A comparative study of two open-source state-of-the-art geometric VOF methods

Esteban, Adolfo; López, Joaquin; Gómez, Pablo; Zanzi, Claudio; Roenby, Johan; Hernández, Julio

Published in:
Computers Fluids

DOI:
10.1016/j.compfluid.2022.105725

Publication date:
2023

Document Version
Publisher's PDF, also known as Version of record

Citation for published version (APA):
Esteban, A., López, J., Gómez, P., Zanzi, C., Roenby, J., & Hernández, J. (2023). A comparative study of two open-source state-of-the-art geometric VOF methods. *Computers Fluids*, *250*(January), [105725]. https://doi.org/10.1016/j.compfluid.2022.105725

Computers and Fluids 250 (2023) 105725

Contents lists available at ScienceDirect

Computers and Fluids

journal homepage: www.elsevier.com/locate/compfluid

A comparative study of two open-source state-of-the-art geometric VOF methods

Adolfo Esteban [a], Joaquín López [b], Pablo Gómez [a], Claudio Zanzi [a], Johan Roenby [c], Julio Hernández [a,*]

[a] Dept. de Mecánica, ETSII, Universidad Nacional de Educación a Distancia (UNED), E-28040, Madrid, Spain
[b] Dept. de Ingeniería Mecánica, Materiales y Fabricación, ETSII, Universidad Politécnica de Cartagena, E-30202, Cartagena, Spain
[c] Dept. of Science and Environment, Roskilde University, DK-4000 Roskilde, Denmark

ARTICLE INFO

Dataset link: https://doi.org/10.5281/zenodo.7260888

Keywords:
Geometric volume of fluid methods
Unsplit interface advection methods
Interface reconstruction
Isosurface extraction

ABSTRACT

We present a systematic study of geometric volume of fluid (VOF) methods provided in the gVOF and TwoPhaseFlow packages, which include algorithms that are among the most accurate proposed in recent years. In addition to contributing to their further validation, the main purpose is to evaluate, in terms of accuracy and efficiency, the relative advantages of the advection and reconstruction algorithms used in the two packages (mainly, FMFPA-CLCIR and isoAdvector-plicRDF, respectively), and to investigate the suitability of combining them. Since TwoPhaseFlow is available in OpenFOAM, gVOF was also coupled with this open source CFD toolbox to maintain the same conditions in common solvers when obtaining and comparing their results, including discretization schemes, tolerances and meshes. For the same reason, identical computational resources were also maintained. The use of a common software and hardware framework that guarantees strictly the same simulation conditions overcomes many of the limitations and uncertainties of comparisons made in previous studies. Several reconstruction and advection tests are presented, showing the differences between the algorithms in terms of accuracy, as measured by several error norms, and in terms of efficiency, as measured by CPU times consumed. Simulations of the rise of a bubble and the impact of a drop on a pool were also performed, in which the VOF methods were coupled to the same solver of the Navier–Stokes equations, and the results obtained with the two combinations of algorithms FMFPA-CLCIR and isoAdvector-plicRDF are compared with each other and, in the case of the second test, with our own experimental results. The relative advantages and limitations of the analyzed algorithms are discussed, and it is suggested that a combination of isoAdvector for advection and CLCIR for reconstruction can provide a good compromise between accuracy and efficiency.

1. Introduction

Numerical simulation of free-surface and interfacial flows, which generally involves solving the Navier–Stokes equations together with an interface tracking method, is an active field of research. Different interface tracking methods have been developed over the years, but, in particular, the volume of fluid (VOF) method has received special attention due to several advantages, especially its good volume conservation capability compared to that of other methods. The various VOF methods developed have improved in accuracy and efficiency over time (see, e.g., [1–4]), and have been adapted for use on arbitrary convex or nonconvex meshes [5,6].

The open source CFD toolbox OpenFOAM [7] has become a widely used software for interfacial flow simulation due to its relative simplicity of use and its wide diffusion among the scientific community. Initially, OpenFOAM had only an algebraic VOF method implemented, based on the flux corrected transport technique, developed by Boris and Book [8] and later improved by Zalesak [9], and usually referred to as MULES because it also uses the multidimensional universal limiter with explicit solution to keep the volume fraction solution bounded. However, in recent years several authors have contributed to extending the capabilities of the software in this field of research. Albadawi et al. [10] implemented a coupled level-set volume of fluid (CLSVOF) method in OpenFOAM making use of the MULES advection scheme for volume

* Corresponding author.
E-mail addresses: aesteban@ind.uned.es (A. Esteban), joaquin.lopez@upct.es (J. López), pgomez@ind.uned.es (P. Gómez), czanzi@ind.uned.es (C. Zanzi), johan@ruc.dk (J. Roenby), jhernandez@ind.uned.es (J. Hernández).

https://doi.org/10.1016/j.compfluid.2022.105725
Received 27 July 2022; Received in revised form 29 October 2022; Accepted 7 November 2022
Available online 11 November 2022

fraction advection. Cifani et al. [11] implemented a piecewise linear interface calculation (PLIC)-based VOF method in OpenFOAM using the advection algorithm proposed by Puckett et al. [12]. They used the volume fraction gradient for interface orientation calculation along with the analytical method of Scardovelli and Zaleski [13] to solve the problem of forcing volume conservation in PLIC positioning, thus limiting the application of the VOF method to rectangular 2D and 3D meshes only. They also coupled the VOF method with the OpenFOAM solver of the Navier–Stokes equations, and conducted several advection tests and simulated a rising bubble problem. Roenby et al. [5] developed the isoAdvector method, which is based on an isosurface concept for interface reconstruction and volume fraction advection. They carried out several advection tests, obtaining good results in terms of volume conservation, boundedness and efficiency. They released the code as an OpenFOAM extension [14]. Dai and Tong [15] used the isoAdvector advection algorithm along with several analytical PLIC-positioning algorithms for 2D unstructured polygonal meshes, and then extended the algorithms to arbitrary convex polyhedral cells [16]. They performed interface reconstruction tests on single cells and simulated 2D and 3D incompressible multiphase flow problems.

Dianat et al. [17] implemented a CLSVOF method in OpenFOAM. They used the algorithms presented by Ahn and Shashkov [18] for interface reconstruction and the MULES algorithm of OpenFOAM for interface advection, while the interface orientation was calculated through the gradient of the level-set function. This methodology was validated for hexahedral and tetrahedral meshes using several advection tests and coupled with OpenFOAM's Navier–Stokes equations solver to simulate the impact of a drop on a solid surface. Following this work, Skarysz et al. [19] introduced an iterative PLIC-positioning method valid for convex cells, which is based on the decomposition into tetrahedral cells for the volume calculation of the truncated polyhedron. Haghshenas et al. [20] also implemented a CLSVOF method in OpenFOAM. More recently, Scheufler and Roenby [21] proposed, among other schemes, an iterative interface reconstruction scheme, plicRDF, based on reconstructing a distance function using the approach proposed by Cummins et al. [22], which significantly reduces reconstruction errors and improves the overall performance of the VOF method, achieving second-order convergence for CFL numbers below 0.2 for all mesh types. This approach was validated using several reconstruction and advection tests. The authors also released the code on the OpenFOAM platform [23]. On the other hand, the gVOF package [6] consists of several geometric VOF methods for arbitrary structured or unstructured meshes with convex or non-convex cells. The algorithms included are based on multidimensional unsplit advection and PLIC schemes. One of the purposes of the software is to facilitate the use of accurate and efficient geometric unsplit VOF methods in computational fluid dynamics codes.

All these methods and others published over the years are usually compared with each other on the basis of the results reported by the authors. In fact, the methods under study in the present work have already been compared in different papers with several other recent methods, and even with each other (see, e.g., [6]). Unfortunately, it is not common to find exactly the same conditions in the tests performed, resulting in a lack of consistency in the results to be compared. However, it is sometimes possible to perform the tests and directly obtain the results for the desired conditions since the corresponding codes are publicly available. This is the case in this work, in which we carry out a consistent and systematic comparison between different algorithms of the gVOF and VoFLibrary (included in TwoPhaseFlow [24,25]) packages, focusing on their accuracy and efficiency. We use the same computational resources in the tests performed with the different algorithms under comparison, including compiling optimizations flags, as well as all parameters, tolerances and discretization schemes in the common solvers used in the numerical simulations. In addition to achieving the main objective of carrying out a rigorous and systematic comparison of the considered methods as described above, the present work improves the completeness of the validation and comparison tests between algorithms of the gVOF and TwoPhaseFlow packages carried out in [6]. To this end, the advection and reconstruction algorithms under study are compared independently and not only coupled with each other; three different mesh types are considered; the performances of the methods under parallel execution are preliminarily compared; hydrodynamic tests, in addition to canonical tests with prescribed velocity field, are used to compare the results; and some of the tests presented here are more comprehensive than in [6] (e.g., regarding the analysis of the influence of the CFL number in the translation test or the three-dimensional character of the shearing flow test). On the other hand, the present work focuses, as a first step, only on the methods indicated above, although it would be desirable to extend it in the future to include other advanced advection and reconstruction methods, following the same strategy of implementing them in a common simulation framework. The computational aspects related to the implementation of the VOF methods considered in the OpenFOAM framework, as well as the meshes employed and the conditions used for testing, are presented in Section 2. In Section 3, we show several reconstruction tests, comparing the accuracy and computational time for each algorithm. Next, in Section 4, we show several advection tests with prescribed velocity fields. Finally, in Section 5, we present results for some incompressible flow problems, where the VOF methods are coupled to a Navier–Stokes equations solver.

2. Numerical methods

2.1. Computational details

Different algorithms of the two geometric VOF packages considered are compared in this work: the isoAdvector advection algorithm proposed by Roenby et al. [5] and the reconstruction algorithms described by Scheufler and Roenby [21], used in combination with isoAdvector, and several reconstruction and advection algorithms used in gVOF [6]. Since isoAdvector and the accompanying reconstruction algorithms have been developed in the OpenFOAM framework, gVOF has also been coupled within this framework for ease of comparison, as mentioned above. The gVOF package incorporates, among others, the reconstruction algorithms CLCIR, ELCIR, LLCIR (conservative, extended and local level-contour interface reconstruction, respectively) and LSGIR (least-squares gradient interface reconstruction), along with the EMFPA and FMFPA (edge-matched and face-matched flux polygons/polyhedra advection) algorithms, all of which are extensively described in [6,26,27]. The package uses the libraries VOFTools [28–30] and isoap [31] to perform geometric operations on arbitrary polyhedra. These libraries, as well as the advection and reconstruction algorithms of gVOF, are written in FORTRAN, and they are compiled together as a single shared library for use with OpenFOAM, which is written in C++. In addition, the OpenMP application programming interface is used to parallelize this library and improve computational efficiency on shared-memory architectures. The communication between the FORTRAN shared library and the OpenFOAM solvers is done through wrapper functions and several loops within the OpenFOAM code to update variables such as volume fraction, among others.

In the tests performed in this work, meshes of three different types and various resolutions are used. Hexahedral meshes are generated with the OpenFOAM's `blockMesh` utility. Unstructured tetrahedral meshes are generated with the open source tetGen v1.5 [32] mesh generator and then converted to the OpenFOAM mesh format using `tetgenToFoam`. For 2D cases, gmsh v4.4.1 and `gmshToFoam` are used to construct triangular meshes. Irregular polyhedral meshes are generated using OpenFOAM's `polyDualMesh` tool, starting from a tetrahedral mesh previously obtained with tetGen v1.5. Since, for a given cell, its resulting faces may not necessarily be planar, every face is triangulated by joining its vertex centroid with two consecutive face vertices. To maintain the same conditions in all simulations, and for

the sake of consistency when comparing results, we have not used any procedure to selectively check if a given face is planar or not. For 2D polygonal meshes, gmsh v4.4.1 is used along with the `polyDualMesh` and `extrudeMesh` tools and, since the resulting cells are convex planar polygons, no triangulation is needed. Table A.1 shows the characteristics of the 3D meshes considered in the advection and reconstruction tests. The size of hexahedral meshes is given by the number of cells in each spatial direction, n, and therefore the total number of cells is calculated as n^3. However, the total number of cells of tetrahedral and irregular polyhedral meshes is approximately equal to that of their equivalent hexahedral meshes.

In the drop impact test of Section 5.2 we use an octree graded, statically refined mesh generated with `snappyHexMesh`, which consists of a root mesh that is iteratively refined to a certain maximum refinement level in predefined regions. Different regions with four refinement levels are superimposed on the root mesh, as described below, keeping the same finest resolution near the interface.

It should be noted that the accuracy of the volume fraction initialization method determines to a considerable extent the accuracy of the test results. The initialization error is defined as the difference between the volume enclosed by the exact interface and the initialized volume. To maintain consistency in comparisons by ensuring the same initialization error, we use in all tests the volume fraction initialization procedure described in [33] and extended to arbitrary convex and nonconvex cells in [29,34], respectively. We have found that this procedure yields an error significantly smaller than OpenFOAM's `setAlphaField` tool, although the latter is usually faster when the simulation is performed on a single core.

In all reconstruction and advection tests with prescribed velocity field, a tolerance for the volume fraction of 10^{-12} is used (a cell is considered to be an interfacial cell if $10^{-12} < F < 1 - 10^{-12}$, where F is the volume fraction). When also solving the Navier–Stokes equations, the tolerance value for the volume fraction is set to 10^{-8} in order to reduce the effects of the velocity field not being exactly divergence-free. All simulations were performed on an Intel Xeon W with 96 GB of DDR4-2500 MHz and 28 cores. For the distributed memory parallelization with Message Passing Interface (MPI) used by OpenFOAM and TwoPhaseFlow, computation times are highly dependent on the domain decomposition used in the parallelization, and therefore all reconstruction and advection tests where computation time is measured are performed on a single core. For non-prescribed velocity tests, all available cores are used in all simulations, with the corresponding parallelization algorithm used for each method, as mentioned above, resulting in less favorable execution conditions for the gVOF algorithms and thus lower speedup.

At the current development stage, isoAdvector-plicRDF is available with OpenMPI via OpenFOAM to run on thousands of cores, whereas CLCIR and FMFPA are currently limited in scaling by the shared memory size of OpenMP systems.

2.2. Coupling with the Navier–Stokes equations

We consider viscous, incompressible, unsteady flows of two immiscible fluids with constant, uniform properties separated by an interface, for which the governing equations can be written as

$$\nabla \cdot \mathbf{u} = 0, \tag{1}$$

$$\frac{\partial(\rho\mathbf{u})}{\partial t} + \nabla \cdot (\rho\mathbf{u}\mathbf{u}) = -\nabla p^* - \mathbf{g} \cdot \mathbf{x}\nabla\rho + \nabla \cdot \{\mu[\nabla\mathbf{u} + (\nabla\mathbf{u})^{\mathrm{T}}]\} + \mathbf{f}_v, \tag{2}$$

where $\mathbf{u}$ is the velocity vector, ρ the density, μ the dynamic viscosity, $\mathbf{g}$ the gravity vector, $\mathbf{x}$ the position vector, $p^* = p - \rho\mathbf{g} \cdot \mathbf{x}$ the modified pressure, p the pressure, and $\mathbf{f}_v$ any body force per unit volume. The evolution of the interface is described by the VOF approach, using any of the different methods studied in this work. The fluid properties are calculated as

$$\rho = \chi\rho_l + (1-\chi)\rho_g, \quad \mu = \chi\mu_l + (1-\chi)\mu_g, \tag{3}$$

where the l and g subscripts denote liquid and gas, respectively, and χ is an indicator function, which is continuous everywhere except at the interface, where it jumps from zero to one and whose evolution is described by the advection equation

$$\frac{\partial\chi}{\partial t} + \nabla \cdot (\chi\mathbf{u}) = 0. \tag{4}$$

In the tests of Section 4, this equation will be solved with different prescribed velocity fields.

The continuum surface force (CSF) method [35] is used to reproduce the effect of the surface tension by including in $\mathbf{f}_v$ the body force per unit volume

$$\mathbf{f}_\sigma = \sigma\kappa\nabla F, \tag{5}$$

where F is the discretized value of χ over cell Ω,

$$F = \frac{1}{V_\Omega}\int_\Omega \chi \,\mathrm{d}V, \tag{6}$$

σ the surface tension coefficient, and κ the interface curvature, defined as

$$\kappa = -\nabla \cdot \left(\frac{\nabla F}{|\nabla F|}\right) \tag{7}$$

(the minus sign is a convention to make the interface normal point from the $F = 1$ fluid to the $F = 0$ fluid).

A common feature of the interface advection algorithms used in isoAdvector and gVOF (the main ones in the latter being the EMFPA and FMFPA mentioned above) is that they use face-centered velocities for volume fraction advection, which are obtained from the volumetric fluxes through the faces. In addition, the EMFPA and FMFPA algorithms require the use of cell-vertex velocities, which are obtained from interpolation from the cell faces using the inverse distance weighting method.

The input data for the tests performed and the data corresponding to most of the figures and tables presented in the paper, along with links to the versions of the gVOF and TwoPhaseFlow packages and to programs to generate the non-hexahedral meshes used in the simulations are publicly available in [36].

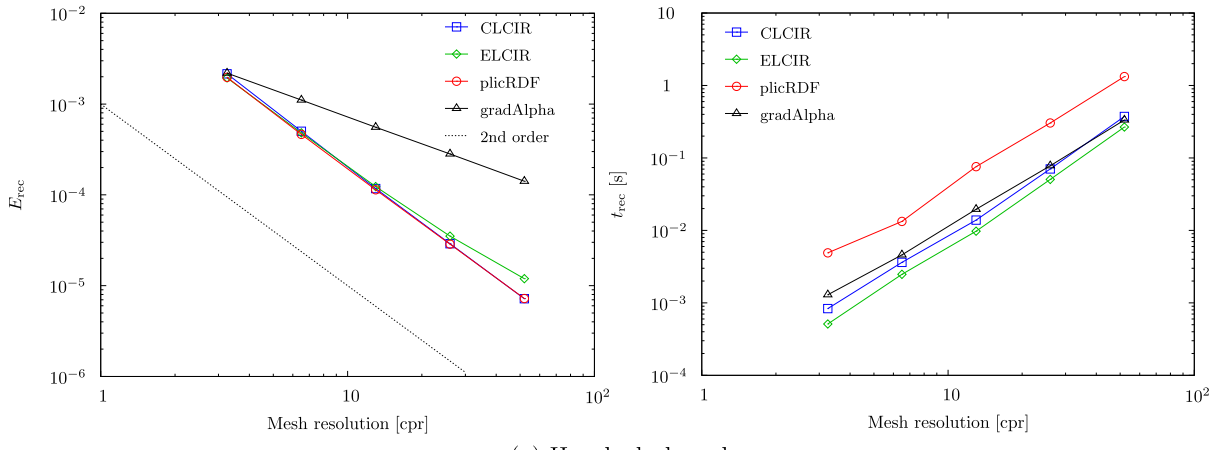
3. Reconstruction tests

To compare the accuracy of the reconstruction procedures, the following error, defined as the volume between the exact interface and its approximate representation, is used:

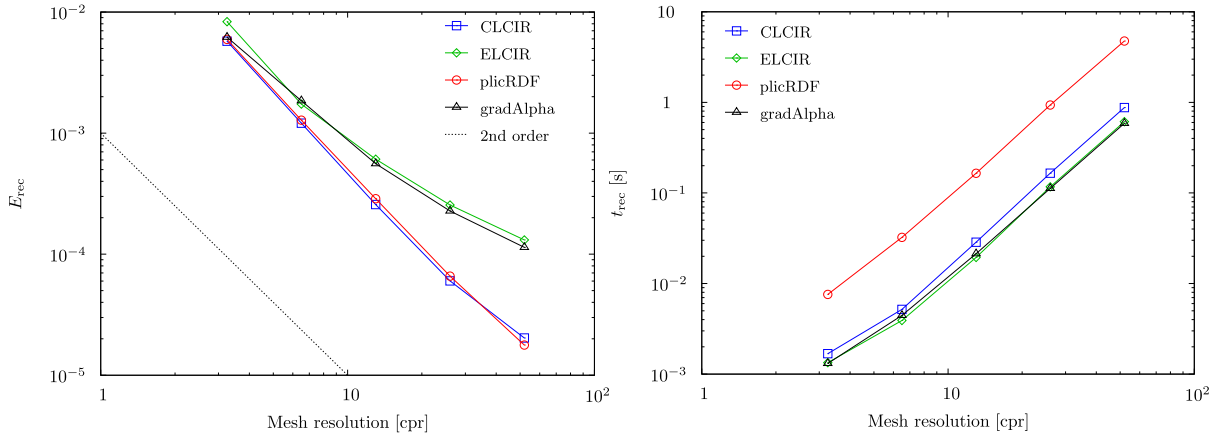
$$E_{\mathrm{rec}} = \sum_i (V_{i,\mathrm{exact}} \setminus V_{i,\mathrm{rec}}) \cup (V_{i,\mathrm{rec}} \setminus V_{i,\mathrm{exact}}), \tag{8}$$

where $V_{i,\mathrm{exact}}, V_{i,\mathrm{rec}} \in \mathbb{R}^3$ are the exact and reconstructed liquid volumes in cell i, respectively, and the sum is over all cells. The evaluation of Eq. (8) is performed using the VOFTools subroutine `initf3d` [34], called from the gVOF subroutine `recerr` [6], which gives a precise value of E_{rec} when a very accurate volume fraction initialization procedure is used to calculate $V_{i,\mathrm{exact}}$, such as that described in [33] and used in the present work. For the sake of clarity in the presentation of the results, the reconstruction algorithms compared in this section are only CLCIR and ELCIR [27], and gradAlpha and plicRDF [21], which yield results that are among the best in terms of combined accuracy and efficiency provided by the algorithms included in gVOF and TwoPhaseFlow, respectively. The plicRDF algorithm determines how many iterations, up to a maximum set at 5, it needs to reach convergence using a criterion based on the calculated residuals. To compare the computational efficiency of the algorithms, we also measure the CPU time consumed in the interface reconstruction, t_{rec}.

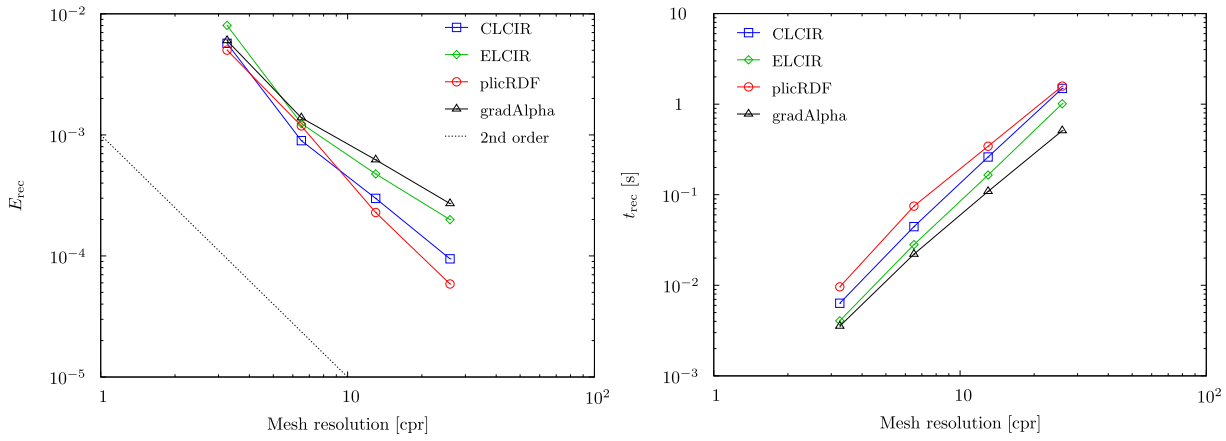
A sphere of radius 0.325 is reconstructed in a unit domain. To avoid artificial regularity of the results due to mesh dependence, the sphere is centered at (0.525, 0.464, 0.516) [27]. Fig. 1 shows the reconstruction error and the CPU time consumed for the algorithms



(a) Hexahedral meshes



(b) Tetrahedral meshes



(c) Irregular polyhedral meshes

Fig. 1. Reconstruction error, E_{rec}, and CPU time consumed, t_{rec}, obtained in the sphere reconstruction test using different mesh types and sizes.

considered as a function of mesh resolution (measured in cells per sphere radius, cpr), obtained with hexahedral, tetrahedral and irregular polyhedral meshes (note that volume errors need not be normalized when using a unit domain). The number of cells of the unstructured meshes is approximately equal to that of the corresponding hexahedral meshes. As shown in Fig. 1(a), for hexahedral meshes the CLCIR and plicRDF algorithms show the best results in terms of reconstruction error and convergence, which are practically coincident with each other for all mesh resolutions. ELCIR shows a second-order convergence and reconstruction errors similar to those of CLCIR and plicRDF for low and medium mesh resolutions, while gradAlpha has a first-order convergence, showing the largest E_{rec} values. In terms of computational efficiency, plicRDF shows the largest values for the CPU time consumed, an order of magnitude higher than those obtained with the CLCIR algorithm for almost all mesh resolutions. The gradAlpha algorithm gives a similar reconstruction time to CLCIR for high mesh resolutions, although at low mesh resolutions it is 35% longer, and ELCIR gives the lowest t_{rec}.

For tetrahedral meshes (Fig. 1(b)), CLCIR and plicRDF give almost identical reconstruction errors. ELCIR and gradAlpha give very similar error and order of convergence, the latter lower than the remaining two algorithms. In this case, ELCIR also takes the shortest time to reconstruct the interface for all mesh resolutions, while plicRDF requires the longest times, again an order of magnitude greater than those required by CLCIR.

On irregular polyhedral meshes (Fig. 1(c)), CLCIR and plicRDF give similar results for low mesh resolutions, although for higher resolutions plicRDF shows lower reconstruction errors than CLCIR due to the reduction in convergence order of the latter. gradAlpha is now the fastest method, although it gives the largest errors for medium-high mesh resolutions. CLCIR requires slightly less reconstruction time than plicRDF for the finest mesh and, although for coarser meshes CLCIR is faster than plicRDF, the large difference shown for the other mesh types is now reduced to only 35%.

The above comparisons have been repeated for the hollow sphere reconstruction test of Liovic et al. [37], giving rise to practically identical trends in results and conclusions.

In an analysis performed to identify the possible causes that make plicRDF significantly slower than CLCIR, it has been found that the most time consuming step in plicRDF is the resolution of the volume conservation enforcement (VCE) problem to position the PLIC from the volume fraction value in the cell. Although, due to the iterative nature of plicRDF, it is difficult to compare in a straightforward manner the performance of the methods used to solve the VCE problem by plicRDF and CLCIR, a preliminary study of CPU time consumption has been carried out. For this purpose, we used the 3D deformation flow test described in the next section, in which the interface normal in plicDRDF is estimated from the previous time step (note that, in a reconstruction test, the lack of a proper estimation of the interface normal would penalize this method with an even greater increase in CPU time consumed). It has been found that, for example, for a hexahedral mesh of size $n = 64$, the method used by plicRDF to solve the VCE problem consumes about 60% of t_{rec}, while the methods used by CLCIR, i.e., the one proposed in [13] for hexahedral meshes and CIBRAVE [38] for meshes of other types, only consume about 6% and 28%, respectively. This is mainly because the CPU time consumed by the method used by plicRDF for the VCE problem is about 85 and 14 times (taking into account the average number of iterations in plicRDF) higher than those consumed by the method proposed in [13] and CIBRAVE, respectively. Although the use of the latter methods in plicRDF would reduce t_{rec} by a factor of about 2.5, plicRDF would still consume about 3 times more CPU time than CLCIR. The above figures may obviously change depending on the type and size of the mesh and the type of test.

4. Advection tests

In this section, three pure advection tests are used to evaluate the relative accuracy and efficiency of the algorithms under analysis. Among the gVOF algorithms, FMFPA is chosen as the advection algorithm in the comparison due to its better computational efficiency, although EMFPA could be used for better accuracy in certain situations. The differences in CPU time consumed by EMFPA and FMFPA are due to the larger number of truncation operations and the longer time spent by the flux polyhedra construction step required by the former (e.g., in the 3D deformation flow test presented below for a hexahedral mesh of size $n = 64$, truncation operations and flux polyhedra construction in EMFPA are 1.9 and 1.7 times, respectively, slower than in FMFPA, whereas differences in accuracy are only about 2% for E_g). For the reconstruction step, the CLCIR algorithm is used due to the better overall performance shown in the previous section. In combination with the isoAdvector advection scheme, the plicRDF reconstruction algorithm is chosen due to its better results compared to those of the remaining algorithms analyzed by Scheufler and Roenby [21], as reported by these authors.

The geometric error is estimated with an L_1 error norm defined as

$$E_g = \sum_i V_i |F_i - F_{e,i}|, \tag{9}$$

where V_i is the volume of cell i, and F_i and $F_{e,i}$ are the calculated and exact volume fractions, respectively, at the final instant of the test, $t = T$. The order of convergence can be determined from

$$\mathcal{O} = \frac{\ln[E_g(2n)/E_g(n)]}{\ln(1/2)}, \tag{10}$$

where $E_g(n)$ and $E_g(2n)$ are the errors obtained using meshes with n^3 and $(2n)^3$ cells, respectively.

The volume conservation error is quantified as

$$E_{\text{vol}} = |\sum_i V_i \left(F_i - F_{e,i}\right)|, \tag{11}$$

where the volume fractions are obtained, again, at instant $t = T$. The boundedness of the solution for the fluid volume fraction at instant t is measured by the error

$$E^{(t)}_{\text{bound}} = \max\left\{E^{(t-\Delta t)}_{\text{bound}}, \max\left[|\min_i\left(V_i\, F_i^{(t)}\right)|, \max_i\left(V_i(F_i^{(t)} - 1)\right)\right]\right\}, \tag{12}$$

where Δt is the time step, and $F_i^{(t)}$ the calculated volume fraction in cell i at instant t. This definition allows us to estimate the unboundedness of the solution throughout the entire simulation time.

To compare the computational efficiency of the algorithms, we measure the advection time, t_{adv}, defined as the average CPU time consumed per advection step, and the reconstruction time introduced in Section 3, t_{rec}, which is now defined as the average CPU time consumed per reconstruction step. We also define the total CPU time per time step, t_{tot}, as the summation of the advection and reconstruction times.

An essential point when comparing the performance of different advection algorithms is the need to maintain the same criteria for the choice of the time step depending on how the CFL number is defined. In OpenFOAM, the CFL number is calculated as

$$\text{CFL} = \Delta t \max_i\left(\frac{1}{2}\frac{\sum_f |\mathbf{u}_{f,i}\cdot\mathbf{S}_{f,i}|}{V_i}\right), \tag{13}$$

where the term $\frac{1}{2}\sum_f |\mathbf{u}_{f,i}\cdot\mathbf{S}_{f,i}|$ is the volumetric flux through cell i such that CFL = 1 corresponds to a situation where the cell volume, V_i, is exactly replaced for the time step Δt. A first approach is to specify a maximum value of the CFL number and calculate an adaptive time step from Eq. (13), then using a procedure that avoids unstable oscillations by limiting the increase of each new time step.

Instead of calculating a value of the quantity in parentheses of Eq. (13) for each computational cell and then selecting the maximum value obtained over the entire domain, another possible approach would be to define a CFL number as follows:

$$\text{CFL} = \Delta t \max\left[\frac{\max_f |u|}{\min_i(\Delta x)}, \frac{\max_f |v|}{\min_i(\Delta y)}, \frac{\max_f |w|}{\min_i(\Delta z)}\right], \tag{14}$$

where u, v, and w are the components of the velocity vector at face f, $\mathbf{u}_f$, and $\Delta x = x_{\max} - x_{\min}$, $\Delta y = y_{\max} - y_{\min}$, and $\Delta z = z_{\max} - z_{\min}$ are the maximum cell sizes in the x, y, and z directions, respectively, in a Cartesian coordinate system (note that the mesh is not required to be aligned with the reference system). Note that the face for which $|u|$ is maximum may not correspond to the cell for which Δx is minimum, just as it may occur in the two other directions. The new time step would be calculated in this case from Eq. (14), also subsequently limiting the increase of each new time step.

Both approaches may give similar time step sizes under certain conditions, e.g., when there is only one non-zero velocity component in a hexahedral mesh, but, in general, Eq. (13) is more restrictive due to the flux averaging, yielding smaller time step sizes than Eq. (14). The question then arises as to which criterion would be preferable to

use. The alternative option of using a non-adaptive but fixed time step would not be consistent between different tests, as it would imply very different CFL numbers for different velocity fields, and therefore we opt to use an adaptive time step, with a prefixed maximum value of CFL. Furthermore, in all advection tests in this section the CFL limiting time step criterion will be applied only in cells close to the interface (parameter `maxAlphaCo` in OpenFOAM). Finally, since isoAdvector convergence is only first order for high CFL numbers [21], we choose to use the more restrictive OpenFOAM definition of CFL given by Eq. (13). Keep in mind, however, that this choice may contribute to overlooking isoAdvector's limitations in dealing with moderately high CFL numbers. Nevertheless, these are not related with the isoAdvector advection concept but to certain situations in which during a time step the interface enters a cell that was not initially labeled as an interface cell. Work on this issue is in progress [39].

The way the CFL number is defined must be emphasized when analyzing the performance of the advection algorithms of VOF-type methods. Depending on the velocity field used in each test, the differences in the errors obtained due to differences in the time step sizes obtained with various criteria can be almost an order of magnitude, which clearly highlights a lack of consistency in some comparisons between advection algorithms reported in the literature.

In the isoAdvector method, we set the parameter `nAlphaBounds` equal to 5, thereby redistributing fluid from cells with out-of-bounds volume fractions to preserve volume conservation [5]. This volume redistribution is accomplished by applying a conservative bounding step five times per advection–reconstruction step. Besides, isoAdvector offers the possibility to use a non-conservative bounding step to force the volume fraction value of unbounded cells, but we prefer to keep this option disabled. In FMFPA and especially in EMFPA, the way the flux polyhedra are constructed results in greatly reducing the over/underlap between them, so instead of using special algorithms to redistribute the very small fluid volumes that cause the volume fraction to be outside the 0 and 1 bounds, the volume fraction is simply adjusted by making $F \leftarrow \max[\min(F, 1.0), 0.0]$ [6].

4.1. Simple translation

The main purpose for doing this simple test was to analyze the influence of the CFL number on the algorithms accuracy. A sphere of radius 0.25 centered at (0.25, 0.25, 0.25) is translated in a steady, uniform flow with velocity components (1, 1, 1) for time $T = 0.5$, in a cubic domain of size $1 \times 1 \times 1$. Fig. 2 shows the geometric error obtained with FMFPA-CLCIR and isoAdvector-plicRDF on hexahedral meshes of different sizes, as a function of CFL number. At high CFL numbers, the error obtained with isoAdvector-plicRDF is significantly larger than that obtained with the FMFPA-CLCIR algorithms for all mesh sizes, with the difference increasing as the mesh resolution increases. As the CFL number decreases, the error tends to a constant value for each method, dependent on the mesh resolution, as indicated by Harvie and Fletcher [40]. For $n = 10$ and $n = 20$, the error provided by FMFPA-CLCIR at low CFL numbers is 11%–14% lower than that obtained with isoAdvector-plicRDF whereas for $n = 40$ the error of the latter is 11% lower than that of the former. For the finest mesh resolution, both algorithm combinations give similar geometric error.

It should be noted that the definition of the CFL number given by Eq. (13) used in this work results in higher errors with the gVOF algorithms than those reported, for example, in [31], where the definition of Eq. (14) was used. This is because the latter yields larger time step sizes, for which the geometric error obtained with the gVOF algorithms is reduced. This trend in the geometric error was explained by Harvie and Fletcher [40], who argued that, as the time step size decreases and more time steps are required to complete a simulation of given duration, more interface reconstruction steps (one per time step) are performed. Since each reconstruction step introduces a discrete amount of error, if the number of reconstructions is increased so does the total error introduced in the simulation. On the other hand, the error obtained with the isoAdvector algorithm does not follow this trend. On the contrary, it decreases with decreasing time step size, which suggests that the error introduced by the advection step dominates over the reconstruction error for high CFL numbers.

Fig. 3 shows the L_1 error norm obtained for different mesh types and three CFL numbers as a function of mesh resolution. Fig. 3(a) shows the error obtained on hexahedral meshes, for which the differences between the two algorithm combinations considered are more evident. The convergence rate of isoAdvector-plicRDF increases as the CFL number decreases, reaching second order for CFL = 0.1 over the entire range of mesh resolutions represented, whereas the convergence of FMFPA-CLCIR is of second order for all mesh resolutions and CFL numbers considered. These differences in convergence rates on hexahedral meshes translate into large differences in error for high mesh resolutions and higher CFL numbers in the range considered, of about two orders of magnitude for CFL = 1. For tetrahedral meshes (Fig. 3(b)), the results are less dependent on the CFL number, and both algorithm combinations show second-order convergence over the entire range of CFL numbers. However, for low CFL numbers at high mesh resolutions, FMFPA-CLCIR yields higher errors than those obtained with isoAdvector-plicRDF. For irregular polyhedral meshes (Fig. 3(c)), both algorithm combinations give very similar errors for low CFL numbers and high mesh resolutions, while for the coarsest mesh FMFPA-CLCIR yields slightly lower errors. For CFL = 1, isoAdvector-plicRDF gives a lower error for the coarsest mesh, while, as the mesh resolution increases, FMFPA-CLCIR gives lower errors.

4.2. 3D deformation flow

The well-known benchmark test introduced by Enright et al. [41] consists of a sphere of radius 0.15 initially centered at (0.35 0.35 0.35) within a unit cubic domain, which deforms in a solenoidal velocity field given by

$$\begin{aligned} u(x,y,z,t) &= 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos(\pi t/T), \\ v(x,y,z,t) &= -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)\cos(\pi t/T), \\ w(x,y,z,t) &= -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos(\pi t/T), \end{aligned} \tag{15}$$

where a period $T = 3$ is used.

Table 1 shows the errors and CPU times measured for the two algorithm combinations compared in the test. The CPU times provided, $\tilde{t}_{\text{adv}}$, $\tilde{t}_{\text{rec}}$ and $\tilde{t}_{\text{tot}}$, are relative to the lowest total time, which in this case is $t_{\text{tot}} = 23.9$ ms, obtained with the isoAdvector-plicRDF method on a hexahedral mesh with $n = 32$. For all mesh types and sizes, the L_1 error norm obtained with FMFPA-CLCIR is always lower than that obtained with isoAdvector-plicRDF except for the coarsest polyhedral mesh, for which isoAdvector-plicRDF yields a 3.7% lower error. The differences for the other mesh types and sizes range from the 4.9% lower error obtained with FMFPA-CLCIR for tetrahedral meshes with $n = 32$ to the 56.3% lower error also obtained with FMFPA-CLCIR for hexahedral meshes with $n = 256$. For hexahedral and tetrahedral meshes, as the mesh resolution increases, so do the differences in geometric error due to the higher order of convergence of FMFPA-CLCIR, although both algorithm combinations achieve second-order convergence. For irregular polyhedral meshes, the differences between the two combinations considered decrease, with their convergence rates becoming almost equal for $n = 128$. This behavior might be attributed to a better performance of the reconstruction step of isoAdvector-plicRDF on this type of mesh.

The errors in volume conservation for hexahedral meshes are around $10^{-13} - 10^{-15}$ for isoAdvector-plicRDF while for FMFPA-CLCIR are close to machine precision $10^{-14} - 10^{-16}$. For the other two mesh types, FMFPA-CLCIR yields errors that, although small ($\lesssim 10^{-6}$), are several orders of magnitude larger than those obtained with isoAdvector-plicRDF ($\lesssim 10^{-12}$). The boundedness error obtained with

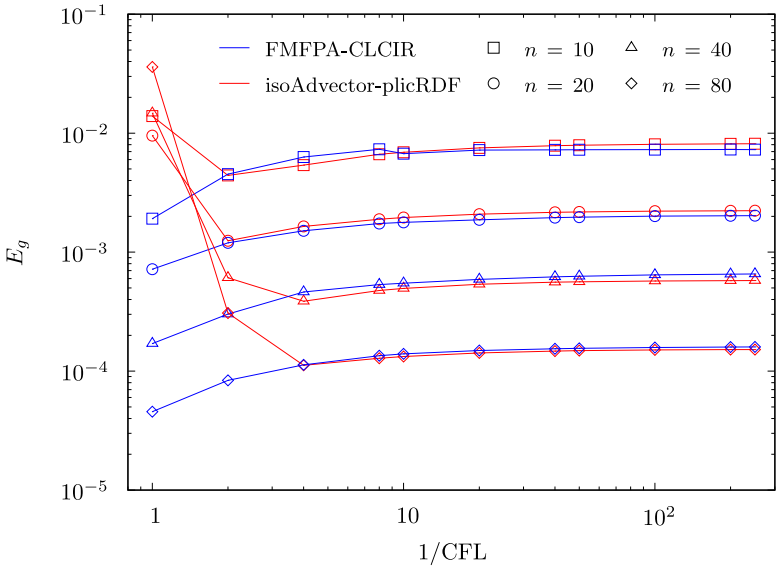
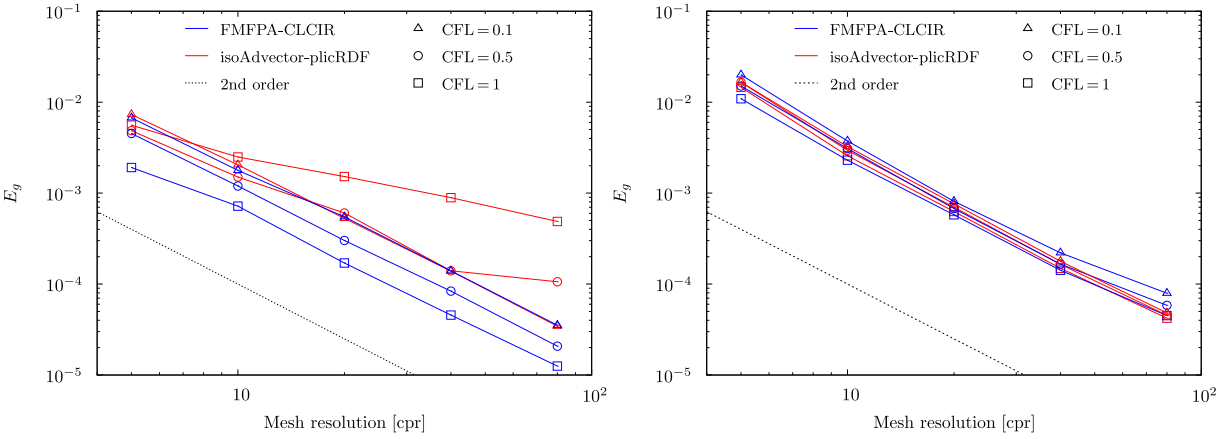
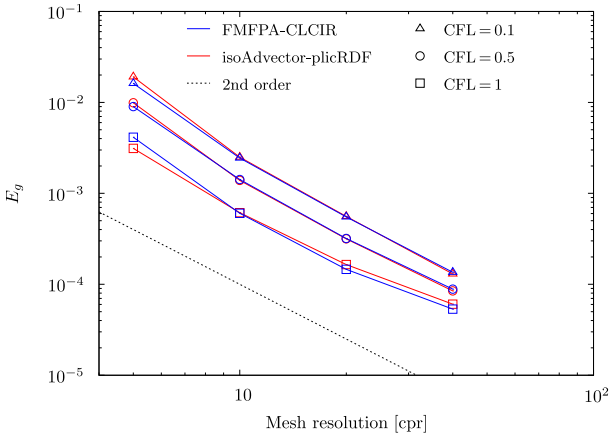


Fig. 2. L_1 error norm obtained in the translation test using hexahedral meshes of four different sizes, as a function of the CFL number.



(a) Hexahedral meshes

(b) Tetrahedral meshes



(c) Irregular polyhedral meshes

Fig. 3. L_1 error norm obtained in the translation test on three different mesh types with three different CFL numbers, as a function of mesh resolution.

Table 1
Errors and CPU times (relative to the lowest $t_{tot} = 23.9$ ms) obtained in the 3D deformation flow test using a CFL = 0.5 and different mesh types and resolutions.

Algorithms	n	E_g	$\mathcal{O}$	E_{vol}	E_{bound}	$\tilde{t}_{tot}$	$\tilde{t}_{adv}$	$\tilde{t}_{rec}$
Hexahedral meshes								
FMFPA-CLCIR	32	6.49e−03	–	5.20e−17	1.27e−18	1.94e+00	1.78e+00	1.55e−01
isoAdvector-plicRDF	32	7.97e−03	–	8.73e−16	2.17e−17	1.00e+00	1.82e−01	8.18e−01
FMFPA-CLCIR	64	2.07e−03	1.65	7.98e−17	8.01e−19	9.90e+00	9.16e+00	7.46e−01
isoAdvector-plicRDF	64	3.02e−03	1.40	5.94e−15	2.01e−09	5.57e+00	1.43e+00	4.14e+00
FMFPA-CLCIR	128	4.31e−04	2.26	4.16e−17	5.55e−19	5.68e+01	5.32e+01	3.66e+00
isoAdvector-plicRDF	128	7.06e−04	2.10	1.88e−14	1.60e−10	3.56e+01	1.26e+01	2.31e+01
FMFPA-CLCIR	256	6.13e−05	2.81	9.28e−15	2.93e−19	3.75e+02	3.53e+02	2.19e+01
isoAdvector-plicRDF	256	1.06e−04	2.74	1.72e−13	1.92e−11	2.41e+02	1.08e+02	1.33e+02
Tetrahedral meshes								
FMFPA-CLCIR	32	1.33e−02	–	6.98e−06	1.12e−08	1.76e+01	1.70e+01	5.37e−01
isoAdvector-plicRDF	32	1.39e−02	–	1.30e−15	4.89e−06	1.88e+00	2.29e−01	1.65e+00
FMFPA-CLCIR	64	4.23e−03	1.65	3.29e−06	3.05e−10	7.79e+01	7.51e+01	2.83e+00
isoAdvector-plicRDF	64	5.86e−03	1.25	2.48e−15	1.18e−06	1.97e+01	2.90e+00	1.68e+01
FMFPA-CLCIR	128	9.39e−04	2.17	3.35e−07	2.11e−10	3.97e+02	3.80e+02	1.68e+01
isoAdvector-plicRDF	128	1.46e−03	2.00	4.67e−15	1.26e−07	1.26e+02	2.34e+01	1.03e+02
FMFPA-CLCIR	256	1.58e−04	2.57	2.16e−07	9.49e−11	2.69e+03	2.56e+03	1.32e+02
isoAdvector-plicRDF	256	3.62e−04	2.02	2.62e−15	1.87e−08	1.12e+03	2.45e+02	8.77e+02
Irregular polyhedral meshes								
FMFPA-CLCIR	32	1.05e−02	–	2.52e−07	8.87e−09	5.67e+01	5.4e+01	2.71e+00
isoAdvector-plicRDF	32	1.01e−02	–	6.52e−16	4.48e−07	9.12e+00	3.02e+00	6.10e+00
FMFPA-CLCIR	64	3.57e−03	1.56	3.12e−09	4.47e−09	2.80e+02	2.59e+02	2.05e+01
isoAdvector-plicRDF	64	4.17e−03	1.28	1.63e−16	5.49e−08	1.62e+02	3.90e+01	1.23e+02
FMFPA-CLCIR	128	6.23e−04	2.52	4.32e−08	2.38e−08	1.49e+03	1.35e+03	1.41e+02
isoAdvector-plicRDF	128	6.95e−04	2.59	2.12e−16	8.78e−09	8.02e+02	2.75e+02	5.26e+02

FMFPA-CLCIR is several orders of magnitude lower than that obtained with isoAdvector-plicRDF except for $n = 128$. In terms of computational efficiency, isoAdvector-plicRDF consumes less total CPU time, $\tilde{t}_{tot}$, than FMFPA-CLCIR for the three types of meshes. The differences in $\tilde{t}_{tot}$ between the two algorithm combinations decrease with increasing mesh resolution for tetrahedral meshes and, albeit only slightly, for hexahedral meshes, while for irregular polyhedral meshes they tend to decrease with mesh resolution up to a certain level of refinement, and then increase slightly with further refinement. On hexahedral meshes, the time consumed by advection is of the same order for both algorithm combinations for the finest meshes and up to one order of magnitude higher for FMFPA-CLCIR for the coarsest grid, whereas the time consumed by the reconstruction step is much less for FMFPA-CLCIR, probably due to the iterative nature of the plicRDF algorithm. These trends are similar, although more pronounced for polyhedral and, particularly, tetrahedral meshes.

The large increase in CPU time consumed by the FMFPA algorithm on tetrahedral and irregular polyhedral meshes is mainly due to the face advection flux calculation procedure, although the reasons are different for each mesh type. In this algorithm, fluid advection is performed using face flux polyhedra, which are first constructed using the interpolated velocities at the cell nodes and then truncated with the reconstructed interface at the cell and at the neighboring cells surrounding the face. Therefore, if the number of neighboring cells containing the interface increases, the number of truncation operations will also increase. For an internal tetrahedral cell, the average number of neighbor cells is between 74 and 78 for the mesh sizes considered in this test (see Table A.1), while for a hexahedral cell this number is only 26 and does not depend on size. Therefore, as the number of interfacial cells increases with mesh refinement, so does the number of truncations and, consequently, the advection time. Conversely, since isoAdvector does not perform such truncation operations for the calculation of fluxes at faces, the time consumed by the advection step is considerably less. This is the main advantage and strength of isoAdvector.

For the irregular polyhedral meshes considered in this test, the average number of neighbor cells per internal cell is 15, whereas the average number of faces and points per internal cell varies from 40.8 to 41.4 and from 77.7 to 78.8, respectively, which is a large increase compared to hexahedral ($\overline{n}_{f,in} = 6$ and $\overline{n}_{p,in} = 8$) or tetrahedral meshes ($\overline{n}_{f,in} = \overline{n}_{p,in} = 4$). Then, in a polyhedral cell, the number of truncations of the polyhedron of a given face is substantially less than in the other two types of meshes since the number of neighbor cells is considerably less than in the latter, but, as the number of flux polyhedra is highly increased due to the large number of faces, the total number of truncation operations is increased. This increase in the average number of faces per polyhedral cell also increases the time consumed by the isoAdvector algorithm, since it is based on the calculation of the area described by the interface-face intersection movement within the considered time interval.

Figs. 4 and 5 show the PLIC interfaces obtained with the two algorithm combinations under comparison for the intermediate ($t = 1.5$) and final ($t = 3$) instants of the test, where the differences in the geometric error shown in Table 1 are difficult to see with the naked eye.

Although it is difficult to compare the performance of the algorithms in parallel execution due to the use of OpenMP in gVOF and MPI in TwoPhaseFlow, the reduction in CPU time consumed by the reconstruction and advection algorithms and the resulting variations in errors can at least be evaluated separately for each parallelization interface. For this purpose, we performed simulations in which the number of threads (gVOF) or processors (TwoPhaseFlow) was increased from 1 to 20. In the case of isoAdvector-plicRDF, we used a simple geometric decomposition of the domain of the form $(n_{proc,x}, n_{proc,y}, n_{proc,z})$, where $n_{proc,m}$ is the number of processors for direction m, which was kept in all simulations as $(1, 1, n_{proc})$. Fig. 6 shows an example of the results for the CPU time consumed and speedup as a function of the number of threads or processors, obtained for this test with a hexahedral mesh of size $n = 128$. Note that the speedup is very similar for the advection and reconstruction steps in FMFPA-CLCIR up to 6 threads, and above this value the reconstruction step scales faster up to 20 threads, reaching a maximum speedup of about 7. For isoAdvector-plicRDF, the advection step always has a better speedup compared to the reconstruction step.

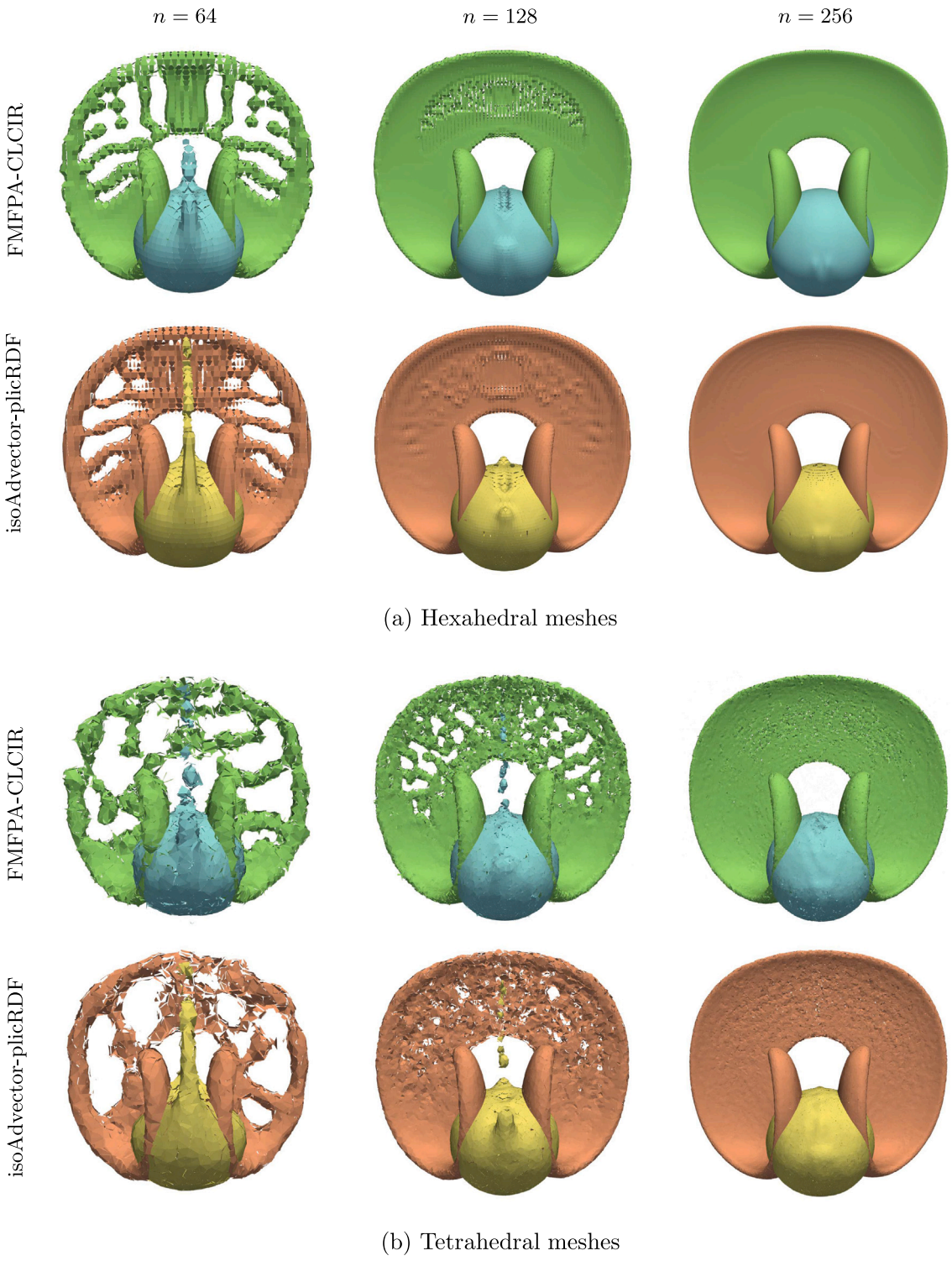


Fig. 4. PLIC interfaces for the 3D deformation flow test at $t = 1.5$ (green and orange) and $t = 3$ (blue and yellow) for different mesh types and resolutions. Results obtained with CFL = 0.5 for FMFPA-CLCIR and isoAdvector-plicRDF.

However, the maximum speedup is reached for about 16 processors for the advection step, while for the reconstruction step it is reached for 20 processors. This behavior may be due to the domain decomposition method used and the processor distribution chosen, i.e., there may be different methods and/or processor distributions that may provide better results in terms of speedup but without ensuring smooth changes between the number of processors used. As far as variations in errors due to parallel execution are concerned, they are negligible in the case of FMFPA-CLCIR, and only minimal oscillations in the error E_{bound} and even smaller for E_{vol} from 8 processors onwards are observed in the case of isoAdvector-plicRDF.

4.3. 3D shearing flow

In this test, proposed by Liovic et al. [37], a sphere of fluid of radius 0.15, initially centered at (0.5, 0.75, 0.25) in a domain of size $1 \times 1 \times 1$,

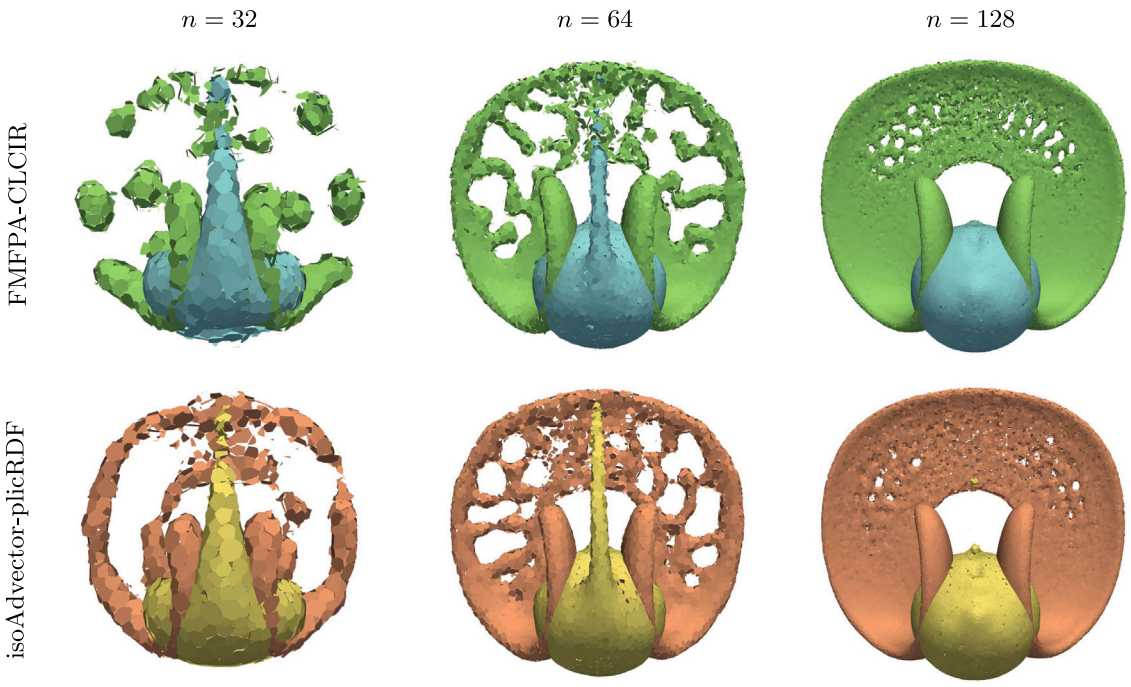


Fig. 5. Same results as in Fig. 4, but for irregular polyhedral meshes and different mesh resolutions.

is deformed in the velocity field defined as

$$\begin{aligned} u(x,y,z,t) &= \sin^2(\pi x)\sin(2\pi y)\cos(\pi t/T), \\ v(x,y,z,t) &= -\sin^2(\pi y)\sin(2\pi x)\cos(\pi t/T), \\ w(x,y,z,t) &= \left\{1-2\left[(x-0.5)^2+(y-0.5)^2\right]^{1/2}\right\}^2\cos(\pi t/T), \end{aligned} \tag{16}$$

during a period $T = 3$.

Table 2 shows the errors and CPU times consumed for the two algorithm combinations under comparison. The times are also provided here in values relative to those obtained in the simulation that yielded the lowest total time, which in this test corresponds to the one performed with the isoAdvector-plicRDF method using a hexahedral mesh of size $n = 32$, for which $t_{\text{tot}} = 21.8$ ms. For all mesh types and sizes, the E_g error measure obtained with FMFPA-CLCIR is always lower than with isoAdvector-plicRDF, with differences ranging from 1.2% for the polyhedral mesh of size $n = 128$ to 68.9% for the finest tetrahedral mesh. Both algorithm combinations achieve second-order convergence, or are very close to it, on hexahedral and irregular polyhedral meshes. For tetrahedral meshes, isoAdvector-plicRDF does not reach second-order convergence while FMFPA-CLCIR does. The isoAdvector-plicRDF volume conservation errors are all near machine precision, while the FMFPA-CLCIR errors are in the range $10^{-5} - 10^{-9}$, whereas the boundedness error is relatively similar for both algorithm combinations on all mesh types and sizes, generally smaller for FMFPA-CLCIR, except for $n = 32$ and 64 on the hexahedral mesh and $n = 128$ on the polyhedral mesh.

The trends in computational efficiency and order of magnitude of the total CPU time are similar to those of the previous test. However, the differences between the two algorithm combinations under comparison are now less significant, particularly for tetrahedral meshes. This is attributed to the smaller number of interfacial cells in the 3D shearing flow test, which mainly affects the advection CPU time consumed. The differences in $\tilde{t}_{\text{tot}}$ decrease with increasing mesh resolution for tetrahedral meshes, while for hexahedral and irregular polyhedral meshes they tend to decrease with mesh resolution up to a certain level of refinement (higher for the former), and then increase with further refinement. Again, isoAdvector gives remarkably low values of $\tilde{t}_{\text{adv}}$ and CLCIR low values of $\tilde{t}_{\text{rec}}$.

Figs. 7 and 8 depict the PLIC interfaces obtained in this test, where the differences in the geometric error shown in Table 2 are again difficult to see with the naked eye.

5. Two-phase flow tests

5.1. Single bubble rising

This test case was described by Hysing et al. [42] and recently used by Gamet et al. [43] in the validation of several reconstruction schemes along with the isoAdvector advection algorithm. A bubble of gas with physical properties $\rho_g = 1\,\text{kg m}^{-3}$ and $\mu_g = 0.1\,\text{Pa s}$, of initial diameter $D_0 = 0.5\,\text{m}$, rises along the positive z direction in a liquid of properties $\rho_l = 1000\,\text{kg m}^{-3}$ and $\mu_l = 10\,\text{Pa s}$. The surface tension coefficient $\sigma = 1.96\,\text{N m}^{-1}$ and the gravity acceleration $\mathbf{g} = (0, 0, -0.98)\ (\text{m s}^{-2})$. Simulations in 2D and 3D geometries were performed, with the bubble center initially placed at (D_0, D_0) and (D_0, D_0, D_0), within computational domains of sizes $2D_0 \times 4D_0$ and $2D_0 \times 2D_0 \times 4D_0$, respectively. Square, triangular and polygonal meshes of sizes $n = 40$, 80, 160, 320, and 640, where n is the number of cells in the x direction, generated as described in Section 2.1, are considered in the 2D simulations. For the 3D test, only a hexahedral mesh with $n = 80$ is considered, and generated with the `snappyHexMesh` tool by imposing on a root mesh of size $10 \times 10 \times 20$ a cylindrical region of diameter $1.4D_0$ centered at the bubble center, with its axis parallel to the z direction, located in the range $0.4D_0 \leq z \leq 3.7D_0$.

The discretization and numerical schemes used in this test are the same for all the combinations of advection–reconstruction algorithms. Following the work by Gamet et al. [43], the Crank–Nicolson scheme with a blending coefficient of 0.9 is used for temporal discretization. This coefficient sets the weight to blend the Crank–Nicolson scheme with the first-order implicit Euler scheme. For the gradient terms, the Gauss scheme is selected, and for the convective term, a TVD scheme with a Sweby limiter function is used. As for the boundary conditions, the lateral walls are considered as slip walls, while the top and bottom boundaries are defined as no-slip walls. For the pressure and volume fraction, the zero normal gradient boundary condition is used on all the walls.

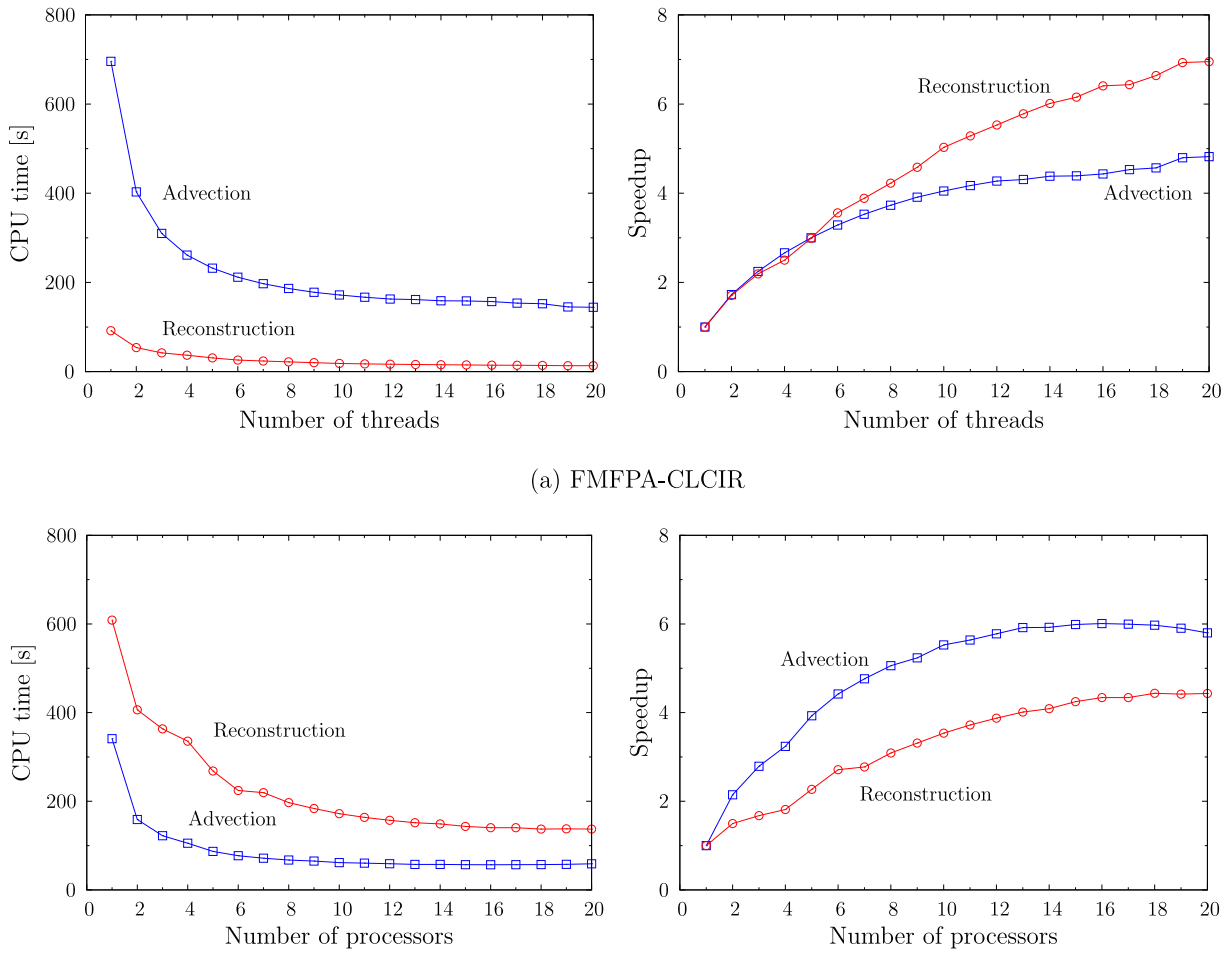


Fig. 6. CPU time consumed and speedup as a function of the number of (a) threads (FMFPA-CLCIR) and (b) processors (isoAdvector-plicRDF), obtained in the 3D deformation flow test with a hexahedral mesh of size $n = 128$.

To solve the Navier–Stokes equations, the PISO algorithm with the momentum predictor step and 3 iterations is used with a single non-orthogonal corrector step for triangular and polygonal meshes. The condition CFL < 0.075 is used for all meshes and resolutions, although a study of the influence of this condition on square meshes has been carried out, as shown below. For the solution of the linear systems of equations, the GAMG is used for the Poisson pressure equation, whereas for the momentum equation an iterative solver with a DIC smoother is selected.

To quantitatively compare the results obtained with the two algorithm combinations under comparison, bubble velocity, circularity (2D), and sphericity (3D) are computed. The bubble velocity is calculated as a weighted average over the whole domain, using the gas volume fraction as weight, following

$$\mathbf{u}_b = \frac{\sum_i \mathbf{u}_i \, (1 - F_i) \, V_i}{V_b}, \tag{17}$$

where $\mathbf{u}_i$ is the velocity, F_i the volume fraction, V_i the volume of cell i, and V_b the bubble volume, calculated as $V_b = \sum_i (1 - F_i) \, V_i$.

Circularity and sphericity are calculated as

$$\frac{[V_b/(\pi \Delta y)]^{1/2}}{A_b/(2\pi \Delta y)} \quad \text{and} \quad \frac{[3V_b/(4\pi)]^{2/3}}{A_b/(4\pi)} \tag{18}$$

respectively, where Δy is the mesh size in the direction perpendicular to the xz plane where the flow is studied in the 2D simulations, and A_b the bubble area. The latter quantity is obtained from the 0.5-isosurface area, which is calculated using the isosurface extraction algorithm available in the isoap library [31] along with the triangulation of this surface to calculate the total bubble area as the summation of the areas of the resulting triangles.

Fig. 9 shows the bubble rise velocity, u_b, as a function of time, calculated from the results obtained from a 2D simulation with a square mesh of size $n = 160$. As the CFL number decreases, the velocity converges to the same solution for both algorithm combinations with almost no difference. Furthermore, it can be seen that the difference between the solutions for the two smallest CFL numbers is very small; for example, the maximum velocities for those CFL numbers differ by less than 1% for FMFPA-CLCIR. Thus, these results indicate that using the condition CFL < 0.075 versus CFL < 0.0375 does not result in a significant decrease in accuracy, while the time step size is increased by a factor of 2.

Fig. 10 shows the time evolution of the bubble rise velocity and circularity for different 2D mesh types and sizes, and Fig. 11 shows the 0.5-isosurfaces for the three finest mesh sizes at the final instant of the test. For square meshes, isoAdvector-plicRDF yields slightly higher velocities for coarser meshes, but as mesh size is increased, both algorithm combinations produce very similar results. Differences in circularity are slightly clearer but, as mesh is refined, both combinations also converge to very similar solutions. For $n = 40$, the secondary bubbles detach almost at the same time ($t \approx 2.64$ ms) in both combinations. The overall agreement can also be seen in Fig. 11(a), where the shape of the bubble is very similar for both combinations at all resolutions. This

Table 2
Errors and CPU times (relative to the lowest $t_{tot} = 21.8$ ms) for the 3D shearing flow test using a CFL = 0.5, for different mesh types and resolutions.

Algorithms	n	E_g	$\mathcal{O}$	E_{vol}	E_{bound}	$\tilde{t}_{tot}$	$\tilde{t}_{adv}$	$\tilde{t}_{rec}$
Hexahedral meshes								
FMFPA-CLCIR	32	3.51e−03	–	7.03e−07	1.17e−08	1.90e+00	1.74e+00	1.66e−01
isoAdvector-plicRDF	32	4.49e−03	–	9.18e−16	1.51e−09	1.00e+00	2.00e−01	8.00e−01
FMFPA-CLCIR	64	9.69e−04	1.86	8.53e−08	8.90e−10	9.95e+00	9.15e+00	7.98e−01
isoAdvector-plicRDF	64	1.28e−03	1.81	5.59e−15	3.14e−10	5.77e+00	1.56e+00	4.21e+00
FMFPA-CLCIR	128	2.37e−04	2.03	1.03e−08	6.93e−11	6.00e+01	5.58e+01	4.23e+00
isoAdvector-plicRDF	128	3.24e−04	1.98	2.84e−14	2.26e−10	5.55e+01	1.99e+01	3.56e+01
FMFPA-CLCIR	256	5.42e−05	2.13	1.08e−09	4.65e−12	4.06e+02	3.79e+02	2.68e+01
isoAdvector-plicRDF	256	7.69e−05	2.07	8.06e−14	5.69e−11	2.82e+02	1.25e+02	1.57e+02
Tetrahedral meshes								
FMFPA-CLCIR	32	6.18e−03	–	1.90e−05	1.19e−08	1.38e+01	1.34e+01	3.67e−01
isoAdvector-plicRDF	32	8.68e−03	–	6.85e−16	4.12e−06	2.21e+00	2.82e−01	1.93e+00
FMFPA-CLCIR	64	1.54e−03	2.01	6.50e−07	3.98e−10	6.03e+01	5.80e+01	2.34e+00
isoAdvector-plicRDF	64	2.86e−03	1.60	1.95e−15	6.25e−07	1.63e+01	2.85e+00	1.34e+01
FMFPA-CLCIR	128	4.03e−04	1.93	8.92e−09	8.49e−11	3.16e+02	3.03e+02	1.35e+01
isoAdvector-plicRDF	128	9.29e−04	1.62	4.12e−15	5.86e−08	1.31e+02	2.61e+01	1.05e+02
FMFPA-CLCIR	256	1.04e−04	1.95	1.94e−09	5.33e−11	1.77e+03	1.69e+03	8.45e+01
isoAdvector-plicRDF	256	3.34e−04	1.47	7.62e−16	3.85e−09	1.22e+03	2.71e+02	9.52e+02
Irregular polyhedral meshes								
FMFPA-CLCIR	32	5.35e−03	–	9.61e−08	2.79e−09	4.56e+01	4.15e+01	4.10e+00
isoAdvector-plicRDF	32	5.93e−03	–	5.27e−16	4.26e−07	1.00e+01	2.99e+00	7.05e+00
FMFPA-CLCIR	64	1.57e−03	1.77	4.57e−09	1.92e−10	2.22e+02	2.02e+02	1.98e+01
isoAdvector-plicRDF	64	1.64e−03	1.85	3.99e−16	4.76e−08	1.98e+02	5.32e+01	1.45e+02
FMFPA-CLCIR	128	4.18e−04	1.91	2.72e−08	3.31e−08	1.41e+03	1.25e+03	1.60e+02
isoAdvector-plicRDF	128	4.22e−04	1.96	9.15e−15	4.62e−09	8.69e+02	2.91e+02	5.78e+02

implies that, for square meshes, the results are not very dependent on the advection–reconstruction algorithms compared here, but depend mainly on other factors such as the PISO algorithm settings or the schemes used to discretize the Navier–Stokes equations.

For triangular meshes, the differences in rise velocity and circularity are now more evident in Fig. 10(b). For all mesh resolutions, isoAdvector-plicRDF yields a higher value of the rise velocity from about $t = 0.7$ ms. For $n = 160$, the results obtained with both algorithm combinations are very similar, but for the remaining sizes they do not coincide as well as in square meshes. Note the worse convergence with mesh refinement of the results obtained with both FMFPA-CLCIR and isoAdvector-plicRDF when using meshes of this type with respect to square meshes. In the time evolution of circularity, the isoAdvector-plicRDF method gives higher values for all mesh resolutions. The coincidence in the results provided by the two algorithm combinations for $n = 160$ in Fig. 10(b) is also apparent in the shape of the 0.5-isosurface shown in Fig. 11(b). For $n = 320$ and 640, the two combinations predict interface shapes that are slightly different from each other and show more apparent asymmetry than for square meshes.

For polygonal meshes (Fig. 10(c)), the differences between the two algorithm combinations considered in the bubble rise velocity are now very small for all mesh resolutions, and for $n = 640$ both combinations give almost indistinguishable solutions. However, in the time evolution of circularity, at low resolutions and after $t = 2.5$ ms, the results are slightly different, although the results of both algorithm combinations follow the same trend. If attention is now focused on the isosurface contours depicted in Fig. 11(c), it can be seen that, although the results are again very similar, usually FMFPA-CLCIR gives rise to more evolved secondary bubbles, i.e., bubbles that appear to detach a few moments earlier than in the results provided by isoAdvector-plicRDF.

Figs. 12 and 13 show the results obtained with the 3D hexahedral mesh and $n = 80$. The rise velocity time evolution (Fig. 12(a)) shows that, again, both algorithm combinations yield very similar solutions, although after about $t = 0.5$ s the velocity obtained with isoAdvector-plicRDF is slightly higher, with the opposite occurring after $t = 3$ s. Fig. 13 shows the 0.5-isosurfaces at different instants, which have been extracted with ParaView [44] for visualization purposes. Only a half of the bubble is represented for each algorithm combination in order to show more clearly the differences between their numerical results. From $t = 1$ s to $t = 2.5$ s, both algorithm combinations yield very similar results. At instant $t = 3$s, the holes that appear in the bubble tail are due to the lack of mesh resolution. The undulations observed at the lower part of the tail, which remain at $t = 3.5$ s, are very similar in the results of both algorithm combinations, as is the shape of the tail break. The top part of the bubble shows almost no difference in the two results. These findings are consistent with the evolutions of the rise speed and circularity, which are very similar.

5.2. Drop impact on a deep pool

The test consists of the impact of a water drop of initial diameter $D_0 = 2.8$ mm on a deep pool of the same liquid at a velocity $U = 1.5\,\mathrm{m\,s^{-1}}$. The experimental results used for comparison were obtained using the equipment used in [45,46] to record the images. The Weber and Froude numbers are We = 88 and Fr = 82. In the numerical simulation, the pool has an initial depth of $4.5D_0$ and the drop center is initially placed at $(0, 0, 5.1D_0)$ (coordinate origin located on the symmetry axis, at the bottom of the tank), with the drop almost in contact with the free surface to avoid possible differences in the velocity impact between both methods. Taking advantage of the symmetry of the problem, only a quarter of the physical domain is solved using a computational domain of size $7D_0 \times 7D_0 \times 7D_0$, which is discretized by a root mesh of $14 \times 14 \times 14$ cells that is statically refined. Two contiguous graded octree patches with four refinement levels are superimposed on the root mesh. One is cylindrical, of diameter $3.57D_0$, with its axis coincident with the z-axis, extending across the region $2.14D_0 \le z \le 6.43D_0$. The second patch horizontally extends the region of maximum refinement of the first, over a region $3.93D_0 \le z \le 5.36D_0$, to the domain boundaries, and matches the root mesh above and below this region using the same graded refinement. The maximum equivalent mesh resolution is 16 cpr, which is maintained around the interface throughout the simulation.

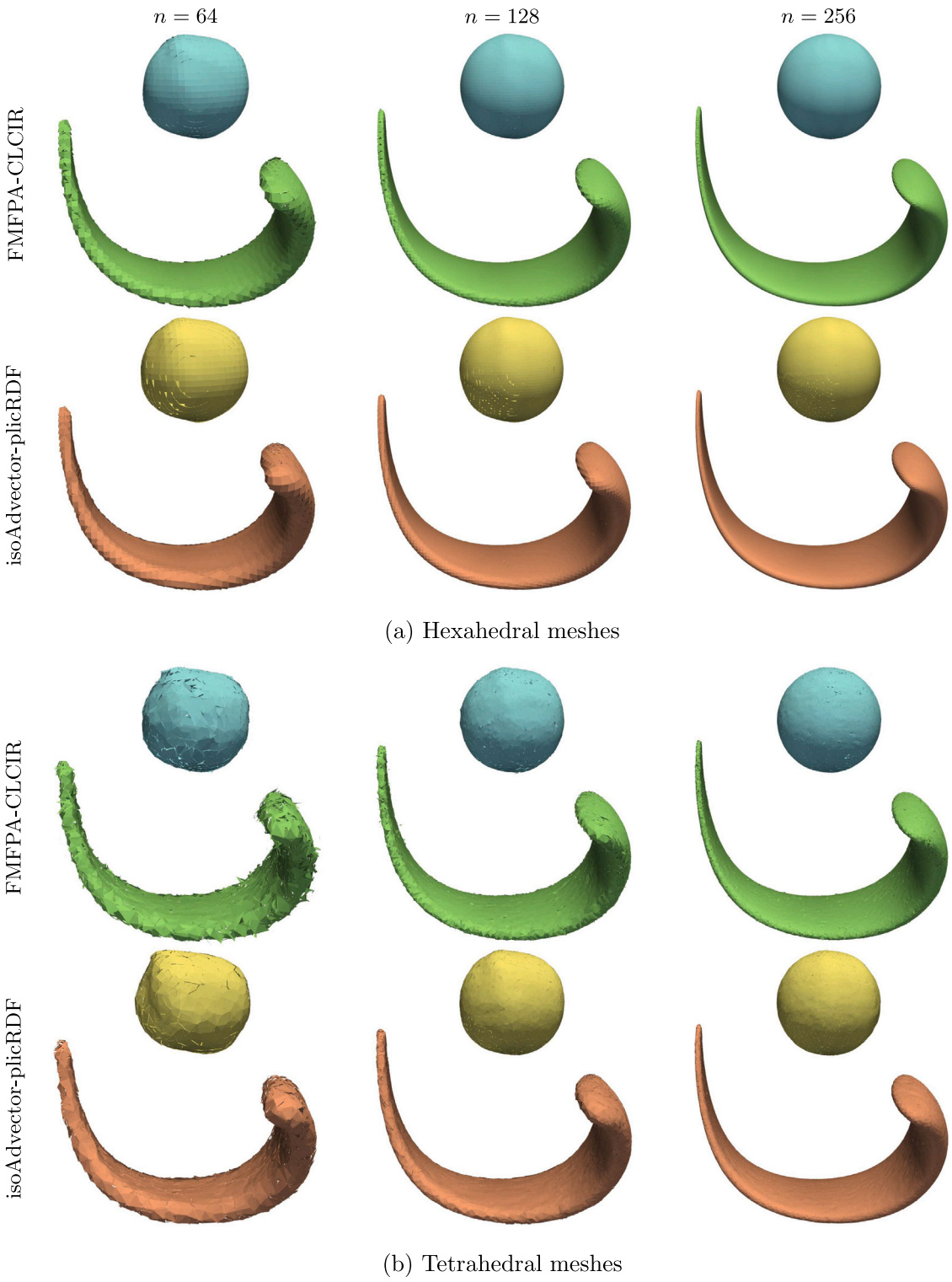


Fig. 7. Same results as in Fig. 4, but for the 3D shearing flow test.

To simulate a quarter of the drop impact problem, the boundaries normal to the x and y directions passing through the coordinate origin are considered as symmetry planes. The remaining boundaries are considered as walls, with a no-slip boundary condition for the velocity. For the volume fraction and pressure, the corresponding gradients normal to the walls are set to zero. For the discretization of gradient terms, the Gauss scheme is used, whereas for the convective terms a TVD scheme is preferred, based on the central differencing and upwind schemes, related through a Sweby limiter function. For the temporal discretization, the first-order implicit Euler scheme is chosen. To solve the systems of algebraic equations, the GAMG solver with a DIC smoother is used for the pressure and velocity terms. To solve the Navier–Stokes equations, the PISO algorithm with three corrections is used, and the condition CFL $<$ 0.5 is set for the calculation of the

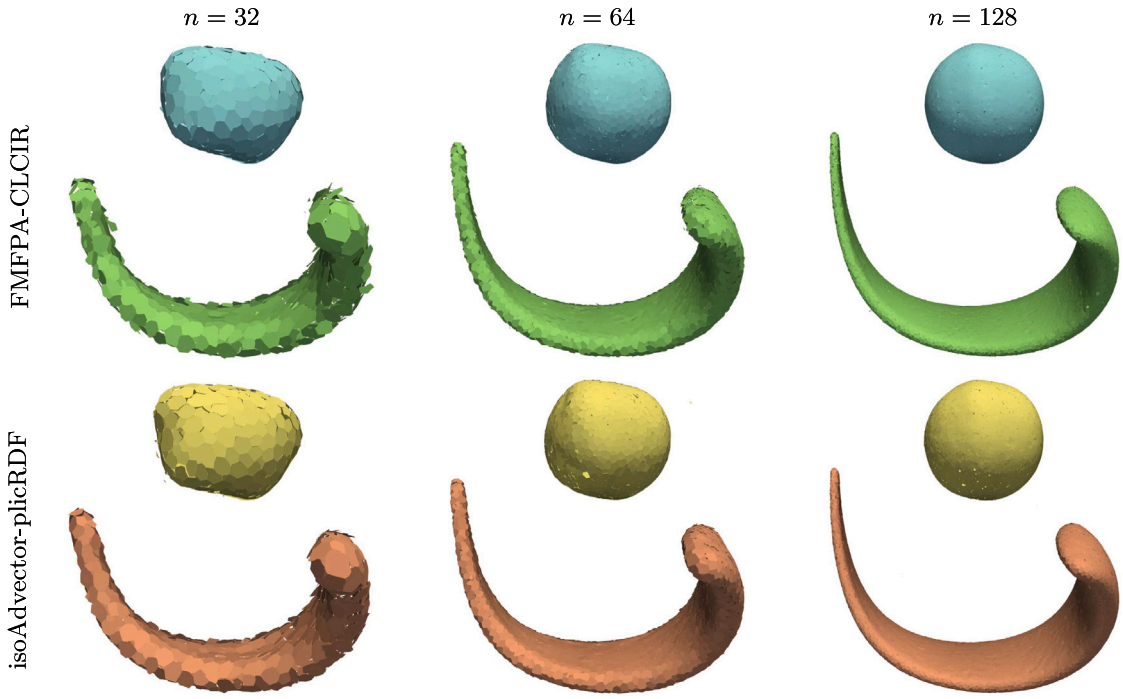


Fig. 8. Same results as in Fig. 7, but for irregular polyhedral meshes and different mesh resolutions.

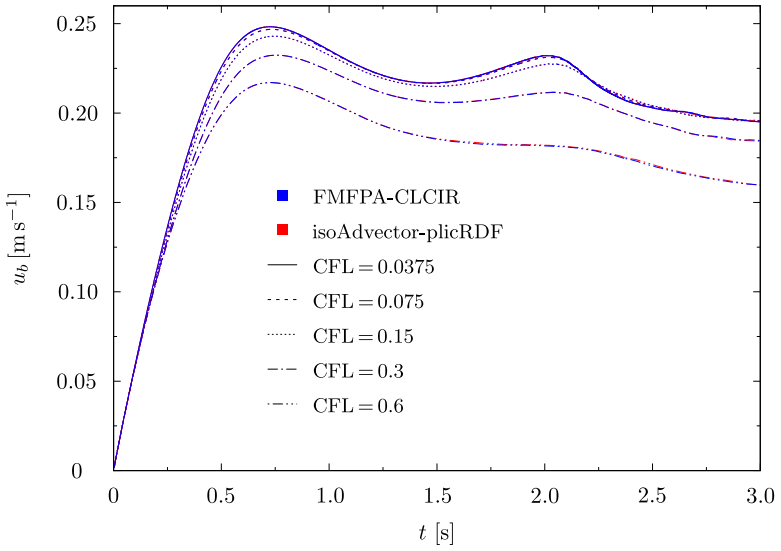


Fig. 9. Time evolution of the bubble rise velocity obtained with different CFL numbers on a 2D square mesh of size $n = 160$. Comparison between isoAdvector-plicRDF and FMFPA-CLCIR.

variable time step. The tolerance for the volume fraction is set to 10^{-6} in the two algorithm combinations compared here.

Fig. 14 shows a comparison between experimental results and the numerical results obtained with FMFPA-CLCIR and isoAdvector-plicRDF, for which the 0.5-isosurfaces are plotted in blue and yellow, respectively. The sequence of images shows the trapping process of an air bubble during the contraction of the crater-shaped cavity, due to the propagation of capillary waves along the side walls of the cavity that converge at its bottom [47]. The retraction of the air filament after the bubble pinch-off results in a thin Worthington jet that quickly breaks into a series of small droplets (the first droplet emerging from the crater is observed in the snapshot for $t = 20.6$ ms, and the last droplet detaches at 22.6 ms, as seen in the video included as supplementary material). The jet then thickens as the crater collapse evolves. The forming droplet observed at 29.8 ms does not subsequently detach. Reasonable repeatability was found despite the sensitivity of the experimental results to impact conditions, which mainly affects the size and number of ejected droplets and the degree of verticality of their trajectory, but also the size of the trapped bubble and the Worthington jet.

As far as the main objective of this test is concerned, which is to compare the results of the two combinations of algorithms considered, a high degree of agreement is observed. Slight differences are observed in the bubble detachment, although in both cases practically the same pinch-off instant is predicted, coinciding with that observed experimentally in the case of isoAdvector-plicRDF and delayed by 0.2 ms in the case of FMFPA-CLCIR, and in the shape of the Worthington jet. The retraction of the air filament after the bubble pinch-off predicted by FMFPA-CLCIR produces a higher momentum in the detached bubble that causes it to reach a greater depth, which agrees slightly

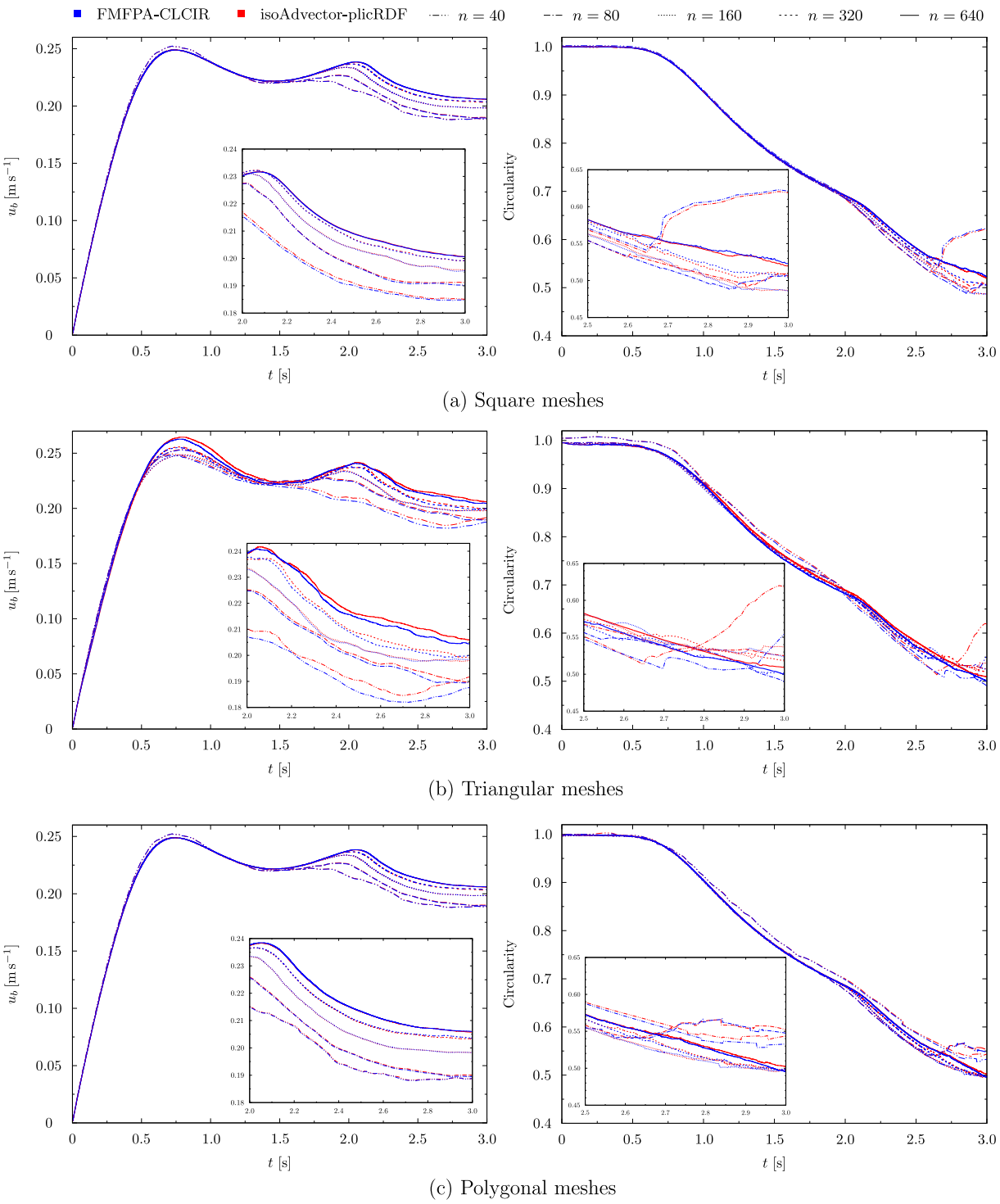


Fig. 10. Bubble rise velocity and circularity as a function of time obtained with FMFPA-CLCIR and isoAdvector-plicRDF on different 2D mesh types and sizes. The insets show a detailed view of the last instants.

better with the experimental observation. The numerical results show a thicker jet than that experimentally observed. Note that the initial thin jet formation and droplet ejection occur on much smaller spatial and temporal scales than the other processes involved in the droplet impact, which would require a very refined computational mesh for a detailed simulation of such phenomena. Nevertheless, the generation of droplets, although in a number of two and of smaller size than those observed in the experimental images, can be seen, albeit with difficulty, from the 20.4 ms instant in the FMFPA-CLCIR results included in the supplementary video.

The small differences found in the results obtained with the two combinations of algorithms for the tests in this section are also relatively small in other two-phase flows with a similar level of complexity. As the level of complexity increases, the relative accuracy and efficiency of the models become progressively more relevant and may produce more appreciable differences depending on the mesh resolution used. In any case, we believe that in general the reasonable agreement of the results is very remarkable and significant, which constitutes a good point of support to continue improving methods under development.

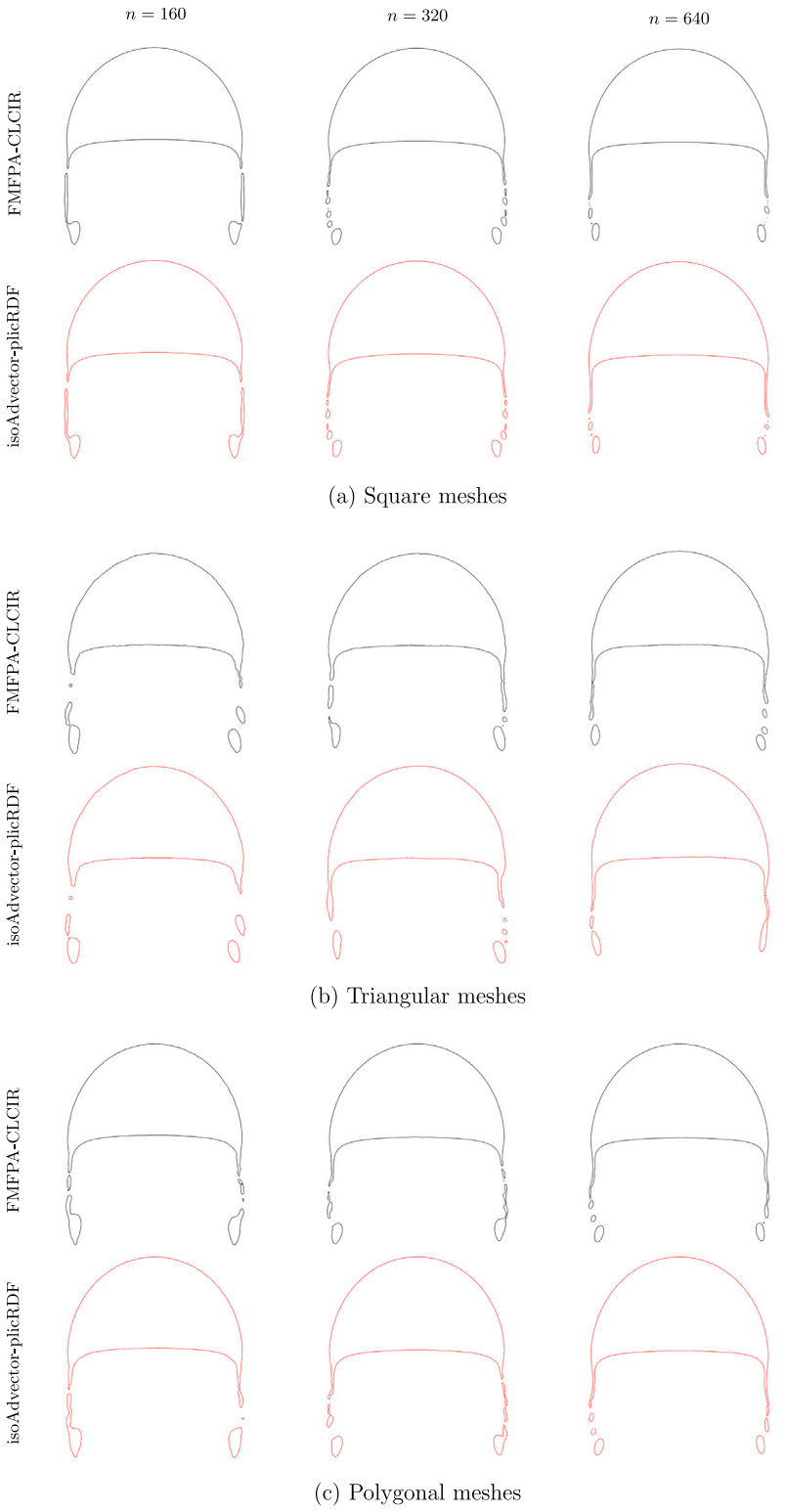


Fig. 11. 0.5-isosurface in the 2D bubble rise test at $t = 3$ ms, obtained on different mesh types and sizes. Comparison between FMFPA-CLCIR (black contours) and isoAdvector-plicRDF (red contours).

6. Conclusions

Different advection and reconstruction algorithms, among those included in the gVOF and TwoPhaseFlow packages, have been thoroughly and systematically compared and further validated, showing their relative capabilities and limitations in terms of accuracy and efficiency on different types of hexahedral, tetrahedral and irregular polyhedral meshes. For this purpose, the algorithms have been tested in the OpenFOAM framework, which has allowed maintaining the same comparison conditions, using the same common solvers, numerical settings and boundary conditions implementations. Identical computational resources have also been used in all simulations.

CLCIR has been found to be about an order of magnitude more efficient than plicRDF, although both algorithms yield very similar

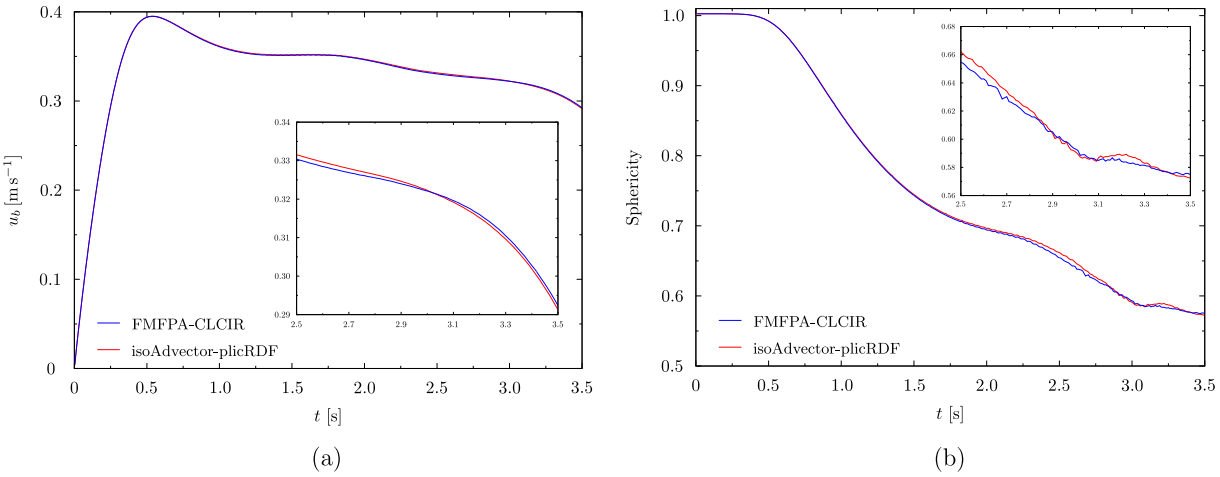


Fig. 12. Rise velocity (a) and sphericity (b) as a function of time, obtained with FMFPA-CLCIR and isoAdvector-plicRDF on a 3D hexahedral mesh and $n = 80$. The insets show a detailed view of the last instants.

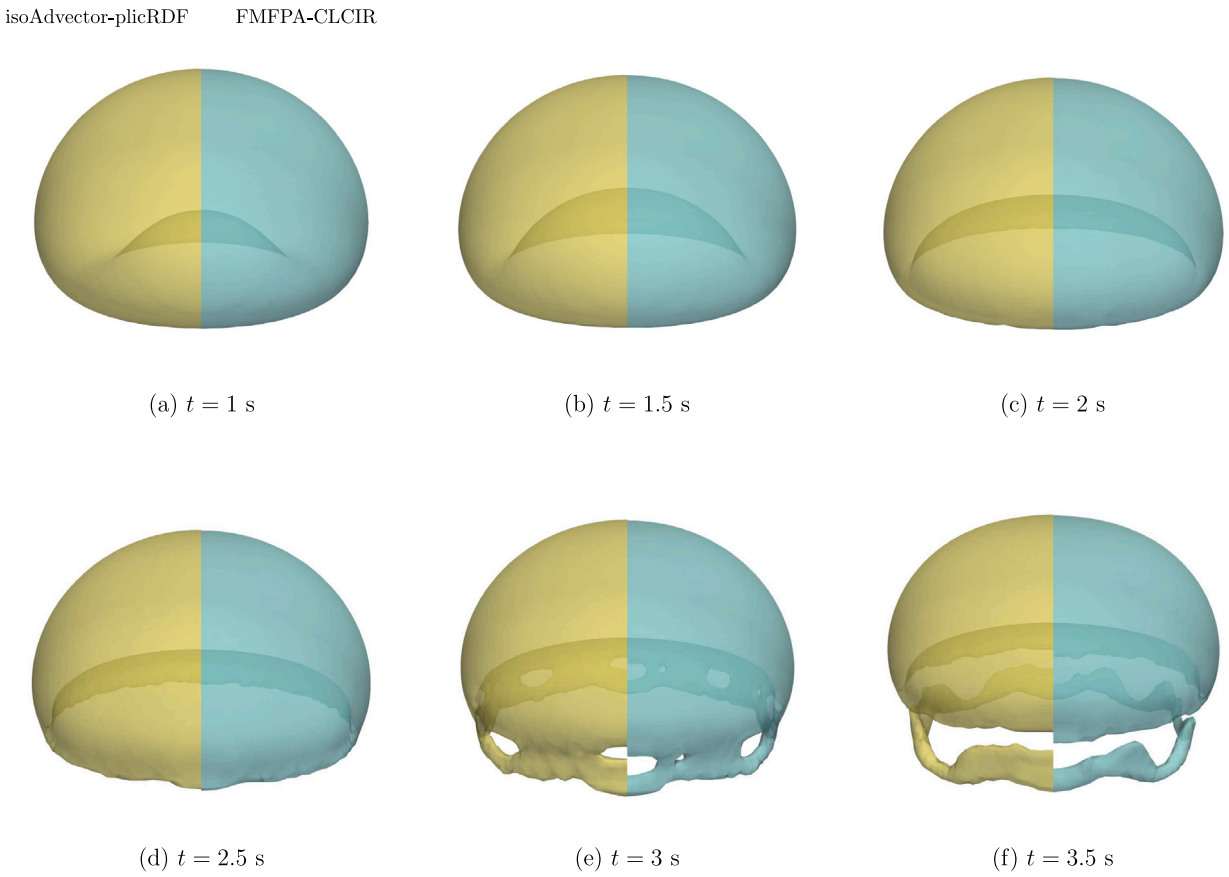


Fig. 13. Results for the 0.5-isosurface contours for the 3D bubble test at different instants, obtained with FMFPA-CLCIR (blue) and isoAdvector-plicRDF (yellow) on a hexahedral mesh and $n = 80$. Only a half of the complete bubble is represented.

results in terms of accuracy on all types of meshes except for fine irregular polyhedral meshes, where plicRDF performs better in terms of accuracy. Through several advection tests with prescribed velocity field for CFL = 0.5, FMFPA-CLCIR has been found to be more accurate than isoAdvector-plicRDF for all types of meshes and resolutions, although at a higher computational cost in the advection steps, which contribute the most to the total CPU time consumed, due to the high number of truncation operations required by FMFPA in the construction of the flux polyhedra. However, the differences in overall efficiency usually decrease as the mesh is refined, except particularly in the case of fine irregular polyhedral meshes. The efficiency of isoAdvector, which is about an order of magnitude faster than FMFPA, constitutes the main advantage of the method.

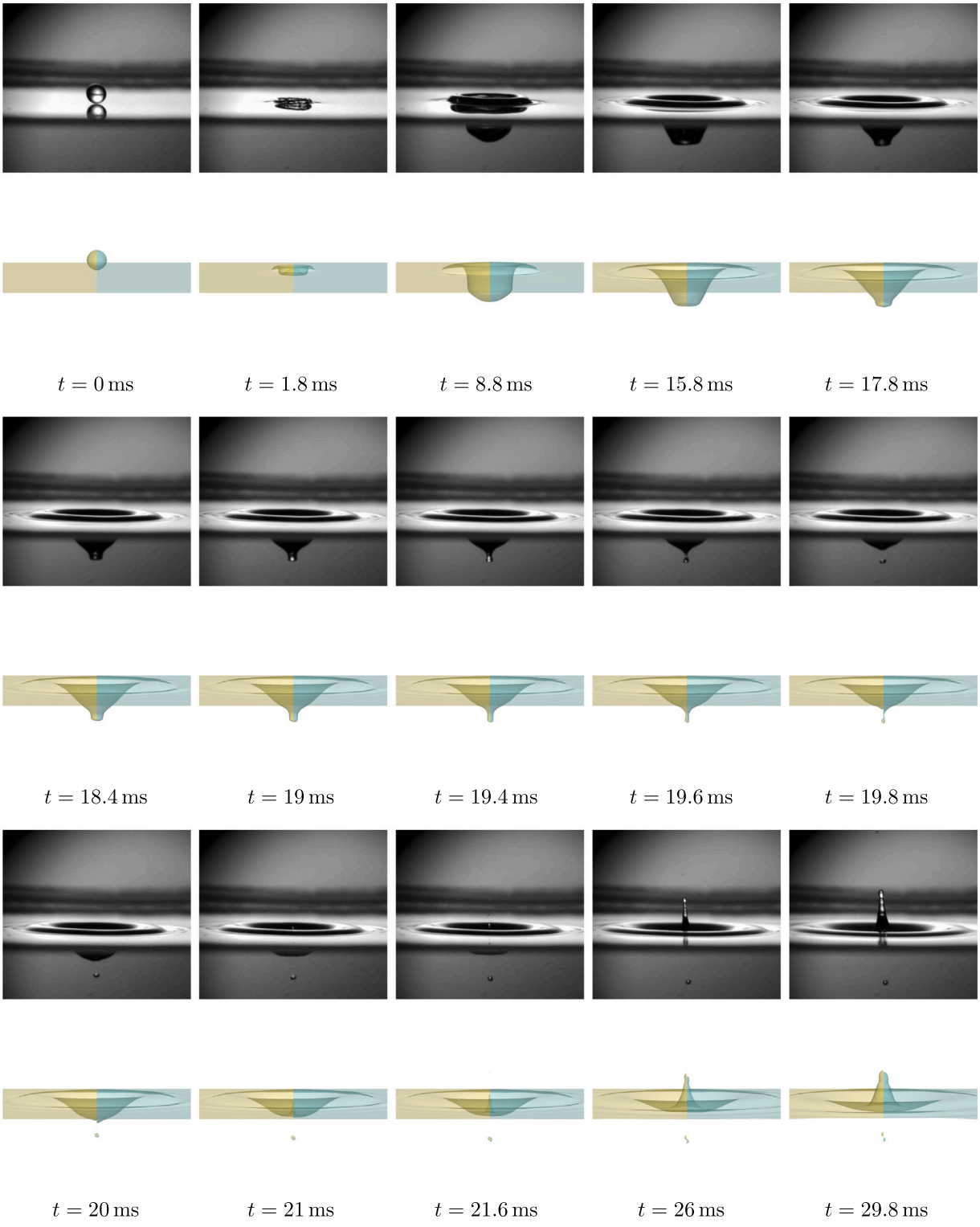


Fig. 14. Comparison between experimental results and the numerical results obtained with FMFPA-CLCIR (blue) and isoAdvector-plicRDF (yellow) in the deep pool drop impact test.

The use of isoAdvector for advection in combination with CLCIR for reconstruction can be a good compromise between accuracy and efficiency, although with limitations for moderate or large CFL numbers, which may require the use of a more accurate advection algorithm. It has also been found that, in general, the two combinations of algorithms studied, when used coupled to the PISO algorithm to solve two-phase flows such as the rise of a bubble or the impact of a drop on a pool, provide relatively similar results in terms of accuracy on different types of meshes, which may make the combination of isoAdvector with CLCIR a suitable choice when computational efficiency is a priority.

Table A.1
Average number of points, faces and neighbor cells per cell ($\overline{n}_p$, $\overline{n}_f$ and $\overline{n}_n$) and per internal cell ($\overline{n}_{p,in}$, $\overline{n}_{f,in}$ and $\overline{n}_{n,in}$) for the different 3D meshes and sizes used in the reconstruction and advection tests. The total number of points (n_p), faces (n_f) and cells (n_c) for each mesh type and size is also provided.

n	$\overline{n}_{p,in}$	$\overline{n}_p$	$\overline{n}_{f,in}$	$\overline{n}_f$	$\overline{n}_{n,in}$	$\overline{n}_n$	n_p	n_f	n_c
Hexahedral meshes									
	8	8	6	6	26				
10						21	1 331	330	1 000
20						23.4	9 261	25 200	8 000
32						24.3	35 937	101 376	32 768
40						24.7	68 921	196 800	64 000
64						25.2	274 625	798 720	262 144
80						25.3	531 441	1 555 200	512 000
128						25.6	2 146 689	6 340 608	2 097 152
160						25.7	4 173 281	12 364 800	4 096 000
256						25.8	16 974 593	50 528 256	16 777 216
Tetrahedral meshes									
	4	4	4	4					
10					67.8	53.1	320	2 271	999
20					73.5	63.4	1 982	17 229	7 981
32					74.8	68.7	6 766	68 636	32 790
40					74.7	69.4	12 979	133 069	63 965
64					76.5	73.2	47 374	536 692	262 139
80					75.1	72.5	93 742	1 056 634	518 135
128					77.3	75.6	352 489	4 245 287	2 097 390
160					75.9	74.4	687 720	8 164 336	4 038 731
256					77.7	76.8	2 687 095	33 443 678	16 620 595
Irregular polyhedral meshes									
10	43.3	33.7	82.9	63.3	15.8	11	13 379	36 054	1 012
20	41.9	36.2	79.9	68.5	15.3	12.8	105 620	290 502	8 077
32	41.4	37.4	78.8	70.7	15.1	13.4	432 805	1 195 863	32 756
40	41.3	38.1	78.5	72.1	15	13.4	853 761	2 634 735	64 076
64	41	38.9	78.1	73.9	15	14.1	3 538 197	9 819 651	262 122
80	41	39.3	77.9	74.5	15	14.3	6 929 638	19 247 250	510 961
128	40.8	39.7	77.7	75.5	14.9	14.5	28 523 863	79 293 605	2 086 918
160	40.8	39.9	77.6	75.8	14.9	14.6	57 097 689	158 781 022	4 164 794

CRediT authorship contribution statement

Adolfo Esteban: Methodology, Software, Validation, Investigation, Data curation, Writing – original draft, Visualization. **Joaquín López:** Conceptualization, Methodology, Software, Formal analysis, Writing – review & editing, Funding acquisition. **Pablo Gómez:** Conceptualization, Formal analysis, Writing – review & editing, Project administration. **Claudio Zanzi:** Investigation, Resources, Data curation, Writing – review & editing. **Johan Roenby:** Methodology, Software, Formal analysis, Writing – review & editing, Supervision, Funding acquisition. **Julio Hernández:** Conceptualization, Methodology, Investigation, Writing – original draft, Supervision, Funding acquisition.

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.

Data availability

Data and links to open-source software are publicly available at https://doi.org/10.5281/zenodo.7260888.

Acknowledgments

Authors from UNED and UPCT gratefully acknowledge the joint support of the Spanish Ministerio de Ciencia, Innovación y Universidades - Agencia Estatal de Investigación and FEDER, Spain 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. J.R. is grateful for funding from Independent Research Fund Denmark via the DFF Sapere Aude Research Leader grant 9063-00018B. The Spanish Ministerio de Ciencia e Innovación and FSE supported A.E. through research fellowship No. PRE2018-087214. A.E. also thanks the University of Toronto and Aalborg University for the invitation for a research stay and the support provided.

Appendix A. Characteristics of the 3D meshes used in the advection and reconstruction tests

Table A.1 shows the total number of points, faces and cells for the meshes considered in the reconstruction and advection tests. The average number of faces, points and neighbor cells (cells sharing a node of a given cell) per cell is also provided, as well as the average number of faces, points and neighbor cells per internal cell, i.e., a cell whose faces are all internal. The latter distinction between cells allows to show more clearly the actual number of neighbor cells, faces and points per cell that are not influenced by the boundary cells.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.compfluid.2022.105725.

References

[1] Mirjalili S, Jain S, Dodd M. Interface-capturing methods for two-phase flows: An overview and recent developments. Center Turbul Res Ann Res Briefs 2017;117–35.
[2] Marić T, Kothe DB, Bothe D. Unstructured un-split geometrical Volume-of-Fluid methods – A review. J Comput Phys 2020;420:109695. http://dx.doi.org/10.1016/j.jcp.2020.109695.
[3] Tryggvason G, Scardovelli R, Zaleski S. Direct numerical simulations of gas–liquid multiphase flows. Cambridge University Press; 2011.

[4] François MM. Recent numerical and algorithmic advances within the volume tracking framework for modeling interfacial flows. Proc IUTAM 2015;15:270–7. http://dx.doi.org/10.1016/j.piutam.2015.04.037, IUTAM Symposium on Multiphase Flows with Phase Change: Challenges and Opportunities.

[5] Roenby J, Bredmose H, Jasak H. A computational method for sharp interface advection. Roy Soc Open Sci 2016;3(11):160405. http://dx.doi.org/10.1098/rsos.160405.

[6] López J, Hernández J. gVOF: An open-source package for unsplit geometric volume of fluid methods on arbitrary grids. Comput Phys Commun 2022;277:108400. http://dx.doi.org/10.1016/j.cpc.2022.108400.

[7] OpenFOAM - The open source CFD toolbox v1812 https://www.openfoam.com/.

[8] Boris JP, Book DL. Flux-corrected transport. I. SHASTA, a fluid transport algorithm that works. J Comput Phys 1973;11(1):38–69. http://dx.doi.org/10.1016/0021-9991(73)90147-2.

[9] Zalesak ST. Fully multidimensional flux-corrected transport algorithms for fluids. J Comput Phys 1979;31(3):335–62. http://dx.doi.org/10.1016/0021-9991(79)90051-2.

[10] Albadawi A, Donoghue D, Robinson A, Murray D, Delauré Y. Influence of surface tension implementation in Volume of Fluid and coupled Volume of Fluid with Level Set methods for bubble growth and detachment. Int J Multiphas Flow 2013;53:11–28. http://dx.doi.org/10.1016/j.ijmultiphaseflow.2013.01.005.

[11] Cifani P, Michalek W, Priems G, Kuerten J, van der Geld C, Geurts B. A comparison between the surface compression method and an interface reconstruction method for the VOF approach. Comput & Fluids 2016;136:421–35. http://dx.doi.org/10.1016/j.compfluid.2016.06.026.

[12] Puckett EG, Almgren AS, Bell JB, Marcus DL, Rider WJ. A high-order projection method for tracking fluid interfaces in variable density incompressible flows. J Comput Phys 1997;130(2):269–82. http://dx.doi.org/10.1006/jcph.1996.5590.

[13] Scardovelli R, Zaleski S. Analytical relations connecting linear interfaces and volume fractions in rectangular grids. J Comput Phys 2000;164(1):228–37. http://dx.doi.org/10.1006/jcph.2000.6567.

[14] isoAdvector https://github.com/isoAdvector/isoAdvector.

[15] Dai D, Tong AY. An analytical interface reconstruction algorithm in the PLIC-VOF method for 2D polygonal unstructured meshes. Int J Numer Meth Fl 2018;88(6):265–76. http://dx.doi.org/10.1002/fld.4664.

[16] Dai D, Tong AY. Analytical interface reconstruction algorithms in the PLIC-VOF method for 3D polyhedral unstructured meshes. Int J Numer Meth Fl 2019;91(5):213–27. http://dx.doi.org/10.1002/fld.4750.

[17] Dianat M, Skarysz M, Garmory A. A Coupled Level Set and Volume of Fluid method for automotive exterior water management applications. Int J Multiphas Flow 2017;91:19–38. http://dx.doi.org/10.1016/j.ijmultiphaseflow.2017.01.008.

[18] Ahn HT, Shashkov M. Geometric algorithms for 3D interface reconstruction. In: Brewer ML, Marcum D, editors. Proceedings of the 16th international meshing roundtable. Berlin, Heidelberg: Springer Berlin Heidelberg; 2008, p. 405–22.

[19] Skarysz M, Garmory A, Dianat M. An iterative interface reconstruction method for PLIC in general convex grids as part of a Coupled Level Set Volume of Fluid solver. J Comput Phys 2018;368:254–76. http://dx.doi.org/10.1016/j.jcp.2018.04.044.

[20] Haghshenas M, Wilson JA, Kumar R. Algebraic coupled level set-volume of fluid method for surface tension dominant two-phase flows. Int J Multiphas Flow 2017;90:13–28. http://dx.doi.org/10.1016/j.ijmultiphaseflow.2016.12.002.

[21] Scheufler H, Roenby J. Accurate and efficient surface reconstruction from volume fraction data on general meshes. J Comput Phys 2019;383:1–23. http://dx.doi.org/10.1016/j.jcp.2019.01.009.

[22] Cummins SJ, Francois MM, Kothe DB. Estimating curvature from volume fractions. Comput Struct 2005;83(6):425–34. http://dx.doi.org/10.1016/j.compstruc.2004.08.017.

[23] VoFLibrary https://github.com/DLR-RY/VoFLibrary.

[24] TwoPhaseFlow Library https://github.com/DLR-RY/TwoPhaseFlow/tree/of1812.

[25] Scheufler H, Roenby J. TwoPhaseFlow: An openfoam based framework for development of two phase flow solvers. 2021, arXiv e-prints https://doi.org/10.48550/arXiv.2103.00870.

[26] Hernández J, López J, Gómez P, Zanzi C, Faura F. A new volume of fluid method in three dimensions—Part I: Multidimensional advection method with face-matched flux polyhedra. Int J Numer Meth Fl 2008;58(8):897–921. http://dx.doi.org/10.1002/fld.1776.

[27] López J, Zanzi C, Gómez P, Faura F, Hernández J. A new volume of fluid method in three dimensions—Part II: Piecewise-planar interface reconstruction with cubic-Bézier fit. Int J Numer Meth Fl 2008;58(8):923–44. http://dx.doi.org/10.1002/fld.1775.

[28] López J, Hernández J, Gómez P, Faura F. VOFTools - A software package of calculation tools for volume of fluid methods using general convex grids. Comput Phys Commun 2018;223:45–54. http://dx.doi.org/10.1016/j.cpc.2017.09.032.

[29] López J, Hernández J, Gómez P, Faura F. Non-convex analytical and geometrical tools for volume truncation, initialization and conservation enforcement in VOF methods. J Comput Phys 2019;392:666–93. http://dx.doi.org/10.1016/j.jcp.2019.04.055.

[30] López J, Hernández J, Gómez P, Zanzi C, Zamora R. VOFTools 5: An extension to non-convex geometries of calculation tools for volume of fluid methods. Comput Phys Commun 2020;252:107277. http://dx.doi.org/10.1016/j.cpc.2020.107277.

[31] López J, Esteban A, Hernández J, Gómez P, Zamora R, Zanzi C, Faura F. A new isosurface extraction method on arbitrary grids. J Comput Phys 2021;444:110579. http://dx.doi.org/10.1016/j.jcp.2021.110579.

[32] Si H. TetGen, a delaunay-based quality tetrahedral mesh generator. ACM T Math Softw 2015;41(2). http://dx.doi.org/10.1145/2629697.

[33] López J, Zanzi C, Gómez P, Zamora R, Faura F, Hernández J. An improved height function technique for computing interface curvature from volume fractions. Comput Method Appl M 2009;198(33):2555–64. http://dx.doi.org/10.1016/j.cma.2009.03.007.

[34] López J, Hernández J, Gómez P, Zanzi C, Zamora R. VOFTools 3.2: Added VOF functionality to initialize the liquid volume fraction in general convex cells. Comput Phys Commun 2019;245:106859. http://dx.doi.org/10.1016/j.cpc.2019.07.022.

[35] Brackbill J, Kothe D, Zemach C. A continuum method for modeling surface tension. J Comput Phys 1992;100(2):335–54. http://dx.doi.org/10.1016/0021-9991(92)90240-Y.

[36] Esteban A, López J, Gómez P, Zanzi C, Roenby J, Hernández J. Vof tests. 2022, http://dx.doi.org/10.5281/zenodo.7260888.

[37] Liovic P, Rudman M, Liow J-L, Lakehal D, Kothe D. A 3D unsplit-advection volume tracking algorithm with planarity-preserving interface reconstruction. Comput & Fluids 2006;35(10):1011–32. http://dx.doi.org/10.1016/j.compfluid.2005.09.003.

[38] López J, Hernández J, Gómez P, Faura F. A new volume conservation enforcement method for PLIC reconstruction in general convex grids. J Comput Phys 2016;316:338–59. http://dx.doi.org/10.1016/j.jcp.2016.04.018.

[39] Missios K. Spatial extension of the advection step of the geometric Volume Of Fluid algorithm isoAdvector. In: Nilsson H, editor. Proceedings of CFD with opensource software. 2021, http://dx.doi.org/10.17196/OS_CFD#YEAR_2021.

[40] Harvie DJ, Fletcher DF. A new volume of fluid advection algorithm: The stream scheme. J Comput Phys 2000;162(1):1–32. http://dx.doi.org/10.1006/jcph.2000.6510.

[41] Enright D, Fedkiw R, Ferziger J, Mitchell I. A hybrid particle level set method for improved interface capturing. J Comput Phys 2002;183(1):83–116. http://dx.doi.org/10.1006/jcph.2002.7166.

[42] Hysing S, Turek S, Kuzmin D, Parolini N, Burman E, Ganesan S, Tobiska L. Quantitative benchmark computations of two-dimensional bubble dynamics. Int J Numer Meth Fl 2009;60(11):1259–88. http://dx.doi.org/10.1002/fld.1934.

[43] Gamet L, Scala M, Roenby J, Scheufler H, Pierson J-L. Validation of volume-of-fluid openFOAM® isoAdvector solvers using single bubble benchmarks. Comput & Fluids 2020;213:104722. http://dx.doi.org/10.1016/j.compfluid.2020.104722.

[44] ParaView https://www.paraview.org/.

[45] Palacios J, Hernández J, Gómez P, Zanzi C, López J. On the impact of viscous drops onto dry smooth surfaces. Exp Fluids 2012;52:1449–63. http://dx.doi.org/10.1007/s00348-012-1264-x.

[46] Gómez P, Zanzi C, López J, Hernández J. Simulation of high density ratio interfacial flows on cell vertex/edge-based staggered octree grids with second-order discretization at irregular nodes. J Comput Phys 2019;376:478–507. http://dx.doi.org/10.1016/j.jcp.2018.09.043.

[47] Pumphrey HC, Elmore PA. The entrainment of bubbles by drop impacts. J Fluid Mech 1990;220:539–67. http://dx.doi.org/10.1017/S0022112090003378.