An interface reconstruction method based on an analytical formula for 3D arbitrary convex cells

Steven Diot^a, Marianne M. François^a,

^aFluid Dynamics and Solid Mechanics (T-3), Los Alamos National Laboratory, NM 87545, USA.
(diot@lanl.gov)

Abstract. In this paper, we are interested in an interface reconstruction method for 3D arbitrary convex cells that could be used in multi-material flow simulations for instance. We assume that the interface is represented by a plane whose normal vector is known and we focus on the volume-matching step that consists in finding the plane constant so that it splits the cell according to a given volume fraction. We follow the same approach as in the recent authors' publication for 2D arbitrary convex cells in planar and axisymmetrical geometries, namely we derive an analytical formula for the volume of the specific prismatoids obtained when decomposing the cell using the planes that are parallel to the interface and passing through all the cell nodes. This formula is used to bracket the interface plane constant such that the volume-matching problem is rewritten in a single prismatoid in which the same formula is used to find the final solution. The proposed method is tested against an important number of reproducible configurations and shown to be at least five times faster.

Introduction

Interface reconstruction methods are intensively used in multi-material flow simulations codes when the interfaces between two or more materials must remain sharp like when two fluids are not miscible or when one material is a solid and the other a gas. In some numerical methods, the interface is explicitly tracked by marker points [12, 24, 22, 8] while in others, the interface is captured by the material volume fractions. The volume-tracking methods belong to the latter class. They were pioneered by [7, 16, 17] and substantially improved since then [5, 14, 3]. In these methods, only the volume fractions of the materials are known and the interfaces have to be reconstructed at each iteration in order to geometrically solve the evolution equations for each volume fraction.

In this work, as in most interface reconstruction methods [13, 11, 1], we assume a linear interface representation given by the equation

$$n_x x + n_y y + n_z z = d,$$

where $\vec{n} = (n_x, n_y, n_z)$ and d are the normal vector and plane constant of the interface, respectively. The reconstruction procedure consists of two steps: 1) the computation of the normal vector to the interface $\vec{n}$ and 2) the computation of the interface constant d so that it splits the cell according to a prescribed volume fraction. The focus of this work is on the latter step and we assume that $\vec{n}$ has already been obtained using one of the many existing methods [26, 4, 18].

In the sequel, we refer to this problem as the *volume-matching* step of the interface reconstruction and propose an original technique based on an analytical formula to solve it in arbitrary three-dimensional convex cells. Several other analytical methods exist for specific type of cells, *e.g.* for tetrahedra [25] or right parallelepipeds [21], but up to our knowledge, only one analytical approach is valid for convex polyhedra [15] and is based on the formula for the volume of a polyhedron that uses the nodes coordinates. We believe that our approach is conceptually simpler and will show that it is computationally more efficient.

Classically, reconstructing an interface in a polyhedral cell is performed by an iterative Brent’s-based method [20] intensively using plane-polyhedron intersections that may be slow and not always robust due to float comparisons in particular. Another approach is to tetrahedrize the polyhedral cell and to use the analytical formulae of [25] within an Brent’s-like iterative process. The main interest of this alternative method is to be valid for non-convex cells, however we do not consider them in this work. Therefore, all our numerical comparisons will be carried out against the Brent’s-based method of [20].

The method that we develop is based on the three following requirements. It has (1) to enforce volume-matching accuracy within machine precision, (2) to be significantly faster than the Brent’s-based method and (3) to derive the analytical formula from simple geometrical calculations. More specifically, these requirements are the extension of the ones we used to develop the volume-matching technique based on analytical formulae for arbitrary two-dimensional convex cells in planar and axisymmetrical geometries in [9]. They can be summed up as: first decompose the cell in elementary elements (here, prismatoids) by slicing it with the planes parallel to the interface and passing through all the nodes, and then derive an analytical formula for the volume of these specific elements and use it to bracket and find the plane constant.

Finally we point out that in the present paper, we deal with the reconstruction of an interface between two materials only. However in multi-material flow simulations more than two materials may be present in a given cell. In that case, the proposed technique would have to be used within a multi-material reconstruction scheme, *e.g.* the onion-skin [5] or nested-dissection [10] schemes, that consists in a series of interface reconstructions between two entities (*i.e.* one material and the remaining others) given a specific materials order. We want to emphasize that for the nested-dissection scheme, the reconstruction of more than two materials in a cell always involves general convex polyhedra since each reconstructed material volume is subtracted from the cell. Therefore, even on Cartesian meshes, the use of a volume-matching technique for general convex polyhedra as the one we propose is mandatory.

The paper is organized as follows: in the first section, we describe the cell decomposition and the plane constant bracketing step. The second section is dedicated to the description of the prismatoids obtained during the cell decomposition and more importantly to the derivation of the analytical formula for their volumes. The technical details of the implementation are described in the third section while numerical results are provided in the last section to demonstrate the gain in computational efficiency of the method against a classical approach.

1 Cell decomposition and plane constant bracketing

Notation. Let us first define the notation that we will use throughout the paper in an analogous way to the one used in [9]. We consider an arbitrary convex polyhedral cell Ω with N_e edges, N_f faces and N_n nodes. The nodes coordinates are denoted by $X_l = (x_l, y_l, z_l)$. By definition of a convex polyhedron, each face f has coplanar nodes, thus having a unique outward normal vector denoted by $\vec{n}_f = (n_{f,x}, n_{f,y}, n_{f,z})$. The volume of the cell is denoted by V_Ω and the material volume we want to match during the interface reconstruction is $\bar{V} = \alpha V_\Omega$ with $\alpha \in [0, 1]$ the material volume fraction.

As mentioned in the introduction, we assume that the unit normal vector to the interface $\vec{n} = (n_x, n_y, n_z)$ is known and points from the material under consideration to the exterior. In consequence, any plane parallel to the interface is defined by the equation $n_x x + n_y y + n_z z = d$ and will be called a d -plane. Moreover the material volume corresponding to a d -plane will be referred to as d -volume and denoted by V_d . Finally, the intersection between a d -plane and the cell is a convex polygon called d -polygon.

Rotation of the reference frame. In order to simplify the derivation of the formulae (and so the formulae themselves), we start by rotating the cell such that the interface normal $\vec{n}$ becomes $\vec{e}_z = (0, 0, 1)$. To this end, we use the 3×3 matrix corresponding to the rotation defined by

- the angle θ with $\cos \theta = \vec{n} \cdot \vec{e}_z = n_z$ and $\sin \theta = \|\vec{n} \times \vec{e}_z\| = \sqrt{n_x^2 + n_y^2}$,
- the axis for which a unitary direction vector is $\frac{\vec{n} \times \vec{e}_z}{\|\vec{n} \times \vec{e}_z\|} = \frac{(n_y, -n_x, 0)}{\sqrt{n_x^2 + n_y^2}}$.

This rotation matrix can be written only in terms of the normal vector components as

$$R(\vec{n}) = \begin{pmatrix} n_y^2 \Gamma + n_z & -n_x n_y \Gamma & -n_x \\ -n_x n_y \Gamma & n_x^2 \Gamma + n_z & -n_y \\ n_x & n_y & n_z \end{pmatrix} \quad \text{with} \quad \Gamma = \frac{1 - n_z}{n_x^2 + n_y^2}. \quad (1)$$

Two situations have to be treated separately: the trivial case $\vec{n} = \vec{e}_z$ for which $R(\vec{e}_z)$ is the identity matrix, and the case $\vec{n} = -\vec{e}_z$ for which a rotation of angle π around the x - or y -axis has to be used to preserve the direct orientation of the reference frame.

Once obtained, the rotation matrix is used to compute the nodes coordinates and the faces normal vectors in the rotated reference frame in which all computations will be done. For the sake of clarity, we consider that in the sequel all coordinates and vector components are given in the rotated frame of reference and keep the same notation.

Cell decomposition. In order to decompose the cell analogously to [9], we compute and sort in increasing order the plane constants d_l of the planes parallel to the interface and passing through all nodes X_l . We emphasize that $d_l = z_l$ in the rotated reference frame. The sorted plane constants are denoted d_k and satisfy $d_1 < d_2 < \dots < d_{\tilde{N}_n}$, where $\tilde{N}_n \leq N_n$ is the number of distinct

plane constants (which differs from N_n only when at least two nodes are included in the same d_k -plane). These $\tilde{N}_n$ d_k -planes define a cell partition made of (possibly degenerated) prismatoids, that we further denote P_k , formed by the portions of the cell lying between the d_k - and d_{k+1} -planes. We recall that a prismatoid is a polyhedron whose vertices lie in two parallel planes. We provide an illustration of the result of this procedure in Figure 1.

Figure 1: Illustration of the cell decomposition procedure for a unit cube. In this example, the number of distinct d_k -planes is $\tilde{N}_n = 6$ leading to 5 prismatoids, the first and last of which are degenerated. The darkest blue volume corresponds to V_{d_2} while the sum of all the blue volumes corresponds to V_{d_5} .

The volume-matching problem we seek to solve writes

$$\textbf{Find } d \textbf{ such that } d_1 \leq d \leq d_{\tilde{N}_n} \textbf{ and } V_d = \bar{V}. \quad (2)$$

It has a unique solution since the volumes V_{d_k} are strictly increasing with d_k . In order to simplify the problem (2), we want to bracket the interface plane constant, meaning that we seek the closest lower and upper bounding plane constants d_{lo} and d_{up} in $\{d_1, \dots, d_{\tilde{N}_n}\}$ such that $V_{d_{lo}} \leq \bar{V} \leq V_{d_{up}}$. To do so, we compute the d_k -volumes V_{d_k} until $\bar{V}$ is bounded, by iteratively adding the volume of the previous prismatoid P_{k-1} using the formula that we derive in the next section. In other words, $V_{d_k} = \sum_{l=1}^{k-1} V_{P_l}$ where V_{P_l} is the volume of the P_l prismatoid.

Once d_{lo} and d_{up} are found, the volume-matching problem (2) is equivalent to

$$\textbf{Find } d \textbf{ such that } d_{lo} \leq d \leq d_{up} \textbf{ and } V_d = \bar{V},$$

and can be rewritten using the volume $V_h = V_d - V_{d_{lo}}$ as

$$\textbf{Find } h \textbf{ such that } 0 \leq h \leq (d_{up} - d_{lo}), d = d_{lo} + h \textbf{ and } V_h = \bar{V} - V_{d_{lo}}. \quad (3)$$

Figure 2: Illustration of a typical prismatic volume defined by the d_{lo} -plane and the d_{up} -plane after plane constant bracketing. In this paper, an analytical formula for the blue volume is derived in function of h . Note that the $V_{d_{lo}}$ volume, depicted in gray, has already been computed during the plane constant bracketing step.

As thus rewritten, the problem takes place solely in the $P_{d_{lo}}$ prismatic volume as illustrated in Figure 2. The next section is therefore dedicated to the derivation of an analytical formula for the prismatic volume in function of h using only the d_{lo} -polygon information, the normal vectors to the faces of Ω and specific direction vectors of the edges of Ω present in $P_{d_{lo}}$. This analytical formula will then be used to both compute the prismatic volumes during the plane bracketing step, and to find the interface plane constant during the volume-matching step, analogously to the procedure developed in [9].

2 Analytical formula for the prismatic volumes

In this section we derive an analytical formula for the volume of a prismatic volume in function of its height h . In the first subsection we describe the prismatic configuration while the second subsection is dedicated to the analytical formula derivation.

2.1 Description of the prismatic volume

As mentioned previously, a result of the plane constant bracketing is that the volume-matching problem (2) is simplified into problem (3) to take place in the $P_{d_{lo}}$ prismatic volume only. Inside $P_{d_{lo}}$, an infinite number of prismatic volumes following the same configuration can be parametrized by h ($0 \leq h < d_{up} - d_{lo}$) such that they are defined by the d_{lo} -polygon and the d -polygon where

$d = d_{lo} + h$. In order to derive the analytical formula for the volume in function of h using neither the d - nor the d_{up} -polygon, the following three elements are needed:

- the d_{lo} -polygon made of $\mathcal{N}_n$ nodes and $\mathcal{N}_f$ faces (note that only the x - and y -coordinates of its nodes are relevant since the z -coordinates all equal d_{lo}),
- for each face f of the d_{lo} -polygon, the normal vector $\vec{n}_f$ of the face of Ω present in $P_{d_{lo}}$ in which f is included and
- for each node n of the d_{lo} -polygon, the set $\mathbb{E}_n = \{\vec{e}_{n(k)}, k = 1, \dots, \mathcal{E}_n\}$ of direction vectors (from n) of the edges of Ω present in $P_{d_{lo}}$ passing through n .

An illustration of this configuration for the blue prismatoid of Figure 2 is given in Figure 3. It is important to highlight that, in the following, the distinction between face index f and node index n is only present for the sake of clarity and refer to the same index since $\mathcal{N}_f = \mathcal{N}_n$. Therefore the f -th face of the d_{lo} -polygon is always assumed to be the one between nodes n and $n + 1$, *i.e.* the n -th face.

Figure 3: Illustration of the prismatoidal configuration obtained after plane constant bracketing. a) The $P_{d_{lo}}$ prismatoid truncated at a given h : the normal vector $\vec{n}_2$ of the second face of the d_{lo} -polygon is shown as an example; b) The d_{lo} -polygon: nodes indices are in black, faces indices are in dark blue and circled and the sets of ordered edges direction vectors are drawn in red.

Let us emphasize that the order in which the information is stored is very important to ensure that the algebraic volumes later use in the analytical formula have the right signs. In the following, we assume that the nodes of the d_{lo} -polygon are stored in the counter-clockwise order when seen from $h > 0$ to ensure a positive area. Furthermore for each node of the d_{lo} -polygon, the sets of edges is ordered in clockwise direction when seen from outside of the prismatoid. Details on the implementation of these constraints are given in section 3.

2.2 Analytical formula for the volume of a generic prismatoid

In this subsection, we derive the analytical formula to compute the volume of a generic prismatoid given the elements described in the previous subsection. As in [9], the main idea of this derivation is to subdivide the prismatoid in simpler geometrical elements and to compute its volume as a

sum of the algebraic volumes of its simpler elements. We emphasize that no special treatment is needed to get the signs of the algebraic volumes right thanks to the ordering we described previously. Therefore the final formula we derive hereafter is valid in every situations. The decomposition of $P_{d_{lo}}$, illustrated in Figures 4 and 5, is made as follows:

- a **right prism**, obtained by extrusion of the d_{lo} -polygon along the z -axis with height h ,
- a **tetrahedron** for each pair of consecutive edges in $\mathbb{E}_n$ of each node n of the d_{lo} -polygon, the height of which is h in regard to its basis included in the d -plane and defined by the projection on the d -plane of the direction vectors $(\vec{e}_{n(k)}, \vec{e}_{n(k+1)})$,
- a **wedge** for each face f of the d_{lo} -polygon, defined by the face of the right prism corresponding to f and the projection on the d -plane of the direction vectors $\vec{e}_{n(\mathcal{E}_n)}$ and $\vec{e}_{n+1(1)}$ of the last edge of the first node of f and the first edge of its second node, respectively.

We now derive the formulae for the volume of each of these elements using only the information provided in the previous subsection. Since most calculations take place in a two-dimensional plane parallel to the interface, we introduce the notation $\vec{u}$ for the projection of a three-dimensional vector $\vec{u} = (u_x, u_y, u_z)$ onto such a plane. In the rotated frame of reference in which we work, this projected vector $\vec{u}$ trivially corresponds to (u_x, u_y) since the z -direction corresponds to the plane normal direction. Therefore, no projection operator is actually used and the notation is only introduced for the sake of clarity. Moreover in the following, all direction vectors $\vec{e}_{n(k)} = (e_{n(k)}^x, e_{n(k)}^y, e_{n(k)}^z)$ are computed such that $\|\vec{e}_{n(k)}\| = 1$ to simplify the calculations.

Volume of the right prism. The volume of the right prism V_{prism} that serves as basis to the prismatoid volume calculation is simply given by:

$$V_{prism} = A_{d_{lo}} h, \quad (4)$$

where $A_{d_{lo}}$ is the positive area of the d_{lo} -polygon.

Volume of a tetrahedron for a node n . The volume of such a tetrahedron is simply obtained as the third of the product of h with the area of its face included in the d -plane. This face is defined by the two projected unitary vectors $\vec{e}_{n(k)}$ and $\vec{e}_{n(k+1)}$ associated with the positive lengths of the two sides $L_{e_{n(k)}}$ and $L_{e_{n(k+1)}}$, respectively. These lengths are given in function of h by

$$L_{e_{n(k)}} = h \frac{\|\vec{e}_{n(k)}\|}{e_{n(k)}^z} = h \frac{1}{e_{n(k)}^z} \quad \text{and} \quad L_{e_{n(k+1)}} = h \frac{1}{e_{n(k+1)}^z}. \quad (5)$$

Hence the algebraic volume of the tetrahedron is

$$V_{tetra}(\vec{e}_{n(k)}, \vec{e}_{n(k+1)}) = \frac{h}{3} \frac{(L_{e_{n(k)}} \vec{e}_{n(k)}) \times (L_{e_{n(k+1)}} \vec{e}_{n(k+1)})}{2} = \frac{h^3}{6} \left(\frac{\vec{e}_{n(k)} \times \vec{e}_{n(k+1)}}{e_{n(k)}^z e_{n(k+1)}^z} \right), \quad (6)$$

where the sign of the volume is given by the cross product $\vec{e}_{n(k)} \times \vec{e}_{n(k+1)}$ and implied by the ordering of the two edges direction vectors. An illustration of such a situation is given in Figure 4 (a). Finally let us point out that the formulae (5) are trivially obtained in the planes $(\vec{e}_z, \vec{e}_{n(k)})$ and $(\vec{e}_z, \vec{e}_{n(k+1)})$ and that if $\mathcal{E}_n = 1$, no tetrahedron exists for this node.

Volume of the wedge for a face f . The volume V_{wedge} of the wedge associated to a face f is decomposed in two sub-volumes; 1) a right triangular prism defined by the face of the right prism corresponding to f , the face of Ω attached to f and the unknown h and 2) a tetrahedron defined by the direction vectors $\vec{e}_{n(\varepsilon_n)}$ of the last edge of the first node of f and $\vec{e}_{n+1(1)}$ of the first edge of its second node. This decomposition is illustrated in Figure 4 (b).

The volume of the right triangular prism is computed as half the volume of the corresponding rectangular prism while the tetrahedron volume use the above formula (6), leading to:

$$V_{wedge}(\vec{n}_f, \vec{e}_{n(\varepsilon_n)}, \vec{e}_{n+1(1)}) = \frac{h}{2} B_f L_f(\vec{n}_f) + V_{tetra}(\vec{e}_{n(\varepsilon_n)}, \vec{e}_{n+1(1)}) \quad (7)$$

with B_f the length of the rectangular basis (included in the d -plane) that is equal to the polygon face f length, and L_f its width given by

$$L_f(\vec{n}_f) = h \frac{-n_{f,z}}{\|\vec{n}_f\|}. \quad (8)$$

Hence the wedge volume in function of h is given by

$$V_{wedge}(\vec{n}_f, \vec{e}_{n(\varepsilon_n)}, \vec{e}_{n+1(1)}) = \frac{h^2}{2} B_f \frac{-n_{f,z}}{\|\vec{n}_f\|} + \frac{h^3}{6} \left(\frac{\vec{e}_{n(\varepsilon_n)} \times \vec{e}_{n+1(1)}}{e_{n(\varepsilon_n)}^z e_{n+1(1)}^z} \right), \quad (9)$$

where the sign of the volume of the right triangular prism is given by $-n_{f,z}$ as expected. Indeed $n_{f,z}$ positive means that the face of Ω is *entering* the extruded prism and so must have a negative volume. Finally note that formula (8) is trivially obtained in the plane $(\vec{e}_z, \vec{n}_f)$.

Figure 4: Computation of the elementary volumes of the prismatoid decomposition. a) The tetrahedron volume equals a third of the product between the blue triangle area and h ; b) The wedge is decomposed in a triangular prism whose volume is half of the product between the green rectangle area and h , and two tetrahedra merged in one whose volume is given by a).

Volume of the prismatoid. We now have all the formulae for the volumes of each element of the prismatoid decomposition. The prismatoid volume is therefore given by

$$V(h) = V_{prism} + \sum_{f=1}^{\mathcal{N}_f} V_{wedge}(\vec{n}_f, \vec{e}_{n(\mathcal{E}_n)}, \vec{e}_{n+1(1)}) + \sum_{n=1}^{\mathcal{N}_n} \sum_{k=1}^{\mathcal{E}_n-1} V_{tetra}(\vec{e}_{n(k)}, \vec{e}_{n(k+1)}), \quad (10)$$

which corresponds to a cubic polynomial that can be written under the form

$$V(h) = Ch + B\frac{h^2}{2} + A\frac{h^3}{6} \quad (11)$$

where, according to equations (4),(6) and (9), we have

$$A = \sum_{n=1}^{\mathcal{N}_n} \left(\frac{\vec{e}_{n(\mathcal{E}_n)} \times \vec{e}_{n+1(1)}}{e_{n(\mathcal{E}_n)}^z e_{n+1(1)}^z} + \sum_{k=1}^{\mathcal{E}_n-1} \frac{\vec{e}_{n(k)} \times \vec{e}_{n(k+1)}}{e_{n(k)}^z e_{n(k+1)}^z} \right), \quad B = \sum_{f=1}^{\mathcal{N}_f} B_f \frac{-n_{f,z}}{\|\vec{n}_f\|} \quad \text{and} \quad C = A_{d_{lo}}. \quad (12)$$

In this formula $n+1$ becomes 1 for $n = \mathcal{N}_n$ and the sum over k is null when $\mathcal{E}_n = 1$. An illustration of this approach to computing the volume of the prismatoid of Figure 3 is provided in Figure 5.

Figure 5: Illustration of the prismatoid decomposition used to compute its volume. a) The right prism (in blue) of height h extruded from the d_{lo} -polygon; b) The wedges (in green) with positive volumes for faces with $n_{f,z} < 0$; c) The wedges (in red) with negative volumes for faces with $n_{f,z} > 0$ and the tetrahedron (in red) with negative volume defined by the two edges of node 1; d) The prismatoid volume is obtained by adding the algebraic volumes of a), b) and c).

3 Implementation details of the complete algorithm

This section is dedicated to the details of the different steps of the overall volume-matching algorithm. The main focus is on providing a data structure and related tools in order to be able to evaluate the volume using formula (11)-(12). Though the formula in itself is not complex, defining the proper data structure and updating it for the next prismatoid is not straightforward. Consequently, we describe in the first part the necessary data structure attached to a polyhedron and provide a way to ensure the right ordering for the sets of edges of each node. In the following two paragraphs, the initialization of the d_1 -polygon and the update from a d_k -polygon to the d_{k+1} -polygon are described. Finally the overall algorithm of the method is drawn in the last part.¹

Polyhedron representation. Several different ways to represent the convex polyhedral cell Ω exist and using the minimal one is not always the best solution, particularly in terms on computational costs. In our implementation, we have chosen to represent a polyhedron as

- a list of nodes: for each node, its x -, y -, and z -coordinates are stored,
- a list of edges: for each edge, the indices of its two nodes are stored,
- a list of faces: for each face, the indices of its edges are stored in counter-clockwise order when seen from the outside.

This representation is good enough to easily navigate through the different geometrical entities of the polyhedron, but we need two more lists to simplify the rest of the algorithm:

- the two faces of Ω attached to each edge,
- the list of edges of Ω passing through each node,

note that the second list is not ordered at this point and that in our implementation, these two extra lists are computed at the beginning of the algorithm.

As mentioned in subsection 2.1, the ordering of the list of edges passing through a node is very important. Therefore we provide in Algorithm 1 a general procedure, based on the convex nature of the polyhedron, to ensure the right ordering for each node n .

Algorithm 1 Ordering the set of edges passing through a node

- 1: Choose a first edge e in the list $\mathbb{E}_n$ of edges of the node n
 - 2: Compute its direction vector $\vec{e}$, from n to the next node of e
 - 3: Compute the triple product $TP = (\vec{n}_{f_1} \times \vec{n}_{f_2}) \cdot \vec{e}$, where f_1, f_2 are the two faces linked to e
 - 4: **if** $TP > 0$ **then**
 - 5: The ordering starts with $f_1 \rightarrow f_2$
 - 6: **else**
 - 7: The ordering starts with $f_2 \rightarrow f_1$
 - 8: **end if**
 - 9: Complete the rest of the list by alternately computing the next edge from the last face and the next face from the last edge until all edges of Ω passing through n are considered
-

¹A Fortran implementation of the algorithm is available upon request by email at steven.diot@gmail.com.

We emphasize that this ordering procedure is not performed at the initialization of the algorithm since 1) the right order is needed only for the nodes that are present in the prismatoid of the current step of the algorithm and 2) only the edges that have a direction vector with a strictly positive z -component are used to compute the volume of the current prismatoid, such that the actual lists of ordered edges for the current d_k -polygon nodes is a subset of the list given by the above procedure. Therefore in order to get the list of edges for the current d_k -polygon nodes in the right order, we first apply the Algorithm 1 and then only select the edges with a strictly positive z -component starting from the one for which the previous edge has a negative or null z -component. Note that there is always a first one except maybe at initialization as detailed in next paragraph.

Initialization of the d_1 -polygon. As explained in section 1, once the distinct plane constants are in increasing order, the bracketing step consists in incrementally computing the prismatoids volumes. But in order to compute the first prismatoid volume, corresponding to V_{d_2} , the d_1 -polygon has to be initialized. The initialization procedure is described in Algorithm 2.

Algorithm 2 Initialization of the d_1 -polygon.

- 1: Store the number, N , of nodes of Ω in the d_1 -plane
 - 2: **if** $N = 1$ **then**
 - 3: The d_1 -polygon is made of only one node and the list of edges passing through this node is directly given by the global ordering procedure of Algorithm 1
 - 4: **else if** $N = 2$ **then**
 - 5: The d_1 -polygon corresponds to the edge e of Ω made of the two nodes in the d_1 -plane. For both nodes, the ordering of the edges results from Algorithm 1 and its accompanying remark since the edge e must not be part of these lists,
 - 6: **else**
 - 7: The d_1 -polygon corresponds to the only face f of Ω included in the d_1 -plane. For each node of f the ordering of the edges results from Algorithm 1 and its accompanying remark since none of the edges of f must be part of these lists
 - 8: **end if**
-

The initialization of the algorithm being set, the next step is to go from a d_k -polygon to the next d_{k+1} -polygon without having to compute the intersection between the d_{k+1} -plane and Ω .

Update from the d_k - to the d_{k+1} -polygon. In order to get the d_{k+1} -polygon from the d_k -polygon, we make intensive use of all the elements that we defined previously and the only delicate point is to make sure that a node of the d_{k+1} -polygon is not repeated several times. This is ensured by using the list of nodes of Ω that correspond to d_{k+1} . The procedure to create the d_{k+1} -polygon is given by Algorithm 3.

Let us point out that if an edge of Ω is included in the d_{k+1} -plane, it has to be excluded from the lists of edges of the nodes of the d_{k+1} -polygon obtained using the above procedure.

Algorithm 3 Creating the d_{k+1} -polygon from the d_k -polygon

```
1: for  $n$  node of the  $d_k$ -polygon do
2:   for  $e_{n(k)} \in \mathbb{E}_n$  edge of node  $n$  do
3:     Get index  $n_2$  of the node of  $e_{n(k)}$  that has a  $z$ -coordinate greater than  $d_k$ 
4:     if  $n_2$  is one of the node corresponding to  $d_{k+1}$  then
5:       Add the  $x$ - and  $y$ -coordinates of node  $n_2$  to the  $d_{k+1}$ -polygon
6:       Get the list of edges of this node using Algorithm 1 and its accompanying remark
7:     else
8:       Add the new node coordinates,  $(x_n, y_n) + hL_{e_{n(k)}}\vec{e}_{n(k)}$ , to the  $d_{k+1}$ -polygon
9:       Store  $e_{n(k)}$  as the only edge passing through this new node
10:    end if
11:  end for
12: end for
```

The overall algorithm for the volume-matching technique. Using the tools we just described, the overall algorithm for the volume-matching method that we propose in this paper is given in Algorithm 4.

Algorithm 4 Overall algorithm for the proposed volume-matching technique.

```
1: Rotate the frame of reference using the rotation matrix (1)
2: Compute and sort the plane constants  $d_k$  for all nodes  $X_k$ 
3: Store the distinct values of  $d_k$  and their corresponding nodes
4: Initialize the  $d_1$ -polygon using Algorithm 2
5: for  $k = 1, \dots, \tilde{N}_n$  do
6:   Compute  $h$  as  $d_{k+1} - d_k$ 
7:   Compute the coefficients A, B, C according to (12)
8:   Compute the volume of the  $P_{d_k}$  prismatoid using (11) and deduce  $V_{d_{k+1}}$ 
9:   if  $V_{d_{k+1}} = \bar{V}$  then
10:     The interface plane constant  $d$  is  $d_{k+1}$ . Exit the loop
11:   else if  $V_{d_{k+1}} > \bar{V}$  then
12:     The interface plane constant  $d$  is between  $d_{lo} = d_k$  and  $d_{up} = d_{k+1}$ . Exit the loop
13:   else
14:     Create the  $d_{k+1}$ -polygon using Algorithm 3
15:   end if
16: end for
17: if  $d \neq d_{k+1}$  then
18:   Find  $d = d_{lo} + h$  by solving  $V(h) - \bar{V} = 0$  where  $V(h)$  is defined by (11)
19: end if
20: Deduce the material polyhedron and/or the remaining polyhedral cell
```

Note that we do not describe the final step of recovering the material polyhedron or the remaining part of the cell as it can be done in many ways. The most trivial, but not optimal, way is to simply compute the intersection of the d -plane with Ω using any existing technique, e.g. [23, 2]. However in our implementation, we make intensive use of our specific data structure and

of the history of the algorithm (*i.e.* we already know that the the part of Ω below the d_{lo} -plane belongs to the material polyhedron) to reduce the computational time. This leads to an algorithm that is about twice faster than without taking advantage of the history.

4 Numerical tests

This section is dedicated to a series of numerical tests carried out to demonstrate the computational interest of the proposed method over a classical approach for convex 3D cells. In the first section we describe in full details and in an entirely reproducible way the conditions of the different tests. Then we apply these conditions to test the accuracy and computational cost of the proposed volume-matching approach and compare the results to a classical iterative technique [20] using a Brent's method [6].

4.1 Testing conditions

In order to mimic the conditions that arise in real simulations — plane normal vectors covering all 3D directions, volume fractions ranging between 0 and 1 and arbitrary cell shapes — we propose to assess the performance of the proposed method over a very large number of *normal/volume fraction* combinations in six different polyhedra and to define these combinations in a deterministic manner for the sake of reproducibility. For each polyhedron, the computational times needed by the methods to complete the sets of combinations along with the average and maximal volume-matching errors are used to provide a fair comparison between the different techniques.

This subsection is divided in three parts, the first of which contains the details of the test cases configurations, the second describes the comparison measures while the last one provides details on the parameters used in the methods.

Test cases: cell shapes, volume fractions and normal vectors. As mentioned previously, we consider six different polyhedra to render realistic conditions of simulations and to assess the sensitivity of the proposed technique to the number of faces, edges and vertices. Illustrations of these cells are provided in Figure 6. These six polyhedra are defined as:

1. Trirectangular tetrahedron: 4 faces, 6 edges and 4 vertices whose coordinates are
 $(0, 0, 0), (1, 0, 0), (0, 1, 0)$ and $(0, 0, 1)$
2. Cube: 6 faces, 12 edges and 8 vertices whose coordinates are
 $(0, 0, 0), (1, 0, 0), (0, 1, 0), (0, 0, 1), (1, 1, 0), (0, 1, 1), (1, 0, 1)$ and $(1, 1, 1)$
3. Irregular convex polyhedron: 11 faces, 27 edges and 18 vertices obtained by intersecting the unit cube with the 5 (arbitrary chosen) following planes
 - Plane 1: $\vec{n} = (0.3, 0.7, 0.2)$ passing through $X = (0.65, 0.8, 0.6)$
 - Plane 2: $\vec{n} = (2, -1, 1)$ passing through $X = (0.8, 0.25, 0.6)$
 - Plane 3: $\vec{n} = (-0.85, -0.45, 0.25)$ passing through $X = (0.1, 0.4, 0.55)$

- Plane 4: $\vec{n} = (-0.9, 0.05, 0.5)$ passing through $X = (-0.01, 0.5, 0.5)$
 - Plane 5: $\vec{n} = (0.75, -0.35, -0.55)$ passing through $X = (0.9, 0.3, 0.2)$
4. Rhombic dodecahedron: 12 faces 24 edges and 14 vertices obtained by permutations of
 - 6 vertices from $(\pm 2, 0, 0)$
 - 8 vertices from $(\pm 1, \pm 1, \pm 1)$
 5. Regular dodecahedron: 12 faces, 30 edges and 20 vertices obtained by permutations of
 - 8 vertices from $(\pm 1, \pm 1, \pm 1)$
 - 12 vertices from $(0, \pm\phi^{-1}, \pm\phi), (\pm\phi^{-1}, \pm\phi, 0), (\pm\phi, 0, \pm\phi^{-1})$ with $\phi = (1 + \sqrt{5})/2$
 6. Regular isocahedron: 20 faces, 30 edges and 12 vertices obtained by permutations of
 - $(0, \pm 1, \pm\phi), (\pm 1, \pm\phi, 0), (\pm\phi, 0, \pm 1)$ with $\phi = (1 + \sqrt{5})/2$

Figure 6: Illustrations of the six convex polyhedra used in the numerical tests.

We now define the two sets S_1 and S_2 of configurations (*i.e.* couples formed by a normal vector $\vec{n}$ and a volume fraction α_m) that will be used as test cases. They are defined in an analogous

manner to [9] by combinations of a set $S_{\vec{n}}$ of normal vectors and two sets of volume fractions S_{α}^1 and S_{α}^2 respectively, that is

$$S_1 = \{(\vec{n}, \alpha_m) \mid \vec{n} \in S_{\vec{n}}, \alpha_m \in S_{\alpha}^1\} \quad \text{and} \quad S_2 = \{(\vec{n}, \alpha_m) \mid \vec{n} \in S_{\vec{n}}, \alpha_m \in S_{\alpha}^2\}.$$

The 259200 normal vectors of the set $S_{\vec{n}}$ are defined from two angles, ϕ and θ , partitioning the intervals $[0, \pi]$ and $[0, 2\pi]$ in 360 and 720 uniform subdivisions respectively, that is

$$S_{\vec{n}} = \left\{ \vec{n} = (\sin \phi \cos \theta, \sin \phi \sin \theta, \cos \phi) \mid \begin{array}{l} \phi = (n-1)\pi, \quad n = 1, \dots, 360 \\ \theta = (m-1)2\pi, \quad m = 1, \dots, 720 \end{array} \right\}.$$

Then we consider the same two sets of volume fractions S_{α}^1 and S_{α}^2 that we used in [9], once again to point out the small sensitivity of our approach to small volume fractions (in contrast with a Brent's-based method [20]). Both sets contain 1000 values so that the total number of combination in S_1 and S_2 , denoted N_{comb} , is 259.2 millions. Such a big N_{comb} is necessary to cover a relevant range of possibilities and to ensure fair comparison between methods.

The set S_{α}^1 uses a constant increment between consecutive volume fractions ranging from 10^{-3} to 1, that is

$$S_{\alpha}^1 = \{\alpha = m \cdot 10^{-3} \mid m = 1, 2, 3, \dots, 10^3\},$$

while the set S_{α}^2 focuses more on small volume fractions by considering 100 uniformly distributed volume fractions in each of the 10 intervals $[10^{-p}, 10^{-p+1}]$ with $p = 1, \dots, 10$, that is

$$S_{\alpha}^2 = \left\{ \alpha = 10^{-p} + \frac{m}{10^2} (10^{-p+1} - 10^{-p}) \mid m = 1, 2, \dots, 10^2 \text{ and } p = 1, 2, \dots, 10 \right\}.$$

Comparison measures. Accuracy of the volume-matching process is measured by a L^{∞} error norm

$$\max_{1 \leq i \leq N_{comb}} \frac{|V_d^i - V_m^i|}{V_{\Omega}},$$

and a L^1 error norm

$$\frac{1}{N_{comb}} \sum_{i=1}^{N_{comb}} \frac{|V_d^i - V_m^i|}{V_{\Omega}},$$

where the superscript i refers to a particular configuration $(\vec{n}, \alpha_m)$ in S_1 or S_2 so that the volume to match is $V_m^i = \alpha_m V_{\Omega}$ while V_d^i is the d -volume corresponding to the interface plane constant d computed by the method under consideration.

Moreover computational costs are given in Relative Time Units (R.T.U.) to allow quick and easy comparisons where we set one R.T.U. as the computational time that the method proposed in this paper takes to complete the set under consideration on the tetrahedron.

Methods parameters. The proposed and Brent's-like methods have been implemented in the same code such that the computational costs comparison is as fair as possible. It has been written in Fortran and compiled by GNU Fortran with the -O2 optimization flag. All computations have been run on one single core of the Intel Core i7 2.7Ghz processor of a MacBook Pro 9.1. We

hereafter give the implementation details that needs to be highlighted for both methods.

In the proposed technique, the interface plane constant is the root of $V(h) - \bar{V}$ for $0 < h < d_{up} - d_{lo}$ and is found by using a Newton's method with the initial guess $h = (d_{up} - d_{lo})/2$ and the stopping criterion $|V(h) - \bar{V}| < \epsilon$ with ϵ set to 10^{-15} .

The implementation of the Brent's-based iterative method follows the original paper [20]. Since it was designed for the two-dimensional case, we used a straightforward extension to the three-dimensional case. We recall that it consists of three steps: 1) for all cell nodes, compute the plane constant and the volume behind the interface that is passing through the node; 2) deduce the closest lower and upper bounds (within the plane constants computed at step 1) such that the solution lies in between; 3) use a Brent's iterative method to find the plane constant of the interface that corresponds to the volume to match. This algorithm makes extensive use of costly plane-polyhedron intersections and polyhedron volume computations from nodes coordinates.

Moreover we point out that the implementation of the Brent's method itself has been taken from [19] and uses a tolerance of 10^{-14} . This value ensures that the volume-matching errors are the smallest possible, meaning that reducing it won't change the accuracy of the results. We emphasize that increasing the tolerance value does not significantly lower the computational times but importantly deteriorate the final accuracy, as illustrated in [15].

Finally let us point out that if the volume fraction α is greater than 0.5, we apply the volume-matching technique with $1 - \alpha$ and the opposite normal vector since it is faster on average.

4.2 Numerical results

In this subsection we provide the numerical results of the volume-matching procedure for the sets of configurations described in previous subsection. The errors and computational times for the sets S_1 and S_2 are gathered in Tables 1 and 2, respectively. As expected, both methods provide volume-matching errors within machine tolerance independently of the type of cell and of the values of the volume fraction to match.

For the set of configurations S_1 , the proposed method is between 4.7 (for the cube) and 5.9 (for the icosahedron) times faster than the classical Brent's method, while these numbers increase to 7.2 (for the irregular polyhedron) and 9.0 for the set S_2 . This is a very significant speed-up that demonstrates the computational gain of our approach. Let us point out that in comparison, the analytical approach of [15] was only between 2 and 3.5 times faster than the Brent's approach for 3D cells. Let us note that the speed of our implementation of the Brent's method has been indirectly compared with the one of the Brent's method used in [15] and they are comparable.

Moreover, we see that independently of the set of configurations the number of Brent's iterations is about 1.5 larger than the one of the Newton's method used in our approach. This mainly explains the better speed-up over the set S_2 , for which these numbers of iterations are twice larger than for the set S_1 , since the cost of a Newton's iteration is negligible (one evaluation of a cubic polynomial and one of its derivative) compared to the cost of a Brent's one (one

plane-polyhedron intersection and one polyhedron volume computation).

Finally, in order to compare the R.T.U. values of Tables 1 and 2, let us add that the computational time taken by our method to complete the set S_1 on the tetrahedral cell is only 5% slower than for the set S_2 . Therefore comparisons of computational times for the two sets show that, except on simple cells (tetrahedron, cube), our approach is about 25% faster over the set S_2 than over the set S_1 . This is easily explained by its incremental nature that makes the bracketing step much quicker for very small volume fractions.

Table 1: Results of the volume-matching test cases of the set S_1 . Relative Time Units (R.T.U.) and volume-matching errors for the different cell shapes obtained by the Proposed and the Brent's method. The average number of (Brent's or Newton's) iterations over the whole set of combinations is also specified.

Method	Cell Shape	R. T. U.	L^∞ Error	L^1 Error	Avg. It. Nb.
Proposed	Tetrahedron	1.00	2.06E-15	1.19E-16	5.058
	Cube	1.29	9.99E-16	7.33E-17	3.179
	Irregular Polyhedron	2.08	3.46E-15	6.03E-16	3.746
	Rhombic Dodecahedron	1.88	6.11E-16	7.65E-17	4.575
	Regular Dodecahedron	2.19	5.52E-16	6.83E-17	4.032
	Regular Icosahedron	2.06	5.60E-16	7.16E-17	4.798
Brent's	Tetrahedron	5.71	1.30E-14	3.81E-16	7.201
	Cube	6.10	3.72E-15	1.31E-16	5.046
	Irregular Polyhedron	11.79	5.36E-15	2.69E-16	5.931
	Rhombic Dodecahedron	10.83	1.39E-15	1.07E-16	6.452
	Regular Dodecahedron	12.99	1.47E-15	1.05E-16	6.351
	Regular Icosahedron	12.31	1.37E-15	1.13E-16	6.636

Table 2: Results of the volume-matching test cases of the set S_2 . Relative Time Units (R.T.U.) and volume-matching errors for the different cell shapes obtained by the Proposed and the Brent's method. The average number of (Brent's or Newton's) iterations over the whole set of combinations is also specified.

Method	Cell Shape	R. T. U.	L^∞ Error	L^1 Error	Avg. It. Nb.
Proposed	Tetrahedron	1.00	2.01E-15	5.27E-17	10.425
	Cube	1.12	9.99E-16	2.03E-17	8.700
	Irregular Polyhedron	1.49	3.46E-15	1.08E-16	7.170
	Rhombic Dodecahedron	1.40	5.55E-16	1.42E-17	8.803
	Regular Dodecahedron	1.51	4.91E-16	1.39E-17	7.751
	Regular Icosahedron	1.50	5.09E-16	1.42E-17	8.902
Brent's	Tetrahedron	8.99	1.31E-14	1.60E-16	15.396
	Cube	9.19	3.72E-15	4.35E-17	13.165
	Irregular Polyhedron	10.78	1.46E-14	7.40E-17	10.941
	Rhombic Dodecahedron	11.89	1.33E-15	2.10E-17	13.225
	Regular Dodecahedron	11.45	1.47E-15	2.16E-17	11.920
	Regular Icosahedron	13.56	1.32E-15	2.34E-17	13.373

Summary and conclusion

In this paper, we have proposed a volume-matching technique based on an analytical formula for 3D arbitrary convex cells. This work can be seen as an extension of our recent work in 2D [9] since the main principles are identical, namely a decomposition of the cell into elementary volumes (prismatoids in 3D vs trapezoids in 2D) and the derivation of analytical formulae for the volume of these specific elementary volumes using a sub-decomposition in simpler geometrical elements.

The complexity inherent to the three-dimensional framework have led to theoretical and computational difficulties for which we presented solutions and possible implementations in section 3. For instance, the right ordering of the elements of the data structure of the polyhedral cell is critical to ensure that the algebraic volumes used in the final formula for the prismatoidal volumes have the right signs.

The proposed technique has been extensively validated against two sets of 259.2 millions reproducible volume-matching configurations in six different polyhedra and have always provided errors within machine precision. Furthermore, comparisons with a classical Brent's method have shown that our approach is between 4.7 and 9.0 times faster and less sensitive to the values of the volume fractions to match.

Finally, we remind that the method we have presented is valid for an interface between two materials only, but that it could be used in cells with more than two materials by simply following a multi-material reconstruction scheme [5, 10].

In the future, we plan to extend this approach, along with its 2D version [9], to non-convex cells such that this approach could be used in Arbitrary-Eulerian-Lagrangian methods in which non-convex cells naturally arise, as noted in [1].

Acknowledgment

This work was performed under the auspices of the National Nuclear Security Administration of the US Department of Energy at Los Alamos National Laboratory under Contract No. DE-AC52-06NA25396 and supported by the DOE Advanced Simulation and Computing (ASC) program. Approved for unlimited release under LA-UR-15-20420.

References

- [1] H.T. Ahn, M. Shashkov, Multi-material interface reconstruction on generalized polyhedral meshes, *J. Comput. Phys.* 226 (2007) 2096–2132.
- [2] H.T. Ahn, M. Shashkov, Multi-material interface reconstruction on generalized polyhedral meshes, Tech. Rep. LA-UR-07-0656, Los Alamos National Laboratory (2007). Also available as <http://cnls.lanl.gov/shashkov/>.

- [3] W. Aniszewski, T. Ménard, M. Marek, Volume of Fluid (VOF) type advection methods in two-phase flow: A comparative study, *Comput. Fluids* 97 (2014) 52–73.
- [4] T.J. Barth, P.O. Fredrickson, Higher-Order Solution of the Euler Equations on Unstructured Grids Using Quadratic Reconstruction, *AIAA Aerospace Sciences Meeting*, 28th Reno (1990).
- [5] D.J. Benson, Volume of fluid interface reconstruction methods for multi-material problems, *Appl. Mech. Rev.* 55 (2002) 151–165.
- [6] R.P. Brent, *Algorithms for Minimization Without Derivatives*, Prentice-Hall, Englewood Cliffs, New Jersey (1973) 195 pp. ISBN 0-13-022335-2.
- [7] R. DeBar, Fundamentals of the KRAKEN Code, LLNL Report, UCIR-760 (1974).
- [8] F.S. de Sousa, N. Mangiavacchi, L.G. Nonato, A. Castelo, M.F. Tomé, V.G. Ferreira, J.A. Cuminato, S. McKee, A front-tracking/front-capturing method for the simulation of 3D multi-fluid flows with free surfaces, *J. Comput. Phys.* 198 (2004) 469–499.
- [9] S. Diot, M. François, E. Dendy, An interface reconstruction method based on analytical formulae for 2D planar and axisymmetric arbitrary convex cells, *J. Comput. Phys.* 275 (2014) 53–64.
- [10] V Dyadechko, M Shashkov, Reconstruction of multi-material interfaces from moment data, *J. Comput. Phys.* 227 (2008) 5361–5384.
- [11] D. Gueyffier, J. Li, A. Nadim, R. Scardovelli, S. Zaleski, Volume-of-Fluid Interface Tracking with Smoothed Surface Stress Methods for Three-Dimensional Flows, *J. Comput. Phys.* 152 (1999) 423–456.
- [12] J. Glimm, J.W. Grove, X.L. Li, K. Shyue, Y. Zeng, Q. Zhang, Three-dimensional front tracking, *SIAM Journal on Scientific Computing* 19 (1998) 703–727.
- [13] D.B. Kothe, W.J. Rider, S.J. Mosso, J.S. Brock, Volume Tracking of Interfaces Having Surface Tension in Two and Three Dimensions, *AIAA Aerospace Sciences Meeting*, 34th Reno (1996) 96-0859.
- [14] 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.
- [15] J. López, J. Hernández, Analytical and geometrical tools for 3D volume of fluid methods in general grids, *J. Comput. Phys.* 227 (2008) 5939–5948.
- [16] B.D. Nichols, C.W. Hirt, *Methods for Calculating Multi-Dimensional, Transient Free Surface Flows Past Bodies*, LANL Report LA-UR-75-1932 (1975).
- [17] W.F. Noh, P. Woodward, SLIC (Simple Line Interface Calculation), *Lecture Notes in Physics*, Springer-Verlag, New York 59 (1976) 330–340.

- [18] J.E. Pilliod Jr., E.G. Puckett, Second-order accurate volume-of-fluid algorithms for tracking material interfaces, *J. Comput. Phys.* 199 (2004) 465–502.
- [19] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, *Numerical Recipes in Fortran; the Art of Scientific Computing* (2nd ed.). Cambridge University Press, New York, NY, USA (1993).
- [20] W.J. Rider, D.B. Kothe, Reconstructing Volume Tracking, *J. Comput. Phys.* 141 (1998) 112–152.
- [21] R. Scardovelli, S. Zaleski, Analytical Relations Connecting Linear Interfaces and Volume Fractions in Rectangular Grids, *J. Comput. Phys.* 164 (2000) 228–37.
- [22] S. Shin, D. Juric, Modeling Three-Dimensional Multiphase Flow Using a Level Contour Reconstruction Method for Front Tracking without Connectivity, *J. Comput. Phys.* 180 (2002) 427–470.
- [23] M.B. Stephenson, H.N. Christiansen, A polyhedron clipping and capping algorithm and a display system for three dimensional finite element models, *A CM SIGGRAPH Computer Graphics* 9 (1975) 1–16.
- [24] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, Y.-J. Jan, A Front-Tracking Method for the Computations of Multiphase Flow, *J. Comput. Phys.* 169 (2001) 708–759.
- [25] X. Yang, A.J. James, Analytic relations for reconstructing piecewise linear interfaces in triangular and tetrahedral grids, *J. Comput. Phys.* 214 (2006) 41–54.
- [26] D.L. Youngs, Time-dependent multi-material flow with large fluid distortion, *Numerical Methods for Fluids Dynamics*, Ed. K.W. Morton and M.J. Baines, Academic Press, New-York (1982) 273–285.