Loughborough University

This item was submitted to Loughborough's Research Repository by the author.
Items in Figshare are protected by copyright, with all rights reserved, unless otherwise indicated.

An iterative interface reconstruction method for PLIC in general convex grids as part of a Coupled Level Set Volume of Fluid solver

PLEASE CITE THE PUBLISHED VERSION

https://doi.org/10.1016/j.jcp.2018.04.044

PUBLISHER

© Elsevier

VERSION

VoR (Version of Record)

PUBLISHER STATEMENT

This work is made available according to the conditions of the Creative Commons Attribution 4.0 International (CC BY 4.0) licence. Full details of this licence are available at: http://creativecommons.org/licenses/by/4.0/

LICENCE

CC BY 4.0

REPOSITORY RECORD

Skarysz, Maciej, Andrew Garmory, and Mehriar Dianat. 2018. “An Iterative Interface Reconstruction Method for PLIC in General Convex Grids as Part of a Coupled Level Set Volume of Fluid Solver”. Loughborough University. https://hdl.handle.net/2134/32948.

Journal of Computational Physics 368 (2018) 254–276

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

An iterative interface reconstruction method for PLIC in general convex grids as part of a Coupled Level Set Volume of Fluid solver

M. Skarysz *, A. Garmory, M. Dianat

Department of Aeronautical and Automotive Engineering, Loughborough University, Loughborough, LE11 3TU, UK

ARTICLE INFO

Article history:
Received 21 September 2017
Received in revised form 23 March 2018
Accepted 23 April 2018
Available online 27 April 2018

Keywords:
Interface reconstruction
PLIC
VOF
CLSVOF
Root-finding

ABSTRACT

Reconstructing the interface within a cell, based on volume fraction and normal direction, is a key part of multiphase flow solvers which make use of piecewise linear interface calculation (PLIC) such as the Coupled Level Set Volume of Fluid (CLSVOF) method. In this paper, we present an iterative method for interface reconstruction (IR) in general convex cells based on tetrahedral decomposition. By splitting the cell into tetrahedra prior to IR the volume of the truncated polyhedron can be calculated much more rapidly than using existing clipping and capping methods. In addition the root finding algorithm is designed to take advantage of the nature of the relationship between volume fraction and interface position by using a combination of Newton's and Muller's methods. In stand-alone tests of the IR algorithm on single cells with up to 20 vertices the proposed method was found to be 2 times faster than an implementation of an existing analytical method, while being easy to implement. It was also found to be 3.4–11.8 times faster than existing iterative methods using clipping and capping and combined with Brent's root finding method. Tests were then carried out of the IR method as part of a CLSVOF solver. For a sphere deformed by a prescribed velocity field the proposed method was found to be up to 33% faster than existing iterative methods. For simulations including the solution of the velocity field the maximum speed up was found to be approximately 52% for a case where 12% of cells lie on the interface. Analysis of the full simulation CPU time budget also indicates that while the proposed method has produced a considerable speed-up, further gains due to increasing the efficiency of the IR method are likely to be small as the IR step now represents only a small proportion of the run time.

© 2018 The Authors. Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

Among the most popular and widely used multiphase flow simulation methods are those based on a Volume of Fluid (VOF) formulation. In these methods, a function α is defined on a fixed grid which denotes the volume fraction of liquid within each cell. Piecewise linear interface calculation (PLIC) is used in geometric VOF methods to define the position of the interface within the cell. Besides geometric VOF [1,2], many high-order methods based on PLIC-VOF formulation have been implemented such as Moment of Fluid (MOF) [3], MOF coupled with Level Set [4] and Coupled Level Set and Volume

* Corresponding author.
E-mail address: m.skarysz@lboro.ac.uk (M. Skarysz).

https://doi.org/10.1016/j.jcp.2018.04.044
0021-9991/© 2018 The Authors. Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

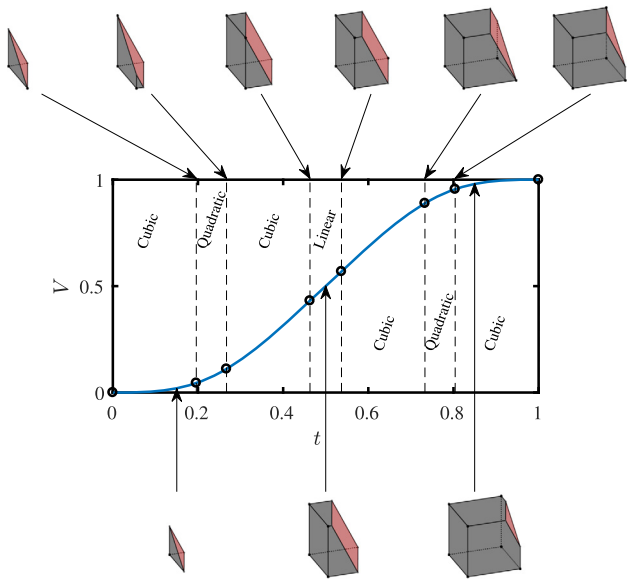


Fig. 1. Truncated cell volume as a function of plane position parameter t for a unit cube. Plane normal vector in spherical coordinates system $\vec{n} = [\phi = 65°, \theta = 70°]$.

of Fluid (CLSVOF) [5–8]. One common operation in each of these methods is an interface reconstruction (IR) step in which the interface between phases is defined within the cell. This is important for two reasons; the first is obtaining an exact position for the interface and the second is using this to define the volume flux through cell faces in the VOF advection step. This ensures mass conservation while limiting the smearing of the interface over multiple cells.

When using PLIC, IR is the process of finding a planar interface within a cell that satisfies a consistency condition, i.e., a plane of given normal direction truncates the polyhedral cell such that the volume on either side of the plane matches a known volume fraction. Any planar surface can be expressed as:

$$\vec{n} \cdot \vec{x} + D = 0 \tag{1}$$

where D is a constant and $\vec{n}$ is the normal vector of the plane which we assume is already given e.g., as a gradient of VOF or gradient of LS. The IR procedure is thus to determine D such that the plane splits the cell in two parts with given volume fraction α_T and $1 - \alpha_T$, where $\alpha_T \in [0, 1]$ and the subscript T indicates a target volume fraction. This involves solving the following equation for D:

$$\alpha(D) - \alpha_T = 0 \tag{2}$$

where $\alpha(D)$ is volume fraction obtained by splitting the cell by plane with constant D. We can re-scale the plane constant D to the parameter $t \in [0, 1]$such that $t = 0$ and $t = 1$ represent the cases of $\alpha_T = 0$ and $\alpha_T = 1$, respectively.

IR can refer to the entire multiphase interface in the computational domain or the local operation in the cell, which is also called local volume conservation enforcement. Here we will use *Interface Reconstruction* in both meanings, i.e., local and global. The complexity of the geometrical operations involved can result in significant CPU time cost. In this paper we consider the computational efficiency of the IR method, both in isolation and as part of full multiphase flow calculation. The latter is done to understand the potential for further improvements in a VOF type algorithm.

For IR problems in general 3D cells the most widely used and robust method was proposed by [9]. The solution is obtained in an iterative process by calculating the truncated volume for a series of values of the parameter D and comparing it to the prescribed target volume fraction until given tolerance is reached. An iterative root-finding method is needed such as Brent's [10] method. For simple geometries, more efficient methods were proposed by [11] for rectangular meshes and [12] for triangular and tetrahedral meshes, both based on an analytical approach. However as Eq. (2) changes its characteristics between linear, quadratic and cubic, as is shown for example in Fig. 1, any analytical approach must contain an initial step (usually iterative) which brackets the solution to the range where the degree of the equation is constant. Recently, analytical methods have been introduced for general convex cells [13–16] reducing CPU time cost considerably in comparison to previous iterative methods, for example in the case of a hexahedron cell the analytical method proposed by [14] is reported to be 4.7 times faster than Brent's method. The performance improvement rises to 5.9 for more complex cells.

In this paper, we propose an iterative method for interface reconstruction in a PLIC formulation that performs comparably with analytical methods and is simpler in implementation. The novelty of the paper is an efficient volume calculation

method (based on tetrahedral decomposition of the cell prior to IR) along with an iterative root-finding method tailored for the IR problem. As the IR procedure is never used by itself, the focus of the present paper is a comparison of IR methods both in isolation and in a multiphase solver to assess the real advantage of using higher efficiency methods and the potential for further improvement. For this purpose the IR methods are implemented in a Coupled Level Set Volume of Fluid solver [8]; (1) without momentum solver for the deformation of a sphere and (2) with momentum solver for drop impingement cases and rivulet flow.

The outline of the paper is as follows: in Section 2 available iterative and analytical IR methods which will be used for comparison are described. The proposed method is introduced in Section 3. Section 4 comprises efficiency tests of all discussed methods for single cell calculations. The performance of the proposed method in a multiphase solver, implemented in the OpenFOAM CFD suite [17], is assessed in Section 5 in comparison to results employing an existing iterative method for IR.

2. Interface reconstruction methods

In this section, two existing IR methods are presented. The first is a 'Clipcap' method [18] combined with Brent's method [10] or secant-bisection (used in [18,1]) root-finding methods, which represent popular iterative approaches used in interface reconstruction problems. The second method is an analytical method described in [14] and [16], the principle in both references is similar.

2.1. Clipping and capping with root-finding method

Combined secant and bisection methods were proposed by [18] and used in [1] as a root-finding method for solving IR problem. The advantage of the secant method is fast convergence but in some conditions it may diverge, whereas bisection is a slow method with guaranteed convergence. Another approach is Brent's method which is a well-known, highly reliable, effective root-finding algorithm combining the bisection method, the secant method, and inverse quadratic interpolation [10]. Both methods are commonly used in IR algorithms, however they require a method to find the volume of the truncated polyhedron at each iteration. The 'Clipcap' algorithm proposed in [18] is such a method to construct a truncated polyhedron of which the volume can be calculated.

In iterative methods, the planar interface is shifted in the direction of the surface normal until the volume bounded by the cell and the interface matches the target cell volume fraction α_T. The method to find the truncated volume is not straightforward on an arbitrary grid as the number of faces and edges, as well as their angles to each other, is not known in advance. In addition these numbers can also change during the iterative process as the interface is moved. The approach proposed by [18] and used in [1] and [8] is based on a clipping and capping algorithm. In this method, intersected faces of the cell are 'clipped' by the interface to form liquid polygons on the faces which are subsequently 'capped' by the interface polygon which joins these to form a liquid polyhedron. Once intersection points of the cutting plane and the cell's edges are found their relationship to each other, and to the existing cell vertices, must be established as this connectivity information is needed for the calculation of volume, which can be calculated by a pyramid rule or Goldman's formula [19]. This determines the volume of polyhedron with n_f faces by the expression:

$$V = \frac{1}{6}\sum_{j=1}^{n_f}\left[(\vec{n}_j \cdot \vec{x}_{j,1})\vec{n}_j \cdot \sum_{i=1}^{n_{vj}}(\vec{x}_{j,i} \times \vec{x}_{j,i+1})\right] \tag{3}$$

where $\vec{n}_j$ is a unit outward pointing vector normal to face j; $\vec{x}_{j,i}$ is the i-th vertex of the j-th face and n_{vj} is the number of vertices in j-th face. Note that this formula requires $n_f n_{vj}$ cross products and $2n_f$ dot products. For example, for a hexahedron cell it is 24 cross products and 12 dot products. Note that n_{vj} will also change as the plane is moved across the cell. The volume of the truncated polyhedron can be calculated by different methods, for example in [1] a tetrahedral decomposition is used to calculate the volume of a polyhedron formed by sweeping a polygonal face. This approach, however, requires a decomposition in every step of the iterative process.

2.2. Analytical method

For simple grid geometries such as an orthogonal rectangular mesh the interface location can by established analytically by solving an equation based on the known geometry of the cell (see for example [6,7]). This approach was extended to arbitrary 2D cells in [13].

The case is more complicated for arbitrary 3D cells. The first analytical IR method for arbitrary convex 3D cells was proposed by [15] and improved by [16] and [14]. The main idea of these methods is similar: bracket the solution and use an analytical formula valid in the prismatic truncated shape (see Fig. 2). Bracketing the solution involves determining all the vertices of the original polyhedron that will remain in the truncated one. As one of the most expensive parts of analytical methods, the bracketing procedure has been significantly improved by [16], ultimately achieving a number of bracketing steps of order $O(\log_2(\log_2(N)))$, where N is the number of vertices for the polyhedron. However, when the number of

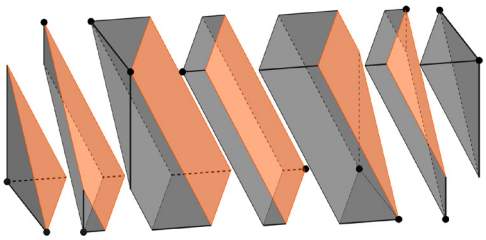


Fig. 2. Cell decomposition in analytical methods for unit cube. Plane normal vector in spherical coordinates system $\vec{n} = [\phi = 65°, \theta = 70°]$. In this example, the decomposition creates 7 prismatoids.

vertices of the cell to be truncated is small the time benefit is not very significant and a simpler bracketing procedure referred as a standard sequential bracketing (SSB) performs sufficiently well, with the average number of bracketing steps in the order of $O(N/4)$. The improved bracketing procedure (referred to as interpolation bracketing (IB) in [16]) reduces the consumed CPU time in the cases when the number of vertices is large, i.e $N > 8$. This method has been tested in cells with a number of vertices up to 30 in [16].

Once the bracketing step is accomplished the prismatic truncated shape is created from a polygon that has the given normal vector and includes the nearest vertex of the cell. In Fig. 2 the prismatic truncated shapes found by bracketing a cube are shown. For each polygon node, a list of prismatoid edges must be stored. The second parallel polygon of the prismatoid is formed by the interface plane. To find the position of the second polygon (and therefore the interface) a volume of the polyhedron is expressed as a function of prismatoid height. In the general case, this is a cubic formula for which coefficients are found from equations consisting of vector operations performed on the prismatoid's edges. The last step in this method is to solve the cubic equation to match to the target volume and this can be done in many ways, e.g. by Newton method as used in [14].

3. Proposed tetrahedral decomposition method

A new method for solving the IR problem is proposed here. The method consists of two parts: a method of finding the volume of truncated polyhedron based on tetrahedral decomposition prior to IR, and a root finding method tailored to the particular demands of the IR problem. The first part, calculating the volume of a truncated polyhedron based on tetrahedral decomposition, is a more efficient alternative to the Clipcap method described in the previous section. The second part of the proposed method is a root-finding method that is a combination of Newton's and Muller's methods [20,21] which uses quadratic interpolation (not to be confused with inverse quadratic interpolation which is part of Brent's method).

3.1. Problem formulation

The first stage in the proposed method is to formulate the problem such that the current volume fraction α is a function of a single variable t which represents the position of the plane. The aim then is to find t that $\alpha(t)$ matches the target volume fraction α_T. Consider a plane $\mathcal{P} : \vec{n} \cdot \vec{x} + D = 0$, which is moved along its normal vector $\vec{n}$ from one side of a polyhedron Ω to the other. We need to define the start and end of the line over which the interface plane is swept. Let $\vec{x}_k$, $k = 0, \ldots, N$ be the vertices of polyhedron Ω, and let $\mathcal{L} : \vec{x}_\mathcal{L} = \vec{a}_0 + \vec{n}t$, $t \in \mathbb{R}$ be any line orthogonal to the plane $\mathcal{P}$, so $\vec{a}_0$ could be any point in $\mathbb{R}^3$. Projection of the vertices $\vec{x}_k$ onto the line $\mathcal{L}$ results in a set of points defined by:

$$\tilde{\vec{x}}_k = \vec{a}_0 + \vec{n}(\vec{n} \cdot \vec{x}_k) \tag{4}$$

If $\tilde{\vec{x}}_k$ are sorted in ascending order of the value of $\tilde{x}_k = \vec{n} \cdot \vec{x}_k$ we can identify the vertices which represent the extremes at which the plane with normal $\vec{n}$ can cut the cell. We can then find $\vec{a}_0 = \vec{x}_{lo}$ such that $\tilde{x}_{lo} = \min_k(\vec{n} \cdot \vec{x}_k)$. Similarly the maximum point is defined as $\tilde{x}_{hi} = \max_k(\vec{n} \cdot \vec{x}_k)$. The line can now be redefined as:

$$\mathcal{L} : \vec{x}_\mathcal{L} = \vec{x}_{lo} + \vec{n}t(\tilde{x}_{hi} - \tilde{x}_{lo}), t \in \mathbb{R} \tag{5}$$

Since $\vec{x}_{lo}$ and $\vec{x}_{hi}$ have been defined as the extreme points of Ω, $0 \leq t \leq 1$ represents all planes $\mathcal{P}(D)$ that cross the polyhedron Ω. We now have a function $\alpha(t)$ which needs to be solved to find the value of t which gives $\alpha = \alpha_T$. The function $\alpha(t)$ is monotonic, increasing and smooth.

3.2. Volume calculation by tetrahedral decomposition

Here we propose a volume calculation method based on tetrahedral decomposition. The first observation is as follows: to calculate the volume of a tetrahedron the only information needed are the coordinates of its vertices in any order, unlike the analytical and Clipcap methods where the order of vertices and edges matters introducing additional complexity in

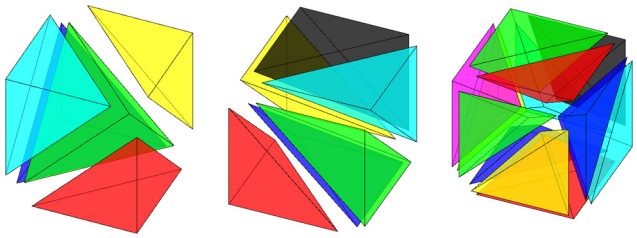


Fig. 3. Hexahedron decomposition (from left to right): 5 (optimal, can be obtained by 'peeling' the vertices of degree 3), 6 (can be obtained by 'coning' from left-front-top corner) and 12 tetrahedrons (can be obtained by method used in OpenFOAM).

implementation and storage requirements which are avoided by the proposed method. A tetrahedron's volume V_{tet} can be found from only one dot product and one cross product:

$$V_{tet} = \frac{1}{6}|\vec{r}_{03} \cdot (\vec{r}_{13} \times \vec{r}_{23})| \tag{6}$$

where $\vec{r}_{ij} = \vec{x}_i - \vec{x}_j$ is a vector from $\vec{x}_j$ to $\vec{x}_i$, and $\vec{x}_{i=0,\dots,3}$ stand for tetrahedron vertices.

3.2.1. Cell decomposition

In order to take advantage of this and reduce complexity and number of geometrical operations, a decomposition to non-intersecting tetrahedrons is proposed. Such a decomposition can be easily defined in any solver as the geometry of cells is precisely known from the mesh generation process and the decomposition pattern can be established in advance for specific cell geometries. The time penalty is very low and can be constant if the decomposition is made once in the initialization steps before the simulation has started. However, the only information required from this decomposition is a set of 4 nodes' labels or nodes' coordinates, therefore the decomposition can be made once a time step as an initial step of the IR algorithm for every time step during the simulation whilst incurring an insignificant time penalty.

Decomposition can be carried out for any type of cell but must be carefully executed to result in a minimum number of tetrahedra to achieve the best IR performance. This is not straightforward as the number of possible decompositions is very large and it is important to use a method that will identify the minimum or close to it. The OpenFOAM software suite [17] has the capability to perform tetrahedral decomposition, which is used to increase the efficiency of its Lagrangian particle tracking algorithm. This is robust but not sufficiently efficient for our purposes, being based on triangulating faces and joining these to the cell centre which leads to as many as 12 tetrahedra for a cube when the minimum is 5.

To establish a decomposition pattern general methods such as 'coning' or 'peeling' algorithms can be used. Coning (also called 'pulling' or 'starring') [22] is a relatively simple method which can be used for any convex polyhedron resulting in slightly larger than optimal decomposition. In this method one vertex is chosen, each face of the polyhedron that is not adjacent to this vertex is triangulated and each triangle is connected to the chosen vertex to form tetrahedrons. The number of tetrahedrons depends on the choice of the first vertex and this can be done in N ways (N is a number of vertices). Therefore, all N possibilities can be tested to find the best solution. Peeling [23] is based on iteratively removing the cap of one vertex. The cap is easily decomposed to tetrahedrons, and the remaining polyhedron is one vertex smaller until only 4 vertices remain, thus producing the last tetrahedron. As the 'peeled' vertex can be chosen on $N - i$ ways on each iteration ($i = 0, \dots, N - 5$), a 'greedy' version of this algorithm was proposed, where the vertex with minimal degree is used. For more details see Edelsbrunner et al. [23]. In our implementation when more than one vertex with a minimal degree was found a random selection was used and this was repeated 100 times, which obtained the optimal result in almost all cases.

However, in this paper we have used the optimal (minimal) decomposition due to de Loera et al. [24] of which details can be found in Appendix A. We have implemented decompositions based on this for selected cells with number of vertices up to 20, i.e., prism, hexahedron, 10 nodes polyhedron, rhombic dodecahedron, regular icosahedron and regular dodecahedron (see Fig. 5). We tested the decomposition for each geometry and compared the optimal result with that from coning, peeling and OpenFOAM. For details see Table 1. The alternative decompositions from these methods for a hexahedron are shown in Fig. 3 as an example. The decomposition method used here is extendable to computational meshes with a range of convex cell shapes. In practical cases of CFD simulations, the computational meshes consist of a finite number of cell shapes which can be decomposed as a preprocessing step using the method described in [24], and used here, or alternatively coning or peeling could be used resulting in potentially a small penalty in the number of tetrahedra produced. For a fuller discussion of the topic of tetrahedral decomposition, including extension to non-convex cells, see [23,25–27,24] and references therein.

Once the decomposition has taken place the volume of the polyhedron can be calculated as a sum of the volume of the tetrahedra $\mathcal{T}_i$:

$$V_\Omega = \sum_i^{N_t} V(\mathcal{T}_i) \tag{7}$$

Table 1
Numbers of tetrahedrons used in decomposition compared to optimal.

Cell shape	Decomposition			
	optimal	coning	peeling	OpenFOAM
Prism	3	3	3	8
Hexahedron	5	6	5	12
10 vertices polyhedron	9	10	9	16
Rhombic dodecahedron	12	16	12	24
Regular icosahedron	15	15	16	20
Regular dodecahedron	23	27	23	36

where the volume of tetrahedron $V(\mathcal{T})$ can be found from Eq. (6) and N_t stands for the number of tetrahedra that the polyhedron Ω has been split into. Note that for hexahedral cells the volume calculated by Eq. (7) requires only 5 cross products and 5 dot products whereas in classical Goldman's formula of Eq. (3) it is 24 and 12 respectively.

It should be noted that, as is the case with the clipping and capping and analytical methods, the method presented here is strictly based on an assumption of planar cell faces. The presence of non-planar faces would lead to an error in the IR algorithm and hence in the face flux used in the advection step [8]. The flux would remain conservative leading to a mass conservative scheme with reduced order of accuracy, as is generally the case with non-planar faces in CFD, as is discussed in [28], for example and for this reason meshing algorithms attempt to minimise non-planarity. The method presented here could be further developed to deal with non-planar faces, for example by sharing the triangulation information of these faces between adjoining cells, but this is not considered here.

3.3. Calculation of cut cell volume fraction using tetrahedral decomposition

Once the polyhedral cell has been decomposed into tetrahedra the volume of the shape created by the intersection of the plane $\mathcal{P}$ with the cell can be found. This is calculated as the sum of the volumes formed by the intersection of $\mathcal{P}$ with the tetrahedra that make up the cell. If $V_\alpha(\mathcal{P},\mathcal{T})$ represents the volume created when tetrahedron $\mathcal{T}$ is truncated by the plane $\mathcal{P}$ then the global cell volume fraction created by $\mathcal{P}$, $\alpha(\mathcal{P})$, can be expressed as:

$$\alpha(\mathcal{P}) = \frac{\sum_i^{N_t} V_\alpha(\mathcal{P},\mathcal{T}_i)}{V_\Omega} \tag{8}$$

An algorithm for finding $V_\alpha(\mathcal{P},\mathcal{T})$ is needed. This is much easier for a tetrahedron than a general polyhedron as the cut plane can take only two shapes as shown in Fig. 4. The cut plane can be a triangle in which case a tetrahedron is formed or a quadrilateral in which case two prisms are formed. The first step is to identify where the four vertices of the tetrahedron lie relative to the plane. The following is checked for all $\vec{x}_{i=0,\dots,3}$:

$$\vec{x}_i \quad \text{is} \quad \begin{cases} \text{on the plane } \mathcal{P}, & \text{if } \vec{n}\cdot\vec{x}_i + D = 0 \\ \text{above the plane } \mathcal{P}, & \text{if } \vec{n}\cdot\vec{x}_i + D > 0 \\ \text{below the plane } \mathcal{P}, & \text{otherwise} \end{cases} \tag{9}$$

Following this $(n_\mathcal{P}, n_{up}, n_{down})$ represent the number of vertices which meet the above conditions, i.e. on, above and below the plane. From this, those conditions requiring no further operations can be identified:

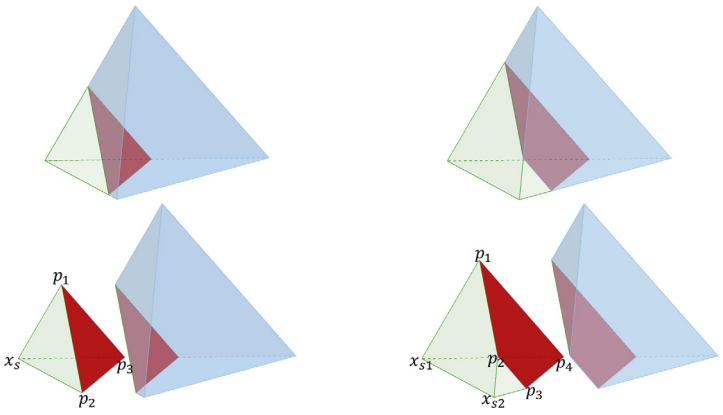
- $n_{up} = 4$ or $n_{down} = 4$ means the plane does not cut the tetrahedron,
- $n_\mathcal{P} = 3$ means that the plane lies on a face,
- $n_\mathcal{P}$ equals 1 or 2 when the other vertices lie on the same side of the plane.

These cases imply that the plane does not intersect the tetrahedron and it is full or empty depending on n_{up} and n_{down}.

For all other cases the intersection point must be found on those edges that connect vertices on opposite sides of the plane. The advantage of the proposed method is that there is limited need for information about connectivity, as in a tetrahedron all combinations of $\vec{x}_i$ and $\vec{x}_j$ where $i \neq j$ form an edge. The intersection of edge $\vec{e}_{i,j} = \vec{x}_i + t(\vec{x}_j - \vec{x}_i)$, $t \in [0,1]$ with the plane $\mathcal{P} : \vec{n}\cdot\vec{x} + D = 0$ can be found from the following equation:

$$t_k = -\frac{D + \vec{n}\cdot\vec{x}_i}{\vec{n}\cdot(\vec{x}_j - \vec{x}_i)} \tag{10}$$

Intersection point $\vec{p}_k$ can then be found using the edge equation $\vec{p}_k = \vec{x}_i + t_k(\vec{x}_j - \vec{x}_i)$, where $k = 0, \dots, N_k - 1$ and N_k is the number of intersecting points. The intersection plane, defined by the points $\vec{p}_k$ and $n_\mathcal{P}$, will then have either three or four points. In the case of three points then a tetrahedron will lie on one side of the plane and the volume of this can easily be calculated. In the case of four points prisms are created either side of the plane and the volume of either can be



(a) Triangle cross-section. New tetrahedron is formed.

(b) Quadrilateral cross-section. Both shapes after truncation are prisms.

Fig. 4. Two possible cases of tetrahedron truncation by a plane. In the first one, the triangle is formed in the cross-section whereas in the second case cross-section is a quadrilateral.

calculated by further decomposition as in Appendix A. To perform this decomposition some connectivity information must be used.

3.4. Root-finding method

The root-finding method employed in this work has been developed based on the specific characteristics of the IR problem. IR involves the resolution of $f(t) = \alpha(t) - \alpha_T = 0$. It has been proved in [15,16,14] that function $\alpha(t)$ has a degree of 3 at the most, but locally could be linear or quadratic. The characteristics of the function change when the cutting plane passes through a vertex. The particular characteristics of the truncated volume function depend on cell shape as well as the normal vector $\vec{n}$. We propose here a root finding method, for solving $f(t) = 0$, that converges very fast on a linear part of the function (only a few iterations are required) but also converges fast in the quadratic range. In the first step Newton's method is applied but instead of using an analytical derivative, a numerical one is taken. Unless the root is found at this first step, the subsequent root approximations are obtained by quadratic interpolation based on points t_i, t_{i-1} and t_{i-2}. This part is similar to Muller's method [20]. The algorithm of the proposed root finding method is described in detail in Algorithms 1 and 2. In case of single tetrahedron cell, the Newton step in the first iteration can be omitted, as a tetrahedron never exhibits the linear region of the volume function.

The order of convergence (taken from [20,21]) of the quadratic method in the general case is 1.84, whereas Brent's method, which uses either inverse quadratic or linear interpolation, has order of convergence of 1.84 or 1.62, respectively. The highest order of convergence, equal to 2, is for Newton's method, but without having a derivative of the function this rate is unachievable. Although the difference between Brent's method and our proposed method is small our method performs better in the case of IR as Muller's method was constructed for polynomial equations, as encountered here, where the function is linear, quadratic or cubic. Moreover, if the root happens to be in the linear regime, which has a high probability for a cube for example, our method will calculate the root with only few steps.

Algorithm 1 Root-finding algorithm.

$t_0 \leftarrow$ starting point ▷ Starting point could be 0.5
$f_0 \leftarrow f(t_0)$
$t_1 \leftarrow t_0 + small$
while $|f_0| > error$ **do**
  if first iteration **then** ▷ Newton's method
    $f_1 \leftarrow f(t_1)$
    $\delta \leftarrow -f_0 \frac{small}{f_1 - f_0}$
  else
    $\delta \leftarrow$ QuadSolve($t_{0,1,2}$, $f_{0,1,2}$) ▷ quadratic interpolation
  end if
  $t_2 \leftarrow t_1$, $t_1 \leftarrow t_0$, $t_0 \leftarrow t_0 + \delta$
  $f_2 \leftarrow f_1$, $f_1 \leftarrow f_0$, $f_0 \leftarrow f(t_0)$
end while
return t_0 ▷ t_0 solves IR problem

Algorithm 2 Quadratic interpolation.

procedure QUADSOLVE($x_0, x_1, x_2, y_0, y_1, y_2$)
  $w \leftarrow \frac{y_0-y_1}{x_0-x_1} + \frac{y_0-y_2}{x_0-x_2} - \frac{y_1-y_2}{x_1-x_2}$
  $k \leftarrow \frac{y_2-y_1}{(x_2-x_1)(x_2-x_0)} - \frac{y_1-y_0}{(x_1-x_0)(x_2-x_0)}$
  if $w^2 - 4ky_0 > 0$ **then**
    $\delta \leftarrow \frac{-2y_0}{w+\mathrm{sgn}(w)\sqrt{w^2-4ky_0}}$ ▷ reduces round-off error
  else
    $\delta \leftarrow -\frac{w}{2k}$ ▷ if quadratic root does not exist
  end if
  return δ ▷ if root exists $f(x_0+\delta)=0$
end procedure

3.5. Initial guess for iterative process

Usually, in root-finding methods such as bisection, secant or Brent's the starting interval $[a, b]$ must be provided in the first step. In the proposed method, instead of an interval, a starting point must be chosen and this could be $t_0 = 0.5$. In a practical implementation of IR algorithms in fluid flow solvers due to small time steps in time integration the change in interface location and normal vector for many cells is small. Therefore, instead of finding the root from a fixed starting point the previous solution presents a good starting approximation for the next IR step, resulting in better performance which is discussed in Section 5.2.

3.6. Rotation of the reference frame

A significant improvement in CPU time consumption can be achieved by rotating the polyhedron such that the plane normal vector becomes a unit vector along the z-axis, i.e $\vec{n} \rightarrow [0, 0, 1]$. This approach was proposed by [14]. After such a transformation all vector calculations involving $\vec{n}$ are reduced, e.g. the dot product becomes simply taking the z-component. The transformed coordinates can be found from the following:

$$\vec{x}_{new} = R\vec{x} \tag{11}$$

$$R = \begin{pmatrix} n_y^2\Gamma + n_z & -n_x n_y \Gamma & -n_x \\ -n_x n_y \Gamma & n_x^2\Gamma + n_z & -n_y \\ n_x & n_y & n_z \end{pmatrix} \tag{12}$$

$$\Gamma = \frac{1 - n_z}{n_x^2 + n_y^2} \tag{13}$$

where $\vec{x}_{new}$ is a point in the rotated reference. Note that there is no need to perform the inverse rotation. The advantage of this operation is discussed in Section 4.

4. Performance of IR in single cells

In this section, a number of numerical tests are provided to demonstrate the computational efficiency of the proposed method for IR in isolation and compare it with existing iterative and analytical methods. Tests were carried out for single convex 3D cells. Accuracy and computational cost were measured and compared for the proposed iterative method, an implementation of an analytical method and an existing iterative approach based on the Clipcap volume calculating method with Brent's method and secant-bisection method.

4.1. Test conditions

The choice of normal vectors and volume fractions for testing conditions should cover all possible combinations that could arise in real simulation to assess the performance of all methods. Therefore we propose volume fractions equally distributed in the range of 0 and 1 and normal vectors covering all 3D directions. Eight different 3D cells were tested: tetrahedron, prism, cube, irregular hexahedron, 10 vertices polyhedron, rhombic dodecahedron, regular icosahedron and regular dodecahedron (see Fig. 5). The position of the cell vertices are shown below:

1. Trirectangular tetrahedron, vertex coordinates:
   $(0, 0, 0)$, $(1, 0, 0)$, $(0, 1, 0)$, $(0, 0, 1)$
2. Prism, vertex coordinates:
   $(0, 0, 0)$, $(1, 0, 0)$, $(0, 1, 0)$, $(0, 0, 1)$, $(1, 0, 1)$, $(0, 1, 1)$
3. Cube, vertex coordinates:
   $(0, 0, 0)$, $(1, 0, 0)$, $(1, 1, 0)$, $(0, 1, 0)$, $(0, 0, 1)$, $(1, 0, 1)$, $(1, 1, 1)$, $(0, 1, 1)$

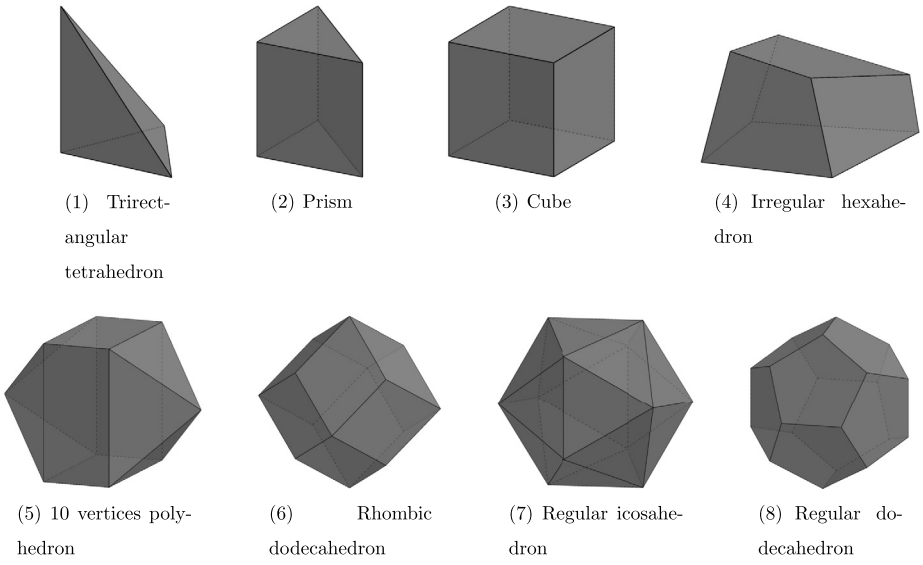


Fig. 5. Shapes of cells used in the numerical tests.

4. Irregular hexahedron, vertex coordinates:
   $(0,0,0)$, $(1,0,0)$, $(1.2,1.2,0)$, $(0,1,0)$, $(0.16,0.16,0.8)$, $(0.785,0.136,0.68)$, $(1.085,1.315,0.384)$, $(0.119,1.228,0.595)$
5. 10 vertices polyhedron, vertex coordinates:
   $(\pm 0.25,\pm 0.5,\pm 0.5)$, $(\pm 0.75,0,0)$
6. Rhombic dodecahedron, 14 vertices:
   $(\pm 1,0,0)$, $(0,\pm 1,0)$, $(0,0,\pm 1)$, $(\pm 0.5,\pm 0.5,\pm 0.5)$,
7. Regular icosahedron, 12 vertices:
   $(\pm\phi,\pm 1,0)$, $(0,\pm\phi,\pm 1)$, $(\pm 1,0,\pm\phi)$, where $\phi=(1+\sqrt{5})/2$
8. Regular dodecahedron, 20 vertices:
   $(0,\pm\phi^{-1},\pm\phi)$, $(\pm\phi,0,\pm\phi^{-1})$, $(\pm\phi^{-1},\pm\phi,0)$, $(\pm 1,\pm 1,\pm 1)$,
   where $\phi=(1+\sqrt{5})/2$

The following set of normal vectors was tested:

$$\vec{n}_{ij}=(\sin\phi_i\cos\theta_j,\sin\phi_i\sin\theta_j,\cos\theta_j) \tag{14}$$

where

$$\phi_i=\frac{i}{100}\pi \quad ,i=0,\ldots,100 \tag{15}$$

$$\theta_j=\frac{j}{100}2\pi \quad ,j=0,\ldots,100 \tag{16}$$

Volume fractions were evenly distributed in the range of 0-1 with $N=1000$:

$$\alpha_T^k=\frac{k}{N} \quad ,k=1,\ldots,N-1 \tag{17}$$

This set S gives $N_S=10$ millions of combinations $(\vec{n}_{ij},\alpha_T^k)$. Such a number of tests is required partly to all conditions that could arise in a real simulation but mainly to provide a test case with a relatively long run time to reduce noise in measurement of CPU time.

For each cell shape, the CPU time consumed to complete the set of $(\vec{n},\alpha_T)$ combinations is measured. The error tolerance $|\alpha_T-\alpha_{reconstructed}|$ was set to be 10^{-15} and was achieved in all tests. Relative Time Units (R.T.U.) were used to show the computational cost for quicker and easier comparison of different techniques. R.T.U is set to be one for tetrahedron cell case performed by the method proposed in this paper. The absolute CPU execution time for the tetrahedron case was 8.52 s to complete all the set which is equivalent of 1 R.T.U.

4.2. Technical details

The proposed method as well as an analytical method and Clipcap, with Brent's iteration, have been implemented in C++ in the same code and compiled by the GNU C++ compiler with -O3 optimization flag (gcc version 4.8.5 SUSE Linux). All

Table 2
CPU times for completing the set of test cases by different Interface Reconstruction approaches in different cell geometries. Time is scaled to Relative Time Units (R.T.U.). Absolute CPU time for 1 R.T.U is equal to 8.52 s. Analytical refers to our implementation of [14], and Clipcap refers to Clipcap solved by Brent's method without bracketing and with bracketing labelled as SNSB.

Cell shape	R.T.U			
	Proposed	Analytical	Clipcap Brent's	Clipcap Brent's SNSB
Tetrahedron	1.00	2.23	11.76	14.34
Prism	1.65	3.52	13.28	20.04
Cube	2.23	4.92	15.76	27.26
Irregular hexahedron	2.72	4.96	17.43	27.48
10 vertices cell	4.53	8.75	27.87	53.50
Rhombic dodecahedron	5.16	12.56	29.93	69.32
Regular icosahedron	6.66	12.74	40.40	90.35
Regular dodecahedron	9.35	15.83	31.89	101.06

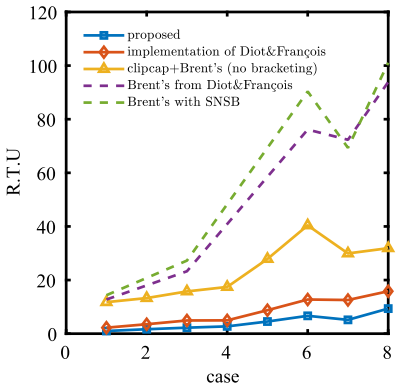


Fig. 6. The CPU time in R.T.U consumed to complete the testing set by different methods for cell shapes in Fig. 5. Brent's from Diot and François [14] are scaled to our implementation of the analytical method of [14].

results were run on a single Intel Xeon CPU E5-1650 3.50 GHz core. The CPU execution time was measured by `clock()` function (C Time Library) which returns the processor time in clock ticks consumed by the program, subsequently divided by clock ticks per second to obtain absolute time. The initial guess for the proposed method was 0.5. The analytical method was carefully implemented according to all details in Diot and François [14], and similarly, the root of the cubic equation was found by the Newton method, as in [14]. The Clipcap method was implemented according to the method described in [8]. Brent's method and secant-bisection method were initialized with extreme positions of the intersecting planes (0 and 1 respectively) as the interval and were run without any bracketing procedure. The combination of Clipcap and Brent's (without bracketing) is our baseline in tests in CLSVOF solver. Additionally, to assess the effectiveness of our implementation of the analytical method, we use as a reference Clipcap and Brent's method with standard non-sequential bracketing procedure (SNSB), which is the same configuration that was used in [14]. Note that in this application it appears that the extra volume calculations in the bracketing procedure increase the calculation time. Reproduction of relative values reported in [14] allows us to benchmark our implementation compared to the original. It should be pointed out that the absolute CPU times reported in this paper may vary depending on particular implementation, programing language or compiler.

4.3. Numerical results

The results obtained for different cell shapes by different methods are summarized in Table 2. The CPU times are presented as a function of the number of vertices in the test cells in Fig. 6. The proposed method was found to be between 3.4 and 11.8 times faster than Clipcap with Brent's method for the regular dodecahedron and tetrahedron respectively, and was on average 6.8 times faster in all cases. The proposed method has a comparable efficiency to our implementation of the analytical method with the proposed method found to be approximately twice as fast as the analytical method in our tests. To demonstrate similarity of performance of our implementation of the analytical method the relative results of [14] (Brent's with SNSB) are superimposed in the Fig. 6, with reference to our implementation of analytical method. The analytical approach in Diot and François [14] was found to be 4.7 (for the cube), 5.9 (for the dodecahedron), and 5.6 (on average) times faster than Brent's approach in [14], whereas in this implementation a speed-up of 5.5 to 6.4 and 6.0 on average

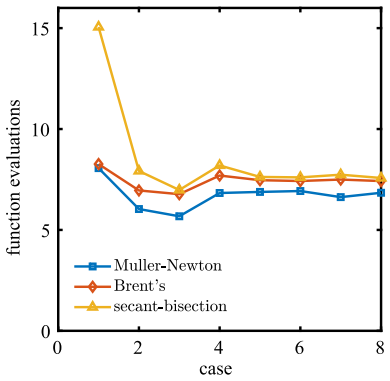


Fig. 7. Average number of volume function evaluations in Muller's–Newton, Brent's and secant-bisection methods for cell shapes in Fig. 5.

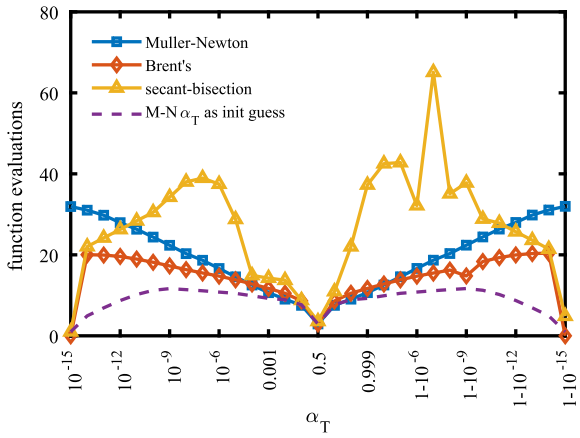


Fig. 8. Average number of volume function evaluations in Muller's–Newton (M–N), Brent's and secant-bisection methods for target volume fractions down to 10^{-15} and up to $1-10^{-15}$. Note the log scale. Average over interface normal directions (Eq. (14)) and cell shapes (Fig. 5). In M–N two initial guesses were tested: $t_0 = 0.5$ and $t_0 = \alpha_T$ (target volume).

was observed. To what extent differences in IR performance will affect overall performance of a multiphase flow solver is considered in the next section.

4.4. Root-finding methods comparison

To test how much of the speed up of our method is due to the root finding method three root-finding methods were tested: that proposed in this paper a combination of Muller's–Newton method, Brent's method, and a secant-bisection method proposed in [18] and used in [1] where secant is used followed by bisection if secant fails. The methods were tested on the same set of cases as described earlier. The volume evaluation function was performed by the proposed tetrahedral decomposition method. In our tests (Fig. 7) we found Brent's method to be slightly faster than secant except for the case of a pure tetrahedron for which secant method performs very poorly resulting in almost double number of iterations due to the secant method failing and falling back on bisection very often. On average we obtained 7.43 function evaluations in Brent's method, and 8.58 in secant-bisection (mostly dominated by tetrahedron case). The proposed Muller's–Newton method needed on average 6.73 function evaluations, which resulted in being 10% and 27% faster than the other methods. Further advantage of using proposed Muller's–Newton method is presented in Section 5.2 where initial guess for IR is discussed.

Additionally, we have successfully tested volume fractions down to 10^{-15} and up to $1-10^{-15}$ to assess the robustness of our method and to compare with Brent's and secant-bisection methods (see Fig. 8). The poor performance of secant-bisection method is mainly caused by using bisection method once secant fails. For stability purposes and better convergence in the Muller–Newton method for volume fractions greater than 0.5 we solved the inverse problem for the other fraction i.e. $1-\alpha_T$. The probability of solving for very small fractions is also very small and thus the performance in these particular cases will not affect the average performance of the methods in multiphase solvers. Nonetheless, for small volume fractions, we also show in the Fig. 8 an advantage of using different starting guess in Muller–Newton method i.e. $t_0 = 0.5$ and $t_0 = \alpha_T$.

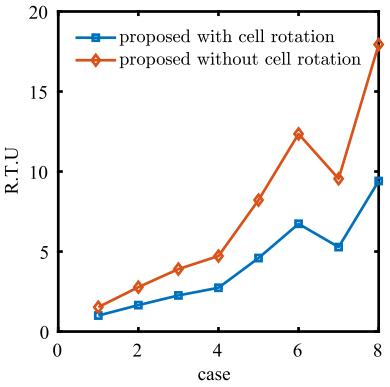


Fig. 9. The CPU time consumed to complete the testing set with and without rotation of reference frame for cell shapes in Fig. 5.

4.5. Effect of rotation of the reference frame

The effect of the rotation of the polyhedron such that the plane normal vector becomes a unit vector along the z-axis is explored in the same number of tests as in previous cases. This approach was found to save about half of CPU time per function evaluation compared to the cases without rotation (see Fig. 9).

5. IR performance within a CLSVOF solver

In this section, four different tests are presented in a multiphase solver implemented in OpenFOAM [17]. This is done to assess the performance of IR methods in practical applications and also to demonstrate computational cost contribution of interface reconstruction algorithms in the entire flow simulation. This will give an understanding of the potential benefits of further increases in IR efficiency. The solver used in the following test is a Coupled Level Set and Volume of Fluid method for unstructured grids [8]. Reconstruction is performed based on volume fraction and normal vector found from the gradient of Level Set. Where needed, the momentum equation is solved by a PISO algorithm with 3 corrector steps. The full description of the method can be found in Dianat et al. [8].

Four tests were chosen as follows: sphere deformation in a 3D vortex field (without solving momentum equation), and three examples with momentum solver, namely drop impingement on a non-orthogonal grid, a rivulet filling a channel and a splashing droplet which has a greater proportion of interface cells. The simulations were again run on a single processor of the hardware described in Section 4.2.

As was shown in the previous section the computational cost of the proposed method and analytical method is comparable and both are significantly faster than the classical approach. As such the analytical method had not been implemented for purpose of the following tests. Therefore two methods are compared in the following tests: the proposed one and Clipcap with Brent's iteration.

5.1. IR test case: 3D deformation of a sphere

A 3D test case proposed in [29], and also applied by [30], was used in which a spherical interface is stretched into a thin film. A sphere of radius 0.15 is placed within the domain $[0, 1]^3$ with its centre at (0.35, 0.35, 0.35). A uniform hexagonal mesh of $200 \times 200 \times 200$ cells is used. The velocity field is specified by

$$\begin{aligned} u(x, y, z, t) &= 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos(\pi t/T) \\ v(x, y, z, t) &= -\sin^2(\pi y)\sin(2\pi x)\sin(2\pi z)\cos(\pi t/T) \\ v(x, y, z, t) &= -\sin^2(\pi z)\sin(2\pi x)\sin(2\pi y)\cos(\pi t/T) \end{aligned} \tag{18}$$

where $T = 3$ s is the period. As the velocity is imposed analytically, the momentum and pressure solvers do not need to be solved and only operations concerning the Level-set and VOF fields need be performed.

Simulation results from the proposed method are shown in Fig. 10 for time $t = 0, 0.1T, 0.15T, 0.2T$ and $0.5T$ (the largest deformation). Identical results are obtained with the Clipcap method and have been omitted. The size of the interface, measured by the number of cells that contain the interface, increased during the simulation from 0.2% of total mesh cells up to ~0.8%, as can be seen in Fig. 11.

The time taken for the IR step per time step for the whole mesh can be seen in Fig. 12. As expected the CPU time increases with the number of interface cells. The IR step for the whole mesh consumes 1.3–1.9 s and 4.3–7.7 s of CPU time for proposed and Clipcap with Brent's methods respectively. This corresponds to average speed-up of the proposed method of 3.68 over Clipcap with Brent's method.

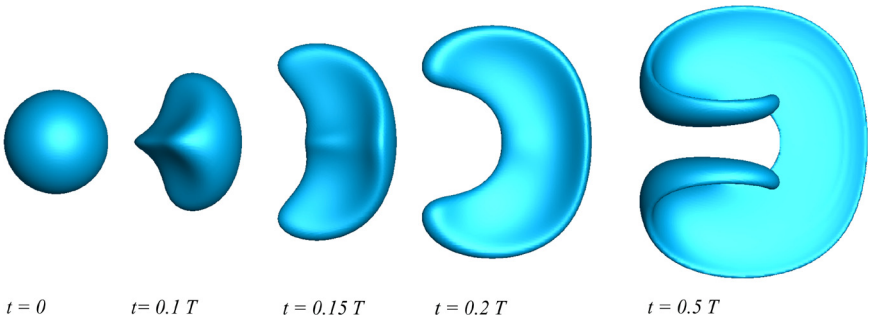


Fig. 10. Deformation of a sphere in a 3D vortex field predicted by CLSVOF method. From left to right: initial sphere at $t = 0$, deformation at $t = 0.1T$, $t = 0.15T$, $t = 0.2T$ and largest deformation at $t = 0.5T$, where $T = 3$ s.

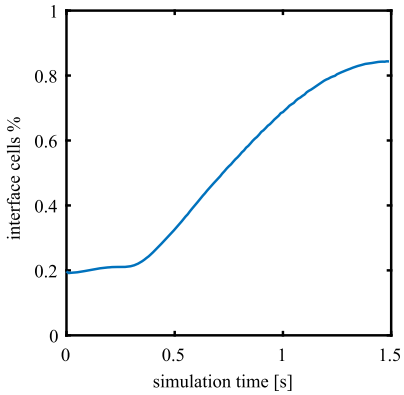


Fig. 11. Number of cells comprising interface as a function of simulation time of sphere deformation case.

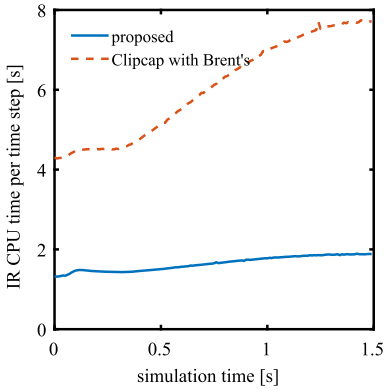


Fig. 12. Sphere deformation case: CPU time consumed by IR per time step done by different methods as a function of simulation time.

The speed ups for the IR alone are smaller than the results obtained for cells in isolation because IR consists of necessary additional numerical operations which were not simulated for the single cell but must be performed in multiphase solvers. These include calculations of gradient or face area of fluid (which is subsequently used in VOF advection). Some CPU time must also be spent identifying interface cells. Therefore, the CPU time spent on IR can be expressed as $T_{IR} = C + n_I\theta_{Icell}$, where C stands for constant time of IR regardless the number of interfacial cells, n_I, and θ_{Icell} is CPU time spent on IR for a single cell. To assess the IR performance per additional cell, θ_{Icell}, the CPU time, T_{IR}, was plotted as a function of number of interfacial cells in Fig. 13. The CPU time consumed by the proposed IR method per additional cell, found from the gradient of this line, was 7.03 times lower than Clipcap with Brent's method, which is similar to the single cell results from Section 4.

The CPU times consumed by the whole CLSVOF time step are shown in Fig. 14. The maximum time reduction obtained by proposed method is 1.33 and the mean 1.23 in comparison to Clipcap with Brent's method. The reduced speed up compared to the IR step in isolation is due to CLSVOF solver, even without momentum and pressure equations, requiring other operations in addition to reconstruction. These operations comprise VOF advection, LS advection, LS reinitialization, data writing operations and adjusting velocity to the prescribed field.

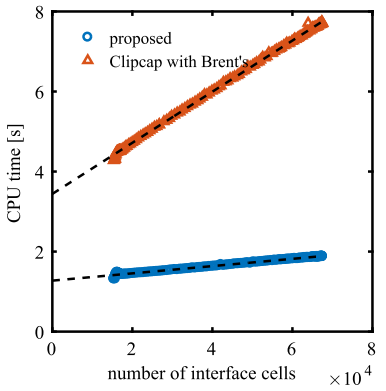


Fig. 13. Sphere deformation case: CPU time consumed by IR done by different methods as a function of number of interfacial cells. The dashed lines represents a linear approximation.

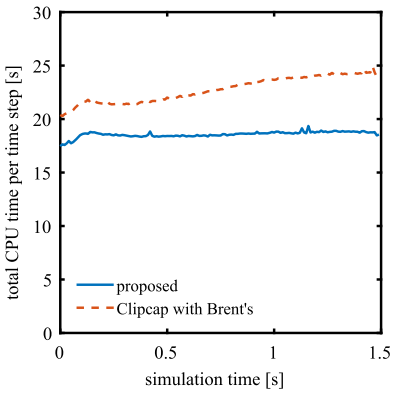


Fig. 14. Sphere deformation case: total CPU time consumed by CLSVOF solver per time step for different IR methods as a function of simulation time.

5.2. IR performance in 'O-Ring' mesh using full CLSVOF simulation

A droplet impacting on a dry surface has been simulated with the full implementation of the CLSVOF model with momentum and pressure solvers and contact angle models, in order to demonstrate the performance of interface reconstruction algorithms in a physical problem with varying size of gas/liquid interface. The experimental data of [31] for the behaviour of a droplet impacting on a dry surface was used and the accuracy of the CLSVOF solver for this test case has been shown in [8]. The mesh used in this simulation is shown in Fig. 15 and consists of a central block of 70 x 70 elements surrounded by four blocks with 42 elements in the radial direction. A total of 80 rows in the vertical direction are used, leading to approximately 1.3M elements. The variable time step option was used with Courant number set to 0.2. The near wall mesh spacing is 0.02 mm with the expansion ratio of 1.05. Note that the mesh elements, while still hexahedral, are no longer cuboid in shape particularly at the edges of the blocks as can be seen in Fig. 15.

The results for this simulation are shown in Fig. 16. The proportion of interfacial cells changes during the simulation from 0.5% to 1.2% (see Fig. 17). The CPU time consumed by the IR step for the two different methods is shown in Fig. 18 as a function of simulation time and in Fig. 19 as a function of the number of interface cells. The dashed lines in Fig. 19 represent linear approximation and show that even if a number of interface cells is 0 the CPU cost is greater than 0, this is because, as discussed above, IR consumes some CPU time regardless of the number of interface cells as some operations are always performed such as calculation of gradient of Level Set field and searching for interface cells.

The proposed IR method was 5.83 times faster per additional interface cell compared to Clipcap with Brent's iteration whereas it was on average 3.82 faster when the entire IR step is considered. The speed up per cell is comparable to those for the case in when momentum and pressure solvers are not used. The contributions of parts of the whole CLSVOF method to the CPU consumption are shown in Fig. 20. The majority of CPU time is spent on solving pressure and velocity equations regardless the IR method used. The IR methods contribute 1% and 5% to the total CPU time for proposed and Clipcap with Brent's method. The overall speed up of the CLSVOF method compared to Clipcap with Brent's iteration for this case is shown in Fig. 21. The average speed up for this case is 1.09 and the maximum speed up seen was 1.16.

The influence of using the result of the previous IR operation for that cell as the initial guess for the next was tested in this drop impingement case. (See Fig. 22.) Using information from previous time step was found to require on average only 4.65 function evaluations in the root-finding process, whereas initializing IR algorithm with a default starting point of 0.5

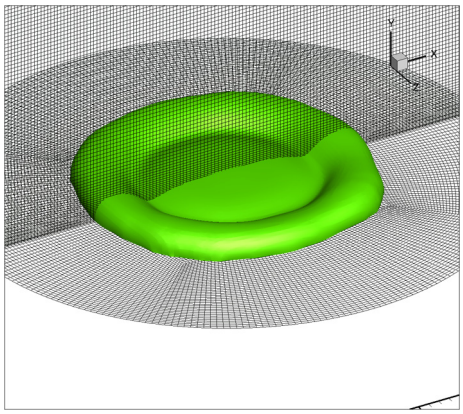


Fig. 15. A droplet impingement at $t = 3.5$ ms when the largest deformation was observed. Details of the mesh are shown.

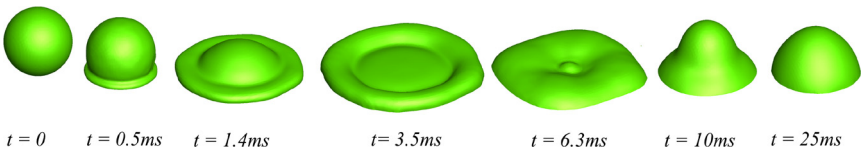


Fig. 16. A droplet impingement: shape variation at different times. The largest area of interface was at $t = 3.5$ ms.

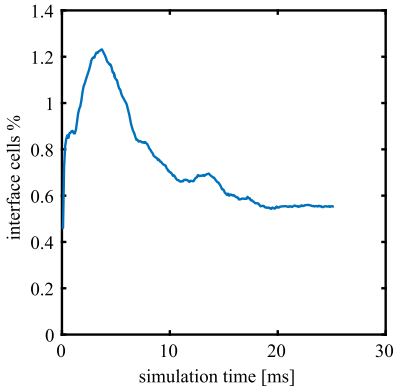


Fig. 17. Number of cells comprising interface as a function of simulation time of drop impingement case.

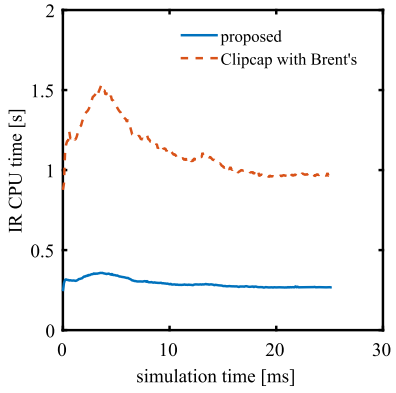


Fig. 18. Drop impingement case: CPU time consumed by IR per time step done by different methods as a function of simulation time.

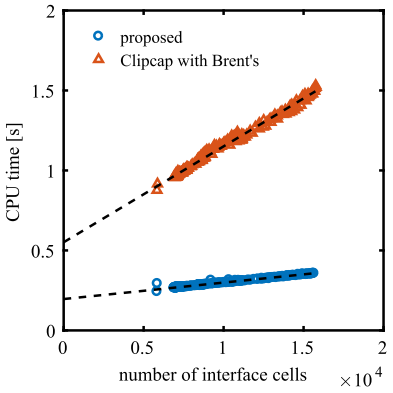


Fig. 19. Drop impingement case: CPU time consumed by IR done by different methods as a function of number of interfacial cells. The dashed lines represents a linear approximation.

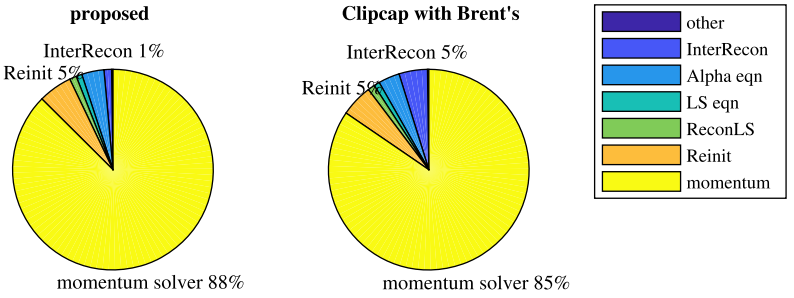


Fig. 20. The pie charts show the contribution to CPU time by different operations during CLSVOF simulation of the droplet impingement case. Two methods were used: proposed, and Clipcap with Brent's iteration. Times are shown for the largest deformation i.e., $t = 3.5$ ms.

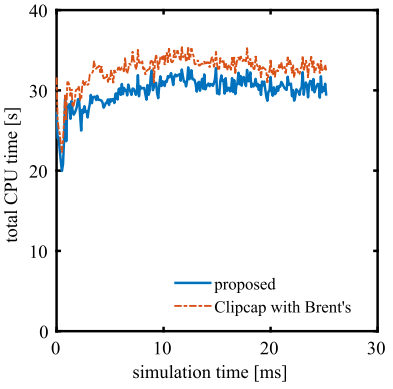


Fig. 21. Drop impingement case: total CPU time consumed by CLSVOF solver per time step with momentum equation for different IR methods as a function of simulation time.

requires on average 8.04 function evaluations which presents a speed-up of 73%. This is the main advantage of having used proposed Muller's–Newton method in comparison to other root-finding methods for IR problem.

5.3. IR performance for rivulet flow simulation with full CLSVOF method

In this section, a simulation of a rivulet flowing into a channel is discussed. The rivulet flows down a dry substrate into a channel placed perpendicular to the direction of flow. This is an example of practical exterior water management application for which a CLSVOF method may be employed [32]. Compared to the previous test this simulation has a larger proportion of interface cells and this proportion grows during the simulation.

This test comprises an inclined solid surface of slope 7° with a channel of width 12.5 mm and height 2.5 mm located 50 mm downstream of the inlet perpendicular to the oncoming flow. The floor downstream of the channel has a length of 22.5 mm. The width in the spanwise direction is 30 mm and the vertical height is 12.5 mm. A mesh of just over a million hexahedral cells has been used. The grid spacing in the proximity of the walls is around 0.1 mm. The mesh is generally

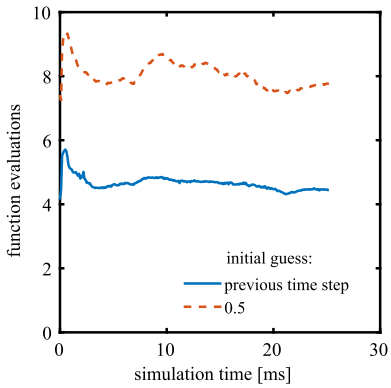


Fig. 22. Drop impingement case: the influence of choice of initial guess in root-finding algorithm in IR step. Two approaches were tested: using a starting guess of 0.5 and using the result from previous time step as an initial approximation for subsequent time step.

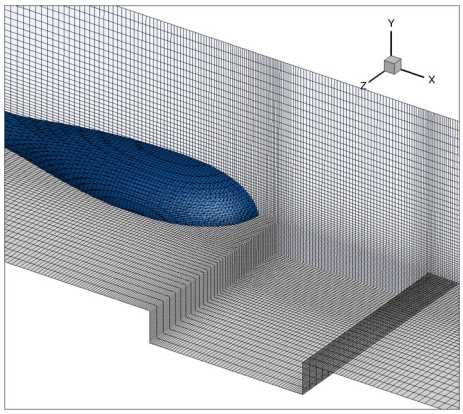


Fig. 23. Details of the mesh used in simulation of rivulet (1M hexahedral cells).

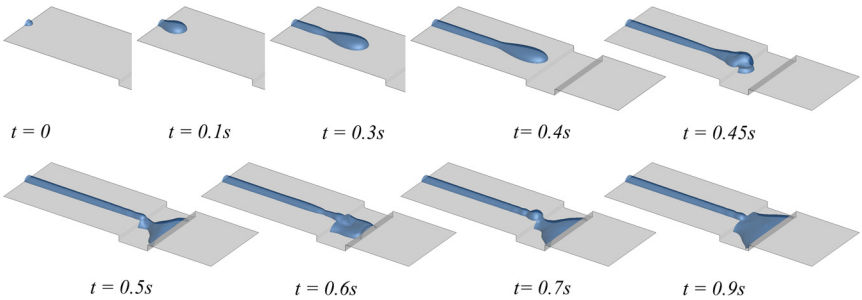


Fig. 24. Simulation of a rivulet flowing to the channel on inclined substrate.

non-uniform being refined near the surface, in the regions close to the channel and in the central region in the spanwise direction where the rivulet flows (see Fig. 23).

The liquid is injected through an inlet at the flow rate of 1 ml/s. The inlet velocity of the rivulet front is $U_L = 16$ cm/s. The inlet area for the rivulet is calculated to be 6.25 mm^2, to satisfy inlet velocity and flow rate. The Hoffman–Voinov–Tanner (HVT) contact angle model was used with the equilibrium contact angle of 80°. To make the shape of the inlet area consistent with the contact angle, a circular cap is used with the same area and the same measured contact angle at the surface.

The results of this simulation are shown in the Fig. 24 at different times. The head of the rivulet approaches the edge of the channel at approximately 0.4 s. After that, water fills the channel and the simulation ends at 0.9 s when the water surface reaches the sides of the computational domain. This case is therefore representative of the sort of changes in interfacial area that may be expected in practical water management problems.

The number of interfacial cells increases from zero to almost 2.5% during the simulation as the rivulet elongates before decreasing slightly as the water pools in the channel (see Fig. 25). The CPU time consumed by the IR step is shown in

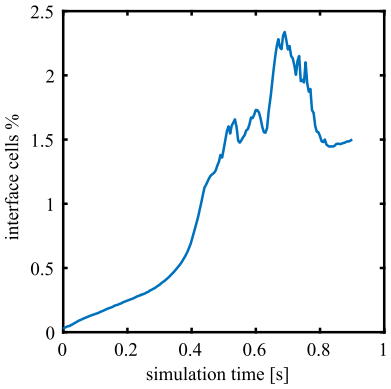


Fig. 25. Number of cells comprising interface as a function of simulation time for the rivulet case.

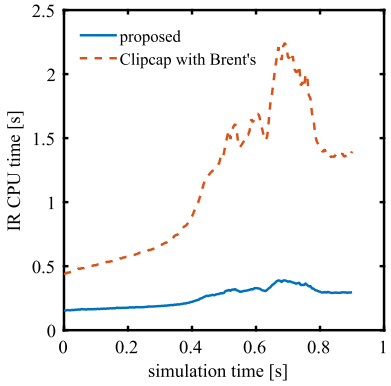


Fig. 26. Rivulet case: CPU time consumed by different IR methods as a function of simulation time.

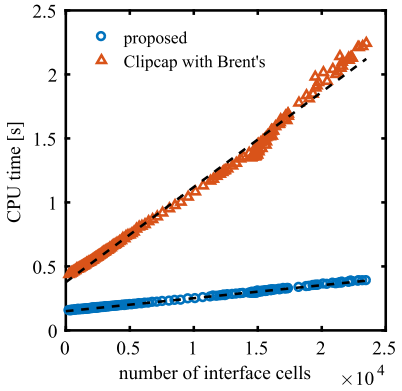


Fig. 27. Rivulet case: CPU time consumed by different IR methods as a function of number of interfacial cells.

Fig. 26 as a function of simulation time and Fig. 27 as a function of number of interface cells. The proposed IR method was 7.32 times faster per additional cell compared to Clipcap with Brent's method, which is very similar to results found in the previous sections. With respect to the entire IR step, the proposed method reduced time on average by a factor of 4.44, which is more than in the case of drop impingement and can by explained by the larger interface. The average speed-up of the entire CLSVOF simulation shown in the Fig. 28 was 1.087 whereas the maximum speed-up was 1.186.

5.4. IR performance for droplet impingement with high velocity on a dry surface

A case of a 4 mm droplet impinging onto a dry surface with a high velocity, $U = 9$ m/s, presents a case with a higher proportion of interfacial cells in contrast to previous examples. The impingement results in a splash with many small satellite droplets produced, as can be seen in the Fig. 29, thus more interface has to be resolved. In this simulation a 3.2M

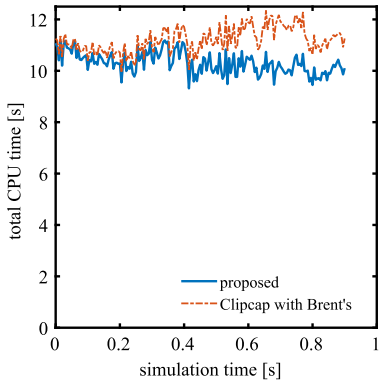


Fig. 28. Rivulet case: total CPU time consumed by CLSVOF solver with momentum equation for different IR methods as a function of simulation time.

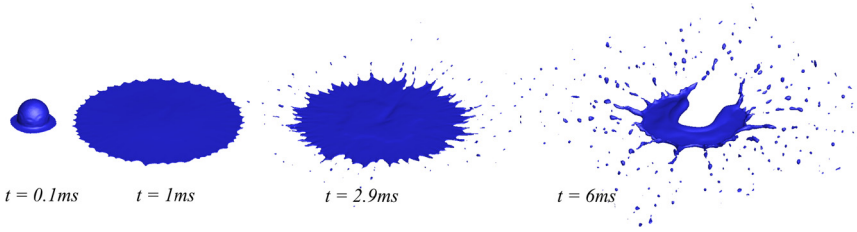


Fig. 29. Simulation of a droplet impingement with high velocity onto dry surface.

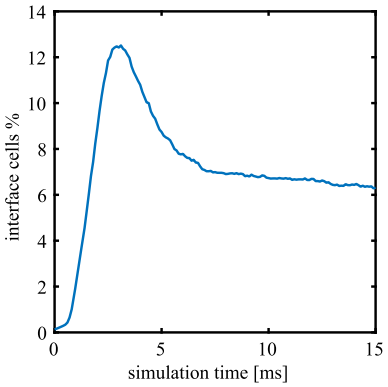


Fig. 30. Number of cells comprising interface as a function of simulation time for the drop impingement with high velocity.

regular hexahedral mesh was used in $0.04 \times 0.005 \times 0.04$ m domain (width×height×length), with mesh refinement near to the wall up to 0.03 mm. The HVT contact angle model was used with the equilibrium contact angle of 135°.

The maximum number of interfacial cells in this simulation was seen at $t = 2.9$ ms and was almost 12.5% (see Fig. 30). As with previous examples, the CPU time consumed by the IR step is shown as a function of simulation time (Fig. 31) and as a function of number of interface cells (Fig. 32). Similarly to results found in the previous sections, the proposed IR method was 5.72 times faster per additional cell compared to Clipcap with Brent's method. With respect to the entire IR step, the proposed method reduced time on average by a factor of 4.95 and average speed-up of the entire CLSVOF simulation shown in the Fig. 33 was 31% whereas the maximum speed-up was 52%. These, higher than in previous sections, values can be explained by the larger proportion of interfacial cells and thus higher contribution of IR to the CPU time of entire simulation.

5.5. Discussion of speed up from the IR method

It is clear that more efficient interface reconstruction methods lead to an improvement in speed of the overall CLSVOF algorithm for the tests considered. However due to the need to carry out other operations, particularly solving momentum and pressure equations, the speed up of the complete solver is significantly reduced compared to tests of the reconstruction algorithm in isolation. The largest speed ups for the complete solver were found to be approximately 52% compared to Clipcap with Brent's iterative method. A speed up of this size would represent a useful reduction in simulation time in a

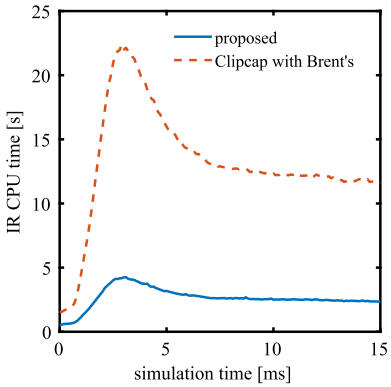


Fig. 31. CPU time consumed by different IR methods as a function of simulation time for the drop impingement with high velocity case.

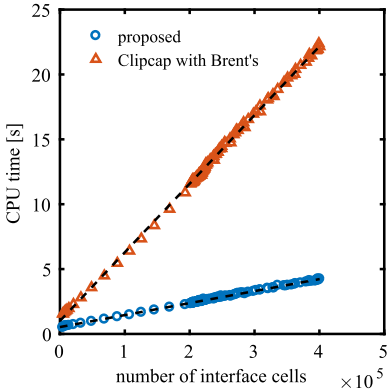


Fig. 32. CPU time consumed by different IR methods as a function of number of interfacial cells in the drop impingement with high velocity case.

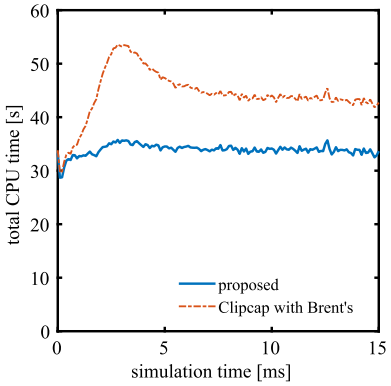


Fig. 33. Total CPU time consumed by CLSVOF solver with momentum equation for different IR methods as a function of simulation time in the drop impingement with high velocity case.

practical problem. The analytical method was not tested in this section, but given its relative performance to the tetrahedral decomposition method for IR in individual cells it is reasonable to assume that it would give a small performance penalty compared to the proposed method for the full CLSVOF solver, with greater complexity in implementation. The difference in run time between the two methods for the IR step is likely to be relatively small compared to other operations required in the solver.

The total speed up of the CLSVOF method when the proposed IR method is used compared to the baseline Clipcap with Brent's method is shown in Fig. 34 as a function of percentage of cells containing interface. It can be seen that for each case (rivulet and splashing) the trend is linear. The different gradient of this speed up line is due to the different work done by solver for these different physical cases. The 50% speed up at high interface percentages shows that in cases which involve a larger proportion of interfacial cells; for example in simulations of liquid atomization (see for example [6,7]) the overall speed up can be significant. To generate this speed up for a mesh with approximately 12% interface cells a speed

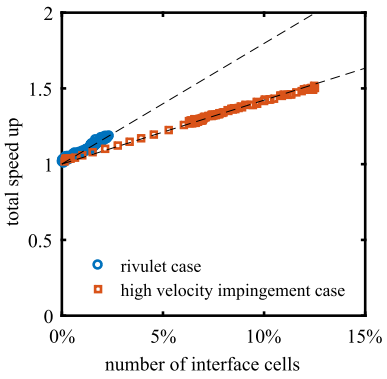


Fig. 34. Speed-up of entire CLSVOF simulation as a function of the number of interface cells for the rivulet and splashing drop cases.

up of around 7.5 in the IR method is needed. For the cases seen here any further improvements in the IR algorithm are unlikely to yield significant improvements in overall speed and more benefits could be obtained by focusing on improving the efficiency of the momentum solver or the reinitialisation solver used to maintain the signed distance property of the level set.

6. Conclusions

A new interface reconstruction method has been proposed and demonstrated on convex cells with up to 20 vertices. It has been shown that the performance of the method is comparable to the analytical approach described in [14] with the proposed method being around two times faster for individual cells. The proposed method is also relatively easy to implement. A significant performance gain over previous, widely used methods for IR (such as Clipcap in [18]) has been achieved by decomposing the cell to tetrahedra prior to the IR step which allows for efficient calculation of the truncated polyhedron volume at each iteration and by using a root-finding method designed purposely for the characteristics of the IR problem.

A number of numerical tests have been provided to assess the numerical performance of different IR methods, i.e. the proposed one, Clipcap with Brent's and an analytical method. Methods were tested on single cells with different geometries as well as in a multiphase flow solver using a CLSVOF approach. For single cell tests, the proposed method was 3.4–11.8 times faster than Clipcap with Brent's method, depending on cell geometry, and 6.8 on average. The method was found to have a similar run time to the analytical method with speed ups of around 2. The average speed-up observed in entire CLSVOF simulations compared to Clipcap was found to be approximately 9% in drop impingement case and rivulet case and 31% in drop impingement with high velocity. The biggest instantaneous speed-up of 52% compared to Clipcap was achieved when interface cells cover 12.5% of the mesh, which was the highest observed interface percentage, in the splashing drop case. The relatively small overall speed up compared to the IR speed up for individual cells leads to the conclusion that further improvements in IR methods would not necessarily bring significant benefits in CPU cost of the entire simulation. It has been also shown that geometrical operations related to the proposed method have a small contribution to CPU cost of the entire simulation. Therefore, we can also conclude that significant gains in accuracy can be achieved by using methods based on geometric VOF-PLIC over algebraic VOF methods with relatively little effort and computational penalty.

Acknowledgements

M. Skarysz has been financially supported by a Jaguar Land Rover/UK Engineering and Physical Sciences Research Council (EPSRC) industrial CASE studentship ref 1689977 which is gratefully acknowledged by the authors. The authors would also like to acknowledge the help of Mr Adrian Gaylard of Jaguar Land Rover and Prof Jim McGuirk for comments on the manuscript.

Appendix A. Optimal tetrahedral decomposition of selected geometries

In this appendix, we present an optimal (minimal) tetrahedral decomposition of geometries used in this paper, namely, prism, hexahedron, 10 vertices polyhedron constructed as a hexahedron with two pyramids attached to two opposite faces; rhombic dodecahedron; regular icosahedron; and regular dodecahedron. The optimal decomposition was found by using the C++ program provided by de Loera [24] with the final step done in CPLEX optimization tool [33]. This decomposition algorithm of [24] consists of using interior cocircuit equations of the polyhedron to solve linear programming problem. A decomposition is described as a list of tetrahedron where tetrahedron is a set of four indices of original polyhedron. We recall vertices coordinates with appropriate numbers:

Prism

$p_1 = (0, 0, 0)$ $p_2 = (1, 0, 0)$ $p_3 = (0, 1, 0)$
$p_4 = (0, 0, 1)$ $p_5 = (1, 0, 1)$ $p_6 = (0, 1, 1)$

Decomposition into 3 tetrahedrons:

(1 2 3 4) (2 3 4 6) (2 4 5 6)

Hexahedron (cube)

$p_1 = (0, 0, 0)$ $p_2 = (1, 0, 0)$ $p_3 = (1, 1, 0)$ $p_4 = (0, 1, 0)$
$p_5 = (0, 0, 1)$ $p_6 = (1, 0, 1)$ $p_7 = (1, 1, 1)$ $p_8 = (0, 1, 1)$

Decomposition into 5 tetrahedrons:

(1 5 6 8) (3 6 7 8) (1 3 4 8) (1 2 3 6) (1 3 6 8)

10 vertices polyhedron

$p_1 = (-.25, -.5, -.5)$ $p_2 = (.25, -.5, -.5)$ $p_3 = (.25, .5, -.5)$
$p_4 = (-.25, .5, -.5)$ $p_5 = (-.25, -.5, .5)$ $p_6 = (.25, -.5, .5)$
$p_7 = (.25, .5, .5)$ $p_8 = (-.25, .5, .5)$
$p_9 = (-.75, 0, 0)$ $p_{10} = (.75, 0, 0)$

Decomposition into 9 tetrahedrons:

(1 2 3 6) (1 3 4 8) (1 3 6 8)
(1 5 6 8) (3 6 7 8) (1 4 8 9)
(1 5 8 9) (2 3 6 10) (3 6 7 10)

Rhombic dodecahedron

$p_1 = (1, 0, 0)$ $p_2 = (0, 1, 0)$ $p_3 = (0, 0, 1)$
$p_4 = (-1, 0, 0)$ $p_5 = (0, -1, 0)$ $p_6 = (0, 0, -1)$
$p_7 = (.5, .5, .5)$ $p_8 = (.5, .5, -.5)$ $p_9 = (.5, -.5, .5)$
$p_{10} = (.5, -.5, -.5)$ $p_{11} = (-.5, .5, .5)$ $p_{12} = (-.5, .5, -.5)$
$p_{13} = (-.5, -.5, .5)$ $p_{14} = (-.5, -.5, -.5)$

Decomposition into 12 tetrahedrons:

(1 2 3 4) (1 3 4 5) (1 2 4 6) (1 4 5 6) (1 2 3 7) (1 2 6 8)
(1 3 5 9) (1 5 6 10) (2 3 4 11) (2 4 6 12) (3 4 5 13) (4 5 6 14)

Regular icosahedron

$p_1 = (\phi, 1, 0)$ $p_2 = (\phi, -1, 0)$ $p_3 = (-\phi, 1, 0)$
$p_4 = (-\phi, -1, 0)$ $p_5 = (0, \phi, 1)$ $p_6 = (0, \phi, -1)$
$p_7 = (0, -\phi, 1)$ $p_8 = (0, -\phi, -1)$ $p_9 = (1, 0, \phi)$
$p_{10} = (1, 0, -\phi)$ $p_{11} = (-1, 0, \phi)$ $p_{12} = (-1, 0, -\phi)$

where $\phi = (1 + \sqrt{5})/2$.
Decomposition into 15 tetrahedrons:

(1 3 5 6) (2 3 7 8) (3 4 7 8) (1 2 3 9) (1 3 5 9)
(2 3 7 9) (1 2 3 10) (1 3 6 10) (2 3 8 10) (3 4 7 11)
(3 5 9 11) (3 7 9 11) (3 4 8 12) (3 6 10 12) (3 8 10 12)

Regular dodecahedron

$p_1 = (0, \phi^{-1}, \phi)$ $p_2 = (0, \phi^{-1}, -\phi)$ $p_3 = (0, -\phi^{-1}, \phi)$
$p_4 = (0, -\phi^{-1}, -\phi)$ $p_5 = (\phi, 0, \phi^{-1})$ $p_6 = (\phi, 0, -\phi^{-1})$
$p_7 = (-\phi, 0, \phi^{-1})$ $p_8 = (-\phi, 0, -\phi^{-1})$ $p_9 = (\phi^{-1}, \phi, 0)$
$p_{10} = (\phi^{-1}, -\phi, 0)$ $p_{11} = (-\phi^{-1}, \phi, 0)$ $p_{12} = (-\phi^{-1}, -\phi, 0)$
$p_{13} = (1, 1, 1)$ $p_{14} = (1, -1, 1)$ $p_{15} = (1, -1, -1)$
$p_{16} = (1, 1, -1)$ $p_{17} = (-1, 1, 1)$ $p_{18} = (-1, -1, 1)$
$p_{19} = (-1, -1, -1)$ $p_{20} = (-1, 1, -1)$

where $\phi = (1 + \sqrt{5})/2$.
Decomposition into 23 tetrahedrons:

(2 3 4 6)	(2 3 4 8)	(2 3 6 9)	(3 5 6 9)	(3 4 6 10)	(3 5 6 10)
(2 3 8 11)	(3 7 8 11)	(2 3 9 11)	(3 4 8 12)	(3 7 8 12)	(3 4 10 12)
(3 5 9 13)	(1 3 11 13)	(3 9 11 13)	(3 5 10 14)	(4 6 10 15)	(2 6 9 16)
(1 3 11 17)	(3 7 11 17)	(3 7 12 18)	(4 8 12 19)	(2 8 11 20)	

References

[1] T. Maric, H. Marschall, D. Bothe, voFoam-A geometrical Volume of Fluid algorithm on arbitrary unstructured meshes with local dynamic adaptive mesh refinement using OpenFOAM, Prepr., pp. 1–30, arXiv:1305.3417, 2013.
[2] R. Scardovelli, S. Zaleski, Direct numerical simulation of free-surface and interfacial flow, Annu. Rev. Fluid Mech. 31 (1999) 567–603.
[3] V. Dyadechko, M. Shashkov, Moment-of-fluid interface reconstruction, Math. Model. Anal. 836 (2005) 1–41.
[4] M. Jemison, E. Loch, M. Sussman, M. Shashkov, M. Arienti, M. Ohta, Y. Wang, A coupled level set-moment of fluid method for incompressible two-phase flows, J. Sci. Comput. 54 (2013) 454–491.
[5] M. Sussman, E.G. Puckett, A coupled level set and volume-of-fluid method for computing 3D and axisymmetric incompressible two-phase flows, J. Comput. Phys. 162 (2000) 301–337.
[6] F. Xiao, M. Dianat, J.J. McGuirk, Large eddy simulation of single droplet and liquid jet primary breakup using a coupled level set/volume of fluid method, At. Sprays 24 (2014) 281–302.
[7] F. Xiao, M. Dianat, J. McGuirk, Les of turbulent liquid jet primary breakup in turbulent coaxial air flow, Int. J. Multiph. Flow 60 (2014) 103–118.
[8] M. Dianat, M. Skarysz, A. Garmory, A Coupled Level Set and Volume of Fluid method for automotive exterior water management applications, Int. J. Multiph. Flow 91 (2017) 19–38.
[9] W.J. Rider, D.B. Kothe, Reconstructing volume tracking, J. Comput. Phys. 141 (1998) 112–152.
[10] R.P. Brent, Algorithms for Minimization Without Derivatives, Prentice-Hall, Englewood Cliffs, N.J., 1973.
[11] R. Scardovelli, S. Zaleski, Analytical relations connecting linear interfaces and volume fractions in rectangular grids, J. Comput. Phys. 164 (2000) 228–237.
[12] X. Yang, A.J. James, Analytic relations for reconstructing piecewise linear interfaces in triangular and tetrahedral grids, J. Comput. Phys. 214 (2006) 41–54.
[13] S. Diot, M.M. François, E.D. Dendy, An interface reconstruction method based on analytical formulae for 2D planar and axisymmetric arbitrary convex cells, J. Comput. Phys. 275 (2014) 53–64.
[14] S. Diot, M.M. François, An interface reconstruction method based on an analytical formula for 3D arbitrary convex cells, J. Comput. Phys. 305 (2016) 63–74.
[15] J. López, J. Hernández, Analytical and geometrical tools for 3D volume of fluid methods in general grids, J. Comput. Phys. 227 (2008) 5939–5948.
[16] J. López, J. Hernández, P. Gómez, F. Faura, A new volume conservation enforcement method for PLIC reconstruction in general convex grids, J. Comput. Phys. 316 (2016) 338–359.
[17] OpenFOAM, v. 2.1.1, OF Build: 2.1.1-221db2718bbb, www.openfoam.org.
[18] H.T. Ahn, M. Shashkov, Multi-material interface reconstruction on generalized polyhedral meshes, J. Comput. Phys. 226 (2007) 2096–2132.
[19] R.N. Goldman, Area of planar polygons and volume of polyhedra, in: Graphics Gems II, vol. 2, 1991, pp. 170–171.
[20] D.E. Muller, A method for solving algebraic equations using an automatic computer, Math. Tables Other Aids Comput. 10 (1956) 208–215.
[21] H. Antia, Numerical Methods for Scientists and Engineers, 2nd ed., Birkhäuser, Basel, 2002.
[22] F.Y. Chin, S.P. Fung, C.-A. Wang, Approximation for minimum triangulations of simplicial convex 3-polytopes, Discrete Comput. Geom. 26 (2001) 499–511.
[23] H. Edelsbrunner, F.P. Preparata, D.B. West, Tetrahedrizing point sets in three dimensions, J. Symb. Comput. 10 (1990) 335–347.
[24] J.A. de Loera, S. Hosten, F. Santos, B. Sturmfels, The polytope of all triangulations of a point configuration, Doc. Math. 1 (1996) 119.
[25] M. Bern, D. Eppstein, Mesh generation and optimal triangulation, Comput. Euclidean Geom. 1 (1992) 23–90.
[26] B. Chazelle, L. Palios, Triangulating a nonconvex polytope, Discrete Comput. Geom. 5 (1990) 505–526.
[27] J. Ruppert, R. Seidel, On the difficulty of triangulating three-dimensional Nonconvex Polyhedra, Discrete Comput. Geom. 7 (1992) 227–253.
[28] A. Katz, V. Sankaran, Mesh quality effects on the accuracy of cfd solutions on unstructured meshes, J. Comput. Phys. 230 (2011) 7670–7686.
[29] R.J. LeVeque, High-resolution conservative algorithms for advection in incompressible flow, SIAM J. Numer. Anal. 33 (1996) 627–665.
[30] T. Ménard, S. Tanguy, A. Berlemont, Coupling level set/VOF/ghost fluid methods: validation and application to 3D simulation of the primary break-up of a liquid jet, Int. J. Multiph. Flow 33 (2007) 510–524.
[31] K. Yokoi, D. Vadillo, J. Hinch, I. Hutchings, Numerical studies of the influence of the dynamic contact angle on a droplet impacting on a dry surface, Phys. Fluids 21 (2009).
[32] M. Dianat, M. Skarysz, G. Hodgson, A. Garmory, M.A. Passmore, Coupled level-set volume of fluid simulations of water flowing over a simplified drainage channel with and without air coflow, SAE Int. J. Passeng. Cars 2017 (2017) 1–8.
[33] A. Adfgh, IBM ILOG CPLEX User's Manual, IBM, Tech. Rep, 2015.