Computers & Fluids 118 (2015) 293–304

Contents lists available at ScienceDirect

Computers & Fluids

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

A three-dimensional volume of fluid & level set (VOSET) method for incompressible two-phase flow

Kong Ling [a], Zhao-Hui Li [a], Dong-Liang Sun [b], Ya-Ling He [a], Wen-Quan Tao [a,*]

[a] Key Laboratory of Thermo-Fluid Science & Engineering of MOE, School of Energy & Power Engineering, Xi'an Jiaotong University, China
[b] School of Mechanical Engineering, Beijing Institute of Petrochemical Technology, China

ARTICLE INFO

Article history:
Received 30 September 2014
Received in revised form 2 May 2015
Accepted 13 June 2015
Available online 18 June 2015

Keywords:
VOF
Level set
Distance function
Two-phase flow
3D VOSET

ABSTRACT

This paper presents a three-dimensional VOSET method, which combines both advantages of the VOF and level set. The basic idea is the same with that in 2D VOSET, but new methods are proposed to deal with new geometric problems in three dimensions. An iterative root-finding method is used for implementing Piecewise Linear Interface Construction (PLIC) in three dimensions. An iterative geometric method is presented to calculate the level-set function. The feasibility and accuracy of this method are analyzed by some classical test problems. Combining the use of projection method that solves the flow field, the 3D VOSET is adopted to simulate a liquid dam break problem and a single-bubble rising problem. Our numerical results are found to be in good accordance with those in previous studies.

1. Introduction

Flow with moving interfaces can be widely found in the daily life and industrial processes, such as moving of droplets and bubbles in gas–liquid flow, ash deposition on tubes in gas–solid flows and phase change heat transfer. In recent years, with the fast development of computer's performance, direct numerical simulation is becoming an increasingly important tool for studying these kinds of problems.

In the past decades, a number of different methods were developed for dealing with moving interfaces. Among these methods, volume of fluid (VOF) [1] and level set [2] were probably the most widely used ones. VOF was proposed by Hirt and Nichols [1]. In this method, a color function C ranging from 0 to 1, which we call VOF function here, is used to identify the two phases divided by the interface. The VOF function denotes the volume proportion of one phase in a computational cell. $C = 1$ demonstrates that this cell is filled with one phase and $C = 0$ means that the cell is filled with the other phase. If C values between 0 and 1, the cell is regarded to contain both phases. The major advantage of VOF is that the total mass conservation of the two phases can be kept very well. However, the interface normal and curvature cannot be solved accurately in a convolved scheme due to the fact that the VOF function is a step function, and it will produce large numerical error in surface tension. Level set method was proposed by Sethian and Osher [2]. In this method, a continuous function ϕ named level-set function is used to identify the interface and the two phases. The whole computational domain is divided into two parts according to the sign of the level-set function. The first part refers to the area occupied by the first phase where it satisfies $\phi > 0$, and $\phi < 0$ represents the location in the other phase. The interface, therefore, is denoted by the zero contour of level-set function. In the level set method, interface normal and curvature can be computed much more accurately due to the use of the continuous function. On the other side, however, the level set method suffers from the weakness of loss/gain of mass.

As can be seen that, both VOF and level set have disadvantages, but in different aspects. Therefore, the combination of the two methods could be a good choice for improvement. In 2000, Sussman proposed a coupled level set and VOF (CLSVOF) method [3], which successfully combines both advantages of VOF and LS. However, the advection equations for the VOF and level-set functions both need to be solved, which makes this method more complicated. As a simpler approach, a coupled method of volume-of-fluid and level set (VOSET), which combines the advantages of VOF and level-set as well, was proposed by Sun and Tao [4]. In this method, only the advection equation of the VOF function is solved, and the level-set function is computed from the VOF function independently. However, Sun and Tao's method is developed only in two dimensions. Recently, Sun [5] has extended VOSET to three dimensions. However, Sun's method needs to use much more classifications on different interface shapes. On the extension of VOSET to 3D, the present authors try simpler approaches.

* Corresponding author. Tel.: +86 029 82669106.
E-mail address: wqtao@mail.xjtu.edu.cn (W.-Q. Tao).

http://dx.doi.org/10.1016/j.compfluid.2015.06.018
0045-7930/

The main purpose of this study is to implement the major idea of VOSET [4] for three-dimension case. That is to say, Piecewise Linear Interface Construction (PLIC) [6] is used to reconstruct the interface shape and geometrical method is adopted in computing the level-set function. However, in order to implement this idea for 3D case, more complicated geometrical problems have to be dealt with, thus one needs to develop new techniques different from those in 2D VOSET [4] so that the 3D VOSET can be conducted in a simple way. Concretely, in the 2D VOSET, the interface is reconstructed as a set of straight line segments by PLIC. In 3D case, however, the interface should be approximated as a set of polygons. In the present article, we will extend PLIC to 3D cases, and the extension has some differences from existing 3D PLIC methods in literatures [5,7]. Furthermore, a geometrical approach will be proposed to compute the level-set function based on the reconstructed polygonal interfaces. These approaches will make the extension of VOSET from 2D to 3D simple. The present study is limited to three-dimensional Cartesian coordinates with uniform grid system.

The rest of the present paper is organized as follows. In Section 2, PLIC interface reconstruction method for solving three dimensional VOF function is introduced. Section 3 presents the method for solving the signed distance function in three dimensions. The algorithm for solving the flow field is illustrated in Section 4. In Section 5, some test problems are studied by the proposed 3D VOSET. Finally some conclusions are made in Section 6.

2. 3D PLIC interface reconstruction

For incompressible flow, the advection equation of VOF function can be written as:

$$\frac{\partial C}{\partial t} + \nabla \cdot (\boldsymbol{u} C) = 0. \tag{1}$$

The discretization of this equation leads to the following result:

$$C^{n+1} = C^n - \frac{\Delta t}{\Delta V} \oint_\Gamma \mathbf{u}^{n+1/2} C^n \mathrm{d}A. \tag{2}$$

For solving this equation numerically, an interface reconstruction method is needed to keep the sharpness of interfaces. For this problem, Piecewise Linear Interface Construction (PLIC) presented by Youngs [6] was widely used in the past decades. However, the extension of the PLIC from 2D to 3D is not straightforward, since the implementation of PLIC in 2D is a plane geometric problem, while in 3D it becomes a solid geometric problem.

To resolve this difficulty, Gueyffier et al. proposed a cube-cutting function [7], which thereafter became the basic method for three-dimensional PLIC. The cube-cutting function is presented briefly as follows.

For a 3D computational cell containing interface, by using some turning and scaling, the reconstructed interface can be described by a plane equation in a unit cube:

$$n_x x + n_y y + n_z z = \alpha, \tag{3}$$

where n_x, n_y and n_z are the three components of the interface normal in x, y and z directions, respectively. The interface normal can be easily estimated by level-set function, but α remains to be determined for fully description of the reconstructed interface.

For a given value of α, the volume below the interface in cube $(0, c_1) \times (0, c_2) \times (0, c_3)$, as illustrated in Fig. 1, can be determined by:

$$Volume(\alpha) = \frac{1}{6 n_1 n_2 n_3} \left[\alpha^3 - \sum_{j=1}^{3} H(\alpha - n_j c_j)(\alpha - n_j c_j)^3 + \sum_{j=1}^{3} H(\alpha - \alpha_{\max} + n_j c_j)(\alpha - \alpha_{\max} + n_j c_j)^3 \right]. \tag{4}$$

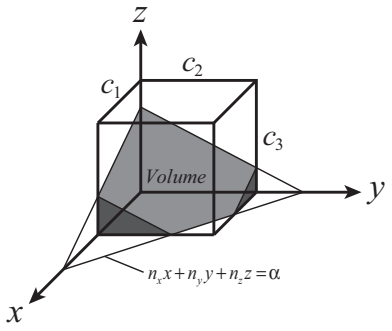

Fig. 1. The volume below an interface in cubic cell.

Here n_1, n_2 and n_3 are equivalent with n_x, n_y and n_z. c_1, c_2 and c_3 are the lengths of three sides of the cube, $\alpha_{\max} = n_x c_1 + n_y c_2 + n_z c_3$, and H is the Heaviside function:

$$H(x) = \begin{cases} 0 & (x < 0) \\ 1 & (x \geqslant 0) \end{cases}.$$

Therefore, the key of interface reconstruction is how to compute the value of α. By using Eq. (4), the VOF function can be obtained directly from a given α. However, what we need here is the inverse, that is, to solve α from a given VOF function. For solving this problem, Gueyffier et al. [7] classified the interface shapes into 12 cases. For each case, Eq. (4) turns out to be a cubic polynomial in α and the root can be computed directly. As a revision, Annaland et al. classified different interface shapes into 5 cases [8]. Figure 2 shows some examples of different interface shapes inside a cubic cell. Anyway, the classification makes the coding complicated.

Actually, the cube-cutting function *Volume* in Eq. (4) is a piecewise cubic polynomial in α, and it is applicable to various interface shapes owing to the use of Heaviside function. Furthermore, it is smooth and monotonically increasing with α within the range $0 \leqslant \alpha \leqslant \alpha_{\max}$. Figure 3 shows some examples of the cube-cutting function (Eq. (4)) with some different normal directions, which demonstrates the two features of this function. According to these features, one can use an iterative root-finding method to solve the value of α that corresponds to a given VOF function. An advantage of using an iterative root-finding method is that we do not need to write different codes for different kinds of interface shapes, which makes the extension of PLIC from 2D to 3D much easier. In fact, it has been mentioned in Ref. [7] that a root-finding method may be feasible. In the present study, a simple iterative secant method is adopted. Here we write a pseudo code as follows to describe how we use this method to solve the value of α from a VOF function (denoted by C) with given $n_{1\sim3}$ in a unit cube. Figure 4 illustrates one iteration in this method.

```
Set: error = 10^-9
Set: α1 = 0, C1 = 0, α2 = n1 + n2 + n3, C2 = 1
Set: αmiddle = (α1 + α2)/2
Loop
  Set: Cmiddle = Volume(αmiddle) /*See Eq. (4)*/
  If (|Cmiddle − C| < error) then
    Set: α = αmiddle
    Exit loop
  End if
  If (Cmiddle < C) Then
    Set: α1 = αmiddle, C1 = Cmiddle
  Else
    Set: α2 = αmiddle, C2 = Cmiddle
  End if
  Set: αmiddle = α1 + (α1−α2)/(C1−C2) (C − C1)
End loop
```

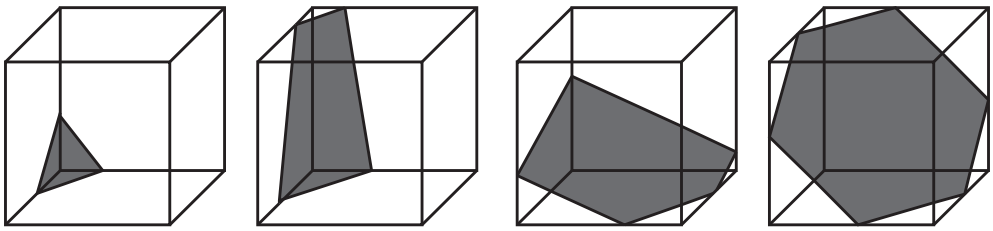

Fig. 2. Some examples of different polygon interface shapes inside a cube cell.

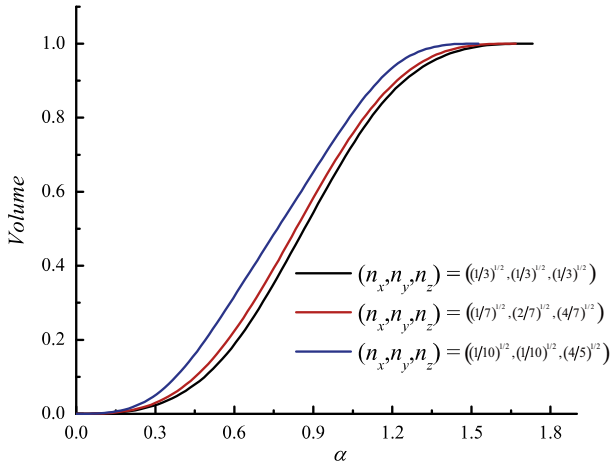

Fig. 3. Some examples of the cube-cutting function: a smooth and monotonically increasing function with α.

It has been found that by using the secant method the relative error can fall below 10^{-9} within 10 iterations in most conditions.

3. Geometric method for computing level-set function

Level-set function, denoted by ϕ, is defined as a signed distance function. The absolute value of ϕ refers to the shortest distance to interfaces. It has been demonstrated in Section 1 that both the commonly used CLSVOF [3,17] and VOSET [4] are combinations of VOF and level set methods. The major difference between them lies in the strategy how to update the level-set function. In CLSVOF [3,17], the level-set function is first updated by solving the advection equation of the level-set function based on the velocity field, and then it may be modified. In VOSET [4], the level-set function is computed directly from the VOF function that has been updated. The basic idea of computing the level-set function in 2D VOSET is

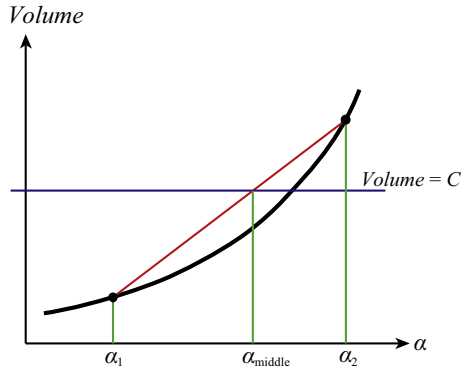

Fig. 4. Schematic of the procedure of one iteration in secant root-finding method used in solving α from a VOF function.

to find the shortest distance between a line segment and a given point, as shown in Fig. 5(a). It can be easily computed in a triangle for 2D case.

However, in 3D case, the problem becomes much more complicated due to the fact that the reconstructed interface in a 3D cube becomes a polygon. The corresponding problem is to find the point on the polygon closest to a given point (see Fig. 5(b)). The number of the polygon's sides, affected by the normal direction as well as the VOF function, may vary from 3 to 6 (see Fig. 2). Thus, the calculation of the level-set function in 3D case needs another method.

Before presenting our method for solving the level-set function, the two types of interfaces which are taken into account are described here. An interface belonging to the first type (we call Type-1 interface hereafter) lies between two adjacent cells which are filled by different phases. The interface is actually the face dividing the two cells, as shown in Fig. 6(a), and we call the two cells as Type-1 interfacial cells. An interface belonging to the second type (we call Type-2 interface hereafter) lies inside a cell that contains both phases, i.e., $0 < C < 1$, as shown in Fig. 6(b), and we call this cell as a Type-2 interfacial cell. In general, the interfaces appear mostly in Type-2 during the movement of the interface. Type-1 interface is actually a special case compared with the second one. Nevertheless, it usually appears in the first time step, especially if one initializes the distribution of the two phases simply by setting the VOF function in part of the computational cells as $C = 1$ and setting $C = 0$ in the rest cells.

A method for solving the distance function in three dimensions has been proposed by Sun [5]. However, in Sun's method, Type-1 interface (see Fig. 6(a)) was ignored. For Type-2 interfaces (see Figs. 2 and 6(b)), Sun [5] considered 12 possible interface shapes (partial cases are shown in Fig. 2) and carried on a unique calculating routine for each of them. In the method presented here, Type-1 interface will be taken into account individually, while for Type-2 interfaces, we present a simpler method with no need for classification on different interface shapes. In this regard, Wang et al. [17] has proposed an algorithm that also needs no classification on interface shapes to deal with the Type-2 interface. However, our method is different and has better applicability in different kinds of cells, which will be discussed later.

An iterative geometric method is adopted for the calculation of the 3D level-set function, which is the same as that in 2D VOSET. The procedure consists of five steps described as follows.

Step 1: Solve the interface normal and reconstruct the interface

Interfaces are reconstructed based on PLIC by an iterative method. In the first iteration, the interface normal is estimated by the smoothed VOF function $\widetilde{C}$:

$$\mathbf{n} = \frac{\nabla \widetilde{C}}{|\nabla \widetilde{C}|}. \tag{5}$$

By using PLIC, the interface is estimated by a plane plate, the equation of which is written as Eq. (3).

Step 2: Set initial value of level-set function

The initial level-set function is given as:

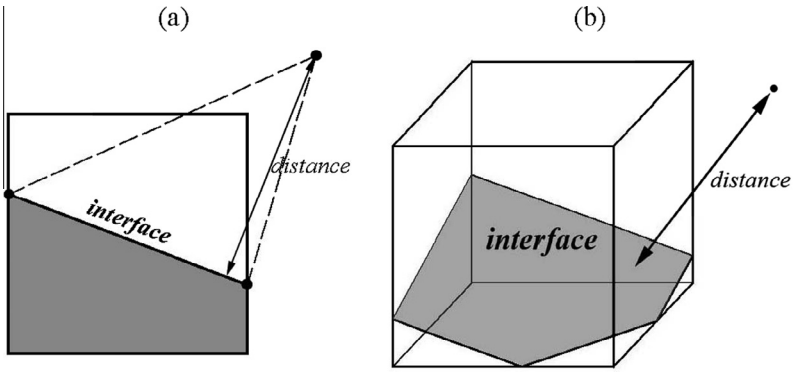

Fig. 5. Computing the shortest distance. (a) 2D case; (b) 3D case.

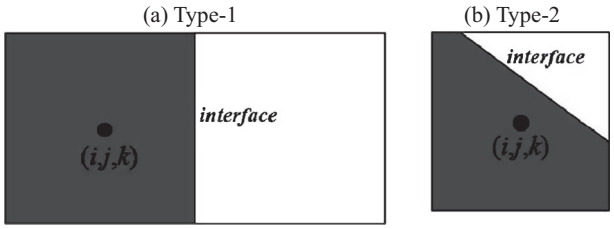

Fig. 6. Two types of interfaces: (a) Type-1; (b) Type-2.

$$\phi_0 = \begin{cases} 10h & \text{if } C \geqslant 0.5 \\ -10h & \text{if } C < 0.5 \end{cases}, \tag{6}$$

where h refers to the size of a computational cell.

Step 3: Mark the interfacial cells and their types

The interfacial cells, either containing interfaces inside them (Type-2, Fig. 6(b)) or having interfaces on their faces (Type-1, Fig. 6(a)), are marked in the present step. Concretely, the following two sub-steps are executed.

(1) Visit all the cell faces and check whether they are Type-1 interfaces. In one visit to a cell face, if the VOF functions in the two adjacent cells divided by the face satisfy: $C_{\text{one}} < \epsilon$, and $C_{\text{the other}} > 1-\epsilon$, it indicates that the face is a Type-1 interface. Thus, the two cells are marked as Type-1 interfacial cells. Note that ϵ is the critical value used in numerical implementation and we set $\epsilon = 10^{-6}$.

(2) Visit all the cells and check whether they have interfaces inside them. In one visit to a cell, if the VOF function satisfies: $\epsilon \leqslant C \leqslant 1-\epsilon$, it indicates that the cell contains a Type-2 interface and the cell is marked as a Type-2 interfacial cell.

Step 4: Mark the cells near interfaces

The purpose of this step is to save computational time, because the level-set function is needed only in the regions near the interface. In this study, cells within $5h$ away from the interfacial cells (Type-1 or Type-2) are marked. This distance can make the marked region thick enough for computing the interface normal and curvature. The distance function will be calculated merely in the cells marked in the present step, and the method will be discussed in Step 5.

The above four steps are basically the same with those in 2D VOSET proposed by Sun and Tao [4].

Step 5: Calculate level-set function in marked region

This step is the key for calculating the level-set function. As shown in Fig. 7, for a marked cell M (i,j,k), nearby cells with distance less than $5h$ are checked whether they are the interfacial cells marked in Step 3. Supposing M′ (i',j',k') is one of these cells, we need to find the shortest distance from M to the interface related with cell M′. For this purpose, the type of the interfacial cell M′ (see Fig. 6), which has been marked in Step 3, will be examined here.

We first consider the case that M′ belongs to Type-1 (see Fig. 6(a)) which is actually much simpler to deal with compared with the second type. Without loss of generality, we suppose that the interface normal is in x direction. The interface center is $(x_f, y_{j'}, z_{k'})$, in which $x_f = x_{i'} + h/2$ or $x_f = x_{i'} - h/2$. The shortest distance is computed by:

$$distance = \sqrt{x_{\text{diff}}^2 + y_{\text{diff}}^2 + z_{\text{diff}}^2}, \tag{7}$$

where $x_{\text{diff}} = |x_i - x_f|$, $y_{\text{diff}} = \max(|y_j - y_{j'}| - h/2, 0)$, $z_{diff} = \max(|z_k - z_{k'}| - h/2, 0)$. Figure 8 illustrates an example of Type-1 interface as well as the method calculating the shortest distance to a given point.

Now attention is turned to the case that M′ is a Type-2 interfacial cell (see Fig. 6(b)). As mentioned above, the interface inside this cell is in Type-2 and it has been reconstructed as a polygon. In this case, the following procedures are implemented.

First, the projection point of M on the interface, denoted by point P in Fig. 9, is solved. According to the reconstructed interface equation (Eq. (3)), the coordinates of P can be computed by:

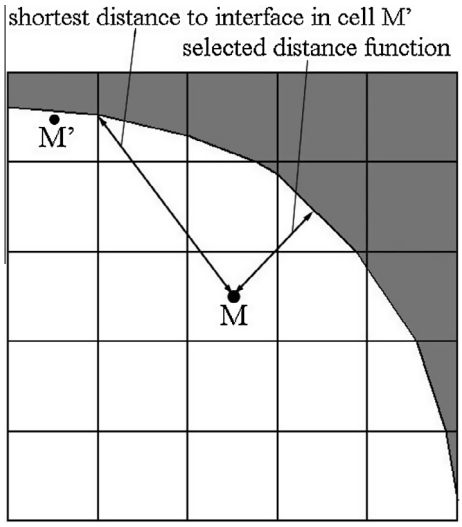

Fig. 7. Computing the distance function.

$$d = \alpha - (n_x x_i + n_y y_j + n_z z_k), \tag{8-a}$$

$$x_P = x_i + n_x d, \tag{8-b}$$

$$y_P = y_j + n_y d, \tag{8-c}$$

$$z_P = z_k + n_z d. \tag{8-d}$$

If P is located inside cell M′, i.e. $x_{i'} - h/2 \leqslant x_P \leqslant x_{i'} + h/2$, $y_{j'} - h/2 \leqslant y_P \leqslant y_{j'} + h/2$ and $z_{k'} - h/2 \leqslant z_P \leqslant z_{k'} + h/2$, point P is the closest point to M. Therefore, the distance is computed by:

$$distance = |d| = |\alpha - (n_x x_i + n_y y_j + n_z z_k)|. \tag{9}$$

If P is outside the cell, it implies that the closest point is on one side of the polygon. In order to find the shortest distance in this condition, we use a straightforward method, i.e., solving the shortest distance to each side of the polygon and then choosing the minimum value of them.

The polygon has some characteristics that advantages can be taken of.

a. All the polygon's sides are on the faces of the cube. These sides are the intersecting lines of the interface and the cube's faces.
b. Each face of the cube contains only one side at most, since two planes cannot have two or more intersection lines.
c. The vertices of the polygon are the intersection points of the interface and the cube's edges.

Based on these characteristics, an algorithm is presented for computing the distance for Type-2 interface (Fig. 6(b)).

(1) Compute intersection points of the interface and the cube's 12 edges. As an example, we illustrate the case for edge AB in Fig. 9 in z-direction. According to the interface's equation (Eq. (3)), one can get the coordinates of the intersection point:

$$x_{\text{inter}} = x_{i'} + h/2, \tag{10-a}$$

$$y_{\text{inter}} = y_{j'} - h/2, \tag{10-b}$$

$$z_{\text{inter}} = (\alpha - n_x x_{\text{inter}} - n_y y_{\text{inter}})/n_z. \tag{10-c}$$

If it satisfies $z_{k'} - h/2 \leqslant z_{\text{inter}} \leqslant z_{k'} + h/2$, it indicates that edge AB contains an intersection point (point C shown in Fig. 9).

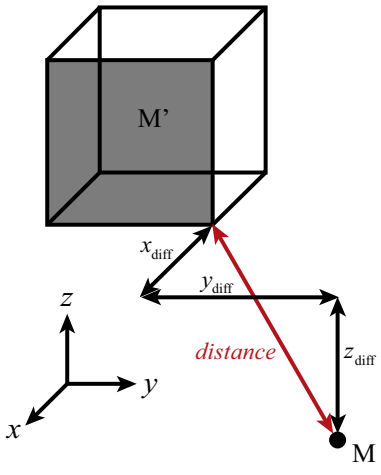

Fig. 8. The method computing the distance from a Type-1 interface to a given point.

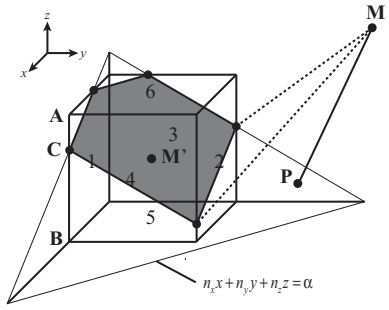

Fig. 9. The method computing the distance from a Type-2 interface to a given point.

(2) Check all the 6 faces surrounding the cube. For each face, check the four surrounding edges and count how many of them contain intersection points. If the face has 2 edges that contain intersection points, it indicates that the face contains one side of the crossing polygon. For examples, faces 1, 2, 3, 4 and 6 in Fig. 9 are in this situation. In this case, the shortest distance from point M to the side can be simply solved in a triangle whose vertices are M and the two end points of the side. An example of this triangle is marked by dashed lines in Fig. 9. This distance from point M to one of the polygon's side solved in this triangle is regarded as a candidate for the shortest distance from point M to the polygon. Face 5 in Fig. 9 contains no side of the polygon because none of its four surrounding edges contains intersection points.
(3) As the shortest distances to all the polygon's edges have been solved, the minimum of them is chosen as the shortest distance from the given point to the crossing polygon.

It deserves to be pointed out here that our algorithm is different from the one proposed by Wang et al. [17], although both of them calculate the level-set functions from reconstructed interfaces. The algorithm by Wang et al. [17] is based on geometrical characteristics of regular hexahedron, thus it is difficult to be applied in unstructured mesh which may contain many non-cubic cells. Differently, our algorithm is based on the relationship between the computational cell and the polygon interface inside it. Thus it is also applicable in other kinds of computational cells. It can be illustrated by an example of a tetrahedron cell shown in Fig. 10, where the distance from point M to the interface inside the cell can be solved by using similar procedures described in Step 5.

Here we summarize the routine of Step 5. For a given marked cell whose level-set function should be calculated, its nearby cells within certain distance are visited. If one of these cells is interfacial (Type-1 or Type-2), the distance from the center of the given marked cell to the corresponding interface (Type-1 or Type-2) is computed. As the visit to the nearby cells is finished, the smallest value of the distances obtained is selected as the distance function, i.e., the absolute value of the level-set function, in this marked cell. In numerical implementation, the shortest distance is selected by updating the current minimum distance, denoted by *distance* here, when a smaller distance is found during the visit to nearby cells.

For saving computational time, the nearby cells are visited from the core to outer, layer by layer. By adopting this visiting sequence, outer layers can be skipped as the visit in inner layers is enough. As shown in Fig. 11, in the nth layer, the shortest distance cannot be smaller than $(n - 1/2)h$. Therefore, after the visit to the $(n - 1)$th layer, if the current smallest distance has been found to be smaller than $(n - 1/2)h$, it can be regarded as the distance function and the outer layers can be skipped. It is different from the line-by-line visiting order adopted in [5]. Here we call this visiting order as the first improvement in saving computational time, and an example will be presented below to test its effect.

At the end of Step 5, the sign of the level-set function is determined by the VOF function:

$$\phi = \begin{cases} distance & (C > 0.5) \\ 0 & (C = 0.5) \\ -distance & (C < 0.5) \end{cases}, \tag{11}$$

which is the same as that executed in 2D VOSET [4].

As the above five steps are finished, an initial level-set function is obtained. However, in Step 1 the interface normal is computed by the VOF function. For a better accuracy, Steps 1–5 are repeated until the number of iterations reaches the preset value. From the second iteration, interface normal is computed by the level-set function obtained from the last iteration:

$$\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|}. \tag{12}$$

As recommended by Sun and Tao [4], three iterations are implemented for solving the level-set function.

Actually, the purpose of the former two iterations is to obtain a more accurate interface normal. For this purpose, we only need the level-set function in one-layer cells around the interfaces. Therefore, in the first and second iterations, we mark the cells within only one layer around the interfacial cells (Step 4) and compute the level-set functions there (Step 5). Only in the third iteration, cells within five layers are marked. This treatment also can save much computational time. It is the second improvement in saving computational time.

In order to check the feasibility of our level-set function generating algorithm as well as the effect of the two improvements in saving computational time, we computed the level-set function for a given round ball in a computational domain of $1 \times 1 \times 1$. The ball is located at the center with radius of 0.25. Computation was executed on a 2.00 GHz Intel Xeon CPU. Table 1 displays the computational time cost by using different methods, i.e., the original method used in [5], the first and second improvements in for saving computational time. From the comparison we can find obvious advantages of the proposed techniques in saving computational time. Figure 12 displays the contours of the signed distance function around the interface in a symmetric cross section (computed in 64^3 grids using the 1st & 2nd improvements). It demonstrates the feasibility of the method presented in this Section for the calculation of the level-set function in three dimensions.

4. Method for solving pressure and flow fields

The Navier–Stocks equation and continuity equation are solved in fluid region, either liquid or gas phase. For incompressible two-phase flow, the governing equations can be written as follows.

$$\rho\frac{\partial\mathbf{u}}{\partial t} + \rho(\mathbf{u}\cdot\nabla)\mathbf{u} = -\nabla p + \nabla\cdot\eta(\nabla\mathbf{u} + \nabla\mathbf{u}^T) + \rho\mathbf{g} + \sigma\kappa\nabla H, \tag{13}$$

$$\nabla\cdot\mathbf{u} = 0. \tag{14}$$

where $\sigma\kappa\nabla H$ is the term representing the effect of surface tension.

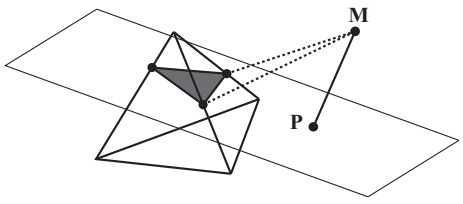

Fig. 10. A method solving the distance from point M to an interface in a tetrahedron cell.

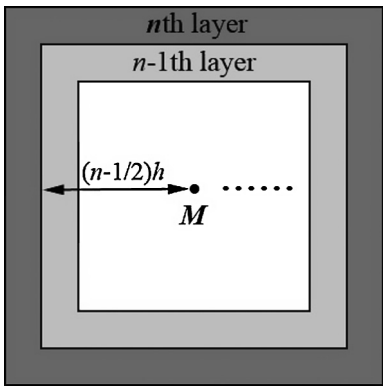

Fig. 11. Visiting sequence for solving a distance function.

Table 1
Computational time cost in computing level-set function by original and improved methods.

Grids	Original (s)	1st improvement (s)	1st & 2nd improvements (s)
32^3	1.342	0.624	0.219
64^3	5.195	2.401	0.89
128^3	21.029	9.61	3.76

The momentum equation is discretized by finite volume method (FVM) [9,10] in staggered grid. We applied projection method [11] for solving the pressure and velocity fields. A second-order Runge–Kutta method is used for time advancement. QUICK scheme [12] is used for the discretization of the convection term and central difference scheme for the diffusion term. Surface tension is computed as a source term of momentum equation by the balanced CSF (bCSF) [13], a revision of CSF model. Bi-CGSTAB [14,15], a Krylov subspace method, is adopted for solving the algebraic equations.

In the momentum equation, fluid properties and interface curvature are computed by the level-set function. The method has been illustrated in [4]. For readers' convenience, here they are briefly listed as follows.

$$H(\phi) = \begin{cases} 0 & (\phi < -\varepsilon) \\ \frac{1}{2}\left[1 + \frac{\phi}{\varepsilon} + \frac{1}{\pi}\sin\left(\frac{\pi\phi}{\varepsilon}\right)\right] & (|\phi| \leqslant \varepsilon) \\ 1 & (\phi > \varepsilon) \end{cases}, \tag{15}$$

$$\rho = H\rho_l + (1-H)\rho_g, \tag{16-a}$$

$$\eta = H\eta_l + (1-H)\eta_g, \tag{16-b}$$

$$\kappa = \nabla\cdot\frac{\nabla\phi}{|\nabla\phi|}. \tag{17}$$

In summary, the procedure of 3D VOSET is presented as follows:

Computation starts from given initial value of the VOF function C^0, velocity $\mathbf{u}^0$, and pressure p^0.

At the nth time step:

1. Compute the level-set function ϕ^n from the VOF function C^n using the method presented in Section 3.
2. Compute the fluid properties, interface curvature and surface tension term by ϕ^n using Eqs. (15)–(17).
3. Solve the pressure p^{n+1} and velocity field $\mathbf{u}^{n+1}$ using the projection method. In this procedure, a velocity field at middle time $\mathbf{u}^{n+1/2}$ is computed of the second-order time advancement.

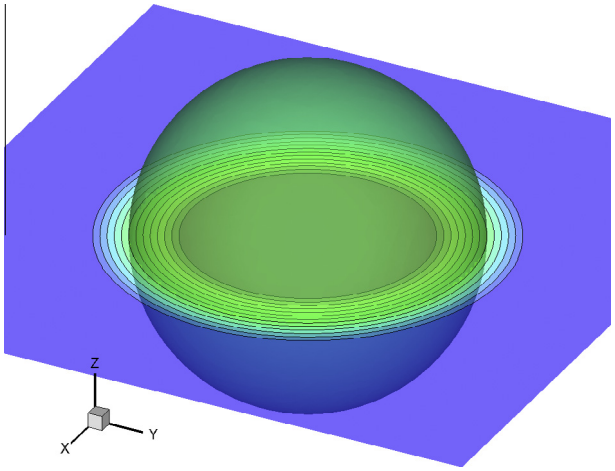

Fig. 12. The level-set function for a round ball.

4. Advect the VOF function by $\mathbf{u}^{n+1/2}$ using the method presented in Section 2, i.e., compute C^{n+1} from C^n and $\mathbf{u}^{n+1/2}$.

5. Numerical tests

5.1. 3D deformation test

In order to examine the three-dimensional VOSET presented in this paper, a test problem proposed by LeVeque [16] was studied. In this problem, an initial sphere is deformed and then recovered in a unit cube. The velocity field is given by:

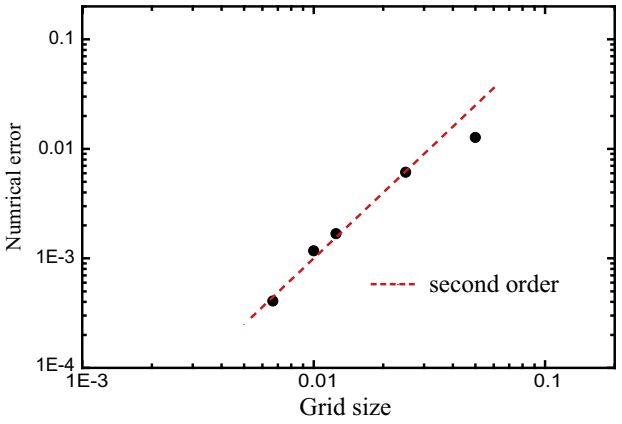

Fig. 13. Numerical error versus grid size for 3D deformation test.

$$\begin{aligned} u(x,y,z,t) &= 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos(\pi t/3) \\ v(x,y,z,t) &= -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)\cos(\pi t/3) \\ w(x,y,z,t) &= -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos(\pi t/3) \end{aligned} \tag{18}$$

The initial sphere's radius equals to 0.15 and it is centered at (0.35,0.35,0.35). After a whole period, the interface should return to the initial status. Therefore, the numerical error of this problem can be defined as:

$$Error = \sum |C - C^0|\Delta V. \tag{19}$$

By the proposed three-dimensional VOSET, the interface was captured in five grid sizes: 32^3, 64^3, 80^3, 100^3 and 150^3. Figure 13 displays the numerical errors versus grid sizes, showing a second-order accuracy.

Figure 14 shows the interfaces computed in grid of 150^3. At $t = 1.5$ (Fig. 14(b)), the deformation process is completed and the shape is stretched to the largest extent. The grid is still not fine enough to eliminate the holes on the thin film. Similar result was obtained by Wang et al. [17]. In this regard, Menard et al. [18] gave a better result. However, their method is based on the more complicated CLSVOF where level-set function and VOF functions both need to be advected. At the end of the recovering process when $t = 3$ (Fig. 14(c)), the interface generally returns spherical with a small deformation compared with the initial shape.

5.2. Spherical drop at equilibrium

The computation of this problem was executed in a domain of $0.5 \times 0.5 \times 0.5$. A spherical drop with radius of $R = 0.125$ is located at the center of the domain. The two phases have the same density and viscosity ($\rho = 4$ and $\eta = 1$). The surface tension is given as $\sigma = 0.357$ and there is no gravity. This problem was studied to test our surface tension model as well as the accuracy in calculation of interface curvature. As the exact solution, the velocity should be equal to zero in the whole domain. However, in numerical simulation, false flow will occur because of inevitable numerical error in computing surface tension. Simulations were carried out from $t = 0$ to $t = 0.01$ in three grid sizes, i.e., $16 \times 16 \times 16$, $32 \times 32 \times 32$ and $50 \times 50 \times 50$. The extent of the false flow can be reflected by the maximum velocity. Figure 15 shows the maximum velocities in the results by the three grids. It can be seen that, the maximum velocities are restricted at order of 10^{-3}. Furthermore, it can be decreased by using a finer mesh.

The exact pressure difference across the interface can be calculated by Young–Laplace equation:

$$\Delta p = 2\sigma/R. \tag{20}$$

Fig. 16 shows the pressure distribution on the center line in x direction obtained in 50^3 grids, and it is compared with the exact solution. Evidently, the pressure solved by the proposed 3D VOSET

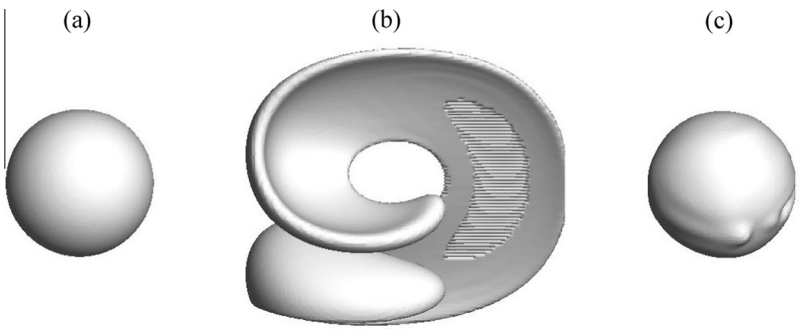

Fig. 14. Interface shape for 3D deformation test. (a) $t = 0$; (b) $t = 1.5$; (c) $t = 3$.

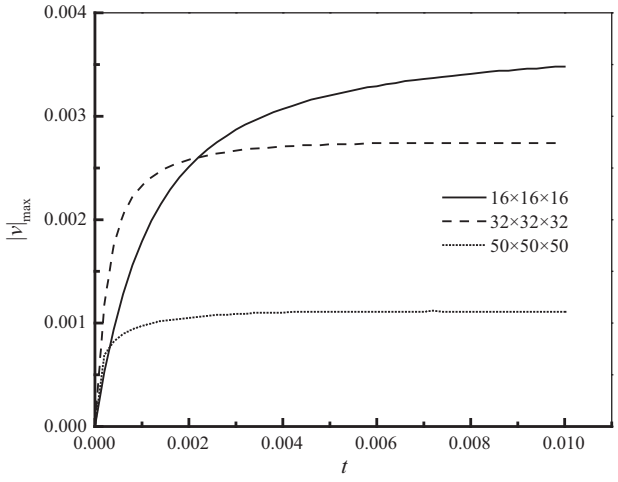

Fig. 15. The biggest velocities of static spherical drop problem.

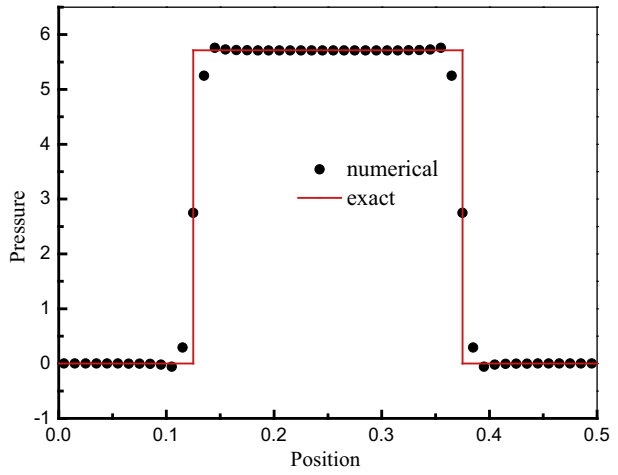

Fig. 16. Pressure distribution of static spherical drop problem.

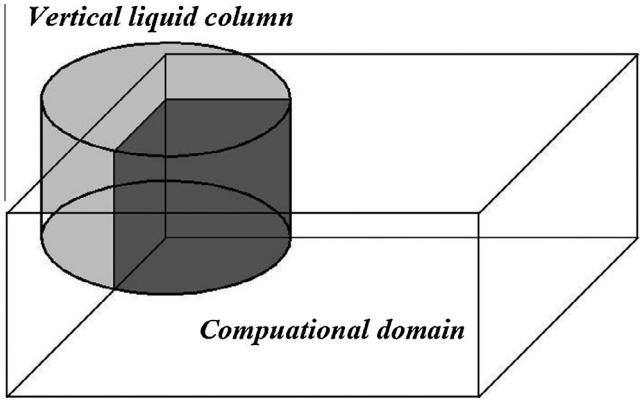

Fig. 17. Schematic of vertical cylindrical dam break problem.

coordinates well with the exact solution except for the slight smoothness around the interfaces, which is a result of the continuous-force surface tension model adopted here.

5.3. Vertical cylindrical liquid dam break

In order to testify the ability for simulating actual problems of the proposed three-dimension VOSET, we computed the vertical cylindrical dam-break problem, which has been experimentally studied by Martin and Moyce [19]. The problem and the computational domain we used are sketched in Fig. 17. The radius and height of the initial liquid column both equal to L (L = 2.25 inches). The densities of liquid and gas are ρ_l = 1000 kg/m^3 and ρ_g = 1.25 kg/m^3, respectively. The viscosities of the two phases are η_l = 0.001 Pa s and $\eta_g = 1.8 \times 10^{-5}$ Pa s, respectively. The surface tension coefficient is 0.072 N/m and the acceleration of gravity is 9.8 m/s^2.

Although cylindrical coordinate is a good choice for simulating this problem, it also can be computed in three-dimensional Cartesian coordinate discussed here (see Fig. 17). The computational domain is $3.6L \times 3.6L \times 1.2L$, and the mesh of $90 \times 90 \times 30$ with uniform size was used. Wall boundary condition was applied on top and bottom boundaries and the other four vertical faces were treated as symmetric boundary.

The interface evaluation is shown in Fig. 18. In the first stage, the base of liquid column expands under the effect of gravity, and the front of the column base keeps a circular shape before it reaches the far walls (Fig. 18(a)–(d)). Fig. 19 illustrates the radius of the liquid column base versus dimensionless time. Martin and Moyce's result (Table 5 in [19]) is displayed as well in this figure. It can be seen that our numerical results agrees very well with the experimental data, which proves the accuracy of 3D VOSET presented in this paper. After the liquid front collides with the wall, it begins to flow back and dashes against the following liquid, and then several waves are formed (Fig. 18(e), (f)).

5.4. Single rising bubble

A single rising bubble was simulated by the proposed 3D VOSET. A sphere gas bubble is initially placed in a static liquid and the bubble will rise due to the effect of buoyancy. The rising bubble may deform into different shapes under different conditions. It was concluded by Grace [20] that the terminal bubble shape can be determined by two dimensionless numbers, i.e., Morton number (M) and Eotvos number (Eo). Also, the Reynolds number (Re) reflecting the terminal rising velocity can be determined by M and Eo numbers. These dimensionless numbers are defined as follows:

$$M = \frac{g\Delta\rho\eta_l^4}{\sigma^3\rho_l^2}, \tag{21-a}$$

$$Eo = g\Delta\rho d_e^2/\sigma, \tag{21-b}$$

$$Re = \rho_l u_t d_e/\mu_l. \tag{21-c}$$

Here, d_e denotes the diameter of the initial bubble and $\Delta\rho = \rho_l - \rho_g$ is the density difference. u_t refers to the bubble's terminal rising speed.

In this problem, we simulated 4 cases with M and Eo numbers that have been studied by Bhaga and Weber [21] experimentally. Also, numerical simulations were conducted by Hua et al. [22] using front tracking method and by Bower and Lee [23] using lattice Boltzmann method. The dimensionless numbers and the expected terminal bubble shapes of these cases are displayed in Table 2.

Fig. 20 sketches the computational domain adopted for this problem. Symmetric boundary conditions were used for saving computational resources. The other two side walls were also set as symmetric boundaries. Periodic condition was given at the top

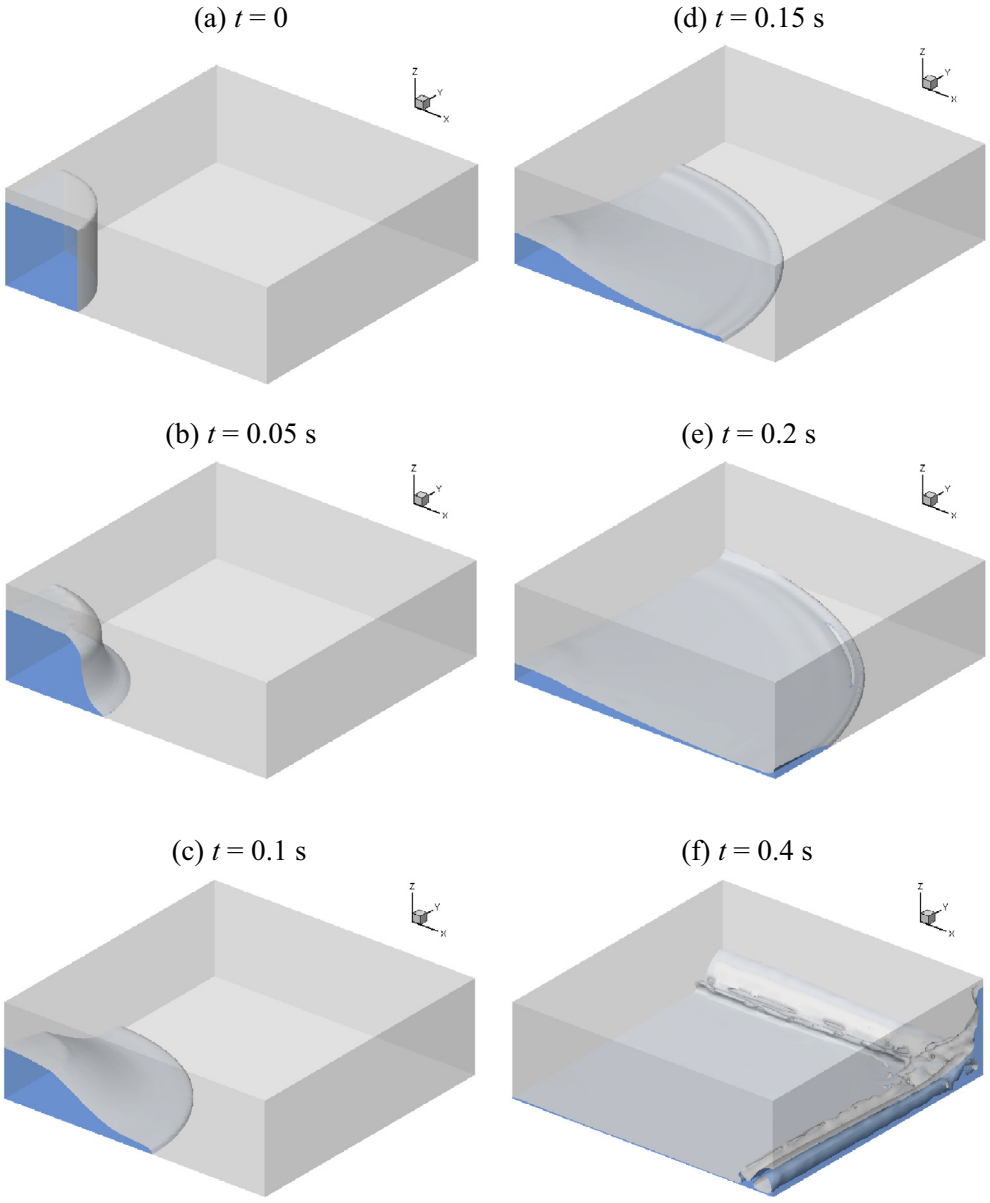

Fig. 18. Interface evolution of cylindrical dam break problem.

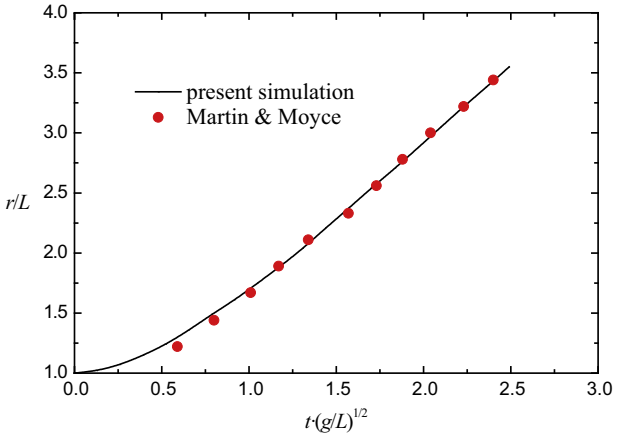

Fig. 19. History of liquid base radius.

and bottom boundaries. Densities of the liquid and gas phases are given as $\rho_l = 1000$ and $\rho_g = 1$, respectively. Viscosity ratio is $\eta_l/\eta_g = 100$. It has been proved that the density ratio and viscosity ratio used in this study are large enough [8,22,23]. The diameter of the initial bubble is $d_e = 0.01$ and the bubble is initially centered at $(0, 0, 0.01)$. The domain size should be large enough to decrease the wall viscous effect under an acceptable extent. In this regard, it has been discussed by former researchers [8,22,23] that a domain size of $5d_e$ is enough for most conditions. Thus, we chose this size in Case 2, Case 3 and Case 4. Since we adopted symmetric conditions on the bubble's cross sections, only 1/4 of the domain size is needed (see Fig. 20). For Case 1, a larger domain size was used because of the larger value of viscosity. The same as the study by Annaland [8], a mesh size of 0.001 was used for Case 1. For the other cases, smaller grid sizes were used because larger deformations of the bubble were expected. The computational domain sizes and the grid sizes are summarized in Table 2.

Figure 21 shows the terminal bubble shapes obtained by the proposed 3D VOSET method and by the experiments [21].

Table 2
Parameters used in single bubble rising simulation.

Cases	Case 1	Case 2	Case 3	Case 4
Morton number	711	8.2×10^{-4}	266	43.1
Eotvos number	17.7	32.2	243	339
Expected shape	Spherical	Elliptic	Dimpled	Skirted
Computational domain	$4d_e \times 4d_e \times 6d_e$	$2.5d_e \times 2.5d_e \times 10d_e$	$2.5d_e \times 2.5d_e \times 10d_e$	$2.5d_e \times 2.5d_e \times 10d_e$
Grids	$40 \times 40 \times 60$	$32 \times 32 \times 128$	$32 \times 32 \times 128$	$50 \times 50 \times 200$

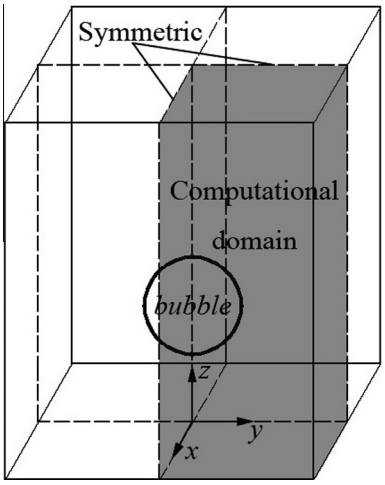

Fig. 20. Schematic of single bubble rising problem.

The figures of the experimental bubble shapes were copied form Hua et al. [22]. Evidently, the terminal bubble shapes obtained by our simulation are in accordance with the experimental results. The velocity fields around the bubbles in the central vertical planes are displayed in Fig. 22.

Table 3 shows the Reynolds numbers of the 4 cases obtained by our simulation. Also, results reported in literatures are listed for comparison. Generally, the Reynolds numbers obtained by our simulations are in good agreement with those by experimental study [21]. Compared with the results obtained by Bower and Lee [23] using Lattice Boltzmann Method, our results are closer to the experimental data. For Cases 2–4, however, Hua et al. [22] got the bubbles' rising velocities that are closer to those by experiments. The reason lies in that they used a larger domain size ($8d_e$) in x and y directions. In Case 1 where the viscous effect is more significant, we also used the larger domain size ($8d_e$), and a Reynolds number was obtained closer to the experimental result [21] compared with that by Hua et al. [22]. In summary, the results of the single bubble rising can prove the ability of the proposed 3D VOSET for simulating three-dimensional gas-liquid two-phase flows.

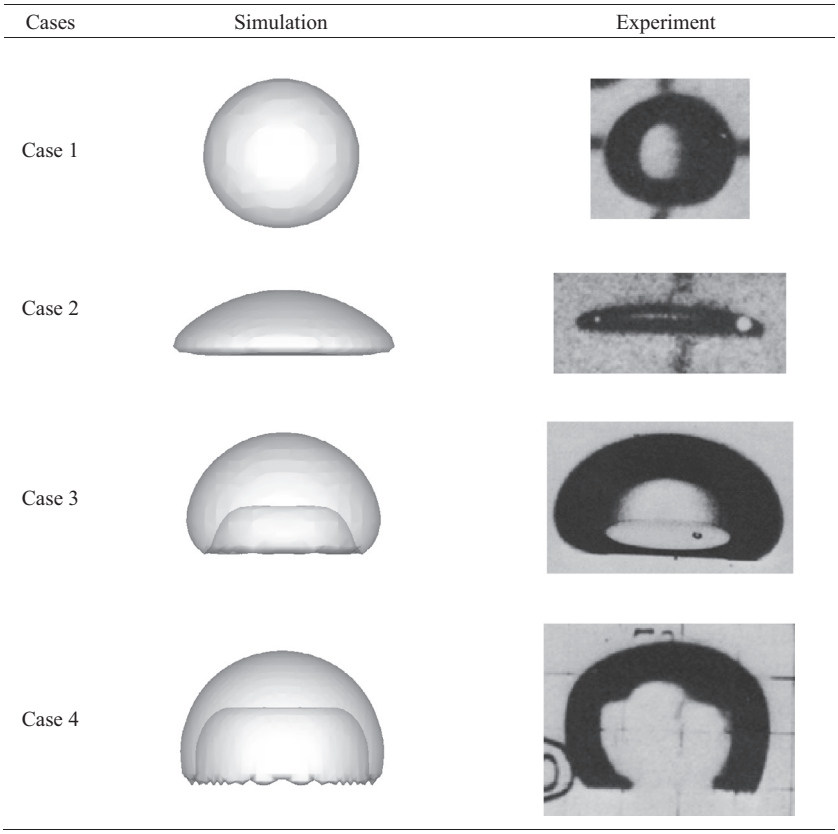

Fig. 21. Terminal bubble shapes.

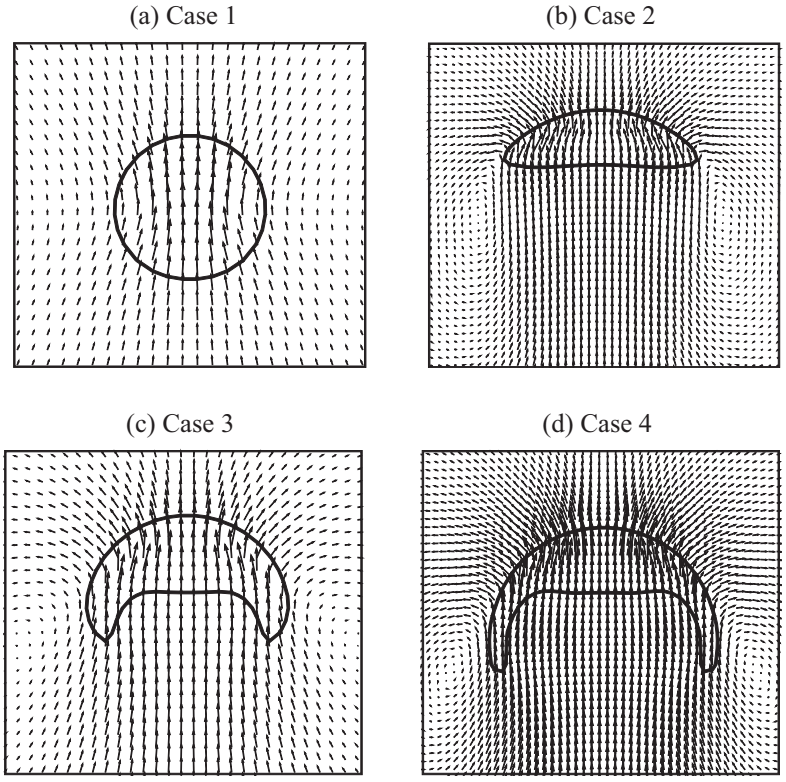

Fig. 22. Velocity fields in terminal state. (a) Case 1; (b) Case 2; (c) Case 3; (d) Case 4.

Table 3
Reynolds numbers of single rising bubble obtained by simulations and experiments.

Cases	Case 1	Case 2	Case 3	Case 4
Re_{exp} [21]	23.2	55.3	7.8	18.3
Re_{sim} in [22]	18.2	54.8	7.6	17.8
Re_{sim} in [23]	–	51.7	6.2	15.2
Re_{sim} present	20.3	52.5	7.3	16.8

6. Conclusion

In the present paper, we describe the extension of the coupled interface capturing method, VOSET, from 2D to 3D. A root-finding method based on the cube-cutting function is used in the implementation of 3D PLIC. For calculating the level-set function, we consider two types of interface, where the first type is a special case and the second one corresponds to a polygon interface inside a cubic cell. For the second type, an algorithm is presented without the need for classification on different interface shapes. By the use of these methods, geometrical difficulties in applying VOSET in three dimensions are overcome without any lost of accuracy. Furthermore, an iterative geometric method is presented to calculate the level-set function. In the numerical tests, the result of 3D deformation test shows a second order accuracy in interface tracking, and the accuracy in computing interface curvature and surface tension is illustrated by result of static spherical drop. In the numerical study of dam break problem and single rising bubble problem, the results obtained by the presented 3D VOSET agree well with those by experiments both qualitatively and quantitatively.

Acknowledgments

We gratefully acknowledge the support of this work by the National Key Basic Research Program of China (973 Program) (2013CB228304) and the Key Project of the National Natural Science Foundation of China (51136004).

References

[1] Hirt CW, Nichols BD. Volume of fluid (VOF) method for the dynamics of free boundary. J Comput Phys 1981;39:201–25.
[2] Osher S, Sethian JA. Fronts propagating with curvature dependent speed: algorithms based on Hamilton–Jacobi formulations. J Comput Phys 1988;79:12–49.
[3] Sussman M, Puckett EG. A coupled level set and volume-of-fluid method for computing 3D and axisymmetric incompressible two-phase flows. J Comput Phys 2000;162:301–37.
[4] Sun DL, Tao WQ. A coupled volume-of-fluid and level set (VOSET) method for computing incompressible two-phase flows. Int J Heat Mass Transfer 2010;53:645–55.
[5] Sun DL. Development of advanced velocity-pressure coupling algorithm and interface capturing method. Thesis for Doctor's Degree. Xi'an (PR China): Xi'an Jiaotong University; 2009.
[6] Youngs DL. Time-dependent multi-material flow with large fluid distortion, in numerical method for fluid dynamics. New York: Academic press; 1982.
[7] Gueyffier D, Li J, Nadim A, Scardovelli R, Zaleski S. Volume-of-fluid interface tracking with smoothed surface stress methods for three-dimensional flows. J Comput Phys 1999;152:423–56.
[8] van Sint Annaland M, Deen NG, Kuipers JAM. Numerical simulation of gas bubbles behaviour using a three-dimensional volume of fluid method. Chem Eng Sci 2005;60:2999–3011.
[9] Patankar SV. Numerical heat transfer and fluid flow. New York: McGraw-Hill; 1980.
[10] Tao WQ. Numerical heat transfer. 2nd ed. Xi'an: Xi'an Jiaotong University Press; 2001.
[11] Tryggvason G, Scardovelli R, Zaleski S. Direct numerical simulations of gas-liquid multiphase flows. New York: Cambridge University Press; 2011.
[12] Leonard BP. A stable and accurate convective modeling procedure based on quadratic upstream interpolation. Comput Methods Appl Mech Eng 1979;29:59–98.
[13] Francois MM, Cummins SJ, Dendy ED, Kothe DB, Sicilian JM, Williams MW. A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework. J Comput Phys 2006;213:141–73.
[14] Tao WQ. Recent advances in computational heat transfer. Beijing: Science Press; 2000.

[15] Van Der Vorst HA. BI-CGSTAB: a fast and smoothly converging variant of BICG for the solution of nonsymmetric linear systems. SIAM J Sci Stat Comput 1992;13:631–44.
[16] LeVeque R. High-resolution conservative algorithms for advection in incompressible flow. SIAM J Numer Anal 1996;33:627–65.
[17] Wang ZY, Yang JM, Koo B, Stern F. A coupled level set and volume-of-fluid method for sharp interface simulation of plunging breaking waves. Int J Multiphase Flow 2009;35:227–46.
[18] Menard T, Tanguy S, Berlemont A. Coupling level set/VOF/ghost fluid methods: validation and application to 3D simulation of the primary break-up of a liquid jet. Int J Multiphase Flow 2007;33:510–24.
[19] Martin JC, Moyce WJ. An experimental study of the collapse of liquid columns on a rigid horizontal plane. Philos Trans Roy Soc Lond Ser A 1952;224:312–24.
[20] Grace JR. Shapes and velocities of bubbles rising in infinite liquids. Trans Inst Chem Eng 1973;51:116–20.
[21] Bhaga D, Weber ME. Bubbles in viscous liquids: shapes, wakes and velocities. J Fluid Mech 1981;105:61–85.
[22] Hua J, Stene JF, Lin P. Numerical simulation of 3D bubbles rising in viscous liquids using a front tracking method. J Comput Phys 2008;227:3358–82.
[23] Amaya-Bower L, Lee T. Single bubble rising dynamics for moderate Reynolds number using lattice Boltzmann method. Comput Fluids 2010;39:1191–207.