Journal of Computational Physics 448 (2022) 110740

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A parallel cell-centered adaptive level set framework for efficient simulation of two-phase flows with subcycling and non-subcycling

Yadong Zeng [a], Anqing Xuan [a], Johannes Blaschke [b], Lian Shen [a,*]

[a] Department of Mechanical Engineering and Saint Anthony Falls Laboratory, University of Minnesota, Minneapolis, MN 55414, USA
[b] National Energy Research Scientific Computing Center (NERSC), Lawrence Berkeley National Lab, Berkeley, CA 94720, USA

ARTICLE INFO

Article history:
Available online 4 October 2021

Keywords:
Adaptive mesh refinement (AMR)
Two-phase flow
Level set
Subcycling
Non-subcycling

ABSTRACT

We develop a unified adaptive level set (LS) framework using the multi-level collocated grid for incompressible two-phase flows. This framework allows us to advance all variables level by level using either the subcycling or the non-subcycling method such that the data advancement on each level is fully decoupled. A series of synchronization operations are designed to keep the momentum and mass conserved across all levels. A multi-level re-initialization method for the LS function is also proposed, which greatly improves the mass conservation of the two-phase flows. The collocated grid allows the use of a single set of differential schemes and interpolation operations for all variables, which greatly simplifies the numerical implementation. The capability and robustness of the computational framework are validated by a variety of canonical problems, including the inviscid shear layer, gravity wave, rising bubble, and Rayleigh-Taylor instability. It is shown that the present multi-level scheme can accurately resolve the interfaces of the two-phase flows with gravitational and surface tension effects while having good momentum and energy conservation. At last, a three-dimensional dam breaking problem is simulated to show the efficiency and significant speedup of the proposed framework.

1. Introduction

Numerical simulation of air–water interactions is of great interest in many environmental problems and engineering applications, such as wind over breaking waves [1,2], ship hydrodynamics [3], bubbly flows [4,5], and liquid jets [6]. To capture the complex physical processes involved in the two-phase flow problems, a high grid resolution is needed around the free surface to resolve small flow structures [7]. However, those fine structures may not be present everywhere, therefore one does not need or maybe cannot afford a fine grid of uniformly high resolution across the whole domain. The need to resolve local fine features can be addressed by the adaptive mesh refinement (AMR) method. AMR increases the grid resolution in regions of interest as needed during the simulation while leaving general estimates in other regions. Since proposed in the 1980s [8,9], various classes of AMR have been developed and applied to a wide range of physical problems, including the wave energy converters [10,11], shallow water flows [12], marine ice sheet [13], island dynamics [14], supersonic flows [15], surfactant driven flows [16], viscous finger flow [17], stratified oceanic flows [18], and the astrophysics [19].

* Corresponding author.
E-mail address: shen@umn.edu (L. Shen).

https://doi.org/10.1016/j.jcp.2021.110740
0021-9991/

Based on the grid hierarchy and data structure, AMR methods can generally be classified into two groups, the quadtree/octree-based AMR (TBAMR) [20–22] and block-structured AMR (BSAMR) [8,9,23–27]. In the TBAMR, each cell can be split into four cells in two dimensions or eight cells in three dimensions and the hierarchy of the grid cells is organized using a tree structure [28]. Although the tree-based structure is an intuitive representation of the grid hierarchy for the multi-level grid and simplifies the management of the grid refinement and coarsening, it is relatively difficult to implement the data structure [28,29]. Moreover, the connectivity between adjacent cells and the refinement history need to be stored at every time step [30,31]. The BSAMR, on the other hand, builds the mesh as nested Cartesian grids [8,9,23–26,32]. It is relatively easy to use the domain decomposition method for parallelization [33]. Equations on the nested grid can also be solved efficiently utilizing the multigrid (MG) solver [23,26].

For both TBAMR and BSAMR, the choice of the grid layout can affect the complexity of the discretization scheme and multi-level algorithm on the adaptive grid [34,35]. Most existing grid layouts fall into three categories, the collocated grid, the staggered grid, and the semi-staggered grid. On a semi-staggered grid [23,36], the velocities are defined at the cell center whereas the pressure is defined at the nodal center. As a result, different interpolation schemes are required for the velocities and pressure on the multi-level grid. Moreover, two types of implicit solvers are needed for the velocity and pressure equations. The staggered grid (or MAC layout), while facilitating a divergence-free velocity field, still needs different interpolation schemes for the velocities and pressure [32,35,37–39]. Furthermore, producing a compact, accurate implicit solver for the viscous terms is not straightforward in the context of the MAC layout [20]. The collocated grid is attractive for non-orthogonal grids. Variables on different levels are coupled through the coarse-fine boundaries and are solved simultaneously. However, because of this coupling, the time step is restricted by the finest grid spacing for numerical stability. On the other hand, the level-by-level advancement method decouples the time advancement among different levels [9,23]. This method can be further divided into the non-subcycling method and subcycling methods [24,26]. The non-subcycling method uses a uniform time step for all levels. Thus, the time step is also restricted by the finest level for numerical stability. The subcycling method, where variables on different levels are advanced with different time steps, can reduce the number of advancement steps and save the simulation time. However, the variables across different levels need to be interpolated in time for the subcycling method, which is not required in the non-subcycling method. Furthermore, the subcycling method needs recursive advancement steps between different levels, making it relatively difficult to implement [35].

In the past several decades, many researchers have combined AMR with the interface-tracking techniques, such as the front-tracking method [40], and the interface-capturing techniques, such as the volume-of-fluid (VOF) method [41–43] and level set (LS) method [44–47], to simulate two-phase flow problems. Pivello et al. [39] presented a BSAMR-based adaptive front tracking method to simulate bubbly flows. This method represents the interface precisely with a Lagrangian mesh but needs frequent re-meshing when the interface deforms significantly, e.g., in a violent two-phase flow. Sussman et al. [36] proposed an adaptive LS approach for the incompressible two-phase flow within the BSAMR framework. The LS method is also developed for the unstructured TBAMR by Kohno and Tanahashi [48]. Antepara et al. [49] embedded a conservative LS method into the TBAMR framework. Popinet [50] combined the VOF method with the TBAMR for surface-tension-driven interfacial flows and de Langavant et al. [16] presented a LS method on the non-graded tree-based adaptive grids for surfactant driven flows. In the works cited above, the two-phase flow solutions are updated in time using the composite advancement method, i.e., discretized equations of velocity and pressure are constructed and solved for multiple levels simultaneously. To the best of our knowledge, there has been no implementation of the AMR framework that can utilize the level-by-level advancement method, especially the subcycling method, for the simulation of two-phase flows. Considering that complex flow structures are often present in two-phase flows, advancement using the subcycling method is desired for reducing the computational cost.

In this paper, we propose a unified BSAMR framework to simulate two-phase flows using the LS function with both the subcycling and non-subcycling methods on a collocated grid. The *AMReX* package [51] is utilized to manage the multi-level grid and perform parallel operations. There are four contributions of this paper. First, we propose a level-by-level method for time advancement in the context of two-phase BSAMR. To the best of our knowledge, this is the first framework that unifies the subcycling and non-subcycling methods to simulate two-phase flows. Second, the use of the collocated grid is also the first among the two-phase BSAMR framework. The collocated grid significantly simplifies the implementation of multi-level differential operators and interpolation schemes. Third, we design the synchronization operations, including the averaging, refluxing, and synchronization projection, which ensures that the flow field is divergence-free on the multi-level grid. Numerical tests show that our algorithm has a good conservation of momentum and energy. Lastly, we propose a robust and efficient re-initialization algorithm to maintain the LS function as the signed distance function across multiple levels. This algorithm substantially improves the mass conservation of the two fluid phases.

The remainder of this paper is organized as follows: we start with the mathematical formulation of the Navier–Stokes equations and LS advection equation for incompressible two-phase flows in section 2, followed by a description of the variables and operators on the multi-level adaptive grid in section 3; next, the time advancement algorithm is presented in section 4; validation cases and numerical tests are then given in section 5; at last, the conclusions are given in section 6.

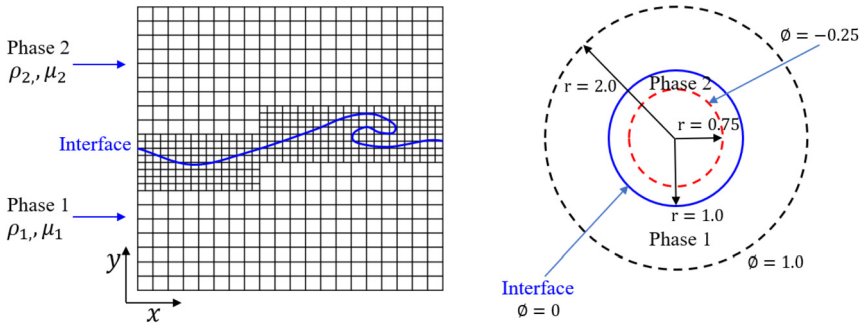

Fig. 1. Left: two-phase flow on a multi-level Cartesian grid. Right: schematic definition of the LS function.

2. Mathematical formulation

In this paper, we consider two-phase incompressible flows with gravity and surface tension effects. As illustrated in the left part of Fig. 1, the densities of the two phases are denoted by ρ_1 and ρ_2, respectively; and the dynamic viscosities are μ_1 and μ_2, respectively. The Navier–Stokes equations for the fluid flow with variable density and viscosity read

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u}\boldsymbol{u}) = \frac{1}{\rho(\phi)} \left[-\nabla p + \frac{1}{Re} \nabla \cdot 2\mu(\phi)\tilde{\boldsymbol{D}} + \rho(\phi)\frac{\hat{\boldsymbol{g}}}{Fr^2} - \frac{1}{We}\kappa(\phi)\delta(\phi)\boldsymbol{n} \right], \tag{1}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{2}$$

where $\boldsymbol{u}$ is the velocity vector, p is the pressure, ρ is the density, and μ is the dynamic viscosity. The above equations are normalized by a characteristic velocity U, characteristic length L, and the density and dynamic viscosity of phase 1, ρ_1 and μ_1. In the viscous term $\nabla \cdot 2\mu(\phi)\tilde{\boldsymbol{D}}/Re$, $Re = \rho_1 UL/\mu_1$ is the Reynolds number and $\tilde{\boldsymbol{D}} = (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T)/2$ is the strain rate tensor. For the gravitational term $\rho(\phi)\hat{\boldsymbol{g}}/Fr^2$, $\hat{\boldsymbol{g}}$ is the unit vector in the direction of gravity and the Froude number is defined as $Fr = U/\sqrt{gL}$ with g being the gravitational acceleration. In the surface tension term $\kappa(\phi)\delta(\phi)\boldsymbol{n}/We$, $\kappa(\phi)$ is the curvature of the interface, $\delta(\phi)$ is the Dirac function, $\boldsymbol{n}$ is the unit vector of the surface normal, and $We = \rho_1 U^2 L/\sigma$ is the Weber number, with σ being the surface tension coefficient.

The two immiscible fluids are tracked by the LS function ϕ [36,52]. As illustrated in the right part of Fig. 1, ϕ is the signed distance from the two-phase interface, with $\phi > 0$ in the phase 1 and $\phi < 0$ in the phase 2. The advection equation of ϕ is

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\boldsymbol{u}\phi) = 0. \tag{3}$$

The unit vector $\boldsymbol{n}$ and curvature $\kappa(\phi)$ of the interface in Eq. (1) are calculated as

$$\boldsymbol{n} = \frac{\nabla \phi}{|\nabla \phi|}, \tag{4}$$

$$\kappa(\phi) = \nabla \cdot \boldsymbol{n} = \nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|} \right). \tag{5}$$

For the numerical treatment, both the Dirac function $\delta(\phi)$ and the Heaviside function $H(\phi)$ are smoothed around the interface as [36]

$$\delta_\epsilon(\phi) = \begin{cases} 0 & |\phi| < \epsilon \\ \frac{1}{2}\left[\frac{1}{\epsilon} + \frac{1}{\epsilon}\cos\left(\frac{\pi\phi}{\epsilon}\right)\right] & |\phi| \le \epsilon, \end{cases} \tag{6}$$

$$H_\epsilon(\phi) = \begin{cases} 0 & \phi < -\epsilon \\ \frac{1}{2}\left[1 + \frac{x}{\epsilon} - \frac{1}{\pi}\sin\left(\frac{\pi\phi}{\epsilon}\right)\right] & |\phi| \le \epsilon \\ 1 & \phi > \epsilon, \end{cases} \tag{7}$$

where ϵ is the smearing width and is usually set to be twice the grid spacing [36,52]. Finally, the dimensionless density $\rho(\phi)$ and dynamic viscosity $\mu(\phi)$ are given by

$$\rho(\phi) = \lambda + (1-\lambda)H(\phi), \tag{8}$$

$$\mu(\phi) = \eta + (1-\eta)H(\phi). \tag{9}$$

In the above equations, $\lambda = \rho_2/\rho_1$ and $\eta = \mu_2/\mu_1$ are the normalized density and dynamic viscosity of phase 2.

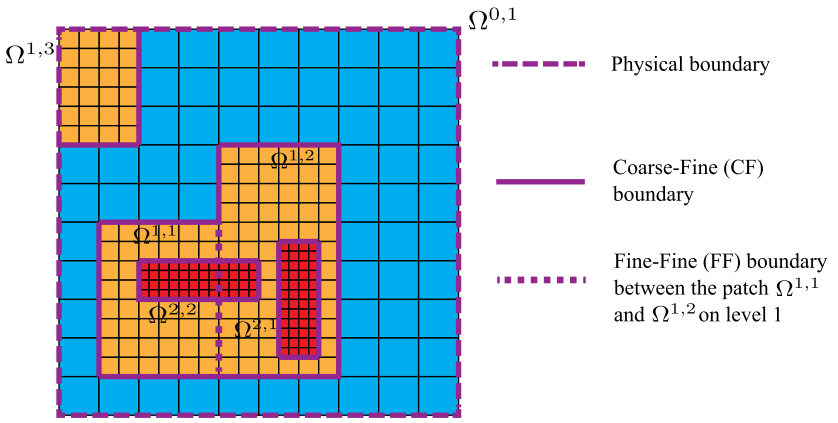

Fig. 2. Diagram of a three-level adaptive grid with three types of boundaries. $\Omega^{i,j}$ represents the patch j on level i for all $i \geq 0$, $j \geq 1$.

3. Variables and operators on multi-level adaptive grid

3.1. Concepts and definitions in BSAMR

This section introduces some important concepts of the BSAMR. In this paper, the coarsest level of the grid in the entire computational domain Γ is referred to as level 0. The finest level that the grid can be refined to is denoted as level l_{max}. In other words, the total number of levels is $l_{max}+1$. Fig. 2 illustrates a three-level adaptive grid with $l_{max}=2$ as an example.

Grid cells can be dynamically tagged and refined following certain criteria [9]. In BSAMR, although the tagging is done on individual cells, we do not refine or de-refine the cells individually. Instead, these tagged cells are grouped to form a series of rectangular patches for two-dimensional grids or cuboid patches for three-dimensional grids. There can be more than one patch on a specific level and these patches are refined simultaneously to the next level. For example, in Fig. 2, level 2 consists of two patches. Because BSAMR uses a nesting hierarchy of rectangular patches, the union patches on level $l+1$ must be contained in the union patches on level l for all $0 \leq l < l_{max}$, i.e. $\Gamma^{l+1} \subset \Gamma^{l}$, where Γ^{l} denotes the union of the patches on level l. Because of this nesting property, three types of boundaries exist on the adaptive grid:

- Physical boundary: the boundary that encloses the computational domain, illustrated using the dashed lines in Fig. 2.
- Coarse-fine (CF) boundary: the boundary between the grid cells of different levels. These boundaries are illustrated using the thick solid lines in Fig. 2.
- Fine-fine (FF) boundary: the boundary between two patches at the same level, marked using the dotted lines in Fig. 2.

Ghost cells are defined at all boundaries and their values are assigned to represent the boundary effects. The ghost cells at the physical boundaries are filled based on the physical boundary conditions. At the CF boundaries, we adopt a conservative interpolation ($\mathcal{I}_{cons}$) method [23,51] to fill the ghost cells of the fine level. To be specific, we reconstruct a continuous functional form, $f(x)$, on the ghost cells by combining the values of the coarse level and the values of the fine level. Besides satisfying the continuity across the CF boundary, the function $f(x)$ is also subject to the requirement that the average of $f(x)$ over the area of a coarse cell is equal to the original coarse cell value [53]. The conservative interpolation scheme maintains the second-order accuracy of the proposed multi-level algorithms, as verified in section 5. At the FF boundaries, the ghost cell values are copied from neighboring patches.

3.2. Definitions of variables and operators

We first define the following types of regions for a specific level l.

- Valid region (Γ^{l}_{valid}): grid cells on level l that are not covered by finer patches.
- Invalid region ($\Gamma^{l}_{invalid}$): grid cells on level l that are covered by finer patches.

We note that on level l_{max}, $\Gamma^{l_{max}}_{valid}=\Gamma^{l_{max}}$ and $\Gamma^{l_{max}}_{invalid}=\varnothing$. On level $l<l_{max}$, $\Gamma^{l}_{valid}=\Gamma^{l}\backslash\Gamma^{l+1}$ and $\Gamma^{l}_{invalid}=\Gamma^{l+1}$. Fig. 3 shows an example, where the green and orange cells represent the valid regions of level l and $l+1$, respectively. The orange area is also the invalid region of level l.

In the present work, all variables, including the velocity $\boldsymbol{u}$, pressure p, and LS function ϕ, are defined at cell centers, i.e., the collocated grid is used. Because our multi-level scheme uses a level-by-level advancement method (section 4), variables need to be available in both the valid and invalid regions. Depending on which regions we use to describe the flow field, we have the following two sets of variables.

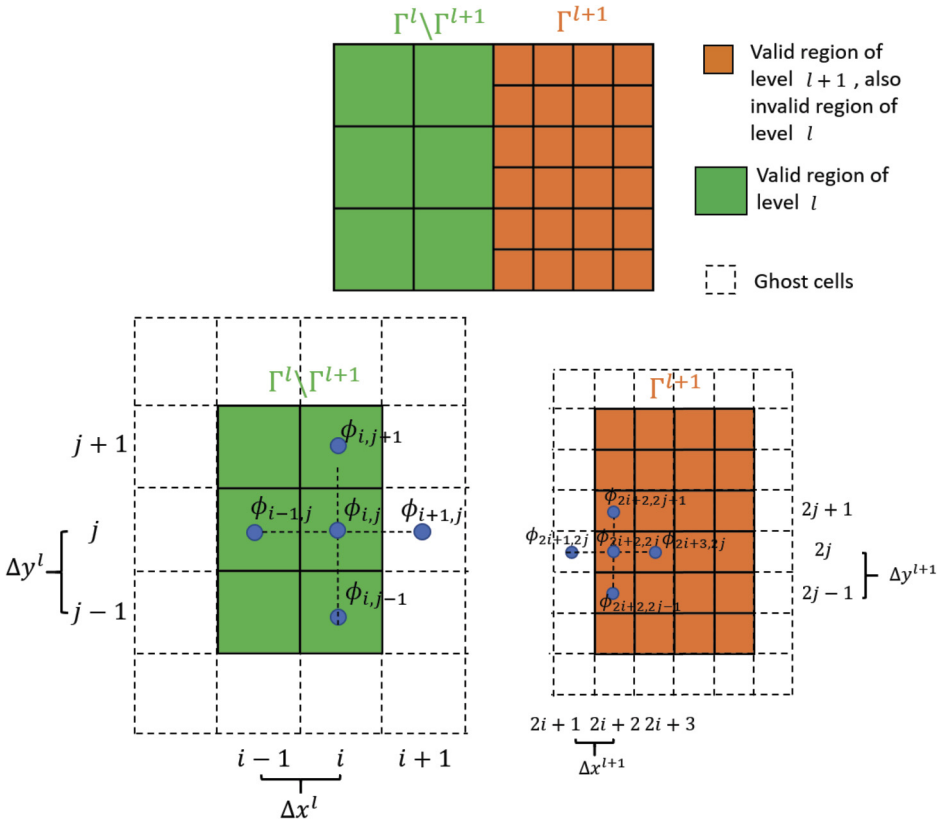

Fig. 3. Diagram of the variable definitions on a multi-level grid and the stencil of the discretization operators. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

- Composite variables: variables defined in the valid regions across a multi-level grid.
- Level variables: variables defined on the whole level, including both the valid and invalid regions.

All simulation results in this paper are presented using the composite variables unless specified otherwise. Corresponding to the composite and level variables, there are also two sets of operators in the adaptive grid algorithms.

- Composite operators: operators defined in the valid regions across multiple levels using cell values on different levels.
- Level operators: operators extended from the composite operators to all regions on a single level.

We use a 2D LS function ϕ to show the definitions of the above two types of operators, as illustrated in Fig. 3. Although only 2D operators are presented in this section, 3D operators can be defined in a straightforward way. For any point $(i, j) \subset \Gamma^l_{valid}$ on level l, the 2D composite gradient operator $\boldsymbol{G}^{cc,\mathrm{comp},l}$ is defined as

$$\begin{aligned}\left(G^{cc,\mathrm{comp},l}\phi\right)^x_{i,j} &= \frac{\phi_{i+1,j}-\phi_{i-1,j}}{2\Delta x^l},\\ \left(G^{cc,\mathrm{comp},l}\phi\right)^y_{i,j} &= \frac{\phi_{i,j+1}-\phi_{i,j-1}}{2\Delta y^l},\end{aligned} \tag{10}$$

Here, the superscript ^{cc} means that the operator applies to the cell-centered variables; the superscripts $^{\mathrm{comp}}$ and ^{l} indicate that the composite operator G is evaluated in the valid regions of level l; the superscripts x and y denote the x- and y-components of the gradient, respectively; Δx^l and Δy^l are the grid spacings in the ^{x} and ^{y} directions on level l, respectively. If the grid cell is away from the CF boundary, no ghost cell values are needed. If the grid cell is adjacent to the CF boundary, the values in the ghost cells are used for the evaluation of the composite gradient. For example, for the stencil shown in Fig. 3, the value of ghost cell $\phi_{i+1,j}$ at the CF boundary is averaged from level $l+1$. Similarly, for all $\phi_{2i+2,2j} \subset \Gamma^{l+1}_{valid}$ on level $l+1$, the composite gradient operator $\boldsymbol{G}^{cc,\mathrm{comp},l+1}$ is defined as

$$\left(G^{cc,\mathrm{comp},l+1}\phi\right)^{x}_{2i+2,2j} = \frac{\phi_{2i+3,2j} - \phi_{2i+1,2j}}{2\Delta x^{l+1}},$$
$$\left(G^{cc,\mathrm{comp},l+1}\phi\right)^{y}_{2i+2,2j} = \frac{\phi_{2i+2,2j+1} - \phi_{2i+2,2j-1}}{2\Delta y^{l+1}}. \tag{11}$$

The value of ghost cell $\phi_{2i+1,2j}$ at the CF boundary is filled using the aforementioned conservative interpolation by combining the data on both level l and level $l+1$.

Similar to the composite gradient operator $\boldsymbol{G}^{cc,\mathrm{comp},l}$, the composite divergence operator $D^{cc,\mathrm{comp},l}$ and the composite Laplacian operator $L^{cc,\mathrm{comp},l}$ are defined as

$$\left(D^{cc,\mathrm{comp},l}\phi\right)_{i,j} = \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x^{l}} + \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta y^{l}}, \tag{12}$$

$$\left(L^{cc,\mathrm{comp},l}\phi\right)_{i,j} = \frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{(\Delta x^{l})^2} + \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{(\Delta y^{l})^2}. \tag{13}$$

To simplify the notations in the following sections, we denote the union set of the composite operators as

$$G^{cc,\mathrm{comp}} = \bigcup_{i=0}^{l_{max}} G^{cc,\mathrm{comp},l},$$
$$D^{cc,\mathrm{comp}} = \bigcup_{i=0}^{l_{max}} D^{cc,\mathrm{comp},l}, \tag{14}$$
$$L^{cc,\mathrm{comp}} = \bigcup_{i=0}^{l_{max}} L^{cc,\mathrm{comp},l}.$$

For a specified level l, the expressions of the level divergence operator $D^{cc,\mathrm{level},l}$, the level gradient operator $\boldsymbol{G}^{cc,\mathrm{level},l}$, and the level Laplacian operator $L^{cc,\mathrm{level},l}$ are the same as Eqs. (10), (12), and (13), respectively. However, these level operators also apply to the invalid regions of level l, i.e., $\phi_{i,j} \subset (\Gamma^{l}_{valid} \cup \Gamma^{l}_{invalid})$. For example, when evaluating the level gradient of $\phi^{x}_{i,j}$ in Eq. (10), $\phi_{i+1,j}$ is the value from the invalid region of level l, instead of the averaged value from level $l+1$. For conciseness, we replace the $D^{cc,\mathrm{level},l}$, $\boldsymbol{G}^{cc,\mathrm{level},l}$, and $L^{cc,\mathrm{level},l}$ with the conventional mathematical expressions ∇, $\nabla\cdot$, ∇^2 hereafter.

At last, we note that the level and composite operators are closely related to the approximate projection method used in this work, which is discussed in section 4.1.1 and section 4.2.2.

4. Time advancement

In this work, we use a level-by-level method [24,25] for the time advancement on the multi-level grid. This method updates the solution on each level individually in a certain order and synchronizes the composite solution across different levels. Because the multi-level advancement algorithm is based on the single-level advancement, we first introduce the single-level algorithm in section 4.1. Then, the multi-level algorithm is described in section 4.2.

4.1. Single-level advancement

4.1.1. Time discretization

For a single level, the momentum equation (1) is advanced by a fractional step method with the approximate projection [54–56] to enforce the incompressibility condition (equation (2)). The LS advection Eq. (3) is updated using the Godunov scheme [23,36,57,58].

At the beginning of each of the time advancement of level l, the velocity $\boldsymbol{u}^{n,l}$ and the LS function $\phi^{n,l}$ at time $t^{n,l}$ are given. Owing to the fractional step method we used, the pressure is staggered in time [23,36,37] and thus the pressure $p^{n-1/2,l}$ at time $t^{n-1/2,l}$ is known. Because the single-level advancement concerns only level l, we omit the superscript ^{l} in this section. To obtain the updated velocity $\boldsymbol{u}^{n+1}$, pressure $p^{n+1/2}$, and LS function ϕ^{n+1} on level l, the solver performs the following steps:

1. Advance the LS function as

$$\phi^{n+1} = \phi^{n} - \Delta t\,[\nabla\cdot(\boldsymbol{u}\phi)]^{n+1/2}. \tag{15}$$

The advection term in the above equation is calculated using the Godunov scheme detailed in section 4.1.2.

2. Solve the intermediate velocity $\boldsymbol{u}^*$ semi-implicitly

$$\boldsymbol{u}^{*,n+1} - \frac{\Delta t}{2\rho(\phi^{n+1/2})Re}\nabla\cdot\mu(\phi^{n+1})\nabla\boldsymbol{u}^{*,n+1} = \boldsymbol{u}^{n} - \Delta t\left[\nabla\cdot(\boldsymbol{u}\boldsymbol{u})\right]^{n+1/2} + \frac{\Delta t}{\rho(\phi^{n+1/2})}\left[-\nabla p^{n-1/2} + \frac{1}{2Re}\nabla\cdot\mu(\phi^{n})\nabla\boldsymbol{u}^{n} + \rho(\phi^{n+1/2})\frac{z}{Fr^2} - \frac{1}{We}\kappa(\phi^{n+1/2})\delta(x^{n+1/2})\boldsymbol{n}\right]. \tag{16}$$

In Eq. (16), the detailed discretization of the advection term $\nabla\cdot(\boldsymbol{u}\boldsymbol{u})$, viscous term $\nabla\cdot(\mu(\phi)\nabla\boldsymbol{u})$, and surface tension term $\kappa(\phi)\delta(x)\boldsymbol{n}/We$ are given in section 4.1.2. The LS function at $t^{n+1/2}$ is calculated by

$$\phi^{n+1/2} = \frac{1}{2}(\phi^{n} + \phi^{n+1}), \tag{17}$$

where ϕ^{n+1} is obtained from step 1 (Eq. (15)). The $\rho(\phi^{n+1/2})$, $\mu(\phi^{n})$, and $\mu(\phi^{n+1})$ are then obtained from Eqs. (8) and (9).

3. Apply the projection method to obtain the pressure and a solenoidal velocity field. To conduct the level projection, a temporary variable $\boldsymbol{V}$ is defined as

$$\boldsymbol{V} = \frac{\boldsymbol{u}^{*,n+1}}{\Delta t} + \frac{1}{\rho(\phi^{n+1/2})}\nabla p^{n-1/2}. \tag{18}$$

Then the updated pressure $p^{n+1/2}$ is calculated by

$$L^{cc,\text{level}}_{\rho^{n+1/2}} p^{n+1/2} = \nabla\cdot\boldsymbol{V}, \tag{19}$$

where $L^{cc,\text{level}}_{\rho^{n+1/2}} p^{n+1/2}$ is a density-weighted approximation to $\nabla\cdot(1/\rho^{n+1/2}\nabla p^{n+1/2})$. Finally, the velocity can be calculated as

$$\boldsymbol{u}^{n+1} = \Delta t\left(\boldsymbol{V} - \frac{1}{\rho^{n+1/2}}\nabla p^{n+1/2}\right). \tag{20}$$

As defined in section 3.2, $\nabla\cdot$ and ∇ are the cell-centered level divergence operator $D^{cc,\text{level}}$ and level gradient operator $G^{cc,\text{level}}$, respectively. The level gradient operator $G^{cc,\text{level}}$ is not the minus transpose of the level divergence operator $D^{cc,\text{level}}$, i.e., $G^{cc,\text{level}} \neq -(D^{cc,\text{level}})^T$ [24,25,59]. As a result, the idempotency of the approximate projection $\boldsymbol{P} = I - G^{cc,\text{level}}(L^{cc,\text{level}})^{-1}D^{cc,\text{level}}$ is not ensured [23], i.e., $\boldsymbol{P}^2 \neq \boldsymbol{P}$. Yet, this nonidempotent approximate projection is stable and appears to be well-behaved in various numerical tests [24,54,60] and practical applications [25,36]. Notably, for a uniform single grid with periodic boundary conditions, Lai [61] theoretically proved that this approximate projection method is stable, in that $\|\boldsymbol{P}\| \leq 1$. It should be noted that the approximate projection is applied to the intermediate velocity $\boldsymbol{u}^{*,n+1}$ (Eq. (18)). Compared with the form that projects the increment velocity $\boldsymbol{u}^{*,n+1} - \boldsymbol{u}^{n}$, e.g. as that used in Almgren et al. [23], the projection method used here can reduce the accumulation of pressure errors and lead to a more stable algorithm [54,62]. We also validate the effectiveness and stability of this approximate projection in section 5.1 using a numerical test [63].

4. Reinitialize the LS function ϕ to maintain ϕ as a signed distance function of the interface and guarantee the conservation of the mass of the two phases. In this step, a temporary LS function $d(\boldsymbol{x},\tau)$ is updated iteratively using the following pseudo evolution equation,

$$\frac{\partial d}{\partial \tau} = S(\phi)(1 - |\nabla d|), \tag{21}$$

with the initial condition

$$d(\boldsymbol{x}, \tau = 0) = \phi^{n+1}(\boldsymbol{x}), \tag{22}$$

where

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

Here, τ is the pseudo time for iterations. A second-order essentially non-oscillatory (ENO) scheme is used to discretize the distance function and a second-order Runge–Kutta (RK) method is applied for the pseudo time advancing. To ensure the mass conservation, $d(\boldsymbol{x},\tau)$ is further corrected by minimizing the differences of the volume of each fluid between $\tau = 0$ and the final iteration [36,52]. Finally, the LS function ϕ is re-initialized by the volume corrected d.

At last, we give a summary of the single-level advancement algorithm in Algorithm 1 as follows.

Algorithm 1 Single-level advancement algorithm.

1: Advance the LS function using Eq. (15);
2: Solve the intermediate velocity using Eq. (16);
3: Apply the projection method to update the pressure and velocity field following Eqs. (18)–(20);
4: Re-initialize the LS function on the single level using Eqs. (21)–(23).

4.1.2. Discretization of the advection, viscous, and surface tension terms

In this part, all discretized formulas use the level operator (section 3.2). For simplicity, only the 2D discretized formulas are given in this section. The 3D formulas can be extended in a straightforward way.

For the discretization of the advection terms $[\nabla \cdot (\boldsymbol{u}\phi)]^{n+1/2}$ in Eq. (15) and $\nabla \cdot (\boldsymbol{u}\boldsymbol{u})^{n+1/2}$ in Eq. (16), we employ the Godunov scheme [57], which is robust for a wide range of Reynolds number values. These terms are determined by four sub-steps:

1. On the edges perpendicular to the x direction, the unsplit Godunov method is used to approximate the edge centered velocity ($\boldsymbol{u}^{n+1/2,L}$, $\boldsymbol{u}^{n+1/2,R}$) and edge centered LS function ($\phi^{n+1/2,L}$, $\phi^{n+1/2,R}$) at the middle time step $t^{n+1/2}$. The superscripts ^{L} and ^{R} denote that the edge centered values are approximated from the left and right sides of that edge, respectively. On the edges perpendicular to the y direction, the edge centered velocity ($\boldsymbol{u}^{n+1/2,U}$, $\boldsymbol{u}^{n+1/2,D}$) and edge centered LS function ($\phi^{n+1/2,U}$, $\phi^{n+1/2,D}$) at the middle time step $t^{n+1/2}$ can be calculated in the same way, where the superscripts ^{U} and ^{D} denote that the edge centered values are approximated from the up and down sides of that edge, respectively.
2. The Mark And Center (MAC) projection [23,36,58] is applied to obtain the divergence-free edge centered advection velocity $\boldsymbol{u}^{adv}$.
3. The advection velocity $\boldsymbol{u}^{adv}$ is then used to determine the edge centered approximate state $\boldsymbol{u}^{n+1/2}$ and $\phi^{n+1/2}$ from $\boldsymbol{u}^{n+1/2,L}$, $\boldsymbol{u}^{n+1/2,R}$, $\boldsymbol{u}^{n+1/2,U}$, $\boldsymbol{u}^{n+1/2,D}$, $\phi^{n+1/2,L}$, $\phi^{n+1/2,R}$, $\phi^{n+1/2,U}$, $\phi^{n+1/2,D}$.
4. The advection velocity $\boldsymbol{u}^{adv}$ is applied to advect the approximate state $\boldsymbol{u}^{n+1/2}$ and $\phi^{n+1/2}$. Finally, $\nabla \cdot (\boldsymbol{u}\boldsymbol{u})^{n+1/2}$ and $[\nabla \cdot (\boldsymbol{u}\phi)]^{n+1/2}$ are calculated as $\nabla \cdot (\boldsymbol{u}^{adv}\boldsymbol{u}^{n+1/2})$ and $\nabla \cdot (\boldsymbol{u}^{adv}\phi^{n+1/2})$.

For the discretization of the viscous term, the x-component of $\nabla \cdot \mu(\phi)\nabla \boldsymbol{u}$ at point (i, j) is calculated as

$$(\nabla \cdot \mu(\phi)\nabla u)_{i,j} = \frac{\mu_{i+1/2,j}(u_{i+1,j} - u_{i,j}) - \mu_{i-1/2,j}(u_{i,j} - u_{i-1,j})}{\Delta x^2} + \frac{\mu_{i,j+1/2}(u_{i,j+1} - u_{i,j}) - \mu_{i,j-1/2}(u_{i,j} - u_{i,j-1})}{\Delta y^2}, \tag{24}$$

where the edge-centered viscosity $\mu_{i+1/2,j}$ and $\mu_{i,j+1/2}$ are defined as

$$\mu_{i+1/2,j} = \frac{1}{2}\left[\mu(\phi)_{i,j} + \mu(\phi)_{i+1,j}\right], \tag{25}$$

$$\mu_{i,j+1/2} = \frac{1}{2}\left[\mu(\phi)_{i,j} + \mu(\phi)_{i,j+1}\right]. \tag{26}$$

The y-component of the viscous term is calculated for the velocity component v in a similar way. For the surface tension term $\kappa(\phi)\delta(x)\boldsymbol{n}/We$, we have

$$(\nabla\phi)_{i,j} = \left(\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x}, \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta y}\right), \tag{27}$$

$$\boldsymbol{n}_{i,j} = \frac{(\nabla\phi)_{i,j}}{(|\nabla\phi|)_{i,j}}, \tag{28}$$

$$\kappa(\phi)_{i,j} = \frac{n^1_{i+1,j} - n^1_{i-1,j}}{2\Delta x} + \frac{n^2_{i,j+1} - n^2_{i,j-1}}{2\Delta y}, \tag{29}$$

where $\boldsymbol{n} := (n^1, n^2)$ and the delta function $\delta(x)$ is calculated from Eq. (6).

4.2. Multi-level advancement

In this section, we describe how we apply the single-level advancement algorithm to the multi-level advancement algorithm using the subcycling and non-subcycling methods (section 4.2.1) and introduce the synchronization step (section 4.2.2). We also devise a multi-level re-initialization algorithm for the LS function across multiple levels (section 4.2.3). The multi-level initialization of the flow field is introduced in section 4.2.4. At last, a summary of the multi-level advancement algorithm is given in section 4.2.5. We note that, although some BSAMR [51,64] and TBAMR [28,31,63] frameworks have supported a high refining ratio, in this work, we limit the refining ratio to 2 between two consecutive levels for easier implementation.

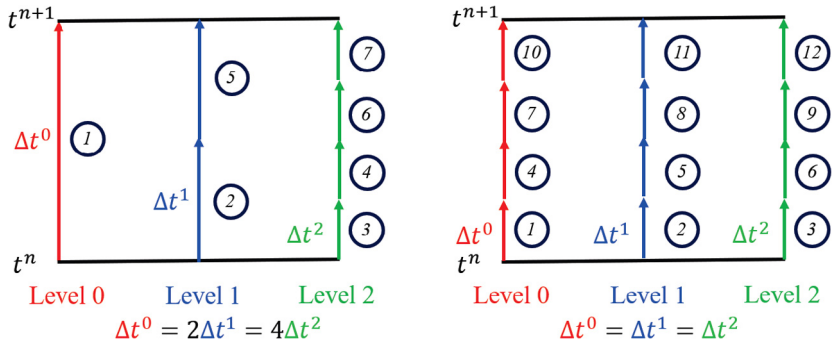

Fig. 4. Schematic of the sub-steps in the level-by-level advancement method for a three-level grid ($l_{max} = 2$). Left: the subcycling method. Right: the non-subcycling method. The sub-steps are represented by the circled numbers.

4.2.1. Subcycling and non-subcycling methods

We consider two cycling methods, subcycling and non-subcycling, to advance variables on a multi-level grid. For the subcycling method, the solutions on different levels are advanced with different time steps. The larger grid spacings on the coarser levels allow a larger time step if the CFL number is kept the same on different levels. For example, if the refining ratio between two neighboring levels is two and if the velocities are approximately the same on both levels, the time step on the coarser level, Δt^l, and on the finer level, Δt^{l+1}, has the relation $\Delta t^l = 2\Delta t^{l+1}$. In the non-subcycling method, the variables on different levels advance with the same time step, restricted by the finest level l_{max} to maintain the numerical stability. In this scenario, all levels are always at the same time instant.

Fig. 4 schematically shows how the individual levels are advanced with the subcycling and non-subcycling methods for a multi-level grid with $l_{max} = 2$. As shown in the sketch, seven sub-steps are needed to advance all levels from t^n to $t^n + \Delta t^0$ using the subcycling method. For each sub-step, the single-level advancement algorithm in section 4.1 is used for time advancement. By comparison, it takes 12 sub-steps for the non-subcycling method. Although the non-subcycling method has more steps, the subcycling method needs the time interpolation because of the mismatch of the time among different levels. The values at the middle time instant are obtained using a mid-point averaging, i.e. $f(t^n + \Delta t^l/2) = [f(t^n) + f(t^n + \Delta t^l)]/2$, which gives a second-order time accuracy. This interpolation is avoided in the non-subcycling method because all levels are at the same time instant.

4.2.2. Synchronization

Synchronization is the process of modifying the data on multiple levels to make them consistent and to better represent the composite solution. The synchronization step is needed for both the subcycling and the non-subcycling methods [23–25]. There are three sub-steps of the synchronization step.

Sub-step 1. Average

Because variables on the finer levels are considered more accurate, the velocity $\boldsymbol{u}$, pressure p, and LS function ϕ are replaced by those on the finer levels after the averaging process. In this way, the composite solution can be obtained in all valid regions of the multi-level grid. Because the collocated grid is used, the same averaging operator, i.e. the conservative interpolation operator introduced in section 3.2, can be used for all flow variables.

Sub-step 2. MAC synchronization and refluxing

When calculating the advection terms, the MAC projection (section 4.1.2) is only applied level by level. As a result, the advection velocity $\boldsymbol{u}^{adv}$ is only divergence-free on the specific level where it is calculated but not across all levels [23,25]. For example, at the CF boundary, the advection velocity $\boldsymbol{u}^{adv,l}$ on the coarser level l is not equal to the edge average of the advection velocity $\boldsymbol{u}^{adv,l+1}$ on the finer level, leading to an imbalance of the momentum fluxes. As a result, the freestream preservation is violated while advancing the variables level by level.

To remedy this problem, the differences between $\boldsymbol{u}^{adv}$ on the coarser level and the finer level are quantified during the single-level advancement (section 4.1). These velocity differences, together with flux differences, form the registers to make the corrections for each level. Specifically, the velocity registers that hold the difference of the edge centered advection velocity are given by

$$\delta \boldsymbol{u}^l = -A^l \boldsymbol{u}^{adv,l} + \frac{1}{2}\sum_{k=1}^{2}\sum_{faces} A^{l+1}\boldsymbol{u}^{adv,k,l+1}. \tag{30}$$

In the above equation, the subscript k represents the sub-steps of the finer level $l+1$ since it takes two sub-steps for level $l+1$ to catch up with level l (Fig. 4), $\sum_{faces}$ is the sum over the cell faces, and A is the area of each faces. The velocity flux

registers, including both the advective flux register $\delta \boldsymbol{f}_{\boldsymbol{u}}^{adv,l}$ and the viscous flux register $\delta \boldsymbol{f}_{\boldsymbol{u}}^{visc,l}$, are defined in a similar way as

$$\delta \boldsymbol{f}_{\boldsymbol{u}}^{adv,l} = \Delta t^l \left(A^l \boldsymbol{f}_{\boldsymbol{u}}^{adv,l} + \frac{1}{2} \sum_{k=1}^{2} \sum_{faces} A^{l+1} \boldsymbol{f}_{\boldsymbol{u}}^{adv,k,l+1} \right), \tag{31}$$

$$\delta \boldsymbol{f}_{\boldsymbol{u}}^{visc,l} = \Delta t^l \left(A^l \boldsymbol{f}_{\boldsymbol{u}}^{visc,l} + \frac{1}{2} \sum_{k=1}^{2} \sum_{faces} A^{l+1} \boldsymbol{f}_{\boldsymbol{u}}^{visc,k,l+1} \right). \tag{32}$$

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

$$\delta \boldsymbol{f}_{\phi}^{adv,l} = \Delta t^l \left(A^l \boldsymbol{f}_{\phi}^{adv,l} + \frac{1}{2} \sum_{k=1}^{2} \sum_{faces} A^{l+1} \boldsymbol{f}_{\phi}^{adv,k,l+1} \right). \tag{33}$$

In Eqs. (31)–(33), $\boldsymbol{f}_{\boldsymbol{u}}^{adv,l}$, $\boldsymbol{f}_{\phi}^{adv,l}$, and $\boldsymbol{f}_{\boldsymbol{u}}^{visc,l}$ are given by

$$\boldsymbol{f}_{\boldsymbol{u}}^{adv,l} = \boldsymbol{u}^{adv} \phi^{n+1/2}, \tag{34}$$

$$\boldsymbol{f}_{\phi}^{adv,l} = \boldsymbol{u}^{adv} \phi^{n+1/2}, \tag{35}$$

$$\boldsymbol{f}_{\boldsymbol{u}}^{visc,l} = \frac{1}{2Re} \left(\mu(\phi^{n,l}) \nabla \boldsymbol{u}^{\boldsymbol{n,l}} + \mu(\phi^{n+1,l}) \nabla \boldsymbol{u}^{\boldsymbol{*,n+1,l}} \right). \tag{36}$$

The mismatch of the velocity register in Eq. (30) forms the right hand side of a MAC solve for the correction δe^l on level l,

$$\nabla \cdot \left(\frac{A^l}{\rho^{n+1/2,l}} \nabla(\delta e^l) \right) = \nabla \cdot \delta \boldsymbol{u}^l. \tag{37}$$

After solving Eq. (37), a velocity correction $\boldsymbol{u}_{corr}^l$ is obtained by

$$\boldsymbol{u}_{corr}^l = \frac{-\nabla(\delta e^l)}{\rho^{n+1/2,l}}. \tag{38}$$

The flux corrections associated with the above velocity correction are

$$\boldsymbol{f}_{\phi}^{corr,l} = \boldsymbol{u}_{corr}^l \phi^{n+1/2,l}, \tag{39}$$

$$\boldsymbol{f}_{\boldsymbol{u}}^{corr,l} = \boldsymbol{u}_{corr}^l \boldsymbol{u}^{n+1/2,l}. \tag{40}$$

The final correction to the LS function on level l, ϕ_{sync}^l, is determined by the flux correction $\boldsymbol{f}_{\phi}^{corr,l}$ in Eq. (39) and the advective flux register $\delta \boldsymbol{f}_{\phi}^{adv,l}$ in Eq. (33) as

$$\phi_{sync}^l = -\nabla \cdot \boldsymbol{f}_{\phi}^{corr,l} - \frac{\delta \boldsymbol{f}_{\phi}^{adv,l}}{\Delta t \cdot Vol^l}, \tag{41}$$

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

$$\phi^{n+1,l} := \phi^{n+1,l} + \Delta t^l \phi_{sync}^l. \tag{42}$$

The flux correction about the velocity $\boldsymbol{f}_{\boldsymbol{u}}^{corr,l}$ in Eq. (40), together with its advective flux register $\delta \boldsymbol{f}_{\boldsymbol{u}}^{adv,l}$ in Eq. (31) and viscous flux register $\delta \boldsymbol{f}_{\boldsymbol{u}}^{adv,l}$ in Eq. (32), form a subsequent parabolic equation,

$$\boldsymbol{u}_{\boldsymbol{sync}}^{\boldsymbol{l}} - \frac{\Delta t}{2\rho^{n+1/2,l} Re} \nabla \cdot \left(\mu(\phi^{n+1}) \nabla \boldsymbol{u}_{\boldsymbol{sync}}^{\boldsymbol{l}} \right) = -\nabla \cdot \boldsymbol{f}_{\boldsymbol{u}}^{corr,l} - \frac{1}{\Delta t \cdot Vol^l} \left(\delta \boldsymbol{f}_{\boldsymbol{u}}^{adv,l} + \frac{1}{\rho^{n+1/2,l}} \delta \boldsymbol{f}_{\boldsymbol{u}}^{adv,l} \right), \tag{43}$$

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

$$\boldsymbol{u}^{n+1,l} := \boldsymbol{u}^{n+1,l} + \Delta t^l \boldsymbol{u}_{sync}^l. \tag{44}$$

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

$$\phi^{n+1,q} := \phi^{n+1,q} + \Delta t^l \mathcal{I}_{cons}(\phi_{sync}^l), \tag{45}$$

Table 1
Parameters for cases of the counter vortex problem.

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

and

$$\boldsymbol{u}^{n+1,q} := \boldsymbol{u}^{n+1,q} + \Delta t^l \mathcal{I}_{cons}(\boldsymbol{u}^l_{sync}) \tag{46}$$

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

At last, for any level $l > 0$, the velocity registers and flux registers on the coarser level $l-1$ are affected by the above correction and thus need to be updated as follows,

$$\delta \boldsymbol{u}^{l-1} := \delta \boldsymbol{u}^{l-1} + \frac{1}{2} \sum_{faces} (A^l \boldsymbol{u}^l_{corr}), \tag{47}$$

$$\delta \boldsymbol{f}^{adv,l-1}_{\boldsymbol{u}} := \delta \boldsymbol{f}^{adv,l-1}_{\boldsymbol{u}} + \frac{1}{2} \Delta t^{l-1} \sum_{faces} (A^l \boldsymbol{f}^{corr,l}_{\boldsymbol{u}}), \tag{48}$$

$$\delta \boldsymbol{f}^{visc,l-1}_{\boldsymbol{u}} := \delta \boldsymbol{f}^{visc,l-1}_{\boldsymbol{u}} + \frac{1}{2} \Delta t^{l-1} \sum_{faces} \left(\frac{1}{2} A^l \mu(\phi^{n+1}) \nabla \boldsymbol{V}^l_{sync} \right), \tag{49}$$

$$\delta \boldsymbol{f}^{adv,l-1}_{\phi} := \delta \boldsymbol{f}^{adv,l-1}_{\phi} + \frac{1}{2} \sum_{faces} (A^l \boldsymbol{f}^{corr,l}_{\phi}). \tag{50}$$

As a reminder, the above MAC synchronization and refluxing sub-step is used to maintain the conservation of momentum and scalar in the whole flow field. To validate the efficacy of this sub-step, we assess the conservation of a passive scalar in the inviscid flow of a counter-rotating vortex pair [24]. The initial azimuthal velocity $u_\theta(r)$ of one vortex is given by

$$u_\theta(r) = \begin{cases} \Gamma\left(\frac{8}{3R^3} r^4 - \frac{5}{R^4} r^3 + \frac{10}{3R^2} r\right), & r < R, \\ \Gamma\left(\frac{1}{r}\right), & r \geq R, \end{cases} \tag{51}$$

where R is the radius of the vortex core, r is the distance from the vortex center (x_c, y_c), and Γ is the vortex strength. One vortex is centered at $(x_c, y_c) = (0.3, 0.35)$ with $\Gamma = -0.35$, $R = 0.15$ and the other is at $(x_c, y_c) = (0.3, 0.65)$ with $\Gamma = 0.35$, $R = 0.15$. The computational domain is $\Omega : [0, 1] \times [0, 1]$. The grid size on level 0 is 100×100. A passive scalar advected by the above vortex pair is simulated. The initial scalar field is set as

$$s(x, y) = \begin{cases} 2.0, & \text{if } x \in [0.2, 0.8] \text{ and } y \in [0.2, 0.8], \\ 1.0, & \text{otherwise .} \end{cases} \tag{52}$$

As the LS function is essentially a passive scalar governed by the advection equation, the simulation of s is carried out using Eq. (3). A total of four subcycling cases are considered, varying in the mesh refinement and whether the refluxing step is performed, as listed in Table 1. For the mesh refinement, we consider the static and dynamic refinement. For the 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 $t = 0.36$ for the dynamic refinement case with refluxing (case 4) is shown in the left part of Fig. 5, which is in good agreement with [24]. The right part of Fig. 5 shows the scalar concentration and grid hierarchy at $t = 0.36$ for case 4. Because of the advection by the vortices, a high concentration of the scalar crosses the CF boundary, which can lead to errors in the conservation of the scalar if the MAC synchronization and refluxing operations are not considered. To quantify this error, the relative change of the total amount of scalar compared to the initial time is evaluated as

$$e(t) = \frac{\int_\Omega (s|_t - s|_{t=0})\, dx}{\int_\Omega s|_{t=0} dx}. \tag{53}$$

The results for the above four cases are plotted in Fig. 6. When the refluxing is used (cases 2 and 4), the relative error is within 10^{-16} for both the static and dynamic refinement, while noticeable errors are present in simulations without refluxing (cases 1 and 3). This test shows that the MAC synchronization and refluxing operations are necessary and can help the conservation of the scalar.

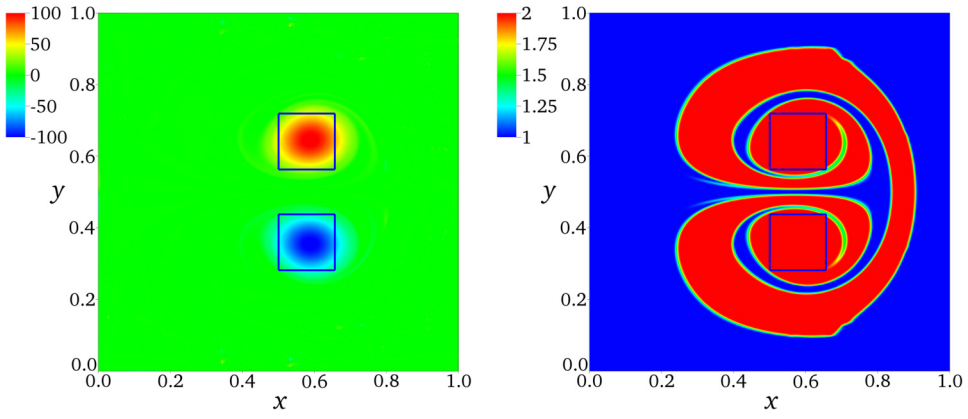

Fig. 5. Left: Vorticity contours and grid hierarchy at $t = 0.36$ of the two-level subcycling case with the dynamic mesh refinement with refluxing (case 4) for the counter vortex problem. Right: Concentration of the passive scalar s and grid hierarchy at $t = 0.36$ for case 4. The blue lines are the grid patches on level 1.

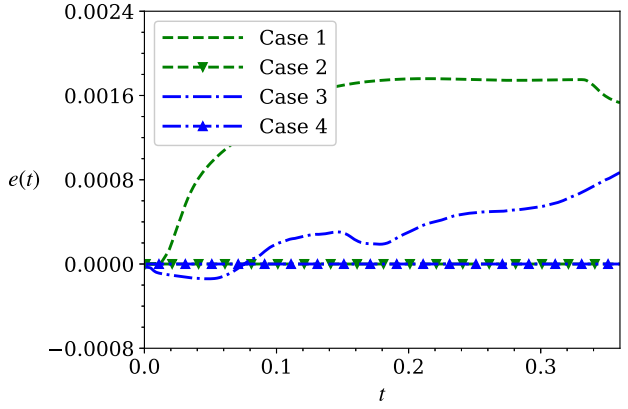

Fig. 6. Comparison of the conservation errors of the passive scalar among the four cases of the counter rotating vortex. Case 1 and case 2 use the static refinement while case 3 and case 4 use the dynamic refinement. Case 2 and case 4 consider the refluxing while case 1 and case 3 do not.

Sub-step 3. Synchronization projection

Because the level projection is only applied level by level, it does not guarantee that the velocity is divergence-free across all levels. The synchronization projection, as the last sub-step of the synchronization, is applied to enforce this constraint [23,24]. Using the composite operators defined in section 3.1, we first solve a correction field e_s by projecting velocities on all levels [26],

$$L^{cc,\mathrm{comp}}_{\rho^{n+1}} e_s = \frac{1}{\Delta t^{sync}} D^{cc,\mathrm{comp}} \boldsymbol{u}^{n+1}, \tag{54}$$

where Δt^{sync} is equal to the time step of level 0, i.e. $\Delta t^{sync} = \Delta t^0$; $L^{cc,\mathrm{comp}}_{\rho^{n+1}} e_s$ is a density-weighted approximation to $\nabla \cdot (1/\rho^{n+1} \nabla e_s)$ on all levels. We note that the ghost cell values of e_s need to be appropriately specified for the evaluation of the composite operators $L^{cc,\mathrm{comp}}$, $D^{cc,\mathrm{comp}}$, and $G^{cc,\mathrm{comp}}$ in Eq. (14). On level 0, the ghost cell values of e_s are determined by the physical boundary conditions. On level l ($l > 0$),

$$e^l_s = \mathcal{I}_{cons}(e^l_s, e^{l-1}_s), \tag{55}$$

which means that the ghost cell values e^l_s on level of l at the CF boundaries are computed using the conservative interpolation (section 3.1) by combining the values of e_s on both level l and $l-1$ [24,25]. We remark that a single conservative interpolation scheme can be used with all composite operators to fill the ghost cells of all cell-centered variables at the CF boundaries [23,65]. Finally, the multi-level velocity field is updated as

$$\boldsymbol{u}^{n+1} := \boldsymbol{u}^{n+1} - \Delta t^{sync} \boldsymbol{G}^{cc,\mathrm{comp}} e_s, \tag{56}$$

after which $\boldsymbol{u}^{n+1}$ becomes divergence-free in a multi-level sense.

The stability of the projection operation is an important issue. For example, for the non-graded TBAMR, the stability of the discrete projection can be affected by the presence of the large size ratio of the adjacent cells, for which a different projection formulation was proposed to enforce the orthogonality of the projection [63,66]. The stability of the synchronization projection method used in the present work has been demonstrated by Martin and Colella [24] using a three-vortex problem. The robustness and the stability of the approximate projection are related to the careful interpolations of the ghost cell values [62,67]. Before doing the approximate projection, we use the constant extrapolation for the ghost cell values at the physical boundaries [62]. We also apply the conservative interpolation for those on the finer levels at the CF boundaries [23]. Another factor contributing to the stability of the algorithm is the form of the projection. The projection is applied to the updated velocity at the new time ($\boldsymbol{u}^{n+1}$) rather than the increment velocity ($\boldsymbol{u}^{n+1} - \boldsymbol{u}^{n}$), which helps to stabilize the synchronization projection as found by Martin and Colella [24]. Furthermore, the nesting property of the BSAMR guarantees the regularity of the multi-level grid and probably eases the burden on the convergence of the multi-level multigrid solver compared to the non-graded TBAMR. In case a higher refining ratio is required in BSAMR, one may need to carefully implement the interpolation scheme to ensure the stability of the projection. Later in section 5.1, both the level projection and the synchronization projection methods are tested with a sample problem [63], which further confirms the stability of the approximate projection method.

As the final remark of this section, we discuss some differences of our synchronization algorithm from other implementations in the literature. For the averaging step, all variables can share the same averaging operator because of our adoption of the collocated layout. We believe this averaging process is simpler than that in Almgren et al. [23] because the latter, with a semi-staggered layout, requires different averaging operators for velocity and pressure. Secondly, in our algorithm for the MAC synchronization and refluxing, the errors are collected from the instantaneous field as velocity and flux registers, which are then used to correct the multi-level solution. In Martin and Colella [24] and Martin et al. [25], the volume discrepancy method is used to maintain the freestream preservation, where an auxiliary scalar is used as the indicator of the errors and is advected in time along with the flow field. Finally, in our algorithm, the synchronization projection is performed only when all the finer levels catch up with level 0. In other words, for every time step from t^n to t^{n+1}, only one synchronization projection step is conducted. Compared to Almgren et al. [23], where the synchronization projection is performed iteratively on pairs of two consecutive levels, our algorithm has fewer projection steps but is still effective as demonstrated.

4.2.3. Multi-level re-initialization of the LS function

In the synchronization step, the property of the LS function as a signed distance function is not guaranteed in the multi-level sense. To maintain the regularity of the LS function and improve the mass conservation, a multi-level re-initialization algorithm (Algorithm 2) is proposed here. The core part of the algorithm to synchronize the LS functions on two consecutive levels using the single-level re-initialization and interpolation iteratively, corresponding to the inner loop in Algorithm 2. First, we apply the single-level re-initialization algorithm (section 4.1.1) to the LS function and interpolate the function onto the finer level. Then the single-level re-initialization is carried out on the finer level, after which the LS function is averaged back to the coarser level. From our testing, the LS function can usually be corrected on these two levels after three iterations of the pair re-initialization. The above pair re-initialization process is applied to all levels, from level $l_{max} - 1$ to level 0, as indicated by the outer loop of Algorithm 2. This ensures the LS function as the signed distance function and mitigates the mass loss on all levels, as demonstrated by the test cases in section 5. Furthermore, our tests also show that the above re-initialization algorithm is computationally efficient.

Algorithm 2 Multi-level re-initialization of the LS function.

```
for l = l_max − 1, 0, −1 do
    for j = 1, N_iter do
        φ̂^l ← single-level re-initialization of φ^l on level l
        φ^{l+1} ← I_cons(φ̂^l)
        φ^{l+1} ← single-level re-initialization of φ^{l+1} on level l + 1
        φ^l ← average φ̂^{l+1}
    end for
end for
```

We note that our multi-level re-initialization technique is different from the one used in Sussman et al. [36], which solves the pseudo evolution equation of ϕ (Eq. (21)) from the coarsest level to the finest level using the subcycling method. In this paper, the multi-level re-initialization step starts from the finest level. The LS function on two consecutive levels are synchronized using the iteration technique, and thus the subcycling method is not needed here.

4.2.4. Multi-level initialization of flow field

All field values, including the velocity $\boldsymbol{u}$, pressure p, and LS function ϕ, need to be initialized on all levels at the beginning of the simulation. Firstly, the velocity $\boldsymbol{u}$ and LS function ϕ on the coarsest level (level 0) are assigned based on the initial conditions. Ghost cell values for these variables on level 0 are also filled by the physical boundary conditions. Then, the grid on the next level (level 1) is generated based on the refinement criteria. After the refinement, the velocity $\boldsymbol{u}$ and LS function ϕ on level 1 are determined based on the initial conditions. This "refining and filling" procedure is repeated

till the finest level l_{max} is reached, or till there is no need to refine the grid based on the refinement criteria. The pressure p is initialized as zero on all levels and corrected by the level projection at the first step.

4.2.5. Summary of multi-level advancement

Algorithm 3 summarizes the unified multi-level advancement algorithm for both the subcycling and non-subcycling methods. After the initialization, we can use either the subcycling or non-subcycling method for time advancement. The synchronization step and the multi-level re-initialization step are then applied when a coarser level catches up with a finer level. Finally, the grid refinement is applied before moving to the next time step. In the multi-level advancement algorithm, the MAC projection, semi-implicit viscous solver (Eq. (16)), level projection (Eq. (19)), MAC synchronization (Eq. (37)), and refluxing (Eq. (43)) steps use the multigrid (MG) solver on each level. The MLMG solver incorporates the mesh information across multiple levels and is only used with the synchronization projection sub-step (Eq. (54)).

As a final remark, we emphasize that our multi-level advancement algorithm is a level-by-level advancement method, which is different from the composite advancement method [36–38,68] in several aspects. In the level-by-level advancement method, the level variables are used for time advancement. Each level can be advanced individually without considering the finer levels before the synchronization step. Because the time advancements at different levels are decoupled, the constraints of the time step on the coarser levels are alleviated. This is in contrast to the composite advancement method, where the multi-level time advancement is based on the composite variables and only variables in the valid regions are utilized. The MLMG solver is employed to simultaneously update the velocity and pressure in the valid regions of all levels. This distinctly different treatment is the reason why the composite advancement method is not flexible enough to embed both the subcycling and non-subcycling methods in a straightforward way, while the level-by-level method in the present work can handle both cycling methods with relative ease.

Algorithm 3 Multi-level advancement algorithm.

Initialize $\boldsymbol{u}^0$, 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 $\boldsymbol{u}^0$, ϕ^0, and p^0 on level $l+1$
  $l \leftarrow l+1$
end while
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 to be simulated
  LevelCycling(0, t_n^0, $t_n^0 + \Delta t^0$, Δt^0)
  Apply the synchronization projection using Eqs. (54)–(56)
  Perform the multi-level re-initialization of ϕ ▷ **Algorithm 2**
  Regrid the patch hierarchy and interpolate $\boldsymbol{u}$, ϕ, and p onto new patches
end for

procedure LevelCycling(l, t^l, t^l_{max}, Δt^l)
  while $t^l < t^l_{max}$ **do**
    Perform single-level advancement on level l from t^l to $t^l + \Delta t^l$. ▷ **Algorithm 1**
    if $l < l_{max}$ **then**
      LevelCycling($l+1$, t^l, $t^l + \Delta t^l$, Δt^{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. (37)–(46)
  end if
end procedure

5. Results

This section presents several canonical test cases to validate the proposed AMR framework from different aspects. First, we shall clarify some common parameters used by these cases unless stated otherwise. For all of the following cases, Δt_0 denotes the time step on level 0. We use Δx_0, Δy_0, and Δz_0 to represent the grid spacings in x-, y-, and z-directions, respectively, on level 0. For the multi-level grid, 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 $0 \le l \le l_{max}$.

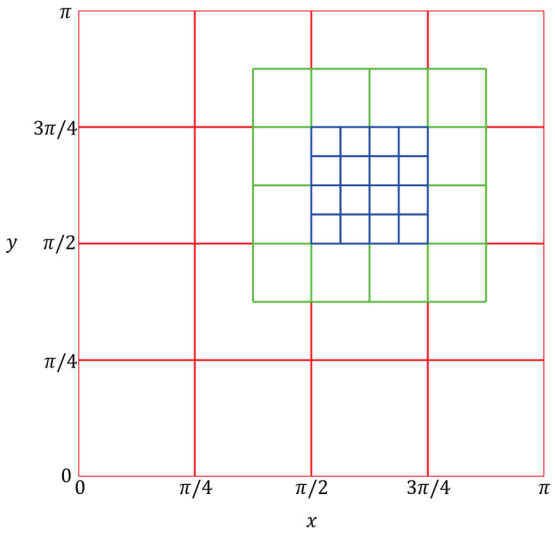

Fig. 7. Patches on the block-structured adaptive grid for testing the stability of the projections. The red, green, and blue rectangles represent the grid patches on levels 0, 1, and 2, respectively. The equivalent grid resolutions are 64×64, 128×128, and 256×256 on the red, green, and blue patches, respectively.

5.1. Stability of the projection

In this test, we examine the effectiveness and stability of the level projection (Eqs. (18) and (19)) on the single-level grid and the synchronization projection (Eqs. (54) and (56)) on the multi-level grid. The projection is performed on the manufactured velocity field $\boldsymbol{u}^* = (u^*, v^*)$ from Min and Gibou [63], given by

$$u^*(x, y) = \sin(x)\cos(y) + x(\pi - x)y^2\left(\frac{y}{3} - \frac{\pi}{2}\right), \tag{57}$$

$$v^*(x, y) = -\cos(x)\sin(y) + y(\pi - y)x^2\left(\frac{x}{3} - \frac{\pi}{2}\right). \tag{58}$$

This manufactured velocity can be decomposed as $\boldsymbol{u}^* = \boldsymbol{u}_{\text{div}} + \nabla\phi$, where $\boldsymbol{u}_{\text{div}}$ is a divergence-free velocity field and $\phi = -(x^3/3 - \pi x^2/2)(y^3/3 - \pi y^2/2)$. We consider a computational domain $\Omega = [0, \pi]$ with $\boldsymbol{u}^* \cdot \boldsymbol{n} = 0$ on the domain boundary $\partial\Omega$. For the single-level grid, the grid number on level 0 is 64×64, 128×128, 256×256, and 512×512, respectively. We iteratively apply the level projection to get the approximately divergence-free velocity $\boldsymbol{u}_{\text{appr}}$ on the single level. For the multi-level grid, we statically refine the patches in the upper-right part of the grid. As shown in Fig. 7, the grid patches in the rectangular region $(x, y) \in [3\pi/8, 5\pi/8]$ are refined to level 1, and the grid patches in the region $(x, y) \in [\pi/2, 3\pi/4]$ are further refined to level 2. We test four multi-level grids. For each grid, the finest resolutions on level 2 are the same as those on the single-level grid. The synchronization projection is applied to obtain $\boldsymbol{u}_{\text{appr}}$.

Figs. 8 and 9 show the evolution of the L^∞ norm $||\boldsymbol{u}^N_{\text{appr}} - \boldsymbol{u}_{\text{div}}||_\infty$ and the L^2 norm $||\boldsymbol{u}^N_{\text{appr}}||_2$ using the single-level projection and the multi-level synchronization projection, respectively. As shown in Fig. 8, the norm errors become almost unchanged after the first several iterations, which proves the stability of the level projection in this test. As the grid number increases, $||\boldsymbol{u}^N_{\text{appr}} - \boldsymbol{u}_{\text{div}}||_\infty$ decreases [63]. The results of the multi-level synchronization projection in Fig. 9 lead to the same conclusion, which indicate that the projection schemes in this work maintain the desired stability with the mesh refinement, consistent with the literature [24,25,67].

5.2. Taylor Green Vortex

The Taylor Green Vortex (TGV) is a canonical problem to verify the order of convergence for new algorithms. The theoretical solution of the TGV problem is given by

$$u(x, y, t) = -\cos(\pi x)\sin(\pi y)e^{-2\pi^2\mu t}, \tag{59}$$

$$v(x, y, t) = \sin(\pi x)\cos(\pi y)e^{-2\pi^2\mu t}, \tag{60}$$

$$p(x, y, t) = -\frac{\cos(2\pi x) + \sin(2\pi y)}{4}e^{-4\pi^2\mu t}, \tag{61}$$

where $\mu = 0.001$ is the dynamic viscosity.

Both the single-level and multi-level performances of our algorithms are examined here. For tests with a single level, a periodic computational domain with size 1×1 is employed for all five cases, where the grid number on level 0 is 16×16,

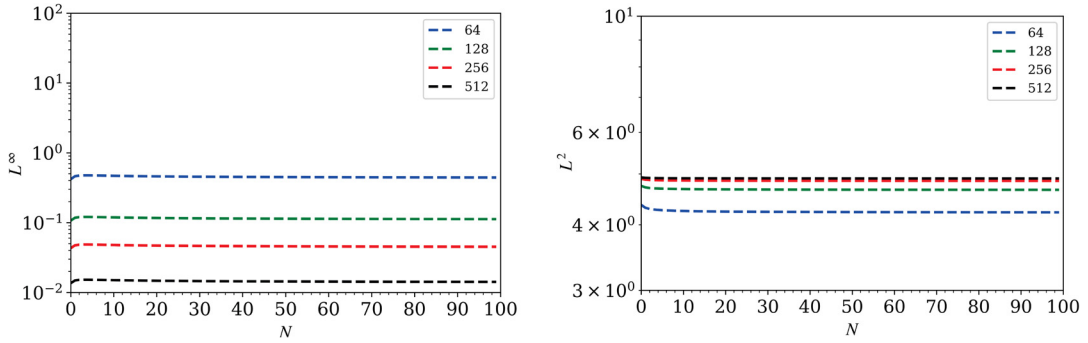

Fig. 8. Errors of the single-level projection. The N is the iteration number and the resolutions on level 0 are 64^2, 128^2, 256^2, and 512^2. Left: $||\boldsymbol{u}^N_{\text{appr}} - \boldsymbol{u}_{\text{div}}||_\infty$. Right: $||\boldsymbol{u}^N_{\text{appr}}||_2$.

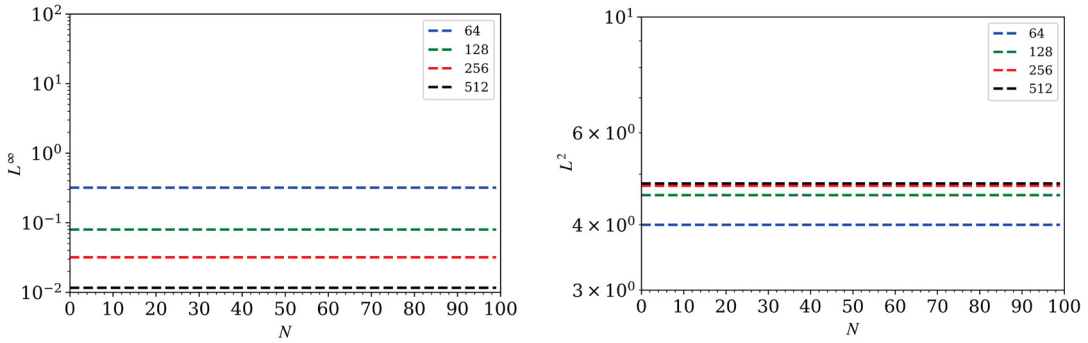

Fig. 9. Errors of the multi-level synchronization projection. The N is the iteration number and the resolutions on the finest level ($(x, y) \in [\pi/2, 3\pi/4]$) are the same as those of the single-level grid with 64^2, 128^2, 256^2, and 512^2. Left: $||\boldsymbol{u}^N_{\text{appr}} - \boldsymbol{u}_{\text{div}}||_\infty$. Right: $||\boldsymbol{u}^N_{\text{appr}}||_2$.

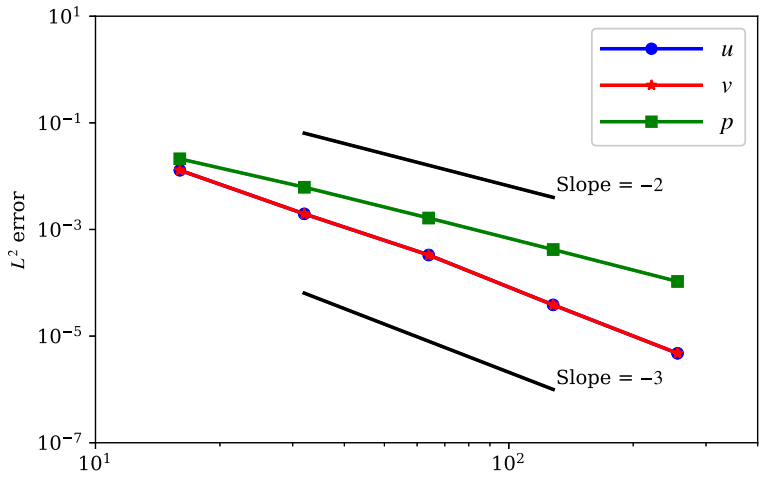

Fig. 10. Grid convergence of u, v, and p for the TGV problem on a single-level grid.

32×32, 64×64, 128×128, and 256×256, respectively. The CFL number is kept as a constant 0.5. Numerical results are compared with the theoretical results at $t = 1.0$ and the L^2 errors are calculated to obtain the point-wise convergence rate. Fig. 10 shows the order of convergence for the single level cases. It is shown that the algorithms achieve the second-order accuracy for the pressure p and a higher-order accuracy (approximately the third order) for the velocity $\boldsymbol{u}$. The higher-order convergence rate for $\boldsymbol{u}$ is also observed in the literature [23,62].

For cases with the multiple levels, the refinement criterion is based on the magnitude of the vorticity, i.e., the grid cells on the coarser levels are refined to the finer levels if $\omega_z > 0.95|\omega_z^{max}|$. The finest level in this problem l_{max} is 2. However, we note that only the static mesh refinement is used here, i.e., the grid cells are only refined at the beginning and kept unchanged throughout the simulation. This is justified because the vortex cores in this TGV problem have no translational motions. For the case with the grid number 32×32 on level 0, Fig. 11 shows the generated grid and it is seen that the grid

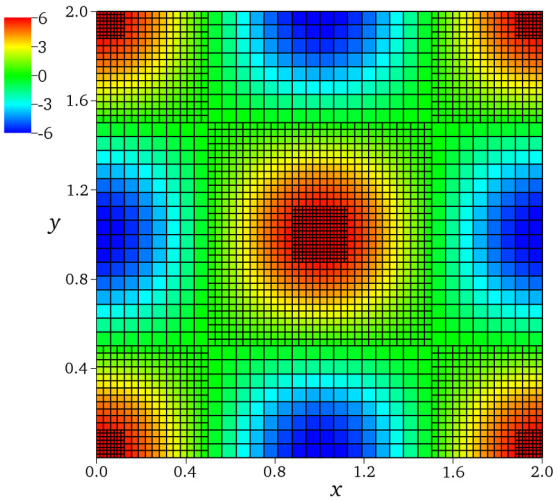

Fig. 11. Grid hierarchy for the subcycling TGV case with a grid number of 32×32 on level 0. The contours of vorticity ω_z at $t = 0$ are also shown.

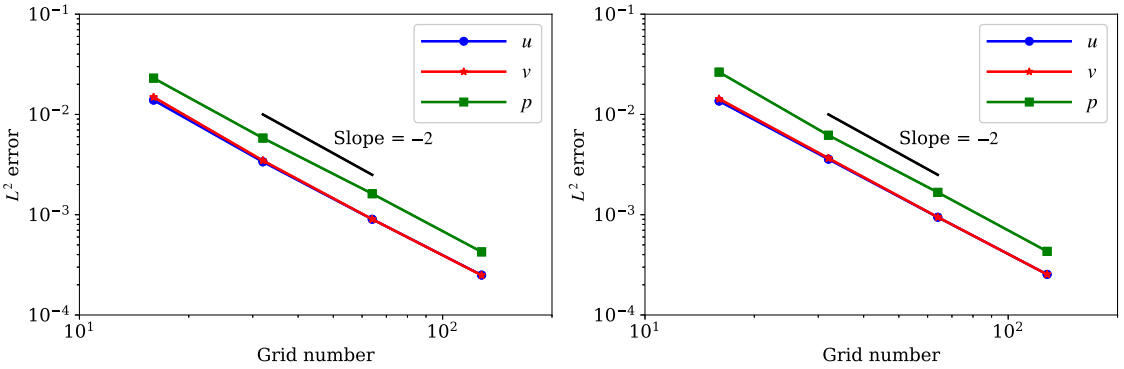

Fig. 12. Grid convergence of u, v, and p for the TGV problem on the multiple levels under the static mesh refinement. Left: the subcycling method. Right: the non-subcycling method.

resolution is higher as the vortex core is approached. Both the subcycling and non-subcycling methods are tested. Similar to the single-level cases, we compare the composite solution with the theoretical results at $t = 1.0$ using the L^2 measure of the errors. As shown in Fig. 12, the L^2 errors of $\boldsymbol{u}$ and p decrease as the grid number increases at a rate of second-order convergence. Moreover, the L^2 errors at a given grid spacing for both the subcycling method and the non-subcycling method are comparable, indicating that the two cycling methods produce consistent results. These tests show that our numerical schemes can achieve the desired second order of accuracy on a static multi-level mesh for both the subcycling and non-subcycling methods.

5.3. Four-way vortex merging

The four-way vortex merging problem is used to validate the order of convergence of our algorithms for dynamically refined meshes [22,23]. Here, four vortices are placed in a unit square domain and centered at $(0.5, 0.5)$, $(0.59, 0.5)$, $(0.455, 0.5 + 0.045\sqrt{3})$, and $(0.455, 0.5 - 0.045\sqrt{3})$, respectively. Their vortex strengths are -150, 50, 50, and 50, respectively. For each vortex, the vorticity ω_z decays from the center (x_i, y_i) as $(1 + \tanh(100\,(0.03 - r_i))/2$, where $r_i = \sqrt{(x - x_i)^2 + (y - y_i)^2}$. To initialize the velocity field, the vorticity field ω_z is used as the source term of the Poisson equation for the stream function ψ

$$L^{cc,\mathrm{comp}}\psi = \omega_z. \tag{62}$$

The initial velocity field is then calculated as $u(x, y) = \partial\psi/\partial y$ and $v(x, y) = -\partial\psi/\partial y$. The Reynolds number is set to $Re = 1000$. The vorticity criterion is used for the dynamic mesh refinement, i.e., the grid cells on the coarser levels are refined to the finer levels as long as $|\omega_z| > 0.05|\omega_z^{max}|$. The finest level l_{max} is 2 in this problem. As shown in Fig. 13, the patches on the finer levels change dynamically to capture the merging vortices. The evolution of the vortices also agrees well with the results in the literature [22,23].

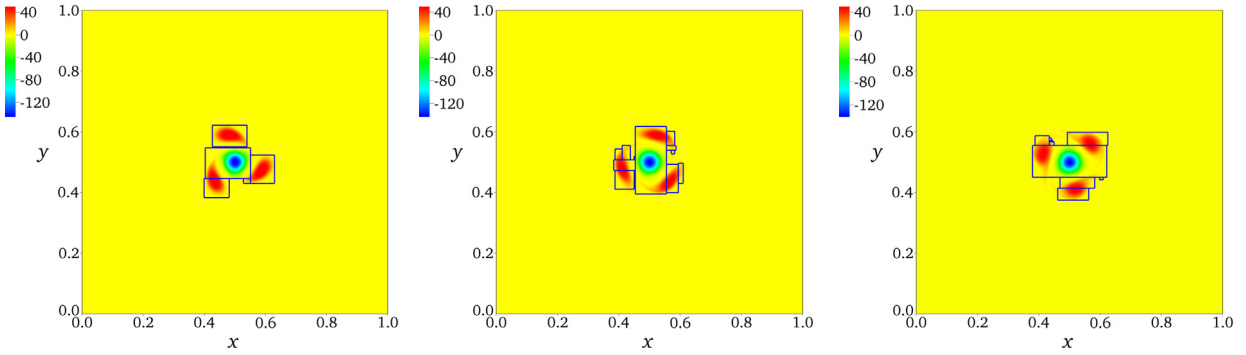

Fig. 13. Evolution of vorticity ω_z and grid hierarchy for the non-subcycling four-way vortex merging case. The grid number on level 0 is 64×64 and the patches are dynamically refined to $l_{max} = 2$. The blue rectangles represent the patches on level 2.

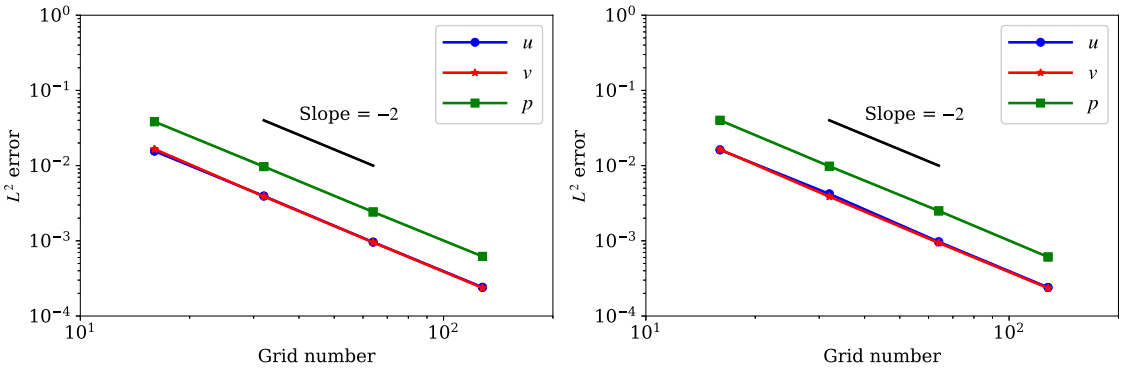

Fig. 14. Grid convergence of u, v, and p for the four-way merging vortex problem on the multiple levels with the dynamic mesh refinement. Left: the subcycling method. Right: the non-subcycling method.

To obtain the point-wise convergence rate on the multi-level grid, we consider five cases here, of which the grid number on level 0 is 16×16, 32×32, 64×64, 128×128, and 256×256, respectively. The CFL number is kept as a constant 0.9. Because there is no exact solution for this problem, we use the result on a 1024×1024 uniform grid as the reference solution. Numerical results are compared with the reference solution at $t = 0.25$ and the L^2 errors are calculated. Fig. 14 shows the L^2 errors of $\boldsymbol{u}$ and p as a function of the grid number. We can see that our numerical scheme maintains the second-order accuracy in the context of dynamic mesh refinement with two cycling methods.

5.4. Inviscid shear layer

The inviscid shear layer problem is used to validate the proposed algorithms under the Euler limit within the BSAMR framework. Similar to the setup in Bell et al. [58], the computational domain is 1×1 with periodic boundary conditions in both the horizontal and vertical directions. The properties of the inviscid fluid are $\rho(\phi) = \mu(\phi) = 1.0$. The initial velocity is given by

$$u(x, y) = \begin{cases} \tanh(\sigma_1(y - 0.25)) & y \leq 0.5 \\ \tanh(\sigma_1(0.75 - y)) & y > 0.5, \end{cases} \tag{63}$$

$$v(x, y) = \sigma_2 \sin(2\pi x), \tag{64}$$

where $\sigma_1 = 30$ and $\sigma_2 = 0.05$. For this problem, we consider four cases, the parameters of which are listed in Table 2. These cases have different refinement levels and the vorticity magnitude $|\omega_z| > 0.75|\omega_z^{max}|$ is used as the refinement criterion, same as the four-way merging vortex problem. The comparison between the subcycling and non-subcycling methods are also considered in these cases.

Fig. 15 plots the evolution of the vorticity ω_z and the patches for the three-level subcycling case (case 3). It shows that our simulation captures the very fine vortex structure and has a good agreement with Bell et al. [58] and Huang et al. [69].

At the Euler limit, the kinetic energy should remain constant. To validate the property of energy conservation, we evaluate the kinetic energy change $\Delta E(t) = E(t) - E(0)$, where the kinetic energy $E(t) = \int_\Omega [u^2(t) + v^2(t)]/2dx$. The time series

Table 2
Parameters for cases of the inviscid shear layer problem.

Case No.	Grid number on level 0	l_{max}	Δt_0	Cycling method
1	128×128	1	0.001	Subcycling
2	128×128	1	0.001	Non-subcycling
3	128×128	2	0.001	Subcycling
4	128×128	2	0.001	Non-subcycling

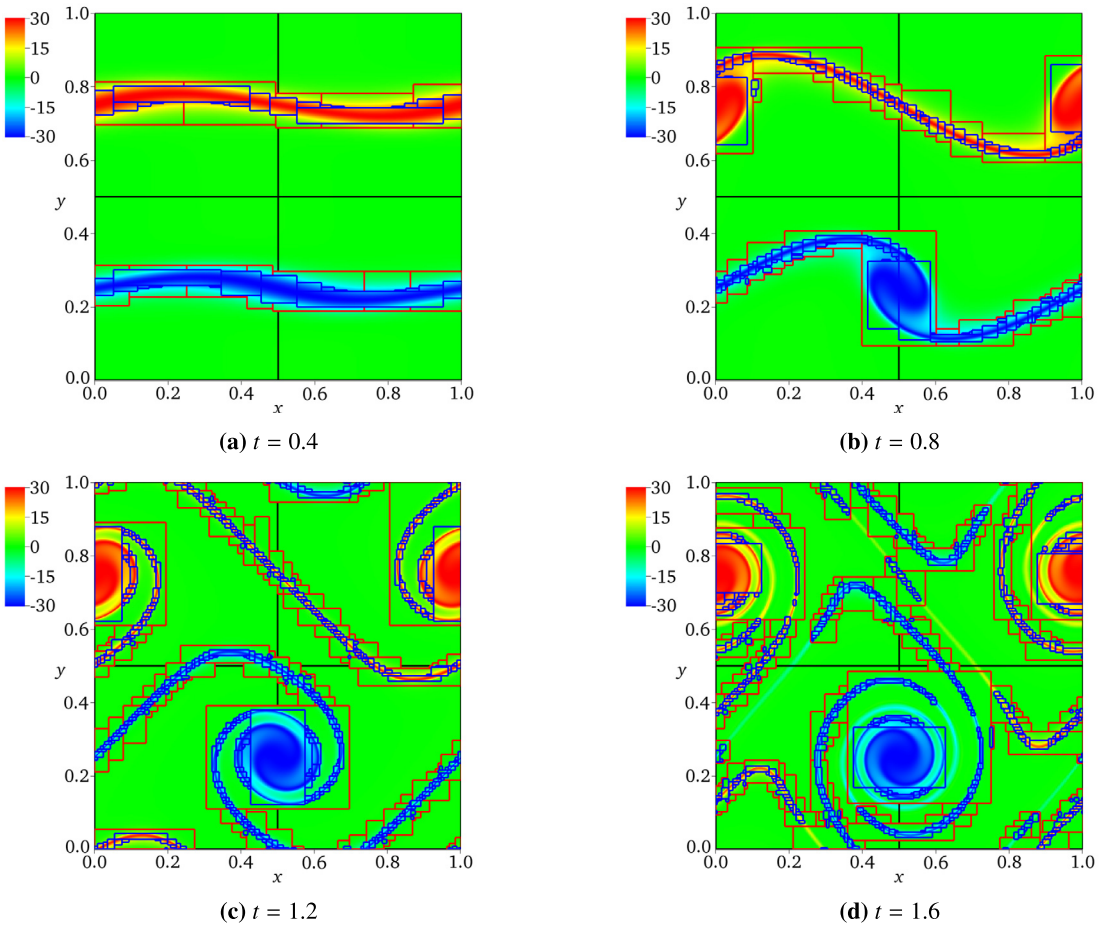

Fig. 15. Evolution of the vorticity field and grid hierarchy of the inviscid shear layer problem for case 3 (three levels with subcycling). The black, red, and blue rectangles represent the patches on levels 0, 1, and 2, respectively.

of the relative kinetic energy change, $\Delta E(t)/E(0)$, for each case are plotted in Fig. 16. Although $\Delta E(t)/E(0)$ shows small oscillations, which might be caused by the regridding and interpolation operations across multiple levels, the maximum relative kinetic energy variation of all these cases are within 0.12%, comparable to the 0.3% in Huang et al. [69]. Comparing cases 1 and 3 (or cases 2 and 4), one can see that the finer grid cells enabled by the additional level of mesh improve the conservation of the kinetic energy. In summary, our algorithm has a fairly good performance in conserving the kinetic energy for both the subcycling and non-subcycling methods.

5.5. Zalesak's problem

The rotation of a notched disk, i.e., the Zalesak's problem [70], is used to validate the advection of the LS function, the single-level re-initialization algorithm, and the multi-level re-initialization algorithm. The computational domain is a 1×1 periodic rectangle. A notched disk with radius $r = 0.15$ is initially placed at $(0.5, 0.75)$ using the LS function ϕ, as shown in Fig. 17. The height and width of the notch are 0.25 and 0.05, respectively. The disk is transported by a prescribed steady velocity field given by

$$u(x, y) = 0.5 - y, \quad v(x, y) = x - 0.5. \tag{65}$$

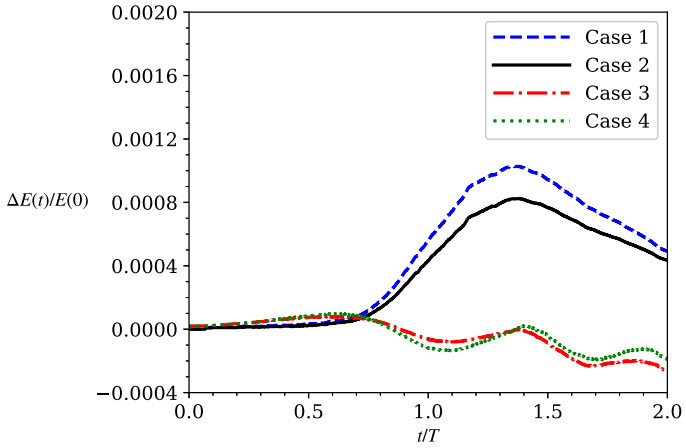

Fig. 16. Relative kinetic energy error $\Delta E(t)/E(0)$ among the four multi-level cases in the inviscid shear layer problem. Case 1 and case 3 use the subcycling method while case 2 and case 4 use the non-subcycling method. For case 1 and case 2, $l_{max} = 1$. For case 3 and case 4, $l_{max} = 2$.

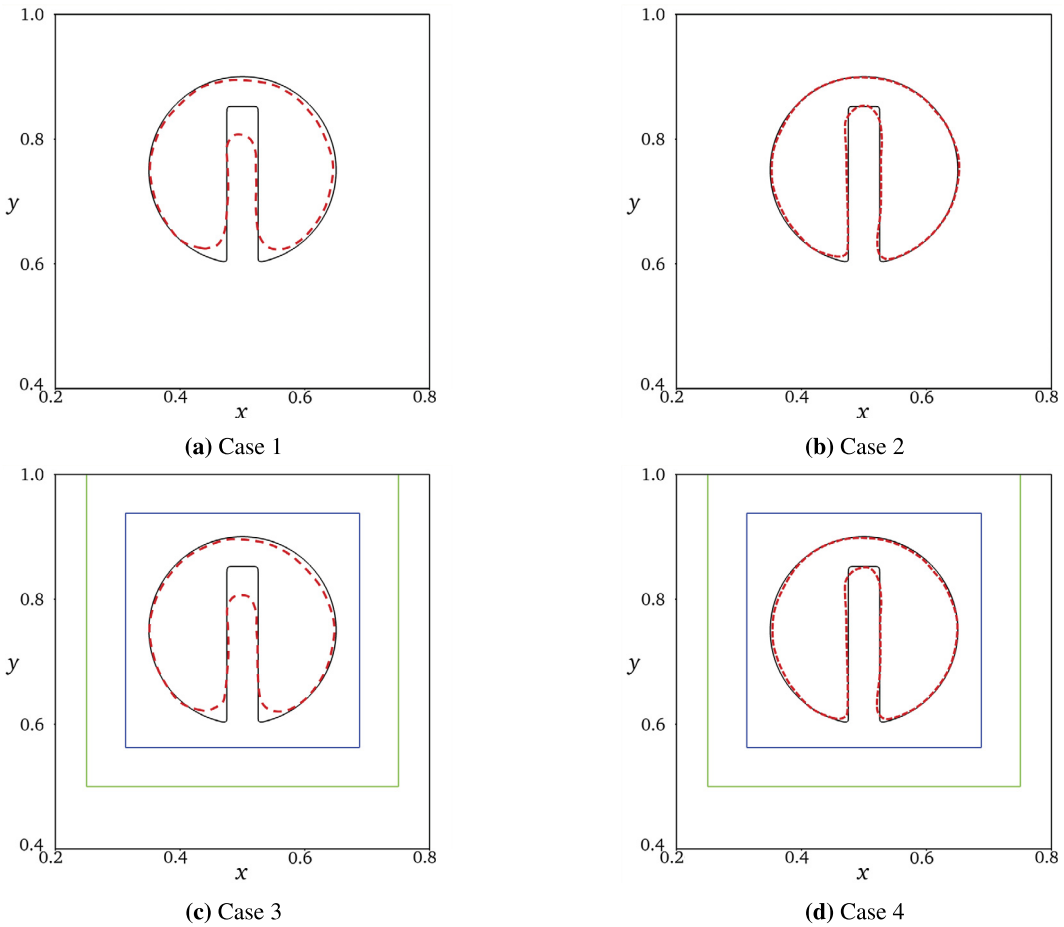

Fig. 17. Comparison of the shapes of the initial Zalesak disk (solid black line) and the disk after one revolution (red dashed line). Case 1 and case 2 use a single-level grid while case 3 and case 4 use the multi-level grid with $l_{max} = 2$. Case 2 and case 4 employ the single-level re-initialization and the multi-level re-initialization, respectively. Case 1 and case 3 do not consider the re-initialization. Only part of the computational domain ($[0.2, 0.8] \times [0.4, 1.0]$) is displayed of with LS function for better visualization. The green and blue lines represent patches on level 1 and level 2, respectively.

Table 3
Parameters for cases of the Zalesak's problem.

Case No.	Grid number on level 0	l_{max}	Δt_0	Cycling method	With reinitialization?
1	192×192	0	0.002	–	No
2	192×192	0	0.002	–	Yes
3	48×48	2	0.0005	Subcycling	No
4	48×48	2	0.0005	Subcycling	Yes

Table 4
Comparison of the relative difference of the disk area at $t = 2\pi$ after one revolution among the four cases for the Zalesak's problem. Case 1 and case 2 use the single-level grid. Case 3 and case 4 use the multi-level grid with $l_{max} = 2$. Case 2 and 4 have the re-initialization while case 1 and 3 do not.

	Case 1	Case 2	Case 3	Case 4
$\delta(2\pi)$	0.042	0.00078	0.056	0.00081

The parameters of the simulation cases are given in Table 3. Cases 1 and 2 are single-level simulations and cases 3 and 4 are multi-level simulations. The refinement criterion is the distance to the interface, i.e., the grid cells (i, j) on level l $(0 \le l < l_{max})$ are refined to the finer level if $|\phi_{i,j}| < 3.0 \max(\Delta x^l, \Delta y^l)$, where Δx^l and Δy^l are the grid spacings in the x and y directions, respectively. The finest level l_{max} is 2 in this problem. Among the four cases considered here, the re-initialization is only performed in cases 2 and 4 to show the effect of the re-initialization operations.

The Zalesak disk rotates counterclockwise under the prescribed velocity. Ideally, the shape of the disk should stay the same and return to its initial state after one evolution. Fig. 17 shows the shapes of the Zalesak disk, denoted by $\phi = 0$, at the initial moment and after one revolution. For case 1 and case 3, the notched disk deviates from its original shape noticeably. For case 2 and case 4 with the re-initialization process, the disk shape is preserved much better. To quantify the errors in ϕ, we calculate the relative errors of the disk area, $\delta(t)$, as [71]

$$\delta(t) = \frac{1}{L} \int_{\Omega} |H(\phi(t)) - H(\phi_e(t))| dx, \tag{66}$$

where $\phi_e(t)$ is the exact LS function and L is the perimeter of the interface. Table 4 shows the relative error after one revolution, i.e. $t = 2\pi$, for different cases. The errors in case 2 and case 4 are two orders of magnitude smaller than case 1 and case 3, which shows the efficacy of the single-level and multi-level re-initialization algorithms. Specifically, the errors of case 2 and case 4 have the same order of magnitude as those in [71], in which the coupled level set and volume-of-fluid (CLSVOF) method is used. We also remark that the refluxing issue (section 4.2.2) is not a concern in this problem because the velocity is prescribed, and the non-subcycling results (not shown here) have negligible differences from the subcycling results. Therefore, we conclude that the single-level and the multi-level re-initialization algorithms in this work can maintain the LS function as the signed distance function and keep the mass conserved. Our advection schemes for the LS function are also validated with both the subcycling and non-subcycling methods.

5.6. Gravity wave

To confirm that the LS advection scheme works well when coupled with the momentum equations for two-phase flows, a canonical decaying gravity wave case is tested here. The surface profile of a linear deep-water wave is initialized as

$$\eta(x, y) = a_0 \cos(kx - \omega t). \tag{67}$$

The velocities are

$$u(x, y) = a_0 \omega e^{ky} \cos(kx - \omega t), \quad v(x, y) = a_0 \omega e^{ky} \sin(kx - \omega t). \tag{68}$$

Here, a_0 is the initial wave amplitude, k is the wave number, and $\omega = \sqrt{gk}$ is the angular frequency according to the dispersion relationship. In our tests, the wave steepness $a_0 k$ is set to 0.1 such that the linear wave theory is still valid. According to Lamb [72], the wave decays with time because of viscous dissipation. Its amplitude evolution is

$$a(t) = a_0 e^{-2\nu k^2 t}. \tag{69}$$

The Reynolds number is set to $Re = \omega / \nu k^2 = 110$, where ν is the kinematic viscosity of the water. Other dimensionless number are Froude number $Fr = (\omega k^{-1}) / \sqrt{g k^{-1}} = 1.0$, Weber number $We = \rho_w k^{-2} g / \sigma = \infty$, density ratio $\lambda = \rho_2 / \rho_1 = 0.0011$, and dynamic viscosity ratio $\eta = \mu_2 / \mu_1 = 0.0085$. The computational domain size is $2\pi \times 2\pi$ and the mean water depth is π. The free slip boundary condition is imposed at the bottom and top of the domain, and the periodic boundary

Table 5
Parameters for cases of the gravity wave problem.

Case No.	Grid number on level 0	l_{max}	Δt_0	Cycling method
1	128×128	0	0.0005	–
2	256×256	0	0.00025	–
3	64×64	2	0.001	Subcycling
4	128×128	1	0.0005	Subcycling

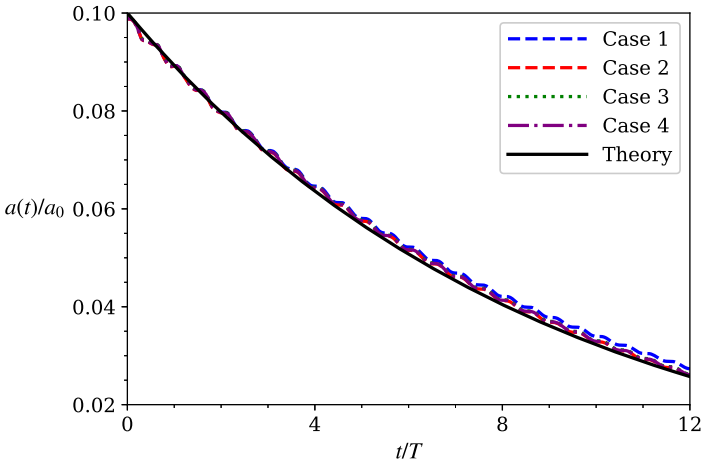

Fig. 18. Comparison of the evolution of the wave amplitude among the single-level cases (case 1 and case 2), the subcycling three-level case (case 3), the subcycling two-level case (case 4), and the theoretical result for the decaying gravity wave problem. Case 1 has the coarsest resolution while case 2, case 3, and case 4 have the same finest resolution.

condition is applied at the left and right boundaries. The parameters of the four cases considered in this problem are given in Table 5. Cases 1 and 2 are single-level cases with the latter having a higher resolution. Cases 3 and 4 use adaptive grids and their grid spacings on the finest level are the same as case 2. The grid is refined based on the distance to the air–water interface as in the Zalesak's problem (section 5.5), therefore the grid resolution near the wave surface in cases 2–4 is the same.

The amplitude evolution of the above four cases is plotted in Fig. 18. Results show that case 1, which has a relative coarse grid resolution, has small deviation from the theoretical result. Meanwhile, the amplitude evolution in the single-level high-resolution case (case 2), the three-level subcycling case (case 3), and the two-level subcycling case (case 4) agree well with the linear wave theory. This result shows that, as the grid resolution increases near the water surface, the simulation results converge to the theoretical solution. Comparing cases 3 and 4 with case 2, we note that the locally refined mesh can yield the same result as the uniform single-level fine mesh. We remark that the small oscillations in the numerical results could be due to using a potential flow solution as the initial condition of a viscous incompressible two-phase flow solver [72,73]. In summary, we conclude that our algorithms satisfy the grid convergence and can accurately simulate the two-phase gravity wave flow when multiple levels are considered.

5.7. Rising bubble

Next, a spherical-cap bubble rising in a liquid is simulated to validate our algorithms for a two-phase flow problem with surface tension. Compared with the gravity wave case, where patches on the finer levels change slowly because of the slow decay of the wave, the grid cells here are refined more dynamically to capture the rising bubble. A large computational domain $[-12, 12] \times [-18, 30]$ is chosen to circumvent wall effects. Free-slip conditions are applied at all boundaries. A spherical bubble of dimensionless radius one is put at $(x, y) = (0, 3)$ surrounded by the stationary fluid as the initial condition. Based on the steady rise velocity $V = 0.215$ m/s and the bubble radius $r = 0.0061$ m in Sussman et al. [36] and Hnat and Buckmaster [74], the dimensionless parameters in our simulation are set as $Re = \rho_1 Vr/\mu_1 = 9.8$, $Fr = V/\sqrt{gr} = 0.872$, $We = \rho_1 V^2 r/\sigma = 7.6$, $\lambda = \rho_2/\rho_1 = 0.0011$, and $\eta = \mu_2/\mu_1 = 0.0085$. Three cases are simulated, of which the parameters are listed in Table 6. For the subcycling case 2 and the non-subcycling case 3, the grid number on level 0 is 128×256 and then refined to the $l_{max} = 2$, i.e., three levels in total, such that the resolution on the finest level is equivalent to case 1 with grid number 512×1024. The refinement criterion is based on the distance to the air–water interface, same as the gravity wave problem (section 5.6).

Fig. 19 shows the movement of the mass centroid of the bubble for different cases. In all of the four cases considered here, the bubble quickly reaches a steady rising velocity. Theoretically, the steady dimensionless rising velocity V_t in this problem is 1.0 [36,74]. Table 7 compares it with the values from the simulations. The relative errors of the four cases are

Table 6
Parameters for cases of the rising bubble problem.

Case No.	Grid number on level 0	l_{max}	Δt_0	Cycling method
1	512×1024	0	0.00025	–
2	128×256	2	0.001	Subcycling
3	128×256	2	0.00025	Non-subcycling

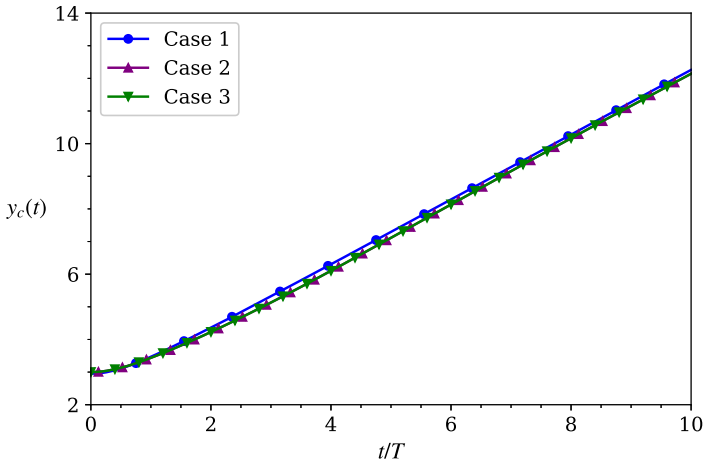

Fig. 19. Comparison of the time series of the bubble centroid ($y_c(t)$) between the single-level case (case 1), the three-level subcycling case (case 2), and the three-level non-subcycling case (case 3).

Table 7
Comparison of the steady rising velocity (V_t) among the cases for the rising bubble problem. Case 1: single-level case; case 2: three-level subcycling case; case 3: three-level non-subcycling case.

	Case 1	Case 2	Case 3	Theory
V_t	0.971	0.966	0.965	1.0

within 4%. Fig. 20 shows the time evolution of the bubble shape. Under the combined effects of buoyancy force, viscosity, and surface tension, the bottom of the bubble moves faster than its top at the initial stage, compressing the bubble in the vertical direction and flattening it in the horizontal direction. As shown in Fig. 20, at the later stage $t = 6.0$–8.0, the bubble rises with a constant speed and its shape keeps nearly unchanged. The numerically computed bubble shapes in Fig. 20 are found in agreement with the experiment of Hnat and Buckmaster [74] and the simulation of Ryskin and Leal [75] (results not plotted here). This test proves that our algorithms can correctly capture the dynamics of the two-phase flow when the surface tension effect is involved. The results also indicate that the AMR technique can reproduce the results accurately while using the fewer grid number compared with the single-level fine-grid simulation. Furthermore, the nearly identical results between the subcycling and non-subcycling methods, as shown in Fig. 20, validate the consistency of these two cycling methods in our unified BSAMR framework.

5.8. Rayleigh–Taylor instability

The Rayleigh–Taylor (RT) instability problem is simulated here to validate the adaptive two-phase flow algorithms when small vorticity structures are involved. This instability phenomenon occurs for any perturbation to the interface between a lighter fluid (ρ_2) at the bottom and a heavier fluid (ρ_1) at the top. In the simulation, we follow the same setup as Guermond and Quartapelle [76]. The computational domain is $[0, 1] \times [0, 4]$. The initial interface is given by $y(x) = 2.0 + 0.1\cos(2\pi x)$. The density ratio is set to $\lambda = \rho_2/\rho_1 = 1/3$ and the Reynolds number is set to be $Re = \rho_1 g^{1/2}/\mu_1 = 3000$. Five cases with different parameters are presented in Table 8. The single-level case (case 1) has the same grid number and time step Δt_0 as in Guermond and Quartapelle [76] and Ding et al. [77]. For the multi-level cases, the refinement criterion is based on the distance to the air–water interface, same as the gravity problem. From case 2 to case 5, we keep the same resolution on the finest level as case 1 while varying the grid number on level 0. The time step is changed accordingly for the subcycling method or the non-subcycling method.

The evolution of the air–water interface for the three-level non-subcycling case (case 5) is shown in Fig. 21. Refined patches on the finer levels are also presented to show the change of the adaptive meshes. We observe a good agreement when comparing the shape of the interface with Ding et al. [77] (not plotted here). A small perturbation of the interface

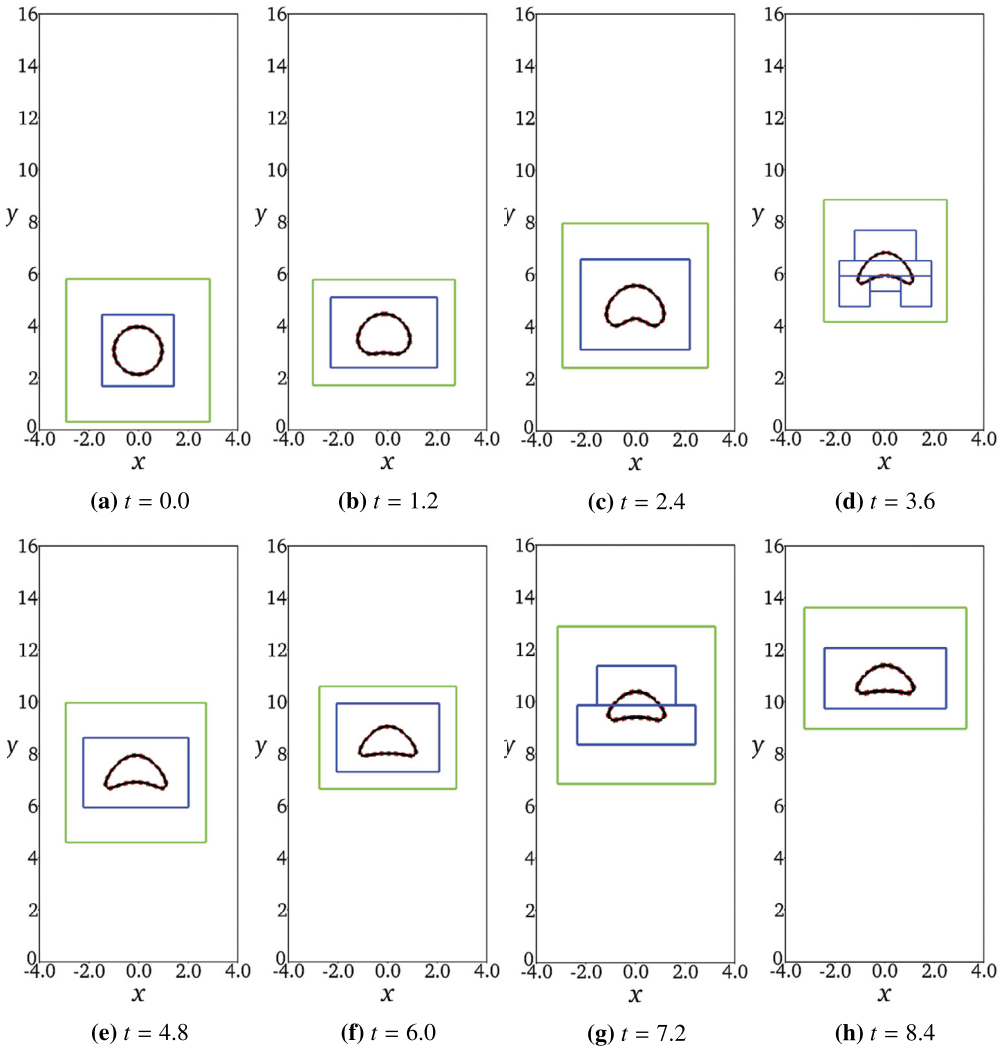

Fig. 20. Evolution of the shape of the rising bubble for the three-level subcycling case (case 2) and three-level non-subcycling case (case 3). The bubble shape is shown by the isoline of $\phi = 0$. The solid black line and dashed red line correspond to case 2 and case 3, respectively. The green and blue lines represent the patches on level 1 and level 2, respectively.

Table 8
Parameters for cases of the Rayleigh–Taylor Instability problem.

Case No.	Grid number on level 0	l_{max}	Δt_0	Cycling method
1	200×800	0	0.0005	–
2	100×400	1	0.001	Subcycling
3	100×400	1	0.0005	Non-subcycling
4	50×200	2	0.002	Subcycling
5	50×200	2	0.0005	Non-subcycling

appears at $t = 0$ and begins to grow due to the gravity effects. The interface then rolls up into the lighter fluid, and a long tail is then formed from $t = 1$ to $t = 1.75$. It is seen that the curling tip of the interface, as well as the secondary vortices of the roll-ups, are fully resolved by the adaptively generated mesh. The left side of Fig. 22 compares the transient locations of the falling fluid $y_f(t)$ and rising fluid $y_r(t)$ between the single-level case (case 1) and previous research. A good agreement is obtained. The multi-level cases (cases 2-5) agree well with the single-level fine-grid case (case 1), as shown on the right side of Fig. 22. By comparing the transient locations between case 2 and case 3 and between case 4 and case 5, we conclude that the subcycling and non-subcycling methods are consistent with each other, same as the rising bubble problem (section 5.7).

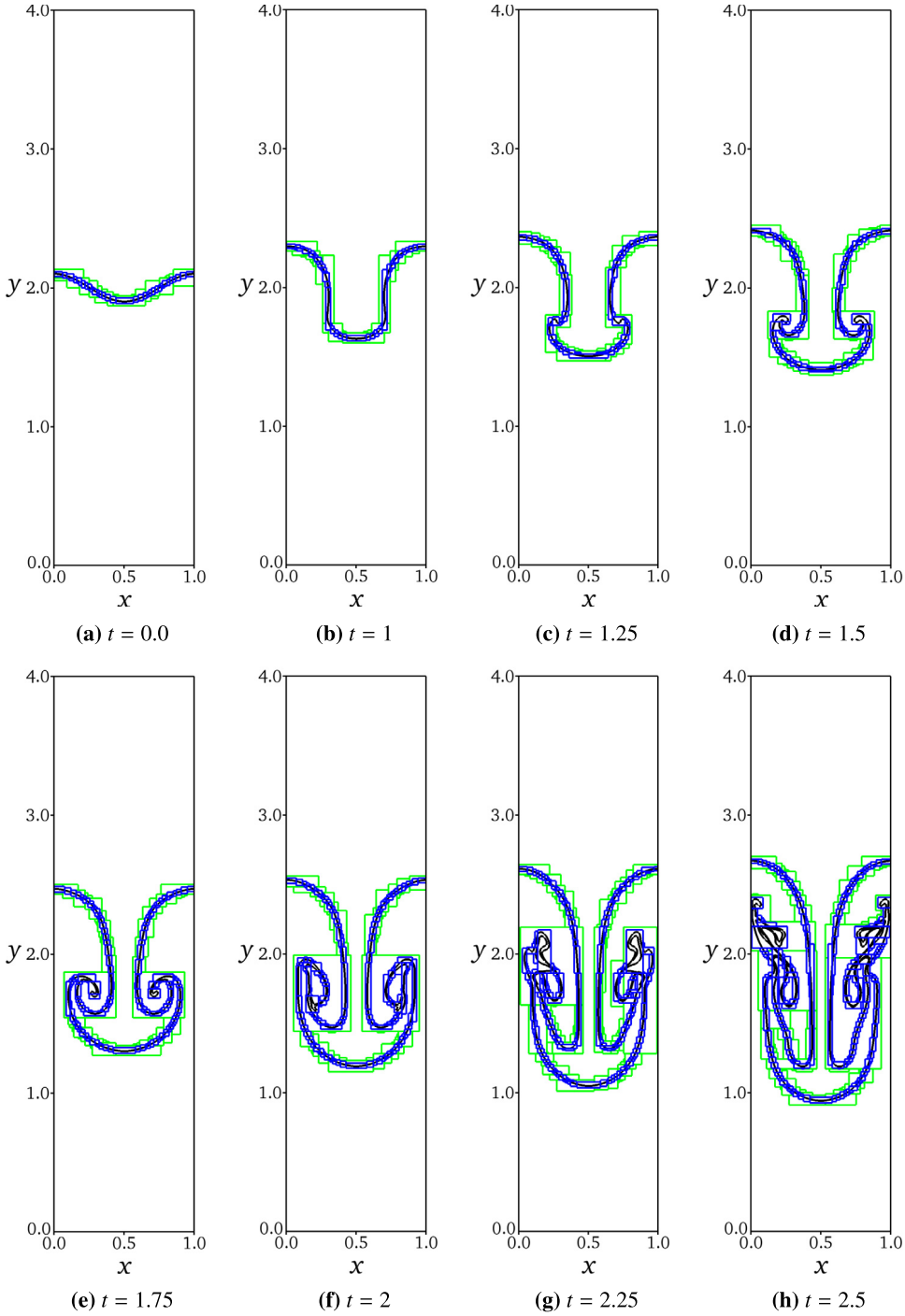

Fig. 21. Evolution of the air–water interface for the three-level non-subcycling Rayleigh–Taylor instability case (case 5 in Table 8). The black line represents the isoline of $\phi = 0$. The green and blue lines represent the patches on level 1 and level 2, respectively.

The above results show that our algorithms can accurately capture the transient locations of the fluid appearing in the Rayleigh–Taylor instability problem, both for the subcycling method and the non-subcycling method. The refined patches can help increase the resolution of small flow structures. This test further validates the capability of our proposed framework for simulating incompressible two-phase flow problems.

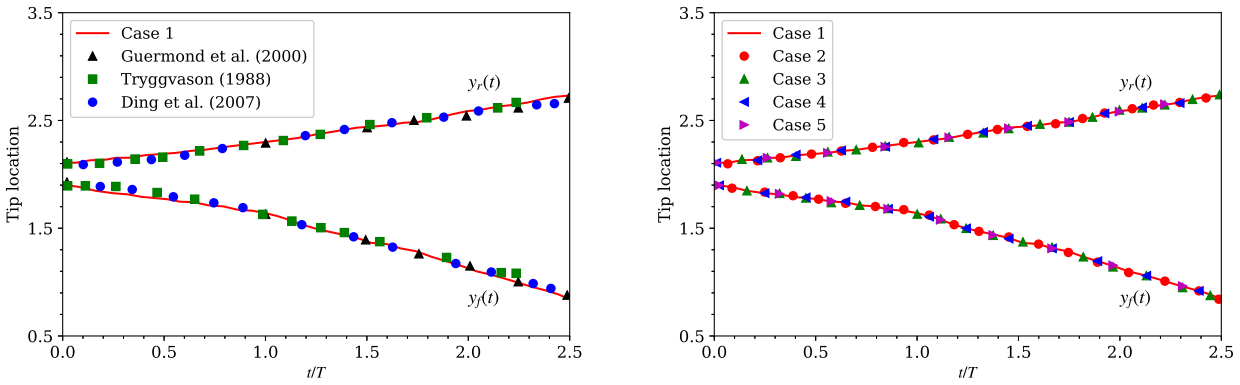

Fig. 22. Left: comparison of the tip locations of the falling fluid ($y_f(t)$) and the rising fluid ($y_r(t)$) between the single-level case (case 1) and the literature: (▲) Guermond and Quartapelle [76]; (■) Tryggvason [78]; (●) Ding et al. [77]. Right: comparison of the tip locations of the falling fluid $y_f(t)$ and the rising fluid $y_r(t)$ between the single-level case (case 1), the two-level subcycling case (case 2), the two-level non-subcycling case (case 3), the three-level subcycling case (case 4), and the three-level non-subcycling case (case 5).

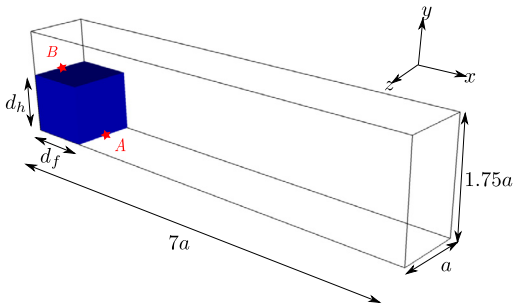

Fig. 23. Sketch of 3D dam breaking problem. A: dam front position; B: dam height position.

Table 9
Parameters for cases of the dam breaking problem.

Case No.	Grid number on level 0	l_{max}	Δt_0	Cycling method
1	$512 \times 128 \times 96$	0	1.25×10^{-4}	–
2	$256 \times 64 \times 48$	1	2.50×10^{-4}	Subcycling
3	$256 \times 64 \times 48$	1	1.25×10^{-4}	Non-subcycling
4	$128 \times 32 \times 24$	2	5.00×10^{-4}	Subcycling
5	$128 \times 32 \times 24$	2	1.25×10^{-4}	Non-subcycling

5.9. 3D dam breaking

This section investigates the 3D dam breaking, a dynamic and complex problem which is traditionally computationally expensive. Besides validating the adaptive two-phase flow algorithms for 3D problems, another objective is to compare the computational cost among the single-level, the subcycling, and the non-subcycling cases.

For this problem, a cubic water block with the side length $a = 5.715 \times 10^{-2}$ m is put at the left-bottom corner. The computational domain size is $[7a, a, 1.75a]$, as shown in Fig. 23. No-slip boundary conditions are imposed on the bottom wall, while all other walls are free-slip boundaries. The d_f and d_h in Fig. 23 refer to the dimensional front (point A) and the dimensional height (point B), respectively. The dimensionless front and height are then defined as $\tilde{d}_f = d_f/a$ and $\tilde{d}_h = d_h/a$, respectively. The gravity is in the $-y$ direction. Dimensionless parameters are set as $Re = \rho_1 U a/\mu_1 = 2950$, $Fr = U/\sqrt{ga} = 1.0$, $We = \rho_1 U^2 a/\sigma = 0.54$, $\lambda = \rho_2/\rho_1 = 0.0012$, and $\eta = \mu_2/\mu_1 = 0.016$. Table 9 gives the parameters of five simulation cases, where case 1 is the single-level case and all other cases are the multi-level cases. From case 2 to case 5, we refine the grid to $l_{max} = 2$ or 3 using either the subcycling or the non-subcycling method. The refinement criterion is based on the distance to the air–water interface, same as the Rayleigh–Taylor instability problem (section 5.8).

Fig. 24 compares the dimensionless front $\tilde{d}_f$ and dimensionless height $\tilde{d}_h$ of the single-level case (case 1) with previous experimental results and numerical results. Our results agree well with the literature. Fig. 25 shows that the time evolution of $\tilde{d}_f$ and $\tilde{d}_h$ for the above five cases, which again indicates the consistency of our numerical algorithms, where the subcycling cases, the non-subcycling cases, and the single-level case can obtain nearly the same accuracy. The evolution of the

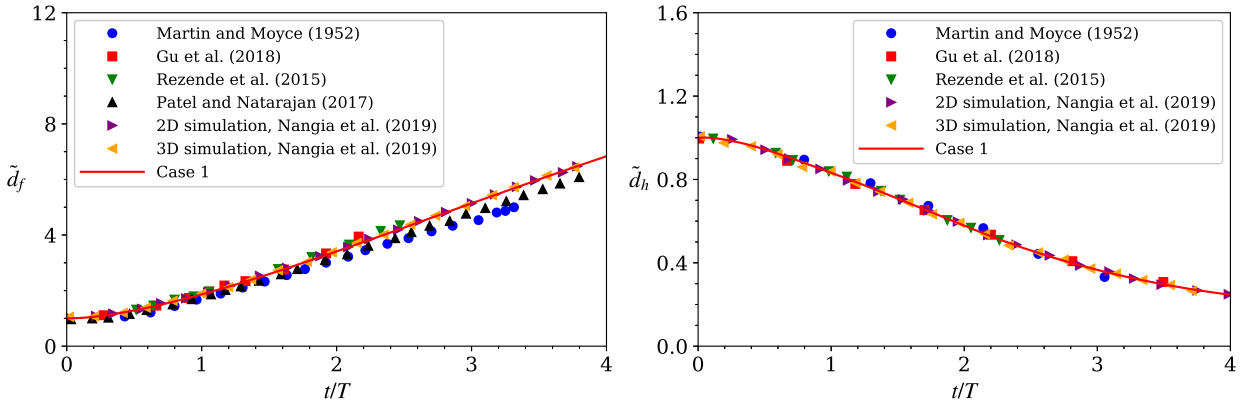

Fig. 24. Comparison of the evolution of the dimensionless front $\tilde{d}_f$ (left) and the dimensionless height $\tilde{d}_h$ (right) between the single-level case (case 1) and the literature: (●) Martin et al. [79]; (■) Gu et al. [80]; (▼) Rezende et al. [81]; (▲) Patel and Natarajan [82]; (▶) Nangia et al. [38]; (◀) Nangia et al. [38].

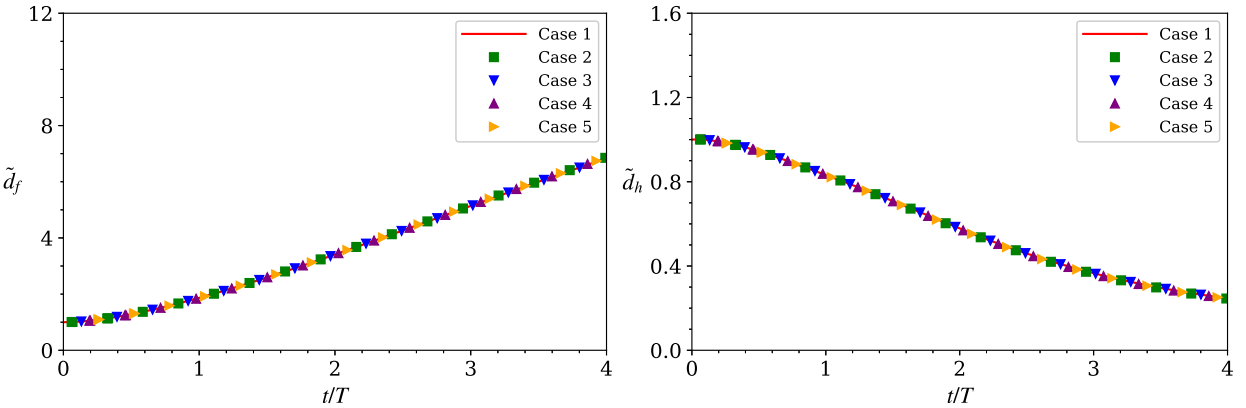

Fig. 25. Comparison of the evolution of the dimensionless front (left) $\tilde{d}_f$ and the dimensionless height $\tilde{d}_h$ (right) among the cases listed in Table 9.

breaking dam for the three-level subcycling case (case 4) is depicted in Fig. 26. Patches are dynamically refined around the interface as time evolves. The shape of the dam is consistent with the results in [38].

To compare the computational cost of different cases, we profile each case for $t/T = 0 - 0.3$ using 64 CPU cores on the Cray XE6m HPC machine without considering the I/O cost. Table 10 shows the total number of grid cells for different cases at $t/T = 0.1$. Compared with the cases with $l_{max} = 2$, the single-level case and the cases with $l_{max} = 1$ have nearly 6.32 times and 1.45 times more cells, respectively, which indicates that the adaptive refinement could reduce the total number of grid cells considerably. We emphasize that the time spent for each case also depends on the time step and the subcycling cases (case 2 and case 4) have less time steps than the non-subcycling cases (case 3 and case 5).

Table 11 compares the total wall clock time among different cases. Compared with the single-level case (case 1), it is seen that the two-level subcycling case (case 2) can obtain a 4.8 times speedup and thus save the computational cost. The three-level subcycling case (case 4) achieves more speedup (6.4 times) compared with the single-level case. In addition, when comparing the wall clock time of the subcycling cases (case 2 and case 4) with the corresponding non-subcycling cases (case 3 and case 5), we find that subcycling cases reduce more computational cost. The reason is that, compared to the non-subcycling method, the subcycling method can use larger time steps for the coarser levels.

Besides the total wall clock time, the percentages of the time spent on some key parts of the algorithms are also documented, which can help us to identify the most time-consuming parts for optimizing the algorithms in future research. As shown in Fig. 27, they include the MAC projection, viscous solver, level projection, and synchronization. Among them, the level projection takes the most time ($> 35\%$), followed by the MAC projection step ($\approx 30\%$). Therefore, optimization of the two projection algorithms is desired. At last, the part denoted as the "Others", including the regridding, the interpolation operations, and the multi-level re-initialization steps, only account for about 5% of the total computation time. This result shows that our multi-level re-initialization algorithm is an economical way to regularize the LS function on the multi-level grid.

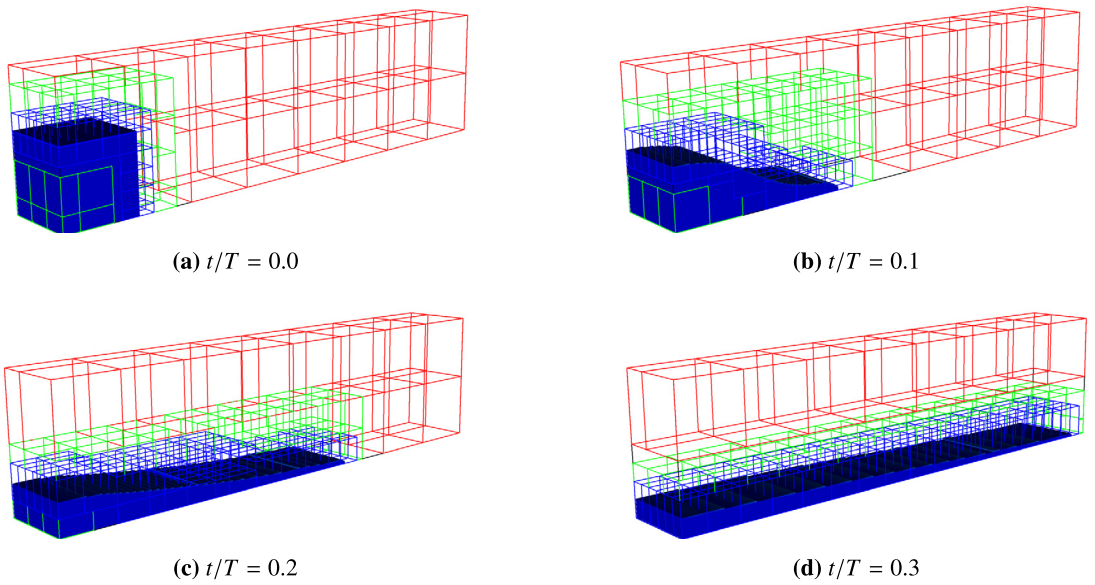

Fig. 26. Profiles of the breaking dam for the three-level subcycling case (case 4) at different time instants. The red, green, and blue lines represent patches on levels 0, 1, 2, respectively.

Table 10
Number of grid cells of the cases in the dam breaking problem at $t/T = 0.1$.

Case No.	Cells on level 0	Cells on level 1	Cells on level 2	Total cells	Total cells normalized by case 4
1	6,291,456	–	–	6,291,456	6.32
2	786,432	657,408	–	1,443,840	1.45
3	786,432	657,408	–	1,443,840	1.45
4	98,304	239,616	657,408	995,328	1
5	98,304	239,616	657,408	995,328	1

Table 11
Comparison of the total wall clock time T_{total} and the speedup among the single-level case (case 1), the two-level subcycling case (case 2), the two-level non-subcycling case (case 3), the three-level subcycling case (case 4), and the three-level non-subcycling case (case 5) for the dam breaking problem.

	Case 1	Case 2	Case 3	Case 4	Case 5
Total wall time (hrs)	1.67	0.35	0.50	0.26	0.38
Speedup compared with case 1	1.0	4.8	3.3	6.4	4.4

6. Conclusions

In this work, we have developed a collocated BSAMR framework with both the subcycling and non-subcycling advancement methods for the simulations of incompressible two-phase flows. The proposed multi-level advancement algorithm based on the level-by-level advancement method uses variables in both the valid and invalid regions and decouples the time advancement for different levels. Because of this decoupling, the time step constraint on the coarser levels is relaxed compared with that on the finest level if the subcycling method is used. On the other hand, the non-subcyling method avoids the time interpolation process across the different levels because data on all levels are located at the same time instant during the simulation.

Compared with the staggered grid and the semi-staggered grid, the collocated grid used here have several benefits. For example, the Godunov scheme, which is robust to for flows with a wide range of Reynolds number, can be implemented in a straightforward way when the collocated grid is used, as done in the present work. In addition, one set of interpolation schemes and average operations is used for all variables in the context of the collocated grid. Moreover, only the cell-centered MG solver is needed for the velocity and pressure fields.

The synchronization algorithm proposed in this work is the key to maintain the consistency of variables across multiple levels so that the variables can better represent the composite solution during the level-by-level advancement. When a fine level catches up with a coarse level, the cell-centered averaging sub-step first replaces the variables on the coarser levels with the more accurate solution on the finer levels. Then, the MAC synchronization and refluxing sub-step provide

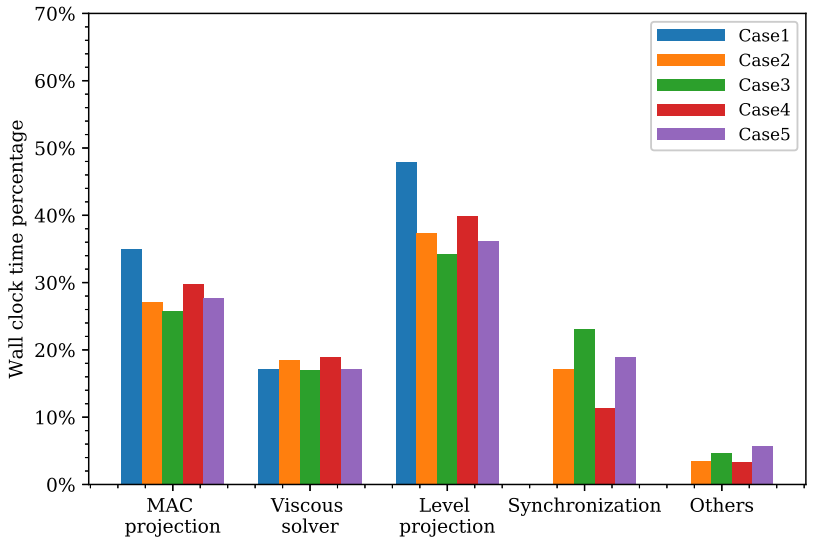

Fig. 27. Percentages of the wall clock time for different parts in the BSAMR algorithms. The cases considered include the single-level case (case 1), the two-level subcycling case (case 2), the two-level non-subcycling case (case 3), the three-level subcycling case (case 4), and the three-level non-subcycling case (case 5).

corrections to the multi-level solution from the velocity and flux registers to ensure the momentum conservation on the entire flow field.

In this work, we have also developed a novel re-initialization algorithm for the LS function on the multi-level grid to improve the accuracy of the two-phase interface capturing in the BSAMR framework. It employs an iteration technique to synchronize the LS function on two consecutive levels pair by pair. This algorithm leads to a substantial improvement in the mass conservation of the two-phase flow.

The accuracy and robustness of the computational framework are validated with several canonical tests. The results have shown that our numerical schemes obtain the second-order accuracy as designed and conserve the mass, momentum, and energy well. The subcycling and non-subcycling methods produce consistent and accurate results. We have also shown that the multi-level cases can achieve the same level of accuracy with fewer grid cells than the single-level fine-grid cases. In particular, for the 3D dam breaking problem, the multi-level simulation is able to capture the evolution of the dam accurately with substantial speedup compared with the single-level simulation. The synchronization and multi-level re-initialization algorithms developed in this work are also shown to be computationally efficient. Therefore, we conclude that the proposed AMR framework is promising for high-fidelity simulations of complex two-phase flows. In the future, we plan to extend this framework to support a high refining ratio between different levels.

CRediT authorship contribution statement

Yadong Zeng: Conceptualization, Methodology, Writing – original draft. **Anqing Xuan:** Formal analysis, Validation, Writing – review & editing. **Johannes Blaschke:** Software, Validation. **Lian Shen:** Funding acquisition, Project administration, Supervision, Writing – review & editing.

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.

Acknowledgements

Y. Z., A. X., and L. S. gratefully acknowledge the support to this work by the Office of Naval Research (N00014-17-1-2658 and N00014-19-1-2139) and National Science Foundation (OCE-1924799) on this work. Y. Z. would also like to gratefully thank the researchers in the Lawrence Berkeley National Lab (LBNL) for the discussions about the synchronization algorithms.

References

[1] A. Iafrati, F. De Vita, R. Verzicco, Effects of the wind on the breaking of modulated wave trains, Eur. J. Mech. B, Fluids 73 (2019) 6–23.

[2] Z. Yang, B.-Q. Deng, L. Shen, Direct numerical simulation of wind turbulence over breaking waves, J. Fluid Mech. 850 (2018) 120–155.
[3] V. Bertram, Practical Ship Hydrodynamics, Elsevier, 2011.
[4] J. Hua, J. Lou, Numerical simulation of bubble rising in viscous liquid, J. Comput. Phys. 222 (2007) 769–795.
[5] L. Deike, E. Ghabache, G. Liger-Belair, A.K. Das, S. Zaleski, S. Popinet, T. Séon, Dynamics of jets produced by bursting bubbles, Phys. Rev. Fluids 3 (2018) 013603.
[6] T. Ménard, S. Tanguy, A. Berlemont, Coupling level set/VOF/ghost fluid methods: validation and application to 3D simulation of the primary break-up of a liquid jet, Int. J. Multiph. Flow 33 (2007) 510–524.
[7] A. Prosperetti, G. Tryggvason, Computational Methods for Multiphase Flow, Cambridge University Press, 2009.
[8] M.J. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys. 53 (1984) 484–512.
[9] M.J. Berger, P. Colella, Local adaptive mesh refinement for shock hydrodynamics, J. Comput. Phys. 82 (1989) 64–84.
[10] Y.-H. Yu, Y. Li, Reynolds-averaged Navier–Stokes simulation of the heave performance of a two-body floating-point absorber wave energy system, Comput. Fluids 73 (2013) 104–114.
[11] Y. Zeng, L. Shen, Modelling Wave Energy Converter (WEC) pointer absorbers using AMR techniques with both subcycling and non-subcycling, Earth Space Sci. Open Arch. (2020) 1.
[12] S. Popinet, A quadtree-adaptive multigrid solver for the Serre–Green–Naghdi equations, J. Comput. Phys. 302 (2015) 336–358.
[13] S.L. Cornford, D.F. Martin, D.T. Graves, D.F. Ranken, A.M. Le Brocq, R.M. Gladstone, A.J. Payne, E.G. Ng, W.H. Lipscomb, Adaptive mesh, finite volume modeling of marine ice sheets, J. Comput. Phys. 232 (2013) 529–549.
[14] P. Mistani, A. Guittet, D. Bochkov, J. Schneider, D. Margetis, C. Ratsch, F. Gibou, The island dynamics model on parallel quadtree grids, J. Comput. Phys. 361 (2018) 150–166.
[15] M. Al-Marouf, R. Samtaney, A versatile embedded boundary adaptive mesh method for compressible flow in complex geometry, J. Comput. Phys. 337 (2017) 339–378.
[16] C.C. de Langavant, A. Guittet, M. Theillard, F. Temprano-Coleto, F. Gibou, Level-set simulations of soluble surfactant driven flows, J. Comput. Phys. 348 (2017) 271–297.
[17] L. Ding, C. Shu, H. Ding, N. Zhao, Stencil adaptive diffuse interface method for simulation of two-dimensional incompressible multiphase flows, Comput. Fluids 39 (2010) 936–944.
[18] V.K. Chalamalla, E. Santilli, A. Scotti, M. Jalali, S. Sarkar, SOMAR-LES: a framework for multi-scale modeling of turbulent stratified oceanic flows, Ocean Model. 120 (2017) 101–119.
[19] M. Zingale, A. Almgren, M.B. Sazo, V. Beckner, J. Bell, B. Friesen, A. Jacobs, M. Katz, C. Malone, A. Nonaka, et al., Meeting the Challenges of Modeling Astrophysical Thermonuclear Explosions: Castro, Maestro, and the AMReX Astrophysics Suite, J. Phys. Conf. Ser. 1031 (2018) 012024, IOP Publishing.
[20] A. Guittet, M. Theillard, F. Gibou, A stable projection method for the incompressible Navier–Stokes equations on arbitrary geometries and adaptive Quad/Octrees, J. Comput. Phys. 292 (2015) 215–238.
[21] M. Mirzadeh, A. Guittet, C. Burstedde, F. Gibou, Parallel level-set methods on adaptive tree-based grids, J. Comput. Phys. 322 (2016) 345–364.
[22] S. Popinet, Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries, J. Comput. Phys. 190 (2003) 572–600.
[23] A.S. Almgren, J.B. Bell, P. Colella, L.H. Howell, M.L. Welcome, A conservative adaptive projection method for the variable density incompressible Navier–Stokes equations, J. Comput. Phys. 142 (1998) 1–46.
[24] D.F. Martin, P. Colella, A cell-centered adaptive projection method for the incompressible Euler equations, J. Comput. Phys. 163 (2000) 271–312.
[25] D.F. Martin, P. Colella, D. Graves, A cell-centered adaptive projection method for the incompressible Navier–Stokes equations in three dimensions, J. Comput. Phys. 227 (2008) 1863–1886.
[26] M.L. Minion, A projection method for locally refined grids, J. Comput. Phys. 127 (1996) 158–178.
[27] Y. Zeng, L. Shen, 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, pp. S19–001.
[28] C. Burstedde, L.C. Wilcox, O. Ghattas, p4est: scalable algorithms for parallel adaptive mesh refinement on forests of octrees, SIAM J. Sci. Comput. 33 (2011) 1103–1133.
[29] T. Isaac, C. Burstedde, L.C. Wilcox, O. Ghattas, Recursive algorithms for distributed forests of octrees, SIAM J. Sci. Comput. 37 (2015) C497–C531.
[30] M. Williamschen, C.P. Groth, Parallel anisotropic block-based adaptive mesh refinement algorithm for three-dimensional flows, in: 21st AIAA Computational Fluid Dynamics Conference, 2013, p. 2442.
[31] C. Min, F. Gibou, A second order accurate level set method on non-graded adaptive Cartesian grids, J. Comput. Phys. 225 (2007) 300–321.
[32] M. Vanella, P. Rabenold, E. Balaras, A direct-forcing embedded-boundary method with adaptive mesh refinement for fluid–structure interaction problems, J. Comput. Phys. 229 (2010) 6427–6449.
[33] B.T. Gunney, R.W. Anderson, Advances in patch-based adaptive mesh refinement scalability, J. Parallel Distrib. Comput. 89 (2016) 65–84.
[34] C. Burstedde, D. Calhoun, K. Mandli, A.R. Terrel, ForestClaw: hybrid forest-of-octrees AMR for hyperbolic conservation laws, Parallel Comput. 25 (2014) 253–262.
[35] B.E. Griffith, R.D. Hornung, D.M. McQueen, C.S. Peskin, An adaptive, formally second order accurate version of the immersed boundary method, J. Comput. Phys. 223 (2007) 10–49.
[36] M. Sussman, A.S. Almgren, J.B. Bell, P. Colella, L.H. Howell, M.L. Welcome, An adaptive level set approach for incompressible two-phase flows, J. Comput. Phys. 148 (1999) 81–124.
[37] A.P.S. Bhalla, R. Bale, B.E. Griffith, N.A. Patankar, A unified mathematical framework and an adaptive numerical method for fluid–structure interaction with rigid, deforming, and elastic bodies, J. Comput. Phys. 250 (2013) 446–476.
[38] N. Nangia, B.E. Griffith, N.A. Patankar, A.P.S. Bhalla, A robust incompressible Navier–Stokes solver for high density ratio multiphase flows, J. Comput. Phys. 390 (2019) 548–594.
[39] M.R. Pivello, M.M. Villar, R. Serfaty, A.M. Roma, A.d. Silveira-Neto, A fully adaptive front tracking method for the simulation of two phase flows, Int. J. Multiph. Flow 58 (2014) 72–82.
[40] S.O. Unverdi, G. Tryggvason, A front-tracking method for viscous, incompressible, multi-fluid flows, J. Comput. Phys. 100 (1992) 25–37.
[41] C.W. Hirt, B.D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, J. Comput. Phys. 39 (1981) 201–225.
[42] M. Renardy, Y. Renardy, J. Li, Numerical simulation of moving contact line problems using a volume-of-fluid method, J. Comput. Phys. 171 (2001) 243–263.
[43] J.E. Pilliod Jr, E.G. Puckett, Second-order accurate volume-of-fluid algorithms for tracking material interfaces, J. Comput. Phys. 199 (2004) 465–502.
[44] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations, J. Comput. Phys. 79 (1988) 12–49.
[45] D. Adalsteinsson, J.A. Sethian, A fast level set method for propagating interfaces, J. Comput. Phys. 118 (1994).
[46] J.A. Sethian, A fast marching level set method for monotonically advancing fronts, Proc. Natl. Acad. Sci. USA 93 (1996) 1591–1595.
[47] F. Gibou, R. Fedkiw, S. Osher, A review of level-set methods and some recent applications, J. Comput. Phys. 353 (2018) 82–109.
[48] H. Kohno, T. Tanahashi, Numerical analysis of moving interfaces using a level set method coupled with adaptive mesh refinement, Int. J. Numer. Methods Fluids 45 (2004) 921–944.

[49] O. Antepara, N. Balcázar, J. Rigola, A. Oliva, Numerical study of rising bubbles with path instability using conservative level-set and adaptive mesh refinement, Comput. Fluids 187 (2019) 83–97.
[50] S. Popinet, An accurate adaptive solver for surface-tension-driven interfacial flows, J. Comput. Phys. 228 (2009) 5838–5866.
[51] W. Zhang, A. Almgren, V. Beckner, J. Bell, J. Blaschke, C. Chan, M. Day, B. Friesen, K. Gott, D. Graves, et al., AMReX: a framework for block-structured adaptive mesh refinement, J. Open Sour. Softw. 4 (2019).
[52] M. Sussman, E. Fatemi, An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow, SIAM J. Sci. Comput. 20 (1999) 1165–1191.
[53] M. Zingale, Introduction to Computational Astrophysical Hydrodynamics, Open-Astrophysics-Bookshelf, 2017.
[54] W.J. Rider, Approximate Projection Methods for Incompressible Flow: Implementation, Variants and Robustness, LANL Unclassified Report LA-UR-94-2000, Los Alamos National Laboratory, 1995.
[55] W.J. Rider, Filtering non-solenoidal modes in numerical solutions of incompressible flows, Int. J. Numer. Methods Fluids 28 (1998) 789–814.
[56] A.S. Almgren, J.B. Bell, W.Y. Crutchfield, Approximate projection methods: Part I. Inviscid analysis, SIAM J. Sci. Comput. 22 (2000) 1139–1159.
[57] P. Colella, A multidimensional second order Godunov scheme for conservation laws, J. Comput. Phys. 87 (1990) 171–200.
[58] J.B. Bell, P. Colella, H.M. Glaz, A second-order projection method for the incompressible Navier–Stokes equations, J. Comput. Phys. 85 (1989) 257–283.
[59] M.F. Lal, A Projection Method for Reacting Flow in the Zero Mach Number Limit, Ph.D. thesis, University of California at Berkeley, 1993.
[60] A.S. Almgren, J.B. Bell, W.G. Szymczak, A numerical method for the incompressible Navier–Stokes equations based on an approximate projection, SIAM J. Sci. Comput. 17 (1996) 358–369.
[61] M.F. Lai, A projection method for reacting flow in the zero Mach number limit, Ph.D. thesis, University of California, Berkeley, 1993.
[62] R.D. Guy, A.L. Fogelson, Stability of approximate projection methods on cell-centered grids, J. Comput. Phys. 203 (2005) 517–538.
[63] C. Min, F. Gibou, A second order accurate projection method for the incompressible Navier–Stokes equations on non-graded adaptive grids, J. Comput. Phys. 219 (2006) 912–929.
[64] P. Colella, D.T. Graves, T. Ligocki, D. Martin, D. Modiano, D. Serafini, B. Van Straalen, Chombo software package for AMR applications design document, Available at the Chombo website: http://seesar.lbl.gov/ANAG/chombo/(September2008), 2009.
[65] L.H. Howell, J.B. Bell, An adaptive mesh projection method for viscous incompressible flow, SIAM J. Sci. Comput. 18 (1997) 996–1013.
[66] F. Gibou, L. Chen, D. Nguyen, S. Banerjee, A level set based sharp interface method for the multiphase incompressible Navier–Stokes equations with phase change, J. Comput. Phys. 222 (2007) 536–555.
[67] D.L. Brown, R. Cortez, M.L. Minion, Accurate projection methods for the incompressible Navier–Stokes equations, J. Comput. Phys. 168 (2001) 464–499.
[68] N. Nangia, N.A. Patankar, A.P.S. Bhalla, A DLM immersed boundary method based wave-structure interaction solver for high density ratio multiphase flows, J. Comput. Phys. 398 (2019) 108804.
[69] Z. Huang, G. Lin, A.M. Ardekani, A mixed upwind/central WENO scheme for incompressible two-phase flows, J. Comput. Phys. 387 (2019) 455–480.
[70] S.T. Zalesak, Fully multidimensional flux-corrected transport algorithms for fluids, J. Comput. Phys. 31 (1979) 335–362.
[71] M. Sussman, E.G. Puckett, A coupled level set and volume-of-fluid method for computing 3D and axisymmetric incompressible two-phase flows, J. Comput. Phys. 162 (2000) 301–337.
[72] H. Lamb, Hydrodynamics, Cambridge University Press, 1993.
[73] S. Xie, D. Yang, Y. Liu, L. Shen, Simulation-based study of wind loads on semi-submersed object in ocean wave field, Phys. Fluids 28 (2016) 015106.
[74] J. Hnat, J. Buckmaster, Spherical cap bubbles and skirt formation, Phys. Fluids 19 (1976) 182–194.
[75] G. Ryskin, L. Leal, Numerical solution of free-boundary problems in fluid mechanics. Part 2. Buoyancy-driven motion of a gas bubble through a quiescent liquid, J. Fluid Mech. 148 (1984) 19–35.
[76] J.-L. Guermond, L. Quartapelle, A projection FEM for variable density incompressible flows, J. Comput. Phys. 165 (2000) 167–188.
[77] H. Ding, P.D. Spelt, C. Shu, Diffuse interface model for incompressible two-phase flows with large density ratios, J. Comput. Phys. 226 (2007) 2078–2095.
[78] G. Tryggvason, Numerical simulations of the Rayleigh–Taylor instability, J. Comput. Phys. 75 (1988) 253–282.
[79] J.C. Martin, W.J. Moyce, J. Martin, W. Moyce, W.G. Penney, A. Price, C. Thornhill, Part IV. An experimental study of the collapse of liquid columns on a rigid horizontal plane, Philos. Trans. R. Soc. Lond. A 244 (1952) 312–324.
[80] Z. Gu, H. Wen, C. Yu, T.W. Sheu, Interface-preserving level set method for simulating dam-break flows, J. Comput. Phys. 374 (2018) 249–280.
[81] R.V. Rezende, R.A. Almeida, A.A.U. de Souza, S.M.G.U. Souza, A two-fluid model with a tensor closure model approach for free surface flow simulations, Chem. Eng. Sci. 122 (2015) 596–613.
[82] J.K. Patel, G. Natarajan, A novel consistent and well-balanced algorithm for simulations of multiphase flows on unstructured grids, J. Comput. Phys. 350 (2017) 207–236.