C_l, is used as a criterion for convergence as shown in Fig. 12.

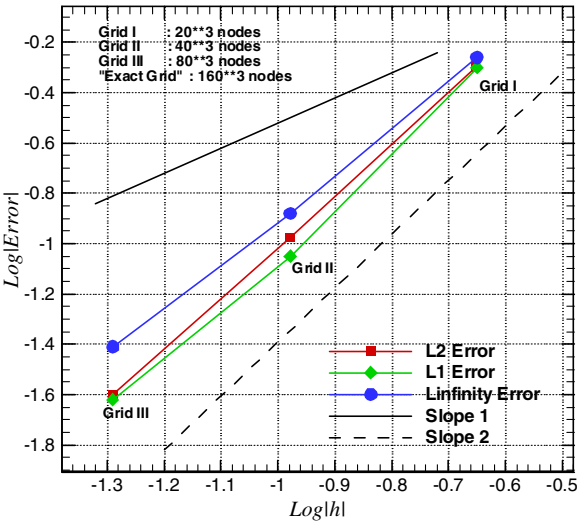
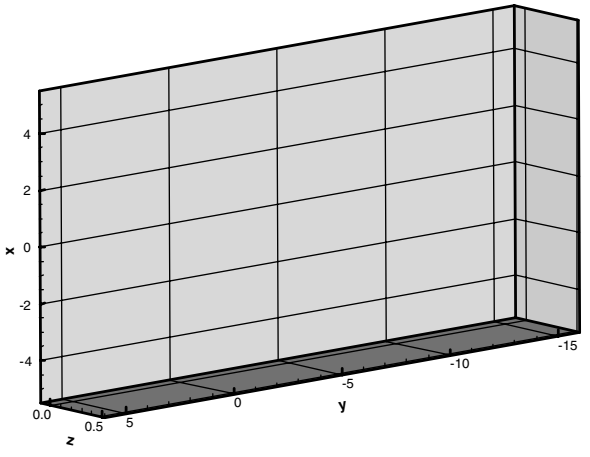


Fig. 10. Convergence of the $L_\infty L_\infty$, L_1 and L_2 error norms for the velocity field of oscillating sphere in a closed cube. Slope 1 and Slope 2 are the reference lines for 1-order and 2-order accuracy, respectively.

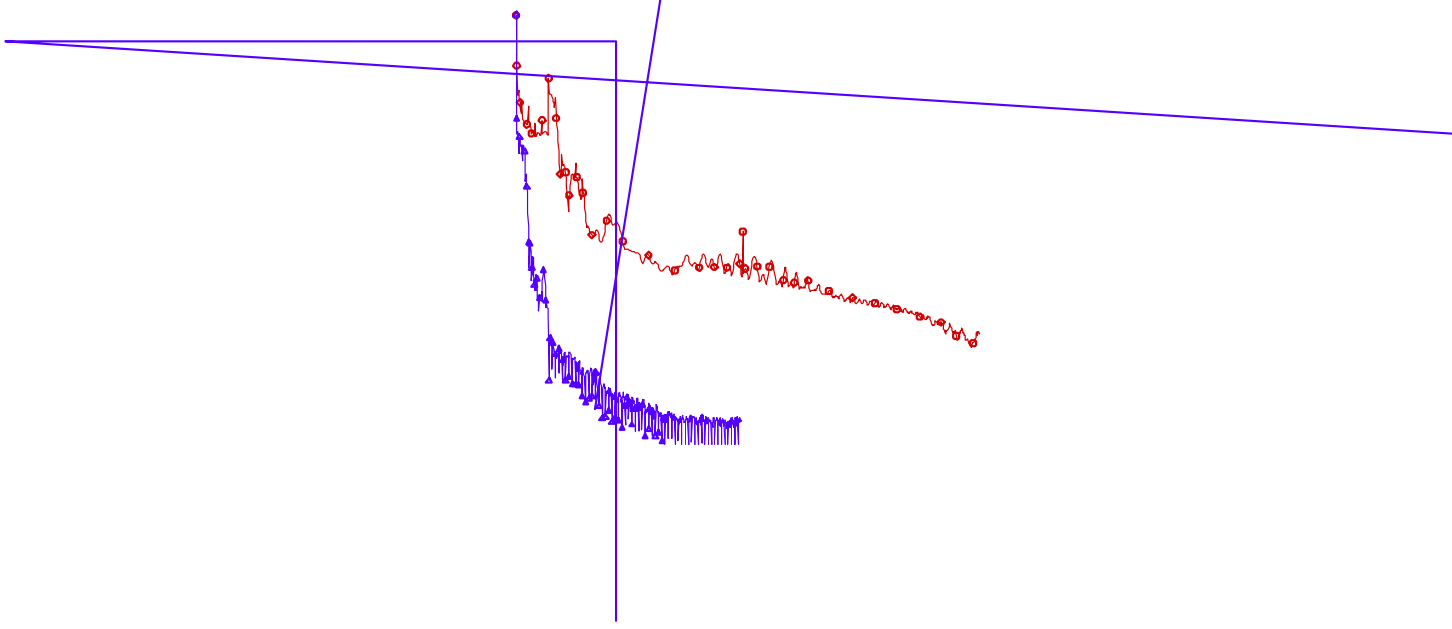
Comparing the results shown in Fig. 12, the peak value of C_l obtained using 84,054 nodes is 0.56 as compared to 0.64 obtained by both grids of 122,202 and 182,536 nodes. And similarly, comparing the results obtained by both 122,202 and 182,536 nodes, the value of C_l does not deviate from each other significantly. Since there is not much difference in the results when the nodes density is increased from 122,202 to 182,536 nodes and to minimize computational time, a grid size of 122,202 nodes is employed in the present work.



5.3.1. Steady flow

The low Reynold number specified in this computation is 40.0 and the inflow Mach number is set to be a very small value. Thus the flow can be considered as steady two-dimensional, which is started from quiescent initial condition in the simulation. It is run using the low speed preconditioning with the parallel-MG method as shown in the convergence history plot in Fig. 13. It is noted that for the same CPU time there is significant improvement in residual reduction using the preconditioning, i.e., improvement in real convergence rate.

Fig. 14(a) and (b) show the streamline plots in the wake region obtained with and without the preconditioning method, respectively. Based on qualitative comparison with the experimental result of Dyke et al. [50], the wake formed behind the cylinder predicted by the preconditioning method has better agreement with Dyke's data than the non-preconditioning one. A quantitative comparison of the aspect ratio (separation



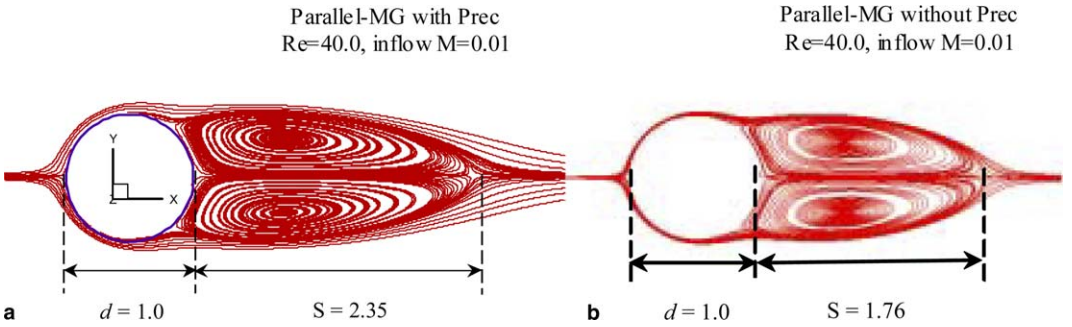


Fig. 14. Streamlines plots ($z = 0.25$) for flow over a three-dimensional stationary circular cylinder at $Re = 40$ and inflow Mach = 0.01 using (a) Parallel-MG and preconditioning, (b) Parallel-MG without preconditioning.

bubble length, S/cylinder diameter, d) with the experimental results obtained by Nishioka and Sato [51] is also carried out. Fig. 15 shows the aspect ratio versus Re. With a Re of 40.0, an aspect ratio of 2.35 is obtained, which agrees well with our preconditioned result (as shown in Fig. 14(a)). As can be observed, the low speed preconditioning method not only gives better convergence rate, it also helps to improve the quality of the numerical results when the flow speed is extremely low. This favorable characteristic is very crucial for compressible unsteady flow computation, especially for the cases where high speed flow field is embedded in low speed flow region.

5.3.2. Unsteady flow

The purpose here is to validate and assess the capability of the current parallel Navier–Stokes solver utilizing multi-grid and low speed preconditioning method as the basic convergence acceleration techniques. The computed results are compared with numerical ones obtained by other researchers, and with available experimental results. The computed lift and drag coefficients on the cylinder versus non-dimensional time with parallel-MG for an 8-partition mesh are shown in Fig. 16. A pronounced asymmetric wake begins to appear at non-dimensional time of 30. And the flow becomes completely periodic at a time of 63. It is found that the

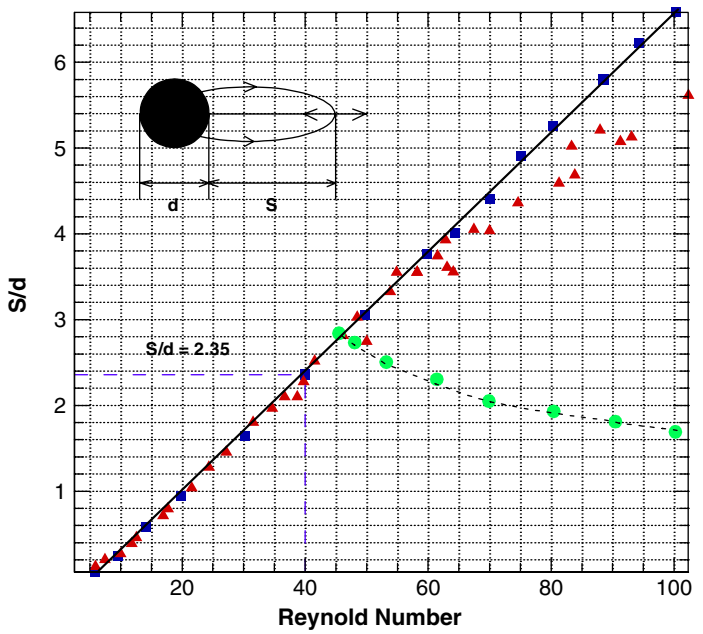


Fig. 15. Length of seperation bubbles behind cylinder vs. Re [51].

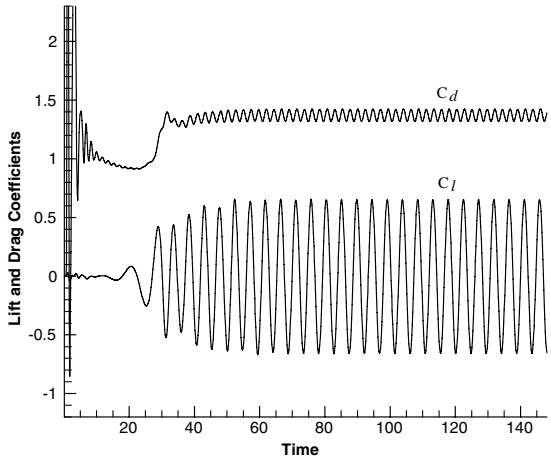


Fig. 16. Lift and drag coefficients versus physical time for flow over a circular cylinder using 3-level Parallel-MG and preconditioning (number of sub-iterations = 60 V-cycles, $Re = 200$, inflow Mach = 0.2).

number of sub-iterations for MG is much less than that required by single grid (SG) computation, which signifies that the MG method takes a shorter time than the SG to produce the vortex shedding phenomenon, thus less CPU time is needed for the flow to become fully periodic. Figs. 17 and 18 present the instant streamlines

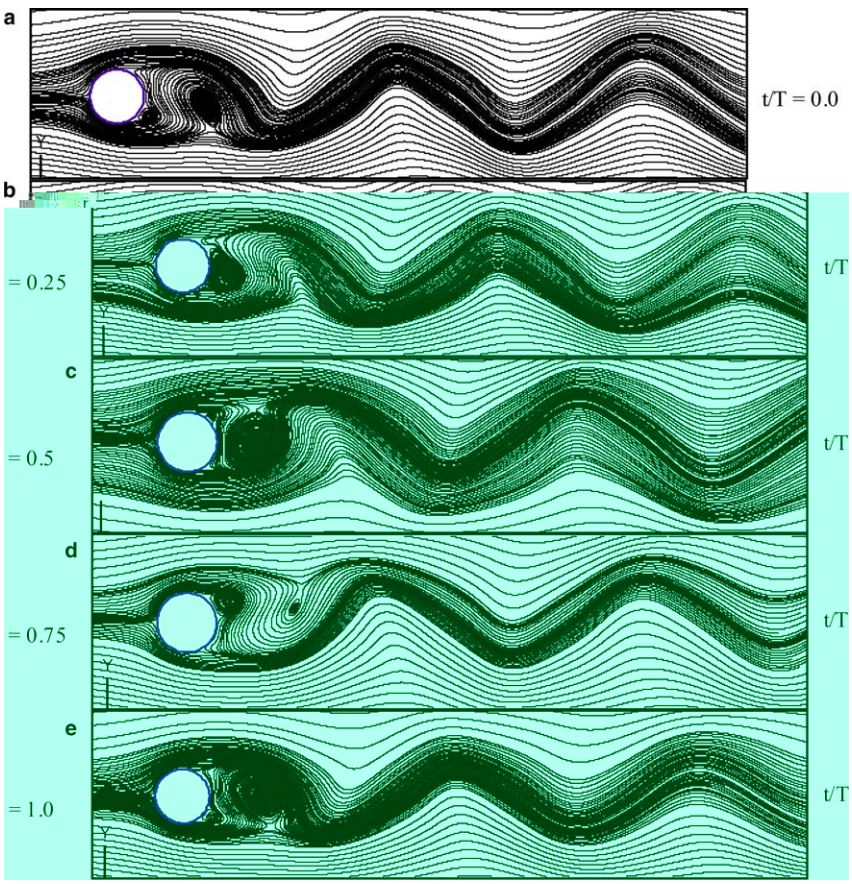
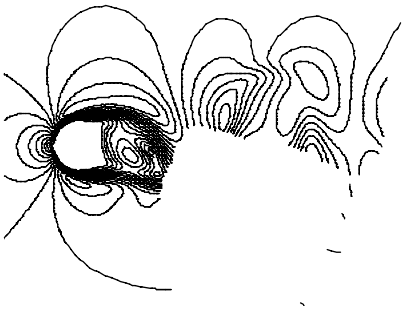


Fig. 17. Streamline patterns showing one cycle of vortex shedding using 3-level Parallel-MG and preconditioning ($z = 0.25$, number of sub-iterations = 60 V-cycles, $Re = 200$, inflow Mach = 0.2).



and Mach-number contours in one cycle of the von Kármán vortex shedding. Fig. 19 shows the convergence history in terms of numbers of subiterations in each physical time step for single grid, MG and MG with preconditioning method for a Reynolds number equal to 200. It is obvious that the combination of preconditioning and MG contributes significantly to the improvement in convergence within each physical time step, which is very useful for efficiently computing 3D unsteady flows. The lift coefficient, C_l, drag coefficient, C_d, and Strouhal number, S_t, obtained in this work are ± 0.65, 1.38 ± 0.046 and 0.196, respectively, and they agree well with numerical solutions obtained by other researchers as well as with experimental measurements [42,52–54]. And these results are tabulated in Table 7.3.

The performance of the parallel solver for simulating unsteady flow over a cylinder with $Re = 200$ using both parallel SG and MG is estimated based on the speedup characteristics, efficiency of parallelization and comparison between percentage computation and communication time, as shown in Table 7.4. The speed-up and parallel efficiency of the proposed method is found to be reasonably good. Although parallel MG has slightly low speed-up compared with parallel SG, the MG computation is still much more efficient because it requires much less number of sub-iterations for every physical time step.

5.4. Flow over an immersed fixed membrane

This case is to validate and assess the capability and accuracy of the developed method for thin structure problems. The model examined in this case is a 3D channel flow with a sinus cavity in the middle of the bottom wall. A rigid membrane is attached to the rigid channel just before the sinus cavity. The channel has a length of 10 L and a width of 1L (L is equal to 20 mm). The radius of the sinus cavity is 0.5 L. The immersed membrane has a length of 0.5 L and thickness of 0.5 L/100, and it is attached to the bottom wall at an angle of $\alpha = 42.5°$. In the region near the membrane the mesh is further refined in order to capture the fine details of the flow. The

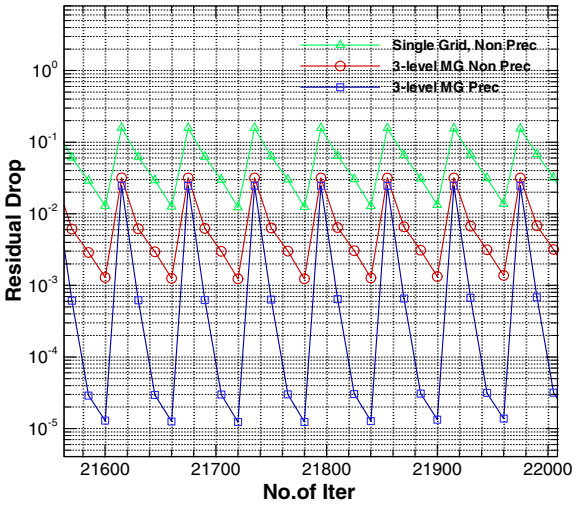


Fig. 19. Unsteady flow convergence history plot (MG used "V" cycle, number of sub-iterations = 60, $Re = 200$, inflow Mach = 0.2).

Table 7.3
Lift and drag coefficients and Strouhal number for unsteady flow over a three-dimensional circular cylinder ($Re = 200$)

Researchers	C_l	C_d	S_t
Present (parallel-MG grid)	± 0.65	1.38 ± 0.046	0.196
Tai and Zhao (parallel-MG grid, incompressible solvers) [42]	± 0.64	1.31 ± 0.041	0.195
Liu et al. [52]	± 0.69	1.31 ± 0.049	0.192
Williamson (Expt.) [53]	–	–	0.197
Wille (Expt.) [54]	–	1.30	–

Table 7.4
Performance for parallel computation of unsteady flow past a circular cylinder ($Re = 200$, $M = 0.3$ with preconditioning)

Performance measuring techniques	Single grid				Multigrid			
	Number of CPU				Number of CPU			
	2	4	8	16	2	4	8	16
Speed-up	1.77	3.48	7.02	13.4	1.72	3.29	6.39	10.18
Efficiency	0.89	0.88	0.85	0.81	0.88	0.83	0.79	0.72
Computation time and communication time (% to total simulation wall-clock time)	99.3	95.2	87.7	79.3	97.2	91.1	82.4	70.1

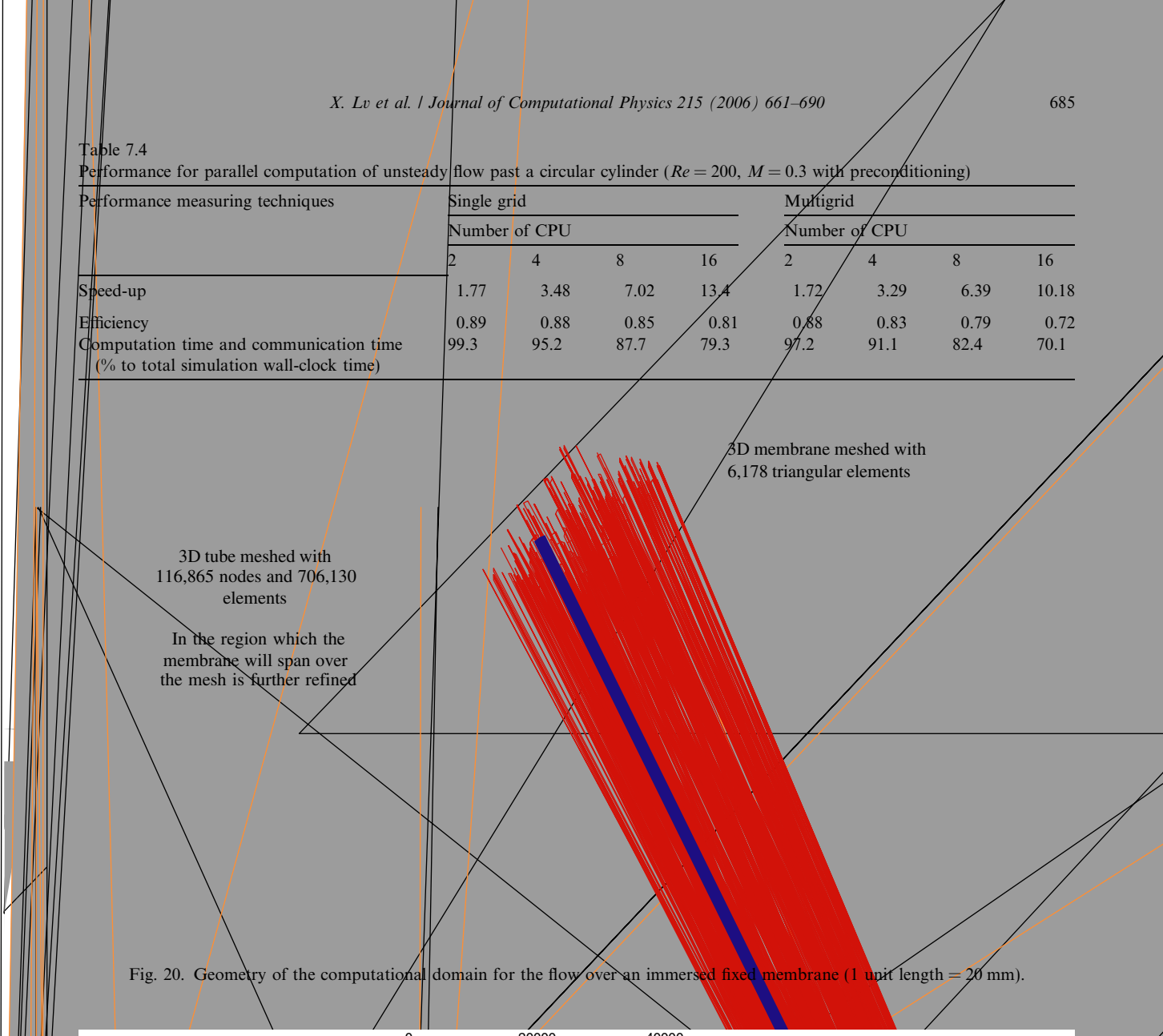


Fig. 20. Geometry of the computational domain for the flow over an immersed fixed membrane (1 unit length = 20 mm).

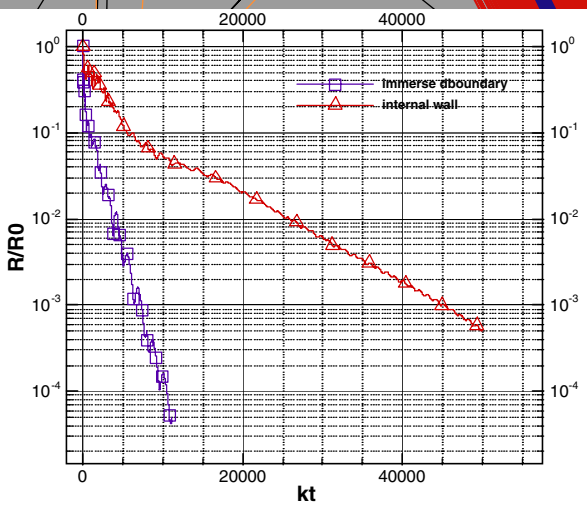


Fig. 21. Convergence histories with immersed membrane and internal boundary.

immersed membrane is discretized into triangular cells. A close-up view of the mesh and the immersed membrane is shown in Fig. 20. The inflow Mach number is 0.3 and the Reynolds number is 100. The results are then compared with those using an internal boundary calculated by the baseline preconditioned compressible parallel-MG solver. The internal boundary has the same geometry as the immersed membrane, and it is under the same flow conditions as used by the immersed membrane method.

The convergence history of the simulation given in Fig. 21 shows that the new solver based on the IMM actually converges faster and better than the baseline solver. Fig. 22 confirms that the two flow fields have

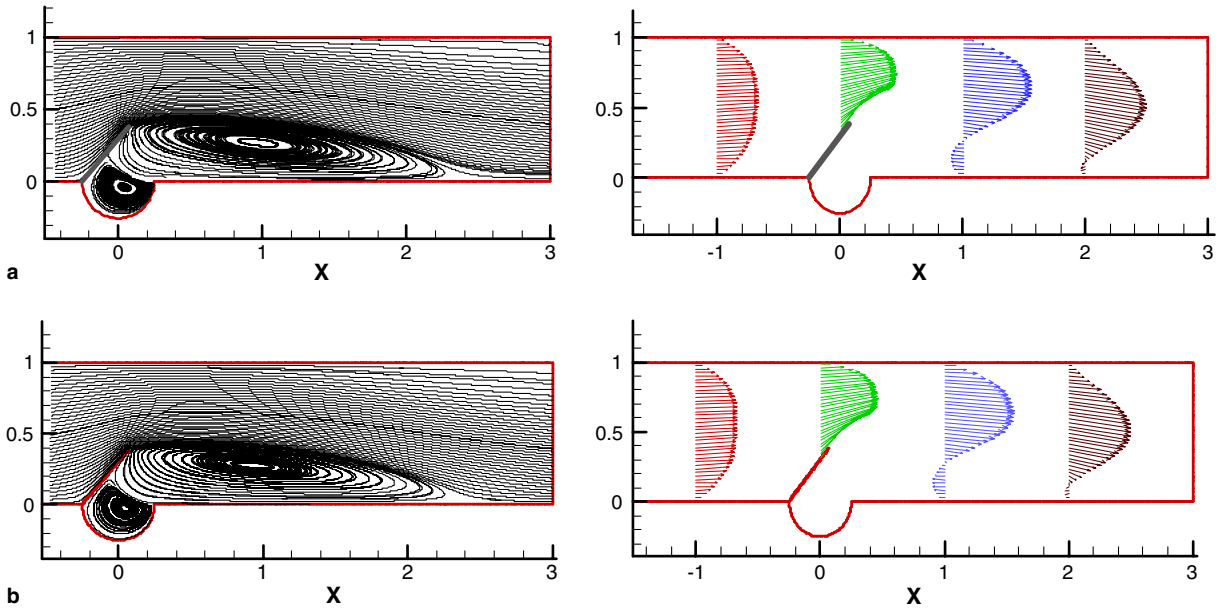


Fig. 22. (a) Streamlines on the channel central plane with the immersed membrane represented by the IMM; (b) flow field with an internal boundary computed by baseline solver.

Table 7.5
Properties of vortices of flow field with immersed membrane and flow field with internal boundary

	Immersed membrane		Internal boundary	
	Center of the primary vortex	Center of the secondary vortex	Center of the primary vortex	Center of the secondary vortex
X (mm)	0.932 L	0.067 L	0.923 L	0.047 L
Y (mm)	0.272 L	−0.031 L	0.273 L	−0.027 L
Vorticity	−5.013	0.126	−4.997	0.117

the same number of vortices with similar shapes, and the u-velocity profiles at the same locations agree well with each other as found in Fig. 23. Table 7.5 compares the two flow fields quantitatively, which demonstrates that the results obtained by the IMM agree well with those obtained by the baseline Navier–Stokes solver using an internal boundary.

6. Conclusions

A new method combining preconditioning, parallel unstructured MG method and a novel treatment of moving objects in fluids for efficient simulation of 3D unsteady compressible flows has been successfully developed and validated. The convergence of numerical solutions is found to be significantly improved with a combination of the preconditioning and unstructured MG methods. And the use of gauge pressure in pressure gradient terms is found to be important to eliminate round-off errors while the flow speed is very low. The parallel speed-up and efficiency of the method for both steady and unsteady flows are found to be reasonably good. The newly developed immersed membrane method is shown to work well with very large displacements of immersed moving objects. Compared with the immersed boundary method, this method allows sharp changes of fluid conditions across immersed thin structures without complicated and time-consuming interpolation and extrapolation. A grid convergence study for the flow generated by a steadily rotating sphere is carried out, which shows that the method is second-order accurate. The method is also found to have similar order of accuracy in another study, where a sphere immersed in a cube filled with fluid oscillates with large amplitude. Results from a flow over a circular cylinder computed by the proposed method are found to agree well with existing numerical and experimental ones. Finally flow over an immersed membrane is calculated and results are compared with those with the membrane erected as a wall, and satisfactory agreement is observed. These studies demonstrate that the method proposed is an effective tool to solve 3D unsteady low-Mach-number compressible flows with arbitrarily moving objects.

Acknowledgements

This research work is supported by a research scholarship provided by Nanyang Technological University (NTU). The provision of computing facilities by Nanyang Centre for Supercomputing and Visualization (NCSV), NTU is acknowledged. The first author would like to thank Dr. Tai Chin Hoe for his valuable discussion and provision of some data.

Appendix. Preconditioner definition

$$\Gamma_1 = \begin{pmatrix} 1/\beta'\gamma T' & 0 & 0 & 0 & -\rho/T' \\ u/\beta'\gamma T' & \rho & 0 & 0 & -\rho u/T' \\ v/\beta'\gamma T' & 0 & \rho & 0 & -\rho v/T' \\ w/\beta'\gamma T' & 0 & 0 & \rho & -\rho w/T' \\ H/\beta'\gamma T' - 1 & \rho u & \rho v & \rho w & \rho[\gamma/(\gamma-1) - H/T'] \end{pmatrix}$$

where

$$\gamma = C_p/C_v$$
$$c^2 = \gamma p/\rho$$
$$T' = p/\rho = c^2/\gamma$$
$$H = (\rho e_t + p)/\rho$$
$$\beta' = \beta/[1 + (\gamma - 1)\beta]$$

Noted that Γ_1 is a rank one modification of M (which is defined in the main body) allows one to easily compute the matrix products $\Gamma_1^{-1}M$ and $M^{-1}\Gamma_1$(which will be needed) via the Sherman–Morrison–Woodbury formula.

The general preconditioned Jacobian matrix is

$$\widetilde{H}_p = \Gamma_1^{-1}H_p = \Gamma_1^{-1}\frac{\partial F_c}{\partial W_p} = \begin{bmatrix} \beta U & \rho\beta\gamma T' n_x & \rho\beta\gamma T' n_y & \rho\beta\gamma T' n_z & 0 \\ n_x/\rho & U & 0 & 0 & 0 \\ n_y/\rho & 0 & U & 0 & 0 \\ n_z/\rho & 0 & 0 & U & 0 \\ U(\gamma-1)(\beta-1)/(\rho\gamma) & (\gamma-1)\beta T' n_x & (\gamma-1)\beta T' n_y & (\gamma-1)\beta T' n_z & U \end{bmatrix}$$

where $U = n_x u + n_y v + n_z w$, with n_x, n_y, n_z are the unit normalized vectors. The eigenvalues of $\widetilde{H}_p$ are

$$\lambda(\widetilde{H}_p) = \left(\lambda_0 = U, \lambda_0 = U, \lambda_0 = U, \lambda_{1,2} = \frac{(\beta+1)U \pm S}{2}\right)$$

where $S = \sqrt{U^2(\beta-1)^2 + 4\beta c^2}$.

The right eigenvectors of $\widetilde{H}_p$ are

$$X_{\widetilde{H}_p,R} = \begin{bmatrix} 0 & 0 & 0 & (\beta U - \lambda_2)/S & (\lambda_1 - \beta U)/S \\ 0 & -n_z & n_y & n_x/(\rho S) & -n_x/(\rho S) \\ n_z & 0 & -n_x & n_y/(\rho S) & -n_y/(\rho S) \\ -n_y & n_x & 0 & n_z/(\rho S) & -n_z/(\rho S) \\ n_x & n_y & n_z & (\gamma-1)(\beta U - \lambda_2)/(\gamma\rho S) & (\gamma-1)(\lambda_1 - \beta U)/(\gamma\rho S) \end{bmatrix}$$

And the corresponding left eigenvectors matrix is given by

$$X_{\widetilde{H}_p,L} = \begin{bmatrix} -(\gamma-1)n_x/(\gamma\rho) & 0 & n_z & -n_y & n_x \\ -(\gamma-1)n_y/(\gamma\rho) & -n_z & 0 & n_x & n_y \\ -(\gamma-1)n_z/(\gamma\rho) & n_y & -n_x & 0 & n_z \\ 1 & \rho(\lambda_1 - \beta U)n_x & \rho(\lambda_1 - \beta U)n_y & \rho(\lambda_1 - \beta U)n_z & 0 \\ 1 & \rho(\lambda_2 - \beta U)n_x & \rho(\lambda_2 - \beta U)n_y & \rho(\lambda_2 - \beta U)n_z & 0 \end{bmatrix}$$

References

[1] X.Y. Luo, T.J. Pedley, A numerical simulation of unsteady flow in a two-dimensional collapsible channel, J. Fluid Mech. 314 (1996) 191–225.
[2] X.Y. Luo, T.J. Pedley, Numerical simulation of steady flow in a 2-D Collapsible channel, J. Fluids Struct. 9 (1995) 149–197.
[3] X.Y. Luo, T.J. Pedley, The effect of wall inertia on flow in a two-dimensional collapsible channel, J. Fluid Mech. 363 (1998) 253–280.
[4] Y. Zhao, F. Ahmed, A general method for simulation of fluid flows with moving and compliant boundaries using unstructured grids, Comput. Methods Appl. Mech. Eng. 192/39–40 (2003) 4439–4466.
[5] Y. Zhao, C.H. Tai, F. Ahmed, Simulation of micro flows with moving boundaries using high-order upwind FV method on unstructured grids, Comput. Mech. 28 (1) (2002) 66–75.
[6] A.L. Gaitonde, A dual-time method for two-dimensional unsteady incompressible flow calculations, Int. J. Numer. Methods Eng. 41 (1998) 1153–1166.

[7] A.L. Gaitonde, An artificial compressibility method for the solution of the 2D incompressible Navier–Stokes equations, Report no. 715, Aero. Eng. Dept., Bristol University, 1995.
[8] P.A. Mendes, F.A. Branco, Analysis of fluid–structure interaction by an arbitrary lagrangian–eulerian finite element formulation, Int. J. Numer. Methods Fluids 30 (1999) 897–919.
[9] A. Anju, A. Marauoka, M. Kawahara, 2-D Fluid–Structure Interaction Problems by an Arbitrary Lagrangian–Eulerian Finite Element Method, The Gordon and Breach Publishing Group, 1995.
[10] M. Heil, Stokes flow in an elastic tube – A large-displacement fluid–structure interaction problem, Int. J. Numer. Methods Fluids 28 (1998) 243–265.
[11] E. Lefrancois, G. Dhatt, D. Vandaromme, Fluid–structure interaction with application to rocket engines, Int. J. Numer. Methods Fluids 30 (1999) 865–895.
[12] Peskin Charles Samuel, Flow patterns around heart valves: a digital computer method for solving the equations of motion, Ph.D Thesis, Albert Einstein College of Medicine, Yeshiva University, 1972.
[13] L. Zhu, C.S. Peskin, Simulation of a flapping flexible filament in a flowing soap film by the immersed boundary method, J. Comput. Phys. 179 (2002) 452–468.
[14] M.F. McCracken, C.S. Peskin, A vortex method for blood flow past heart valves, J. Comput. Phys. 35 (1980) 183–205.
[15] A.L. Fogelson, A mathematical model and numerical method for studying platelet adhesion and aggregation during blood clotting, J. Comput. Phys. 56 (1984) 111–134.
[16] A.L. Fogelson, Mathematical and computational aspects of blood clotting, in: Proceedings of the 11th IMACS World Congress on System Simulation and Scientific Computation, vol. 3, 5–8, North Holland, New York, 1985.
[17] J.M. Stockie, S.I. Green, Simulating the motion of flexible pulp fibres using the immersed boundary method, J. Comput. Phys. 147 (1998) 147–165.
[18] R. Glowinski et al., A fictitious domain for dirichlet problems and applications, Comput. Methods Appl. Mech. Eng. 111 (1994) 283–303.
[19] R. Glowinski et al., A fictitious domain for external incompressible viscous flow modeled by Navier–Stokes equations, Comput. Methods Appl. Mech. Eng. 112 (1994) 133–148.
[20] J. De Hart, Fluid–structure interaction in the aortic valve: a three-dimensional computational analysis, Ph.D. Thesis, Eindhoven University of Technology, 2002.
[21] F.P.T. Baaijens, A fictitious domain/mortar element method for fluid–structure interaction, Int. J. Numer. Methods Fluids 35 (7) (2001) 743–761.
[22] J. De Hart et al., A two-dimensional fluid–structure interaction model of the aortic valve, J. Biomech. 33 (9) (2000) 1079–1088.
[23] J. De Hart et al., A three-dimensional computational analysis of fluid–structure interaction in the aortic valve, J. Biomech. 36 (2003) 103–112.
[24] J. De Hart, Fluid–structure interaction in the aortic valve: a three-dimensional computational analysis, Ph.D. Thesis, Eindhoven University of Technology, 2002.
[25] R. Glowinski et al., A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: application to particulate flow, J. Comput. Phys. 169 (2001) 363–426.
[26] Yu. Zhaosheng, A DLM/FD method for fluid/flexible-body interactions, J. Comput. Phys. 207 (2005) 1–27.
[27] C.H. Tai, Y. Zhao, K.M. Liew, Parallel computation of unsteady incompressible viscous flows around moving rigid bodies using an immersed object method with overlapping grids, J. Comput. Phys. 207 (20 July) (2005) 151–172.
[28] C. Liu, X. Zheng, C.H. Sung, Preconditioned multigrid methods for unsteady incompressible flows, J. Comput. Phys. 139 (1998) 35–57.
[29] P.T. Lin, Implicit time dependent calculations for compressible and incompressible flows on unstructured meshes, M.Sc. Thesis, Department of Mechanical and Aerospace Engineering, Princeton University, 1994.
[30] C.H. Tai, Y. Zhao, A finite volume unstructured multigrid method for efficient computation of unsteady incompressible viscous flows, Int. J. Numer. Methods Fluids 46 (1) (2004) 59–84.
[31] Y. Zhao, B.L. Zhang, A high-order characteristics upwind fv method for incompressible flow and heat transfer simulation on unstructured grids, Comput. Methods Appl. Mech. Eng. 25 (6) (2001) 523–536.
[32] N.A. Pierce, M.B. Giles, Preconditioning compressible flow calculations on stretched meshes, AIAA Paper 96-0889, 1996.
[33] P. Moinier, M.B. Giles, Stability analysis of preconditioned approximations of the Euler equations on unstructured meshes, J. Comput. Phys. 178 (2002) 498–519.
[34] P. Moinier, M.B. Giles, Preconditioned Euler and Navier–Stokes calculations on unstructured grids, in: 6th ICFD Conference on Numerical Methods for Fluid Dynamics, Oxford, UK, 1998.
[35] S. Venkateswaran, D. Li, C.L. Merkle, Influence of stagnation regions on preconditioned solutions at low speeds, AIAA-2003-0435, 41st Aerospace Sciences Meeting and Exhibit, Reno, NV, January 6–9, 2003.
[36] S. Venkateswaran, C.L. Merkle, Dual-time stepping and preconditioning for unsteady computations, AIAA Paper 95-0078, 33rd Aerospace Sciences Meeting and Exhibit, January 9–12, 1995.
[37] S. Venkateswaran, P.E.O. Buelow, C.L. Merkle, Development of linearized preconditioning methods for enhancing robustness and efficiency of Euler and Navier–Stokes computations, AIAA Paper 97-2030, 1997.
[38] S. Venkateswaran, C.L. Merkle, Analysis of time-derivative preconditioning for the Navier–Stokes equations, in: 6th International Symposium on Computational Fluid Dynamics, 1995, pp.1323–1328.
[39] E. Turkel, Preconditioned methods for solving the incompressible and low speed compressible equations, J. Comput. Phys. 72 (1987).

[40] B. Van Leer, W.T. Lee, P.L. Roe, Characteristic time-stepping or local preconditioning of the Euler equations, AIAA Paper 91-1552, 1991.
[41] J.M. Weiss, W.A. Smith, Preconditioning applied to variable and constant density flows, AIAA J. 33 (11) (1995) 2050.
[42] C.H. Tai, Y. Zhao, Parallel unsteady incompressible viscous flow simulation using an unstructured multigrid method, J. Comput. Phys. 192 (1) (2003) 277–311. http://www.academicpress.com/jcp.
[43] A. Singh, Y. Zhao, Parallel unstructured dynamic grid direct Monte Carlo simulation of molecular gas dynamics and the associated thin film deposition, computational fluid and solid mechanics, in: K.J. Bathe (Ed.), Elsevier Science, The First MIT Conference on Computational Fluid and Solid Mechanics, June 11–13, 2001, MIT, Cambridge, MA 02139, USA.
[44] W. Gropp, E. Lusk, A. Skjellum, Using MPI: portable parallel programming with the message-passing interface, The MIT Press, Cambridge, MA, 1994.
[45] G. Karypis, V. Kumar, Metis: a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices, Version 4.0, University of Minnesota, Department of Computer Science, September 1998.
[46] G.H. Xia, Y. Zhao, J.H. Yeo, An immersed membrane method for simulation of fluid–structure interaction in bio-fluid flows, in: 1st International BioEngineering Conference (IBEC 2004), 08–10 September 2004.
[47] R. Mittal, G. Iaccarino, Immersed boundary method, Annu. Rev. Fluid Mech. 37 (2005) 239–261.
[48] A. Gilmanov, F. Sotiropoulos, A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies, J. Comput. Phys. 207 (2) (2005) 457–492.
[49] S.C.R. Dennis, S.N. Singh, D.B. Ingham, The steady flow due to a rotating sphere at low and moderate Reynolds numbers, Phys. Fluids 101 (1980) 257–279.
[50] Van Dyke, D. Milton, An Album of Fluid Motion, Parabolic Press, Stanford, CA, 1982.
[51] M. Nishioka, H. Sato, Mechanism of determination of the shedding frequency of vortices behind a cylinder at low Reynolds numbers, J. Fluid Mech. 89 (1978) 49–60.
[52] C. Liu, X. Zheng, C.H. Sung, Preconditioned multigrid methods for unsteady incompressible flows, J. Comput. Phys. 139 (1998) 35–57.
[53] C.H.K. Williamson, Defining a universal and continuous Strouhal–Reynolds number relationship for the laminar vortex shedding of a circular cylinder, Phys. Fluids 31 (1988) 2742–2744.
[54] R. Wille, Karman Vortex Streets, in: Advances in Applied Mechanics, vol. 6, Academic, New York, 1960, pp. 273–287.