Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 227 (2008) 5939–5948
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Short Note

Analytical and geometrical tools for 3D volume of fluid methods in general grids

J. López [a,*], J. Hernández [b]

[a] Departamento de Ingeniería de Materiales y Fabricación, ETSII, Universidad Politécnica de Cartagena, E-30202 Cartagena, Spain
[b] Departamento de Mecánica, ETSII, UNED, E-28040 Madrid, Spain

Received 11 August 2007; received in revised form 4 March 2008; accepted 10 March 2008
Available online 20 March 2008

Abstract

It is well known that volume of fluid (VOF) methods in three-dimensions, especially those based on unsplit advection schemes, involve highly complex geometrical operations. The objective of this work is to propose, for general grids and three-dimensional Cartesian geometry, simple and efficient geometrical tools for volume truncation operations that typically arise in VOF methods and an analytical method for local volume enforcement. The results obtained for different tests and grid types show that the proposed analytical method may be as much as three times faster than Brent's iterative method. Advection tests were carried out using hexahedral grids obtained from deformation of a cubic grid to assess the accuracy of the proposed tools in combination with a recently proposed unsplit PLIC–VOF method.

Keywords: Volume tracking; Volume of fluid method; Volume truncation; Local volume conservation enforcement

1. Introduction

One of the most widely used methods for simulating interfacial flows is the volume of fluid (VOF) method. Algebraic formulations in VOF methods are being replaced by purely geometrical procedures with PLIC (piecewise linear interface calculation) reconstruction of the interface, which provide second-order accuracy. Numerous successful implementations of high-order PLIC–VOF methods have been developed in two dimensions (2D) during the last two decades (see for example the reviews of Scardovelli and Zaleski [14] or Rider and Kothe [13] and the references therein). However, the great complexity of the geometrical operations involved in these methods makes their extension to three dimensions (3D) relatively difficult (successful implementations of high-order PLIC–VOF methods in 3D can be found in [2,4,5,8–10,12]).

Two basic operations are involved in any geometrical PLIC–VOF method: volume truncation and the enforcement of local volume conservation. The first typically arises in the computation of the fluid volume advected through cell boundaries, a volume obtained from the intersection between the reconstructed interface

* Corresponding author. Tel.: +34 968325962; fax: +34 968326445.
E-mail address: joaquin.lopez@upct.es (J. López).

0021-9991/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2008.03.010

and the donating flux regions constructed at cell boundaries. In spatially varying velocity fields, the use of unsplit advection schemes gives rise to discretized flux regions that are generally irregular convex polyhedra (polygons in 2D), thus introducing additional complexity.

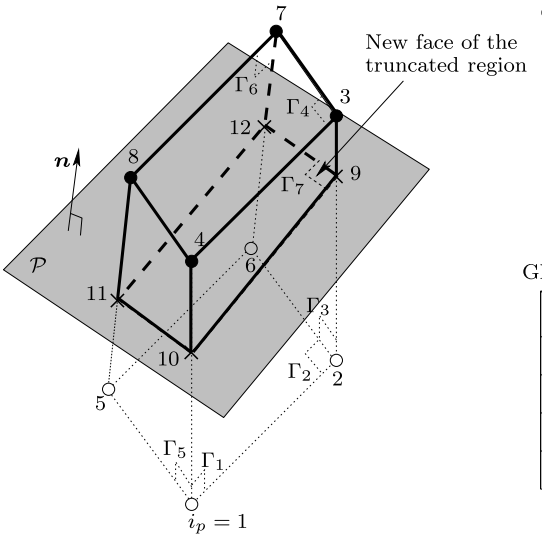
The second operation consists of determining the position of a planar interface (with a given orientation) that produces a truncated polyhedron of a given volume. This problem is usually encountered as part of the interface reconstruction step, in which local volume conservation is enforced in each interfacial cell of the computational domain. The solution to this problem may also be useful in the advection step, to ensure that the volume of the flux region constructed at a given cell boundary satisfies the local conservation constraint given by the computed volumetric flux through the cell boundary (see Refs. [4,6,7]). The analytical solution to this problem eliminates the need to iterate, generally reducing CPU time. Scardovelli and Zaleski [15] and Yang and James [18] solved this problem analytically for orthogonal hexahedral and tetrahedral grids, respectively. However, for general grids, or even for orthogonal hexahedral grids when an unsplit advection scheme is used and the flux regions constructed at cell boundaries are non-orthogonal, a more general analytical method is needed.

The tools for volume truncation operations and the analytical method to enforce local volume conservation in general grids are described in Sections 2 and 3, respectively. The proposed analytical method is assessed in Section 4 by comparing its efficiency with that of an iterative method typically used in PLIC–VOF methods. The source codes and pseudo-codes for the proposed algorithms are available for download at [19].

2. Polyhedron truncation procedure

In this section, a general procedure for obtaining the truncated polyhedron resulting from the intersection between a plane $\mathscr{P}$ and a generic convex polyhedron Ω, either regular or irregular, will be described. A relatively similar procedure can be found in the work of Stephenson and Christiansen [17], although few implementation details are given by these authors. The truncation procedure is carried out in 2D and 3D problems using two different algorithms, the corresponding codes and pseudo-codes of which are available at [19]. The output of the algorithms is the set of ordered vertices of the truncated polyhedron or polygon.

Let us consider a polyhedron of J faces, with I_j vertices in each face j, and a plane $\mathscr{P}$, defined by $\boldsymbol{n}\cdot\boldsymbol{x}+C=0$, where $\boldsymbol{n}$ is the unit-length vector normal to $\mathscr{P}$, $\boldsymbol{x}$ is the position vector of a generic point on $\mathscr{P}$ and C is a constant. Fig. 1 shows an example of the arrangement of polyhedron vertices using a global



Global i_p vertex indices for the original polyhedron

i \ j	1	2	3	4	5	6
1	1	2	3	4	1	6
2	2	1	2	3	4	5
3	3	5	6	7	8	8
4	4	6	7	8	5	7

Global i_p vertex indices for the truncated polyhedron

i \ j	1	2	3	4	5	6	7
1	9	−−	3	4	10	11	9
2	3	−−	9	3	4	8	10
3	4	−−	12	7	8	7	11
4	10	−−	7	8	11	12	12

Discarded face

New face

Fig. 1. Truncation of a polyhedron by a plane $\mathscr{P}$. Symbols • and ○ denote vertices with positive and negative ϕ values, respectively, and × denotes the intersection point between plane $\mathscr{P}$ and a polyhedron edge.

vertex index, i_p, assigned to every vertex index i of face j ($\boldsymbol{x}_{j,i} \equiv \boldsymbol{x}_{i_p}$). The signed distance, ϕ_{i_p}, from every vertex, $\boldsymbol{x}_{i_p}$, of Ω to $\mathscr{P}$ is computed as $\phi_{i_p} = \boldsymbol{n} \cdot \boldsymbol{x}_{i_p} + C$. Note that if the normal vector $\boldsymbol{n}$ points to $\boldsymbol{x}_{i_p}$, the sign of ϕ_{i_p} will be positive. Whether or not intersection between Ω and $\mathscr{P}$ exists can be determined by comparing the relative signs of ϕ_{i_p} values. The truncated polyhedron (edges drawn with thicker lines in the example of Fig. 1) will be defined by the vertices where the signed distance function is positive and by the points of intersection between $\mathscr{P}$ and the edges of Ω. In the example of Fig. 1, in which one of these edges is defined by the two adjacent vertices, $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$, the resulting position vector of the intersection point is obtained as $\boldsymbol{x}_9 = \boldsymbol{x}_2 - \frac{\phi_2}{\phi_3 - \phi_2}(\boldsymbol{x}_3 - \boldsymbol{x}_2)$.

In 2D, a new vertex index, corresponding to an intersection point between line $\mathscr{P}$ and the polygon considered, is inserted between every two adjacent polygon vertices with opposite ϕ values. The vertices of the truncated polygon are then arranged in counterclockwise order.

In 3D, the truncation procedure begins by discarding the polyhedron faces in which all the vertices have a negative ϕ value (face Γ_2 of Fig. 1). Next, the array of the ordered global indices corresponding to the vertices of the new face, Γ_{J+1}, of the truncated polyhedron (face Γ_7 in Fig. 1) is constructed. Finally, the new array of global vertex indices for every truncated face of the polyhedron (faces Γ_1, Γ_3, Γ_5 and Γ_6 in Fig. 1) is constructed following a procedure similar to that indicated above for polygon truncation in 2D problems.

Note that the polygon truncation procedure described above could also be applied to non-convex polygons with no loss of generality. In 3D, this would require a small modification to allow for more than one new face in the resulting truncated polyhedron. However, since the present work is focused on the context of VOF methods, for which all the polyhedra arising during computation are generally convex, the modification of the algorithm for more general applications will not be presented here. It should be mentioned that the intersection between a convex polyhedron and a plane also produces a convex polyhedron, and so subsequent truncation operations will be made over convex polyhedra.

3. Local volume enforcement

The problem consists of determining the constant C of a plane $\mathscr{P}$ with a given normal $\boldsymbol{n}$, which truncates a polyhedron Ω, producing a polyhedron, Ω_T, with a given volume, V_T. The procedure proposed in this section consists of bracketing the solution followed by applying an analytical method valid for the truncation of any convex polyhedron in general grids.

The volume of the polyhedron can be expressed as [16]

$$V = \frac{1}{6}\sum_{j=1}^{J}\left[(\boldsymbol{n}_{\Gamma j}\cdot\boldsymbol{x}_{j,1})\boldsymbol{n}_{\Gamma j}\cdot\sum_{i=1}^{I_j}(\boldsymbol{x}_{j,i}\times\boldsymbol{x}_{j,i+1})\right] \tag{1}$$

(subscript $i+1$ must be replaced by 1 for $i = I_j$). Here, $\boldsymbol{n}_{\Gamma j}$ is the unit-length vector normal to face Γ_j (pointing outwards from the polyhedron) and $\boldsymbol{x}_{j,i}$ is the position vector of the counterclockwise (viewed from outside the polyhedron) ordered vertex i of face Γ_j. The constant C will be obtained by imposing the condition $V = V_T$ in Eq. (1). Since the unit-length vector normal to each face of the truncated polyhedron, Ω_T, is chosen to point outwards from the polyhedron, the plane containing the new face Γ_c of Ω_T will be given by $\boldsymbol{n}_{\Gamma c}\cdot\boldsymbol{x} + C_{\Gamma c} = 0$, with $\boldsymbol{n}_{\Gamma c} = -\boldsymbol{n}$ and $C_{\Gamma c} = -C$ (see the examples of Figs. 2(a) and (b)). The bracketing and analytical steps of the proposed procedure used to determine the constant $C_{\Gamma c}$ are described in the following two sections.

3.1. Bracketing the solution

The vertices of the original polyhedron that will remain in the truncated polyhedron are first determined using a bracketing procedure. This part of the algorithm may be the most time consuming, especially when the number of vertices of Ω is large.

The procedure consists of successively truncating the polyhedron Ω by the planes parallel to $\boldsymbol{n}\cdot\boldsymbol{x} = 0$ passing through its different vertices, and comparing the corresponding truncated volumes with V_T, thus determining whether the distance ϕ_{i_p} from each vertex i_p to the plane containing Γ_c is positive (if $V_{i_p} < V_T$, where V_{i_p} is

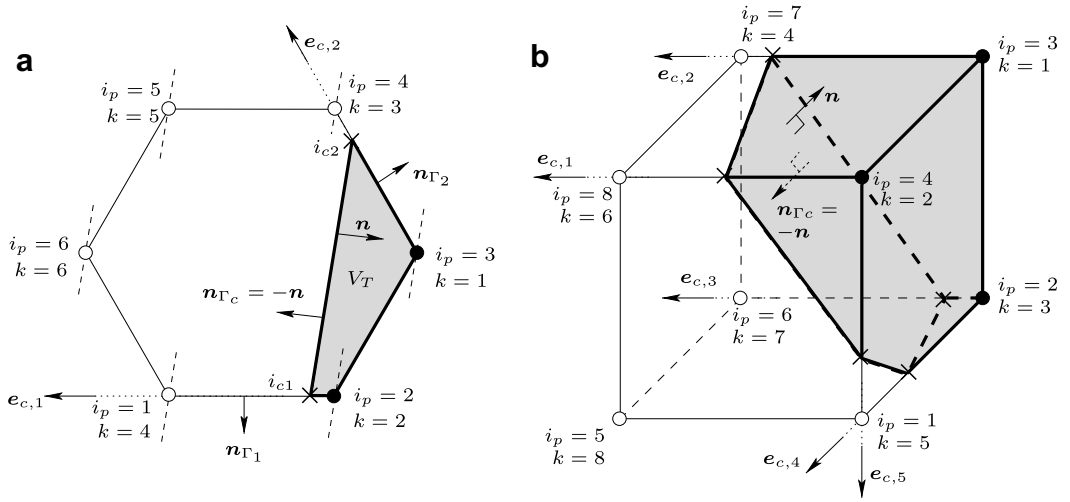


Fig. 2. Schematic representation of the procedure used to determine the location of the interface Γ_c: (a) two-dimensional case; (b) three-dimensional case.

the volume truncated by the plane which goes through vertex i_p) or negative (if $V_{i_p} > V_T$). To improve the efficiency of the algorithm, the I_p vertices of Ω are first ordered from higher to lower distance, $\phi^0_{i_p} = \boldsymbol{n} \cdot \boldsymbol{x}_{i_p}$, to the plane $\boldsymbol{n} \cdot \boldsymbol{x} = 0$, parallel to $\mathscr{P}$ and passing through the origin, using index $k = 1, \ldots, I_p$ (for example, in Fig. 2(a), $\phi^0_3 > \phi^0_2 > \phi^0_4 > \phi^0_1 > \phi^0_5 > \phi^0_6$).

The first truncation plane considered in the bracketing procedure will be that passing through the vertex corresponding to the central index $k_c = \mathtt{INT}[(I_p + 1)/2]$ of the ordered sequence of $\phi^0_{i_p}$ values (vertex index $i_p = 4$ and $k_c = 3$ in the case of Fig. 2(a), and vertex index $i_p = 7$ and $k_c = 4$ in Fig. 2(b)). Depending on the value of the resulting truncation volume, V_{k_c}, the central index k_c is assigned to the bracketing index $k_{\max}$ if $V_T < V_{k_c}$, or to $k_{\min}$ if $V_T > V_{k_c}$ (in both cases of Figs. 2(a) and (b), $k_{\max} = k_c$), and planes passing through vertices with the preceding $(k_{\max} - 1)$ or subsequent $(k_{\min} + 1)$ indices are used, respectively, to obtain the next truncated volume. Values I_p and 1 will be set as initial guesses for the bracketing indices $k_{\max}$ and $k_{\min}$, respectively. The process will finish when the difference between $k_{\max}$ and $k_{\min}$ reaches unity. Obviously, every vertex, i_p, for which $V_{i_p} < V_T$ (and so the corresponding index k is lower than $k_{\max}$) will have a positive value of the distance to Γ_c, ϕ (black circles in Fig. 2), whereas the rest of vertices will have a negative ϕ value (open circles in Fig. 2). Note that the solution for $C_{\Gamma c}$ being sought is bracketed between the two values $C_{\max} = \phi^0(k_{\min})$ and $C_{\min} = \phi^0(k_{\max})$.

When the number of vertices I_p of the polyhedron to be truncated is large, the above procedure may be relatively slow. The consumed CPU time, which depends on the value of the volume fraction V_T/V_Ω, is, on average, in the order of $\mathrm{O}(\frac{1}{4}I_p + 1)$ (note that the time consumed by the simpler procedure of sorting in ascending or descending order may be as much as double). An alternative procedure, based on selecting the plane passing through the vertex with index $(k_{\max} + k_{\min})/2$ to obtain the next truncated volume, consumes a CPU time in the order of $\mathrm{O}(\log_2 I_p)$. This choice has been used for $I_p > 8$ (note that $\log_2 I_p$ becomes lower than $\frac{1}{4}I_p + 1$ for $I_p > 8$).

When the number of vertices with a positive distance function to Γ_c is larger than that of the rest of the polyhedron vertices, the inverse problem is solved using the algorithm proposed above. This consists of determining the position of the plane with normal vector $-\boldsymbol{n}$ that truncates Ω into a polyhedron of volume $V_\Omega - V_T$, which involves a lower number of vertices and so requires a lower computational time.

3.2. Analytical solution

Once the solution to the problem has been bracketed, the next step is to determine the edges of the polyhedron Ω that join vertices located on both sides of the interface. Then, the position vector, $\boldsymbol{x}_{c,i}$, of every intersection point between the interface plane Γ_c, given by $\boldsymbol{n}_{\Gamma c} \cdot \boldsymbol{x} + C_{\Gamma c} = 0$, and the polyhedron edges can be

expressed analytically. Let us denote by $\boldsymbol{x}_{c,i}^{\text{out}}$ and $\boldsymbol{x}_{c,i}^{\text{in}}$ the vertices of the polyhedron edge L_i ($i = 1, \ldots, I_c$, where I_c is the number of vertices of the new face) passing through point $\boldsymbol{x}_{c,i}$ (in the case of Fig. 2(a), vertices $i_p = 1$ and 2 and $i_p = 4$ and 3 defining edges L_1 and L_2, respectively, intersecting Γ_c at $\boldsymbol{x}_{c,1}$ and $\boldsymbol{x}_{c,2}$). For convenience, $\boldsymbol{x}_{c,i}$ will be expressed as a function of the constant $C_{\Gamma c}$,

$$\boldsymbol{x}_{c,i} = \boldsymbol{x}_{c,i}^{0} + \beta_{c,i} C_{\Gamma c} \boldsymbol{e}_{c,i}, \tag{2}$$

where $\beta_{c,i} = -1/(\boldsymbol{n}_{\Gamma c} \cdot \boldsymbol{e}_{c,i})$, $\boldsymbol{e}_{c,i} = (\boldsymbol{x}_{c,i}^{\text{out}} - \boldsymbol{x}_{c,i}^{\text{in}})/|\boldsymbol{x}_{c,i}^{\text{out}} - \boldsymbol{x}_{c,i}^{\text{in}}|$ and $\boldsymbol{x}_{c,i}^{0}$ is the position vector of the intersection point between edge L_i and the plane defined by $\boldsymbol{n}_{\Gamma c} \cdot \boldsymbol{x} = 0$ (parallel to Γ_c and passing through the origin of the coordinate system), which may be expressed as

$$\boldsymbol{x}_{c,i}^{0} = \boldsymbol{x}_{c,i}^{\text{in}} + \beta_{c,i} (\boldsymbol{n}_{\Gamma c} \cdot \boldsymbol{x}_{c,i}^{\text{in}}) \boldsymbol{e}_{c,i}. \tag{3}$$

The next step is to order the vertices of the truncated polyhedron (i.e., those of the original polyhedron with $\phi > 0$ and those of the new face defined by Eq. (2)) using the criterion indicated in Section 2. Then, introducing Eq. (2) and the position vectors of the rest of the truncated polyhedron vertices into Eq. (1) and imposing the volume constraint given by $V = V_T$ yields an analytical expression for $C_{\Gamma c}$.

3.2.1. 2D problems

In a 2D problem

$$\alpha_2 C_{\Gamma c}^2 + \alpha_1 C_{\Gamma c} + \alpha_0 = 2V_T, \tag{4}$$

where

$$\alpha_2 = -(\boldsymbol{n}_{\Gamma 2}^{\perp} \cdot \boldsymbol{n}_{\Gamma 1}) \beta_{c,1} \beta_{c,2}, \tag{5}$$

$$\alpha_1 = -2(C_{\Gamma 2} \beta_{c,2} + C_{\Gamma 1} \beta_{c,1}), \quad \text{and} \tag{6}$$

$$\alpha_0 = -\left(\boldsymbol{n}_{\Gamma c} \cdot \boldsymbol{x}_{c,1}^{\text{in}}\right) C_{\Gamma 1} \beta_{c,1} - \left(\boldsymbol{n}_{\Gamma c} \cdot \boldsymbol{x}_{c,2}^{\text{in}}\right) C_{\Gamma 2} \beta_{c,2} + \sum_{\forall i | \phi_i, \phi_{i+1} > 0} \boldsymbol{x}_i \times \boldsymbol{x}_{i+1}, \tag{7}$$

where the summation is extended to every pair of adjacent vertices of the truncated polygon, except those located on Γ_c. Subscripts 1 and 2 refer to the edges of the original polygon, defined by $\boldsymbol{n}_{\Gamma 1} \cdot \boldsymbol{x} + C_{\Gamma 1} = 0$ and $\boldsymbol{n}_{\Gamma 2} \cdot \boldsymbol{x} + C_{\Gamma 2} = 0$, respectively, intersected by Γ_c, while $\boldsymbol{x}_{c,1}^{\text{in}}$ and $\boldsymbol{x}_{c,2}^{\text{in}}$ are the position vectors of the vertices of these edges that have a positive ϕ value (vertex indices $i_p = 2$ and 3, respectively, in the case of Fig. 2(a)). Subscripts 1 and 2 are chosen so that all the vertices of the truncated polygon with positive ϕ values can be followed in counterclockwise order from $\boldsymbol{x}_{c,1}^{\text{in}}$ to $\boldsymbol{x}_{c,2}^{\text{in}}$, so that $\boldsymbol{n}_{\Gamma 1}^{\perp} = -\boldsymbol{e}_{c,1}$ and $\boldsymbol{n}_{\Gamma 2}^{\perp} = \boldsymbol{e}_{c,2}$. Note that, when edges Γ_1 and Γ_2 are parallel, the coefficient α_2 is zero and Eq. (4) becomes a linear function of $C_{\Gamma c}$. In the example of Fig. 2(a), this would occur when $V_{i_p=4} < V_T < V_{i_p=1}$. Note that, within this range, the length of edge Γ_c is constant.

3.2.2. 3D problems

For a 3D case, after a relatively laborious manipulation, the following expression, equivalent to Eq. (4), is obtained:

$$\alpha_3 C_{\Gamma c}^3 + \alpha_2 C_{\Gamma c}^2 + \alpha_1 C_{\Gamma c} + \alpha_0 = 6V_T, \tag{8}$$

where

$$\alpha_3 = -\boldsymbol{M}_{\Gamma c} \cdot \boldsymbol{n}_{\Gamma c},$$

$$\alpha_2 = -\boldsymbol{L}_{\Gamma c} \cdot \boldsymbol{n}_{\Gamma c} - \sum_j \boldsymbol{M}_{\Gamma j} \cdot \boldsymbol{n}_{\Gamma j} C_{\Gamma j},$$

$$\alpha_1 = -\boldsymbol{K}_{\Gamma c} \cdot \boldsymbol{n}_{\Gamma c} - \sum_j \boldsymbol{L}_{\Gamma j} \cdot \boldsymbol{n}_{\Gamma j} C_{\Gamma j},$$

$$\alpha_0 = -\sum_j \boldsymbol{K}_{\Gamma j} \cdot \boldsymbol{n}_{\Gamma j} C_{\Gamma j},$$

in which the summations are extended over all the faces of the truncated polyhedron, except face Γ_c. In the above expressions, $C_{\Gamma j} = -\boldsymbol{n}_{\Gamma j} \cdot \boldsymbol{x}_{j,1}$,

$$\boldsymbol{K}_{\Gamma j} = \sum_{i=1}^{I_j} \boldsymbol{x}^0_{j,i} \times \boldsymbol{x}^0_{j,i+1},$$

$$\boldsymbol{L}_{\Gamma j} = \sum_{i=1}^{I_j} \boldsymbol{e}_{j,i} \times (\boldsymbol{x}^0_{j,i+1} - \boldsymbol{x}^0_{j,i-1})\beta_{j,i},$$

$$\boldsymbol{M}_{\Gamma j} = \sum_{i=1}^{I_j} (\boldsymbol{e}_{j,i} \times \boldsymbol{e}_{j,i+1})\beta_{j,i}\beta_{j,i+1}$$

(subscripts $i+1$ and $i-1$ must be replaced by 1 and I_j, respectively, for $i = I_j$ and $i = 1$). For vertices that are not placed on face Γ_c, $\boldsymbol{x}^0_{j,i} = \boldsymbol{x}_{j,i}$, $\boldsymbol{e}_{j,i} = (0,0,0)$ and $\beta_{j,i} = 0$.

By introducing Eq. (2) into the expression giving the area of the new face Γ_c,

$$A_c = \frac{1}{2}\boldsymbol{n}_{\Gamma_c} \cdot \sum_{i=1}^{I_c} \boldsymbol{x}_{c,i} \times \boldsymbol{x}_{c,i+1}, \tag{9}$$

where subscript $i+1$ must be replaced by 1 for $i = I_c$, and, by rearranging terms, A_c can be expressed as a function of $C_{\Gamma c}$

$$A_c = \frac{1}{2}\left(\boldsymbol{M}_{\Gamma c} \cdot \boldsymbol{n}_{\Gamma c} C^2_{\Gamma c} + \boldsymbol{L}_{\Gamma c} \cdot \boldsymbol{n}_{\Gamma c} C_{\Gamma c} + \boldsymbol{K}_{\Gamma c} \cdot \boldsymbol{n}_{\Gamma c}\right). \tag{10}$$

Then, combining Eqs. (10) and (8), the following expression is obtained:

$$-C_{\Gamma c}\left(2A_c + \sum_j \boldsymbol{M}_{\Gamma j} \cdot \boldsymbol{n}_{\Gamma j}\, C_{\Gamma j} + \sum_j \boldsymbol{L}_{\Gamma j} \cdot \boldsymbol{n}_{\Gamma j}\, C_{\Gamma j}\right) + \alpha_0 = 6V_T.$$

Note that, when A_c is constant or a linear function of $C_{\Gamma c}$ in the bracketed range determined in Section 3.1, Eq. (8) becomes, respectively, a linear or quadratic function of $C_{\Gamma c}$. Also note that A_c is linear when $\boldsymbol{M}_{\Gamma c} \cdot \boldsymbol{n}_{\Gamma c} = 0$ and constant when, additionally, $\boldsymbol{L}_{\Gamma c} \cdot \boldsymbol{n}_{\Gamma c} = 0$, which occurs if all $\boldsymbol{e}_{c,i}$ are parallel vectors.

Eqs. (4) and (8) can easily be solved analytically (see, for example, Refs. [1,11,15]). We have found that explicit formulae for cubic roots generally consume less computational time than Newton–Raphson iterations (a similar finding was observed in [15], although the relative advantage of both methods may depend on the computer, compiler and intrinsic function library used). Only in the very few cases in which roundoff errors may produce unstable analytical solutions are Newton–Raphson iterations used. These cases were found to occur when the analytical solution is out of the bracketing interval $[C_{\min}, C_{\max}]$. In the tests presented in Section 4, the analytical solution may go out of the bracketing interval in the very few cases in which V_T/V_Ω or $1 - V_T/V_\Omega$ is lower than about 10^{-6} and the dimensionless size of the bracketing interval ($(C_{\max} - C_{\min})/h$, where h is the cell size) is lower than about 0.05. Therefore, the global computational efficiency of the local volume enforcement procedure is not practically affected by the eventual use of an iterative root finding procedure (in the 3D shearing flow test of Section 4.1, when the finest and most deformed grid is used, for example, this occurs only in a 0.005% of the total cases).

4. Assessment of the proposed analytical method

The computational efficiency and robustness of the proposed analytical method for local volume enforcement in general grids will be assessed by comparing results with those obtained with the iterative method of Brent, which is extensively used in VOF-type methods. Brent's method [11] has been implemented in the present work following the indications given by Rider and Kothe [13], who used a bracketing procedure different from that described in Section 3.1. In this procedure, the volumes of the polyhedra truncated by all the planes parallel to $\mathscr{P}$ passing through every vertex of Ω_T are computed, and then the two planes that delimit the actual material volume V_T are determined. The routines were compiled using the GNU g77 compiler and all test

cases were run on a workstation with dual 2 GHz Intel T7200 processors. A number of 10^6 and 10^5 combinations of the volume V_T and interface orientation, for 2D and 3D problems, respectively, were randomly selected and solved for each test.

Table 1 shows the CPU time consumed by the iterative method relative to that consumed by the analytical method for five grid types and different values of the convergence tolerance in the solution for $C_{\Gamma c}$. The corresponding mean volume error and mean number of iterations needed by the iterative method to reach the required tolerance are also included in the table. Note that the CPU time consumed by the bracketing procedure suggested by Rider and Kothe [13] is in the order of $O(I_p)$, which makes the iterative method particularly time-consuming for high values of I_p. To assess the importance of using an appropriate bracketing procedure in an iterative method, we have also implemented Brent's method in combination with the bracketing procedure proposed in Section 3.1. Table 2 shows the CPU time consumed by the iterative method (relative to that consumed by the analytical method) using the bracketing procedure of Rider and Kothe [13] and that proposed in this work for three different grid types. The convergence tolerance in the solution was set to 10^{-10}. Note that, although the proposed bracketing procedure reduces by a factor of 2.5 the CPU time consumed by the iterative method for the case with $I_p = 20$, Table 2 shows that the iterative method is always more time-consuming than the analytical method, regardless of the bracketing procedure used.

4.1. Advection tests

In this section, we present results obtained using two types of hexahedral grids, obtained by deformation of a cubic grid, for two different tests: the rigid-body rotation of a slotted sphere, taken from the work of Enright et al. [3], and the single-vortex flow problem of Rider and Kothe [13] with a laminar pipe flow in the z direction. The first grid is a parallelepiped grid, obtained by transforming the location (x, y, z) of each node of a cubic grid to $(x + y/\xi, y, z)$, where ξ is a deformation parameter. In the second grid considered, referred to as sinusoidal grid, the coordinates of the grid nodes are given by $(x + \gamma/\xi, y + \gamma/\xi, z)$, where $\gamma = \sin(2\pi x)\sin(2\pi y)$. Note that the lower the value of ξ, the higher the deformation of the grid. Although similar tests have also been performed in two dimensions, only results for 3D cases will be presented here.

The analytical and geometrical tools described in this work have been combined with the unsplit advection (FMFPA-3D) method proposed by Hernández et al. [4] and with the PLIC reconstruction (CLC-CBIR) method proposed by López et al. [8], both of which can be used with deformed hexahedral grids without loss of generality. The error, E, is estimated with an L_1 error norm defined as

Table 1
Mean volume error and CPU time consumed by Brent's method (relative to that required by the proposed analytical method) for different grid types and convergence tolerances

Grid type	Tolerance/mean number of iterations	Mean volume error (%)	Relative mean CPU time
Square ($I_p = 4; 2D$)	10^{-4}/4.12	1.47×10^{-2}	1.46
	10^{-6}/4.48	7.13×10^{-5}	1.49
	10^{-10}/4.86	3.20×10^{-9}	1.58
Hexagonal ($I_p = 6; 2D$)	10^{-4}/4.49	1.14×10^{-2}	1.66
	10^{-6}/5.06	7.34×10^{-5}	1.74
	10^{-10}/5.87	4.32×10^{-9}	1.84
Cubic ($I_p = 8$)	10^{-4}/4.74	3.84×10^{-3}	1.90
	10^{-6}/5.35	2.71×10^{-5}	1.99
	10^{-10}/6.10	1.62×10^{-9}	2.11
Icosahedral ($I_p = 12$)	10^{-4}/4.51	2.91×10^{-3}	2.42
	10^{-6}/5.29	2.00×10^{-5}	2.53
	10^{-10}/6.21	1.48×10^{-9}	2.70
Dodecahedral ($I_p = 20$)	10^{-4}/4.22	2.96×10^{-3}	3.23
	10^{-6}/4.96	2.06×10^{-5}	3.32
	10^{-10}/5.91	1.29×10^{-9}	3.44

Table 2
CPU time (relative to that required by the proposed analytical method) using Brent's method with two different bracketing procedures, for square, cubic and dodecahedral grids, and a convergence tolerance of 10^{-10}

Grid type	Bracketing procedure	
	Rider and Kothe [13]	This work
Relative CPU time		
Square ($I_p = 4$)	1.58	1.38
Cubic ($I_p = 8$)	2.11	1.44
Dodecahedral ($I_p = 20$)	3.44	1.39

$$E = \sum_{i,j,k} V_{\Omega}^{(i,j,k)} |F^{(i,j,k)} - F_{\mathrm{e}}^{(i,j,k)}|, \tag{11}$$

where $V_{\Omega}^{(i,j,k)}$ is the volume of cell (i,j,k), and $F^{(i,j,k)}$ and $F_{\mathrm{e}}^{(i,j,k)}$ are, respectively, the calculated and exact volume fractions at the end of the test.

4.1.1. Zalesak's sphere test

This test involves the rotation, in a uniform vorticity field, of a slotted sphere of fluid of radius 0.15, initially centered at $(0.5, 0.75, 0.5)$, around an axis parallel to the z axis and centered in a unit domain. The slot is 0.05 wide and 0.125 deep. Results after one complete revolution of the slotted sphere obtained with a cubic grid, a parallelepiped grid with $\xi = 7.5$ and a sinusoidal grid with $\xi = 20$ are presented in Fig. 3 for the same grid size (100^3 cells) and time step used by Enright et al. [3]. The E values are included at the top of each picture. The minimum value of the ratios obtained between the minimum and maximum edge lengths for all the cells (r_{g}) is used to measure the deformation degree of the grid. The values of r_{g} corresponding to the deformed grids used are 0.99 for the parallelepiped grid and 0.56 for the sinusoidal grid. It can be observed that the proposed analytical and geometrical tools, in combination with the advection and reconstruction methods proposed in [4,8], respectively, perform well even for very deformed grids (the L_1 error norm for the greatest grid deformation only increases by about 5% with respect to that of the cubic grid). Note that there are no appreciable differences between the iso-surface results presented in Fig. 3 for different grid types. It has been found that using Brent's method for local volume conservation enforcement with a sufficiently low convergence tolerance yields E values that are almost identical to those obtained with the proposed analytical method, although, as in the tests depicted in Table 1, at the cost of a higher computational time. A visual comparison with the corresponding results obtained by Enright et al. [3] shows that the results presented in Fig. 3 compare favorably with their standard level set results. The net change in total volume at the end of the test obtained by the method proposed here is about $1 \times 10^{-7}\%$ for the cubic and parallelepiped grids and $1 \times 10^{-4}\%$ for the sinusoidal grid.

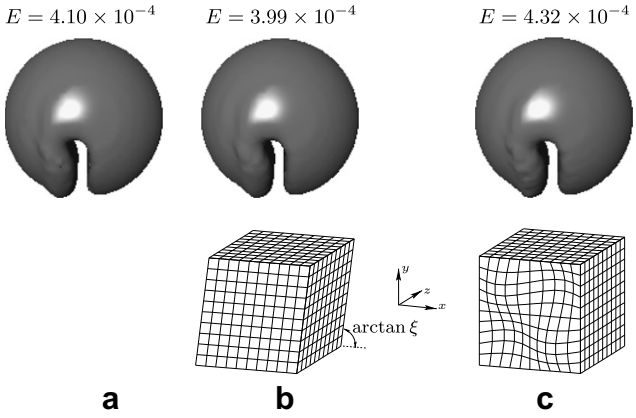


Fig. 3. Results for the $F = 0.5$ iso-surfaces after one complete revolution of the slotted sphere for three grid types: (a) cubic grid; (b) parallelepiped grid with $\xi = 7.5$; (c) sinusoidal grid with $\xi = 20$. The L_1 error norm is indicated at the top of each picture.

Table 3
L_1 error norm, E, and order of convergence, $\mathcal{O}$, obtained in the 3D shearing flow test at $t = T$, using different grid types (the grid deformation rates r_g are in parentheses) and three grid sizes

Grid size		Grid type			
		Cubic ($r_g = 1.0$)	Parallelepiped with $\xi = 7.5$ ($r_g = 0.99$)	Sinusoidal with $\xi = 30$ ($r_g = 0.67$)	Sinusoidal with $\xi = 20$ ($r_g = 0.56$)
$32 \times 32 \times 64$	E	1.06×10^{-2}	1.04×10^{-2}	1.11×10^{-2}	1.18×10^{-2}
	$\mathcal{O}$	1.59	1.55	1.57	1.59
$64 \times 64 \times 128$	E	3.53×10^{-3}	3.54×10^{-3}	3.74×10^{-3}	3.92×10^{-3}
	$\mathcal{O}$	1.82	1.82	1.81	1.81
$128 \times 128 \times 256$	E	1.00×10^{-3}	1.00×10^{-3}	1.07×10^{-3}	1.12×10^{-3}

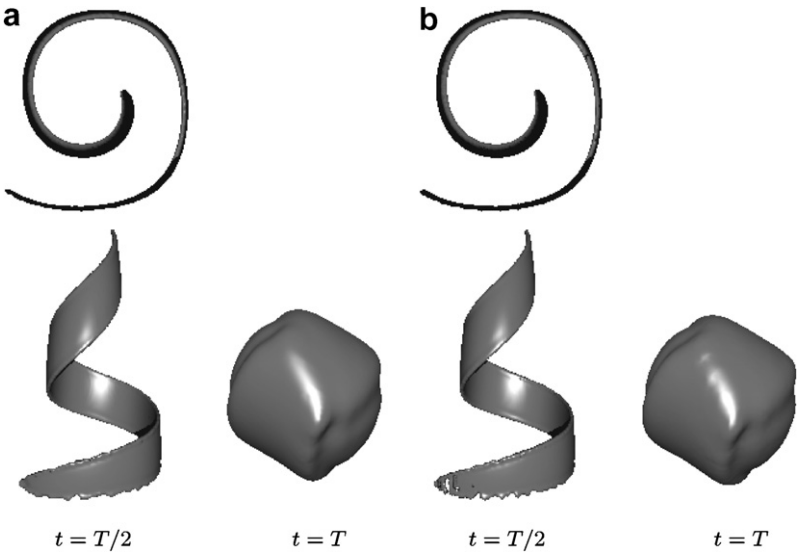


Fig. 4. Results for the $F = 0.5$ iso-surfaces at instants $t = T/2$ and T for the 3D shearing flow test, a grid size of $128 \times 128 \times 256$ and two grid types: (a) cubic grid; (b) sinusoidal grid with $\xi = 20$. The top and bottom pictures corresponding to $t = T/2$ show, respectively, xy-plane and xz-plane views.

4.1.2. 3D shearing flow test

In this test, taken from the work of Liovic et al. [5], a sphere of fluid of radius 0.15, initially centered at $(0.5, 0.75, 0.25)$ in a domain of size $1 \times 1 \times 2$, is deformed using a combination of the classical single vortex test of Rider and Kothe [13] with a laminar pipe flow in the z direction. A period of $T = 6$ (the fluid body should reverse to the initial location at $t = T$) and a CFL number based on the maximum velocity component in a cubic grid equal to 0.5 were used. Table 3 shows the L_1 error norm obtained at $t = T$ using different grid types and three grid sizes. Note that, for sufficiently fine grids, nearly second-order accuracy is reached for all the cases shown in the table. Again, the differences in accuracy between the results obtained with different grids are relatively small. It can be observed from the iso-surfaces of Fig. 4 that there are no appreciable differences between the results obtained with a cubic grid and those obtained with a highly deformed grid. The net change in total volume at the end of the test obtained with the grid with the highest deformation and a $128 \times 128 \times 256$ size is about $1 \times 10^{-5}\%$.

5. Conclusions

Simple and efficient geometrical tools for volume truncation operations and an analytical method for local volume enforcement in three-dimensional Cartesian geometry have been proposed for general grids. Different test cases have demonstrated the efficiency of the proposed analytical method, which compares favorably with

Brent's iterative method. The proposed tools may help reduce the implementation difficulties associated with the highly complex geometrical operations involved in 3D PLIC–VOF methods based on unsplit advection schemes. Different tests demonstrated that the proposed tools, in combination with a recently proposed unsplit PLIC–VOF method, perform well in terms of accuracy, even for very deformed grids.

Acknowledgments

The authors gratefully acknowledge the support of the Spanish Ministerio de Educación y Ciencia under Grants DPI2004-08198 and DPI2006-07047.

References

[1] M. Abramowitz, I.A. Stegun, Handbook of Mathematical Functions, National Bureau of Standards, Washington, DC, 1964.
[2] E. Aulisa, S. Manservisi, R. Scardovelli, S. Zaleski, Interface reconstruction with least-squares fit and split advection in three-dimensional Cartesian geometry, J. Comput. Phys. 225 (2007) 2301–2319.
[3] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level set method for improved interface capturing, J. Comput. Phys. 183 (2002) 83–116.
[4] J. Hernández, J. López, P. Gómez, C. Zanzi, F. Faura, A new volume of fluid method in three dimensions. Part I: Multidimensional advection method with face-matched flux polyhedra, Int. J. Numer. Methods Fluids, doi:10.1002/fld.1776.
[5] P. Liovic, M. Rudman, J.-L. Liow, D. Lakehal, D. Kothe, A 3D unsplit-advection volume tracking algorithm with planarity-preserving interface reconstruction, Comput. Fluids 35 (2006) 1011–1032.
[6] J. López, J. Hernández, P. Gómez, F. Faura, A volume of fluid method based on multidimensional advection and spline interface reconstruction, J. Comput. Phys. 195 (2004) 718–742.
[7] J. López, J. Hernández, P. Gómez, F. Faura, An improved PLIC–VOF method for tracking thin fluid structures in incompressible two-phase flows, J. Comput. Phys. 208 (2005) 51–74.
[8] J. López, C. Zanzi, P. Gómez, F. Faura, J. Hernández, A new volume of fluid method in three dimensions. Part II: Piecewise-planar interface reconstruction with cubic-Bézier fit, Int. J. Numer. Methods Fluids, doi:10.1002/fld.1775.
[9] D. Lörstad, L. Fuchs, High-order surface tension VOF-model for 3D bubble flows with high density ratio, J. Comput. Phys. 200 (2004) 153–176.
[10] G.H. Miller, P. Colella, A conservative three-dimensional Eulerian method for coupled solid–fluid shock capturing, J. Comput. Phys. 183 (2002) 26–82.
[11] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in Fortran, Cambridge University Press, Cambridge, UK, 1986.
[12] Y. Renardy, M. Renardy, PROST: a parabolic reconstruction of surface tension for the volume-of-fluid method, J. Comput. Phys. 183 (2002) 400–421.
[13] W.J. Rider, D.B. Kothe, Reconstructing volume tracking, J. Comput. Phys. 141 (1998) 112–152.
[14] R. Scardovelli, S. Zaleski, Direct numerical simulation of free-surface and interfacial flow, Annu. Rev. Fluid Mech. 31 (1999) 567–603.
[15] R. Scardovelli, S. Zaleski, Analytical relations connecting linear interfaces and volume fractions in rectangular grids, J. Comput. Phys. 164 (2000) 228–237.
[16] P.J. Schneider, D.H. Eberly, Geometric Tools for Computer Graphics, Morgan Kaufman Publishers, London, 2003.
[17] M.B. Stephenson, H.N. Christiansen, A polyhedron clipping and capping algorithm and a display system for three dimensional finite element models, ACM SIGGRAPH Computer Graphics 9 (1975) 1–16.
[18] X. Yang, A.J. James, Analytical relations for reconstructing piecewise linear interfaces in triangular and tetrahedral grids, J. Comput. Phys. 214 (2006) 41–54.
[19] The code and pseudo-code are available at the web site http://www.dimf.upct.es/personal/lrj/voftools.html.