Computer Physics Communications 281 (2022) 108506

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

An optimized Vofi library to initialize the volume fraction field ☆,☆☆

A. Chierici [a,b], L. Chirco [a,c], V. Le Chenadec [d], R. Scardovelli [a,*], Ph. Yecko [e], S. Zaleski [c,f]

[a] DIN – Lab. di Montecuccolino, Università di Bologna, Via dei Colli 16, 40136 Bologna, Italy
[b] Department of Mathematics and Statistics, Texas Tech University, Lubbock, TX 79409, USA
[c] Sorbonne Université & CNRS, UMR 7190, Institut Jean Le Rond d'Alembert, Paris, F-75005, France
[d] MSME, Univ. G. Eiffel, UMR CNRS 8208, Marne-la-Vallée, 77454, France
[e] Physics Department, Cooper Union, New York, NY, USA
[f] Institut Universitaire de France (IUF), Paris, France

ARTICLE INFO

Article history:
Received 6 May 2021
Received in revised form 1 August 2022
Accepted 16 August 2022
Available online 22 August 2022

Keywords:
Implicit function
Numerical integration
Volume fraction function
VOF method

ABSTRACT

The Vofi (Volume Of Fluid Initializer) library has been developed to initialize the volume fraction field determined by implicitly-defined interfaces. The major conceptual changes in the numerical algorithms of the library are discussed and a few new features, including the computation of the reference phase centroid and of the interface length/area, are presented and applied to grid cells that are cuboids. Several numerical tests are considered to demonstrate both the accuracy of the new features, as the grid resolution and the number of integration points are varied, and the considerably improved efficiency of the library with respect to its previous version. A few of these tests are also included in the software distribution written in `C`, examples of `C++` and `Fortran` interfaces are also provided.

Program summary
Program title: Vofi – Volume Of Fluid Initializer
CPC Library link to program files: https://doi.org/10.17632/mbmzpbfxdz.1
Code Ocean capsule: https://codeocean.com/capsule/3817905
Licensing provisions: GPLv3
Programming language: C
Journal reference of previous version: Comput. Phys. Commun. 200 (2016) 291-299
Does the new version supersede the previous version?: Yes.
Reasons for the new version: Optimization of the library and new features has been added.
Summary of revisions: Most of the routines have been rewritten, several numerical algorithms have been revised, as detailed in the paper, added features include the computation of the reference phase centroid and of the interface length in 2D and area in 3D; furthermore heights and triangulation data can now be printed for graphics.
Nature of problem: The Vofi library computes the volume fraction of a cuboid cut by an interface described by a user-defined implicit function, and optionally the centroid of the cut volume and the area (length in 2D) of the portion of the interface inside the cell.
Solution method: The Vofi library reorders the three Cartesian directions, x, y, z, in ascending order, x_3, x_2, x_1, computes the integration limits along the third and second directions, respectively x_3 and x_2, and determines the local height function, along x_1, that is the integrand of a double Gauss-Legendre integration with a variable number of nodes. Optionally, the same heights are used to compute the centroid of the cut volume and to triangulate the interface.

1. Introduction

Direct numerical simulation (DNS) is a popular method for detailed studies of multiphase flows which are encountered in many scientific disciplines, such as thermo-fluid dynamics, oceanography, chemistry and engineering. The Volume-of-Fluid (VOF) method is one of the numerical techniques that is used to predict the evolution of interfaces in multiphase flows within the framework of DNS

☆ The review of this paper was arranged by Prof. David W. Walker.
☆☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).
* Corresponding author.
E-mail address: ruben.scardovelli@unibo.it (R. Scardovelli).

https://doi.org/10.1016/j.cpc.2022.108506
0010-4655/

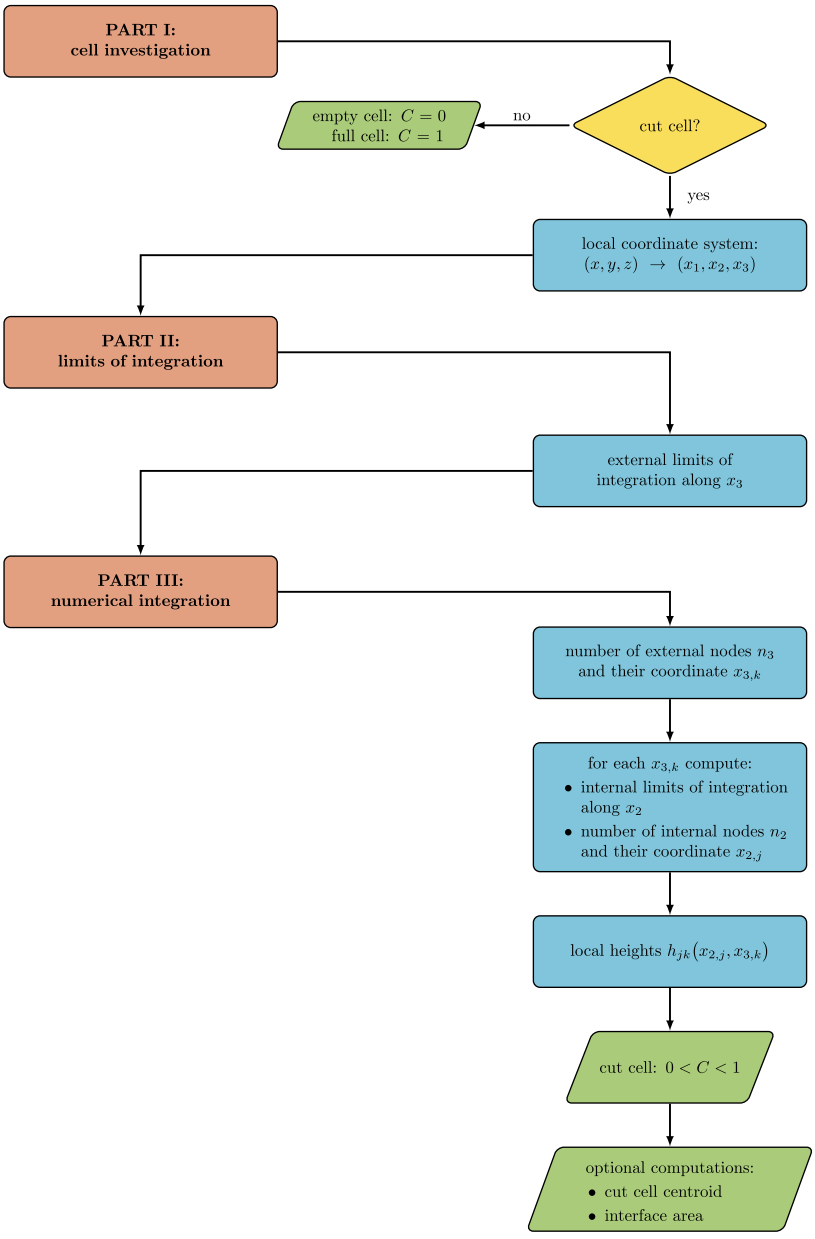

Fig. 1. A graphical representation of the VOFI algorithm in three dimensions.

[1,2]. The method is based on the characteristic function $\chi(\mathbf{x}, t)$, a multidimensional Heaviside step function, with value 1 in the reference phase and 0 elsewhere, at time t. The integral of the function χ over a computational domain D provides the volume of the portion of D that is occupied by the reference phase. The volume fraction C is associated to the function χ and it represents the fraction of a grid cell that is occupied by the reference phase, as defined by Eq. (2).

An accurate representation of the volume fraction field is necessary for an exact initialization of the total mass at the beginning of a numerical simulation. It is also required to compute convergence rates with grid spacing of geometric properties of the interface, such as its normal vector and curvature, and of instability growth rates [3]. In a previous paper [4], we have presented the first release of the VOFI library that can be used to initialize the volume fraction C in a computational grid with cubic cells of edge l_0, given a user-defined implicit equation of the interface, $f(\mathbf{x}) = 0$. The library is based on two main assumptions: 1) in a cell cut by the interface the implicit equation can be locally written explicitly as $x_1 = h(x_2, x_3)$, where each x_i is one of the three coordinate directions; 2) the interface cannot intersect a cell edge more than twice. Both assumptions are verified if the local radius of curvature R of the interface is greater than the spatial step l_0.

The VOFI algorithm is divided into three parts, as shown in the flowchart of Fig. 1. In the first part each grid cell is investigated to determine if it is empty, full or cut by the interface. In a cut cell, the function gradient is estimated in the cell with finite differences to change from the reference system with coordinates (x, y, z) to a local coordinate system with (x_1, x_2, x_3). In the second part the limits of integration are computed. In the third part the local height function $h(x_2, x_3)$ is computed numerically along the first direction x_1 at each node (x_2, x_3) and the cut volume is computed with a double Gauss-Legendre quadrature rule. The function h satisfies the relation $0 \leq h \leq l_0$. The internal integration is performed along direction x_2 with n_2 nodes, while the external integration along direction x_3 with n_3 nodes.

In this paper we discuss the major changes in the numerical algorithms, demonstrate the improved efficiency of the library with

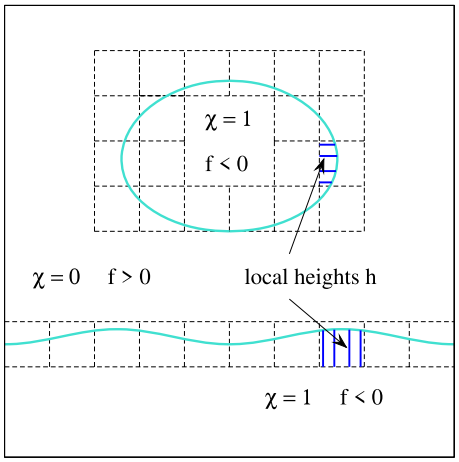

Fig. 2. A droplet impacting on a wavy interface in two dimensions. Inside the reference phase $\chi = 1$ and $f < 0$, in the secondary phase $\chi = 0$ and $f > 0$. In the cells crossed by the interface the local heights h are used to compute numerically the cut area.

respect to its previous version, and present a few new features. Most of the routines have been completely rewritten in order to considerably reduce the number of calls of the implicit function, where the CPU-time is mostly spent. The grid cell analysis and the numerical integration scheme are based on different algorithms, while the new features include the computation of the reference phase centroid and of the interface length in 2D and area in 3D. Moreover, grid cells now can be cuboids.

The library is written in C, as was the first release of VOFI, and a few two-dimensional and three-dimensional tests are provided with the software distribution, including visualization of the height data and interface triangulation. A version of the numerical tests, written both in Fortran and in C++, is also included to show how to call the library routines from these two languages.

2. Revised numerical algorithms and new features

Let $D \subset R^n$ be a computational domain and Ω the portion occupied by the reference phase

$$\Omega = \{\mathbf{x} \in D : f(\mathbf{x}) \leq 0\}\,, \tag{1}$$

where points on the interface satisfy $f(\mathbf{x}) = 0$, and $f(\mathbf{x}) < 0$ for points in the interior of Ω. The interface Γ is a planar curve when $n = 2$, and a surface when $n = 3$. The characteristic function $\chi(\mathbf{x}, t_0)$ at the initial time t_0 is equal to 1 inside Ω and 0 elsewhere.

We consider a Cartesian subdivision of the domain D with cuboids with edges of length l_x, l_y and l_z. The edges length may change from cell to cell. The volume fraction $C(t_0)$ is defined by the integral

$$C(t_0) = \frac{1}{V_0} \int_V \chi(\mathbf{x}, t_0)\, dV\,, \tag{2}$$

where V is the grid cell and V_0 its volume, $V_0 = l_x l_y l_z$. We use the implicit function $f(\mathbf{x})$ to determine the cell type. If the cell is empty or full with respect to the reference phase, the function $f(\mathbf{x})$ does not change its sign inside the cell, and the integration in (2) is straightforward. On the other hand if the interface cuts the cell, we compute the limits of integration and perform the integration to evaluate the measure of the volume delimited by the cell boundary and the interface itself.

A two-dimensional configuration is shown in Fig. 2 to illustrate the relation among the different variables. The reference phase,

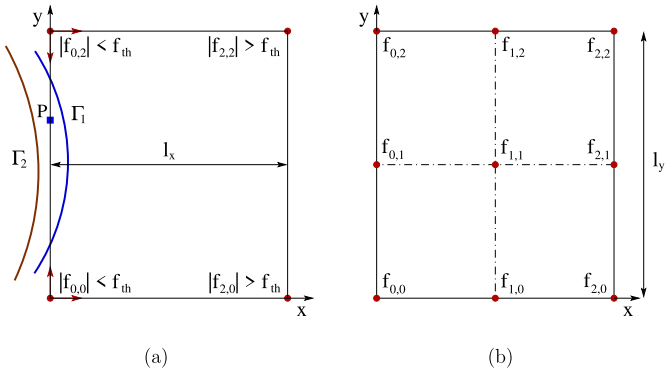

Fig. 3. A two–dimensional cell with edges l_x and l_y: (a) the function value is positive at the four vertices for both interface lines, but only interface Γ_1 cuts the vertical edge; (b) the function value is computed at the nodes of a local 3×3 subgrid to change from a system with coordinates (x, y) to (x_1, x_2).

where the characteristic function χ is equal to 1, is inside the droplet and below the wavy interface. The implicit function f is negative inside the reference phase and its sign determines whether a cell with no interface is empty or full. In the cells cut by the interface local heights h are computed to approximate the two-dimensional integral of Eq. (2) with a one-dimensional numerical integration.

2.1. Cell analysis

In the first release of the library the user was required to call a routine to estimate a threshold value f_{th}, a characteristic function value for the whole interface. This routine has been removed from the library and the value f_{th} is now computed in each cell. In the two-dimensional example of Fig. 3a the function is evaluated at the N_{ve} vertices of the cell ($N_{ve} = 2^2$ in 2D, $N_{ve} = 2^3$ in 3D) to approximate the components of the local gradient $\boldsymbol{\nabla} f$ with centered finite differences

$$\begin{aligned} \frac{\partial f}{\partial x} &\approx \frac{(f_{2,2} + f_{2,0}) - (f_{0,2} + f_{0,0})}{2 l_x} \\ \frac{\partial f}{\partial y} &\approx \frac{(f_{2,2} + f_{0,2}) - (f_{2,0} + f_{0,0})}{2 l_y}\,. \end{aligned} \tag{3}$$

The threshold value is then given by the expression $f_{th} = l_m\, |\boldsymbol{\nabla} f|$, where $l_m = \max(l_x, l_y)/2$ is the maximum distance of a point on the cell boundary from a vertex. Afterwards, we compute f_{min}, the minimum in absolute value of the N_{ve} function values. If the function has the same sign at the vertices and $f_{min} > f_{th}$ then the cell is either full, when its sign is negative, or empty. In 3D the approximation of the local gradient follows from (3) in a straightforward manner, for example

$$\frac{\partial f}{\partial z} \approx \frac{(f_{2,2,2} + f_{0,2,2} + f_{2,0,2} + f_{0,0,2}) - (f_{2,2,0} + f_{0,2,0} + f_{2,0,0} + f_{0,0,0})}{4 l_z}\,, \tag{4}$$

and $l_m = \max(l_x, l_y, l_z)/\sqrt{2}$. For empty or full cells with $f_{min} > f_{th}$ the analysis ends with 2^n function evaluations, whereas in the previous release of the library the function was evaluated at $N_{no} = 3^n$ nodes of a local submesh, as shown in Fig. 3b for $n = 2$.

More computations are required when the cell is closer to the interface and $f_{min} < f_{th}$. The two points in Fig. 3a with positive values $f_{0,0}$ and $f_{0,2}$ satisfy that inequality and we have to check if the configuration is compatible with a sign change in the function value. We consider the lower-left vertex $\mathbf{x}_0$, the unit horizontal vector $\mathbf{i}_x$, and compute $f_\alpha = f(\mathbf{x}_0 + \mathbf{i}_x\, \varepsilon)$, with $\varepsilon \ll l_x$. The following condition is verified

$$|f_\alpha| > |f_{0,0}|\,, \tag{5}$$

therefore we assume that the interface does not cut the horizontal edge. As a matter of fact, condition (5) implies that points along the horizontal edge are initially moving away from the interface, then a sign change in the function value along that edge would be associated with a perturbation in the interface line with a characteristic lengthscale smaller than l_x. This feature is not consistent with the requirement that the relevant lengthscales of the implicit function should be a few times greater than the spatial step.

Afterwards, along the vertical edge we compute $f_\beta = f(\mathbf{x}_0 + \mathbf{i}_y\,\varepsilon)$, but now $|f_\beta| < |f_{0,0}|$. In these conditions the analysis proceeds with a minimum search along that edge. For the interface Γ_1 a sign change is detected at point P, therefore the interface intersects the edge and the search is stopped; for the interface line Γ_2 a positive minimum is found and the edge is not cut by the interface. In 3D we check again for a sign change along the cell edges, but we have to consider the cell faces as well. The minimum search routines along a cell edge and on a cell face have not been changed and we refer to [5] for their detailed description.

Next we consider a cut cell and we assume that the implicit function $f(\mathbf{x}) = 0$ can be written in the explicit form $x_1 = h(x_2, x_3)$, where each x_i is a different coordinate direction. In the previous release of the library we computed the gradient with centered finite differences in the N_{pt} points of the local submesh of Fig. 3b where the function value f satisfied $|f| < f_{th}$, with $1 \le N_{pt} \le N_{no}$. The numerical computation of a gradient required four more function evaluations in 2D and six in 3D. The cell gradient was then defined as the average of the point gradients, and the main coordinate direction x_1 was associated to the cell gradient component with the greatest absolute value, while the second and third directions, x_2 and x_3, to the other two components in decreasing absolute value.

In the new release, we consider again the function values at the N_{no} nodes of the submesh, but the cell gradient is now the average of the gradients computed in the 4 subcells of Fig. 3b (8 subcells in 3D) with centered finite differences and no further function evaluation. To decrease the influence of nodes far from the interface, we assign a weight w to the gradient components in each subcell. For example for the lower-left subcell of Fig. 3b we have

$$\begin{aligned}\frac{\partial f}{\partial x} &\approx w\,\frac{(f_{1,1}+f_{1,0})-(f_{0,1}+f_{0,0})}{l_x}\\ \frac{\partial f}{\partial y} &\approx w\,\frac{(f_{1,1}+f_{0,1})-(f_{1,0}+f_{0,0})}{l_y}\,.\end{aligned} \tag{6}$$

The value of the weight w is related to the number of sign changes of the function at the subcell vertices and is given in a lookup table. For the cell of Fig. 3a and interface Γ_1 the first direction x_1 is along the x-axis, the second direction x_2 along the y-axis.

The final step in the analysis of a cut cell is the determination of a tentative number of integration nodes n_*, which is now based on an estimate of the local curvature κ. For an implicit equation of the interface line the curvature κ in absolute value is given by the expression

$$|\kappa| = \frac{\left|-f_y^2 f_{xx} + 2 f_x f_y f_{xy} - f_x^2 f_{yy}\right|}{(f_x^2+f_y^2)^{3/2}} = \frac{1}{R}\,, \tag{7}$$

involving the first and second partial derivatives of the function f, with R being the local radius of curvature. We approximate the partial derivatives with centered finite differences in a nondimensional fashion by defining a characteristic length l_c. For example the two terms f_x and f_{xy} are approximated by the expressions

$$f_x \approx \frac{(f_{2,1}-f_{0,1})}{l_x}\,l_c\,, \quad f_{xy} \approx \frac{(f_{2,2}-f_{0,2})-(f_{2,0}-f_{0,0})}{l_x\,l_y}\,l_c^2\,. \tag{8}$$

We have considered three different expressions for l_c that provide the same value in the case of square cells

$$l_c = \sqrt{l_x\,l_y}\,, \quad l_c = (l_x + l_y)/2\,, \quad l_c = \max(l_x, l_y)\,,$$

and several elongation ratios l_x/l_y. The geometric mean provides a tentative number of nodes n_* very close to the value obtained with square cells, while the third expression considerably overestimates it. We have decided to select the intermediate expression, i.e. the arithmetic mean, because of the elongation of the cell and the mild overestimate. In 3D we compute the curvature on the two cell faces perpendicular to the third direction x_3 and in the midplane between them and then average the results. We have considered circles and spheres of different radius, center position and resolution of the unit domain for a total of 1200 test cases in both 2D and 3D. For each test we have computed the error E in absolute value between the analytical value of the area/volume and the numerical integration with a fixed number of points in each cell. We have fixed a target error $E \approx 10^{-14}$ in 2D and $E \approx 10^{-12}$ in 3D and computed a cubic relation with a least-squares method between the nondimensional curvature $|\kappa_*|$ and the number of integration nodes n_*. These two relations have been implemented in the library and are shown in Fig. 4.

2.2. Internal and external limits of integration

In 2D the limits of integration are defined by the intersections of the interface line with the cell edges along the second direction x_2. A single intersection is computed by the root-finding routine, a double intersection, as shown on the right of Fig. 5, requires the detection of a sign change at point P and then the computation of the two zeros x_{21} and x_{22}. The measure of the colored area in the central rectangle is computed with a numerical integration.

In 3D the internal limits of integration are computed as in the 2D problem, while the external limits of integration include the intersections of the interface with the four cell edges along the third direction x_3. However, there is also the possibility of a very small intersection of the interface with a cell that is represented by the cap on the left of Fig. 5, with the two external limits x_{31} and x_{32}, that are computed starting from point S.

To illustrate the changes in the new release we consider the ellipsoid $f(x', y', z') = (x'/4)^2 + (y'/5)^2 + (z'/6)^2 - 1 = 0$. The axis z' is parallel to the coordinate axis z, while the other two axis x' and y' are rotated 60 degrees counterclockwise with respect to x and y. The ellipsoid center in Fig. 6a is at $(0.35, 0.35, -5.97)$ and the ellipse represents the intersection of its surface with the plane $z = 0$, while in Fig. 6b the center is at $(0.26, 0.26, -5.97)$. In both cases the cap height is $\Delta h = 0.03$ and the grid cells are cubes of edge length $l_0 = 1$.

In the top-right cell of Fig. 6a the function is positive at the four vertices of the face at $z = 0$ and the minimum search routine detects a sign change at point S. Another routine, which is based on the root-finding algorithm, is then called to get a sequence of approximations converging to the external limit x_{33}. This routine has been discussed in [5] and the first two steps of the iterative procedure are shown graphically in Fig. 6a. The other two integration limits x_{31} and x_{32} are found as intersections of the interface with a cell edge.

In the top-right cell of Fig. 6b the function value is negative at the bottom-left vertex and positive at the other three. In the first

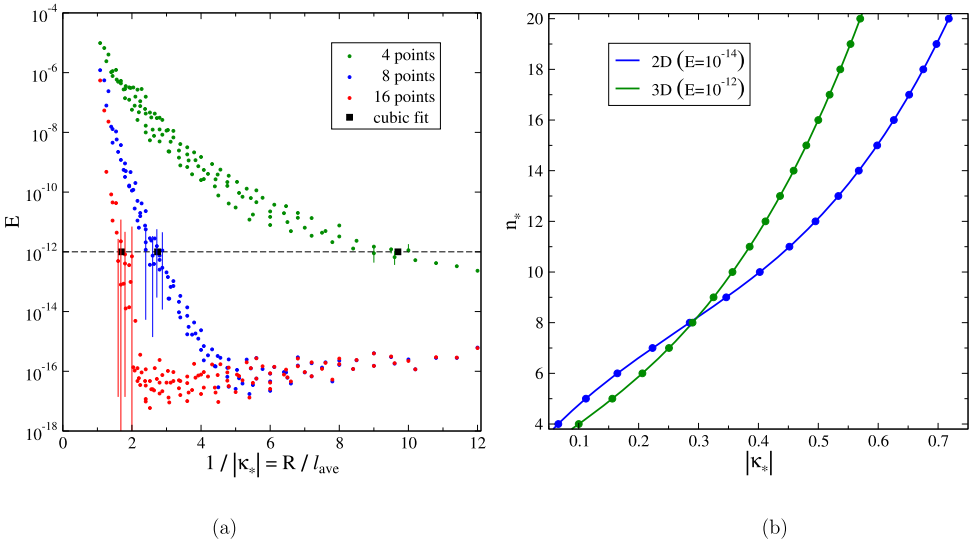

Fig. 4. (a) Error E as a function of the nondimensional curvature $|\kappa_*|$ for a fixed number of integration nodes, $n_2 = n_3 = 4, 8, 16$. Each point corresponds to an average over 10 different spheres with the same radius R. For a few points near the corresponding value of the cubic fit (black squares) the minimum and maximum errors are shown as vertical bars. (b) Cubic relations between the tentative number of integration nodes n_* and the nondimensional numerical curvature $|\kappa_*|$. (For interpretation of the colors in the figures, the reader is referred to the web version of this article.)

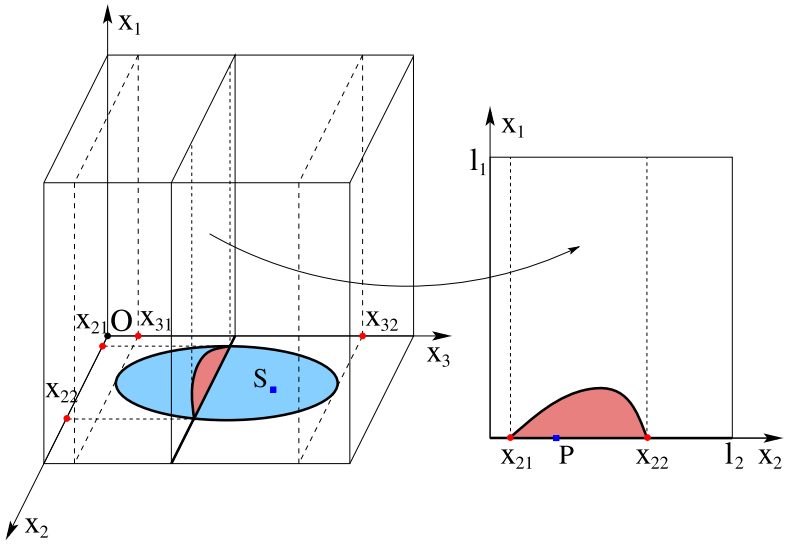

Fig. 5. The grid cell is subdivided into three cuboids after the computation of the two external limits of integration x_{31} and x_{32}, starting from point S (left); each internal integration requires the computation of two internal limits of integration x_{21} and x_{22}, starting from point P (right).

release of the library the routine to compute the external limit x_{32} was not called in this case, and the tiny volume between the two limits x_{31} and x_{32} was not computed. A possible solution to this issue is to call the routine starting from any internal point S_1 and converging to the external limit x_{32}. If we do not discriminate among cells, we should also call the routine from a point S_2 in the bottom-right cell that converges to x_{31}, an integration limit which is also found as an intersection of the interface with a cell edge. Since an external limit in the interior of a face is a rather rare situation, we would waste a considerable amount of CPU time. To avoid this, from the external limits on the edges, such as x_{31} in Fig. 6b, we now compute the function value at two points along the second direction x_2, the first one very close to x_{31} and the other on the opposite edge. In the top-right cell the function has a different sign at these two points, hence the routine to compute the external limits will be called, while in the bottom-right cell the sign is the same and there will be no further action.

2.3. Numerical integration

The root-finding algorithm implemented in the previous release was a hybrid secant-bisection method. We now consider three consecutive iterations (y_0, f_0), (y_1, f_1) (y_2, f_2) and compute the next one in the following way

$$y_3 = y_2 - \frac{f(y_2)}{n'_k(y_2)} \tag{9}$$

where $n_k(y)$ is Newton's divided-differences polynomial of order k and $n'_k(y)$ its derivative. For $k = 1$ we recover the secant method with $n'_1(y_2) = f_{21} = (f_2 - f_1)/(y_2 - y_1)$. For $k = 2$ we define $f_{10} = (f_1 - f_0)/(y_1 - y_0)$ and $f_{210} = (f_{21} - f_{10})/(y_2 - y_0)$, then $n'_2(y_2) = f_{21} + f_{210}(y_2 - y_1)$. The method requires only one function evaluation at each iteration, and as $k \to \infty$ the order of convergence p approaches 2 from below [6]. With $k = 2$ the order of convergence increases from the value $p = 1.6180$ of the

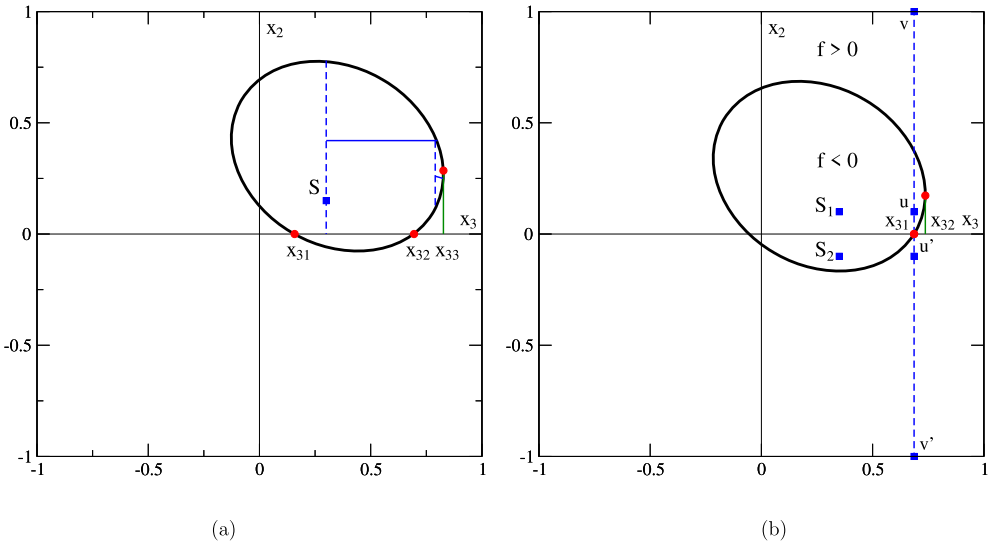

Fig. 6. Intersection of the ellipsoid surface with the plane $z = 0$. (a) Ellipsoid center at $(0.35, 0.35, -5.97)$: in the top-right cell the external limit x_{33} is computed iteratively from point S. (b) Ellipsoid center at $(0.26, 0.26, -5.97)$: in the top-right cell the function sign is evaluated at points u and v to start the iterative computation of x_{32} from S_1, while in the bottom-right cell the function sign at u' and v' is the same with no further action.

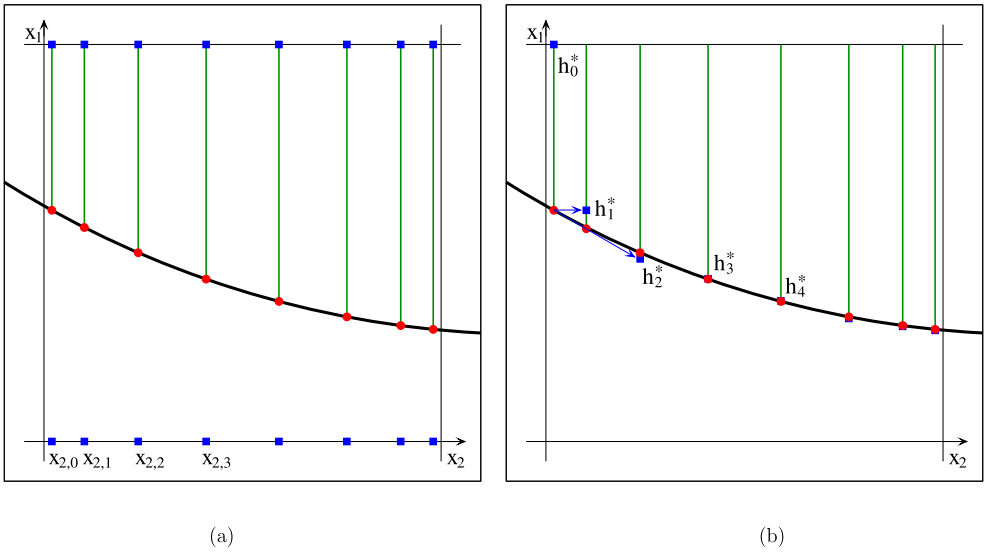

Fig. 7. Integration with 8 nodes: a) in the previous hybrid secant-bisection method the algorithm is initialized with the function value at two points (blue squares), on opposite edges, that bracket the zero (red circles), b) in the new generalized secant method the starting value h_i^* (blue square) is extrapolated from previous height values.

secant method to $p = 1.8393$, both values rounded to four significant digits. The bisection portion of the root-finding algorithm has not been changed. At the beginning of the algorithm or after a bisection step a secant iteration is performed.

In Fig. 7 we illustrate an integration with 8 nodes in 2D. In the previous version each height h_i was computed independently from the neighboring ones. The function value was evaluated at two points on opposite edges (blue squares of Fig. 7a) to initialize the root-finding routine and compute the local height h_i (red circles). In the new release we consider three consecutive couples $(x_{2,i}, h_i)$, $i = 0, 1, 2$, and again Newton's divided-differences polynomial of order 2

$$h^*(x) = h_0 + h_{10}(x - x_{2,0}) + h_{210}(x - x_{2,0})(x - x_{2,1}) \tag{10}$$

where h_{10} and h_{210} are defined as f_{10} and f_{210}, respectively. The starting point for the root-finding routine is the extrapolation h_3^* of the polynomial (10) at $x = x_{2,3}$. In Fig. 7b the blue squares represent the starting points h_i^*: for the first height at $x_{2,0}$ we use the old method, at point $x_{2,1}$ we use $h_1^* = h_0$, at $x_{2,2}$ we use a linear extrapolation and from the fourth point $x_{2,3}$ the extrapolation is quadratic, thereafter the three points for the extrapolation (10) are shifted by one position. To start the iteration (9) we also need to approximate the derivative, that we extrapolate from three consecutive couples $\left(x_{2,i}, n_2'(x_{2,i})\right)$ and the associated Newton's polynomial. With the polynomial extrapolation and the generalized secant method we usually save at least a couple of iterations per height calculation.

2.4. New features: cut volume centroid and interface area

In the first release of the library the grid cells were cubes of edge l_0, now they can be cuboids of edges that may vary from cell to cell. Once the external limits of integration are computed, the cell is further subdivided into subcells and a double Gauss–Legendre integration is performed when the subcell is cut by the interface to compute the cut volume and its centroid. Let us consider a cell of edges l_1, l_2 and l_3 along the three ordered co-

ordinate directions x_1, x_2 and x_3. Furthermore, we assume that the cell has been divided in three subcells: the first is full and fills up the region $0 \le x_3 \le x_{31}$, the second is a cut subcell with $x_{31} \le x_3 \le x_{32}$, the third is empty with $x_{32} \le x_3 \le l_3$. The volume of the first subcell is $V_{01} = l_1\, l_2\, x_{31}$ and its centroid is positioned at $\mathbf{x}_{c1} = \big(l_1/2,\, l_2/2,\, x_{31}/2\big)$. The empty subcell gives no contribution. The cut volume V_{02} and the centroid $\mathbf{x}_{c2}$ of the central subcell are given by the following expressions

$$V_{02} = \int_{V_2} \chi\,(\mathbf{x}, t_0)\, dV\ ; \quad \mathbf{x}_{c2} = \frac{1}{V_{02}} \int_{V_2} \mathbf{x}\, \chi\,(\mathbf{x}, t_0)\, dV \tag{11}$$

where V_2 is the domain of integration of the subcell. To illustrate how the numerical integrations are done we consider the central subcell of Fig. 5 and let $\Delta x_3 = (x_{32} - x_{31})$. The internal limits x_{22} and x_{21} and the length of the interval $\Delta x_2 = (x_{22} - x_{21})$ vary with the third coordinate x_3, even when not explicitly stated in the following expressions. The cut volume V_{02} is then approximated by

$$V_{02} = \int_{V_2} \chi\,(\mathbf{x}, t_0)\, dV = \int_{x_{31}}^{x_{32}} A(x_3)\, dx_3 \approx \frac{\Delta x_3}{2} \sum_{k=1}^{n_3} \omega_k\, A_k\,, \tag{12}$$

where the coefficients ω_k are the integration weights, and the areas A_k are the integrals of the local height function $h(x_2, x_{3,k})$ along the second direction x_2

$$A_k = \int_{x_{21}}^{x_{22}} dx_2 \int_0^h dx_1 = \int_{x_{21}}^{x_{22}} h(x_2, x_{3,k})\, dx_2 \approx \frac{\Delta x_{2,k}}{2} \sum_{j=1}^{n_2} \omega_j\, h_{j,k}\,. \tag{13}$$

The number of nodes n_2 of the internal integration is the minimum value between the previously–defined parameter n_* and that of a linear function of the nondimensional length $\delta_2 = \Delta x_{2,k}/l_n$, with $l_n = \max(l_1, l_2, l_3)$, while n_3 of the external integration is given by a linear function of the length $\delta_3 = \Delta x_3/l_n$. Furthermore, the user can also set a minimum and a maximum value for both n_2 and n_3, which should be in the range between 3 and 20. The user-defined values are compared with and can supersede those determined by the library routines.

The coordinates of the centroid $\mathbf{x}_{c2}$ are then given by the following expressions

$$\begin{aligned}
x_{c2;1} &= \frac{1}{V_{02}} \int_{x_{31}}^{x_{32}} dx_3 \int_{x_{21}}^{x_{22}} dx_2 \int_0^h x_1\, dx_1 \\
&= \frac{\Delta x_3}{2\, V_{02}} \sum_{k=1}^{n_e} \omega_k \left(\frac{\Delta x_{2,k}}{2} \sum_{j=1}^{n_i} \frac{\omega_j}{2}\, h_{j,k}^2 \right) \\
x_{c2;2} &= \frac{1}{V_{02}} \int_{x_{31}}^{x_{32}} dx_3 \int_{x_{21}}^{x_{22}} x_2\, dx_2 \int_0^h dx_1 \\
&= \frac{\Delta x_3}{2\, V_{02}} \sum_{k=1}^{n_e} \omega_k \left(\frac{\Delta x_{2,k}}{2} \sum_{j=1}^{n_i} \omega_j\, x_{2,j,k}\, h_{j,k} \right) \\
x_{c2;3} &= \frac{1}{V_{02}} \int_{x_{31}}^{x_{32}} x_3\, dx_3 \int_{x_{21}}^{x_{22}} dx_2 \int_0^h dx_1 = \frac{\Delta x_3}{2\, V_{02}} \sum_{k=1}^{n_e} \omega_k\, x_{3,k}\, A_k
\end{aligned}$$

Finally, the centroid $\mathbf{x}_c$ of the cut cell is

$$\mathbf{x}_c = \frac{V_{01}\, \mathbf{x}_{c1} + V_{02}\, \mathbf{x}_{c2}}{V_{01} + V_{02}} \tag{14}$$

In 2D an n-point Gauss-Legendre quadrature rule uses n nodes in the interior of the interval of integration $[x_{2a}, x_{2b}]$. In order to calculate the length of the interface line which is inside a two-dimensional cell, we first compute the height at the two endpoints x_{2a} and x_{2b}. In this way we divide the interval $[x_{2a}, x_{2b}]$ in $n+1$ subintervals. We have implemented a node-based method that approximates the interface arc in each subinterval with a polynomial and then computes its length by quadrature [7]. The method is fourth-order accurate and it requires the computation of the height $h_{j+1/2}$ at the midpoint $x_{2,j+1/2}$ between two consecutive nodes $x_{2,j}$ and $x_{2,j+1}$. The Newton's polynomial approach of Section 2.3 is used to initialize the root-finding routine to compute $h_{j+1/2}$. The method considers three points in each subinterval, $\mathbf{x}_{2,j} = \big(x_{2,j}, h_j\big)$, $\mathbf{x}_{2,j+1/2} = \big(x_{2,j+1/2}, h_{j+1/2}\big)$ and $\mathbf{x}_{2,j+1} = \big(x_{2,j+1}, h_{j+1}\big)$, and the arclength L of the subinterval is approximated by the expression

$$L\big(x_{2,j}, x_{2,j+1}\big) \approx |\mathbf{x}_{2,j+1} - \mathbf{r}| + |\mathbf{r} - \mathbf{x}_{2,j}|\,, \tag{15}$$

where the shifted intermediate point $\mathbf{r}$ is

$$\mathbf{r} = \tfrac{1}{2}\big(\mathbf{x}_{2,j} + \mathbf{x}_{2,j+1}\big) + \tfrac{1}{3}\sqrt{3}\big(-\mathbf{x}_{2,j} + 2\,\mathbf{x}_{2,j+1/2} - \mathbf{x}_{2,j+1}\big)\,. \tag{16}$$

To illustrate the surface triangulation in three-dimensions we consider again the ellipsoidal cap of Section 2.2 but with center at $(0.5, 0.45, -5.97)$, then the cap is inside a single grid cell and the coordinates ordering is $x_1 = z$, $x_2 = y$ and $x_3 = x$. We compute the cap volume with $n_2 = n_3 = 6$ nodes for both the internal and external integrations and show the position of all the nodes in Fig. 8a (blue squares). From this nodes distribution and considering the internal integration at $x_{3,k}$, we compute the height at the two endpoints $(x_{21}, x_{3,k})$ and $(x_{22}, x_{3,k})$, and substitute them into the first and last elements of the array containing nodes position and height at $x_{3,k}$. We do the same for the external integration, we remove the first and last arrays of internal nodes and substitute them with the nodes computed at x_{31} and x_{32}. The new nodes are shown as red circles in Fig. 8b, in particular in the external limits at x_{31} and x_{32} only a single node is present. We have decided upon this choice for a more homogeneous distribution of the triangles position and size. Indeed, by simply adding the new nodes, the triangles tend to accumulate near the boundaries and become much smaller.

We then consider two consecutive arrays of nodes, say at $x_{3,k}$ and $x_{3,k+1}$. If the number of internal nodes n_2 is the same we take couples of consecutive nodes on both arrays, and define intermediate nodes by computing their average position, shown as green squares in Fig. 8b, and their average height to start the iteration (9) to calculate the local height. The two couples of consecutive nodes on both arrays and their intermediate node are then connected to form four triangles, that share a vertex at the intermediate node, as shown in Fig. 8b. If the number of internal nodes is not the same, a few triangles will be defined with two consecutive nodes on the array with more nodes and one on the other array, until the number of remaining nodes is the same. This is what is done with the single node in the external limits at x_{31} and x_{32}. The number of external nodes n_3 can change from cell to cell and nodes may not match across the cell boundary, therefore the triangulation is continuous within a cell but it may have holes at the boundary between cells where the edges of the triangles can connect different nodes. The measure of the interfacial area is approximated by the sum of the triangles area.

3. User manual

In the previous release the implicit function $f(\mathbf{x})$ was defined by the user, possibly with a set of fixed parameters, for example the center coordinates and the radius of a sphere. These parameters now can be passed also dynamically as an additional

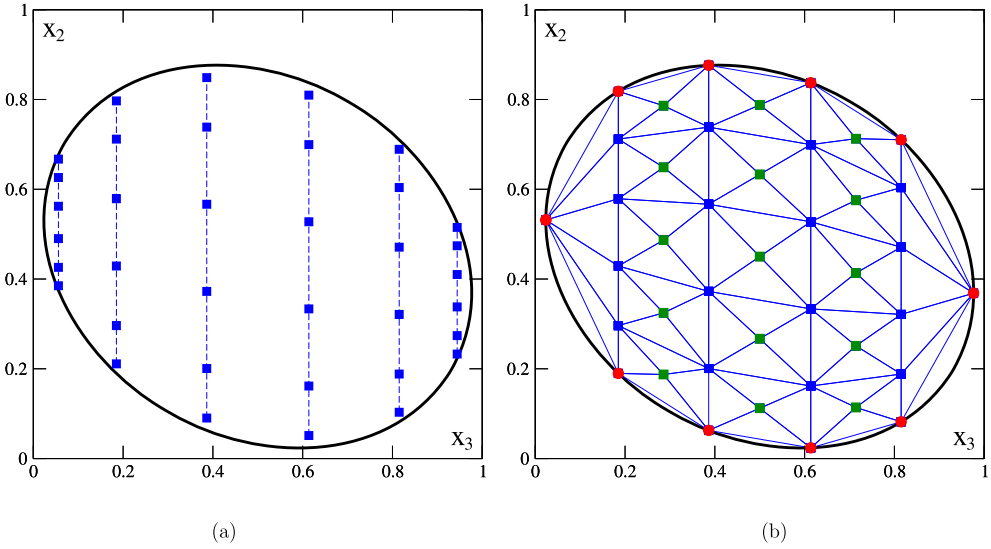

Fig. 8. Triangulation of an ellipsoidal cap: (a) nodes position for the volume integration with $n_2 = n_3 = 6$ (blue squares). The nodes of an internal integration form an array of nodes (dashed lines); (b) first and last elements of internal arrays and first and last arrays are substituted with boundary nodes (red circles). Two couples of consecutive nodes on consecutive arrays are used to define intermediate nodes (green squares) and to form four triangles.

argument of the function call. Furthermore, we provide the possibility to print heights and triangulation data in `TECPLOT ASCII` file format that can be easily visualized with the help of graphics software, as shown in Fig. 10 with the ParaView visualization application.

3.1. C/C++ calls

The Vofi library has been developed in `C` and the user can call two functions. The first function `vofi_get_cc` computes the volume fraction in a given cell and returns a real number (in double precision)

```
cc = vofi_get_cc(impl_func, par, x0, h0,
xext, next, npts, nvis, ndim);
```

The interface position is often used in adaptive mesh refinement as a criterion to increase the local resolution. In this case it is required only to know if the cell is full, empty or cut, and not the actual value of the volume fraction. The second function `vofi_get_cell_type` returns an integer with one of the three values 1, 0 and -1 if the cell is respectively full, empty or cut by the interface

```
icc = vofi_get_cell_type(impl_func, par, x0,
h0, ndim);
```

The input arguments are defined as follows

- `impl_func`: the external function that computes $f(\mathbf{x})$ and is expected to be declared as
  `double impl_func(const double x0[], void * const par) {...}`
- `par`: parameters to be passed to `impl_func` (pointer to a data structure defined by the user)
- `x0`: coordinates of cell vertex with smallest values (1D array of 3 real numbers)
- `h0`: cell edges (1D array of 3 real numbers)
- `xext`: center of mass coordinates and interface length/area (1D array of 4 real numbers, the length/area is always the fourth element)
- `next`: switches either to compute or not to compute the center of mass and/or the interface length/area (1D array of 2 integer numbers, with each value respectively 1 or 0)
- `npts`: minimum and maximum number of nodes allowed by the user for both the internal and external integrations (1D array of 4 integer numbers, to be effective each integer n should be in the range $3 \leq n \leq 20$)
- `nvis`: switches either to print or not to print heights and triangulation data (1D array of 2 integer numbers, with each value respectively 1 or 0). This feature should be used sparingly, as the size of the output files grows very rapidly with grid resolution and number of integration nodes.
- `ndim`: space dimension (integer number, either 2 or 3)

The prototypes for the external implicit function and the two library functions are

```
typedef double (*integrand) (const double [],
void * const);
double vofi_get_cc(integrand,void * const,
const double [],const double [],double [], const
int[],const int[],const int [],const int);
int vofi_get_cell_type(integrand,void *
const,const double [], const double [],const
int);
```

The prototypes are contained in the header file `vofi.h` that should be included when using the Vofi library. To link the library it is necessary to add `-lvofi` to the compiler command line. More details are given in the `README` file of the software distribution.

3.2. Fortran calls

The Vofi library has been developed in `C` and for its usage in `Fortran` codes we have written the interface module `fvofi.f90` that calls the corresponding `C` functions. The examples in `Fortran` use the `ISO_C_BINDING` module that defines intrinsic procedures for `C` interoperability. The user should read the `README` file and look at the examples for different ways to pass the parameters to the external function that computes $f(\mathbf{x})$.

4. Testing and validation

To illustrate the new features of the library we consider the circle with radius $r = 0.25$ and center of coordinates $(x_C, y_C) = (0.623, 0.377)$, inside a unit square domain subdivided into N^2 square cells. As we double the edge resolution starting from $N =$

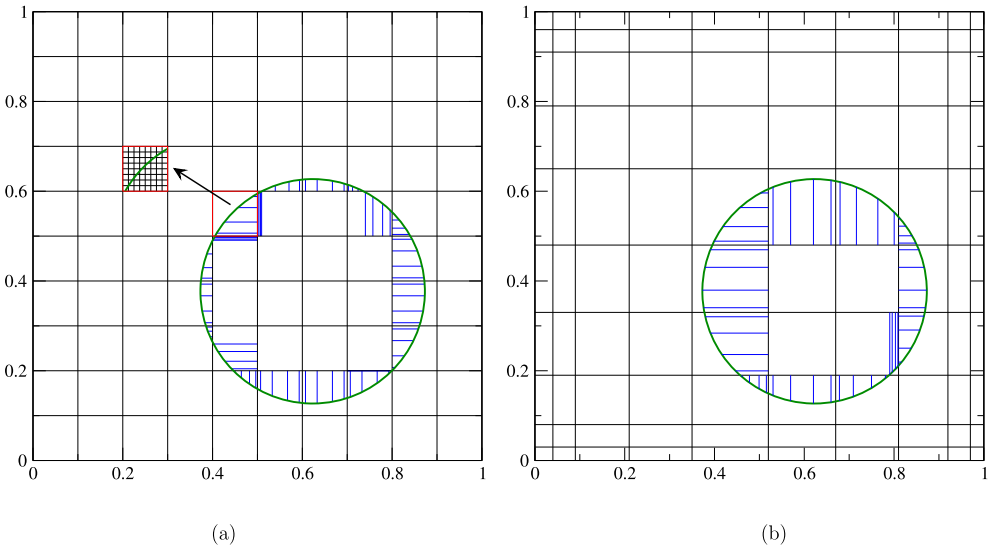

Fig. 9. Integration of the circle with center at $(0.623, 0.377)$ and radius $r = 0.25$ with $n_2 = 4$ nodes and $n_s = 5$ sectors in each cell and edge resolution $N = 10$: (a) square cells, a cut cell at the highest resolution $N = 80$ is also shown; (b) rectangular cells.

Table 1
Number of subdivisions N of the unit edge and number of cut cells m_I for the circle of Fig. 9; errors of the numerical computation of the circle area, E_A, centroid coordinates, E_x and E_y, and circumference length, E_L, with a fixed number of integration nodes, $n_2 = 4$, and of arc sectors, $n_s = 5$, per cell; convergence rate p of the circumference length with edge resolution N.

N	m_I	E_A	E_x	E_y	E_L (5)	p
5	12	4.17e-07	1.12e-07	1.10e-07	2.08e-06	–
10	20	4.68e-08	2.47e-09	5.52e-08	3.58e-07	3.446
20	40	1.16e-10	1.02e-11	1.02e-11	2.26e-08	3.983
40	80	5.04e-13	9.36e-14	9.36e-14	1.36e-09	4.059
80	160	1.03e-15	1.55e-15	2.22e-16	6.59e-11	4.364

Table 2
Number of subdivisions N of the unit edge and number of cut cells m_I for the circle of Fig. 9; errors of the numerical computation of the circumference length, E_L, with a fixed edge resolution N and different arc sectors, $n_s = 5, 10, 20$, per cell; convergence rate p of the circumference length with number of sectors n_s.

N	m_I	E_L (5)	E_L (10)	E_L (20)	p (5 → 10)	p (10 → 20)
5	12	2.08e-06	1.16e-07	6.74e-09	4.159	4.110
10	20	3.58e-07	2.00e-08	1.16e-09	4.160	4.112
20	40	2.26e-18	1.26e-09	7.28e-11	4.163	4.116
40	80	1.36e-09	7.56e-11	4.36e-12	4.165	4.117
80	160	6.59e-11	3.67e-12	2.12e-13	4.165	4.110

10 and up to $N = 80$, the number of cut cells m_I doubles as well. The number of integration nodes n_2 is the same in all the cells and we compare the numerical area, centroid coordinates and circumference length with their analytical values. The results are presented in Tables 1 and 2 where each error is defined as the difference in absolute value between the numerical and analytical values. The interface arclength changes from cell to cell and Gauss nodes are not evenly spaced inside each cut cell, as seen in Fig. 9a. Therefore, we define the convergence rate p of the length of the circumference in the following way

$$p \approx \frac{\log\left(E_{L,1}/E_{L,2}\right)}{\log\left(k_2/k_1\right)} \tag{17}$$

In Table 1 the error $E_{L,1}$ is computed with a fixed number $n_2 = 4$ of integration nodes, corresponding to $n_s = 5$ sectors of the interface arc in each cell, and edge resolution N, and $E_{L,2}$ with the same number of sectors n_s, but edge resolution $2N$. The integer k in the denominator of (17) is then the number of cut cells, $k = m_I$.

Table 3
Number of internal and external nodes, n_2 and n_3, for the sphere of Fig. 10 with edge resolution $N = 10$; errors of the numerical computation of the sphere volume, E_V, centroid coordinates, E_x, E_y and E_z, and interface surface, E_S.

n_2, n_3	E_V	E_x	E_y	E_z	E_S
4	4.25e-09	6.49e-10	2.16e-10	1.81e-09	1.65e-03
8	2.23e-14	5.33e-15	8.33e-16	1.47e-14	6.36e-04
16	0.00e-00	6.66e-16	2.22e-16	5.55e-17	2.96e-04

In Table 2 the edge resolution N is kept fixed as we change the number of sectors, $n_s = 5, 10, 20$, in each cell, hence $k = n_s$ (in the usual definition of the convergence rate a step size appears in the denominator, which is inversely proportional to the resolution that is used in (17)).

The results of Table 1 suggest that the accuracy of the computation of the centroid of a geometric figure is limited from above by the accuracy of the area computation, but the error for one of the centroid components can be one or two orders of magnitude smaller in particular symmetrical conditions. This is what we observe in both 2D and 3D numerical tests. The theoretical convergence order of 4 for the length of the interface line is recovered both by doubling the edge resolution N at constant number of sectors n_s in each cell and by doubling the number of sectors n_s at constant N.

In Fig. 9b we show the heights with $n_2 = 4$ of the same circle in a rectangular grid with $N = 10$. The area error is $E_A = 5.71\,10^{-7}$ which is similar to the error $E_A = 4.17\,10^{-7}$ of Table 1 with $N = 5$, but with the same number of cut cells, $m_I = 12$.

In 3D we consider the sphere with radius $r = 0.34$ and center of coordinates $(x_C, y_C, z_C) = (0.503, 0.451, 0.463)$, inside a unit cube with edge resolution $N = 10$ and cubic cells. In Table 3 we present the results with a different number of integration nodes, but keeping $n_2 = n_3$. At this resolution the convergence rate for the measure of the interfacial surface is linear. However, as we increase the grid resolution N we observe that the measure of the surface tends to saturate as the convergence rate decreases. The triangulation of the interface with $N = 10$ and $n_2 = n_3 = 4$ is shown in Fig. 10, together with the heights and triangles of a selected cell.

Finally we compare the number of function calls that are required to initialize circular and spherical droplets. In 2D the edge resolution of the unit square is N and the total number of grid cells N^2, while the number of cut cells is proportional to N and the total number of heights to the product $N\,n_2$. The ratio of cut cells to

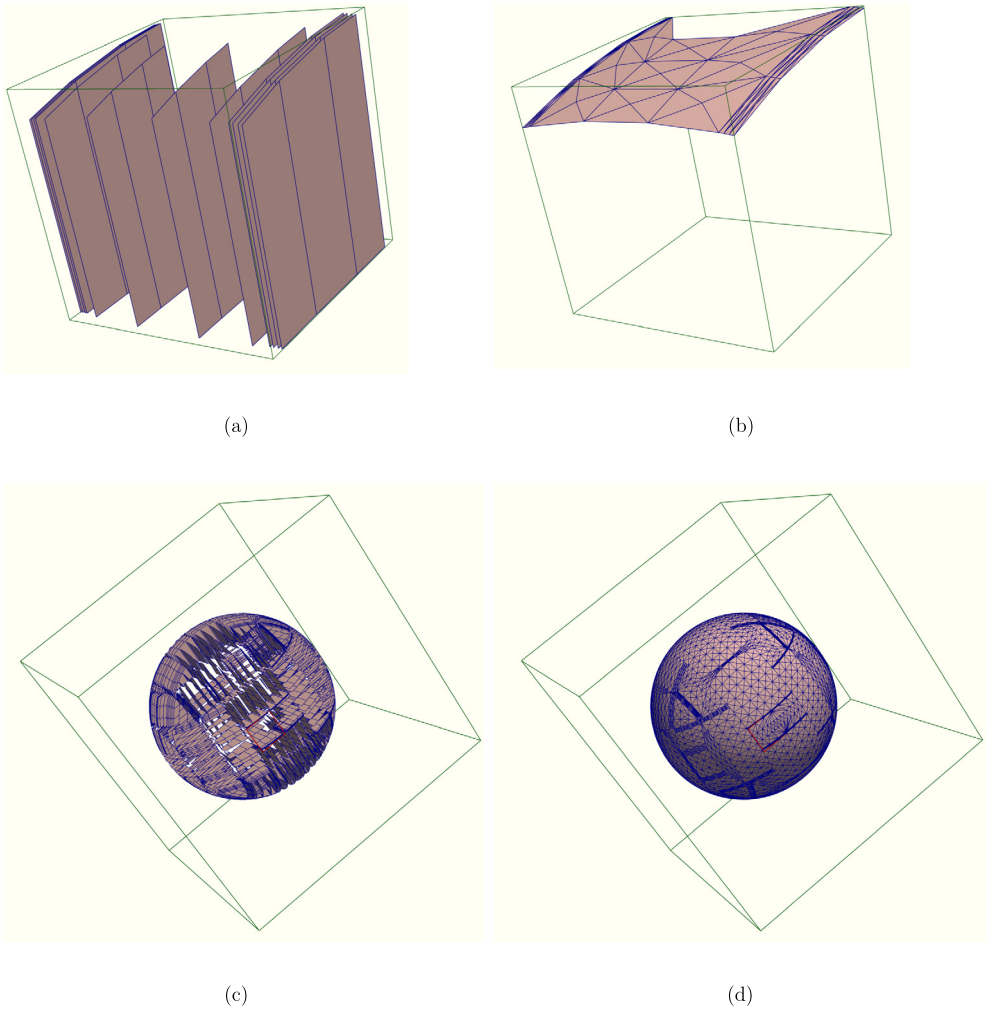

Fig. 10. Integration of the sphere with center at $(0.503, 0.451, 0.463)$ and radius $r = 0.34$ in the unit cube with edge resolution $N = 10$, and $n_2 = n_3 = 4$ in each cut cuboid: (a,b) heights and triangles in a selected cell subdivided in 3 cuboids; (c,d) heights and triangulation of the sphere (top face of the selected cell outlined in red).

Table 4
Ratio of implicit function calls between the old and new versions of the VOFI library in 2D: at constant number of internal nodes $n_2 = 4$ (left), and at constant number of subdivisions $N = 5$ of the unit edge (right).

N	n_2	ratio	N	n_2	ratio
5	4	2.037	5	4	2.037
10	4	2.133	5	8	2.058
20	4	2.134	5	16	2.118
40	4	2.213			
80	4	2.222			

Table 5
Ratio of implicit function calls between the old and new versions of the VOFI library in 3D: at constant number of integration nodes $n_2 = n_3 = 4$ (left), and at constant number of subdivisions $N = 5$ of the unit edge (right).

N	$n_2 = n_3$	ratio	N	$n_2 = n_3$	ratio
5	4	4.607	5	4	4.607
10	4	4.844	5	8	3.325
20	4	4.740	5	16	2.043
40	4	4.400			
80	4	4.038			

the total number of cells scales as $1/N$, therefore the number of empty or full cells becomes more and more predominant with N and the ratio of implicit function calls between the old and new versions should tend to the value $9/4 = 2.25$, where 9 and 4 are the minimum numbers of function calls in the two library versions to determine the cell type. This asymptotic behavior is clearly observed in the results on the left of Table 4. The value of the ratio remains always greater than 2, while the average number of function calls to initialize a cut cell in the new version increases from an average value of 33 when $n_2 = 4$ to 70 when $n_2 = 16$. It is also interesting to note that the ratio increases slightly with the value of n_2 as seen on the right of Table 4. The reason for this behavior is that when $n_2 = 4$ the quadratic extrapolation (10) is applied only once, while with $n_2 = 16$ it is applied to 13 nodes. Therefore, the distance between two consecutive nodes diminishes, the initial guess (10) is more precise, and the average number of iterations to compute the local height is slightly reduced.

In 3D the edge resolution of the unit cube is still N and the total number of grid cells is now N^3, while the number of cut cells is proportional to N^2 and the total number of heights to the product $N^2 n_2 n_3$. The asymptotic value at large N is now $27/8 = 3.375$, and the results on the left of Table 5 show that the numerical ratio is slowly decreasing towards this value. The library optimization is more efficient in 3D than in 2D, as the ratio between the function calls is a bit less than 5 for values of N up to 20 and $n_2 = n_3 = 4$, which implies the computation of 16 local heights. However at fixed N, the numerical ratio decreases with n_2 and n_3. First, we observe that the number of computed heights divided by the number of grid cells is now $n_2 n_3/N$ (it was only n_2/N in 2D)

so that the CPU time spent in their calculation becomes predominant in 3D, as there are 256 local heights when $n_2 = n_3 = 16$. More precisely, with the optimized algorithm to compute the local height, that is described in Section 2.3, we gain at most a factor of order 2, and the results on the right of Table 5 clearly demonstrate this behavior.

To summarize our findings in 3D, at moderate edge resolution N of the unit cube and number of local heights in each cut cell, we observe that the implicit function calls in the new version of the VOFI library are reduced by a factor close to 5 with respect to its first release. If we increase only the resolution N, the initialization of empty or full cells becomes predominant and the factor tends to the asymptotic value 3.375. On the other hand, if we increase only the number of local heights, that factor tends to the value 2.

5. Conclusions

Several numerical algorithms of the VOFI library have been rewritten in order to reduce the number of calls of the user-defined function that defines implicitly the interface. In particular, the grid cell analysis and the numerical integration scheme have been considerably optimized. New features of the software distribution include the subdivision of the computational domain in cuboids and the calculation of the reference phase centroid and the interface length in 2D and area in 3D in each cut cell. The library is written in `C`, and its functions can be called from `Fortran` routines using a supplied interface module and the `ISO_C_BINDING` module.

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

No data was used for the research described in the article.

Acknowledgements

We thank Dr. S. Manservisi for useful discussions on the topics of this paper and in particular on its revision. The simulation data in 3D are visualized by the open-source software PARAVIEW.

References

[1] R. Scardovelli, S. Zaleski, Annu. Rev. Fluid Mech. 31 (1999) 567–603.
[2] G. Tryggvason, R. Scardovelli, S. Zaleski, Direct Numerical Simulations of Gas–Liquid Multiphase Flows, Cambridge University Press, Cambridge, UK, 2011.
[3] A. Bagué, D. Fuster, S. Popinet, R. Scardovelli, S. Zaleski, Phys. Fluids 22 (2010) 092104.
[4] S. Bnà, S. Manservisi, R. Scardovelli, P. Yecko, S. Zaleski, Comput. Phys. Commun. 200 (2016) 291–299.
[5] S. Bnà, S. Manservisi, R. Scardovelli, P. Yecko, S. Zaleski, Comput. Fluids 113 (2015) 42–52.
[6] A. Sidi, Appl. Math. E-Notes 8 (2008) 115–123.
[7] M.S. Floater, A.F. Rasmussen, J. Comput. Appl. Math. 196 (2006) 512–522.