Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

gVOF: An open-source package for unsplit geometric volume of fluid methods on arbitrary grids [☆],[☆☆]

Joaquín López [a],[*], Julio Hernández [b]

[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

A R T I C L E   I N F O

Article history:
Received 4 November 2021
Received in revised form 17 March 2022
Accepted 26 April 2022
Available online 4 May 2022

Keywords:
Volume of fluid method
Volume fraction initialization
Multidimensional unsplit advection
PLIC interface reconstruction
Arbitrary grids
Non-convex geometry

A B S T R A C T

The gVOF package implements several accurate and efficient geometric volume of fluid (VOF) methods on arbitrary grids, either structured or unstructured with convex or non-convex cells, based on multidimensional unsplit advection and piecewise linear interface calculation (PLIC) schemes, with the purpose of facilitating and extending the use of advanced unsplit geometric VOF methods in new or existing computational fluid dynamics codes. The package includes a complete and self-contained set of routines for VOF initialization, interface reconstruction and fluid advection, and uses as external libraries a set of publicly available in-house tools to perform several analytical and geometrical operations. These operations may involve handling of high-complex non-convex flux polyhedra, even with self-intersecting faces, which are robustly and efficiently treated in this work without the need of costly techniques based on convex decomposition. Results for the accuracy, computational efficiency, and volume (local and global) conservation properties of different combinations of the implemented advection and reconstruction methods are presented for several numerical tests on structured and unstructured grids. An extensive comparison with results obtained by other authors using advanced geometric VOF methods shows the outstanding performance of the gVOF package in terms of efficiency and accuracy. To demonstrate the performance of the package in solving complex two-phase flow problems, the implemented methods are combined with an existing in-house code to simulate the impact of a water drop on a free surface.

Program summary
Program Title: gVOF
CPC Library link to program files: https://doi.org/10.17632/9gmn6gsb6p.1
Licensing provisions: GPLv3
Programming language: FORTRAN and C, with C interfaces
External routines/libraries: VOFTools (https://doi.org/10.17632/brrgt645bh.3) and isoap (https://doi.org/10.17632/4rcf98s74c.1) libraries
Nature of problem: The software package includes efficient and accurate routines for volume of fluid initialization, reconstruction of interfaces and fluid advection on arbitrary grids, either structured or unstructured with convex or non-convex cells, which are used to implement advanced unsplit geometric VOF methods. In particular, the package includes a fluid volume fraction initialization method, six PLIC reconstruction methods and three multidimensional unsplit advection methods. Routines to visualize interfaces and compute reconstruction errors; tests to assess the accuracy, computational efficiency, and volume conservation of the implemented methods for the reconstruction and advection of complex interfaces on arbitrary grids; and a user manual have also been included in the supplied package.
Solution method: Basically, the implemented methods, which can be used on grids with polyhedral cells of arbitrary geometry, have the following general characteristics. The methods implemented for fluid volume fraction initialization and reconstruction error calculation are based on a recursive grid refinement procedure to compute the fluid volume bounded by cell edges and an implicitly-defined fluid interface. The six implemented interface reconstruction methods are the following: a least-squares gradient

[☆] The review of this paper was arranged by Prof. Hazel Andrew.
[☆☆] This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).
[*] Corresponding author.
E-mail address: joaquin.lopez@upct.es (J. López).

https://doi.org/10.1016/j.cpc.2022.108400

interface reconstruction method; three isosurface-based interface reconstruction methods; an extension to 3D of the iterative Swartz interface reconstruction; and a least-squares fit interface reconstruction method. Three unsplit advection methods, which are based on edge-matched, face-matched and non-matched flux polyhedra and are valid for 3D arbitrary grids, are implemented. All the implemented methods can be combined by the user to solve different tests on arbitrary grids. The implemented routines can be used in FORTRAN or C. The OpenMP application programming interface is used to improve the computational efficiency.

1. Introduction

There are many methods for interface capturing. Among them (see, for example, Reference [1] for a complete review), the volume of fluid (VOF) method is one of the most popular. This method uses an auxiliary function, referred to in this work as f, to implicitly capture the interface. Mass conservation is among its main advantages and the recent improvements in the computation of geometric characteristics such as interface curvature (e.g., [2–10]) or orientation (e.g., [9,11–17]) make this method even more competitive compared to others. The VOF methods could be classified in two main groups: (1) algebraic and (2) geometric methods. In algebraic type VOF methods, the auxiliary function f is usually represented algebraically by a polynomial or trigonometric function while in geometric type VOF methods, a discretized version of f is represented geometrically by the region delimited by the considered cell and the interface, which is generally defined by a line in two dimensions (2D) or a plane in three dimensions (3D). The geometric VOF methods, although they require a relatively high complex implementation, are more accurate in the computation of fluxes through cell faces and may be more computationally efficient since only the closest cells to the interface are involved in the computation (algebraic VOF methods involve larger grid cell stencils). However, probably due to the above geometric implementation complexity, algebraic VOF methods are still widely used in many commercial (for example, STAR-CCM+ or ANSYS Fluent) and non-commercial codes (for example, the OpenFOAM interface flow solver interFoam). A recent review of interface-capturing methods for two-phase flows [18] concludes that geometric VOF methods are among the most promising interface-capturing methods for future investment. An overview of different geometric VOF methods, mainly focused on unstructured grids and three dimensions, is recently presented in [19].

Geometric VOF methods basically consist of two steps: interface reconstruction and fluid advection. At each time interval, these two steps can be computed in only one stage using unsplit schemes or in multiple stages using split schemes that perform one dimensional advection for each spatial dimension (2 stages in 2D or 3 stages in 3D). The most advanced geometric VOF methods used today generally involve unsplit schemes with PLIC (piecewise-linear interface calculation) reconstruction, although competitive geometric VOF methods based on split schemes can also be found, e.g., in [14,20,21]. PLIC reconstruction is being used since the first implemented geometric VOF method [22] and prevails over the less accurate piecewise-constant SLIC (simple line interface calculation) reconstruction [23]. Unsplit schemes, although they require higher complex geometric operations than split schemes, especially in 3D, are computationally efficient because only one interface reconstruction per time interval is required and have the potential to be readily implemented on unstructured grids. This work is focused on these schemes.

Among others, Kothe et al. [24], Rider and Kothe [25], Harvie and Fletcher [26,27], López et al. [15,16] and Pilliod and Puckett [12], for 2D, and Rider and Kothe [25], Miller and Colella [28], Liovic et al. [13], Hernández et al. [29], López et al. [17], and more recently Owkes and Desjardins [30] or Marić et al. [31] for 3D, developed geometric-unsplit VOF methods, but only results on structured square or cubic grids were provided. Some of the first results on 2D unstructured grids can be found, for example, in the works by Mosso et al. [32,33]. Successful implementations of unsplit geometric VOF methods coupled with a level set method were also developed by Ningegowda and Premachandran [34] and Cao et al. [35] on 2D structured and unstructured grids, respectively. Due to the highly complex geometric operations involved in these methods, very scarce results can be found in the literature on 3D grids with non-cubic cells. Thanks to using the VOFTools routines, which have also been used by many of the authors referenced in this section, the first results on grids with non-cubic cells were obtained by López and Hernández [36]. Later, Ivey and Moin [37] and Jofre et al. [38] implemented methods that use, like in the edge-matched flux polygon advection (EMFPA) method proposed by López et al. [15] and following the suggestion made by Hernández et al. [29], the velocities in cell vertices to construct flux regions in cell faces on 3D unstructured grids, minimizing overlaps between them and reducing bounding errors in the fluid volume fraction distribution. Very recently, Ngo et al. [39] developed a similar geometric-unsplit VOF method coupled with a level set method and provided results on unstructured triangular and tetrahedral grids. Ivey and Moin [40] proposed the non-intersecting flux polyhedron advection (NIFPA) method that uses an iterative procedure to satisfy conservation and boundedness of the liquid volume fraction irrespective of the underlying flux polyhedron geometry on structured and unstructured grids. A recent advanced geometric VOF method based on isosurface constructions for general grids with arbitrary polyhedral cells was proposed by Roenby et al. [41] and improved later in [42] by using reconstructed distances to PLIC interfaces.

The main objective of this work is to implement a publicly available software which includes several accurate and efficient PLIC reconstruction and unsplit advection methods with the purpose to spread their use to simulate complex interface dynamics in arbitrary grids, with convex or non-convex cells. In particular, the following nine methods, described in Section 2, are implemented:

(i) Six PLIC-based reconstruction methods:
- an interface reconstruction method based on a least-squares gradient technique proposed by Barth and Frederickson [43] (Section 2.1.1),
- three new versions of the isosurface-based interface reconstruction methods proposed by López et al. [17] (Section 2.1.2),
- a 3D version of the iterative Swartz [44] interface reconstruction method (Section 2.1.3), and
- a 3D version of the least-squares fit interface reconstruction methods of Scardovelli and Zaleski [11] and Aulisa et al. [14].

(ii) Three unsplit advection methods:

- an extension to 3D arbitrary grids of the edge-matched flux polygon advection, also referred to hereafter as EMFPA (edge-matched flux polyhedron advection), method proposed by López et al. [15] (Section 2.2.1),
- a new version of the face-matched flux polyhedron advection (FMFPA) method proposed by Hernández et al. [29] (Section 2.2.2), and
- an extension to 3D arbitrary grids of the unsplit advection method proposed in [25], which will be referred to as non-matched flux polyhedron advection (NMFPA) method (Section 2.2.3).

Also, a fluid volume fraction initialization method based on the refinement procedure proposed in [7,45] and routines for interface visualization are implemented.

To the best of our knowledge, this is the first freely available package based on state-of-the-art unsplit geometric VOF solvers with a wide range of possibilities for efficient and accurate initialization of the fluid volume fraction and solution of complex interfacial dynamics on unstructured grids, with convex or non-convex cells, avoiding decomposition into convex sub-cells. The initialization routine allows obtaining the required degree of accuracy by using the appropriate refinement level. Depending on the accuracy, fluid volume conservation or computational efficiency required by the user, eighteen different geometric VOF combinations can be chosen between unsplit advection and PLIC reconstruction routines. Results that compare favorably with previous ones can be obtained efficiently with `gVOF` using any arbitrary grid, with second-order accuracy and excellent fluid volume conservation.

A brief description of the routines included in the `gVOF` package is presented in Section 3 and further details can be found in the user manual included in the supplied software. In Section 4, an analysis of accuracy, computational efficiency, and volume conservation properties of different combinations of the implemented advection and reconstruction methods is carried out for several numerical tests using structured and unstructured grids with convex and non-convex polyhedral cells. Also, an exhaustive comparison with results obtained with previous advanced geometric-unsplit VOF methods is included. The implemented package has been combined with an existing code developed by our group [7,16,29,46,47] to simulate complex phenomena involved in the impact of a water drop onto a free surface, and its results are presented in Section 5. Finally, the parallel performance of the `gVOF` package is assessed in Section 6. To avoid doing an excessively long work, the detailed analysis of the high number of parameters involved in the implemented algorithms will be published elsewhere.

2. Problem statement

The `gVOF` package solves the time-evolution equation

$$\frac{\partial f}{\partial t} + \nabla \cdot (\boldsymbol{u} f) - f \nabla \cdot \boldsymbol{u} = 0, \tag{1}$$

where the function f is equal to 1 in the fluid and 0 otherwise, and $\boldsymbol{u}$ is the velocity field. This equation is integrated over a time interval from t^n to t^{n+1} and a given cell, Ω, of volume V_Ω, to obtain, at each time step,

$$F^{n+1} = F^n - \frac{1}{V_\Omega} \int_{t^n}^{t^{n+1}} \int_{\Omega} \nabla \cdot (\boldsymbol{u} f)\, \mathrm{d}\Omega\, \mathrm{d}t + \frac{F^{n+1} + F^n}{2 V_\Omega} \int_{t^n}^{t^{n+1}} \int_{\Omega} \nabla \cdot \boldsymbol{u}\, \mathrm{d}\Omega\, \mathrm{d}t, \tag{2}$$

where F is a discretized version of the function f, whose value in each cell of the computational grid is the fraction of the cell occupied by the fluid. F will be denoted hereafter as fluid volume fraction. As suggested by Rider and Kothe [25], the application of the last term in Eq. (2), even for incompressible flows, can help to improve the local and global volume conservation. Note that this term must be null if the velocity vector field $\boldsymbol{u}$ is discretely solenoidal. The first integral in Eq. (2) represents the net volume of fluid advected out of the cell. This volume is computed geometrically using an unsplit advection method, for which the fluid interface must be previously reconstructed using a PLIC method. Both reconstruction and advection represent the most complex and time-consuming steps in any advanced geometric VOF code. Below, the methods implemented in the `gVOF` package for these two steps are briefly described.

2.1. Interface reconstruction

Based on the value of F, the grid cells are classified in two types:

- uniform, if $F < \epsilon$ or $F > 1 - \epsilon$, and
- interfacial, otherwise,

where ϵ is a small value close to zero (in the order of 10^{-10}). For all the uniform cells, the volume fraction is cut off before reconstructing as

$$F = \begin{cases} 1, & \text{if } F > 1 - \epsilon \\ 0, & \text{if } F < \epsilon. \end{cases} \tag{3}$$

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

$$\boldsymbol{n} \cdot \boldsymbol{x} + C = 0, \tag{4}$$

where the unit vector $\boldsymbol{n}$ normal to the interface and pointing into the fluid is firstly determined from any of the methods amenable to arbitrary grids described below, and then, the constant C is computed so that the interface splits the cell Ω of volume V_Ω into two sub-cells of volumes $F V_\Omega$ and $(1 - F) V_\Omega$. In this work, the CIBRAVE (coupled interpolation-bracketed analytical volume enforcement) method of López et al. [48] is generally used to compute C, except when using grids with rectangular parallelepiped cells, for which the efficient analytical method of Scardovelli and Zaleski [49] is used. The implementation of these two volume conservation enforcement methods is included in the `VOFTools` package [50–52].

This work implements into the `gVOF` package: a weighted least-squares gradient technique (Section 2.1.1), also known as Youngs' method, which shows very good performance in terms of efficiency and accuracy in regions where the grid resolution is not enough; three methods based on isosurface extractions (Section 2.1.2), which generally show superior accuracy in regions with enough grid resolution; and two interface reconstruction methods which can be applied iteratively (Sections 2.1.3 and 2.1.4).

2.1.1. Least-squares gradient interface reconstruction, LSGIR

The unit vector normal to the interface, $\boldsymbol{n}$, is obtained for every interfacial cell from the gradient of the fluid volume fraction distribution ∇F as

$$\boldsymbol{n} = \frac{\nabla F}{|\nabla F|} \tag{5}$$

using a weighted least-squares technique [43], amenable to any grid. This technique, which can be considered as an extension of the Youngs' finite difference approximations for ∇F [53] to arbitrary grids [24,25], has been implemented as follows.

A stencil involving neighbor cells of every interfacial cell is considered, in which each neighbor cell k, with geometric centers given by $\boldsymbol{x}_k \equiv (x_k, y_k, z_k)$, shares at least one vertex with the interfacial cell, with geometric center given by $\boldsymbol{x} \equiv (x, y, z)$. Then, the sum $\sum_{k=1}^{n} \left(\widetilde{F}_k - F_k\right)^2$, over all the n neighbor cells, of the quadratic differences between the Taylor series expanded $\widetilde{F}_k$ and F_k values, is minimized using a weighted least-squares fit that yields the following equation for the volume fraction gradient:

$$\nabla F = \left(A^T A\right)^{-1} A^T \boldsymbol{b}, \tag{6}$$

where

$$A = \begin{pmatrix} w_1(x_1 - x) & w_1(y_1 - y) & w_1(z_1 - z) \\ w_2(x_2 - x) & w_2(y_2 - y) & w_2(z_2 - z) \\ \vdots & \vdots & \vdots \\ w_n(x_n - x) & w_n(y_n - y) & w_n(z_n - z) \end{pmatrix}, \tag{7}$$

$$\boldsymbol{b} = \begin{pmatrix} w_1(F_1 - F) \\ w_2(F_2 - F) \\ \vdots \\ w_n(F_n - F) \end{pmatrix}, \tag{8}$$

and the weight

$$w_k = \frac{1}{|\boldsymbol{x} - \boldsymbol{x}_k|^{\beta}} \tag{9}$$

is used to soften the influence of remote stencil points on the results of the least squares approximation. A detailed analysis of the effect of the β parameter on the accuracy of the LSGIR method will be published elsewhere.

2.1.2. Isosurface-based interface reconstruction

The vector $\boldsymbol{n}$ is obtained with the aid of the isosurface extracted from the volume fraction distribution interpolated at cell vertices, F^*. For every interfacial cell whose maximum and minimum interpolated F^* values satisfy the condition

$$F^*_{min} < 0.5 < F^*_{max}, \tag{10}$$

the isosurface corresponding to the isovalue 0.5 is constructed using the procedure proposed by López et al. [54] (see the example of Fig. 1).

The interfacial cells that do not satisfy the condition of Eq. (10) or produce more than one isosurface, situations that frequently occur in regions of low grid resolutions, are reconstructed using the LSGIR method, which, as mentioned, performs well in such cases.

Below, three different variants of the interface reconstruction based on isosurface extraction will be briefly described. Considerations for the domain boundaries can be found in [17] and further details will be published elsewhere.

Local level contour-based interface reconstruction, LLCIR. The extracted isosurface, which generally is non-planar, is triangulated using its geometric center, which is obtained by simple averaging the position vectors of the isosurface vertices, as it is sketched in the example of Fig. 2, and the interface unit vector $\boldsymbol{n}$ in the considered interfacial cell is obtained from a weighted average of the unit vector normals to the isosurface triangles (see the example in Fig. 3). Different weighted parameters, based on angles, edge lengths or areas of the isosurface triangles, are considered in the implemented package and a detailed analysis of the performance of each one will be carried out elsewhere.

Extended level contour-based interface reconstruction, ELCIR. The vector $\boldsymbol{n}$ is obtained from an extended triangulated isosurface involving adjacent cells. The extension to adjacent cells is made by connecting the geometric centers of the set of vertices of the extracted isosurfaces, as sketched in the example of Fig. 4(a), resulting in a new triangulated surface (see the extended triangulated surface in the example of Fig. 4(b)) from which $\boldsymbol{n}$ is finally obtained by averaging as mentioned above (Fig. 3). If the extended triangulated surface yields an orientation that differs by more than 1.2 rad with respect to that provided by the LLCIR method, the vector $\boldsymbol{n}$ is obtained from the previous local triangulated isosurface.

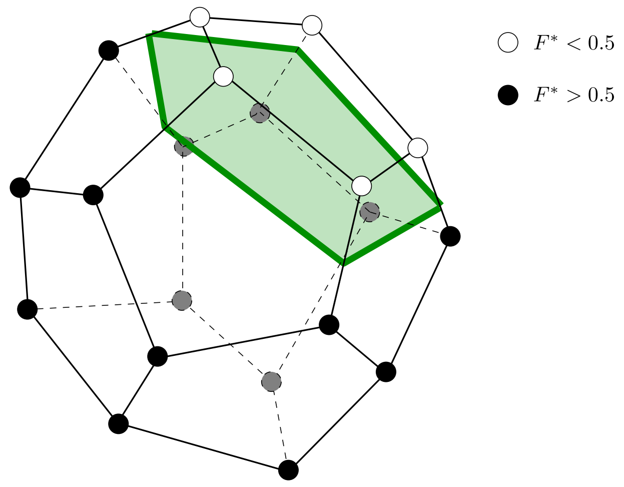


Fig. 1. Isosurface extracted on an irregular polyhedral cell from the volume fraction values interpolated at its vertices.

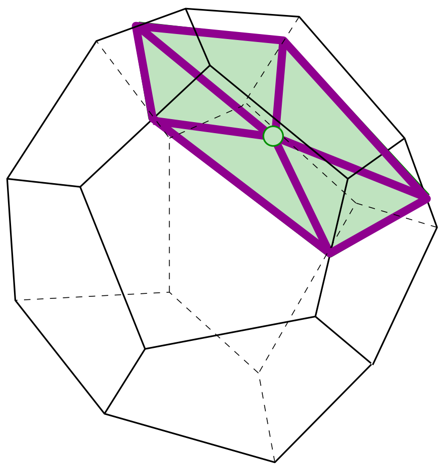


Fig. 2. Triangulation of the isosurface of Fig. 1 used in the LLCIR method.

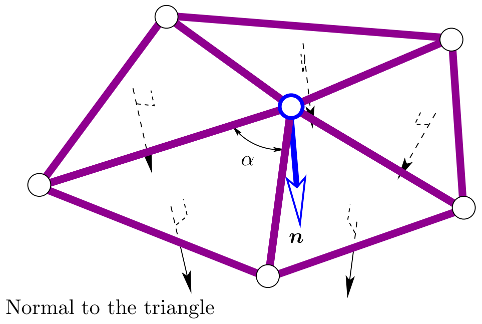
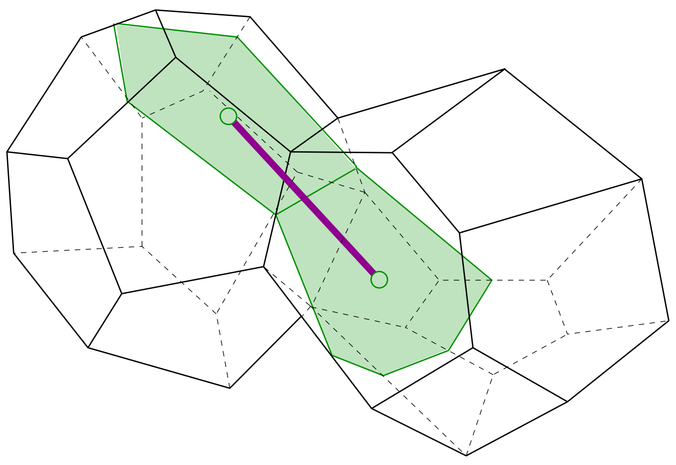


Fig. 3. Interface orientation vector, $\boldsymbol{n}$, obtained by averaging the unit vectors normal to the triangles. The angle α, along with other parameters of each triangle, are used for some of the weight factors included in the gVOF package that can be applied to obtain the average interface normal.

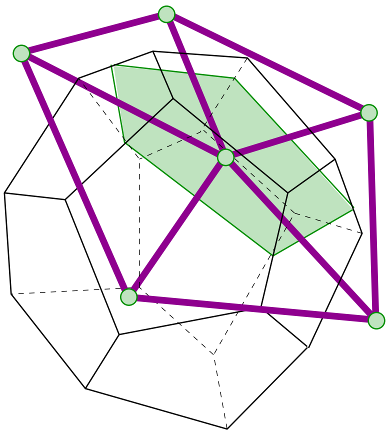
Conservative level contour-based interface reconstruction, CLCIR. After applying the ELCIR method and the volume conservation procedure on each interfacial cell to obtain the position of the PLIC interface (Fig. 5 shows the reconstructed PLIC interface that encloses the fluid volume corresponding to the value of F in the cell), the vertices of the extended triangulated surface are translated to the geometric centers of the set of vertices of the corresponding PLIC interfaces (see the example in Fig. 6(a)), resulting in a conservative extended triangulated surface (Fig. 6(b)) from which $\boldsymbol{n}$ is again obtained by averaging. A filter like that used in the above section for orientation differences higher than 1.2 rad is also applied here.

2.1.3. Swartz interface reconstruction, SWIR

The unit normal vector $\boldsymbol{n}$ is obtained using a procedure based on the iterative second-order interface reconstruction method of Swartz [44], which uses the fact that for a pair of certain neighbor interfacial cells there exists a “common orientation” that can be obtained from the fluid volume fractions in the two cells. Neighbor cells are those sharing at least one vertex. For a given interfacial



(a)



(b)

Fig. 4. Sketch of the ELCIR method. (a) Extension to an adjacent cell by connecting the geometric centers of the set of vertices of the extracted isosurfaces. (b) Triangulated surface extended to the geometric centers of the set of vertices of the isosurfaces extracted from adjacent cells.

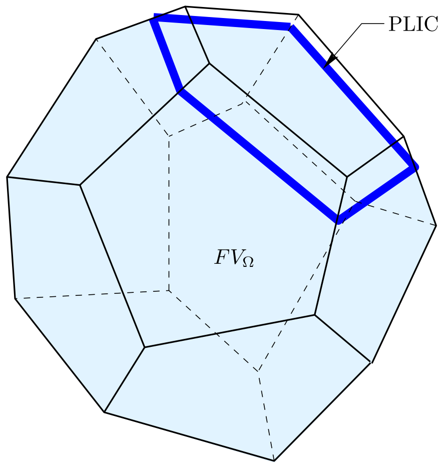


Fig. 5. PLIC interface that encloses the fluid volume FV_Ω in the cell.

cell, $\boldsymbol{n}$ is computed by averaging its common orientations with some of the neighbor interfacial cells, as described below. The method implemented in this work is similar, although with some considerations, to the variants described by Dyadechko and Shashkov [55] and Garimella et al. [56], which are somewhat different from the previous description given by Mosso et al. [32].

For a given interfacial cell, two types of iterations (inner and outer) are performed as follows. The inner iteration is applied over every valid neighbor interfacial cell until the common orientation reaches a prescribed tolerance, while the outer iteration involves all its valid neighbor interfacial cells as follows.

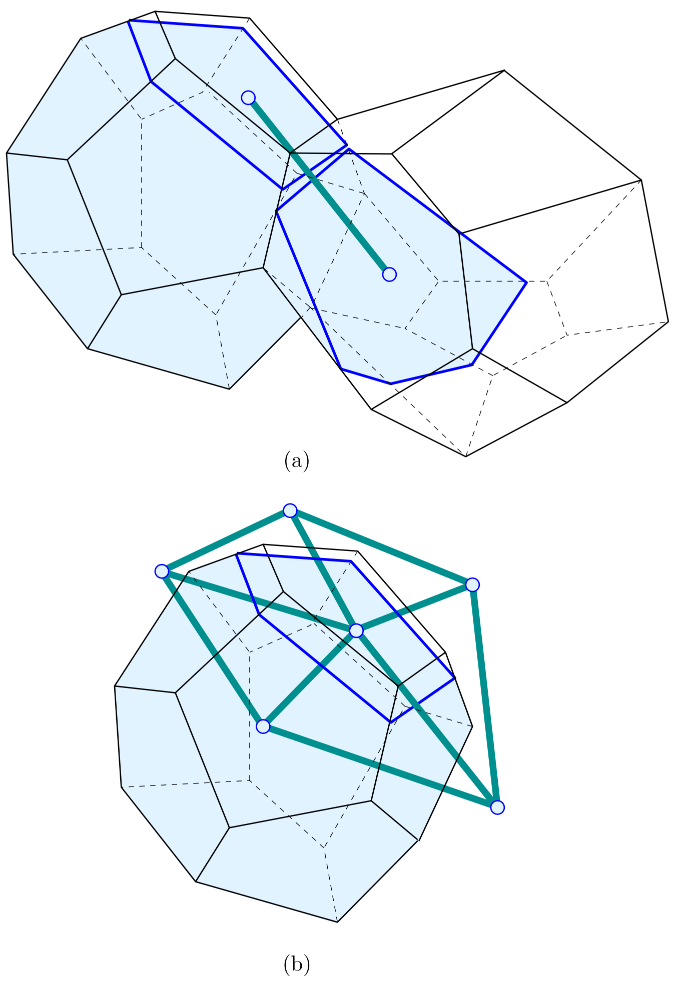


Fig. 6. Sketch of the CLCIR method. (a) Connection to an adjacent cell by joining the geometric centers of the set of vertices of the corresponding PLIC interfaces. (b) Resulting triangulated surface involving adjacent interfacial cells.

Inner iteration. A pair of neighbor interfacial cells are valid to perform the inner iteration if their interface orientations obtained in the previous outer iteration differ by less than 45°. Every valid pair must be iterated over as follows until the difference of the common orientation is below 10^{-6} rad.

1. Connect the geometric centers of the set of vertices of the paired PLIC interfaces.
2. Compute the perpendicular to the common segment joining the geometric centers.
3. Use this perpendicular to estimate the common orientation and locate the PLIC interfaces to conserve the associated fluid volume fractions.

Outer iteration. The interface orientation computed in the previous outer iteration is updated by the new orientation if it differs by less than $30°/I_{\text{out}}$, where I_{out} is the number of repetitions performed in the outer iteration, and the PLIC interface is located to match the corresponding fluid volume fraction. It should be mentioned that the update of an interface orientation is made only after the new orientation is computed over all the interfacial cells. The outer iteration is repeated until the angle between the previous $(I_{\text{out}} - 1)$ and current (I_{out}) computed interface orientation reaches a value lower than a prescribed tolerance or a maximum n_{out} of repetitions (the prescribed tolerance and the maximum number of repetitions of the outer iteration can be set by the `gVOF` user). The previous interface orientation values for the first outer iteration $(I_{\text{out}} = 1)$ are obtained using the LSGIR method from Section 2.1.1.

2.1.4. Least squares fit interface reconstruction, LSFIR

The unit normal vector $\boldsymbol{n}$ is obtained using a version applied to arbitrary polyhedral grids of the least-squares fit interface reconstruction methods presented by Scardovelli and Zaleski [11] and Aulisa et al. [14].

The interface is first reconstructed using the LSGIR method of Section 2.1.1 and the geometric center of the set of vertices of every reconstructed PLIC interface is computed. Using the centroid of the polygonal PLIC interface, like in [14], instead of the above geometric center provides similar results at the cost of a higher CPU time. The interface orientation of a given interfacial cell is updated from the orientation of the plane passing through its PLIC geometric center $\boldsymbol{x}$ that minimizes the distances δ_k to every PLIC geometric center $\boldsymbol{x}_k$ of neighbor interfacial cell k. The solution is obtained using a least-squares procedure that minimizes the functional H defined by

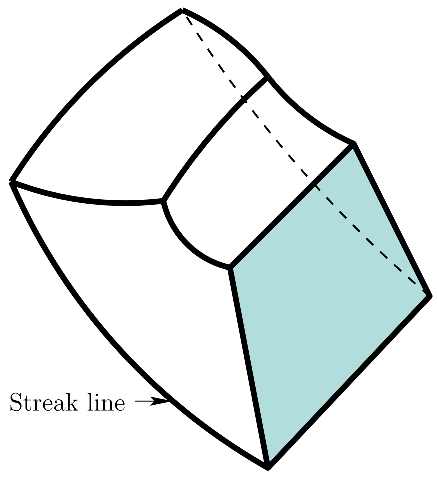


Fig. 7. Example of a flux region constructed on a cell face (shaded region).

$$H = \sum_{k}^{n_{\text{valid}}} w_k \delta_k^2, \tag{11}$$

where n_{valid} is the total number of valid neighbor interfacial cells and the weight $w_k = 1/|\boldsymbol{x}_k - \boldsymbol{x}|^{2.5}$ (the exponent 2.5 is roughly obtained by trial-and-error analysis). A neighbor interfacial cell is valid if it satisfies the same condition imposed for the inner iteration in the SWIR method. The LSFIR method can be applied using the outer iteration described in Section 2.1.3.

2.2. Fluid advected through cell faces

The following procedure is applied to determine the net volume of fluid advected out of the cell (first integral in Eq. (2)). On each polygonal cell face j of I vertices, the flux region of volume

$$V_{d_j} = \left(\boldsymbol{u}_j^{n+\frac{1}{2}} \cdot \boldsymbol{n}_j \right) A_j \Delta t, \tag{12}$$

where $\boldsymbol{n}_j$ is the unit vector normal to it pointing out of the cell, $\boldsymbol{u}_j^{n+\frac{1}{2}}$ is the velocity vector at its center and intermediate time $t^{n+\frac{1}{2}} = \frac{1}{2}\left(t^n + t^{n+1}\right)$, and A_j is its area, is delimited by $I+2$ flux region faces (see the example of Fig. 7 corresponding to a quadrilateral cell face (shaded region)), one of which is planar and coincides with the considered cell face (first flux region face) and the other $I+1$ flux region faces may be non-planar, self-intersected and have curved edges: I faces that have a common edge with the first flux region face (each one of these faces can be considered as the streak surfaces emanating from the corresponding common edge) and the last one is opposite to the first one. To make this complex flux region computationally affordable, its edges and faces will be approximated by straight lines and planar surfaces, respectively. Hereafter this discrete flux region will be denoted as flux polyhedron Ω_j^p and may be non-convex and even have self-intersecting faces.

Three different schemes, the edge and face matched flux polyhedron advection schemes described respectively in Sections 2.2.1 and 2.2.2, and the non-matched flux polyhedron advection scheme described in Section 2.2.3, are used to construct the flux polyhedron Ω_j^p. Hereafter, the unsplit advection procedure implemented in 3D in combination with the construction of edge-matched, face-matched, or non-matched flux polyhedra will be referred to as EMFPA, FMFPA or NMFPA method, respectively.

The volume of fluid V_{F_j} advected through the cell face j, which is taken to be positive when the fluid leaves the cell and negative otherwise, will depend on the shapes of Ω_j^p and the fluid regions determined by the reconstructed PLIC interfaces in cells intersected by Ω_j^p. Note that donating regions from several cells need to be considered. This is the most expensive task, in which recursive intersections between half-spaces and the generally non-convex Ω_j^p are required to compute V_{F_j} (the resulting truncated fluid regions are highlighted with red dashed lines in the 2D example of Fig. 8). The use of the non-convex tools proposed in [45] allows to perform these recursive intersections efficiently without the need to additionally decompose Ω_j^p, which will have a very positive impact on the computational efficiency of the whole advection scheme. The details of the procedure used to perform the recursive intersection operations will be published elsewhere.

Finally, the new volume of fluid fraction at t^{n+1} is obtained from Eq. (2) as

$$F^{n+1} = \left[F^n \left(1 + \frac{V_{d_T}}{2V_\Omega} \right) - \frac{V_{F_T}}{V_\Omega} \right] \left(1 - \frac{V_{d_T}}{2V_\Omega} \right)^{-1}, \tag{13}$$

where

$$V_{d_T} = \sum_j V_{d_j} \tag{14}$$

is the total net flux volume in the cell (second integral in Eq. (2)), and

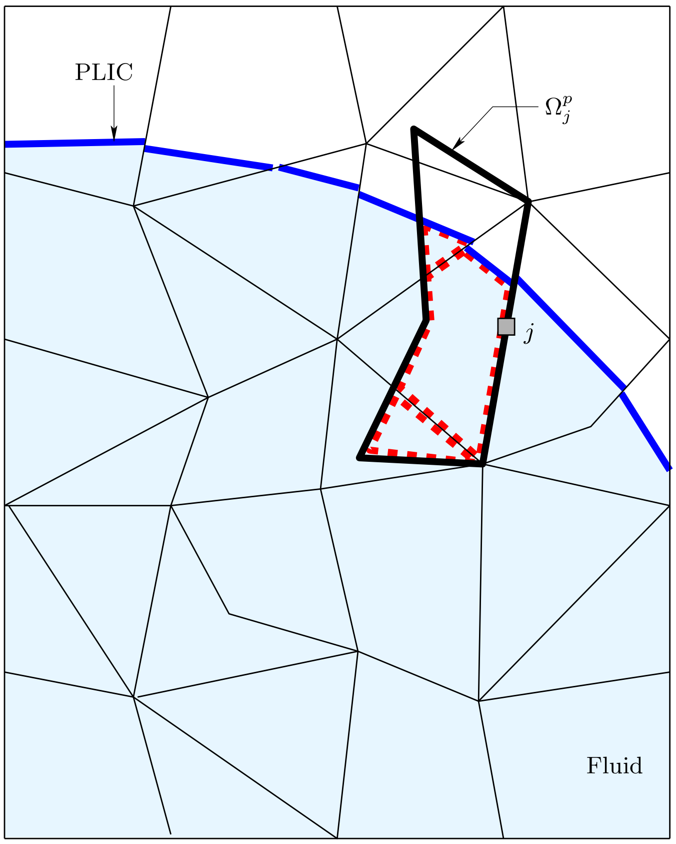


Fig. 8. 2D example for the computation of V_{F_j}. The truncated fluid regions are highlighted with red dashed lines. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

$$V_{F_T} = \sum_j V_{F_j} \tag{15}$$

is the total net volume of fluid that leaves (or enters) the cell (first integral in Eq. (2)), and the summations extend over all the faces of the considered grid cell.

The `gVOF` package does not use additional algorithms to redistribute the very small liquid volumes which are out of the limits given by the volume fraction. Instead, the updated F^{n+1} value is finally adjusted by simply making

$$F^{n+1} = \max\left[\min(F^{n+1}, 1.0), 0.0\right]. \tag{16}$$

2.2.1. Edge-matched flux polyhedron

The procedure used to construct an edge-matched flux polyhedron is analogous to that of the 2D version proposed by López et al. [15], in which the flux polygons on cell edges that have a vertex in common are constructed by having an edge with a common orientation, thus avoiding over/underlaps between flux polygons, provided that the time interval is low enough. In 3D, the proposed procedure also avoids over/underlaps between flux polyhedra on cell faces with a common vertex. The flux region corresponding to a given cell face of I vertices is delimited by a polyhedron Ω_j^p with $5I+1$ faces and $3I+1$ vertices as described below (see the example of Fig. 9).

The first face of Ω_j^p is matched to the considered cell face. It will be assumed that all the computational cells are composed by planar faces. Each vertex i (white circles in Fig. 9) of the first face is transported back in time along its streak line by solving

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \boldsymbol{u}. \tag{17}$$

The following approximation is used in this work to integrate Eq. (17) from $t = t^{n+1}$ to t^n with $\boldsymbol{x}(t) = \boldsymbol{x}_i$ at $t = t^{n+1}$

$$\boldsymbol{x}_{I+i} = \boldsymbol{x}_i - \Delta t \boldsymbol{u}_i^{n+\frac{1}{2}}, \tag{18}$$

where $\boldsymbol{x}_{I+i}$ is the position vector of the transported face vertex i at time t^n (gray circles in Fig. 9). All the results presented in this work were obtained with $\boldsymbol{u}_i^{n+\frac{1}{2}}$ defined at the intermediate time $t^{n+\frac{1}{2}} = \frac{1}{2}(t^n + t^{n+1})$.

Each edge of the first face and the approximated streak lines corresponding to the back-traced of its vertices delimit a flux face which is generally non planar. The next step is to replace this generally non-planar face by four triangles using its geometric center obtained by simple averaging of vertex positions (white square in Fig. 9), as shown in the sketch of Fig. 9.

The remaining flux region face is delimited by the I back-traced vertices, producing a generally non planar face which is also triangulated to obtain a surface of I triangles. Instead of directly using the geometric center of the I back-traced vertices in the triangulation,

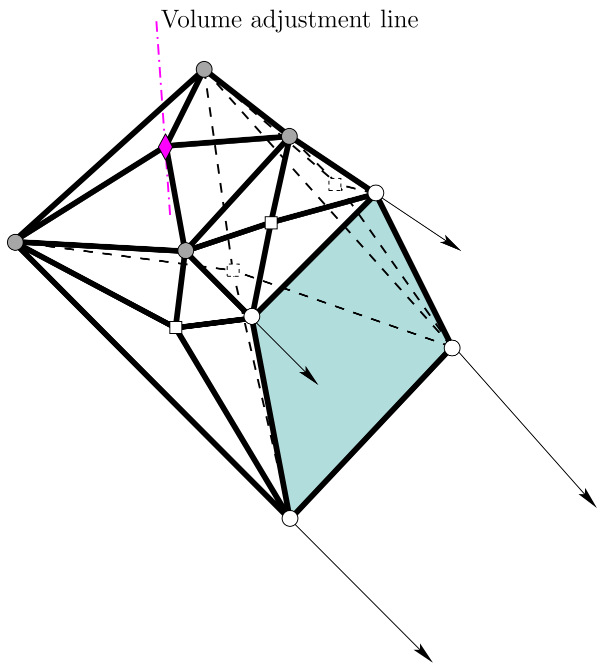


Fig. 9. Example of an edge-matched flux polyhedron construction. The arrows represent the velocity vectors on the cell face vertices.

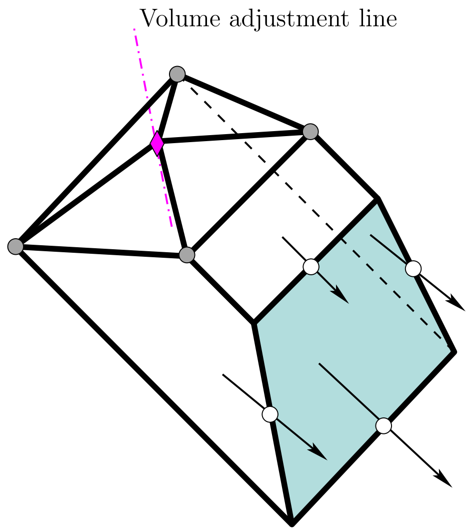


Fig. 10. Example of a face-matched flux polyhedron construction. The arrows represent the velocity vectors on the centers of the cell face edges.

which would produce a flux region without the required volume, this geometric center is translated in the direction normal to the face (volume adjustment line indicated in Fig. 9) so that the volume of the flux polyhedron coincides with V_{d_j} (the final point is depicted with a rhombus symbol in Fig. 9). This approach is like that used in [30].

2.2.2. *Face-matched flux polyhedron*

A face-matched flux polyhedron Ω_j^p is constructed using a new version of the procedure proposed in [29]. This new procedure produces flux polyhedra defined by $2I+1$ faces and vertices (see the example of Fig. 10), which obviously is computationally less expensive compared to the flux polyhedra produced by the EMFPA method (this can be clearly seen by comparing the examples in Figs. 9 and 10). Following the work in [29], the basic idea is to ensure that the flux polyhedra constructed on cell faces with a common edge have a face with a common orientation. The details of this face-matched flux polyhedron construction are given below.

As in Section 2.2.1, the first face of Ω_j^p coincides with the considered cell face. Each face of Ω_j^p that has a common edge with the first face is made parallel to the velocity vector at the center of the common edge (white circles in Fig. 10). The vertices of the remaining face of the flux region (gray circles in Fig. 10), which are initially obtained by back tracing along the lines of intersection of the planes that contain the faces contiguous to the cell face, determine a generally non-planar flux region face which is adjusted by triangulation using the same volume conservation enforcement mentioned in the above section. This approach is found to be more robust than that used by Hernández et al. [29] to impose the volume constraint of Ω_j^p, although also slightly more complex geometrically (in [29], Ω_j^p is defined by only $I+2$ faces and $2I$ vertices). As it was mentioned, the FMFPA method, either the original or the new version, avoids over/underlaps

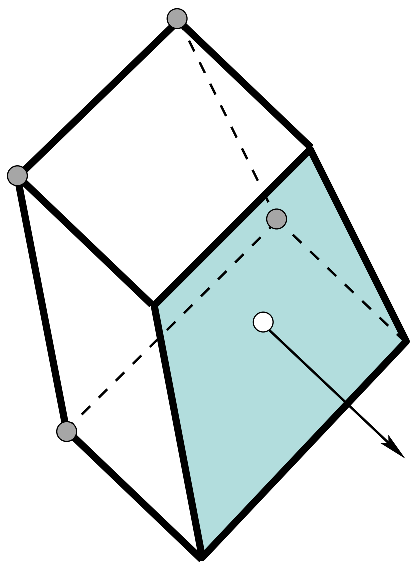


Fig. 11. Example of a non-matched flux polyhedron construction. The arrow represents the velocity vector at the face center.

between flux polyhedra constructed on cell faces with a common edge, although over/underlaps may still occur between flux polyhedra constructed on cell faces with only a common vertex.

2.2.3. Non-matched flux polyhedron

A non-matched flux polyhedron Ω_j^p is constructed using solely the velocity vector defined at the center of face j. The procedure described below, which could be considered as an extension to 3D arbitrary grids of the procedure described by Rider and Kothe [25], produces a flux polyhedron defined by $I+2$ faces and $2I$ vertices (see the example of Fig. 11), which further reduces the geometric complexity of the methods described in Sections 2.2.2 and 2.2.1. Liovic et al. [13] developed a similar procedure which was applied to cubic grids. Note that with this procedure the absence of over/underlaps between flux polyhedra constructed on cell faces with some vertex (just one or both) in common is not warranted.

As in Sections 2.2.1 and 2.2.2, the first face of Ω_j^p is made coincident with the considered cell face. Each vertex of the first face is transported back in time from an expression like that of Eq. (18), but using a unique velocity vector (the face center velocity vector) for all the other face vertices. Because of this, (1) each edge of the first face and the approximated streak lines corresponding to the back-traced of its vertices delimit a flux face which is planar, (2) the I back-traced vertices also delimit a planar flux region face, and (3) the resulting volume of the flux polyhedron must coincide with the volume V_{d_j}. Therefore, the triangulation of non-planar faces and the additional step used in the EMFPA and FMFPA methods to impose the volume constraint given by $V_{\Omega_j^p} = V_{d_j}$ are not required.

3. Description of the routines

The files included in the supplied software package are listed in Table 1. The `gVOF` software is implemented in `FORTRAN`. `OpenMP` application programming interface [61] is used to parallelize its execution on shared-memory architectures. To enable the software to be used with `C` codes, the declarations in `C` of the implemented routines are also included in the distributed software. The relevant arrays are pre-allocated for user-specified values of the parameters `ns`, `nv`, `nf`, `np` and `nn` (all these parameters are defined in the files `dim.h`, `dimpol.h` and `dimgrid.h` files for `FORTRAN` codes and `dimc.h`, `dimpolc.h` and `dimgridc.h` for `C` codes).

The main routines implemented to solve Eq. (1) are presented below (a brief description of each is summarized in Table 2). It must be mentioned that `gVOF` requires the `VOFTools` v5 [45,52] and `isoap` [54,57] external libraries. For the sake of brevity, the calling convection and the list of input and output parameters for each routine described hereafter, as well as others auxiliary routines not mentioned here, are left to be provided in the user manual included in the supplied software package.

3.1. Grid construction

The `defgrid` routine has been implemented to construct the computational grid. The grid is defined by a structure like that used by other codes, such as `OpenFOAM` [58]. The computational domain is divided into discrete non-overlapping polyhedral cells to form the grid. Each polyhedral cell is defined by a set of nodes and is bounded by planar polygonal faces. These faces may be internal or belong to a grid boundary. An internal face is shared by two cells, which are referred to in the example of Fig. 12 as owner and neighbor cells, while a face located in the domain boundary belongs to only one cell. The nodes of each face are arranged sequentially so that the vector joining two consecutive nodes leaves the face to the left (counterclockwise order) when viewed from outside the owner cell (see the node order illustrated in the example of Fig. 12).

The `neigbcell` routine has been implemented to obtain the list of cells that share at least one node with a given cell and the `compgrid` routine has been implemented to compute different geometric parameters related with the cells and faces of the computational grid, such as sizes, areas, volumes, geometric centers, or orientations, among others.

Table 1
Description of the gVOF package content.

File/directory Name	Description
`gvof.f`	source code of routines used to implement different advanced unsplit geometric VOF methods with PLIC reconstruction
`usergvof.f`	source code of user-defined routines and functions to be used in the tests
`cgvof.h`	declaration of the main routines included in the file `gvof.f` to be used in `C` codes
`cusergvof.h`	declaration of user-defined routines and functions included in the file `usergvof.f` to be used in `C` codes
`testrec.f`	`FORTRAN` test for interface reconstruction problems
`testrec.c`	`C` test for interface reconstruction problems
`testvof.f`	`FORTRAN` test for interfacial dynamics problems
`testvof.c`	`C` test for interfacial dynamics problems
`gvofvardef`	text file with input data for the tests
`Makefile.linux` and `Makefile.mac`	script examples for the set of tasks to construct the `gvof` library and make executable files for tests in `C` and `FORTRAN` on different platforms
`user-manual-gvof.pdf`	user manual in `pdf` format
`readme`	information about the contents of the supplied software package
`COPYING`	copy of the GNU General Public License, Version 3
`grids`	directory with non-uniform grid samples for tests (details about the grids included in the supplied software package are given in Section 4)

Table 2
Brief description of the main routines included in the gVOF package.

Name	Description
`defgrid`	constructs the grid
`neigbcell`	obtains the list of cells that share at least one node with a given cell
`compgrid`	computes different geometric parameters related with the cells and faces of the grid
`printgrid`	writes the geometry of the grid to a file in `VTK` format
`initfgrid`	initializes the fluid volume fraction in the grid cells
`taggrid`	interpolates the fluid volume fraction to the grid nodes and tags the nodes, faces and cells of the grid
`clcir`	reconstructs the PLIC interfaces using the isosurface-based CLCIR method
`elcir`	reconstructs the PLIC interfaces using the isosurface-based ELCIR method
`llcir`	reconstructs the PLIC interfaces using the isosurface-based LLCIR method
`lsgir`	reconstructs the PLIC interfaces using the gradient of the fluid volume fraction distribution
`swir`	reconstructs the PLIC interfaces using the iterative SWIR method
`lsfir`	reconstructs the PLIC interfaces using the LSFIR method
`recerr`	computes the interface reconstruction errors
`printplic`	writes the PLIC interfaces to a `file` in `VTK` format
`printvoxel`	remeshes the main grid onto a Cartesian grid (voxel grid) preserving the fluid volume and writes the computed fluid volume fraction at the voxel grid nodes to a file in `VTK` format
`faceflux`	computes the volume of fluid advected during a time interval through the grid faces
`emfp`	constructs a flux polyhedron using the EMFPA method
`fmfp`	constructs a flux polyhedron using the FMFPA method
`nmfp`	constructs a flux polyhedron using the NMFPA method
`vofadv`	advances the volume fraction distribution to the next time step

The `printgrid` routine has been implemented to write the geometry of the constructed grid to a file in `VTK` (visualization toolkit [59]) format which can be visualized by using, for example, the `ParaView` program [60]. As an example, Fig. 13 shows four different types of grids included in the supplied software with around 20^3 cells in a unit domain.

3.2. Fluid volume fraction initialization

The `initfgrid` routine has been implemented to initialize the fluid volume in the grid cells using an accurate refinement procedure [7,45,52], which allows to obtain the volume fraction of the fluid contained in a convex or non-convex polyhedral cell. The shape of the fluid interface is defined by implicit external functions. The degree of refinement (given by the number of divisions along each coordinate axis of the Cartesian cell superimposed to a given cell in which the fluid volume fraction is computed) can be increased by the users to improve the accuracy of the initialization. Detailed assessments of the accuracy of this initialization procedure can be found in [7,45,48].

3.3. Grid tagging

The nodes, faces and cells of the grid are tagged as described below to improve the computational efficiency of the implemented interface reconstruction and fluid advection algorithms (note that these algorithms only need to be applied in a region close to the interface). The `taggrid` routine first computes the volume of fluid fraction at every grid node (F^*) using an inverse distance weighting

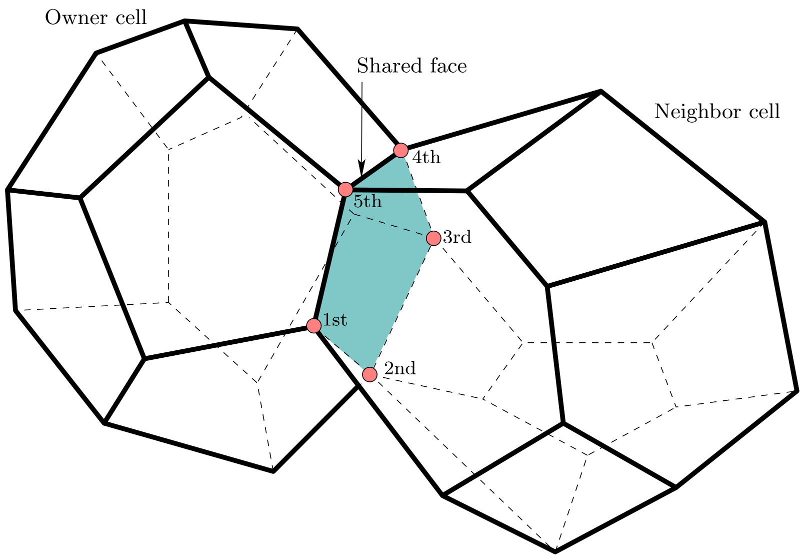


Fig. 12. Owner and neighbor cells that share a common face, with indication of the sequential order of nodes that define the shared face.

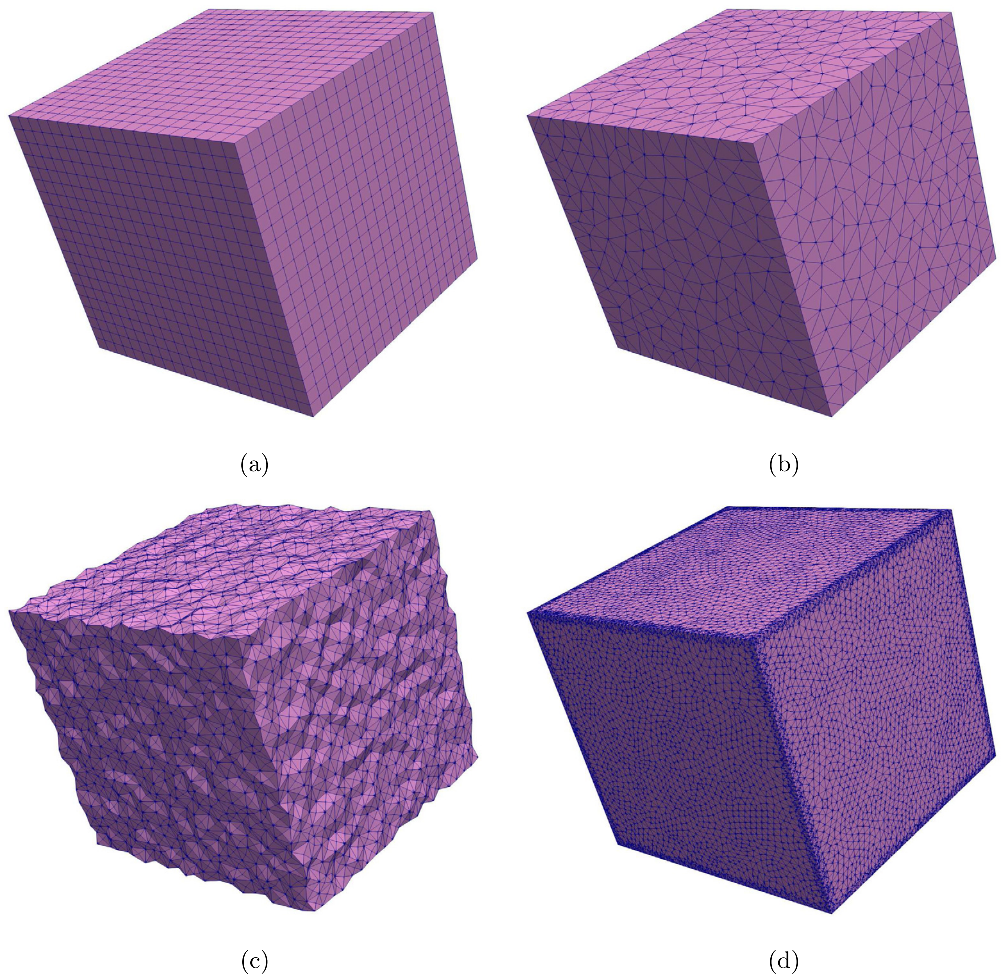


Fig. 13. Grids with around 20^3 cells in a unit domain (convex cells on the top and non-convex cells on the bottom): (a) uniform grid with rectangular parallelepipedal cells and non-uniform grids with (b) convex and non-convex cells ((c) distorted cubic and (d) irregular polyhedrical cells).

interpolation from the volume fractions in the cells that share it. To illustrate the tagging procedure used in `gVOF`, the 2D example of Fig. 14 is used for clarity, although the package works on 3D. As it will be seen in Section 4.2.4, the users can simulate 2D (see the example in Fig. 15(a)) or axisymmetric (Fig. 15(b)) problems by using 3D grids with only one cell along one of the coordinates directions (circumferential direction for axisymmetric cases). The tags are assigned to nodes, faces and cells as, respectively,

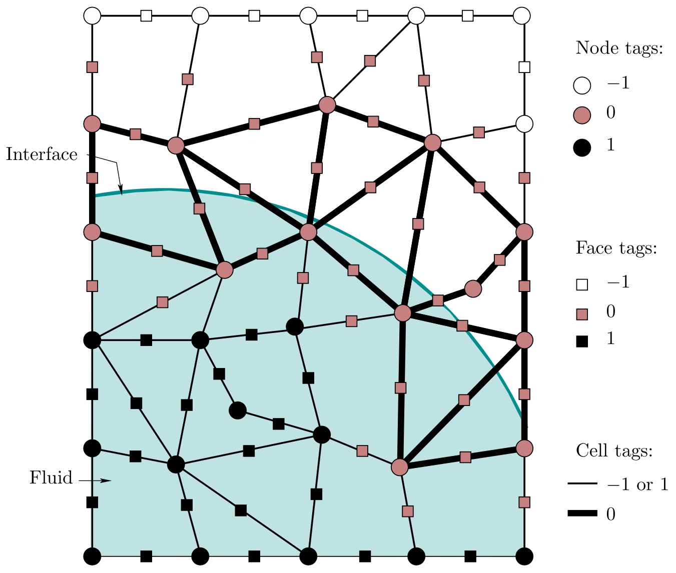


Fig. 14. 2D example of node (circle symbols), face (square symbols) and cell tagging.

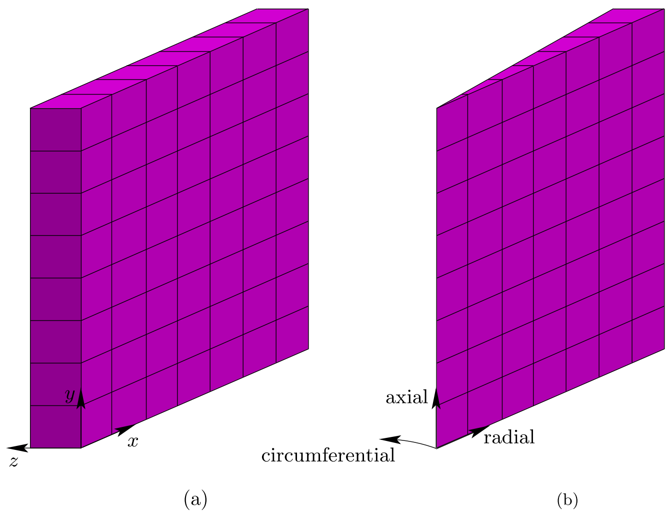


Fig. 15. Examples of 3D grids used to solve (a) 2D and (b) axisymmetric problems.

$$\text{Node tag} = \begin{cases} -1 & \text{if } F^* < \epsilon, \\ 1 & \text{if } F^* > (1-\epsilon) \text{ or} \\ 0 & \text{otherwise,} \end{cases}$$

$$\text{Face tag} = \begin{cases} -1 & \text{if all its nodes are tagged with } -1, \\ 1 & \text{if all its nodes are tagged with 1 or} \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\text{Cell tag} = \begin{cases} -1 & \text{if } F < \epsilon, \\ 1 & \text{if } F > (1-\epsilon) \text{ or} \\ 0 & \text{otherwise.} \end{cases}$$

Provided that the CFL number is limited by a maximum value equal to 1, the following considerations about the above face tagging can be made. If a face is tagged with −1, the volumetric flux of fluid must be zero, and therefore, no geometric operation will be carried out to compute the flux of fluid. Note that any flux polyhedron constructed on a face tagged with −1 (□ symbols in Fig. 14) will be far enough away from the fluid region. If a face is tagged with 1, the volumetric flux of fluid is approximately obtained in the `gVOF` package without geometrical operations from the fluid velocity defined at the considered face. Note that any flux polyhedron constructed on a face tagged with 1 (■ symbols in Fig. 14) will be completely inside the fluid region. Otherwise, a flux polyhedron must be constructed using the EMFPA, FMFPA or NMFPA method (Sections 2.2.1, 2.2.2 or 2.2.3, respectively) at the considered face and a series of recursive

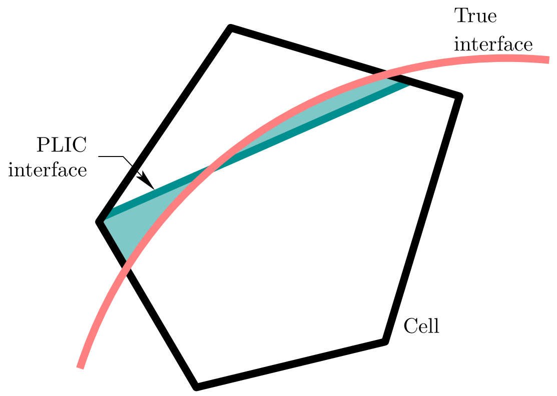


Fig. 16. 2D example with a sketch of the reconstruction error definition (shaded region).

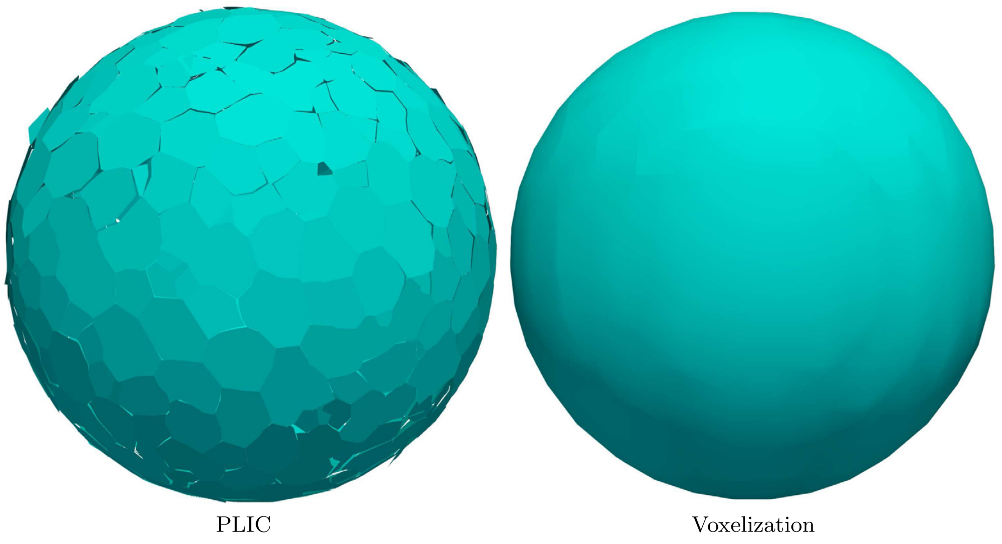


Fig. 17. Comparison between the spherical interface resulting from PLIC using the `printplic` routine (left picture) and voxelization using the `printvoxel` routine (right picture) in a non-convex irregular polyhedral grid.

truncation operations are needed to obtain the corresponding volumetric flux of fluid. Also note that only the cells tagged with 0 are interfacial (thick lines in Fig. 14) and any of the interface reconstruction routines described in the next section must be applied to all of them to obtain the corresponding PLIC interfaces.

3.4. Interface reconstruction

The `lsgir`, `llcir`, `elcir`, `clcir`, `swir` and `lsfir` routines have been implemented to reconstruct the PLIC interfaces using, respectively, the least-squares gradient interface reconstruction of Section 2.1.1, the three isosurface-based interface reconstruction of Section 2.1.2, the iterative Swartz interface reconstruction of Section 2.1.3 and the least squares fit interface reconstruction of Section 2.1.4.

The `recerr` routine has been implemented to compute the errors of the PLIC reconstruction of an interface given by an implicitly defined function. The reconstruction error is defined within a given cell as the volume enclosed by the true interface and the PLIC interface (shaded region in the 2D example in Fig. 16). To compute this volume, a procedure like that used in the `initfgrid` routine to compute the initial fluid volume in a cell is used. Additionally, this routine returns the fluid volume initialization error which is computed as the relative absolute difference between the exact volume of the fluid body and total initialized fluid volume.

To visualize the reconstructed interface, the following two routines have also been implemented. The `printplic` routine writes the reconstructed PLIC interface into a `VTK`-format file. Due to the discontinuities of the PLIC reconstructed in adjacent cells, this interface representation may be unpleasant for visualization purposes. To get a smoother visualization, the `printvoxel` routine remeshes the main grid onto a Cartesian grid (voxel grid) where a new volume fraction distribution is computed by truncation from the PLIC interface reconstructed on the main grid to exactly preserve the total fluid volume, and writes the computed fluid volume fraction at the voxel grid nodes to a `VTK`-format file, which can be visualized using programs like `ParaView` [60] (see the comparison in Fig. 17 between the spherical interface resulting from PLIC using the `printplic` routine (left picture) and from voxelization using the `printvoxel` routine (right picture) in a non-convex irregular polyhedral grid). The details and assessment of the voxelization procedure implemented in this routine will be published elsewhere. It must be mentioned that the `isovtkgrid` routine, which is part of the `isoap` library [54,57], is also used in this work to write isosurfaces extracted on arbitrary grids.

3.5. Fluid advection

The faceflux routine has been implemented to compute the volume of fluid advected during a time interval through the grid faces using the EMFPA (Section 2.2.1), FMFPA (Section 2.2.2) or NMFPA (Section 2.2.3) method by calling, respectively, the emfp, fmfp or nmfp subroutine also included in the gVOF package. This routine also returns the volumes of the flux polyhedra constructed at the grid faces.

The vofadv routine has been implemented to advance the volume fraction distribution to the next time step by solving Eq. (13) using the volumetric fluxes computed from the above routines.

4. Tests and performance analysis

Two test files are included in the supplied software package to assess the performance of the implemented algorithms to solve different static (Section 4.1) and dynamic (Section 4.2) problems with prescribed velocity fields using arbitrary grids. The following grids with convex and non-convex cells are considered (further details of the generation of grids used in this work can be found in [54] and some of these non-uniform grids are also included in the supplied software package):

- Uniform grids of rectangular parallelepiped cells.
- Unstructured grids with tetrahedral cells obtained using the program TetGen [62].
- Structured grids with non-convex cells obtained by distorting cubic cells. Each of the eight corner vertices of every initial uniform cubic cell of size h is randomly moved to the surface of a sphere with radius $0.25h$ and centered in the corresponding initial vertex position. The faces of the distorted cell, which are generally non-planar, are triangulated by joining its center with two consecutive vertices of each face, which results in a non-convex polyhedron of 14 vertices and 24 triangular faces.
- Unstructured grids with non-convex irregular polyhedral cells obtained with the aid of TetGen [62] and two OpenFOAM's programs [58] (tetgenToFoam and polyDualMesh). The faces of the cells generated in this way are generally non-planar and must be triangulated, as for the distorted cubic grids, by joining its center with two consecutive vertices of each face.
- Unstructured grids of different types available at [63] which were used to compare our results with those presented in [41].

The grid size $\mathcal{N}$ is obtained as

$$\mathcal{N} = \widetilde{N}_{\rm CELL}^{1/\mathcal{D}} \tag{19}$$

where $\widetilde{N}_{\rm CELL}$ is the number of grid cells in an equivalent unit domain (note that for the cases presented below with a domain different to a unit one, $\widetilde{N}_{\rm CELL}$ will not coincide with the true number of grid cells $N_{\rm CELL}$) and $\mathcal{D}$ is the dimensions number of the problem (2 and 3 for 2D and 3D problems, respectively). Note that in this work the 2D simulations are performed using 3D grids with only one cell along one of the coordinate directions.

In this and the following sections, CPU-times are estimated using the same resources as in Section 6 for the assessment of the parallel performance of the gVOF package, with a number of threads equal to twice the number of available cores to take advantage of the hyper-threading capability of the machine. For the LSGIR method of Section 2.1.1, a value $\beta = 1.5$ in the weight of Eq. (9) was used, which provides good results, especially for the cubic grids. For the isosurface-based interface reconstruction methods of Section 2.1.2, the weight factor introduced by Max [64] was used for cubic grids, the weight factor given by the angle α of each triangular face of the constructed isosurface (see the sketch of Fig. 3) if $\alpha \le \pi/2$ or $\pi - \alpha$ otherwise, was used for the tetrahedral and distorted cubic grids and the weight factor, also introduced by Max [64], given by the area of each triangular facet was used for the non-convex irregular polyhedral grids. For the SWIR method of Section 2.1.3 and the LSFIR method of Section 2.1.4, a tolerance of 10^{-3} rad and $n_{\rm out} = 4$ for the outer iteration are used (different values of the tolerance and maximum number of repetitions of the outer iteration can be set by the gVOF). A detailed analysis of the different weightings and iteration parameters available in the gVOF package will be published elsewhere. A volume fraction tolerance ϵ of 10^{-12} is used for all the results, except for the 2D deformation test of Section 4.2.4 and a particular case mentioned later, for which $\epsilon = 10^{-10}$ is used. Ten divisions along each coordinate direction are used to initialize the fluid volume fraction and compute the reconstruction error on each interfacial cell.

4.1. Static test cases

Algorithm 1 Interface reconstruction test.

```
1: Define the test case and allocate arrays for grid construction
2: Call defgrid and allocate arrays for reconstruction and assessment
3: Call printgrid
4: Call neigbcell
5: Call compgrid
6: Call initfgrid
7: Call taggrid
8: Call clcir, elcir, llcir, lsgir, swir or lsfir to reconstruct the interface
9: Call printplic, isovtkgrid or printvoxel to print the interface shape
10: Call recerr to obtain the reconstruction error $E^{L_1}_{\rm rec}$
11: Deallocate arrays
```

The test used to assess the performance of the different interface reconstruction methods implemented in the gVOF package is briefly presented in Algorithm 1. Two static interface reconstruction test cases are considered:

1. Sphere of radius 0.325 centered at (0.525, 0.464, 0.516).

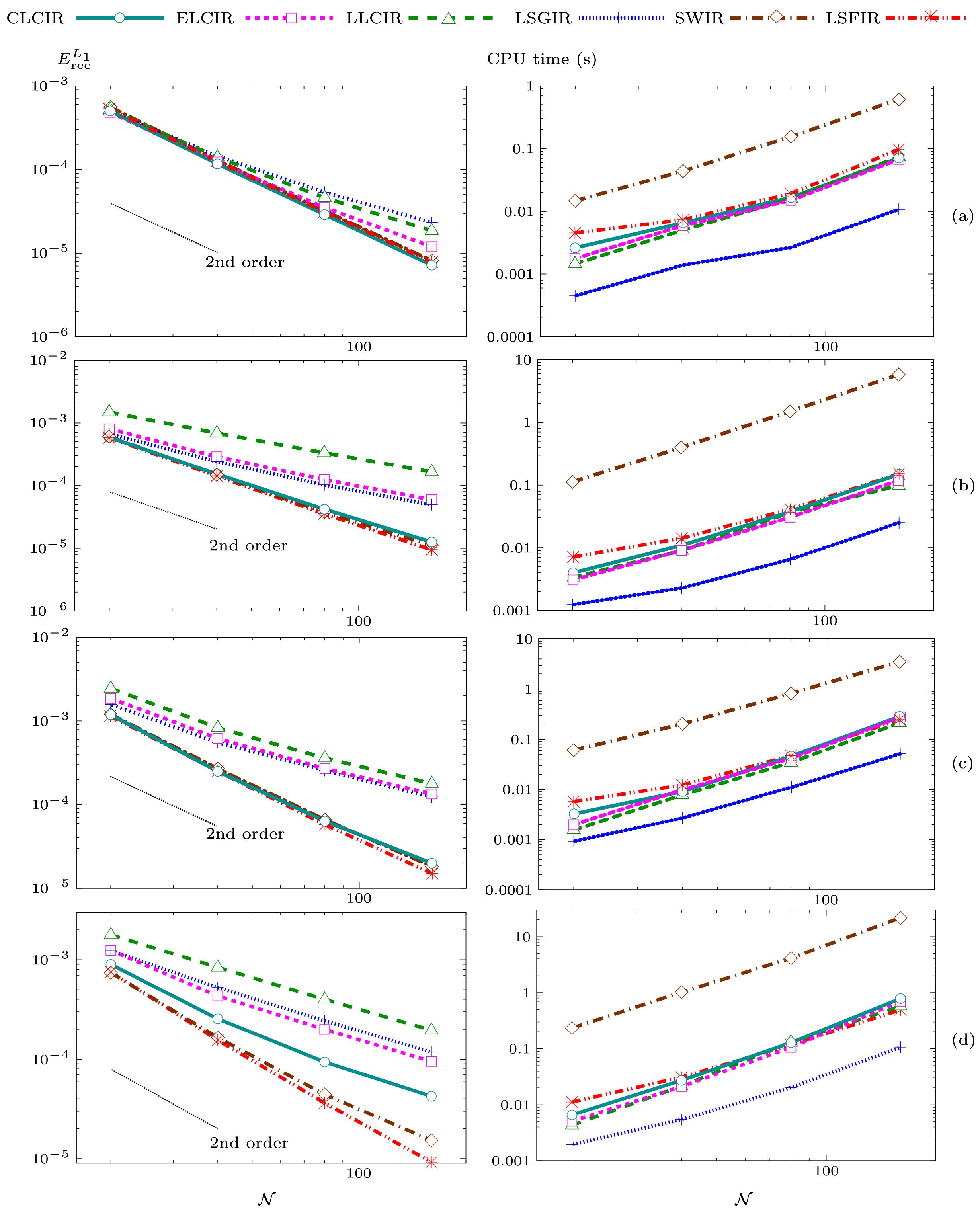


Fig. 18. Error $E_{\text{rec}}^{L_1}$ (left pictures) and CPU time (right pictures) for the reconstruction of the spherical fluid body using (a) cubic, (b) distorted cubic, (c) tetrahedral and (d) non-convex irregular polyhedral grids.

2. Torus centered at (0.525, 0.464, 0.516) with minor and major radius of 0.1 and 0.2.

Figs. 18 and 19 show the error $E_{\text{rec}}^{L_1}$ (left pictures), which is obtained by summing the error volumes computed in all the interfacial cells, and execution time (right pictures) for the reconstruction of the spherical and toroidal fluid bodies using different grids. It should be mentioned that changes in the relative differences in CPU time between methods could be observed depending on the architecture or compilation options used in the comparison. For the grids with cubic, non-convex distorted cubic and tetrahedral cells, the CLCIR, SWIR and LSFIR methods are the most accurate providing similar accuracy with second-order convergence. However, the iterative SWIR method is clearly less computationally efficient being, on average, 20 and 100 times more time consuming than the CLCIR and LSGIR methods,

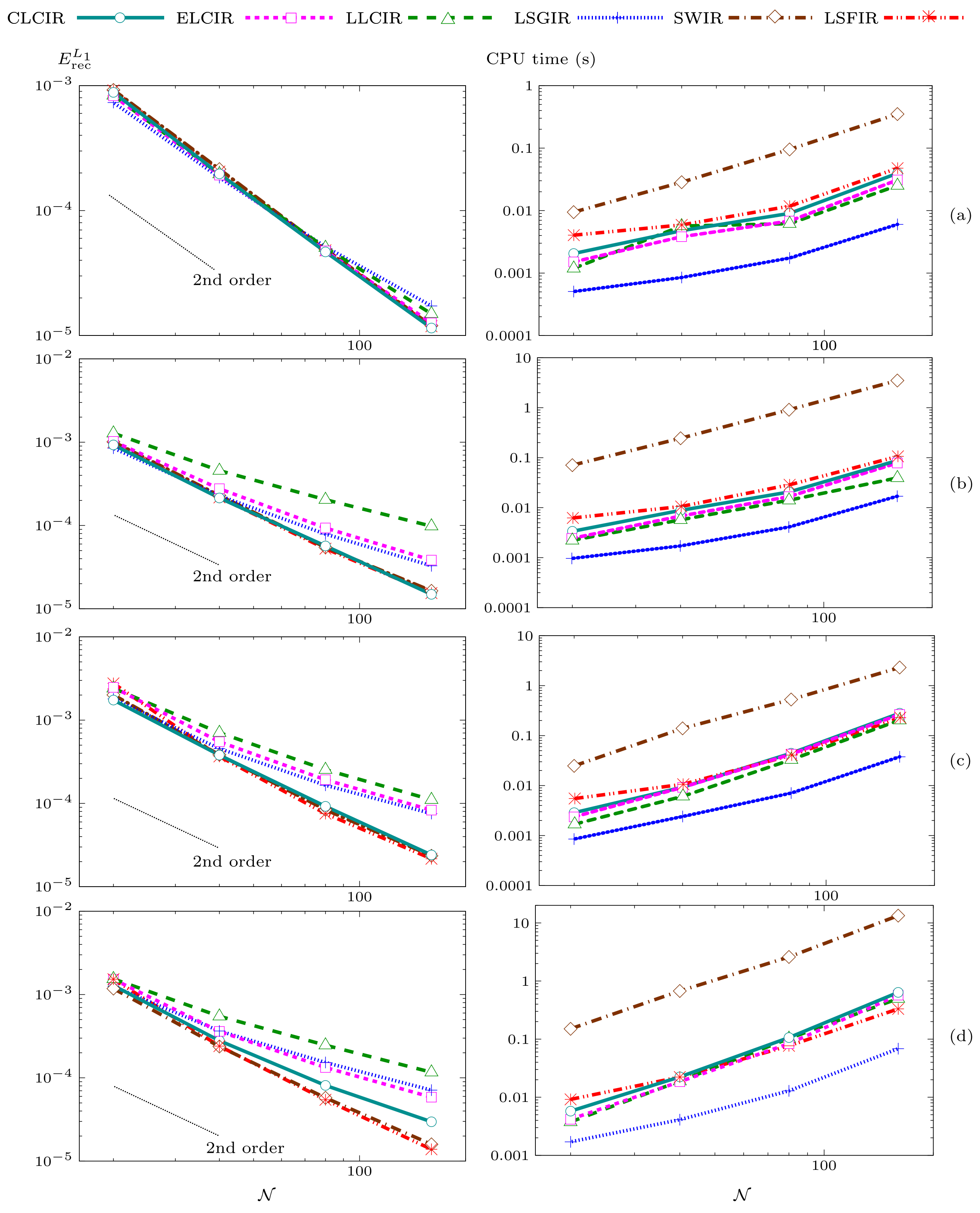


Fig. 19. Same results as in Fig. 18, but for the toroidal fluid body.

respectively. Note that the computational efficiency of the CLCIR and LSFIR methods is relatively similar. For the grids with non-convex irregular polyhedral cells, the LSFIR method is the most accurate exhibiting second-order convergence. The LSGIR method is the least time consuming, showing very good performance at low grid resolutions. As an example, Figs. 20 and 21 show the reconstructed PLIC interfaces for the spherical and toroidal fluid bodies using the CLCIR method and different grids with $\mathcal{N} = 40$.

4.2. Dynamic test cases with prescribed velocity field

Algorithm 2 presents the test used for the assessment of the coupling between the different interface reconstruction and advection methods implemented in the gVOF package to solve dynamic problems. Extensive comparisons with some results available in the literature

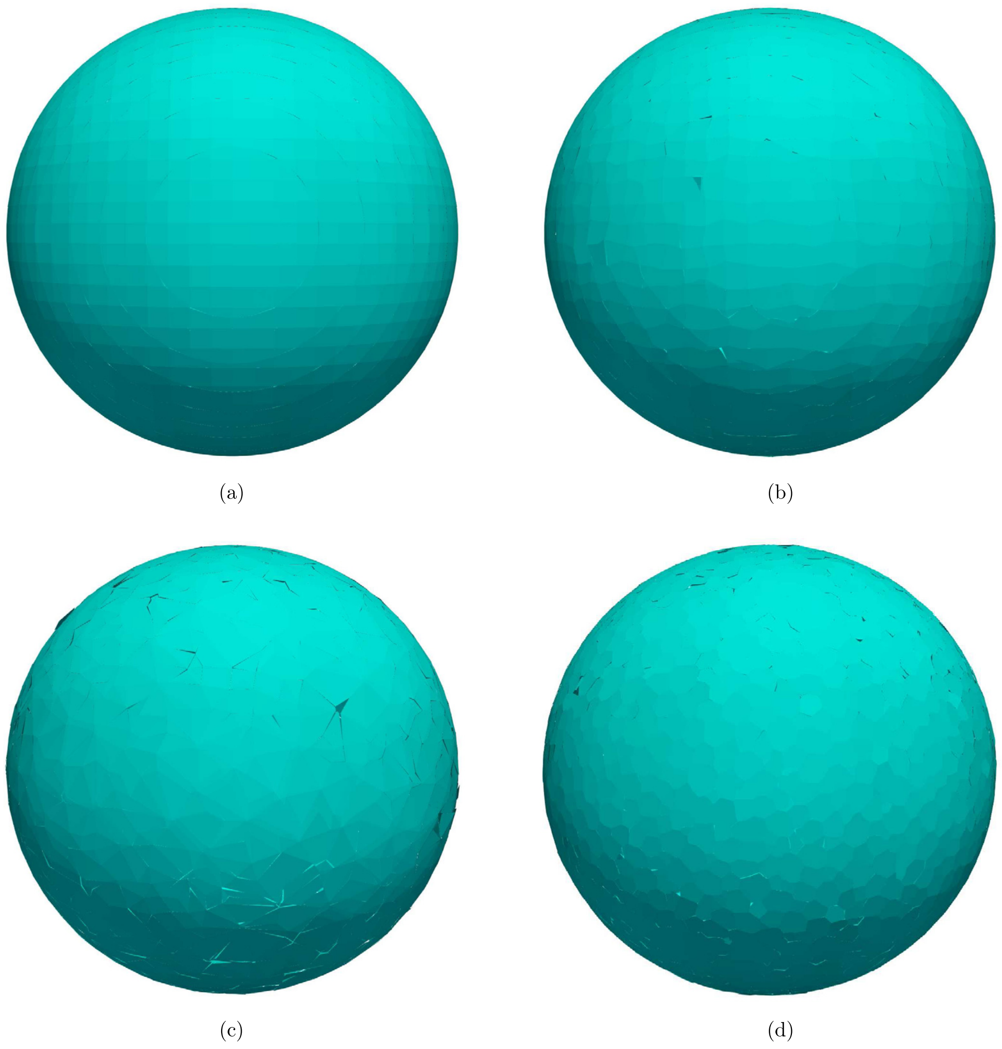


Fig. 20. PLIC interfaces for the spherical fluid body reconstructed using the CLCIR method and (a) cubic, (b) distorted cubic, (c) tetrahedral and (d) non-convex irregular polyhedral grids with $\mathcal{N} = 40$.

and obtained using other advanced geometric VOF methods are included in this section. Other results, which can also be found in the literature, are left out of this comparison for several reasons, among which, apart from brevity, are: the use of unstructured grids which are not provided and cannot be easily reproduced to be able to compare or the lack of information about parameters required for a rigorous comparison.

Algorithm 2 Fluid advection test.

```
 1: Define the test case and allocate arrays for grid construction
 2: Call defgrid and allocate arrays for reconstruction, advection, and assessment
 3: Call printgrid
 4: Call neigbcell
 5: Call compgrid
 6: Call initfgrid
 7: Call taggrid
 8: Call clcir, elcir, llcir, lsgir, swir or lsfir and print the interface
 9: $t_0 = 0$ and $t = 0$
10: Set the velocity field at the initial instant $t_0$
11: Set the initial time step $\Delta t$
12: while $t + \Delta t < t_{\text{end}}$ do
13:     $t_0 = t$ and $t = t_0 + \Delta t$
14:     Set the velocity field at the instant $\frac{1}{2}(t_0 + t)$
15:     Call faceflux
16:     Call vofadv
17:     Call taggrid
18:     Call clcir, elcir, llcir, lsgir, swir or lsfir and print, if required, the interface
19:     Set the time step $\Delta t$
20: end while
21: Errors computation
22: Deallocate arrays
```

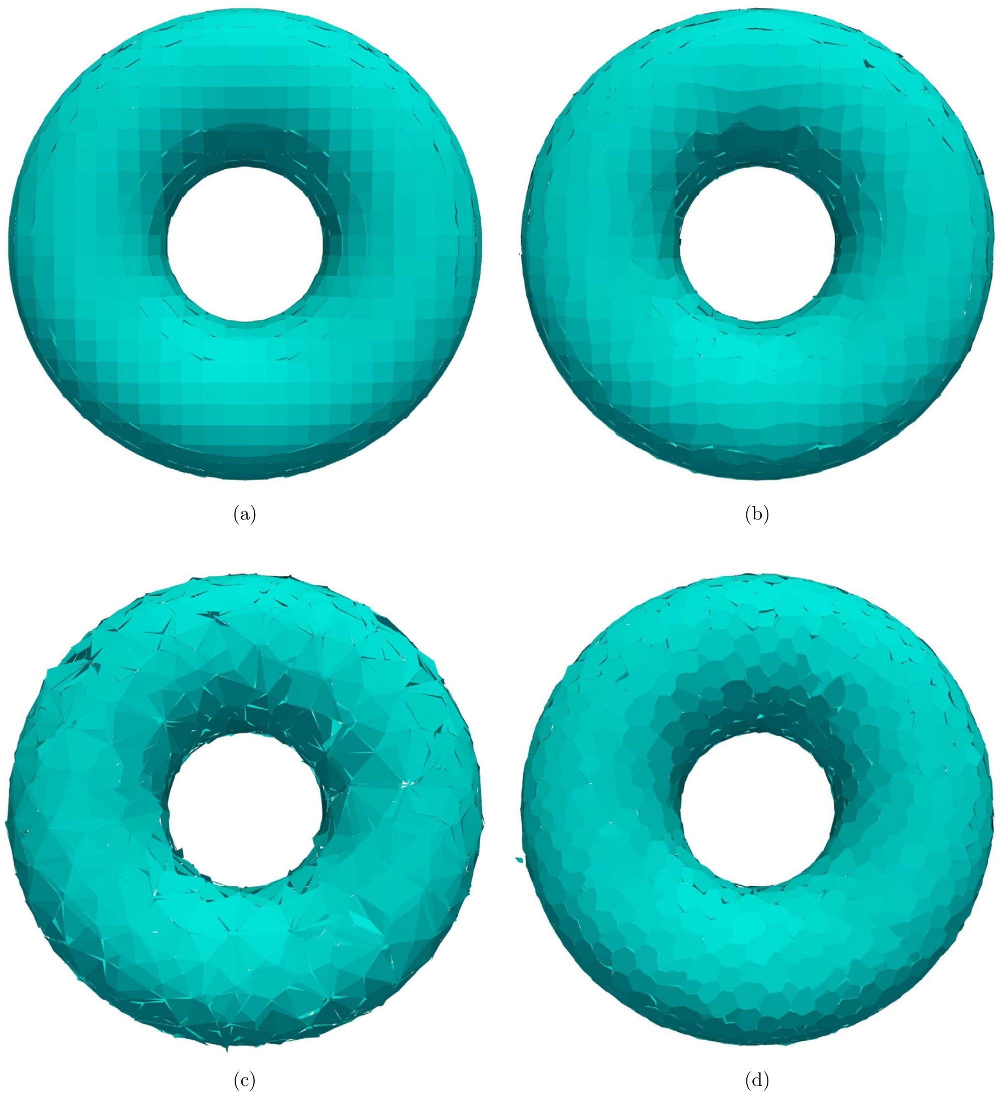


Fig. 21. Same results as in Fig. 20, but for the toroidal fluid body.

The time step Δt used to solve Eq. (1) is determined at each instant t from the following expression

$$\Delta t = \min\left[\frac{\min\limits_{i=1,\cdots,N_{\mathrm{CELL}}}(h_{x_i})}{\max\limits_{j=1,\cdots,N_{\mathrm{FACE}}}(|u_j|)}, \frac{\min\limits_{i=1,\cdots,N_{\mathrm{CELL}}}(h_{y_i})}{\max\limits_{j=1,\cdots,N_{\mathrm{FACE}}}(|v_j|)}, \frac{\min\limits_{i=1,\cdots,N_{\mathrm{CELL}}}(h_{z_i})}{\max\limits_{j=1,\cdots,N_{\mathrm{FACE}}}(|w_j|)}\right]\mathrm{CFL}, \tag{20}$$

where CFL is the Courant-Friedrich-Levy number, u_j, v_j and w_j are the components of the velocity vectors at the centers of every grid face j, N_{FACE} is the number of faces in the grid, and 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 every cell i. Other strategies can be found in the literature to fix the time step from the CFL number. For example, Roenby et al. [41], and Scheufler and Roenby [42], adjusted the time step by using a CFL number that only concerns cells near the interface, or Ivey and Moin [40] fixed the time step by choosing a sufficiently low CFL number to maintain the bounding volume errors close to the machine precision. Some authors are not very clear on this respect, therefore the comparisons should be seen with certain reservations given the great influence of the chosen time step on the final accuracy.

To quantify the accuracy of the interface shape, the following error norm, used for example by López et al. [15], López et al. [17] or Owkes and Desjardins [30], among others, is considered:

$$E^{L_1}_{\mathrm{shape}}(t) = \sum_{i=1}^{N_{\mathrm{CELL}}} V_{\Omega_i}\left|F^{\mathrm{e}}_i(t) - F_i(t)\right|, \tag{21}$$

where $F^{\mathrm{e}}_i(t)$ and $F_i(t)$ are, respectively, the exact and computed fluid volume fraction of cell i at instant t. Other authors, such as Roenby et al. [41], used the same error norm but relative to the exact total fluid volume:

Table 3
Results for the translation test using tetrahedral grids and CFL = 0.5. Comparison with results from [41]. The CPU times reported in [41] were obtained using a single core of an Intel Xeon 3.1 GHz CPU (E5-2687W).

Grid size, $\mathcal{N}$	$E^{L_1}_{\text{shape}^*}(t=4)$	$E^{L_1}_{\text{vol}}(t=4)$	$E^{L_1}_{\text{bound}}$	t_{cpu} (s)
gVOF (FMFPA, CLCIR)				
41	3.1×10^{-2}	5.7×10^{-10}	7.7×10^{-12}	22
71	1.3×10^{-2}	4.6×10^{-9}	1.1×10^{-11}	162
gVOF (FMFPA, SWIR)				
41	3.1×10^{-2}	3.7×10^{-10}	1.1×10^{-12}	104
71	1.3×10^{-2}	1.5×10^{-9}	2.4×10^{-12}	601
gVOF (FMFPA, LSFIR)				
41	3.9×10^{-2}	3.4×10^{-10}	1.0×10^{-11}	22
71	1.6×10^{-2}	2.5×10^{-9}	1.0×10^{-11}	145
isoAdvector [41]				
41	4.6×10^{-2}	5.1×10^{-13}	−−	157
71	2.1×10^{-2}	3.9×10^{-12}	−−	1411
MULES [41]				
71	4.2×10^{-1}	2.9×10^{-14}	−−	7306

$$E^{L_1}_{\text{shape}^*}(t)=\frac{\sum_{i=1}^{N_{\text{CELL}}}V_{\Omega_i}\left|F^{\text{e}}_i(t)-F_i(t)\right|}{\sum_{i=1}^{N_{\text{CELL}}}V_{\Omega_i}F^{\text{e}}_i(t)}. \tag{22}$$

To quantify the change in the total fluid volume, the following error norm is used

$$E^{L_1}_{\text{vol}}(t)=\left|\sum_{i=1}^{N_{\text{CELL}}}V_{\Omega_i}F^{\text{e}}_i(t)-\sum_{i=1}^{N_{\text{CELL}}}V_{\Omega_i}F_i(t)\right|. \tag{23}$$

To quantify the unboundedness of the volume of fluid fractions (values lower and higher than 0.0 and 1.0, respectively), the error norm used by Owkes and Desjardins [30] is also considered in this work

$$E^{L_\infty}_{\text{bound}}(t)=\max\left[-\min_{i=1,\cdots,N_{\text{CELL}}}V_{\Omega_i}F_i(t),\ \max_{i=1,\cdots,N_{\text{CELL}}}V_{\Omega_i}\left(F_i(t)-1\right)\right]. \tag{24}$$

The corresponding maximum and average values produced during the simulation are obtained, respectively, as

$$E^{L_\infty}_{\text{bound}}=\max_{i=1,\cdots,N_{\text{STEP}}}E^{L_\infty}_{\text{bound}}(t_i) \tag{25}$$

$$E^{L_1}_{\text{bound}}=\frac{\sum_{i=1}^{N_{\text{STEP}}}E^{L_\infty}_{\text{bound}}(t_i)}{N_{\text{STEP}}}, \tag{26}$$

where N_{STEP} is the number of time steps required to complete the numerical test. It must be emphasized that unlike other VOF methods referenced to compare, the gVOF package, as mentioned, does not use special algorithms to redistribute the very small liquid volumes out of the bounds 0 and 1 for the volume fraction F.

Results for the total execution time $t_{\text{cpu}}=t_{\text{rec}}+t_{\text{adv}}$, where t_{rec} and t_{adv} are the total CPU times consumed by the reconstruction and advection steps, respectively, and its average value per time step $\widetilde{t}_{\text{cpu}}$ are also presented below.

All the results presented in this section were obtained using the exact velocity field at both the face centers and cell vertices. It has been checked that when the velocities at the cell vertices are interpolated from the prescribed velocities at the face centers using a simple average, the accuracy values of the interface shape estimated using any of the error norms considered in this work are very close to those presented below. For the complex 3D deformation test of Section 4.2.3, the observed variations in the estimated accuracy by using prescribed or interpolated velocities at the cell vertices are in the order of only 1%, being negligible for the finest grids.

4.2.1. Translation test

This test is used to compare with the results obtained by Roenby et al. [41] using unstructured grids available in [63]. A sphere of fluid with radius 0.25, which is initially centered at (0.5, 0.5, 0.5) in a domain $[0,1]\times[0,1]\times[0,5]$, is translated by a velocity field given by (0, 0, 1) from time 0 to $t_{\text{end}}=4$. Table 3 compares the $E^{L_1}_{\text{shape}^*}(t=4)$ error values obtained using gVOF, combining the FMFPA method with the CLCIR, SWIR and LSFIR methods, and those obtained using two VOF methods: the algebraic VOF MULES (multidimensional universal limiter with explicit solution) [65] and the geometric VOF isoAdvector [41], both implemented in the OpenFOAM software [58]. All these results correspond to CFL = 0.5. Note that for this test, the final shape errors are reduced on average by 35% compared to the results obtained using the isoAdvector method and 97% compared to the MULES method, which represents a significant improvement in accuracy. The computational efficiency of the FMFPA and CLCIR methods, which produce errors like those obtained using the SWIR method but with

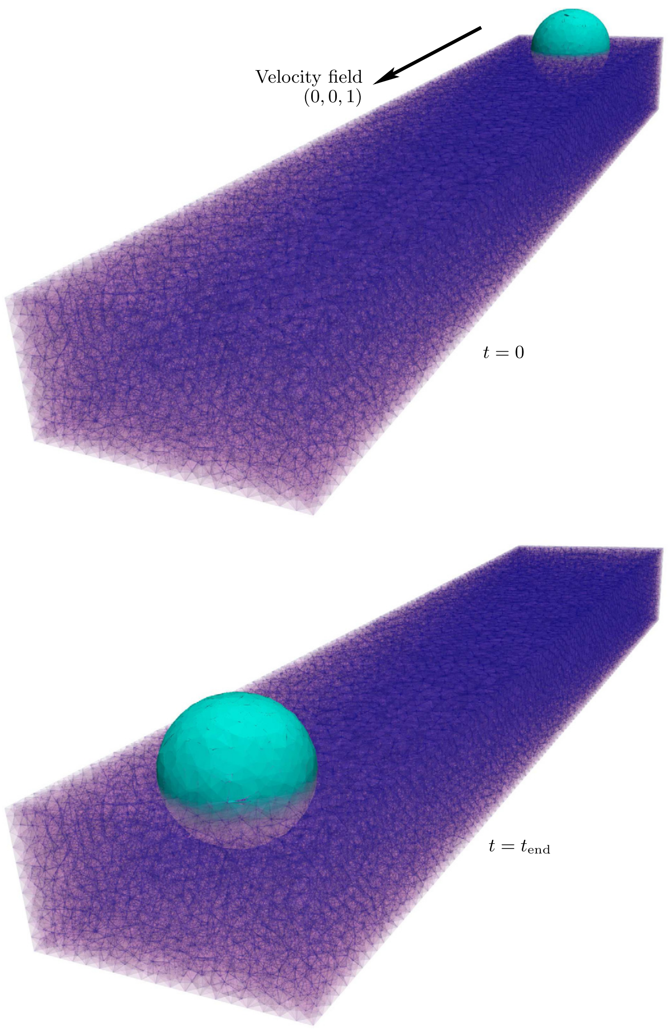


Fig. 22. Results for the translation case obtained using the FMFPA and CLCIR methods, the tetrahedral grid with $\mathcal{N} = 41$ and CFL $= 0.5$. The grid is clipped and made partially transparent to better see the PLIC interfaces reconstructed at $t = 0$ (top picture) and $t = t_{\text{end}}$ (bottom picture).

lower consumed CPU times, can also be observed in the results presented in the table. It must be considered that the execution times reported in [41] were obtained using a processor different to that used in this work and, therefore, those results should be viewed as a qualitative reference rather than for direct comparison. It must be mentioned that the EMFPA and NMFPA methods give almost identical results (not reported in the table) in terms of shape accuracy since the advection errors for this test are negligible using any of the implemented advection methods. For this test, however, the EMFPA and NMFPA methods consume CPU-times around a 20% higher and only 1% lower, respectively, than that consumed by the FMFPA method. The errors reported in Table 3 are due to the reconstruction step and to the roundoff errors of the geometric operations involved during the simulation. A more detailed analysis on this subject will be carried out elsewhere.

Fig. 22 shows the PLIC interface at $t = 0$ (top picture) and $t = t_{\text{end}}$ (bottom picture) obtained using the tetrahedral grid with $\mathcal{N} = 41$. The grid is clipped and made partially transparent to better see the reconstructed PLIC interfaces. A visual comparison with the isoAdvector and MULES methods is presented in Fig. 23, where results for the isosurfaces, corresponding to the 0.5 isovalue of the fluid volume fraction, extracted at the end of the translation test ($t = t_{\text{end}}$) can be seen. It can be observed that the extracted isosurfaces obtained using the FMFPA and CLCIR methods are better adjusted to the exact surfaces represented in transparent red color than those of the isoAdvector method (the differences with the exact solution can be clearly appreciated by observing colors change and specially the gap between the extracted isosurface and the exact sphere on the left of each picture) and considerably much better than those of the MULES method, which produces a highly distorted fluid body even for this simple translation test.

4.2.2. Rotation test

In this test, a sphere of fluid with radius 0.15, which is initially centered at (0.5,0.75,0.5), is rotated to complete a revolution at $t_{\text{end}} = 2\pi$ around an axis parallel to the z-axis and centered in a domain $[0, 1] \times [0, 1] \times [0, 1]$. The results obtained with `gVOF` (the FMFPA method combined with the CLCIR, SWIR and LSFIR methods and the NMFPA method combined with the CLCIR method) are compared in Table 4 with those obtained by Jofre et al. [38], using an advection method similar to the EMFPA method and the second-order LVIRA [66] reconstruction method, and Scheufler and Roenby [42], using a new version of the isoAdvector method (isoAdvector-plicRDF) which is improved by a distance function reconstructed from PLIC interfaces. All these results were obtained using CFL $= 1$. As in the above test, the EMFPA method produces an accuracy almost identical to that of FMFPA method but with higher time consumptions. Note that the final

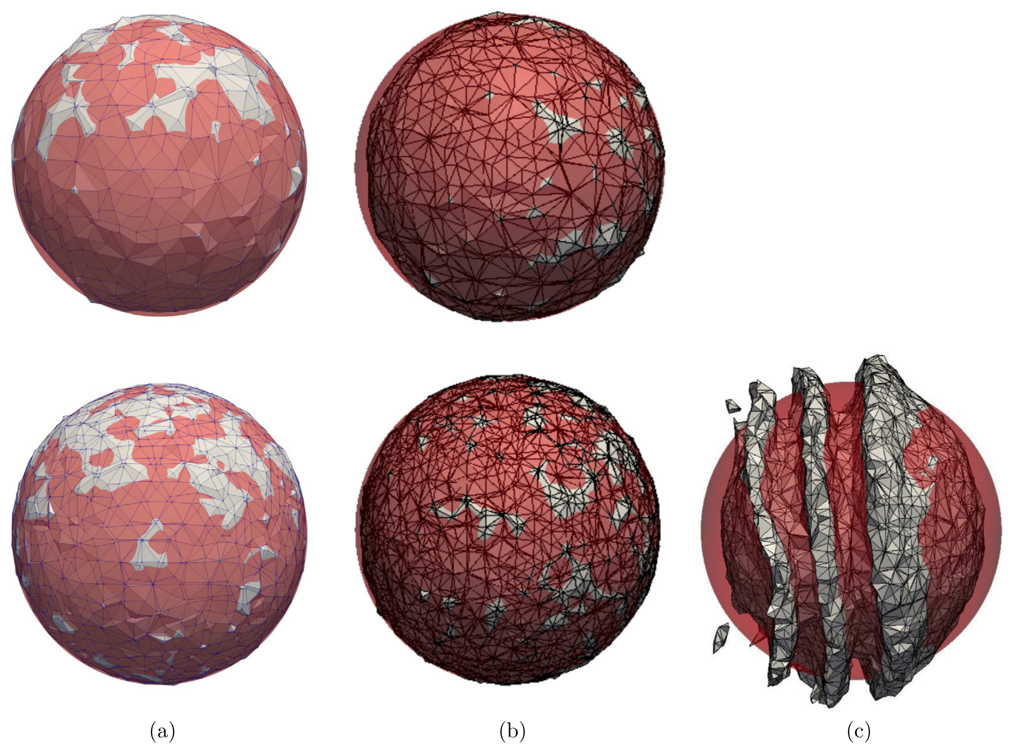


Fig. 23. Isosurfaces, corresponding to the 0.5 isovalue of the fluid volume fraction, extracted on the tetrahedral grids with $\mathcal{N}=41$ (top pictures) and 71 (bottom pictures) at the end of the translation test ($t=t_{\text{end}}$). Results obtained using (a) gVOF (FMFPA and CLCIR), (b) isoAdvector [41] and (c) MULES [41] methods. The exact solution is represented in transparent red color.

shape errors are reduced around 50% with respect to those of isoAdvector-plicRDF and around 30% with respect to those obtained by Jofre et al. [38], which represents a significant accuracy improvement. Results for the computational efficiency and volume conservation are also included in the table. The FMFPA method produces out of bounds volume errors close to the machine precision and shows a very good computational efficiency (the advantage of the CLCIR and LSFIR methods in this respect is clearly seen in the table). It is also noteworthy that the NMFPA method is less accurate than the FMFPA method and not strictly conservative due to the over/underlaps between flux polyhedra.

4.2.3. 3D deformation test

In this test proposed by Enright et al. [67], a sphere of fluid with radius 0.15 and initially centered at (0.35,0.35,0.35) in a domain $[0,1]\times[0,1]\times[0,1]$, is deformed in the following velocity field:

$$\left.\begin{aligned} u(x,y,z,t) &= 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos(\pi t/t_{\text{end}}),\\ v(x,y,z,t) &= -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)\cos(\pi t/t_{\text{end}}),\\ w(x,y,z,t) &= -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos(\pi t/t_{\text{end}}),\end{aligned}\right\} \tag{27}$$

for $t_{\text{end}}=3$. This test produces a high deformation requiring a very fine grid to correctly solve the thinnest fluid structures.

A comparison with the results obtained using the isoAdvector method [41] is presented in Table 5 for cubic grids and in Table 6 for a tetrahedral grid (when using this grid and the NMFPA methods, ϵ has been increased to 10^{-10} to attenuate the effect of the over/underlaps). From the above results, it is evident the advantage in accuracy of the gVOF algorithms (on average, a reduction of around 60% in the shape errors is observed in Table 5). This improvement can be visually observed from results like those shown in Fig. 24. When using cubic grids, the FMFPA results are slightly more accurate than the NMFPA results and almost identical to the EMFPA results. The EMFPA method increases the consumed CPU-time in around 20% and 30% compared to the FMFPA and NMFPA methods, respectively. When using the tetrahedral grid, the EMFPA method is slightly more accurate but with an increment of around 45% and 70% in the consumed CPU time compared to the FMFPA and NMFPA methods, respectively. Although the shape error value for the case with the tetrahedral grid of Table 5 and the isoAdvector method is not available, a volume variation of 0.63% and a CPU time of around 3 days at the end of the test were reported in [41] using a single core of an Intel Xeon 3.10 GHz CPU (E5-2687W). The very good performance and computational efficiency of gVOF, especially when using the CLCIR or LSFIR methods, is clearly observed in the tables.

Now, a comparison with the isoAdvector-plicRDF method [42] and many of the most advanced geometric-unsplit VOF methods recently published is presented in Tables 7 and 8 (the last three combined EMFPA-like advection methods with second-order reconstruction methods). For ease of comparison, Fig. 25 shows in charts the corresponding values of $E^{L_1}_{\text{shape}}(t=3)$ and $\widetilde{t}$.

On average, all the methods included in the table show second-order convergence and gVOF provides the lowest shape errors. Although SWIR and LSFIR are the most accurate methods for $\mathcal{N}\leq 128$, the CLCIR method provides acceptable shape errors with a very good computational efficiency and higher accuracy when the grid resolution is sufficiently high to correctly solve the thinnest fluid structures at the maximum fluid body deformation time. In [31], where an unsplit advection EMFPA-like advection method is combined with a

Table 4
Results for the rotation test using cubic grids and CFL = 1. Comparison with results from [42] and [38]. The execution times $\widetilde{t}_{\text{cpu}}$ reported in [42] were obtained using four cores of a dual Intel Xeon 2687W v2 processor.

Grid size, $\mathcal{N}$	$E^{L_1}_{\text{shape}}(t=2\pi)$	$E^{L_1}_{\text{vol}}(t=2\pi)$	$E^{L_\infty}_{\text{bound}}(t=2\pi)$	$\widetilde{t}_{\text{cpu}}$ (s)
gVOF (NMFPA, CLCIR)				
32	6.74×10^{-4}	1.5×10^{-4}	7.4×10^{-8}	0.0055
64	2.06×10^{-4}	6.7×10^{-5}	2.2×10^{-8}	0.019
128	6.52×10^{-5}	3.0×10^{-5}	1.5×10^{-9}	0.090
256	2.31×10^{-5}	1.3×10^{-5}	1.0×10^{-10}	0.64
gVOF (FMFPA, CLCIR)				
32	3.33×10^{-4}	8.7×10^{-18}	6.9×10^{-19}	0.0055
64	9.31×10^{-5}	2.3×10^{-17}	2.7×10^{-20}	0.020
128	2.40×10^{-5}	7.3×10^{-17}	2.7×10^{-19}	0.092
256	6.34×10^{-6}	1.8×10^{-16}	2.5×10^{-19}	0.65
gVOF (FMFPA, SWIR)				
32	3.65×10^{-4}	3.8×10^{-17}	4.0×10^{-19}	0.013
64	9.11×10^{-5}	0.0	2.7×10^{-20}	0.040
128	2.38×10^{-5}	5.9×10^{-17}	2.7×10^{-19}	0.16
256	6.37×10^{-6}	6.6×10^{-17}	2.5×10^{-19}	0.94
gVOF (FMFPA, LSFIR)				
32	3.62×10^{-4}	2.3×10^{-17}	1.5×10^{-18}	0.0074
64	9.30×10^{-5}	3.6×10^{-17}	2.7×10^{-20}	0.019
128	2.38×10^{-5}	3.8×10^{-17}	2.4×10^{-19}	0.092
256	6.35×10^{-6}	9.7×10^{-17}	1.4×10^{-19}	0.66
isoAdvector-plicRDF [42]				
32	7.50×10^{-4}	3.5×10^{-18}	5.4×10^{-21}	0.034
64	1.86×10^{-4}	2.7×10^{-16}	7.4×10^{-22}	0.1
128	4.77×10^{-5}	3.6×10^{-15}	8.9×10^{-23}	0.44
256	1.41×10^{-5}	2.5×10^{-14}	5.4×10^{-14}	2.6
Jofre et al. [38]				
32	5.47×10^{-4}	––	––	––
64	1.29×10^{-4}	––	––	––
128	3.46×10^{-5}	––	––	––

Table 5
Results for the 3D deformation test using cubic grids and CFL = 0.5. Comparison with results from [41], where the execution times t_{cpu} were obtained using a single core of an Intel Xeon 3.1 GHz CPU (E5-2687W).

Grid size, $\mathcal{N}$	$E^{L_1}_{\text{shape}*}(t=3)$	$E^{L_1}_{\text{vol}}(t=3)$	$E^{L_\infty}_{\text{bound}}(t=3)$	t_{cpu} (s)
gVOF (NMFPA, CLCIR)				
64	1.52×10^{-1}	8.2×10^{-6}	4.0×10^{-10}	11
128	3.13×10^{-2}	4.0×10^{-6}	2.3×10^{-10}	106
256	4.40×10^{-3}	1.8×10^{-6}	1.3×10^{-11}	1426
gVOF (FMFPA, CLCIR)				
64	1.39×10^{-1}	6.8×10^{-17}	4.0×10^{-19}	13
128	3.04×10^{-2}	6.6×10^{-17}	1.4×10^{-19}	113
256	4.23×10^{-3}	5.5×10^{-15}	6.8×10^{-20}	1455
gVOF (EMFPA, CLCIR)				
64	1.39×10^{-1}	3.6×10^{-17}	1.4×10^{-19}	16
128	3.04×10^{-2}	5.2×10^{-18}	1.2×10^{-19}	135
256	4.23×10^{-3}	5.5×10^{-15}	1.0×10^{-19}	1617
gVOF (FMFPA, SWIR)				
64	7.39×10^{-2}	3.5×10^{-18}	1.2×10^{-19}	31
128	2.15×10^{-2}	2.6×10^{-17}	1.4×10^{-19}	248
256	4.39×10^{-3}	5.4×10^{-15}	7.8×10^{-20}	2672
gVOF (FMFPA, LSFIR)				
64	7.65×10^{-2}	4.9×10^{-17}	1.2×10^{-19}	12
128	1.90×10^{-2}	5.7×10^{-17}	8.7×10^{-20}	112
256	4.41×10^{-3}	5.5×10^{-15}	7.7×10^{-20}	1505
isoAdvector [41]				
64	2.2×10^{-1}	3.7×10^{-15}	0	173
128	4.7×10^{-2}	3.7×10^{-14}	3.0×10^{-13}	2626
256	1.2×10^{-2}	2.3×10^{-13}	1.1×10^{-10}	46706

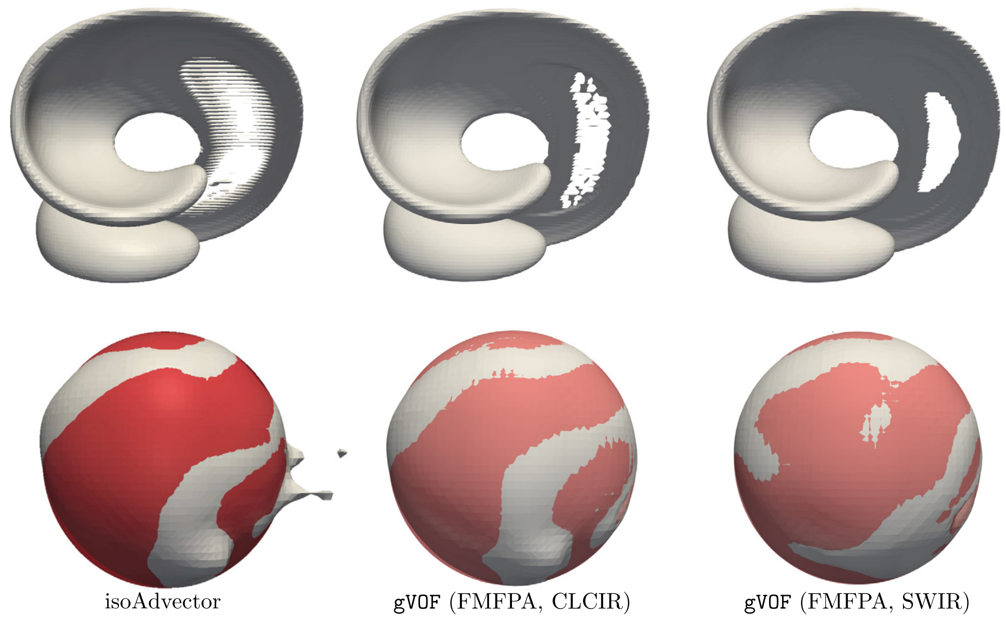
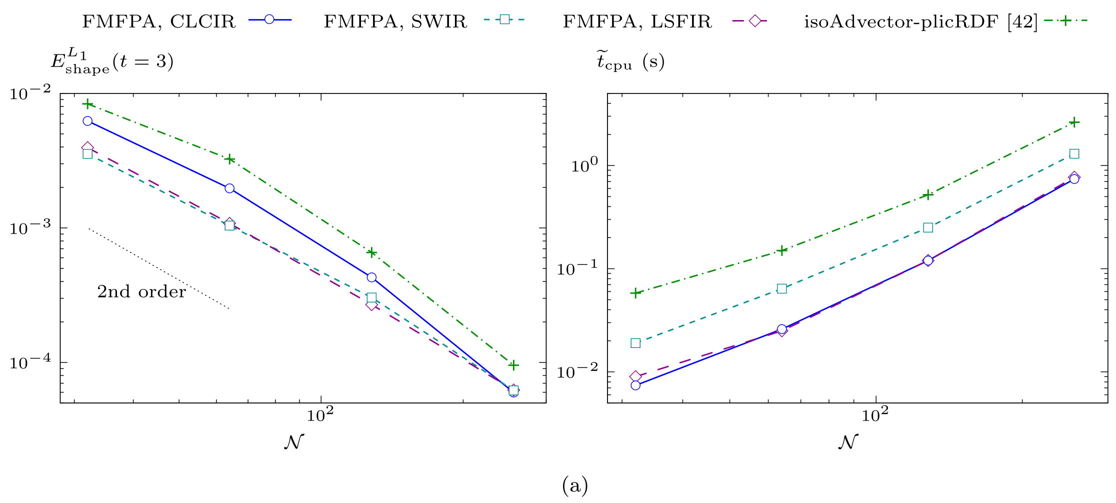


Fig. 24. Isosurfaces extracted on a cubic grid with $\mathcal{N} = 128$ at the instant of maximum deformation (top pictures) and end instant of the test (bottom pictures). The results are compared with those presented in [41] using the isoAdvector method.



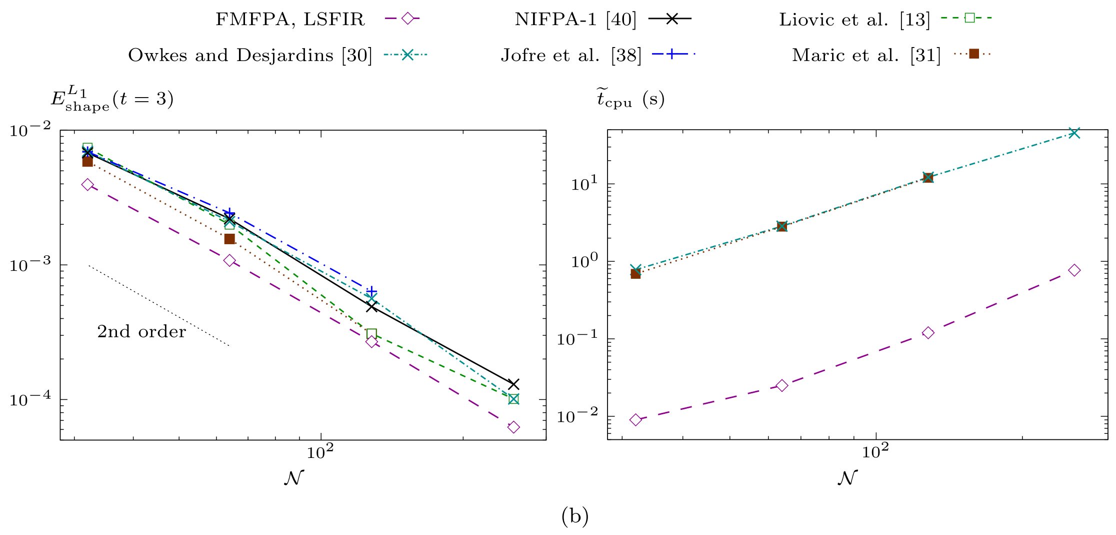


Fig. 25. Values of the error $E^{L_1}_{\text{shape}}(t=3)$ (left picture) and average CPU time $\widetilde{t}$ (s) (right picture) from (a) Table 7, and (b) Table 8 (the results obtained using the FMFPA and LSFIR methods are also included for comparison).

Table 6
Same results as in Table 5, but using the tetrahedral grid with $\mathcal{N} = 317$ from [63].

$E^{L_1}_{\text{shape*}}(t=3)$	$E^{L_1}_{\text{vol}}(t=3)$	$E^{L_\infty}_{\text{bound}}(t=3)$	t_{cpu} (s)
gVOF (NMFPA, CLCIR)			
4.06×10^{-3}	3.3×10^{-8}	4.7×10^{-12}	4217
gVOF (FMFPA, CLCIR)			
4.01×10^{-3}	2.6×10^{-9}	1.3×10^{-13}	4985
gVOF (EMFPA, CLCIR)			
3.94×10^{-3}	4.0×10^{-9}	8.4×10^{-14}	7196
gVOF (EMFPA, SWIR)			
4.63×10^{-3}	9.8×10^{-9}	5.4×10^{-14}	39128
gVOF (EMFPA, LSFIR)			
7.92×10^{-3}	1.22×10^{-8}	4.9×10^{-12}	7783
isoAdvector [41]			
−−	8.9×10^{-5}	−−	$\simeq 259200$

Table 7
Results for the 3D deformation test using cubic grids and CFL = 0.5. Comparison with results from (the resources used to execute the tests is included in parentheses) [42] (4 cores of a dual Intel Xeon 2687W v2).

Grid size, $\mathcal{N}$	$E^{L_1}_{\text{shape}}(t=3)$	$E^{L_1}_{\text{vol}}(t=3)$	$E^{L_\infty}_{\text{bound}}(t=3)$	$\widetilde{t}_{\text{cpu}}$ (s)
gVOF (FMFPA, CLCIR)				
32	6.24×10^{-3}	1.0×10^{-16}	1.6×10^{-19}	0.0074
64	1.97×10^{-3}	6.8×10^{-17}	4.0×10^{-19}	0.026
128	4.29×10^{-4}	6.6×10^{-17}	1.4×10^{-19}	0.12
256	5.98×10^{-5}	5.5×10^{-15}	6.8×10^{-20}	0.74
gVOF (FMFPA, SWIR)				
32	3.55×10^{-3}	3.0×10^{-16}	1.7×10^{-19}	0.019
64	1.04×10^{-3}	3.5×10^{-18}	1.2×10^{-19}	0.064
128	3.04×10^{-4}	2.6×10^{-17}	1.4×10^{-19}	0.25
256	6.20×10^{-5}	5.4×10^{-15}	7.8×10^{-20}	1.30
gVOF (FMFPA, LSFIR)				
32	3.95×10^{-3}	1.4×10^{-16}	1.9×10^{-19}	0.0090
64	1.08×10^{-3}	4.9×10^{-17}	1.2×10^{-19}	0.025
128	2.69×10^{-4}	5.7×10^{-17}	8.7×10^{-20}	0.12
256	6.24×10^{-5}	5.5×10^{-15}	7.7×10^{-20}	0.77
isoAdvector-plicRDF [42]				
32	8.36×10^{-3}	1.1×10^{-16}	2.1×10^{-14}	0.058
64	3.25×10^{-3}	9.7×10^{-16}	6.4×10^{-20}	0.15
128	6.57×10^{-4}	3.7×10^{-15}	1.1×10^{-15}	0.52
256	9.54×10^{-5}	2.0×10^{-14}	2.4×10^{-16}	2.63

modified version of the Swartz reconstruction method proposed by Dyadechko and Shashkov [55], zero bounding errors are reported, which is somewhat surprising given the high geometric complexity of the unsplit advection method. The results obtained by Owkes and Desjardins [30] and Jofre et al. [38], who also use EMFPA-like advection methods and, respectively, the second-order ELVIRA and LVIRA reconstruction methods [12], are relatively similar and on average their shape errors are around 90% higher than those of gVOF. A visual comparison with the results obtained with the isoAdvector-plicRDF method, which shows an accuracy close to that of Owkes and Desjardins [30] and Jofre et al. [38], can also be seen on Fig. 26, where the improvement achieved using gVOF is clearly seen. Note that SWIR method better preserves the integrity of the thin fluid structures (the LSFIR method has a similar behavior in this respect). The NIFPA-1 method proposed by Ivey and Moin [40] combined with the embedded height-function method proposed also by Ivey and Moin [9] provides accuracies comparable to those of other advanced geometric unsplit VOF methods. Although, as it was mentioned, the results of $\widetilde{t}_{\text{cpu}}$ included in the table should only be considered as a qualitative reference, one of the reasons for the clear advantage of gVOF with respect to others in terms of computational efficiency, is the use of the non-convex version of VOFTools [45,52], which avoids the use of costly techniques to decompose the generally non-convex flux polyhedra into convex sub-polyhedra.

Fig. 27 shows the PLIC interfaces for the 3D deformation test at different instants obtained using gVOF (EMFPA and CLCIR), CFL = 0.5 and different grids of $\mathcal{N} = 160$ with convex and non-convex cells, where the capacity of gVOF to solve tests of high interface deformation on 3D arbitrary grids can be clearly seen.

4.2.4. 2D deformation test

In this test introduced by Bell et al. [68], a cylinder of fluid with radius 0.15, symmetry axis parallel to the y-axis and initially centered at (0.5,0,0.75) in a domain $[0, 1] \times [0, 1] \times [0, 1]$, is deformed in the following velocity field:

Table 8
Continuation of Table 7. Comparison with results from [40] (no execution time reported; the results were obtained with CFL sufficiently low to get $E^{L_1}_{\text{vol}}(8) < 10^{-12}$), [30] (4 cores of a dual Intel Xeon X5670), [38] (no execution time reported) and [31] (4 cores of an Intel Xeon E5-2680 v3).

Grid size, $\mathcal{N}$	$E^{L_1}_{\text{shape}}(t=3)$	$E^{L_1}_{\text{vol}}(t=3)$	$E^{L_\infty}_{\text{bound}}(t=3)$	$\widetilde{t}_{\text{cpu}}$ (s)
NIFPA-1 [40]*				
33	6.8×10^{-3}	$< 1.4 \times 10^{-14}$	$< 1.4 \times 10^{-14}$	−−
65	2.2×10^{-3}	$< 1.4 \times 10^{-14}$	$< 1.4 \times 10^{-14}$	−−
129	4.9×10^{-4}	$< 1.4 \times 10^{-14}$	$< 1.4 \times 10^{-14}$	−−
257	1.3×10^{-4}	$< 1.4 \times 10^{-14}$	$< 1.4 \times 10^{-14}$	−−
Liovic et al. [13]				
32	7.41×10^{-3}	−−	−−	−−
64	1.99×10^{-3}	−−	−−	−−
128	3.09×10^{-4}	−−	−−	−−
256	7.03×10^{-5}	−−	−−	−−
Owkes and Desjardins [30]				
32	6.98×10^{-3}	1.2×10^{-15}	1.2×10^{-17}	0.78
64	2.10×10^{-3}	2.5×10^{-15}	2.3×10^{-17}	2.85
128	5.63×10^{-4}	1.7×10^{-14}	2.8×10^{-17}	12.2
256	1.01×10^{-4}	3.9×10^{-14}	4.7×10^{-17}	45.5
Jofre et al. [38]				
32	6.92×10^{-3}	−−	−−	−−
64	2.43×10^{-3}	−−	−−	−−
128	6.37×10^{-4}	−−	−−	−−
Marić et al. [31]				
32	5.86×10^{-3}	2.5×10^{-15}	0.0	0.69
64	1.56×10^{-3}	6.0×10^{-15}	0.0	2.81
128	3.08×10^{-4}	1.6×10^{-14}	0.0	12.0

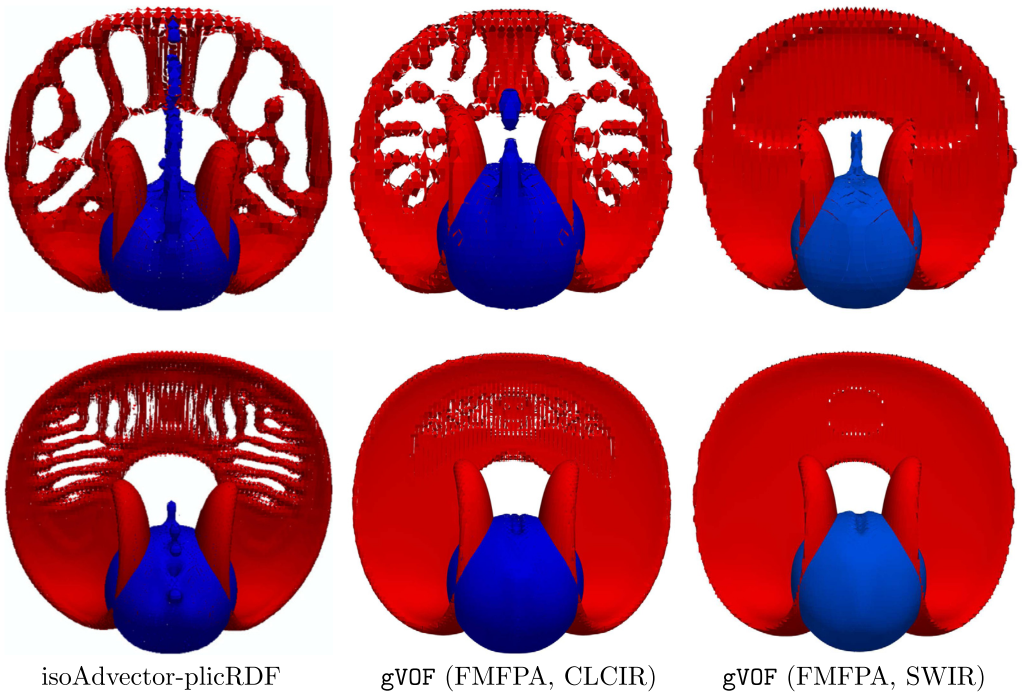


Fig. 26. PLIC interfaces at $t = 1.5$ (in red color) and $t = 3$ (in blue color) obtained using the isoAdvector-plicRDF method [42] (left pictures) and gVOF, combining FMFPA method with CLCIR and SWIR methods (middle and right pictures, respectively), for the 3D deformation test with two cubic grids with $\mathcal{N} = 64$ (top pictures) and 128 (bottom pictures), and CFL = 0.5.

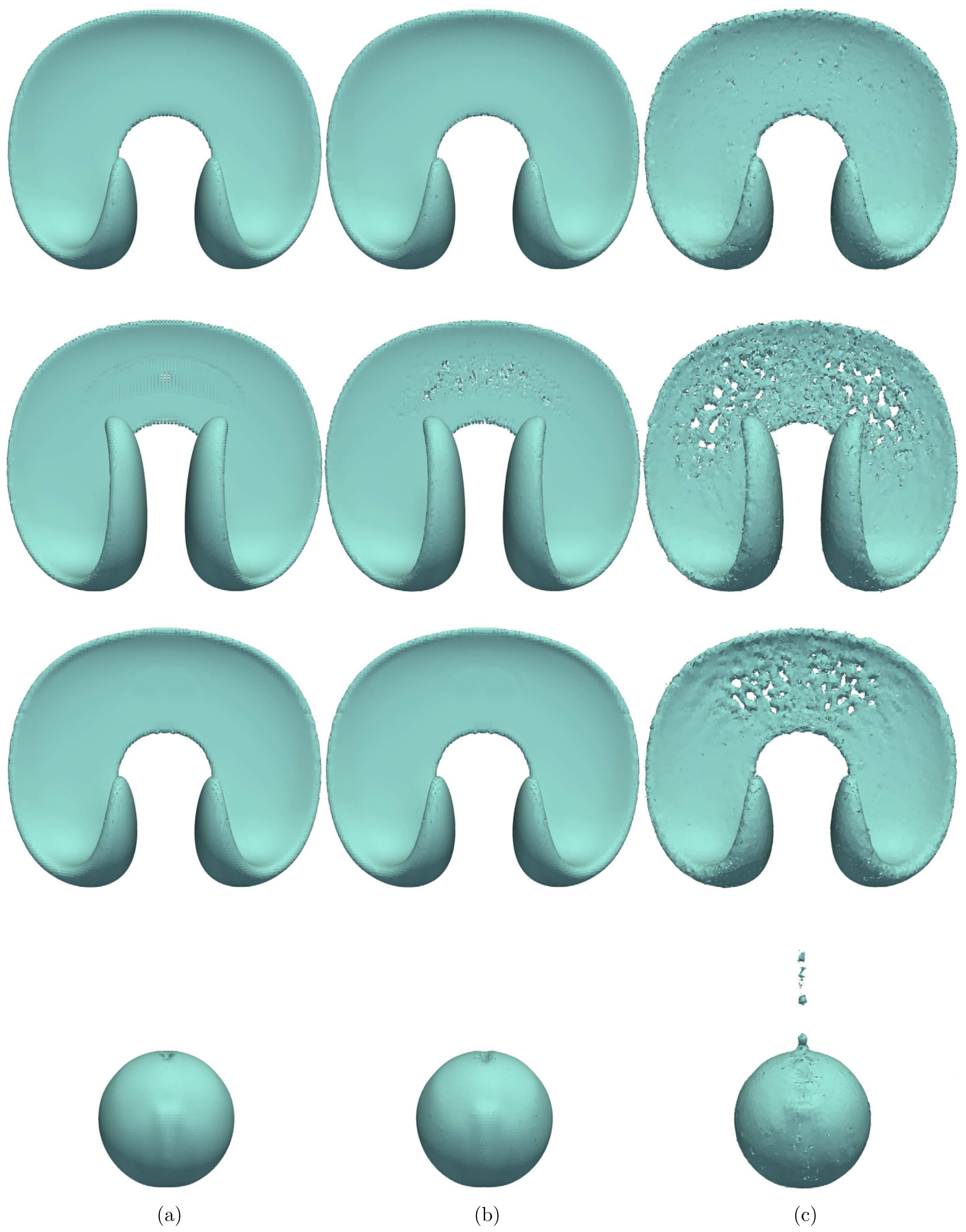


Fig. 27. PLIC interfaces for the 3D deformation test at $t = 1.0$, 1.5, 2.0 and 3.0 (from top to bottom) obtained using gVOF (EMFPA and CLCIR), CFL = 0.5 and (a) cubic, (b) non-convex distorted cubic and (c) tetrahedral grids with $\mathcal{N} = 160$.

$$\left.\begin{aligned} u(x,y,z,t) =& -2\sin^2(\pi x)\sin(\pi z)\cos(\pi z)\cos(\pi t/t_{\text{end}}),\\ v(x,y,z,t) =& 0,\\ w(x,y,z,t) =& 2\sin^2(\pi z)\sin(\pi x)\cos(\pi x)\cos(\pi t/t_{\text{end}}),\end{aligned}\right\} \qquad (28)$$

for $t_{\text{end}} = 8$. Note that this 2D deformation test is simulated in this work using 3D grids with a single cell along the y-axis. These quasi-2D grids will be denoted hereafter as square or triangular grids.

Table 9 compares the $E^{L_1}_{\text{shape}*}$ error values at $t = 8$ obtained using gVOF (combining the FMFPA method with the CLCIR, SWIR and LSFIR methods) and CFL = 0.5 with those obtained using the isoAdvector and MULES methods [41] on square and unstructured grids available in [63] (the results corresponding to the MULES method were obtained using CFL = 0.1). A substantial reduction of the error values, being on average around a 60% with respect to the isoAdvector method and higher than 90% with respect to the MULES method, has been achieved by the gVOF code. The corresponding PLIC interfaces obtained with gVOF (FMFPA and CLCIR) at $t = 4$ (dark blue color) and $t = 8$ (red color) can be seen in Fig. 28 (the exact fluid region at $t = 8$ is depicted in light blue color). Note that for 2D problems, the EMFPA and FMFPA methods produce equivalent flux polyhedron shapes, although, as mentioned, the EMFPA method uses more faces and vertices for each flux polyhedron to construct the flux region, affecting this, obviously, to the computational efficiency. As expected,

Table 9
$E^{L_1}_{\text{shape}*}$ error values at $t = 8$ for the 2D deformation test using different grids and CFL = 0.5. Comparison with the isoAdvector and MULES methods [41]. The results presented for MULES were obtained in [41] using CFL = 0.1.

Grid size,	gVOF, FMFPA+			isoAdvector	MULES
$\mathcal{N}$	CLCIR	SWIR	LSFIR	[41]	[41]
Square grids					
100	2.71×10^{-2}	1.57×10^{-2}	1.61×10^{-2}	4.7×10^{-2}	–
200	7.60×10^{-3}	3.60×10^{-3}	3.82×10^{-3}	1.2×10^{-2}	7.2×10^{-2}
400	1.24×10^{-3}	1.32×10^{-3}	1.77×10^{-3}	2.3×10^{-3}	–
Triangular grids [63]					
140	3.30×10^{-2}	2.23×10^{-2}	2.23×10^{-2}	5.4×10^{-2}	–
281	8.77×10^{-3}	5.47×10^{-3}	6.15×10^{-3}	2.0×10^{-2}	6.6×10^{-1}
564	2.45×10^{-3}	8.18×10^{-4}	1.62×10^{-3}	9.5×10^{-3}	–

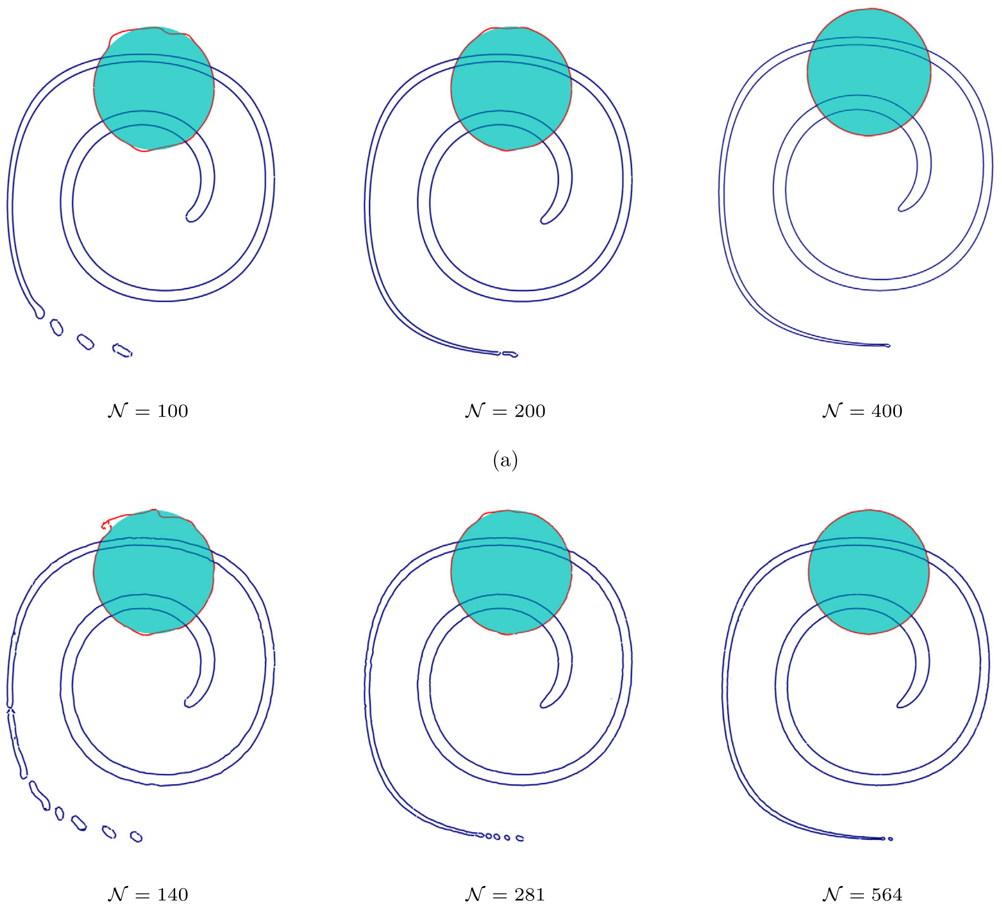


Fig. 28. PLIC interfaces (in blue for $t = 4$ and in red for $t = 8$) obtained using FMFPA and CLCIR methods with CFL = 0.5 and different grids. (a) Square grids with $\mathcal{N} = 100$, 200 and 400. (b) Triangular grids with $\mathcal{N} = 140$, 281 and 564. The exact solution at the end of the test is represented in light blue color.

the results obtained using the NMFPA method, which are omitted in this test for brevity, are less accurate than those obtained using the EMFPA or FMFPA method.

Now, Table 10 compares the $E^{L_1}_{\text{shape}}(8)$ error values obtained using gVOF (combining now the EMFPA method with the CLCIR and SWIR methods) with those obtained using several of the most advanced geometric VOF methods developed to date. The shape error reduction achieved by gVOF for this test of high deformation is around 60% and 50% compared to the methods implemented by Owkes and Desjardins [30] and Marić et al. [31], respectively, and around 80% and 70% compared to the isoAdvector-plicRDF method of Scheufler and Roenby [42] and NIFPA-1 method of Ivey and Moin [40], respectively, which represents a remarkable achievement.

Table 10
$E^{L_1}_{\text{shape}}$ error values at $t = 8$ for the 2D deformation test using square grids and different CFL values. Comparison with recent advanced geometric-unsplit VOF methods.

Grid size,	gVOF, EMFPA +		Owkes and	isoAdvector-	Marić	NIFPA-1
$\mathcal{N}$	CLCIR	SWIR	Desjardins [30]	plicRDF [42]	et al. [31]	[40][*]
CFL = 1						
64	3.51×10^{-3}	4.15×10^{-3}	–	8.70×10^{-3}	5.74×10^{-3}	–
128	8.85×10^{-4}	7.20×10^{-4}	–	2.27×10^{-3}	1.45×10^{-3}	–
256	1.94×10^{-4}	2.14×10^{-4}	–	1.61×10^{-3}	3.77×10^{-4}	–
512	6.37×10^{-5}	4.45×10^{-5}	–	1.93×10^{-3}	–	–
1024	1.59×10^{-5}	1.35×10^{-5}	–	2.75×10^{-3}	–	–
CFL = 0.5						
64	6.35×10^{-3}	4.37×10^{-3}	7.75×10^{-3}	1.26×10^{-2}	–	1.2×10^{-2}
128	1.19×10^{-3}	8.18×10^{-4}	1.87×10^{-3}	2.61×10^{-3}	–	2.7×10^{-3}
256	2.07×10^{-4}	1.72×10^{-4}	4.04×10^{-4}	5.71×10^{-4}	–	5.4×10^{-4}
512	5.45×10^{-5}	4.11×10^{-5}	8.32×10^{-5}	1.04×10^{-4}	–	1.7×10^{-4}
1024	1.32×10^{-5}	1.11×10^{-5}	2.35×10^{-5}	3.50×10^{-5}	–	4.6×10^{-5}

[*] These results were obtained for $\mathcal{N}+1$ and a CFL sufficiently low to get $E^{L_1}_{\text{vol}}(8) < 7 \times 10^{-14}$.

5. Coupling gVOF with an in-house CFD code

To show the performance of gVOF coupled with an in-house code that solves the flow conservation equations (previous results obtained with this code coupled with VOF and level-set methods can be found in [7,16,29,46]), the impact of a water drop onto a free surface is solved. The code solves the conservation equations on both sides of the interface using a projection method. The projection step incorporates a continuous surface tension model based on the balanced-force algorithm proposed by François et al. [69], in which the interface curvature is computed using the height function technique that incorporates the improvements proposed in [7] and [8]. The pressure Poisson equation resulting from the projection step, which is the most expensive part of the code, is solved using a preconditioned Krylov solver with the help of LIS (library of iterative solvers) [70]. A CFL number of 0.2 was used to determine the time step as

$$\Delta t = \min\left(\sqrt{\frac{[\rho_l + \rho_g]h^3}{4\pi\sigma}}, \frac{h}{u_{\max}}\right)\text{CFL},$$

where ρ_l and ρ_g and the densities of the liquid and gas phases, respectively, σ is the surface tension coefficient, h is the cell size and $u_{\max}$ is the maximum absolute value of the velocity components at each instant.

A water drop of diameter $D = 2.9$ mm impacting a deep water pool with velocity $U = 2.5\ \text{m}\,\text{s}^{-1}$ is considered. The Froude and Weber numbers are $Fr = 220$ and $We = 248$, respectively. Due to the symmetry of the problem, only one quarter of the physical domain was considered. The computational domain used was $8D \times 3.5D \times 3.5D$. The pool depth was $4.5D$, the water drop was initially located at a height equal to $6.0D$ and the domain was discretized on a grid of $175 \times 70 \times 70$ cells. To reach the desired impact velocity, a fictitious gravitational force was used to accelerate the drop. Fig. 29 shows results for the 0.5-isosurfaces obtained using the EMFPA and CLCIR methods at different instants after the drop had made contact with the pool surface (right pictures). A relatively good degree of agreement can be observed with the visualization results obtained experimentally in [29] (left pictures; note that the images of the interface are magnified when observed through water). A quantitative comparison can be seen in Fig. 30, where numerical and experimental results of the evolution of the free-surface depth at the symmetry axis, D_c, are compared. A reasonable agreement can be observed for the cavity depth evolution during growth and collapse processes. Note that the numerical predictions obtained using the EMFPA and FMFPA methods are very closed for this drop impact test. On average, the CPU time consumed by the advection and reconstruction schemes represent around 20% of the total. In all simulations, the net change in total volume at the end of the test was lower than 4×10^{-6}%.

6. Parallel performance

To assess the parallel performance of the gVOF package, the 3D deformation test of Section 4.2.3 was executed using an increasing number of threads. The gfortran compiler with -O3 optimization and the OpenMP application programming interface [61] were used on an iMac Pro 2017 with an 18-core Intel Xeon W 2.3 GHz CPU. Each iteration is performed over a single cell and is treated independently in the parallel loop. The distribution of loop iterations among threads is left to be done automatically by the compiler. The same conditions as in Table 7 but using the EMFPA and CLCIR methods, and a cubic grid with $\mathcal{N} = 256$ are considered for this analysis. The top picture in Fig. 31 shows the total CPU times of the interface reconstruction and fluid advection as a function of the number of threads used during the execution of the test case. The corresponding speedup achieved with respect to the execution with one thread is also shown in Fig. 31 (bottom picture). Note that fluid advection spends around twenty times the CPU time consumed by the interface reconstruction when using a single thread, while this difference tends to reduce to around ten when the number of threads increases. It should be mentioned that the automatic distribution of loop iterations used in this work may cause uneven progress during execution, which could explain the slight changes in speedup observed in Fig. 31. Future work will investigate additional considerations based, for example, on the VOF fraction distribution or local grid resolution, to better control how loop iterations are distributed.

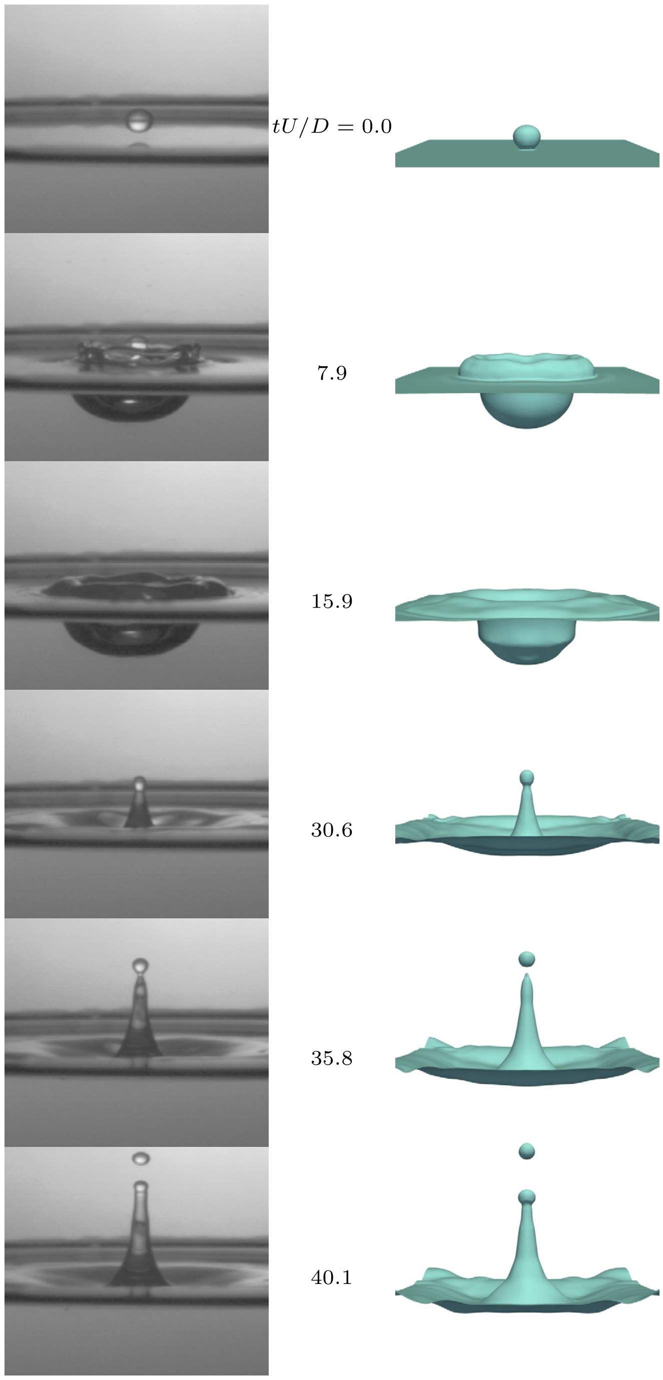


Fig. 29. Water drop impact test. Comparison between the experimental results for the interface shape [29] (left pictures) and numerical predictions for the 0.5-isosurfaces obtained using the EMFPA and CLCIR methods (right pictures) at different instants after the drop had made contact with the pool surface.

7. Conclusions

The `gVOF` package, which is provided as open source, includes routines for the implementation of advanced geometric VOF methods on arbitrary grids that achieve accuracies and computational efficiencies which are cutting edge. The package uses the `VOFTools` and `isoap` libraries to implement a VOF initialization method and several PLIC interface reconstruction and unsplit advection methods valid for arbitrary grids with convex or non-convex cells. The routines are written in `FORTRAN` and can be used with `C` codes through interface routines included in the distributed software. To improve the computational efficiency, the `OpenMP` application programming interface is used. To assess the performance of the package, interface reconstruction and advection tests are also included in the supplied software. A thorough comparison with existing state-of-the-art geometric VOF methods has been carried out and has provided very favorable results. Also, the package has been incorporated into an existing in-house code to simulate the impact of a water drop on a free surface. The numerical results have been compared with experimental results, and a good agreement has been found.

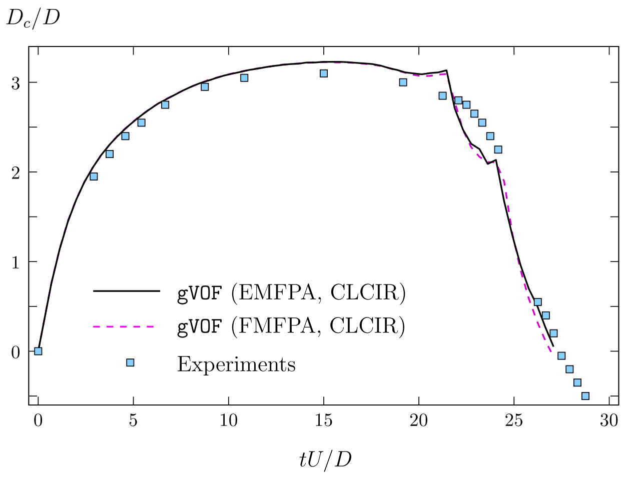


Fig. 30. Comparison between numerical predictions for the drop impact test of Fig. 29 and experimental results [29] for the evolution of the free-surface depth, D_c, at the symmetry axis.

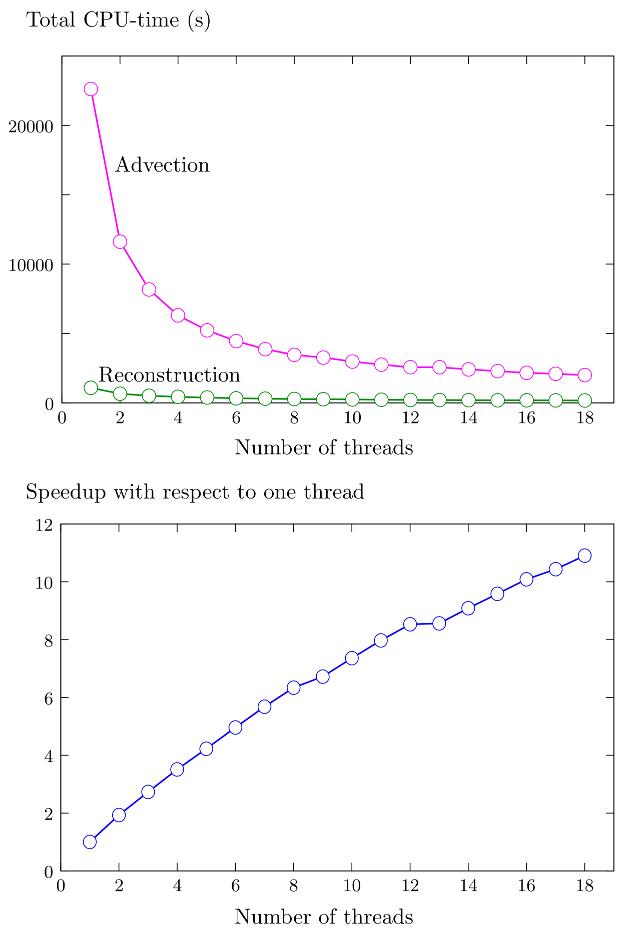


Fig. 31. Total CPU time consumed by interface reconstruction and fluid advection (top picture) and speedup with respect to the execution with one thread (bottom picture), as a function of the number of threads, obtained for the 3D deformation test using `gVOF` (EMFPA and CLCIR), a cubic grid with $\mathcal{N} = 256$ and CFL = 0.5, on an 18-core Intel Xeon W 2.3 GHz CPU.

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 joint 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, and the Spanish Ministerio de Ciencia e Innovación - Agencia Estatal de Investigación (MCIN/ AEI / 10.13039/501100011033) through projects PID2020-120100GB-C21 and PID2020-120100GB-C22.

Appendix A. Nomenclature

$A =$	matrix in Eq. (7); area of the cell face
$\boldsymbol{b} =$	vector in Eq. (8)
$C =$	constant that determines the position of the PLIC interface
$D =$	drop diameter
$D_c =$	free-surface depth at the symmetry axis
$\mathcal{D} =$	dimensions number (2 for 2D and 3 for 3D)
$E^{L_1}_{\text{bound}} =$	average $E^{L_\infty}_{\text{bound}}(t)$ value during the complete simulation
$E^{L_\infty}_{\text{bound}} =$	maximum $E^{L_\infty}_{\text{bound}}(t)$ value during the complete simulation
$E^{L_\infty}_{\text{bound}}(t) =$	maximum unboundedness error of the fluid volume at a time t
$E^{L_1}_{\text{rec}} =$	interface reconstruction error
$E^{L_1}_{\text{shape}} =$	interface shape error
$E^{L_1}_{\text{shape}^*} =$	relative interface shape error
$E^{L_1}_{\text{vol}} =$	fluid volume error
$f =$	auxiliary VOF function
$F =$	fluid volume fraction (discretized version of f)
$\widetilde{F} =$	Taylor series expanded value of F
$F^{\text{e}} =$	exact fluid volume fraction of a cell
$F^* =$	fluid volume fraction interpolated at a grid node
$Fr =$	Froude number
$h =$	cubic cell size
$h_x, h_y, h_z =$	sizes along the coordinate axis x, y, z of the minimum-size rectangular parallelepiped that encloses a cell
$I =$	number of vertices of the first face in a flux polyhedron
$n =$	current time step
$\boldsymbol{n} =$	unit vector normal to the interface pointing into the fluid or normal to the cell face pointing out of the cell
$\mathcal{N} =$	grid size
$N_{\text{CELL}} =$	number of grid cells in the computational domain
$N_{\text{FACE}} =$	number of grid faces in the computational domain
$N_{\text{STEP}} =$	number of time steps required to complete the advection test
$\widetilde{N}_{\text{CELL}} =$	number of grid cells in an equivalent unit domain
$t =$	time
$t_0 =$	previous time in an advection test
$t_{\text{adv}} =$	total CPU time consumed by the advection step
$t_{\text{cpu}} =$	total execution CPU time
$t_{\text{end}} =$	end time of an advection test
$t_{\text{rec}} =$	total CPU time consumed by the reconstruction step
$\widetilde{t}_{\text{cpu}} =$	average total execution CPU time per time step
$U =$	impact velocity
$\boldsymbol{u} =$	velocity vector
$u_{max} =$	maximum absolute value of the velocity components
$u, v, w =$	components of $\boldsymbol{u}$
$V_d =$	volume of the flux region

$V_{d_T} =$	total net flux volume at the cell
$V_F =$	volume of the fluid advected through a cell face
$V_{F_T} =$	total net volume of fluid that leaves (or enters) the cell
$V_\Omega =$	volume of the grid cell Ω
$w =$	weighting factor
$We =$	Weber number
$\boldsymbol{x} =$	position vector
$x, y, z =$	Cartesian coordinates

Subscripts

$i =$	face vertex index
$j =$	cell face index
$k =$	neighbor grid cell index

Superscripts

$n =$	time step

Greek characters

$\alpha =$	facet angle in the triangulated isosurface
$\beta =$	parameter in Eq. (9)
$\Delta t =$	time step
$\epsilon =$	fluid volume fraction tolerance
$\Omega =$	grid cell
$\Omega^p =$	flux polyhedron
$\rho_g =$	gas density
$\rho_l =$	liquid density
$\sigma =$	surface tension coefficient

Acronyms

CFD =	computational fluid dynamics
CFL =	Courant-Friedrich-Levy number
CIBRAVE =	coupled interpolation-bracketed analytical volume enforcement
CLCIR =	conservative level contour interface reconstruction
ELCIR =	extended level contour interface reconstruction
EMFPA =	edge-matched flux polygon/polyhedron advection
FMFPA =	face-matched flux polyhedron advection
LLCIR =	local level contour interface reconstruction
LSFIR =	least-squares fit interface reconstruction
LSGIR =	least-squares gradient interface reconstruction
NIFPA =	non-intersecting flux polyhedron advection
NMFPA =	non-matched flux polyhedron advection
PLIC =	piecewise linear interface calculation
SLIC =	simple line interface calculation
SWIR =	Swartz interface reconstruction
VOF =	volume of fluid

References

[1] G. Tryggvason, R. Scardovelli, S. Zaleski, Direct Numerical Simulations of Gas-Liquid Multiphase Flows, Cambridge University Press, 2011.
[2] Y. Renardy, M. Renardy, J. Comput. Phys. 183 (2002) 400–421.
[3] S.J. Cummins, M.M. François, D.B. Kothe, Comput. Struct. 83 (2005) 425–434.
[4] S. Afkhami, M. Bussmann, Int. J. Numer. Methods Fluids 57 (2007) 453–472.
[5] S. Afkhami, M. Bussmann, Int. J. Numer. Methods Fluids 61 (2008) 827–847.
[6] S. Popinet, J. Comput. Phys. 228 (2009) 5838–5866.
[7] J. López, C. Zanzi, P. Gómez, R. Zamora, F. Faura, J. Hernández, Comput. Methods Appl. Mech. Eng. 198 (2009) 2555–2564.

[8] J. López, J. Hernández, J. Comput. Phys. 229 (2010) 4855–4868.
[9] C.B. Ivey, P. Moin, J. Comput. Phys. 300 (2015) 365–386.
[10] Z. Jibben, N.N. Carlson, M.M. François, Comput. Math. Appl. 78 (2019) 643–653.
[11] R. Scardovelli, S. Zaleski, Int. J. Numer. Methods Fluids 41 (2003) 251–274.
[12] J.E. Pilliod, E.G. Puckett, J. Comput. Phys. 199 (2004) 465–502.
[13] P. Liovic, M. Rudman, J-L. Liow, D. Lakehal, D. Kothe, Comput. Fluids 35 (2006) 1011–1032.
[14] E. Aulisa, S. Manservisi, R. Scardovelli, S. Zaleski, J. Comput. Phys. 225 (2007) 2301–2319.
[15] J. López, J. Hernández, P. Gómez, F. Faura, J. Comput. Phys. 195 (2004) 718–742.
[16] J. López, J. Hernández, P. Gómez, F. Faura, J. Comput. Phys. 208 (2005) 51–74.
[17] J. López, C. Zanzi, P. Gómez, F. Faura, J. Hernández, Int. J. Numer. Methods Fluids 58 (2008) 923–944.
[18] S. Mirjalili, S.S. Jain, M.S. Dodd, in: Center for Turbulence Research, Annual Research, 2017, pp. 117–135.
[19] T. Marić, D.B. Kothe, D. Bothe, J. Comput. Phys. (2018) 109695.
[20] E. Aulisa, S. Manservisi, R. Scardovelli, S. Zaleski, J. Comput. Phys. 192 (2003) 355–364.
[21] A. Baraldi, M.S. Dodd, A. Ferrante, Comput. Fluids 96 (2014) 322–337.
[22] R. DeBar, Fundamentals of the Kraken code, Lawrence Livermore Laboratory, 1974, UCIR-760.
[23] C.W. Hirt, B.D. Nichols, J. Comput. Phys. 39 (1981) 201–225.
[24] 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 34th Aerospace Sciences Meeting and Exhibit.
[25] W.J. Rider, D.B. Kothe, J. Comput. Phys. 141 (1998) 112–152.
[26] D.J.E. Harvie, D.F. Fletcher, J. Comput. Phys. 162 (2000) 1–32.
[27] D.J.E. Harvie, D.F. Fletcher, Int. J. Numer. Methods Fluids 35 (2) (2001) 151–172.
[28] G.H. Miller, P. Colella, J. Comput. Phys. 183 (2002) 26–82.
[29] J. Hernández, J. López, P. Gómez, C. Zanzi, F. Faura, Int. J. Numer. Methods Fluids 58 (2008) 897–921.
[30] M. Owkes, O. Desjardins, J. Comput. Phys. 270 (2014) 587–612.
[31] T. Marić, H. Marschall, D. Bothe, J. Comput. Phys. 371 (2018) 367–393.
[32] S. Mosso, B. Swartz, D. Kothe, R. Ferrell, in: P. Schiano, A. Ecer, J. Periaux, N. Satofuka (Eds.), Parallel Comput. Dyn. Algorithms Results Using Adv. Comput., MAY, Capri, Italy, ISBN 9780444823274, 1996, pp. 368–375.
[33] S.J. Mosso, B.K. Swartz, D.B. Kothe, S.P. Clancy, Recent enhancements of volume tracking algorithms for irregular grids, Technical Report LA-UR-96-277, Los Alamos Natl. Lab., Los Alamos, NM, 1996, pp. 20–23.
[34] B.M. Ningegowda, B. Premachandran, Int. J. Heat Mass Transf. 79 (2014) 532–550.
[35] Z. Cao, D. Sun, J. Wei, B. Yu, Chem. Eng. Sci. 176 (2018) 560–579.
[36] J. López, J. Hernández, J. Comput. Phys. 227 (2008) 5939–5948.
[37] C.B. Ivey, P. Moin, in: Center Turb. Res. Ann. Res. Briefs, 2012, pp. 179–192.
[38] L. Jofre, O. Lehmkuhl, J. Castro, A. Oliva, Comput. Fluids 94 (2014) 14–29.
[39] L.C. Ngo, H.G. Choi, K. Chang, J. Mech. Sci. Technol. 35 (2021) 625–634.
[40] C.B. Ivey, P. Moin, J. Comput. Phys. 350 (2017) 387–419.
[41] J. Roenby, H. Bredmose, H. Jasak, R. Soc. Open Sci. 3 (2016) 160405.
[42] H. Scheufler, J. Roenby, J. Comput. Phys. 383 (2019) 1–23.
[43] T.J. Barth, P.O. Frederickson, in: 28th AIAA Aerosp. Sci. Meeting, 1990.
[44] B. Swartz, Math. Comput. 52 (1989) 675–714.
[45] J. López, J. Hernández, P. Gómez, F. Faura, J. Comput. Phys. 392 (2019) 666–693.
[46] P. Gómez, J. Hernández, J. López, Int. J. Numer. Methods Eng. 63 (2005) 1478–1512.
[47] P. Gómez, C. Zanzi, J. López, J. Hernández, J. Comput. Phys. 376 (2019) 478–507.
[48] J. López, J. Hernández, P. Gómez, F. Faura, J. Comput. Phys. 316 (2016) 338–359.
[49] R. Scardovelli, S. Zaleski, J. Comput. Phys. 164 (2000) 228–237.
[50] J. López, J. Hernández, P. Gómez, F. Faura, Comput. Phys. Commun. 223 (2018) 45–54.
[51] J. López, J. Hernández, P. Gómez, C. Zanzi, R. Zamora, Comput. Phys. Commun. 245 (2019) 106859.
[52] J. López, J. Hernández, P. Gómez, C. Zanzi, R. Zamora, Comput. Phys. Commun. 252 (2020) 107277.
[53] D.L. Youngs, An Interface Tracking Method for a 3D Eulerian Hydrodynamics Code, Technical Report 44/92/35, AWRE, 1984.
[54] J. López, A. Esteban, J. Hernández, P. Gómez, R. Zamora, C. Zanzi, F. Faura, J. Comput. Phys. 444 (2021) 110579.
[55] V. Dyadechko, M. Shashkov, Moment-of-Fluid Interface Reconstruction, LA-UR-05-7571, Los Alamos National Laboratory, 2005.
[56] R.V. Garimella, V. Dyadechko, B.K. Swartz, M. Shashkov, in: Proceedings of the 14th International Meshing Rountable, 2005, pp. 19–32.
[57] J. López, J. Hernández, isoap: A Software for Isosurface Extraction on Arbitrary Polyhedra, Mendeley Data, V1, 2021.
[58] The OpenFOAM Foundation, Openfoam, www.openfoam.org.
[59] Kitware, https://www.vtk.org.
[60] A. Henderson, J. Ahrens, C. Law, The `ParaView` Guide, 2004.
[61] L. Dagum, R. Menon, IEEE Comput. Sci. Eng. 5 (1998) 46–55.
[62] H. Si, A quality tetrahedral mesh generator and a 3D Delaunay triangulator, http://wias-berlin.de/software/tetgen/, v1.5.
[63] J. Roenby, H. Bredmose, H. Jasak, Data from: a computational method for sharp interface advection, dryad digital repository, https://doi.org/10.5061/dryad.66840.
[64] N. Max, J. Graph. Tools 4 (2) (1999) 1–6.
[65] S.S. Deshpande, L. Anumolu, M.F. Trujillo, Comput. Sci. Discov. 5 (2012) 014016.
[66] J.E. Pilliod, E.G. Puckett, Second-order volume-of-fluid algorithms for tracking material interfaces, Technical Report LBNL-40744, Lawrence Berkeley National Laboratory, 1997.
[67] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, J. Comput. Phys. 183 (2002) 83–116.
[68] J.B. Bell, P. Colella, H.M. Glaz, J. Comput. Phys. 85 (1989) 257–283.
[69] M.M. François, S.J. Cummins, E.D. Dendy, D.B. Kothe, J.M. Sicilian, M.W. Williams, J. Comput. Phys. 213 (2006) 141–173.
[70] LIS, Library of iterative solvers for linear systems, user guide, www.ssisc.org/lis.