Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A new isosurface extraction method on arbitrary grids

Joaquín López [a,*], Adolfo Esteban [b], Julio Hernández [b], Pablo Gómez [b], Rosendo Zamora [a], Claudio Zanzi [b], Félix Faura [a]

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

ARTICLE INFO

Article history:
Available online 21 July 2021

Keywords:
Volume of fluid method
Interface reconstruction
Isosurface extraction
Non-convex polyhedra
Unstructured irregular grids
Signed distance

ABSTRACT

The development of interface-capturing methods (such as level-set, phase-field or volume of fluid (VOF) methods) for arbitrary 3D grids has further highlighted the need for more accurate and efficient interface reconstruction procedures. In this work, we propose a new method for the extraction of isosurfaces on arbitrary polyhedra that can be used with advantage for this purpose. The isosurface is extracted from volume fractions by a general polygon tracing procedure, which is valid for convex or non-convex geometries, even with non-planar faces. The proposed method, which can be considered as an extension of the marching cubes technique, produces consistent results even for ambiguous situations in polyhedra of arbitrary shape. To show the reproducibility of the results presented in this work, we provide the open source library `isoap`, which has been developed to implement the proposed method and includes test programs to demonstrate the successful extraction of isosurfaces on several grids with polyhedral cells of different types. We present results obtained not only for isosurface extraction from discrete volume fractions resulting from a volume of fluid method, but also from data sets obtained from implicit mathematical functions and signed distances to scanned surfaces. The improvement provided by the proposed method for the extraction of isosurfaces in arbitrary grids will also be very useful in other fields, such as CFD visualization or medical imaging.

1. Introduction

Isosurface extraction is a procedure often used in various applications, such as data visualization and numerous physical and engineering problems (see, for example, [1,2]). It is increasingly common to develop codes capable of simulating different physical phenomena using complex unstructured grids with cells without any predetermined geometric configuration. The complex geometry and topology of this type of grids makes extremely difficult the extraction of isosurfaces to visualize or reconstruct material volumes. This difficulty also appears in the simulation of multiphase flows using VOF-type methods and arbitrary convex or non-convex grids, when trying to reconstruct the interface using advanced schemes based on the extraction of isosurfaces from the distribution of the volume fraction of fluid. Examples of these schemes can be found, for example, in [3–6]. López et al. [3] developed several piecewise linear interface calculation (PLIC) methods to represent the interface in a given grid cell by a plane whose position is obtained to match the fluid volume contained in the cell and its orientation is obtained by a weighted-average procedure. This procedure is based on triangulated surfaces

[*] Corresponding author.
E-mail address: joaquin.lopez@upct.es (J. López).

https://doi.org/10.1016/j.jcp.2021.110579

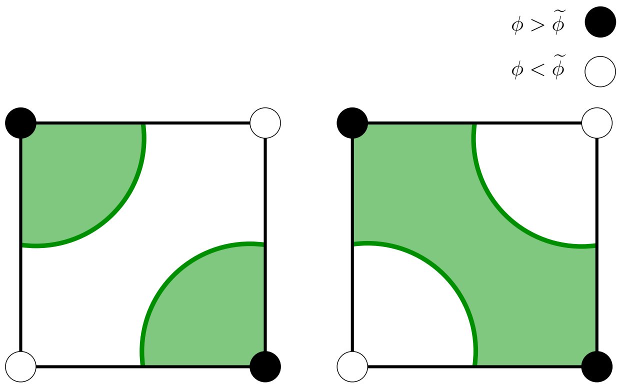


Fig. 1. 2D example of an ambiguous situation produced in the isosurface extraction from discrete data.

constructed from the extraction of generally non-planar isosurfaces corresponding to a 0.5 value of the fluid volume fraction distribution. In particular, the LLCIR (local level-contour interface reconstruction) method constructs the triangulated surface by joining the vertices and centroid of the extracted isosurface in the cell. The ELCIR (extended level-contour interface reconstruction) method constructs the triangulated surface by joining the centroid of the isosurface extracted in the cell with those of the isosurfaces extracted in adjacent cells. The CLCIR (conservative level-contour interface reconstruction) method translates the above mentioned centroids to the corresponding PLIC geometric centers, and the CLC-CBIR (conservative level-contour cubic-Bézier-based interface reconstruction) improves the orientation obtained from the CLCIR method by constructing a cubic-Bézier patch over each triangle of the triangulated surface. Results obtained using these reconstruction methods were presented in [3] for cubic grids and results obtained using the CLC-CBIR method were presented in [4] for grids with deformed hexahedral cells. Roenby et al. [5] proposed an advanced VOF method, referred to as isoAdvector, based on the extraction of isosurfaces corresponding to a non-fixed value of the fluid volume fraction that is adjusted in each cell to enforce volume conservation without the need for PLIC reconstructions. Later, Scheufler and Roenby [6] improved the isoAdvector method by using reconstructed distances to PLIC interfaces (isoAdvector-plicRDF). Results obtained using these isosurface-based methods were presented in [5,6] for general grids with arbitrary polyhedral cells. Shin and Juric [7] also used an isosurface extraction technique, referred to as level contour reconstruction, which enables front-tracking methods to robustly merge and breakup interfaces in three-dimensional flows (later enhancements and applications can be found in references [8–11]).

The following describes the problem posed in this work and discusses the background to the issue. Given a scalar field $\phi : \mathbb{R}^3 \to \mathbb{R}$ and a value $\widetilde{\phi} \in \mathbb{R}$, the goal is to extract the set $\{(x, y, z) : \phi(x, y, z) = \widetilde{\phi}\}$, which is usually called an isosurface. The scalar field can be given, for example, by implicit mathematical functions, thresholded data obtained by computer fluid dynamics simulations or by computer tomography, which is of particular relevance in medical applications, or unorganized points obtained, for example, by scanning systems (which usually requires creating a distance function from the unorganized points). In this work, with an approach that could also be used in applications other than the reconstruction of interfaces in VOF methods, it will be assumed that ϕ is defined by discrete data, generally in the form of experimental or computed volumetric datasets, available only at the cell vertices of a grid. Therefore, the location of the extracted isosurface points must be approximated by interpolation, and the reconstructed isosurface can be represented, for example, with linear edges constructed by sequentially connecting the obtained isosurface points ordered by some tracing procedure. It should be mentioned that the use of discrete data may introduce ambiguities like that shown in the example of Fig. 1. These situations can produce inconsistencies in isosurface extraction and can be solved by increasing the number of sample points.

There are several methods that use polygons to reconstruct isosurfaces from discrete data; among them, the “marching cubes” algorithm, originally introduced by Lorensen and Cline [12] for 3D medical data disposed in a cubic grid, is one of the best known. This algorithm creates a triangular surface of a given $\widetilde{\phi}$ level. In an individual cube there are 2^8 different configurations based on the relative values of the ϕ-sampled data assigned to its 8 vertices with respect to $\widetilde{\phi}$. The original algorithm proposed in [12] uses two different symmetries to reduce the number of distinctive configurations to only 14 (see Fig. 3 in [12]), which are stored in a lookup table to speed up execution of the algorithm. However, this reduction introduces a topological inconsistency in the vertex data when at least one face of the cube contains two opposite vertices, one on one side of the isosurface, and the other on the other side (similar to the ambiguous situation shown in Fig. 1). This inconsistency produces isosurfaces that may have holes and may not be a 2-manifold (see the example of Fig. 2). Several efforts have been made to avoid this inconsistency and to extend the use of the original marching cubes algorithm to other grid types (see, for example, References [13–20]).

Many codes, such as `OpenFOAM` [21] or `STAR-CCM+` [22], are capable of simulating different physical phenomena using complex unstructured grids. However, isosurface extraction to visualize or reconstruct material volumes has not been solved in a fully satisfactory way for such complex grids. The marching tetrahedra algorithm, originally introduced by Doi and Koide [15], is one of the most widely used isosurface extraction algorithms to deal with inconsistencies. It can also be

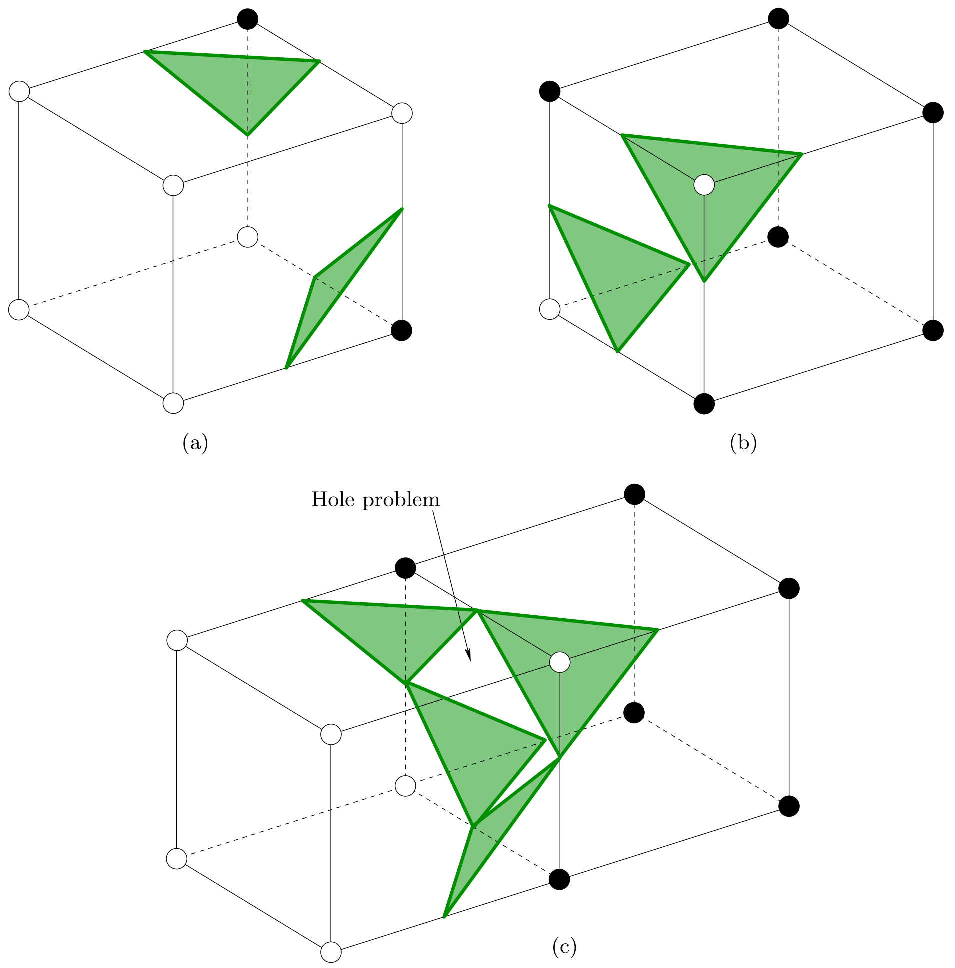


Fig. 2. Inconsistency of the original marching cubes algorithm. A combination of cells with (a) configuration 3 of Fig. 3 in [12] and (b) the complementary symmetric case produces (c) the hole observed on the bottom picture.

applied to unstructured grids, although with the added cost of an extra step to decompose complex polyhedral cells into tetrahedra. A promising neural network-based approach to efficiently reconstruct and visualize interfaces using square, cubic, triangular or tetrahedral grids has been recently proposed by Ataei et al. [23] in the context of VOF methods.

In this work, we propose a method, described in Section 2, that avoids the above mentioned inconsistencies by using general rules that do not rely on predefined configurations and can be applied to any polyhedral cell, either convex or non-convex, even with non-planar faces, without using cell decompositions. To the best of our knowledge, there is no similar published method that provides this generalization. In Section 3, different tests on several grids are considered to extract polygonal isosurfaces from implicit mathematical functions, signed distances to a scanned surface, and volume fraction data obtained from a VOF scheme. Results obtained with an extension to arbitrary meshes of the CLCIR method based on the proposed isosurface extraction method will be compared with those obtained by other authors. The method has been implemented in the `isoap` library, publicly available in the Mendeley Data repository [24], under the terms of GPLv3 license [25].

2. Method for isosurface extraction on arbitrary polyhedral cells

The new method proposed in this work aims to efficiently and consistently extract the isosurface corresponding to a value $\widetilde{\phi}$ from the distribution of values ϕ_{i_p} of the scalar variable ϕ assigned to every vertex i_p of an arbitrary polyhedral cell. As already mentioned, the ϕ_{i_p} values may be obtained, for example, from implicit functions or discrete data such as signed distances or volume fractions, among other sources. The isosurface is approximated locally at each cell by a polygonal surface, which will be referred to as an isopolygon, using a general polygon tracing procedure. The vertices of the isopolygon will be called $\widetilde{\phi}$-vertices. The following additional considerations will be made:

- the $\widetilde{\phi}$-vertices are located at cell edges connecting cell vertices with assigned ϕ values above and below $\widetilde{\phi}$,

Algorithm 1 Isosurface extraction.

```
1: Tag cell vertices
2: Call to Algorithm 2 to insert the $\widetilde{\phi}$-vertices
3: Call to Algorithm 3 to define the isopolygons by sequentially arranging the $\widetilde{\phi}$-vertex indices
4: Calculate the position vectors $\boldsymbol{x}_{\widetilde{\phi}}$ of the $\widetilde{\phi}$-vertices from interpolation
```

Table 1
Array IPV of index i_p assigned to every vertex i of face boundary j of the polyhedra of Fig. 3.

Vertex index	Face boundary j					
i	1	2	3	4	5	6
Cube of Fig. 3(a)						
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
Non-convex pentapyramid of Fig. 3(b)						
1	1	1	2	3	4	5
2	5	2	3	4	5	1
3	4	6	6	6	6	6
4	3	–	–	–	–	–
5	2	–	–	–	–	–

- at most, one $\widetilde{\phi}$-vertex is inserted at each cell edge, and
- the $\widetilde{\phi}$-vertices inserted at cell edges are sequentially joined by line segments, forming polygons that may be non-planar.

The extracted polygonal surface for the whole domain can be used in the implementation of advanced PLIC schemes in VOF methods (examples of such schemes can be found, for example, in [3]), but also for visualization with conventional graphics-rendering tools, among other purposes. The construction of the polygonal isosurface may be followed by a decomposition procedure to convert each isopolygon into triangles, but this is beyond the scope of the present work. In any case, most visualization applications can handle non-planar polygons, as can be seen in the results presented in Section 3.

Algorithm 1 performs the proposed isosurface extraction procedure, which basically consists of the following four steps:

(1) The cell vertices are tagged according to their relative assigned value of ϕ with respect to $\widetilde{\phi}$.
(2) A $\widetilde{\phi}$-vertex is inserted on every cell edge connecting cell vertices with different tags.
(3) The inserted $\widetilde{\phi}$-vertices are sequentially ordered by a polygon tracing procedure. An anticlockwise order is chosen to sequentially connect all the $\widetilde{\phi}$-vertices when the isosurface is viewed from the outside of the region with positive scalar field (reference medium whose surface is represented). This is the most complex step of the algorithm and is somehow similar to the capping procedure valid for convex or non-convex regions presented in [26].
(4) The $\widetilde{\phi}$-vertices are finally positioned on cell edges by interpolation.

Each of these steps is detailed in the next four subsections. The following arrangement of vertices, similar to that used in [26], is considered for a polyhedral cell. Let us consider a generic (convex or non-convex) polyhedron Ω with $\mathtt{NIPV}(j)$ vertices on each face boundary j, either planar or non-planar. Note that a face may be defined by a simple polygon with holes, with its non-simply connected interior delimited by two or more face boundaries. The vertices of each face boundary are arranged sequentially, so that the vector joining two consecutive vertices leaves the face boundary to the left when viewed from outside the polyhedron. The vertex with index i_p, assigned to vertex i of face boundary j, is stored using the two-dimensional array $\mathtt{IPV}(j,i) = i_p$. The notation used in the description of the algorithms considers that the vertices for each face boundary are ordered to form a closed loop. More details about the arrangement of vertices of Ω can be found in [26]. As an example, Table 1 presents the arrangement of vertices in the cube and non-convex pentapyramid of Fig. 3, showing the index i_p assigned to every vertex i of face boundary j ($\boldsymbol{x}_{j,i} \equiv \boldsymbol{x}_{i_p}$). For ease of explanation, many examples of surface extraction on these polyhedra will be presented below in this section. Illustrative examples of the application of the algorithm to more complex polyhedral cells will be presented throughout the paper.

2.1. Cell vertex tagging

At every cell vertex with index i_p, the array element $\mathtt{IA}(i_p)$ is set equal to 1 if $\phi_{i_p} > \widetilde{\phi}$, or 0 otherwise. From now on, cell vertices with assigned IA element values equal to 1 and 0 will be depicted with • and ∘ symbols, respectively, as in Fig. 4.

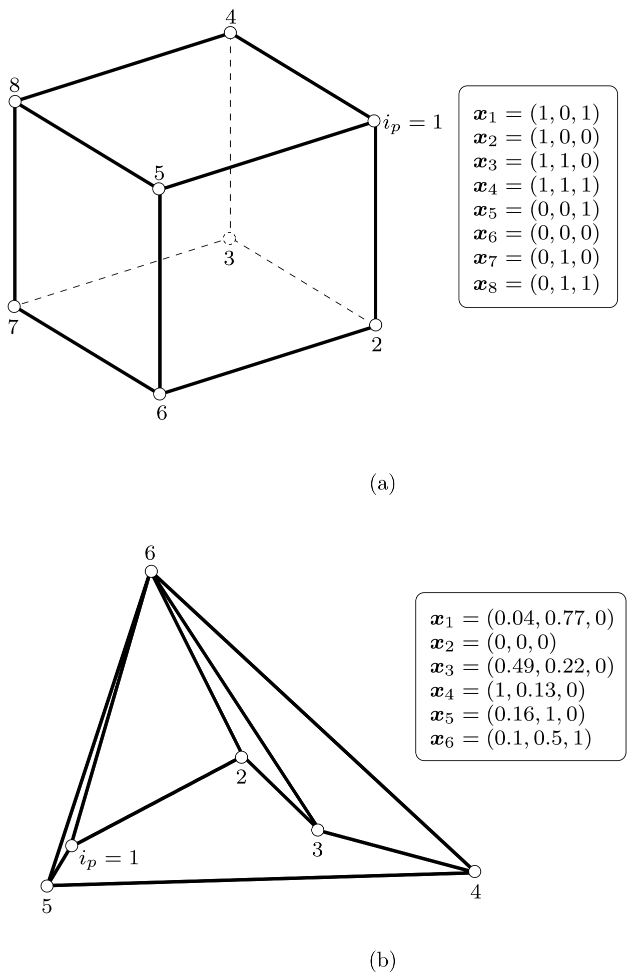

4
8
$i_p = 1$
5
3
7
2
6
$\boldsymbol{x}_1 = (1, 0, 1)$
$\boldsymbol{x}_2 = (1, 0, 0)$
$\boldsymbol{x}_3 = (1, 1, 0)$
$\boldsymbol{x}_4 = (1, 1, 1)$
$\boldsymbol{x}_5 = (0, 0, 1)$
$\boldsymbol{x}_6 = (0, 0, 0)$
$\boldsymbol{x}_7 = (0, 1, 0)$
$\boldsymbol{x}_8 = (0, 1, 1)$
(a)
6
$\boldsymbol{x}_1 = (0.04, 0.77, 0)$
$\boldsymbol{x}_2 = (0, 0, 0)$
$\boldsymbol{x}_3 = (0.49, 0.22, 0)$
$\boldsymbol{x}_4 = (1, 0.13, 0)$
$\boldsymbol{x}_5 = (0.16, 1, 0)$
$\boldsymbol{x}_6 = (0.1, 0.5, 1)$
2
$i_p = 1$
3
5
4
(b)

Fig. 3. Vertex coordinates of (a) a cube and (b) a non-convex pentagonal pyramid.

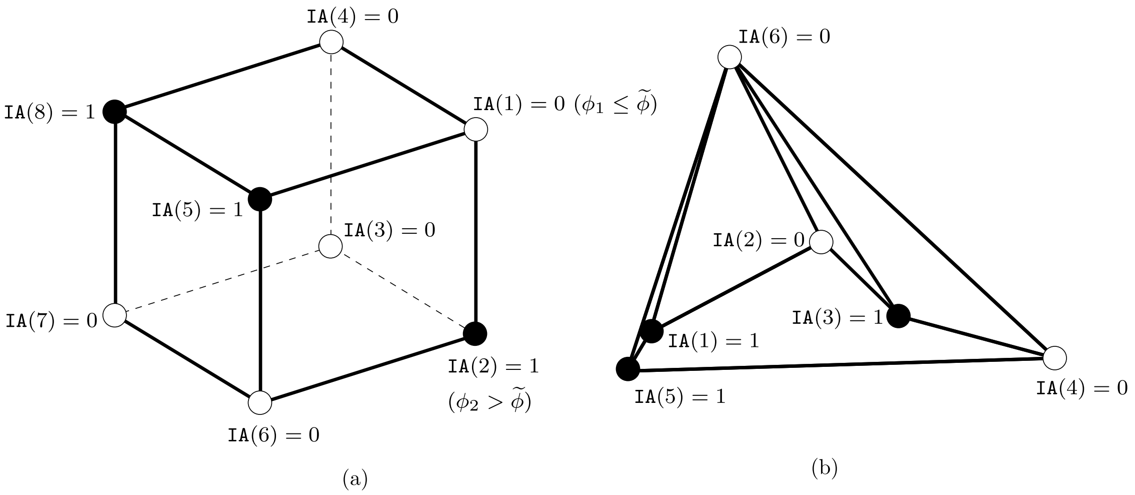


Fig. 4. Examples of vertex tagging for the cells shown in Fig. 3.

Algorithm 2 Insertion of the $\widetilde{\phi}$-vertex indices.

```
1: $i_{\widetilde{\phi}} = 0$
2: for face boundaries $j$ intersected by the isosurface do
3:     $i' = 0$
4:     for $i = 1$ to NIPV($j$) do
5:         $i_{p1}$ = IPV($j, i$) and $i_{p2}$ = IPV($j, i+1$)
6:         if IA($i_{p1}$) $\neq$ IA($i_{p2}$) then
7:             if edge defined by $i_{p1}$ and $i_{p2}$ was previously visited then
8:                 Identify the previously inserted vertex with index $i_{\widetilde{\phi}}$
9:             else
10:                Insert a new $\widetilde{\phi}$-vertex, with index $i_{\widetilde{\phi}} = i_{\widetilde{\phi}} + 1$
11:            end if
12:            $i' = i' + 1$ and IPVINT($j, i'$) $= i_{\widetilde{\phi}}$
13:            if IA($i_{p2}$) $= 1$ then IPVINT($j, i'$) is marked as "key vertex"
14:        end if
15:    end for
16:    NIPVINT($j$) $= i'$
17: end for
```

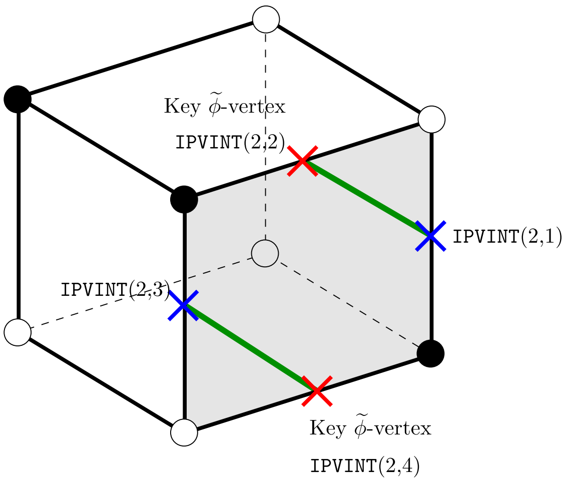


Fig. 5. Key (red cross) and previous (blue cross) $\widetilde{\phi}$-vertices for $j = 2$ (boundary of the shaded face) in the example of Fig. 6(a). The isoedges are depicted in green. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

2.2. $\widetilde{\phi}$-Vertex insertion

Algorithm 2 inserts a $\widetilde{\phi}$-vertex at each cell edge intersected by the isosurface. To illustrate the procedures used here and in the next section, the inserted $\widetilde{\phi}$-vertices are depicted with $\times$ symbols at the middle of the corresponding intersected cell edges (the final location of the $\widetilde{\phi}$-vertices will be obtained from the interpolation procedure described in Section 2.4). Each of these intersected edges is defined by the two vertices of the edge, with different IA element values. A cell is considered to be intersected by the isosurface if any of its edges is. The procedure goes in sequential order over the NIPV(j) vertices of each intersected face boundary j (lines 4-15 in Algorithm 2) to obtain the array IPVINT(j, i') that stores the index $i_{\widetilde{\phi}}$ assigned to every intersected edge i' of j (line 12 in Algorithm 2). The array NIPVINT(j) stores the total number of edges of the face boundary j intersected by the isosurface (line 16 in Algorithm 2). Note that each intersected edge is shared by two face boundaries, and index $i_{\widetilde{\phi}}$ is increased by one only if the corresponding edge was not previously visited in this sequential procedure over all the intersected face boundaries (line 10 in Algorithm 2). Otherwise, the index $i_{\widetilde{\phi}}$ identified in a previous "visit" to the edge (line 8 in Algorithm 2) is assigned to the array IPVINT.

On every intersected face boundary, the inserted $\widetilde{\phi}$-vertices will be marked as follows (see the example in Fig. 5):

- A $\widetilde{\phi}$-vertex inserted at an intersected edge i', defined by vertices with indices i_{p1} = IPV(j, i) and i_{p2} = IPV($j, i+1$), of face boundary j will be marked as a "key $\widetilde{\phi}$-vertex" if IA(i_{p2}) $= 1$ (line 13 in Algorithm 2; vertices denoted by red cross symbols in the example of Fig. 5).
- If IPVINT(j, i') stores an index corresponding to a key $\widetilde{\phi}$-vertex in the face boundary j, the edge joining it to the previously inserted $\widetilde{\phi}$-vertex, whose index is stored as IPVINT($j, i'-1$) (blue cross symbols in the example of Fig. 5), is part of an isopolygon (each of the green lines in the example of Fig. 5). Note that the previous $\widetilde{\phi}$-vertex of index IPVINT($j, i'-1$) must be a key $\widetilde{\phi}$-vertex at the other face boundary of the cell that shares the cell edge in which this $\widetilde{\phi}$-vertex is inserted. The array IPVINT is considered to form a closed loop; thus, for $i' = 1$, IPVINT($j, i'-1$) = IPVINT(j, NIPVINT(j)).

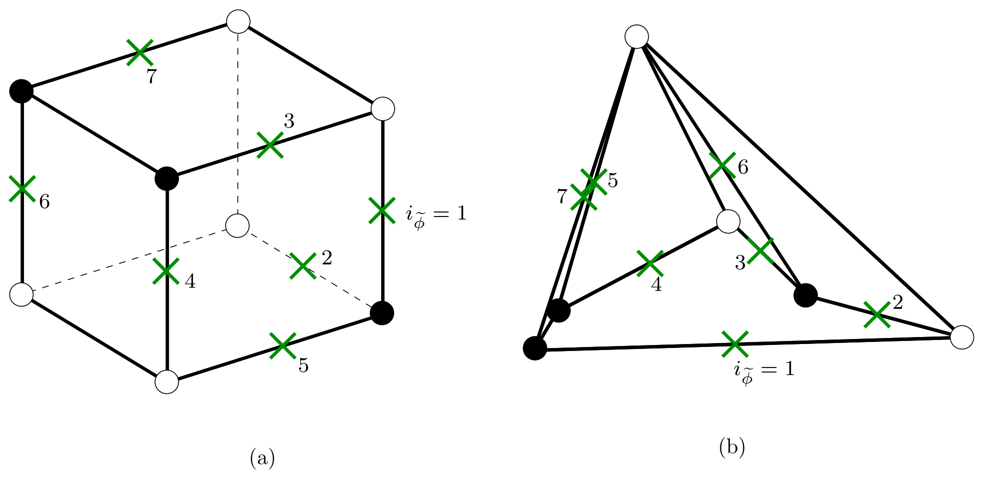


Fig. 6. Examples of the application of the insertion procedure of Algorithm 2 for the two polyhedra of Fig. 3. Symbols • and ◦ denote cell vertices with assigned `IA` element values equal to 1 and 0, respectively, and symbol × denotes $\widetilde{\phi}$-vertices (the same notation will be used in other figures).

Table 2
Array `IPVINT` of indices $i_{\widetilde{\phi}}$ of the vertices inserted on the intersected face boundaries of the polyhedra of Fig. 6. The key $\widetilde{\phi}$-vertices inserted on every face boundary are boundboxed.

Intersected edge	Face boundary j					
index i'	1	2	3	4	5	6
Cube						
1	$\boxed{1}$	1	$\boxed{2}$	$\boxed{6}$	$\boxed{7}$	$\boxed{4}$
2	2	$\boxed{3}$	5	7	3	6
3	–	4	–	–	–	–
4	–	$\boxed{5}$	–	–	–	–
Non-convex pentapyramid						
1	1	4	$\boxed{3}$	2	$\boxed{1}$	5
2	$\boxed{2}$	$\boxed{5}$	6	$\boxed{6}$	7	$\boxed{7}$
3	3	–	–	–	–	–
4	$\boxed{4}$	–	–	–	–	–

Fig. 6 illustrates the application of Algorithm 2 to the polyhedra of Fig. 3. Table 2 shows the resulting array `IPVINT` of indices $i_{\widetilde{\phi}}$, of the vertices inserted at the intersected face boundaries of the polyhedra (the key $\widetilde{\phi}$-vertices inserted on each face boundary are highlighted in the table by a box).

2.3. $\widetilde{\phi}$-Vertex arrangement

Algorithm 3 sequentially arranges all the indices of the inserted $\widetilde{\phi}$-vertices in anticlockwise order when viewed from outside the region with positive scalar field (fluid or another medium considered as reference), to form one or more closed isopolygons. The isopolygons constructed to approximate the isosurface are identified with index k. The array `NIPVISO`(k) is used to store the number of $\widetilde{\phi}$-vertices on each isopolygon k, and the two-dimensional array `IPVISO`(k, i_k) is used to store every $\widetilde{\phi}$-vertex index $i_{\widetilde{\phi}}$ assigned to vertex i_k of isopolygon k.

The proposed vertex arrangement procedure consists of the following steps:

1. Assign the $\widetilde{\phi}$-vertex index corresponding to the first vertex of isopolygon k (line 2 in Algorithm 3).
   The first vertex ($i_k = 1$) of the first isopolygon ($k = 1$) is assumed to be the first vertex ($i_{\widetilde{\phi}} = 1$) obtained in the insertion procedure of Section 2.2 (line 1 in Algorithm 3). Note, however, that a different choice could have been made with no loss of generality.
2. Apply the following recursive procedure (lines 3-6 in Algorithm 3):

   2.i. For the $\widetilde{\phi}$-vertex index previously assigned to `IPVISO`(k, i_k), find the intersected face boundary j for which `IPVINT`(j, i') = `IPVISO`(k, i_k) is a key $\widetilde{\phi}$-vertex (line 3 in Algorithm 3).
   2.ii. If the previous $\widetilde{\phi}$-vertex index stored at `IPVINT`$(j, i' - 1)$ is not coincident with `IPVISO`$(k, 1)$, increase index i_k by one, assign `IPVINT`$(j, i' - 1)$ to `IPVISO`(k, i_k) and go to 2.i (lines 4-6 in Algorithm 3).

Algorithm 3 Arrangement of the $\widetilde{\phi}$-vertices.

```
1: $k = 1$ and $i_{\widetilde{\phi}} = 1$
2: $i_k = 1$ and IPVISO$(k, i_k) = i_{\widetilde{\phi}}$
3: Find $j$ and $i'$ for which IPVINT$(j, i')$ = IPVISO$(k, i_k)$ is a "key $\widetilde{\phi}$-vertex"
4: if IPVINT$(j, i' - 1) \neq$ IPVISO$(k, 1)$ then
5:     $i_k = i_k + 1$ and IPVISO$(k, i_k)$ = IPVINT$(j, i' - 1)$
6:     Go to line 3
7: else
8:     NIPVISO$(k) = i_k$
9:     if there are still unassigned $\widetilde{\phi}$-vertices then
10:        Pick an unassigned $\widetilde{\phi}$-vertex $i_{\widetilde{\phi}}$ and $k = k + 1$
11:        Go to line 2 to construct a new isosurface $k$
12:    end if
13: end if
```

3. If there are still unassigned $\widetilde{\phi}$-vertices, pick one of them, increase index k by one and go to Step 1 to construct a new isopolygon (lines 9-12 in Algorithm 3).

Figs. 7 and 8 illustrate the sequence of application of Algorithm 3 to the examples of Figs. 4(a) and 4(b), respectively. The edge of the isopolygons are depicted with thick green lines.

Examples of the construction of multiple isopolygons in polyhedral cells can be seen in Figs. 9(a) (four isopolygons) and 9(b) (eight isopolygons).

2.4. Positioning the $\widetilde{\phi}$-vertices by interpolation

Finally, the position vectors $\boldsymbol{x}_{\widetilde{\phi}}$ of the $\widetilde{\phi}$-vertices are obtained from the following linear interpolation (line 4 in Algorithm 1):

$$\boldsymbol{x}_{\widetilde{\phi}} = \boldsymbol{x}_{i_{p_2}} - \frac{\phi_{i_{p_2}} - \widetilde{\phi}}{\phi_{i_{p_2}} - \phi_{i_{p_1}}} \left(\boldsymbol{x}_{i_{p_2}} - \boldsymbol{x}_{i_{p_1}} \right). \tag{1}$$

Note that, due to the use of strict inequalities (line 6 in Algorithm 2) to determine the ϕ values involved in Eq. (1), it is guaranteed that $\phi_{i_{p_2}} \neq \phi_{i_{p_1}}$, and, therefore, the divide by zero issue never occurs and the consistency of the isosurface extraction is maintained in cases where the isosurface coincides with a face, edge, or vertex of the cell. It should be mentioned that high-order interpolations based, for example, on the gradient of ϕ could have been considered (see, for example, Reference [27]), but this will be the subject of future works.

Fig. 10 shows the resulting isopolygons for the cells of Fig. 3 and the ϕ-node values indicated, which are obtained by using the interpolation of Eq. (1) with $\widetilde{\phi} = 0$ to compute the position vectors $\boldsymbol{x}_{\widetilde{\phi}}$.

2.5. Software description

The implemented algorithm extracts, from discrete data, isosurfaces on arbitrary, convex or non-convex polyhedra, even with non-planar face boundaries. The software package [24] also includes a user manual and routines for writing the geometry of polyhedra and isosurfaces in external files. Tests programs are included to extract isosurfaces on single polyhedral cells and grids of any type, either from implicit mathematical functions or discrete data such as volume fractions or signed distances to scanned surfaces. The implemented routines can be used in `FORTRAN` and `C`, with `C` interfaces. To speed up the operations performed over grids, the `OpenMP` application programming interface is used.

3. Results and discussion

A comparison in terms of consistency and computational efficiency between the proposed and the marching cubes algorithms is presented in Section 3.1. Results for isosurfaces constructed from implicit mathematical functions, signed distances to scanned surfaces and volume fraction data obtained from a volume of fluid scheme, using several grids with convex and non-convex cells, are presented in Section 3.2. The results presented in these sections are visualized using the `ParaView` program [28].

3.1. Comparison between the proposed algorithm and the marching cubes algorithm

Fig. 11 shows the isosurfaces extracted using the proposed method for the pre-defined cubic cell configurations considered in [12]. Note that the results for ambiguous configurations, like those shown in Fig. 1, differ from those obtained in [12]. The inconsistency observed in Fig. 2 when using the original marching cubes algorithm [12] is produced because the $\widetilde{\phi}$-vertices lying on the isoedges on an ambiguous cell face are connected using rules that may be different in the two cells

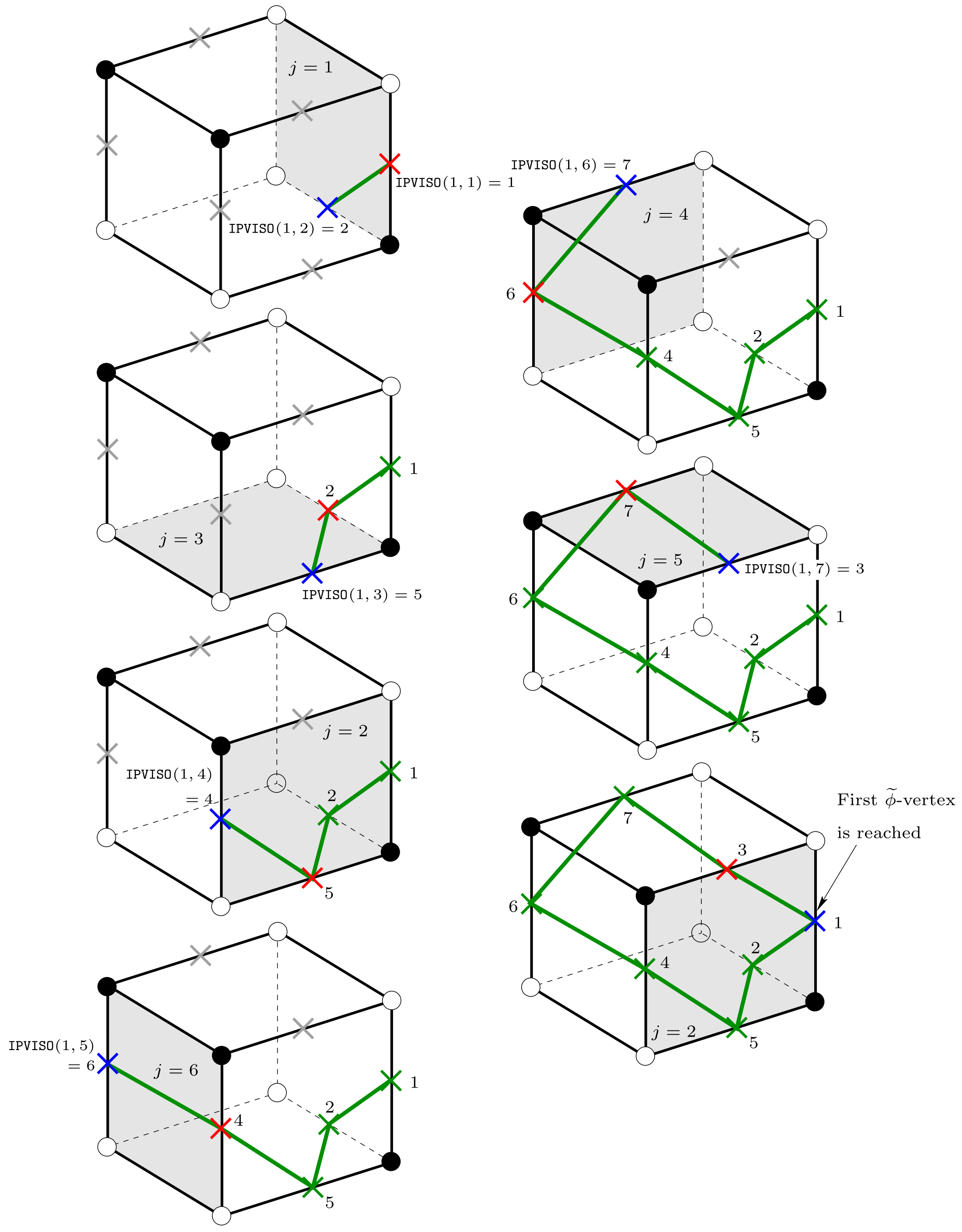


Fig. 7. Construction sequence (from top to bottom and from left to right) of the isopolygon for the example of Fig. 4(a). The cell face boundary involved in each step of the sequence is depicted gray filled.

that share the face. Note that the extracted isosurface separates the black circles on the left cell in Fig. 2, but the empty circles on the right cell, producing a hole at the shared face. The proposed method avoids this inconsistency because all the face boundaries are treated systematically, using the same rules presented in Algorithms 2 and 3. The extracted isopolygons obtained using the proposed method always separate the cell vertices with IA values equal to zero, as shown in the example of Fig. 12. Bloomenthal [29], Wyvill and Jevans [30] or Delibasis et al. [31], among others, also used a polygon tracing technique to avoid these inconsistencies, although only for cubic cells.

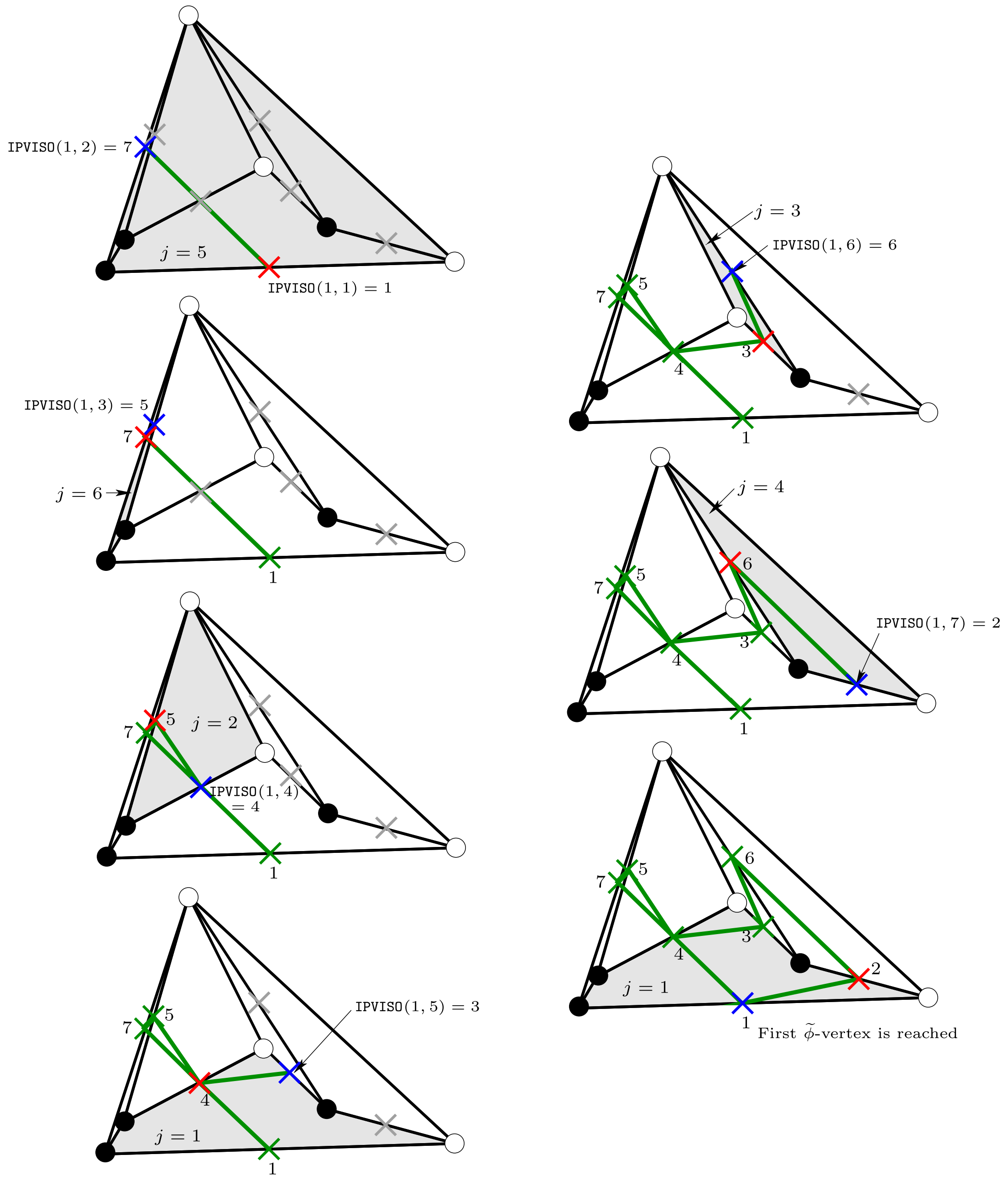


Fig. 8. Same isopolygon construction sequence as in Fig. 7, but for the example of Fig. 4(b).

Note that the proposed method relies on general rules which are able to extract isosurfaces on arbitrary polyhedral cells, while the marching cubes technique is restricted to a list of standard configurations for cubic cells. For comparison purposes, Fig. 13 presents the isosurfaces of two tests described in [19], where the marching cube algorithm with an extended modified lookup table with 21 patterns is used. The use of lookup tables in the marching cubes algorithm speeds up the surface extraction, making our more general algorithm to be, on average, 1.8 times slower (the tests were run for the algorithm in [19] and the proposed algorithm on an iMac Pro (2017) with a 2.3 GHz Intel Xeon W processor using the `gcc` compiler with `-O3` compilation option; a $\times 6$ speedup is achieved for the proposed algorithm using the `-fopenmp` compilation option and all the available threads). However, our method is not constrained to using a given cell geometry. Obviously, to accelerate the CPU-time and obtain a computational efficiency similar to that of the marching cubes algorithm, the $\widetilde{\phi}$-vertex connectivity could have been pre-computed with the proposed method for each of the possible 2^8 cube configurations and stored in a lookup table for rapid access. The same applies for other cell types.

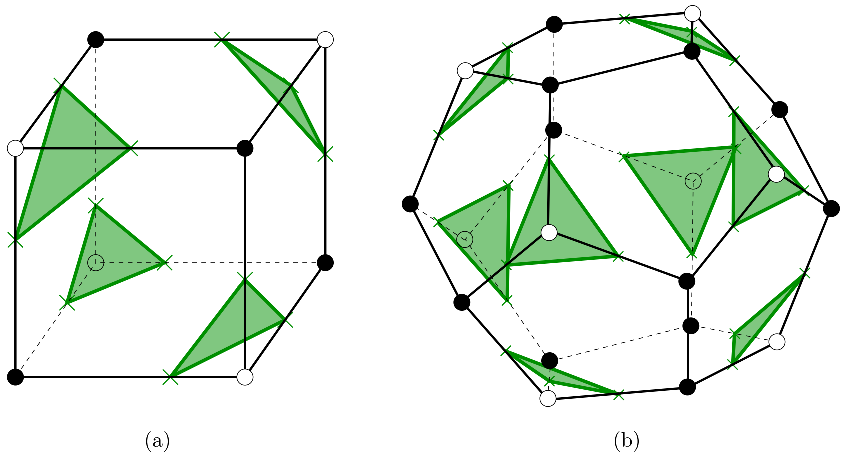


Fig. 9. Examples of multiple isopolygons constructed on (a) a cubic cell and (b) a dodecahedral cell.

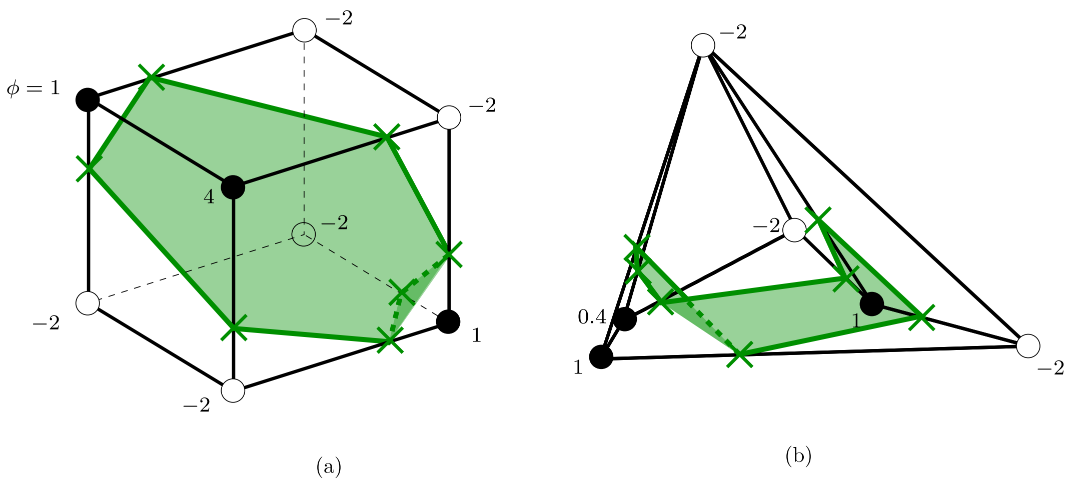


Fig. 10. Resulting isopolygons defined by the position vectors $\boldsymbol{x}_{\tilde{\phi}}$ computed from the interpolation given by Eq. (1) with $\tilde{\phi} = 0$.

3.2. Application to several grid types

In this section, the following grids in a unit domain are used:

1. Structured grid with cubic cells (Fig. 14(a)).
2. Unstructured grid with tetrahedral cells (Fig. 14(b)), obtained using `TetGen` [32] (version 1.5) by typing

```
tetgen option domain.poly
```

   where the content of file `domain.poly` is included in Appendix A and the values of the parameter `option`, used to generate the grids with different resolutions considered in this work, are shown in Table 3.
3. Structured grid with non-convex cells (Fig. 14(c)), obtained by distorting cubic cells as follows. Each of the eight corner vertices of the initial uniform cubic grid of cell size h is randomly moved to the surface of a sphere with radius $0.25h$ and centered in the corresponding vertex. Each face of the distorted cell, which is generally non-planar, is triangulated by joining its center with two consecutive vertices of the face, resulting in a non-convex polyhedron of 14 vertices and 24 triangular faces.
4. Unstructured grid with non-convex irregular polyhedral cells (Fig. 14(d)), obtained with the aid of `TetGen` [32] and `OpenFOAM`'s `tetgenToFoam` and `polyDualMesh` tools (see [21] for information on how to use these tools) by typing

```
tetgen option domain.poly
tetgenToFoam domain.1
polyDualMesh 75 -concaveMultiCells
```

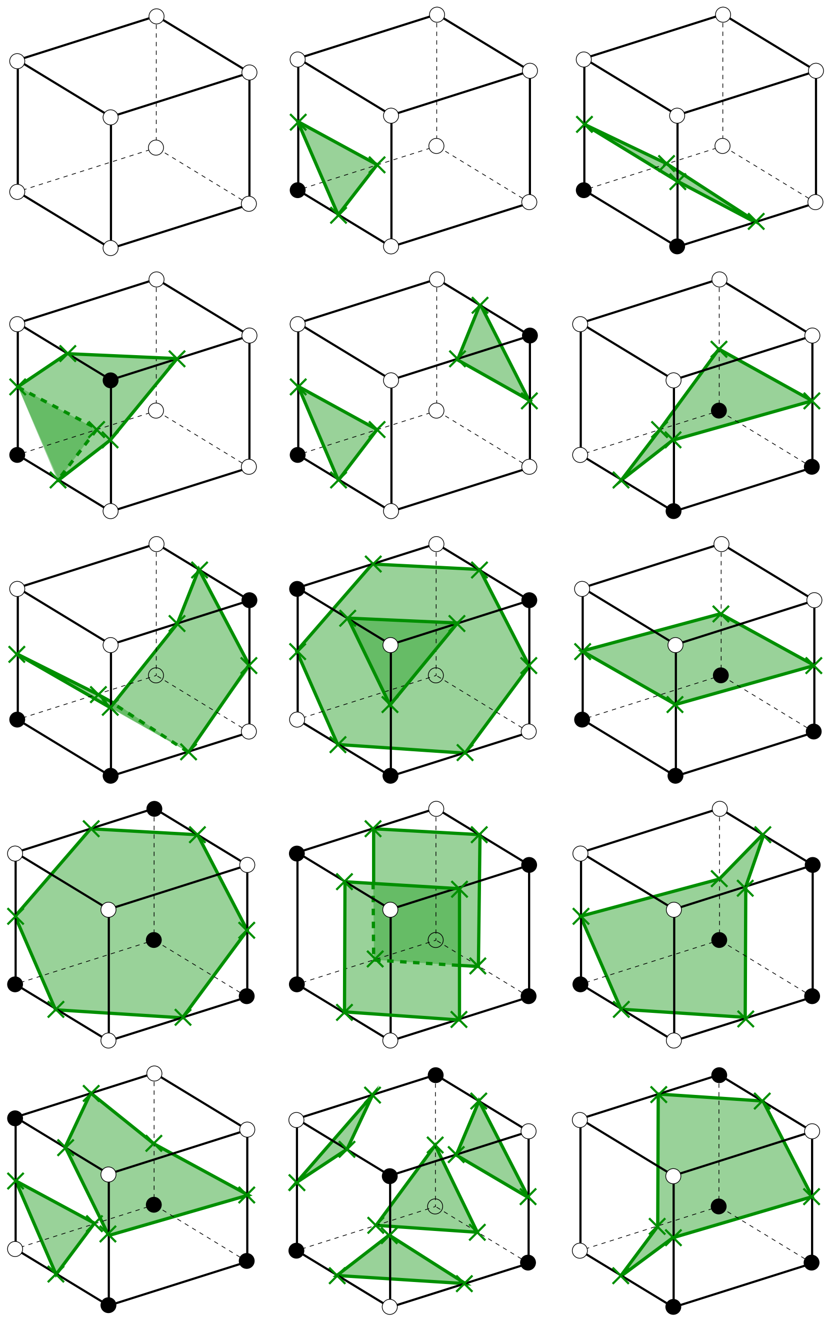


Fig. 11. Results obtained with the proposed method for the 14 pre-defined cell configurations considered in [12].

It should be mentioned that the faces of these cells are generally non-planar. For the VOF simulation carried out in Section 3.2.3, each cell face has additionally been triangulated by joining its geometric center with two consecutive face vertices.

The isosurfaces extracted from ϕ values obtained in different ways are presented below for grids with $n \simeq 80^3$ cells, in order to demonstrate the versatility of the proposed algorithm.

3.2.1. Scalar field given by implicit functions

The scalar field ϕ is defined at each cell vertex i_p by the following implicit functions:

$$\phi_{i_p} = 0.325^2 - (x_{i_p} - x_c)^2 - (y_{i_p} - y_c)^2 - (z_{i_p} - z_c)^2 \tag{2}$$

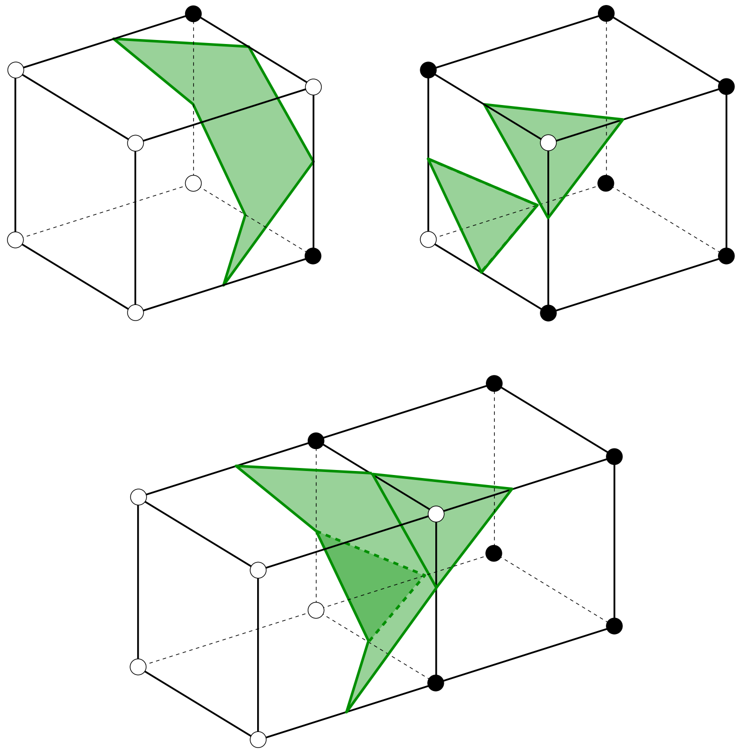


Fig. 12. The proposed approach avoids the inconsistency of Fig. 2.

Table 3
Details of the tetrahedral and non-convex irregular polyhedral grids of different resolutions used to assess the proposed methods. The numbers of grid vertices and faces obtained after triangulating the faces of the non-convex irregular polyhedral grids used in the VOF simulation are included in parenthesis.

Approx. $n^{1/3}$	`tetgen` option	Grid cells, n	Grid vertices	Grid faces	Avg. vertices per cell	Avg. faces per cell
Tetrahedral grids						
20	`-pq1.2a0.00031k`	7 923	1 831	16 907	4	4
32	`-pq1.2a0.00007k`	33 010	6 823	69 080	4	4
40	`-pq1.2a0.000034k`	63 988	12 400	132 550	4	4
64	`-pq1.2a0.0000079k`	261 051	47 201	534 515	4	4
80	`-pq1.2a0.0000039k`	513 711	89 902	1 046 584	4	4
128	`-pq1.2a0.00000093k`	2 094 575	351 874	4 239 540	4	4
160	`-pq1.2a0.00000047k`	4 138 041	687 618	8 360 839	4	4
Non-convex irregular polyhedral grids						
20	`-pq1.2a0.000056k`	8 077	48 947	56 673	22.8	13.4
32	`-pq1.2a0.000012k`	32 887	201 401	233 551	23.6	13.8
			(434 952)	(1 201 904)	(37.4)	(70.8)
40	`-pq1.2a0.0000056k`	64 207	396 276	459 748	24.1	14.0
64	`-pq1.2a0.00000127k`	262 122	1 638 789	1 899 408	24.6	14.3
			(3 538 197)	(9 819 651)	(38.9)	(73.9)
80	`-pq1.2a0.00000063k`	510 961	3 210 090	3 719 548	24.8	14.4
			(6 929 638)	(19 247 250)	(39.3)	(74.5)
128	`-pq1.2a0.000000147k`	2 101 279	13 312 234	15 410 474	25.2	14.6
			(28 722 708)	(79 847 053)	(39.7)	(75.5)
160	`-pq1.2a0.0000000745k`	4 081 207	25 935 456	30 013 624	25.3	14.6

for a sphere,

$$\phi_{i_p} = 0.1^2 - \left\{0.2 - \left[(x_{i_p} - x_c)^2 + (y_{i_p} - y_c)^2\right]^{1/2}\right\}^2 - (z_{i_p} - z_c)^2 \tag{3}$$

for a torus,

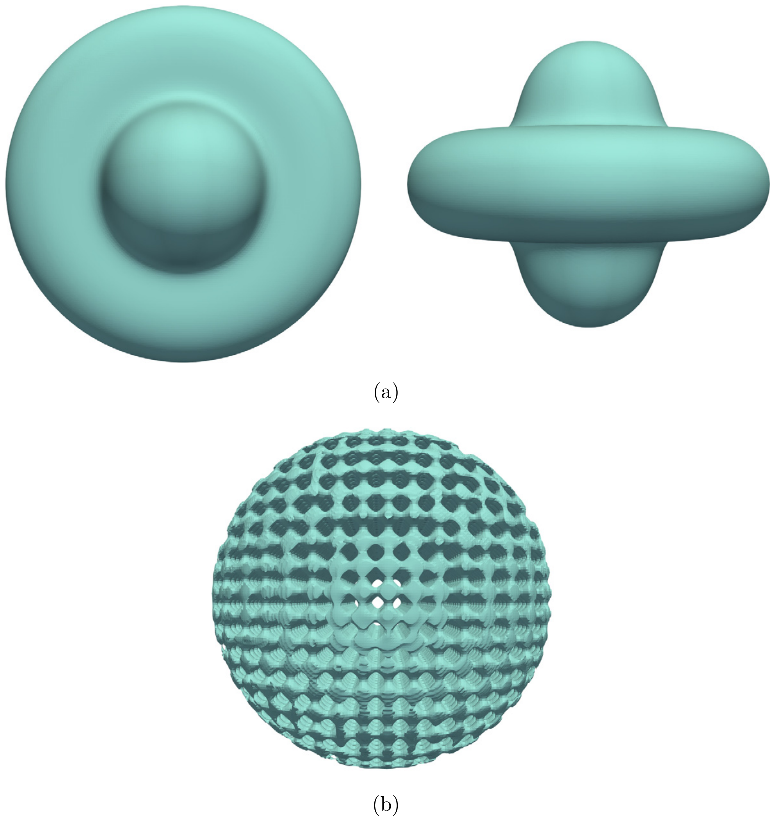


Fig. 13. Extracted isosurfaces obtained from the 200^3 volumetric data provided by Masala et al. [19] for two tests, using a grid with 199^3 cubic cells: (a) 'visual' ($\widetilde{\phi} = 0.5$) and (b) 'net' ($\widetilde{\phi} = 1.05$) tests.

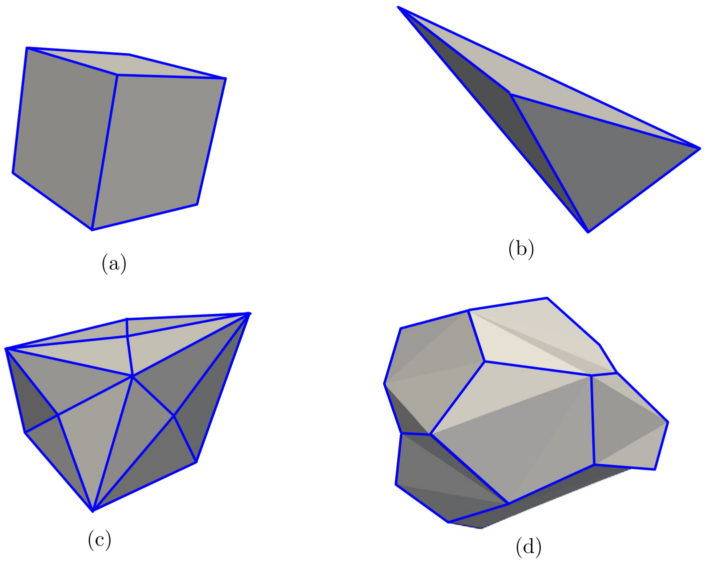


Fig. 14. Cell types: (a) cube, (b) tetrahedron, (c) distorted cube and (d) non-convex irregular polyhedron (note that this cell has non-planar faces).

$$\phi_{i_p} = 1 - \left(\frac{x_{i_p} - x_c}{0.4}\right)^2 - \left(\frac{y_{i_p} - y_c}{0.3}\right)^2 - \left(\frac{z_{i_p} - z_c}{0.2}\right)^2 \tag{4}$$

for an ellipsoid, and

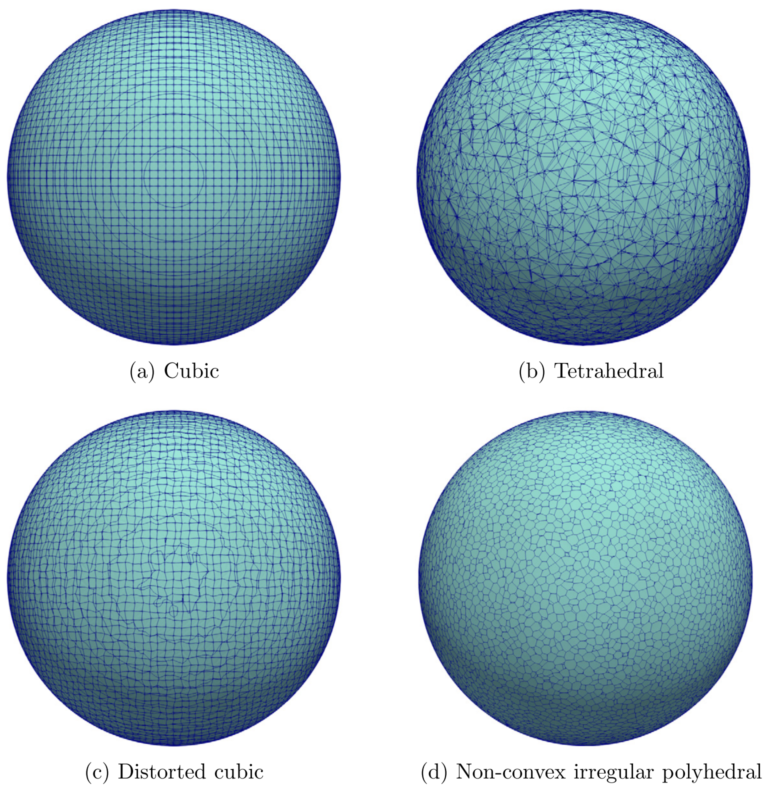


Fig. 15. Extracted isosurfaces of $\widetilde{\phi}=0$ for the sphere of Eq. (2), obtained using the (a) cubic, (b) tetrahedral, (c) distorted cubic and (d) non-convex irregular polyhedral grids with $n \simeq 80^3$.

$$\phi_{i_p} = -\frac{1}{\sqrt{3}}(x_{i_p} - x_c) - \frac{1}{\sqrt{3}}(y_{i_p} - y_c) - \frac{1}{\sqrt{3}}(z_{i_p} - z_c) \tag{5}$$

for a half-space, where $x_c = 0.525$, $y_c = 0.464$ and $z_c = 0.516$, and x_{i_p}, y_{i_p} and z_{i_p} are the position vector coordinates of vertex i_p.

Figs. 15, 16 and 17 show the extracted isosurfaces of $\widetilde{\phi}=0$ for the sphere of Eq. (2), the torus of Eq. (3) and the ellipsoid of Eq. (4), respectively, obtained using the (a) cubic, (b) tetrahedral, (c) distorted cubic and (d) non-convex irregular polyhedral grids.

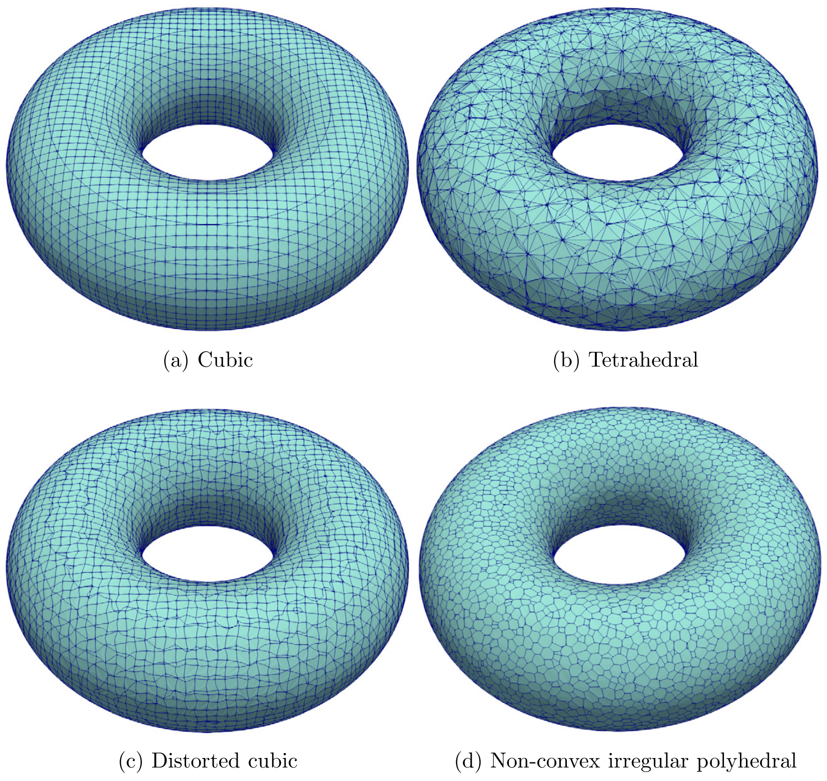
To quantify the accuracy of the isosurface extraction, the error norms

$$E^{L_\infty}_{\widetilde{\phi}} = \max\left(\left|\phi(\boldsymbol{x}_{\widetilde{\phi}})\right|\right) \tag{6}$$

and

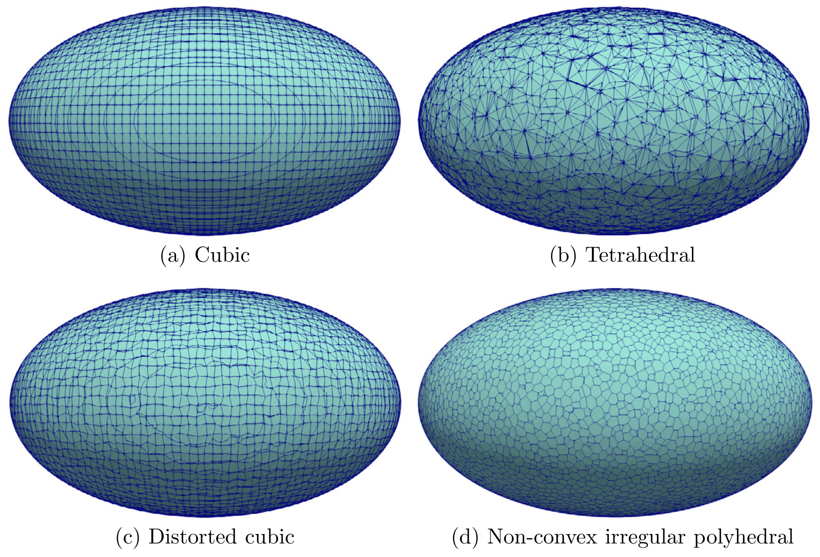
$$E^{L_1}_{\widetilde{\phi}} = \frac{\sum \left|\phi(\boldsymbol{x}_{\widetilde{\phi}})\right|}{N_{\widetilde{\phi}}} \tag{7}$$

are considered, where the max operator and the summation extend over all the $N_{\widetilde{\phi}}$ $\widetilde{\phi}$-vertices, and $\phi(\boldsymbol{x}_{\widetilde{\phi}})$ is obtained from Eqs. (2), (3), (4) or (5) by replacing x_{i_p}, y_{i_p} and z_{i_p} with the corresponding coordinates of each extracted $\widetilde{\phi}$-vertex $\boldsymbol{x}_{\widetilde{\phi}}$. The values of these error norms, along with the corresponding convergence orders, are shown in Tables 4, 5, 6 and 7 for the sphere, torus, ellipsoid and half space, respectively. As expected for the linearly approximated isosurface of Eq. (1) [33], it can be observed from these table, for any of the grids considered, a second-order convergence for the sphere, torus and ellipsoid, and errors on the order of the machine precision for the half-space. The test programs supplied with the software package use an interval-based approach to create the list of cells over which the algorithms presented in this work are applied (i.e., cells with maximum and minimum scalar values greater and lower, respectively, than $\widetilde{\phi}$). The user could incorporate more sophisticated search approaches, such as those described, for example, in [34]. The execution times consumed to identify these cells, t_{i}, and extract the isosurface on them, $t_{\widetilde{\phi}}$, have also been included in the tables. It can be seen that, for a particular combination of grid type and implicit function, the variation of t_{i} and $t_{\widetilde{\phi}}$ with respect to the total number of grid cells, n, and identified cells, $n_{\widetilde{\phi}}$, respectively, is roughly linear (see the example of Fig. 18). The slopes of these variations decrease as the number of threads used in the simulation execution increases. Using the maximum


(a) Cubic (b) Tetrahedral

(c) Distorted cubic (d) Non-convex irregular polyhedral

Fig. 16. Same results as in Fig. 15, but for the torus of Eq. (3).


(a) Cubic (b) Tetrahedral

(c) Distorted cubic (d) Non-convex irregular polyhedral

Fig. 17. Same results as in Fig. 15, but for the ellipsoid of Eq. (4).

number of threads allowed by the processor used for this test, instead of only one, the execution can be speed up in some cases by a factor of around 10 for t_{i} and 20 for $t_{\widetilde{\phi}}$. Also note that, as expected, for a particular implicit function and grid size n, the execution times tend to increase as the grid cell complexity increases (the high complexity of the non-convex irregular polyhedral grids becomes evident from the high values of the average number of vertices and faces per cell shown in Table 3).

The $E_{\widetilde{\phi}}^{L_\infty}$ values obtained for $\phi_{i_p} = 1 - (x_{i_p} - 4)^2 - (y_{i_p} - 4)^2 - (z_{i_p} - 4)^2$ in a domain 8^3 using grids with cubic cells of size h, are compare in Fig. 19 with those reported in [33] and obtained with six different isosurface extraction codes:

Table 4
$E_{\widetilde{\phi}}^{L_\infty}$, $E_{\widetilde{\phi}}^{L_1}$, convergence order ($\mathcal{O}$) and execution time values for the isosurface corresponding to the sphere of Eq. (2) extracted on different grids.

n	$n_{\widetilde{\phi}}$	$E_{\widetilde{\phi}}^{L_\infty}$	$\mathcal{O}$	$E_{\widetilde{\phi}}^{L_1}$	$\mathcal{O}$	t_i (µs)	$t_{\widetilde{\phi}}$ (µs)
Cubic cells							
20^3	786	6.25×10^{-4}		4.07×10^{-4}		354	252
40^3	3174	1.56×10^{-4}	2.0	1.05×10^{-4}	2.0	543	317
80^3	12720	3.91×10^{-5}	2.0	2.59×10^{-5}	2.0	2965	562
160^3	50936	9.77×10^{-6}	2.0	6.50×10^{-6}	2.0	20022	2153
Tetrahedral cells							
20^3	883	9.14×10^{-3}		2.72×10^{-3}		308	218
40^3	3847	2.10×10^{-3}	2.1	6.17×10^{-4}	2.1	453	272
80^3	16158	5.17×10^{-4}	2.0	1.49×10^{-4}	2.0	2656	640
160^3	66642	1.35×10^{-4}	1.9	3.63×10^{-5}	2.0	20842	3123
Distorted cubic cells							
20^3	827	1.30×10^{-3}		3.48×10^{-4}		274	277
40^3	3297	3.31×10^{-4}	2.0	8.71×10^{-5}	2.0	509	549
80^3	13264	8.47×10^{-5}	2.0	2.20×10^{-5}	2.0	2866	1830
160^3	53030	2.13×10^{-5}	2.0	5.48×10^{-6}	2.0	22831	7247
Non-convex irregular polyhedral cells							
20^3	555	4.50×10^{-4}		1.13×10^{-4}		309	273
40^3	2523	7.20×10^{-5}	2.6	2.44×10^{-5}	2.2	780	621
80^3	11033	2.18×10^{-5}	1.7	5.70×10^{-6}	2.1	5280	3275
160^3	45263	5.60×10^{-6}	2.0	1.38×10^{-6}	2.0	78783	16057

Table 5
Same results as in Table 4, but for the torus of Eq. (3).

n	$n_{\widetilde{\phi}}$	$E_{\widetilde{\phi}}^{L_\infty}$	$\mathcal{O}$	$E_{\widetilde{\phi}}^{L_1}$	$\mathcal{O}$	t_i (µs)	$t_{\widetilde{\phi}}$ (µs)
Cubic cells							
20^3	460	6.25×10^{-4}		3.52×10^{-4}		315	226
40^3	1820	1.56×10^{-4}	2.0	8.93×10^{-5}	2.0	570	304
80^3	7296	3.91×10^{-5}	2.0	2.13×10^{-5}	2.1	2803	415
160^3	29256	9.77×10^{-6}	2.0	5.47×10^{-6}	2.0	20326	1448
Tetrahedral cells							
20^3	498	7.25×10^{-3}		1.88×10^{-3}		316	209
40^3	2264	2.21×10^{-3}	1.7	4.73×10^{-4}	2.0	489	247
80^3	9643	4.56×10^{-4}	2.3	1.14×10^{-4}	2.1	2540	493
160^3	39439	1.22×10^{-4}	1.9	2.79×10^{-5}	2.0	20421	1884
Distorted cubic cells							
20^3	468	1.27×10^{-3}		2.80×10^{-4}		292	220
40^3	1888	3.25×10^{-4}	2.0	6.85×10^{-5}	2.0	642	388
80^3	7856	8.29×10^{-5}	2.0	1.69×10^{-5}	2.0	2577	1200
160^3	30779	2.17×10^{-5}	1.9	4.27×10^{-6}	2.0	22835	4521
Non-convex irregular polyhedral cells							
20^3	335	2.90×10^{-4}		8.23×10^{-5}		290	251
40^3	1543	7.22×10^{-5}	2.0	1.87×10^{-5}	2.1	679	467
80^3	6435	1.91×10^{-5}	1.9	4.43×10^{-6}	2.1	5052	2019
160^3	27053	4.98×10^{-6}	1.9	1.06×10^{-6}	2.1	81421	9710

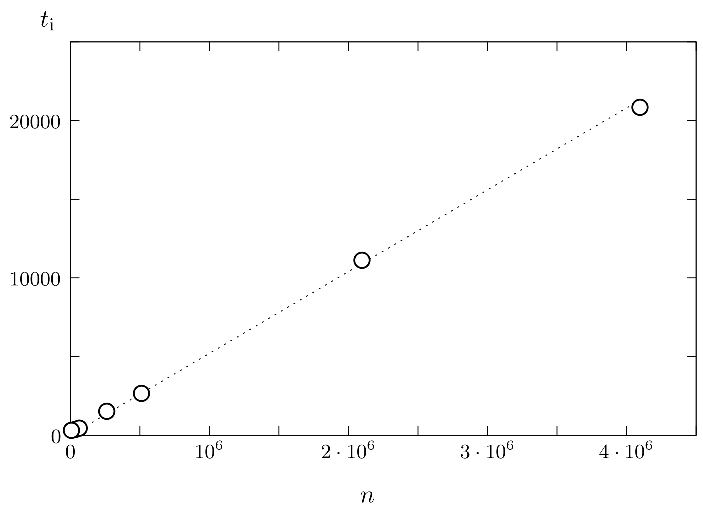
vtk Marching Cubes [12], SnapMC [35], Macet [36], Dual Contouring [37], Afront [38] and DelIso [39]. As expected, the $E_{\widetilde{\phi}}^{L_\infty}$ values obtained with the proposed method almost coincide with the vtk Marching Cubes and Snap MC codes since both use the same linear approximation of Eq. (1). As mentioned in Section 2.4, higher-order interpolations must be used to obtained more accurate results.

3.2.2. *Scalar field given by signed distances to scanned surfaces*

In this section, the scalar field ϕ is obtained from the scanned surface of two objects. The first corresponds to the well-known Stanford bunny [40] and the second to an automobile part available in our laboratory.

The scanned Stanford bunny is available in [40] for download in polygon (PLY) format. We used the file composed by 35947 vertices and 69451 triangles. The scalar field ϕ is assigned to each grid point from its signed distance (negative inside the surface and positive outside) to the triangulated bunny surface. This signed distance is computed using the `dist3d` routine presented in [41,42]. Fig. 20 shows the extracted isosurfaces of $\widetilde{\phi} = 0$ for the four grid types considered.

An aluminum alloy A413 automobile part (Fig. 21(a)) was scanned to provide a triangulated surface in PLY format composed of 156229 vertices and 312458 triangles (Fig. 21(b)). As in the previous case, the scalar field ϕ at each grid point is obtained from the signed distance to the triangulated surface. Fig. 22 shows the extracted isosurfaces of $\widetilde{\phi} = 0$ for



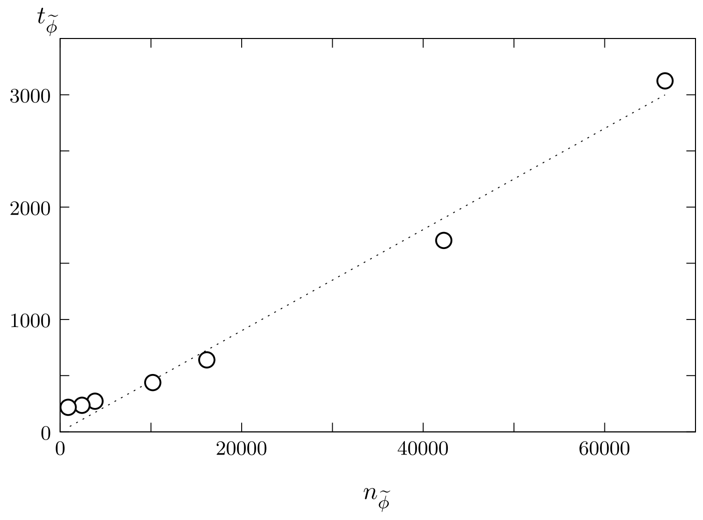


Fig. 18. Execution times t_i (top picture) and $t_{\widetilde{\phi}}$ (bottom picture) as a function of n and $n_{\widetilde{\phi}}$, respectively, for the sphere of Eq. (2) and tetrahedral grids of different sizes.

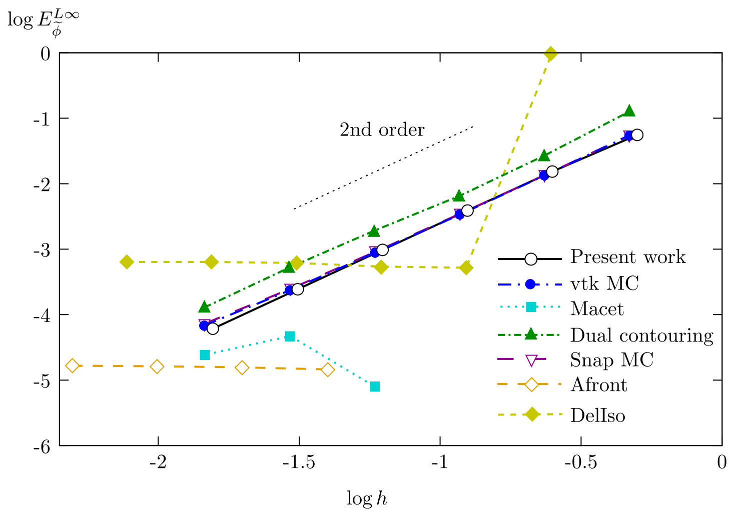


Fig. 19. Error $E^{L_\infty}_{\widetilde{\phi}}$ as a function of cell size, h, obtained for a cubic grid and $\phi_{i_p} = 1 - (x_{i_p} - 4)^2 - (y_{i_p} - 4)^2 - (z_{i_p} - 4)^2$, in a domain of size 8^3. Comparison of the proposed method with different isosurface extraction methods.

Table 6
Same results as in Table 4, but for the ellipsoid of Eq. (4).

n	$n_{\widetilde{\phi}}$	$E_{\widetilde{\phi}}^{L_\infty}$	$\mathcal{O}$	$E_{\widetilde{\phi}}^{L_1}$	$\mathcal{O}$	t_i (μs)	$t_{\widetilde{\phi}}$ (μs)
Cubic cells							
20^3	664	1.56×10^{-2}		6.76×10^{-3}		365	269
40^3	2616	3.91×10^{-3}	2.0	1.70×10^{-3}	2.0	584	305
80^3	10436	9.77×10^{-4}	2.0	4.29×10^{-4}	2.0	2265	490
160^3	41808	2.44×10^{-4}	2.0	1.07×10^{-4}	2.0	19210	1717
Tetrahedral cells							
20^3	707	1.75×10^{-1}		3.71×10^{-2}		337	236
40^3	3227	4.26×10^{-2}	2.0	9.57×10^{-3}	2.0	395	260
80^3	13646	1.07×10^{-2}	2.0	2.25×10^{-3}	2.1	2763	573
160^3	55745	3.18×10^{-3}	1.8	5.47×10^{-4}	2.0	18920	2399
Distorted cubic cells							
20^3	697	2.98×10^{-2}		5.44×10^{-3}		248	238
40^3	2762	7.83×10^{-3}	1.9	1.32×10^{-3}	2.0	583	443
80^3	10939	2.04×10^{-3}	1.9	3.32×10^{-4}	2.0	2615	1536
160^3	43858	5.43×10^{-4}	1.9	8.39×10^{-5}	2.0	21396	6008
Non-convex irregular polyhedral cells							
20^3	478	8.59×10^{-3}		1.69×10^{-3}		307	278
40^3	2144	1.75×10^{-3}	2.3	3.68×10^{-4}	2.2	803	580
80^3	9203	4.50×10^{-4}	2.0	8.65×10^{-5}	2.1	5143	2734
160^3	38088	1.13×10^{-4}	2.0	2.08×10^{-5}	2.1	77255	13453

Table 7
Same results as in Table 4, but for the half-space of Eq. (5) (obviously, the convergence orders are dropped).

n	$n_{\widetilde{\phi}}$	$E_{\widetilde{\phi}}^{L_\infty}$	$E_{\widetilde{\phi}}^{L_1}$	t_i (μs)	$t_{\widetilde{\phi}}$ (μs)
Cubic cells					
20^3	898	5.55×10^{-17}	8.24×10^{-18}	323	273
40^3	3598	5.55×10^{-17}	8.05×10^{-18}	596	296
80^3	14398	5.55×10^{-17}	8.73×10^{-18}	2540	578
160^3	57598	6.25×10^{-17}	9.71×10^{-18}	21066	2130
Tetrahedral cells					
20^3	995	1.11×10^{-16}	2.25×10^{-17}	334	235
40^3	4041	1.11×10^{-16}	2.30×10^{-17}	570	273
80^3	16442	1.11×10^{-16}	2.41×10^{-17}	2888	656
160^3	66130	1.67×10^{-16}	2.39×10^{-17}	21560	2792
Distorted cubic cells					
20^3	903	1.39×10^{-16}	2.37×10^{-17}	265	289
40^3	3603	1.67×10^{-16}	2.38×10^{-17}	576	511
80^3	14401	1.67×10^{-16}	2.38×10^{-17}	3178	1494
160^3	57605	1.77×10^{-16}	2.41×10^{-17}	23151	6443
Non-convex irregular polyhedral cells					
20^3	660	1.11×10^{-16}	2.27×10^{-17}	300	273
40^3	2729	1.63×10^{-16}	2.43×10^{-17}	730	629
80^3	11106	1.67×10^{-16}	2.40×10^{-17}	5329	3429
160^3	45499	1.67×10^{-16}	2.40×10^{-17}	79377	16192

the four grid types considered. The resolution of the tetrahedral grid was increased to $\mathcal{N} = 128$ to avoid losing too much detail.

It should be mentioned that the accuracy of the extraction of isosurfaces using signed distances to scanned surfaces obviously depends, apart from the factors discussed in the previous section, on the surface scanner and signed distances accuracies. A detailed assessment of the procedure used in this work to computed signed distances can be found in [43].

3.2.3. *Scalar field given by volume fractions*

For dealing with the complex interfacial shapes that typically arise in simulations of multiphase fluid flows using VOF methods, the evolution equation for an indicator function $\chi(\boldsymbol{x}, t)$, which is equal to 1 if $\boldsymbol{x}$ is inside the fluid and 0 otherwise,

$$\frac{\partial \chi(\boldsymbol{x}, t)}{\partial t} + \nabla \cdot [\boldsymbol{v} \chi(\boldsymbol{x}, t)] - \chi(\boldsymbol{x}, t) \nabla \cdot \boldsymbol{v} = 0, \tag{8}$$

is integrated over a given cell, Ω, of volume V_Ω, and the time interval $\Delta t = t^{\mathrm{n}+1} - t^{\mathrm{n}}$, to obtain, at each time step,

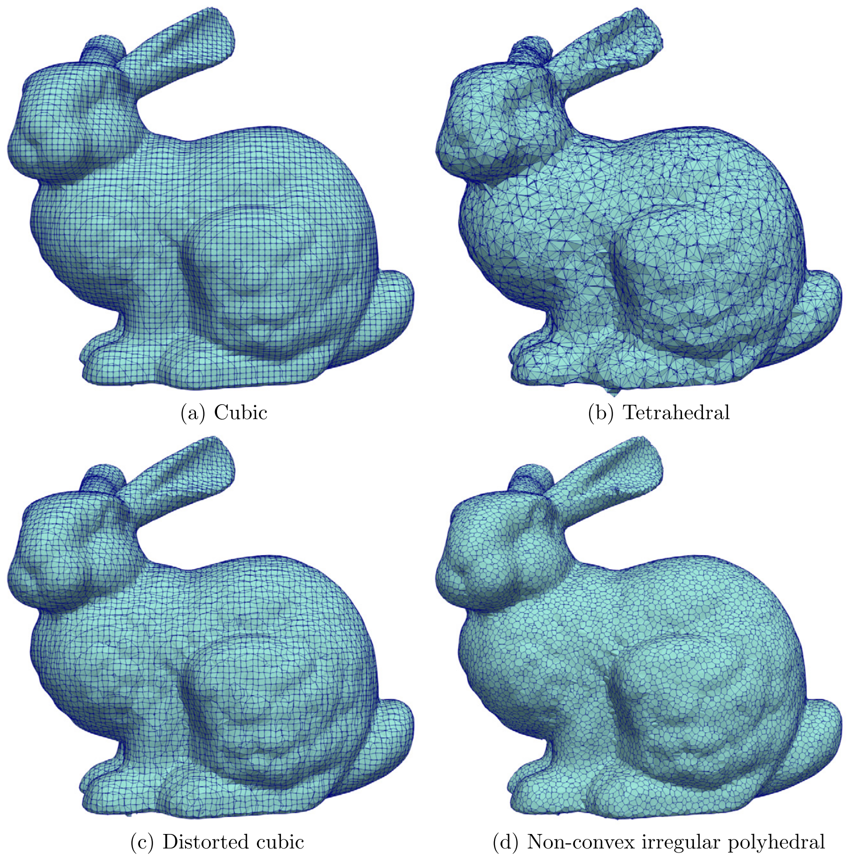


Fig. 20. Same results as in Fig. 15, but for the scanned Stanford bunny.

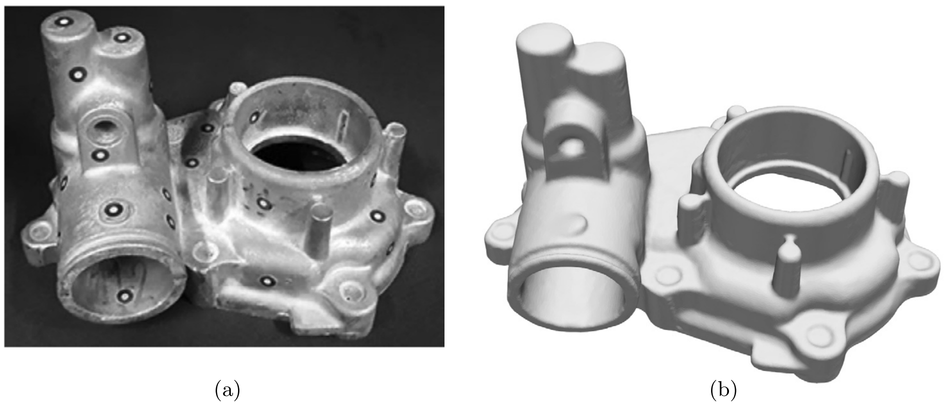


Fig. 21. Aluminum alloy A413 automobile part: (a) real and (b) scanned.

$$
\begin{aligned}
F(t^{\mathrm{n}+1}) = F(t^{\mathrm{n}}) - \frac{1}{V_\Omega} \int_{t^{\mathrm{n}}}^{t^{\mathrm{n}+1}} \int_{\Omega} \nabla \cdot [\boldsymbol{v}\chi(\boldsymbol{x},t)] \,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t \\
+ \frac{F(t^{\mathrm{n}+1}) - F(t^{\mathrm{n}})}{2V_\Omega} \int_{t^{\mathrm{n}}}^{t^{\mathrm{n}+1}} \int_{\Omega} \nabla \cdot \boldsymbol{v} \,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t,
\end{aligned}
\tag{9}
$$

where F is a discretized version of function χ, whose value in each cell of the computational grid is the fraction of the cell occupied by the fluid. In the test described below, a spherical fluid volume is subjected to advection in a prescribed

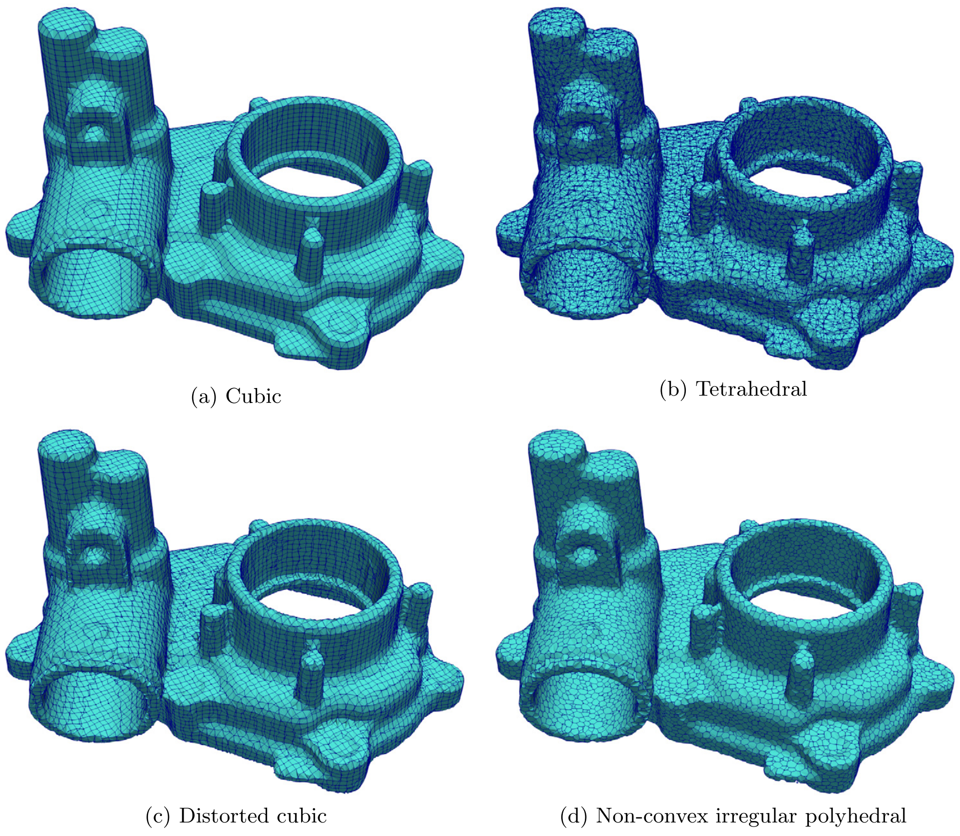


Fig. 22. Same results as in Fig. 15, but for the scanned automobile part of Fig. 21(b). To avoid losing too much detail, the grid resolution for the tetrahedral grid has been increased from 80^3 to 128^3 cells.

velocity field. At the initial time step, $F(t^{n=0})$ is obtained in each cell using a grid refinement procedure [44,45], extended to non-convex grid cells in [26], for a spherical fluid body of radius 0.15 and centered at $(0.5, 0.75, 0.25)$ in a unit box domain. For subsequent time steps, $F(t^{n+1})$ is obtained from the solution of Eq. (9) assuming that the interface was previously reconstructed at the instant t^n using an extension to arbitrary grids of the CLCIR method proposed in [3], which has been implemented with the aid of the isosurface extraction procedure presented in this work to estimate the PLIC interface orientation on every cell where $10^{-12} < F(t^n) < (1 - 10^{-12})$, and is described in Appendix B. The first integral in Eq. (9) represents the net volume of fluid advected out of the cell and will be solved geometrically using the extension to 3D [46] of the edge-matched flux polygon advection (EMFPA) method proposed by López et al. [47], which avoids the over/underlapping between flux regions constructed at cell faces with common vertices. The repeated intersection operations required by this method over polyhedra that are generally non-convex, even with self-intersecting faces, are performed with the aid of the tools proposed in [26,48], thus avoiding the use of decomposition techniques. Note that the last term in Eq. (9) must be null if the velocity vector field $\boldsymbol{v}$ is discretely solenoidal.

3D vortex in a box test The solenoidal velocity field $\boldsymbol{v} = (u, v, w)$, defined by

$$\begin{aligned} u &= \sin^2(\pi x)\sin(2\pi y)\cos(\pi t/3) \\ v &= -\sin^2(\pi y)\sin(2\pi x)\cos(\pi t/3) \\ w &= \left\{1 - 2\left[(x-0.5)^2 + (y-0.5)^2\right]^{1/2}\right\}^2 \cos(\pi t/3), \end{aligned} \tag{10}$$

corresponds to a combination of the classical vortex-in-a-box test of Rider and Kothe [49] with a laminar pipe flow in the z-direction, as in the work of Liovic et al. [50] (variants of this single-vortex test can also be found, for example, in [51,52], among many other references). Figs. 23 and 24 show the extracted isosurfaces for $\widetilde{\phi} = 0.5$ at different instants using the cubic and tetrahedral grids, and the distorted cubic and non-convex irregular polyhedral grids, respectively, with $n \simeq 80^3$ cells. Note that the fluid interface reaches its maximum deformation at instant 1.5 and should return to its initial state at instant 3 (any visual difference with the spherical body shape at $t = 0$ is due to numerical errors in the solution of Eq. (9)).

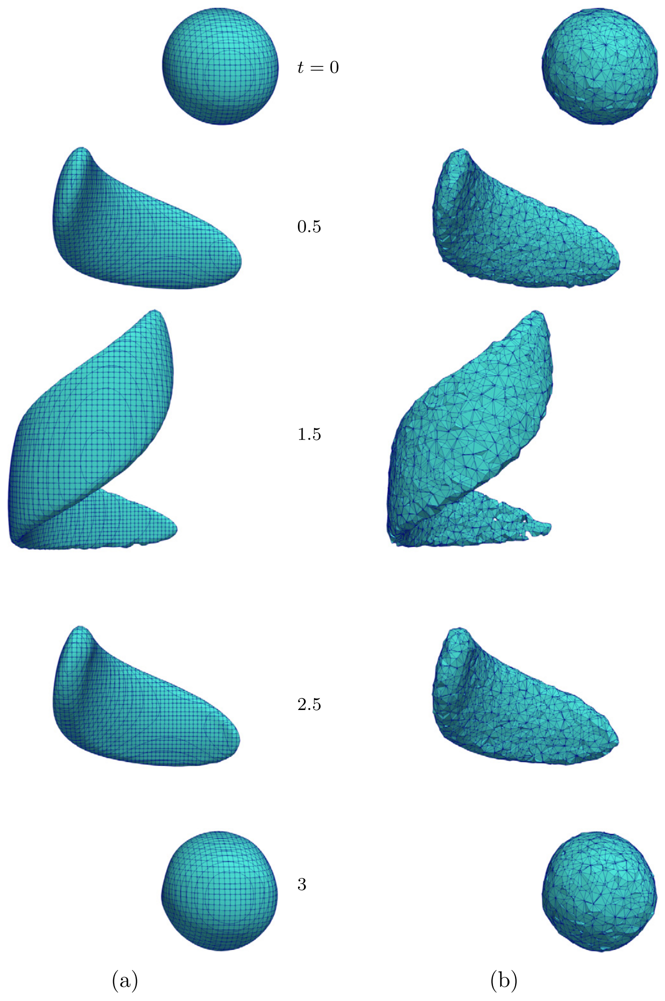


Fig. 23. Extracted isosurfaces for $\widetilde{\phi} = 0.5$ at different instants in the 3D vortex in a box test, using the EMFPA and CLCIR methods and the (a) cubic and (b) tetrahedral grids with $n \simeq 80^3$.

To assess the performance of the implemented algorithms, results obtained using grids of different types and resolutions are compared with the exact solution at the end of the VOF simulation ($t = t_{\text{end}}$). Fig. 25 shows an example of this comparison (the pictures are depicted in semitransparent colors to better see the comparison) for cubic grids. The EMFPA method was used in combination with two PLIC methods: the CLCIR method (left pictures) and the well-known method of Youngs (right pictures). Although some differences can be qualitatively perceived from the figure, especially for the coarser grids, the quantitative differences can be estimated using the following error norm:

$$E^{L_1}_{\text{shape}} = \sum_{i=1}^{n} V_{\Omega_i} \left| F^{\text{e}}_i(t_{\text{end}}) - F_i(t_{\text{end}}) \right| , \tag{11}$$

where $F^{\text{e}}_i(t_{\text{end}})$ and $F_i(t_{\text{end}})$ are, respectively, the exact and computed fluid volume fraction in cell i at instant $t_{\text{end}} = 3$. It should be mentioned that results obtained using unstructured grids are very scarce in the literature, and the few that can be found do not allow a rigorous comparison due to the lack of information about the grid generation. Scheufler and Roenby [6] and Jofre et al. [53] provide results on tetrahedral grids with similar number of cells, although not generated in the same way. Results on unstructured polyhedral grids for this test can also be found in [6], although details on the degree of complexity of the polyhedral cells are not provided. As mentioned, detailed information about the grids used in this work is included at the beginning of Section 3.2, which will allow for more rigorous future comparisons. Table 8 compares results obtained in this work using a Courant-Friedrich-Levy (CFL) number equal to 0.5 and grids of $n \simeq 32^3$, 64^3 and 128^3 cells, with those obtained in [6] using an improved version (isoAdvector-plicRDF, which reconstructs a distance function from PLIC interfaces) of the method based on isosurface extractions proposed by Roenby et al. [5], and with those

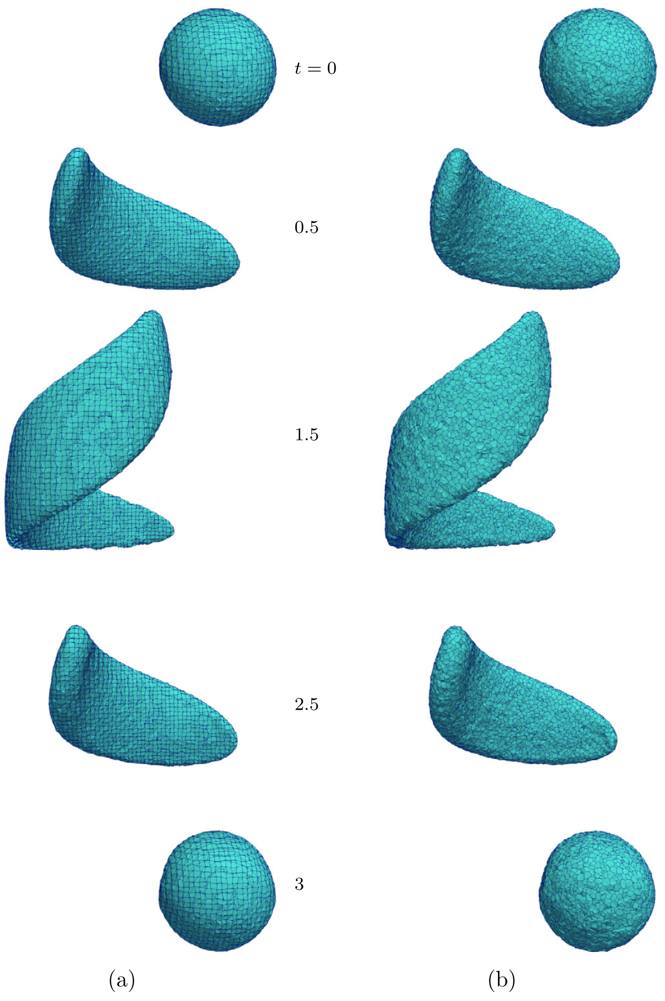


Fig. 24. Same results as in Fig. 23, but using the (a) distorted cubic and (b) non-convex irregular polyhedral grids with $n \simeq 80^3$.

obtained in [53] using an EMFPA-like advection method combined with the LVIRA (least-squares volume-of-fluid interface reconstruction algorithm) method. The time step Δt is determined at each instant t from

$$\Delta t = \min\left[\frac{\min(h_{x_i})}{\max(u_j)}, \frac{\min(h_{y_i})}{\max(v_j)}, \frac{\min(h_{z_i})}{\max(w_j)}\right]\mathrm{CFL} \tag{12}$$

where the min and max operators inside the square brackets involve, respectively, all the cells and faces for cubic grids and only cells and faces near the interface for unstructured grids; h_{x_i}, h_{y_i} and h_{z_i} are the sizes along the corresponding coordinate axis of the minimum-size rectangular parallelepiped that encloses cell i (note that, for cubic grids with n cells, $\min(h_{x_i}) = \min(h_{y_i}) = \min(h_{z_i}) = 1/n^{1/3}$); and u_j, v_j and w_j are the components of the velocity vector at the center of grid face j. The average execution time per time step, $\widetilde{t}_{\text{cpu}}$, is also included in the table (execution times can also be found in [6] for this test, although using different resources to execute the simulations, therefore these values should only be used as an approximated reference). For all the grids used, the results obtained with EMFPA-CLCIR methods show second-order convergence and lower error values than those obtained using the EMFPA and Youngs methods, although with a slightly higher CPU time consumed. For the cubic and tetrahedral grids, the EMFPA-CLCIR methods also show error values lower than those obtained in [6] and [53]. For the non-convex irregular polyhedral grids, the isoAdvector-plicRDF method provides lower errors on grids with $n \simeq 64^3$ and 128^3. Fig. 26 shows the PLIC interfaces at $t = 1.5$ and 3 using the cubic, tetrahedral and non-convex irregular polyhedral grids. A visual comparison with the results obtained using the isoAdvector-plicRDF method can be seen in Fig. 13 of reference [6]. It is clear from this comparison that the unstructured grids used in [6] are not identical to those used in this work and therefore, as mentioned above, conclusions about this comparison must be viewed with certain reservations.

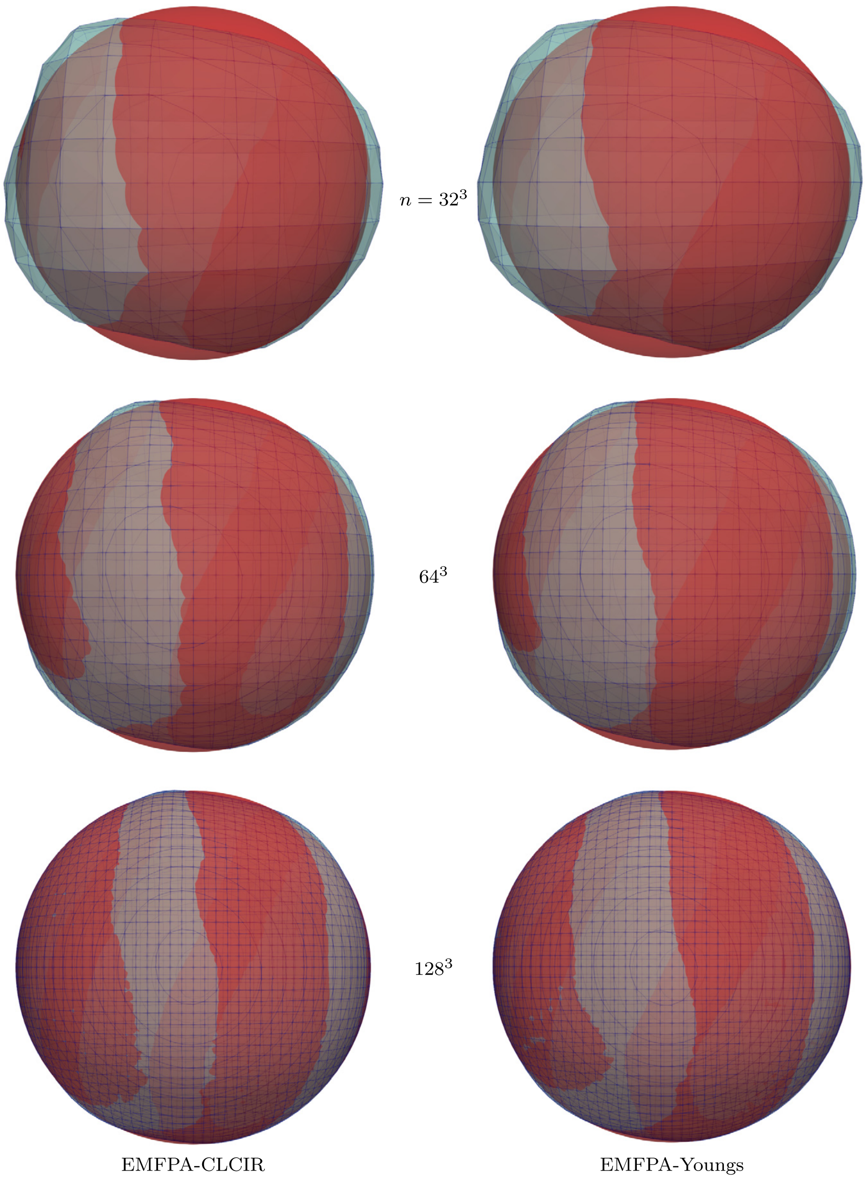


Fig. 25. Extracted isosurfaces for $\widetilde{\phi} = 0.5$ (semitransparent light blue color) at the end of the VOF simulation of the 3D vortex in a box test, using the EMFPA method in combination with the CLCIR (left pictures) and Youngs (right pictures) methods and cubic grids of different sizes (the exact solution is depicted in semitransparent red color).

4. Conclusions

In this paper we have proposed a novel method for isosurface extraction from a discrete data field, which can be advantageously used for interface reconstruction in volume of fluid methods on arbitrary grids. The performance and versatility of the new method, which can be applied not only to the simulation of multiphase flows but also in fields such as CFD visualization and medical imaging, among others, have been assessed through several tests using different grids with polyhedral cells, either convex or non-convex, with planar or non-planar faces, where the discrete values assigned to the grid nodes are obtained from different sources, such as implicit mathematical functions, signed distances to scanned surfaces or volume fractions obtained from a VOF scheme. The main novelties and advantages of the proposed method with respect to the existing ones are the following:

1) It maintains the local character of the marching cubes algorithm (only discrete data in the vertices of the considered cell are involved in the isosurface extraction), which has the advantage of allowing for easy parallelization, and avoids the

Table 8
$E^{L_1}_{\text{shape}}$ error obtained with the VOF simulation using the EMFPA method in combination with the CLCIR and Youngs reconstruction methods using different grids. Comparison with results from [6] and [53]. The convergence orders are included in parenthesis between the corresponding error values and the averaged execution times, $\widetilde{t}_{\text{cpu}}$ (s), are included on the right in square brackets.

Grid size	EMFPA CLCIR [$\widetilde{t}_{\text{cpu}}$]	EMFPA Youngs [$\widetilde{t}_{\text{cpu}}$]	isoAdvector plicRDF [6]	Jofre et al. [53]
Cubic cells				
32^3	3.21×10^{-3} [0.0088]	3.73×10^{-3} [0.0067]	4.06×10^{-3}	4.08×10^{-3}
	(1.9)	(1.8)	(1.9)	(1.5)
64^3	8.80×10^{-4} [0.028]	1.10×10^{-3} [0.023]	1.08×10^{-3}	1.46×10^{-3}
	(2.0)	(1.9)	(2.1)	(2.0)
128^3	2.14×10^{-4} [0.13]	2.91×10^{-4} [0.11]	2.44×10^{-4}	3.53×10^{-4}
Tetrahedral cells				
32^3	5.60×10^{-3} [0.018]	6.92×10^{-3} [0.015]	8.43×10^{-3}	5.97×10^{-3}
	(1.9)	(1.7)	(1.5)	(1.9)
64^3	1.46×10^{-3} [0.063]	2.15×10^{-3} [0.054]	2.88×10^{-3}	1.64×10^{-3}
	(2.1)	(1.8)	(2.4)	(1.6)
128^3	3.51×10^{-4} [0.37]	6.00×10^{-4} [0.25]	5.50×10^{-4}	5.37×10^{-4}
Distorted cubic cells				
32^3	3.64×10^{-3} [0.069]	4.00×10^{-3} [0.062]	−−	−−
	(1.8)	(1.7)		
64^3	1.02×10^{-3} [0.23]	1.23×10^{-3} [0.21]	−−	−−
	(2.0)	(1.9)		
128^3	2.60×10^{-4} [0.93]	3.41×10^{-4} [0.84]	−−	−−
Non-convex irregular polyhedral cells				
32^3	4.66×10^{-3} [0.089]	4.92×10^{-3} [0.076]	5.99×10^{-3}	−−
	(1.9)	(1.8)	(2.4)	
64^3	1.29×10^{-3} [0.36]	1.38×10^{-3} [0.31]	1.17×10^{-3}	−−
	(1.9)	(1.9)	(2.2)	
128^3	3.44×10^{-4} [2.19]	3.76×10^{-4} [1.81]	2.59×10^{-4}	−−

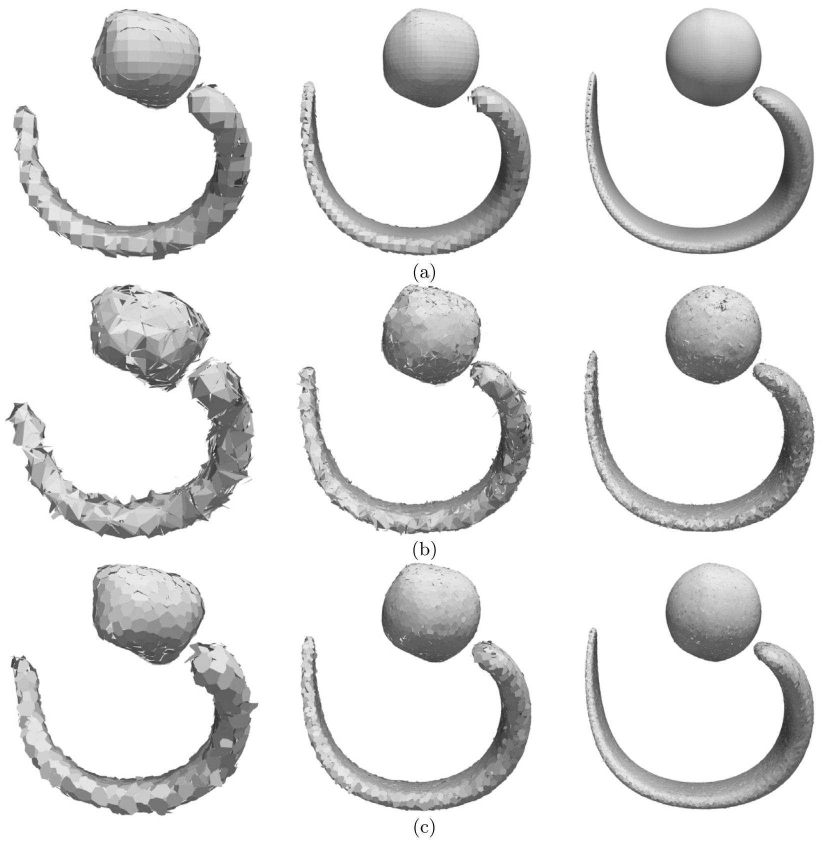


Fig. 26. PLIC interfaces at $t = 1.5$ and 3 for the VOF simulation using (a) cubic, (b) tetrahedral and (c) non-convex irregular polyhedral grids with (from the left to the right) $n \simeq 32^3$, 64^3 and 128^3 cells.

associated inconsistencies. It is also important to emphasize that this is achieved not only when the algorithm is applied to the extraction of isosurfaces on cubic cells but also over convex or non-convex arbitrary polyhedral cells.

2) The method not only allows the systematic extraction of isosurfaces in any type of grid, but can also be applied to generate “lookup tables” like those used in the marching cubes algorithm, but with two important advantages: a) the tables are consistent and b) they can be obtained not only for cubic cells but for any type of cells used in grids with predefined cell types. Having “lookup tables” previously generated by using the proposed algorithm would reduce the calculation time for the extraction of isosurfaces.

A new software that implements the proposed method has been released as open-source [24], allowing the results presented in the paper to be reproduced.

CRediT authorship contribution statement

Joaquín López: Conceptualization, Methodology, Software, Writing – original draft. **Adolfo Esteban:** Formal analysis, Resources, Validation. **Julio Hernández:** Conceptualization, Methodology, Writing – review & editing. **Pablo Gómez:** Software, Writing – original draft. **Rosendo Zamora:** Resources, Visualization. **Claudio Zanzi:** Investigation, Resources. **Félix Faura:** Project administration, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors gratefully acknowledge the support of the Spanish Ministerio de Ciencia, Innovación y Universidades - Agencia Estatal de Investigación and FEDER through projects DPI2017-87826-C2-1-P and DPI2017-87826-C2-2-P.

Appendix A. Content of file `domain.poly`

```
# Part 1 - node list
# Node count, 3 dim, no attribute, no boundary marker
8  3  0  0
# Node index, node coordinates
1  0.0 0.0 0.0
2  1.0 0.0 0.0
3  1.0 1.0 0.0
4  0.0 1.0 0.0
5  0.0 0.0 1.0
6  1.0 0.0 1.0
7  1.0 1.0 1.0
8  0.0 1.0 1.0
# Part 2 - facet list
# Facet count, no boundary marker
6  0
# Facets
1            # 1 polygon, no hole, no boundary marker
4  1 2 3 4   # front
1
4  5 6 7 8   # back
1
4  1 2 6 5   # bottom
 1
4  2 3 7 6   # right
1
4  3 4 8 7   # top
1
4  4 1 5 8   # left
# Part 3 - hole list
0            # no hole
# Part 4 - region list
0            # no region
```

Appendix B. Extended CLCIR PLIC reconstruction method

For each interfacial cell, the interface is represented by a plane as

$$\boldsymbol{n} \cdot \boldsymbol{x} + C = 0, \tag{B.1}$$

Algorithm 4 Extended CLCIR method.

```
1: Obtain $\phi$ at the grid vertices from Eq. (B.2)
2: for every grid cell do
3:     Maximum ($\phi_{\max}$) and minimum ($\phi_{\min}$) values of $\phi$ at the cell vertices
4:     if $\phi_{\max} > 0.5$ and $\phi_{\min} < 0.5$ then
5:         Call Algorithm 1 with $\widetilde{\phi} = 0.5$ to extract the isosurface
6:         if there is only one extracted isopolygon then
7:             Mark the grid cell as valid isosurface cell
8:             Construct the 'local triangulated surface'
9:             Compute $\boldsymbol{n}^l$ from Eq. (B.3)
10:        end if
11:    end if
12: end for
13: for every interfacial cell do
14:    if it is a valid isosurface cell then
15:        Construct the 'extended triangulated surface' $\mathcal{T}$
16:        Compute $\boldsymbol{n}^e$ from the equation equivalent to Eq. (B.3)
17:        if $\arccos(\boldsymbol{n}^e \cdot \boldsymbol{n}^l) < 1.2$ rad then
18:            $\boldsymbol{n} = \boldsymbol{n}^e$
19:        else
20:            $\boldsymbol{n} = \boldsymbol{n}^l$
21:        end if
22:    else
23:        Compute $\boldsymbol{n}$ from the LSGIR method
24:    end if
25:    Compute $C$ to locate the PLIC
26: end for
27: for every interfacial cell marked as valid isosurface cell do
28:    Update the vertices of $\mathcal{T}$ with the corresponding PLIC centers
29: end for
30: for every interfacial cell marked as valid isosurface cell do
31:    Compute $\boldsymbol{n}^c$ from the equation equivalent to Eq. (B.3) using the updated $\mathcal{T}$
32:    if $\arccos(\boldsymbol{n}^c \cdot \boldsymbol{n}) < 1.2$ rad then
33:        Update $\boldsymbol{n}$ with $\boldsymbol{n}^c$
34:        Compute $C$ to relocate the PLIC
35:    end if
36: end for
```

where the unit-length vector $\boldsymbol{n}$, normal to the interface and pointing to the fluid, is determined from the PLIC reconstruction method presented in the Algorithm 4 described below, which has been implemented with the aid of the proposed isosurface extraction procedure. The PLIC position is defined by the constant C, which is computed so that the interface splits cell Ω, of volume V_Ω, into two sub-cells of volumes $F(t^{\mathrm{n}})V_\Omega$ and $(1 - F(t^{\mathrm{n}}))V_\Omega$. In this work, the CIBRAVE (coupled interpolation-bracketed analytical volume enforcement) method of López et al. [46] is used to compute C, except when using grids with cubic cells, for which the efficient analytical method of Scardovelli and Zaleski [54] is used. The implementation of these two volume conservation enforcement methods is included in the `VOFTools` package [41,42,48].

The scalar field ϕ at each instant t^{n} and cell vertex i_p of the computational domain is obtained from

$$\phi_{i_p} = \frac{\sum\limits_{l} F_l(t^{\mathrm{n}}) w_l}{\sum\limits_{l} w_l}, \tag{B.2}$$

where the summations extend to all cells l containing the vertex i_p and $w_l = 1/|\boldsymbol{x}_{i_p} - \boldsymbol{x}_l|$, where $\boldsymbol{x}_{i_p}$ and $\boldsymbol{x}_l$ are the position vectors of the i_p vertex and geometric center of cell l, respectively (line 1 in Algorithm 4). For grid cells whose maximum and minimum interpolated ϕ values satisfy the condition (line 4) $\phi_{\min} < 0.5 < \phi_{\max}$, the isosurface corresponding to $\widetilde{\phi} = 0.5$ is extracted by calling Algorithm 1 (line 5). When the extracted isosurface consists of a single isopolygon (line 6), the grid cell is marked as a valid isosurface cell (line 7). A 'local triangulated surface' is then constructed around the geometric center of the extracted isosurface in such a way that each triangle is formed by this geometric center and two consecutive $\widetilde{\phi}$-vertices (line 8). The unit vector normal to the PLIC interface is obtained as

$$\boldsymbol{n}^l = \frac{\sum\limits_{t=1}^{N_t} w_t \boldsymbol{n}_t / \sum\limits_{t=1}^{N_t} w_t}{\left| \sum\limits_{t=1}^{N_t} w_t \boldsymbol{n}_t / \sum\limits_{t=1}^{N_t} w_t \right|}, \tag{B.3}$$

where the summation extends over the $N_t = $ `NIPVISO(1)` facets of the triangulated surface, $\boldsymbol{n}_t$ is the unit-length vector normal to the triangular facet t and w_t is a weighting factor (line 9). For cubic grids, w_t is defined as the ratio of the sine of the angle between the two inner edges of each triangular facet t and the product of their lengths [55] (an inner edge joins a $\widetilde{\phi}$-vertex with the geometric center of the extracted isosurface), and for the rest of grids, $w_t = 1$. To increase the accuracy of the PLIC reconstruction, the initial triangulated surface is substituted by an 'extended triangulated surface' obtained by connecting its geometric center to those of the isosurfaces extracted at adjacent cells (line 15) and the corresponding vector $\boldsymbol{n}^e$ is obtained from the equation equivalent to Eq. (B.3) (line 16). If $\arccos(\boldsymbol{n}^e \cdot \boldsymbol{n}^l) < 1.2$ rad, $\boldsymbol{n} = \boldsymbol{n}^e$ and otherwise $\boldsymbol{n} = \boldsymbol{n}^l$ (line 17-21). For interfacial cells that are not valid isosurface cells, situations that frequently occur in regions of low grid resolution, $\boldsymbol{n}$ is computed using a least-squares gradient interface reconstruction (LSGIR) method [56] (line 23), which can be considered as an extension of the Youngs' finite difference approximation for the gradient of the fluid volume fraction [57] to arbitrary grids [49,58]. Once $\boldsymbol{n}$ is obtained, the constant C is computed to locate the PLIC (line 25). Finally, the vertices of the extended triangulated surfaces are moved to the corresponding PLIC centers (lines 27-29). For every interfacial cell which is considered as a valid isosurface cell, $\boldsymbol{n}^c$ is computed from the equation equivalent to Eq. (B.3) using the updated extended triangulated surface (line 31). If $\arccos(\boldsymbol{n}^c \cdot \boldsymbol{n}) < 1.2$ rad, $\boldsymbol{n}$ is updated with $\boldsymbol{n}^c$ and the constant C is again computed to relocate the PLIC (lines 32 to 35).

Appendix C. Nomenclature

$C =$	PLIC interface constant in Eq. (B.1)
$E^{L_1}_{\text{shape}} =$	L_1 error norm to quantify interface shape accuracy in a VOF simulation
$E^{L_1}_{\widetilde{\phi}} =$	L_1 error norm to quantify the accuracy of isosurface extraction
$E^{L_\infty}_{\widetilde{\phi}} =$	L_∞ error norm to quantify the accuracy of isosurface extraction
$F =$	fluid volume fraction in a grid cell
$F^{\text{e}} =$	exact fluid volume fraction in a grid cell
$h =$	size of a cubic cell
$i =$	index of each vertex of a face boundary; index of a grid cell
$i' =$	edge index of intersected cell
$i_k =$	index of each $\widetilde{\phi}$-vertex of an isopolygon
$i_p =$	vertex index
$i_{\widetilde{\phi}} =$	$\widetilde{\phi}$-vertex index
$j =$	face boundary index
$k =$	isopolygon index
$l =$	index of a grid cell containing a given grid vertex i_p
$n =$	number of grid cells
$\boldsymbol{n} =$	unit vector normal to the PLIC interface
$\boldsymbol{n}^c =$	unit normal vector averaged on the updated surface $\mathcal{T}$
$\boldsymbol{n}^e =$	unit normal vector averaged on surface $\mathcal{T}$
$\boldsymbol{n}^l =$	unit normal vector averaged on the local triangulated surface
$N_t =$	number of facets of the triangulated surface
$\boldsymbol{n}_t =$	unit vector normal to the triangular facet t
$n_{\widetilde{\phi}} =$	number of grid cells where the isosurface is extracted
$N_{\widetilde{\phi}} =$	number of $\widetilde{\phi}$-vertices on the grid
$\mathcal{O} =$	order of convergence
$t =$	time; triangular facet index
$\mathcal{T} =$	extended triangulated surface
$\widetilde{t}_{\text{cpu}} =$	average execution time per time step in the VOF simulation
$t_{\text{end}} =$	time at the end of VOF simulation
$t_{\text{i}} =$	execution time consumed to identify the grid cells over which the isosurface is extracted
$t_{\widetilde{\phi}} =$	execution time consumed to extract the isosurface on the grid
$u, v, w =$	Cartesian components of $\boldsymbol{v}$
$\boldsymbol{v} =$	velocity vector
$V_\Omega =$	volume of the grid cell Ω
$w =$	weighting factor
$x, y, z =$	Cartesian coordinates
$\boldsymbol{x} =$	position vector of a grid point or of a generic point in the PLIC plane
$\boldsymbol{x}_{i_p} =$	position vector of vertex i_p
$\boldsymbol{x}_l =$	position vector of the geometric center of grid cell l
$\boldsymbol{x}_{\widetilde{\phi}} =$	position vector of $\widetilde{\phi}$-vertex

Greek characters

$\Delta t =$	time interval

$\phi =$	scalar field
$\widetilde{\phi} =$	isovalue of ϕ
$\phi_{i_p} =$	scalar value assigned to vertex i_p
$\phi_{\max} =$	maximum ϕ value at cell vertices
$\phi_{\min} =$	minimum ϕ value at cell vertices
$\Omega =$	polyhedron; grid cell
$\chi =$	indicator function

Arrays

$\mathtt{IA}(i_p) =$	tag value (1 or 0) assigned to vertex i_p
$\mathtt{IPV}(j, i) =$	polyhedron vertex with index i_p assigned to vertex i of face boundary j
$\mathtt{IPVINT}(j, i') =$	$\widetilde{\phi}$-vertex index $i_{\widetilde{\phi}}$ corresponding to the intersected edge i' of face boundary j
$\mathtt{IPVISO}(k, i_k) =$	$\widetilde{\phi}$-vertex index $i_{\widetilde{\phi}}$ corresponding to vertex i_k of isopolygon k
$\mathtt{NIPV}(j) =$	number of vertices of each polyhedron face boundary j
$\mathtt{NIPVINT}(j) =$	number of intersected edges of face boundary j
$\mathtt{NIPVISO}(k) =$	number of $\widetilde{\phi}$-vertices of isopolygon k

References

[1] T.J.R. Hughes, J.T. Oden, M. Papadrakakis (Eds.), Isogeometric analysis, Comput. Methods Appl. Mech. Eng. 284 (2015) 1–1182 (special issue).
[2] T.J.R. Hughes, J.T. Oden, M. Papadrakakis (Eds.), Isogeometric analysis: progress and challenges, Comput. Methods Appl. Mech. Eng. 316 (2017) 1–1270 (special issue).
[3] 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 58 (2008) 923–944.
[4] J. López, J. Hernández, Analytical and geometrical tools for 3D volume of fluid methods in general grids, J. Comput. Phys. 227 (2008) 5939–5948.
[5] J. Roenby, H. Bredmose, H. Jasak, A computational method for sharp interface advection, R. Soc. Open Sci. 3 (11) (2016) 160405.
[6] H. Scheufler, J. Roenby, Accurate and efficient surface reconstruction from volume fraction data on general meshes, J. Comput. Phys. 383 (2019) 1–23.
[7] 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.
[8] S. Shin, S.I. Abdel-Khalik, V. Daru, D. Juric, Accurate representation of surface tension using the level contour reconstruction method, J. Comput. Phys. 203 (2005) 493–516.
[9] S. Shin, Computation of the curvature field in numerical simulation of multiphase flow, J. Comput. Phys. 222 (2007) 872–878.
[10] S. Shin, I. Yoon, D. Juric, The Local Front Reconstruction Method for direct simulation of two- and three-dimensional multiphase flows, J. Comput. Phys. 230 (2011) 6605–6646.
[11] S. Shin, J. Chergui, D. Juric, L. Kahouadji, O.K. Matar, R.V. Craster, A hybrid interface tracking - level set technique for multiphase flow with soluble surfactant, J. Comput. Phys. 359 (2018) 409–435.
[12] W.E. Lorensen, H.E. Cline, Marching cubes: a high resolution 3D surface construction algorithm, Comput. Graph. 21 (1987) 163–168.
[13] R. Wenger, Isosurfaces: Geometry, Topology and Algorithms, A K Peters/CRC Press, Boca Raton, 2013.
[14] T.S. Newman, H. Yi, A survey of the marching cubes algorithm, Comput. Graph. 30 (2006) 854–879.
[15] A. Doi, A. Koide, An efficient method of triangulating equi-valued surfaces by using tetrahedral cells, IEICE Trans. Inf. Syst. E74-D (1991) 214–224.
[16] E. Chernyaev, Marching cubes 33: construction of topologically correct isosurfaces, CERN Report, CN/95-17, 1995.
[17] Y. Zhang, J. Qian, Dual contouring for domains with topology ambiguity, Comput. Methods Appl. Mech. Eng. 217–220 (2012) 34–45.
[18] Y. Zhang, J. Qian, Resolving topology ambiguity for multiple-material domains, Comput. Methods Appl. Mech. Eng. 247–248 (2012) 166–178.
[19] G.L. Masala, B. Golosio, P. Oliva, An improved marching cube algorithm for 3D data segmentation, Comput. Phys. Commun. 184 (2013) 777–782.
[20] J.L. Barrera, K. Maute, Ambiguous phase assignment of discretized 3D geometries in topology optimization, Comput. Methods Appl. Mech. Eng. 369 (2020) 113201.
[21] https://www.openfoam.com.
[22] https://mdx.plm.automation.siemens.com/star-ccm-plus.
[23] M. Ataei, M. Bussmann, V. Shaayegan, F. Costa, S. Han, C.B. Park, NPLIC: a machine learning approach to piecewise linear interface construction, arXiv:2007.04244, 2020.
[24] J. López, J. Hernández, isoap: A software for isosurface extraction on arbitrary polyhedra, Mendeley Data, V1, https://doi.org/10.17632/4rcf98s74c.1.
[25] https://www.gnu.org/licenses/gpl.html.
[26] J. López, J. Hernández, P. Gómez, F. Faura, Non-convex analytical and geometrical tools for volume truncation, initialization and conservation enforcement in VOF methods, J. Comput. Phys. 392 (2019) 666–693.
[27] S. Fuhrmann, M. Kazhdan, M. Goesele, Accurate isosurface interpolation with Hermite data, in: 2015 International Conference on 3D Vision, Lyon, 2015, pp. 256–263.
[28] A. Henderson, J. Ahrens, C. Law, The ParaView guide, 2004.
[29] J. Bloomenthal, Polygonization of implicit surfaces, Comput. Aided Geom. Des. 5 (1988) 341–355.
[30] B. Wyvill, D. Jevans, Table driven polygonization, in: SIGGRAPH Course Notes (Modeling and Animating with Implicit Surfaces), 1990, pp. 7.1–7.6.
[31] K.K. Delibasis, G.K. Matsopoulos, N.A. Mouravliansky, K.S. Nikita, A novel and efficient implementation of the marching cubes algorithm, Comput. Med. Imaging Graph. 25 (2001) 343–352.
[32] H. Si, TetGen, a Delaunay-based quality tetrahedral mesh generator, ACM Trans. Math. Softw. 41 (2015) 1–36.
[33] T. Etiene, C. Scheidegger, L.G. Nonato, R.M. Kriby, C.T. Silva, Verifiable visualization for isosurface extraction, IEEE Trans. Vis. Comput. Graph. 15 (2009) 1227–1234.
[34] Y. Livnat, H.W. Shen, C.R. Johnson, A near optimal isosurface extraction algorithm using the span space, IEEE Trans. Vis. Comput. Graph. 2 (1996) 73–84.
[35] S. Raman, R. Wenger, Quality isosurface mesh generation using an extended marching cubes lookup table, Comput. Graph. Forum 27 (2008) 791–798.
[36] C.A. Dietrich, C. Scheidegger, J. Schreiner, J.L.D. Comba, L.P. Nedel, C. Silva, Edge transformations for improving mesh quality of marching cubes, IEEE Trans. Vis. Comput. Graph. 15 (2008) 150–159.
[37] T. Ju, F. Losasso, S. Schaefer, J. Warren, Dual contouring of Hermite data, in: SIGGRAPH'02, ACM, 2002, pp. 339–346.

[38] J. Schreiner, C. Scheidegger, C. Silva, High-quality extraction of isosurfaces from regular and irregular grids, IEEE Trans. Vis. Comput. Graph. 12 (2006) 1205–1212.
[39] T.K. Deyand, J.A. Levine, Delaunay meshing of isosurfaces, in: SMI'07: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2007, IEEE Computer Society, 2007, pp. 241–250.
[40] Stanford University Computer Graphics Laboratory, Stanford Bunny, 1994, http://graphics.stanford.edu/data/3Dscanrep/.
[41] J. López, J. Hernández, P. Gómez, F. Faura, VOFTools - a software package of calculation tools for volume of fluid methods using general convex grids, Comput. Phys. Commun. 223 (2018) 45–54.
[42] J. López, J. Hernández, P. Gómez, C. Zanzi, R. Zamora, VOFTools 3.2: added VOF functionality to initialize the liquid volume fraction in general convex cells, Comput. Phys. Commun. 245 (2019) 106859.
[43] J. López, P. Gómez, J. Hernández, F. Faura, A two-grid adaptive volume of fluid approach for dendritic solidification, Comput. Fluids 86 (2013) 326–342.
[44] S.J. Cummins, M.M. Francois, D.B. Kothe, Estimating curvature from volume fractions, Comput. Struct. 83 (2005) 425–434.
[45] J. López, C. Zanzi, P. Gómez, R. Zamora, F. Faura, J. Hernández, An improved height function technique for computing interface curvature from volume fractions, Comput. Methods Appl. Mech. Eng. 198 (2009) 2555–2564.
[46] J. López, P. Gómez, C. Zanzi, J. Hernández, F. Faura, Application of non-convex analytic and geometric tools to a PLIC-VOF method, in: Proceedings of the ASME 2016 International Mechanical Engineering Congress and Exposition, November 11-17, Phoenix, Arizona, 2016, IMECE2016-67409.
[47] 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.
[48] J. López, J. Hernández, P. Gómez, C. Zanzi, R. Zamora, VOFTools 5: an extension to non-convex geometries of calculation tools for volume of fluid methods, Comput. Phys. Commun. 252 (2020) 107277.
[49] W.J. Rider, D.B. Kothe, Reconstructing volume tracking, J. Comput. Phys. 141 (1998) 112–152.
[50] 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.
[51] A. Baraldi, M.S. Dodd, A. Ferrante, A mass-conserving volume-of-fluid method: volume tracking and droplet surface-tension in incompressible isotropic turbulence, Comput. Fluids 96 (2014) 322–337.
[52] R. Comminal, J. Spangenberg, J.H. Hattel, Cellwise conservative unsplit advection for the volume of fluid method, J. Comput. Phys. 283 (2015) 582–608.
[53] L. Jofre, O. Lehmkuhl, J. Castro, A. Oliva, A 3-d volume-of-fluid advection method based on cell-vertex velocities for unstructured meshes, Comput. Fluids 94 (2014) 14–29.
[54] R. Scardovelli, S. Zaleski, Analytical relations connecting linear interfaces and volume fractions in rectangular grids, J. Comput. Phys. 164 (2000) 228–237.
[55] N. Max, Weights for computing vertex normals from facet normals, J. Graph. Tools 4 (2) (1999) 1–6.
[56] T.J. Barth, P.O. Frederickson, Higher-order solution of the Euler equations on unstructured grids using quadratic reconstruction, in: 28th AIAA Aerosp. Sci. Meeting, 1990.
[57] D.L. Youngs, An Interface Tracking Method for a 3D Eulerian Hydrodynamics Code, Technical Report 44/92/35, AWRE, 1984.
[58] D.B. Kothe, W.J. Rider, S.J. Mosso, J.S. Brock, J.I. Hochstein, Volume tracking of interfaces having surface tension in two and three dimensions, Technical Report AIAA 96-0859, AIAA, 1996 [Presented at the 34rd Aerospace Sciences Meeting and Exhibit].