A simple volume-of-fluid reconstruction method for three-dimensional two-phase flows

Akio Kawano*

Japan Agency for Marine-Earth Science and Technology, 3173-25, Showa-machi, Kanazawa-ku, Yokohama, 236-0001, Japan

Abstract

A new PLIC (piecewise linear interface calculation)-type VOF (volume of fluid) method, called APPLIC (approximated PLIC) method, is presented. Although the PLIC method is one of the most accurate VOF methods, the three-dimensional algorithm is complex. Accordingly, it is hard to develop and maintain the computational code. The APPLIC method reduces the complexity using simple approximation formulae. Three numerical tests were performed to compare the accuracy of the SVOF (simplified volume of fluid), VOF/WLIC (weighed line interface calculation), THINC/SW (tangent of hyperbola for interface capturing/slope weighting), THINC/WLIC, PLIC, and APPLIC methods. The results of the tests show that the APPLIC results are as accurate as the PLIC results and are more accurate than the SVOF, VOF/WLIC, THINC/SW, and THINC/WLIC results. It was demonstrated that the APPLIC method is more computationally efficient than the PLIC method.

Keywords: Free interface, VOF method, PLIC method, two-phase flows

1. Introduction

Two-phase flows are essential in many research fields; for example, in relation to cloud and precipitation droplets in the atmosphere, water waves, cooling devices, oil and gas pipelines, chemical industrial plants, and thermal power stations. In the recent decades, many interface tracking methods for simulating two-phase flows have been developed. The VOF (volume of fluid) method, originated by Hirt and Nichols [1], is one of the most widely used algorithms. Excellent reviews of the VOF method have been given by Rudman [2], Rider and Kothe [3], Scardovelli and Zaleski [4], and Pilliod and Puckett [5].

The VOF method is based on the spatial discretization of a characteristic function to distinguish between two phases, and the reconstruction of the interfaces for advection. Suppose that we wish to simulate an incompressible two-phase (‘light’ and ‘dark’) flow in the three-dimensional Cartesian space \( x = (x_1, x_2, x_3) \). The characteristic function for the flow is defined as

\[
\chi(x) = \begin{cases} 
0 & \text{if there is light fluid at point } x, \\
1 & \text{if there is dark fluid at point } x.
\end{cases}
\]  

(1)

The interfaces between the two phases are represented by the discontinuity of the characteristic function. In this paper, we suppose that a computational grid composed of cubic cells of a edge \( \Delta x \) is used. Extension of our analysis to general regular grids is straightforward. By discretizing the characteristic function in a computational cell \( (i, j, k) \), we can obtain the volume fraction

\[
C_{i,j,k} = \frac{1}{(\Delta x)^3} \int_{\Omega_{i,j,k}} \chi(x) \, dx,
\]  

(2)
where $\Omega_{i,j,k}$ is the domain of the cell. It is obvious from the definition that

$$C_{i,j,k} = \begin{cases} 
0 & \text{if the cell is filled by light fluid,} \\
(0, 1) & \text{if the cell contains both fluids (interface cell),} \\
1 & \text{if the cell is filled by dark fluid.} 
\end{cases}$$

The VOF method reconstructs the shape of the interface in each interface cell to evaluate VOF advection fluxes. Various schemes for VOF reconstruction have been presented. The PLIC (piecewise linear interface calculation) method \[6\] \[7\] reconstructs an interface in a cell as a plane (in three-dimensional space) or a line (in two-dimensional space) with a given normal vector. The SLIC (simple line interface calculation) method \[8\] assumes the shape of an interface to be a plane parallel to one of the cell faces. The VOF/WLIC (weighted line interface calculation) method \[9\] evaluates an advection flux through a cell face as a weighted sum of SLIC fluxes. The SVOF (simplified volume of fluid) method \[10\] is similar to the VOF/WLIC method, except for the weight formula. In the THINC (tangent of hyperbola for interface capturing) method \[11\], interfaces are represented by the use of the hyperbolic tangent. Improved THINC methods have also been proposed \[9\] \[12\] \[13\].

Although it is known to be one of the most accurate reconstruction methods, a three-dimensional implementation of the PLIC method is a troublesome task. The PLIC method requires the solution of two geometric problems, as to a cut-volume of a cube by a plane, which are very complicated especially in three-dimensional cases. Scardovelli and Zaleski have provided two sophisticated algorithms (hereafter called the SZ algorithms) to solve these problems \[14\]. Although the SZ algorithms make the implementation of the PLIC method easier because of their compactness, these are still too complex for quick and easy implementation. Computational routines that implement the SZ algorithms must involve multiple “if” statements, which make it hard to develop and maintain the routines, and potentially inhibit its optimal compilation, especially for processors susceptible to conditional branches, e.g., deeply pipelined processors, processors with SIMD (single instruction multiple data) operations, vector processors, and GPUs (graphics processing units) \[15\].

In this paper, a PLIC-type VOF method called the APPLIC (approximated PLIC) method is presented. In the APPLIC method, interfaces are reconstructed in a similar manner as in the PLIC method, except that the geometric problems are solved through the use of simple approximation formulae.

This paper is organized as follows. In Section 2, we describe the APPLIC method. Section 3 compares the accuracy and computational efficiency of the APPLIC method with some existing VOF methods. Finally, conclusions are summarized in Section 4.

The following vector notation is used throughout this paper. Bold letters denote three-dimensional vectors and the corresponding non-bold letters with subscripts 1, 2, or 3 denote the vector components. For example, $\mathbf{u} = (u_1, u_2, u_3)$ and $\mathbf{m}_A'' = (m_{A,1}'', m_{A,2}'', m_{A,3}'')$. We use $\|\mathbf{m}\|_n$ to represent the $n$-norm of $\mathbf{m}$; namely, $\|\mathbf{m}\|_1 = |m_1| + |m_2| + |m_3|$ and $\|\mathbf{m}\|_2 = (m_1^2 + m_2^2 + m_3^2)^{1/2}$. The expression $\mathbf{m} \geq a$ stands for the condition $m_1 \geq a$, $m_2 \geq a$, and $m_3 \geq a$.

2. Method

2.1. The PLIC method using the SZ algorithms

In this paper, we use directional splitting for advection and a regular staggered grid where velocity components $u_1$, $u_2$, and $u_3$ are stored at the centers of the cell faces $\{(i + 1/2, j, k)\}$, $\{(i, j + 1/2, k)\}$, and $\{(i, j, k + 1/2)\}$, respectively. We assume that the Courant-Friedrichs-Lewy (CFL) condition,

$$\frac{|u_l| \Delta t}{\Delta x} < 1 \quad \text{for all } l \in \{1, 2, 3\},$$

holds, where $\Delta t$ is the time step size.

Let $\phi$ be the face, and $u_l$ ($l = 1, 2, 3$) the velocity component placed on $\phi$. Let $\Omega$ be the donor cell, which is the cell that has $\phi$ as a cell face and lies on the upwind side of $u_l$. Let $\phi^*$ be the opposite face of the face $\phi$ in the cell $\Omega$. The cell $\Omega$ is partitioned into two subcells by the section $\sigma$ parallel to $\phi$ and laid $|u_l| \Delta t$...
away from $\phi$. Let $\Omega_A$ and $\Omega_B$ be the subcells of $\Omega$ between $\phi$ and $\sigma$ and between $\sigma$ and $\phi^*$, respectively (see Fig. 1). The section $\sigma$ is always located between $\phi$ and $\phi^*$ because of the CFL condition. Let $C_A$ and $C_B$ be the partial volume fractions in $\Omega_A$ and $\Omega_B$, respectively, defined as

$$C_A = \frac{1}{(\Delta x)^3} \int_{\Omega_A} \chi(x) \, dx,$$

(5)

$$C_B = \frac{1}{(\Delta x)^3} \int_{\Omega_B} \chi(x) \, dx.$$

(6)

It is obvious that $C_A \in [0, 1]$ and $C_B \in [0, 1]$. From the definitions, we have

$$C_A + C_B = C,$$

(7)

where $C$ is the volume fraction in the donor cell $\Omega$.

![Figure 1: Schematic illustration of a donor cell with respect to a flux through a cell face $\phi$. The cell is divided into two subcells, $\Omega_A$ and $\Omega_B$, by the section $\sigma$ parallel to $\phi$ and laid $|u_I|\Delta t$ away from $\phi$.](image)

The computational advection flux of the volume fraction through the face $\phi$ (i.e., the amount of the volume fraction through the face during $\Delta t$) is obtained via

$$F = C_A \, \text{sgn} \, u_I,$$

(8)

where sgn is the sign function defined as

$$\text{sgn} \, x = \begin{cases} 1 & \text{if } x \geq 0, \\ -1 & \text{if } x < 0. \end{cases}$$

(9)

In some cases, $C_A$ is easily determined by

$$C_A = \begin{cases} 0 & \text{if } C = 0 \text{ or } |g| = 0, \\ |g| & \text{if } C = 1, \end{cases}$$

(10)

where $|g|$ denote the local CFL number in the cell $\Omega$ with respect to the flux through the face $\phi$:

$$g = \frac{u_I \Delta t}{\Delta x}.$$
Because of the CFL condition, \( g \) must be in the range \((-1, 1)\). If \( C \in (0, 1) \) and \( |g| > 0 \), \( CA \) is determined through the reconstruction of the interface in the donor cell \( \Omega \).

Here, we define the two geometric problems crucial to the PLIC method, which are mutually inverse. Consider a unit cube \( U = \{ x \in [0, 1]^3 \} \) and an oriented plane \( P(\alpha, m) = \{ x | m \cdot x < \alpha \} \), where \( m \) is the normal vector of the plane, and \( \alpha \) is the plane constant. Note that an oriented plane is not a thin object without volume, but is a solid object with an inside and an outside. Let \( V \) be the volume of the intersection between the unit cube and the oriented plane. One of the problems, called the forward problem, is to determine the value of \( V \) for given \( \alpha \) and \( m \). The other problem, called the inverse problem, is to determine the value of \( \alpha \) for given \( V \) and \( m \). Namely,

\[
V(\alpha, m) = \int_{U \cap P(\alpha, m)} dx, \tag{12}
\]

\[
\alpha(V', m) = \alpha' \quad \text{such that} \quad V(\alpha', m) = V'. \tag{13}
\]

To reduce the complexity of the problems, the SZ algorithms restrict \( m \) to a vector so that \( m \geq 0 \) and \( \|m\|_1 = 1 \).

The functions \( V(\alpha, m) \) and \( \alpha(V, m) \) have the following properties [4].

(I) The value of \( V(\alpha, m) \) is within the range \([0, 1]\) and

\[
V(\alpha, m) = \begin{cases} 
0 & \text{if } \alpha \leq 0, \\
1 & \text{if } \alpha \geq 1.
\end{cases} \tag{14}
\]

(II) The value of \( \alpha(V, m) \) is within the range \([0, 1]\) for \( V \in [0, 1] \).

(III) The functions \( V \) and \( \alpha \) are invariant with respect to a permutation of \( m_1, m_2, \) and \( m_3 \).

(IV) The functions \( V \) and \( \alpha \) are continuous, one-to-one, and monotonically increasing functions of \( \alpha \) and \( V \) in the ranges \( \alpha \in [0, 1] \) and \( V \in [0, 1] \), respectively.

(V) The first derivatives \( \partial V/\partial \alpha \) and \( \partial \alpha/\partial V \) are continuous and monotonically nondecreasing functions of \( \alpha \in (0, 1/2] \) and \( V \in (0, 1/2] \), respectively.

(VI) The curve \((\alpha, V)\) passes through the points \((0, 0)\), \((1/2, 1/2)\), and \((1, 1)\).

(VII) If \( m = (0, 0, 1) \) and \((0, 1/2, 1/2)\), the values of \( V(\alpha, m) \) become \( \alpha \) and \((2\alpha)^2/2\), respectively, for \( \alpha \in [0, 1/2] \).

(VIII) The curve \((\alpha, V)\) has point symmetry (or odd symmetry) with respect to \((\alpha, V) = (1/2, 1/2)\); namely,

\[
V(\alpha, m) = 1 - V(1 - \alpha, m), \tag{15}
\]

\[
\alpha(V, m) = 1 - \alpha(1 - V, m). \tag{16}
\]

All the properties except (VII) hold for the arbitrary \( m \).

Figure 2 shows an example implementation of the SZ algorithms written in Fortran 90. The cbart function, used in line 44, is an intrinsic function that returns the real cube root of the argument. Although this is not included in the Fortran 90 standard, many Fortran compilers support this. The abs functions in lines 11 and 40 are used to prevent \( \text{vm2} \) being negative owing to the numerical error in floating point arithmetic when \( \text{vm3} \approx 1 \).
module constants

real(8), parameter :: CONST_TINY = 1d-25 ! an extremely small constant
real(8), parameter :: CONST_PI = 3.14159265358979323846d0 / \pi 
end module constants

function calc_v(alpha, vma, vmb, vmc) result(v)
    ! Preconditions: vma \in [0,1], vmb \in [0,1], vmc \in [0,1], vma + vmb + vmc = 1.
    use constants
    real(8), intent(in) :: alpha, vma, vmb, vmc

    v = 0d0
    if (a > 0d0) then
        vm1 = min(vma, vmb, vmc) ! Sort vm1, vm2, vm3
        vm3 = max(vma, vmb, vmc) ! such that vm1 \leq vm2 \leq vm3.
        vm2 = abs(1d0 - vm3 - vm1)
        vm12 = vm1 + vm2
        if (a < vm1) then
            v = a ** 3 / (6d0 * vm1 * vm2 * vm3)
        else if (a < vm2) then
            v = a * (a - vm1) / (2d0 * vm2 * vm3) + vm1 ** 2 / (6d0 * vm2 * vm3 + CONST_TINY)
        else if (a < min(vm12, vm3)) then
            v = (a ** 2 * (3d0 - 2d0 * a) + vm1 ** 2 * (vm1 - 3d0 * a) + vm2 ** 2 * (vm2 - 3d0 * a)) / (6d0 * vm1 * vm2 * vm3)
        else
            v = (a - 0.5d0 * vm12) / vm3
        end if
    else if (a < vm1) then
        v = a ** 3 / (6d0 * vm1 * vm2 * vm3)
    else if (a < vm2) then
        v = a * (a - vm1) / (2d0 * vm2 * vm3) + vm1 ** 2 / (6d0 * vm2 * vm3 + CONST_TINY)
    else if (a < min(vm12, vm3)) then
        v = (a ** 2 * (3d0 - 2d0 * a) + vm1 ** 2 * (vm1 - 3d0 * a) + vm2 ** 2 * (vm2 - 3d0 * a)) / (6d0 * vm1 * vm2 * vm3)
    else
        v = (a - 0.5d0 * vm12) / vm3
    end if
    if (alpha > 0.5d0) v = 1d0 - v
end function calc_v

function calc_alpha(v, vma, vmb, vmc) result(alpha)
    ! Preconditions: v \in (0,1), vma \in [0,1], vmb \in [0,1], vmc \in [0,1], vma + vmb + vmc = 1.
    use constants
    real(8), intent(in) :: v, vma, vmb, vmc
    real(8) :: alpha, w, vm1, vm2, vm3, vm12, v1, v3, a0, a1, a2, q0, sp, th

    w = min(v, 1d0 - v)
    vm1 = min(vma, vmb, vmc) ! Sort vm1, vm2, vm3
    vm3 = max(vma, vmb, vmc) ! such that vm1 \leq vm2 \leq vm3.
    vm2 = abs(1d0 - vm3 - vm1)
    vm12 = vm1 + vm2
    v1 = vm1 ** 2 / (6d0 * vm2 * vm3 + CONST_TINY)
    if (w < v1) then
        alpha = cbrt(6d0 * vm1 * vm2 * vm3 * w)
    else if (w < v1 + (vm2 - vm1) / (2.0 * vm3)) then
        alpha = 0.5d0 * (vm1 + sqrt(vm1 ** 2 + 8d0 * vm2 * vm3 * (w - v1)))
    else
        alpha = 0d0
        if (vm3 < vm12) then
            v3 = (vm3 ** 2 * (3d0 - 2d0 * vm3 - vm1) + vm1 ** 2 * (vm1 - 3d0 * vm3)) / (6d0 * vm1 * vm2 * vm3)
        else
            v3 = 0.5d0 * vm12 / vm3
        end if
        if (v3 <= w) alpha = vm3 * w + 0.5d0 * vm12
    end if
    if (alpha == 0.0) then
        if (w < v3) then
            a2 = -3d0 * vm1
            a1 = 3d0 * (vm1 ** 2 + vm2 ** 2)
            a0 = -(vm1 ** 3 + vm2 ** 3 - 6d0 * vm1 * vm2 * vm3 * w)
        else
            a2 = -1.5d0
            a1 = 1.5d0 * (vm1 ** 2 + vm2 ** 2 + vm3 ** 2)
            a0 = -0.5d0 * (vm1 ** 3 + vm2 ** 3 + vm3 ** 3 - 6d0 * vm1 * vm2 * vm3 * w)
        end if
        q0 = (1d0/6d0) * (a1 + a2 * 3d0 * a0) - a2 ** 3 * (1d0/27d0)
        sp = sqrt((1d0/3d0) * (a1 + a2 * 3d0 * a0) + a2 ** 2)
        th = (1d0/3d0) * acos(q0 / (sp ** 3))
        alpha = 2d0 * sp * cos(th + (4d0/3d0 * CONST_PI) - (1d0/3d0) * a2)
    end if
end function calc_alpha

Figure 2: Example implementation of the SZ algorithms written in Fortran 90.
By using the SZ algorithms to evaluate the functions $V(\alpha, \mathbf{m})$ and $\alpha(V, \mathbf{m})$, the volume fraction $C_A$ for the PLIC method is determined as

$$C_A(g, C, \mathbf{n}) = |g| V(\alpha_A''', m_A'''), \quad (17a)$$

with

$$\alpha_A''' = \begin{cases} Q_A' \alpha' & \text{if } n_I g \leq 0, \\ Q_A' (\alpha' - r_A) & \text{if } n_I g > 0, \end{cases} \quad (17b)$$

$$m_A''' = \begin{cases} Q_A' m_l'' |g| & \text{if } l = I, \\ Q_A' m_l'' & \text{if } l \neq I, \end{cases} \quad (17c)$$

$$Q_A' = \frac{1}{1 - r_A}, \quad (17d)$$

$$r_A = m_l'' (1 - |g|), \quad (17e)$$

$$\alpha' = \alpha(C, m'), \quad (17f)$$

$$m_l'' = \frac{|n_l|}{\|n\|_1}, \quad (17g)$$

where $l$ is an index running from one to three, and $\mathbf{n}$ is the normal vector of the interface in the donor cell $\Omega$ oriented from the dark fluid to the light fluid. See Appendix A for the derivation of the equations. The vector $\mathbf{n}$ is determined as

$$\mathbf{n} = -\nabla C, \quad (18)$$

where $\nabla C$ is a numerical gradient of the volume fraction. Accurate evaluation of numerical gradients is required for accurate results. Various algorithms to evaluate numerical gradients for the VOF method can be found in the literature [5, 16–20].

Figure 3 shows an example implementation of the PLIC method written in Fortran 90. Here, the argument vn1 corresponds to $n_I$, and vn2 and vn3 correspond to the other components of $\mathbf{n}$.

2.2. Approximation of the forward and the inverse problems

A basic idea of the APPLIC method is to evaluate $V(\alpha, \mathbf{m})$ and $\alpha(V, \mathbf{m})$ by use of simple approximation formulae. In the APPLIC method, the function $V(\alpha, \mathbf{m})$ for $\alpha \in [0, 1/2]$ is approximated as

$$\tilde{V}(\alpha, \mathbf{m}) = \frac{1}{2} (2\alpha)^p, \quad (19)$$
where \( p \) is a positive-valued function of \( m \). The approximation of the function \( \alpha(V, m) \) for \( V \in [0, 1/2] \) is derived by solving Eq. (19) for \( \alpha \):

\[
\tilde{\alpha}(V, m) = \frac{1}{2}(2V)^{1/p}.
\]

Here a tilde (\( \tilde{\phantom{\alpha}} \)) indicates an approximation. Using Eqs. (16) and (20), we can give the formula of \( \tilde{\alpha} \) for \( V \in [0, 1] \) as

\[
\tilde{\alpha}(V, m) = \begin{cases} 
\frac{1}{2}(2V)^{p} & \text{if } V \leq 1/2, \\
1 - \frac{1}{2}(2(1 - V))^{p} & \text{if } V > 1/2.
\end{cases}
\]

Similarly, using Eqs. (14), (15), and (19), we have

\[
\tilde{V}(\alpha, m) = \begin{cases} 
0 & \text{if } \alpha \leq 0, \\
\frac{1}{2}(2\alpha)^{p} & \text{if } 0 < \alpha \leq 1/2, \\
1 - \frac{1}{2}(2(1 - \alpha))^{p} & \text{if } 1/2 < \alpha < 1, \\
1 & \text{if } \alpha \geq 1.
\end{cases}
\]

The functions \( \tilde{V} \) and \( \tilde{\alpha} \) satisfy properties (I), (II), (IV), (V), (VI), and (VIII).

Let us formulate the function \( p \). First, we determine the optimal \( p \), denoted by \( p_{\text{opt}} \), that minimizes the square error \( D \), defined as

\[
D(m) = \int_{0}^{1/2} [\tilde{V}(\alpha, m) - V(\alpha, m)]^2 d\alpha + \int_{1/2}^{1} [\tilde{\alpha}(V, m) - \alpha(V, m)]^2 dV.
\]

The function \( p_{\text{opt}} \) is plotted in Fig. 4.

Figure 4: Plot for \( p_{\text{opt}} \) as a function of \( m \). The component \( m_3 \) is given by \( 1 - m_1 - m_2 \). The contour lines are drawn at \( p_{\text{opt}} = 1, 1.2, 1.4, \ldots, 2.6 \).
Next, we construct an arithmetic expression of \( p \) that approximates \( p_{\text{opt}} \). We found that \( p_{\text{opt}} \) can be approximated as

\[
p = \frac{c_2 \xi^2 + c_1 \xi + c_0}{\xi + c_0},
\]

with

\[
\xi = (b - m_1)(b - m_2)(b - m_3) - a,
\]

where \( c_0, c_1, c_2, a, \) and \( b \) are constants. To satisfy property (III), Eq. (25) is designed to be symmetric with respect to \( m_1, m_2, \) and \( m_3 \). We impose the following relations on the constants to satisfy property (VII):

\[
c_0 = \frac{b}{16}(c_2 b + 4 c_1 - 8),
\]

\[
a = b^2(b - 1).
\]

The optimal values of \( b, c_1, \) and \( c_2 \), shown in Table 1, were determined by a least-square procedure that minimizes the mean square error \( \bar{D} \) with respect to \( m \), defined as

\[
\bar{D} = \frac{\iint_{S_1} D \left( \frac{n}{\|n\|_1} \right) dS}{\iint_{S_1} dS},
\]

where \( S_1 \) is the part of the unit spherical surface in the first octant, namely,

\[
S_1 = \{ x \mid x \geq 0 \text{ and } \| x \|_2 = 1 \},
\]

\( n \) is a vector that scans \( S_1 \), and \( dS \) is the surface element. Figure 5 shows \( p \) given by Eq. (24) with the optimal \( b, c_1, \) and \( c_2 \). Here, \( p_{\text{opt}} \) is also plotted for comparison, denoted by white and dotted contour lines. The function \( p \) given by Eq. (24) fits with \( p_{\text{opt}} \) extremely well.

| Table 1: Optimal values of the constants |
|---|
| \( b \) | 1.49 |
| \( c_1 \) | 0.132 |
| \( c_2 \) | 0.239 |
Figure 5: Plot for $p$ (the color image and the black solid contour lines), as obtained by Eq. (24) as a function of $m$. The component $m_3$ is given by $1 - m_1 - m_2$. The contour lines for $p_{\text{opt}}$ are also shown as white dotted lines for comparison. The contour lines are drawn at $p = 1, 1.2, 1.4, \ldots, 2.6$.

The square error $D$ for $p$ obtained by Eq. (24) is plotted in Fig. 6. This figure shows that $D$ becomes the maximum, $D_{\text{max}} = 2.70 \times 10^{-4}$, at $m = (0.734, 0.133, 0.133)$, $(0.133, 0.734, 0.133)$, and $(0.133, 0.133, 0.734)$, and becomes zero at $m = (1, 0, 0)$, $(0, 1, 0)$, $(0, 0, 1)$, $(0, 1/2, 1/2)$, $(1/2, 0, 1/2)$, and $(1/2, 1/2, 0)$. In Fig. 7, we show $V$ and $\tilde{V}$ as functions of $\alpha$ for various orientations of $m$. In each pane, curves $V(\alpha, m)$ and $\tilde{V}(\alpha, m)$ are plotted for a specific $m$. This figure shows that the discrepancy between $V$ and $\tilde{V}$ is sufficiently small for any $m$. Panel (q) indicates the case of $m = (0.125, 0.125, 0.75)$, which has the largest square error ($D = 2.68 \times 10^{-4}$) among the cases plotted in Fig. 7. Note that $(0.125, 0.125, 0.75)$ is close to $(0.133, 0.133, 0.734)$, a case with the maximum square error.
Figure 6: Plot for the square error defined as Eq. (23) as a function of $m$. The component $m_3$ is given by $1 - m_1 - m_2$. The contour lines are drawn at $D = 0.5 \times 10^{-4}, 1.0 \times 10^{-4}, 1.5 \times 10^{-4}, \ldots, 2.5 \times 10^{-4}$.
Figure 7: Comparison between the functions $V(\alpha, m)$ (dashed lines) and $\tilde{V}(\alpha, m)$ (solid lines) as functions of $\alpha$ for various $m$. 

$V_{\alpha, m}$
2.3. The crude APPLIC method

A are mathematically equivalent if both \( A_1 \) and 24, the expression \( \exp(\log(A) \times B) \) is used instead of the expression \( A \times B \). Both the expressions are mathematically equivalent if both \( A \) and \( B \) are positive. Note that the variables \( a, w, p, \) and \( \text{invp} \) in Fig. 8 are always positive. However, the former expression requires slightly less computational cost than the latter expression in most computer environments because the evaluation of \( A \times B \) internally involves multiple conditional branches, e.g., if \( A \) is positive/zero/negative, if \( B \) is positive/zero/negative, and if \( B \) is an integer or not. If C/C++ is used, the expression \( \exp(\log(A) \times B) \) may slightly more efficient.

\[
\text{Figure 8: Example implementation of the approximation functions } \tilde{V} \text{ and } \tilde{\alpha} \text{ written in Fortran 90.}
\]

\[
2.3. \text{The crude APPLIC method}
\]

Using the approximation functions \( \tilde{V}(\alpha, m) \) and \( \tilde{\alpha}(V, m) \) instead of functions \( V(\alpha, m) \) and \( \alpha(V, m) \) in Eq. (17), we can determine computational advection fluxes via Eq. (8). This straightforward method is called the crude APPLIC method.

The crude APPLIC method is not practical because of defects described below. Let us examine whether fluxes evaluated by the crude APPLIC method have properties which are essential to be satisfied by fluxes of volume fractions. The computational advection flux \( F \) should satisfy the following conditions:

\[
F(g, C, n) = \begin{cases} 
  \geq 0 & \text{if } g > 0, \\
  = 0 & \text{if } g = 0, \\
  \leq 0 & \text{if } g < 0, 
\end{cases}
\]  

(30)

\[
F[g, C, (n_1, n_2, n_3)] = s_1 F[s_1 g, C, (s_1 n_1, s_2 n_2, s_3 n_3)],
\]  

(31)

\[
F[g, C, (n_1, n_2, n_3)] = F[g, C, (n_1, n_3, n_2)],
\]  

(32)

\[
g = F(g, C, n) + F(g, 1 - C, -n),
\]  

(33)

\[
|F[g, C, n]| \leq C,
\]  

(34)

where \( s_i \) in Eq. (31) is either 1 or \(-1\). In Eq. (32) we suppose that \( I = 1 \) for simplicity. Condition (30) specifies the sign of \( F \). Condition (31) stems from the symmetry of positive and negative directions along
the coordinate axes. Conditions (32) stems from the permutation symmetry between the second- and the third-coordinate axes. As shown in (a) and (b) of Fig. 9, the second term in the right-hand side of Eq. (33) corresponds to the flux of the light fluid. Therefore, condition (33) means that the total flux, given by $g$, is the sum of the light fluid flux and the dark fluid flux. Condition (34) provides the upper limit of $|F|$ under the CFL condition.

A lower limit of $|F|$ is derived from Eqs. (30), (33) and (34) as follows. From Eqs. (30) and (33), we have

$$|F(g,C,n)| + |F(g,1-C,n)| = |g|,$$

$$|F(g,1-C,n)| \leq 1-C.$$  

(35)

These leads

$$|F(g,C,n)| \geq |g| - (1-C).$$

(36)

![Figure 9: Two-dimensional schematic of the relation among partial volume fractions in a donor cell. The sides of the cells are scaled to be unity.](image)

Although conditions (30), (31), (32), and (33) are always satisfied by the crude APPLIC method, conditions (34) and (36) are not. To demonstrate this, a set of sample points, $S = \{(g^{(l)}, C^{(l)}, n^{(l)}) | l = 1, 2, \ldots, N\}$ ($N = 10,000,000$), is used. The sample points in the set $S$ were generated such that $\{g^{(l)}\}$, $\{C^{(l)}\}$, and $\{n^{(l)}\}$ are uniformly distributed on $[0, 1]$, $[0, 1]$, and the unit spherical surface, respectively, by use of pseudorandom numbers. Among the sample points in the set $S$, 5.4% of the points do not satisfy the condition $|g| - (1-C) \leq |\tilde{F}_{\text{APPLIC}}| \leq C$, where $\tilde{F}_{\text{APPLIC}}$ is a flux obtained by the crude APPLIC method. An easy remedy for the defect is to adopt the limiter as follows:

$$\tilde{F}_{\text{APPLIC/L}} = \min\{C, \max\{|\tilde{F}_{\text{APPLIC}}|, |g| - (1-C)\}\} \sgn g,$$

(37)

where $\tilde{F}_{\text{APPLIC/L}}$ is a flux obtained by the crude APPLIC method with the limiter.
2.4. The APPLIC method

Consider the following relation:

\[ \tilde{C}_A + \tilde{C}_B = C, \] (38)

where \( \tilde{C}_B \) is obtained by

\[ \tilde{C}_B(g, C, n) = \tilde{C}_A(1 - |g|) \text{sgn} \ g, \ C, -n. \] (39)

See (a) and (c) of Fig. 9 for a geometric interpretation of Eq. (39). Equation (38) is identical to Eq. (7) except that \( C_A \) and \( C_B \) are obtained by use of the approximation functions \( \tilde{V} \) and \( \tilde{\alpha} \). Generally, Eq. (38) does not hold because of approximation errors in \( \tilde{V} \) and \( \tilde{\alpha} \).

To improve the crude APPLIC method, we take advantage of the defect that Eq. (38) does not hold. There are two ways to calculate \( F \) by use of \( \tilde{V} \) and \( \tilde{\alpha} \):

\[ \tilde{F}_A(g, C, n) = \tilde{C}_A(g, C, n) \text{sgn} \ g, \] (40)

\[ \tilde{F}_B(g, C, n) = [C - \tilde{C}_B(g, C, n)] \text{sgn} \ g. \] (41)

In general, \( \tilde{F}_A \) and \( \tilde{F}_B \) are close but not equal. The crude APPLIC method uses only Eq. (40) to evaluate flux (i.e., \( \tilde{F}_{\text{APPLIC}} \) is identical to \( \tilde{F}_A \)), whereas the APPLIC method uses either \( \tilde{F}_A \) or \( \tilde{F}_B \) as follows:

\[ \tilde{F}_{\text{APPLIC}}(g, C, n) = \begin{cases} \tilde{F}_A(g, C, n) & \text{if} \ pred(g, C, n) \text{ is true}, \\ \tilde{F}_B(g, C, n) & \text{otherwise} \end{cases} \] (42)

where \( \text{pred} \) is a choice criterion, which is a logical (or Boolean-valued) function that returns either a true or false value. The ideal (i.e., impractical) criterion returns true if \( |\tilde{\alpha}_{A}'' - 1/2| < |\tilde{\alpha}_{B}'' - 1/2| \) and false otherwise, where \( F_{\text{PLIC}} \) denotes the flux obtained by the PLIC method.

Using Eq. (17), we have

\[ \tilde{C}_A(g, C, n) = |g| \tilde{V}(\tilde{\alpha}_A'', \ m_{A}'), \] (43a)

\[ \tilde{C}_B(g, C, n) = (1 - |g|) \tilde{V}(\tilde{\alpha}_B'', \ m_{B}'), \] (43b)

with

\[ \tilde{\alpha}_A'' = \begin{cases} Q_A' \tilde{\alpha}' & \text{if} \ n_I \ g \leq 0, \\ Q_A' (\tilde{\alpha}' - r_A) & \text{if} \ n_I \ g > 0, \end{cases} \] (43c)

\[ \tilde{\alpha}_B'' = \begin{cases} Q_B' (\tilde{\alpha}' - r_B) & \text{if} \ n_I \ g \leq 0, \\ Q_B' \tilde{\alpha}' & \text{if} \ n_I \ g > 0, \end{cases} \] (43d)

\[ m_{B,l}' = \begin{cases} Q_B' m_l' (1 - |g|) & \text{if} \ l = I, \\ Q_B' m_l' & \text{if} \ l \neq I, \end{cases} \] (43e)

\[ Q_B' = \frac{1}{1 - r_B}, \] (43f)

\[ r_B = m_l' |g|, \] (43g)

\[ \tilde{\alpha}' = \tilde{\alpha}(C, m). \] (43h)

The author proposes the following choice criterion for the APPLIC method: true if \( |\tilde{\alpha}_A'' - 1/2| > |\tilde{\alpha}_B'' - 1/2| \) and false otherwise. See Appendix B for the derivation of the choice criterion. Figure 10 shows an example implementation of the APPLIC method with the proposed criterion written in Fortran 90.
function calc_flux_applic(g, c, vn1, vn2, vn3) result(f)
!
use constants
real(8), intent(in) :: g, c, vn1, vn2, vn3

real(8) :: f, absg, alpha, alphb, ra, rb, qa, qb, vm1, vm2, vm3, v, sw

absg = abs(g)
vm1 = abs(vn1)
vm2 = abs(vn2)
vm3 = abs(vn3) + CONST_TINY
qa = 1d0 / (vm1 + vm2 + vm3)

ra = vm1 * qa
vm1 = vm1 * qa
vm2 = vm2 * qa
vm3 = vm3 * qa
alpha = calc_approx_alpha(c, vm1, vm2, vm3)
rb = vm1 * absg

ra = vm1 - rb
qa = 1d0 / (1d0 - ra)

qb = 1d0 / (1d0 - rb)

sw = 0d0
if(abs(alphb - 0.5d0) > abs(alpha - 0.5d0)) then
    sw = 1d0
    alpha = alphb
    qa = qb
    rb = ra
end if
v = calc_approx_v(alpha, rb * qa, vm2 * qa, vm3 * qa)
f = sign((c - v) * sw + v * absg, g)
end function

Figure 10: Example implementation of the APPLIC method with the proposed criterion written in Fortran 90.

Now the approximation accuracy of fluxes obtained by the APPLIC method is examined. Table 2 compares the statistics for the approximation errors of the crude APPLIC method, crude APPLIC method with Eq. (37) as a limiter, APPLIC method, and APPLIC method with the ideal criterion, for the point set $S$. Here, the approximation errors of a flux is defined as the difference between the flux and that obtained by the PLIC method. It is observed that applying the limiter to the crude APPLIC method reduces the mean error and the maximum error. The mean error of the APPLIC method is smaller than the crude APPLIC method with the limiter. The last row is for the APPLIC method with the ideal criterion, and indicates the lower limits for the mean errors and the maximum errors for any choice criteria.

| Method                      | Correct answer rate $^a$ | Mean error $^b$  | Maximum error $^c$ |
|-----------------------------|--------------------------|------------------|--------------------|
| Crude APPLIC                | 50.0%                    | $2.89 \times 10^{-3}$ | $3.32 \times 10^{-2}$ |
| Crude APPLIC with limiter   | —                        | $2.54 \times 10^{-3}$ | $2.77 \times 10^{-2}$ |
| APPLIC                      | 71.5%                    | $1.88 \times 10^{-3}$ | $2.77 \times 10^{-2}$ |
| APPLIC with ideal criterion | 100%                     | $1.43 \times 10^{-3}$ | $1.67 \times 10^{-2}$ |

$^a$The percentage of the sample points where the criterion provides true if $|F_A - F_{PLIC}| < |F_B - F_{PLIC}|$ and false otherwise.

$^b$The arithmetic mean of the absolute errors.

$^c$The maximum value of the absolute errors.

Figures 11 and 12 show the dependence of approximation errors of fluxes on $|g|$ and $C$. The errors in the figures were evaluated by using 50000 three-dimensional vectors distributed uniformly on the unit spherical surface as sample points of $n$. All the plots are axisymmetric with respect to $C = 0.5$ since the PLIC, crude APPLIC, crude APPLIC with limiter, APPLIC, and APPLIC with ideal criterion satisfy Eq. (33). Moreover, the plots for the APPLIC and the APPLIC with ideal criterion are axisymmetric with respect to $|g| = 0.5$ since the PLIC, APPLIC, and APPLIC with ideal criterion satisfy that $F_A$ equals to $F_B$. 

15
The APPLIC method satisfies conditions (34) and (36) as well as conditions (30)-(33). Moreover, the following condition tighter than conditions (30), (34), and (36) is also satisfied:

\[ F \in [LB, UB], \]  
(44)
with

\[
\text{LB} = \begin{cases} 
  gC & \text{if } n_I \leq 0, \\
  \max[g - (1 - C), 0] & \text{if } n_I > 0 \text{ and } g \geq 0, \\
  \max[g, -C] & \text{if } n_I > 0 \text{ and } g < 0,
\end{cases} 
\]  

(45)

\[
\text{UB} = \begin{cases} 
  gC & \text{if } n_I \geq 0, \\
  \min[g + (1 - C), 0] & \text{if } n_I < 0 \text{ and } g \leq 0, \\
  \min[g, C] & \text{if } n_I < 0 \text{ and } g > 0.
\end{cases} 
\]  

(46)

These bounds are derived from fluxes for SLIC-type fluid configurations [10].

We now demonstrate that the APPLIC method satisfies condition (44) using the set \( S \). Among the sample points in the set \( S \), 5.6% of the points do not satisfy the relation \( \tilde{F}_A \in [\text{LB}, \text{UB}] \). Similarly, 5.6% of the points do not satisfy \( \tilde{F}_B \in [\text{LB}, \text{UB}] \). However, all the points meet either \( \tilde{F}_A \in [\text{LB}, \text{UB}] \) or \( \tilde{F}_B \in [\text{LB}, \text{UB}] \).

Figure 13 depicts the points in the set \( S \) such that \( \tilde{F}_A \in [\text{LB}, \text{UB}] \) (blue dots) and \( \tilde{F}_B \notin [\text{LB}, \text{UB}] \) (red dots). Each dot is placed at \((|\tilde{\alpha}_A'' - 1/2|, |\tilde{\alpha}_B'' - 1/2|)\) on the plot. All the blue and the red dots lie in the regions \(|\tilde{\alpha}_A'' - 1/2| < |\tilde{\alpha}_B'' - 1/2|\) and \(|\tilde{\alpha}_A'' - 1/2| > |\tilde{\alpha}_B'' - 1/2|\), respectively. This indicates that fluxes evaluated by the APPLIC method with the proposed choice criterion always satisfy Eq. (44).

3. Numerical tests

3.1. The accuracy of advection

We compare the accuracy of advection for the APPLIC method with other VOF methods over three test problems. The VOF methods examined in this section are as follows: SVOF, VOF/WLIC, THINC/SW, THINC/WLIC, PLIC, and APPLIC. Each test problem is designed so that the initial distribution (at \( t = 0 \)) of the light and dark fluids is theoretically identical with the final distribution. The \( L_1 \) error, defined as

\[
\sum_{i,j,k} (\Delta x)^3 |C_{i,j,k}(t = 0) - C_{i,j,k}(t = T)|,
\]

is employed to compare the accuracy, where \( T \) is the final time.

Figure 13: The sample points in the set \( S \) so that \( \tilde{F}_A \notin [\text{LB}, \text{UB}] \) (blue dots) and \( \tilde{F}_B \notin [\text{LB}, \text{UB}] \) (red dots). The solid line represents the line \(|\tilde{\alpha}_A'' - 1/2| = |\tilde{\alpha}_B'' - 1/2|\).
Parker and Youngs’ method [16, 21] is adopted to evaluate surface normals. We adopt an operator splitting algorithm for the advection of volume fractions as follows:

\[ C_{i,j,k}^* = C_{i,j,k}^{(n)} - F_{i,j,k}^{(n)} + \frac{C_{i,j,k}^{(n)}}{\Delta x_1} (u_{1;i,j,k}^{1/2} - u_{1;i,j,k}^{-1/2}), \]  

\[ C_{i,j,k}^{**} = C_{i,j,k}^* - F_{i,j,k}^* + \frac{C_{i,j,k}^{(n)}}{\Delta x_2} (u_{2;i,j,k}^{1/2} - u_{2;i,j,k}^{-1/2}), \]  

\[ C_{i,j,k}^{(n+1)} = C_{i,j,k}^{**} - F_{i,j,k}^{**} + \frac{C_{i,j,k}^{(n)}}{\Delta x_3} (u_{3;i,j,k}^{1/2} - u_{3;i,j,k}^{-1/2}), \]  

(47a)  

(47b)  

(47c)

where the superscript \((n)\) refers to the temporal indices, and the superscripts \(*\) and \(**\) represent quantities at the first and second intermediate steps, respectively. The order of directions of Eq. (47) is changed at each time step to minimize possible asymmetries. All floating point arithmetic is done in double-precision.

3.1.1. Test 1

In the first test problem, a shape defined as the union of the rectangular parallelepipeds \(\{x \in [0.08, 0.48] \times [0.2, 0.36] \times [0.2, 0.36]\}\) and the sphere with center \((0.3, 0.28, 0.28)\) and radius 0.15 is translated in a computational domain \([0, 1] \times [0, 1] \times [0, 1]\). We set the final time as \(T = 0.8\). The shape \(B\) is advected in the following uniform velocity field:

\[ u = \begin{cases} 
(1, 1, 1) & \text{if } t < T/2, \\
(-1, -1, -1) & \text{if } t > T/2.
\end{cases} \]  

(48)

Figure 14 compares the initial shape and the final shapes using a 100×100×100 (\(\Delta x = 0.01\)) grid with \(\Delta t = 0.005\), where the CFL number is 0.5. The THINC/SW method produces significantly deformed shape. Sliced plots of volume fractions are shown in Fig. 15. This figure shows that the PLIC and the APPLIC methods keep the interface widths more compact than the SVOF, VOF/WLIC, THINC/SW, and THINC/WLIC methods.

Table 3 presents the \(L_1\) errors and the corresponding convergence rate for 25 × 25 × 25 (\(\Delta x = 0.04\)), 50 × 50 × 50 (\(\Delta x = 0.02\)), and 100 × 100 × 100 (\(\Delta x = 0.01\)) grids, where \(\Delta t = 0.02, 0.001,\) and 0.005, respectively. The errors of the PLIC and the APPLIC methods are the lowest and the second lowest for all cases.
Figure 14: The initial \((t = 0)\) and the final \((t = T)\) shapes for test 1, using a \(100 \times 100 \times 100\) grid with \(\Delta t = 0.005\).
Figure 15: Slice plots of the initial \((t = 0)\) and final \((t = T)\) volume fractions for test 1 at \(x_3 = 0.24\), using a 100 × 100 × 100 grid with \(\Delta t = 0.005\). Volume fractions are drawn by contour lines of 0.00001, 0.01, 0.5, 0.99, and 0.99999.

Table 3: \(L_1\) errors and convergence rates using different grid sizes for test 1.

| Method       | \(25 \times 25 \times 25\) Rate | \(50 \times 50 \times 50\) Rate | \(100 \times 100 \times 100\) Rate |
|--------------|---------------------------------|---------------------------------|----------------------------------|
| SVOF         | \(4.19 \times 10^{-3}\) 0.74    | \(2.50 \times 10^{-3}\) 1.03    | \(1.22 \times 10^{-3}\)         |
| VOF/WLIC     | \(3.77 \times 10^{-3}\) 0.84    | \(2.11 \times 10^{-3}\) 1.23    | \(8.99 \times 10^{-4}\)         |
| THINC/SW     | \(3.35 \times 10^{-3}\) 0.46    | \(2.44 \times 10^{-3}\) 1.03    | \(1.19 \times 10^{-3}\)         |
| THINC/WLIC   | \(4.57 \times 10^{-3}\) 0.86    | \(2.51 \times 10^{-3}\) 1.17    | \(1.12 \times 10^{-3}\)         |
| PLIC         | \(2.71 \times 10^{-3}\) 0.69    | \(1.68 \times 10^{-3}\) 1.15    | \(7.58 \times 10^{-4}\)         |
| APPLIC       | \(2.81 \times 10^{-3}\) 0.67    | \(1.77 \times 10^{-3}\) 1.17    | \(7.87 \times 10^{-4}\)         |

3.1.2. Test 2

Secondly, we examine a test problem proposed by Enright et al. [22], which is analogous with Zalesak’s disk problem in two-dimensional space [23]. A sphere of 0.16 radius with a slot of 0.04 wide and 0.2 deep is located initially at \((0.5, 0.72, 0.24)\) in a computational domain \([0, 1] \times [0, 1] \times [0, 0.48]\) and undergoes a rigid body rotation. The velocity field is static and is represented by

\[
    u_1 = (2\pi/T)(0.5 - x_2), \tag{49}
\]

\[
    u_2 = (2\pi/T)(x_1 - 0.5), \tag{50}
\]

\[
    u_3 = 0. \tag{51}
\]
We set the final time as \( T = 6 \). Figure 16 shows the evolution of the problem.

Figure 16: Evolution of test 2, calculated using the APPLIC method on a 200 \( \times \) 200 \( \times \) 96 grid with \( \Delta t = 0.005 \).

Figure 17 compares the initial (exact) shape and the final shapes using a 100 \( \times \) 100 \( \times \) 48 (\( \Delta x = 0.01 \)) grid and \( \Delta t = 0.01 \), where the maximum CFL number is approximately 0.5. The SVOF and VOF/WLIC methods produce significantly deformed shapes. Sliced plots of volume fractions are shown in Fig. 18. This figure shows that the PLIC and the APPLIC methods keep the interface widths more compact than the SVOF, VOF/WLIC, THINC/SW, and THINC/WLIC methods. Table 4 presents the \( L_1 \) errors and the corresponding convergence rates for 25 \( \times \) 25 \( \times \) 12 (\( \Delta x = 0.04 \)), 50 \( \times \) 50 \( \times \) 24 (\( \Delta x = 0.02 \)), and 100 \( \times \) 100 \( \times \) 48 (\( \Delta x = 0.01 \)) grids, where \( \Delta t = 0.04, 0.02, \) and 0.01, respectively. The errors of the APPLIC and the PLIC are the smallest and the second smallest for all cases.
Figure 17: The initial \((t = 0)\) and the final \((t = T)\) shapes for test 2, using a \(100\times100\times48\) grid with \(\Delta t = 0.01\).
Figure 18: Slice plots of the initial \((t = 0)\) and final \((t = T)\) volume fractions for test 2 at \(x_3 = 0.24\), using a \(100 \times 100 \times 48\) grid with \(\Delta t = 0.01\). Volume fractions are drawn by contour lines of 0.00001, 0.01, 0.5, 0.99, and 0.99999.

Table 4: \(L_1\) errors and convergence rates using different grid sizes for test 2.

| Method      | 25 \(\times\) 25 \(\times\) 12 Rate | 50 \(\times\) 50 \(\times\) 24 Rate | 100 \(\times\) 100 \(\times\) 48 Rate |
|-------------|-------------------------------------|-----------------------------------|-------------------------------------|
| SVOF        | \(5.25 \times 10^{-3}\) 0.63        | \(3.40 \times 10^{-3}\) 1.32      | \(1.36 \times 10^{-3}\) 1.36        |
| VOF/WLIC    | \(5.39 \times 10^{-3}\) 0.57        | \(3.63 \times 10^{-3}\) 1.35      | \(1.42 \times 10^{-3}\) 1.42        |
| THINC/SW    | \(4.42 \times 10^{-3}\) 0.67        | \(2.79 \times 10^{-3}\) 1.50      | \(9.84 \times 10^{-4}\) 9.84        |
| THINC/WLIC  | \(5.51 \times 10^{-3}\) 0.71        | \(3.36 \times 10^{-3}\) 1.30      | \(1.37 \times 10^{-3}\) 1.37        |
| PLIC        | \(4.00 \times 10^{-3}\) 1.07        | \(1.91 \times 10^{-3}\) 1.58      | \(6.39 \times 10^{-4}\) 6.39        |
| APPLIC      | \(3.99 \times 10^{-3}\) 0.98        | \(2.02 \times 10^{-3}\) 1.59      | \(6.73 \times 10^{-4}\) 6.73        |

3.1.3. Test 3

In the third test problem, a sphere of 0.15 radius centered at \((0.35, 0.35, 0.35)\) in a computational domain \([0, 1] \times [0, 1] \times [0, 1]\) is deformed in an incompressible flow field proposed by LeVeque [24], expressed by

\[
\begin{align*}
    u_1 &= 2\sin^2(\pi x_1)\sin(2\pi x_2)\sin(2\pi x_3)\cos(\pi t/T), \\
    u_2 &= -\sin(2\pi x_1)\sin^2(\pi x_2)\sin(2\pi x_3)\cos(\pi t/T), \\
    u_3 &= -\sin(2\pi x_1)\sin(2\pi x_2)\sin^2(\pi x_3)\cos(\pi t/T).
\end{align*}
\]

We set \(T = 3\). Figure 19 shows the evolution of the problem.
Figure 19: Evolution of test 3, calculated using the APPLIC method on a $200 \times 200 \times 200$ grid with $\Delta t = 0.00125$.

Figure 20 compares the initial (exact) shape and the final shapes using a $100 \times 100 \times 100$ ($\Delta x = 0.01$) grid with $\Delta t = 0.0025$, where the maximum CFL number is approximately 0.5. The SVOF and the VOF/WLIC methods produce significantly deformed shapes. Sliced plots of volume fractions are shown in Fig. 21. Artificial low-density particles and voids are generated by all the VOF methods. In particular, the THINC/SW and the THINC/WLIC methods generate many particles. The PLIC and the APPLIC methods keep the interface widths more compact than the SVOF, VOF/WLIC, THINC/SW, and THINC/WLIC methods. The shapes produced by the THINC/WLIC method have more prominent bumps than those by the PLIC and the APPLIC methods. Table 5 presents the $L_1$ errors and the corresponding convergence rates for $50 \times 50 \times 50$ ($\Delta x = 0.02$), $100 \times 100 \times 100$ ($\Delta x = 0.01$), and $200 \times 200 \times 200$ ($\Delta x = 0.005$) grids with $\Delta t = 0.005$, 0.0025, and 0.00125, respectively. The errors of the PLIC and the APPLIC are the smallest and the second smallest for all cases.

24
Figure 20: The initial ($t = 0$) and the final ($t = T$) shapes for test 3, using a $100 \times 100 \times 100$ grid with $\Delta t = 0.0025$. 
3.2. Computational efficiency

We investigate the computational efficiency of the following method: VOF/WLIC, THIC/WLIC, PLIC, and APPLIC. The computational efficiency of the THINC/SW and the SVOF methods is comparable to that of the THINC/WLIC and the VOF/WLIC methods, respectively. The computational time required to evaluate fluxes for the 10,000,000 sample points in the set $S$, defined in section 2.3, for each method is used as a measure of efficiency.

The benchmark program is written in the C language. Calculations using both single-precision and double-precision floating point arithmetic were conducted. Three different computing platforms, summarized in Table 6, were used to measure computational times.
The measured computational times are shown in Table 7 (single-precision) and in Table 8 (double-precision). As shown in the tables, the APPLIC method was 1.6-2.4 times faster than the PLIC method. The VOF/WLIC and the PLIC methods were the fastest and the slowest among the methods, respectively. Although the algorithm of the THINC/WLIC method is more simple than that of the APPLIC method, the computational times of the two methods were comparable. This is due to the calculations of transcendental functions, which are much more time-consuming than the basic arithmetic operations. To evaluate each flux, the THINC/WLIC and the APPLIC methods requires five (exp, log×2, and cosh×2) and four (exp×2 and log×2) transcendental functions, respectively.

4. Conclusions

We have presented a new PLIC-type VOF method called the APPLIC method. In this method, the complicated forward and inverse problems that arise with the PLIC method are approximately solved through the use of the extremely simple formulae. Accordingly, the APPLIC method is easier to develop and to maintain the computational codes than the standard PLIC method. The APPLIC method satisfies Eqs. (30)-(34), which are essential to be satisfied for any VOF methods.

We conducted computational tests to compare accuracy of the APPLIC method with other VOF methods: SVOF, VOF/WLIC, THINC/SW, THINC/WLIC, and PLIC. The results of the tests show that the APPLIC method is as accurate as the PLIC method and more accurate than the SVOF, VOF/WLIC, THINC/SW, and THINC/WLIC methods. It was demonstrated that the computational time of the APPLIC method is shorter than that of the PLIC method and comparable to that of the THINC/WLIC method.

Acknowledgments

This research is partially supported by the Center of Innovation Program from Japan Science and Technology Agency, JST. The author thanks Akira Sou, Ippei Oshima, and Kensuke Yokoi for engaging in insightful
Appendix A.

In this appendix, we explain the derivation of Eq. \([A.7]\) to evaluate \(C_A\).

The PLIC method assumes the shape of the dark fluid in the donor cell \(\Omega\) as the intersection of \(\Omega\) and an oriented plane \(\{x | n \cdot x < \alpha\}\). The SZ algorithms work with a unit cube and a normal vector \(m\) so that \(m \geq 0\) and \(\|m\|_1 = 1\). Therefore, we apply a coordinate transformation from \(x\) to \(x'\) so that the donor cell \(\Omega\) is mapped to the unit cube \(\{x' \in [0,1]^3\}\), and the normal vector \(n\) is mapped to a vector \(m'\) satisfying the relation \(m' \geq 0\). The transformation, represented by \(T_0\), is written as

\[
x'_I = \text{sgn} n_I \frac{n_I \Delta x}{\Delta x} (x_I - \xi_I),
\]

where \(\xi\) is the origin of the new coordinate, which is chosen from the eight vertices of \(\Omega\) depending on the signs of \(n_1, n_2,\) and \(n_3\) so that the image of the cell \(\Omega\) coincides with the unit cube \(\{x' \in [0,1]^3\}\). The vertex \(\xi\) is placed on the face \(\phi^*\) if the signs of \(m_I\) and \(n_I\) are identical, or on the face \(\phi\) otherwise. We can express the image of the oriented plane \(\{x | n \cdot x < \alpha\}\) under \(T_0\) as \(\{x' | m' \cdot x' < \alpha'\}\), where \(\alpha'\) is the transformed plane constant. The vector \(m'\) is given by

\[
m'_I = \frac{n_I \Delta x}{\text{sgn} n_I} + \frac{n_2 \Delta x}{\text{sgn} n_2} + \frac{n_3 \Delta x}{\text{sgn} n_3} = \frac{|n_I|}{\|n\|_1},
\]

(A.2)

The plane constant \(\alpha'\) is determined by the solution of the inverse problem

\[
\alpha' = \alpha(C,m').
\]

(A.3)

Let \(\Omega'_A\) be the image of \(\Omega_A\) under \(T_0\). We apply a further coordinate transformation \(T_A\) from \(x'\) to \(x''\) so that \(\Omega'_A\) is mapped to the unit cube \(\{x'' \in [0,1]^3\}\) and all the components of the transformed normal vector are nonnegative. The transformation \(T_A\) is written as

\[
x''_I = \begin{cases} x'_I & \text{if } l \neq I, \\ x'_I/|g| & \text{if } l = I \text{ and } \xi \text{ is placed on } \phi, \\ [x'_I - (1 - |g|)]/|g| & \text{if } l = I \text{ and } \xi \text{ is placed on } \phi^*. \end{cases}
\]

(A.4)

We can express the image of the oriented plane \(\{x' | m' \cdot x' < \alpha'\}\) under \(T_A\) as \(\{x'' | m''_A \cdot x'' < \alpha''_A\}\). The normal vector \(m''_A\) and plane constant \(\alpha''_A\) are given by

\[
m''_{A,I} = \begin{cases} Q'_A m'_I |g| & \text{if } l = I, \\ Q'_A m'_I & \text{if } l \neq I, \end{cases}
\]

(A.5)

\[
\alpha''_A = \begin{cases} Q'_A [\alpha' - m'_I (1 - |g|)] & \text{if } \xi \text{ is placed on } \phi^*, \\ Q'_A \alpha' & \text{if } \xi \text{ is placed on } \phi, \end{cases}
\]

(A.6)

where \(Q'_A\) is the normalization factor determined by

\[
Q'_A = \frac{1}{m'_1 + m'_2 + m'_3 - m'_I + m'_I |g|} = \frac{1}{1 - m'_I (1 - |g|)}.
\]

(A.7)

Let \(\Omega''_A\) be the image of \(\Omega'_A\) under \(T_A\). The volume of the shape \(\Omega''_A \cap \{x'' | m''_A \cdot x'' < \alpha''_A\}\) in the \(x''\) coordinate system is determined by solving the forward problem \(V(\alpha'', m'')\). The partial volume fraction \(C_A\) is thus obtained via Eq. \([A.7]\).
Appendix B.

In this appendix, we derive the choice criterion \(|\bar{\alpha}_C' - 1/2| > |\bar{\alpha}_B'' - 1/2|\).

We define the approximation errors in \(\bar{F}_A\) and \(\bar{F}_B\) as

\[ \varepsilon_A = (\bar{F}_A - F_{\text{PLIC}}) \text{sgn } g, \]  
\[ \varepsilon_B = (\bar{F}_B - F_{\text{PLIC}}) \text{sgn } g. \]  

(B.1)
(B.2)

Because \(F_{\text{PLIC}} \text{sgn } g = V(\alpha''_A, m''_A)|g| = |C - (1 - |g|)V(\alpha''_B, m''_B)|\), we have

\[ \varepsilon_A = |\bar{V}(\bar{\alpha}'', \bar{m}'') - V(\alpha''_A, m''_A)||g|, \]  
\[ \varepsilon_B = -|\bar{V}(\bar{\alpha}'', \bar{m}'') - V(\alpha''_B, m''_B)|(1 - |g|), \]  

where The aim of this appendix is to find a quick and easy criterion to guess whether \(|\varepsilon_A| < |\varepsilon_B|\) or not.

The approximation error in \(\bar{V}\) and \(\bar{\alpha}\) are defined as

\[ \varepsilon_{\alpha}(V, m) = \bar{\alpha}(V, m) - \alpha(V, m), \]  
\[ \varepsilon_V(\alpha, m) = \bar{V}(\alpha, m) - V(\alpha, m). \]  

(B.5)
(B.6)

Using Eqs. (B.17b) and (B.17), we obtain

\[ \alpha' = \bar{\alpha}' - \varepsilon_{\alpha}(C, m'), \]  
\[ \alpha'' = \bar{\alpha}'' - Q'_{A} \varepsilon_{\alpha}(C, m'). \]  

(B.7)
(B.8)

The approximation error \(\varepsilon_A\) is then given by

\[ \varepsilon_A = |\bar{V}(\bar{\alpha}'', \bar{m}'') - V(\alpha''_A, m''_A)| |g| = \left| \varepsilon_V(\bar{\alpha}'', \bar{m}'') + Q'_{A} \varepsilon_{\alpha}(C, m') \frac{\partial}{\partial \alpha} V(\bar{\alpha}'', \bar{m}'') \right| \]  
\[ + O(Q'_{A}|g| \varepsilon_{\alpha}^2). \]  

(B.9)

The value of \(Q'_{A}|g|\) is included in the range [0, 1] because \(m'_f \in [0, 1]\) and \(|g| \in (0, 1]\).

Let \(V^*\) be an arbitrary value within [0, 1]. The following relation holds:

\[ \bar{V}[\bar{\alpha}(V^*)] = V[\alpha(V^*)] = V^*, \]  

(B.10)

Here the second arguments regarding \(m\) are omitted for brevity. On the other side, the following can be obtained:

\[ V[\alpha(V^*)] = V[\bar{\alpha}^* - \varepsilon_{\alpha}(V^*)] \]  
\[ = V(\bar{\alpha}^*) - \varepsilon_{\alpha}(V^*) \frac{\partial}{\partial \alpha} V(\bar{\alpha}^*) + O(\varepsilon_{\alpha}^2), \]  

(B.11)

where \(\bar{\alpha}^*\) stands for \(\bar{\alpha}(V^*)\). Using Eqs. (B.10) and (B.11), we have the relation between \(\varepsilon_V\) and \(\varepsilon_{\alpha}\) as

\[ \varepsilon_V(\bar{\alpha}^*) = -\varepsilon_{\alpha}(V^*) \frac{\partial}{\partial \alpha} V(\bar{\alpha}^*) + O(\varepsilon_{\alpha}^2). \]  

(B.12)

Substituting Eq. (B.12) into (B.9) and neglecting the term \(O(\varepsilon_{\alpha}^2)\), we obtain

\[ \varepsilon_A \approx |g| \delta_{\Lambda} \frac{\partial}{\partial \alpha} V(\bar{\alpha}'', \bar{m}''), \]  

(B.13)

with

\[ \delta_{\Lambda} = Q'_{A} \varepsilon_{\alpha}(C, m') - \varepsilon_{\alpha}(V^*_\Lambda, m^*_\Lambda). \]  

(B.14)
Similarly,

$$\varepsilon_B \approx -(1 - g)\delta_B \frac{\partial}{\partial \alpha} V(\tilde{\alpha}_B'', m_B''),$$  \hspace{1cm} (B.15)

with

$$\delta_B = Q_B'\varepsilon_\alpha(C, m') - \varepsilon_\alpha(\tilde{V}_B'', m_B'').$$  \hspace{1cm} (B.16)

Here $\tilde{V}_A''$ and $\tilde{V}_B''$ stand for $\tilde{V}(\tilde{\alpha}_A'', m_A')$ and $\tilde{V}(\tilde{\alpha}_B'', m_B'')$, respectively.

The value of $\delta_A$ becomes zero when $|g|$ is equal to one (because $\tilde{V}_A'' = C, m''_A = m'$, and $Q_A' = 1$ if $|g| = 1$). Similarly, $\delta_B$ becomes zero when $|g| = 0$. It is evident that $\delta_A$ and $\delta_B$ are smooth functions with respect to $|g|$. Considering these conditions, we adopt the following approximations:

$$\frac{|\delta_A|}{|\delta_A| + |\delta_B|} \approx 1 - |g|,$$ \hspace{1cm} (B.17)

$$\frac{|\delta_B|}{|\delta_A| + |\delta_B|} \approx |g|,$$ \hspace{1cm} (B.18)

where we attach importance on simplicity rather than on accuracy. Figure B.22 demonstrates the accuracy of the approximations. The data points are distributed roughly around the line $|\delta_A|/(|\delta_A| + |\delta_B|) = 1 - |g|$.

By use of Eqs. (B.17) and (B.18), the inequality $|\varepsilon_A| < |\varepsilon_B|$ can be written as

$$\frac{\partial}{\partial \alpha} V(\tilde{\alpha}_A'', m_A'') < \frac{\partial}{\partial \alpha} V(\tilde{\alpha}_B'', m_B''),$$  \hspace{1cm} (B.19)

We employ the following approximation for simplicity:

$$\frac{\partial}{\partial \alpha} V(\alpha, m_A'') = \frac{\partial}{\partial \alpha} V(\alpha, m_B''),$$  \hspace{1cm} (B.20)
where $\alpha$ is an arbitrary value. Because $\partial V / \partial \alpha$ is a monotonically nondecreasing function of $\alpha < 1/2$ and has even symmetry to $\alpha = 1/2$, inequality (B.19) becomes

$$|\tilde{\alpha}_A'' - 1/2| > |\tilde{\alpha}_B'' - 1/2|.$$  \hfill (B.21)

Thus we obtain the criterion.
References

[1] C. W. Hirt, B. D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, Journal of Computational Physics 39 (1981) 201–225.

[2] M. Rudman, Volume-tracking methods for interfacial flow calculations, International Journal for Numerical Methods in Fluids 24 (1997) 671–691.

[3] W. J. Rider, D. B. Kothe, Reconstructing volume tracking, Journal of Computational Physics 141 (1998) 112–152.

[4] R. Scardovelli, S. Zaleski, Direct numerical simulation of free-surface and interfacial flow, Annual Review of Fluid Mechanics 31 (1999) 567–603.

[5] J. E. Pilliod Jr, E. G. Puckett, Second-order accurate volume-of-fluid algorithms for tracking material interfaces, Journal of Computational Physics 199 (2004) 465–502.

[6] 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, New York, 1982, pp. 273–285.

[7] J. Li, Calcul d’interface affine par morceaux (piecewise linear interface calculation), Comptes Rendus des Seances de l’ Academie des Sciences Paris, Série IIb 320 (1995) 391–396.

[8] W. F. Noh, P. Woodward, Slic (simple line interface method), Lecture Notes in Physics 24 (1976) 330–340.

[9] K. Yokoi, Efficient implementation of THINC scheme: A simple and practical smoothed VOF algorithm, Journal of Computational Physics 226 (2007) 1985–2002.

[10] M. Marek, W. Aniszewski, A. Bogusławski, Simplified volume of fluid method (SVOF) for two-phase flows, TASK Quarterly 12 (2008) 255–265.

[11] F. Xiao, Y. Honma, T. Kono, A simple algebraic interface capturing scheme using hyperbolic tangent function, International Journal for Numerical Methods in Fluids 48 (2005) 1023–1040.

[12] F. Xiao, S. Ii, C. Chen, Revisit to the THINC scheme: a simple algebraic vof algorithm, Journal of Computational Physics 230 (2011) 7086–7092.

[13] S. Ii, K. Sugiyama, S. Takeuchi, S. Takagi, Y. Matsumoto, F. Xiao, An interface capturing method with a continuous function: The THINC method with multi-dimensional reconstruction, Journal of Computational Physics 231 (2012) 2328–2358.

[14] R. Scardovelli, S. Zaleski, Analytical relations connecting linear interfaces and volume fractions in rectangular grids, Journal of Computational Physics 164 (2000) 228–237.

[15] J. Hennessy, D. Patterson, Computer Architecture: A Quantitative Approach, The Morgan Kaufmann Series in Computer Architecture and Design, Elsevier Science, 2006.

[16] R. Scardovelli, S. Zaleski, Interface reconstruction with least-square fit and split Eulerian–Lagrangian advection, International Journal for Numerical Methods in Fluids 41 (3) (2003) 251–274.

[17] E. Aulisa, S. Manservisi, R. Scardovelli, S. Zaleski, Interface reconstruction with least-squares fit and split advection in three-dimensional Cartesian geometry, Journal of Computational Physics 225 (2) (2007) 2301–2319.

[18] G. Weymouth, D. K.-P. Yue, Conservative Volume-of-Fluid method for free-surface simulations on Cartesian-grids, Journal of Computational Physics 229 (8) (2010) 2853–2865.
[19] C. S. Wu, D. L. Young, H. C. Wu, Simulations of multidimensional interfacial flows by an improved volume-of-fluid method, International Journal of Heat and Mass Transfer 60 (2013) 739–755.

[20] T. Vignesh, S. Bakshi, Noniterative interface reconstruction algorithms for volume of fluid method, International Journal for Numerical Methods in Fluids 73 (1) (2013) 1–18.

[21] B. Parker, D. Youngs, Two and Three Dimensional Eulerian Simulation of Fluid Flow with Material Interfaces, Atomic Weapons Establishment, 1992.

[22] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level set method for improved interface capturing, Journal of Computational Physics 183 (1) (2002) 83–116.

[23] S. T. Zalesak, Fully multidimensional flux-corrected transport algorithms for fluids, Journal of Computational Physics 31 (3) (1979) 335–362.

[24] R. J. LeVeque, High-resolution conservative algorithms for advection in incompressible flow, SIAM Journal on Numerical Analysis 33 (2) (1996) 627–665.