A structured adaptive mesh refinement strategy with a sharp interface direct-forcing immersed boundary method for moving boundary problems

Mehdi BADRI GHOMIZAD∗, Hosnieh KOR∗ and Koji FUKAGATA∗

∗ Department of Mechanical Engineering, Keio University
Hiyoshi 3-14-1, Kohoku-ku, Yokohama 223-8522, Japan
E-mail: fukagata@mech.keio.ac.jp

Received: 14 April 2021; Revised: 7 June 2021; Accepted: 11 June 2021

Abstract

We develop a versatile and accurate structured adaptive mesh refinement (S-AMR) strategy with a moving least square sharp-direct forcing immersed boundary method (IBM) for incompressible fluid-structure interaction (FSI) simulations. The computational grid consists of several nested blocks in different refinement levels. While blocks with the coarsest grid cover the entire computational domain, the computational domain is locally refined at the location of solid boundary (moving or fixed) by bisecting selected blocks in every coordinate direction. The grid topology and data structure is managed by an extended version of Afivo toolkit (Teunissen and Ebert, 2018), where a novel technique is introduced for conservative data transfer between the coarser and the finer blocks, particularly in velocity transformation for which the mass conservation plays a crucial role. In the present study, the continuity and Navier-Stokes equations for incompressible flows are spatially discretized with a second order central finite difference method using a collocated arrangement and are time-integrated using a semi-implicit second order fractional step method, although the proposed S-AMR strategy can be used with different discretization schemes. An IBM using a moving least square approach is utilized to impose boundary conditions. To handle FSI problems, all the governing equations for the dynamics of fluid and structure are simultaneously advanced in time by a predictor-corrector strategy. Several test cases of increasing complexity are solved in order to demonstrate the robustness and accuracy of the proposed method as well as its capability in simulation-driven mesh adaptivity.

Keywords : Adaptive mesh refinement, Conservative interpolation, Fluid-structure interaction, Moving boundary, Immersed boundary

1. introduction

Broadly speaking, numerical solutions of fluid dynamics problems, which are originally defined in a physical domain, are obtained in numerical domains except rare cases of mesh-free methods. This numerical domain is then discretized with a fine enough grid, which results in the geometrical information of the grid nodes covering the domain. Ultimately, the governing equations are algebraically solved by using such geometrical information. Aiming at capturing the detailed characteristics of the physics and proper analysis of them, a high resolution mesh is necessary for the spots bearing the crucial aspects of the flow; namely, in the presence of high gradients (or even shocks) in the flows. Essentially two classes of mesh (i.e., grid system) are used in the computational fluid dynamics studies: structured and unstructured. Whereas each cell on a structured mesh can easily be found by a predefined relation, the cells on an unstructured mesh are arbitrarily tagged and the relation between them must be kept explicitly. While both methods can be employed for regular or irregular domains, the unstructured mesh fits better for complex geometries though with more computational costs in comparison with the structured mesh.

To a large extent, the accuracy of the numerical solution is a function of the grid quality. While both structured and unstructured meshes can be utilized for well-understood problems to meet the requirements, a dynamic adaptive mesh would be desirable for more complex cases such as problems involving moving boundaries.
A significant amount of study on adaptive mesh has been carried out over the past decades based on unstructured methods, thanks to their versatile data structure which allows us for a straightforward implementation of different refinement strategies. The most prevalent approach for mesh adaptation is called h-refinement (e.g., Rencis and Mullen, 1986), in which the localized refinement is obtained by splitting the current cells into the smaller ones, or by introducing local nodes. For moving boundary problems, mesh topology must also be altered (i.e., re-refinement) to keep the quality of the mesh around the solid interfaces. For problems with largely deforming boundaries, the aforementioned methods lack the required mesh quality, thus have an adverse impact on the accuracy and stability of the solver.

A successful strategy to cope with the difficulties in the unstructured mesh method is employing a non-conforming mesh formulation such as immersed boundary methods (IBMs). These methods solve the governing equations on a fixed Cartesian mesh. This mesh is almost never aligned with physical boundaries of the objects immersed in the flow, either fixed or moving. Instead, a fictitious force term is added to the governing equations in order to account for the presence of those boundaries. A wide variety of those methods have been put forward by researchers, which can principally be categorized into two classes: sharp and smooth interface methods. In both methods, the force can be derived either by using the physical laws (Peskin, 1972) or directly from discrete equations (Fadlun et al., 2000; Kor et al., 2017; Nishikawa et al., 2020; Yao et al., 2021). These non-conforming mesh based methods, however, lack the enough flexibility to selectively distribute more grid points in the region with high velocity gradient unless unnecessarily high mesh resolution is adopted over the other parts of the computational domain. Therefore, a formulation involving local grid refinement will be highly beneficial in terms of the computational costs.

In addition to the method to obtain the immersed boundary forces, the method by which the interface is reconstructed plays a critical role in the accuracy of IBMs, and this is another factor to categorize IBMs into two distinct classes: diffused interface and sharp interface methods. Although the merits and defects of these methods have been introduced in detail in our previous work (Badri Ghomizad et al., 2021), in both approaches, the values required for interface reconstruction is usually interpolated along a specific line direction (albeit ad-hoc), which is not consistent with the strong elliptic characteristic of the incompressible flows. However, the reconstructed velocities at the hybrid nodes (either the forcing or the ghost point) have substantial impact on the overall accuracy of the solution (Kor et al., 2018; Kor, 2018). This elliptic characteristic of the incompressible flows demands a dispersed and relatively larger stencil for interpolation rather than those typical local and ad-hoc directional interpolations.

A method similar to mesh-free methods can be used for this interface reconstruction to take into account this nature of the governing equations. Wang et al. (2010) and Ding et al. (2004) numerically solved the Navier-Stokes equations with moving boundaries by considering a cloud of nodes at the vicinity of the boundary. They applied a standard finite difference method to discretize the equations on the Cartesian nodes, while they employed the generalized finite difference (GFD) for the cloud surrounding the immersed boundary. Soltani and Badri Ghomizad (2016) also used a cloud of nodes in their interpolations by exploiting the Shepard method for heat transfer problems in conjunction with the velocity-vorticity formulation of the Navier-Stokes equations. In our previous work (Badri Ghomizad et al., 2021), we proposed a sharp interface direct-forcing IBM which can provide more accurate solution with a versatile and relatively simple scheme. In this method, using a predictor-corrector approach, we impose the desired velocity boundary condition in the intermediate velocity field by a moving least square (MLS) approximation. Then, the calculated IB forces are exerted into the momentum equation to enforce the presence of the immersed body.

Theoretically, all of aforementioned IBMs might be equipped with an adaptive mesh refinement method (AMR) to handle the complex and moving boundary problems with proper refinement around the surface and inside the domain. The AMR technique was first used by Berger and Oliner (1984) to solve hyperbolic equations, and later for shock capturing (Berger and Collela, 1989). Roma et al. (1999) employed a diffused interface IBM in an AMR framework, and their results showed a good agreement with those of the stretched nonuniform grids. In almost two decades, the AMR methods were used to resolve the required resolution for boundary layer problems (Berger and Collela, 1989; Griffith et al., 2007; Harris, 2013; Popinet, 2003; Vanella et al., 2010), mainly based on the projection schemes (Berger and Collela, 1989; Chorin, 1968).

Regarding the AMR methods for incompressible flow problems, one needs to be careful about the divergence-free data transformation between the coarser and finer grids either in prolongation or flux calculations (Martin and Collela, 2000; Minion, 1996; Almgren et al., 1998). In order to satisfy this constraint, Griffith et al. (2007) used a different discrete scheme at the interface between two different grid resolutions to get around the pressure discontinuity and to ensure the global mass conservation. Utilizing a node-based finite difference discretization, Angelidis et al. (2016) proposed a hanging node method, which satisfies the divergence-free constraint in flux calculations across the coarser and finer grids.
but not in prolongation. Stating that the accuracy of the solution highly depends on the local conservation properties of the interpolations and extrapolations, Vanella et al. (2010) put forth a divergence-free prolongation operator for the staggered grid configuration. Although they could successfully manage the problem of mass conservative interpolations, their approach was limited to the staggered configuration of variables which are notorious for complicated programming especially in IB implementation.

Usually for AMR methods, two main strategies can be identified in order to keep the needed data structure while introducing finer grids: i) isotropic splitting of individual coarse cells and merging new created cells to the exist data structure of the cells — the whole data structure is managed by a hierarchical tree (Bayyuk et al., 1993) or a fully unstructured data structure (Ham et al., 2002); ii) grid embedding, where block-structure grids composed of nested rectangular patches are used (Berger and Oliger, 1984), which is also called structured adaptive mesh refinement method (S-AMR). The S-AMR approach enjoys most merits of uniform structured grid solvers including a good quality of mesh for turbulent flows (Vanella et al., 2010). There are several libraries which implemented a variant of AMR with IBM. Popinet (2003) developed Gerries library, which adopts a non-structured grid based quad/oct tree data structure as well as a classical IBM, and Vanella et al. (2010) developed a block-based AMR method on a staggered grid configuration with a smooth IBM using an MLS interpolation. Liu and Hu (2018a) also presented a block-based algorithm with a special finite volume discretization (Liu and Hu, 2018b) and a variant of MLS for their interpolation. Their proposed algorithm could be classified as a sharp interface IBM, however without addressing the inherent spurious oscillations typically occurring in the sharp interface implementation. Therefore, an AMR solution with a sharp interface IBM with less spurious oscillations would be highly beneficial for studying the fluid-structure interaction (FSI) problems, especially in presence of multi solid moving objects.

Having introduced an accurate and efficient collocated scheme for a moving least square-based sharp direct forcing IBM (MLS-IBM) in our previous work (Badri Ghomizad et al., 2021), here we propose a novel S-AMR approach combined with MLS-IBM for further efficiency in FSI problems, especially with moving boundaries. This strategy inherits the accuracy and versatility of the boundary imposition from MLS-IBM and efficiency of S-AMR methods with collocated configuration, while satisfying the mass conservation constraints. We have adopted a predictor-corrector (Liao et al., 2010) scheme for the direct forcing implementation with an MLS approximation to increase the accuracy of this sharp interface method while reducing the spurious forcing oscillation. In contrast to the other version of the MLS based algorithm (Vanella et al., 2010), forces are directly calculated on the Eulerian nodes (and inside the entire solid body), thus the boundary condition is not diffused at the interface. In contrast to the majority of the aforementioned solutions which use the distributed memory for their implementation (Popinet, 2003; Vanella et al., 2010; Liu and Hu, 2018) and are somewhat limited to the fixed boundary problems (Popinet, 2003), the present method is based on a shared-memory parallel solver, which allows us to efficiently utilize the moving least square sharp interface immersed boundary method for moving boundary problems without the load balancing bottleneck in those distributed memory methods. In addition to the efficiency, the sharp interface boundary representation in the proposed method will equip us with a suitable tool for the internal or external turbulent flow studies compared to those having smeared boundary (Vanella et al., 2010).

The remainder of the paper is organized as follows. The governing equations and the numerical methods for MLS-IBM are outlined in Section 2. Section 3 introduces the present S-AMR method and the corresponding operators used for this study — especially, the conservative treatments in the prolongation and restriction operators. In Section 4, we thoroughly analyze some test cases to validate the code and show the capability of the proposed method particularly for FSI problems. Finally, Section 5 gives a brief summary of the present method with some outlooks for the future research.

2. Mathematical model

2.1. Governing equations

As schematically shown in Fig. 1(a), we consider an FSI problem in a domain \( \Omega \). The fluid and solid body respectively occupy the sub-domains \( \Omega_f \) and \( \Omega_s \), and interaction takes place at their interface \( \partial \Omega_f = \Omega_f \cap \Omega_s \). We assume the whole system, whose outer boundary is indicated by \( \partial \Omega_f \), is under the gravitational acceleration \( g \). The motion of the solid body is represented by the translational velocity \( v_s \) and the angular velocity \( \omega_s \). Also, the vector \( x^s \) points to the center of the mass, \( x^f \) is the location vector at the surface of the solid object, and the vector \( r^s = x^s - x^f \) is the location vector of the surface point with respect to the center of the mass.

The fluid motion is governed by the incompressible continuity and Navier-Stokes equations,

\[
\nabla \cdot u = 0, \tag{1a}
\]

\[

\]
\[ \frac{\partial u}{\partial t} + \nabla \cdot (uu) = \nabla \cdot \sigma_f + g + f, \quad (1b) \]

where \( u \) is the fluid velocity vector, and the presence of the solid is accounted for by the body force \( f \). The fluid stress tensor \( \sigma_f \) is defined as

\[ \sigma_f = -\frac{P}{\rho_f} I + \nu \left( \nabla u + (\nabla u)^T \right), \quad (2) \]

where \( p \) is the pressure, \( \rho_f \) is the fluid density (assumed to be constant in the present study), \( I \) is the identity matrix, and \( \nu \) is the kinematic viscosity.

Three-dimensional motion of the rigid object can be described by three translational \( x = (x, y, z) \) and three rotational \( \theta = (\theta_x, \theta_y, \theta_z) \) degrees of freedom, while these are reduced to only three variables \( (x, y, \theta_z) \) in two-dimensional problems.

The rigid body motion is determined by the force \( F_s \) and torque \( T_s \) acting from the fluid, i.e.,

\[ m_s \frac{du_s}{dt} = F_s + \left( 1 - \frac{\rho_s}{\rho_f} \right) m_s g, \quad (3a) \]

\[ I_s \cdot \frac{d\omega_s}{dt} = T_s - \omega_s \times (I_s \cdot \omega_s), \quad (3b) \]

where \( m_s \) and \( I_s \) the mass, the moment of inertia (tensor) of the solid body, \( u_s \) and \( \omega_s \) are the translational and rotational velocities, and \( \rho_s \) is the density of the body. These force and torque are obtained as

\[ F_s = \int_{\partial \Omega_s} \sigma_f \cdot n \, dS, \quad (4a) \]

\[ T_s = \int_{\partial \Omega_s} r \times (\sigma_f \cdot n) \, dS, \quad (4b) \]

where \( n \) is the outward normal vector on the fluid-solid interface, \( \partial \Omega_s \). It is proven that those surface forces can be recast to the body forces (De, 2018); therefore, by the definition of the immersed boundary forces (i.e., \( f \) in Eq. (1b)), \( F_s \) and \( T_s \) can be computed as

\[ F_s = -\rho_f \int_{\Omega_s} f \, dV + \rho_f m_s \frac{du_s}{dt}, \quad (5a) \]

\[ T_s = -\rho_f \int_{\partial \Omega_s} (r \times f) \, dV + \rho_f \left( I_s \cdot \frac{d\omega_s}{dt} + \omega_s \times (I_s \cdot \omega_s) \right), \quad (5b) \]

with \( r \) being the location vector for each element in the solid domain. The above equations are used to manage the dynamics of the solid body. Although these equations are in many ways akin to the strong form of fictitious domain methods (Glowinski et al., 2001), the sharp interface direct forces, \( f \), being discussed more in Section 2.2, are distinctly different from their counterpart in those method. The updated position of the rigid body can be found by solving a displacement equation \( dX/dt = U \) where \( X \) is the mass center of the body. The second order Adams-Bashforth scheme is employed to to solve this equation. Also it should be emphasised that the solid body must be rotated with angular velocity whenever it is needed.
2.2. Fluid flow solution with moving least square reconstruction

We employ a cell-centered collocated configuration of the variables (Almgren et al., 1998) with the fractional step method of Kim and Moin (1985), since it is more convenient for AMR. Although readers are referred to our previous work (Badri Ghomizad et al., 2021) for more details on the present MLS-IBM, we briefly overview in this subsection the discretization and flow reconstruction methods used in the present study, which is common to our previous work.

This algorithm start with calculating a provisional velocity, \( \tilde{u} \), as

\[
\frac{\tilde{u} - u^n}{\Delta t} = -\frac{3}{2} \nabla_h \cdot (U u)^n - \frac{1}{2} \nabla_h \cdot (U u)_{n+1} + \frac{1}{2} \nabla_h^2 (u^n + \tilde{u}) - \nabla_h \mathcal{P} + f^{n+1},
\]

where \( U \) is the contravariant component of the velocity field residing at the cell faces, \( \mathcal{P} = \rho^n / \rho_f \), and the immersed boundary force \( f^{n+1} \) is added to impose the desired boundary condition within a fixed time step, \( \Delta t \). In this equation, \( \tilde{u} \) is an estimation of the cell center velocity at time step \( n + 1 \), which is not necessarily divergence-free although it meets the boundary condition because of the immersed boundary force term. The immersed boundary force can be computed as (Uhlmann, 2005)

\[
f^{n+1} = u_f - \tilde{u} - H^{1/2},
\]

where \( \tilde{u} \) is a predicted provisional velocity obtained by the explicit form of Eq. (6), i.e., using the second-order Adams-Bashforth method. The second term, \( H^{1/2} \), is the summation of the advection, pressure, and viscosity terms in the Navier-Stokes equation evaluated using the explicit form. The target velocity \( u_f \) complies with the solid motion in \( \Omega_s \) at time step \( n + 1 \) (i.e. \( u_f = (u_f^{n+1} + \omega_s^{n+1} \times r) / \Delta t \)) and takes into play the desired boundary condition by using MLS approximation, whose details have been discussed in our previous work (Badri Ghomizad et al., 2021). In this study, we impose the forcing term not only at the vicinity of the solid boundary, but also inside the entire solid domain. This will improve the accuracy particularly in terms of spurious oscillation in the calculated force (Liao, 2010; Tajiri et al., 2014; Badri Ghomizad et al., 2021).

Then, the divergence-free velocity field, \( u^{n+1} \) and \( U^{n+1} \), at time step \( n + 1 \) is calculated as

\[
\begin{align*}
\frac{u^{n+1} - \tilde{u}}{\Delta t} &= -\nabla_h \varphi_{cc}, \\
\frac{U^{n+1} - \tilde{U}}{\Delta t} &= -\nabla_h \varphi_{fc},
\end{align*}
\]

where \( \nabla_h \varphi_{cc} \) and \( \nabla_h \varphi_{fc} \) denote the gradients of \( \varphi \) at the cell center and the cell face, respectively. The pressure corrector term \( \varphi \) is calculated from the following Poisson equation under the homogeneous Neumann boundary condition on all boundaries:

\[
\nabla^2 \varphi = \frac{\nabla_h \cdot \tilde{U}}{\Delta t}.
\]

Finally, the updated pressure, \( \mathcal{P}^{n+1} = \mathcal{P} + \varphi - (1/2) \nabla \cdot \tilde{u} \), is obtained as well. It should be mentioned that elliptic partial differential equations in this study such as the Poisson equation (9) are solved by using a multigrid method. The error in the solution is iteratively damped on a hierarchy of grids, in which the low frequency (i.e., long wavelength) error components are reduced on the coarse grids, and the high frequency errors on the finer ones. Since the fine grid on an octree mesh generally does not cover the whole domain, a Full Approximation Scheme (FAS) version of multigrid methods is employed. For further details one can refer to (Teunissen and Ebert, 2017; Brandt and Livne, 2011) and the references therein.

The so-called forcing nodes, where the immersed boundary force is enforced, are grid nodes inside the fluid domain neighboring at least one node inside the solid domain. For these nodes, the velocity field must be reconstructed so that it satisfies the boundary condition. We employed the MLS approximation to obtain a velocity at forcing nodes satisfying the physical boundary condition as well as the velocity at the surrounding nodes. To this end, for each forcing node, the surrounding nodes inside the fluid domain are sought. As illustrated in Fig. 1(b), one can obtain the stencil points for the forcing node \( f \) using its associated normal vector to the surface, \( n_f = (\nabla \phi) / ||\nabla \phi||_f \), where \( \phi \) is the distance function (see Section 2.3 for its calculation). Assuming that the indices of the forcing node are \((i, j)\), we can find five nodes around it (inside the fluid domain) by moving along \( n_f \). These nodes are exemplified in Fig 1(b) and classified by \( A = \{A_1, \ldots, A_5\} \). Interrelated to \( A_1, A_2 \) and \( A_3 \), the location of the points \( S = \{S_1, \ldots, S_3\} \) on the solid-fluid interface are also obtained as \( S_i = x_A - n_A \phi_A \) (\( i = 1, \ldots, 3 \)).
The domain for MLS reconstruction of a forcing node is composed of both the nodes in \( \mathcal{A} \), which are not a forcing nodes, and all nodes in \( \mathcal{S} \). Each component of the velocity vector at a forcing node is then approximated as

\[
u(x)^h = \sum_{i=1}^{n} p_i(x) a_i(x) = P^T a, \quad (10)
\]

where \( p_i(x) \) and \( a_i(x) \) the basis functions and coefficients, respectively, and the number of basis functions used is denoted by \( m \). In the matrix form, these bases and coefficients are written as \( P \) and \( a \). Given \( n \) nodes and their related values, \( u = \{ u_j, \ (j = 1, \ldots, n) \} \) in the approximation domain, the coefficients in vector \( a \) in Eq. (10) can be obtained by minimizing the error functional,

\[
J(x) = \sum_{j=1}^{n} \left[ (u^d(x_j) - u_j)^2 w(x_j) \right] = (Pa - u)^T W (Pa - u), \quad (11)
\]

where the weight matrix is \( W = \text{diag}(w(||x - x_j||), j = 1, \ldots, n) \). Here we chose the cubic spline function as the weight function, \( w \). By minimizing Eq. (11), the approximated function, \( u^d(x) = p^T (p^T W p)^{-1} p^T W u \), is computed. For more details, readers are referred to Badri Ghomizad et al. (2021) and the references therein.

2.3. Interface representation

Due to the significant impact of the interface representation on the accuracy of the immersed boundary imposition especially for sharp interface methods, a precise shape reconstruction plays a pivotal role, particularly for moving boundary problems. The shape of the object can implicitly be represented by the zero-value contour of a distance function, \( \phi(x), \ x \in \mathbb{R}^d, \) with \( d = 2, 3 \). From a set of markers put on the solid-fluid interface, \( \Gamma \), the distance function \( \phi \), for a node, \( x, \ x \in \Omega \), from the nodes located on \( x^f \), is described as

\[
\phi(x_i) = \text{sgn}(n_i \cdot (x_i - x^f_i)) \min(||x_i - x^f_i||), \quad (12)
\]

where \( x^f_i \in \Gamma \) indicates the coordinate of Lagrangian markers laid on the interface \( \Gamma \), and \( n_i \) denotes the surface-normal vector on them. Differed from our earlier work (Badri Ghomizad et al., 2021), we opt for a radial basis function interpolation to directly reconstruct the interface with \( \phi \) as suggested by Biancolini (2017). This strategy is computationally more efficient and on-target mechanism for distance function calculation; therefore, it can be used in every time step to competently update the interface of fixed and moving objects.

Given a set of \( n \) Lagrangian markers put on the fluid-solid interface \( (x_i, \ i = 1, \ldots, n) \) with an arbitrary positive value \( \lambda \) for the surface isoline, we seek for an interpolant \( s(x) \) with a radial basis interpolation, so that \( s(x) \) is enforced to be constant \( s(x_i) = \lambda \) on the surface, while at other locations, it is required to be a linear combination of translated radial bases and a low order polynomial as

\[
s(x) = \sum_{i=1}^{n} \alpha_i \phi(||x - x_i||) + \sum_{k=1}^{M} \beta_k p_k(x), \quad (13)
\]

where \( \psi \) denotes a radial basis function from \( \mathbb{R}^d \) to \( \mathbb{R} \), and \( p_k(x) \) are all independent multivariate polynomials up through degree \( m \), with \( M = \binom{m+d}{m} \) where \( \binom{m+d}{m} = (m + d)!/(m!d!) \) denotes the binomial coefficient. The second term in Eq. (13) is a regularization term required for providing non-singularity: see, e.g., Bayona et al. (2019) for details.

The radial basis fit is uniquely determined if the weights, \( \alpha_i \), and the coefficients of the polynomial, \( \beta_k \), can be found by solving the system of equations in which \( s(x_i) = \lambda \) for all given points, i.e., \( \lambda \in \mathbb{R}^n \); but here, the number of equations is insufficient because the number of unknowns is \( n + M \). Therefore, extra degrees of freedom are taken up by orthogonality conditions on the coefficients \( \alpha_i \) (Biancolini, 2017; Bayona et al., 2019), i.e.,

\[
\sum_{i}^{n} \alpha_i p_k(x_i) = 0, \quad k = 1, \ldots, M. \quad (14)
\]

By adding those extra constraints, one can find the following system of linear equations for the required coefficients:

\[
\begin{bmatrix}
\Psi & P \\
 P^T & 0
\end{bmatrix}
\begin{bmatrix}
\alpha \\
\beta
\end{bmatrix}
=
\begin{bmatrix}
\lambda \\
0
\end{bmatrix},
\quad (15)
\]

where \( \Psi \in \mathbb{R}^{nxn} \) and \( P \in \mathbb{R}^{nxm} \) are respectively the radial basis and the added polynomials in a matrix form. The unknowns \( \alpha \in \mathbb{R}^n \) and \( \beta \in \mathbb{R}^M \) are the vector of coefficients in Eq. (13), where \( \beta \) can be interpreted as the Lagrange multipliers that constrain \( \alpha \) to the space \( P^T \alpha = 0 \) (Flyer et al., 2016).
Having found the coefficients in Eq. (15), we can calculate the distance function \( \phi \) in each time step by Eq. (13).

\[
\phi(x) = -[s(x) - \lambda]
\]

where its zero isoline would represent the shape of the object and the positive and negative values respectively indicate the outside and inside of the object.

3. Structured adaptive mesh refinement (S-AMR) strategy

In this section, we present the details of the present S-AMR, i.e., the mesh topology and the adaptation in addition to its corresponding interpolation operators for ghost cells, prolongation and restriction. Note that, although we consider only two-dimensional problems in the present paper, this method can be straightforwardly extended to the three-dimensional problems.

3.1. Grid topology

A nested grid blocks with \( n_x \times n_y \) computational cells at different refinement levels, \( \Omega^l \), \( l = 0, 1, \cdots, l_{\text{max}} \), forms the computational grid for our calculations, where the coarsest grid blocks at level \( \Omega^0 \) always cover the entire computational domain. The refined grid block is achieved by bi-sectioning the coarse block grid in every coordinate direction. Therefore, in two-dimensional problems, the block \( b \) at level \( l \) will be the parent of four children grid blocks at level \( l + 1 \), occupying the same area of their parent. We define the leaf block as the block with highest level of refinement in a particular region of the domain. The leaf blocks in any refinement level might share their boundaries with physical boundaries of the domain or other blocks either in the same level or at most one level different in the refinement level.

Figure 2(a) exemplifies a computational domain with three levels of refinement. The domain pertained to each level is denoted by \( \Omega^n \), where the superscript \( n \) indicates the level of refinement. For instance, in Fig. 2(a), three levels of nested grids are shown. In our algorithm, the computational domain is covered by a series of boxes with fixed number of cells in the coarsest resolution. Then, wherever required, the specified box is refined by four (in two-dimensional cases) boxes with a twice finer resolution. For example, in Fig. 2(a), the entire domain has been covered by two square grid boxes at level \( \Omega^0 \). The right square box in the figure has been refined with four boxes in the level \( \Omega^1 \) and the lower right box of the level 1 has also been refined with four boxes with level 2. We should point out that, in this work, we use a 1 : 2 constraint for our refinement levels. That means the refinement level of each box is not allowed to be different more one level with its neighbors. Consequently, when a box is going to be refined, we also adjust the refinement of all its surrounding boxes with its new refinement level. In present study, a layer of one cell surrounding each box, called ghost cells, is employed to transfer data between adjacent boxes. This ghost cell is served as a tool for transferring data between neighbor blocks either in the same refinement level or different levels. The detail of the data transformation are set out in the following section (Section 3.2.1). It should be emphasized that by the constraining the adjacent mesh grids the \( \Delta x_i \) for two neighbor grids with different refinement level will differ by factor two. The resulting grid structure is managed using the oct-tree data structure in the Afivo toolkit (Teunissen and Ebert, 2017; Teunissen and Ebert, 2018), which delivers an efficient tools for a robust implementation. In the Afivo library, a variable can be stored in cell center or face center; thus, it simplifies the implementation of our collocation scheme. In following sections, we discuss a customized prolongation and restriction methods elaborated for our FSI study.
3.2. Interpolation operators
To complete each step of the proposed direct forcing fractional step method, without special treatment for boundaries, one needs to fill the ghost cells, which can be achieved by proper interpolations. Moreover, upon refinement or coarsening the blocks, the field variables must be interpolated from coarse to fine grids (prolong) or from fine to coarse (restrict).

3.2.1. Interpolation for ghost cells
As mentioned in Section 3.1, ghost point layers in blocks are defined for cell-center variables to facilitate the calculation at the refined boundaries. We opted for the conservative interpolation method among the available choices in the Afivo library. We provide a brief description for this scheme. For more detail, one can refer to Teunissen and Ebert (2018) and their references.

An interface between coarse and fine level grids is depicted in Fig. 2(b). A generic cell-center variable in this figure is indicated by \( \phi^H \) on the coarse side and by \( \phi^h \) on the refined side. We assume that each block contains \( N \times N \) nodes and the ghost cells of the box are shown by \( g \) with a \( H \) or \( h \) superscript for coarse and fine grid, respectively, while the flux is denoted by \( f \) with the same superscripts. Adopted from the Afivo library, the ghost cell value for a coarse node is obtained by a bilinear interpolation from the surrounding fine nodes. The ghost cell value for a finer cell is obtained conservatively such that the flux computed by the coarse cell is consistent with that of fine cells. The fine and coarse grid fluxes are shown in Fig. 2(b) respectively with \( f^h \) and \( f^H \). To conservatively interpolate the ghost cells, these must satisfy

\[
f^H = \left( f^h_1^1 + f^h_2^2 \right)/2. \tag{17}\]

Thus, for a fine-grid ghost cell, for example \( g^h_{1,1} \) in Fig. 2(b), it is proved (Teunissen and Ebert, 2018) that any scheme satisfying the flowing constraint, will result in a conservative discretization:

\[
g^h_{1,1} + g^h_{1,2} = \frac{3}{4} (\phi^h_{1,1} + \phi^h_{1,2}) - \frac{1}{4} (\phi^h_{2,1} + \phi^h_{2,2}). \tag{18}\]

Analogous to the bilinear interpolation, the bilinear extrapolation meets the requirement in Eq. (18) and leads to the following scheme for \( g^h_{1,1} \):

\[
g^h_{1,1} = \frac{1}{2} \phi^h_{2,N} + \frac{9}{8} \phi^h_{1,1} - \frac{3}{8} \left( \phi^h_{1,2} + \phi^h_{2,1} + \frac{1}{8} \phi^h_{2,2} \right). \tag{19}\]

To fill the ghost cell for each cell-center variable, we have used the Eq. (19), although the Afivo library has provided several other schemes.

3.2.2. Prolongation and restriction
Aiming at transferring data between parent and children blocks, a specific interpolation is needed for collocated scalar and vector variables as well as the contravariant variables to make sure the flux consistency and mass conservation. A restriction operator transfers data from a children block (with finer grid resolution) to its parent (with coarser grid resolution), while a prolongation operator works in the reverse direction. In this study, we use the default averaging method delivered by the Afivo library for restriction of all variables. For prolongation, on the other hand, we use different strategies in different cases. Scalar variables are prolonged using a conservative interpolation, among several methods provided by the Afivo library, which has been elaborated in Teunissen and Ebert (2018). However, vector variables residing on cell-face or cell-center are prolonged using a divergence-free interpolation.

The configuration of a two dimensional problem is illustrated in Fig. 3. In this figure, all uppercase variables \( (U \) and \( V) \) are reserved for the cell-face variables, whereas the lowercase variables \( (u \) and \( v) \) are for the cell-center ones. The variables on the parent cell are indicated by superscript \( H \), while superscript \( h \) is used for children. To be more distinguishable, different indices are used for fine and coarse grids. The indices with a prime \( (i' \) and \( j') \) refer to the children blocks, and the indices without a prime \( (i \) and \( j) \) are associated with the parent block.

Using these notations, we describe the interpolation scheme tailored for the present study, where the grid size between consecutive refinement levels can only differ by a factor two. Let us define the discrete divergence, \( D \), at an arbitrary cell \((i,j)\) of the parent block as

\[
D_{ij} = \frac{U_{ij}^H - U_{i-1,j}^H}{\Delta^H x} + \frac{V_{ij}^H - V_{i,j-1}^H}{\Delta^H y}, \tag{20}\]

where \( U_{ij}^H \) and \( V_{ij}^H \) represent the contravariant components of the velocity on the parent block faces, while \( \Delta^H x \) and \( \Delta^H y \) are the cell widths in \( x \) and \( y \) directions, respectively. The divergence in Eq. (20) is deemed as the target divergence for the
calculated velocity in the children block. To facilitate future references, we define two subsets of all velocity components as follows.

- All contravariant components on the children faces that are located on the cell face of the parent cell too:
  \[ \mathcal{S}_1 = \left\{ U_{i,j}^{h}, U_{i,j+1}^{h}, U_{i-1,j}^{h}, U_{i-1,j+1}^{h}, V_{i,j}^{h}, V_{i,j-1}^{h}, V_{i-1,j}^{h}, V_{i-1,j-1}^{h}, V_{i,j+1}^{h}, V_{i,j}^{h} \right\} \]  
  \[ \text{(21)} \]

- All contravariant components of the coarse cell:
  \[ \mathcal{S}_2 = \left\{ U_{i,j+1}^{H}, U_{i,j-1}^{H}, U_{i+1,j}^{H}, U_{i-1,j}^{H}, U_{i,j}^{H} \right\}. \]
  \[ \text{(22)} \]

To obtain the velocity components of the children block, all components in \( \mathcal{S}_1 \) are conservatively interpolated. For example, on the right face of the parent grid, the velocity can be approximated as \( U(y) = a_0 + a_1 y + a_2 y^2 \). Using the mass-flux conservation, \( U_{i,j}^{H} = 0.5(U_{i,j}^{h} + U_{i,j+1}^{h}) \), and the known values, \( U_{i,j+1}^{H} \), \( U_{i,j}^{H} \) and \( U_{i,j-1}^{H} \), the following system of equations can be assembled (Vanella et al., 2010):

\[
\begin{bmatrix}
1 & \frac{\Delta y}{2} & \frac{\Delta y^2}{2}
\frac{3 \Delta y}{2} & \frac{(3 \Delta y)^2}{2}
\frac{5 \Delta y}{2} & \frac{(5 \Delta y)^2}{2}
\end{bmatrix}
\begin{bmatrix}
\alpha_0 \\
\alpha_1 \\
\alpha_2
\end{bmatrix}
=
\begin{bmatrix}
U_{i,j-1}^{H} \\
U_{i,j}^{H} \\
U_{i,j+1}^{H}
\end{bmatrix}.
\]
\[ \text{(23)} \]

Having obtained the values of coefficients \( \alpha_0 \), \( \alpha_1 \), and \( \alpha_1 \) from Eq. (23), one would be able to calculate the velocities at the midpoints of the cells. As an example, we can say \( U_{i,j}^{h} = a_0 + 5a_1 \Delta y^2/4 + a_2 (5 \Delta y^2/4)^2 \). Note that the system in Eq. (23) is a Vandermonde system that can be analytically solved. This procedure is repeated for all faces on the parent cell to obtain all velocity components in set \( \mathcal{S}_1 \).

For each coarse cell, a divergence-free enforced least square approximation of the velocity field is constructed by utilizing the already calculated velocity components in \( \mathcal{S}_1 \) and the cell-center velocity of the coarse cell \( (u_{ij}^{H}, v_{ij}^{H}) \). To this end, a set of polynomial basis functions are chosen for each component of the velocity, and a constraint least square approximation is obtained for the vector field by simultaneously solving the least square problem for both \( u \) and \( v \) components. In this study, the following basis functions and coefficients are utilized for each component:

\[
\begin{align*}
\mathbf{p}(x) &= \begin{bmatrix} 1, x, y, x^2, xy, y^2 \end{bmatrix}, \\
\mathbf{a}(x) &= \begin{bmatrix} a_0, a_1, a_2, a_3, a_4, a_5 \end{bmatrix}, \\
\mathbf{b}(x) &= \begin{bmatrix} b_0, b_1, b_2, b_3, b_4, b_5 \end{bmatrix},
\end{align*}
\]
\[ \text{(24a)} \]
\[ \text{(24b)} \]
\[ \text{(24c)} \]

where \( \mathbf{p}(x) \) is a set of monomial functions and the coefficients defined in Eq. (24b) and Eq. (24c) (i.e., \( \mathbf{a}(x) \) and \( \mathbf{b}(x) \)) are employed to reconstruct \( u \) and \( v \), respectively. Therefore, the velocity components can be written as

\[
\begin{align*}
\mathbf{u} &= a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2, \\
\mathbf{v} &= b_0 + b_1 x + b_2 y + b_3 x^2 + b_4 xy + b_5 y^2.
\end{align*}
\]
\[ \text{(25a)} \]
\[ v = b_0 + b_1 x + b_2 y + b_3 x^2 + b_4 xy + b_5 y^2. \]  

(25b)

Then, we modify the coefficients of the basis functions such that the reconstructed velocity guaranties the continuity by satisfying \( \partial u / \partial x = -\partial v / \partial y \). Besides, when obtaining the MLS approximation for internal components, the center of the axis is offset to the center of the coarse grid (where we set \( u_{ij}^H \) and \( v_{ij}^H \) in Fig. 3) which yields in \( a_0 = u_{ij}^H \) and \( b_0 = v_{ij}^H \). Consequently, the following polynomials representation of the velocity components are considered:

\[ u - u_{ij}^H = a_1 x + a_2 y + a_3 x^2 + a_4 y^2 + a_5 xy, \]  

(26a)

\[ v - v_{ij}^H = b_1 x - a_1 y + b_3 x^2 - \frac{a_5}{2} y^2 - 2a_3 xy. \]  

(26b)

By compiling both polynomial in Eq. (26b) with all elements in \( \Xi_1 \) as well as \( \Xi_2 \), one can obtain the following system of equations in the matrix form for the unknown coefficients (i.e. Eq. (24b) and Eq. (24c)):

\[ A \mathbf{c} = \mathbf{U}, \]  

(27)

where \( \mathbf{c} \) is the vector of the unknowns \( (a_1, \cdots, a_n, b_1, b_2) \), and \( \mathbf{U} \) is the vector of the velocity components on the face of parent block, either for the parent cell or the children cells. The matrix \( A \) is the coefficients matrix obtained from Eq. (26b). The system in Eq. (27) in an extended form reads

\[
\begin{bmatrix}
    x_1 & y_1 & x_1^2 & y_1^2 & x_1y_1 & 0 & 0 \\
    x_2 & y_2 & x_2^2 & y_2^2 & x_2y_2 & 0 & 0 \\
    \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
    x_n & y_n & x_n^2 & y_n^2 & x_ny_n & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
    a_1 \\
    a_2 \\
    a_3 \\
    a_4 \\
    a_5 \\
    a_6 \\
    b_1 \\
    b_2 \\
\end{bmatrix}
= 
\begin{bmatrix}
    U_1 \\
    U_2 \\
    \vdots \\
    U_n \\
\end{bmatrix},
\]

(28)

where \( x_1, \cdots, x_n \) and \( y_1, \cdots, y_n \) are the relative (with respect to the center of the coarse cell) \( x \) and \( y \) coordinates of the velocity components in the sets \( \Xi_1 \) and \( \Xi_2 \), and \( n \) is the number of elements in those sets; \( U_1 \) to \( U_n \) on the right hand side are all elements in \( \Xi_1 \cup \Xi_2 \) subtracted by \( u_{ij}^H \) or \( v_{ij}^H \), accordingly.

In two-dimensional problems, Eq. (27) is an overestimated system with twelve equations (eight of which for the velocities in \( \Xi_1 \), and four for those of in \( \Xi_2 \)) for eight unknowns \( (a_1, \cdots, a_n, b_1, b_2) \). Therefore, the vector \( \mathbf{c} \) is calculated by \( \mathbf{c} = (A^T A)^{-1} A^T \mathbf{U} \). The product \( (A^T A)^{-1} A^T \) is the left pseudo-inverse of the system, which is needed to be found merely once per each level of the nested grid.

4. Numerical results

In this section, we carry out a series of tests to demonstrate the accuracy and robustness of the present S-AMR method. Increasing the complexity of the cases, we first solve the Taylor–Green vortex problem, which has an exact solution, for detailed error analyses on the different components of the proposed method. Then, we simulate the uniform flow over a fixed circular cylinder producing the periodic Kármán vortex shedding to demonstrate that the solver can accurately and efficiently capture the flow phenomena. Subsequently, a predefined oscillatory motion of two cylinders passing each other is studied to show the capability of the proposed method for relatively complex flow with moving boundaries. Finally, we study the sedimentation of a circular cylinder in a fluid at rest, which reveals the ability of the present method for FSI problems.

4.1. Taylor–Green vortex

This problem defines an array of periodic counter-rotating vortices in a \( 2\pi \times 2\pi \) square domain that decay in time. The velocity components \( u, v \), and the pressure \( p \) for the exact solution are expressed as

\[ u = -e^{-2\pi t} \cos x \sin y, \]  

(29a)

\[ v = -e^{-2\pi t} \sin x \cos y, \]  

(29b)

\[ p = -\frac{e^{-2\pi t}}{4} (\cos 2x + \cos 2y). \]  

(29c)
Fig. 4 An example of the grid refinement used for the accuracy test and a sample snapshot \((t = 0.03)\) of the vorticity field of the Taylor–Green vortex problem. The white circle indicates a surface of the immersed body considered in the second test problem.

Fig. 5 The \(L_\infty\) error norms \(||\varepsilon_\omega||_\infty\) against the analytical solution as functions of the minimum grid spacing \(h\). (a) without the immersed object; (b) with the immersed object. Blue, linear interpolation in ghost layers between boxes; red, quadratic interpolation; black, uniform mesh results. The markers represent the examined resolutions.

Mixed homogeneous Dirichlet and Neumann velocity boundary conditions are imposed to be consistent with the exact solution. For the pressure Poisson equation (9), we impose the homogeneous Neumann boundary condition as we discussed in the previous section.

First, to separately analyze the effect of interpolation method used for the present S-AMR on the overall accuracy, two sets of different static meshes are considered. First, the entire domain is divided into four blocks containing uniform mesh. As exemplified in Fig. 4, the entire domain is subdivided into four boxes with uniform grid points; then, two blocks are refined for one level. The base uniform grid covering the entire domain is varied as 16\(^2\), 32\(^2\), 64\(^2\) and 128\(^2\) for calculating the error and its convergence. For all cases, the equations are integrated up to \(t = 0.03\) with \(\Delta t = 5 \times 10^{-3}\), and the error of the solution is obtained for two different ghost-layer boundary interpolations, i.e. the linear and the quadratic. This will show the effect of ghost-layer filling scheme on the overall accuracy of the algorithm. A sample snapshot of the vorticity field is also shown in Fig. 4. The same computation is carried out also on 32\(^2\), 64\(^2\), 128\(^2\) and 256\(^2\) uniform grids (i.e., without S-AMR) for comparison.

Figure 5(a) shows the convergence rates for two different ghost-layer interpolations, compared with that of the uniform grid. Note that \(h\) for S-AMR represents the minimum grid resolution. As expected, the convergence rate with the linear interpolation is about one, and that of the quadratic interpolation is about two. Although the accuracy with the
Fig. 6 The $L_{\infty}$ error norms $||\varepsilon_\omega||_\infty$ as functions of $h$ at $t = 0.03$ with different prolongation methods: blue, linear; red, quadratic non-divergence-free; black, divergence-free. (a) without the immersed object; (b) with the immersed object. The markers represent the examined resolutions.

Fig. 7 Time evolution of $||\varepsilon_\omega||_\infty$ under dynamic mesh refinement with different prolongation methods: blue, linear; red, quadratic non-divergence-free; black, quadratic divergence-free.

uniform grid is slightly higher than that with the S-AMR with quadratic interpolation, the convergence rate is similar to each other.

Subsequently, to investigate the accuracy of the present IBM coupled with the interpolation of S-AMR, the same problem is solved with assuming an immersed solid boundary inside the domain as depicted by a white circle in Fig. 4. The velocity boundary condition at the surface of the immersed object is obtained from the exact solution in Eq. (29). As can be seen in Fig. 5(b), the convergence rates are only marginally less than those without immersed boundary, which can be attributed to the splitting errors produced by the IBM.

In the two tests above, the grid is locally refined but does not evolve with time. Therefore, to evaluate the effects of the dynamic adaptation of the grid, we also utilize this problem with the same mesh configuration. However, for this test, refinement procedure is repeated as the solution is evolved in time. Starting from a single level grid, the domain is refined every 5 iterations by adding one refinement level, and then derefined every 5 iterations. That means, the refinement is performed at iterations 5, 15, 20, \ldots, and the derefinement at 10, 15, \ldots. Then, at time $t = 0.03$, the solution is compared with the exact one and the error norm is calculated. The accuracy of the solution with dynamics (de)refinement should highly depend on the prolongation of the solution in the newly generated children blocks on refinement and the restriction of the solution from the eliminated leaf-blocks to its parent on derefinement. Therefore, we examine three different prolongation methods, i.e., the linear prolongation, a quadratic prolongation without considering the divergence-free constraint, and the quadratic divergence-free prolongation discussed in Section 3.2. For all cases, a conservative interpolation (as described in Section 3.2.1) is used for the ghost-layer filling, and the base uniform grid with the resolution
of 128² is used.

Figure 6(b) shows the spatial accuracy of the solution with dynamic refinement using different prolongation schemes in the case without the immersed object. It is evident that the convergence rate is not much affected by the choice of prolongation operator. With all prolongation schemes, the second-order accuracy maintained, although the amount of errors is a function of the prolongation accuracy. On the other hand, the time evolution of the error, shown in Fig. 7, reveals an interesting feature of the prolongation scheme. The non-divergence-free prolongation operators result in a monotonically increasing error. However, with the divergence-free prolongation scheme, the error is kept constant on the order $O(10^{-4})$.

The same simulation with immersed boundary is conducted to investigate the effect of IBM along with the AMR strategy. Figure 6(b) demonstrates the accuracy of the present MLS-IBM with the dynamic S-AMR. The figure shows that the accuracy is slightly decreased with the IBM; however, the convergence rates are kept similar to the cases without the immersed boundary.

4.2. Uniform flow over a fixed cylinder

This test case addresses a major challenge in IBM's, that is the calculation of the force and capturing the correct boundary condition with immersed boundary. To accurately capture the boundary, in addition to the accuracy of IBM, the resolution of the grid plays an important role (Kor et al., 2018; Kor, 2018; Nishida and Tajiri, 2009). Although one can refine the grid around the object using mesh stretching or fixed mesh refinement, an AMR could serve an accurate and efficient solution by concentrating the grid where they are actually needed. This method let us have adequate resolution at the vicinity of the solid boundary and at the core of vortices in the wake, where the high gradients of the velocity should be captured with higher grid resolution.

The computation has been performed in a domain of $\Omega = [70D \times 40D]$, which is large enough to minimize the blockage effect (Kor et al., 2017). For the pressure, a homogeneous Neumann boundary condition, $\partial p / \partial n = 0$, is imposed on all the outer boundaries. A slip boundary condition is imposed for the velocity on the side boundaries. A uniform velocity, $u_\infty = 1$, is enforced at the inlet, while the convective boundary condition, i.e.,

$$\frac{\partial u}{\partial t} = -u_\infty \frac{\partial u}{\partial x},$$

is adopted at the outlet.

We study two cases at different Reynolds numbers, Re = 100 and 200. In both cases, the cylinder center is put at 10D downstream of the inlet and 20D apart from the upper boundary. The same strategy for grid resolution is adopted for both simulations. The entire computational domain is initially subdivided into 35 and 20 square boxes in the streamwise and spanwise directions, respectively. The side length of each box is 2, and each box contains $16^2$ uniform grids, so that $\Delta x/D = \Delta y/D = 0.125$. Then, the boxes containing the cylinder are four times refined and their neighbors are also refined appropriately to keep $1:2$ refinement criterion between neighbouring boxes. The mesh around the solid boundary is kept unchanged during the simulation (i.e., $\Delta x/D = \Delta y/D = 0.016$), but in the wake of the cylinder the grid resolution is dynamically adjusted (refined or de–refined) using the present S-AMR method. This dynamic adjustment is done every five time steps based on the vorticity amplitude, $|\omega|$, by checking whether it is larger or smaller than a predefined threshold ($|\omega| = 0.2$ was used in the present study).

Figure 8 shows the snapshot of the vorticity field for the case of Re = 100 in upper panel and Re = 200 in the lower panel, at $t = 1$ in the left column and $t = 130$ in the right column. The regular periodic vortex shedding is clearly observed in both cases. Moreover, in this figure, the hierarchy of grids around the cylinder and the core of vortices have been shown.

Figure 9 shows the pressure field for the case with Re = 100 and Re = 200 in upper and lower panel, respectively, at the same time step as the vorticity fields. In the zoom-up views on the right, the body of the cylinder is not shown purposely, to put emphasis on the fact that in both cases the pressure field inside the object varies smoothly due to the treatment of the domain inside the rigid body. It should be noted that in this figure the discontinuity in the pressure contours stems from the post-processing procedure for block-structured meshes.

Figure 10 presents the time evolution of the lift and drag coefficients. Both coefficients have a regular sinusoidal pattern upon the establishment of the periodical stable vortex shedding. This result indicates that the proposed method succeeded to capture the desired physical phenomena (i.e., the periodic vortex shedding) accurately.

Table 1 summarizes the result in these two cases and compares them with the literature. We here compare the Strouhal number, $St = f D / u_\infty$ (where $f$ is the frequency of the vortex shedding, $D$ is the diameter of the cylinder, and $u_\infty$...
Fig. 8  Contour of vorticity for the flow over a fixed cylinder at \( t = 1 \) in the left column and \( t = 130 \) in the right column with different Reynolds numbers: (a) \( \text{Re} = 100 \); (b) \( \text{Re} = 200 \). Grid lines are drawn only for grid blocks, i.e., every 16 grids.

Fig. 9  Contour of pressure for the flow over a fixed cylinder at \( t = 150 \) with different Reynolds numbers. Top row shows the result \( \text{Re} = 100 \) and bottom row is for \( \text{Re} = 200 \). (a) pressure field and adapted grid blocks in the entire domain, where grid lines are drawn only for grid blocks, i.e., every 16 grids; (b) zoom-up view of the pressure field including that computed inside and around the boundary of the cylinder.

is the free stream velocity), the fluctuations of the drag coefficient, \( C_d' \), and the lift coefficient, \( C_l' \), as well as the average of lift coefficient, \( C_l \). Calculated quantities are in good agreement with those of other numerical studies using body fitted or immersed boundary methods including our MLS-IBM without S-AMR (Badri Ghomizad et al., 2021), as well as the experimental results (Williamson, 1989).
The Reynolds number for this case is \( Re = \frac{\Omega y}{v} \) about their initial location for two periods, then move towards each other at \( \Omega = \frac{a}{D} \) rectangular box with lengths \( 4 \times 5 \times \) whereas it is \( 2 \times 4 \times \) in Fig. 11, two cylinders, \( S_1 \) and \( S_2 \), with the same diameter, \( D = 1 \), are initially put at \( (8D, 8D) \) and \( (24D, 9.5D) \). The \( x \)-component of time-dependent location of upper and lower cylinders, \( x_{r1} \) and \( x_{r2} \), are defined as

\[
x_{r1} = \begin{cases} 
\frac{4}{\pi} \sin \left( \frac{\pi}{8} \right) - 8, & 0 \leq t < 16, \\
-\frac{t}{24} & 16 \leq t \leq 32 
\end{cases}
\]

(31)

and

\[
x_{r2} = \begin{cases} 
8 - \frac{4}{\pi} \sin \left( \frac{\pi}{8} \right), & 0 \leq t < 16, \\
\frac{24 - t}{t} & 16 \leq t \leq 32 
\end{cases}
\]

(32)

while the \( y \)-components of them are kept the same as they move. Note that in this way, these two cylinders will oscillate about their initial location for two periods, then move towards each other at \( t = 16 \). The no-slip boundary condition for velocity and the homogeneous Neumann boundary condition for pressure is set at the domain boundaries (see Fig. 11). The Reynolds number for this case is \( Re = 40 \).

**Table 1** Comparison of the Strouhal number and force coefficients for the flow over the circular cylinder at \( Re = 100 \) and \( Re = 200 \). The results with * have been obtained from experimental study.

|            | \( C_d \) | \( C_f \) | \( C_l \) | \( St \) |
|------------|----------|----------|----------|--------|
| \( Re = 100 \) Williamson (1989) | 1.453 ±0.011 | ±0.339 | 0.164 |
| Uhlmann (2005) | 1.359 ±0.019 | ±0.293 | 0.16 |
| Braza et al. (1986) | 1.350 ±0.012 | ±0.339 | 0.165 |
| Liu et al. (1998) | 1.40 ±0.010 | ±0.32 | 0.165 |
| Mimeau et al. (2015) | 1.423 ±0.013 | ±0.34 | 0.171 |
| Xu and Wang (2006) | 1.389 ±0.011 | ±0.344 | 0.162 |
| Present S-AMR (\( \Omega = 70D \times 40D, h_{\min} = 0.016 \)) | 1.392 ±0.011 | ±0.344 | 0.160 |
| \( Re = 200 \) Williamson (1989) | - | - | - | 0.197 |
| Taira and Colonius (2007) | 1.35 ±0.048 | ±0.68 | 0.196 |
| Braza et al. (1986) | 1.386 ±0.040 | ±0.766 | 0.20 |
| Liu et al. (1998) | 1.31 ±0.049 | ±0.69 | 0.192 |
| Mimeau et al. (2015) | 1.44 ±0.05 | ±0.75 | 0.200 |
| Xu and Wang (2006) | 1.42 ±0.04 | ±0.66 | 0.202 |
| Present S-AMR (\( \Omega = 70D \times 40D, h_{\min} = 0.016 \)) | 1.368 ±0.044 | ±0.692 | 0.201 |
| Present S-AMR (\( \Omega = 70D \times 40D, h_{\min} = 0.016 \)) | 1.370 ±0.044 | ±0.692 | 0.198 |

**4.3. Two cylinders moving towards each other**

Besides vortex capturing inside the domain, accurate and efficient object representation would be fulfilled if the grid resolution over a moving solid boundary could adaptability be changed according to its path. Taken from Russel and Wang (2003), this example deals with simulation of fluid flows past multiple objects. Here, the computational domain is a rectangular box with lengths \( \Omega = [32D \times 16D] \), which is initially discretized with \( 16 \times 8 \) uniform coarse blocks. Each of this block in our simulation contains \( 8 \times 8 \) grid resolution. Based on the location of cylinders, blocks are refined up to five levels; therefore, at the vicinity of the solid boundary the grid resolution comes to the \( \Delta x/D = \Delta y/D = 7.8 \times 10^{-3} \) whereas it is \( 2.5 \times 10^{-3} \) for the region far from the boundaries. The time step is kept constant at \( \Delta t = 0.003 \). As shown in Fig. 11, two cylinders, \( S_1 \) and \( S_2 \), with the same diameter, \( D = 1 \), are initially put at \( (8D, 8D) \) and \( (24D, 9.5D) \). The \( x \)-component of time-dependent location of upper and lower cylinders, \( x_{r1} \) and \( x_{r2} \), are defined as

\[
x_{r1} = \begin{cases} 
\frac{4}{\pi} \sin \left( \frac{\pi}{8} \right) - 8, & 0 \leq t < 16, \\
-\frac{t}{24} & 16 \leq t \leq 32 
\end{cases}
\]

(31)

and

\[
x_{r2} = \begin{cases} 
8 - \frac{4}{\pi} \sin \left( \frac{\pi}{8} \right), & 0 \leq t < 16, \\
\frac{24 - t}{t} & 16 \leq t \leq 32 
\end{cases}
\]

(32)
Fig. 11  The schematic configuration for two cylinders moving towards each other.

$$\begin{align*}
\mathbf{u} &= 0, \quad \frac{\partial p}{\partial y} = 0 \\
D &= 6.5D \\
\mathbf{u}_{s1} &= \mathbf{u} = 0, \quad \frac{\partial p}{\partial x} = 0 \\
\mathbf{u}_{s2} &= \mathbf{u} = 0, \quad \frac{\partial p}{\partial y} = 0
\end{align*}$$

Fig. 12  The contours of variables for moving two cylinders towards each other at different time: (a) pressure contour; (b) vorticity contour; (c) vorticity contour with adapted grid blocks (grid lines are drawn only for grid blocks, i.e., every 16 grids).

Figure 12 shows the contours of the pressure, vorticity, and the adaptive mesh blocks in four different time instants, $t = 4, 16, 24,$ and $32$. The mesh is adapted based on the vorticity of the field. To this end, the refinement of the computational domain is adjusted according to the vorticity with a predefined threshold ($|\omega| = 0.2$). It can be seen that the solver is able to accurately follow both object locations as well as the wake of them.

Figure 13 compares the time evolution of lift and drag coefficients with the result of Xu and Wang (2006) obtained by immersed interface method. It is worth mentioning that Xu and Wang (2006) obtained the same result with much smaller time step, $\Delta t = 0.0005$ than ours. Our results are in good agreement with both Xu and Wang (2006) and those reported by Horng et al. (2018) who used an immersed boundary method with a solid fraction force calculation to obtain their results.

We observe that the drag on each cylinder increases as they approach each other, then drops to the minimum when the cylinders are in the closest distance. Then drag profiles return back to approximately the drag profile for a single cylinder. On the other hand, the objects are repulsive when they go towards each other and attractive after passing each
4.4. Sedimentation of a cylinder

The sedimentation of a cylinder is a canonical test case which needs considering a two-way interaction between fluid and structure. For convenience of comparison, the computational parameters are chosen as the same as those of Glowinski et al. (2001). Note that, in this example, the quantities are presented in dimensional forms following Glowinski et al. (2001). The diameter of cylinder is \( D_c = 0.25 \text{ cm} \), its density is \( \rho_c = 1.5 \text{ g/cm}^3 \), and its initial position is at \((1 \text{ cm}, 4 \text{ cm})\). The fluid with the density and dynamic viscosity \( \rho_f = 1.00 \text{ g/cm}^3 \) and \( \mu = 0.01 \text{ g/(cm} \cdot \text{s}) \) is initially assumed at rest. The computational domain is \( \Omega = [2 \text{ cm} \times 6 \text{ cm}] \).

In present study, a series of different mesh sizes are studied. The finest level resolution of these mesh are \( 1/18 \), \( 1/36 \) and \( 1/64 \) of the particle diameter. That means the finest level mesh \((h = \Delta x / D = \Delta y / D)\) would be \( h_1 = 0.139 \text{ cm} \), \( h_2 = 0.007 \text{ cm} \) and \( h_3 = 0.004 \text{ cm} \), respectively. In all of these cases, 150 Lagrangian points on the surface is used to represent the solid boundary. For the sake of quantitative comparison, the maximum Reynolds number is defined as

\[
\text{Re}_{\text{max}} = \max(\text{Re}(t)) = \max \left[ \frac{\rho_c \sqrt{u_c(t)^2 + v_c(t)^2} \cdot D_c}{\mu} \right]
\]

where \( u_c(t) \) and \( v_c(t) \) are the \( x \) and \( y \) components of the cylinder velocity.

Moreover, a collision model is applied in order to prevent the so-called particles from interpenetration to walls. Glowinski et al. (2001) proposed a short-range repulsive model which was improved later by Singh et al. (2003) and Wan and Turek (2007). In the original model given by Glowinski et al. (2001), the cylinder (particle) is not allowed to overlap the other cylinder or walls while the formulation of Singh et al. (2003) allows slight interpenetration. On the other hand, the work of Wan and Turek (2007) combines these two models and eventually does not allow particles to overlap. Here we use Wan and Turek’s (2007) model in order to simulate the collision of the cylinder with wall. For cylinder-wall collision the collision force read

\[
F^w = \begin{cases} 
\frac{1}{\epsilon} (X_i - X'_i) (2R_i - d'_i) & d_{i,j} < 2R_i \\
\frac{1}{\epsilon'} (X_i - X'_i) (2R_i + \xi - d'_i) & 2R_i \leq d_{i,j} < 2R_i + \xi \\
0 & 2R_i + \xi \leq d_{i,j}
\end{cases}
\]

where \( R_i \) is the radius of cylinder, \( X_i \) the center of it, \( X'_i \) the coordinate vector of the imaginary cylinder located on the physical boundary, and \( d_{i,j} = |X_i - X'_i| \). The parameter \( \xi \) is the range of the repulsive force, while the \( \epsilon \) and \( \epsilon' \) represent the small positive parameters for the collision.

Figure 14 shows some representative snapshots of the vorticity field and the adapted grid blocks. When the cylinder rebounds back, a pair of vortices is produced below the cylinder (Fig. 14(b)). Moving towards the top of the cylinder this
Fig. 14 The vorticity fields the collision process of a sedimenting cylinder with the wall. Note that the coordinates are dimensional (in cm) in this problem, and grid lines are drawn only for grid blocks, i.e., every 16 grids.

Table 2 Comparison of the maximum Reynolds number during the sedimentation of a cylinder ($\rho_s = 1.5 \text{ g/cm}^3$, $\nu = 0.01 \text{ cm}^2/\text{s}$, $\Delta t = 3.125 \times 10^{-5} \text{ s}$).

| Present | Wang et al. (2008) | Glowinski et al. (2001) | Wan and Turek (2007) | Uhlmann (2005) |
|---------|--------------------|-------------------------|----------------------|-----------------|
| $h$ (cm) | $1/72$ | $1/144$ | $1/256$ | $1/72$ | $1/144$ | $1/256$ | $1/72$ | $1/144$ | $1/256$ | $1/72$ | $1/144$ | $1/256$ | $1/72$ | $1/144$ | $1/256$ |
| $Re_{\text{max}}$ | 498.56 | 499.94 | 499.91 | 502.37 | 503.26 | 503.38 | 438.6 | 450.7 | 466 | 442.19 | 465.52 | 495 |

pair of vortices hit and combine with the vortices at the top side of the cylinder (Fig. 14(c)). The combined vortices move towards the wall side (Figs. 14(d) – (e)). Upon the next rebounding, the same procedure takes place but with less intense vortices (Fig. 14(f)). It is worth mentioning that here due to the symmetric scheme, relatively short domain, and not large enough Reynolds number, the pair vortex behind the cylinder does not shed to the domain. Besides those physical phenomena, the current algorithm can dynamically change the computational grids as can be seen in Fig. 14.

In this figure, the fine resolution are dynamically concentrated around the object and in the vicinity of walls where there are high gradient in velocity, where the vorticity is higher than a predefined threshold ($|\omega| = 0.2$).

Table 2 shows the comparison of the maximum Reynolds number $Re_{\text{max}}$. In our numerical simulation, $Re_{\text{max}}$ is 498.56, 499.94 and 499.91 for the finest meshes $h_1$, $h_2$ and $h_3$, respectively. The result are in reasonable agreement with result of Wan and Turek (2007) and Glowinski et al., (2001) utilizing a fictitious boundary method with finite element discretization, whereas they are in good agreement with Uhlmann (2005) and Wang et al. (2008) who used a class of immersed boundary method.

Because the initial position is symmetric, and the spatial discretization scheme keeps the symmetry, the perturbation grows slowly and a slight lateral change in position is observed before the collision. After collision, however, based on the stiffness parameter in Eq. (34), the rebouncing velocity varies, hence the perturbation in velocity field as shown in Fig. 15 for two different values of stiffness.

Figure 15(a) shows the evaluation of $x$ and $y$ locations of the cylinder with two stiffness parameters, $\epsilon_w = 2.5 \times 10^{-2}$ (cm$^2$/g) and $\epsilon_w = 2.5 \times 10^{-3}$ (cm$^2$/g) (with fixed $\epsilon_w' = 5 \times 10^{-2}$ (cm$^2$/g)). As we can see, the cylinder moves on the symmetric line before the collision, the cylinder tends to move toward the left wall of the domain after the collision due to the larger perturbation. When the cylinder falls on the bottom, it rebounds back for several times as can be seen in Fig. 15(b). It this figure, the $y$ component of the velocity clearly shows the rebouncing of the cylinder upon colliding the wall. Although this result is different from the numerical result of Glowinski et al. (2001), it is closer to the physical observation (Wang et al., 2008).
4.5. Computational time

Finally, we compare the computational time needed for solving the last example problem with the present S-AMR and that with a uniform grid to demonstrate the efficiency of the present method and the influential factor on it. In both approaches, first, the entire domain is covered by base boxes. In the present S-AMR, the boxes are distributed among threads, and a task parallelization strategy is utilized to solve the problem (Teunissen and Ebert, 2018); therefore, the grid resolution in each box has a considerable impact on the efficiency of the solver. To investigate the effect of the box grid resolution in the S-AMR approach, we consider two different cases. In Case A1, the domain is covered by $4 \times 12$ boxes and the grid resolution of each box is $16^2$. In Case A2, $8 \times 24$ boxes having $8^2$ grids are laid down on the computational domain. As a common practice in AMR approaches, wherever the solution needs finer mesh, boxes are dynamically refined up to four and five levels for Case 1 and Case 2, respectively; therefore, the finest resolution for both would be $\Delta x = \Delta y = 0.002$. In uniform grid approach, too, two different cases for the base boxes are considered. In Case U1, the entire domain is covered by a $128 \times 384$ boxes where each of them has $8^2$ grids. Case U2, $64 \times 96$ boxes with $16^2$ grids are laid down on the domain. Thus for both cases the resolution is $\Delta x = \Delta y = 0.002$.

The average computational time taken for each time step is shown in Fig. 16(a). As a task-parallel method is used here, the required computational time is slightly different between Case A1 and Case A2 (and also between Case U1 and Case U2). This difference is associated with the memory of the system and the grid size. However, the figure indicates that the present AMR solution is almost 20 times faster than its uniform counterpart at the same number of threads. Figure 16(b) shows the speed-up rate against the single thread computation. All cases show a linear increase in the speed-up rate up to 16 threads for this problem.
5. Summary and Conclusions

In this work, we proposed an adaptive mesh refinement (AMR) strategy suited to a moving least square (MLS) immersed boundary method (IBM) for numerical simulations of fluid-structure interactions (FSI) in viscous incompressible flows. A single block solver was employed on series of nested grids with different resolutions. Being a part of a tree data-structure covering the entire computation domain, each of these boxes has a structured Cartesian topology. Time integration was fulfilled by fractional step method. All spatial derivatives were approximated by a second-order central finite difference scheme on a cell center collocated configuration. While the Afivo library (Teunissen and Ebert, 2017; Teunissen and Ebert, 2018) is utilized to keep track of the grid hierarchy, we augmented it for vector field manipulation.

This AMR solver was coupled with a two-step MLS IBM to accurately and efficiently solve FSI problems. In the present IBM, we first obtain an approximation of the field in the first step to calculate the immersed boundary forces on the boundary and the whole domain, then in a correction step, those forces are imposed to the Navier-Stokes equation to take into account the presence of the boundary. According to our numerical demonstrations, the combination of these two approaches can considerably reduce the computational cost while keeping the accuracy high, particularly in terms of boundary condition accuracy and less spurious oscillations in force. Although here we just considered two-dimensional cases, the authors believe that for three-dimensional cases it will have more impact on the efficiency and it can be the future plan for this study.

The proposed method has a great potential to be used for FSI problems in high Reynolds number flows and flexible solid bodies, which is the subject of our future research.

References

Almgren, A. S., Bell, J. B., Colella, P., Howell, L. H., and Welcome, M. L., A conservative adaptive projection method for the variable density incompressible Navier-Stokes equations, Journal of Computational Physics, Vol. 142 (1998), pp. 1–46.
Angelidis, D., Chawdhary, S., and Sotiropoulos, F., Unstructured Cartesian refinement with sharp interface immersed boundary method for 3D unsteady incompressible flows, Journal of Computational Physics, Vol. 325 (2016), pp. 272–300.
Badri Ghomizad, M., Kor, H., and Fukagata, K., A sharp interface direct-forcing immersed boundary method using the moving least square approximation, Journal of Fluid Science and Technology, Vol. 16, No. 2 (2021), DOI:10.1299/jfst.2021jfst0013.
Bayyuk, S. A., Powell, K. G., and van Leer, B., A simulation technique for 2-D unsteady inviscid flows around arbitrarily moving and deforming bodies of arbitrary geometry, Proceedings of the 11th Computational Fluid Dynamics Conference (1993), Paper No. AIAA-93-3391-CP.
Berger, M. J. and Colella, P., Local adaptive mesh refinement for shock hydrodynamics, Journal of Computational Physics, Vol. 82 (1989), pp. 64–84.
Berger, M. J. and Oliger, J., Adaptive mesh refinement for hyperbolic partial differential equations, Journal of Computational Physics, Vol. 53 (1984), pp. 484–512.
Bhalla, A. P. S., Bale, R., Griffith, B. E., and Patankar, N. A., A unified mathematical framework and an adaptive numerical method for fluid–structure interaction with rigid, deforming, and elastic bodies, Journal of Computational Physics, Vol. 250 (2013), pp. 446–476.
Biancolini, M. E., Fast Radial Basis Functions for Engineering Applications, Springer (2017).
Bayona, V., Flyer N., and Fornberg, B., On the role of polynomials in RBF-FD approximations: III. Behavior near domain boundaries, Journal of Computational Physics, Vol. 380 (2019), pp. 378–399.
Brandt, A. and Livne, O. E., Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics, Revised Edition, Society for Industrial & Applied Mathematics (2011).
Braza, M., Chassaing, P. H., and Minh H. H., Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder, Journal of Fluid Mechanics, Vol. 165 (1986), pp. 79–130.
Chen, W., Fu, Z. J., and Chen, C. S., Recent Advances in Radial Basis Function Collocation Methods, Springer (2014).
Ding, H., Shu, C., Yeo, K. S., and Xu, D., Simulation of incompressible viscous flows past a circular cylinder by hybrid FD scheme and meshless least square-based finite difference method, Computer Methods in Applied Mechanics and Engineering, Vol. 193 (2004), pp. 727–744.

[DOI: 10.1299/jfst.2021jfst0014] © 2021 The Japan Society of Mechanical Engineers
De, A. K., A diffuse interface immersed boundary method for complex moving boundary problems, Journal of Computational Physics, Vol. 366 (2018), pp. 226–251.

Fadlun, E. A., Verzicco, R., Orlandi, P., and Mohd-Yusof, J., Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations, Journal of Computational Physics, Vol. 161 (2000), pp. 35–60.

Flyer, N., Fornberg, B., Bayona, V., and Barnett, G. A., On the role of polynomials in RBF-FD approximations: I. Interpolation and accuracy, Journal of Computational Physics, Vol. 321 (2016), pp. 21–38.

Gallier, S., Lemaire, E., Lobry, L., and Peters, F., A fictitious domain approach for the simulation of dense suspensions, Journal of Computational Physics, Vol. 256 (2014), pp. 367–387.

Griffith, B. E., Hornung, R. D., McQueen, D. M., and Peskin, C. S., An adaptive, formally second order accurate version of the immersed boundary method, Journal of Computational Physics, Vol. 223 (2007), pp. 10–49.

Glowinski, R., Pan, T. W., Hasla, T. I., Joseph, D. D., and Périaux, J., A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: Application to particulate flow, Journal of Computational Physics, Vol. 169 (2001), pp. 363–426.

Ham, F. E., Lien, F. S., and Strong, A. B., A Cartesian grid method with transient anisotropic adaptation, Journal of Computational Physics, Vol. 179 (2002), pp. 469–494.

Harris, R. E., Adaptive Cartesian immersed boundary method for simulation of flow over flexible geometries, AIAA Journal, Vol. 51 (2013), pp. 53–69.

Horng, T. L., Hsieh, P. W., Yang, S. Y., and You, C. S., A simple direct-forcing immersed boundary projection method with prediction-correction for fluid-solid interaction problems, Computers & Fluids, Vol. 176 (2018), pp. 135–152.

Ji, C., Munjiza, A., and Williams, J. J., A novel iterative direct-forcing immersed boundary method and its finite volume applications, Journal of Computational Physics, Vol. 231 (2012), pp. 1797–1821.

Kim, J. and Moin, P., Application of a fractional-step method to incompressible Navier-Stokes equations, Journal of Computational Physics, Vol. 59 (1985), pp. 308–323.

Kor, H., Development of an efficient immersed boundary method for simulation of flows around stationary and moving bodies. Doctoral Thesis, Keio University (2018).

Kor, H., Badri Ghomizad, M., and Fukagata, K., Extension of the unified interpolation stencil for immersed boundary method for moving boundary problems, Journal of Fluid Science and Technology, Vol. 13, No. 2 (2018), DOI:10.1299/jfst.2018jfst0008.

Kor, H., Badri Ghomizad, M., and Fukagata, K., A unified interpolation stencil for ghost-cell immersed boundary method for flow around complex geometries, Journal of Fluid Science and Technology, Vol. 12, No. 1 (2017), DOI:10.1299/jfst.2017jfst0011.

Liao, C. C., Chang, Y. W., Lin, C. A., and McDonough, J. M., Simulating flows with moving rigid boundary using immersed-boundary method, Computers & Fluids, Vol. 39 (2010), pp. 152–167.

Liu, C. and Hu, C., Block-based adaptive mesh refinement for fluid-structure interactions in incompressible flows, Computer Physics Communications, Vol. 232 (2018a), pp. 104–123.

Liu, C. and Hu, C., An adaptive multi-moment FVM approach for incompressible flows, Journal of Computational Physics, Vol. 359 (2018b), pp. 239–262.

Liu, C., Zheng, X., and Sung, C. H., Preconditioned multigrid methods for unsteady incompressible flows, Journal of Computational Physics, Vol. 139 (1998), pp. 35–57.

Martin, D. F. and Colella, P., A cell-centered adaptive projection method for the incompressible Euler equations Journal of Computational Physics, Vol. 163 (2000), pp. 271–312.

Mimeau, C., Gallizio, F., Cottet, G. H., and Mortazavi, I., Vortex penalization method for bluff body flows, International Journal for Numerical Methods in Fluids, Vol. 79 (2015), pp. 55–83.

Minion, M. L., A projection method for locally refined grids, Journal of Computational Physics, Vol. 127 (1996), pp. 158—178.

Nishida, H. and Tajiri, K., Numerical simulation of incompressible flows around a fish model at low Reynolds number using seamless virtual boundary method, Journal of Fluid Science and Technology, Vol. 4, No. 3 (2009), DOI:10.1299/jfst.4.500.

Nishikawa, K., Konno, K., and Hattori, Y., Assessment of immersed boundary method as a tool for direct numerical simulation of aeroacoustic sound, Journal of Fluid Science and Technology, Vol. 15, No. 1 (2020), DOI:10.1299/jfst.2020jfst0004.

Peskin, C. S., Flow patterns around heart valves: A numerical method, Journal of Computational Physics, Vol. 10 (1972),
pp. 252–271.

Popinet, S., Gerris: A tree-based adaptive solver for the incompressible Euler equations in complex geometries, Journal of Computational Physics. Vol. 190 (2003), pp. 572–600.

Rencis, J. J. and Mullen, R. L., Solution of elasticity problems by a self adaptive mesh refinement technique for boundary element computation, International Journal for Numerical Methods in Engineering. Vol. 23 (1986), pp. 1509–1527.

Roma, A. M., Peskin, C. S., and Berger, M. J., An adaptive version of the immersed boundary method, Journal of Computational Physics, Vol. 153 (1999), pp. 509–534.

Russell, D. and Wang, Z. J., A cartesian grid method for modeling multiple moving objects in 2D incompressible viscous flow, Journal of Computational Physics, Vol. 191 (2003), pp. 177–205.

Singh, P., Hesla, T. I., and Joseph, D. D., Distributed Lagrange multiplier method for particulate flows with collisions, International Journal of Multiphase Flow, Vol. 29 (2003), pp. 495–509.

Soltani, E. and Badri Ghomizad, M., Immersed boundary method for the simulation of heat transfer and fluid flow based on vorticity-velocity formulation, Numerical Heat Transfer, Part B: Fundamentals, Vol. 70 (2016), pp. 25–46.

Teunissen, J. and Ebert, U., Simulating streamer discharges in 3D with the parallel adaptive Afivo framework, Journal of Physics D: Applied Physics, Vol. 50 (2017), 474001.

Taira, K. and Colonius, T., The immersed boundary method: A projection approach, Journal of Computational Physics, Vol. 225 (2007), pp. 2118–2137.

Tajiri, K., Nishida, H., and Tanaka, M., Property of seamless immersed boundary method for large eddy simulation of incompressible turbulent flows, Journal of Fluid Science and Technology, Vol. 9, No. 2 (2014), DOI:10.1299/jfst.2014jfst0020.

Uhlmann, M., An immersed boundary method with direct forcing for the simulation of particulate flows, Journal of Computational Physics, Vol. 209 (2005), pp. 448–476.

Vanella, M., Rabenold, P., and Balaras, E., A direct-forcing embedded-boundary method with adaptive mesh refinement for fluid-structure interaction problems, Journal of Computational Physics, Vol. 229 (2010), pp. 6427–6449.

Wan, D. and Turek, S., An efficient multigrid-FEM method for the simulation of solid-liquid two phase flows, Journal of Computational and Applied Mathematics, Vol. 203 (2007), pp. 561–580.

Wang, X. Y, Yu, P., Yeo, K. S., and Khoo, B. C., SVD–GFD scheme to simulate complex moving body problems in 3D space, Journal of Computational Physics, Vol. 229 (2010), pp. 2314–2338.

Wang, Z., Fan, J., and Luo, K., Combined multi-direct forcing and immersed boundary method for simulating flows with moving particles, International Journal of Multiphase Flow, Vol. 34 (2008) pp. 283–302.

Williamson C. H. K., Oblique and parallel modes of vortex shedding in the wake of a circular cylinder at low Reynolds numbers, Journal of Fluid Mechanics, Vol. 206 (2006), pp. 579–627.

Xu, S. and Wang, Z. J., An immersed interface method for simulating the interaction of a fluid with moving boundaries, Journal of Computational Physics, Vol. 216 (2006), pp. 454–93.

Yao, H., Nambu, T., and Mizobuchi, Y., An immersed boundary method for practical simulations of high-Reynolds number flows by $k$-$\varepsilon$ RANS models, Journal of Fluid Science and Technology, Vol. 16, No. 1 (2021), DOI:10.1299/jfst.2021jfst0007.