Available online at www.sciencedirect.com
ScienceDirect
Computers & Fluids 35 (2006) 1011–1032
computers & fluids
www.elsevier.com/locate/compfluid

A 3D unsplit-advection volume tracking algorithm with planarity-preserving interface reconstruction

Petar Liovic [a,*], Murray Rudman [b], Jong-Leng Liow [c], Djamel Lakehal [a], Doug Kothe [d]

[a] Institute of Energy Technology, ETH-Zürich, CH-8092 Zürich, Switzerland
[b] CSIRO Manufacturing and Infrastructure Technology, PO Box 56, Highett, Victoria 3190, Australia
[c] School of Engineering, James Cook University, Townsville, Queensland 4811, Australia
[d] Computer and Computational Sciences Division, Continuum Dynamics Group, MS B296, Los Alamos National Laboratory, NM 87545, USA

Received 15 December 2004; received in revised form 11 May 2005; accepted 5 September 2005
Available online 5 January 2006

Abstract

A new volume tracking method is introduced for tracking interfaces in three-dimensional (3D) geometries partitioned with orthogonal hexahedra. The method approximates interface geometries as piecewise planar, and advects volumes in a single unsplit step using fully multidimensional fluxes that have their definition based in backward-trajectory remapping. By using multidimensional unsplit advection, the expense of high-order interface reconstruction is incurred only once per timestep. Simple departures from strict backward-trajectory remapping remove any need for consideration of volume computations involving shapes consisting of non-planar ruled surfaces. Second-order accuracy of the method is demonstrated even for vigorous 3D deformations.

1. Introduction

Interface tracking, or specialized numerical solution techniques for modeling interface kinematics on computational meshes, is required for the accurate simulation of interfacial physics in multi-fluid flows. Accurate preservation of discontinuities in these problems is particularly important to the overall accuracy of the flow solver, since jump conditions and body forces at the interfaces are only represented as faithfully as the simulated locations of interfaces allow them to be. Concerted development over decades has yielded three general classes of interface tracking methods:

- *capturing*: continuum advection [1]; Level Set [2,3]; cubic-interpolated propagation (CIP) [4]; volume tracking (Volume-of-Fluid, or VOF) methods [5–7];
- *tracking*: front tracking [8,9]; particles [10];
- *Lagrangian*: conventional Lagrangian [11]; free Lagrange [12].

1.1. Motivations for improved 3D volume tracking

Volume tracking, or Volume-of-Fluid methods, have proven remarkably successful in computational fluid dynamics (CFD) simulation of interfacial flows [13–17]. Higher-fidelity improvements are still desirable—especially so in 3D, given that the lower resolution achievable in 3D simulation (as compared to 2D) means interface tracking errors become more significant and under-resolved regions become more prevalent. We seek a 3D volume tracking method that possesses the robustness of typical volume tracking, while raising the bar on 3D performance to the levels of accuracy attainable with the optimal 2D methods.

To illustrate our desire for improved volume tracking, we consider the example of gas injection through a pipe into a liquid bath. In the case of bubbles from the pipe bursting at the bath free surface, Fig. 1(a) and (b) show

* Corresponding author.
E-mail address: liovic@iet.mavt.ethz.ch (P. Liovic).

0045-7930/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compfluid.2005.09.003

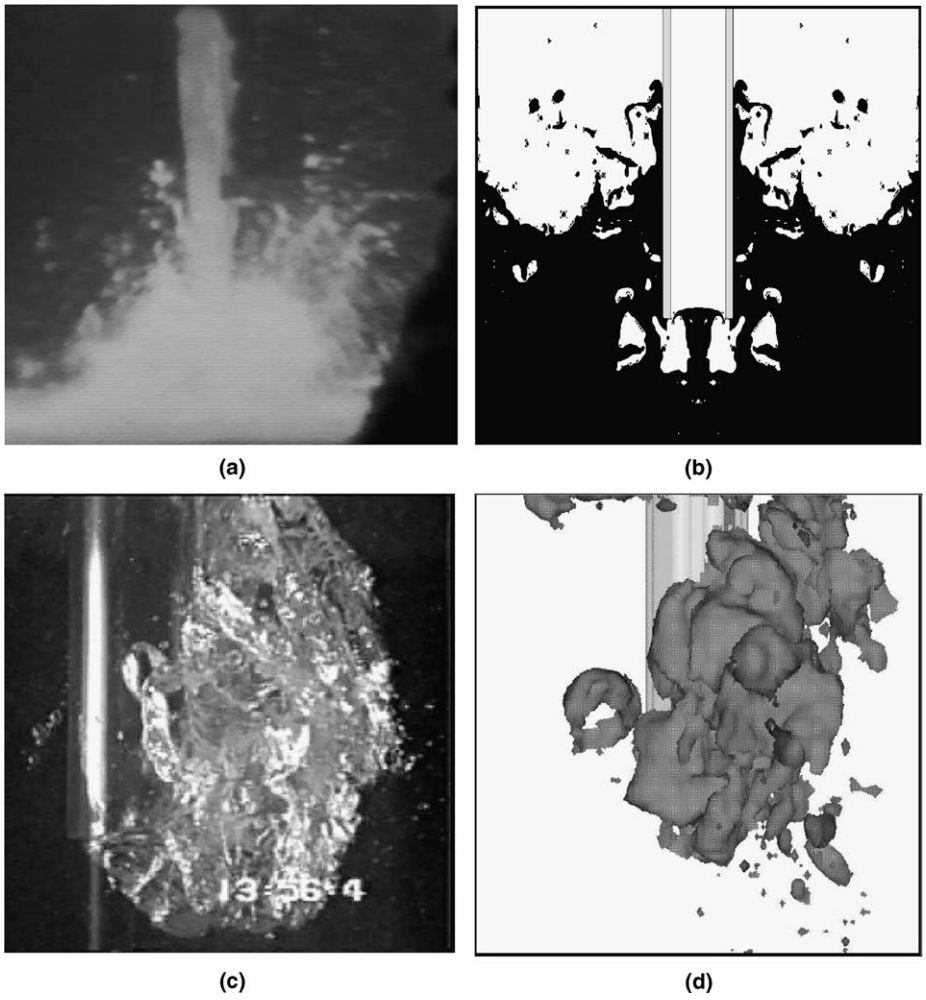

Fig. 1. Photographed and simulated features of gas injection into a liquid bath: (a) above the free surface, the dispersed phase is liquid, in the form of ligament sheets and splash [46,47]; (c) below the free surface, the dispersed phase is gas, in the form of bubbles of various scales [14,17]; (b) a 2D axisymmetric simulation using volume tracking [16,47] handles the fragmentation and coalescence naturally, but 3D effects cannot be represented; and (d) 3D volume tracking [17].

that the gas phase is the dispersed phase below an average free surface level, while the liquid phase is the dispersed phase above that level.

Fig. 1(a) and (b) display splash drops fragmenting from bath liquid that has been stretched into ligament sheets, while Fig. 1(b)–(d) display small bubbles fragmenting from larger bubbles growing at the pipe tip. Liquid splash drops then re-coalesce with each other in mid-flight or with the bulk liquid bath, while gas bubbles re-coalesce in bubble rise or coalesce with the freeboard gas upon disengagement at the free surface. For the flows in Fig. 1, we propose volume tracking to be the approach most capable of handling the extreme interface transformations in the flow. We also expect improved volume tracking to contribute to the enhanced fidelity of simulated solutions for this flow, through accuracy and efficiency gains promoting better resolution of interfacial features.

In bubbling and splashing of the type shown in Fig. 1, it is necessary for the flow simulation to capture the regaining of large length-scale topological structures from the deformation and coalescence of smaller length-scale structures. This is possible in simulations where interfaces are tracked while conserving mass locally and globally; irreversible information loss through large local departures from mass conservation should be minimized. Using volume tracking algorithms described in this paper, the simulations of Fig. 1 are able to capture the gross gas/bubble structures seen in experiment, even after multiple high-curvature interfaces are created and destroyed. Further details of these simulations can be found in [17]. For a more intimate coupling of the interface tracking solution into Eulerian flow solvers, Rudman [18] and Bussmann et al. [19] show the manner in which fluxes from volume tracking algorithms can be used to generate highly accurate local density estimates for conservative momentum advection; the idea is shown in Fig. 2. With minimal computational overhead, fluxes from a VOF algorithm allow for a consistent approach to mass and momentum conservation that results in the more accurate fulfillment of jump conditions at the interfaces. Such consistency allows Eulerian flow solvers to

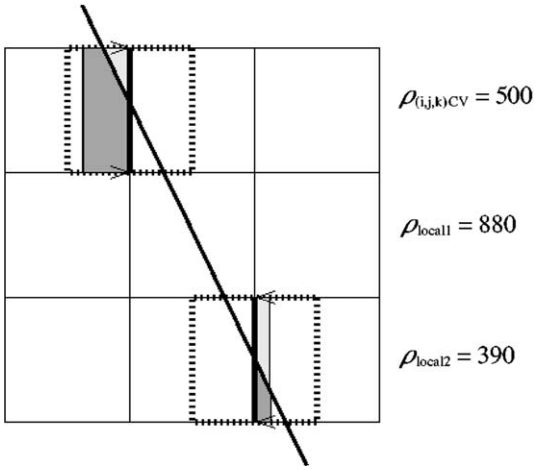

Fig. 2. Density estimates using VOF-augmented momentum advection, using the example of the update of $u_{i+1/2,j,k}$, and using the momentum-conservative form of the advective flux $F_{i,j,k} \approx u^{fl}_{i,j,k}[\rho u]_{i,j,k}$. Volume-weighted averaging within the fluid parcels that cross the (i, j, k) faces—the VOF fluxes—ensures stability and promotes conservation. Using less representative density estimates, such as simple averaging over the entire control volume about (i, j, k), has been observed to result in poor robustness.

remain stable for large density-ratio multi-fluid flow simulation, as shown by Bussmann and coworkers for density ratios of 10^9 and above [19].

Given the demands of flow problems similar to those associated with Fig. 1, the ideal 3D interface tracking method for CFD application should maintain a compact interface width no larger than the mesh size, and conserve mass locally. In terms of accuracy, we establish planarity preservation—the ability of an interface tracking method to reconstruct a 3D plane throughout the history of advection in a non-deforming velocity field—as a key to reducing numerically-induced curvature evolution; this requirement is equivalent to asymptotic second-order accuracy in space and time. Topological robustness is an even more necessary requirement, to avoid the case-by-case logic failures in capturing highly arbitrary interface merging and tearing that may be inevitable using other approaches. The interface tracking method should also be easily implemented and efficient.

Up to now, no interface tracking method (or class of methods) has fulfilled all criteria. Volume tracking methods are popular, because their established conservation, robustness and versatility properties are well known in 2D, with the achievable property of rigorous local volume conservation being particularly valued. Preservation of formal high-order accuracy, ease of implementation and efficiency remain issues associated with the extension of volume tracking to 3D. An approach adopted in other work to overcome these difficulties has been to develop VOF-based hybrid schemes, such as the Coupled Level Set and Volume-of-Fluid (CLSVOF) method of Sussman and Puckett [20], and the mixed markers and Volume-of-Fluid method of Aulisa et al. [21]. In this paper, we alternatively demonstrate that the superior properties of 2D linearity-preserving volume tracking are readily realizable in 3D, without any need for hybridization.

1.2. Overview of documented volume tracking methods

We proceed by considering the Piecewise Linear Interface Calculation based on the works of Debar [5] and Youngs [22] (hereafter described generically as PLIC-VOF) to define modern volume tracking. In PLIC-VOF, arbitrarily oriented line/plane segment interface reconstructions introduced first-order accuracy [23], ensured discontinuities were maintained within one cell rather than smeared over several cells [18], and suppressed the generation of flotsam and jetsam that is characteristic of lower accuracy piecewise constant schemes [24,25]. Second-order PLIC-VOF interface reconstruction schemes have been developed in 2D [23,26–28], and extended to 3D in [29] (albeit in a tedious manner). Increased accuracy has been sought in 2D [30] and in 3D [31] through the use of piecewise parabolic interface reconstruction.

For time integration in volume tracking, 1D flux calculations combined with operator splitting along coordinate directions continues to be the norm. Unsplit advection using fully multidimensional fluxes defined in an Eulerian manner [23,24,32–34] has been shown to improve accuracy, with the additional benefit of requiring only a single-stage algorithm. Mosso et al. [27,35] has demonstrated linearity preservation using a Lagrangian full-remap approach to unsplit advection in 2D volume tracking, while Shahbazi et al. [36] has preserved second-order accuracy in vortical flows using a similar method implemented on triangular meshes. The Lagrangian algorithm of Scardovelli and Zaleski [28] is derived from a Lagrangian-remap point of view, and is essentially a 1D version of the Mosso work. All of the methods listed above are 2D. In 3D, Miller and Colella [29] have presented an extension of the unsplit advection scheme of Pilliod and Puckett [23].

1.3. Overview of paper

The tedium necessary for implementation of any PLIC-based volume tracking in 3D, at first glance, remains high compared to other interface tracking schemes. More advanced PLIC-VOF schemes (second-order interface reconstruction, unsplit advection) have not been extended from 2D to 3D, because the increase in complexity of the geometry primitives involved has made implementation excessively difficult and ultimately infeasible. In this paper, we present a new 3D volume tracking scheme, featuring planarity-preserving interface reconstruction and unsplit advection using fully multidimensional fluxes. The approach is also described in sufficient detail for implementation, with at least one simple framework presented for completing the bulk of the geometric tasks that need to be performed.

2. The volume tracking problem

In interface tracking that captures the interface representation in a variable distribution on an underlying Eulerian mesh, solutions are sought for the standard mass conservation equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{U}) = 0, \tag{1}$$

where ρ is the fluid density and $\mathbf{U}$ is the fluid velocity. In the case of volume tracking algorithms, we introduce the definition $\rho = \sum_k C_k \rho_k^0$ for the fluid density of interfacial flows comprised of k incompressible fluids, where C_k and ρ_k^0 are the volume fraction and density of fluid k respectively. Upon substituting this definition into Eq. (1), and commuting the summations over k, an evolution equation is obtained for each volume fraction C_k:

$$\frac{\partial C_k}{\partial t} + \nabla \cdot (C_k \mathbf{U}) = 0. \tag{2}$$

Variable C_k in Eq. (2) is ultimately a discrete Heaviside (or "color" C) function being advected with the fluid velocity $\mathbf{U}$. Volume tracking algorithms therefore begin with Eq. (2) as a model for interface kinematics, with particular attention thereafter paid to geometrically-based numerical solutions obtained for the advection term $\nabla \cdot (C_k \mathbf{U})$. Using a discrete Heaviside model for the interface, the location of the interface is deduced from the color function distribution; a unique, exact interface location is not possible as a solution.

A typical algorithm for the volume tracking of interfaces bounding incompressible fluids starts by integrating Eq. (2) over volume, giving an evolution equation for the volume V_k of the kth fluid:

$$V_k^{n+1} = V_k^n - \delta t \int \nabla \cdot (\mathbf{U} C_k) \mathrm{d}V, \tag{3}$$

where a first-order time discretization has been used and the definitions $V_k = \int C_k \mathrm{d}V = C_k V$ and $V = \sum_k V_k$ apply. Upon converting the volume integral to a surface integral, and approximating this integral with a spatially second-order quadrature over discrete faces f bounding the control volume V, Eq. (3) can be approximated discretely as

$$V_k^{n+1} = V_k^n - \sum_f \delta V_{k,f}^n, \tag{4}$$

where δV_f, the total volume change across face f (given by $\mathbf{U}_f \cdot \mathbf{A}_f \delta t$) must be partitioned into individual volumes $\delta V_{k,f}$, i.e., $\delta V_f = \sum_k \delta V_{k,f}$. Here $\mathbf{A}_f$ is the outward-facing area vector of face f, $\mathbf{A}_f = \hat{\mathbf{n}}_f A_f$, where $\hat{\mathbf{n}}$ is the face unit normal vector. For second-order quadratures, all face quantities are evaluated at the face centroid locations. For real or numerical departures from incompressibility, it is sometimes useful to retain the last "divergence source term" (from the product-rule expansion of the advection term in Eq. (3)) as

$$V_k^{n+1} = V_k^n - \sum_f \delta V_{k,f}^n + \int C_k^n (\nabla \cdot \mathbf{U}) \mathrm{d}V \tag{5}$$

to help keep C_k bounded between zero and unity. While we prefer to work numerically with fluid volumes V_k, Eq. (5) above can be rewritten as a volume fraction evolution equation:

$$C_k^{n+1} = C_k^n - \frac{1}{V^{n+1}} \sum_f F_{k,f}^n + \frac{1}{V^{n+1}} \int C_k^n (\nabla \cdot \mathbf{U}) \mathrm{d}V, \tag{6}$$

where we alternatively describe partitioned volume changes across cell faces as C-fluid fluxes ($F_{k,f} = \delta V_{k,f}$).

Volume tracking algorithms consist of two main geometrically-based steps: (i) interface reconstruction, and (ii) interface advection. The details of our treatment of each of these steps in the new 3D volume tracking scheme are presented next.

3. Interface reconstruction

3.1. Introductory remarks

Error-minimizing PLIC-VOF schemes (such as the 2D schemes in [23,24,27,28]) are pursued for extension to 3D in the current work, as they are far more amenable to coupling with various advection schemes and unstructured meshes in 3D (as compared to curved interface reconstruction methods). In PLIC-VOF methods, the interface geometry is described by $\hat{\mathbf{n}} \cdot \mathbf{x} = \rho$, where $\hat{\mathbf{n}}$ is the interface unit normal, and ρ is the "plane constant" (distance from the origin) that fixes the interface location. We base the current work on the PLIC-VOF paradigm introduced by Youngs [22], in which a non-unique line segment is defined in each mesh cell to approximate the interface. In this paradigm, interface continuity across mesh cell faces is not enforced, such that most reconstructed interfaces feature small discontinuities at the faces shared by adjacent mesh cells. The $(\hat{\mathbf{n}}, \rho)$ solution in any interface cell is constrained to enclose volume $V_{\mathrm{tr}} = C_{\mathrm{cell}} V_{\mathrm{cell}}$; error-minimizing methods then seek a solution that is unique by being "optimized" in some sense.

One approach that has been used to optimize interface plane location uses volume-based error minimization, in the form of 2D Fast Least-Squares (FLS) method of Pilliod and Puckett [23], and the 3D ELVIRA method of Miller and Colella [29]. In this orthogonal-mesh approach, an $(\hat{\mathbf{n}}, \rho)$ combination minimizes the least-squares error:

$$\epsilon_{i,j,k} = \sum_{kk=-pz}^{pz} \sum_{jj=-py}^{py} \sum_{ii=-px}^{px} [(C_{i+ii,j+jj,k+kk} - \overline{C}_{i+ii,j+jj,k+kk}) \delta x_{i+ii} \delta y_{j+jj} \delta z_{k+kk}]^2, \tag{7}$$

where (px, py, pz) are the radial widths (in mesh cells) of the stencil support, C is the actual color function, and $\overline{C}$ is the color function resulting from extrapolation of the interface plane beyond cell (i, j, k) throughout the rest of the stencil. When a list of candidate approximants is gener-

ated from an adequately-sized stencil and using all possible variations in spatial differencing, volume-based error-minimization methods are second-order accurate, and preserve linearity (in 2D) or planarity (in 3D). The original 2D FLS version [23] achieves this without the need for outer-loop iteration on the $(\hat{\mathbf{n}}, \rho)$ solution, using six candidate approximants and a 9-cell stencil ($px = py = 1$, $pz = 0$). The 3D ELVIRA extension proposed by Miller and Colella [29], while also non-iterative, requires the use of 72–144 candidate approximants and a 125-cell stencil ($px = py = pz = 2$). The 3D ELVIRA method is hardly a straightforward extension of the 2D FLS method; the 3D concept to achieve error minimization is more complex, the increase in computation from 2D to 3D is large, and the proportion of interface topology features with radii of curvature less than the stencil width is also increased. These issues motivate consideration of alternatives for high-order planar interface reconstruction in 3D.

An alternative approach to interface reconstruction is to progressively refine interface orientation $\hat{\mathbf{n}}$, by averaging across mesh cell faces, in an attempt to flatten out discontinuities between adjacent interface reconstructions. In the 2D method developed by Swartz [26] (and first described in algorithmic form by Mosso [27]), a normal candidate in the target interface cell is exported to the surrounding interface cells in the 3×3 stencil. Connecting the centroid of the interface reconstruction in the target cell with a centroid in any neighboring cell results in a line segment, of modified $\hat{\mathbf{n}}$, that is continuous across the common mesh cell face. All such modifications of $\hat{\mathbf{n}}$ are averaged, leading to a new normal estimate in the target cell. Iteration of the procedure results in linearity-preserving and second-order accurate interface reconstruction in 2D [24,27].

3.2. Centroid-vertex triangle-normal averaging (CVTNA)

A design guideline adopted here for "feasible" interface reconstruction is that high-order accuracy should not be achieved at the cost of increased stencil width: interface reconstruction must be based on a 27-cell stencil. No error-minimizing PLIC-VOF method has been developed as yet for 3D that meets this guideline. In this paper, we propose a 3D piecewise-planar interface reconstruction method, based on a conceptually simple 3D extension of the 2D algorithm described in [27]. Referred to by us as Centroid-Vertex Triangle-Normal Averaging (CVTNA), it is the first interface reconstruction method in 3D that reliably achieves high-order accuracy within a 27-cell stencil. We only describe the orthogonal-mesh implementation here: the CVTNA method is readily extensible to interface reconstruction on unstructured and Lagrangian meshes.

3.2.1. CVTNA algorithm

In the CVTNA algorithm, an estimate of the normal $\hat{\mathbf{n}}_{i,j,k}$ is copied to all interface cells within the 27-cell stencil about cell (i, j, k). Once the correct volumes are truncated by the interface planes throughout the stencil, the centroids of the interface reconstructions are extracted. The core of our extension of Swartz's method to 3D is in the construction of triangles from the centroids of interface reconstructions. The basic rule is that any L-shaped group of three cells (as shown in Fig. 3(a)) can be used to form a *triplet*, as long as the three cells all possess interfaces. The centroids in the triplet (Fig. 3(b)) form a *centroid-vertex triangle*, and Fig. 3(c) shows the manner in which the *centroid-vertex triangle normal* is extracted. The unit-normals are averaged to generate the new, improved estimate of $\hat{\mathbf{n}}_{i,j,k}$, thereby completing one outer-loop iteration of the CVTNA computation for the normal in cell (i, j, k). Iterations are repeated until changes in $\hat{\mathbf{n}}_{i,j,k}$ are no longer significant—usually two to four iterations. The stencil-based CVTNA algorithm is shown in Fig. 5.

With the naïve method described in the previous paragraph, cases exist resulting in low-amplitude oscillation between "brackets" on $\hat{\mathbf{n}}$, that nevertheless represent devi-

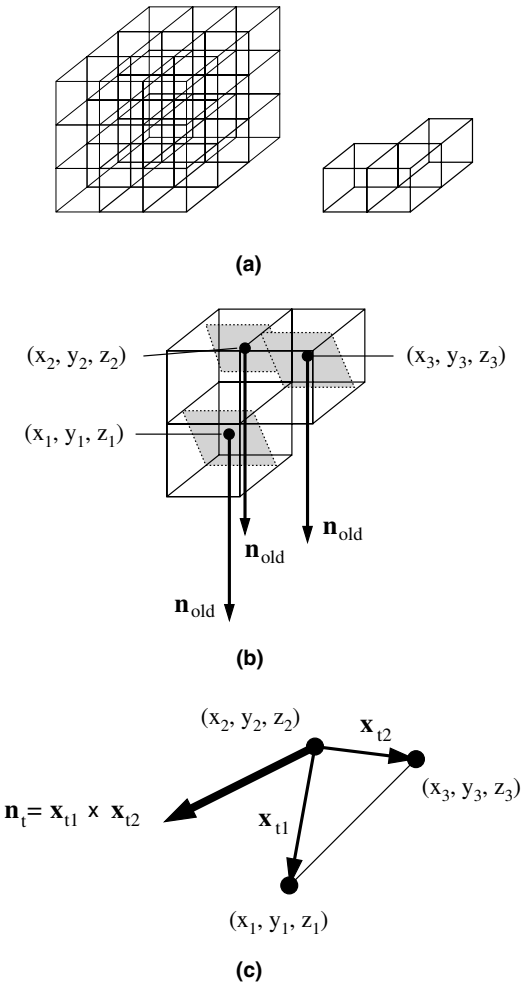

Fig. 3. Calculation of normal vectors for the sample to be averaged in the CVTNA method: (a) a sample L-shape cell triplets consisting of interface cells; (b) using the current normal, interface planes are located in each of the cells in the triplet, and the centroid of each reconstruction in the triplet is computed; (c) identifying the triangle created from the centroids, the normal estimate from the triplet is the vector orthogonal to the triangle.

ations from robust convergence. An example of such a case is shown in Fig. 4: every second outer-loop iteration, a triangle of centroids have the same y-coordinate, thus the normal to the plane containing this triangle of centroids must be $\hat{\mathbf{n}} \approx (0, 1, 0)$, despite the current normal estimate having a very small magnitude component in the y-direction. In a CVTNA implementation that is not naïve, the triplets that cause the oscillation are deemed unsuitable, and excluded from the CVTNA computation; Fig. 5 includes this step. Unsuitable triplet exclusion is achieved in the current work through a *filtering* procedure. The vast majority of centroid-vertex triangle normals are close to the optimizing solution, thus the normal in the sample causing the oscillatory behavior is easily identified as an outlier. Our procedure for filtering centroid-vertex triangle normals in any given outer-loop iteration is:

(1) Average the sample of unit-normals from the centroid-vertex triangles of all triplets.
(2) Compute $|\hat{n}^{\text{average}} - \hat{n}^{\text{triplet}_i}|$, for all centroid-vertex triangle normals in the sample.
(3) If $|\hat{n}^{\text{average}} - \hat{n}^{\text{triplet}_i}| > \epsilon_l$, then identify triplet i as unsuitable, and remove the normal from this triplet from the sample in all subsequent computations.
(4) Repeat (1)–(3) for the current outer-loop iteration.

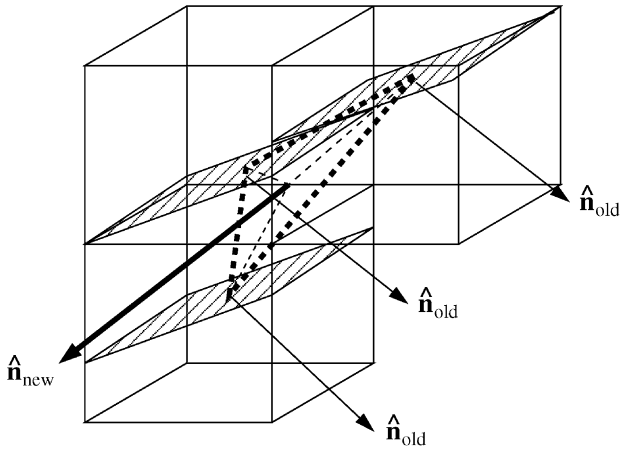

Fig. 4. A case of the normal estimate from a triplet resulting in oscillation in the CVTNA method. Such cases are identified and excluded from future samples in the CVTNA outer-loop iteration.

In the current work, four filtering sweeps with progressively smaller ϵ_l are used.

It should be noted that filtering during the CVTNA algorithm is not the only means of removing oscillation in the convergence of the CVTNA scheme; unsuitable triplets may also be identified before the outer-loop iteration commences. Unsuitable triplet exclusion in the current work has some analogy in Swartz's discussion of "usable" numbers. Restricting consideration of values of C in the stencil to "usefully usable" numbers [26] represents an alternative filtering approach for promoting strong outer-loop iteration convergence. The current work will use filtering during the CVTNA algorithm and demonstrate it to be

Estimate normal in central cell of stencil, $(\hat{n}_x, \hat{n}_y, \hat{n}_z)^1_{0,0,0}$, using Youngs gradient.
Start outer-loop iteration l
  Sweep through $3 \times 3 \times 3$ stencil (ii, jj, kk)
    If cell (ii, jj, kk) is a mixed cell
      Set normal estimate as $(\hat{n}_x, \hat{n}_y, \hat{n}_z)^l_{ii,jj,kk} = (\hat{n}_x, \hat{n}_y, \hat{n}_z)^l_{0,0,0}$.
      Compute $\rho_{ii,jj,kk}$ to enclose $V_{tr} = C_{cell} V_{cell}$, using Brent's method.
      Compute intercepts of $(\hat{n}_x, \hat{n}_y, \hat{n}_z, \rho)^l_{ii,jj,kk}$ with the edges of cell (ii, jj, kk).
      Store as $(x, y, z)^m_{ii,jj,kk}$, where $1 \geq m \geq 12$ denotes the mesh cell edge.
      From the intercepts stored in $(x, y, z)^m_{ii,jj,kk}$, generate the centroid.
    End If
  End sweep through $3 \times 3 \times 3$ stencil (ii, jj, kk).
  Re-express coordinates of all centroids relative to a single origin in the $3 \times 3 \times 3$ stencil.
  Centroid-vertex triangle-normal sample size starts at $n = 0$.
  Sweep through the 48 possible L-shaped triplets within the $3 \times 3 \times 3$ stencil
    If all cells of triplet are mixed cells, have a valid centroid-vertex triangle
      Sample size $n = n + 1$.
      Use cross-product to compute centroid-vertex triangle normal $(\hat{n}_x, \hat{n}_y, \hat{n}_z)^{CVT}_n$.
    End If
  End sweep through the 48 possible L-shaped triplets.
  Exclude normals from unsuitable triplets from sample.
  Average sample of $(\hat{n}_x, \hat{n}_y, \hat{n}_z)^{CVT}_n$ values to get new CVTNA estimate $(\hat{n}_x, \hat{n}_y, \hat{n}_z)^{l+1}_{0,0,0}$.
End outer-loop iteration l if CVTNA convergence criteria met.

Fig. 5. Outline of algorithm for the stencil-based computation in the CVTNA interface reconstruction scheme.

cheap and effective, while future work will compare the properties of the alternative approaches applied to CVTNA.

3.3. Stationary interface reconstruction testing

Given second-order accuracy in volume tracking interface reconstruction equates with planarity preservation in 3D, initial testing of the CVTNA interface reconstruction scheme focused on reconstructing arbitrarily-oriented domain-spanning planes. The following interface reconstruction methods were proposed in initial testing:

- the Youngs gradient method for normal orientation [22];
- naïve CVTNA;
- CVTNA using loose filter (0.4, 0.2, 0.1, 0.05);
- CVTNA using medium filter (0.4, 0.15, 0.05, 0.015);
- CVTNA using tight filter $(0.3, 0.125, 0.025, 0.02/n^2)$, $n =$ outer-loop iteration count.

In the plane testing, the criterion for determining the planarity preservation of an interface reconstruction scheme was

$$|\hat{n}_x - \hat{n}_x^{\text{exact}}|, |\hat{n}_y - \hat{n}_y^{\text{exact}}|, |\hat{n}_z - \hat{n}_z^{\text{exact}}| < 10^{-4}. \tag{8}$$

The initial testing using plane reconstruction tests was encouraging, in that tight and medium filtering resulted in reliable convergence of the CVTNA outer-loop iteration to the analytical solution. Using the tight filter, a handful of cases barely missed meeting the criterion for planarity preservation (less than 0.02% of the total)—all of them near-homogeneous cell cases (deviations away from 0 or 1 to a few decimal places) that disappear with fine-tuning or the application of a "usable numbers" definition such as introduced by Swartz [26]. The medium filter is only slightly less performed. The loose filter results in oscillatory convergence arising more often and clear deviation away from reliable planarity preservation, while the interface reconstruction computations that were non-oscillatory converged to the correct solution. Having seen the effect of not excluding unsuitable triplets on CVTNA outer-loop convergence behavior, it was necessary to test the different filtering options on a curved interface reconstruction problem.

Hollowed sphere: The hollowed sphere test is a 3D variant of the ubiquitous 2D circle test that features both convex and concave surfaces. In the test, a sphere of radius 0.4 is initialized in a unitary domain, and a spherical core of radius 0.2 is then hollowed out of it. Both concave and convex surfaces are smooth, making the test ideal for determining the order of accuracy of interface reconstruction schemes. An L_1 error norm is used, which in the continuous limit is the integral [23]

$$L_1 = \frac{1}{L}\int\int\int |C(x,y,z) - \tilde{C}(x,y,z)|\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z, \tag{9}$$

Table 1
L_1 errors in interface reconstruction in the hollowed sphere test, using the Youngs gradient and CVTNA schemes

Mesh size	CVTNA loose filter	CVTNA medium filter	Youngs gradient
10^3	2.06×10^{-2}	2.03×10^{-2}	3.27×10^{-3}
	3.78	*3.69*	*1.78*
20^3	1.50×10^{-3}	1.57×10^{-3}	9.53×10^{-4}
	2.57	*2.17*	*1.42*
40^3	2.52×10^{-4}	3.50×10^{-4}	3.56×10^{-4}
	1.99	*2.22*	*1.15*
80^3	6.36×10^{-5}	7.49×10^{-5}	1.60×10^{-4}
	1.70	*2.13*	*1.06*
160^3	1.95×10^{-5}	1.71×10^{-5}	7.70×10^{-5}

The computed orders of accuracy are in italics between mesh entries.

where $\tilde{C}(x,y,z)$ and $C(x,y,z)$ are the color functions for the analytical and reconstructed topologies, respectively. Multiple tests are performed with the fluid of the spherical core alternating between $C = 0$ and $C = 1$, and the L_1 errors averaged; in this manner, the numbers of interface reconstructions of convex and concave surface are equal. The Youngs gradient scheme, CVTNA with loose filter, and CVTNA with medium filter, are the interface reconstruction schemes used in the hollowed sphere test.

Table 1 shows the errors resulting from the hollowed sphere reconstruction tests. The Youngs gradient scheme results converge to first-order quickly, while second-order accuracy is sustained for CVTNA when moderate filtering is used. In the case of CVTNA with loose filtering, first-order effects creep in at higher mesh resolutions, demonstrating the importance of unsuitable triplet exclusion for ensuring high-order accuracy.

In terms of absolute accuracy, while the error associated with CVTNA is many times smaller than that associated with the Youngs gradient scheme at high mesh resolutions, the Youngs gradient scheme is far more accurate at very coarse mesh resolutions. In the CVTNA schemes, the accuracy gains from using more stringent triplet exclusion are realized at relatively high resolutions. For interface reconstruction in generic problems, an adaptive kernel is recommended, combining Youngs gradient for high-curvature interfaces, CVTNA for better-resolved interfaces, and curvature-dependent filter modification.

For the hollowed-sphere test, CPU timings were obtained to determine the effort required to achieve higher-order planarity preservation using CVTNA, as compared to the Youngs gradient scheme. The same scheme used for computing distance is used in both the Youngs gradient and CVTNA schemes, and the tests were done on the same serial platform. The results are shown in Table 2. The CVTNA scheme requires about fifty times the computational effort needed by the Youngs gradient scheme. This is a moderate cost for achieving second-order accuracy, and seems competitive compared to the 3D ELVIRA alternative of Miller and Colella [29].

Table 2
Relative single-processor CPU timings, using the Youngs gradient method and the CVTNA method (using Filter 1): (a) relative to coarse-grid Youngs gradient timing, and (b) relative to Youngs gradient timing for the same mesh size

	Mesh	CVTNA	Youngs
(a)	10^3	65.65	1.00
	20^3	234.84	4.07
	40^3	747.39	15.99
	80^3	2831.47	64.77
(b)	10^3	65.65	1.00
	20^3	57.66	1.00
	40^3	46.74	1.00
	80^3	43.71	1.00

3.4. Upgrading of existing 3D VOF schemes to CVTNA

Beyond the 3D extensions of the earliest VOF methods (SLIC [6] or Hirt-Nichols VOF [7]), the Youngs gradient is the most commonly used scheme for interface reconstruction in 3D VOF-based CFD and multi-physics codes. A major advantage of CVTNA is that the upgrade from the Youngs gradient scheme to CVTNA is relatively straight forward, thus allowing the benefits of second-order planarity-preserving VOF to be easily achieved. Looking at the individual operations in the algorithm for the stencil-based computation in Fig. 5, the Youngs gradient normal estimate and the Brent's method root-finder for plane constant ρ are well-established operations that are commonly implemented in existing VOF codes. The remainder of the operations required to carry out CVTNA—line-plane intercepts, cross-products, coordinate translations, and arithmetic averaging—are relatively trivial by comparison.

In the case of a sweep through the mesh for Youngs gradient interface reconstruction coming across a mixed cell, a $3\times3\times3$ array of C data about cell (i, j, k), along with the accompanying mesh cell dimensions data, is scattered as input to the stencil-level interface reconstruction object. The CVTNA scheme uses the same $3\times3\times3$ stencil used by the Youngs gradient scheme, therefore requiring no change to the VOF module source code, beyond the addition of a new stencil-level interface reconstruction object. Importantly, in the case of load-balanced parallel implementations, any existing message-passing infrastructure developed for Youngs gradient-based VOF remains adequate for CVTNA. [For contrast, in the case of an upgrade to a scheme that works on a $5\times5\times5$ stencil, input arrays for stencil-level interface reconstruction objects must be broadened, and existing message-passing infrastructure may need to be modified to accommodate the wider stencils.] Finally, recognizing the Youngs gradient scheme as an initial estimate of CVTNA, it is straight-forward to implement an adaptive Youngs-gradient/CVTNA kernel, to combine the advantage of the Youngs gradient scheme in high-curvature interface reconstruction with the planarity-preserving quality of CVTNA in the lower-curvature majority of interfaces.

4. Interface advection and flux computation

4.1. Introduction

In volume tracking, advective fluxes are defined from geometric interpretations, whether explicitly as advection flux volumes across mesh cell faces (referred to in [36] as flux-based Eulerian methods), or obtained as remapped volumes from polygon intersection operations using the underlying and the Lagrangian mesh (referred to in [36] as Lagrangian–Eulerian (LE) methods). A more usual distinction of volume tracking advection relates to the number of stages required to complete time integration in multiple spatial dimensions, where a "stage" consists of one interface reconstruction mesh sweep and one flux computation mesh sweep. Fluxes that take into account velocity vector components acting in all directions are fully multidimensional, and the single-stage time integration $C^n \rightarrow C^{n+1}$ makes the method an unsplit advection scheme. On a 3D orthogonal mesh, assuming a solenoidal velocity field, the discretization of Eq. (6) becomes

$$C^{n+1}_{i,j,k} = C^n_{i,j,k} - \frac{1}{V_{i,j,k}}\left(F_{i+1/2,j,k} - F_{i-1/2,j,k} + F_{i,j+1/2,k} - F_{i,j-1/2,k} + F_{i,j,k+1/2} - F_{i,j,k-1/2}\right). \tag{10}$$

In an alternative approach, one-dimensional fluxes can be generated by considering individual velocity vector components in each stage, and a three-stage "direction split" update $C^n \rightarrow C^* \rightarrow C^{**} \rightarrow C^{n+1}$ is the minimum required to resolve the advection in all three spatial directions. Relative to unsplit advection, direction-split volume tracking is well documented [18,24,25,28]. While easy to implement in 3D, direction splitting is a procedure that can be geometrically distorting, that results in a loss of symmetry [24], and contributes a "direction splitting error" [37]. Direction splitting requires at least three interface reconstruction sweeps to complete a volume tracking update. Unless an unsplit advection scheme is grossly inefficient, second-order accuracy in volume tracking using direction splitting will always be significantly more expensive than in volume tracking using unsplit advection.

4.2. A new 3D unsplit advection scheme for volume tracking

4.2.1. Introductory remarks

As in the case of high-order interface reconstruction, the development of unsplit advection schemes for 3D has lagged behind developments in 2D volume tracking. Extension of most of the published 2D schemes to 3D is not feasible at the moment, due to disproportionately large increases in mathematical complexity and computation. Fig. 6 gives a geometric interpretation of several published 2D advection schemes, namely those of Mosso [27,35] and Shahbazi et al. [36] (Fig. 6(a)), Pilliod and Puckett [23] (Fig. 6(c)), Rider and Kothe [24] (Fig. 6(d)), and Garrioch and Baliga [33,34] (Fig. 6(e)). Other approaches can be

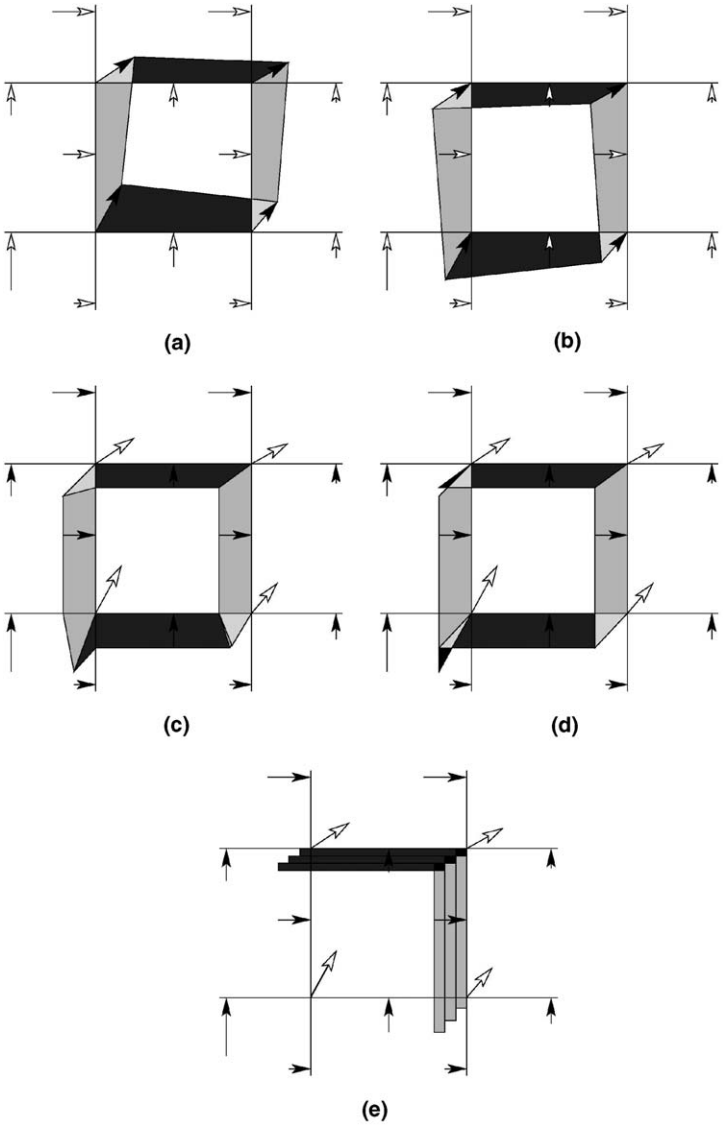

Fig. 6. 2D advection in volume tracking using documented multidimensional flux definition concepts: (a) Lagrangian [27], (b) Eulerian, (c) first method of Pilliod and Puckett [23], (d) method of Rider and Kothe [24], and (e) sub-advection method of Garrioch and Baliga [34]. Velocity vector locations used in constructing the fluxes have solid arrow heads.

found in Aulisa et al. [38], Lopez et al. [32], and Miller and Colella [29]. The 2D unsplit advection schemes based in remapping (e.g. [24,27,35,36]) are most amenable to 3D.

On 3D meshes consisting of Cartesian (hexahedral) mesh cells and four-edged faces, tracing along characteristics from the vertices of cell faces yields multidimensional remapped volumes that are also hexahedral. The volume an arbitrarily oriented plane truncates in a hexahedral shape—the core geometric task of computing the C-fluid fluxes in Eq. (10)—can be done using concise, efficient geometric formulae and operations. The schemes based on the rigorous full-remap therefore allow single-stage volume tracking to be achieved accurately and economically. The multidimensional total-fluid fluxes from the method of Rider and Kothe [24] are not equivalent to the remapped volumes from rigorous full-remapping, but the departure from the rigorous definition is small and particularly useful (as will be demonstrated later). We therefore present a single, general remapping-based framework for unsplit advection in 3D volume tracking, before discussing implementation specifics. Mosso has had success in directly extending the 2D Lagrangian full-remap method described in [27,35] to 3D; as yet unpublished, we recognize that work as the first successful implementation of unsplit advection for single-stage volume tracking in 3D. In the same timeframe, we worked on developing alternative 3D unsplit volume tracking advection schemes using fully multidimensional flux definitions; this work is described below.

4.2.2. Hexahedral definition of 3D multidimensional fluxes

The example illustrated in Fig. 7(a) shows an orthogonal mesh cell, and its right face, consisting of vertices rnb

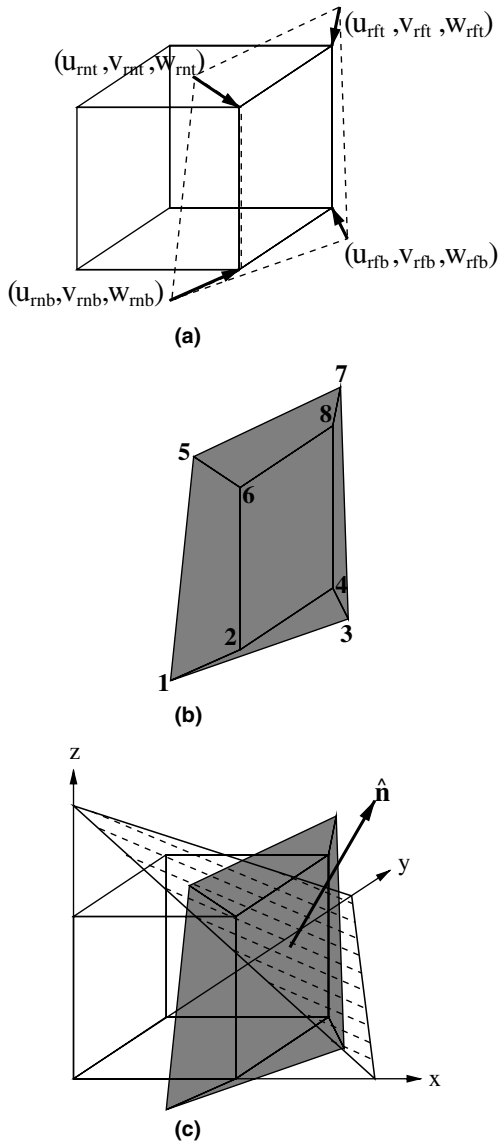

Fig. 7. Flux computation problem of unsplit–advection volume tracking. (a) Every mesh cell face is traced back along Lagrangian trajectories for time δt. (b) The total flux volume enclosed by the vertices is a logical cube. (c) A planar interface reconstruction (dotted triangle) truncates the total flux volume; the volume on the opposite side to the interface normal is the C-fluid flux. Note: the displayed interface reconstruction is only valid for the portion of the total flux volume intersecting the volume of the underlying mesh cell.

(δx, 0, 0), rfb (δx, δy, 0), rnt (δx, 0, δz), and rft (δx, δy, δz). Tracing from each vertex of the cell face, the opposite face of the total flux volume across the right cell face will consist of the following vertices: near bottom ($\delta x \pm u_{\mathrm{rnb}}\delta t$, $0 \pm v_{\mathrm{rnb}}\delta t$, $0 \pm w_{\mathrm{rnb}}\delta t$), far bottom ($\delta x \pm u_{\mathrm{rfb}}\delta t$, $\delta y \pm v_{\mathrm{rfb}}\delta t$, $0 \pm w_{\mathrm{rfb}}\,\delta t$), near top ($\delta x \pm u_{\mathrm{rnt}}\delta t$, $0 \pm v_{\mathrm{rnt}}\,\delta t$, $\delta z \pm w_{\mathrm{rnt}}\delta t$), far top ($\delta x \pm u_{\mathrm{rft}}\delta t$, $\delta y \pm v_{\mathrm{rft}}\delta t$, $\delta z \pm w_{\mathrm{rft}}\delta t$). The use of the $\pm$ sign reflects a dependence on the direction of the trajectory used: the sign is negative if the vertices are traced backwards along Lagrangian trajectories, and is positive if forward trajectories are used. Fig. 7(b) illustrates the flux volume generated for the case of backwards tracing.

The volume in Fig. 7(b) is hexahedral—a logical cube, consisting of six faces, eight vertices and twelve edges. An important property of this logical cube is that planes coplanar with mesh cell faces can, at most, split the total flux volume into two smaller parts that themselves remain logical cubes, i.e. volumes with greater than six faces are not formed. Fig. 7(c) illustrates the core problem of unsplit-advection volume tracking, that is, computing the volume of a multidimensional flux volume truncated by an arbitrarily-oriented interface reconstruction plane.

4.2.3. Backward- vs. forward-trajectory remapping, Eulerian vs. Lagrangian unsplit advection

In the case where mesh vertices are traced by a unique (given) cell-vertex velocity field, hexahedral flux volumes need not be explicitly constructed, but rather remapped between the underlying mesh and the Lagrangian mesh using intersection operations. Starting from the basis of equivalence between explicit advective flux definition and remapping, advection in volume tracking can be achieved in two basic ways. In *Lagrangian advection* using *forward-trajectory remapping* [27,35,36,39], cell volumes on the underlying mesh are projected forward along Lagrangian trajectories. Interface reconstructions are performed in the cells on the new Lagrangian grid, and this information is then transferred back to the underlying mesh. Alternatively, in *Eulerian advection* using *backward-trajectory remapping* [39] (illustrated in Fig. 6(b)), interface reconstructions are performed in cells on the underlying mesh. The update of C is completed by integrating the distribution of C on the underlying mesh, using as control volumes the cells on the traced-back Lagrangian mesh.

Even if the underlying mesh is orthogonal, cells on the Lagrangian mesh are usually distorted, resulting in cell-faces that are non-planar ruled surfaces. In the case of the forward-trajectory remapping option, interface reconstruction on the Lagrangian mesh and subsequent remapping of volumes back to the target mesh become non-trivial geometric problems. The option of using a framework featuring simpler geometry is therefore promoted here.

Eulerian unsplit advection using backward-trajectory remapping can be interpreted as a flux-based method, with the flux volumes being the parts of the truncated cell volumes on the underlying mesh not overlapped by the corresponding cells on the old Lagrangian mesh. In the case of a Cartesian underlying mesh and rigorous adherence to backward-trajectory remapping, interface reconstruction is eliminated as a source of non-planar ruled surfaces, but such surfaces remain in the flux computation. However, small and simple departures away from rigorous adherence can eliminate non-planar ruled surfaces from flux volume definitions altogether. For this primary reason, backward-trajectory remapping is adopted here as the framework for our flux-based Eulerian unsplit advection scheme.

4.2.4. Contributions to the total flux volume

Tracing characteristics back from the vertices of the cell face will usually result in the total flux volume across that face spanning multiple cells on the underlying mesh. In our Eulerian method, piecewise planar interface reconstructions are only valid for individual mesh cells, which means that flux calculations using a particular interface reconstruction can only consider the portion of the total flux volume residing in that cell.

Incremental remapping contributions to the flux across a given cell face on a 3D orthogonal mesh are restricted to an 18-cell stencil about that face. Taking the example of the flux of C across the $(i+1/2, j, k)$ cell face, the flux is determined by

$$F_{i+1/2,j,k} = \sum_{kk=-1}^{1} \sum_{jj=-1}^{1} \sum_{ii=0}^{1} V^{\mathrm{tr}}_{i+ii,j+jj,k+kk}, \tag{11}$$

where V^{tr} is the volume truncated by the planar interface reconstruction in any stencil cell.

To decompose a total flux volume amongst underlying mesh cells, planes of infinite extent that are coplanar with mesh cell faces (henceforth referred to as cell-face planes) are used in a process of vertex replacement. The intersection of a total flux volume edge with a cell-face plane becomes a new vertex. Given a specific stencil cell of interest, original vertices of the total flux volume are retained if they are cell-side of the cell-face plane, and discarded if they not cell-side. The result of the decomposition is multiple *total flux volume contributions*; truncation computations ultimately use these contributions, rather than the full total flux volume.

As an example, consider the case in Fig. 8(a), in which fluid from the nine bordering cells to the left of face $(i+1/2, j, k)$ is forced through the face. Fig. 8(b) shows Fig. 8(a) looking in the positive x-direction. To isolate the contribution that lay within cell (i, j, k), we first identify the mesh cell face that was traced backwards along characteristics to form the total flux volume in the first place—in our case the $(i+1/2, j, k)$ wall face 2486. The cell-face plane coinciding with face 2486 is used in the first decomposition of the total flux volume, in order to decouple potential "bow-tie" surfaces. In this case the $(i+1/2)$ cell-face plane is used, and no vertices are replaced.

After bow-tie decoupling, vertex replacement proceeds with all remaining cell-face planes, one plane at a time. Decomposing with the $(j-1/2)$ cell-face plane, vertex 1 is replaced by the intersection of the cell-face plane with edge 15, and vertex 3 is replaced by the intersection of the cell-face plane with edge 37. The vertex replacements made using the $(j-1/2)$ cell-face plane are shown in Fig. 8(c), along with the replacements using the $(j+1/2)$ plane of the opposing cell-face. The $(k-1/2)$ and $(k+1/2)$ cell-face planes are similarly applied to complete the decomposition, with new vertices being formed by intersections along edges 13 and 57; the end result is shown in Fig. 8(d).

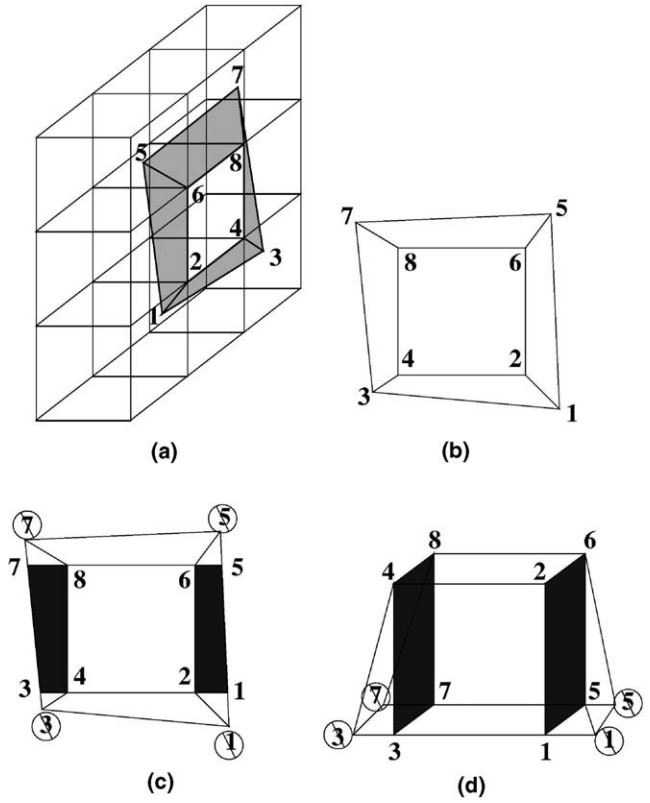

Fig. 8. Vertex modification to isolate contributions to the total flux volume that come from individual mesh cells: (a) the fully multidimensional 3D flux, (b) view of flux normal to the mesh cell face, (c) planes coplanar with the mesh cell faces other than the flux face and coplanar with lines 24 and 68 truncate the initial total flux, modifying vertices 5 and 7, (d) planes coplanar with the mesh cell faces other than the flux face and coplanar with lines 26 and 48 truncate the intermediate total flux from (c), modifying vertices 1, 3, 5 and 7.

4.2.5. Flux-surface definition

Rigorous adherence to the full-remap uses a unique cell-vertex velocity solution $\mathbf{U}_v = (u_v, v_v, w_v)$ to generate a Lagrangian mesh that is logically orthogonal, and consisting of no gaps or overlaps between adjacent traced-back cells. Fig. 9(a) and (b) show cases of bow-tie surfaces resulting from backward tracing using cell-vertex vectors that point in different directions relative to the flux face. In both cases, it is apparent that most faces of the total flux volumes are non-planar ruled surfaces.

The use of cell-vertex velocity vectors from any unique solution to $\mathbf{U}_v$ will usually result in the total flux volume having five non-planar faces. In the case of a unique velocity solution on a staggered mesh, the known normal velocity at any cell-face centroid may simply be assigned to its vertices; this represents a deviation from strict adherence to remapping. Tracing characteristically using such non-unique estimates of the velocities at cell-face vertices results in total flux volumes featuring only planar faces, as shown in Fig. 9(c). We refer to multidimensional flux calculation using cell-face velocities assigned to cell-vertices as the *Piecewise-Constant Flux Surface Calculation* (PCFSC). On the surface, the PCFSC scheme looks like a 3D

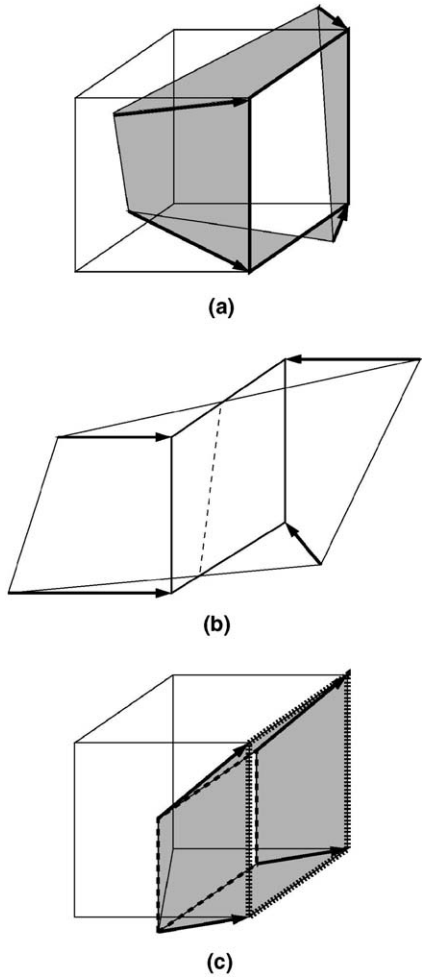

Fig. 9. 3D total flux volume definitions obtains by tracing backwards characteristically: (a) traces back using a unique cell-vertex velocity field, which can result in bow-tie flux volumes and surfaces (shown more dramatically in (b)); (c) demonstrates the less distorted, planar-faced total flux volume using the PCFSC scheme, with the flux surface (dotted) being parallel to the mesh cell flux-face (railed).

extension of the 2D volume tracking unsplit advection scheme of Rider and Kothe [24].

Our main motivation for presenting PCFSC here is that simpler, published methods and formulae can be used for computing C-fluid fluxes. Appendix A presents a conceptually simple framework for computing C-fluid fluxes within the PCFSC unsplit advection method. PCFSC can also be run as a special case of unsplit advection implementations using strict backward-trajectory remapping, with a switch used to change between cell-vertex and cell-face velocity inputs.

4.3. Introduced local extrema: undershoots, overshoots and wisps

When applied to problems other than translation, most of the known 2D volume tracking methods result in generation of undershoots ($C^n \to C^{n+1} < 0$) and overshoots ($C^n \to C^{n+1} > 1$) after time integration (Eq. (10)). Another form of instability that cannot be detected as easily as undershoots and overshoots is the case of C in cells away from the interface deviating from 0 or 1. New extrema in the distribution of C are introduced in such cases, appearing as flotsam and jetsam (referred to in [40] as "wisps") that destroy the convergence of the scheme. Introducing new local extrema can also represent a consistent under- or over-prediction of C-fluid fluxes, that can result in an accumulation of phase error.

Rewriting the evolution equation (Eq. (10)) as a change in C (and assuming $\nabla \cdot \mathbf{U}_f = 0$),

$$C^{n+1} = C^n + \delta C, \tag{12}$$

undershoots occur if $\delta C > 0$ and $|C^n| < \delta C$, and overshoots occur if $\delta C > 0$ and $C^n > 1 - \delta C$. One cause for computed $\delta C \neq 0$ away from interface cells is that the divergence, based on geometric interpretations of total fluid fluxes, is non-zero:

$$\sum_{\text{faces}} F^{\mathrm{T}} \neq 0. \tag{13}$$

Other causes of undershoots and overshoots include inaccurately shaped fluid fluxes, including the case of adjacent fluid fluxes using non-unique Lagrangian trajectories resulting in fluid being fluxed twice or not at all [24].

At this stage, there is no mechanism in the PCFSC scheme to ensure δC is reconstructed in a manner that inherently prevents new local extrema in C being introduced. Unsplit advection in 3D based on the full-remap, or a 3D extension of the 2D Edge Flux Polygon Matching scheme [32], may ameliorate the introduction of new local extrema to a certain extent. However, the merits and implementation of these alternatives are moot points, if we wish to avoid consideration of non-planar surfaces in 3D.

As a treatment to handle the source of undershoots and overshoots described by Eq. (13), we assert that total flux volume estimates should be corrected in proportion to the under/over-estimation compared to the correct total flux volume. The numerical velocity divergence correction [24] does not adequately discriminate between good and poor total flux volume estimates. Excessively correcting good flux estimates, and not correcting poor flux estimates enough, was seen to result in an inadequate amelioration of phase error; the prescription of Rider and Kothe [24] is therefore not used here.

An alternative approach developed here for handling multidimensional flux estimates generated using spatially-varying velocity fields is to use a scaling procedure for all fluxes. The proposed scaling resembles that proposed by Lopez et al. [32] for near-parallel flow in high-CFL number simulations. In our case, the effective C-fluid fluxes used in Eq. (10) take the form

$$F_{i+1/2,j,k} = f F^{*}_{i+1/2,j,k}, \tag{14}$$

where

$$f = 1 + \frac{|u_{i+1/2,j,k}\delta t \delta y_j \delta z_k| - |F^{\mathrm{T}}_{i+1/2,j,k}|}{|F^{\mathrm{T}}_{i+1/2,j,k}|}, \tag{15}$$

and $F^{*}_{i+1/2,j,k}$ is the C-fluid flux generated as the volume of the geometrically defined total fluid flux $F^{\mathrm{T}}_{i+1/2,j,k}$ truncated by the piecewise planar interface reconstruction. With this approach, geometric flux volumes are corrected in proportion to how under/over-estimating they are.

The action of Eqs. (14) and (15) removes undershoots, overshoots and wisps as an issue for the vast majority of interface cells. Global volume conservation has been found, in our early work, to be satisfied to machine error if the integration occurs over the entire solution domain. In reality, we need to confine computations to a thin support about all interface cells for computational efficiency reasons. Localized redistribution and fluid accounting become useful for eliminating any small-scale deviations from global conservation, along the lines of the procedures presented in [40].

4.4. Advection tests

Given the wider context of the work, our 3D volume tracking implementation features two variants of unsplit advection: (i) a backward-trajectory full-remap; and (ii) PCFSC. The PCFSC scheme is the option for which all geometric tools for implementation are presented in this paper: therefore, only the results of the PCFSC variant are presented. All advection tests using the CVTNA scheme use the loose filter from our earlier interface reconstruction tests.

Given the lack of documented benchmarks for 3D volume tracking, we first apply our 3D algorithm to a test suite of current standard tests for 2D interface tracking. These tests also demonstrate that our volume tracking method has complete redundancy from 3D down to 2D; the 2D tests performed here are done using the 3D volume tracking module, on $nx \times ny \times 1$ meshes. Finally, a range of 3D advection tests are presented.

Given all advection tests used here return advected shapes to their original locations, the L_1 error norm

$$L_1 = \sum_{i=1}^{nx} \sum_{i=1}^{ny} \sum_{i=1}^{nz} |C_{i,j,k} - \tilde{C}_{i,j,k}| \delta x_i \delta y_j \delta z_k \tag{16}$$

is used. Variables $\tilde{C}_{i,j,k}$ and $C_{i,j,k}$ are the color functions before and after advection, respectively. [The simplification to 2D is trivial.]

4.4.1. Standard 2D problem suite

Summary findings from simple 2D pure translation tests include: (i) absolutely rigorous local conservation ($f \equiv 1$, no extrema, no redistribution); (ii) unsplit advection is slightly more accurate than direction-split advection; (iii) linearity is preserved at flat interfaces. For more demanding tests of the 3D volume tracking implementation applied to 2D problems, rotating flow field circle advection and the single vortex circle advection test [24] are considered.

Rotating flow field: Inducing no change in interface topology, this test is more useful as a useful verification test for volume tracking implementations, rather than as a demanding test of schemes. The rotating flow field test is equivalent to that performed in [24]: a circular body of radius 0.15 and centered at (0.5, 0.75) is advected for the period of time required for a particle to return to its initial position. In our case, we use CFL numbers of 0.25, 0.5 and 1.0. In these tests, three alternative interface-reconstruction/advection combinations are used: Youngs-gradient + PCFSC unsplit; CVTNA + direction-split; CVTNA + PCFSC unsplit.

Table 3 shows the errors and orders of accuracy that result from the 2D rotating velocity field advection test. First, the choice of interface reconstruction scheme is clearly most important in determining the accuracy of the scheme, confirming the observation of Aulisa et al. [38]. The volume tracking schemes using Youngs gradient for interface reconstruction display first-order accuracy, while our CVTNA method is able to preserve second-order accuracy even after advection. For the case of CFL = 0.5 (used in [24]), the direction-split and PCFSC unsplit advection schemes perform comparably well. Changing the CFL number demonstrates a major difference between unsplit and direction-split advection. As CFL is reduced, PCFSC unsplit advection basically preserves the accuracy of the scheme, while direction-split advection shows more evident degradation. As CFL is increased however, the accuracy of the direction-split scheme improves, more than the PCFSC scheme deteriorates. For CFL = 1.0, PCFSC unsplit advection results in more volume needing to be redistributed, and the donor cell-based trajectory for backward tracing becomes more inaccurate; some degree of formal accuracy loss becomes evident. On the other hand, at lower CFL the backward tracing occurs over shorter distances and are much better estimates of the real trajectory, while the extra number of interface reconstructions performed in direction-splitting result in error being introduced at more points in the simulation.

Table 3
L_1 errors for the advection of a circle for a 2D rotating flow field, and one complete revolution about the center

		CFL = 0.25	CFL = 0.5	CFL = 1.0
CVTNA + direction-split	$32^2 \times 1$	2.09×10^{-3}	1.53×10^{-3}	1.37×10^{-3}
		1.95	*2.01*	*1.95*
	$64^2 \times 1$	5.41×10^{-4}	3.79×10^{-4}	3.54×10^{-4}
		2.03	*2.07*	*2.13*
	$128^2 \times 1$	1.32×10^{-4}	9.03×10^{-5}	8.08×10^{-5}
Youngs + PCFSC unsplit	$32^2 \times 1$	2.25×10^{-3}	1.50×10^{-3}	9.92×10^{-4}
		1.93	*1.90*	*1.62*
	$64^2 \times 1$	5.90×10^{-4}	4.00×10^{-4}	3.22×10^{-4}
		1.32	*1.11*	*1.18*
	$128^2 \times 1$	2.37×10^{-4}	1.86×10^{-4}	1.42×10^{-4}
CVTNA + PCFSC unsplit	$32^2 \times 1$	2.39×10^{-3}	2.03×10^{-3}	1.64×10^{-3}
		2.08	*2.33*	*2.16*
	$64^2 \times 1$	5.65×10^{-4}	4.02×10^{-4}	3.67×10^{-4}
		2.10	*2.03*	*1.76*
	$128^2 \times 1$	1.32×10^{-4}	9.83×10^{-5}	1.08×10^{-4}

The computed orders of accuracy are in italics between mesh entries.

Single vortex: The single vortex velocity field, commonly used in interface tracking testing since used by Rider and Kothe [24], uses the velocity field:

$$u = \sin(2\pi y)\sin^2(\pi x)\cos\left(\frac{\pi t}{T}\right), \tag{17}$$

$$v = -\sin(2\pi x)\sin^2(\pi y)\cos\left(\frac{\pi t}{T}\right). \tag{18}$$

Starting with a circular body of radius 0.15 and centered at (0.5, 0.75), this shearing flow will result in the fluid body stretching and spiraling into towards the center, and the Leveque cosine term returns the advected body to its initial location—a convenient means of establishing temporal accuracy. In this case only CFL = 1.0 is used.

Table 4 shows the resulting errors and orders of accuracy, as well as some benchmark results for the same problem from the wider literature. The results for the CVTNA + PCFSC unsplit combination compare favorably with the literature results, and validate our volume tracking module implementation. The linear-fit scheme of Scardovelli and Zaleski [28] outperforms all of the other schemes, but it is doubtful the scheme extends to 3D readily. The results obtained in this benchmark 2D test verify the ability of PCFSC unsplit advection to sustain the second-order accuracy of CVTNA interface reconstruction, by validating our scaling of geometrically constructed flux volumes to prevent order-degradation.

4.4.2. 3D Problems

As a final test of our 3D volume tracking scheme, 3D shearing flow tests are performed to benchmark the performance of the scheme. In these test flows, three different combinations of interface reconstruction and advection are used: Youngs gradient + direction-split; Youngs gradient + unsplit; CVTNA + unsplit. The combination of CVTNA interface reconstruction and direction-split advection is not used, because the thrice application of CVTNA required with direction-split advection makes tests at high-resolutions too computationally expensive.

3D shearing flow: In this test, the single vortex in the *xy*-plane described by Eqs. 17–18 is combined with laminar pipe flow

$$w = U_{\max}\left(1 - \frac{r}{R}\right)^2 \cos\left(\frac{\pi t}{T}\right), \tag{19}$$

where $x_0 = 0.5$, $y_0 = 0.5$, $R = 0.5$ and $r = \sqrt{(x-x_0)^2 + (y-y_0)^2}$. A sphere of radius 0.15 and centered at (0.5, 0.75, 0.25) is initialized, a value of $U_{\max} = 1.0$ is used, and Leveque's cosine term is used to return the deformed fluid body to its original position.

In the first part of the test, the problem is run for $T = 3$ with a CFL number of 0.5. Fig. 10 illustrates $C = 0.5$ iso-surfaces of the sphere after maximum deformation from two different views. As the resolution increases, the increases in deformations in the swirl and laminar flow directions decelerate toward the grid independent solution to the maximum deformation. In all cases, no fragmentation is simulated. Table 5 shows the errors obtained for the volume tracking schemes in the 3D shearing flow test. When the same interface reconstruction method is used, unsplit advection has a slight advantage in accuracy over direction-splitting. The expected second-order accuracy is achieved when the CVTNA scheme is used.

Isolating consideration to the combination of CVTNA interface reconstruction and PCFSC unsplit advection, Fig. 11 shows the effect on the solution accuracy of repeatedly halving the CFL, for 32^3 and 64^3 runs. In this case, CFL is progressively halved from CFL = 1.0. The increase in error for reducing CFL is a shortcoming that has been identified by Scardovelli and Zaleski [28] as a feature of volume tracking schemes, due more to the larger numbers of interface reconstructions that must be performed, and less as a result of advection error. The 25–30% increase in error seen in Fig. 11 for our 32^3 results is about the same proportional increase seen in [28] for 2D PLIC-VOF schemes on the 32^3 single vortex test. The L_1 error is essentially converged as CFL $\to 0$ on the medium-resolution 64^3 mesh, while it is converging more slowly on the 32^3 mesh. This is due to the advection of larger discontinuities between adjacent planar interface reconstructions, especially in the high-curvature regions of the interface.

On a $64 \times 64 \times 128$ mesh and using CFL = 1.0, more extreme deformations are simulated by running the test for $T = 6$ and $T = 9$. Fig. 12 shows different views of the $C = 0.5$ iso-surface at maximum deformation in both cases. The figure shows the premature fragmentation of the deformed fluid sheets, resulting from numerical surface tension induced by piecewise planar interface reconstruction. The difference in numerical surface tension between the

Table 4
L_1 errors for the advection of a circle for the 2D Rider/Kothe single vortex flow problem, at CFL = 1.0

Mesh	Error (CVTNA + PCFSC unsplit)	Error (Youngs + PCFSC unsplit)	Error (ELVIRA + Rider/Kothe)	Error (linear fit + EI-LE)	Error (ELVIRA + stream)
$32^2 \times 1$	2.34×10^{-3}	2.61×10^{-3}	2.36×10^{-3}	1.75×10^{-3}	2.37×10^{-3}
	2.12	*1.85*	*2.01*	*1.91*	*2.07*
$64^2 \times 1$	5.38×10^{-4}	7.25×10^{-4}	5.85×10^{-4}	4.66×10^{-4}	5.65×10^{-4}
	2.03	*1.66*	*2.16*	*2.19*	*2.10*
$128^2 \times 1$	1.31×10^{-4}	2.29×10^{-4}	1.31×10^{-4}	1.02×10^{-4}	1.32×10^{-4}

The computed orders of accuracy are in italics between mesh entries. The first two columns are results obtained in the current work using the 3D schemes for 2D problems, while the last three columns are comparison results taken from [24,28,40] respectively.

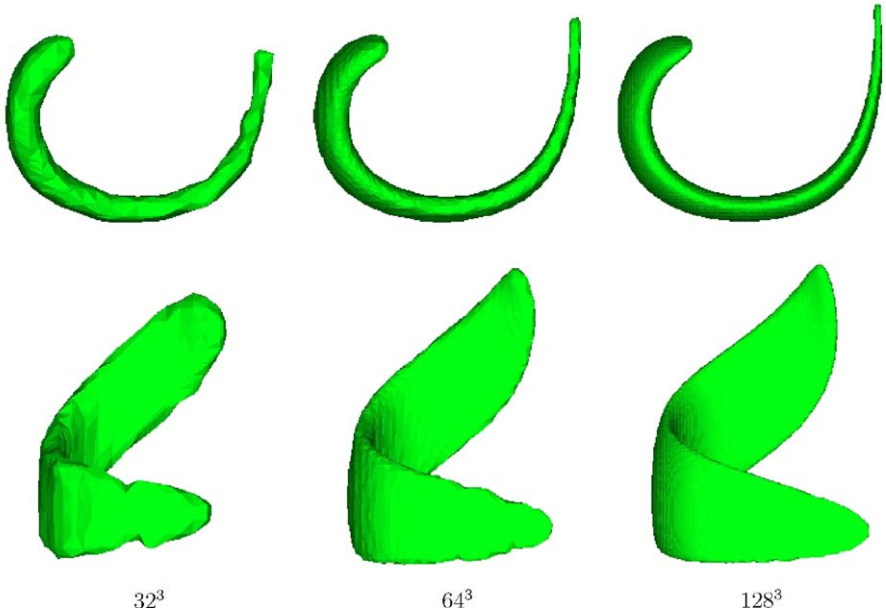

Fig. 10. Profiles at maximum deformation for the 3D shearing flow problem, using $T = 3$ and CFL = 0.5, for successively refined meshes from volume tracking using the CVTNA interface reconstruction scheme and PCFSC unsplit advection. The top row shows the xy-plane view of the $C = 0.5$ iso-surface, while the bottom row shows a side-on view.

Table 5
L_1 errors in volume tracking advection in the 3D shearing flow advection test (single vortex in xy-plane with laminar pipe flow in z-direction), for CFL = 0.5

Mesh	Error (CVTNA + PCFSC unsplit)	Error (Youngs + direction-split)	Error (Youngs + PCFSC unsplit)
32^3	2.86×10^{-3}	3.42×10^{-3}	3.39×10^{-3}
	2.00	*1.59*	*1.60*
64^3	7.14×10^{-4}	1.14×10^{-3}	1.12×10^{-3}
	2.19	*1.60*	*1.63*
128^3	1.56×10^{-4}	3.76×10^{-4}	3.61×10^{-4}

The computed orders of accuracy are in italics between mesh entries.

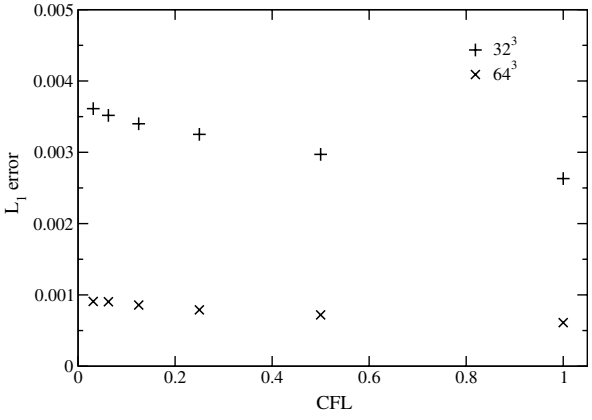

Fig. 11. L_1 error for the 3D shearing flow problem as a function of CFL, for 32^3 and 64^3 meshes.

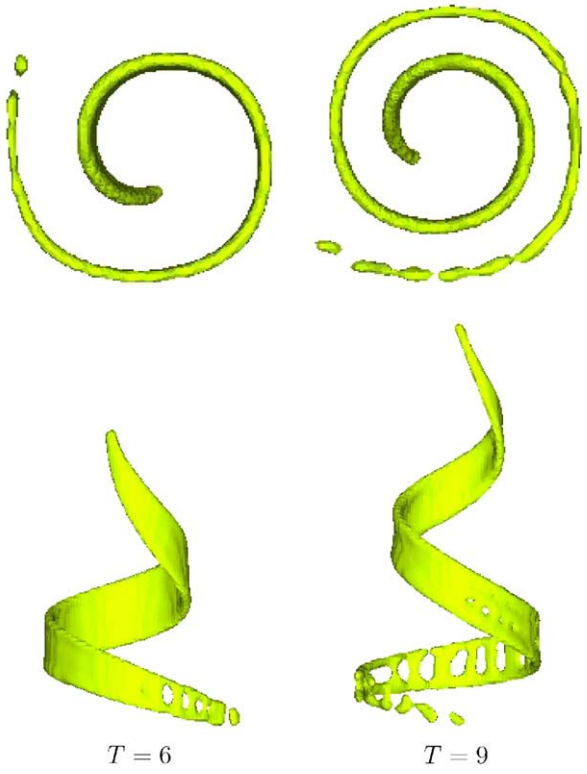

Fig. 12. Profiles at maximum deformation for the 3D shearing flow problem, using CFL = 1.0 and a $64 \times 64 \times 128$ mesh, for the greater deformation cases of $T = 6$ and $T = 9$. The top row shows the xy-plane view of the $C = 0.5$ iso-surface, while the bottom row shows a side-on view.

CVTNA and Youngs gradient schemes is generally not significant. Table 6 shows the results for $T = 3,6,9$ on the $64 \times 64 \times 128$ mesh using CFL = 1.0 upon full flow reversal. While CVTNA interface reconstruction is overall more accurate, the improvement is markedly reduced for greater-deformation runs; numerical surface tension errors

Table 6
L_1 errors in volume tracking advection over increasing deformations ($T = 3$, $T = 6$, $T = 9$) of the 3D shearing flow advection test (single vortex in xy-plane with laminar pipe flow in z-direction), using CFL = 1.0, and the $1.0 \times 1.0 \times 2.0$ domain discretized using a uniform $64 \times 64 \times 128$ mesh

Mesh	Error $T = 3$	Error $T = 6$	Error $T = 9$
Youngs + unsplit	9.99×10^{-4}	4.38×10^{-3}	8.59×10^{-3}
CVTNA + unsplit	6.20×10^{-4}	3.64×10^{-3}	8.01×10^{-3}
Hirt-Nichols + unsplit	3.02×10^{-3}	4.53×10^{-3}	8.36×10^{-3}
Youngs + direction split	1.04×10^{-3}	4.43×10^{-3}	8.59×10^{-3}

in high-curvature regions tend to diminish the general accuracy gains achieved by achieving planarity preservation in interface reconstruction.

For real flow scenarios featuring massively arbitrary fragmentation and coalescence (such as the turbulent bubbling flows in Fig. 1), the best means of reducing the simulation of spurious fragmentation and coalescence is to accurately capture the interfaces that are resolvable, and to increase the proportion of interface length scales in the physical problem that can be resolved on the VOF mesh. The contribution the schemes presented in this paper make in those directions will be reinforced later. Adaptive kernels for using the cheaper Youngs gradient scheme for high-curvature regions, and thin-fluid models, are directions that warrant further work, but are beyond the scope of the current paper.

3D deformation field: The 3D deformation field proposed in [41],

$$u = 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos\left(\frac{\pi t}{T}\right), \tag{20}$$

$$v = -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)\cos\left(\frac{\pi t}{T}\right), \tag{21}$$

$$w = -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos\left(\frac{\pi t}{T}\right), \tag{22}$$

was used in [42] as a final test of their hybrid particle level set method, and clearly highlighted the massive improvement in conservation and accuracy of the hybrid scheme over pure level sets. This flow is also used here as a final test of our 3D volume tracking scheme. As set up in [42], a sphere of radius 0.15 and centered at (0.35, 0.35, 0.35) was advected by the field for a period of $T = 3$. Given the significant thinning of the sphere at maximum deformation, we expect this to be a challenging test of volume tracking at lower mesh resolutions.

Fig. 13 shows the CVTNA interface reconstructions for the 3D deformation field test on 128^3 and 256^3 meshes, at

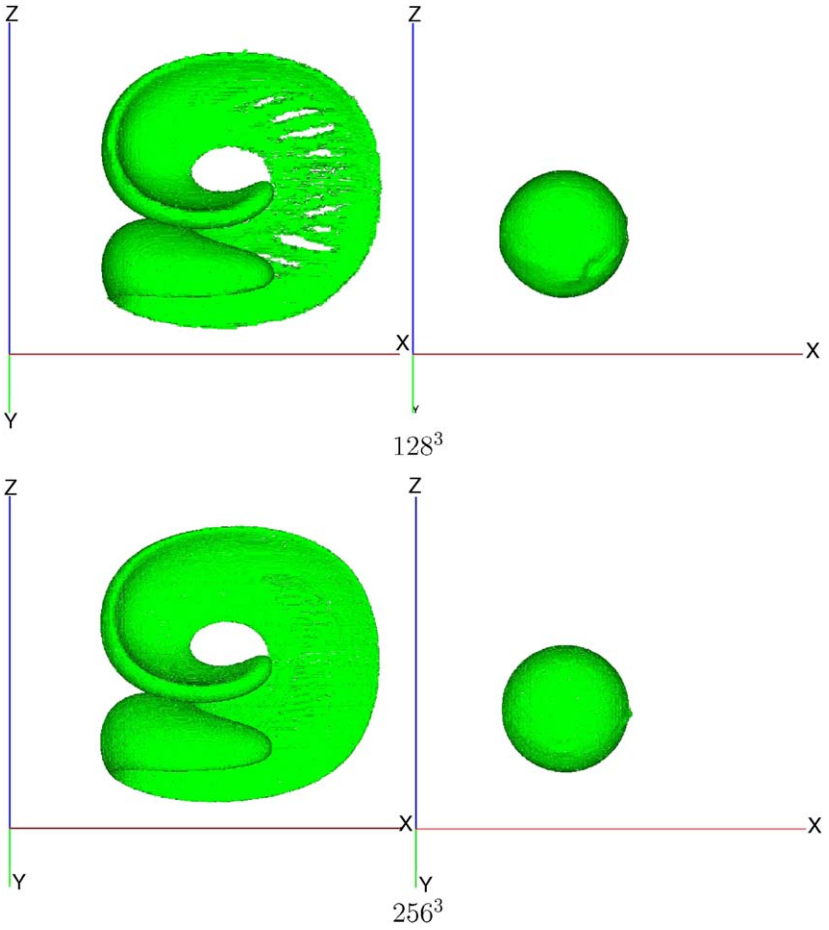

Fig. 13. CVTNA-plane interface representation at maximum deformation ($t = 1.5$), and after full flow reversal ($t = 3$), from the 3D deformation field advection test performed at CFL = 0.5 on 128^3 (top) and 256^3 (bottom) grids, using PCFSC unsplit advection.

Table 7
L_1 errors in volume tracking advection in the 3D deformation field advection test, for CFL = 0.5,using CVTNA interface reconstruction with PCFSC for unsplit advection, and using Youngs interface reconstruction with direction-splitting for advection

Mesh	Error (CVTNA + PCFSC unsplit)	Error (Youngs + direction-split)	Error (Youngs + PCFSC unsplit)
32^3	7.41×10^{-3}	7.71×10^{-3}	7.86×10^{-3}
	1.90	*1.47*	*1.43*
64^3	1.99×10^{-3}	2.78×10^{-3}	2.91×10^{-3}
	2.69	*1.87*	*1.98*
128^3	3.09×10^{-4}	7.58×10^{-4}	7.36×10^{-4}
	2.14	*1.68*	*1.70*
256^3	7.03×10^{-5}	2.37×10^{-4}	2.26×10^{-4}

The computed orders of accuracy are in italics between mesh entries.

maximum deformation and upon flow reversal. Our use of CVTNA reconstructions for visualization shows the appearance of the interface as seen by the volume tracking schemes, as well as by VOF-augmented momentum advection schemes. Single-stage VOF through unsplit advection allowed the use of the CVTNA interface reconstruction scheme for generating the impressive high-resolution results. Smaller computations result in fragmentation induced by numerical surface tension; holes in the extruded centre of a single deformed fluid body in the 128^3 simulation are seen to be replaced in very low-resolution cases by a gulf between two separated bodies centered at the leading and trailing "poles". Even at low resolutions, the VOF algorithm ensures volume is conserved, thereby allowing volume tracking on extremely coarse meshes (e.g. 32^3) to outperform a pure level-set scheme run on a 100^3 meshes [42].

Table 7 shows the L_1 errors for the 3D deformation field problem associated with each of the tested combinations of interface reconstruction and advection. The inaccuracy of the coarse-mesh results is an expected result of the spurious fragmentation caused by numerical surface tension occurring in the early stages of the simulation. The results of our new scheme show the expected second-order behavior, while the results with the Youngs gradient-based scheme tend toward first-order accuracy. The comparison between advection schemes using the same Youngs-gradient interface reconstruction scheme shows the differences in accuracy to be negligible: PCFSC unsplit advection is more accurate at larger mesh sizes, and less accurate at smaller mesh sizes.

5. Flow solver timings

Thus far in the current work, unsplit advection has been shown to be slightly better performed than direction-splitting in terms of solution accuracy on the same problems and using the same meshes. To complete the assessment of the advances we have made in advection and interface reconstruction for volume tracking, a VOF module consisting of the PCFSC and CVTNA schemes was incorporated into a standard incompressible CFD flow solver for multi-material flow, for the purpose of CPU timings. Apart from the volume tracking module, the flow solver features a two-step projection solution algorithm, a 3D version of the fully kernel-based Continuum Surface Force method of Rudman [18], van Leer momentum advection, and a highly efficient multigrid-preconditioned GMRES solver for the Poisson equation.

The VOF-based flow solver was applied to the problem of falling drops, in which a $18D\times18D\times9D$ domain was filled with air, and water drops of diameter D (in the current case $D\approx3$ mm) may be initialized at x- and y-coordinates of $6.9D$, $9.0D$, and $11.1D$, and at z-coordinates of $2.0D$, $4.1D$ and $6.2D$. In the problem therefore, up to 27 drops, configured in a $3\times3\times3$ array, can be initialized to fall through the domain under gravity. The problem is run for a few timesteps until the timings are stabilized, and then averaged. Starting with the 27-drop array, rows of drops are subsequently eliminated, and new timings obtained. With minimal drop deformation and no merging, the interfacial area is well defined and monotonically reduces with reducing drop count, allowing the scaling of CPU timings with surface area to be ascertained.

Fig. 14 shows the timings obtained for the falling drop problem on $64\times64\times32$ and $128\times128\times64$ meshes, respectively. The Poisson solver was the most time-consuming part of the solution algorithm in all cases, and the volume tracking was the second most time-consuming part. These observations are consistent with those from the use of other multi-material flow solvers, although the expense of surface tension has been curtailed here by kernel smoothing using a two-cell thick smoothing length; in the case of the smoothing length increasing to three cells, the computational expense of the surface tension modeling competes with the Poisson solver. The linear scaling of the CPU time spent in volume tracking suggests that for problems featuring large interfacial areas (relative to dispersed phase volumes), the computational expense of the volume tracking may approach that of the Poisson solver. The results in Fig. 14 are a clear motivation for efficient volume tracking, although the $1/\delta x^2$ scaling of the number of interface cells that need to be treated by VOF is better than the $1/\delta x^3$ scaling of the mesh size.

Focusing on the volume tracking algorithm, the comparison of the timings for the CVTNA interface reconstruction and the PCFSC unsplit advection scheme is particularly noteworthy. The results confirm the assertion that high-order interface reconstruction is more expensive than unsplit advection—on average three to five times more expensive. Direction-split advection is understood to be less computationally expensive than unsplit advection, because of the simplicity of the flux volumes that need to be constructed. However, the computational savings from using direction-split advection are only realized if an inexpensive interface reconstruction scheme is used, such as the Youngs gradient scheme. In the case of

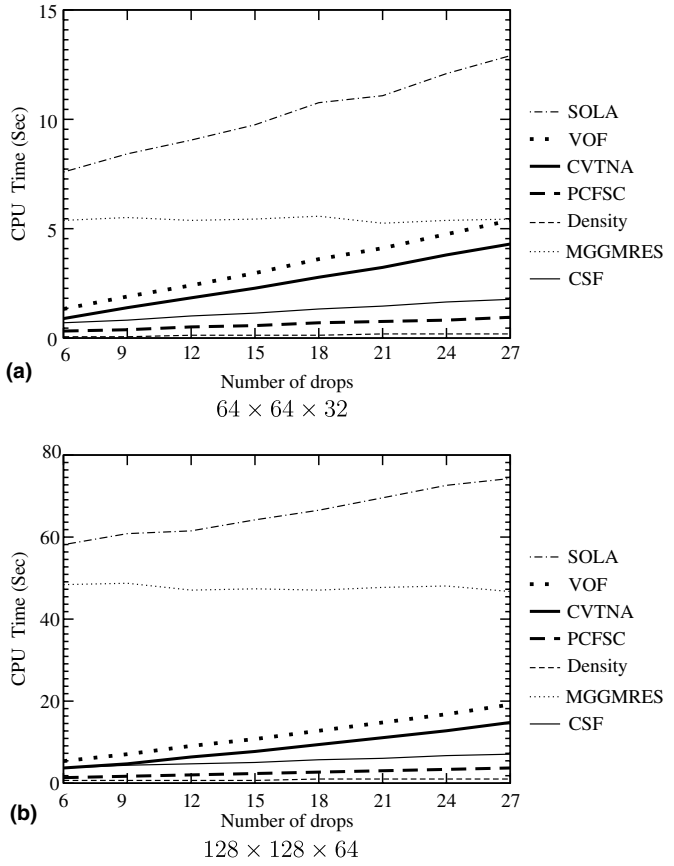

Fig. 14. Breakdown of CPU times spent in different modules of a standard incompressible multi-material Navier–Stokes solver, for a problem of drops falling under gravity, as a function of the number of drops (hence interfacial area). "SOLA" refers to one application of the two-step projection solution algorithm, "VOF" refers to the new volume tracking algorithm featuring CVTNA interface reconstruction and PCFSC unsplit advection (individually timed as well), "Density" refers to the decomposition of VOF interface reconstructions to compute cell-face density estimates, "MGGMRES" refers to our multigrid-preconditioned GMRES solver for the Poisson equation, and "CSF" refers to the kernel-based CSF surface tension model developed by Rudman [18].

high-order volume tracking, multiple application of the interface reconstruction scheme accompanying direction-split advection decimates the efficiency of the VOF module in the flow solver. For the case of 27 drops on the $128 \times 128 \times 64$ mesh, the one-time application of CVTNA facilitated by unsplit advection allows the VOF module in the code to be executed in 19 seconds, while VOF using a three-time application of CVTNA in conjunction with direction-split advection cannot be completed in under 45 seconds. The timings for direction-split advection would be even worse if an alternative, more expensive scheme for high-order interface reconstruction were used. In the case of the CPU timings for the entire solution algorithm being considered acceptable using direction-split VOF, the computational savings from unsplit advection can go towards improvements in accuracy—specifically, mesh refinement (fixed or adaptive) to ameliorate numerical surface tension, and sub-advection for highly vortical flows.

6. Conclusion

A 3D interface reconstruction scheme based (in spirit) on the 2D idea proposed by Swartz [26] has been implemented here, with no disproportionate increase in complexity or computation in upgrading from 2D to 3D. The new scheme, which we refer to as CVTNA, results in interface reconstruction that iteratively tends towards plane preservation. The CVTNA scheme is shown to be second-order accurate and planarity-preserving, and represents a substantial advance in volume tracking interface reconstruction, given the past history of documented 2D schemes not extending readily to 3D.

The PCFSC unsplit advection scheme presented here preserves the spatial accuracy from the interface reconstruction even after advection, and features accuracy on moderate/large meshes that is superior to direction-splitting. Scaling of multidimensional fluxes with exact, conservative, one-dimensional fluxes, is used to preserve monotonicity in time integration and to minimize the introduction of phase error in advection. Departure from the strict definition of backward-trajectory remapping, in assigning cell-face velocities to cell vertices, allows for multidimensional flux computation using simple geometric concepts and documented mathematical formulae. Unsplit advection allowed all large-mesh advection tests to be completed in this work, whereas direction-splitting was found to make CVTNA-based advection tests too computationally expensive and infeasible. The main promise of unsplit advection—making second-order accuracy feasible through the use of single-stage volume tracking—has therefore already been realized here.

Acknowledgments

Computations were performed on the SGI supercomputer of the High Performance Computing Center at James Cook University. Also thanks to Prof. G. Yadigaroglu and Chidu Narayanan (ETH-Zurich) for their guidance and feedback on the manuscript. Special thanks also to Jay Mosso (LANL) for his insights into remapping-based volume tracking. Finally, thanks to Chuck Zemach (LANL), whose novel formulae for truncated-volume computations in arbitrary hexahedra provide much of the inspiration towards making full remap-based volume tracking in 3D a reality, and which are used in various implementations of the authors.

Appendix A. A framework for volume computations in piecewise-constant flux-surface computation

The total flux volume generated in the PCFSC unsplit advection scheme is hexahedral, with all faces being planar. Decomposition of the flux volume results in some combination of a cuboid, triangular prisms, pyramids and tetrahedra. All of these shapes are generalizable as arbitrary

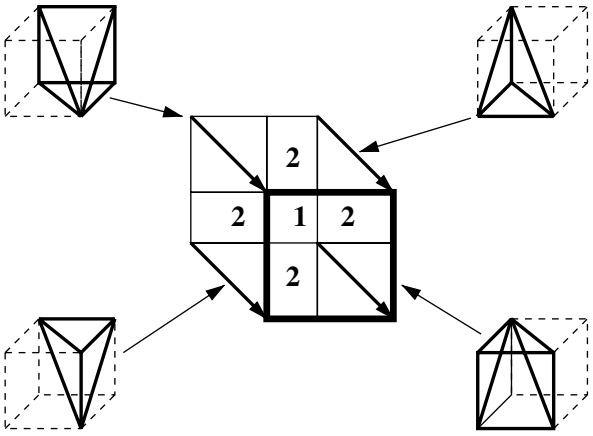

Fig. 15. The decomposition of a PCFSC total flux volume amongst cell in a 9-cell stencil. The total flux volume can be considered the sum of a cuboid, four triangular prisms, and four pyramids/tetrahedra.

hexahedra with planar faces. Fig. 15 shows a decomposition of a sample PCFSC flux.

A number of formulae exist for computing the volumes of cuboids, triangular prisms, pyramids and tetrahedra. Most generally, all arbitrary hexahedral volumes (and shapes with redundant vertices/edges/faces) can be computed using the method of Dukowicz [43]. In the method, the volume is computed as the sum of face contributions:

$$V_{\mathrm{HEX}} = S_{\mathrm{left}} + S_{\mathrm{right}} + S_{\mathrm{near}} + S_{\mathrm{far}} + S_{\mathrm{bottom}} + S_{\mathrm{top}}, \tag{23}$$

where the faces S_i are those of a logical cube. Each of the face contributions to the hexahedron volume computation can be computed as

$$S_{1234} = \frac{1}{12}[\mathbf{r}_1 \cdot (\mathbf{r}_4 \times \mathbf{r}_3) + \mathbf{r}_2 \cdot (\mathbf{r}_1 \times \mathbf{r}_4) + \mathbf{r}_3 \cdot (\mathbf{r}_2 \times \mathbf{r}_1) + \mathbf{r}_4 \cdot (\mathbf{r}_3 \times \mathbf{r}_2)], \tag{24}$$

where 1234 is a face labeled according to the right-hand rule. Computationally cheaper expressions are presented in [43].

The task of computing any C-fluid flux in PCFSC involves *truncation computations*—truncating a planar-surfaced hexahedron total flux volume contribution, using a planar interface reconstruction. The resultant C-fluid flux contribution may feature anywhere up to seven faces, and faces can feature up to six edges. Eqs. (23) and (24) are not prescribed for volume computations featuring seven-faced volumes, or featuring faces with five or more edges. Indirect approaches become necessary to complete all possible C-fluid flux volume shape possibilities generated by the PCFSC scheme on orthogonal meshes.

A.1. Volume of a cuboid truncated by a plane

For the case of an interface reconstruction plane truncating a cuboid, a commonly used indirect approach to V_c computation is documented by Zaleski [44], in which

$$V_{\mathrm{tr}} = V_1 - (V_2 + V_3 + V_4) - (V_5 + V_6 + V_7), \tag{25}$$

where V_i represents the following tetrahedron volumes:

V_1: enclosed by the interface plane, $x = 0$ plane, $y = 0$ plane, and $z = 0$ plane;
V_2: enclosed by the interface plane, $x = \delta x$ plane, $y = 0$ plane, and $z = 0$ plane;
V_3: enclosed by the interface plane, $x = 0$ plane, $y = \delta y$ plane, and $z = 0$ plane;
V_4: enclosed by the interface plane, $x = 0$ plane, $y = 0$ plane, and $z = \delta z$ plane;
V_5: enclosed by the interface plane, $x = \delta x$ plane, $y = \delta y$ plane, and $z = 0$ plane;
V_6: enclosed by the interface plane, $x = \delta x$ plane, $y = 0$ plane, and $z = \delta z$ plane;
V_7: enclosed by the interface plane, $x = 0$ plane, $y = \delta y$ plane, and $z = \delta z$ plane.

Volume V_c is then computed depending on the side of the truncating plane the C-fluid is located in. A straightforward method to implement (indeed derived by us independently for use in 3D direction-split implementations), the method does not extend conveniently to computing V_c in cases of an interface plane truncating a triangular prism, a pyramid, or a tetrahedron.

A.2. Volume of a triangular prism, pyramid or tetrahedron truncated by a plane

A plane of infinite extent cutting a triangular prism, pyramid or tetrahedron divides it into two parts; Fig. 16 shows a range of example truncations. In the case of C-fluid flux computation, the C-fluid may be located on either side of the truncating plane. In Fig. 16(a), the volume of C-fluid, V_c, can be computed as tetrahedral volume V_{abcd} if the

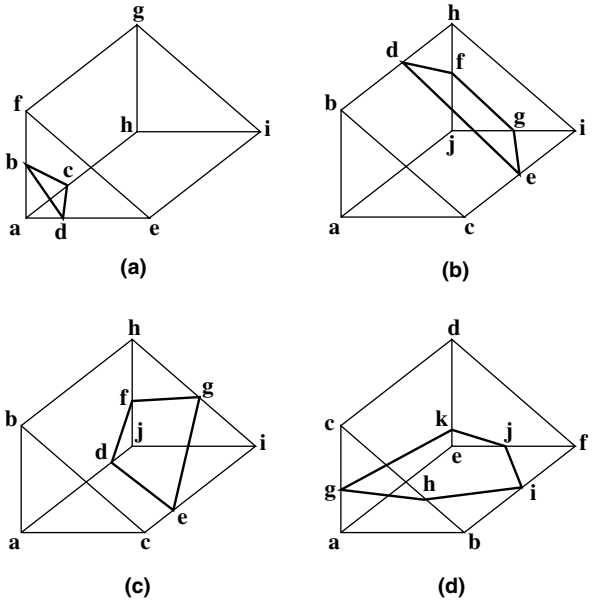

Fig. 16. Sample shapes generated by the truncation of a planar-faced hexahedron with another plane.

outward normal vector from the truncating plane bcd points away from vertex a. In the case of the outward normal vector pointing towards a, the volume can be computed as

$$V_c = V_{\text{edcbfgih}} = V_{\text{aefgih}} - V_{\text{abcd}}. \tag{26}$$

Direct solution of volume V_{edcbfgih} cannot be completed using Eqs. (23) and (24) alone, because of the presence of five-edged faces 'achie' and 'bfghc'. In contrast, indirect computation consists of volume computations V_{aefgih} and V_{abcd}—shapes featuring no more than six faces or four edges per face, for which volumes can be computed using Eqs. (23) and (24).

A.2.1. Decompositions of the total flux volume

As a rule-of-thumb when the interface reconstruction plane truncates the total flux volume contribution into two part-volumes, one part-volume with fewer vertices can be identified, such that the computation of V_c takes the form

$$V_c = V_{\text{HEX}} \quad \text{or} \quad V_c = V_{\text{TOTALFLUX}} - V_{\text{HEX}}, \tag{27}$$

i.e the part-volume with fewer vertices will usually fulfill the logical cube definition, and V_c can be solved by using two applications of Eq. (23). Fig. 16(b) and (c) show further such examples: volumes V_{degfhi} in part (b) and V_{defgij} in part (c) are easily considered as arbitrary hexahedra using Eq. (23) and (24).

In the case of a planar interface reconstruction truncating a triangular prism, there is a possibility that the interface reconstruction becomes a five-edged face on each of the new part-volumes. Such an example is shown in Fig. 16(d), in which neither part-volume can be computed using only one application of Eq. (23). One of the part-volumes can be decomposed into two separate parts, each consisting of no more than six planar faces or four edges per face. The volume of *C*-fluid can therefore be computed as

$$\begin{aligned} &V_c = V^1_{\text{HEX}} + V^2_{\text{HEX}} \quad \text{or} \\ &V_c = V_{\text{TOTALFLUX}} - (V^1_{\text{HEX}} + V^2_{\text{HEX}}). \end{aligned} \tag{28}$$

In the case of a total flux volume contribution in the shape of triangular prism, pyramid or tetrahedron, a useful strategy for decomposing a part-volume featuring a five-edged face, described by us as a Five-Edged Face Decomposition Procedure is

(1) On the truncating-plane face, identify two successive vertices—labeling them A, B—such that edge AB is also an edge of the non-truncating-plane face of the part-volume with the most vertices (usually a five-edged face).

```
Sweep through mesh to initialize homogeneous fluid fluxes, using FOU.
Sweep through mesh second time
  Sweep through faces in vicinity of interface cells.
    Identify vertices of face. Compute coordinates.
    Trace the vertices of face along characteristics to define total PCFSC flux shape.
    Sweep through 18-cell stencil surrounding face. [Stencil cell (ii, jj, kk).]
      Isolate contribution of Total PCFSC flux within stencil cell. [Fig. 8 procedure.]
      Identify shape of contribution (cuboid, triangular prism, pyramid, tetrahedron).
      If interface reconstruction doesn't cut any edges of contribution
        Homogeneous case. Use Dukowicz formulae.
      Else
        If contribution shape = cuboid
          Use method based on eq. 25, with Dukowicz formulae (or otherwise).
        Else
          Truncate shape with interface reconstruction plane.
          Identify vertices and connectivity of each part-volume.
          If one part-volume is an arbitrary hexahedron
            Use eq. 27 with Dukowicz formulae.
          Else
            Apply Five-Edged Face Decomposition Procedure.
            Use eq. 28 with Dukowicz formulae.
          End if
        End if
      End if
    End sweep through 18-cell stencil.
  End sweep through faces.
End second mesh sweep.
```

Fig. 17. Sample outline of algorithm for the PCFSC unsplit advection scheme.

(2) Identify a third vertex, C—a successor of B that is a vertex of the truncating-plane face, but not of the non-truncating-plane face AB belongs to.
(3) On that non-truncating-plane face of the part-volume of which BC is an edge, identify the predecessor of B, and label this vertex D. Then identify the successor of C, and label the vertex E.
(4) Identify ACDE as a single face. This is the face that is defined to decompose the part-volume into two volumes that are arbitrary hexahedra. Compute $V^1_{\mathrm{HEX}} = V_{\mathrm{ABCDE}}$. The volume to the other side of face ACDE (the side that does not feature B as a vertex) is computed to be V^2_{HEX}.

Taking the example of Fig. 16(d), and isolating consideration to the part-volume beneath the truncating plane, step (1) identifies ghijk as the truncating plane, and abije as the non-truncating-plane face possessing five edges. Common to both is edge ij, so we set vertices A = i and B = j. Applying step (2), we assign vertex C = k. Applying step (3), we assign vertices D = e and E = e. In step (4), we identify the new face ACDE = ike, compute $V^1_{\mathrm{HEX}} = V_{\mathrm{ikej}}$, and $V^2_{\mathrm{HEX}} = V_{\mathrm{ikeghba}}$. In the case of swapping A and B, the new partitioning face would be ACDE = bhj, $V^1_{\mathrm{HEX}} = V_{\mathrm{bhji}}$, and $V^2_{\mathrm{HEX}} = V_{\mathrm{bhjaekg}}$. If we instead consider the part-volume above the truncating plane, we can set vertices A = h, B = i, C = j, D = f, E = f, such that the new partitioning face is ACDE = hjf, $V^1_{\mathrm{HEX}} = V_{\mathrm{hjfi}}$, and $V^2_{\mathrm{HEX}} = V_{\mathrm{hjfcdkg}}$. Similarly, if A and B are reversed, then the assignments become ACDE = igc, $V^1_{\mathrm{HEX}} = V_{\mathrm{igch}}$, and $V^2_{\mathrm{HEX}} = V_{\mathrm{igcdfjk}}$.

A.3. Final notes

The beauty of this framework is that the math operations required are simple; Eqs. (23) and (24), and the computing of intercepts between lines and planes, are the extent of the mathematical complexity. In addition, the Five-Edged Face Decomposition Procedure is a robust means of decomposing shapes into logical cubes, that can be coded up relatively easily (similarly for the cuboid treatment). Fig. 17 outlines an algorithm for generating multidimensional fluxes according to the PCFSC method.

The framework presented here is only applicable to the PCFSC scheme for unsplit advection, since it generates planar-faced hexahedral total flux volumes. An alternative framework, that is extensible to full-remapping in which non-planar ruled surfaces are introduced, has been implemented in our unsplit advection modules. This framework, involving more complex mathematical operations, was derived by Zemach [45] and is unpublished, and has not been discussed here. The framework presented in this appendix at least makes the benefits of unsplit advection in volume tracking realizable. In addition, the appendix makes this paper self-containing, such that this paper is all that is required for 3D implementation.

References

[1] Woodward PR, Colella P. The numerical solution of two-dimensional fluid flow with strong shocks. J Comput Phys 1984;54:115–73.
[2] Osher S, Sethian J. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations. J Comput Phys 1988;79:12–49.
[3] Sethian J. Level set methods and fast marching methods. Cambridge University Press; 1999.
[4] Yabe T, Aoki T. A universal solver for hyperbolic equations by cubic polynomial interpolation. Comput Phys Commun 1991;66:219–32.
[5] Debar R. Fundamentals of the KRAKEN code. Technical Report UCIR-760, Lawrence Livermore National Laboratory, 1974.
[6] Noh WF, Woodward PR. SLIC (simple line interface method). In: van de Vooren AI, Zandbergen PJ. editors. Lecture Notes Phys 1976;59:330–40.
[7] Hirt CW, Nichols BD. Volume of fluid (VOF) method for the dynamics of free boundaries. J Comput Phys 1981;39:205–26.
[8] Glimm J, Grove JW, Li XL, Shyue KM, Zeng YN, Zhang Q. 3-Dimensional front tracking. SIAM J Sci Comput 1998;19:703–27.
[9] Unverdi SO, Tryggvason G. A front-tracking method for viscous, incompressible, multi-fluid flows. J Comput Phys 1992;100:25–37.
[10] Torres DJ, Brackbill JU. The Point-Set method: front-tracking without connectivity. J Comput Phys 2000;165:620–44.
[11] Hirt CW, Cook JL, Butler TD. A Lagrangian method for calculating the dynamics of an incompressible fluid with a free surface. J Comput Phys 1970;5:103–24.
[12] Fritts MJ, Boris JP. The Lagrangian solution of transient problems in hydrodynamics using a triangular mesh. J Comput Phys 1979;31:173–215.
[13] Hirt CW, Nichols BD. Adding limited compressibility to incompressible hydrocodes. J Comput Phys 1980;34:390–400.
[14] Meier M, Yadigaroglu G, Andreani M. Numerical and experimental study of large steam–air bubbles injected in a water pool. Nucl Sci Eng 2000;136:363–75.
[15] Liovic P, Liow J-L, Rudman M. A Volume-of-Fluid (VOF) method for the simulation of metallurgical flows. ISIJ Int 2001;41:225–33.
[16] Liovic P, Rudman M, Liow J-L. Numerical modelling of free surface flows in metallurgical vessels. Appl Math Modell. 2002;26:113–40.
[17] Liovic P, Lakehal D, Liow J-L. LES of turbulent bubble formation and break-up based on interface tracking. In: Geurts BJ, Friedrich R, Metais O, editors. Direct and large-eddy simulation V. ERCOFTAC series, vol. 10. Kluwer Academic Publishers; 2004. p. 261–70.
[18] Rudman M. A volume-tracking method for incompressible multifluid flows with large density variations. Int J Numer Methods Fluids 1998;28:357–78.
[19] Bussmann M, Kothe DB, Sicilian JM. Modeling high density ratio incompressible interfacial flows. In: Proceedings of ASME fluids engineering division summer meeting, Montreal, 14–18 July 2002, FEDSM2002-31125.
[20] 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.
[21] Aulisa E, Manservisi S, Scardovelli R. A mixed markers and volume-of-fluid method for the reconstruction and advection of interfaces in two-phase and free-boundary flows. J Comput Phys 2003;188:611–39.
[22] Youngs DL. Time-dependent multi-material flow with large fluid distortion. Morton KW, editor. Numerical methods for fluid dynamics, 1982. p. 273–85.
[23] Pilliod Jr JE, Puckett EG. Second-order Volume-of-Fluid algorithms for tracking material interfaces. Technical Report LBNL-40744, Lawrence Berkeley National Laboratory, 1997.
[24] Rider WJ, Kothe DB. Reconstructing volume tracking. J Comput Phys 1997;141:112–52.
[25] Rudman M. Volume tracking methods for interfacial flow calculations. Int J Numer Methods Fluids 1997;24:671–91.

[26] Swartz B. The second-order sharpening of blurred smooth borders. Math Comput 1989;52:675–714.
[27] Mosso SJ, Swartz BK, Kothe DB, Ferrell RC. A parallel, volume-tracking algorithm for unstructured meshes. In: Schiano P, editor. Parallel computational fluid dynamics '96, Italy, 1996.
[28] Scardovelli R, Zaleski S. Interface reconstruction with least-square fit and split Lagrangian–Eulerian advection. Int J Numer Methods Fluids 2003;41:251–74.
[29] Miller GH, Colella P. A conservative three-dimensional Eulerian method for coupled solid–fluid shock capturing. J Comput Phys 2002; 183:26–82.
[30] Price GR, Reader GT, Rowe RD, Bugg JD. A piecewise parabolic interface calculation for volume tracking. Proceedings of the sixth annual conference of the computational fluid dynamics society of Canada. Victoria, British Columbia: University of Victoria; 1998.
[31] Renardy Y, Renardy MM. PROST: a parabolic representation of surface tension for the volume-of-fluid method. J Comput Phys 2002;183:400–21.
[32] Lopez J, Hernandez J, Gomez P, Faura F. A volume of fluid method based on multidimensional advection and spline interface reconstruction. J Comput Phys 2004;195:718–42.
[33] Garrioch SH, Baliga BR. A multidimensional advection technique for PLIC Volume-of-Fluid methods. In: Djilali N, Dunlop S, editors. Proceedings of the 5th annual conference of the computational fluid dynamics society of Canada, 1997. p. 11.3–8.
[34] Garrioch SH, Baliga BR. Interface tracking using a multidimensional advection technique based on skewed subadvections. In: Proceedings of ASME fluids engineering division summer meeting, 21–25 June 1998, Washington, DC, FEDSM98-5207.
[35] Mosso SJ, Swartz BK, Kothe DB, Clancy SP. Recent enhancements of volume tracking algorithms for irregular grids. Technical Report LA-CP-96-227, Los Alamos National Laboratory, 1996.
[36] Shahbazi K, Paraschivoiu M, Mostaghimi J. Second order accurate volume tracking based on remapping for triangular meshes. J Comput Phys 2003;188:100–22.
[37] Kothe DB, Rider WJ, Mosso SJ, Brock JS, Hochstein JI. Volume tracking of interfaces having surface tension in two and three dimensions. Technical Report AIAA 96-0859, AIAA, 1996. [Presented at the 34th Aerospace Sciences Meeting and Exhibit].
[38] Aulisa E, Manservisi S, Scardovelli R, Zaleski S. A geometrical area-preserving Volume-of-Fluid advection method. J Comput Phys 2003; 192:355–64.
[39] Dukowicz JK, Baumgardner JR. Incremental remapping as a transport/advection algorithm. J Comput Phys 2000;160:318–35.
[40] Harvie DJE, Fletcher DF. A new Volume of Fluid advection algorithm: the Stream scheme. J Comput Phys 2000;162:1–32.
[41] Leveque R. High-resolution conservative algorithms for advection in incompressible flow. SIAM J Numer Anal 1996;33:627–65.
[42] Enright D, Fedkiw R, Ferziger J, Mitchell I. A hybrid Particle Level set method for improving interface capturing. J Comput Phys 2002; 183:83–116.
[43] Dukowicz JK. Efficient volume computation for three-dimensional hexahedral cells. J Comput Phys 1988;74:493–6.
[44] Zaleski S. Multiphase-flow CFD with volume of fluid (VOF) methods. In: Yadigaroglu G, Hetsroni G, editors. Modelling and computation of multiphase flows, ETH-Zurich short course, 19–23 March 2001, vol. 15B, p. 1–43.
[45] Zemach C. Notes on the volume of a ruled hexahedron behind a truncating plane, unpublished manuscript, 1993.
[46] Liow J-L, Gray NB. Experimental study of splash generation in a flash smelting furnace. Metall Mater Trans B 1996;27B:633–46.
[47] Liovic P. Numerical modelling of top-submerged gas injection. PhD Dissertation, The University of Melbourne, 2000.