Conservative volume of fluid advection method on unstructured grids in three dimensions

By C. B. Ivey AND P. Moin

1. Motivation and objectives

The volume of fluid (VOF) method is one of the most widely used formulations to track interfaces between two immiscible fluids in free-surface and interfacial flow simulations. In this method, the interface evolution is implicitly tracked using a discrete indicator function, F, whose value represents the volume fraction of the tagged fluid within a cell. F is the cell average of the fluid marker function, f, that is constant in each phase and jumps at the interface from 0 to 1. As described by Scardovelli & Zaleski (1999), the variations in VOF algorithms can be distinguished by the routines used to locate the implicit interface and by the methods used to integrate the volume fraction equation in time, respectively known as the reconstruction and advection steps.

Of the VOF algorithms in use, the geometric formulations are more accurate than their algebraic counterparts, of which classic upwinding schemes come to mind. Traditional first-order upwinding maintains the boundedness and globally conserves F; however, it quickly smears out the interface. Higher-order upwinding schemes help maintain the integrity of the interface; however, they lack boundedness in F, creating a whole slew of problems. The benefit of the algebraic methods are their simplicity; they do not require any reconstruction of the interface.

Geometric algorithms locally approximate the interface with a lower-dimensional manifold, so that analytic techniques can be encompassed into the VOF method to help preserve the integrity of the interface, while maintaining the local conservation of F. A specific class of the geometric algorithms that describe the interface by a series of disconnected planes, named the piecewise-linear interface calculation (PLIC), has gained popularity within the VOF community for their great foundation of available geometric tools. As a side note, the nomenclature was developed during the era of two-dimensional (2D) VOF, hence the usage of "linear" in lieu of "planar". Specifically, PLIC representations allow for the use of analytic routines to determine the volume fraction in an interface containing cell and to solve the inverse problem, placing the local interface such that a desired volume fraction is maintained; these procedures are integral to the reconstruction and advection steps. Scardovelli (2000) developed the tools for rectangular and hexahedral elements, Yang & James (2006) determined the analytics for triangular and tetrahedral meshes, and López & Hernández (2008) published the geometric operations for general convex grids.

PLIC-VOF can be further categorized by means of integration of the advection equation. The integration can be split along each coordinate direction, or the equation can be integrated all at once, irrespective of the coordinate direction. Split advection is much simpler to implement, but it has the added cost of requiring an advection and reconstruction step for each dimension. Further, continuity is satisfied over the entire space, not along a given direction, so divergence-based forcing terms need to be incorporated into the formulation. Of the many directionally split schemes, Weymouth & Yue (2010) pro-

posed the only one that discretely conserves volume. They cleverly introduced a modified divergence forcing term that ensured F remained bounded during each one-dimensional (1D) advection step and that the full advection sequence was conservative. Continuity combined with volume conservation enforces density conservation, traditionally a zeroth-order constraint for any simulated system, due to the scaling of mass, momentum, and energy fluxes on density. For liquid fuel combustion, a problem of particular interest to the authors of this brief, the fuel density is typically three orders of magnitude larger than that of the oxidizer, making local volume conservation paramount to simulation accuracy.

Unsplit advection schemes do not have any source terms; however, the design of the advection operator is complicated by the added dimensionality of the system. Inconveniently, operator split schemes cannot be used within unstructured mesh environments, a necessary gridding paradigm for practical geometries, yielding unsplit advection algorithms as the only potential basis for realistic liquid fuel combustion simulations. Two seminal works in this area deserve credit for their direct impact on the development of the scheme described herein, López *et al.* (2004) and Hernández *et al.* (2008). Both papers described the use of flux polyhedra, a construct detailed in Section 2.1, to accurately evolve F in time; they quoted accuracies comparable to the top-of-the-line contemporary schemes, somewhere between first- and second-order. López *et al.* (2004) described a 2D algorithm that could conservatively advect F using edge-matched flux polyhedra (EMFPA-2D). Hernández *et al.* (2008) described their three-dimensional (3D) extension of EMFPA, the face-matched flux polyhedra (FMFPA-3D). Whereas EMFPA-2D, with a properly chosen time-step, could prevent the formation of over-/undershoots in the volume fraction by discretely enforcing that the flux polyhedra do not over-/underlap, FMFPA-3D could only reduce the formation of over-/undershoots in the volume fraction. FMFPA-3D required the use of a volume fraction redistribution and/or flux rescaling algorithm; these are unphysical, zeroth-order corrections that destroy the accuracy of the VOF method. Hernández *et al.* (2008) suggested that it would be possible to design an edge-matched flux polyhedra scheme in 3D (EMFPA-3D); however, they left that as a future development for the VOF community. The objective of this brief is to detail EMFPA-3D, an unsplit PLIC-VOF advection scheme that conservatively and accurately evolves F in time on an unstructured 3D mesh by ameliorating the aforementioned flux polyhedra construct.

2. EMFPA-3D

The equations governing the motion of an unsteady, viscous, incompressible, immiscible two-fluid system are the Navier-Stokes equations, augmented by a localized surface tension force:

$$\begin{aligned} \rho\left(\frac{\partial \vec{v}}{\partial t}+\vec{v}\cdot\nabla\vec{v}\right) &= -\nabla p+\nabla\cdot\left(\mu\left[\nabla\vec{v}+\{\nabla\vec{v}\}^{T}\right]\right)+\rho\vec{g}-\sigma\kappa_{\Sigma}\delta_{\Sigma}\hat{n}_{\Sigma} \\ \nabla\cdot\vec{v} &= 0, \end{aligned} \tag{2.1}$$

where, ρ is the density, $\vec{v}$ is the velocity, p is the pressure, μ is the viscosity, $\vec{g}$ is the gravitational force, σ is the surface tension coefficient, Σ represents the interface, κ_{Σ} is the surface curvature at the interface, δ_{Σ} is a delta function located at the interface, and $\hat{n}_{\Sigma}$ is the surface normal at the interface. This one-fluid system of equations requires knowledge of the interface location to determine ρ, μ, and the localized surface tension

force. A marker function, f, is used to implicitly track the interface. f is a Heaviside function that takes on a value of 1 in the tagged fluid and 0 otherwise, so ρ and μ have a simple one-to-one correspondence with f. Because f is a passive tracer, it follows a linear advection equation:

$$\frac{\partial f}{\partial t} + \vec{v} \cdot \nabla f = 0. \tag{2.2}$$

Integration of the above equation over the cell, Ω, and between the time interval, $t' \epsilon \, [t, t + \Delta t]$, yields an equation for the cell volume fraction, F:

$$F\left(t + \Delta t\right) = F\left(t\right) - \frac{1}{V_\Omega} \int_t^{t+\Delta t} \int_\Omega \vec{v} \cdot \nabla f dV dt'. \tag{2.3}$$

Utilization of the continuity condition in conjunction with the divergence theorem along a surface, S, leads to a simple, flux-based update rule for F:

$$F\left(t + \Delta t\right) = F\left(t\right) - \frac{1}{V_\Omega} \int_t^{t+\Delta t} \int_S \left(\vec{v} f\right) \cdot \hat{n}_A dA dt'. \tag{2.4}$$

Equation 2.4 is unsolvable in the form above, with F still dependent on f. The VOF algorithms essentially differ in how the flux term is discretized. This brief will describe the EMFPA-3D discretization developed at the Center for Turbulence Research (CTR). Before beginning the description of EMFPA-3D, certain assumptions need to be mentioned. EMFPA-3D is a class of PLIC-VOF algorithms, so it requires a reconstruction step to describe the interface locally as a plane, $\hat{n}_{\Gamma_c} \cdot \vec{x} + C_{\Gamma_c} = 0$, where Γ designates a plane and c is the index of the plane. This notation is to be used later and is repeated here for consistency. The reconstruction step is not the focus of this brief, so the usage of an acceptable estimator of $\hat{n}_{\Gamma_c}$ is assumed. In order for the scheme to be convergent, the estimation of the normal needs to be at least first-order accurate on general grids; the gradient-based approximation in Parker & Youngs (1992) fits this criterion, for example. Their method essentially calculates the gradient of F at the nodes surrounding the cell and then averages those gradients to get a cell-averaged gradient. The gradient is then negated and normalized to provide an estimation of $\hat{n}_{\Gamma_c}$, which points into the reference fluid. The method uses more information than the traditional central differences to help characterize sub-cell discontinuities. It should be emphasized that a second-order estimation of the normal would augment the total accuracy of the scheme, due to the empirical accuracy of the flux polyhedron, so the development of such a reconstruction is in the works, namely mapping the height-functions described in Weymouth & Yue (2010) to general grids. C_{Γ_c} is determined using the volume enforcement procedure proposed in López & Hernández (2008). Note that the volume enforcement procedure is integral to the construction of the flux polyhedron as well, so a more detailed derivation of the method is provided in Section 2.1.1. The local surface curvature, κ_{Γ_c}, is not needed to advect F, so its estimation is left out of the brief. EMFPA-3D requires a solenoidal velocity field located at the faces of the mesh, limiting the use of the scheme to finite volume methods. Discrete continuity is necessary for the scheme to remain conservative; this is obviated by the usage of the continuity condition above to get Eq. (2.4) in a flux formulation. It is also assumed that there is only one face separating two cells; localized refinement with hanging nodes is not yet handled.

The EMFPA-3D algorithm breaks down the advection operator into a series of local advection steps performed using flux polyhedra. The flux polyhedron is based on the idea of a stream tube, wherein the flux face is swept over a volume defined by the local

velocities and time step. There are many details involved in the construction of a flux polyhedron, due to the restriction of volume conservation and planar geometries; however, the geometric description aims to approximate the simple concept of a stream tube using the information available. The proposed advection algorithm is outlined below:

(a)Use distance-weighted averaging to interpolate the velocities from the faces to their respective adjacent nodes.

(b)Triangulate the flux faces by drawing edges from the face center to each of the nodes bounding the face, as shown in Figure 1. The triangulation is used to constrain the geometry of the flux polyhedron, which will be necessary for the developments described in Section 2.3.

(c)Loop through all the triangular sections of all the faces, completing the steps below:

•Determine the donor cell based on sign $(\vec{v}_{fa} \cdot \hat{n}_{fa})$, where fa represents the face. It is assumed that the grid was designed such that $\hat{n}_{fa}$ has a predefined orientation.

•Calculate the stream tube volume using $V_{st} = \vec{v}_{fa} \cdot \hat{n}_{fa} A_{tri} \Delta t$, where V_{st} is the stream tube volume, A_{tri} is the area of the triangular section, and Δt is the time step.

•Construct a flux polyhedron of volume V_{st} using Δt, the face velocity and the two nodal velocities bounding the triangular section. The construction of a flux polyhedron is described in Section 2.1. If unable to create a flux polyhedron using the current time step, take $\Delta t = \Delta t/2$, go to (c), and double the number of times the calculation is performed.

•If the flux polyhedron is self-intersecting, slice the flux polyhedron by the flux face, and store both the section of the flux polyhedron contained in the donor cell and the section contained in the receiver cell. Self-intersection can occur when sign $(\vec{v}_{fa} \cdot \hat{n}_{fa}) \neq$ sign $(\vec{v}_{no} \cdot \hat{n}_{no})$ for one of the nodes, no, possibly resulting in an inward flux as described in Section 2.3.

•Truncate the flux polyhedra by the PLIC surfaces associated with their cell. Truncation is a geometric procedure described in López & Hernández (2008); however, convexity of the geometry was assumed, so truncation procedures acceptable for non-convex and convex geometries alike are provided in Section 2.2.

•Truncate the flux polyhedra by the cell faces adjacent to the triangular section that are associated with the given polyhedron.

•If the flux polyhedron was self-intersecting, store the volume of the finalized sliced polyhedron of the donor cell as V_{out} and the finalized sliced polyhedron of the receiver cell as V_{in}. If the polyhedron was not self-intersecting, store the finalized sliced polyhedron as V_{out} and set $V_{in} = 0$.

•Update the volume fraction, $F(t + \Delta t) = F(t) - \frac{V_{out} - V_{in}}{V_\Omega}$.

•Update the time, $t = t + \Delta t$.

2.1. *Construction of the flux polyhedron*

The flux region for a given triangular section of a face is delimited by a potentially non-convex, irregular polyhedron which has a variable number of faces and nodes per face. The number of faces and nodes per face depends on the number of edges that are created to preserve the planarity of the polyhedron faces, whether or not any of the velocities are directed into the donor cell, and the orientations and positions of the adjacent flux polyhedra bounding faces. For clarity, this discussion will focus on the case where the velocities are directed out of the donor cell, as shown in Figure 3. Inward velocities will be discussed in Section 2.3 and complications resulting from the capping faces of adjacent flux polyhedra will be addressed in Section 2.4.

A polyhedron is characterized by the vertex positions, $\vec{x}_p$, the outward surface normals

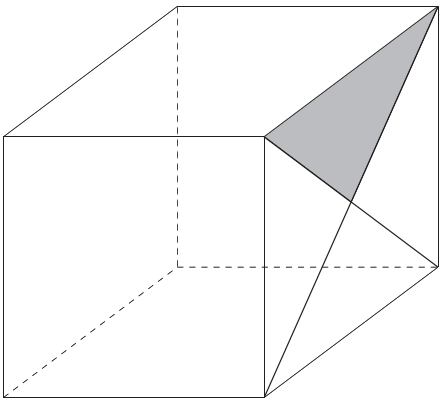

FIGURE 1. Triangulation of the flux face enables the algorithm to handle self-intersection of the flux polyhedron by constraining the geometry.

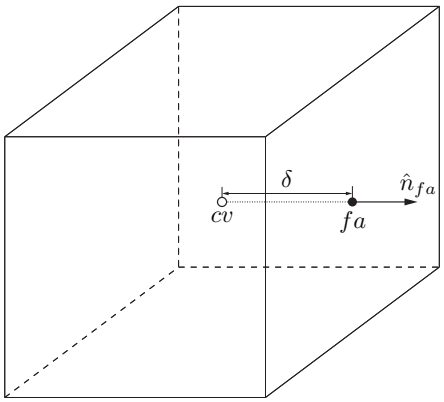

FIGURE 2. Use distance between face and cell center in the normal direction as length scale for creating planarity-preserving edges.

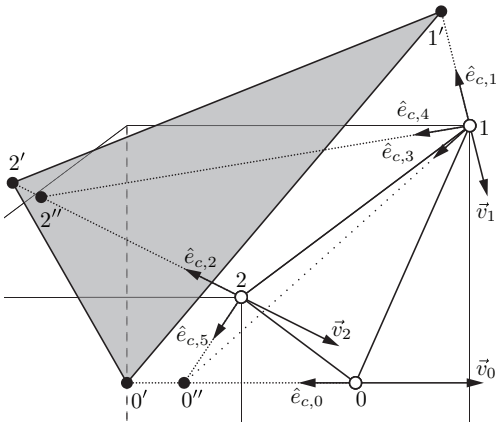

FIGURE 3. Edges bounding stream tube are a combination of vectors anti-parallel to the velocity vectors and the vectors created to preserve planarity.

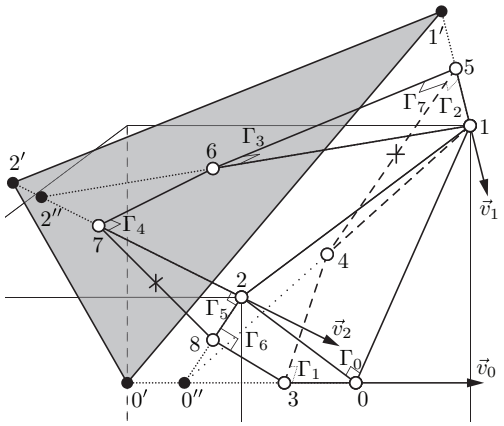

FIGURE 4. Placement of plane such that the volume bound by the flux polyhedron equals that of the flux through the triangular face section, V_{st}.

and constants, $\hat{n}_{\Gamma_j}$ and C_{Γ_j}, and the mapping from the global node index to the local node index for a given surface, $p(j,i)$, where the local indices follow a right-hand rule around the outward normal of the face. For clarity, p will be used solely for global node indexing, j for face indexing, and i for local face indexing. This information is required for any and all geometric operations to be performed on the flux polyhedron. The description provided herein will follow the example fully detailed in Figures 3 and 4. The procedure is quite general; however, the specificity allows for a simpler description.

First, fully characterize the triangular flux face section, face 0 in this example. The nodal positions, $\vec{x}_p$, and velocities, $\vec{v}_p$, are provided for $p = 0, 1, 2$. The respective normal depends on which cell is the donor cell:

$$\hat{n}_0 = \hat{n}_{fa} \text{sign}\left(\vec{v}_0 \cdot \hat{n}_{fa}\right). \tag{2.5}$$

Define the ordering such that it follows the right-hand rule. Because this is the first face, define the global and local indexing to be the same.

For the example described herein, face c is the face that caps the polyhedron to preserve volume. Note that face c translates during the flux polyhedron construction; however, it initially corresponds to the shaded plane defined by the nodes $0'$, $1'$, and $2'$ in Figures 3 and 4. The edge unit vectors point from face 0 to face c, so for every node i of face c there is an associated base node, i_0 on face 0. Using these definitions, and our mapping operator defined earlier, the three velocity-based edge unit vectors can be defined:

$$\hat{e}_{c,i} = -\frac{\vec{v}_{p(0,i_0)}}{\|\vec{v}_{p(0,i_0)}\|} \quad \text{for} \quad i = 0, 1, 2. \tag{2.6}$$

Note that this is not necessarily true for the case with inward directed velocities, which will be described in Section 2.3. Further, these unit vectors are used to define the normal of the bounding face. In order to do so, propagate the velocity vectors back in time along their respective velocity-based edge unit vectors:

$$\vec{x}'_p = \vec{x}_p - \vec{v}_p \Delta t \quad \text{for} \quad p = 0, 1, 2, \tag{2.7}$$

where the nodes are shown in Figure 3. Then define the capping face normal using the auxiliary vertices defined in Eq. 2.7:

$$\hat{n}_{\Gamma_c} = \frac{(\vec{x}'_1 - \vec{x}'_2) \times (\vec{x}'_0 - \vec{x}'_1)}{\| (\vec{x}'_1 - \vec{x}'_2) \times (\vec{x}'_0 - \vec{x}'_1) \|} \text{sign}\left(\vec{v}_0 \cdot \hat{n}_{fa}\right). \tag{2.8}$$

The descriptions of the bounding face normal, velocity-based edges, and auxiliary nodes emphasize that the flux polyhedron is a planar approximation of the local stream tube emanating from the triangular section of the flux face, giving EMFPA-3D its upwinding nature and its competitive accuracy. The rest of this brief focuses on perturbing the definition of the flux polyhedron such that it remains planar and conservative by preventing overlap with neighboring flux polyhedra.

The velocity-based edges described in Eq. (2.6) may not lie in the same plane, so edges may need to be added to preserve the planarity of the faces bounding the flux polyhedron. The following approximation of the surface strives to minimize the total number of faces of the resulting polyhedron; the advantage of this approach over traditional linear triangulation will be discussed in Section 2.2. Determine if any two edge unit vectors are coplanar with the edge joining them. For example, determine if $(\vec{x}_1 - \vec{x}_0)$, $\hat{e}_{c,1}$ and $\hat{e}_{c,0}$ lie in the same plane. Three vectors, $\vec{a}$, $\vec{b}$, and $\vec{c}$, are coplanar if their determinant is 0, $\vec{a} \cdot \left(\vec{b} \times \vec{c}\right) = 0$. Specifically, check if $(\vec{x}_1 - \vec{x}_0) \cdot \hat{e}_{c,1} \times \hat{e}_{c,0} = 0$, $(\vec{x}_2 - \vec{x}_1) \cdot \hat{e}_{c,2} \times \hat{e}_{c,1} = 0$, and $(\vec{x}_0 - \vec{x}_2) \cdot \hat{e}_{c,0} \times \hat{e}_{c,2} = 0$. If the equality is not satisfied, then an edge needs to be created so that the planarity of the flux polyhedron is preserved. Usually any two given velocity-based edge unit vectors are not coplanar with the edge joining them, so for the rest of the discussion three planarity-preserving edges are to be assumed. The following definitions will be used: $\hat{e}_{c,3}$ is planar with $\hat{e}_{c,0}$ and $\hat{e}_{c,1}$, $\hat{e}_{c,4}$ is planar with $\hat{e}_{c,1}$ and $\hat{e}_{c,2}$, and $\hat{e}_{c,5}$ is planar with $\hat{e}_{c,2}$ and $\hat{e}_{c,0}$; this specific case is illustrated in Figure 3.

In order to manufacture planarity, three facts of polygons are exploited: (a) polygons are planar, (b) slicing a polygon yields a polygon, and (c) triangles are polygons. Referring to Figure 3, by drawing an edge from 1 to the line parallel to $\hat{e}_{c,2}$, namely the newly constructed edge parallel to $\hat{e}_{c,4}$, a triangle is created; therefore, by (a) and (c), a plane is created. To define the edge parallel to $\hat{e}_{c,4}$ such that planarity will be observed on both faces bound by the line parallel to $\hat{e}_{c,1}$ and the line parallel to $\hat{e}_{c,2}$, orient the edge parallel to $\hat{e}_{c,4}$ such that it intersects the line parallel to $\hat{e}_{c,2}$ at a distance large enough

such that the bounding plane will slice the newly created triangle. Slicing the triangular plane formed by intersecting the edge parallel to $\hat{e}_{c,4}$ and the line parallel to $\hat{e}_{c,2}$ does not change its planarity, by (b), and the slice will introduce a new triangle between the edge parallel to $\hat{e}_{c,4}$ and the line parallel to $\hat{e}_{c,1}$. Suppose the node created by the edge drawn from node 1 to the line parallel to $\hat{e}_{c,2}$ is called $2''$. Based on this definition, 1, 2, and $2''$ form a triangle and, hence, a plane. Now slice this triangle by the bounding face and define the intersection of the bounding face and the edge parallel to $\hat{e}_{c,4}$ as 6 and the intersection between the bounding face and the edge parallel to $\hat{e}_{c,2}$ as 7. Now 6, 7, 2, and 1 form a plane. Further, define the intersection of the bounding plane with the line defined by $\hat{e}_{c,1}$ as 5. Now 1, 5, and 6 form a new triangle and, hence, a plane. This process is repeated for all the required planarity-based edges.

Because the placement of the bounding face is not known a priori, the intersection distance needs to be defined such that the bounding face will always be positioned at a distance smaller than the intersection distance. In order to ensure that the various flux polyhedra of a cell do not overlap, the bounding face may never be positioned further away from the flux face than the cell center in the direction of the flux face normal. Otherwise, it would be possible for the flux polyhedron to intersect with a flux polyhedron emanating from the opposite side of the cell. The smallest distance that is guaranteed to satisfy such a condition is the length scale along the edge connecting the cell center to the face center, δ, projected onto planarity-based edge unit vector. Minimizing the distance helps split the non-planar region into two planes of comparable area. The length scale, illustrated in Figure 2, where cv is the cell center, follows a simple definition:

$$\delta = |\,(\vec{x}_0 - \vec{x}_{cv}) \cdot \hat{n}_0|, \tag{2.9}$$

with an associated intersection node position:

$$\vec{x}''_{p(0,i_0)} = \vec{x}_{p(0,i_0)} + \tfrac{\delta}{\|\hat{e}_{c,i} \cdot \hat{n}_0\|} \hat{e}_{c,i} \quad \text{for} \quad i = 0, 1, 2. \tag{2.10}$$

Because faces are shared between adjacent triangular sections of a flux face or a neighboring flux face, the edges created to preserve planarity need to be mimicked between shared faces. This is accomplished by two definitions. First, choose the adjoining edge to be such that it emanates from the node with the smaller velocity magnitude towards the line connected to the node with the larger velocity magnitude. This is an arbitrary definition, but it allows for adjacent flux polyhedra to be defined without any necessary communication. Second, δ, for a given shared face, needs to be defined as the maximum of the two flux polyhedra length scales. Flux polyhedra that share a face and are created from the same flux face have the same δ; however, flux polyhedra that share a face between adjacent cells do not necessarily have matching length scales. By choosing the maximum length scale as δ, the slice plane is guaranteed to be positioned between the intersection and the flux face for both flux polyhedra.

Once all the velocity- and planarity-based edges are defined, the surfaces created between the edges need to be characterized by the surface normals and constants. The normals are determined by appropriately crossing the bounding edge unit vectors such that normal points outwards. Then the surface constants can be determined by dotting the normal vector with an adjacent node on the fully characterized triangular face section:

$$C_{\Gamma_j} = -\hat{n}_{\Gamma_j} \cdot \vec{x}. \tag{2.11}$$

Now the geometry is fully specified, excluding the bounding face constant, C_{Γ_c}, and the nodes shared by the bounding face, which are, fortunately, fully parameterized by

C_{Γ_c}. This parameterization is based on a fixed capping face normal, $\hat{n}_{\Gamma_c}$. C_{Γ_c} will shift the bounding plane to enforce the flux polyhedron volume to be V_{st}. There are associated min and max limits on the allowable C_{Γ_c} in order to prevent the polyhedron from extruding beyond the bounds set by the cell center and flux face; if a polyhedron cannot be constructed such that the volume matching occurs, the advection procedure must start over using a smaller time step, as outlined in Section 2. As a side note, $\hat{n}_{\Gamma_c}$ was specified using Eq. (2.7), so the orientation implicitly had a velocity dependence; shifting the plane with a fixed normal does not respect this velocity dependence. If a shift in the capping plane is envisaged as a respective increase or decrease in time, the volume enforcement procedure can further respect the stream tube idealization. Another volume enforcement procedure can be defined with an orientable and shiftable capping surface, where the parameterization with C_{Γ_c} is augmented by the definitions in Eq. (2.7). This procedure is left out of the brief as it has not yet been fully formulated.

2.1.1. *Determination of bounding face position*

The volume of a polyhedron can be determined by the surface normals and vertex positions, where j is the face index, i is the local node index, I_j is the number of nodes in face j, and J is the number of faces. Periodicity in i is assumed:

$$6V_{st} = \sum_{j=0}^{J-1} \left[\left(\hat{n}_{\Gamma_j} \cdot \vec{x}_{j,0} \right) \hat{n}_{\Gamma_j} \cdot \sum_{i=0}^{I_j-1} \left(\vec{x}_{j,i} \times \vec{x}_{j,i+1} \right) \right] = - \sum_{j=0}^{J-1} \left[\sum_{i=0}^{I_j-1} \left(\vec{x}_{j,i} \times \vec{x}_{j,i+1} \right) \cdot \hat{n}_{\Gamma_j} C_{\Gamma_j} \right]. \tag{2.12}$$

Split off the slicing plane from the equation:

$$6V_{st} = - \sum_{i=0}^{I_c-1} \left(\vec{x}_{c,i} \times \vec{x}_{c,i+1} \right) \cdot \hat{n}_{\Gamma_c} C_{\Gamma_c} - \sum_{\substack{j=0 \\ j \neq c}}^{J-1} \left[\sum_{i=0}^{I_j-1} \left(\vec{x}_{j,i} \times \vec{x}_{j,i+1} \right) \cdot \hat{n}_{\Gamma_j} C_{\Gamma_j} \right]. \tag{2.13}$$

Define the unknown polyhedron node cut by the plane, $\vec{x}_{c,i}$, by the slope of the plane, $\hat{n}_{\Gamma_c}$, the direction of the cut edge, $\hat{e}_{c,i}$, and the associated node along the edge located on the flux face, $\vec{x}_{0,i_0}$. $\hat{e}_{c,i}$ and $\vec{x}_{0,i_0}$ were determined earlier during the construction of the flux polyhedron:

$$\begin{aligned} \vec{x}^0_{c,i} &= \vec{x}_{0,i_0} - \frac{\hat{n}_{\Gamma_c} \cdot \vec{x}_{0,i_0}}{\hat{n}_{\Gamma_c} \cdot \hat{e}_{c,i}} \hat{e}_{c,i} = \vec{x}_{0,i_0} + \beta_{c,i} \hat{n}_{\Gamma_c} \cdot \vec{x}_{0,i_0} \hat{e}_{c,i} \\ \vec{x}_{c,i} &= \vec{x}^0_{c,i} - \frac{C_{\Gamma_c}}{\hat{n}_{\Gamma_c} \cdot \hat{e}_{c,i}} \hat{e}_{c,i} = \vec{x}^0_{c,i} + \beta_{c,i} C_{\Gamma_c} \hat{e}_{c,i}. \end{aligned} \tag{2.14}$$

Substitute the parameterization in Eq. (2.14) and simplify:

$$\begin{aligned} \sum_{i=0}^{I_c-1} \left(\vec{x}_{c,i} \times \vec{x}_{c,i+1} \right) \cdot \hat{n}_{\Gamma_c} C_{\Gamma_c} = & \sum_{i=0}^{I_c-1} \left(\hat{e}_{c,i} \times \hat{e}_{c,i+1} \right) \beta_{c,i+1} \beta_{c,i} \cdot \hat{n}_{\Gamma_c} C^3_{\Gamma_c} \\ & + \sum_{i=0}^{I_c-1} \left[\hat{e}_{c,i} \times \left(\vec{x}^0_{c,i+1} - \vec{x}^0_{c,i-1} \right) \right] \beta_{c,i} \cdot \hat{n}_{\Gamma_c} C^2_{\Gamma_c} \\ & + \sum_{i=0}^{I_c-1} \left(\vec{x}^0_{c,i} \times \vec{x}^0_{c,i+1} \right) \cdot \hat{n}_{\Gamma_c} C_{\Gamma_c}. \end{aligned} \tag{2.15}$$

Rewrite the equation using the following terms: $\vec{M}_{\Gamma_c} = \sum_{i=0}^{I_c-1} \left(\hat{e}_{c,i} \times \hat{e}_{c,i+1}\right) \beta_{c,i}\beta_{c,i+1}$, $\vec{L}_{\Gamma_c} = \sum_{i=0}^{I_c-1} \left[\hat{e}_{c,i} \times \left(\vec{x}^0_{c,i-1} - \vec{x}^0_{c,i+1}\right)\right] \beta_{c,i}$, and $\vec{K}_{\Gamma_c} = \sum_{i=0}^{I_c-1} \left(\vec{x}^0_{c,i} \times \vec{x}^0_{c,i+1}\right)$:

$$\sum_{i=0}^{I_c-1} \left(\vec{x}_{c,i} \times \vec{x}_{c,i+1}\right) \cdot \hat{n}_{\Gamma_c} C_{\Gamma_c} = \vec{M}_{\Gamma_c} \cdot \hat{n}_{\Gamma_c} C^3_{\Gamma_c} + \vec{L}_{\Gamma_c} \cdot \hat{n}_{\Gamma_c} C^2_{\Gamma_c} + \vec{K}_{\Gamma_c} \cdot \hat{n}_{\Gamma_c} C_{\Gamma_c}. \quad (2.16)$$

Recognize that face j can be parameterized exactly as c, $x_{j,i} = x^0_{j,i} + \beta_{j,i} C_{\Gamma_j} \hat{e}_{j,i}$, where $\hat{e}_{j,i} = 0$, $\beta_{j,i} = 0$, and $x^0_{j,i} = x_{j,i}$ when i is not an adjoining node of face c. Then recognize that for $j \neq c$, faces will have one less power in C_{Γ_c} than face c:

$$\sum_{i=0}^{I_j-1} \left(\vec{x}_{j,i} \times \vec{x}_{j,i+1}\right) \cdot \hat{n}_{\Gamma_j} C_{\Gamma_j} = \vec{M}_{\Gamma_j} \cdot \hat{n}_{\Gamma_j} C_{\Gamma_j} C^2_{\Gamma_c} + \vec{L}_{\Gamma_j} \cdot \hat{n}_{\Gamma_j} C_{\Gamma_j} C_{\Gamma_c} + \vec{K}_{\Gamma_j} \cdot \hat{n}_{\Gamma_j} C_{\Gamma_j}. \quad (2.17)$$

After substituting the parameterizations and the known terms, a cubic equation is revealed, the solution of which positions the bounding plane, C_{Γ_c}:

$$6V_{st} = \alpha_3 C^3_{\Gamma_c} + \alpha_2 C^2_{\Gamma_c} + \alpha_1 C_{\Gamma_c} + \alpha_0, \quad (2.18)$$

where $\alpha_3 = -\vec{M}_{\Gamma_c} \cdot \hat{n}_{\Gamma_c}$, $\alpha_2 = -\vec{L}_{\Gamma_c} \cdot \hat{n}_{\Gamma_c} - \sum_{j=0,j\neq c}^{J-1} \vec{M}_{\Gamma_j} \cdot \hat{n}_{\Gamma_j} C_{\Gamma_j}$, $\alpha_1 = -\vec{K}_{\Gamma_c} \cdot \hat{n}_{\Gamma_c} - \sum_{j=0,j\neq c}^{J-1} \vec{L}_{\Gamma_j} \cdot \hat{n}_{\Gamma_j} C_{\Gamma_j}$, and $\alpha_0 = -\sum_{j=0,j\neq c}^{J-1} \vec{K}_{\Gamma_j} \cdot \hat{n}_{\Gamma_j} C_{\Gamma_j}$. Substitute C_{Γ_c} into the parameterization of Eq. (2.14) for the unknown node positions to full specify the flux polyhedron, as shown in Figure 4.

2.2. *Truncation of the flux polyhedron*

Because only the tagged volume is tracked, the flux polyhedron should be truncated by the PLIC surface. Further, in order to prevent overlap between adjacent cells sharing a common edge, the flux polyhedron needs to be truncated by the donor cell faces that are adjacent to the flux face. The resulting flux polyhedron volume is donated from the donor to the receiver cell, completing the local advection step. Truncation procedures are described in López & Hernández (2008), the details of which are repeated here because a clarification needs to be made in regards to the non-convexity of the flux polyhedron.

Calculate the signed distance, ϕ_p, from every vertex, $\vec{x}_p$, to the truncation plane, T, using $\phi_p = \hat{n}_{\Gamma_T} \cdot \vec{x}_p + C_{\Gamma_T}$. An edge is defined to be intersected if the relative signs of ϕ for the two nodes bounding the edge change sign. The distance functions can be used to locate the intersected points. For example, if an intersected edge is bound by $\vec{x}_1$ and $\vec{x}_2$, the intersection position can be found using $\vec{x}_t = \vec{x}_1 - \frac{\phi_1}{\phi_2 - \phi_1}\left(\vec{x}_2 - \vec{x}_1\right)$, where t is the index of the node created by the intersection.

The truncation procedure removes the nodes with a negative ϕ and use the the vertices with positive ϕ and the intersected nodes to build up the truncated polyhedron. Due to the non-convexity of the flux region, the truncation procedure can slice the polyhedron in multiple sections, introducing multiple new faces to the polyhedron, or, possibly, creating multiple polyhedra. Because the non-convexity of the flux region is restricted to two adjacent planes, created by the added planarity-preserving edge between two velocity edges, the truncation procedure can introduce up to two new faces to the polyhedron or separate the polyhedron into two polyhedra. In order to use convex truncation procedures, the polyhedron is broken up into a set of convex tetrahedra for the duration of the operation. Nevertheless, because the non-convexity is mild, the truncation procedure usually operates as if the polyhedron is convex, as shown by in Figures 5-7. The truncation step is costly relative to the rest of the algorithm, so the number of reconstructed

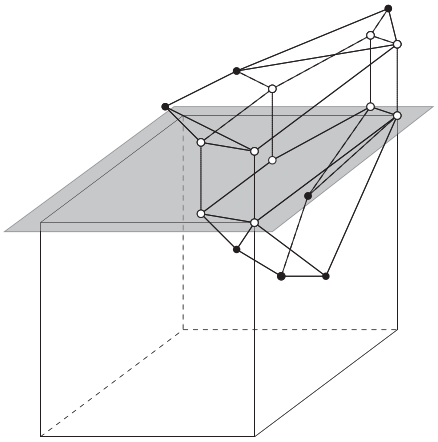

FIGURE 5. Truncation of the flux polyhedron by the top face.

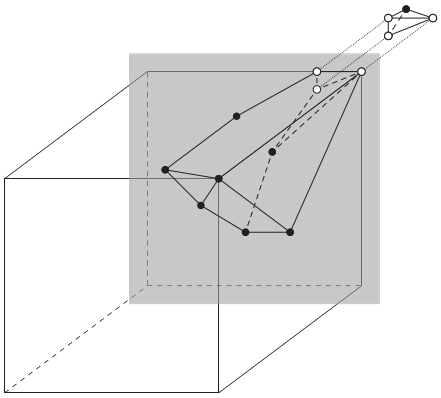

FIGURE 6. Truncation of the flux polyhedron by the back face.

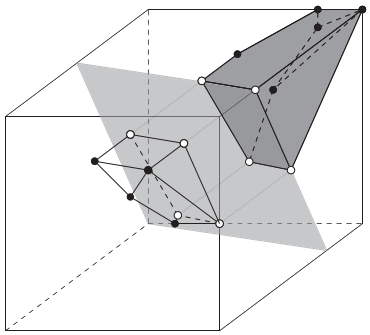

FIGURE 7. Truncation of the polyhedron by the PLIC plane. Shaded region volume is donated to the receiver cell.

tetrahedra should be minimized; this is accomplished by minimizing the number of surfaces bounding the non-convex polyhedron. Linear triangulation of a non-planar surface produces more flux polyhedron faces than the procedure described in Section 2.1.

2.3. *Handling of inward velocities*

If there exists a node where $\text{sign}\,(\vec{v}_{fa} \cdot \hat{n}_{fa}) \neq \text{sign}\,(\vec{v}_{no} \cdot \hat{n}_{no})$, then there is an inward velocity. Inward velocities have the potential to create a self-intersecting flux polyhedron, where some of the resulting flux will be be donated from the receiver cell to the donor cell. We can only determine whether the inward velocity has caused the flux polyhedron to be self-intersecting once we have tried to place the bounding face to match the stream tube volume. Essentially, if the assumption of self-intersection does not permit a placement of the bounding face such that stream tube volume is matched, then it is not self-intersecting. So, first assume the inward velocity makes the polyhedron self-intersecting

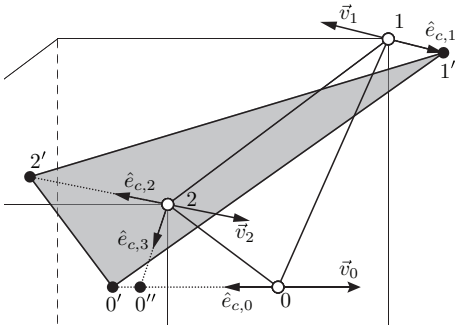

FIGURE 8. Self-intersecting case has velocity-based edges propagate in both directions. Note that a planarity edge is not needed between nodes with velocities in opposing directions.

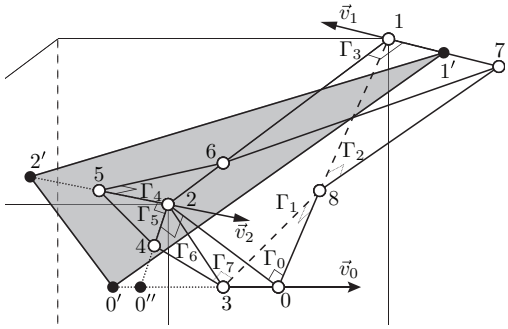

FIGURE 9. The self-intersecting has the capping plane slice the flux face.

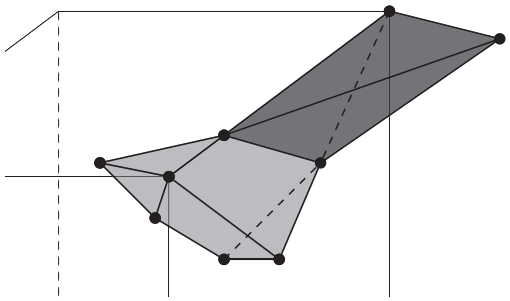

FIGURE 10. The self-intersected flux polyhedron has both positive and negative fluxed volume.

and define the geometry as such. If the plane cannot be placed in such a way that it slices the flux face, then the situation reverts to a case where all the flux is outward and the geometry is defined by the traditional method described above.

The self-intersecting case is systematically constructed in Figures 8-10. The program is essentially the same as the one described above. The only caveat is that there is no need to add planarity-preserving edges to the regions between the nodes that have their velocities pointing in opposite directions; instead, the edges are constrained such that they connect to an edge bounding the triangular flux face section to preserve planarity. This behavior is illustrated in Figure 9 by nodes 6 and 8. Note that the auxiliary nodes used to define the normal of bounding plane still follow the relation, $\vec{x}'_p = \vec{x}_p - \vec{v}_p \Delta t$; however, this definition causes one of the nodes to propagate into the receiving cell. Also, because the volume is negative, all normals for the faces living in the receiving cell, and the associated local indices, need to be reversed to use Eq. (2.12) and (2.18). Note that the bounding plane is considered as one face in this definition and its normal still points outward. If there is a negative volume region, truncation operations for the cell faces and the PLIC planes need to be performed for both the donor and receiver cell. Further, if there are two inward velocities that are non-planar, the planarity-based edges need to be defined using the maximum length scale defined by the receiver cell and the associated face adjacent neighbor.

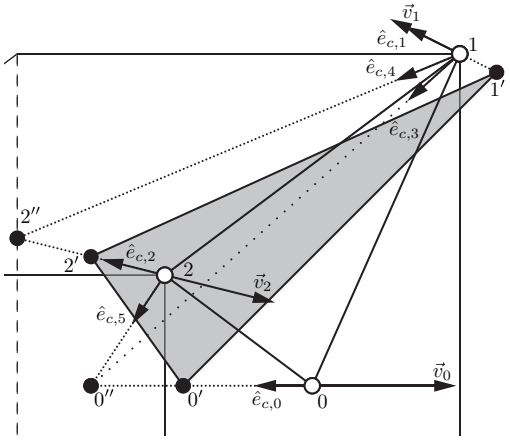

FIGURE 11. Non-intersecting case has the velocity-based edge of the inward velocity flipped because the volume could not be conserved with self-intersection.

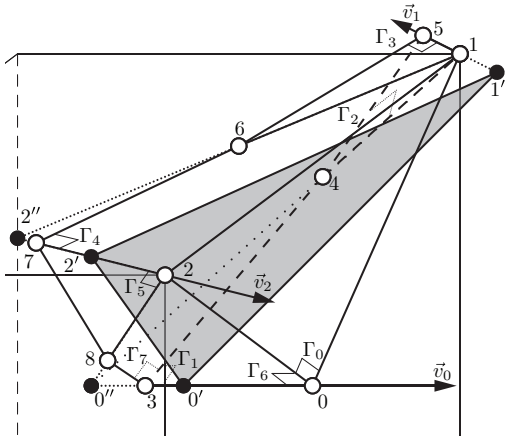

FIGURE 12. In the non-intersecting case, the plane does not slice the flux face even though there was an inward velocity

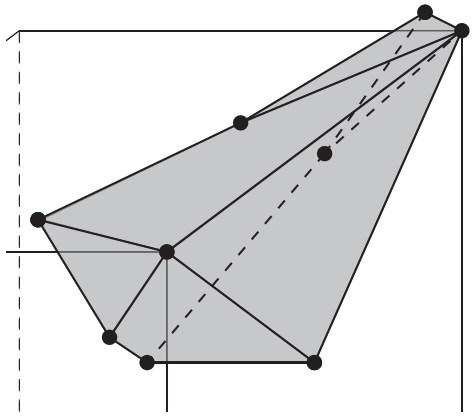

FIGURE 13. The non-intersected flux polyhedron has only positive fluxed volume..

As mentioned above, if the self-intersecting geometry does not permit placement of the capping plane under the limits on C_{Γ_c}, defined by the flux face nodes and cell center positions, then the flux polyhedron falls back to the traditional description above, where more edges need to be added for planarity. The only difference is that the edge unit vector for the inward velocity is parallel to velocity vector, instead of anti-parallel. This is illustrated by $\hat{e}_{c,1}$ in Figure 11. The normal of the bounding face has the same orientation as if it were self-intersecting; however, the flux polyhedron in Figure 13 is geometrically similar to that in Figure 4.

2.4. *Handling overlap due to bounding face orientation and position*

As currently defined, the edge-matched flux polyhedron, truncated by the cell faces, can still overlap. The problem will first be described in 2D, where the overlap is apparent.

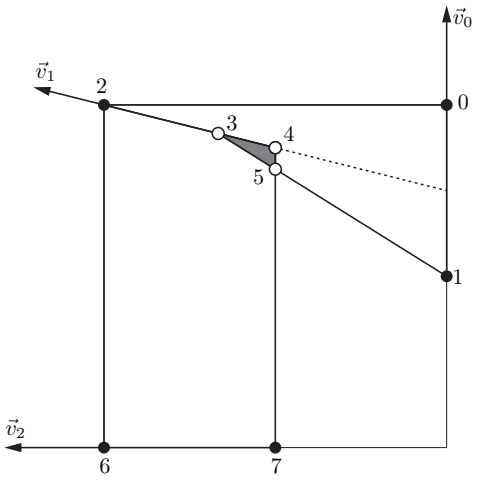

FIGURE 14. 2D schematic of the overlap between adjacent flux polyhedra.

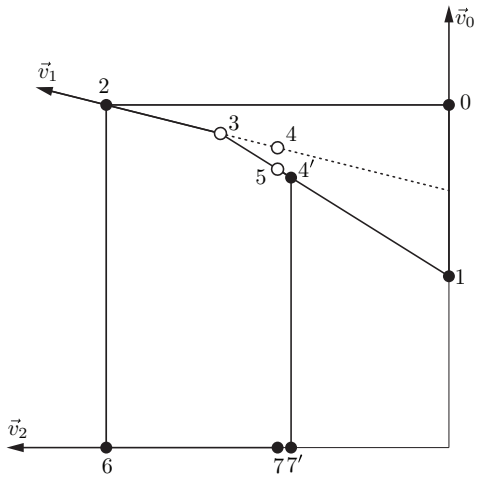

FIGURE 15. 2D schematic of the corrected flux polyhedra to prevent overlap.

Note that in a 2D description, faces are replaced by edges. Following the 2D schematic in Figure 14, the flux polyhedron defined by the nodes 0, 2, 3, and 1 overlap with the adjacent flux polyhedron defined by nodes 4, 2, 6, and 7 over the region bound by nodes 3, 4, and 5. This overlap can be easily accounted for in the volume enforcement procedure described in Section 2.1.1 by storing and using the bounding edge normal and constant from the antecedently reconstructed flux polyhedra of the cell. Figure 15 illustrates the correction for the case where the flux polyhedron emanating from the top face is reconstructed first, so that the flux polyhedron emanating from the left face is modified. The flux polyhedron originally defined by nodes 4, 2, 6, and 7 conservatively deforms such that it is bound by nodes $4'$, 3, 2, 6 and $7'$. Basically the edge bound by nodes 2 and 3 is added to the left face flux polyhedron and the volume enforcement procedure is repeated, using the bounding edge of the top face flux polyhedron, parallel to the edge bound by nodes 3 and 1, as the direction of propagation in lieu of the edge anti-parallel to $\vec{v}_1$. The procedure is repeated for all subsequent flux polyhedra of the cell, namely from the bottom and right faces.

The procedures are very similar in 3D; however, the description is convoluted by the addition of planarity-preserving faces. Store the antecedently reconstructed flux polyhedra bounding surface normals and constants of the cell. If during the volume enforcement procedure, the subsequently reconstructed flux polyhedron's bounding surface intersects an adjacent polyhedron's bounding surface, copy the face(s) shared by the adjacent flux polyhedron, bound by the two respective velocity-based edges, potentially the planarity-based edge, and the edge shared with the antecedently reconstructed flux polyhedron's capping face, into the subsequently reconstructed flux polyhedron. This is equivalent to adding the edge defined by nodes 2 and 3 in the 2D description of Figure 15. Then repeat the volume enforcement procedure, replacing the propagation path defined by velocity-based edges and potentially the planarity-based edge with an edge parallel to the antecedently reconstructed flux polyhedron's bounding face. This is equivalent to the direction defined by the edge bound by nodes 3 and 1 in the 2D description of Figure 15. Repeat for all subsequently reconstructed flux polyhedra of the cell.

3. Concluding remarks and future work

This brief describes a new PLIC-VOF advection method on general grids in 3D. The method formally conserves the local volume and has an empirical accuracy between first- and second-order. The method uses the flux polyhedron construct to locally approximate a stream tube emanating from the flux face, perturbing the definition in order to remove all over-/underlap to prevent the volume fraction from over-/undershooting. Kinematic test cases on unstructured grids have been performed; the results of the study, including estimation of cost and accuracy in comparison to contemporary algorithms, will be included in a future publication.

Future research will include: incorporation of EMFPA-3D into the mass and momentum advection operators in an incompressible Navier-Stokes solver that uses a node-based finite volume method, estimation of the surface normal and curvature on unstructured grids using height functions, augmentation of the volume enforcement procedure to have an orientable and shiftable capping surface, and incorporation of hanging nodes.

Acknowledgements

This work was supported by the Department of Energy Computational Science Graduate Fellowship under grant number DE-FG02-97ER25308 and by the Stanford Graduate Fellowship. The authors wish to thank Sanjeeb Bose, Paul Covington, Olivier Desjardins, Frank Ham, and Vincent Le Chenadec for their helpful discussions during the development of EMFPA-3D.

REFERENCES

Hernández, J., López, J., Gómez, P., Zanzi, C. & Faura, F. 2008 A new volume of fluid method in three dimensions – Part I: Multidimensional advection method with face-matched flux polyhedra. *Int. J. Numer. Meth. Fluids* **58**, 897–921.

López, J. & Hernández, J. 2008 Analytical and geometrical tools for 3D volume of fluid methods in general grids. *J. Comput. Physics* **227**, 5939–5948.

López, J., Hernández, J., Gómez, P. & Faura, F. 2004 A volume of fluid method based on multidimensional advection and spline interface reconstruction. *J. Comput. Physics* **195**, 718–742.

Parker, B. J. & Youngs, D. L. 1992 Two and three dimensional Eulerian simulation and fluid flow with material interfaces. *Tech. Rep.* 01/92. AWE.

Scardovelli, R. 2000 Analytical relations connecting linear interfaces and volume fractions in rectangular grids. *J. Comput. Physics* **164**, 228–237.

Scardovelli, R. & Zaleski, S. 1999 Direct Numerical Simulation of free-surface and interfacial flow. *Annu. Rev. Fluid Mech.* **31**, 567–603.

Weymouth, G. D. & Yue, D. K. P. 2010 Conservative Volume-of-Fluid method for free-surface simulations on Cartesian-grids. *J. Comput. Physics* **229**, 2853–2865.

Yang, X. & James, A. J. 2006 Analytic relations for reconstructing piecewise linear interfaces in triangular and tetrahedral grids. *J. Comput. Physics* **214**, 41–54.