Computers and Fluids 238 (2022) 105358

Contents lists available at ScienceDirect

Computers and Fluids

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

A subcycling/non-subcycling time advancement scheme-based DLM immersed boundary method framework for solving single and multiphase fluid–structure interaction problems on dynamically adaptive grids

Yadong Zeng [a,c], Amneet Pal Singh Bhalla [b], Lian Shen [a,c,*]

[a] Department of Mechanical Engineering, University of Minnesota, Minneapolis, MN, 55455, USA
[b] Department of Mechanical Engineering, San Diego State University, San Diego, CA 92182, USA
[c] St. Anthony Falls Laboratory, University of Minnesota, Minneapolis, MN, 55414, USA

ARTICLE INFO

Keywords:
Adaptive mesh refinement (AMR)
Distributed Lagrange multiplier (DLM)
Multiphase flows
Subcycling
Non-subcycling
AMReX

ABSTRACT

In this paper, we present an adaptive implementation of the distributed Lagrange multiplier (DLM) immersed boundary (IB) method on multilevel collocated grids for solving single- and multiphase fluid–structure interaction (FSI) problems. Both a non-subcycling time advancement scheme and a subcycling time advancement scheme, which are applied to time-march the composite grid variables on a level-by-level basis, are presented; these schemes use the same time step size and a different time step size, respectively, on different levels. This is in contrast to the existing adaptive versions of the IB method in the literature, in which coarse- and fine-level variables are simultaneously solved and advanced in a coupled fashion. A force-averaging technique and a series of synchronization operations are constructed to achieve excellent momentum and mass conservation across multiple levels of grid hierarchy. We demonstrate the versatility of the present multilevel framework by simulating problems with various types of kinematic constraints imposed by structures on fluids, such as imposing a prescribed motion, free motion, and time-evolving shape of a solid body. The DLM method is also coupled to a robust level set method-based two-phase fluid solver to simulate challenging multiphase flow problems, including wave energy harvesting using a mechanical oscillator. The capabilities and robustness of the computational framework are validated against a variety of benchmarking single-phase and multiphase FSI problems from the literature, which include a three-dimensional swimming eel model to demonstrate the significant speedup and efficiency that result from employing the present multilevel subcycling FSI scheme.

1. Introduction

Many engineering fields address problems that involve complex interactions between fluids and solids. Examples include biomedical engineers who address heart valves [1,2] and drug particles [3], control and mechatronic engineers who address swimming fish [4,5], flying drones [6], and tiny insects [7,8], marine engineers who address offshore platforms [9] and risers [10,11], and mechanical and energy engineers who address membrane distillation [12], wind turbines [13] and wave energy converters (WECs) [14,15]. In the past several decades, many researchers have investigated such problems through numerical simulations and experiments, which provide detailed descriptions of the flow field and help reveal the underlying mechanisms involved in the fluid–structure interaction (FSI). Usually, the fluid–solid interfaces in FSI simulations exist within a small spatial region of the computational domain, and high spatial resolution is required only in the vicinity of fluid–solid interfaces to adequately resolve the flow boundary layers. However, one does not need or perhaps cannot afford a uniformly high grid resolution across an entire domain, even if the simulation is executed on distributed memory systems. The need to locally resolve thin fluid–structure interfaces and boundary layers can be addressed by the dynamic adaptive mesh refinement (AMR) technique. In brief, AMR increases the grid resolution in regions of interest as needed during a simulation while providing a reasonable estimate of the flow field in other regions of the domain. Since AMR was proposed in the 1980s [16,17], various classes of AMR methods have been developed and applied to a wide range of fluid mechanics problems, including marine ice sheets [18], surfactant-driven flows [19], wave energy harvesting [14], deforming flags [20,21], swimming eels [4], and stratified oceanic flows [22].

AMR methods can be broadly classified into two categories based on the grid hierarchy and data structures: the quadtree/octree-based AMR

* Corresponding author at: Department of Mechanical Engineering, University of Minnesota, Minneapolis, MN, 55455, USA.
E-mail address: shen@umn.edu (L. Shen).

https://doi.org/10.1016/j.compfluid.2022.105358
Received 21 December 2021; Received in revised form 11 February 2022; Accepted 25 February 2022
Available online 9 March 2022

(TBAMR) technique [23–25] and the block-structured AMR (BSAMR) technique [16,17,26–29]. In TBAMR, each grid cell can be split into four cells in two spatial dimensions or eight cells in three spatial dimensions, and a tree structure [30] is utilized to arrange the hierarchy of grid cells. Although the tree-based structure is an intuitive representation of the multilevel grid hierarchy and simplifies grid refining and coarsening operations, the data structures [30,31] employed in TBAMR are not easy to implement. Moreover, the cell connectivity and the refinement/coarsening history need to be stored at every time step of the simulation [32,33]. In contrast, BSAMR builds the mesh as nested Cartesian patches [16,17,26–29,34,35]. Each grid level includes several rectangular patches that are grouped, and the resulting system of equations is efficiently solved utilizing a multigrid (MG) solver [26,29]. This approach makes domain decomposition of the BSAMR data structures relatively easy [36,37]. More specifically, in the context of immersed boundary (IB) methods, where an immersed body is represented by a Lagrangian mesh, BSAMR allows the moving structure to remain embedded on the finest grid level during a simulation in a natural way. Furthermore, IB velocity interpolation and force spreading routines can be efficiently implemented using the BSAMR framework.

Broadly speaking, FSI problems can be simulated using interface conforming arbitrary Lagrangian–Eulerian (ALE) methods or interface nonconforming fictitious domain/immersed boundary methods. Although an ALE-like method sharply resolves the fluid boundary layer and imposes exact boundary conditions on the fluid–structure [38] or the fluid–fluid [39] interface, for a solid structure with complex geometry and large displacements or deformations, the ALE method poses several numerical and implementation challenges because the computational domain needs to be frequently remeshed to adhere to the evolving structure or the interface [40,41]. In recent years, the IB method has been widely employed for simulating complex FSI problems that involve large deformations of a structure or topological changes of the fluid–fluid interface due to the motion of the structure. In contrast to an ALE approach, in the IB method, the background (Cartesian) fluid grid does not deform due to a moving structure. Instead, the moving structure is accounted for through IB forces that are applied near the structure surface to satisfy the velocity matching boundary condition on the fluid–solid interface [42]. Depending on the forcing technique, IB methods can be further classified as a diffuse interface or a sharp interface IB method [40,43]. In the diffuse-interface IB method, the surface force calculated on the (Lagrangian) immersed boundary is distributed to several adjacent fluid grid nodes by a regularized integral kernel [44–48]. One of the most efficient ways to compute the rigid body IB force is through the distributed Lagrange multiplier (DLM) technique of Patankar et al. [49]. The sharp-interface IB method, on the other hand, directly imposes the velocity of the moving solid boundary by fitting a spatial polynomial through the solid interface and nearby fluid nodes [50–57]. In addition to the IB method, the cut-cell method [50,58,59] and the Brinkman penalization method [60] have also been proposed to simulate complex FSI problems.

To reduce the execution times of FSI simulations, many researchers have applied AMR frameworks to implement some of the aforementioned FSI schemes. Popinet [25] implemented the volume of fluid (VOF) method on a TBAMR framework for simulating incompressible Euler flows (no flow penetration through the solid boundary, but tangential slip is present) in complex stationary domains. The solid boundaries in [25] are represented by a Eulerian level set (LS) function that is embedded on the finest grid level. Guittet et al. [23] developed a LS-based method for the incompressible Navier–Stokes equations adaptive quad/octrees grids, which can address arbitrary refinement/coarsening ratios between adjacent cells. Roma et al. [61] implemented the diffuse-interface IB method on a multilevel staggered grid within the BSAMR framework. However, this method is only suitable for low Reynolds numbers (Re = 10–100) because of the explicit treatment of the nonlinear convective term with a nonconservative central differencing scheme and the lack of limiters. A similar formulation was later implemented by Griffith et al. [62] on both multilevel collocated grids and staggered grids. Vanella and Balaras [63] and Vanella et al. [35] implemented a direct-forcing immersed boundary method within the BSAMR framework for laminar and turbulent flows, in which the governing equations are integrated simultaneously and iteratively using a strong coupling scheme. Bhalla et al. [4] and Nangia et al. [64] combined the DLM method with BSAMR to capture thin boundary layers and vortical structures present in high-density-ratio, wave–structure interaction (WSI) problems. An advantage of the DLM/IB method is that fluid–structure coupling is implicit, which implies that there is no need to explicitly evaluate hydrodynamic stresses on the solid surface or to iterate between fluid domain integrators and solid domain integrators. In this study, we combine the DLM method with a collocated BSAMR framework for simulating both single and multiphase FSI problems.

The choice of the grid layout affects the complexity of the discretization and multilevel algorithms for resolving the FSI [62]. In the works cited in the preceding paragraph, most of the authors considered a staggered Mark And Center (MAC) grid layout [4,23,25,35,61,63,64], in which different interpolation schemes for velocity and pressure fields are required [34,35,65–67]. Compared to the staggered grid, the use of a collocated grid layout simplifies the implementation of multilevel differential and interpolation operators.

Because the coarse and fine grid variables are coupled to each other through the coarse–fine boundaries in both staggered and collocated grid layouts, discrete equations are typically solved on the entire grid hierarchy using a composite grid time marching scheme. Consequently, the time step size of the simulation is restricted by the finest grid spacing to preserve numerical stability. This restriction is especially valid for IB time integrators in the literature [4,23,25,35,61,63,64]. On the other hand, the level-by-level time integration method decouples the time advancement process among different levels [17,26]; this technique has been successfully employed to solve hyperbolic flow problems in the literature but has not been implemented in the context of an IB method. The level-by-level time integration method can be further divided into two types: non-subcycling methods and subcycling methods [27,29]. The non-subcycling method uses a uniform time step size for all levels. Thus, the time step size is also restricted by the finest level for numerical stability reasons. The subcycling method, in which grid variables on different levels are advanced with different time step sizes, can reduce the number of advancement steps and total execution time. Note that the non-subcycling method is different from the typical composite grid time marching method, despite the commonality of employing the same time step size on different levels. In the non-subcycling method, a coarse level provides the required (Dirichlet) boundary conditions to time advance the variables on the next (nested) finer level. To our knowledge, there has been no implementation of an adaptive version of the IB method that utilizes the level-by-level time advancement method, especially the subcycling method. Considering that vortical structures and thin boundary layers are often resolved only on the finest grid level, time advancement using the subcycling method is desired to reduce the FSI computational cost. Furthermore, the proposed method is implemented using the *AMReX* package [68], which already provides an efficient implementation of multilevel grid operations (data coarsening/refining/synchronization, hierarchy regridding and associated data transfer, etc.) and linear solvers.

There are four main contributions of this paper. First, we develop a level-by-level method for time advancing IB equations on collocated grids. To our knowledge, this method is the first framework that utilizes subcycling and non-subcycling methods to solve FSI problems on adaptive meshes. Second, we propose a force-averaging technique and a series of synchronization operations to achieve excellent momentum and mass conservation across multiple levels. Third, we couple the DLM method with a robust LS method-based two-phase flow solver for efficiently simulating multiphase FSI problems. Last, we incorporate several types of solid constraints into this unified framework.

We would like to point out that the present paper is substantially different from our previous work [69]. The latter simulated two-fluid flows only, without considering the presence of the solid structures. The present paper focuses on the FSI problems with different types of kinematic constraints of the solid structures and their coupling with the fluid motions. For the common numerical algorithms shared by the two papers, we minimize the duplication by referring to [69] while only providing a concise description of the numerical features in a self-contained way in the present paper.

The remainder of the paper is organized as follows: we start with the mathematical formulation of a coupled fluid–structure system in Section 2, followed by the design concept of a multilevel adaptive grid in Section 3; the spatial discretization of Eulerian and Lagrangian variables are then described in Section 4; next, the time advancement algorithm is presented in Section 5; validation cases and numerical tests are shown in Section 6; and finally, conclusions are summarized in Section 7.

2. Mathematical formulation

In this section, we describe the governing equations for a fluid–structure system occupying a multilevel Cartesian grid $\Omega \subset \mathbb{R}^d$, where $d = 2, 3$ denotes the spatial dimension. The left part of Fig. 1 shows a schematic of two solid bodies on a two-dimensional multilevel Cartesian grid. The momentum and material incompressibility equations are described using a fixed Eulerian coordinate system $\mathbf{x} = (x_1, \ldots, x_d) \in \Omega$. The immersed body is described using a Lagrangian coordinate system, where $\mathbf{s} = (s_1, \ldots, s_d) \in \Omega_c$ denotes the fixed material coordinate system attached to the structure and $\Omega_c \subset \mathbb{R}^d$ is the Lagrangian curvilinear coordinate domain. The position of the structure at time t is $\mathbf{X}(\mathbf{s}, t)$; it occupies a volumetric region $V_b(t) \subset \Omega$. The equations of motion of the coupled fluid–structure system are

$$\rho(\mathbf{x},t)\left(\frac{\partial \mathbf{u}}{\partial t}(\mathbf{x},t) + \nabla\cdot(\mathbf{u}(\mathbf{x},t)\mathbf{u}(\mathbf{x},t))\right) = -\nabla p(\mathbf{x},t) + \nabla\cdot\left[\mu(\mathbf{x},t)\left(\nabla\mathbf{u}(\mathbf{x},t) + \nabla\mathbf{u}(\mathbf{x},t)^T\right)\right] + \wp(\mathbf{x},t)\mathbf{g} + \mathbf{f}_c(\mathbf{x},t), \quad (1)$$

$$\nabla\cdot\mathbf{u}(\mathbf{x},t) = 0, \quad (2)$$

$$\mathbf{f}_c(\mathbf{x},t) = \int_{V_b(t)} \mathbf{F}_c(\mathbf{s},t)\delta(\mathbf{x} - \mathbf{X}(\mathbf{s},t))\mathrm{d}\mathbf{s}, \quad (3)$$

$$\frac{\partial \mathbf{X}}{\partial t}(\mathbf{s},t) = \mathbf{U}(\mathbf{s},t), \quad (4)$$

$$\mathbf{U}(\mathbf{s},t) = \int_{V_b(t)} \mathbf{u}(\mathbf{x},t)\delta(\mathbf{x} - \mathbf{X}(\mathbf{s},t))\mathrm{d}\mathbf{x}. \quad (5)$$

Here, $\mathbf{u}(\mathbf{x},t)$ is the Eulerian velocity of the coupled fluid–structure system, $p(\mathbf{x},t)$ is the pressure, $\rho(\mathbf{x},t)$ is the Eulerian density field (fluid density in the fluid region $\Omega_f = \Omega \setminus \Omega_c$ and solid density in the solid region Ω_c), $\mu(\mathbf{x},t)$ is the dynamic viscosity of the fluid–structure system, and $\wp(\mathbf{x},t)$ is the modified density, which will be detailed in Section 5.1. Because $\rho(\mathbf{x},t)$, $\wp(\mathbf{x},t)$, and $\mu(\mathbf{x},t)$ are allowed to change spatially and temporally in this work, the solid structure can be heavier (or lighter) and more viscous (or less viscous) than the surrounding fluid. The gravitational acceleration is written as $\mathbf{g} = (g_1, \ldots, g_d)$. In Eq. (1), $\mathbf{f}_c(\mathbf{x},t)$ represents the Eulerian force density, which accounts for the presence of the solid in the domain. $\delta(\mathbf{x}) = \Pi_{i=1}^d \delta(x_i)$ represents the d-dimensional Dirac delta function, which is employed to facilitate the information exchange between the Eulerian quantity and Lagrangian quantity. Specifically, Eq. (3) converts the Lagrangian force density $\mathbf{F}_c(\mathbf{s},t)$ to an equivalent Eulerian force density $\mathbf{f}_c(\mathbf{x},t)$, in an operation that is referred to as *force spreading*. Eq. (5) maps the Eulerian velocity $\mathbf{u}(\mathbf{x},t)$ to the Lagrangian marker velocity $\mathbf{U}(\mathbf{s},t)$, in an operation that is referred to as *velocity interpolation*. Constrained by the no-slip boundary condition at the fluid–solid interface, the velocity of a Lagrangian marker $\mathbf{U}(\mathbf{s},t)$ follows the local fluid velocity. For notational convenience, we denote the force spreading operation in Eq. (3) as

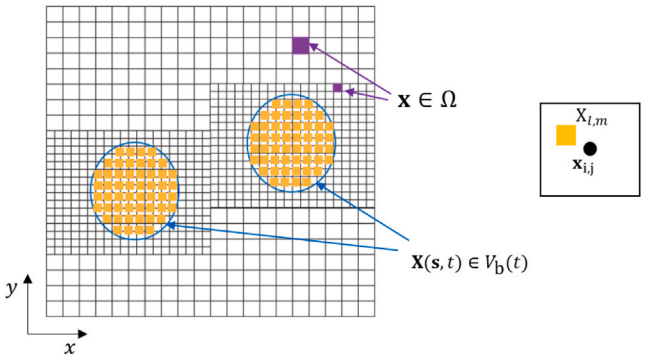

Fig. 1. Left: two solid bodies (confined by blue lines) on a multilevel Cartesian grid. The Eulerian grid cells (■, purple) discretize the Ω region, and the Lagrangian markers (■, orange) discretize the $V_b(t)$ region. Right: schematic representation of a single Cartesian grid cell. The Eulerian quantities are defined at the cell center (●, black); the Lagrangian quantities are defined on the marker points (■, orange), which are free to move on the Eulerian grid.

$$\mathbf{f}_c = \mathcal{S}[\mathbf{X}]\mathbf{F}, \quad (6)$$

where $\mathcal{S}[\mathbf{X}]$ is the force spreading operator. Similarly, the velocity interpolation operation in Eq. (5) is written in shorthand notation as

$$\mathbf{U} = \mathcal{J}[\mathbf{X}]\mathbf{u}, \quad (7)$$

where $\mathcal{J}[\mathbf{X}]$ is the velocity interpolation operator. As shown in [45,64], the Lagrangian–Eulerian coupling operators conserve energy as long as $\mathcal{S}$ and $\mathcal{J}$ are adjoint.

3. Adaptive grid design concept

This section describes the definitions of the variables and operators for a multilevel adaptive grid. The coarsest level, enumerated level 0, occupies the entire computational domain Ω, and level l_{max} is the finest level in the grid hierarchy. Therefore, the total number of levels in the grid hierarchy is $l_{max} + 1$. Grid cells at coarser levels are dynamically tagged, grouped to form a series of rectangular patches, and then refined to finer levels following a nesting criterion [17]. There can be more than one patch on a specific level, and patches from the same level may be mapped to different processors in parallel computing using the domain decomposition technique. The solid structure is always enclosed by the finest level patches. As an example, Fig. 1 illustrates a two-level adaptive grid with $l_{max} = 1$; level 1 has two patches, and the two immersed structures are entirely enclosed by the finest level patches.

We use Ω^l to represent the union of patches on level l. Because BSAMR uses a nested hierarchy of rectangular patches, the union of patches on level $l+1$ must be contained within the union of patches on level l for all $0 \le l < l_{max}$, i.e., $\overline{\Omega}^{l+1} \subset (\Omega^l)^\circ$. Here, the overline indicates the closed set of level $l+1$ and the superscript '$\circ$' represents the interior regions of the patches on level l. Three types of boundaries are involved in an adaptive grid framework: the physical boundary, coarse–fine (CF) boundary, and fine–fine (FF) boundary. A physical boundary refers to the boundary that delineates the entire computational domain. The CF boundary represents the boundary between the grid cells on the finer level $l+1$ and those on the coarser level l for all $0 \le l < l_{max}$. The FF boundary represents the boundary between two patches on the same level. All three boundaries are illustrated using different line styles in Fig. 2. The discrete operators described in Section 4 require filling ghost cells near the aforementioned boundaries; techniques to fill ghost cells at the three types of BSAMR boundaries have been discussed in detail in the literature [26,68,70], and the same operations are followed in this work.

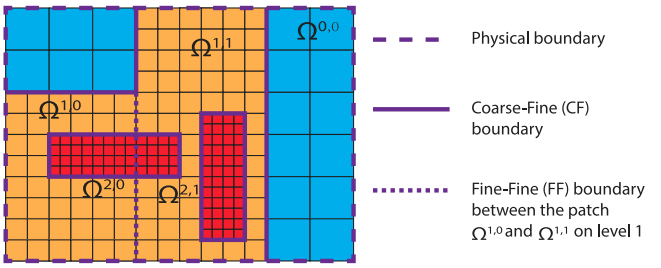

Fig. 2. Schematic of a three-level adaptive grid with three types of boundaries: the physical boundary, coarse–fine boundary, and fine–fine boundary. $\Omega^{i,j}$ represents patch j on level i for all $i \geq 0, j \geq 0$.

4. Spatial discretization on a multilevel adaptive grid

This section describes the spatial discretization of Eulerian and Lagrangian quantities and their coupling on a multilevel adaptive grid. The discretization of two-dimensional spatial operators is presented in this section. The discrete version of the three-dimensional spatial operators can be defined analogously.

4.1. Eulerian discretization

All Eulerian variables, including the fluid velocity $\mathbf{u}$, pressure p, and LS functions ϕ and ψ, are defined at cell centers following the collocated grid variable arrangement setup. Here, the LS functions ϕ and ψ are employed to represent the gas–liquid interface and fluid–solid interface, respectively [64]. On a particular level l, the integer index set (i, j) is used to label the Cartesian grid cells. The position of the cell center is obtained from its index as $\mathbf{x}^l_{i,j} = (i\Delta x^l, j\Delta y^l)$, where Δx^l and Δy^l represent the grid spacings in the x direction and y direction on level l, respectively. Similarly, the discrete velocity $\mathbf{u}$, pressure p, and Eulerian force density $\mathbf{f}_c$ on level l are denoted as $\mathbf{u}(\mathbf{x}^l_{i,j}, t)$, $p(\mathbf{x}^l_{i,j}, t)$, and $\mathbf{f}_c(\mathbf{x}^l_{i,j}, t)$, respectively.

For a multilevel adaptive grid, the union of patches Ω^l on level l can be partitioned into a valid region Ω^l_{valid} that is not covered by finer grid patches and an invalid region $\Omega^l_{\text{invalid}}$ that is covered by finer grid patches. We note that on level $l_{\max}$, $\Omega^{l_{\max}}_{\text{valid}} = \Omega^{l_{\max}}$ and $\Omega^{l_{\max}}_{\text{invalid}} = \varnothing$. On level $l < l_{\max}$, $\Omega^l_{\text{valid}} = \Omega^l \setminus \Omega^{l+1}$ and $\Omega^l_{\text{invalid}} = \Omega^{l+1}$. Fig. 3 shows an example grid, where green grid cells represent the valid region of level l, and orange grid cells represent the valid region of $l+1$ (which is also the invalid region of level l). Because a level-by-level time advancement method (Section 5) is applied in this work, Eulerian variables need to be available in both valid regions and invalid regions for the time advancement process. This method is different from the composite grid time advancement method employed in [4,64]. Composite variables that are defined only in the valid regions across the grid hierarchy can also be employed. All simulation visualizations in this work are presented using the composite variables unless specified otherwise. On the other hand, level variables are defined on an entire level, including in its invalid region. Correspondingly, there are two types of operators in an adaptive grid algorithm: composite operators that are defined only in the valid regions across the multiple levels and level operators that are defined for an entire level.

In Fig. 3, we use a two-dimensional pressure field p to define the aforementioned composite and level operators. For a specified level l, the composite Laplacian operator is denoted as $L^{cc,\text{comp},l}$, where the superscripts cc, comp, and l denote the cell center, composite operator, and level l, respectively. For all $p_{i,j} \subset \Omega^l_{\text{valid}}$ on level l, the two-dimensional composite Laplacian operator $L^{cc,\text{comp},l}$ is defined as

$$L^{cc,\text{comp},l}\, p\Big|_{i,j} := \frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{(\Delta x^l)^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{(\Delta y^l)^2}. \tag{8}$$

The ghost cell $p_{i+1,j} \subset \Omega^l_{\text{invalid}}$ is obtained from the valid p data on level $l+1$ as $p_{i+1,j} = (p_{2i+2,2j} + p_{2i+3,2j} + p_{2i+2,2j+1} + p_{2i+3,2j+1})/4$. Likewise, for all $p_{2i+2,2j} \subset \Omega^{l+1}_{\text{valid}}$ on level $l+1$, the composite Laplacian operator $L^{cc,\text{comp},l+1}$ is defined as

$$L^{cc,\text{comp},l}\, p\Big|_{2i+2,2j} := \frac{p_{2i+3,2j} - 2p_{2i+2,2j} + p_{2i+1,2j}}{(\Delta x^{l+1})^2} + \frac{p_{2i+2,2j+1} - 2p_{2i+2,2j} + p_{2i+2,2j-1}}{(\Delta y^{l+1})^2}. \tag{9}$$

In this equation, if the ghost cell $p_{2i+1,2j}$ belongs to the CF boundary, as shown in Fig. 3, then it is filled using conservative interpolation by combining the valid data on levels l and $l+1$ [70]. The composite gradient operator $\boldsymbol{G}^{cc,\text{comp},l}$ is defined as

$$G_x^{cc,\text{comp},l}\, p\Big|_{i,j} = \frac{p_{i+1,j} - p_{i-1,j}}{2\Delta x^l}, \qquad G_y^{cc,\text{comp},l}\, p\Big|_{i,j} = \frac{p_{i,j+1} - p_{i,j-1}}{2\Delta y^l}. \tag{10}$$

The composite divergence operator $\boldsymbol{D}^{cc,\text{comp},l}\ \cdot\ $ is applied to the edge-centered velocity $\mathbf{u}^{\text{edge},l}$ as

$$\boldsymbol{D}^{cc,\text{comp},l}\ \cdot\ \mathbf{u}^{\text{edge},l}\Big|_{i,j} = \frac{u^{\text{edge},l}_{i+1/2,j} - u^{\text{edge},l}_{i-1/2,j}}{\Delta x^l} + \frac{v^{\text{edge},l}_{i,j+1/2} - v^{\text{edge},l}_{i,j-1/2}}{\Delta y^l}. \tag{11}$$

For a specified level l, the level operators include the level Laplacian $L^{cc,\text{level},l}$, gradient $\boldsymbol{G}^{cc,\text{level},l}$, and divergence $\boldsymbol{D}^{cc,\text{level},l}\ \cdot$ operators. Their definitions are the same as Eqs. (9), (10), and (11), except that the invalid region of level l is now also included in the domain of the operators, i.e., $p_{i,j} \subset (\Omega^l_{\text{valid}} \cup \Omega^l_{\text{invalid}})$ and $\mathbf{u}^{\text{edge},l}\Big|_{i,j} \subset (\Omega^l_{\text{valid}} \cup \Omega^l_{\text{invalid}})$. For ease of describing the time stepping scheme, we replace the discrete level operators $\boldsymbol{D}^{cc,\text{level},l}\ \cdot$, $\boldsymbol{G}^{cc,\text{level},l}$, and $L^{cc,\text{level},l}$ with their continuous counterparts $\boldsymbol{\nabla}\cdot$, $\boldsymbol{\nabla}$, and ∇^2, respectively, in Section 5.

4.2. Lagrangian discretization

In the IB approach, the Lagrangian markers that define the solid structure are free to move on the background Eulerian grid. Following the convention of Nangia et al. [64], the Lagrangian markers are indexed by (q, m) with mesh spacings $(\Delta s_1, \Delta s_2)$ in the two curvilinear directions. Because the present work only considers rigid body kinematic constraints, explicit connectivity information among the marker points is not required. Moreover, the Lagrangian markers are placed on the finest grid level $l_{\max}$, and therefore, the relevant Lagrangian quantities, including the position $(\mathbf{X})_{q,m}$, velocity $(\mathbf{U})_{q,m}$, and force $(\mathbf{F})_{q,m}$ of the marker points, are defined only on the finest grid level.

4.3. Lagrangian–Eulerian coupling

The Eulerian and Lagrangian quantities are transformed through two coupling operators described in Section 2: the force spreading operator $\boldsymbol{\mathcal{S}}[\mathbf{X}]$ and the velocity interpolation operator $\boldsymbol{\mathcal{J}}[\mathbf{X}]$. In this work, a canonical d-dimensional delta function with the tensor product form $\delta_h(\mathbf{x}) = \prod_{i=1}^d \delta_h(x_i)$ is applied to approximate the coupling operators. In each dimension, $\delta_h(x_i)$ is defined as $\delta_h(x_i) = \frac{1}{h}\varphi_4\left(\frac{x_i}{h}\right)$, where $\varphi_4(r)$ is the four-point IB kernel of Peskin [44,45] given by

$$\varphi_4(r) = \begin{cases} \frac{1}{8}(3 - 2|r| + \sqrt{1 + 4|r| - 4r^2}), & 0 \leqslant |r| < 1, \\ \frac{1}{8}(5 - 2|r| - \sqrt{-7 + 12|r| - 4r^2}), & 1 \leqslant |r| < 2, \\ 0, & 2 \leqslant |r|. \end{cases} \tag{12}$$

The solid structure is completely enclosed by the finest level $l_{\max}$, and the Lagrangian markers within the solid region Ω_b are identified by the marker set $\mathcal{M}_b$. Given a Lagrangian force density $\mathbf{F} = (F_1, F_2)$ defined on $\mathcal{M}_b$, the corresponding Eulerian force density

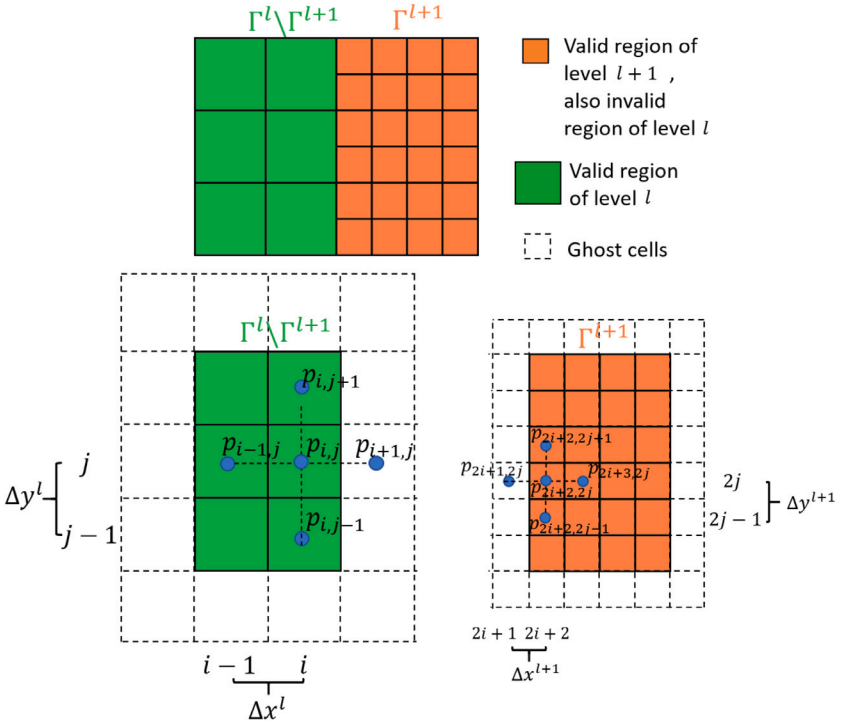

Fig. 3. Schematic of valid and invalid regions on a multilevel grid and five-point stencil used in the discrete Laplacian operator on coarse and fine levels. The refinement ratio between level l and level $l+1$ is 2.

$\mathbf{f}^{l_{\max}} = \left(f_1^{l_{\max}}, f_2^{l_{\max}}\right)$ is obtained through the force spreading operation (Eq. (6)) as

$$\begin{aligned}\left(f_1\right)_{i,j}^{l_{\max}} &= \sum_{(q,m)\in\mathcal{M}_b} \left(F_1\right)_{q,m} \delta_{h^{l_{\max}}}\left(\mathbf{x}_{i,j}^{l_{\max}} - \mathbf{X}_{q,m}\right) \Delta s_1 \Delta s_2,\\ \left(f_2\right)_{i,j}^{l_{\max}} &= \sum_{(q,m)\in\mathcal{M}_b} \left(F_2\right)_{q,m} \delta_{h^{l_{\max}}}\left(\mathbf{x}_{i,j}^{l_{\max}} - \mathbf{X}_{q,m}\right) \Delta s_1 \Delta s_2.\end{aligned} \tag{13}$$

Likewise, with $\mathbf{u}^{l_{\max}} = \left(u_1^{l_{\max}}, u_2^{l_{\max}}\right)$ being the Eulerian velocity on the finest level $l_{\max}$ and $\mathbf{U} = (U_1, U_2)$ being the velocity of the Lagrangian markers, the velocity interpolation operation (Eq. (7)) can be written as

$$\begin{aligned}\left(U_1\right)_{q,m} &= \sum_{(i,j)} \left(u_1\right)_{i,j}^{l_{\max}} \delta_{h^{l_{\max}}}\left(\mathbf{x}_{i,j}^{l_{\max}} - \mathbf{X}_{q,m}\right) \Delta x^{l_{\max}} \Delta y^{l_{\max}},\\ \left(U_2\right)_{q,m} &= \sum_{(i,j)} \left(u_2\right)_{i,j}^{l_{\max}} \delta_{h^{l_{\max}}}\left(\mathbf{x}_{i,j}^{l_{\max}} - \mathbf{X}_{q,m}\right) \Delta x^{l_{\max}} \Delta y^{l_{\max}}.\end{aligned} \tag{14}$$

5. Time advancement

In this work, we use a level-by-level method [27,28] for the time advancement of variables on a multilevel grid. As our multilevel advancement algorithm (Section 5.4) is based on the single-level advancement method, we introduce a single-level advancement algorithm (Section 5.1) in which different types of kinematic constraints (Section 5.2) are considered and the hydrodynamic force and torque acting on the immersed object are calculated (Section 5.3) if needed. The multilevel advancement algorithm, which combines the single-level advancement algorithm with the subcycling and non-subcycling methods, is discussed in Section 5.4.1. To match the data across multiple levels, various synchronization operations are performed every time a finer level catches up with a coarser level; the details of the synchronization operations are discussed in Section 5.4.2.

5.1. Single-level advancement

Our numerical method uses a time-splitting approach, in which we first solve the momentum equation Eq. (1) using the approximate projection method [71–73] to enforce the incompressibility condition Eq. (2) before correcting the Eulerian velocity to enforce the rigid constraint in the solid domain. At time t^n, we are given the collocated Eulerian velocity $\mathbf{u}^n$ and time-staggered pressure $p^{n-1/2}$ [4,26,74]. For the rigid body, the mass center position $\mathbf{X}_r^n$, velocity $\mathbf{U}_r^n$, and angular velocity $\mathbf{W}_r^n$ are also known at time t^n. Our objective is to obtain the updated fluid velocity $\mathbf{u}^{n+1}$, pressure $p^{n+1/2}$, and updated rigid body quantities $\mathbf{X}_r^{n+1}$, $\mathbf{U}_r^{n+1}$, and $\mathbf{W}_r^{n+1}$ at the next time level t^{n+1}. The details of the single-level time advancement are given below.

1. The LS function ψ^n is calculated based on $\mathbf{X}_r^n$ and the geometry of the solid body, and the LS function ϕ^n are calculated based on the location of the gas–liquid interface, both at time t^n. To capture the movement of the fluid–solid and gas–liquid interfaces, ψ^n and ϕ^n are advanced to ψ^{n+1} and ϕ^{n+1} using

$$\psi^{n+1} = \psi^n - \Delta t\left[\nabla\cdot(\mathbf{u}\psi)\right]^{n+1/2}, \tag{15}$$

$$\phi^{n+1} = \phi^n - \Delta t\left[\nabla\cdot(\mathbf{u}\phi)\right]^{n+1/2}, \tag{16}$$

where the advection terms $[\nabla\cdot(\mathbf{u}\psi)]^{n+1/2}$ and $[\nabla\cdot(\mathbf{u}\phi)]^{n+1/2}$ are computed using the Godunov scheme [26,74–76], as described in Appendix A. The midpoint values of ψ and ϕ are calculated as $\psi^{n+\frac{1}{2}} = (\psi^{n+1}+\psi^n)/2$ and $\phi^{n+\frac{1}{2}} = (\phi^{n+1}+\phi^n)/2$, respectively.

2. To set the midpoint density field $\rho^{n+\frac{1}{2}}$ appearing on the left-hand side of the momentum equation (Eq. (1)), two Heaviside functions are defined as

$$\widetilde{H}^{\text{flow}}\left(\phi^{n+\frac{1}{2}}\right) = \begin{cases} 0, & \phi^{n+\frac{1}{2}} < -n_{\text{cells}}\Delta x \\ \frac{1}{2}\left(1 + \frac{1}{n_{\text{cells}}\Delta x}\phi^{n+\frac{1}{2}} + \frac{1}{\pi}\sin\left(\frac{\pi}{n_{\text{cells}}\Delta x}\phi^{n+\frac{1}{2}}\right)\right), & \left|\phi^{n+\frac{1}{2}}\right| \le n_{\text{cells}}\Delta x \\ 1, & \text{otherwise,}\end{cases} \tag{17}$$

$$\widetilde{H}^{\text{body}}\left(\psi^{n+\frac{1}{2}}\right)$$

$$= \begin{cases} 0, & \psi^{n+\frac{1}{2}} < -n_{\text{cells}}\Delta x \\ \frac{1}{2}\left(1 + \frac{1}{n_{\text{cells}}\Delta x}\psi^{n+\frac{1}{2}} + \frac{1}{\pi}\sin\left(\frac{\pi}{n_{\text{cells}}\Delta x}\psi^{n+\frac{1}{2}}\right)\right), & \left|\psi^{n+\frac{1}{2}}\right| \le n_{\text{cells}}\Delta x \\ 1, & \text{otherwise,} \end{cases} \tag{18}$$

where n_{cells} is the smearing width of the fluid–solid or gas–liquid interface assuming a uniform grid spacing in all directions, i.e., $\Delta x = \Delta y$. The density $\rho^{n+\frac{1}{2}}$ is then set via a two-step process [64,77] as

$$\widetilde{\rho}^{n+\frac{1}{2}} = \rho_{\text{g}} + \left(\rho_{\text{l}} - \rho_{\text{g}}\right)\widetilde{H}^{\text{flow}}\left(\phi^{n+\frac{1}{2}}\right), \tag{19}$$

$$\rho^{n+\frac{1}{2}} = \rho_{\text{s}} + \left(\widetilde{\rho}^{n+\frac{1}{2}} - \rho_{\text{s}}\right)\widetilde{H}^{\text{body}}\left(\psi^{n+\frac{1}{2}}\right), \tag{20}$$

where ρ_{l}, ρ_{g}, and ρ_{s} are the liquid density, gas density, and solid density, respectively. Depending on the type of kinematic constraint (Section 5.2), the modified density $\wp$ appearing in the gravitational term of the momentum equation is set to

$$\wp = \begin{cases} \rho_{\text{l}}, & \text{for prescribed motion of the solid} \\ \rho_{\text{s}}, & \text{for free motion and prescribed shape of the solid} \end{cases} \tag{21}$$

so that the inertia and buoyancy effects due to the weight of the solid are properly included in the FSI simulation without producing spurious gravitational currents [64,77].

3. Neither ψ nor ϕ are guaranteed to retain the signed distance property after the advections in Eqs. (15) and (16), even if they are initialized as the signed distance function at the beginning of the simulation. For the structures with relatively simple geometries that are considered in this work, the solid LS function ψ can be directly computed using the centroid information of the body. For immersed bodies with complex geometries, the constructive solid geometry (CGS) and/or R-functions [78] can be employed to determine the analytical expressions for ψ. The fluid LS function ϕ is reinitialized by computing the steady-state solution to the Hamilton–Jacobi equation

$$\frac{\partial d}{\partial \tau} = S(\phi)(1 - |\nabla d|), \tag{22}$$

where

$$S(\phi) = 2\left(H(\phi) - 1/2\right). \tag{23}$$

The initial condition of Eq. (22) is

$$d(\boldsymbol{x}, \tau = 0) = \phi^{n+1}(\boldsymbol{x}), \tag{24}$$

where τ is the pseudotime for iterations. A classical second-order Runge–Kutta (RK) method is applied for the pseudotime advancement of d [74,79], which helps minimize the volume change of each fluid phase and ensure mass conservation [74,79,80]. The flow LS function ϕ is updated by d after this pseudotime advancement.

4. The intermediate velocity $\widetilde{\mathbf{u}}^{*,n+1}$ is solved semi-implicitly without considering the rigidity constraint as

$$\begin{aligned} &\rho^{n+\frac{1}{2}}\left(\frac{\widetilde{\mathbf{u}}^{*,n+1} - \mathbf{u}^{n}}{\Delta t} + \boldsymbol{\nabla}\cdot(\mathbf{u}\mathbf{u})^{n+\frac{1}{2}}\right) \\ &\quad = -\boldsymbol{\nabla}p^{n-\frac{1}{2}} + \frac{1}{2}\left(\boldsymbol{\nabla}\cdot\mu(\psi^{n+1},\phi^{n+1})\boldsymbol{\nabla}\widetilde{\mathbf{u}}^{*,n+1} + \boldsymbol{\nabla}\cdot\mu(\psi^{n},\phi^{n})\boldsymbol{\nabla}\mathbf{u}^{n}\right) + \wp^{n+\frac{1}{2}}\mathbf{g}, \end{aligned} \tag{25}$$

where the convective term $\boldsymbol{\nabla}\cdot(\mathbf{u}\mathbf{u})^{n+\frac{1}{2}}$ is calculated using the Godunov scheme (Appendix A). In this work, the viscosity $\mu(\psi^{n+1},\phi^{n+1})$ or $\mu(\psi^{n},\phi^{n})$ does not depend on the solid LS function ψ and is set to the surrounding fluid viscosity, i.e., $\mu^{n+1} = \mu_{\text{f}}(\phi^{n+1})$, following [64].

5. After obtaining the intermediate velocity, a level projection is applied to obtain the updated velocity $\widetilde{\mathbf{u}}^{n+1}$ and pressure $p^{n+\frac{1}{2}}$ fields. An auxiliary variable $\boldsymbol{V}$ is first calculated by

$$\boldsymbol{V} = \frac{\widetilde{\mathbf{u}}^{*,n+1}}{\Delta t} + \frac{1}{\rho^{n+1/2}}\boldsymbol{\nabla}p^{n-\frac{1}{2}}. \tag{26}$$

Next, $\boldsymbol{V}$ is projected on the divergence-free velocity field to obtain the updated pressure $p^{n+1/2}$ via

$$L^{cc,l}_{\rho^{n+1/2}}p^{n+1/2} = \boldsymbol{\nabla}\cdot\boldsymbol{V}, \tag{27}$$

where $L^{cc,l}_{\rho^{n+1/2}}p^{n+1/2}$ is the density-weighted approximation to $\boldsymbol{\nabla}\cdot(1/\rho^{n+1/2}\boldsymbol{\nabla}p^{n+1/2})$ on level l. The divergence-free velocity $\widetilde{\mathbf{u}}^{n+1}$ on level l is then obtained as

$$\widetilde{\mathbf{u}}^{n+1} = \Delta t\left(\boldsymbol{V} - \frac{1}{\rho^{n+1/2}}\boldsymbol{\nabla}p^{n+1/2}\right). \tag{28}$$

We note that the intermediate velocity on each level is projected without the pressure gradient term, as the pressure gradient term is subtracted in Eq. (26). This step reduces the accumulation of pressure errors and produces a more stable algorithm [71,81]. Moreover, the approximate projection approach effectively removes the issue of the pressure checker-boarding problem that appears on collocated grids [27,69,73,82].

6. The updated velocity $\widetilde{\mathbf{u}}^{n+1}$ satisfies the incompressibility condition but needs to be corrected to satisfy the constraints of motions of the rigid body within the solid region $V_{\text{b}}(t)$. To achieve this, the Lagrangian velocity $(\mathbf{U}_{\text{b}})^{n+1}_{q,m}$ and the marker points position $\mathbf{X}^{n+1}_{q,m}$ need to be approximated. Because these approximations may vary based on the kinematic constraints, for now, we assume that they are known variables; the detailed steps for calculating $(\mathbf{U}_{\text{b}})^{n+1}_{q,m}$ and $\mathbf{X}^{n+1}_{q,m}$ are presented in Section 5.2.

To proceed, the difference between the Lagrangian velocity $(\mathbf{U}_{\text{b}})^{n+1}_{q,m}$ and the background Eulerian velocity $\widetilde{\mathbf{u}}^{n+1}$ is calculated via the velocity interpolation operation as

$$\left(\Delta\mathbf{U}_{\text{c}}\right)^{n+1}_{q,m} = \begin{cases} \left(\mathbf{U}_{\text{b}}\right)^{n+1}_{q,m} - \left(\mathcal{J}\left[\mathbf{X}^{n+1}_{q,m}\right]\tilde{\mathbf{u}}^{n+1}\right)_{q,m}, & \text{for } (q,m) \in V_{\text{b}}(t), \\ \mathbf{0}. & \text{for } (q,m) \notin V_{\text{b}}(t). \end{cases} \tag{29}$$

This velocity difference is then employed to approximate the Lagrangian and Eulerian constraint forces as

$$\mathbf{F}^{n+1}_{q,m} = \frac{\rho_{\text{s}}}{\Delta t}(\Delta\mathbf{U}_{\text{c}})^{n+1}_{q,m}, \tag{30}$$

$$\mathbf{f}^{n+1}_{\text{c}} = \mathcal{S}\left[\mathbf{X}^{n+1}_{q,m}\right]\mathbf{F}^{n+1}_{q,m}. \tag{31}$$

The Eulerian velocity field is corrected by the Eulerian constraint force as

$$\mathbf{u}^{n+1} = \widetilde{\mathbf{u}}^{n+1} + \frac{\Delta t}{\rho_{\text{s}}}\mathbf{f}^{n+1}_{\text{c}}, \tag{32}$$

and the solid effects are properly included in the fluid–solid system [64].

5.2. Types of kinematic constraints

The approximation to the constrained Lagrangian velocity $(\mathbf{U}_{\text{b}})^{n+1}_{q,m}$ and position $\mathbf{X}^{n+1}_{q,m}$ depends on the type of kinematic constraint in the FSI. In this work, we consider three types of kinematic constraints: prescribed motion of the structure, free motion of the structure, and prescribed shape of the structure.

5.2.1. Prescribed motion

If the motion of the structure is prescribed, then the velocity and position of the body are known *a priori* and not influenced by the surrounding fluid. Thus, the centroid position $\mathbf{X}^{n}_{\text{r}}$, centroid velocity $\mathbf{U}^{n+1}_{\text{r}}$, and angular velocity $\mathbf{W}^{n+1}_{\text{r}}$ of the body are given. The constrained Lagrangian velocity $(\mathbf{U}_{\text{b}})^{n+1}_{q,m}$ of the markers is calculated as

$$\left(\mathbf{U}_{\text{b}}\right)^{n+1}_{q,m} = \mathbf{U}^{n+1}_{\text{r}} + \mathbf{W}^{n+1}_{\text{r}} \times \mathbf{R}^{n}_{q,m}, \tag{33}$$

where $\mathbf{R}^{n}_{q,m} = \mathbf{X}^{n}_{q,m} - \mathbf{X}^{n}_{\text{r}}$. Since the marker position $\mathbf{X}^{n}_{q,m}$ is already known from the calculation at the previous time step, the new position $\mathbf{X}^{n+1}_{q,m}$ of the marker points is updated using the midpoint scheme as

$$\mathbf{X}^{n+1}_{q,m} = \mathbf{X}^{n}_{q,m} + \frac{\Delta t}{2}\left(\left(\mathbf{U}_{\text{b}}\right)^{n}_{q,m} + \left(\mathbf{U}_{\text{b}}\right)^{n+1}_{q,m}\right). \tag{34}$$

5.2.2. Free motion

In contrast to the prescribed kinematics case, the motion of a freely moving body is influenced by the surrounding fluid. To account for this two-way coupled interaction, the linear and angular momentum of the fluid occupying the solid domain is redistributed as rigid body momentum, which provides an estimate of the centroid velocities $\mathbf{U}_{\mathrm{r}}^{n+1}$ and $\mathbf{W}_{\mathrm{r}}^{n+1}$ at the new time level. This momentum projection step is carried out using the principle of conservation of linear and angular momentum in the solid domain and is written as

$$\mathrm{M_b}\mathbf{U}_{\mathrm{r}}^{n+1} = \sum_{\mathbf{X}_{q,m}\in V_{\mathrm{b}}} \rho_{\mathrm{s}} \left(\mathcal{J}_h \left[\mathbf{X}^{n+\frac{1}{2}} \right] \widetilde{\mathbf{u}}^{n+1} \right)_{q,m} \Delta s_1 \Delta s_2 + \mathbf{F}_{\mathrm{ext}}\, \Delta t, \tag{35}$$

$$\mathbf{I}_{\mathrm{b}}\mathbf{W}_{\mathrm{r}}^{n+1} = \sum_{\mathbf{X}_{q,m}\in V_{\mathrm{b}}} \rho_{\mathrm{s}} \mathbf{R}_{q,m}^{n+\frac{1}{2}} \times \left(\mathcal{J}_h \left[\mathbf{X}^{n+\frac{1}{2}} \right] \widetilde{\mathbf{u}}^{n+1} \right)_{q,m} \Delta s_1 \Delta s_2, \tag{36}$$

where

$$\mathrm{M_b} = \sum_{\mathbf{X}_{q,m}\in V_{\mathrm{b}}} \rho_{\mathrm{s}} \Delta s_1 \Delta s_2 \tag{37}$$

is the mass of the solid body and

$$\mathbf{I}_{\mathrm{b}} = \sum_{\mathbf{X}_{q,m}\in V_{\mathrm{b}}} \rho_{\mathrm{s}} \left(\mathbf{R}_{q,m}^{n+\frac{1}{2}} \cdot \mathbf{R}_{q,m}^{n+\frac{1}{2}}\, \mathbf{I} - \mathbf{R}_{q,m}^{n+\frac{1}{2}} \otimes \mathbf{R}_{q,m}^{n+\frac{1}{2}} \right) \tag{38}$$

is the moment of inertia tensor, with $\mathbf{I}$ being the d-dimensional identity tensor. In Eq. (35), $\mathbf{F}_{\mathrm{ext}}$ is utilized to account for the effects of external (nonhydrodynamic) forces, such as those provided by springs and dampers that may be attached to a solid body. The centroid position $\mathbf{X}_r^{n+1}$ is then updated using the midpoint scheme as

$$\mathbf{X}_r^{n+1} = \mathbf{X}_r^n + \frac{\Delta t}{2}(\mathbf{U}_r^{n+1} + \mathbf{U}_r^n). \tag{39}$$

The constrained Lagrangian velocity $(\mathbf{U}_{\mathrm{b}})_{q,m}^{n+1}$ and position $\mathbf{X}_{q,m}^{n+1}$ of the marker points are updated using Eqs. (33) and (34), respectively, in a manner similar to the prescribed motion case.

5.2.3. Prescribed shape

In some cases of freely moving bodies, particularly those encountered in aquatic locomotion, the shape of the body changes with time. The deformation of the body can be prescribed as shape mapping $\chi = \chi(\mathbf{s}, t)$. The deformation velocity of the body $\mathbf{U}_{\mathrm{k}}$ can be obtained as $\mathbf{U}_{\mathrm{k}} = \partial\chi(\mathbf{s}, t)/\partial t$. In general, the net linear and angular momentum of the deformation velocity is nonzero. A direct use of such deformation velocity in the FSI algorithm would manifest as a spurious external force or torque acting on the freely swimming body. Therefore, the additional momentum due to the prescribed deformation velocity must be removed from the equations of motion to conserve the system momentum. To project the deformation kinematics onto a space of net zero momentum, the linear and angular momentum redistribution Eqs. (35) and (36) are modified to remove the extraneous momentum as

$$\mathrm{M_b}\mathbf{U}_{\mathrm{r}}^{n+1} = \sum_{\mathbf{X}_{q,m}\in V_{\mathrm{b}}} \rho_{\mathrm{s}} \left((\mathcal{J}_h \left[\mathbf{X}^{n+\frac{1}{2}} \right] \widetilde{\mathbf{u}}^{n+1})_{q,m} - (\mathbf{U}_{\mathrm{k}})_{q,m}^{n+1} \right) \Delta s_1 \Delta s_2, \tag{40}$$

$$\mathbf{I}_{\mathrm{b}}\mathbf{W}_{\mathrm{r}}^{n+1} = \sum_{\mathbf{X}_{q,m}\in V_{\mathrm{b}}} \rho_{\mathrm{s}} \mathbf{R}_{q,m}^{n+\frac{1}{2}} \times \left((\mathcal{J}_h \left[\mathbf{X}^{n+\frac{1}{2}} \right] \widetilde{\mathbf{u}}^{n+1})_{q,m} - (\mathbf{U}_{\mathrm{k}})_{q,m}^{n+1} \right) \Delta s_1 \Delta s_2. \tag{41}$$

In these equations, it is assumed that the deformation velocity $(\mathbf{U}_{\mathrm{k}})_{q,m}^{n+1}$ has been obtained from the prescribed body shape $\chi(\mathbf{s}, t)$. In addition, the deformation velocity $(\mathbf{U}_{\mathrm{k}})_{q,m}^{n+1}$ needs to be considered while updating the constrained Lagrangian velocity $(\mathbf{U}_{\mathrm{b}})_{q,m}^{n+1}$ as

$$\left(\mathbf{U}_{\mathrm{b}}\right)_{q,m}^{n+1} = \mathbf{U}_{\mathrm{r}}^{n+1} + \mathbf{W}_{\mathrm{r}}^{n+1} \times \mathbf{R}_{q,m}^{n} + \left(\mathbf{U}_{\mathrm{k}}\right)_{q,m}^{n+1}. \tag{42}$$

The position of the marker points $\mathbf{X}_{q,m}^{n+1}$ is updated using Eq. (34).

5.3. Hydrodynamic force and torque calculation

In the DLM/IB method, the fluid–structure interaction is handled implicitly. Therefore, there is no need to iterate between fluid domain integrators and solid domain integrators to maintain stability, which is a strict requirement for some sharp-interface immersed boundary methods [52,83,84]. Moreover, there is no need to explicitly evaluate the hydrodynamic force and torque acting on the body to determine its motion. These features make the DLM/IB scheme more efficient than some sharp-interface immersed boundary methods that require velocity interpolation and pressure reconstruction around the solid surface to calculate the required hydrodynamic forces and torques. However, if needed, the hydrodynamic force and torque acting on the immersed object in the DLM/IB method can still be calculated as a postprocessing step using the following equations [64,65]:

$$\mathcal{F}^{n+1} = \sum_{\mathbf{X}_{q,m}\in V_{\mathrm{b}}} \rho_{\mathrm{s}} \left[\frac{\left(\mathbf{U}_{\mathrm{b}}\right)_{q,m}^{n+1} - \left(\mathbf{U}_{\mathrm{b}}\right)_{q,m}^{n}}{\Delta t} - \frac{\Delta \mathbf{U}_{q,m}^{n+1}}{\Delta t} \right] \Delta s_1 \Delta s_2, \tag{43}$$

$$\mathcal{M}^{n+1} = \sum_{\mathbf{X}_{q,m}\in V_{\mathrm{b}}} \rho_{\mathrm{s}} \mathbf{R}_{q,m}^{n+1} \times \left[\frac{\left(\mathbf{U}_{\mathrm{b}}\right)_{q,m}^{n+1} - \left(\mathbf{U}_{\mathrm{b}}\right)_{q,m}^{n}}{\Delta t} - \frac{\Delta \mathbf{U}_{q,m}^{n+1}}{\Delta t} \right] \Delta s_1 \Delta s_2. \tag{44}$$

5.4. Multilevel advancement

This section describes the extension of the single-level advancement algorithm presented in Section 5.1 to the multilevel advancement algorithm using subcycling and non-subcycling methods (Section 5.4.1). A force averaging scheme is proposed to average the Eulerian forces from the finest level to the coarser levels, which conserves the momentum of the system at a discrete level. A synchronization step (Section 5.4.2) is then applied to match the variables on multiple levels, which provides a better representation of the composite solution on multiple levels. Next, a multilevel initialization of the fluid and solid fields is introduced in Section 5.4.3. A summary of the multilevel advancement algorithm is provided in Section 5.4.4.

5.4.1. Subcycling and non-subcycling methods

To advance variables on a multilevel grid, we consider two types of cycling methods, namely, the subcycling method and non-subcycling method. In the subcycling method, variables on different levels are advanced with different time step sizes. The main advantage of the subcycling method is that when the Courant–Friedrichs–Lewy (CFL) number is kept the same on different grid levels, a larger grid spacing on a coarser level allows for a larger time step size on this level. For example, if the refinement ratio between two neighboring levels is two and the L^∞-norm of velocity on both levels is approximately the same, then the time step size on the coarser level Δt^l can be twice as large as that on the finer level Δt^{l+1}. In contrast, in the non-subcycling method, variables on different levels advance with the same time step size that is dictated by the finest level $l_{\max}$. In this case, variables on all levels are always at the same time instant.

Fig. 4 schematically shows how the subcycling and non-subcycling methods are used to advance the variables on a multilevel grid with $l_{\max} = 2$. As shown in the sketch, only 7 substeps are needed to advance all levels from t^n to $t^n + \Delta t^0$ using the subcycling method. In contrast, 12 substeps are needed for the non-subcycling method. Within each substep of the two cycling methods, the single-level advancement algorithm described in Section 5.1 is employed to time march the solution. Although the non-subcycling method allocates more steps than the subcycling method, the latter requires an additional time interpolation of variables due to the mismatch of time step sizes among different levels. For example, to fill the ghost cell values at the coarse–fine boundary of a finer level, spatiotemporal interpolation of variables on the coarser level is required in the subcycling method. In contrast, in the non-subcycling method, spatial interpolation only

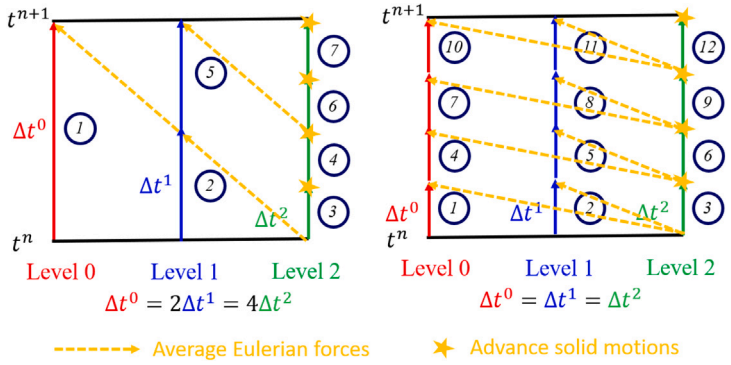

Fig. 4. Schematic of the substeps in the level-by-level advancement method for a three-level grid ($l_{\max} = 2$). Left: the subcycling method. Right: the non-subcycling method. The substeps are represented by circled numbers. The yellow dashed lines represent the averaging of Eulerian forces from the finest level to the coarser levels, and the yellow stars represent the time advancement of the immersed body on the finest level.

suffices. In Section 6.9, the time savings for the subcycling and non-subcycling methods are carefully quantified in the simulations of the three-dimensional swimming eel.

In the context of FSI problems, there are two points of consideration when using the level-by-level time advancement technique. First, the Lagrangian markers are distributed only on the finest level. Thus, the Lagrangian quantities are updated only after the time advancement of the flow variables on the finest level, as indicated by the yellow stars in Fig. 4. Consequently, the solution on the finest level represents the final solid motion. Second, the Eulerian IB forces on a coarser level need to be properly considered. If the IB forces are not included on a coarser level, then the flow field on this level cannot "feel" the existence of the solid structure. Consequently, the updated flow velocity on a coarser level that is used to provide Dirichlet boundary conditions for the finer level will be incorrect, which will further lead to incorrect solutions on the finer level. To resolve this problem, we propose a force averaging algorithm to average the latest Eulerian forces on the finest level onto coarser levels in a sequential manner, as depicted in Fig. 4. Specifically, $\tilde{\mathbf{f}}_{\mathrm{c}}^{t+\Delta t^l,l}$ denotes the Eulerian force that needs to be approximated at time $t + \Delta t^l$ on the coarser level l for all $0 \le l < l_{\max}$. Because the Eulerian force $\tilde{\mathbf{f}}_{\mathrm{c}}^{t,l_{\max}}$ on the finest level $l_{\max}$ is known at time t, we obtain the following Algorithm 1:

Algorithm 1 Force averaging algorithm

if subcycling method is used **then**
  $\Delta t^l = 2^{l_{\max}-l}\Delta t^{l_{\max}}$ for all $0 \le l < l_{\max}$
else
  $\Delta t^l = \Delta t^{l_{\max}}$ for all $0 \le l < l_{\max}$
end if
for $l = l_{\max} - 1, 0, -1$ **do**
  if $l = l_{\max} - 1$ **then**
    $\tilde{\mathbf{f}}_{\mathrm{c}}^{t+\Delta t^l,l} \leftarrow$ average $\tilde{\mathbf{f}}_{\mathrm{c}}^{t,l+1}$
  else
    $\tilde{\mathbf{f}}_{\mathrm{c}}^{t+\Delta t^l,l} \leftarrow$ average $\tilde{\mathbf{f}}_{\mathrm{c}}^{t+\Delta t^{l+1},l+1}$
  end if
end for
Replace $\mathbf{f}_{\mathrm{c}}^{n+1}$ in Eq. (32) with $\tilde{\mathbf{f}}_{\mathrm{c}}^{t+\Delta t^l,l}$ for updating the velocity

In Algorithm 1, four-point averaging operators and eight-point averaging operators [27,28] are employed in two spatial dimensions and three spatial dimensions, respectively.

5.4.2. Synchronization

The synchronization operation is utilized to make the solution data consistent across multiple levels and to obtain the composite solution from the level data. This operation is applied in both cycling methods [26–28]. There are three substeps involved in the synchronization operation:

Substep 1. Averaging

The fluid velocity $\mathbf{u}$, pressure p, and LS functions ψ and ϕ on coarser levels are replaced by the corresponding data on the finer levels after the averaging substep. The variable values on the finer levels are considered to be better resolved than those on the coarser levels. Because a collocated grid framework is applied in this work, the same averaging operator can be employed for all of the aforementioned variables. We note that the averaging operation used here is different from the averaging step of the force averaging algorithm. Here, the averaging operator aims to synchronize all flow variables on multiple levels; it is applied to variables at the same time instance whenever a finer level catches up with a coarser level. On the other hand, the averaging operator in the force averaging algorithm is applied only to the Eulerian IB force variable at different time instants to obtain an approximate value of the IB force on a coarser level.

Substep 2. MAC synchronization and refluxing

At the CF boundary, the advection velocity $\mathbf{u}^{\mathrm{adv},l}$ on the coarser level l is generally not equal to the edge average of the advection velocity $\mathbf{u}^{\mathrm{adv},l+1}$ on the next finer level $l+1$. This difference can create an imbalance of the momentum and scalar fluxes at the CF boundary. As a result, the free stream is not preserved while advancing the variables level by level. To remedy this problem, MAC synchronization and refluxing algorithms are carefully designed to maintain the conservation of momentum and scalar in the entire domain. The algorithms are detailed in Appendix B.

Substep 3. Composite grid projection

Because the level projection is applied on a level-by-level basis, it does not guarantee that the fluid velocity is divergence-free across all levels [26,28]. As a last step of the synchronization operation, a composite grid projection is applied to enforce the divergence-free condition on the velocity field across the entire hierarchy [26,27]. Using the composite grid operators defined in Section 4.1, a multilevel multigrid (MLMG) solver [29] is employed to project the fluid velocity on a divergence-free space as

$$L_{\rho^{n+1}}^{cc,\mathrm{comp}}\Theta = \frac{1}{\Delta t^{\mathrm{sync}}} D^{cc,\mathrm{comp}}\mathbf{u}^{n+1}, \tag{45}$$

$$\mathbf{u}^{n+1} := \mathbf{u}^{n+1} - \Delta t^{\mathrm{sync}} \boldsymbol{G}^{cc,\mathrm{comp}}\Theta, \tag{46}$$

where Δt^{sync} is the time step of level 0, i.e., $\Delta t^{\mathrm{sync}} = \Delta t^0$, and $L_{\rho^{n+1}}^{cc,\mathrm{comp}}\Theta$ is the density-weighted approximation to the term $\nabla \cdot (1/\rho^{n+1}\nabla\Theta)$. We note that $\mathbf{u}^{n+1}$ becomes divergence-free in a multilevel sense after the composite grid projection substep.

5.4.3. Multilevel initialization

All field values need to be initialized on all levels at the beginning of the simulation. For example, the fluid velocity $\mathbf{u}$ on the coarsest level (level 0) is assigned based on the initial condition. The solid LS function ψ is computed analytically based on the position of the center of mass of the body $\mathbf{X}_r^0$ and its geometry. The flow LS function ϕ is initialized based on the position of the gas–liquid interface when multiphase flow effects are included in the FSI simulation. The grid cells on the next level (level 1) are generated based on refinement and nesting criteria. After the level refinement, the velocity $\mathbf{u}$ and LS functions ψ and ϕ values on level 1 are assigned based on the initial conditions. This "refining and filling" procedure is repeated until the finest level $l_{\max}$ is reached or until there is no need to refine the grid based on the refinement criteria.

Next, the Lagrangian markers are initialized on the finest level by placing one Lagrangian marker per Eulerian grid cell. The physical position $\mathbf{X}_{q,m}^0$, velocity $\mathbf{U}_{q,m}^0$ and radius vector $\mathbf{R}_{q,m}^0$ of the Lagrangian markers are determined from the known initial condition of the solid body. The pressure p is initialized to zero on all levels.

5.4.4. Summary of the multilevel advancement algorithm

Algorithm 2 summarizes the unified multilevel advancement algorithm for both the subcycling methods and non-subcycling methods. After the variable initialization, we can use either the subcycling method or the non-subcycling method for time advancement. The force averaging algorithm is employed before each time step to approximate the Eulerian IB forces on all coarser levels. The synchronization step is applied whenever a finer level catches up with a coarser level. Grid refinement is applied before moving to the next time step.

Algorithm 2 Multilevel advancement algorithm

Initialize $\mathbf{X}_r^0$, $\mathbf{U}_r^0$, $\mathbf{W}_r^0$, $\mathbf{u}^0$, ψ^0, ϕ^0, and p^0 on level 0
$l \leftarrow 0$
while refinement criteria are satisfied on level l and $l < l_{\max}$ **do**
  Regrid the patch hierarchy to obtain level $l+1$
  Initialize $\mathbf{X}_r^0$, $\mathbf{U}_r^0$, $\mathbf{W}_r^0$, $\mathbf{u}^0$, ψ^0, ϕ^0, and p^0 on level $l+1$
  $l \leftarrow l+1$
end while
Initialize $\mathbf{X}_{q,m}^0$, $\mathbf{U}_{q,m}^0$ and $\mathbf{R}_{q,m}^0$ for all Lagrangian markers on level $l_{\max}$
if subcycling method is used **then**
  $\Delta t^l = 2^{l_{\max}-l}\Delta t^{l_{\max}}$ for all $0 \le l < l_{\max}$
else
  $\Delta t^l = \Delta t^{l_{\max}}$ for all $0 \le l < l_{\max}$
end if
for $n = 1, n_{\max}$ **do** ▷ $n_{\max}$ is the number of time steps in the simulation
  LevelCycling$(0,\ t_n^0,\ t_n^0 + \Delta t^0,\ \Delta t^0)$
  Apply the synchronization projection using Eqs. (45)–(46)
  Regrid the patch hierarchy and interpolate $\mathbf{u}$, ψ, ϕ, and p onto new patches
end for

procedure LevelCycling$(l,\ t^l,\ t_{\max}^l,\ \Delta t^l)$
  while $t^l < t_{\max}^l$ **do**
    if $l < l_{\max}$ **then**
      Apply the force averaging algorithm on level l ▷ **Algorithm 1**
    end if
    Perform single-level advancement on level l from t^l to $t^l + \Delta t^l$
      • Advect the LS functions $\psi^{n,l}$ and $\phi^{n,l}$ and reinitialize them using Eqs. (15)–(23)
      • Solve momentum Eq. (28) to obtain $\widetilde{\mathbf{u}}^{n+1,l}$
      • Calculate the Lagrangian correction velocity $\Delta\mathbf{U}_c$ based on a specific kinematic constraint
      • Correct the Eulerian velocity field $\mathbf{u}^{n+1,l}$ using Eq. (32)
    if $l < l_{\max}$ **then**
      LevelCycling$(l+1,\ t^l,\ t^l + \Delta t^l,\ \Delta t^{l+1})$
    end if
    $t^l \leftarrow t^l + \Delta t^l$
  end while
  if $l > 0$ **then**
    Average all data from finer levels to the coarser levels
  end if
  if $l < l_{\max}$ **then**
    Perform MAC synchronization and refluxing using Eqs. (51)–(71)
  end if
end procedure

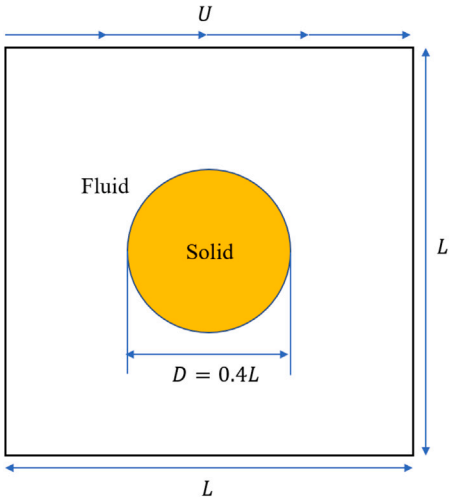

Fig. 5. Schematic of a submerged cylinder in a lid-driven cavity flow.

6. Results

This section presents several canonical testing problems to validate the capabilities and robustness of the proposed AMR framework from different aspects. The following notations are used unless stated otherwise. For each case, Δt_0 denotes the time step on level 0, and Δx_0, Δy_0, and Δz_0 represent the grid spacings in the x-direction, y-direction, and z-direction, respectively, on level 0. For a multilevel grid, the grid spacings on the finer level l satisfy $\Delta x_l = \Delta x_0/2^l$, $\Delta y_l = \Delta y_0/2^l$, and $\Delta z_l = \Delta z_0/2^l$ for all $0 \le l \le l_{max}$.

6.1. Lid-driven cavity with a submerged cylinder

We begin by considering the flow around a cylinder submerged in a lid-driven cavity. As shown in the left part of Fig. 5, a stationary cylinder with diameter $D = 0.4L$ is immersed at the center of a computational domain of extent $\Omega \in [0, L]^2$, with $L = 1$. The lid velocity at the domain top is set to $U = 1$. The Reynolds number of the flow is $Re = \rho_f U L/\mu_f = 1000$. No-slip boundary conditions are applied on all sides of the domain boundaries.

Both the single-level computational case and multilevel computational case are considered for this problem. For the single-level case, we test both the temporal convergence rate and the spatial–temporal convergence rate of the flow field. For the temporal convergence, it is defined as the rate of error reduction with decreasing time step size. When performing a temporal convergence study, it is necessary to design the tests such that the spatial error is minimal [26,85]. We thus use a fixed grid with the smallest grid size and only change the time step size in different tests. For the spatial–temporal convergence, we consider grid numbers of 64 × 64, 128 × 128, 256 × 256, and 512 × 512. The CFL number is kept at a constant value 0.5 for all grids. Because no analytical solution is available for this problem, we compare the numerical solution against a reference solution obtained on a high-resolution 2048 × 2048 uniform grid. When the flow field reaches the steady state at around $tU/L = 30$, the L^2 norm of the velocity errors is computed [43], and the temporal convergence rate and the spatial–temporal convergence rate of the solution are estimated, as shown in Fig. 6. In the left part of Fig. 6, it shows that our mid-point integration scheme can achieve the second order. The right part of Fig. 6 demonstrates that spatial–temporal convergence rate of the velocity is approximately 1.45 for the single-level case. This convergence rate is reasonable considering the smearing of the fluid–solid interface inherent in the DLM/IB method [61,62]. Also consistent

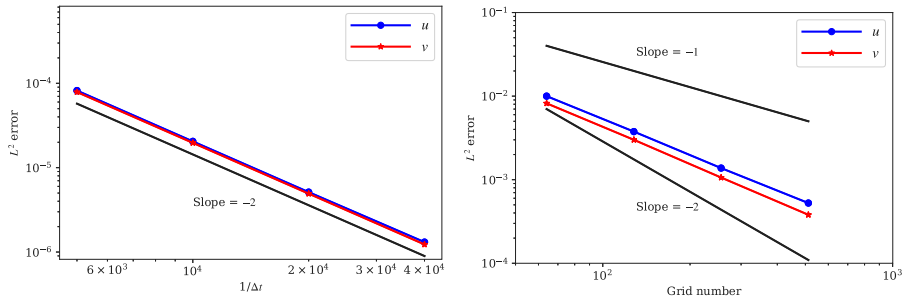

Fig. 6. Temporal convergence (left) and spatial–temporal convergence (right) of u and v components of the flow velocity for the single-level case.

Table 1
Parameters of the lid-driven cavity with a submerged cylinder problem.

Case no.	Grid numbers on level 0	$l_{\max}$	Cycling methods	Force averaging scheme used?
1	2048 × 2048	0	–	–
2	64 × 64	2	Subcycling	Yes
3	64 × 64	2	Subcycling	No

Table 2
Parameters of the cylinder with prescribed motion problem.

Case no.	Grid numbers on level 0	$l_{\max}$	Δt_0	Cycling methods
1	1024 × 512	0	5×10^{-4}	–
2	256 × 128	2	2×10^{-3}	Subcycling
3	256 × 128	2	5×10^{-4}	Non-subcycling

with other IB papers [4,49], the dominant errors are localized near the fluid–structure interface.

For the multilevel grid case, we consider static mesh refinement. Grid cells in the rectangular region $(x, y) \in [0.2, 0.8]$ are refined to level 1, and grid cells in the region $(x, y) \in [0.25, 0.75]$ are further refined to $l_{\max} = 2$. The cylinder is kept on the finest level, and a refinement ratio of two is applied to both levels.

Before proceeding to test the convergence rate on a multilevel grid, we demonstrate the necessity of the force averaging algorithm (Section 5.4.1) for the level-by-level time advancement method. Three cases are considered, as listed in Table 1. Case 1 is the single-level case, which is employed as a reference case. Case 2 has the force averaging scheme, while Case 3 does not. Fig. 7 compares the time evolution of the maximum Eulerian IB forces among the different cases, where $fb_{x,max}$ and $fb_{y,max}$ represent the maximum IB force in the x direction and y direction, respectively. It is clearly seen that the time series of IB forces in Case 3 deviate from those in Case 1 and Case 2 because the IB forces on the finest level in Case 3 are not averaged to the coarser levels to obtain a proper estimation. By using the force averaging scheme, the time evolution of the IB forces in Case 2 shows agreement with that in Case 1, and their difference in the early stage of simulation can be explained by the fact that it takes some time for the simulations on the finer level to be tightly coupled with those on the coarser level before the data across multiple levels match. In addition, the effectiveness of the force averaging scheme is also demonstrated through the velocity fields. As shown in Fig. 8, the contours of the velocity magnitude in Cases 1 and 2 are almost identical, whereas in Case 3, a spurious vortex is generated on the upper right side of the finest level. These results show the necessity and efficacy of using the force averaging scheme when a multilevel grid is employed in the simulations. Although only the subcycling case results are shown here, the non-subcycling case also exhibits a similar behavior.

After validating the force averaging scheme, we now assess the temporal and spatial–temporal convergence rates of our numerical scheme on a multilevel grid. For assessing the temporal convergence rate, the grid size is kept fixed. For the spatial–temporal convergence rate test, the CFL number is fixed and the grid sizes on level 0 are taken to be 32 × 32, 64 × 64, and 128 × 128. Both the subcycling and non-subcycling methods are assessed. Similar to the single-level case, we compare the composite solution with the reference solution at $tU/L = 30$ using the L^2-norm of the error. In Fig. 9, it can be observed that we obtain second-order convergence rate in time for both the non-subcycling and subcycling methods; the latter uses an additional midpoint time interpolation scheme of variables defined on different levels (Section 5.4.1). As seen in Fig. 10, the L^2 errors of $\mathbf{u}$ decrease as the grid resolution increases. The L^2 errors at a given grid spacing for the subcycling method and non-subcycling methods are comparable. The two adaptive methods also achieve a spatial–temporal convergence rate of approximately 1.44, which is nearly the same as those observed for the single-level cases previously. These tests prove the efficacy of the force averaging scheme and demonstrate that our numerical schemes can achieve the expected order of accuracy on (static) multilevel mesh.

6.2. Cylinder with prescribed motion

We first consider the flow past a cylinder oscillating about a mean position with a prescribed velocity. Specifically, the cylinder oscillates in the x direction with the prescribed velocity $U_c = U_{\max}\cos(2\pi t/T)$, where $U_{\max} = 1.0$ is the maximum oscillating velocity and T is the time period of the oscillation. To exclude wall effects, the size of the computational domain is set sufficiently large to $[-16D, 16D] \times [-8D, 8D]$. Here, D is the diameter of the cylinder. The center of the cylinder is located at $(0, 0)$. To match the previous studies by Shen et al. [86] and Bhalla et al. [4], the Keulegan–Carpenter number is set to $KC = U_{\max}T/D = 5$, and the Reynolds number is $Re = \rho_f U_{\max} D/\mu_f = 100$. Three cases are considered, as listed in Table 2. The refinement criterion is the distance to the fluid–solid interface indicated by the LS function ψ. Specifically, the grid cells (i, j) on level l $(0 \le l < l_{\max})$ are refined to the finer level if $|\psi_{i,j}| < 4.0\max(\Delta x^l, \Delta y^l)$, where Δx^l and Δy^l are the grid spacings in the x direction and y direction, respectively, on level l.

The left part of Fig. 11 compares the drag coefficient in the single-level case, defined as $C_d = F_x/(0.5\rho_f U_{\max}^2 D)$, with the results of Shen et al. [86] and Bhalla et al. [4]. We note that both Eqs. (30) and (43) are used to calculate the drag coefficient in this test problem. Eq. (30) only considers the constraint force, while Eq. (43) includes both the constraint force and inertial force. The right part of Fig. 11 shows the comparison of the drag coefficients among the single-level case (Case 1), three-level subcycling case (Case 2), and three-level non-subcycling case (Case 3). The agreement indicates that when inertial effects are present, our FSI algorithm can accurately estimate the hydrodynamic force on the solid surface for both the single-level case (Case 1) and multilevel case (Cases 2 and 3).

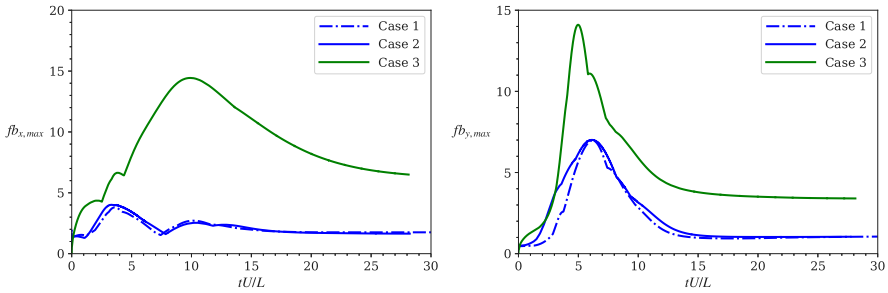

Fig. 7. Comparison of the time evolution of the maximum Eulerian IB force among the different cases listed in Table 1. $fb_{x,max}$ (left) and $fb_{y,max}$ (right) represent the maximum IB force in the x direction and y direction, respectively.

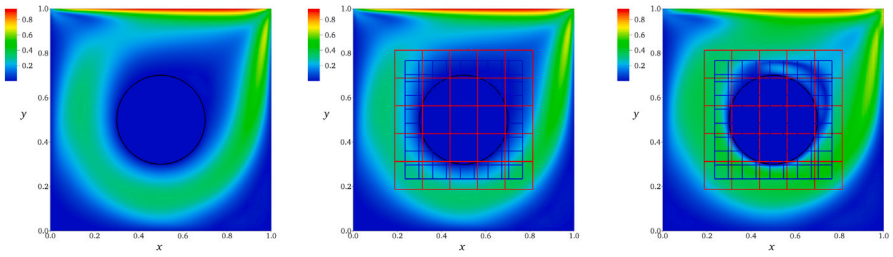

Fig. 8. Comparison of the velocity magnitudes for the cylinder submerged within a lid-driven cavity problem at $tU/L = 30$. Left: single-level case (Case 1); middle: three-level subcycling case (Case 2) with the force averaging scheme; right: three-level subcycling case (Case 3) without the force averaging scheme. Black lines: fluid–solid interface; red lines: patches on level 1; blue lines: patches on level 2.

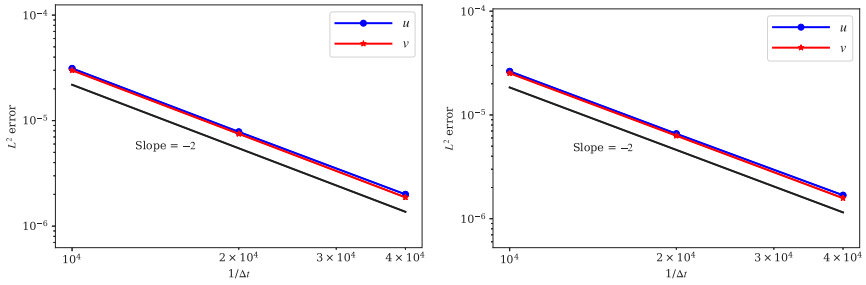

Fig. 9. Temporal convergence of u and v for the problem of a cylinder submerged in a lid-driven cavity with mesh refinement. The finest level is $l_{\max} = 2$. Left: subcycling method. Right: non-subcycling method.

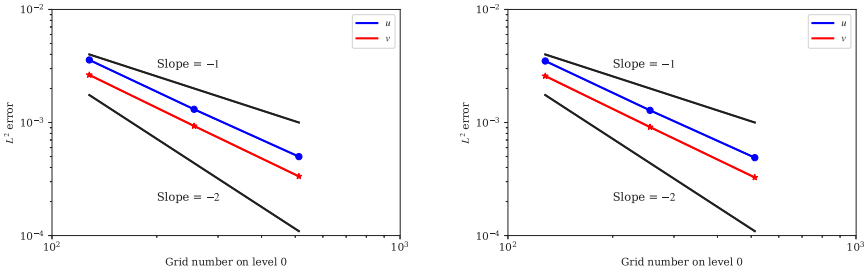

Fig. 10. Spatial–temporal convergence of u and v for the problem of a cylinder submerged in a lid-driven cavity with mesh refinement. The finest level is $l_{\max} = 2$. Left: subcycling method. Right: non-subcycling method.

6.3. Falling sphere in quiescent flow

In the test cases introduced above, the motion of the body was prescribed. In this section, the falling sphere problem is simulated to validate the DLM algorithm with a freely moving body. Here, we follow the same setup as [4]. The computational domain is $[-1\,\mathrm{m}, 1\,\mathrm{m}]\times[0\,\mathrm{m}, 8\,\mathrm{m}]\times[-1\,\mathrm{m}, 1\,\mathrm{m}]$, and the sphere with diameter $D = 0.625\,\mathrm{m}$ is initially centered at $(x, y, z) = (0\,\mathrm{m}, 7\,\mathrm{m}, 0\,\mathrm{m})$. The density of the fluid and the density of the solid are set to $\rho_\mathrm{f} = 2\,\mathrm{kg/m^3}$ and $\rho_\mathrm{s} = 3\,\mathrm{kg/m^3}$, respectively. The fluid viscosity is $\mu_\mathrm{f} = 0.05\,\mathrm{Pa\cdot s}$, and the Reynolds number is $Re = \rho_\mathrm{f} V D/\mu_\mathrm{f}$, where V is the terminal velocity of the falling sphere. No-slip boundary conditions are employed in all directions. Three cases are considered, as listed in Table 3.

When the sphere approaches the terminal state, its velocity in this case is $V = -1.24\,\mathrm{m/s}$, which agrees with prior studies [4,88]. Previous studies have shown that the normalized vertical velocity can be fit using an exponential curve as

$$v/V = 1 - e^{-3t/t_{95}}, \tag{47}$$

where t_{95} is the time required for the sphere to reach 95% of the terminal velocity V. The left part of Fig. 12 compares the time series of

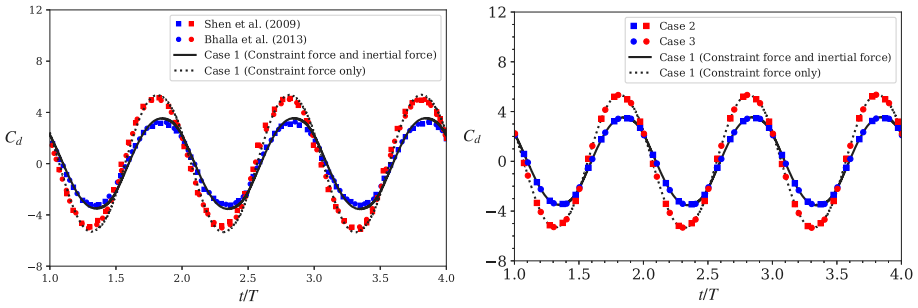

Fig. 11. Time series of the drag coefficient of an oscillating cylinder at $Re = 100$ and $KC = 5$. Left: comparison between the single-level case (Case 1) and previous results. Right: comparison among the single-level case (Case 1), the three-level subcycling case (Case 2), and the three-level non-subcycling case (Case 3). The dotted line, red circle (●), and red square (■) represent the constraint force calculated from Eq. (30). The solid line, blue circle (●), and blue square (■) represent the sum of both the inertial term and the constraint force obtained from Eq. (43).

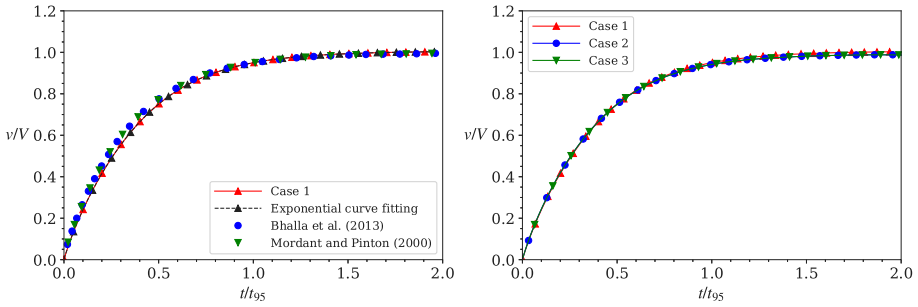

Fig. 12. Time series of the vertical velocity of a falling sphere at $Re = 31$. Left: comparison among the single-level case (Case 1), the exponential curve fitting using Eq. (47), and previous results [4,87]. Right: comparison among the single-level case (Case 1), three-level subcycling case (Case 2), and three-level non-subcycling case (Case 3).

Table 3
Parameters of the falling sphere in quiescent flow problem.

Case no.	Grid numbers on level 0	$l_{\max}$	Δt_0	Cycling methods
1	$64 \times 512 \times 64$	0	1×10^{-3}	–
2	$16 \times 128 \times 16$	2	4×10^{-3}	Subcycling
3	$16 \times 128 \times 16$	1	1×10^{-3}	Non-subcycling

Table 4
Parameters of the rotating cylinder in a shear flow problem.

Case no.	Grid numbers on level 0	$l_{\max}$	Δt_0
1	64×512	0	1×10^{-3}
2	16×128	1	1×10^{-3}
3	16×128	2	1×10^{-3}

the vertical velocity among the single-level case (Case 1), exponential curve fitting results using Eq. (47), and previous numerical [4] and experimental [87] results. The right part of Fig. 12 shows that the results of the multilevel cases (Cases 2 and 3) agree well with the single-level results. These results indicate that our algorithms can accurately capture the transient velocity of the falling sphere when it freely interacts with the surrounding fluid.

6.4. Rotating cylinder in a shear flow

This section considers a rotating cylinder in a shear flow to validate the multilevel DLM algorithm for problems with a rotational degree of freedom (DOF). As shown in Fig. 13, a 2D cylinder is initially immersed in a shear-driven channel flow with its centroid at $(L/4, H/4)$. The flow is driven by two moving plates with velocity U_w in opposite directions. No-slip and periodic boundary conditions are applied in the vertical direction and horizontal direction, respectively. Three non-subcycling cases are considered, as listed in Table 4, at Reynolds number $Re = 8\rho_f U_w D/\mu_f = 40$. The refinement criterion is based on the distance to the fluid–solid interface, i.e., the grid cells (i, j) on level l $(0 \le l < l_{\max})$ are refined to the next finer level if $|\psi_{i,j}| < 4.0 \max(\Delta x^l, \Delta y^l)$.

Fig. 14 shows the time evolution of the angular velocity ω_z and the vertical position of the cylinder centroid y_c. The angular velocity of the cylinder increases rapidly and then reaches a steady state at approximately $2U_w t/H = 200$ at a value of 0.47. This value is consistent with the results of Yeo et al. [89] and Bhalla et al. [4]. Similarly, the

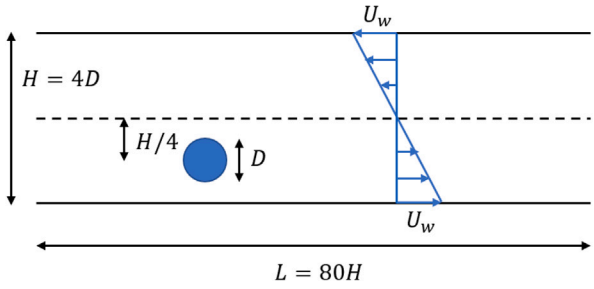

Fig. 13. Sketch of a two-dimensional rotating cylinder in a shear flow.

evolution of the vertical position of the cylinder y_c in the single-level case (Case 1) also agrees with previous studies. Fig. 15 compares the time series of ω_z and y_c between the single-level case (Case 1) and the multilevel non-subcycling cases (Cases 2 and 3). The agreement proves that the proposed multilevel algorithms can accurately simulate FSI problems with rotational degrees of freedom.

6.5. Oscillating cylinder in a spring–mass–damper system

This section considers an oscillating cylinder with different densities in a spring–mass–damper system, aimed at testing the FSI algorithms when external restoring and damping forces are applied to the solid body at each time step. The schematic of the numerical configuration

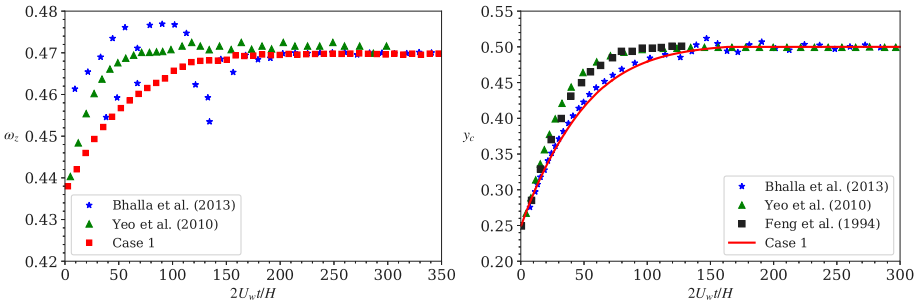

Fig. 14. Comparison of the angular velocity ω_z (left) and the vertical position of the cylinder centroid y_c (right) between the single-level case (Case 1) and previous results.

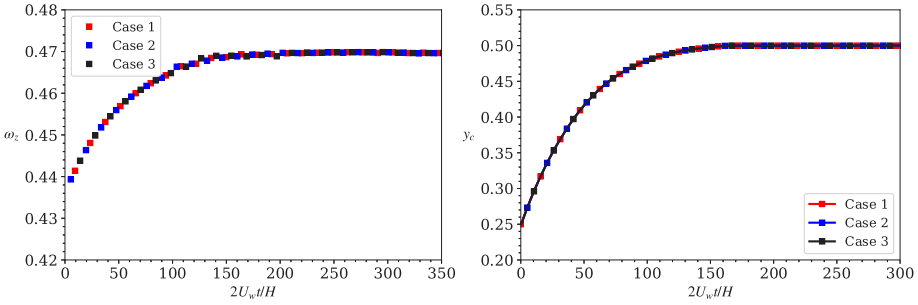

Fig. 15. Comparison of the angular velocity ω_z (left) and vertical position of the cylinder centroid y_c (right) between the single-level case (Case 1) and the multilevel non-subcycling cases (Cases 2 and 3).

is shown on the left part of Fig. 16, in which the computational domain is $[0, 5D] \times [0, 10D]$ and no-slip boundary conditions are utilized on all sides except for at the top boundary [90]. The cylinder is attached to a spring-damper system with a damping constant of b_s and a stiffness constant of k_s, and its initial centroid is located at $(X_0, Y_0) = (2.5D, 8D)$. The free length of the spring is set to $6.5D$, which means that the initial extension of the spring is $1.5D$. The density ratio between the solid and the fluid is $m^* = \rho_s/\rho_f$. Six cases are listed in Table 5, in which we choose two different density ratios, $m^* = 100$ and 0.8, corresponding to a structure in a gas and a structure in a liquid, respectively. For all these cases, we fix the diameter of the cylinder $D = 0.2$ m and spring constant $k_s = 500\,\mathrm{N/m}$. We consider the damper with different values of $b_s/b_{\mathrm{critical}} = \zeta$, where $b_{\mathrm{critical}} = 2\sqrt{k_s M_b}$. We note that M_b is the mass of the cylinder and ζ determines the behavior of the spring-damper system [91]: $\zeta < 1$ represents an underdamped system, $\zeta = 1$ leads to a critically damped system, and $\zeta > 1$ results in an overdamped system.

For the density ratio $m^* = 100$, the left part of Fig. 17 compares the time evolution of the vertical position of the cylinder centroid between the single-level case (Case 1) and the analytical solution. The analytical solution is obtained by disregarding the fluid forces on the solid body because these forces are small compared to the inertial force for this case with a large density ratio. Fig. 17 shows that our results are consistent with the analytical results. The right part of Fig. 17 compares Cases 1–3, which shows that both subcycling results and non-subcycling results match the single-level result. To visualize the flow field, the contours of the velocity vector at $t = 0.5$ and the patch hierarchy for Case 2 are plotted in the right part of Fig. 16.

For the low-density ratio $m^* = 0.8$, the fluid and solid have comparable densities, and the hydrodynamic forces cannot be disregarded, especially at low damping ratios. Although an analytical solution is not available for this density ratio, we compare our numerical results with those of Dafnakis et al. [92], in which a Brinkman penalization (BP) method is used to simulate the oscillating cylinder and WEC problems. As shown on the left part of Fig. 18, the numerical results of our single-level case (Case 4) using the DLM method agree with those using the BP method in [92], which proves the correctness and robustness of our algorithms at low-density ratios between the solid phase and fluid phase. The right part of Fig. 18 further compares the results between the single-level case (Case 4) and the multilevel cases (Cases 5 and 6), which shows the consistency in the numerical results regardless of the presence of AMR.

6.6. Wave energy converter (WEC)

In this section, a WEC simulation is conducted to evaluate the performance of our multilevel DLM algorithms in the presence of multiphase flows and external forces. Here, the LS-based two-phase flow solver [74] is coupled to the multilevel DLM algorithm to simulate the WEC dynamics. A schematic representation of the WEC problem is shown on the left part of Fig. 19. A water wave of height $H = 0.01\,\mathrm{m}$ and period $T = 0.8838\,\mathrm{s}$ is generated from the left boundary using Stokes wave theory. Refer to Appendix C and [92] for details. The wave then propagates to the right, transfers energy to the WEC, and is eventually dissipated in the damping zone at the right end of the domain. The wavelength is $\lambda = 1.216\,\mathrm{m}$, and the water depth is $d = 0.65\,\mathrm{m}$ [90,92]. A cylindrical-shaped WEC device with diameter $D = 0.16\,\mathrm{m}$ and density ratio $m^* = \rho_s/\rho_f = 0.9$ is initially submerged to a depth of $d_s = 0.25\,\mathrm{m}$ from the free water surface (19). The stiffness and damping constants of the power take-off (PTO) system are $k_s = 1995.2\,\mathrm{N/m}$ and $b_s = 80.64\,\mathrm{N\,s/m}$, respectively. Two multilevel computational cases are considered, as listed in Table 6. The air–water interface and water–solid interface are refined to the finest level (Fig. 19). In this work, the WEC device is only allowed to oscillate in the heave and surge directions, i.e., with two free DOFs, and the same k_s and b_s are applied to both directions.

Fig. 20 shows the heave and surge motions of the two-DOF WEC and its power generation during the steady state. Our multilevel results agree with the previous CFD simulation [92], and the slow drift phenomenon appearing in the surge dynamics is also well captured. Compared with the one-DOF results (not shown here), the two-DOF WEC slightly increases the power absorption, which shows agreement with the theoretical analysis of [93]. These results indicate that our multilevel DLM algorithm can accurately resolve the dynamics of a solid body when external forces are applied in a wave environment, for both the subcycling time advancement scheme and non-subcycling time advancement schemes.

Table 5
Parameters of the oscillating cylinder in a spring–mass–damper system problem.

Case no.	Density ratio m^*	Grid numbers on level 0	l_{max}	Δt_0	Cycling methods
1	100	256 × 512	0	1×10^{-3}	–
2	100	64 × 128	2	4×10^{-3}	Subcycling
3	100	64 × 128	2	1×10^{-3}	Non-subcycling
4	0.8	256 × 512	0	1×10^{-3}	–
5	0.8	64 × 128	2	4×10^{-3}	Subcycling
6	0.8	64 × 128	2	1×10^{-3}	Non-subcycling

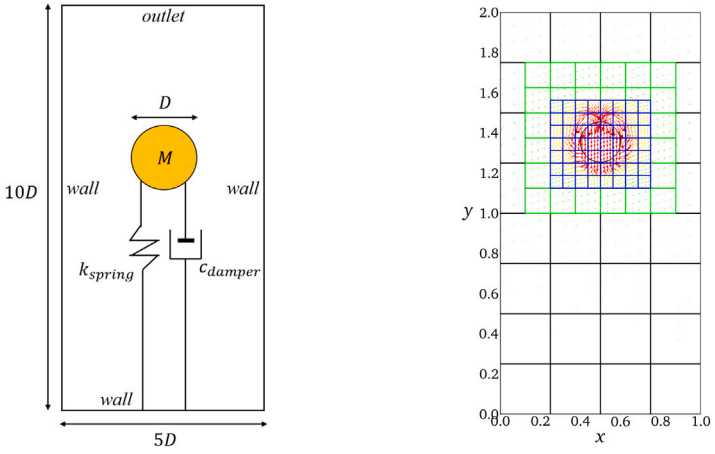

Fig. 16. Left: schematic of an oscillating cylinder in a spring-damper system. Right: velocity vectors around the cylinder in an underdamped regime ($\zeta = 0.5$) for the three-level subcycling case (Case 2). Black lines: patches on level 0; green lines: patches on level 1; blue lines: patches on level 2.

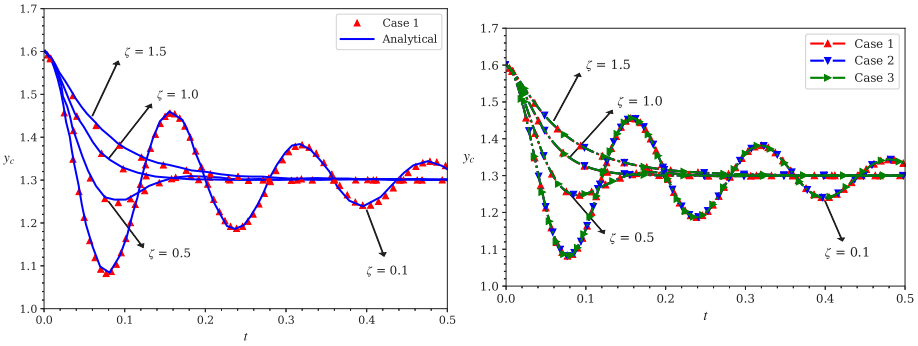

Fig. 17. Left: comparison of the time series of the vertical position of the cylinder centroid at $m^* = 100$ between the single-level case (Case 1) and the analytical solution. Right: comparison of the time series of the vertical position of the cylinder centroid among the single-level case (Case 1), three-level subcycling case (Case 2), and three-level non-subcycling case (Case 3). Various values of damping ratio ζ are considered.

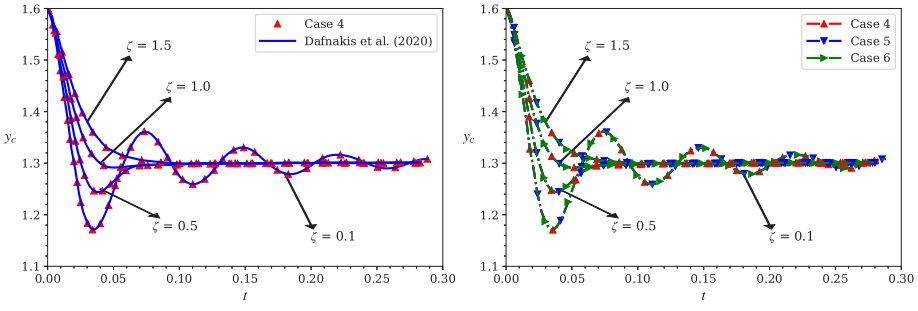

Fig. 18. Left: comparison of the time series of the vertical position of the cylinder centroid at $m^* = 0.8$ between the single-level case (Case 4) and [92]. Right: comparison of the time series of the vertical position of the cylinder centroid among the single-level case (Case 4), three-level subcycling case (Case 5), and three-level non-subcycling case (Case 6). Various values of damping ratio ζ are considered.

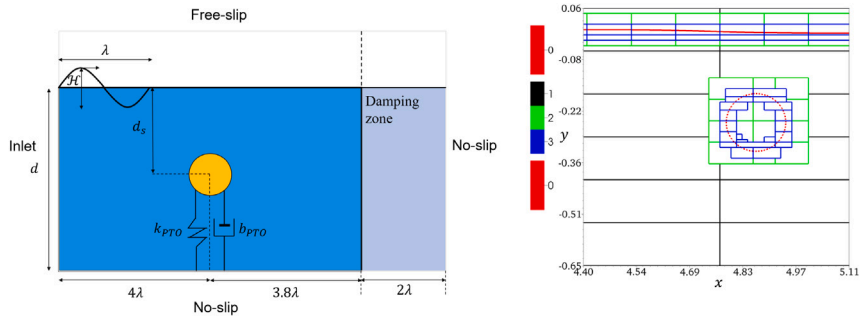

Fig. 19. Left: schematic of the submerged cylindrical WEC device. Right: patch hierarchy and contours among different phases. Red solid line: air–water interface; red dashed line: solid-water interface; black lines: patches on level 0; green lines: patches on level 1; blue lines: patches on level 2.

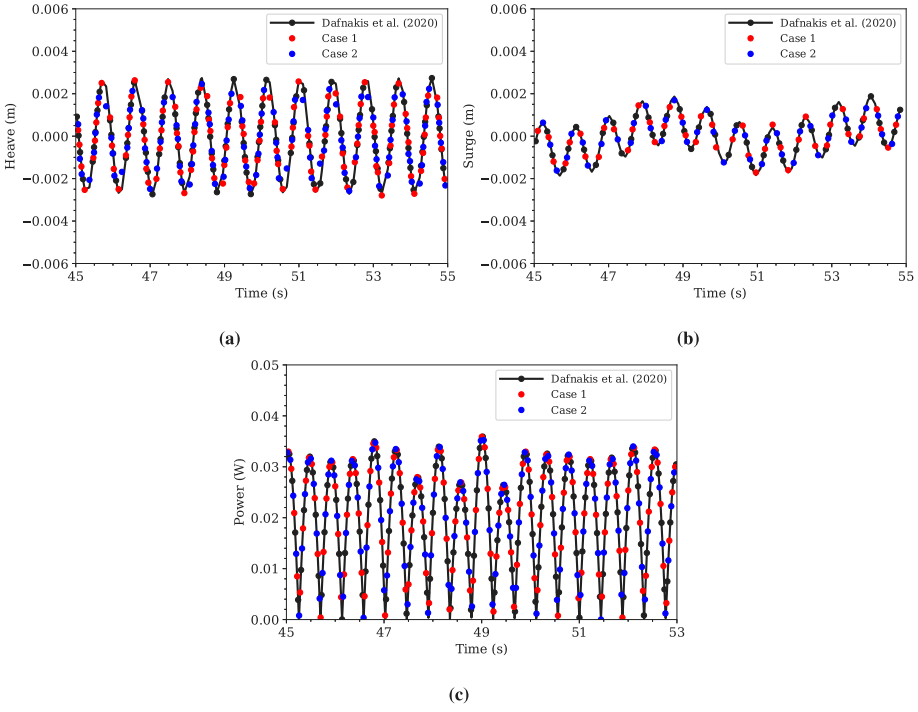

Fig. 20. Comparison of the dynamics of the two-DOF WEC among [92], the three-level subcycling case (Case 1), and the three-level non-subcycling case (Case 2). (a) Heave, (b) surge, and (c) generated power.

Table 6
Parameters of the WEC problem.

Case no.	Grid numbers on level 0	$l_{\max}$	Δt_0	Cycling methods
1	800 × 192	2	8×10^{-4}	Subcycling
2	800 × 192	2	2×10^{-4}	Non-subcycling

Table 7
Parameters of the cylinder splashing onto a two-fluid interface problem.

Case no.	Grid numbers on level 0	$l_{\max}$	Δt_0	Cycling methods
1	128 × 768	2	2×10^{-3}	Subcycling
2	128 × 768	2	5×10^{-4}	Non-subcycling

6.7. Cylinder splashing onto a two-fluid interface

To further validate our multilevel algorithms in the multiphase flow scenario, we consider a cylinder splashing onto a two-fluid interface, which is different from the single-phase falling sphere problem (Section 6.3). The computational domain is $[8D, 48D]$, the upper half of which from $y = 12D$ to $y = 24D$ is filled with a lighter fluid of density $\rho_g = 1\times10^3\,\mathrm{kg/m^3}$ and the lower half of which from $y = 0$ to $y = 12D$ is filled with a heavier fluid of density $\rho_l = 1.25\times10^3\,\mathrm{kg/m^3}$. No-slip boundary conditions are imposed on all sides. A circular cylinder with diameter $D = 2.5\times10^{-3}$ m and density $\rho_s = 1.5\times10^3\,\mathrm{kg/m^3}$ is initially placed at location $[4D, 40D]$. For this FSI problem, two cases using either the subcycling or non-subcycling method are considered, as listed in Table 7.

Fig. 21 shows the time evolution of the density contours of the three phases. The dimensionless time is defined as $T = t\sqrt{g/D}$. As plotted, a cavity forms in the wake of the cylinder as it penetrates the two-fluid interface. As the cavity collapses, a jet forms and breaks up into small droplets. We note that both the cavity and jet dynamics remain approximately symmetric in our simulations due to the initial symmetry of the problem setup. Symmetrical results are also observed in [64].

In Appendix B, we assess the conservation of a passive scalar in the inviscid shear layer. Here, instead, we assess the conservation of mass by our multiphase FSI solver. This is done by tracking the normalized fluid mass $m^\star(t) = |m(t)|/|m(0)|$ as a function of time for the cylinder splashing onto the two-fluid interface case. Here, $m(t) = \int \rho(\mathbf{x}, t)\, dV$ is the total fluid mass in the domain. Fig. 22 plots the time evolution of $m^\star$ over the course of the entire simulation. The results indicate that the

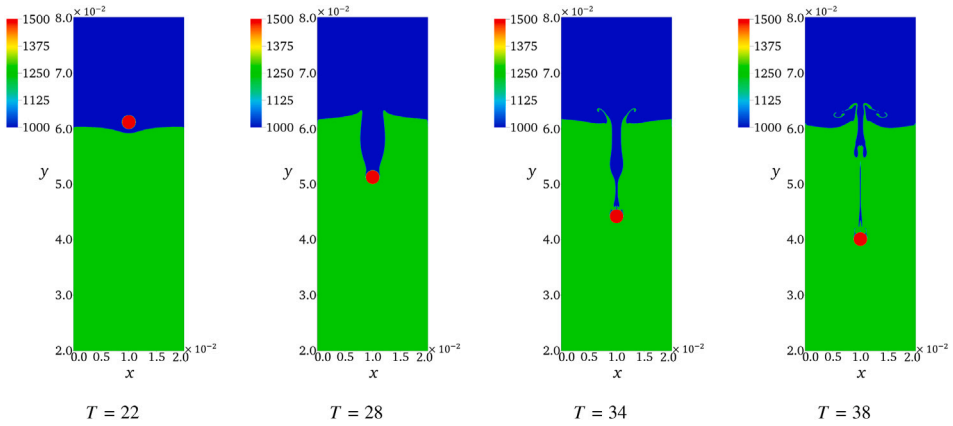

Fig. 21. Evolution of the density field when a cylinder falls into a column containing two fluids at different time instances. Only part of the y axis is plotted for better visualization.

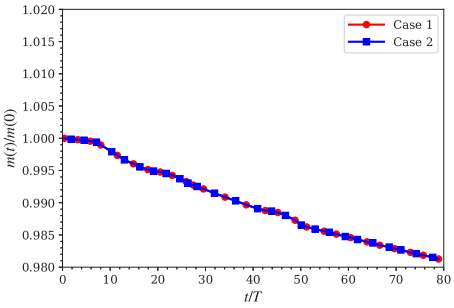

Fig. 22. Comparison of the time evolution of the normalized fluid mass between the subcycling method (Case 1) and the non-subcycling method (Case 2) for a 2D cylinder free-falling into a column containing two fluids.

total mass loss for Case 1 and 2 is less than 2%. We attribute the mass loss to the level set technique, which is known to be non-conservative in nature. Nevertheless, results of Fig. 22 indicate that our algorithm has a reasonably good performance in terms of conserving the total fluid mass for this complex problem including gas–liquid–solid interaction using both subcycling and non-subcycling methods.

To further quantitatively validate the accuracy of our algorithms, the time series of the vertical velocity and vertical position of the cylinder are plotted in Fig. 23. Our results show agreement with the studies of Nangia et al. [64], in which the same DLM approach was applied to capture the motion of the splashing cylinder. In summary, our simulations can accurately simulate the dynamics of the splashing cylinder and achieve good performance of the mass conservation in multiphase flow scenarios with both the subcycling time advancement scheme (Case 1) and the non-subcycling time advancement scheme (Case 2).

6.8. Two-dimensional self-propelled eel

In this section, we consider the two-dimensional self-propelled eel problem to validate the proposed algorithms when the geometry and deformation of the body are prescribed (Section 5.2.3). Eels are elongated ray-finned fish that belong to the order *Anguilliformes*. Fig. 24 sketches the two-dimensional swimming eel considered in this study. The eel has translation and deformation motions at $t > 0$ while interacting with the surrounding fluid. For this problem, the eel is assumed to move forward in the negative x direction, and thus, the local $h-\zeta$ coordinate, which is attached to the body of the eel, retains the same orientation as the fixed global $y-x$ coordinate. To describe the geometry of the fish, the vertical displacement of its middle line

Table 8
Parameters of the two-dimensional self-propelled eel problem.

Case no.	Grid numbers on level 0	$l_{\max}$	Cycling methods
1	4096 × 2048	0	–
2	512 × 256	3	Subcycling
3	512 × 256	3	Non-subcycling

$h(\zeta,t)$ is given by [4,54]

$$h(\zeta,t)=0.125L\frac{\zeta/L+0.03125}{1.03125}\sin\left[2\pi\left(\frac{\zeta}{L}-\frac{t}{T}\right)\right], \tag{48}$$

where L is the projected body length and T is the oscillating period of the tail tip. The cross-section of the fish body is centered about the middle line $h(\zeta,t)$, and its half width $r(\zeta)$ is given by

$$r(\zeta)=\begin{cases}\sqrt{2w_{\mathrm{H}}\zeta-\zeta^2} & \text{for } 0\leqslant\zeta<\zeta_{\mathrm{H}},\\ w_{\mathrm{H}}\frac{L-\zeta}{L-\zeta_{\mathrm{H}}} & \text{for } \zeta_{\mathrm{H}}\leqslant\zeta<L.\end{cases} \tag{49}$$

Here, $w_{\mathrm{H}}=0.04L$ is obtained from the observations reported in [4,94]. A periodic computational domain of size $8L\times4L$ is employed for the two-dimensional eel simulation. The Reynolds number is $Re=V_{\max}L/\mu_{\mathrm{f}}$, where $V_{\max}=0.785L/T$ is the maximum undulatory velocity of the tail tip. At $t=0$, the head of the eel is centered at $(6L,2L)$. The eel then moves forward along the negative x direction when $t>0$ because of self-propulsion. Other useful parameters for different cases are listed in Table 8. There are two refinement criteria in the eel simulation problem. The first criterion is the distance to the interface, i.e., the grid cells (i,j) on level l $(0\leq l<l_{\max})$ are refined to the finer level if $|\psi_{i,j}|<4.0\max(\Delta x^l,\Delta y^l)$, where Δx^l and Δy^l are the grid spacings in the x direction and y direction, respectively, on level l. The second criterion is based on the vorticity magnitude, in which grid cells are also tagged and refined to the next finer level if $|\omega|>0.7|\omega^{\max}|$. The refinement stops when $l_{\max}$ is reached.

For the two-dimensional eel simulation, the left part of Fig. 25 shows the results of the evolution of the forward-moving velocity, which agrees with the results of Bhalla et al. [4]. Fig. 26 shows vorticity contours associated with the swimming eel and the grid hierarchy at different time instants. Vortices are generated on both sides of the eel and shed from the tail tip in the form of a reverse Von Kármán vortex sheet. The frequency of vortex shedding corresponds to the oscillating frequency of undulation. The eel moves forward as a result of self-propulsion. The adaptive grid helps accurately resolve the fluid–solid interface and the essential flow features at a reduced computational cost. The Strouhal number is defined as $St=2A_{\mathrm{tail}}/TU_{\mathrm{steady}}$ [95], in which U_{steady} is the average forward speed of the eel in the quasi-steady state and A_{tail} is the amplitude of the tail tip flapping. In our simulations, we obtain $St=0.41$, which is in the range of experimentally observed St data for eels [96].

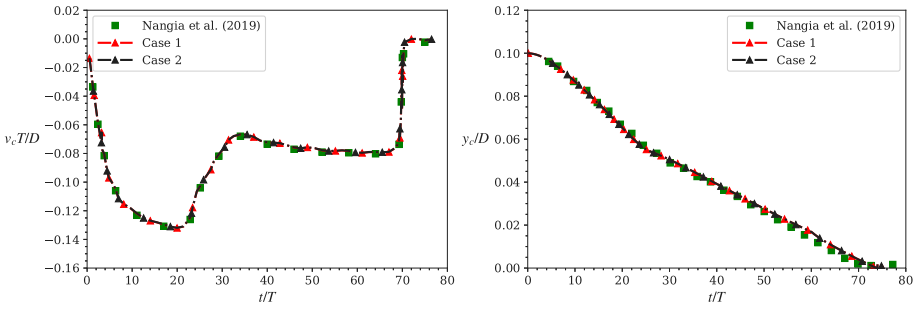

Fig. 23. Time series of vertical velocity (left) and vertical position (right) for a 2D cylinder free-falling into a column containing two fluids. The simulation results are compared with those of [64].

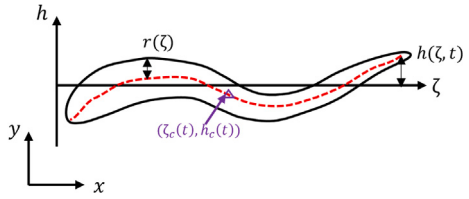

Fig. 24. Sketch of a two-dimensional swimming eel. Here, $(\zeta_c(t), h_c(t))$ is the centroid of the eel, $h(\zeta, t)$ is the vertical displacement of its middle line (red dashed line), and $r(\zeta)$ is the half width of the cross-section centered at the middle line.

In summary, our adaptive algorithms can accurately capture the dynamics of swimming eels with prescribed geometry and deformation.

6.9. Three-dimensional self-propelled eel

This section investigates a three-dimensional, self-propelled swimming eel, a dynamic and complex problem that is considered to be computationally expensive. In addition to validating the adaptive DLM algorithms for 3D problems with prescribed deformations, another objective of this test is to compare the computational cost of the single-level, subcycling, and non-subcycling cases. The simulation parameters and refinement criteria for the three-dimensional eel problem are the same as those for the two-dimensional eel problem of Section 6.8, except that the geometry of its cross-section is an ellipsoid with semimajor and semiminor axis lengths of $a = 0.51L$ and $b = 0.08L$, respectively. The height $h(s)$ is set to

$$h(\zeta) = b\sqrt{1 - \left(\frac{\zeta - a}{a}\right)^2}. \tag{50}$$

A periodic computational domain of size $8L \times 4L \times 4L$ is employed in the simulation. The head of the 3D eel at $t = 0$ is centered at $(6L, 2L, 2L)$. Three cases listed in Table 9 are considered. As seen from the right part of Fig. 25, we obtain agreement with the literature on the forward swimming velocity of the eel. The left part of Fig. 27 shows the vorticity contour and grid hierarchy of the four-level subcycling case (Case 2) at $Re = 5609$. The flow features are consistent with those reported in [94].

To compare the computational cost for different cases, we profile each case for $t/T = 0 - 0.1$ on the Cray XC40/50 (Onyx) system at the U.S. Army Engineer Research and Development Center, excluding the I/O costs. Table 10 shows the total number of grid cells for different cases at $t/T = 0.1$. Compared with the adaptive cases with $l_{\max} = 3$ (Cases 2 and 3), the single-level case (Case 1) has nearly 11.42 times more cells, i.e., the adaptive refinement considerably reduces the total number of grid cells.

The right part of Fig. 27 compares the wall clock time between the single-level case and the multilevel case for the time range $t/T = 0-0.1$. Compared with the single-level case (Case 1), the four-level subcycling case (Case 2) achieves more than a 20× speedup in terms of the wall clock time, which significantly saves the computational cost of the 3D simulation. By comparing the non-subcycling case (Case 3) with the subcycling case (Case 2), we find that the subcycling case further lowers the computational cost by a factor of 1.5. The reason is that, compared to the non-subcycling method, the subcycling method uses a larger time step size for the coarser levels.

Table 9
Parameters of the three-dimensional eel problem.

Case no.	Grid numbers on level 0	$l_{\max}$	Cycling methods
1	$1536 \times 512 \times 256$	0	–
2	$192 \times 64 \times 32$	3	Subcycling
3	$192 \times 64 \times 32$	3	Non-subcycling

In addition to the total wall clock time, the wall clock time spent on some key parts of the algorithm is also documented, including the MAC projection, viscous solver, level projection, synchronization, and DLM algorithm. Among them, the DLM algorithm, which includes the initialization and redistribution of the Lagrangian markers and the time advancement of the solid structure, is the most time-consuming operation due to the suboptimal parallelization of the Lagrangian markers while using the domain decomposition technique for the background (Cartesian) grid. The second most time-consuming part and third most time-consuming part are the level projection and MAC projection, respectively, in which the Poisson equation needs to be solved on the multilevel grid. The optimization of these three parts is beyond the scope of this work and deferred to future studies.

7. Conclusions

In this work, we have established a novel adaptive distributed Lagrangian multiplier (DLM) framework with subcycling and non-subcycling time advancement methods for simulating fluid–structure interaction (FSI) problems on adaptively refined grids. The four main contributions of this work are summarized at the end of Section 1. We note that the proposed multilevel advancement algorithm uses the level-by-level advancement technique for time-marching the variables in valid and invalid regions of the adaptive mesh refinement (AMR) hierarchy and decouples the time advancement at different levels. Because of this decoupling, the time step constraint on the coarser levels is relaxed compared to the finer levels when the subcycling method is applied. On the other hand, the non-subcycling method avoids the time interpolation process across different levels because data on all levels are located at the same time instant during the simulation.

We also developed a force-averaging algorithm to maintain the consistency of Eulerian immersed boundary (IB) forces across multiple levels. The efficacy of the force averaging algorithm is validated using the lid-driven cavity with a submerged cylinder problem, in which the expected order of the convergence rate is obtained for the multilevel

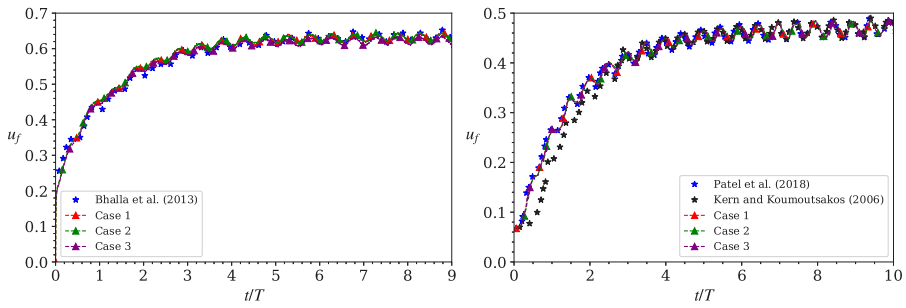

Fig. 25. Comparison of the evolution of the forward velocity u_f of the cases for the two-dimensional eel problem (left) and the three-dimensional eel problem (right). Case 1: single-level case; Case 2: four-level subcycling case; Case 3: four-level non-subcycling case.

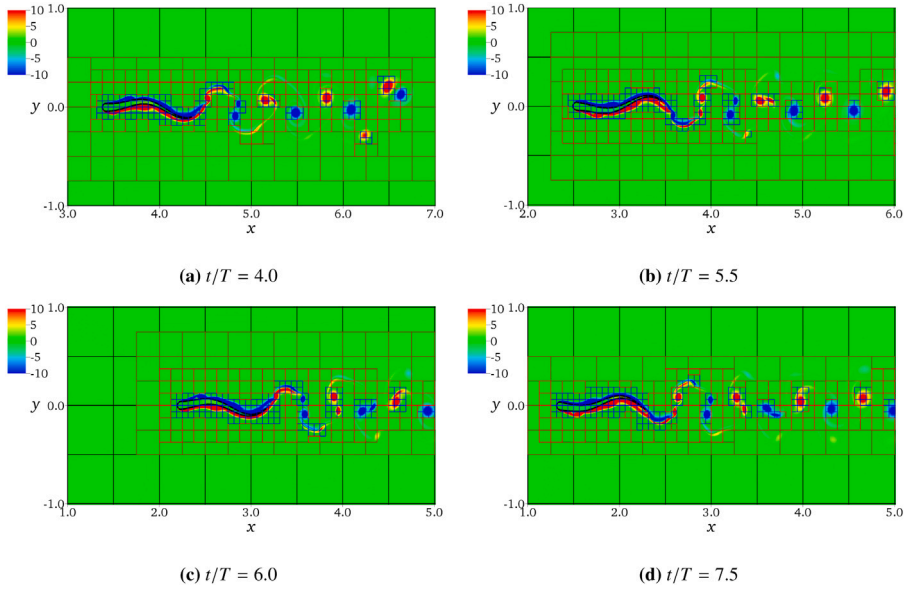

Fig. 26. Vorticity contours of the four-level subcycling case (Case 2) at $Re = 5609$ for the two-dimensional eel problem. Black patches: level 0; brown patches: level 1; red patches: level 2; blue patches: level 3.

Table 10
Number of grid cells for the 3D self-propelled eel problem at $t/T = 0.1$.

Case no.	Level 0 cells	Level 1 cells	Level 2 cells	Level 3 cells	Total cells
1	201,326,592	–	–	–	201,326,592
2	393,216	1,081,344	3,932,160	12,222,464	17,629,184
3	393,216	1,081,344	3,932,160	12,222,464	17,629,184

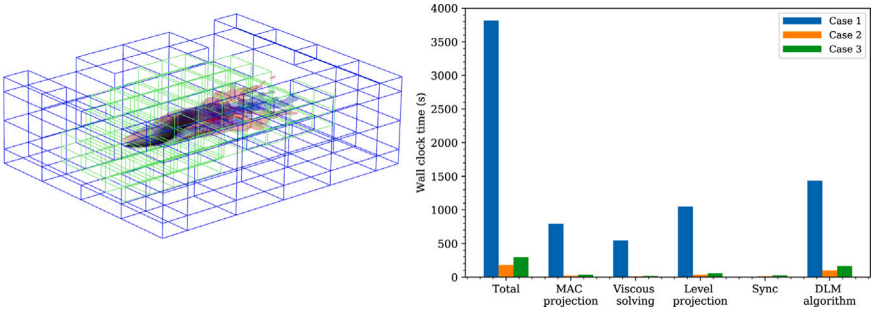

Fig. 27. Left: vorticity contour ($|\omega| = 1.8$) and grid hierarchy of the four-level subcycling case (Case 2) at $Re = 5609$ for the three-dimensional eel problem. Blue lines: patches on level 2; green lines: patches on level 3. Right: comparison of the wall clock time of key advancing steps among the single-level case (Case 1), four-level subcycling case (Case 2), and four-level non-subcycling case (Case 3).

cases. When a fine level catches up with a coarse level, synchronization operations are applied to represent the composite solution during the level-by-level advancement. As shown in the shear layer problem in Appendix B, the MAC synchronization and refluxing operation ensures mass and momentum conservation of the entire flow field by correcting the multilevel solution using velocity and flux registers.

The accuracy and robustness of the computational framework were validated using several canonical test problems. The results show that our multilevel numerical schemes can simulate various FSI problems with different types of solid constraints, including prescribed motion, free motion, and prescribed shape change. The subcycling and non-subcycling methods produced consistent and accurate results for all of these problems. We also combined the DLM algorithm with our previous two-phase flow solver and incorporated external spring-damper forces into the simulation. This approach enabled us to simulate the wave energy converter (WEC) problem.

Finally, we demonstrated that multilevel simulation can achieve the same level of accuracy with substantially fewer grid cells compared to the single-level fine-grid simulation. In particular, for the 3D swimming eel problem, the multilevel simulation is able to accurately capture the forward velocity with a nearly 20× speedup compared to the single-level simulation. In summary, we conclude that our proposed BSAMR framework is promising for high-fidelity and computationally demanding single-phase and multiphase fluid–structure interaction problems.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Y.Z. and L.S. gratefully acknowledge the support from Office of Naval Research awards N00014-17-1-2658 and N00014-19-1-2139. A.P.S.B. acknowledges support from National Science Foundation, USA award OAC-1931368. Y.Z. is grateful to Lawrence Berkeley National Lab researchers for their discussions related to the synchronization algorithms.

Appendix A. Discretization of the advection terms

We use the Godunov scheme [75,97] for discretization of the advection terms $[\nabla\cdot(\mathbf{u}\psi)]^{n+1/2}$ in Eq. (15) and $\nabla\cdot(\mathbf{u}\mathbf{u})^{n+1/2}$ in Eq. (25). This scheme is robust for a wide range of Reynolds numbers. There are four substeps to calculate the advection terms:

1. The unsplit Godunov approach is utilized to estimate the edge-centered velocity ($\mathbf{u}^{n+1/2,L}$, $\mathbf{u}^{n+1/2,R}$) and edge-centered LS function ($\psi^{n+1/2,L}$, $\psi^{n+1/2,R}$) at the middle time step $t^{n+1/2}$ on the edges perpendicular to the x direction. The superscripts ^{L} and ^{R} indicate that the edge-centered values are approximated on the left edge and right edge, respectively. The edge-centered velocity ($\mathbf{u}^{n+1/2,U}$, $\mathbf{u}^{n+1/2,D}$) and edge-centered LS function ($\psi^{n+1/2,U}$, $\psi^{n+1/2,D}$) can be calculated on the edges perpendicular to the y direction at the middle time step $t^{n+1/2}$, where the superscripts ^{U} and ^{D} denote that the edge-centered values are calculated from the up edge and down edge, respectively, of the computational cell.
2. The MAC projection [26,74,76] is then applied to obtain the divergence-free edge-centered advection velocity $\mathbf{u}^{\text{adv}}$.
3. The advection velocity $\mathbf{u}^{\text{adv}}$ is then used to calculate the edge-centered approximate state $\mathbf{u}^{n+1/2}$ and $\psi^{n+1/2}$ based on $\mathbf{u}^{n+1/2,L}$, $\mathbf{u}^{n+1/2,R}$, $\mathbf{u}^{n+1/2,U}$, $\mathbf{u}^{n+1/2,D}$, $\psi^{n+1/2,L}$, $\psi^{n+1/2,R}$, $\psi^{n+1/2,U}$, and $\psi^{n+1/2,D}$.
4. The advection velocity $\mathbf{u}^{\text{adv}}$ is then used to advect the approximate state $\mathbf{u}^{n+1/2}$ and $\psi^{n+1/2}$. The advection terms $\nabla\cdot(\mathbf{u}\mathbf{u})^{n+1/2}$ and $[\nabla\cdot(\mathbf{u}\psi)]^{n+1/2}$ are calculated as $\nabla\cdot(\mathbf{u}^{\text{adv}}\mathbf{u}^{n+1/2})$ and $\nabla\cdot(\mathbf{u}^{\text{adv}}\psi^{n+1/2})$.

We note that the calculation of the advection term $[\nabla\cdot(\mathbf{u}\phi)]^{n+1/2}$ in Eq. (16) is the same as that for $[\nabla\cdot(\mathbf{u}\psi)]^{n+1/2}$.

Appendix B. MAC synchronization and refluxing algorithm

The MAC synchronization and refluxing algorithm is applied as the second substep of the synchronization step, aimed at maintaining the conservation of momentum and scalar on the multilevel grid. Here, the differences between $\mathbf{u}^{\text{adv}}$ on the coarser level and the finer level are quantified during the single-level advancement (Section 5.1). These velocity differences, together with the flux differences, form the registers to make the corrections for each level. Specifically, the velocity registers that hold the difference in the edge-centered advection velocity are given by

$$\delta\mathbf{u}^{l} = -A^{l}\mathbf{u}^{\text{adv},l} + \frac{1}{2}\sum_{k=1}^{2}\sum_{faces} A^{l+1}\mathbf{u}^{\text{adv},k,l+1}. \tag{51}$$

In this equation, the superscript k represents the substeps of the finer level $l+1$ because it takes two substeps for level $l+1$ to catch up with level l in the subcycling method (Fig. 4), $\sum_{faces}$ is the sum over the cell faces, and A is the area of each face. The velocity flux registers, including both the advective flux register $\delta\mathbf{f}_{\mathbf{u}}^{\text{adv},l}$ and the viscous flux register $\delta\mathbf{f}_{\mathbf{u}}^{\text{visc},l}$, are defined in a similar way as

$$\delta\mathbf{f}_{\mathbf{u}}^{\text{adv},l} = \Delta t^{l}\left(A^{l}\mathbf{f}_{\mathbf{u}}^{\text{adv},l} + \frac{1}{2}\sum_{k=1}^{2}\sum_{faces} A^{l+1}\mathbf{f}_{\mathbf{u}}^{\text{adv},k,l+1}\right), \tag{52}$$

$$\delta\mathbf{f}_{\mathbf{u}}^{\text{visc},l} = \Delta t^{l}\left(A^{l}\mathbf{f}_{\mathbf{u}}^{\text{visc},l} + \frac{1}{2}\sum_{k=1}^{2}\sum_{faces} A^{l+1}\mathbf{f}_{\mathbf{u}}^{\text{visc},k,l+1}\right). \tag{53}$$

The LS function has the advective flux register $\delta\mathbf{f}_{\psi}^{\text{adv},l}$ only, which is calculated as

$$\delta\mathbf{f}_{\psi}^{\text{adv},l} = \Delta t^{l}\left(A^{l}\mathbf{f}_{\psi}^{\text{adv},l} + \frac{1}{2}\sum_{k=1}^{2}\sum_{faces} A^{l+1}\mathbf{f}_{\psi}^{\text{adv},k,l+1}\right). \tag{54}$$

In Eqs. (52)–(54), $\mathbf{f}_{\mathbf{u}}^{\text{adv},l}$, $\mathbf{f}_{\psi}^{\text{adv},l}$, and $\mathbf{f}_{\mathbf{u}}^{\text{visc},l}$ are given by

$$\mathbf{f}_{\mathbf{u}}^{\text{adv},l} = \mathbf{u}^{\text{adv}}\psi^{n+1/2}, \tag{55}$$

$$\mathbf{f}_{\psi}^{\text{adv},l} = \mathbf{u}^{\text{adv}}\psi^{n+1/2}, \tag{56}$$

$$\mathbf{f}_{\mathbf{u}}^{\text{visc},l} = \frac{1}{2Re}\left(\mu(\psi^{n,l})\nabla\mathbf{u}^{n,l} + \mu(\psi^{n+1,l})\nabla\mathbf{u}^{*,n+1,l}\right). \tag{57}$$

The mismatch of the velocity register in Eq. (51) forms the right-hand side of a MAC solution for the correction δe^{l} on level l,

$$\nabla\cdot\left(\frac{A^{l}}{\rho^{n+1/2,l}}\nabla(\delta e^{l})\right) = \nabla\cdot\delta\mathbf{u}^{l}. \tag{58}$$

After solving Eq. (58), a velocity correction $\mathbf{u}_{\text{corr}}^{l}$ is obtained by

$$\boldsymbol{u}_{\text{corr}}^{l} = \frac{-\nabla(\delta e^{l})}{\rho^{n+1/2,l}}. \tag{59}$$

The flux corrections associated with the above velocity correction are

$$\mathbf{f}_{\psi}^{\text{corr,l}} = \mathbf{u}_{\text{corr}}^{l}\psi^{n+1/2,l}, \tag{60}$$

$$\mathbf{f}_{\mathbf{u}}^{\text{corr,l}} = \mathbf{u}_{\text{corr}}^{l}\mathbf{u}^{n+1/2,l}. \tag{61}$$

The final correction to the LS function on level l, ψ_{sync}^{l}, is determined by the flux correction $\mathbf{f}_{\psi}^{\text{corr,l}}$ in Eq. (60) and the advective flux register $\delta\mathbf{f}_{\psi}^{\text{adv},l}$ in Eq. (54) as

$$\psi_{\text{sync}}^{l} = -\nabla\cdot\mathbf{f}_{\psi}^{\text{corr,l}} - \frac{\delta\mathbf{f}_{\psi}^{\text{adv},l}}{\Delta t\cdot Vol^{l}}, \tag{62}$$

where Vol^{l} is volume of the grid cell on level l, i.e., $Vol^{l} = \Delta x^{l}\Delta y^{l}$ for the 2D case and $Vol^{l} = \Delta x^{l}\Delta y^{l}\Delta z^{l}$ for the 3D case. The LS function on level l is then updated as

$$\psi^{n+1,l} := \psi^{n+1,l} + \Delta t^{l}\psi_{\text{sync}}^{l}. \tag{63}$$

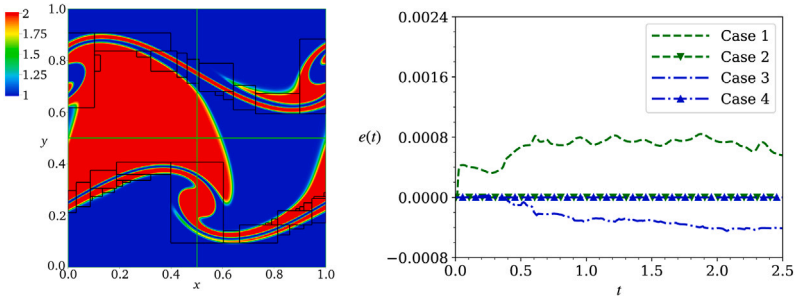

Fig. 28. Left: contours of the scalar field and grid hierarchy at $t = 0.8$ of the two-level subcycling case with dynamic mesh refinement with refluxing (Case 4) for the inviscid shear layer problem. Green line: patches on level 0; black line: patches on level 1. Right: comparison of the conservation errors of the passive scalar among the four cases of the inviscid shear layer problem. Case 1 and Case 2 use static refinement, while Case 3 and Case 4 use dynamic refinement. Case 2 and Case 4 consider refluxing, while Case 1 and Case 3 do not.

The flux correction about the velocity $\mathbf{f}_{\mathbf{u}}^{\text{corr,l}}$ in Eq. (61), together with its advective flux register $\delta\mathbf{f}_{\mathbf{u}}^{\text{adv},l}$ in Eq. (52) and viscous flux register $\delta\mathbf{f}_{\mathbf{u}}^{\text{adv},l}$ in Eq. (53), forms a subsequent parabolic equation,

$$\mathbf{u}_{\text{sync}}^{l} - \frac{\Delta t}{2\rho^{n+1/2,l}\,Re}\boldsymbol{\nabla}\cdot\left(\mu(\psi^{n+1})\boldsymbol{\nabla}\mathbf{u}_{\text{sync}}^{l}\right) = -\boldsymbol{\nabla}\cdot\mathbf{f}_{\mathbf{u}}^{\text{corr,l}} - \frac{1}{\Delta t\cdot Vol^{l}}\left(\delta\mathbf{f}_{\mathbf{u}}^{\text{adv},l} + \frac{1}{\rho^{n+1/2,l}}\delta\mathbf{f}_{\mathbf{u}}^{\text{adv},l}\right), \tag{64}$$

which gives the final correction of the velocity $\mathbf{u}_{\text{sync}}^{l}$ on level l. The updated velocity on level l is then given by

$$\mathbf{u}^{n+1,l} := \mathbf{u}^{n+1,l} + \Delta t^{l}\mathbf{u}_{\text{sync}}^{l}. \tag{65}$$

The corrections also need to propagate to all the finer levels q as

$$\psi^{n+1,q} := \psi^{n+1,q} + \Delta t^{l}\mathcal{I}_{\text{cons}}(\psi_{\text{sync}}^{l}) \tag{66}$$

and

$$\mathbf{u}^{n+1,q} := \mathbf{u}^{n+1,q} + \Delta t^{l}\mathcal{I}_{\text{cons}}(\mathbf{u}_{\text{sync}}^{l}) \tag{67}$$

for all $l < q \le l_{\max}$. Here, the conservative interpolation $\mathcal{I}_{\text{cons}}$ is performed.

For any level $l > 0$, the velocity registers and flux registers on the coarser level $l-1$ are affected by the abovementioned correction and thus need to be updated as follows:

$$\delta\mathbf{u}^{l-1} := \delta\mathbf{u}^{l-1} + \frac{1}{2}\sum_{faces}(A^{l}\mathbf{u}_{\text{corr}}^{l}), \tag{68}$$

$$\delta\mathbf{f}_{\mathbf{u}}^{\text{adv},l-1} := \delta\mathbf{f}_{\mathbf{u}}^{\text{adv},l-1} + \frac{1}{2}\Delta t^{l-1}\sum_{faces}(A^{l}\mathbf{f}_{\mathbf{u}}^{\text{corr,l}}), \tag{69}$$

$$\delta\mathbf{f}_{\mathbf{u}}^{\text{visc},l-1} := \delta\mathbf{f}_{\mathbf{u}}^{\text{visc},l-1} + \frac{1}{2}\Delta t^{l-1}\sum_{faces}\left(\frac{1}{2}A^{l}\mu(\psi^{n+1})\boldsymbol{\nabla}\boldsymbol{V}_{\text{sync}}^{l}\right), \tag{70}$$

$$\delta\mathbf{f}_{\psi}^{\text{adv},l-1} := \delta\mathbf{f}_{\psi}^{\text{adv},l-1} + \frac{1}{2}\sum_{faces}(A^{l}\mathbf{f}_{\psi}^{\text{corr,l}}). \tag{71}$$

As a reminder, the abovementioned MAC synchronization and refluxing substep is utilized to maintain the conservation of momentum and scalar in the whole flow field. To validate the efficacy of this substep, we assess the conservation of a passive scalar in the inviscid shear layer. Similar to the setup in Bell et al. [76], the computational domain is 1×1 with periodic boundary conditions in both the horizontal direction and vertical direction. The density of the inviscid fluid is $\rho_{\text{f}} = 1.0$, and the initial velocity is given by

$$u(x,y) = \begin{cases} \tanh(\sigma_1(y-0.25)) & y \le 0.5 \\ \tanh(\sigma_1(0.75-y)) & y > 0.5, \end{cases} \tag{72}$$

$$v(x,y) = \sigma_2\sin(2\pi x), \tag{73}$$

where $\sigma_1 = 30$ and $\sigma_2 = 0.05$. The grid size on level 0 is 100×100. A passive scalar advected by the abovementioned vortex pair is simulated. The initial scalar field is set to

$$s(x,y) = \begin{cases} 2.0, & \text{if } x \in [0.2,0.8] \text{ and } y \in [0.2,0.8], \\ 1.0, & \text{otherwise}. \end{cases} \tag{74}$$

Table 11
Parameters of the inviscid shear layer problem.

Case no.	Mesh refinement type	Is refluxing performed?
1	Static	No
2	Static	Yes
3	Dynamic	No
4	Dynamic	Yes

As the LS function is essentially a passive scalar governed by the advection equation, the simulation of s is carried out using Eq. (15). A total of four subcycling cases are considered, varying in the mesh refinement and whether the refluxing step is performed, as listed in Table 11. For mesh refinement, we consider static and dynamic refinement. For static refinement, grid cells are refined to $l_{\max} = 1$ in the rectangular region $x \in [0.2, 0.8]$ and $y \in [0.2, 0.8]$. For the dynamic refinement, the vorticity magnitude, $|\omega_z| > 0.75|\omega_z^{\max}|$, is used as the refinement criterion.

The vorticity field at $t = 0.8$ for the dynamic refinement case with refluxing (Case 4) is shown on the left part of Fig. 28. Because of the advection by the vortices, a high concentration of the scalar crosses the CF boundary, which can generate errors in the conservation of the scalar if the MAC synchronization and refluxing operations are not considered. To quantify this error, the relative change in the total amount of the scalar compared to the initial time is evaluated as

$$e(t) = \frac{\int_{\Omega}\left(s|_{t} - s|_{t=0}\right)dx}{\int_{\Omega} s|_{t=0}dx}. \tag{75}$$

The results for the abovementioned four cases are plotted in Fig. 28. When refluxing is used (Cases 2 and 4), the relative error is within 10^{-16} for both static refinement and dynamic refinement, while noticeable errors are present in simulations without refluxing (Cases 1 and 3). This test shows that MAC synchronization and refluxing operations are necessary and can help conserve the scalar.

Appendix C. Wave generation and wave absorption

This section presents the validation of the wave generation and wave absorption algorithms utilized for simulating the WEC problem in Section 6.6. These algorithms are only briefly introduced here; their numerical details can be referenced in [64,90,92]. In short, the horizontal and vertical velocity components are prescribed at the left boundary based on wave theory such that Stokes waves can be generated and propagate towards the right side. To mitigate the reflection of waves from the right boundary, a damping zone is placed at a downstream location to smoothly relax the velocities and LS function ϕ.

A 2D example is presented here for validating the wave generation and wave absorption algorithms. This example has the same computational parameters as the WEC problem of Section 6.6, except that the WEC device is not included in the simulation. Five cases are considered,

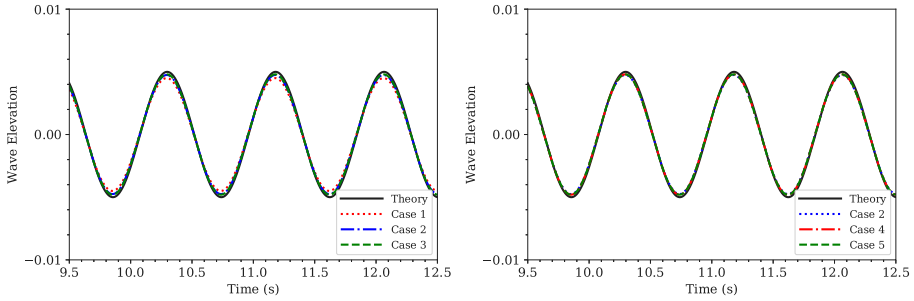

Fig. 29. Left: comparison of time series of the wave elevation at $x = 2.87\lambda$ among the cases with different grid sizes for the convergence study. Right: comparison of time series of the wave elevation at $x = 2.87\lambda$ among the single-level case (Case 2), the three-level subcycling case (Case 4), the three-level non-subcycling case (Case 5), and the theory.

Table 12
Parameters of the wave generation and wave absorption problem.

Case no.	Grid numbers on level 0	l_{max}	Δt_0	Cycling methods
1	1280 × 128	0	0.0002	–
2	2560 × 256	0	0.0001	–
3	5120 × 512	0	0.00005	–
4	640 × 64	2	0.0004	Subcycling
5	640 × 64	2	0.0001	Non-subcycling

as listed in Table 12. The first three single-level cases have different grid sizes and are employed to show the numerical convergence of the wave generation algorithms. As shown in the left part of Fig. 29, the simulations converge to the theoretical wave elevation using Stokes wave theory [98] as the grid resolution increases. We find that the resolution of Case 2 is sufficient to accurately capture the wave profile, in which there are approximately ten grid cells per wave height. Case 4 and Case 5 have the same resolution on their finest level as Case 2. From the right part of Fig. 29, it is seen that the results of these three cases are consistent with each other and agree with the theoretical result.

References

[1] Zhang Z, Kleinstreuer C. Airflow structures and nano-particle deposition in a human upper airway model. J Comput Phys 2004;198(1):178–210.
[2] Kamensky D, Hsu M-C, Schillinger D, Evans JA, Aggarwal A, Bazilevs Y, et al. An immersogeometric variational framework for fluid–structure interaction: Application to bioprosthetic heart valves. Comput Methods Appl Mech Eng 2015;284:1005–53.
[3] Kolahdouz EM, Bhalla APS, Craven BA, Griffith BE. An immersed interface method for discrete surfaces. J Comput Phys 2020;400:108854.
[4] Bhalla APS, Bale R, Griffith BE, Patankar NA. A unified mathematical framework and an adaptive numerical method for fluid–structure interaction with rigid, deforming, and elastic bodies. J Comput Phys 2013;250:446–76.
[5] Deng H-B, Xu Y-Q, Chen D-D, Dai H, Wu J, Tian F-B. On numerical modeling of animal swimming and flight. Comput Mech 2013;52(6):1221–42.
[6] Floreano D, Wood RJ. Science, technology and the future of small autonomous drones. Nature 2015;521(7553):460–6.
[7] Santhanakrishnan A, Robinson AK, Jones S, Low AA, Gadi S, Hedrick TL, et al. Clap and fling mechanism with interacting porous wings in tiny insect flight. J Exp Biol 2014;217(21):3898–909.
[8] Miller LA, Peskin CS. A computational fluid dynamics ofclap and fling'in the smallest insects. J Exp Biol 2005;208(2):195–212.
[9] Faltinsen O. Sea loads on ships and offshore structures, Vol. 1. Cambridge University Press; 1993.
[10] Trim A, Braaten H, Lie H, Tognarelli M. Experimental investigation of vortex-induced vibration of long marine risers. J Fluids Struct 2005;21(3):335–61.
[11] Song L, Fu S, Zeng Y, Chen Y. Hydrodynamic forces and coefficients on flexible risers undergoing vortex-induced vibrations in uniform flow. J Waterw Port Coast Ocean Eng 2016;142(4):04016001.
[12] Lou J, Johnston J, Tilton N. Application of projection and immersed boundary methods to simulating heat and mass transport in membrane distillation. Comput Fluids 2020;212:104711.
[13] Natarajan M, Sitaraman H, Ananthan S, Alan Sprague M. Actuator-line simulations of wind turbines with block-structured adaptive mesh refinement. In: APS division of fluid dynamics meeting abstracts. 2019, p. C09–008.
[14] Yu Y-H, Li Y. Reynolds-Averaged Navier–Stokes simulation of the heave performance of a two-body floating-point absorber wave energy system. Comput Fluids 2013;73:104–14.
[15] Khedkar K, Nangia N, Thirumalaisamy R, Bhalla APS. The inertial sea wave energy converter (ISWEC) technology: device-physics, multiphase modeling and simulations. Ocean Eng 2021;229:108879.
[16] Berger MJ, Oliger J. Adaptive mesh refinement for hyperbolic partial differential equations. J Comput Phys 1984;53(3):484–512.
[17] Berger MJ, Colella P. Local adaptive mesh refinement for shock hydrodynamics. J Comput Phys 1989;82(1):64–84.
[18] Cornford SL, Martin DF, Graves DT, Ranken DF, Le Brocq AM, Gladstone RM, et al. Adaptive mesh, finite volume modeling of marine ice sheets. J Comput Phys 2013;232(1):529–49.
[19] de Langavant CC, Guittet A, Theillard M, Temprano-Coleto F, Gibou F. Level-set simulations of soluble surfactant driven flows. J Comput Phys 2017;348:271–97.
[20] Goza A, Colonius T. A strongly-coupled immersed-boundary formulation for thin elastic structures. J Comput Phys 2017;336:401–11.
[21] Tian F-B. Role of mass on the stability of flag/flags in uniform flow. Appl Phys Lett 2013;103(3):034101.
[22] Chalamalla VK, Santilli E, Scotti A, Jalali M, Sarkar S. SOMAR-LES: A Framework for multi-scale modeling of turbulent stratified oceanic flows. Ocean Model 2017;120:101–19.
[23] Guittet A, Theillard M, Gibou F. A stable projection method for the incompressible Navier–Stokes equations on arbitrary geometries and adaptive Quad/Octrees. J Comput Phys 2015;292:215–38.
[24] Mirzadeh M, Guittet A, Burstedde C, Gibou F. Parallel level-set methods on adaptive tree-based grids. J Comput Phys 2016;322:345–64.
[25] Popinet S. Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries. J Comput Phys 2003;190(2):572–600.
[26] Almgren AS, Bell JB, Colella P, Howell LH, Welcome ML. A conservative adaptive projection method for the variable density incompressible Navier–Stokes equations. J Comput Phys 1998;142(1):1–46.
[27] Martin DF, Colella P. A cell-centered adaptive projection method for the incompressible Euler equations. J Comput Phys 2000;163(2):271–312.
[28] Martin DF, Colella P, Graves D. A cell-centered adaptive projection method for the incompressible Navier–Stokes equations in three dimensions. J Comput Phys 2008;227(3):1863–86.
[29] Minion ML. A projection method for locally refined grids. J Comput Phys 1996;127(1):158–78.
[30] Burstedde C, Wilcox LC, Ghattas O. p4est: scalable algorithms for parallel adaptive mesh refinement on forests of octrees. SIAM J Sci Comput 2011;33(3):1103–33.
[31] Isaac T, Burstedde C, Wilcox LC, Ghattas O. Recursive algorithms for distributed forests of octrees. SIAM J Sci Comput 2015;37(5):C497–531.
[32] Williamschen M, Groth CP. Parallel anisotropic block-based adaptive mesh refinement algorithm for three-dimensional flows. In: 21st AIAA computational fluid dynamics conference. 2013, p. 2442.
[33] Min C, Gibou F. A second order accurate level set method on non-graded adaptive cartesian grids. J Comput Phys 2007;225(1):300–21.
[34] Posa A, Vanella M, Balaras E. An adaptive reconstruction for Lagrangian, direct-forcing, immersed-boundary methods. J Comput Phys 2017;351:422–36.
[35] Vanella M, Rabenold P, Balaras E. A direct-forcing embedded-boundary method with adaptive mesh refinement for fluid–structure interaction problems. J Comput Phys 2010;229(18):6427–49.
[36] Gunney BT, Anderson RW. Advances in patch-based adaptive mesh refinement scalability. J Parallel Distrib Comput 2016;89:65–84.
[37] Natarajan M, Chiodi R, Kuhn M, Desjardins O. An all-mach multiphase flow solver using block-structured AMR. In: ILASS americas, 30th annual conference on liquid atomization and spray systems. may 12th-15th, tempe, Arizona, USA. 2019.
[38] Hu HH, Patankar NA, Zhu M. Direct numerical simulations of fluid–solid systems using the arbitrary Lagrangian–Eulerian technique. J Comput Phys 2001;169(2):427–62.

[39] Ramaswamy B, Kawahara M. Arbitrary Lagrangian–Eulerian finite element method for unsteady, convective, incompressible viscous free surface fluid flow. Internat J Numer Methods Fluids 1987;7(10):1053–75.

[40] Sotiropoulos F, Yang X. Immersed boundary methods for simulating fluid–structure interaction. Prog Aerosp Sci 2014;65:1–21.

[41] Tian F-B, Dai H, Luo H, Doyle JF, Rousseau B. Fluid–structure interaction involving large deformations: 3D simulations and applications to biological systems. J Comput Phys 2014;258:451–69.

[42] Mittal R, Iaccarino G. Immersed boundary methods. Annu Rev Fluid Mech 2005;37(1):239–61.

[43] Balaras E. Modeling complex boundaries using an external force field on fixed cartesian grids in large-eddy simulations. Comput Fluids 2004;33(3):375–404.

[44] Peskin CS. Flow patterns around heart valves: A numerical method. J Comput Phys 1972;10(2):252–71.

[45] Peskin CS. The immersed boundary method. Acta Numer 2002;11:479–517.

[46] Fadlun E, Verzicco R, Orlandi P, Mohd-Yusof J. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. J Comput Phys 2000;161(1):35–60.

[47] Yang X, Zhang X, Li Z, He G-W. A smoothing technique for discrete delta functions with application to immersed boundary method in moving boundary simulations. J Comput Phys 2009;228(20):7821–36.

[48] He S, Yang Z, Sotiropoulos F, Shen L. Numerical simulation of interaction between multiphase flows and thin flexible structures. J Comput Phys 2022;448:110691.

[49] Patankar NA, Singh P, Joseph DD, Glowinski R, Pan T-W. A new formulation of the distributed Lagrange multiplier/fictitious domain method for particulate flows. Int J Multiph Flow 2000;26(9):1509–24.

[50] Udaykumar H, Shyy W, Rao M. Elafint: a mixed Eulerian–Lagrangian method for fluid flows with complex and moving boundaries. Internat J Numer Methods Fluids 1996;22(8):691–712.

[51] Kim J, Kim D, Choi H. An immersed-boundary finite-volume method for simulations of flow in complex geometries. J Comput Phys 2001;171(1):132–50.

[52] Yang J, Balaras E. An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries. J Comput Phys 2006;215(1):12–40.

[53] Gilmanov A, Le TB, Sotiropoulos F. A numerical approach for simulating fluid structure interaction of flexible thin shells undergoing arbitrarily large deformations in complex domains. J Comput Phys 2015;300:814–43.

[54] Cui Z, Yang Z, Jiang H-Z, Huang W-X, Shen L. A sharp-interface immersed boundary method for simulating incompressible flows with arbitrarily deforming smooth boundaries. Int J Comput Methods 2018;15(01):1750080.

[55] Roman F, Armenio V, Fröhlich J. A simple wall-layer model for large eddy simulation with immersed boundary method. Phys Fluids 2009;21(10):10–4.

[56] Kang S. An improved near-wall modeling for large-eddy simulation using immersed boundary methods. Internat J Numer Methods Fluids 2015;78(2):76–88.

[57] Zeng Y, Bhalla AP, He S, Shen L. A subcycling/non-subcycling time advancement scheme-based sharp-interface immersed boundary method framework for solving fluid-structure interaction problems on dynamically adaptive grids. In: APS division of fluid dynamics meeting abstracts. 2021, p. F26–004.

[58] Ye T, Mittal R, Udaykumar H, Shyy W. An accurate cartesian grid method for viscous incompressible flows with complex immersed boundaries. J Comput Phys 1999;156(2):209–40.

[59] Almgren AS, Bell JB, Colella P, Marthaler T. A cartesian grid projection method for the incompressible Euler equations in complex geometries. SIAM J Sci Comput 1997;18(5):1289–309.

[60] Liu Q, Vasilyev OV. A Brinkman penalization method for compressible flows in complex geometries. J Comput Phys 2007;227(2):946–66.

[61] Roma AM, Peskin CS, Berger MJ. An adaptive version of the immersed boundary method. J Comput Phys 1999;153(2):509–34.

[62] Griffith BE, Hornung RD, McQueen DM, Peskin CS. An adaptive, formally second order accurate version of the immersed boundary method. J Comput Phys 2007;223(1):10–49.

[63] Vanella M, Balaras E. A moving-least-squares reconstruction for embedded-boundary formulations. J Comput Phys 2009;228(18):6617–28.

[64] Nangia N, Patankar NA, Bhalla APS. A DLM immersed boundary method based wave-structure interaction solver for high density ratio multiphase flows. J Comput Phys 2019;398:108804.

[65] Nangia N, Griffith BE, Patankar NA, Bhalla APS. A robust incompressible Navier–Stokes solver for high density ratio multiphase flows. J Comput Phys 2019;390:548–94.

[66] Delaney K. An adaptive mesh refinement solver for multiphase incompressible flows with large density ratios. (Ph.D. thesis), The George Washington University; 2014.

[67] Pivello MR, Villar MM, Serfaty R, Roma AM, Silveira-Neto Ad. A fully adaptive front tracking method for the simulation of two phase flows. Int J Multiph Flow 2014;58:72–82.

[68] Zhang W, Almgren A, Beckner V, Bell J, Blaschke J, Chan C, et al. AMReX: a framework for block-structured adaptive mesh refinement. J Open Source Softw 2019;4(37).

[69] Zeng Y, Xuan A, Blaschke J, Shen L. A parallel cell-centered adaptive level set framework for efficient simulation of two-phase flows with subcycling and non-subcycling. J Comput Phys 2022;448:110740.

[70] Zingale M. Introduction to computational astrophysical hydrodynamics. Open-Astrophys-Bookshelf 2014;13:129–37.

[71] Rider WJ. Approximate projection methods for incompressible flow: Implementation, variants and robustness. LANL unclassified report LA-UR-94-2000, Los Alamos National Laboratory; 1995.

[72] Rider WJ. Filtering non-solenoidal modes in numerical solutions of incompressible flows. Internat J Numer Methods Fluids 1998;28(5):789–814.

[73] Almgren AS, Bell JB, Crutchfield WY. Approximate projection methods: Part I. Inviscid analysis. SIAM J Sci Comput 2000;22(4):1139–59.

[74] Sussman M, Almgren AS, Bell JB, Colella P, Howell LH, Welcome ML. An adaptive level set approach for incompressible two-phase flows. J Comput Phys 1999;148(1):81–124.

[75] Colella P. A multidimensional second order Godunov scheme for conservation laws. J Comput Phys 1990;87:171–200.

[76] Bell JB, Colella P, Glaz HM. A second-order projection method for the incompressible Navier–Stokes equations. J Comput Phys 1989;85(2):257–83.

[77] Bhalla APS, Nangia N, Dafnakis P, Bracco G, Mattiazzo G. Simulating water-entry/exit problems using Eulerian–Lagrangian and fully-Eulerian fictitious domain methods within the open-source IBAMR library. Appl Ocean Res 2020;94:101932.

[78] Shapiro V. Semi-analytic geometry with r-functions. Acta Numerica 2007: Volume 16 2007;16:239–303.

[79] Sussman M, Smereka P. Axisymmetric free boundary problems. J Fluid Mech 1997;341:269–94.

[80] Sussman M, Fatemi E. An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow. SIAM J Sci Comput 1999;20(4):1165–91.

[81] Guy RD, Fogelson AL. Stability of approximate projection methods on cell-centered grids. J Comput Phys 2005;203(2):517–38.

[82] Zeng Y, Shen L. A unified AMR framework for multiphase flow and fluid-structure interaction problems with both non-subcycling and subcycling. In: APS division of fluid dynamics meeting abstracts. 2019, p. S19–001.

[83] Borazjani I, Ge L, Sotiropoulos F. Curvilinear immersed boundary method for simulating fluid structure interaction with complex 3D rigid bodies. J Comput Phys 2008;227(16):7587–620.

[84] Calderer A, Kang S, Sotiropoulos F. Level set immersed boundary method for coupled simulation of air/water interaction with complex floating structures. J Comput Phys 2014;277:201–27.

[85] Jacob F, Ted B. a first course in finite elements. Wiley; 2007.

[86] Shen L, Chan E-S, Lin P. Calculation of hydrodynamic forces acting on a submerged moving object using immersed boundary method. Comput Fluids 2009;38(3):691–702.

[87] Mordant N, Pinton J-F. Velocity measurement of a settling sphere. Eur Phys J B 2000;18(2):343–52.

[88] Sharma N, Patankar NA. A fast computation technique for the direct numerical simulation of rigid particulate flows. J Comput Phys 2005;205(2):439–57.

[89] Yeo K, Ang S, Shu C. Simulation of fish swimming and manoeuvring by an SVD-gfd method on a hybrid meshfree-Cartesian grid. Comput Fluids 2010;39(3):403–30.

[90] Zeng Y, Shen L. Modelling wave energy converter (WEC) pointer absorbers using AMR techniques with both subcycling and non-subcycling. Earth Space Sci Open Arch 2020;1.

[91] Nayfeh AH, Pai PF. Linear and nonlinear structural mechanics. John Wiley & Sons; 2008.

[92] Dafnakis P, Bhalla APS, Sirigu SA, Bonfanti M, Bracco G, Mattiazzo G. Comparison of wave–structure interaction dynamics of a submerged cylindrical point absorber with three degrees of freedom using potential flow and computational fluid dynamics models. Phys Fluids 2020;32(9):093307.

[93] Falnes J, Kurniawan A. Ocean waves and oscillating systems: linear interactions including wave-energy extraction, Vol. 8. Cambridge University Press; 2020.

[94] Kern S, Koumoutsakos P. Simulations of optimized anguilliform swimming. J Exp Biol 2006;209(24):4841–57.

[95] Triantafyllou GS, Triantafyllou M, Grosenbaugh M. Optimal thrust development in oscillating foils with application to fish propulsion. J Fluids Struct 1993;7(2):205–24.

[96] Tytell ED. The hydrodynamics of eel swimming II. Effect of swimming speed. J Exp Biol 2004;207(19):3265–79.

[97] Colella P. A direct Eulerian MUSCL scheme for gas dynamics. SIAM J Sci Comput 1985;6(1):104–17.

[98] Lamb H. Hydrodynamics. 6th Ed.. Dover, New York; 1993.