Journal of Computational Physics 229 (2010) 2573–2604

Contents lists available at ScienceDirect

Journal of Computational Physics

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

A novel coupled level set and volume of fluid method for sharp interface capturing on 3D tetrahedral grids

Xin Lv [a,*], Qingping Zou [a], Yong Zhao [b], Dominic Reeve [a]

[a] Centre for Coastal Dynamics and Engineering, School of Engineering, University of Plymouth, Devon PL4 8AA, United Kingdom
[b] Nanyang Technological University, School of MPE, Nanyang Avenue, 639798 Singapore, Singapore

ARTICLE INFO

Article history:
Received 17 December 2007
Received in revised form 23 September 2009
Accepted 3 December 2009
Available online 14 December 2009

Keywords:
VOF
Level set
Free surface
Tetrahedral grid
Finite volume method

ABSTRACT

We present a new three-dimensional hybrid level set (LS) and volume of fluid (VOF) method for free surface flow simulations on tetrahedral grids. At each time step, we evolve both the level set function and the volume fraction. The level set function is evolved by solving the level set advection equation using a second-order characteristic based finite volume method. The volume fraction advection is performed using a bounded compressive normalized variable diagram (NVD) based scheme. The interface is reconstructed based on both the level set and the volume fraction information. The novelty of the method lies in that we use an analytic method for finding the intercepts on tetrahedral grids, which makes interface reconstruction efficient and conserves volume of fluid exactly. Furthermore, the advection of volume fraction makes use of the NVD concept and switches between different high resolution differencing schemes to yield a bounded scalar field, and to preserve both smoothness and sharp definition of the interface. The method is coupled to a well validated finite volume based Navier–Stokes incompressible flow solver. The code validation shows that our method can be employed to resolve complex interface changes efficiently and accurately. In addition, the centroid and intercept data available as a by-product of the proposed interface reconstruction scheme can be used directly in near-interface sub-grid models in large eddy simulation.

1. Introduction

Flow with free surface is common phenomena encountered in a wide variety of environmental, geophysical and engineering situations. Simulating free surface flow, however, remains a formidable challenge because non-linear boundary conditions are required to be satisfied on an arbitrarily moving surface whose position is not prescribed a priori.

Mainly two approaches are used to track the free surface, the Lagrangian and Eulerian. Both approaches have advantages and weaknesses, and the focus of this paper will be on the latter option. Tracking the free surface using the Eulerian description employs an equation for the density:

$$\frac{\partial \rho}{\partial t} + u \cdot \nabla \rho = 0, \tag{1}$$

which is referred to as the transport equation. The density $\rho(x,t)$ can be either continuous or discontinuous. By tracking the change of ρ it is possible to identify the location of the free surface. Direct numerical solution of Eq. (1) leads to excessive numerical diffusion. As a result, the free surface will become thicker and thicker and eventually become diffused into the

* Corresponding author.
E-mail address: xin.lv@plymouth.ac.uk (X. Lv).

0021-9991/$ - see front matter Crown Copyright © 2009 Published by Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.12.005

whole domain. Hence other approaches have been used for such a purpose. Nichols et al. [1] developed the volume of fluid (VOF) method to track the free surface. By using the transport equation and assuming that density is everywhere constant ($\rho = \rho_0$) in the flow domain and zero ($\rho = 0$) in the air domain, it is possible to normalize the transport equation by ρ_0. Defining $F = \rho/\rho_0$ as the fractional volume of fluid function (VOF function), the transport equation becomes:

$$\frac{\partial F}{\partial t} + u \cdot \nabla F = 0. \tag{2}$$

F is a step function, where $F = 1$ in cells containing fluid and $F = 0$ in cells containing air. In free surface cells, the value of F ranges between (0–1). Therefore, by tracking the VOF function F, one can identify the free surface elements at any time step. He numerical solution of Eq. (2) requires some care in order to avoid introducing too much numerical diffusion while maintaining the accuracy of the scheme. Hirt and Nichols [2] proposed the donor–acceptor method, which is the simplest treatment for controlling the numerical diffusion during the advection of the VOF function F. However, it is only first-order accurate. The inaccuracy in the donor–acceptor method is primarily caused by the oversimplification of the free surface geometry during the advection process. In [3], Youngs introduced a more accurate way to reconstruct the free surface by defining the free surface slope within each surface cell according to the gradient of the VOF function F and the intercepts with the slope and the volume of fluid of the element. The resulting free surface is discontinuous at the cell faces, since each element has its own free surface slope. Ashgriz and Poo [4] proposed a variant of this technique for two-dimensional problems which reconstructs the sloping interface at the cell boundaries, avoiding discontinuities in the free surface and providing more accurate results. Although the sloping method is in general more accurate than the donor–acceptor method, it still suffers from a few limitations. First, the sloping interface reconstruction and consequent advection is much more computationally intensive than that of the donor–acceptor method and it is more difficult to code. Furthermore, the extension of the sloping interface method from two dimensions to three dimensions needs much more effort than that of the donor–acceptor method and it is subject to numerical uncertainties that have not been tested and discussed fully. The difficulty in reconstructing the free surface accurately and efficiently prevents the practical application of the sloping interface method, especially in three-dimensional cases. Welch et al. [5] proposed a method which employed a sub-grid counting procedure to determine the accurate location of the interface and the correct advective flux, however, this technique has yet to attract widespread use.

VOF methods have proved to be robust and relatively easy to code, but still have drawbacks [18,44–46]. The overall accuracy of this method relies heavily on the performance of its interface reconstruction scheme. Most currently available reconstruction schemes are developed exclusively for 2D Cartesians meshes. In these methods the cell shapes, usually rectangular, are implicitly included in the reconstruction of the interface. Consequently, it is difficult to extend these methods to arbitrary complex meshes. Extensions to three-dimensional calculations pose similar difficulties. More discussion on this topic can be found in [7]. The interface propagation is also a problem, especially in multi-dimensional problems where an explicit split operator technique (see [3]) is often employed. The basic idea of operator splitting is to apply the one-dimensional equation in separate steps for each of the coordinate directions. This limits the implementation to structured meshes in which the faces of the control volume are aligned with the coordinate axes.

The choice of volume fraction as a phase indicator is a popular one but is prone to problems associated with the advection of a step function across a mesh that have to be overcome. Namely, how to advect the interface without diffusing, dispersing, or wrinkling it; while in the meantime maintaining the boundedness of the F function to avoid the non-physical deformation of the interface shape.

Recently, several strategies to sharpen the interface using VOF methods have been developed. In [8], a compressive discretization scheme called CICSAM (Compressive Interface Capturing Scheme for Arbitrary Meshes) is developed. It makes use of the normalized variable diagram (NVD) concept [10] and switches between different differencing schemes to yield a bounded scalar field, and to preserve both smoothness and sharp definition (over one or two computational cells) of the interface. The scheme is developed in the context of multi-dimensional applications, avoiding the need to use operator splitting. The merit of this method lies in that it can be easily implemented on arbitrary complex meshes. However, the thickness of the free surface still cannot be maintained as constant, and its actual performance depends on the local quality of the grid.

Another technique for interface tracking is the level set method (Sussman et al. [11]). By substituting ρ with the smooth-level set function φ Eq. (2) becomes

$$\frac{\partial \varphi}{\partial t} + u \cdot \nabla \varphi = 0. \tag{3}$$

The function φ gives the normal distance from the interface, in which its original value is set to zero at the interface. By tracking the zero value of φ based on Eq. (3), one may identify the interface during the computation. Because the set level function ϕ is a smooth function, it is much easier to compute and it induces much less numerical diffusion. But it has been shown [6] that this method is subjected to significant mass loss under complicated situations since the scheme does not explicitly impose mass conservation. Lots of studies have been performed by many researchers to solve this problem. Sussman and Puckett combined their method with a VOF method in order to overcome problems with mass conservation [13]. But this improvement is too complex to be implemented in three-dimensional solvers and will pose problems for arbitrary complex grids. In the level set method, maintaining φ as a distance function is essential for providing the interface with a width fixed in time, more importantly, it is very crucial for mass conservation. The technique can be improved further by

'reinitialising' the distance function repeatedly [12]. Conventional routines for reinitializing a distance function have to explicitly find the contour $\varphi = 0$ and reset φ at all grid points close to the front. This can distort the front (e.g. mass loss, non-physical deformation) depending on how one reconstructs the shape of the front ($\varphi = 0$). Traditional the distance reinitialization is achieved by solving the following equation in an iterative manner:

$$\begin{aligned} &\frac{\partial \varphi}{\partial t} - S(\varphi_0)\left(1 - \sqrt{\sum_{i=1}^{k}\left(\frac{\partial \varphi}{\partial x_i}\right)^2}\right) = 0, \\ &S(\varphi_0) = \frac{\varphi_0}{\sqrt{\varphi_0^2 + \varepsilon^2}} \end{aligned} \tag{4}$$

In Eq. (4), φ_0 is the initial distribution of the level set function before reinitialization, and ε a small number to avoid dividing by zero, usually chosen as the local grid length. This procedure avoids finding the interface and has proved to be efficient to implement in Cartesians grid solvers. It must be executed after every time step in order to keep φ as a distance function. In practice, we have found that this algorithm will pose difficulties when implemented in an unstructured grid solver, especially in a finite volume scheme based solver. Enright et al. proposed a hybrid particle level set method to improve the interface capturing [41]. In [42,43], Kriess and Olsson constructed a modified level set method with built in conservation. Where, the level set function ϕ was a regularized characteristic function. And a reinitialization procedure, formulated as a conservation law, was used to preserve the smooth profile of the regularized characteristic function.

The ghost fluid method (GFM) was developed based on level set method in its original form [51]. Essentially, GFM exploits the concept of ghost and real fluid cells and manages them with an overlapping Schwarz like numerical procedure. In the material interface region, it sets the values of the pressure and normal velocity in the ghost fluid cells to those in the real fluid cells. To eliminate an otherwise spurious "over-heating" phenomenon, it computes the density of the ghost fluid using an isobaric fix technique. As explained in [51], this isobaric fix requires the solution of yet another auxiliary partial differential equation and therefore increases further the computational complexity of the method. Also as pointed out by some researchers (for example [53]), the original GFM fails to solve some air/water problems of interest due to the large density ratio. In [55], Kang and the co-authors extended GFM to multiphase incompressible flow including the effects of viscosity, surface tension and gravity. The novelty of their approach is that they incorporated the boundary condition capturing approach for the variable coefficient Poisson equation developed in [54] to treat the interface in a sharp fashion rather than numerical smearing. They have showed improved accuracy and robustness of their approach against the original GFM.

In [56], the authors present an enhanced resolution capturing method for topologically complex two and three-dimensional incompressible free surface flows. The method is based upon the level set method of Osher and Sethian to represent the interface combined with two recent advances in the treatment of the interface, a second-order accurate discretization of the Dirichlet pressure boundary condition at the free surface [57] and the use of massless marker particles to enhance the resolution of the interface through the use of the particle level set method [41].

In [14], a novel three-dimensional problem formulation is introduced for the simulation of turbulent interfacial multifluid flows. The strategy is built around the large eddy simulation (LES) concept and provides two main features: (i) a reconstructed distance function (RDF) is introduced to define a level set interface-normal length scale, and (ii) an interfacial shear velocity is defined on the distance function support for further use in near-interface transport models. Their solution algorithm uses VOF with piecewise planar interface reconstructions on a twice-as-fine mesh, and infers the convective mass fluxes from the interface solution for momentum conservation. The procedure provides the interfacial shear velocity defined on the distance function support to accommodate the asymptotic behaviour of turbulence approaching the interface in a proximity-dependent manner. Provided with highly accurate distance function data, the scheme generates near-interface damping functions that are second-order accurate and independent of interface orientation. Their algorithm is developed for the structured Cartesians grids and discussion on extension to unstructured grids is not covered. However, the basic idea employed in their study is borrowed in current work.

In this study, a novel coupled level set/VOF method for interfacial flow simulations on three-dimensional unstructured tetrahedral grids is presented. The rest of the paper is organized as follows. The details on analytical piecewise linear interface reconstruction (PLIC) and distance reinitialization for level set function are presented in the next section. In Section 3 we describe the governing equations and numerical methods for Navier–Stokes incompressible flow, as well as the evolution for the level set function and the volume fraction. In Section 4, the issues concerning the performance and robustness of the proposed free surface scheme are discussed. The coupling of free surface scheme with IMM is also discussed in Section 5. Code validation and results are presented in Section 6. The conclusion and some plans on future work are given in Section 7.

2. Analytic PLIC and distance reinitialization

2.1. Analytic PLIC on 3D tetrahedral grids

Analytical relations connecting linear interfaces and volume fractions in structured rectangular grids have been presented in [14]. This was extended in [15] to analytical relations connecting linear interfaces and volume fractions in triangular and tetrahedral grids, i.e., given the interface normal vector in a cell, how to find the unique linear segment which also truncates

the cell by the given volume fraction. The merit of this method is that the analytic formulations eliminate the need to iterate, and thus reduce computation time. In addition, the volume of fluid is exactly conserved during interface reconstruction. This method serves as the basis of our interface reconstruction algorithm.

In a three-dimensional grid, a reconstructed linear interface in each tetrahedral cell is a polygonal segment of a plane, which can be represented by the vertices of the polygon (see Fig. 1). Consider an arbitrary tetrahedral cell of a given volume fraction F and a linear interface with a given unit normal vector $\vec{n}$, which can be easily computed from the level set equation define in Eq. (20) by $\vec{n} = \left\{\frac{\partial\phi}{\partial x}, \frac{\partial\phi}{\partial y}, \frac{\partial\phi}{\partial z}\right\}$. The vertices of the tetrahedron are denoted by A, B, C, and D in such an order that we have

$$0 = P_A \leqslant P_B \leqslant P_C \leqslant P_D. \tag{5}$$

The exact meaning of the symbols in Eq. (5) can be clearly explained using Fig. (1). Where plane μ, whose normal vector is $\vec{n}$, passes through vertex A, P_B, P_C and P_D the respective normal distance from vertex B, C and D. With relation Eq. (5) satisfied, the intersection of a linear interface with a tetrahedral grid cell can be categorized into three regimes, which are shown in Fig. 2. Following [15], the intercepts of the planar interface with the tetrahedral cell $ABCD$ can be determined as follows.

For regime I [see Fig. 2(a)], the interface polygon is a triangle with vertices G, H, and I. Their coordinates can be computed from

$$\begin{aligned} X_G &= X_A + \left(\frac{f}{f_B}\right)^{1/3} \frac{P_B}{P_D} \vec{AD}, \\ X_H &= X_A + \left(\frac{f}{f_B}\right)^{1/3} \frac{P_B}{P_C} \vec{AC}, \\ X_I &= X_A + \left(\frac{f}{f_B}\right)^{1/3} \vec{AB}. \end{aligned} \tag{6}$$

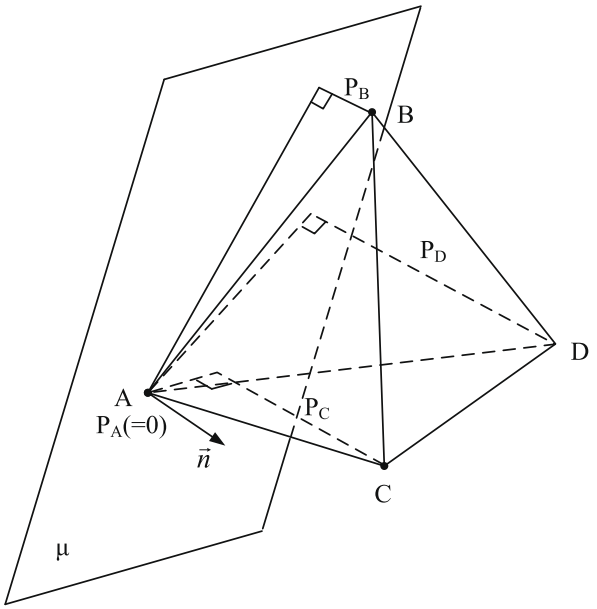


Fig. 1. Schematic of the vertices ordering for a tetrahedral cell.

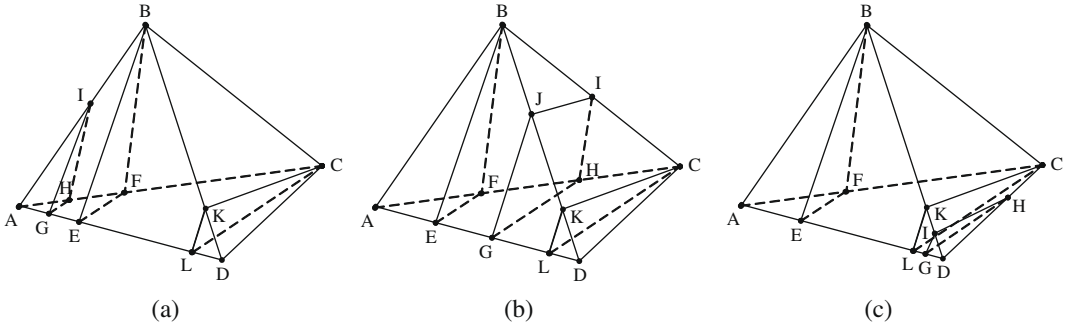


Fig. 2. Intersection of a planar interface with a tetrahedral grid cell. $BEF \perp \vec{n}$ and $KLC \perp \vec{n}$. G, H, I and J are the intercepts of planar interface with the cell. (a) Regime I. $GHI \perp \vec{n}$; (b) regime II. $GHIJ \perp \vec{n}$; and (c) regime III. $GHI \perp \vec{n}$.

For regime III [see Fig. 2(c)], the coordinates for the three intercepts G, H, and I can be determined by

$$\begin{aligned} X_G &= X_D + \left(\frac{1-f}{f_C}\right)^{1/3} \frac{P_D - P_C}{P_D} \vec{DA}, \\ X_H &= X_D + \left(\frac{1-f}{f_C}\right)^{1/3} \vec{DC}, \\ X_I &= X_D + \left(\frac{1-f}{f_C}\right)^{1/3} \frac{P_D - P_C}{P_D - P_B} \vec{DB}. \end{aligned} \tag{7}$$

For regime II [see Fig. 2(b)], the coordinates for the four intercepts G, H, I and J can be determined by

$$\begin{aligned} X_G &= X_A + \frac{P_B}{P_D} \vec{AD} + \alpha \frac{P_C - P_B}{P_D} \vec{AD}, \\ X_H &= X_C + (1-\alpha) \frac{P_C - P_B}{P_C} \vec{CA}, \\ X_I &= X_B + \alpha \vec{BC}, \\ X_I &= X_B + \alpha \frac{P_C - P_B}{P_D - P_B} \vec{BD}. \end{aligned} \tag{8}$$

And

$$\begin{aligned} a &= -\frac{(P_C - P_B)^2}{P_D} \left(\frac{1}{P_C} + \frac{1}{P_D - P_B}\right), \\ b &= \frac{3(P_C - P_B)^2}{P_D P_C}, \\ c &= \frac{3P_B(P_C - P_B)}{P_D P_C}, \\ d &= -\frac{P_B^2}{P_D P_C} - f, \\ \theta &= \frac{1}{3} \arccos\left(-\frac{d/a + 2b^3/(27a^3) - bc/(3a^2)}{2\sqrt{-\left(\frac{c/a - b^2/(3a^2)}{3}\right)^3}} \right), \\ \alpha &= \sqrt{-\frac{c/a - b^2/(3a^2)}{3}} \left(\sqrt{3}\sin\theta - \cos\theta\right) - \frac{b}{3a}. \end{aligned} \tag{9}$$

In the above equations, f is the volume fraction associated with cell $ABCD$, f_C and f_B are defined as

$$\begin{aligned} f_B &= \left(\frac{P_B}{P_C}\right)^2 \frac{P_C}{P_D}, \\ f_C &= \left(\frac{P_D - P_C}{P_D - P_B}\right)^2 \frac{P_D - P_B}{P_D}. \end{aligned} \tag{10}$$

In practice, we have tested this analytic method on a wide range of tetrahedral grids, from very coarse one to highly refined one, and found no problems, even on the meshes with many badly shaped cells. Some special cases may arise and need special treatment as mentioned in [15]. The method is quite efficient compared to its iterative counterparts. The only parameters required are the volume fraction value and the interface normal vector, which should be associated with the centroid of the cell. It is quite straightforward to implement in a cell-centered method. For vertex-centered schemes, such as in our case, it may pose some problems, which will be addressed in Section 3.

2.2. Distance reinitialization for level set function

After the piecewise linear interfaces are constructed for every interfacial cell. The reinitialization of the distance function can be performed in a straightforward manner. For each grid point, its signed distance to the interface can be easily calculated in three steps. Firstly, the initial sign of the distance function is recorded. And then the shortest distance between current point and the reconstructed interfaces is assigned to this point. Lastly, multiply the recorded sign with the shortest distance to ensure the sign of its distance function remains unchanged. The calculation of the shortest distance in step two deserves further discussion. There are a few ways to fulfill this task. As shown in Fig. 3.

In either way listed in Fig. 3, intercepts are necessary for computing the centroid point of any VOF interface plane. However, the use of intercepts when determining the shortest distance represents an increase in computation of threefold or

more. Intercepts of a PLIC VOF interface reconstruction may often be closer to the vertex location A than the centroid from the same plane, but the locations of intercepts are also sensitive to plane orientation. In contrast, the VOF plane centroid is far less sensitive while the free surface is evolving. For these reasons, our approach to distance reinitialization focuses on centroid data, and ignores intercept data. In example calculations we have found that the practical difference between these two methods is negligible. Only a few sub-layers of grid points near the VOF interface contribute to the accuracy of the level set method, as reported in many other publications (e.g. [11–13]).

3. Governing equations and numerical methods

3.1. Governing equations for N–S incompressible flow and free surface evolution

The non-dimensional governing 3D equations, modified by the artificial compression method (ACM) [16], are given as

$$\frac{1}{\beta}\frac{\partial p}{\partial \tau}+\nabla\cdot\vec{\mathbf{U}}=0, \tag{11}$$

$$\frac{\partial u}{\partial \tau}+\frac{\partial u}{\partial t}+\nabla\cdot(u\vec{\mathbf{U}})=-\frac{1}{\rho}\frac{\partial p}{\partial x}+\frac{1}{\rho Re}\cdot\nabla^2\cdot u+\frac{1}{\rho}F_{Sx}+F_{gx}, \tag{12}$$

$$\frac{\partial v}{\partial \tau}+\frac{\partial v}{\partial t}+\nabla\cdot(v\vec{\mathbf{U}})=-\frac{1}{\rho}\frac{\partial p}{\partial y}+\frac{1}{\rho \mathrm{Re}}\cdot\nabla^2\cdot v+\frac{1}{\rho}F_{Sy}+F_{gy}, \tag{13}$$

$$\frac{\partial w}{\partial \tau}+\frac{\partial w}{\partial t}+\nabla\cdot(w\vec{\mathbf{U}})=-\frac{1}{\rho}\frac{\partial p}{\partial z}+\frac{1}{\rho \mathrm{Re}}\cdot\nabla^2\cdot w+\frac{1}{\rho}F_{Sz}+F_{gz}, \tag{14}$$

$$\frac{\partial \phi}{\partial \tau}+\frac{\partial \phi}{\partial t}+\nabla\cdot(\phi\vec{\mathbf{U}})=0, \tag{15}$$

$$\frac{\partial F}{\partial t}+\nabla(F\cdot\vec{U})=0, \tag{16}$$

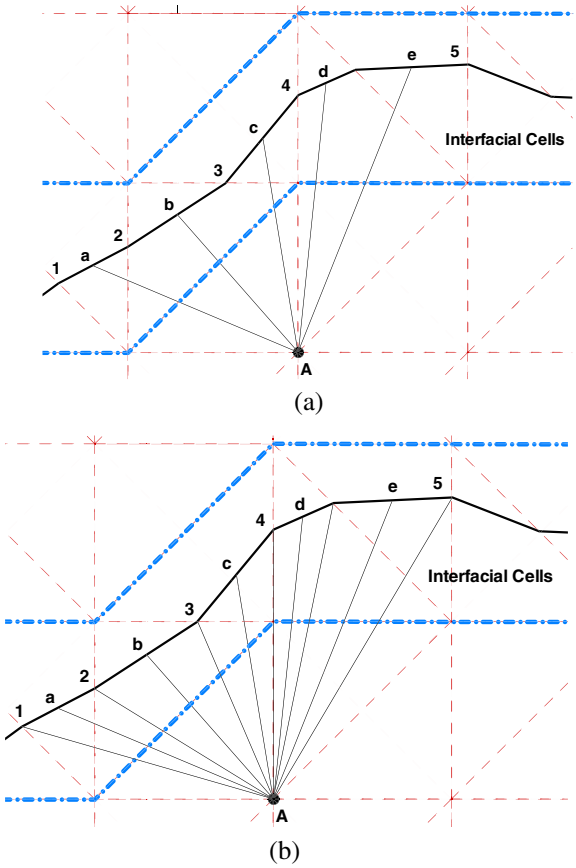


Fig. 3. 2D schematic of determination of the shortest distance from a vertex to the reconstructed planar interfaces. Numbers represent the computed intercepts. Characters represent the centroids of the interfaces. (a) Only centroids are included in the distance computation; (b) centroids and intercepts are included in the distance computation.

where $\vec{\mathbf{U}} = u\cdot\vec{\boldsymbol{i}} + v\cdot\vec{\boldsymbol{j}} + w\cdot\vec{\boldsymbol{k}}$. Please be noted that in current method, the free surface evolution is governed by both level set equation ϕ [Eq. (15)] and the volume of fluid equation F [Eq. (16)]. The detailed description on evolving F will be given in Section 3.6.

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

$$(x,y,z) = \left(\frac{x^*}{L^*},\frac{y^*}{L^*},\frac{z^*}{L^*}\right);\quad (u,v,w) = \left(\frac{u^*}{U^*_\infty},\frac{v^*}{U^*_\infty},\frac{w^*}{U^*_\infty}\right);\quad \rho = \frac{\rho^*}{\rho^*_\infty};\quad t = \frac{t^*}{L^*/U^*_\infty};\quad p = \frac{p^*-p_o}{\rho^*_\infty\left(U^*_\infty\right)^2};\quad \mathrm{Re} = \frac{\rho^*_\infty U^*_\infty L^*}{\mu^*_a};$$

$$Fr = \frac{U^*_\infty}{\sqrt{gL^*}};\quad We = \frac{\rho^*_\infty\left(U^*_\infty\right)^2 L^*}{\sigma^*_\infty},$$

where L^* denotes the reference length and U^*_∞ denotes the reference velocity. Terms with superscript * indicate dimensional quantities, and the subscript ∞ indicates the other reference terms.

Eqs. (11)–(15) can be expressed in non-dimensional vector form as follows:

$$\mathbf{C}\frac{\partial\mathbf{W}}{\partial\tau} + \mathbf{K}\frac{\partial\mathbf{W}}{\partial \mathrm{t}} + \nabla\cdot\vec{\mathbf{F}}_c = \nabla\cdot\vec{\mathbf{F}}_v + \vec{\mathbf{S}}, \tag{17}$$

where

$$\mathbf{W} = \begin{bmatrix} p\\ u\\ v\\ w\\ \varphi\end{bmatrix},\quad \vec{\mathbf{F}}_c = \begin{bmatrix} \vec{\mathbf{U}}\\ u\vec{\mathbf{U}}+p/\rho\delta_{ij}\\ v\vec{\mathbf{U}}+p/\rho\delta_{ij}\\ w\vec{\mathbf{U}}+p/\rho\delta_{ij}\\ \varphi\vec{\mathbf{U}}\end{bmatrix},\quad \vec{\mathbf{F}}_v = \begin{bmatrix} 0\\ \frac{1}{\rho\mathrm{Re}}\cdot\nabla\cdot u\\ \frac{1}{\rho\mathrm{Re}}\cdot\nabla\cdot v\\ \frac{1}{\rho\mathrm{Re}}\cdot\nabla\cdot w\\ 0\end{bmatrix},$$

$$\mathbf{K} = \begin{bmatrix} 0&0&0&0&0\\ 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&0&1&0\\ 0&0&0&0&1\end{bmatrix},\quad \mathbf{C} = \begin{bmatrix} \frac{1}{\beta}&0&0&0&0\\ 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&0&1&0\\ 0&0&0&0&1\end{bmatrix}, \tag{18}$$

$$\vec{\mathbf{S}} = \begin{bmatrix} 0\\ F_{gx}\\ F_{gy}\\ F_{gz}\\ 0\end{bmatrix}.$$

In all the equations above, $\mathbf{W}$ is the vector of dependent variables, p and ρ are pressure and density, respectively, β the constant parameter introduced by ACM. $\vec{\mathbf{F}}_c$ and $\vec{\mathbf{F}}_v$ are the convective flux and viscous flux vectors. $\mathbf{S}$ contains the volume force terms such as gravity acceleration. The first term on the left-hand side of Eq. (17) is a partial derivative with respect to pseudo-time τ (the artificial compression term), which is introduced to couple velocity and pressure fields for the calculation of pressure based on the divergence-free condition. $\mathbf{C}$ is a preconditioning matrix that arises with the implementation of the artificial compressibility method. $\mathbf{K}$ is the unit matrix with its first element being zero. The spatial Cartesian coordinates are x, y and z, and t is the physical time. F_g is the gravity force per unit mass and is given as

$$\vec{F}_g = \frac{\vec{n}_g}{Fr^2}, \tag{19}$$

Where Fr is the Froude number and $\vec{n}_g$ the unit vector along the prescribed direction of gravity.

The level set function φ is defined to be a signed distance function

$$|\varphi(\vec{x})| = d(\vec{x}) = \min_{x_I\in I}(|\vec{x}-\vec{x}_I|), \tag{20}$$

where I is the VOF interface, $\varphi > 0$ on one side of the interface and $\varphi < 0$ on the other. In standard level set methods, the advection of φ, including a reinitialization step to retain φ as a signed distance function, is not done in a conservative way, not even for divergence-free velocity fields. This implies that the total mass bounded by the zero level set is not conserved. This drawback has been addressed in currently proposed method.

For a traditional two-phase model, to represent density and viscosity discontinuities over the interface the smeared out version of Heaviside function can be used for the sake of numerical robustness. In our proposed method, however, the gaseous phase will be deactivated during the computation and the flow variables needed to evolve the interface will be extrapolated from with the fluid phase. The details will be covered in following sections.

Eqs. (11)–(15) can be recast in an integral form as follows:

$$\mathbf{C}\frac{\partial}{\partial\tau}\int\int\int_V \mathbf{W}dV + \mathbf{K}\frac{\partial}{\partial t}\int\int\int_V \mathbf{W}dV + \oint_{S_{cv}} (\vec{\mathbf{F}}_\mathbf{c} - \vec{\mathbf{F}}_\mathbf{v})\cdot dS + \int\int\int_V \vec{\mathbf{S}}dV = 0 \tag{21}$$

Once the artificial steady state is reached, terms involving derivatives with respect to τ become zero and the above equation reduces to

$$\mathbf{K}\frac{\partial}{\partial t}\int\int\int_V \mathbf{W}dV + \oint_{S_{cv}} (\vec{\mathbf{F}}_\mathbf{c} - \vec{\mathbf{F}}_\mathbf{v})\cdot d\mathbf{S} + \int\int\int_V \vec{\mathbf{S}}dV = 0 \tag{22}$$

Eq. (22) shows that the preconditioning matrix does not affect the solution and the original unsteady incompressible Navier–Stokes equations, Eqs. (11)–(15) are fully recovered.

3.2. Unstructured finite volume method

Following [19], Eq. (21) is discretized on an unstructured tetrahedral grid and a cell-vertex scheme is adopted here, i.e., all computed variables in vector **W** are stored at vertices of the tetrahedral cells. For every vertex, as shown in Fig. 4, a control volume is constructed using the median dual of the tetrahedral grid: nodes *A*, *P*, *B* and *C* form the vertex of the tetrahedral cell and *O* is the centre of the element *APBC*; *a*, *b* and *c* are the median duals of the edges *AP*, *BP* and *CP*, respectively; and 1, 2 and 3 are the centre points of triangles *APC*, *CBP* and *ABP*, respectively. In the cell-vertex scheme, the computed variables are stored at vertices *A*, *P*, *B* and *C*. Triangles *O*1a and *O*3a form the control volume surface for the convective term of the governing equation on edge *AP*. Likewise, the rest of the convective term for different edges is computed in an analogous manner. Triangles *APC*, *CBP*, *ABC* and *ABP* form the corresponding control volume surfaces for the calculation of viscous terms.

Finite volume numerical discretizations are based on the solution of the governing equations in integral form and spatial discretization is performed by using Eq. (21). To introduce the upwind scheme using an edge-based procedure, the convective term is transformed into the following equation:

$$\oint_{S_{cv}} \vec{\mathbf{F}}_\mathbf{c}\cdot\vec{\mathbf{n}}\, dS = \sum_{n=1}^{nbseg}\left[(\vec{\mathbf{F}}_\mathbf{c})_{ij}\cdot\vec{\mathbf{n}}\Delta S\right]_n, \tag{23}$$

where *nbseg* is the number of the edges associated with node *P*, $(\vec{\mathbf{F}}_\mathbf{c})_{ij}$ is the convective flux through the part of control volume surface. ΔS_n is part of the control volume surface associated with edge *n*. Therefore, all the convective fluxes are calculated for the edges and then collected at the two ends of each edge for updating of flow variables in time marching. The inviscid flux at the control volume surface associated with an edge *ij* and $(\mathbf{F}_\mathbf{c})_{ij} = (\vec{\mathbf{F}}_\mathbf{c})_{ij}\cdot\vec{\mathbf{n}}_{ij}$ is evaluated based on Roe's approximate Riemann solver [21]:

$$(\mathbf{F}_\mathbf{c})_{ij} = \frac{1}{2}\left[\mathbf{F}_\mathbf{c}^L + \mathbf{F}_\mathbf{c}^R - |\overline{\mathbf{A}}|(\mathbf{W}^R - \mathbf{W}^L)\right], \tag{24}$$

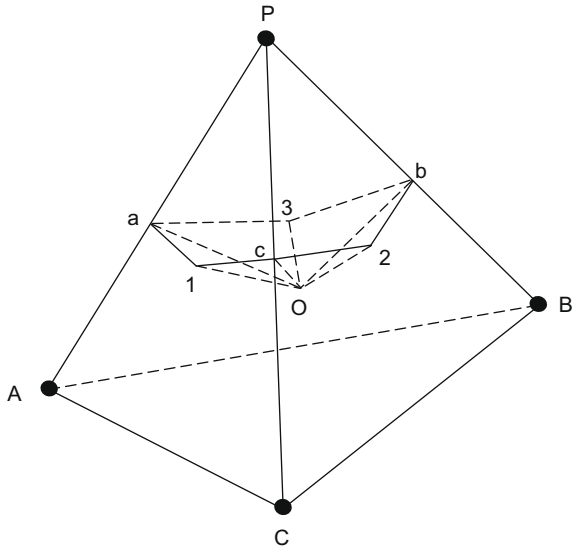


Fig. 4. Construction of control volume within a tetrahedron for a node *P*.

where $|\overline{\mathbf{A}}|$ is defined as Roe-averaged Jacobian matrix.

The viscous term is calculated based on the cell-based method as follows:

$$\oint_{S_{cv}} \vec{\mathbf{F}}_{\mathbf{v}} dS = \sum_{i=1}^{ncell} \left[\vec{\mathbf{F}}_{\mathbf{v}} \cdot \Delta S_c\right]_i, \tag{25}$$

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

$$\oint_s dS = 0. \tag{26}$$

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

$$\Delta S_{ci} = \frac{1}{3}\Delta S_{pi}. \tag{27}$$

Thus, the calculation of viscous terms can be simplified as,

$$\oint_{S_{cv}} \vec{\mathbf{F}}_{\mathbf{v}} dS = \sum_{i=1}^{ncell} \left[\vec{\mathbf{F}}_{\mathbf{v}} \cdot \Delta S_c\right]_i = \frac{1}{3}\sum_{i=1}^{ncell} \left(\vec{\mathbf{F}}_{\mathbf{v}} \cdot \Delta S_p\right)_i, \tag{28}$$

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

$$grad\ \phi_c = -\frac{\sum_{i=1}^{4}\phi_i 9 S_i}{27V} = -\frac{1}{3}\frac{\sum_{i=1}^{4}\phi_i S_i}{V}, \tag{29}$$

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

The edge-based method of [22] is adopted to calculate the total inviscid flux. Such a treatment leads to higher efficiency in computation and reduced data storage requirement. The left and right state vectors $\mathbf{W}_L$ and $\mathbf{W}_R$ at a control volume surface are evaluated using a nominally third-order upwind-biased interpolation scheme. If the left and right state vectors are set to $\mathbf{W}_i$ and $\mathbf{W}_j$ (i and j corresponding to the two end nodes of an edge), it is a first-order upwind scheme, which are shown as follows:

$$\mathbf{W}_L = \mathbf{W}_i + \frac{1}{4}\left[(1-k)\Delta_i^- + (1+k)\Delta_i^+\right], \tag{30a}$$

$$\mathbf{W}_R = \mathbf{W}_j - \frac{1}{4}\left[(1-k)\Delta_j^+ + (1+k)\Delta_j^-\right], \tag{30b}$$

where

$$\Delta_i^+ = \Delta_j^- = \mathbf{W}_j - \mathbf{W}_i,$$

$$\Delta_i^- = \mathbf{W}_i - \mathbf{W}_{i-1} = 2\vec{\boldsymbol{ij}}\cdot\nabla\mathbf{W}_i - (\mathbf{W}_j - \mathbf{W}_i) = 2\vec{\boldsymbol{ij}}\cdot\nabla\mathbf{W}_i - \Delta_i^+,$$

$$\Delta_j^+ = \mathbf{W}_{j+1} - \mathbf{W}_j = 2\vec{\boldsymbol{ij}}\cdot\nabla\mathbf{W}_j - (\mathbf{W}_j - \mathbf{W}_i) = 2\vec{\boldsymbol{ij}}\cdot\nabla\mathbf{W}_j - \Delta_j^-.$$

Therefore, substituting the above equations into Eqs. (32a) and (32b), and obtain the final equations based on upwind-biased interpolation scheme is,

$$\mathbf{W}_L = \mathbf{W}_i + \frac{1}{2}\left[(-k)\vec{\boldsymbol{ij}}\cdot\nabla\mathbf{W}_i + k\Delta_i^+\right], \tag{31a}$$

$$\mathbf{W}_R = \mathbf{W}_j - \frac{1}{2}\left[(1-k)\vec{\boldsymbol{ij}}\cdot\nabla\mathbf{W}_j + k\Delta_j^-\right], \tag{31b}$$

where k is set to 1/3, which corresponds to a nominally third-order accuracy.

Also following the work by Zhao et al. in [20], the values of the primitive variables φ, u, v, w and p at the edge center are determined using an upwind-biased characteristics method:

$$\varphi = \varphi^0, \tag{32a}$$

$$u = fn_x + u^0\left(n_y^2 + n_z^2\right) - v^0 n_x n_y - w^0 n_x n_z, \tag{32b}$$

$$v = fn_y + v^0\left(n_x^2 + n_z^2\right) - w^0 n_y n_z - u^0 n_y n_x, \tag{32c}$$

$$w = fn_z + w^0\left(n_x^2 + n_y^2\right) - u^0 n_z n_x - v^0 n_z n_y, \tag{32d}$$

$$p = p^1 - \lambda^1[(u-u^1)n_x + (v-v^1)n_y + (w-w^1)n_z], \tag{32e}$$

or

$$p = p^2 - \lambda^2\left[(u-u^2)n_x + (v-v^2)n_y + (w-w^2)n_z\right], \tag{32f}$$

where

$$C = \sqrt{(\lambda^0)^2 + \beta},$$

$$f = \frac{1}{2C}\left[(p^1 - p^2) + n_x(\lambda^1 u^1 - \lambda^2 u^2) + n_y(\lambda^1 v^1 - \lambda^2 v^2) + n_z(\lambda^1 w^1 - \lambda^2 w^2)\right],$$

where n_x, n_y and n_z are the three components of edge vector $\vec{n}$. The characteristic variables u^j, v^j, w^j, p^j with $j = 0,1,2$ in Eq. (32) are calculated by upwind differences from the left or the right side of the cell face according to the sign of λ^j by using Eq. (33). Flow quantities at $m+1$ pseudo time level obtained from the preceding equations on the characteristics are then used to calculate convection fluxes at the control volume interface. Those on different characteristics at m time level are approximately evaluated by an upwind scheme using the signs of the characteristics as suggested by [23].

$$\mathbf{W}^j = \frac{1}{2}\left[(1 + sign(\lambda^j))\mathbf{W}_L + (1 - sign(\lambda^j))\mathbf{W}_R\right],$$

where $\mathbf{W}_L$ and $\mathbf{W}_R$ are obtained by the upwind-biased interpolation. And the characteristics are given as

$$\lambda^0 = un_x + vn_y + wn_z, \tag{33a}$$

$$\lambda^1 = \lambda^0 + \sqrt{(\lambda^0)^2 + \beta}, \tag{33b}$$

$$\lambda^2 = \lambda^0 - \sqrt{(\lambda^0)^2 + \beta}. \tag{33c}$$

The computed edge center value of level set function [Eq. (32a)] is then used to calculate the density at the control volume interface, which is needed for the discretized momentum equations.

The advantages of the characteristics scheme are: (i) stable solution without adding artificial viscosity; (ii) less sensitive to grid orientation because flow signals are propagated along characteristics.

3.3. A fixed-point dual time stepping scheme

In this work, a fix-point partial implicit scheme is derived, which is found to be efficient in terms of memory requirement and computing effort per time step. This is due to the fact that no matrix manipulation is required by the scheme.

By re-writing Eq. (21) for a given node P, the spatially discretized equations form a system of coupled ordinary differential equations, which can be re-formulated as

$$\mathbf{C}\frac{\partial}{\partial\tau}(\Delta V_{cv}\mathbf{W}_p) + \mathbf{K}\frac{\partial}{\partial t}(\Delta V_{cv}\mathbf{W}_P) = -\left\{\sum_{n=1}^{nbseg}\left[(\vec{\mathbf{F}}_\mathbf{c})_{PQ}\cdot(\vec{\mathbf{n}}\Delta S)\right]_n - \frac{1}{3}\sum_{i=1}^{ncell}\left(\vec{\mathbf{F}}_\mathbf{v}\cdot\vec{\mathbf{n}}\Delta S_P\right)_i\right\} = -R(\mathbf{W}_P), \tag{34}$$

where $R(\mathbf{W}_P)$ represents the residual error, or deviation from steady state, which includes the convective and diffusive fluxes and ΔV_{cv} is the control volume of node P.

For convenience, the pseudo time derivative term is ignored at the moment. An implicit scheme is adopted to approximate Eq. (34) and the semi-discrete equation is shown as follows:

$$\mathbf{K}\frac{\partial}{\partial t}\left(\Delta V_{cv}^{n+1}\mathbf{W}_p\right) = -R\left(\mathbf{W}_P^{n+1}\right). \tag{35}$$

The superscript $(n+1)$ denotes the time level $(n+1)\Delta t$ and all the variables are evaluated at this time level. In this work, $\frac{\partial}{\partial t}$ is discretized as a second-order accurate backward difference, so that Eq. (35) is re-formulated as follows:

$$\mathbf{K}\left(\frac{1.5\Delta V_{cv}^{n+1}\mathbf{W}_p^{n+1} - 2.0\Delta V_{cv}^{n}\mathbf{W}_p^{n} + 0.5\Delta V_{cv}^{n-1}\mathbf{W}_p^{n-1}}{\Delta t}\right) + R\left(\mathbf{W}_P^{n+1}\right) = \widetilde{R}\left(\mathbf{W}_p^{n+1}\right) = 0, \tag{36}$$

where $\widetilde{R}\left(\mathbf{W}_p^{n+1}\right)$ is the new modified residual, which contains both the time derivative and flux vectors. The advantage of this implicit scheme is that the physical time step size is not restricted by numerical stability, but only by numerical accuracy.

This is especially useful in unsteady flow simulation where the maximum time step size is much smaller than the size permitted by accuracy consideration. From Eq. (36), it is worthwhile to note that there is no second-order accurate backward difference term in the conservation of mass and this is because Eq. (11) does not have a time derivative term. Following Jameson's work [17], the derivative with respect to a fictitious pseudo time τ, is added to Eq. (36) and the unsteady Navier–Stokes equations can be re-formulated as,

$$\mathbf{C}\Delta V_{cv}^{n+1}\frac{d\mathbf{W}_p}{d\tau} = -\widetilde{R}\left(\mathbf{W}_p^{n+1,m+1}\right), \tag{37}$$

whose solution is sought by marching to a pseudo steady state in τ. Here m denotes the pseudo time level $m\Delta\tau$. Once the artificial steady state is reached, the derivative of $\mathbf{W}_p$ with respect to τ becomes zero, and the solution satisfies $\widetilde{R}\left(\mathbf{W}_p^{n+1}\right) = 0$. This is actually the solution of Eq. (36). Hence, the original unsteady Navier–Stokes equations are fully recovered. Therefore, instead of solving each time step in the physical time domain (t), the problem is transformed into a sequence of steady state computations in the artificial time domain (τ). This can be performed using a pseudo time explicit five-stage Runge–Kutta scheme [24,25]. However, the pseudo time step size may be severely restricted if the physical time step size is very small. In view of this, an implicit dual time stepping scheme is adopted in this work. Eq. (37) can now be re-formulated as,

$$\mathbf{C}\Delta V_{cv}^{n+1}\frac{\mathbf{W}_p^{n+1,m+1}-\mathbf{W}_p^{n+1,m}}{d\tau} = -\widetilde{R}\left(\mathbf{W}_p^{n+1,m+1}\right). \tag{38}$$

To facilitate the implicit linearization, an approximate flux function must be introduced first. In [26] it is shown how the total flux (including both inviscid and viscous fluxes, gravity forces are not considered here) across a control volume surface associated with a certain edge PQ (Q is a node neighbouring to P) can be approximated as,

$$\mathbf{R}_{PQ} = \vec{\mathbf{F}}_{PQ} \approx \frac{1}{2}\left[(\vec{\mathbf{F}}_{\mathbf{c}})_P\cdot\vec{\boldsymbol{n}} + (\vec{\mathbf{F}}_{\mathbf{c}})_Q\cdot\vec{\boldsymbol{n}} - |\lambda_{PQ}|(\mathbf{W}_P-\mathbf{W}_Q)\right], \tag{39}$$

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

$$\lambda_{PQ} = \vec{\mathbf{U}}\cdot\vec{\boldsymbol{n}}_{PQ} + \sqrt{(\vec{\mathbf{U}}\cdot\vec{\boldsymbol{n}}_{PQ})^2+\beta^2}. \tag{40}$$

Note that $\mathbf{R}_{PQ}$ can be related to the residual $R(\mathbf{W}_P)$ defined in Eq. (34) as,

$$R(\mathbf{W}_P) \approx \sum_{Q=1}^{nbseg}\mathbf{R}_{PQ}. \tag{41}$$

We are now in a place to perform the Taylor series expansion for the modified residual defined in Eq. (36) with respect to the pseudo time for node P,****

$$\widetilde{R}\left(\mathbf{W}_p^{n+1,m+1}\right) = \widetilde{R}\left(\mathbf{W}_p^{n+1,m}\right) + \left.\frac{\partial\widetilde{R}(\mathbf{W}_P)}{\partial W_P}\right|_{n+1,m}\Delta\mathbf{W}_P + \sum_{Q=1}^{nbseg}\left(\frac{\partial\widetilde{R}(\mathbf{W}_P)}{\partial\mathbf{W}_Q}|_{n+1,m}\Delta\mathbf{W}_Q\right) \approx \widetilde{R}\left(\mathbf{W}_p^{n+1,m}\right) + \left.\frac{\partial\widetilde{R}(\mathbf{W}_P)}{\partial\mathbf{W}_P}\right|_{n+1,m}\Delta\mathbf{W}_P, \tag{42}$$

where $\Delta\mathbf{W}_P = \mathbf{W}_p^{n+1,m+1} - \mathbf{W}_p^{n+1,m}$. The approximation introduced in Eq. (42) is the key to our proposed method. We assume that the link between P and Q, even they are neighbours, are weak, so that the third term at the right hand side of the equation can be dropped. In such a way we solve the governing equations for each node in a fully decoupled manner. This feature renders the solver a great simplicity to implement and extremely low memory requirement [19].

Notice that $\widetilde{R}(\mathbf{W}_p)$ contains both the flux vectors and physical time derivative (recall Eq. (36)), and we need to treat them separately. With the help of Eq. (39) and keeping in mind that the approximate flux function $\mathbf{F}_{PQ}$ is edge-based, we may easily have

$$\frac{\partial\mathbf{R}_{PQ}}{\partial\mathbf{W}_P} = \frac{\partial\mathbf{F}_{PQ}}{\partial\mathbf{W}_P} = \frac{1}{2}\left[\frac{\partial(\vec{\mathbf{F}}_{\mathbf{c}})_{PQ}}{\partial\mathbf{W}_P} - |\lambda_{PQ}|\right] \tag{43}$$

where $\frac{\partial(\vec{\mathbf{F}}_{\mathbf{c}})_{PQ}}{\partial\mathbf{W}_P} = A_P$ is the system Jacobian. For the physical time derivative, we have

$$\begin{aligned}
&\mathbf{K}\left(\frac{1.5\Delta V_{cv}^{n+1}\mathbf{W}_p^{n+1,m+1} - 2.0\Delta V_{cv}^{n}\mathbf{W}_p^{n} + 0.5\Delta V_{cv}^{n-1}\mathbf{W}_p^{n-1}}{\Delta t}\right)\\
&= \mathbf{K}\left(\frac{1.5\Delta V_{cv}^{n+1}\left(\mathbf{W}_p^{n+1,m} + \left(\mathbf{W}_p^{n+1,m+1} - \mathbf{W}_p^{n+1,m}\right)\right) - 2.0\Delta V_{cv}^{n}\mathbf{W}_p^{n} + 0.5\Delta V_{cv}^{n-1}\mathbf{W}_p^{n-1}}{\Delta t}\right)\\
&= \mathbf{K}\left(\frac{1.5\Delta V_{cv}^{n+1}\left(\mathbf{W}_p^{n+1,m} + \Delta\mathbf{W}_p\right) - 2.0\Delta V_{cv}^{n}\mathbf{W}_p^{n} + 0.5\Delta V_{cv}^{n-1}\mathbf{W}_p^{n-1}}{\Delta t}\right)
\end{aligned} \tag{44}$$

After gathering the edge-based residual into respective vertices, the whole-field equivalent Eq. (38) can then be re-written as,

$$\mathbf{C}\Delta\mathbf{V}_{cv}^{n+1}\left(\frac{\Delta t+1.5\Delta\tau}{\Delta t}-\frac{A_P\Delta\tau}{\Delta V_{cv}^{n+1}}\right)\frac{\Delta\mathbf{W}_P}{\Delta\tau}=-R\left(\mathbf{W}_P^{n+1,m}\right)-\mathbf{K}\left(\frac{1.5\Delta V_{cv}^{n+1}\mathbf{W}_P^{n+1,m}-2.0\Delta V_{cv}^{n}\mathbf{W}_P^{n}+0.5\Delta V_{cv}^{n-1}\mathbf{W}_P^{n-1}}{\Delta t}\right),$$

or in short as

$$\mathbf{C}\Delta V_{cv}^{n+1}\widetilde{A}_P\frac{\Delta\mathbf{W}_P}{\Delta\tau}=\widetilde{R}\left(\mathbf{W}_P^{n+1,m}\right). \tag{45}$$

where $\widetilde{A}_P=\frac{\Delta t+1.5\Delta\tau}{\Delta t}-\frac{A_P\Delta\tau}{\Delta V_{cv}^{n+1}}$, and,

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

By inspecting Eq. (45), we can see that the links between a node point and its neighbours have been cut off and each node is solved in a fully decoupled manner. This feature renders the solver a great simplicity to implement and extremely low memory demand, especially in parallel computing environment. In order to solving Eq. (45), the 5×5 Jacobian matrix $\widetilde{A}_P$ has to be inverted, which is trivial and can be done analytically.

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

$$\mathbf{C}\Delta V_{cv}^{n+1}\{\widetilde{A}_P\}_D\frac{\Delta\mathbf{W}_P}{\Delta\tau}=\widetilde{R}\left(\mathbf{W}_P^{n+1,m}\right), \tag{46}$$

where $\{\widetilde{A}_P\}_D=diag\left\{\left(\frac{\Delta t+1.5\Delta\tau}{\Delta t}-\frac{A_P\Delta\tau}{\Delta V_{cv}^{n+1}}\right)\right\}$.

Pseudo time stepping is then performed on Eq. (46). An initial guess of $\mathbf{W}_P^{n+1,m}$ must be provided to start the iteration, one may simply take it as the old solution from last physical time step.

As described, the current point-fixed method is a different approach from those introduced in [50]. When dealing with a matrix-free method (e.g. matrix-free GMRES), in practice, one still needs to form a matrix (or set of matrices) for preconditioning purposes, so this family of methods is properly said to be "Jacobian-free" rather than "matrix-free". Furthermore, the equations for all of the node points in the whole computational domain must be solved in a coupled manner which has been proven to be time consuming and makes it difficult to be implemented in a parallel environment. Basing on the preliminary numerical experiments we have conducted, exact the same numerical results can be obtained from the serial version of the two methods. And good consistency is observed between the serial and parallel version of fixed-point method. However, the parallel GMRES implicit method produces slightly different results from its serial counterpart and sometimes stability problems. The inconsistency in result and deficiency in robustness may be explained by the fact that to some level the approximation has to be introduced into the GMRES method while it is parallelized, more specifically, the connections between far neighbours have to be cut off while performing partitioning. In our present method, no matrix manipulation is needed, the links between any node point and its neighbours have been cut off and the governing equations for each node are solved in a fully decoupled manner. This feature renders the solver a great simplicity to implement and extremely low memory demand. The proposed method is found to be efficient in terms of memory requirement and CPU resources per time step.

In this work, a five-stage Runge–Kutta time integration algorithm is used between each physical time step to iterate the numerical solution in an artificial time τ until convergence is reached [17,19]. Therefore, the converged solution from the artificial steady state equations becomes the time accurate solution at current physical time, t. The selection of artificial time step size $\Delta\tau$ must satisfy the following two constraints:

(1) *Viscous constraint:*

$$\Delta\tau\leqslant\Delta\tau_{vis}=\frac{\rho(\Delta h)^2}{2\mu}. \tag{47}$$

(2) *Convection constraint:*

$$\Delta\tau\leqslant\Delta\tau_{conv}=\frac{\Delta h}{u_{\max}}. \tag{48}$$

In Eqs. (47) and (48), Δh is the characteristic length of the grid. After $\Delta\tau_{vis}$ and $\Delta\tau_{conv}$ are computed, the final pseudo time step size is determined as

$$\Delta\tau=\min(\Delta\tau_{vis},\Delta\tau_{conv})\times L_{CFL}.$$

Benefited from the partial implicit treatments introduced into the solver, the CFL parameter L_{CFL} can be as large as 10. The stopping criterion for pseudo time stepping is

$$\text{Stop if residual drop satisfies}: \frac{R^{n,m}}{R^{n,0}} \leqslant \varepsilon, \tag{49}$$

where ε is a prescribed small number. $R^{n,m}$ is the residual for current sub-iteration and $R^{n,0}$ the initial residual at the beginning of current physical time step. In this work, the residual is computed from the continuity equation [Eq. (11)] as

$$R = \sum_{i=1}^{CVTOT} \left| \int\int\int_{cv} (\nabla \cdot \vec{\mathbf{U}}) d\nu \right|_i = \sum_{i=1}^{CVTOT} \left| \oint_{S_{cv}} (\vec{\mathbf{U}} \cdot \vec{\mathbf{n}}) dS \right|_i = \sum_{i=1}^{CVTOT} \left| \sum_{n=1}^{nbseg} (\vec{\mathbf{U}})_{ij} \cdot \vec{\mathbf{n}} \triangle S]_n \right|_i, \tag{50}$$

where *CVTOT* is the total number of grid points in the domain. Apart from Eq. (49), another necessary check before quit sub-iteration is $R^{n,m}$ less than a prescribed value (10^{-2} for example). For most of the free surface flow cases, 20 iterations are enough to make the residual drop to a given level (usually be 10^{-3}).

As mentioned by many other researchers, the choice of the pseudo-acoustic velocity β is of importance as it affects the stability and convergence characteristics of the scheme. In this work, a local treatment of β at each computational cell introduced in [57] is employed to ensure applicability to problems containing flows with both diffusion and convection dominated regions.

The general idea is that the Mach number be kept below unity. While calculating β, a distinction is made between convective-dominated (high Reynolds numbers) and diffusion-dominated (low Reynolds number) flow regions. This is as the pseudo-acoustic velocity β will be scaled in a different manner in each case. Considering first high Reynolds number flows, β should be kept as close as possible to the local convection velocity [58]. This is implemented as follows:

$$\beta_{conv} = \begin{cases} \varepsilon_{\tau_{conv}} & \text{if } |\vec{U}| \leqslant \varepsilon_{\tau_{conv}} \\ |\vec{U}| & \text{if } |\vec{U}| > \varepsilon_{\tau_{conv}} \end{cases}, \tag{51}$$

where $|\vec{U}|$ is the magnitude of local velocity vector and $\varepsilon_{\tau_{conv}}$ is an adjustable parameter determined as follows:

$$\varepsilon_{\tau_{conv}} = \begin{cases} \mathrm{Re}/10^4 & \text{if } \mathrm{Re} \leqslant 10^4 \\ 1.0|\vec{U}|_{\max} & \text{if } \mathrm{Re} > 10^4 \end{cases}, \tag{52}$$

with $|\vec{U}|_{\max}$ being the maximum velocity magnitude in the whole-field. For low Reynolds number diffusion-dominated flows, β is determined as follows:

$$\beta_{vis} = \left(\frac{CFL\Delta h}{\Delta \tau_{vis}} - \left|\vec{U}\right| \right)^2 - 4\left|\vec{U}\right|^2, \tag{53}$$

where $\Delta\tau_{vis}$ is computed from Eq. (47). The final value of β is calculated as $\beta = \max\{\beta_{conv}, \beta_{vis}\}$.

3.4. Convergence acceleration techniques

In the present code, time-dependent calculations require the convergence of the Navier–Stokes equations to the steady state in pseudo-time for each real time step. To speed-up the convergence rate, an implicit residual smoothing scheme developed for unstructured grids is employed. The smoothing equation for a vertex k can be expressed as follows:

$$\overline{R}_k = R_k + \varepsilon \nabla^2 \overline{R}_k \tag{54}$$

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

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

where CFL^* is the maximum *CFL* number of the basic scheme (typically no bigger than 2). The solution to the above equations can be obtained on an unstructured grid by using the Jacobi iterative method as follows,

$$\overline{R}_k^{(m)} = \frac{R_k^{(0)} + \varepsilon \sum_{i=1}^{numnod(k)} \overline{R}_i^{(m-1,m)}}{1 + \varepsilon \cdot numnod(k)} \tag{55}$$

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

Another technique employed to enhance the convergence rates is the multigrid method. The basic idea of this method is to carry out early iterations on a fine grid and then progressively transfer these flow field variables and residuals to a series of coarser grids. On the coarser grids, the low frequency errors become high frequency ones and they can be easily eliminated by a time stepping scheme. The flow equations are then solved on the coarser grids and the corrections are then interpolated back to the fine grid. The process is repeated over a sufficient number of times until satisfactory convergence on the fine grid is achieved. For ease of implementation, the non-nested mesh method is adopted here, which utilizes independently

generated non-nested (or overset) coarse meshes and has been proven to be accurate and efficient for solving non-linear flow equations on unstructured grids (see [59]). V-cycle strategy has been employed in the present work. The initial solution and residuals on the coarse grid ($h+1$) are transferred from the fine grid (h) using volume-weighted transfer operators [59]. In order to drive the coarser grid solution using the fine grid residual, a forcing function is calculated at the first stage of the implicit Runge–Kutta time stepping scheme and subsequently added to the residual on the coarse grid. After calculating the variables on the coarsest grid, the corrections are evaluated and interpolated back level-by-level to the finest grid. To improve efficiency for the simulation of viscous flows, the viscous terms are only evaluated on the fine grid but not evaluated on the coarser grids. Since the coarser grids are only used to cancel the dominating low frequency errors, this treatment does not affect the accuracy of the solution. The upwind-biased interpolation scheme is also set to first-order at the coarser levels. More specifically, the correction, dW_{h+1}, is the difference between the newly computed value on the coarse grid, W^{+}_{h+1}, and the initial value that was transferred from the fine grid, $W^{(0)}_{h+1}$.

$$dW_{h+1} = W^{+}_{h+1} - W^{(0)}_{h+1}.$$

The corrections are transferred to the fine mesh by the *prolongation* operator, I^{h}_{h+1}:

$$v_h = I^{h}_{h+1} dW_{h+1}.$$

And the solution on fine grid is updated by:

$$W^{+}_{h} = W_h + v_h.$$

According to Fig. 5, the correction of the flow field variables transferred from the coarse nodes 1, 2, 3 and 4 to the fine node a is a weighted average of the corrections at these nodes and the expression for the transferred correction is as follows:

$$(v_a)_h = \frac{V_1(dW_1)_{h+1} + V_2(dW_2)_{h+1} + V_3(dW_3)_{h+1} + V_4(dW_4)_{h+1}}{V_1 + V_2 + V_3 + V_4}$$

where V_1 is the volume of the sub tetrahedron cell with vertices 2, a, 3 and 4. V_2 is the volume of the sub tetrahedron cell with vertices 1, a, 3 and 4. V_3 is the volume of the sub tetrahedron cell with vertices 1, a, 2 and 4. V_4 is the volume of the sub tetrahedron cell with vertices 1, a, 2 and 3. The volume of the corresponding tetrahedron cell is the one opposite to the node.

Above two techniques are described in [19,59] in details and can be applied here without change. This is because similar temporal and spatial discretization techniques are employed by both models.

3.5. Boundary condition

3.5.1. N–S incompressible solver

At the solid wall, slip (for inviscid flows) or no-slip (for viscous flows) and no-injection boundary conditions are imposed, that is, the zero normal fluxes of mass, momentum, and energy are imposed. In addition, the solid surface is assumed to be adiabatic, and the pressure gradient normal to the wall at the surface is considered to be zero. For a far-field or upstream boundary, the flow velocity is given directly, and gradients of variables are assumed to be zero. The pressure at the upstream was calculated while downstream pressure is fixed at a constant value.

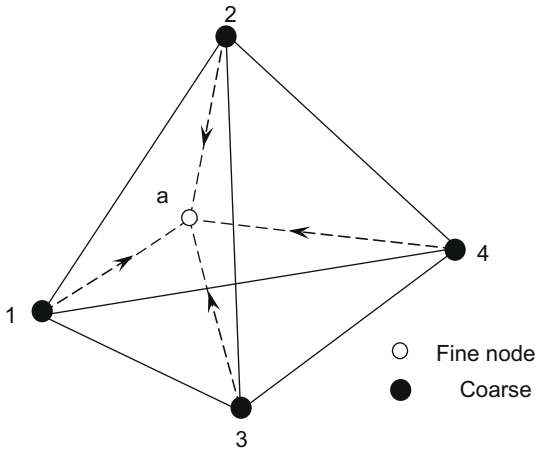


Fig. 5. Transfer of corrections from the coarse mesh to the fine mesh using *prolongation* transfer operator.

3.5.2. Level set equation

For level set equation, some special notes must be point out. At the solid wall, slip (normal velocity to the wall is zero) or no-slip (zero velocity) are both acceptable according to our experiences. They can give similar results for the free surface calculation. For both case, following condition must be imposed [49]:

$$\frac{\partial \phi}{\partial \vec{n}} = 0,$$

where $\vec{n}$ is the normal vector to the wall. This condition can be implemented in a cell-centered scheme in a very straightforward manner. For example, one can use this condition while extrapolating a face value before evaluating the fluxes for a wall boundary cells. It is more complicated in a vertex-centered scheme for there is no explicit way to impose such a condition. In this study, we impose this boundary condition in an implicit manner. After computing the nodal gradients for every node in the domain the normal component of the gradients are reduced to zero for every boundary nodes. We have found that this treatment has little impact on the overall stability of the code and gives satisfactory results at the boundary.

3.6. Governing equations and numerical method for volume fraction evolution (VOF)

The evolution of the volume fraction (VOF) field is governed by Eq. (2). For incompressible flow, this can be re-formulated as Eq. (16) which is repeated here for convenience:

$$\frac{\partial F}{\partial t} + \nabla\left(F \cdot \vec{U}\right) = 0. \tag{56}$$

This equation is in divergence form and can be easily discretized as

$$\Delta V_{cv}^{n+1} \frac{\Delta F}{\Delta \tau} = \left(\frac{\Delta t + 1.5\Delta\tau}{\Delta t} - \frac{A\Delta\tau}{\Delta V_{cv}}\right)^{-1} \left\{ -R^{n+1,m} - \left(\frac{1.5\Delta V_{cv}^{n+1} F^{n+1,m} - 2.0\Delta V_{cv}^{n} F^{n} + 0.5\Delta V_{cv}^{n-1} F^{n-1}}{\Delta t}\right)\right\},$$

where

$$R^{n+1,m} = \sum_{n=1}^{nbseg} \left[F_f \mathbf{F}\right]_n.$$

In above equation, the subscript '*f*' denotes the value at the centre of the edge *ij* and **F** is the volumetric flux defined by

$$\mathbf{F} = \vec{\mathbf{U}}_f \cdot \vec{\mathbf{n}}_{ij} \Delta S,$$

where ΔS is the control volume area associated with edge *ij*. The components of $\vec{\mathbf{U}}_f$ can be determined using Eqs. ((32b)–(32d)), and the computation of volumetric flux is performed simultaneously along with the level set and N–S equations. For the vertex-centered method, the cell-vertex values of volume fraction *F* are used to interpolate the edge centre values F_f on the control volume faces. This interpolation, which can guarantee a bounded solution while maintaining the sharpness of the interface, is presented next.

The upwind-biased characteristic method developed in Section 3.3 [Eq. (32)] is very suitable for determining the edge center values when applied to a smoothly transitioned function, like velocity and level set field, but it is too diffusive for the evolution of step function. The VOF equation is a typical example of this kind. The CICSAM (Compressive Interface Capturing Scheme for Arbitrary Meshes), [8], is employed due to its ability to maintain the sharpness of the interface while keeping reasonable accuracy. It makes use of the NVD concept [10] and switches between different high resolution differencing schemes to yield a bounded scalar field, but one which preserves both the smoothness of the interface and its sharp definition (over one or two computational cells). The special implicit implementation embedded inside makes it applicable to unstructured meshes, which is a desired feature by our solver. This method will be described in steps next.

Step 1. Determine the donor and acceptor control volume (referred by CV herein) according to the sign of the volumetric flux.
A schematic representation of a one-dimensional CV and its neighbours is given in Fig. 6. The centre CV (donor CV), referred to with a subscript *D*, has two neighbours known as the acceptor CV, referred to by subscript *A*, and the upwind CV, referred to by subscript *U*. The local flow direction is used to determine the location of the neighbours. The CV receiving fluid from the centre CV is the acceptor CV and the other CV is the upwind CV. The CV boundary face between the donor and acceptor CVs, referred to with a subscript *f*, is the face under consideration. Note that in unstructured grids, access to an upwind CV is not necessarily readily available and therefore some special method has to be introduced whenever the flow requires an upwind CV.

Step 2. Predict the bounded upwind value of volume fraction F_U^* according to:

$$F_U^* = \min\left\{\max\left\{[F_A - 2(\nabla F)_D \cdot d], 0\right\}, 1\right\},$$

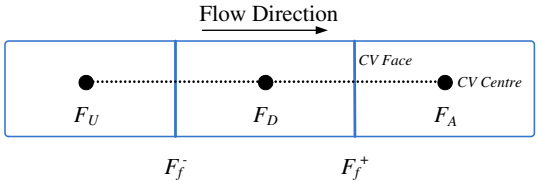


Fig. 6. Schematic of one-dimensional control volume and its neighbours.

where $(\nabla F)_D$ is the volume fraction gradient vector in the donor CV and d is the space vector pointing from donor CV to acceptor CV. As can be seen from Eq. (56), no explicit variables of an upwind CV are needed.

Step 3. Use the predicted upwind value F_U^* to calculate the normalized variable for the donor *CV* $\widetilde{F}_D$.

$$\widetilde{F}_D = \frac{F_D - F_U^*}{F_A - F_U^*}. \tag{57}$$

Step 4. Use the upper bound of the convection boundedness criterion (CBC, [29]) for multi-dimensional flow calculations, which is defined in Eq. (58), to calculate the normalized CV face value.

$$\widetilde{F}_{f_{CBC}} = \begin{cases} \min\left\{1, \frac{\widetilde{F}_D}{c_D}\right\} & \text{when } 0 \leqslant \widetilde{F}_D \leqslant 1 \\ \widetilde{F}_D & \text{when } \widetilde{F}_D < 0 \text{ or } \widetilde{F}_D > 1 \end{cases} \tag{58}$$

Step 5. The ULTIMATE-QUICKEST (UQ, see [1]) for multi-dimensional flow calculations, which is defined in Eq. (59), is used to calculate the normalized CV face value again.

$$\widetilde{F}_{f_{CBC}} = \begin{cases} \min\left\{\frac{8c_D\widetilde{F}_D + (1-c_D)(6\widetilde{F}_D+3)}{8}, \widetilde{F}_{f_{CBC}}\right\} & \text{when } 0 \leqslant \widetilde{F}_D \leqslant 1 \\ \widetilde{F}_D & \text{when } \widetilde{F}_D < 0 \text{ or } \widetilde{F}_D > 1 \end{cases} \tag{59}$$

In steps 4 and 5, c_D is the control volume Courant number, defined by

$$c_D = \sum_{n=1}^{nbseg} \max\left\{\frac{-\mathbf{F}\Delta t}{(\Delta V_{CV})_D}, 0\right\}, \tag{60}$$

where $\mathbf{F}$ is the volumetric flux defined in Eq. (55) and Δt is the physical time step size.

Step 6. Determine the weighting factor which takes into account the interface orientation and the direction of flow motion:

$$\gamma_f = \max\left\{k_\gamma \frac{\cos(2\theta_f) + 1}{2}, 1\right\}, \tag{61}$$

Where

$$\theta_f = \arccos\left(\left|\frac{(\nabla F)_D \cdot d}{|(\nabla F)_D||d|}\right|\right) \tag{62}$$

and $k_\gamma = 1$ is the recommended value.

Step 7. Calculate the normalized CV face value for the CICSAM differencing scheme from:

$$\widetilde{F}_f = \gamma_f \widetilde{F}_{f_{CBC}} + (1 - \gamma_f)\widetilde{F}_{f_{UQ}}. \tag{63}$$

Step 8. The calculated $\widetilde{F}_f$ is then used for the calculation of , the CICSAM weighting factor:

$$\beta_f = \frac{\widetilde{F}_f - \widetilde{F}_D}{1 - \widetilde{F}_D}, \tag{64}$$

Step 9. Finally, solve Eq. (53), the discretized volume fraction equation, to evolve the VOF field. And use Eq. (65) to explicitly calculate the CV face value.

$$F_f = (1 - \beta_f)F_D + \beta_f F_A. \tag{65}$$

As pointed out by the authors in [8], to maintain the best performance of CICSAM, the Crank–Nicolson time discretization is a necessary practice. According to our experiments, this scheme also gives excellent results with the proposed time integration scheme.

4. Single-phase free surface treatment

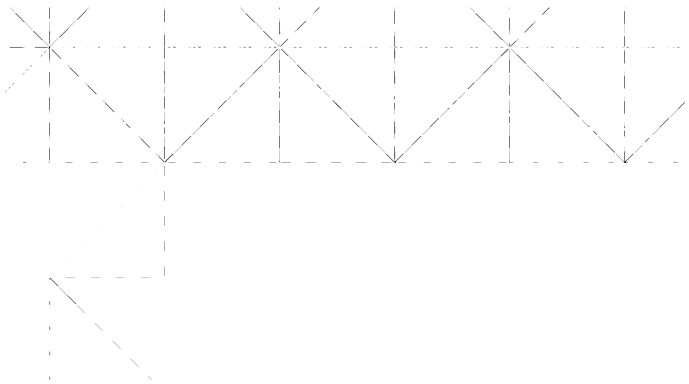
4.1. Interface reconstruction

After the level set and VOF field are evolved to a new physical time step, the interface must be reconstructed. First of all, the interfacial cells are identified. Interfacial cells are those cells cut by the VOF front, which is defined by $F = 0.5$; next, volume fractions and level set gradients for each of the interfacial cells are prepared for the interface reconstruction, and the detailed process is described in Section 2.1. The level set gradient for a given cell are readily accessible because we can easily compute it using Eq. (29). The calculation of volume fraction for a given cell is not so straightforward. Unlike cell-centered schemes, where the volume fraction is stored in the cell centre already, the vertex-centered solver stores all of its solution on the vertex. The simplest way to get the cell-centered volume fraction from the available cell vertices value is to perform an average operation:

$$F_C = \frac{1}{4}\sum_{i=1}^{4}(F_i). \tag{66}$$

Fig. 7 shows a typical interface reconstructed by this method. As can be seen in the figure, the interface is discontinuous and split to segments. This is due to the error introduced when computing the cell-centered volume fraction using the vertex averaged value. A possible solution is to perform a volume-weighted average before averaging over the vertices, as shown in Fig. 8, so

$$\overline{F} = \frac{1}{\Delta V_{all}}\sum_{k=1}^{nbneib}(F_k \Delta V_k), \tag{67}$$



where *nbneib* is the total number of direct ne bours of a given vertex. Eq. (67) is performed for each vertex on every inter- facial cell. Fig. 9 shows the improvement giv by this method.

4.2. Deactivation of air phase

It is well known that solving a two-phase v can cause severe problems when the density ratio between the liquid and gaseous phase is large. This difficulty mainly mes from the fact that in such a case, the pressure gradient distribution is discontinuous across the interface. This imp s that any small pressure gradient that 'pollutes over' from the liquid to the air region due to numerical error will acc ate the air considerably. This in turn will lead to loss of divergence, causing more spurious pressures. The whole cycle n , in fact, lead to a complete divergence of the solution. It should also be pointed out that, some semi-implicit schem have the merit handling very large density ratios without losing accuracy and stability. Among others, projection meth has received much more attentions recently, and it has been widely used by many researchers in practical application f free surface flow computations [43,47,48]. The advantage of this type of multiphase model lies in the fact that the int ction between gaseous phase and liquid phase can be accurately predicted without resorting to approximation or empiri formula. But this comes with the cost of more computational resources and CPU time in that all of the gaseous phase grid ints must be involved in the computation even they are far away the inter- face and of little interest. Also, for large scale t e dimensional computations, the parallelization of projection method is not easy to implement. In the present study, the mputation for the air phase is completely deactivated. All of the necessary information needed in the computation of lic l phase is extrapolated from inside the liquid phase itself. And the detailed procedure will be described next. The basic i employed in this study follows the method presented in [9], however, the extrapolation scheme differs.

4.2.1. Extrapolation of the pressure

The pressure in the gas region needs to be xtrapolated properly in order to obtain correct velocity in the region of the free surface. And this extrapolation is carried t in three steps. Firstly, the pressure for all gaseous points is set to constant, either the atmospheric pressure or, simply zer Next, the gradient of the pressure for those liquid points directly neighbour- ing to the *deactivation interface* are extrapolat from inside the liquid phase. This procedure is repeated for those gaseous points directly neighbouring to the *deactivatic nterface* because the pressure gradient for these points cannot be computed properly from the available data. Using this in mation (i.e. pressure and gradient of pressure), the pressure for the gaseous

ges in solution while attempting to "lump" the gravity term with pressure to define the piezometric pressure. Also we coul not find much difference in solution if we bypass the extrapolation and just apply a uniform pressure at the free surface. owever, we did observe improved robustness by switching to the proposed treatment.

4.2.2. Extrapolation of the velocity

The velocity and pressure for the gaseous p nts needs to be extrapolated properly in order to propagate th ace accurately. This extrapolation can be done in rious ways. One possible way is called "isobaric fix" techniqu duced by Ronald and Fedkiw et al. in their orig al finite difference based ghost fluid method [51]. In this tech ing partial differential equation has to be solv l in an iterative fa ion:

$$I_t \pm N \cdot \nabla I = 0,$$

terface. In our experiment, this method works fine in Cartesian blem n unstructured solver when singularity point presents. The other

way is the Lagrange multiplier approach proposed by Popinet and Zaleski in [52]. In this approach, it is assumed that the local velocity field around a node P in the vicinity of free surface can be expressed as

$$\vec{U} = \vec{U}_P + T \cdot \vec{X},$$

where $\vec{U}_P = \{u_P, v_P, w_P\}$ is constant to be determined. $\vec{X}$ is the position vector and $T_{ij} = \partial u_i/\partial x_j$ is a 3×3 matrix. With this assumption the viscous stress s around point P can be approximated as $\mu(T + T^T)$. If N grid points in the liquid domain are chosen in the neighbourhood of the point P and the unit normal $\vec{n}_P$ to the interface at P as well, then $\vec{U}_P$ and T can be computed by minimizing the error

$$\Phi = \sum_{n=1}^{N} \left\| (\vec{U}_P + T \cdot \vec{X}_n - \vec{U}_n) \right\|^2 + \lambda \cdot \left[\vec{n}_P \times (\mu(T + T^T) \cdot \vec{n}_P) \right].$$

By taking $\partial\Phi/\partial C = 0$, where $C \equiv \left\{ \begin{matrix} u_P, v_P, w_P, T_{11}, T_{12}, T_{13}, T_{21}, \\ T_{22}, T_{23}, T_{31}, T_{32}, T_{33}, \lambda_1, \lambda_2, \lambda_3 \end{matrix} \right\}$. Above equation is solved using a singular value decomposition method or a least-squares approach. In our experiment, this method can produce slightly better result than the one we are currently using. However, one may occasionally experience trouble in find the supporting stencil N for a gas node and the solvability condition of above equation is prone to be violated for splashing simulations where a small part of water is detached from the main body of the water domain. Further difficulties need to be addressed in parallel environment.

In the current study, an approach similar to the field-extension procedure used in [28] is employed for its relative simplicity and robustness. It is comprised of several basic sub steps as described as follows. In the first sub step, all of the computational nodes close to the interface are successively 'levelled'. The level tagging process is depicted in Fig. 10. All liquid nodes directly neighbouring the *deactivation interface* are tagged as '*level* 1' nodes (neighbouring here means connected by an edge). '*level 2*' nodes are grid points in the gaseous region that have one or more '*level* 1' as their neighbouring points. '*level* 3' nodes are grid points in the gaseous region that have one or more '*level* 2' nodes as their neighbouring points (except '*level* 1' points). The rest may be deduced by analogy.

In the second sub step of our approach, for every '*level* 1' point, the labels of the neighbouring points in the fluid phase are stored, as well as the lengths of the edges connecting the current point and the neighbouring points in the fluid phase. For every '*level* 2' point, the labels of the neighbouring '*level* 1' points are stored, as well as the lengths of the edges connecting the current point and the neighbouring '*level* 1' points. The same treatment will be performed for every point of successively levels. Finally, the velocity and pressure gradient are "extended" in the air phase through the levels. The inverse distance weighted method is used to perform the extrapolation, [30]. This method has the property of preserving local maxima and producing smooth reconstruction. The interpolation at a certain location (x,y,z) is

$$\begin{aligned} \phi(x,y,z) &= \sum_{m=1}^{n} w_m \phi_m / q \\ w_m &= \left(\frac{R - h_m}{R h_m} \right)^p \\ q &= \sum_{l=1}^{n} \left(\frac{R - h_l}{R h_l} \right)^p \end{aligned} \tag{68}$$

where ϕ_m represents the solution at a certain location, w_m the weight, and h_m the distance between the location (x,y,z) and the location of ϕ_m. R the maximum of all h_m; p is a constant and normally set to 2. The extrapolation process can be described as follows. The velocity and pressure gradient of the fluid nodes neighbouring to the *deactivation interface* are interpolated to the "*level* 1" points using Eq. (68). And then the velocity and pressure gradient of the "*level* 1" points are interpolated to the "*level* 2" points using Eq. (68) again and so on for successively levels. Generally, 3-level extension is enough, provided that the *CFL* number has not been set to an extraordinarily large value. As a result, not only the velocity and pressure at the newly emerging points, but also their derivatives will have physically realistic values, eliminating problems in the computation of the momentum equations in the next time step for these points.

Another problem which affects the robustness may arise when the gaseous phase is deactivated is the 'newly filled points' problem. It arises when a computational point (filled circle in Fig. 10), which was in the gaseous phase at one time step, emerges into the liquid at the next time step. The evaluation of the momentum equation at step $k + 1$, requires physical values of the velocity vector and pressure, as well as their derivatives from step k at these points. Due to the fact that the *deactivation interface* changes locations, it is possible that some of the required values from step k are not physical. To avoid forbiddingly complex special treatments of the computation of these derivatives every time such a change of flag is detected during the time advancement procedure, a field-extension procedure (for details of this method, see [28]), in which the velocity and pressure fields are "extended" in the gaseous phase at the end of each step. The gas phase deactivation and extrapolation scheme introduced in this section are extremely important to keep the method stable and accurate. A key element of this scheme is the definition of the '*deactivation interface*'. From the numerical point of view, the grid nodes falling into one side of this interface are treated as liquid phase and those falling into the other side of this interface are deactivated and treated as gaseous phase. The definition of this interface controls the accuracy and robustness of the free surface capturing and thus deserves further discussion. A simple way to define this interface is using $F = 0.5$, F is volume fraction (or

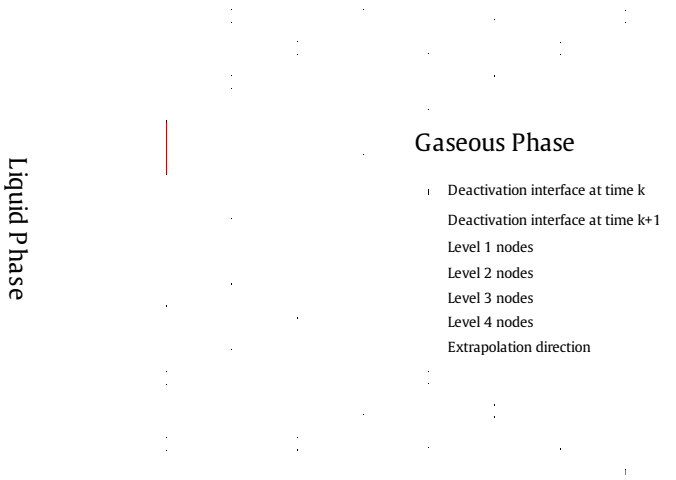


Fig. 10. A two-dimensional example that demonstrates the extrapolation of velocity and pressure for the grid nodes near a moving deactivation interface The classification of the various nodes is according to the notation introduced in the text and is based on the location of the deactivation interface at time k. In this special case, all of the 'level 1' nodes (ghost nodes) will emerge into the fluid at time k + 1, and all of the 'level 2' nodes will become 'level 1' nodes at time k + 1 and so on.

equally $\phi = 0$). The benefit of this method lies in the fact that the scheme is now most stable because the maximum density ratio has been limited to 2 in this manner. One possible problem in this approach is that it may affect the accuracy of the free surface capturing in that a half of the transitional interface region is now evolved by an extrapolated velocity field. In practice, one may tune the position of the *deactivation interface* within $F = 0.2$–0.5 to find a compromise between robustness and accuracy.

4.3. Coupling of level set function and volume fraction function

The level set function ϕ and the volume fractions F are coupled as follows:

- The normals used in the volume of fluid reconstruction step are determined from the level set function.
- The volume fractions are used, together with the slopes from the level set function, to construct the zero level set interface as well as a 'volume preserving' distance function along with providing 'closest point' information to the zero level set.
- The smooth-level set function is used to express the interfacial curvature.
- The "redistanced" level set function is then used to clip the VOF function within the narrow band near the zero level set.

5. Coupling with immersed membrane method (IMM)

The development of a new method (the Immersed Membrane Method or IMM) for imposing an immersed moving boundary condition and perform fluid–structure interaction simulation has been reported in [27,28]. Here we address the problem on how to couple it with currently proposed free surface scheme. In the early stage of every physical time step, the fluid solver is run to solve the fluid domain. And during this stage, IMM is used to impose the boundary conditions across the fluid–structure interface, where those fluid cells crossing the interface need special treatment. As shown in Fig. 11 for example, a fluid cell 1234 is cut by the interface. Nodes 1, 2 and 3 lie in the fluid domain while node 4 is in the structural domain. *abc*, *def* and *ghi* are triangles from the shell mesh of the structural domain. As described in [27,28], convection fluxes are computed based on mesh edges. In the computation of the convection flux along edge 14, the flow conditions at node 1 and ghost node 4 ($g14$) will be involved. The level set and volume fraction at ghost node 4 ($g14$) are determined as follows:

$$\begin{aligned} F_{g14} &= F_1 + \vec{r}_{14} \cdot (\nabla F)_1, \\ \phi_{g14} &= \phi_1 + \vec{r}_{14} \cdot (\nabla \phi)_1, \end{aligned} \tag{69}$$

Where $\vec{r}_{14}$ is the space vector pointing from point 1 to point 4, $(\nabla F)_1$ and $(\nabla \phi)_1$ the gradients for respective functions. Recall that for level set and VOF equation, one must impose zero normal gradient as the boundary condition at the solid wall. And this is the perfect place to impose such condition. Decompose $(\nabla F)_1$ and $(\nabla \phi)_1$ according to the direction of $\vec{r}_{14}$, and set their components in this direction to 0. The resultant gradients are then used in Eq. (69) to determine the ghost value for node 4.

A schematic diagram of our full solution algorithm is shown in Fig. 12 for completeness.

6. Results and discussion

In this section, we examine the integrity and accuracy of our interface reconstruction and volume tracking method, and present numerical results for three selected test problems:

6.1. Single vortex flow

An initially circular fluid body of radius 0.2 centered at the point (0.5,0.75) is evolved by a single vortex flow field. The flow field is given by the stream function

$$\Psi = -\frac{1}{\pi}\sin^2(\pi x)\sin^2(\pi y)\cos(\pi t/9), \tag{70}$$

where the velocity vector is defined by $\left(-\frac{\partial \Psi}{\partial y}, \frac{\partial \Psi}{\partial x}, 0\right)$. This flow field was first introduced in [31], and first used as the flow field in the "single vortex" problem by Rider and Kothe in [6] for assessing the integrity and capability of an interface tracking method in cases where significant interface stretching occurs. The flow field contains a single vortex centered in the domain. The vortex spins fluid elements, stretching them into a thin filament that spirals toward the vortex centre. Fig. 13 shows the simulation results for evolution of the circular fluid body in the single vortex flow field. The dimensions of the computational domain are $1.0 \times 1.0 \times 0.0125$. And the solution is computed using a 3D tetrahedral mesh with $80^2 \times 4$ grid points. It is seen that the fluid body is spun into a thinning filament, which compares very well with those results obtained using other state-of-the-art methods, for example [15,32]. Theoretically, the filament spirals toward the vortex centre, and becomes thinner and thinner continuously. However, Fig. 13 shows that the thin filament breaks into small pieces beginning at its tail after time reaches $t = 4.5$ s. This numerical phenomenon has been reported by many other researchers already. This is due to the fact that the grid resolution has become low relative to the thinning filament when the thickness of the filament is equal to or less than the grid size. Because the interface is represented by a single line segment in each interfacial grid cell, any two interfaces merge automatically whenever they come into the same cell, resulting in an unphysical 'pinch-off'. This unphysical pinch-off can be actually delayed by refining the grid near the interface. However, because the filament asymptotes to infinitesimal thickness, any grid will eventually provide inadequate resolution.

To determine the overall accuracy of the proposed methods, we carry out a grid convergence study for the single vortex problem. Four uniformly spaced and successively finer mesh sizes with $40^2 \times 4$, $80^2 \times 4$, $160^2 \times 4$, and $320^2 \times 4$ grid points, respectively, are used for error analysis, and the analytical solution on finest-mesh at $t = 9$ s is considered to be the 'exact' solution. On all the grids the same physical time step ($\Delta t = 0.01$ s) is employed in order to concentrate on the spatial resolution of the method. For all the grids, the simulation time is set to 9 s, at the end of which the L_∞ and L_q norms of the level set function errors are calculated as follows:

$$\varepsilon_N^\infty = \max_{i=1,M} |\phi_i^N - \phi_i^e|, \quad \varepsilon_N^q = \left[\frac{1}{M}\sum_{i=1}^{M} |\phi_i^N - \phi_i^e|^q\right]^{1/q} \tag{71}$$

where superscript N represents the grid refinement level (40, 80, 160 and 320) and M the number of grid points satisfying the condition $|\phi_i| \leqslant 5\Delta h$ and Δh was defined in Eq. (47). ε_N^∞ and ε_N^q are the infinity and qth error norms, ϕ_i^N is the level set function at node i, and ϕ_i^e is the 'exact' level set field. The results of the grid convergence study are summarized in Table 1 as well as Fig. 14. The graph shows the variation of the L_∞, L_1 and L_2 norms of errors with grid spacing in logarithmic coordinates. The lines with slope one and two are also given as reference. To further demonstrate the accuracy of our method, we also use the Richardson estimation procedure to study the accuracy of the solver as in [33]. Let f^N denote the numerical solution on the $N^2 \times 4$ meshes. Assume that the discrete solution is a γ-order approximation to its value f^{exact}, and the flow field is continuous and has no singularity points, then we have

$$\gamma = \frac{\log\left(\|f^N - f^{N/2}\| / \|f^{N/2} - f^{N/4}\|\right)}{\log 2} \tag{72}$$

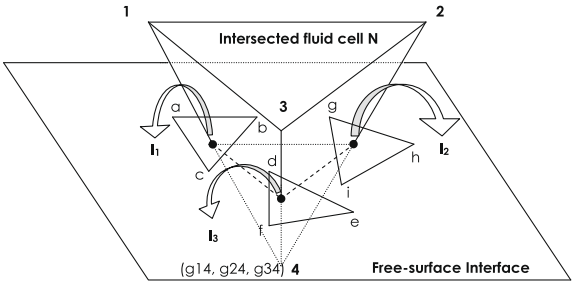
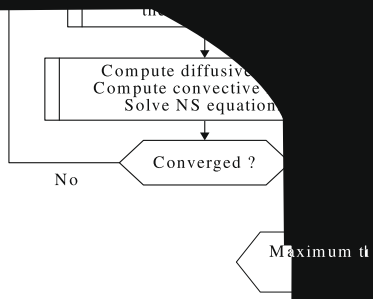


Fig. 11. Schematic of coupling between proposed free surface scheme and immersed membrane method.



where $\|\,\|$ denotes an error norm (L_∞, L_1 or L_2). If $\gamma \approx 2$ the solutio
for $N = 320$ (using solutions obtained on meshes $80^2 \times 4$, and 16
we use all three norms to compute the error and the results are

Upon inspecting Eq. (70), on may find that the flow field rev
position at time T_f. Fig. 13(g) shows the volume fraction contours
tion of the method, we used the time averaged mass error meth

$$E^M = \int_{t=0}^{T_f} \frac{(M(t) - M(0))}{T_f} d_t,$$

where

$$M(t) = \int_\Omega f(\vec{x}, t) d_{\vec{x}}.$$

And T_f is the total integration time (9 s). The computed mass

The quantitative mass error measurements listed above indi
capturing, even when large complex topological change has occ

6.2. Breaking dam problem

A classical experiment [35] used in the validation of the mathe
is the collapse of a liquid column [2,36–39]. Measurements of the

Fig. 126

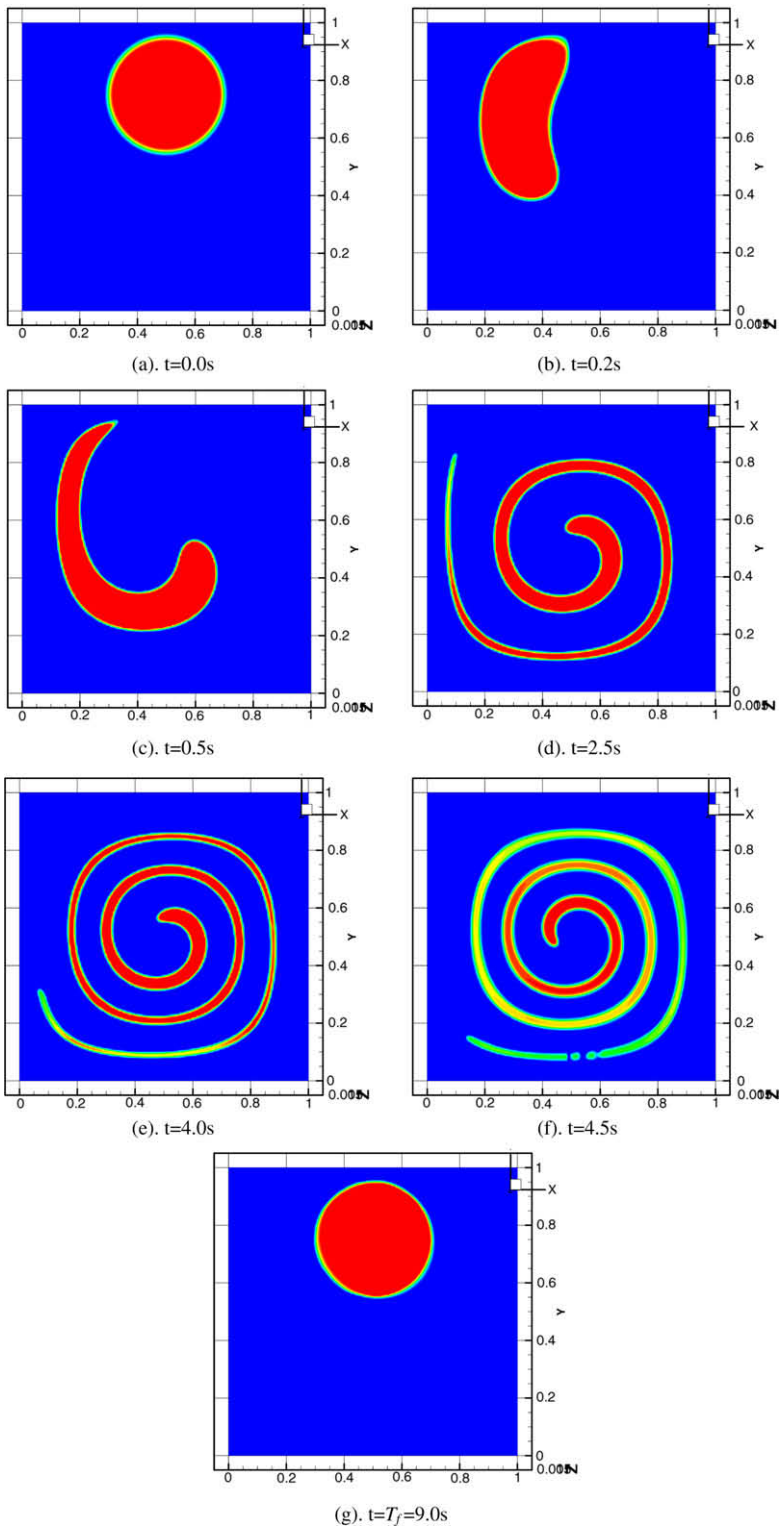


Fig. 13. Time evolution of an initially circular fluid body placed in the single vortex flow. The flow direction is initially anticlockwise and then changes to clockwise. Contoured by volume fraction F between $(0,1)$. 15 levels were used.

data such as the speed of the wave front and the reduction of the column height are available. Furthermore, an analytical solution of the interface evolution can be obtained. However, these are for inviscid flows and only of use for the initial stages. Fig. 15 shows a schematic view of the domain setup which is used for the current flow prediction. The water has dynamic

Table 1
The computed error norms L_∞, L_1 and L_2 from Eq. (71) on four grids for single vortex flow.

Grids	L_∞	L_1	L_2
$40 \times 40 \times 4$	3.2179e−2	3.1414e−2	2.6738e−2
$80 \times 80 \times 4$	1.2221e−2	8.2627e−3	9.3999e−3
$160 \times 160 \times 4$	4.3363e−3	2.0755e−3	2.9317e−3
$320 \times 320 \times 4$	1.6486e−3	5.2134e−4	9.0599e−4

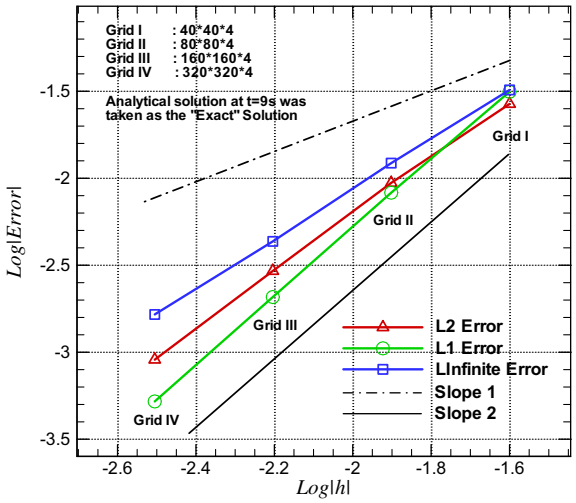


Fig. 14. Convergence of the L_∞, L_1 and L_2 error norms for the single vortex flow testing case. Slope 1 and Slope 2 are the reference lines for first-order and second-order accuracy, respectively. All of the error norms are computed within the transitional region near the zero level set.

Table 2
Rate of convergence γ calculated for different error norms.

Norm	Grids $80^2 \times 4$, $160^2 \times 4$, $320^2 \times 4$
L_∞	1.55
L_1	1.99
L_2	1.68

Table 3
Mass loss calculated by Eqs. (73) and (74) for four different meshes.

Grids	Mass loss
$40^2 \times 4$	9.68E−3
$80^2 \times 4$	3.53E−3
$160^2 \times 4$	4.48E−4
$320^2 \times 4$	2.17E−4

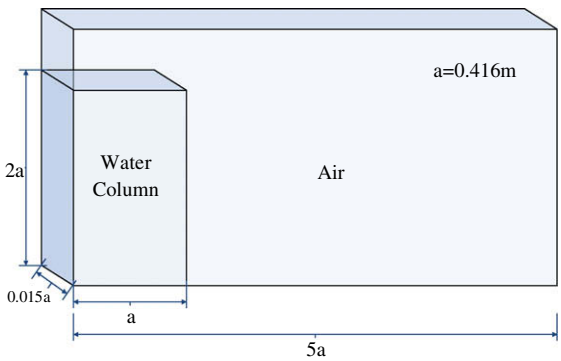


Fig. 15. Schematic of dam-breaking testing case. Slip conditions have been applied to the front and back wall of the tank, non-slip conditions have been applied to the rest of walls (left, right, up and down).

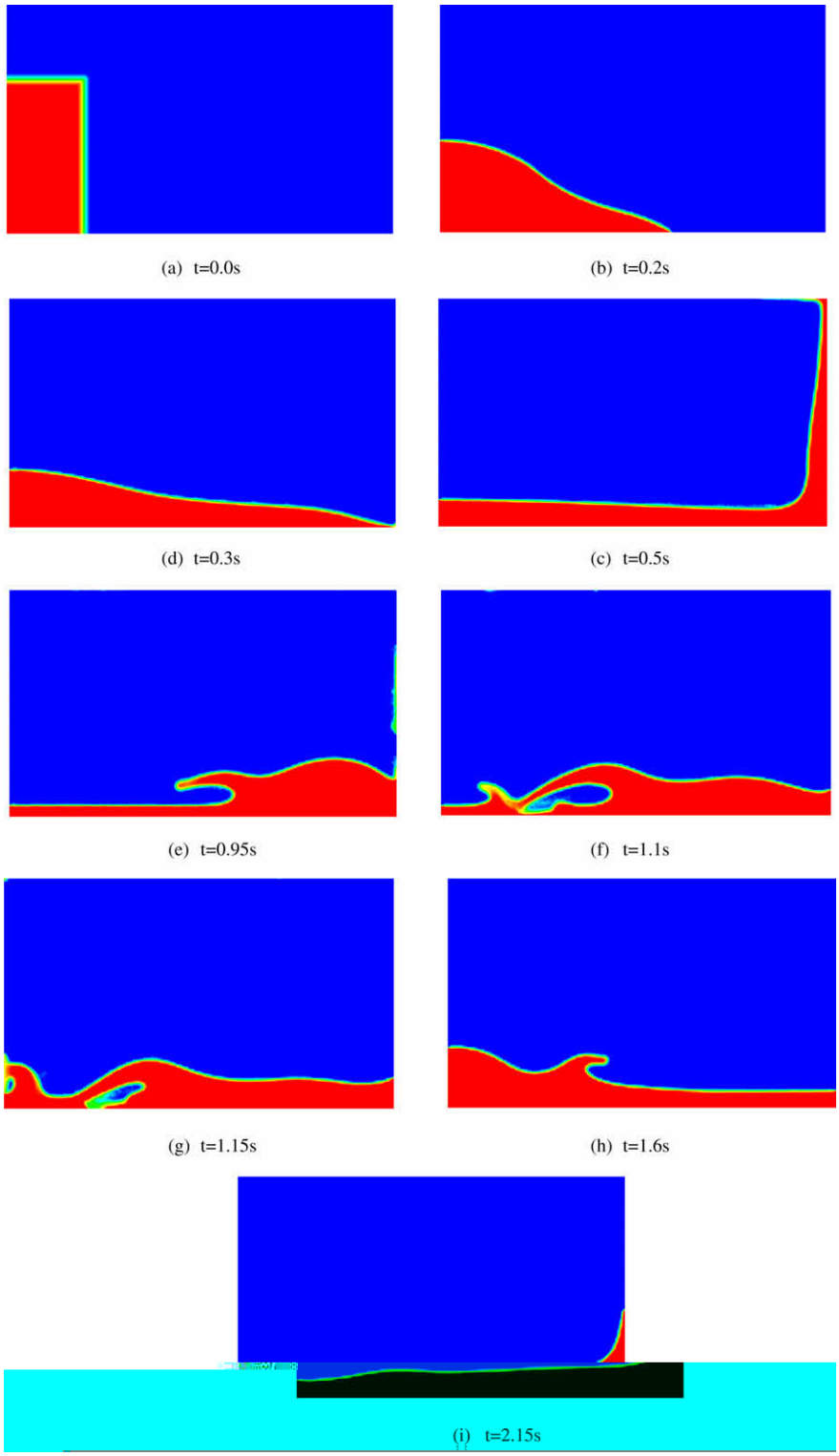


Fig. 16. Simulation result for the breaking of a water column. Nine different time snapshots are presented. Volume fraction contours are of 15 levels between 0 and 1.

viscosity $\mu = 1 \times 10^{-3}$ kg/m/s and density $\rho = 1$ kg/m^3. The acceleration due to gravity is taken to be $g = 9.8$ m/s. The water column is initially supported on the right by a vertical plate drawn up rapidly at time $t = 0.0$ s. Gravitational forces cause

the water column to collapse. The water eventually comes to rest, occupying the bottom of the tank. The initial stages of the flow are dominated by inertia forces with viscous effects increasing rapidly as the water comes to rest. On such a large scale, the effect of surface tension forces is unimportant. For the numerical calculation no-slip boundary conditions have been applied to the bottom and sides of the tank. This experiment models a two-dimensional effect; therefore slip conditions have been applied to the front and back face of the tank.

The predicted position of the interface on a tetrahedral grid generated from a $140 \times 80 \times 4$ uniform spaced structured mesh, are shown in Fig. 16. The air phase was deactivated while performing the simulation. On a PC with 4GB memory and an Intel Dual-Core CPU running at 2.4 GHz, the total computational time (corresponding to 2.4 s in the simulation) is about 1080 min (18 h).

Key features from the simulation are:

- Time $t = 0.2$ s: approximate 60% of the bank base is covered with water.
- Time $t = 0.3$ s: the surge front starts to hit the right wall of the bank.
- Time $t = 0.5$ s: the horizontal interface is almost parallel to the base of the tank and the water against the right wall starts to fall back under the influence of gravity.
- Time $t = 0.95$ s: the backward moving wave has folded. In the experiment however, a big air entrapment region presented in form of small bubbles should emerge behind the wave. The current methodology has been derived for sharp interfaces and tries to keep similar fluid regions together. To predict the behaviour of the small bubbles correctly, either a breakup model needs to be introduced or the mesh needs significant refinement to a resolution smaller than the bubble size. This failure to resolve bubbles or spray without the use of a prohibitively small grid results in the spray on the air side being incorporated in the tongue at the front of the backward moving wave. This results in the tongue being slightly heavier and therefore slightly lower than the position given in the experiments.

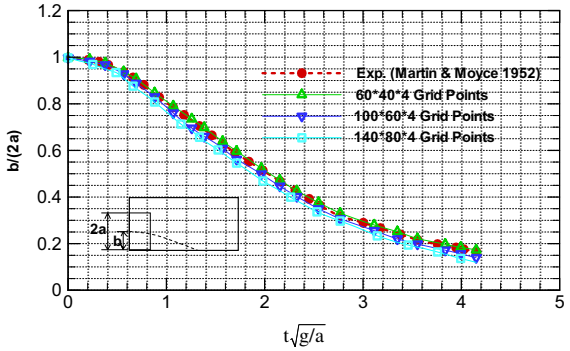


Fig. 17. The height of the collapsing water column versus time.

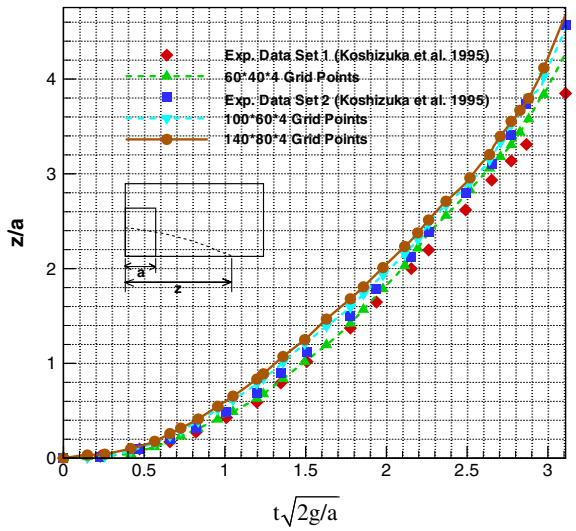


Fig. 18. The position of leading edge versus time.

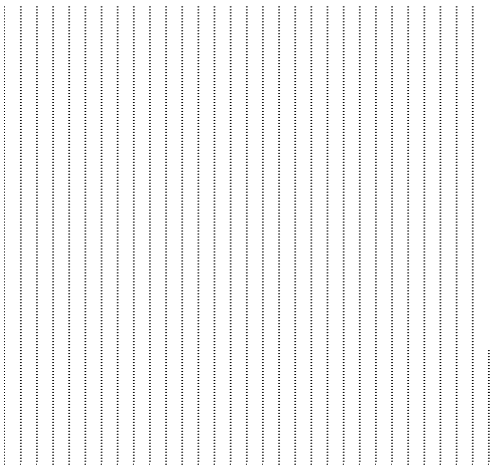
- Time $t = 1.1$ s: the tongue of the backward moving wave plunges into the water, trapping a large air bubble. In the experiment this bubble also contains a dispersed phase not predicted with the current methodology.
- Time $t = 1.15$ s: the tongue of the backward moving wave impinges upon the left wall, trapping an air bubble as well. The spray on the left above the surface shown in the experimental results is concentrated against the left wall in the case of the numerical prediction.
- Time $t = 1.6$ s: after the surge front impinging upon the left wall, a forward moving wave has produced and folded.
- Time $t = 2.15$ s: the forward moving wave hits the right wall of the tank.

A more quantitative comparison for the early stages of this experiment can be made by using the reduction in height and the speed of the wave front. The experimental data used were obtained by Martin and Moyce in [35]. The non-dimensional height of the collapsing water column at the left wall versus the non-dimensional time is shown in Fig. 17. Numerical predictions for three successively finer mesh sizes with $60 \times 40 \times 4$, $100 \times 60 \times 4$, $140 \times 80 \times 4$ grid points, respectively, are presented. For both these calculations the predicted height is the same and it corresponds very well with the experimental data presented by Martin and Moyce. The non-dimensional positions of the leading edge for the same cases are shown in Fig. 18. The calculated results show that the leading edge moves faster when the resolution of the mesh increases. Results presented by other researchers show the same tendency (see [40]). The reason for this is the difficulty to determine the exact position of the leading edge. A thin layer shoots over the bottom and the rest of the bulk flow follows shortly behind it. The difficulty is also confirmed by Martin and Moyce in [35] who present two different sets of experimental data.

The importance of the *deactivation interface* can be understood via Fig. 19. We used the $140 \times 80 \times 4$ grid to perform the breaking dam problem twice, but with two different selections of *deactivation interface*. As can be seen from the comparison, a proper selection of the *deactivation interface* can have big impact on the accuracy of the scheme.

To determine the accuracy of the proposed methods on general multiphase flows, a grid convergence study is also performed for the dam-breaking problem. Four uniformly spaced and successively finer mesh sizes with $60 \times 40 \times 4$, $100 \times 60 \times 4$, $140 \times 80 \times 4$ and $320 \times 160 \times 4$ grid points, respectively, are used for relative error analysis. Because there is no "exact" solution for this problem, the result on the finest grid at $t = 0.3$ s is taken as the reference solution. The results of the grid convergence study are summarized in Table 4 and Fig. 20.

Besides the absolute errors defined in Eq. (71), the relative error serves as another instructive indication of the solver performance. We define it as



$$(\varepsilon_N)_R = \left[\frac{1}{M}\sum_{i=1}^{M}\frac{\left|F_i^N - F_i^{2N}\right|}{F_i^{2N}}\right] \times 100 \tag{75}$$

The involved solution variable has been change to the volume of fluid function. The computed relative error norms for the four grids are tabulated in Table 5.

6.3. Dam-breaking wave interacting with a circular cylinder

The previous examples validated the proficiency and accuracy of the numerical scheme for studying single vortex flow and breaking dam problems. This example considers a three-dimensional dam-breaking wave interacting with a circular cylinder. The tank is 16 m long, 5 m wide, and 7 m high. The water column initially contained behind a vertical plate is

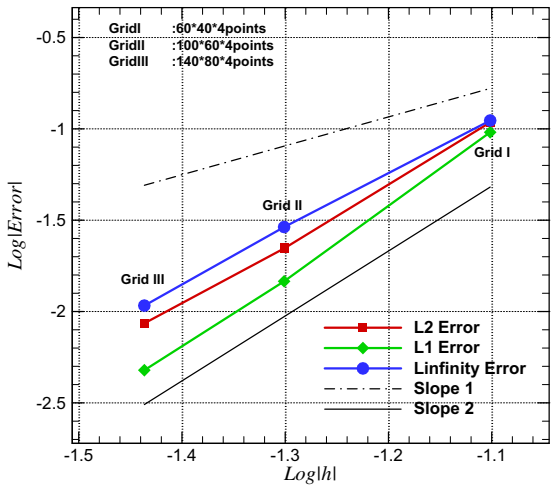


Fig. 20. Convergence of the L_∞, L_1 and L_2 error norms for the dam-breaking testing case. Slope 1 and Slope 2 are the reference lines for first-order and second-order accuracy, respectively. All of the error norms are computed within the transitional region near the zero level set.

Table 5
Relative error norms [Eq. (75)] for dam-breaking problem on four progressively refined grids.

Grids	Relative error L_R (%)
$40 \times 40 \times 4$	35.319
$100 \times 60 \times 4$	9.328
$140 \times 80 \times 4$	2.486

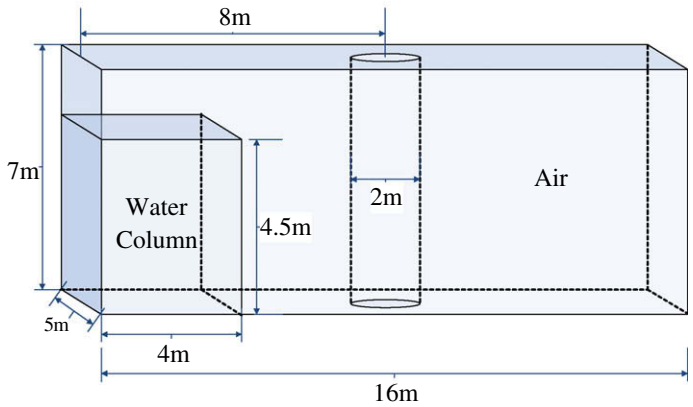


Fig. 21. Schematic of dam-breaking wave interacting with circular. Slip conditions have been applied to the front and back wall of the tank, non-slip conditions have been applied to the rest of walls (left, right, up and down).

4 m × 5 m × 4.5 m. The circular cylinder, which has a radius r = 1 m and height h = 7 m, is placed in the middle of the tank. The problem definition is demonstrated in Fig. 21. The entire tank is selected as the computational domain and meshed with

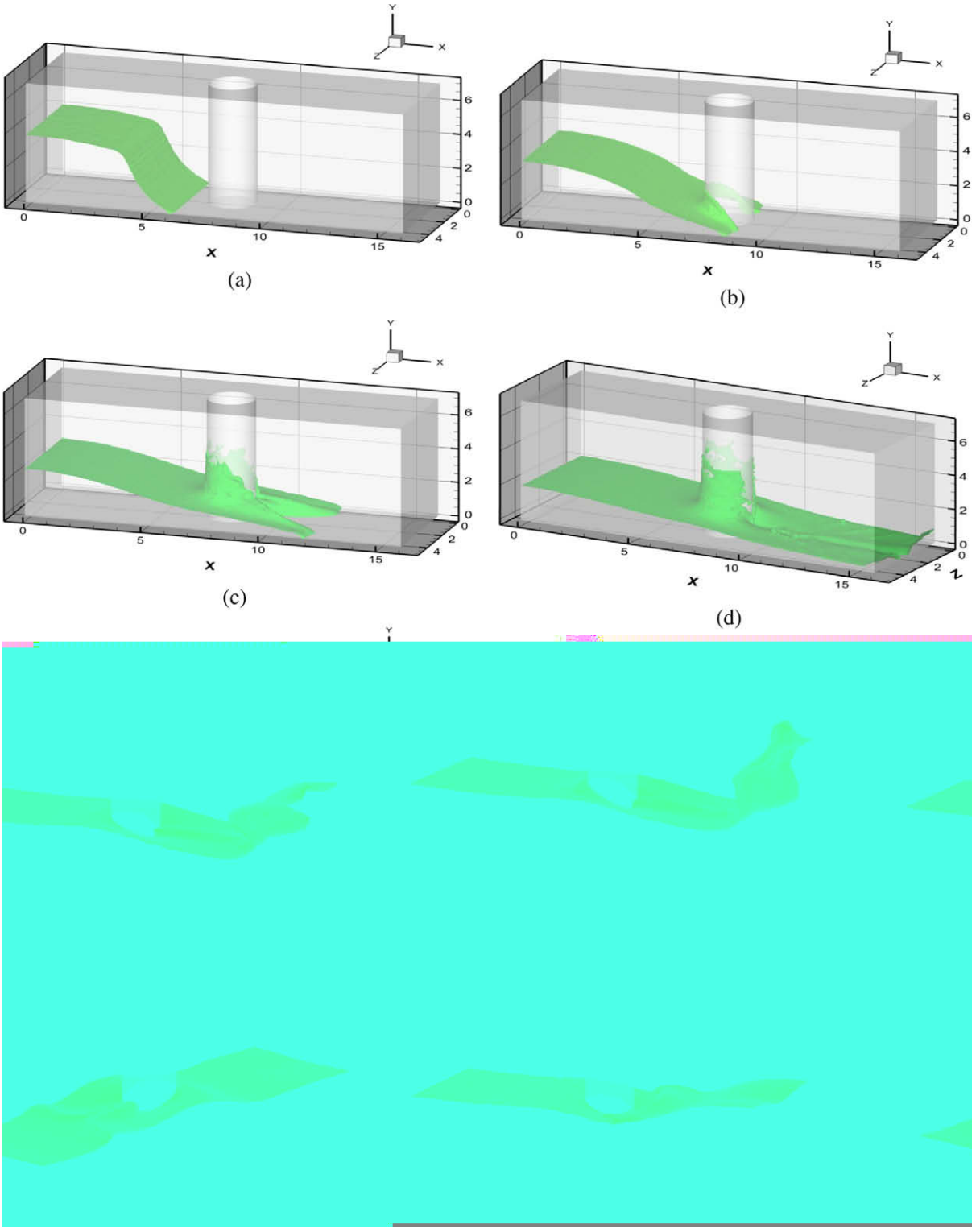


Fig. 22. Simulation result for dam-breaking wave interacting with a circular cylinder. Eight different time snapshots are presented. Free surface is represented by the ISO surface where F = 0.5.

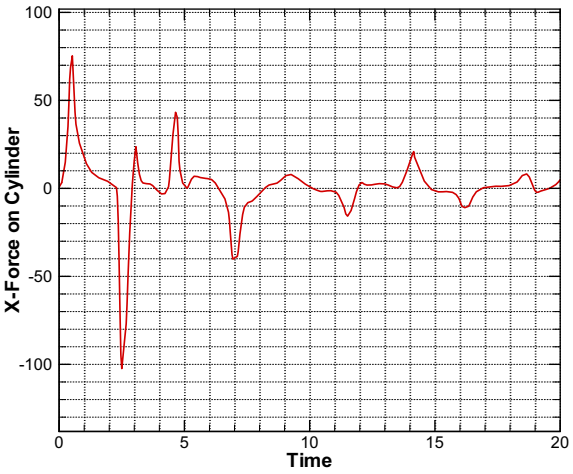


Fig. 23. Time history of the horizontal force acting on the cylinder.

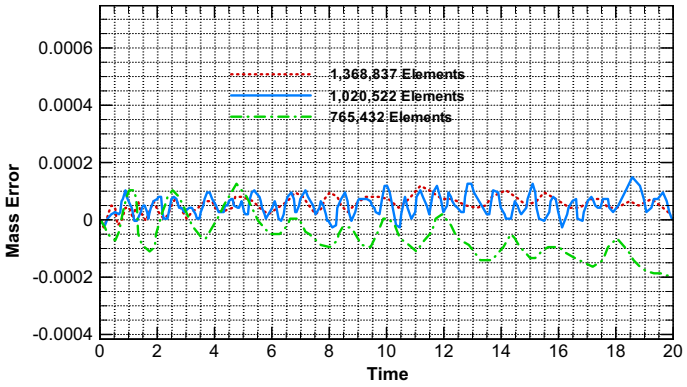


Fig. 24. Time history of the volume difference for different resolution levels.

249,663 grid points and 1,368,837 tetrahedral elements. Fig. 22 shows a sequence of snapshots of the free surface wave elevation, and Fig. 23 the time history of the horizontal force acting on the cylinder. The air phase was deactivated during this simulation. On a PC with 4 GB memory and an Intel Dual-Core CPU running at 2.4 GHz, the total computational time (corresponding to 10 s in the simulation) is about 12,000 min (200 h). The simulation time will be dramatically reduced if using a cluster to perform parallel computation. Running the same case on a cluster (with one hundred Intel QuadCore Xeon E5310 (1.6 GHz)) using 64 CPUs requires only 3.5 h, corresponding to a speed-up factor of 55.

A mass conservation test has also been performed for this testing case. The magnitude of water volume is calculated for every time step and compared to the initial value, which is 4 m $\times$ 4.5 m $\times$ 5m = 90 m^3. The detailed volume comparison time histories for three different grids are shown in Fig. 24.

7. Conclusion and future work

We have developed a novel hybrid coupled level set and volume of fluid method for sharp interface capturing on tetrahedral unstructured grids. The method takes advantage of the strengths of both the level set and volume of fluid methods, which makes the calculation of the interface normal vector easier and more accurate. Mass is also conserved very accurately. The analytic interface reconstruction algorithm we employed has been proven to be efficient, and conserve mass exactly. Numerical validation showed that the accuracy of our proposed method is reasonablely good and is comparable to other state-of-the-art methods. The method is also coupled to a finite volume based Navier–Stokes flow solver. Physical simulations of dam-breaking problem and dam-breaking wave interacting with cylinder showed that our method is capable of analyzing problems of physical interest. In addition, it can resolve complex interface changes and interfaces of high curvature accurately and efficiently.

Due to the fact that the centroid and intercept data is available as a by-product of the proposed interface reconstruction scheme, the development of a damping function for near-interface sub-grid models in large eddy simulation is underway. We will further couple this free surface tracking method with our fluid–structure interaction solver to investigate the wave impact on sea defence structures.

Acknowledgments

We would like to thank Dr. Zhengyi Wang for helpful discussions of free surface flow simulation. We would also like to thank an anonymous referee for his/her constructive criticism that helps to improve this manuscript greatly. This research was supported by the Flood Risk from Extreme Events (FREE) Programme of the UK Natural Environment Research Council (NERC) (NE/E0002129/1), coordinated and monitored by Professor Chris Collier and Paul Hardaker. The numerical calculations have been carried out on the HPC facility at the University of Plymouth.

References

[1] B.D. Nichols, C.W. Hirt, R.S. Hotchkiss, SOLA-VOF: a solution algorithm for transient fluid flow with multiple free boundaries, Tech. Rep. LA-8355, Los Alamos National Laboratory, 1980.
[2] C.W. Hirt, B.D. Nichols, Volume of fluid method for the dynamics of free boundaries, J. Comput. Phys. 39 (1981) 201–225.
[3] D.L. Youngs, Time-dependent multi-material flow with large fluid distortion, in: K.W. Morton, M.J. Baines (Eds.), Numerical Methods for Fluid Dynamics, Academic Press, London, 1982, p. 273.
[4] N. Ashgriz, J.Y. Poo, FLAIR: flux line-segment model for advection and interface reconstruction, J. Comput. Phys. 93 (1991) 449–468.
[5] S. Welch, J. Trapp, G. Mortensen, Interface tracking in two-phase flow simulations using a simple subgrid counting procedure, in: Numerical Methods in Multiphase Flows, ASME, FED., vol. 185, 1994, pp. 293–299.
[6] J. Rider, D.B. Kothe, Stretching and tearing interface tracking methods, in: 12th AIAA Computational Fluid Dynamics Conference, June 20, 1995, San Diego, Paper Number AIAA-95-1717 or LA-UR-95-1145, 1995.
[7] W.J. Rider, D.B. Kothe, Reconstructing volume tracking, J. Comput. Phys. 141 (1998) 112–152.
[8] O. Ubbink, R.I. Issa, A method for capturing sharp fluid interfaces on arbitrary meshes, J. Comput. Phys. 153 (1999) 26–50.
[9] R. Löhner, C. Yang, E. Oñate, On the simulation of flows with violent free surface motion, in: 44th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, 2006.
[10] B.P. Leonard, The ULTIMATE conservative difference scheme applied to unsteady one-dimensional advection, Comput. Meth. Appl. Mech. Eng. 88 (1991) 17–74.
[11] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible 2-phase flow, J. Comput. Phys. 114 (1) (1994) 146–159.
[12] 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 (4) (1999) 1165–1191.
[13] 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 (2) (2000) 301–337.
[14] P. Liovic, D. Lakehal, Multi-physics treatment in the vicinity of arbitrarily deformable gas–liquid interfaces, J. Comput. Phys. 222 (2007) 504–535.
[15] X. Yang, A.J. James, et al, An adaptive coupled level-set/volume-of-fluid interface capturing method for unstructured triangular grids, J. Comput. Phys. 214 (2006) 41–54.
[16] A. Jameson, D.J. Mavriplis, Finite volume solution of the two-dimensional Euler equations on a regular triangular mesh, AIAA J. 24 (1986) 611.
[17] A. Jameson, Time Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings, AIAA Paper 91, 1991, p. 1596.
[18] J.U. Brackbill, D.B. Kothe, C. Zemach, A continuum method for modelling surface tension, J. Comput. Phys. 100 (1992) 335.
[19] Y. Zhao, C.H. Tai, Higher-order characteristics-based methods for incompressible flow computation on unstructured grids, AIAA J. 39 (7) (2001) 1280–1287.
[20] Y. Zhao, H.H. Tan, B.L. Zhang, A high-resolution characteristics-based implicit dual time-stepping VOF method for free surface flow simulation on unstructured grids, J. Comput. Phys. 183 (2002) 233–273.
[21] P.L. Roe, Approximate riemann solvers, parameter vectors and difference schemes, J. Comput. Phys. 43 (1981) 357–372.
[22] Bram van Leer, Towards the ultimate conservation difference scheme V, a second-order sequel to Godunov's method, J. Comput. Phys. 32 (1979) 101–136.
[23] D. Drikakis, P.A. Govatsos, D.E. Papantonis, A characteristic-based method for incompressible flows, Int. J. Numer. Meth. Fluids 19 (8) (1994) 667–685.
[24] S.E. Rogers, D. Kwak, C. Kiris, Steady and unsteady solutions of the incompressible Navier–Stokes equations, AIAA J. 29 (4) (1991) 603–610.
[25] T.J. Pedley, K.D. Stephanoff, Flow along a channel with a time-dependent indentation in one wall: the generation of vorticity waves, J. Fluid Mech. 160 (1985) 337–367.
[26] H. Lou, J.D. Baum, R. Löhner, An accurate fast, matrix-free implicit method for computing unsteady flows on unstructured grids, Comput. Fluids 30 (2001) 137–159.
[27] X. Lv, Y. Zhao, et al, An efficient parallel/unstructured-multigrid preconditioned implicit method for simulating 3D unsteady compressible flows with moving objects, J. Comput. Phys. 215 (2) (2006) 661–690.
[28] X. Lv, Y. Zhao, et al, A matrix-free implicit unstructured multigrid finite volume method for simulating structural dynamics and fluid–structure interaction, J. Comput. Phys. 215 (2) (2006) 661–690.
[29] H. Gaskell, A.K.C. Lau, Int. J. Numer. Meth. Fluids 8 (1988) 617.
[30] R. Franke, Scattered data interpolation: tests of some methods, Math. Comput. 38 (1982) 181–200.
[31] J.B. Bell, P. Colella, H.M. Glaz, A second order projection method of the incompressible Navier–Stokes equations, J. Comput. Phys. 85 (1989) 257–283.
[32] K. Shahbazi, M. Paraschivoiu, J. Mostaghimi, Second order accurate volume tracking based on remapping for triangular meshes, J. Comput. Phys. 188 (2003) 100–122.
[33] A. Gilmanov, F. Sotiropoulos, A hybrid cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies, J. Comput. Phys. 207 (2) (2005) 457–492.
[35] J.C. Martin, W.J. Moyce, An experimental study of the collapse of liquid columns on a rigid horizontal plane, Philos. Trans. Roy. Soc. Lond. A244 (1952) 312–324.
[36] B.D. Nichols, C.W. Hirt, Improved free surface boundary conditions for numerical incompressible-flow calculations, J. Comput. Phys. 8 (1971) 434–448.
[37] B. Ramaswamy, M. Kawahara, Lagrangian finite element analysis applied to viscous free surface fluid flow, Int. J. Numer. Meth. Fluids 7 (1987) 953–984.
[38] A. Ghobadian, Development of a method for numerical simulation of flows with moving interfaces, National Power report TEC/L/0077/M91, 1991.
[39] J.V. Soulis, Computation of two-dimensional dam-break flood flows, Int. J. Numer. Meth. Fluids 14 (1992) 631–664.

[40] S. Koshizuka, H. Tamako, Y. Oka, A particle method for incompressible viscous flow with fluid fragmentation, Comput. Fluid Dynam. J. 4 (1) (1995) 29–46.
[41] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level set method for improved interface capturing, J. Comput. Phys. 183 (1) (2002) 83–116.
[42] E. Olsson, G. Kreiss, A conservative level set method for two phase flow, J. Comput. Phys. 210 (2005) 225–246.
[43] E. Olsson, G. Kreiss, S. Zahedi, A conservative level set method for two phase flow II, J. Comput. Phys. 225 (1) (2007) 785–807.
[44] S.J. Cummins, M.M. Francois, D.B. Kothe, Estimating curvature from volume fractions, Comput. Struct. 83 (2005) 425–434.
[45] M.M. Francois, S.J. Cummins, E.D. Dendy, et al, A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework, J. Comput. Phys. 213 (2006) 141–173.
[46] S. Afkhami, M. Bussmann, Height functions for applying contact angles to 2D VOF simulations, Int. J. Numer. Meth. Fluids 57 (2008) 453–472.
[47] M. Sussman, K.M. Smith, M.Y. Hussaini, M. Ohta, R. Zhi-Wei, A sharp interface method for incompressible two-phase flows, J. Comput. Phys. 221 (2007) 469–505.
[48] P.Z. Lin, Philip L.-F. Liu, A numerical study of breaking waves in the surf zone, J. Fluid Mech. 359 (1998) 239–264.
[49] J.M. Yang, F. Stern, A sharp interface method for two-phase flows interacting with moving bodies, in: AIAA 2007-4578, 18th AIAA Computational Fluid Dynamics Conference, Miami, FL, 25–28 June 2007.
[50] D.A. Knoll, D.E. Keyes, Jacobian-free Newton–Krylov methods a survey of approaches and applications, J. Comput. Phys. 193 (2004) 357–397.
[51] Ronald P. Fedkiw, Tariq Aslam, Barry Merriman, Stanley Osher, A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method), J. Comput. Phys. 152 (2) (1999) 457–492.
[52] S. Popinet, S. Zaleski, Bubble collapse near a solid boundary: A numerical study of the influence of viscosity, J. Fluid Mech. 464 (1999) 137–163.
[53] T.G. Liu, B.C. Khoo, C.W. Wang, The ghost fluid method for compressible gas–water simulation, J. Comput. Phys. 204 (2005) 193–221.
[54] X.-D. Liu, R. Fedkiw, M. Kang, A boundary condition capturing method for Poisson's equation on irregular domains, J. Comput. Phys. 160 (2000) 151–178.
[55] M. Kang, R. Fedkiw, X.-D. Liu, A boundary condition capturing method for multiphase incompressible flow, J. Sci. Comput. 15 (2000) 323–360.
[56] D. Enright, D. Nguyen, F. Gibou, and R. Fedkiw, Using the particle level set method and a second order accurate pressure boundary condition for free surface flows, in: M. Kawahashi, A. Ogut, Y. Tsuji (Eds.), Proceedings of the 4th ASME–JSME Joint Fluids Engineering Conference, FEDSM2003-45144, Honolulu, HI, 2003, pp. 1–6.
[57] F. Gibou, R.P. Fedkiw, L.-T. Cheng, M. Kang, A second-order-accurate symmetric discretization of the Poisson equation on irregular domains, J. Comput. Phys. 176 (1) (2002) 205–227.
[58] A.J. Chorin, A numerical method for solving incompressible viscous flow problems, J. Comput. Phys. 2 (1967) 12–26.
[59] C.H. Tai, Y. Zhao, Parallel unsteady incompressible viscous flow simulation using an unstructured multigrid method, J. Comput. Phys. 192 (1) (2003) 277–311.