1.

Chapter 1

Arbitrary Lagrangian-Eulerian Method for Two-Phase Flows

G. Anjos*, N. Mangiavacchi† and J. Pontes‡

Group of Environmental Studies for Water Reservoirs – GESAR
State University of Rio de Janeiro, R. Fonseca Teles 524, 20550-013
Rio de Janeiro, RJ, Brazil

Metallurgy and Materials Engineering Dept. PEMM/COPPE/UFRJ,
P.O. Box 68505, 21941-972, Rio de Janeiro, RJ, Brazil

A modern numerical method is described to study two-phase flows for single and multiple bubbles. The fluid flow equations are developed in 3-dimensions based on the Arbitrary Lagrangian-Eulerian formulation (ALE) and the Finite Element Method (FEM), creating a new two-phase method with an improved model for the liquid-gas interface. A successful adaptive mesh update procedure is also described for effective management of the mesh at the two-phase interface to remove, add and repair surface elements, since the computational mesh nodes move according to the flow. The Lagrangian description explicitly defines the two-phase interface position by a set of interconnected nodes which ensures a sharp representation of the boundary, including the role of the surface tension. The methodology proposed for computing the curvature leads to accurate results with moderate programming effort and computational cost and it can also be applied to different configurations with an explicit description of the interface.

1. Introduction

In the literature, fluid flow can be expressed by two reference frames commonly used in fluid dynamics, namely *Eulerian* and *Lagrangian* descriptions. The former describes the fluid motion relative to a fixed referential frame where the continuum moves with respect to the mesh nodes. The latter describes the fluid flow through the material derivative, i.e. the referential frame is moving according to the fluid motion. A more generalized way to describe the fluid motion may consider the referential frame not in fixed space or moving with the same velocity of the fluid motion, but instead moving with an arbitrary velocity that does not necessary represent any of the standard description. Such a generalized representation of the

*currently as Post-Doc at State University of Rio de Janeiro, e-mail: gustavo.anjos@uerj.br
†Professor at State University of Rio de Janeiro, e-mail: norberto@uerj.br
‡Professor at Federal University of Rio de Janeiro, e-mail: jopontes@metalmat.ufr.br

flow field is referred as the *Arbitrary Lagrangian-Eulerian* description or simply ALE.

The conservation laws are used to set the general equations for fluid flow problems. However, the modeling of two-phase flows requires an additional description to characterize the different phases involved. In the "one-fluid" approach, one set of equations is used to describe the entire domain, therefore an additional marker function is used to modify the properties of the phases. At the interface, a jump condition must be taken into account so that the transition zone located at the interface between the fluids can be modeled. In this way, the mass, momentum and energy conservation laws will be developed by considering only one set of equations, where the different phases are represented by a scalar function which defines different properties, such as viscosity and density. Note that in this formulation, the fluid properties are considered to be constant in each phase with a jump condition at the interface.

This chapter describes the general non-dimensional equations employed to model two-phase flow. The equations are written in the so-called "one-fluid" formulation using the Arbitrary Lagrangian-Eulerian description in which the computational mesh and the interface between the fluids move with an arbitrary velocity. Due to the constant motion of the mesh nodes and thus distortion of their layouts, continuous geometric operations are required to keep the finite element mesh satisfactorily distributed, which includes insertion and deletion of nodes, as well as contraction and flipping of element edges. Due to the explicitly description of the interface between the fluids by interconnected nodes, the motion of the interface also requires surface geometric operations, such as those found in the mesh nodes (see [**?**] for detailed informations). Finally, this chapter will present the discretization of the surface tension force and the detailed computation of the curvature term.

2. The arbitrary Lagrangian-Eulerian description

A schematic representation of the one-dimensional domain with the *Lagrangian*, *Eulerian* and *ALE* descriptions is shown in Fig. **??**. In Fig. **??**(a), the nodes are moved with the same velocity as the flow field, thus the material nodes are found to be located on the mesh nodes. As can be seen, depending on the flow conditions, in a short period of time the nodes may be poorly distributed, consequently degrading the accuracy of the solution. In Fig. **??**(b), the mesh nodes are fixed in the space domain and the particle motion is evaluated according to both their position and velocity and then interpolated back to the mesh nodes. The adverse condition here is that the quantity in the next time step is interpolated, and thus numerical diffusion may spoil the accuracy of the simulation. Figure **??**(c) shows an example of the ALE motion, where the material nodes are calculated based on the arbitrary motion of the mesh nodes. Such a inherent capability of the ALE description allows the mesh nodes to be repositioned according to some refinement criteria, thus avoiding

the shortcomings of the pure *Eulerian* and *Lagrangian* descriptions.

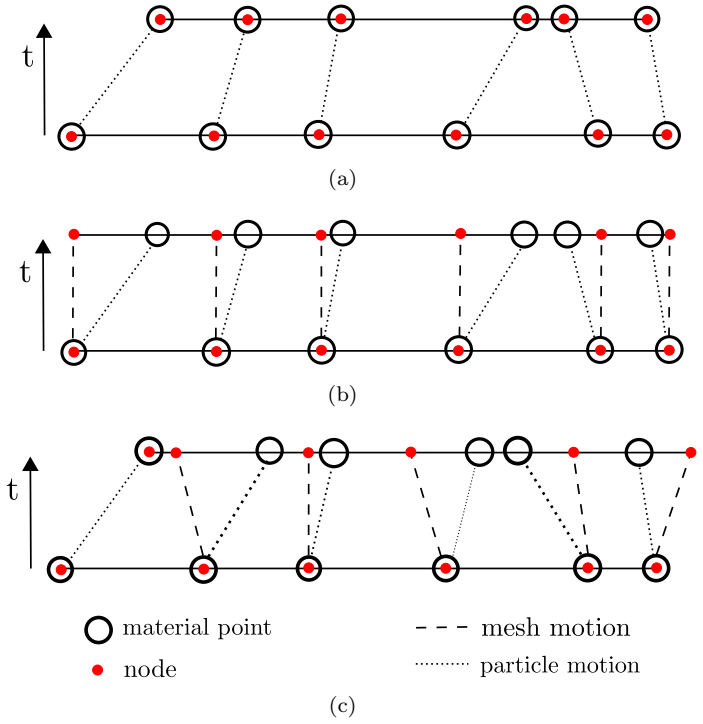


Fig. 1. One-dimensional examples of the (a) *Lagrangian* description, in which the mesh nodes move according to the flow field, (b) *Eulerian*, in which the mesh is fixed in the space and (c) ALE, in which a generalized description is achieved [**?**].

According to [**?**], the ALE description can be represented by three different domains as illustrated in Fig. **??**. These domains are divided as follows: material domain ($\mathbf{X}$), spatial domain ($\mathbf{x}$) and referential domain ($\tilde{\mathbf{X}}$). Both the material and referential domains are moving while the spatial domain is fixed. The velocity of a particle that travels in the material domain relative to the spatial domain may be written according to the operator ϕ, which describes the motion as:

$$\mathbf{v} = \frac{\partial \mathbf{x}}{\partial t} = \frac{\partial}{\partial t} X(\mathbf{x}, t) \tag{1}$$

On the other hand, the velocity of a particle that travels in the referential domain relative to the spatial domain according to the operator $\tilde{\phi}$ may be written as:

$$\hat{\mathbf{v}} = \frac{\partial \mathbf{x}}{\partial t} = \frac{\partial}{\partial t} \tilde{X}(\mathbf{x}, t) \tag{2}$$

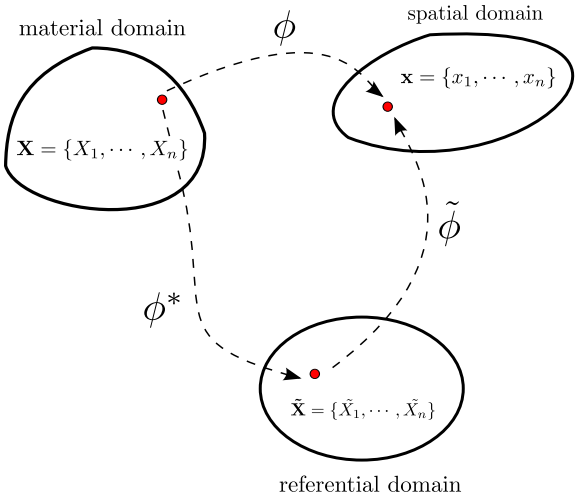


Fig. 2. Material, referential and spatial configuration for the Arbitrary Lagrangian-Eulerian framework.

Consequently, the two velocities $\mathbf{v}$ and $\hat{\mathbf{v}}$ are mapped onto the spatial domain and thus can be rewritten to described the Arbitrary Lagrangian-Eulerian motion as a subtraction of these velocities. So now, let us consider f to be a quantity in the space-time domain expressed as a function of these two mentioned descriptions. The resulting scheme for the quantity f is given by:

$$\frac{Df}{Dt} = \frac{\partial f}{\partial t} + ((\mathbf{v} - \hat{\mathbf{v}}) \cdot \nabla) f = \frac{\partial f}{\partial t} + (\mathbf{c} \cdot \nabla) f \tag{3}$$

In this scheme, if the mesh velocity is $\hat{\mathbf{v}} = \mathbf{v}$, the time variation of the quantity f is exactly the same in two different time steps, and thus the *Lagrangian* description is recovered. By setting the mesh velocity $\hat{\mathbf{v}} = 0$ instead, the referential domain is fixed in the space, while the quantity f varies, and thus in this case the pure *Eulerian* description is achieved.

3. Finite element method for two-phase flows

In this section, the Finite Element Method formulation will be described. Focus will be on the variational formulation (weak form) of the non-dimensional conservation equations for two-phase flows. Additionally, the discrete finite elements used in this work will be presented and a brief discussion given of the tetrahedron mesh generation method through the *Delaunay* algorithm.

3.1. *Variational formulation*

Let us consider the fluid to be incompressible and the momentum, continuity and energy (heat) equations are given in their non-dimensional form as follows:

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{c} \cdot \nabla \mathbf{v} = -\frac{1}{\rho}\nabla p + \frac{1}{Re}\nabla \cdot [\mu(\nabla \mathbf{v} + \nabla \mathbf{v}^T)] + \frac{1}{Fr^2}\mathbf{g} + \frac{1}{We}\mathbf{f} \tag{4}$$

$$\nabla \cdot \mathbf{v} = 0 \tag{5}$$

$$\frac{\partial T}{\partial t} + \mathbf{c} \cdot \nabla T = \frac{1}{RePr}\nabla \cdot (k \nabla T) \tag{6}$$

that are valid in a domain $\Omega \subset \mathbb{R}^m$ with the following boundary conditions:

$$\mathbf{v} = \mathbf{v}_\Gamma \quad \text{in } \Gamma_1 \tag{7}$$

$$\mathbf{v}_t = 0 \quad \text{and } \sigma^{nn} = 0 \quad \text{in } \Gamma_2 \tag{8}$$

$$T = T_\Gamma \quad \text{in}\Gamma_3 \tag{9}$$

where the convective velocity $\mathbf{c}$ represents the relative velocity between the flow field and the mesh, given by the following expression: $\mathbf{c} = \mathbf{v} - \hat{\mathbf{v}}$. Here, $\mathbf{v}$ stands for the flow field velocity and $\hat{\mathbf{v}}$ for the mesh velocity.

Now, let us consider the subspace:

$$\mathbb{V} = H^1(\Omega)^m = \{\mathbf{v} = (v_1, \ldots, v_m) : v_i \in H^1(\Omega), \forall\ i = 1, \ldots, m\} \tag{10}$$

where $H^1(\Omega)$ and the *Sobolev* space given by:

$$H^1(\Omega) = \left\{ v \in L^2(\Omega) : \frac{\partial v}{\partial x_i} \in L^2(\Omega), i = 1, \cdots, m \right\} \tag{11}$$

taking $L^2(\Omega)$ as an infinite dimensional space characterized by the Lesbegue's integral, which is equivalent to the Riemann's integral for continuous functions, and thus it can be treated by the conventional form:

$$L^2(\Omega) = \left\{ v : \Omega \to \mathbb{R}, \int_\Omega v^2 d\Omega < \infty \right\} \tag{12}$$

Note that $\mathbb{V} = H^1(\Omega)^m$ is the Cartesian product of m spaces $H^1(\Omega)$ where:

$$\mathbb{V}_{\mathbf{v}_\Gamma} = \{\mathbf{v} \in \mathbb{V} : \mathbf{v} = \mathbf{v}_\Gamma \text{ in } \Gamma_1\} \tag{13}$$

$$\mathbb{P}_{p_\Gamma} = \{q \in L^2(\Omega) : q = p_\Gamma \text{ in } \Gamma_2\} \tag{14}$$

$$\mathbb{T}_{T_\Gamma} = \{r \in L^2(\Omega) : r = T_\Gamma \text{ in } \Gamma_3\} \tag{15}$$

The variational formulation consists in finding the solutions $\mathbf{v}(x,t) \in \mathbb{V}_{\mathbf{v}_\Gamma}$, $p(x,t) \in \mathbb{P}_0$ and $T(x,t) \in \mathbb{T}_{T_\Gamma}$ so that:

$$\int_\Omega \left\{ \frac{\partial \mathbf{v}}{\partial t} + \mathbf{c}\cdot\nabla\mathbf{v} + \frac{1}{\rho}\nabla p - \frac{1}{Re}\nabla\cdot[\mu(\nabla\mathbf{v} + \nabla\mathbf{v}^T)] - \frac{1}{Fr^2}\mathbf{g} - \frac{1}{We}\mathbf{f} \right\}\cdot\mathbf{w}d\Omega = 0 \tag{16}$$

$$\int_\Omega [\nabla\cdot\mathbf{v}]q d\Omega = 0 \tag{17}$$

$$\int_\Omega \left\{ \frac{\partial T}{\partial t} + \mathbf{c}\cdot\nabla T - \frac{1}{RePr}\nabla\cdot(k\nabla T) \right\} r d\Omega = 0 \tag{18}$$

Developing the equation terms, they become:

$$\begin{aligned}\int_\Omega \left\{ \frac{\partial \mathbf{v}}{\partial t} + \mathbf{c}\cdot\nabla\mathbf{v} \right\}\cdot\mathbf{w}d\Omega + \int_\Omega \left\{ \frac{1}{\rho}\nabla p \right\}\cdot\mathbf{w}d\Omega \\ - \int_\Omega \left\{ \frac{1}{Re}\nabla\cdot[\mu(\nabla\mathbf{v} + \nabla\mathbf{v}^T)] \right\}\cdot\mathbf{w}d\Omega \\ - \int_\Omega \left\{ \frac{1}{Fr^2}\mathbf{g} \right\}\cdot\mathbf{w}d\Omega - \int_\Omega \left\{ \frac{1}{We}\mathbf{f} \right\}\cdot\mathbf{w}d\Omega = 0\end{aligned} \tag{19}$$

$$\int_\Omega \{\nabla\cdot\mathbf{v}\}q d\Omega = 0 \tag{20}$$

$$\int_\Omega \left\{ \frac{\partial T}{\partial t} + \mathbf{c}\cdot\nabla T \right\}\cdot r d\Omega - \int_\Omega \left\{ \frac{1}{RePr}\nabla\cdot(k\nabla T) \right\} r d\Omega = 0 \tag{21}$$

The first terms of Eqs. **??** and **??** will be treated in the *ALE* formulation as a substantive derivative. Its weighing ratio consists in:

$$\int_\Omega \frac{D\mathbf{v}}{Dt}\cdot\mathbf{w}d\Omega = \int_\Omega \left\{ \frac{\partial \mathbf{v}}{\partial t} + \mathbf{c}\cdot\nabla\mathbf{v} \right\}\cdot\mathbf{w}d\Omega \tag{22}$$

$$\int_\Omega \frac{DT}{Dt} r d\Omega = \int_\Omega \left\{ \frac{\partial T}{\partial t} + \mathbf{c}\cdot\nabla T \right\}\cdot r d\Omega \tag{23}$$

The next step is the treatment of the diffusive terms, in which Green's theorem should by applied, thus splitting the volume integral in two other integrals: the inner and the boundary domain integrals, so that:

$$\int_\Omega \nabla \cdot [\mu(\nabla \mathbf{v} + \nabla \mathbf{v}^T)] \cdot \mathbf{w} d\Omega = \\ - \int_\Omega \nu[(\nabla \mathbf{v} + \nabla \mathbf{v}^T) : \nabla \mathbf{w}^T] d\Omega + \int_\Gamma \mathbf{n} \cdot [\mu(\nabla \mathbf{v} + \nabla \mathbf{v}^T) \cdot \mathbf{w}] d\Gamma \quad (24)$$

$$\int_\Omega \nabla \cdot (k \nabla T) r d\Omega = - \int_\Omega (k \nabla T) \cdot \nabla r^T d\Omega + \int_\Gamma \mathbf{n} \cdot (k \nabla T) r d\Gamma \quad (25)$$

where the operator (:) stands for the scalar product between two tensors. The boundary integral Γ in the above equation may be segregated into two other integrals in Γ_1 and Γ_2. Due to $w = 0$ from Eq. **??** and $r = 0$ from Eq. **??**, the integral in Γ_2 is null. Moreover, the integral in Γ_1 is also null due to the boundary conditions in Eq. **??**. Therefore the integral in Γ is null. The treatment of the pressure term is done by applying the same procedure, and thus the integration by parts results in:

$$\int_\Omega \nabla p \cdot \mathbf{w} d\Omega = - \int_\Omega p \nabla \cdot \mathbf{w} d\Omega + \int_\Gamma p \mathbf{w} \cdot \mathbf{n} d\Gamma \quad (26)$$

where the above boundary integral is null due to the boundary conditions $\mathbf{w} = 0$ in Γ_1 and $p = 0$ in Γ_2. The gravity and surface tension terms are treated simply as:

$$\int_\Omega \mathbf{g} \cdot \mathbf{w} d\Omega \qquad \text{and} \qquad \int_\Omega \mathbf{f} \cdot \mathbf{w} d\Omega \quad (27)$$

The resulting equations are:

$$\int_\Omega \frac{D\mathbf{v}}{\partial t} \cdot \mathbf{w} d\Omega - \frac{1}{\rho} \int_\Omega p \nabla \cdot \mathbf{w} d\Omega + \frac{1}{Re} \int_\Omega \mu[\nabla \mathbf{v} + \nabla \mathbf{v}^T] : \mathbf{w} d\Omega \\ - \frac{1}{Fr^2} \int_\Omega \mathbf{g} \cdot \mathbf{w} d\Omega - \frac{1}{We} \int_\Omega \mathbf{f} \cdot \mathbf{w} d\Omega = 0 \quad (28)$$

$$\int_\Omega [\nabla \cdot \mathbf{v}] q d\Omega = 0 \quad (29)$$

$$\int_\Omega \frac{DT}{\partial t} r d\Omega + \frac{1}{RePr} \int_\Omega (k \nabla T) \nabla r^T d\Omega = 0 \quad (30)$$

The integral forms are defined as follows:

$$m(\frac{D\mathbf{v}}{Dt}, \mathbf{w}) = \int_\Omega \frac{D\mathbf{v}}{Dt} \cdot \mathbf{w} d\Omega \tag{31}$$

$$k(\nu, \mathbf{v}, \mathbf{w}) = \int_\Omega \nu[(\nabla \mathbf{v} + \nabla \mathbf{v}^T) : \nabla \mathbf{w}^T] d\Omega \tag{32}$$

$$g(p, \mathbf{w}) = \int_\Omega \nabla p \cdot \mathbf{w} \; d\Omega \tag{33}$$

$$d(p, \mathbf{w}) = \int_\Omega (\nabla \cdot \mathbf{w}) p \; d\Omega \tag{34}$$

$$\tilde{k}(k, T, r) = \int_\Omega k \nabla T \cdot \nabla r^T d\Omega \tag{35}$$

$$\tilde{m}(\frac{DT}{Dt}, r) = \int_\Omega \frac{DT}{Dt} r d\Omega \tag{36}$$

Thus, the weak form of the proposed problem is written as: Find the solutions $\mathbf{v}(x,t) \in \mathbb{V}_{\mathbf{v}_\Gamma}$, $p(x,t) \in \mathbb{P}$ and $T(x,t) \in \mathbb{T}_{T_\Gamma}$ such that

$$m(\frac{D\mathbf{v}}{Dt}, \rho, \mathbf{w}) - g(p, \mathbf{w}) + \frac{1}{Re} k(\mu, \mathbf{v}, \mathbf{w}) - \frac{1}{Fr^2} m(\mathbf{g}, \rho, w) - \frac{1}{We} m(\mathbf{f}, w) = 0 \tag{37}$$

$$d(q, \mathbf{v}) = 0 \tag{38}$$

$$\tilde{m}(\frac{DT}{Dt}, r) + \frac{1}{RePr} \; \tilde{k}(k, T, r) = 0 \tag{39}$$

for all $\mathbf{w} \in \mathbb{V}_0$, $q \in \mathbb{P}_0$ and $T \in \mathbb{T}_0$.

3.2. *Mesh elements*

The computational mesh used in the Finite Element Method allows for a wide variety of elements. They are characterized by their geometry and the polynomial interpolating function used to fit the desired equation. These elements may be classified according to their shape as triangular, quadrilateral, etc. for 2-dimensional problems and tetrahedral, prismatic, parallelepiped, etc. for 3-dimensional problems. Moreover, the order of the polynomial function may be classified as linear, quadratic, cubic, bi-linear, etc. Additionally a particular class of elements, namely *Isoparametric elements*, may be used where the region to be modeled requires elements with more general shapes, which is the case of curvilinear domains (for details, see [**?**]). Even though affordable in moving boundary problems, the Isoparametric elements are unfortunately relatively more complicated to deploy and thus they have not been chosen in the present work.

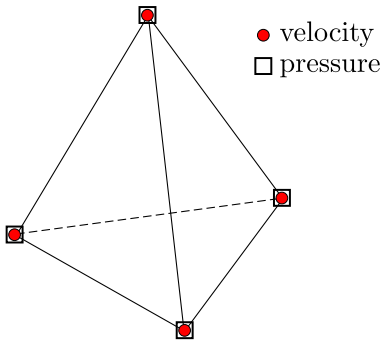
The finite elements are numerically represented by the meaning of their local coordinates. This means that the interpolating functions (also known as shape functions) are built independently of the element's neighborhood, and thus the finite element may be easily changed without modifying significantly the simulation

program. Such an inherent ability is a strong advantage compared to other discretization methods such as finite difference and finite volume methods. In the following subsections, a description of the tetrahedron element with respect to its *volume coordinates* is given and then a brief illustration of different shape functions commonly found in the literature are presented.

3.2.1. *3-dimensional elements*

The choice of the appropriate element is a key step to successfully achieve the required precision in the simulations. In fluid dynamics, the element is responsible for the coupling of velocity and pressure and it should satisfy the requirements of the so-called *Ladyzhenskaya-Babouska-Brezzi* (LBB) stability condition ([**?**], [**?**] and [**?**]). Such a condition imposes the type of velocity and pressure basis functions. One way to avoid the LBB condition is to use stabilizing methods such as pressure stabilization, penalty method or artificial compressibility ([**?**], [**?**],[**?**]). However, such a stabilization process is not part of this work, for which is preferred the use of LBB stable elements. The elements used in the present work are presented below, followed by short descriptions of their features and their applicability in the conservation equations.

Linear element: The unknowns are evaluated at the tetrahedron's corners with interpolation functions of order 1. This element is commonly used to solve scalar equations, such as heat and chemical species transport. This element does not satisfy the LBB condition for fluid flow problems and it cannot be used to solve velocity and pressure without stabilizing methods.

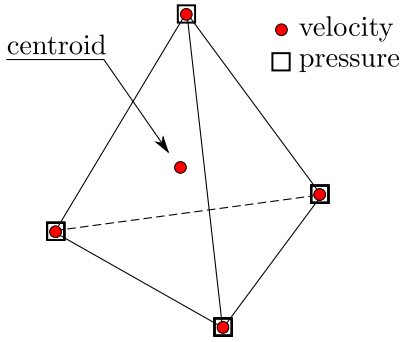


For linear tetrahedron elements with local coordinates L_i, L_j, L_k and L_l, the shape functions are equivalent and written as:

$$N_i = L_i, \qquad i = 1, 2, 3, 4 \tag{40}$$

Mini element: This element is part of the Taylor-Hood family and it is a combination of the linear tetrahedron and an additional "bubble" function, built

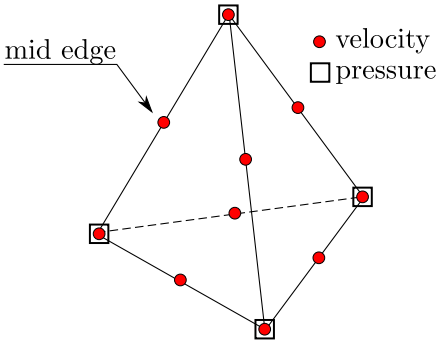
by an additional node localized in its barycenter, thus making it a five node element. The interpolation polynomial is of order 4, but it does not present the second and the third degree terms. The pressure is linear and is evaluated at the tetrahedron vertices and the velocity is cubic incomplete and evaluated at all the 5 nodes.



The shape functions N_i for the 5-node tetrahedron element may be defined as a function of local coordinates L_i, L_j, L_k as follows:

$$\begin{aligned} N_i &= L_i - 64L_1L_2L_3L_4, \qquad i = 1,2,3,4 \\ N_5 &= 256L_1L_2L_3L_4 \end{aligned} \tag{41}$$

10-node element: Defined by nodes in the middle of the tetrahedron edges, the 10-nodes quadratic element is commonly used in fluid flow problems. The interpolation polynomial is of order 2. The pressure is linear and evaluated at the tetrahedron vertices, while the velocity is quadratic and evaluated at all 10 nodes.



The interpolation functions N_i for the 10-node quadratic element may be also expressed as a function of local coordinates L_i, L_j, L_k as follows:

$$
\begin{aligned}
N_i &= (2L_i - 1)L_i, \qquad i = 1, 2, 3, 4 \\
N_5 &= 4L_1L_2 \\
N_6 &= 4L_2L_3 \\
N_7 &= 4L_1L_3 \\
N_8 &= 4L_1L_4 \\
N_9 &= 4L_2L_4 \\
N_{10} &= 4L_3L_4
\end{aligned}
\tag{42}
$$

The above list of available elements is not limited to those above described. Many other geometries and polynomial functions may be used to discretize the fluid flow equations. High order elements, discontinuous pressure elements and combined elements are examples of the wide variety of Finite Elements. This diversity is organized by families such as Taylor-Hood, Crouzeix-Raviart and Serendipity; each one has its particular approach to calculate the quantities. A list of available elements for fluid flow problems can be found in [**?**], [**?**] and [**?**].

Due to its superior mass conservation and the minimum amount of nodes per element to satisfy the LBB condition, the mini-element has been chosen here to discretize pressure and velocity in the conservation equations within the moving mesh context. Such a decision has shown to be appropriate to model two-phase flow problems with minimum implementation efforts and significant accuracy.

3.3. *The Delaunay tetrahedralization*

Tetrahedron elements are extremely powerful to discretize any kind of geometry, extending from simple brick geometries to more complex shapes including curvilinear boundaries. However, for fluid flow problems, the distribution of the elements should respect some physical requirements. While considering a domain discretized by tetrahedron elements and not limiting it to a uniform node distribution, an unstructured grid is expected, i.e. the number of a node's neighbors is not constant. Such a flexibility cannot be assessed if cubic and brick shaped elements are used; however, the Finite Element method allows the concurrent use of different element shapes, if the code implementation takes into account their interconnectivity. Thus, the tetrahedron distribution in the computational domain may change with respect to the investigated problem and some criteria need be imposed to keep the elements bounded to acceptable aspect ratios, since low quality elements may corrupt the accuracy of the computational discretization.

In the present technique, the equations are discretized over an unstructured non-regular tetrahedral mesh with optimal element properties. This is achieved using the *Delaunay* tetrahedralization algorithm which ensures well-shaped elements.

However, due to the constant motion of the mesh nodes, all the Delaunay requirements may not be satisfied. Nevertheless, as will be described bellow, efforts have been made to maintain the mesh quality, in each time step, restricting it to good element aspect ratios.

The Delaunay tetrahedralization is a geometrical construction often used in mesh generation for the Finite Element Method. Such a construction corresponds to the dual graph of the *Voronoi diagram* ([**?**]), which is a special type of space decomposition determined by distances and commonly used for mapping regions and land areas. Figure **??** shows a simplified representation of the Voronoi diagram of a set of nodes and its corresponding tessellation according to the Delaunay properties.

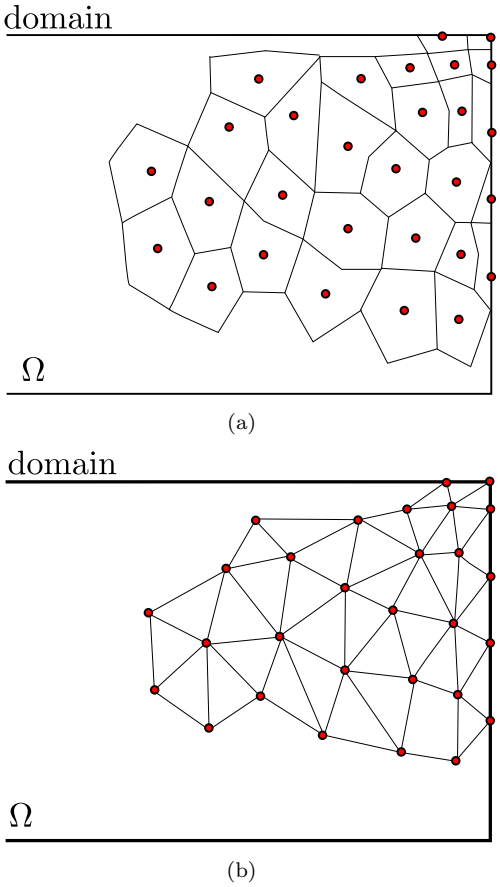


Fig. 3. 2-dimensional representation of a (a) *Voronoi diagram* and (b) its tessellation according to the Delaunay properties.

In 2-dimensional spaces, the Delaunay algorithm maximizes the minimum angle of all triangles and minimizes the largest circumscribed circle, thus avoiding low

quality elements. The Delaunay tetrahedralization follows the same strategy as its correspondent 2-dimensional algorithm, however some of the features may not be assured due to geometrical restrictions. Although this inconvenience, the Delaunay tetrahedralization results in satisfactory elements and thus can be used to discretize the domain of two-phase flow problems by the Finite Element Method (see [**?**] and [**?**]).

4. Adaptive mesh refinement

This section describes the new methodology of adaptive mesh refinement developed to work with the Finite Element method in two-phase flows, in which the interface between the fluids plays an important role. The effective management of the computational mesh is detailed in which two data structures are stored in the computer memory and handled separately.

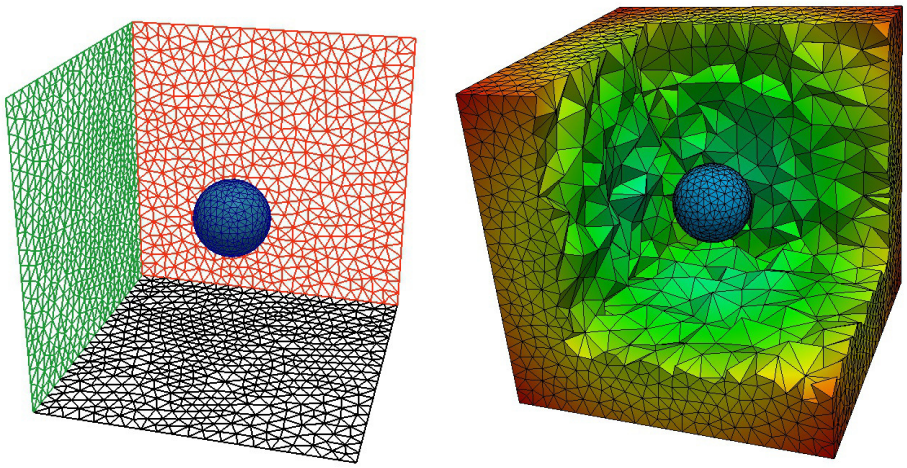
4.1. *Mesh representation*

Two sets of data are stored during the simulation which are treated separately in the framework of the adaptive mesh refinement: the volumetric nodes and the surface mesh. The latter consists of two parts, those on the interface between the phases and those on the domain boundary, both created simultaneously by the software (GMesh, [**?**]). The code is then linked to a tetrahedral mesh generator (TETGEN, [**?**]) that uses the previous generated surface meshes and a volumetric set of nodes as input parameters. The initial volumetric node distribution may be set manually, according to the requirements of the simulation, or by TETGEN, which creates a smooth distribution of nodes according to the edge lengths of the given surface meshes. Then, the 3-dimensional connectivity array is exported to the code. Figure **??** depicts the two data structures used in this work. Such an approach allows for easy maintenance of each region without affecting the node connectivity of the other. Moreover, the data structure makes the mesh management easier in the code programming level, enabling fast access to each structure separately.

In the ALE context the computational mesh is not fixed in the space, but instead it is moved according to an arbitrary velocity. Therefore to avoid collapsing of nodes, edges and elements, the computational mesh requires an extensive topological treatment. Note that due to the separate treatment of the surface mesh and the volumetric nodes, each set of data must be handled in a different way. Additionally, mesh smoothing may be used to reduce the number of operations performed on the meshes.

4.2. *Volumetric nodes*

For each time step, the node distribution of the volumetric mesh (tetrahedrons) is monitored and compared to the previous iteration. The tetrahedron edge length



(a) (b)

Fig. 4. Data-set representation of the meshes used in the present numerical code. (a) The surface meshes, which comprise the interface between the fluids and the domain boundary, are passed as an input parameter to the open source library *TETGEN*, which exports the 3-dimensional connectivity array. (b) The volumetric mesh is then used to discretize the two-phase flow equations.

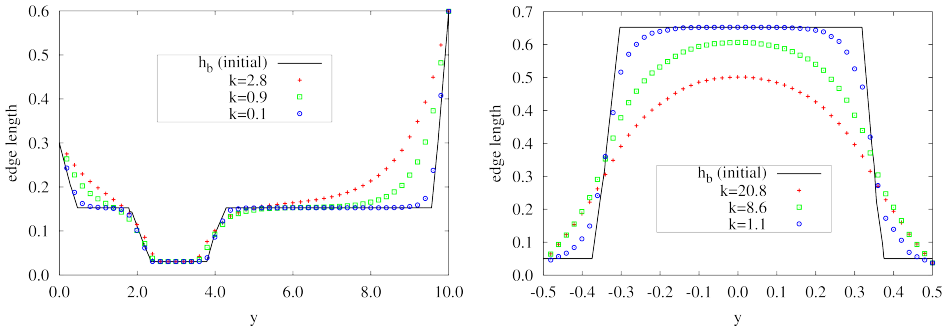
determines whether insertion or deletion is required for a given predefined distance in a specific zone, therewith it is possible to avoid clustering and dispersion of computational nodes. In this work, the edge length distribution h is obtained by the solution of the Helmholtz's equation:

$$\nabla^2 h = \frac{1}{k}(h_b - h) \tag{43}$$

where k is a diffusive parameter and h_b is the initial edge length distribution. Thus, the solution obtained corresponds to a smooth distribution of nodes in the volumetric space. Note that for large values of k in the above equation, the right hand term tends to zero, thus resulting in Laplace's equation ($\nabla^2 h = 0$) in which the solution damps all the sudden changes in the distance between nodes. On the other hand, assuming a small value of k, the solution h approaches the initial node distribution h_b.

As mentioned before, the initial volumetric node distribution h_b may be set according to the flow requirements, where a particular zone may or may not be refined. Figure **??** shows two examples of the solution of the Helmholtz equation. In both cases, the solid lines represent the initial edge length distribution h_b, while the other lines show the solutions for different diffusive parameter k. As can be seen for large values of k, the smoothing effect is more pronounced, while for small values of k, the solution approaches the initial distribution h_b. In Fig. **??**(a), the edge length distribution is seen in the z axis, in which the bubble is located within

the interval $z = \{2, 4\}$. This shows a typical distribution of edge lengths in rising bubble simulations where a dense cluster of nodes is expected near the bubble's interface. However, far from the interface, the region is not as refined as can be seen with large values of edge length h. In Fig. **??**(b) the edge length distribution differs from the previous case. The aim is to achieve an efficient distribution along a pipe or channel, thus more nodes are required close to the domain boundaries and a coarse mesh is sufficient in the middle. Such a refinement is important to investigate liquid film thickness and bubble/boundary interactions present in annular and bubbly flows.



(a) (b)

Fig. 5. Solutions of the Helmholtz's equations for different values of the diffusive parameter k. (a) The sample was taken along the z axis, in which the bubble's location can be seen within the interval $z = \{2, 4\}$. (b) The y component represents the channel cross section. In this case, the mesh is more refined close to the channel's boundaries ($y = -0.5$ and $y = 0.5$) and coarser in the middle.

The solution of the Helmholtz equation (**??**) has been shown to be extremely important in order to achieve a smooth distribution of nodes in the 2-dimensional and 3-dimensional domains. Its continuous solution may overload the computational resources and it may not necessarily bring significant changes in the simulation at each time step. Due to minor modifications in the mesh distribution from one time step to another, mainly driven by small time steps, the solution of the Helmholtz equation may be stored in the memory and kept for a few iterations. After a certain number of iterations, say 5, the procedure is redone. Thus, the successive solution of the Helmholtz equation is avoided but a smooth edge length distribution is achieved during all the simulation.

4.3. *Mesh smoothing*

A proper choice of the mesh velocity and the mesh strategy is necessary to avoid the fast degradation of the computational elements and the onset of non-desirable numerical instabilities found in the pure *Lagrangian* framework. Especially close

to the boundaries, the elements are often stretched and compressed in such a way that the simulation becomes unstable. Therefore it is desirable to chose the mesh velocity to rearrange the nodes, thus keeping the elements bounded within good aspect ratios. Additionally, insertion and deletion of vertices should be performed whenever the edges of the elements are greater or smaller than a predefined size. Unfortunately, the re-meshing process and the rearrangement of mesh elements may become very expensive in terms of processing time and therefore they require special attention. To address this issue, a new fast node repositioning scheme is proposed and described here as well as a new local re-meshing technique that bounds the element aspect ratios within a satisfactory level with no significant time cost in two-phase flows.

The mesh velocity $\hat{\mathbf{v}}$ determines the motion of the nodes of the finite element mesh. This velocity is obtained by a linear combination of two others: the flow velocity itself and an *elastic* velocity, in which the latter is defined according to some smoothing criteria intended to redistribute the nodes optimally, thus minimizing the number of re-meshing steps and avoiding heavy computation requirements. The transfinite mapping (see [**?**]) and Laplacian smoothing methods are examples of mesh-update procedures.

The idea behind the Laplacian smoothing operator is to move the non-uniform mesh nodes by redistributing the distance equally between them, thus achieving a smoothed distribution. In this work, the implementation consists in defining a function in terms of the 1-ring neighboring coordinates; thus the mesh nodes are repositioned in such a manner that the elements satisfy some predetermined geometrical criterion. The smoothing procedure is part of an iterative scheme which converges to a more uniform node distribution. According to [**?**], the new node's position $\hat{\mathbf{x}}_i$ can be approximated using a weighted sum of the 1-ring neighbors of a node as follows:

$$\hat{\mathbf{x}}_i = \sum_{i \in N_1(j)} w_{ij}(\mathbf{x}_j - \mathbf{x}_i) \tag{44}$$

where w_{ij} is the weight that can be set as uniform or proportional to the inverse distance from its neighbor vertices and N_1 is the set of 1-ring neighbors of the $j^{th.}$ node. Thus, the mesh velocity $\hat{\mathbf{v}}_e$ is found by dividing the displacement of each node's position $\mathbf{x}_i$ by the simulation time step dt. This approximation is not sufficient to distribute the mesh nodes optimally in one single application step but once applied systematically, the mesh elements converge to a satisfactory shape. Figure **??** shows a 3-dimensional example of a node reposition scheme based on the Laplacian smooth operator. As can be seen, the Laplacian operator moves the nodes in the direction of the polyhedron's centroid, thus the connected edges approach an uniform spatial distribution.

So far we have seen that the uniform weighted approximation results in significant sliding and shape distortion in an unstructured 3-dimensional mesh, as has

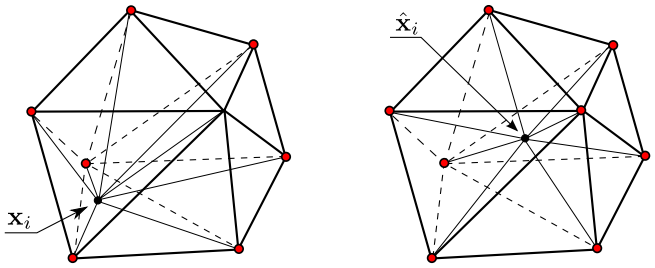


Fig. 6. Laplacian smoothing operation in 3-dimensional space. (a) initial node position and (b) final node position after successively smoothing steps.

been reported by Taubin [**?**]. Therefore, a scale-dependent Laplacian approximation seems to be ideal for all the 3-dimensional simulations. Its difference, compared to the uniform weighted method, is that $w_{ij} = 1/|e_{ij}|$ where e_{ij} is the distance between the node and each neighbor. Thus, the node sliding is less pronounced and the shape of the elements converges to a more equidistant distribution of vertices. Thus, the final Laplacian smoothed distribution is obtained by:

$$\hat{\mathbf{v}}_{e_i} = \frac{\sum\limits_{i \in N_1(j)} e_{ij}^{-1}(\mathbf{x}_j - \mathbf{x}_i)}{dt} \tag{45}$$

Additionally, another technique has been tested here which defines the velocity $\hat{\mathbf{v}}$ as a function of the neighbor's velocities instead. If an iteration process is used, the velocity is spread smoothly over the vicinity of the surface mesh. The advantage of such an approach is that for high velocity gradients, when the elements tend to collapse, the surface mesh velocity is distributed around the neighborhood of each node next to the surface, thus moving them all in the direction of the surface node normal vector. The procedure is represented by the following scheme:

$$\hat{\mathbf{v}}_{v_i} = \frac{1}{n} \sum_{j \in N_1(j)} \mathbf{v}_j \tag{46}$$

where N_1 is the set of the 1-ring neighbors to the jth node, v_i is the velocity associated to the ith node and n is the number of mesh neighbors to the ith node. The mesh velocity is represented by $\hat{\mathbf{v}}_{v_i}$, which is the mesh velocity. Note that with such an approach, the nodes close to the surface mesh (interface and domain boundaries) will behave like those of the surface mesh. Thus, it is expected that these nodes will not collapse nor cross a triangle interface. This scheme has been successfully applied to overcome the fast element distortion close to the interface where the velocity gradient may be high.

Figure **??** illustrates the above velocity repositioning scheme in 2-dimensional space and its simplicity. The surface's velocity is scattered gradually to the nodes near the surface, the closer is the node to the surface, the stronger is the influence of the moving interface. This scheme is an important tool to avoid collapsing nodes and surface elements. Note that its extension to a 3-dimensional space is straightforward by considering an embedded surface in $\mathbb{R}^3$ as the interface between the fluids.

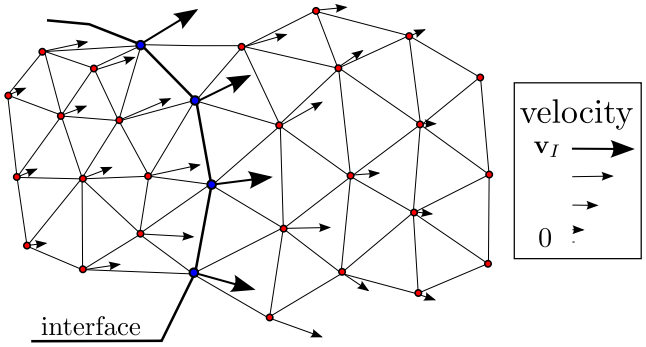


Fig. 7. Velocity smoothing operation in 2-dimensional spaces. Near the interface, the nodes are more influenced by the surface velocity (large arrows), while if the node is located far from the surface, the mesh velocity $\hat{\mathbf{u}}_v$ is less pronounced (small arrows). Its analogy to 3-dimension space is straightforward by considering a surface embedded in $\mathbb{R}^3$ as the interface between the fluids.

Moreover, the velocity smoothing scheme proposed here is useful for simulations involving bubble/bubble and bubble/wall interactions. Close to the boundaries, the volumetric nodes are moved according to the interface's velocity, which is responsible for pulling the nodes away, and the zero boundary mesh velocity, which stops the motion of an approaching node. Thus, the volumetric nodes are compressed and squeezed in the gap between the interface and boundary meshes, such that the precision and the number of nodes remain unchanged.

Due to the separation of the domain and the surface mesh in the above procedure, the mesh distribution treatments may be combined into a scheme and adjusted by parameters varying from 0 to 1, which is a 3-dimensional generalization of the approach presented by Souza and Mangiavacchi [**?**] for 2-dimensional simulations. The domain and surface velocities are therefore treated as follows:

$$\hat{\mathbf{v}}(\mathbf{x}) = \begin{cases} \mathbf{v} - \gamma_1(\mathbf{v}\cdot t)t + \gamma_2(\mathbf{v}_e \cdot t)t & \text{if } \mathbf{x} \text{ belongs to the interface} \\ \beta_1\mathbf{v} + \beta_2\mathbf{v}_v + \beta_3\mathbf{v}_e & \text{if } \mathbf{x} \text{ does not belong to the interface} \end{cases} \tag{47}$$

In such a method, due to the description of the interface mesh by computational elements, the surface should move according to the fluid motion. In the above equation, if $\mathbf{x}$ belongs to the interface, we can define its velocity as $\mathbf{v}_I$. Thus, it is convenient to decompose it into two orthogonal components: $\mathbf{v}_{I_n}$ and $\mathbf{v}_{I_t}$ which

represent the normal and tangential velocities, respectively. To decrease the displacement of nodes in the tangential direction, one may remove partially, or even totally, its velocity from the total interface's velocity. This can be achieved by either projecting the interface's velocity $\mathbf{v}_{I_n}$ to the normal vector associated to the node or, in a simpler manner, by removing the tangent component from the total surface mesh velocity $\mathbf{v}_{I_t} = \mathbf{v} - (\mathbf{v} \cdot t)t$. Figure **??** shows such a decomposition of the interface's velocity $\mathbf{v}_I$ into two orthogonal vectors. Such a procedure may be included into a scheme so that the intensity of the tangential velocity can be easy modified. Therewith, the parameter γ_1 controls the magnitude of the tangent velocity in the total interface's velocity. Letting $\gamma_1 = 1$, only the normal interface's velocity is taken into account in the surface mesh motion, and therefore the surface nodes are not allowed to move in the tangential direction. Additionally, the parameter γ_2 includes the smoothing scheme in Eq. **??** on the surface mesh nodes, thus keeping them all bounded within a good aspect ratio. The parameter β_1 controls the Lagrangian motion of the inner and outer volumetric mesh velocity. By setting $\beta_1 = 1$, the flow velocity $\mathbf{v}$ is fully included in the moving mesh velocity $\hat{\mathbf{v}}$ and, consequently, the volumetric nodes move according to the flow field. Otherwise, letting $\beta_1 = 0$, the flow velocity $\mathbf{v}$ is not taken into account on the moving mesh velocity. The parameters β_2 and β_3 control the intensity of the velocity smoothing scheme $\mathbf{v}_v$ and the Laplacian smoothing scheme $\mathbf{v}_e$ into the moving mesh velocity. Thus, setting both parameters to null, the volumetric mesh smoothing is not performed. Note that the parameters γ and β may vary from 0 to 1 to achieve a desirable node distributions according to the simulation requirements.

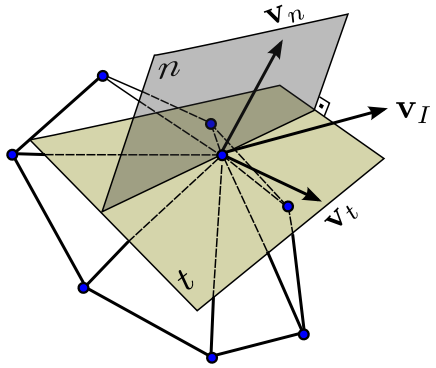


Fig. 8. Normal and tangent components of the interface's velocity vector. The proposed scheme allows to remove partially or totally the tangent component of the interface's velocity $\mathbf{v}_I$ by varying the parameter γ_1.

To illustrate the influence and flexibility of this technique, several cases for choosing these mesh parameters are discussed below.

The most immediate case which requires very strict mesh control is the sim-

ulation of a bubble in a gravity-driven flow. In such a demanding condition, the computed flow field tends to drag the surface nodes from one region to another mainly due to the higher shear stress close to the interface. The flexibility of the parameters presented in Eq. **??** allows one to chose $\gamma_1 = 1$, so that the tangential surface velocity is completely eliminated from the equation and therefore the surface nodes are allowed to move only in the normal direction. Additionally, the wake and vortex formed by the bubble's ascension push the surface nodes around the bubble's surface, compressing them one against the other; thus a stiff mesh is recommended to avoid such a problem and this is achieved by setting $\beta_1 = 0$. One may also adjust the parameters β_2 and γ_2 between 0 and 1 to displace the nodes close to the interface and to optimally redistribute the new surface mesh elements respectively, as well as the value of β_3 to guarantee better tetrahedron aspect ratios.

In contrast to the above example, if the bubble or drop is nearly static or its displacement with respect to the domain is negligible, different mesh parameters are required. Due to the low level of motion of the flow field, the mesh velocity may be set to a pure Lagrangian motion ($\beta_1 = 1$), thus describing the fluid convection with higher precision and a lower number of mesh nodes compared to a standard fixed mesh simulation. The other parameters may be adjusted according to the need to preserve the mesh quality. Moreover, if the flow is controlled by an inflow velocity condition, letting $\beta_1 = 1$ and $\gamma_1 = 0$ may be the appropriate choice for such a problem.

A third case is illustrated for shear-driven simulations. Due to the prescription of a velocity inflow condition, a pure Lagrangian motion is not recommended due to the strong mesh distortions that may appear, especially close to the boundaries. Thus, it is recommended to set the mesh parameter $\beta_1 = 0$. This implies a stiff volumetric mesh. The *elastic* velocity parameters β_3 and γ_2 may be adjusted to 1, thus maintaining the nodes well distributed for both the volumetric and the surface meshes. The parameters β_2 and γ_1 may be arbitrarily chosen to fit the requirements of the mesh. In general, when several of these conditions are present, then one must do a pre-study to find a good choice of parameters that handle the various facets of the problem being faced.

4.4. *Surface remeshing*

Unfortunately, mesh smoothing by itself is not able to keep all the elements bounded to optimal shapes after numerous iterations. Furthermore, the moving front creates a poor distribution of surface nodes which can affect the accuracy of the computed curvature and, consequently, the final solution. Since the connectivity of the surface mesh is handled by the code, a re-meshing technique is thus required to keep the surface element's aspect ratios in a satisfactory range as indicated by ([**?**], [**?**], [**?**], [**?**], [**?**]). The technique proposed here consists of changing the connectivity of the surface nodes and neighboring elements through "flipping" operations. Additionally, insertion and deletion of nodes is required when a coarse surface mesh is

detected (edge length h is too long) or when a dense cluster of surface nodes is not desired, respectively. The new methodology proposed for insertion, deletion, contraction and flipping of triangular edges on the surface mesh is described below.

4.4.1. *Node insertion*

The strategy for insertion of new nodes aims to occupy "barren" areas and to increase the accuracy in certain regions of the surface mesh where a higher precision is required. Different techniques and insertion criteria can be found in the literature, especially in the computer graphics area, and thus it is desired to select a suitable approach. In two-phase flows, the flow field formed by the motion of a single bubble tends to move the surface nodes from one region of the interface to another, producing non-uniform node distributions. Furthermore, when the bubble interface moves away from another surface, the 3-dimensional elements are slightly stretched and consequently their aspect ratios change and thus the insertion of nodes may be required to maintain the desired mesh quality.

Figure **??** shows two connected triangles with vertices numbered from 1 to 4. When, due to stretching, edge $1-2$ becomes longer than a predefined length, a new node v is inserted in the middle of the edge $1-2$, thus dividing the segment into two equal parts, and consequently creating two new elements which are both part of the surface mesh.

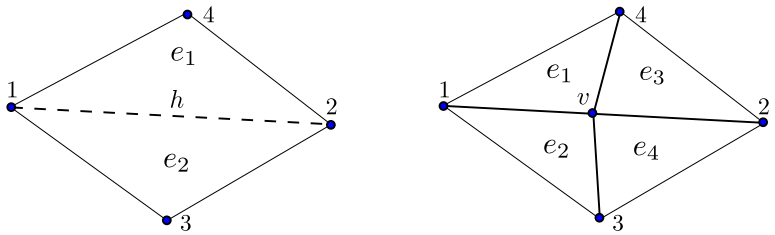


Fig. 9. Insertion of a surface node. (a) The edge $1-2$, which is longer than a fixed parameter h_{max}, is identified. (b) The new node is then added at the mid node of the edge $1-2$.

Secondly, if the new node v is inserted on the segment that defines the edge $1-2$ (see Fig. **??**(a)), it will introduce a local curvature error that is proportional to l/h, where h represents the triangle edge length and l the distance from the edge $1-2$ and its correct position considering the local mean curvature, thus adding a perturbation that affects the accuracy of the computation of the surface tension force. To minimize such an undesirable error, the new node must be placed according to the curvature of its neighbors. This can be achieved by fitting a circular segment which passes through the vertices 1 and 2. As exemplified in Fig. **??**(b), the θ-plane may be defined by the mean normal vector of two adjacent triangular elements, namely

$1-2-3$ and $1-4-2$. The normal vector associated to the nodes 1 and 2 are projected onto the θ-plane forming $\mathbf{n}_1$ and $\mathbf{n}_2$ respectively. The intersection of the line of action of these two vectors is chosen as the approximate center (x_c,y_c) of a circumference $(x-x_c)^2+(y-y_c)^2=r^2$ with radius r. Thus, the solution of this equation is used to determine the displacement l, and consequently the final position of the new node v, as can be seen in Fig. **??**(c).

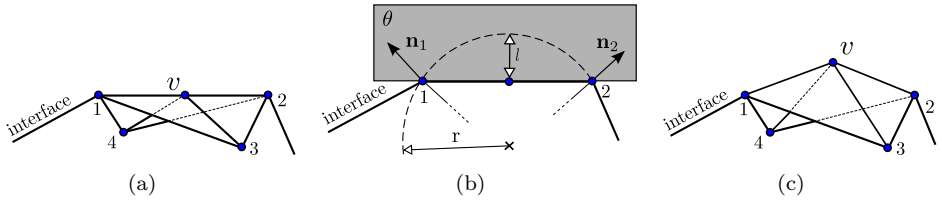


Fig. 10. Representation of the 3-dimensional triangular surface mesh. (a) The node v is added at the mid node of the edge $1-2$; (b) The plane θ is derived by the mean of two element normal vectors which are adjacent to the edge $1-2$. The vectors $\mathbf{n}_1$ and $\mathbf{n}_2$ are the projection of the normal vectors of nodes 1 and 2 onto the plane θ; (c) The node's new position is found by moving it from the edge $1-2$ toward the circle segment in (b), thus the curvature error in v is reduced.

4.4.2. *Node deletion*

The displacement of vertices in the moving mesh technique may cause a dense clustering of nodes in particular areas of the domain, including the triangular surface mesh. Consequently, with the regrouping of these nodes, the elements become smaller and the time step size decreases due to the *Lagrangian* motion restriction. Additionally, the total number of nodes belonging to the mesh increases, increasing the processing time. The additional number of vertices does not necessarily bring real benefits to the accuracy of the final solution and thus these unnecessary vertices should be eliminated. There are many ways to delete a single mesh node and conserve the mesh quality for the next iteration. The present work has adopted two different techniques to keep the mesh bounded to the Delaunay properties. Both approaches attempt to find an edge h that is smaller than a predefined length h_{min}. Once such an edge is detected, the method computes the sum of the edge length of the 1-ring neighbors of both extremity vertices. The one which has the lowest value is then considered for elimination from the surface mesh. Once the node deletion is performed, its neighbors form a polyhedron which must be subdivided to recover the triangular structure. Figure **??** shows the deletion of a surface node and the polyhedron that then has to be remeshed.

Two strategies were tested to reconnect the empty space left by the deletion of the surface node v. The first strategy is simple and its implementation is straightforward. Let us consider the polyhedron $P=\{1,2,3,4,5,6\}$ as show in Fig. **??**. The node 1 is chosen to reconnect successively the nodes 3, 4 and 5 by creating

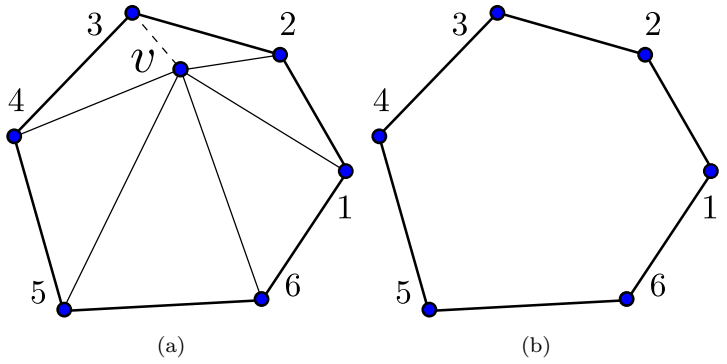


Fig. 11. Deletion of a surface node. (a) The edge $3 - v$ is detected when its length is smaller than a reference length h_{min}. Due to the sum of neighbor edge lengths, the node v is chosen to be deleted. (b) Therefore, the empty polyhedron must be reconnected to achieve a new surface triangulation.

edges on the surface mesh. If the polyhedron nodes have some orientation defined, each triangle may be created by choosing two successive nodes. Thus, the first triangle is formed by connecting the nodes $\{1, 2, 3\}$ and the second by $\{1, 3, 4\}$. The strategy continues to set the third triangle $\{1, 3, 5\}$ and the last one is then defined by $\{1, 5, 6\}$.

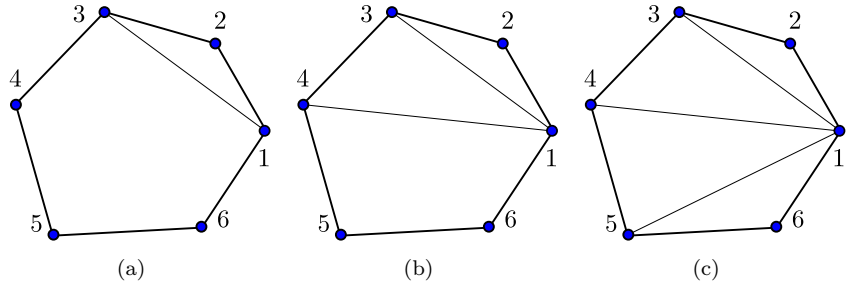


Fig. 12. Remeshing of a surface polyhedron by successive node re-connections. (a) An edge is created by connecting the nodes 1 and 2. (b and c) The node 1 is then connected to the remaining nodes 4 and 5, thus achieving the final surface triangulation.

The second strategy is done based on the work of Devillers [**?**] and Xu et al. [**?**] and extended here to triangular surface meshes. Let us consider a generic polyhedron $P = \{x_0, x_1, ..., x_k, x_0\}$, where the first "ear" of the polyhedron P is defined by the triangle with vertices $x_i - x_{i+1} - x_{i+2}$. Such an ear will be part of the surface triangulation if and only if the segment $[x_i, x_{i+2}]$ is located inside the polyhedron and it does not intercept its boundary. A sub-set of P is formed with the deletion of the node x_{i+1}, and then the new triangular "ear" may be found by repeating the described strategy. The process is iterated until the number of nodes

of the sub-set of P is equal to 3, and therefore the last triangular ear is composed by $x_{k-1} - x_k - x_0$.

Figure **??** shows the schematic representation of the deletion process and remeshing by the "ear" technique. The edge $3-v$ is detected as being smaller than h_{min} and the node v is chosen to be removed. The neighbor triangles of node v are eliminated from the surface mesh and the polyhedron formed by the 1-ring neighbor nodes of v is used to remesh the empty space. In this example, the reconnection of nodes to form the new triangulation is done by defining a polyhedron $P = \{1, 2, 3, 4, 5, 6\}$ and the first "ear" to be $E_1 = \{1, 2, 3\}$. The first triangle is created and the node 2 is deleted from P. The new sub-set of P is defined as $P_{s1} = \{1, 3, 4, 5, 6\}$ and thus the next "ear" as $E_2 = \{3, 4, 5\}$. Node 4 is then deleted from the sub-set P_{s1}, thus creating $P_{s2} = \{1, 3, 5, 6\}$. A new ear is set on the triangle $E_3 = \{5, 6, 1\}$, the node 6 is deleted from the sub-set P_{s2} and consequently the new sub-set $P_{s3} = \{1, 3, 5\}$ is assembled. Since the number of nodes in $P_{s3} = 3$, the last triangle is formed and the local re-meshing is accomplished.

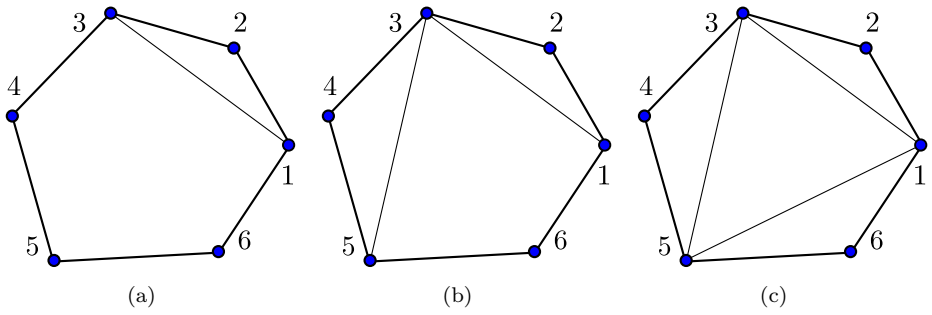


Fig. 13. Reconstruction of the surface mesh by the "ear" technique. (a) First "ear" is achieved by connecting the nodes $1-2-3$ and forming the surface triangle. The node 2 is then deleted from the polyhedron P. (b) The new triangle is formed by connecting the nodes $3-4-5$, and thus the node 4 is eliminated. (c) Last two triangles are created from nodes $5-6-1$ (node 6 is deleted) and $1-3-5$, which are the remaining nodes of the successively deleted polyhedron.

The strategies presented above are not sufficient to guarantee optimal triangular shapes. In fact, the selection of the initial node should be treated with care by considering the polyhedron spatial geometry. For instance, if the polyhedron has a concave shape and the initial node is wrongly chosen, the remeshing procedure will invalidate the triangulation by creating non-triangular elements or even creating elements outside the polyhedron boundaries. "Flipping" operations may be required to achieve an optimal triangle distribution. Such an operation was implemented in this work and it will be described in the next sections.

4.4.3. *Edge contraction*

This strategy is based on the contraction of an edge and it is well discussed by [**?**]. Once an edge h for each $h < h_{min}$ is detected, this scheme aims to collapse the two extremity vertices into the mid node of the edge h; thus the two adjacent triangles are removed from the surface mesh, as shown in Fig. **??**. After it has been contracted, the edge h is no longer part of the surface mesh, so that nodes 1 and 2 occupy the same position while nodes 3 and 4 remain at their locations. The benefit of such an approach, compared to the previous node deletion strategy, is its geometrical simplicity since the surrounding connectivity of the mesh is not affected.

As can be seen in Fig. **??**, the curvature of two adjacent nodes found in the insertion strategy should be taken into account when collapsing two vertices, thus avoiding displacement errors and undesirable loss of mass. This is done by fitting the equation of a circle to the nodes 1 and 2 and considering the curvature of the adjacent nodes $(1-2-3-4)$. Thus, the collapsed node is displaced according to the neighbor curvature values.

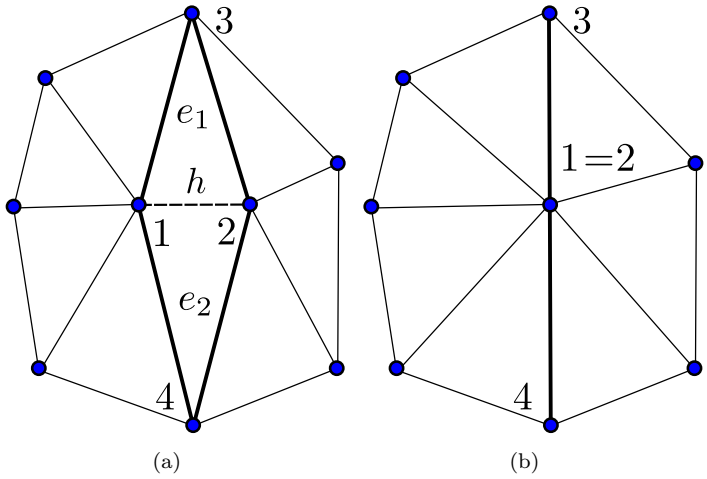


Fig. 14. Contraction of a surface edge. (a) The edge h (segment 1-2) is found to be smaller than h_{min} and (b) so it is collapsed to the mid node of the same edge. Due to its simplicity, only triangles e_1 and e_2 are eliminated from the surface mesh and the remaining node connectivity is not affected. The new location of node 1 should respect the curvature of its neighbors as described in the insertion strategy.

4.4.4. *Edge flipping*

Since insertion, deletion and collapsing of vertices may deteriorate the mesh, edge flipping may be required to restore the mesh quality. Such an operation in a 3-

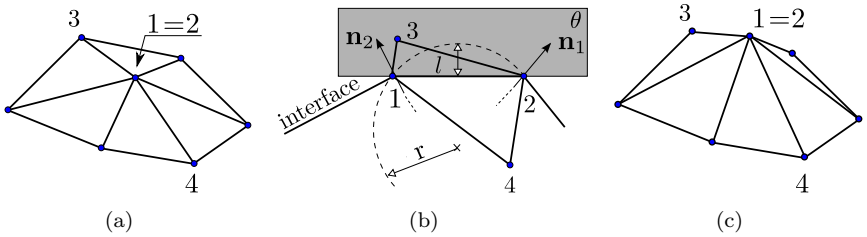


Fig. 15. Node displacement according to neighbor's curvature in the process of edge contraction. (a) The plane Ω is found using the curvature vectors of nodes 1 and 2, thus a circle equation is fitted and (b) its solution is used to displace the node and avoid losses of mass. (c) The resulting scheme of edge contraction considering the neighbor's curvature.

dimensional surface is more restrictive compared to 2-dimensional spaces. The flipping criteria implemented here considers four different measures:

- sum of the triangle's aspect ratios,
- curvature of neighboring nodes,
- sum of the triangular areas,
- circumcenter of each triangle.

These parameters are evaluated at every time step to check if any flipping operations need to be performed. This is achieved by comparing the quality of the initial and the modified pair of triangles; thus if one criteria fail, the flipping operation is not performed.

In the literature, there are many ways to check the triangle aspect ratio such as edge ratio measurement, relative size squared, maximum and minimum angle, etc. (see [**?**], [**?**] and [**?**]). The one chosen to use here considers the radius of the inscribed circle and the longest edge length. This scheme provides quantitatively the quality of a given triangle and thus can be used as parameter in the flipping operation. The neighbor's curvature should be also considered before flipping an edge due to a strong restriction on the surface embedded in $\mathbb{R}^3$. If the curvature κ is too high, the flipping operation can damage the surface mesh, thus forcing the simulation to shut down. In this work this curvature limiter has been adopted to be $\kappa_{max} = 40$, i.e. above this limit flipping is not performed. Note that the maximum allowed curvature of 40 was chosen due to fast degeneration of surface elements, observed in many different numerical experiments. Additionally, the sum of the triangle areas are taken into account. This measure stops the flipping operation if the resulting sum of the areas is smaller than before flipping. Finally, the circumcenter of each triangle is also taken into account as a quality ratio parameter to the final local mesh. Note that these flipping parameters are required to avoid strong mesh degradation and large losses of mass. However, the flipping operation is especially required to keep the mesh bounded to within a satisfactory aspect ratio.

Figure **??**(a) shows a typical flipping operation done on the surface edge. Ac-

cording to the criteria mentioned above, the triangles with vertices $1-2-3$ and $1-4-2$ have lower ratio qualities compared to the triangles with vertices $1-4-3$ and $3-4-2$, and thus the flipping operation is performed and the new mesh is achieved. On the other hand, Fig. **??**(b) shows a case where a diagonal flip should be avoided since it contradicts the triangular surface mesh. The new generated element $3-4-1-2$ is not triangular and the surface mesh is consequently corrupted. As pointed out by [**?**], pronounced loss of mass occurs if the flipping operation is performed when the angle of two consecutive faces is lower than $90°$, where the resulting elements may have a better aspect ratio but the loss of mass may reduce significantly the precision of the simulation (see Fig **??**(c)).

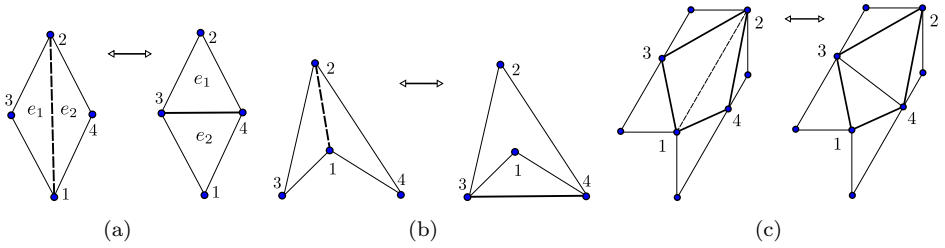


Fig. 16. Triangular surface flipping operations: (a) The triangle aspect ratio, the curvature of neighboring nodes, and the triangle circumcenter are taken into consideration to perform the flipping from edge $1-2$ to $3-4$; (b) the flipping of edge $1-2$ cannot be assigned due to an inconsistent mesh generation. (c) The flipping operation may lead to local loss of mass and it should be treated with care.

4.5. *Volume conservation*

Due to the constant surface mesh treatment, the front-tracking methods are known to accumulate spurious loss of mass during a simulation. However, for incompressible flows, the volume of both phases should remain constant and excessive geometrical operations may lead to excessive loss of mass. To avoid this pitfall, one should minimize the number of flipping operations and limit the deletion and insertion of new nodes. In certain cases where the surface mesh is at constant shear, the number of geometrical operations cannot be reduced. As mentioned before, successive mesh smoothing and the displacement of a new inserted node according to its curvature should be performed. Nevertheless, due to the inherent truncation and rounding errors, the combined mass of the two phases may slightly change. Therefore, a simple treatment has been implemented to compensate for the spurious mass variation, thus avoiding the accumulation of mass conservation errors. This correction is done by moving the surface nodes in the direction of their normal vector. Such a displacement is calculated based on the initial phase volume, which is compared to the current iteration; thus a successive relaxation method is applied to find the final node's positions. The difference between the initial phase volume

and the current time step volume may be chosen according to a given tolerance, in which, for the present work, is on the order of 10^{-8}. The volume conservation algorithm is described in Table 1.

Table 1. Initial surface volume

```
TOL = 1.0E-08;
While abs(error) > TOL
{
 ForEach surface node i
 {
  xNormal(i) = normal vector component of node i;
  edge = local surface edge length size;
  x(i) = x(i) + xNormal(i)*edge*error;
 }
 surfaceVolume = compute surface volume;
 error = 1.0 - surfaceVolume/initSurfaceVolume;
}
```

The geometrical operations on the surface and volumetric meshes are performed preferably at all time steps, as well as the volume correction algorithm and consequently the adaptive mesh refinement is successfully achieved. Thus, the surface and the volumetric meshes are corrected, the phase volumes are adjusted, and the simulation can reach the final state with no significant loss of mass.

5. Interface discretization procedure

This section describes the interface discretization procedure, from its geometrical representation to its nodal surface tension calculation. The Heaviside function is presented and the distribution of fluid properties in the numerical domain is compared to other standard methods found in the literature.

5.1. *Geometrical representation*

In front-tracking codes the interface is constructed by a set of geometrical objects, such as triangles, edges and nodes, which are moved in Lagrangian fashion, while instead the background mesh is fixed in the space. An additional function is required to communicate from one mesh to another, since there is no implicit interconnectivity. This approach leads to the so-called zero-thickness interface, in which a sharp representation is achieved. Although its excellent geometrical definition of a sharp interface, the fluid properties close to the interface require a numerical treatment to avoid undesirable instabilities. Thus, these properties are smoothed along the

transition zone and the zero-thickness is no longer guaranteed.

Unlike front-tracking codes, the present surface and background meshes are part of the same computational mesh, and thus no additional equation is required to pass the information from one mesh to another. The 3-dimensional mesh comprises a set of tetrahedron elements distributed through out the domain and the interface is found by a scalar function, namely Heaviside, which defines the nodes belonging to each phase and the interface itself. To achieve a zero-thickness representation, the interface's nodes must be connected consistently so that the piecewise discrete interface may be represented by a set of interconnected triangles. In other words, each triangle is a face of two adjacent tetrahedral elements. Figure **??**(a) shows a schematic representation of the discrete interface between the two different phases. The same triangular face is shared by two adjacent tetrahedrons, and therefore the zero-thickness interface is successfully achieved.

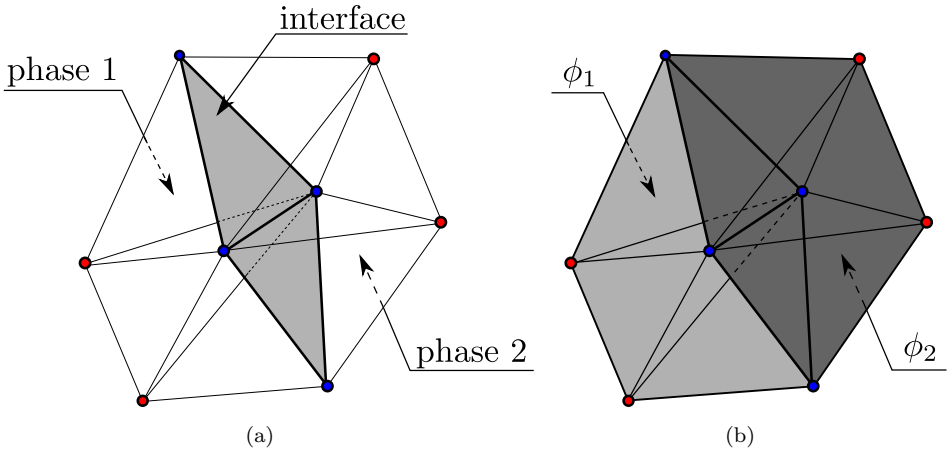


Fig. 17. Geometrical representation of the interface between the phases. (a) The interface (gray colored) is represented by a set of triangles, edges and nodes which are part of the tetrahedron mesh. (b) The fluid property ϕ, such as density or viscosity, is sharply defined in phase 1 and phase 2 with a zero thickness interface in the transition zone.

Advantages and drawbacks are found in such an approach, but one feature that is especially interesting regards the definition of the fluid properties in the transition area. From a macroscopic point of view, the physical meaning of an interface is the region that sharply divides the volume occupied by each phase. Thus, it is desirable that such an interface's thickness should be kept as thin as possible. The Lagrangian description guaranties the geometrical part, but due to the abrupt change in properties from one phase to another, numerical instabilities may appear and deteriorate the accuracy of the solution. Such a problem is mainly due to the location of the interface somewhere in-between two computational elements. This can be circumvented with the ALE and the Finite Element formulation, in which the interface is not located in between mesh elements but it shares the faces of

two adjacent computational mesh elements and thus the fluid properties remain constant in each mesh element. A sharp transition of properties is thus successfully achieved and does not require the use of any smoothing functions, consequently assuring good accuracy in the balance of forces close to the interface.

Figure **??**(b) shows the transition zone between the two phases colored by dark and light gray, which was purposely drawn to highlight the methodology proposed by this work. As can be seen, the property ϕ_1 fills the elements of phase 1 and the property ϕ_2 fills exactly the elements of phase 2. Even for a high property ratio $\phi_1/\phi_2 = 1000$, the methodology proposed here does not present spurious oscillation in the pressure and the velocity field. Figure **??** depicts a 1-dimensional plot of density distribution along the phases and shows a comparison of the density distribution used in the implemented ALE-FEM scheme and two smoothed distributions commonly found in Level-Set methods. Due to the Finite Element Method formulation, each phase property ϕ is assigned to each tetrahedron element, and thus a sharp transition of properties is achieved.

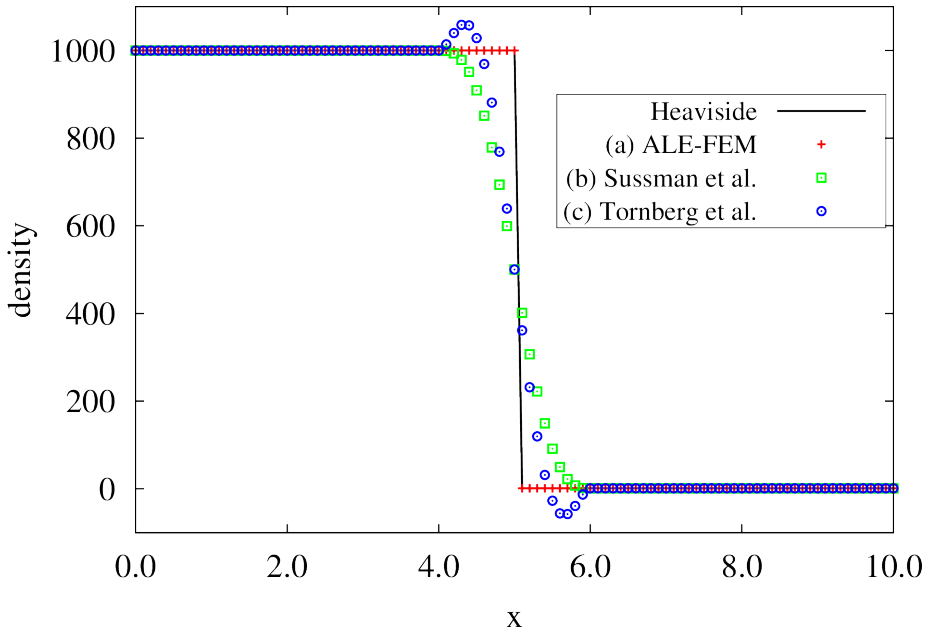


Fig. 18. Density distribution in two-phase flows. Phase 1 has a density $\rho_1 = 1000$ and phase 2 has a density $\rho_2 = 1$. The interface is at $x \approx 0.5$. (a) The sharp transition is achieved by the ALE-FE method, in which no artificial smoothing is required. (b) Smoothed distribution of density commonly found in Level-Set methods (see [**?**] and [**?**]).

Despite the sharp definition of the interface and the fluid properties, topological changes are not naturally handled in this method, thus requiring an implementation effort on the modeling of coalescence and break-up of bubbles and drops. To address

this issue, a geometrical model may be used to merge or split two surfaces. For instance, when the film thickness between two bubbles are smaller than a predefined parameter, the surfaces are connected and coalescence takes place. Although the topological change occurs, the physical aspects are not fulfilled. In fact, the mechanisms of bubble coalescence and break-up is still an open issue and, consequently, a potential field of future research.

5.2. *Curvature and normal vectors in* $\mathbb{R}^3$

The non-dimensional form of the surface tension term can be written as $\mathbf{f} = \mathbf{n}\kappa\delta$, where κ is the curvature and $\mathbf{n}$ is the surface unity outward normal vector. Additionally, δ represents the Dirac delta function with support on the interface. Now, let us consider a distributed surface tension force scheme based on the Heaviside function:

$$\mathbf{f} = \sigma\kappa\nabla H \tag{48}$$

In such a scheme, the distributed interface force $\mathbf{f}$ is a volume force and its intensity $\sigma\kappa$ is calculated and applied in the direction of the gradient of the linear Heaviside function ∇H, where σ stands for the surface tension coefficient. Thus, at the interface, all the surface tension force is well distributed on the free node's neighbors and the effects of overshooting and undershooting are eliminated.

A new scheme is proposed to compute the mean curvature and the normal vectors in 3-dimensional spaces, which will later be used in Eq. **??** to calculate the surface tension force. The interface between the fluids is represented by a set of geometrical objects which defines a surface, hence the new scheme should take into consideration the different topologies that such a surface may present. This scheme is described below.

Let Ω_s be an embedded surface in $\mathbb{R}^3$, n_i be the set of nodes n lying on the surface Ω_s and e_i^j be the set of surface triangles associated to the ith node, where j is the number of triangles, which may vary according to the structure of the surface mesh (Fig. **??**). The surface tension force $\mathbf{f}$ requires the calculation of the mean curvature κ which, in the present model, is defined at each node i. An evaluation of this nodal mean curvature κ_i is done by integrating the elemental force contribution over the 1-ring triangle neighbors and dividing by the corresponding barycentric area. To calculate the elemental force, one should find two unit normal vectors, which lie on the triangle surface, and integrate them on the segments that connect the mid-edges node to the triangle's centroid as shown in Fig. **??**(a). Due to the Stokes theorem, the distributed elemental force $\mathbf{t}_n d$ in Fig. **??**(c) is equivalent to the integral over the segments connecting the triangle mid-edge nodes to the centroid. Thus, a simplified way to calculate the elemental force is achieved by orthogonalizing one of the two vectors ($\mathbf{t}_1$ or $\mathbf{t}_2$ - Fig. **??**(b), finding its unit vector, and integrating the result to the segment d, which connects two triangle mid-edges Fig. **??**(c). Consequently, an

evaluation of the *ith* node mean curvature is found by integrating the intensity of elemental force $\mathbf{t_n}d$ in the 1-ring triangle neighbors e^i_j and dividing it by the sum of the barycentric areas of e^i_j. The mentioned expression is calculated as follows:

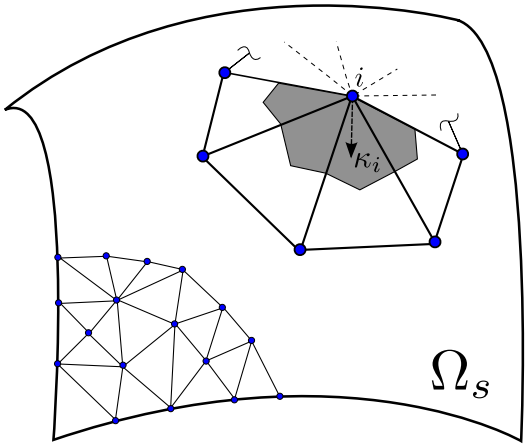
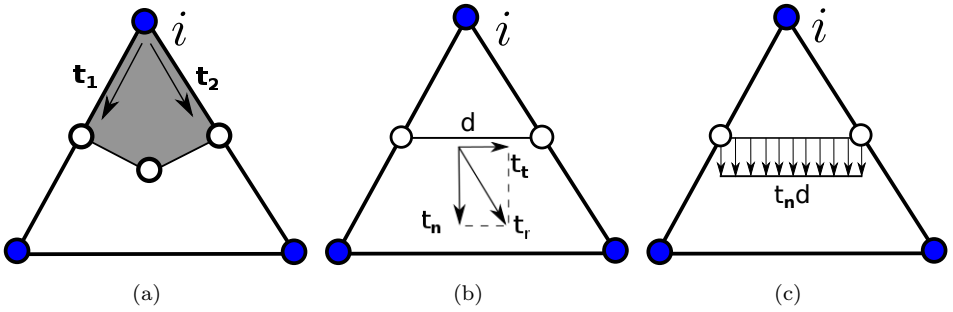


Fig. 19. Schematic representation of the curvature evaluation κ_i at a common surface node, which is calculated using geometric operations at all the triangular neighboring elements and weighted by the barycentric area (gray colored).



(a) (b) (c)

Fig. 20. Surface triangle. (a) An elemental force evaluation is done using the sum of the distributed forces $\mathbf{t}_{1\mathbf{n}}d_1$ and $\mathbf{t}_{2\mathbf{n}}d_2$. (b) Using Stokes theorem, the elemental distributed force may be calculated orthogonalizing one of the two linearly independent vectors t_1 and t_2 to the segment d which connects two mid-edge nodes. (c) An evaluation of the node mean curvature is found by dividing the sum of the module of the calculated distributed forces ($|t_n d|$) by the sum of the barycentric areas (Eq. **??**.)

$$\kappa_i = \frac{\left| \sum_{j=1}^{m} (\mathbf{t_n} d)_j \right|}{\sum_{j=1}^{m} A_j^i} \tag{49}$$

In the above equation, κ_i is the node mean curvature and A_j^i is the barycentric area of the jth triangle neighbor of i, which is equivalent to 1/3 of the triangle area, and m is the number of neighbor elements of i. The direction of application of such a curvature is given by the normal vector, that unfortunately cannot always be defined by the direction of the vector k_i itself due to a singular case where the set of triangular faces e_i^j are on the same plane, and thus the length of the computed vector is equal to zero and its direction is then not defined. In this singular case, to find the normal vector of the node n_i one can approximate it by the sum of the cross product of two vectors of e_i^j since the 1-ring neighbor nodes are consistently sorted. This scheme is an extension for surfaces embedded in $\mathbb{R}^3$ of the previously presented 2-dimensional scheme shown in Fig. **??**.

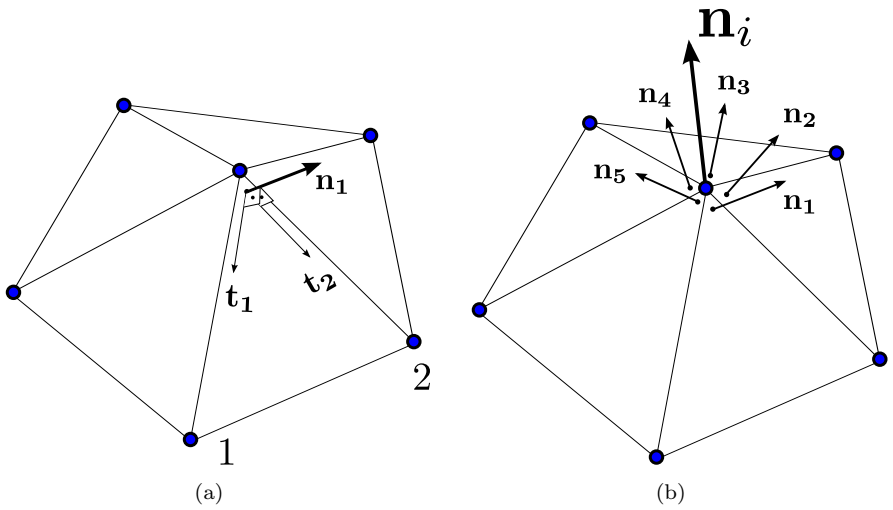


Fig. 21. Normal vector evaluation in 3-dimensional spaces. (a) The normal vector of each triangle in the surface may be found by applying the cross product of two tangent vectors which lie in the same triangle plane. (b) The final nodal normal vector $\mathbf{n}_i$ is found by summing the normal vectors n_e for $e = \{1..j\}$. In the illustrated case $j = 5$.

Due to the unstructured character of the surface mesh, the number of neighbor triangular elements e associated to the node i must be taken into account, since it may vary from one node to another. Moreover, a consistent surface orientation is required in this scheme so that it can be successfully applied. Substituting such a distributed force into Eq. **??** yields:

$$\mathbf{f} = \sigma \frac{\left| \sum_{j=1}^{m} (\mathbf{t_n} d)_j \right|}{\sum_{j=1}^{m} A_j^i} \mathbf{n} \nabla H \tag{50}$$

and the calculation of the surface tension force in 3-dimensions is thus achieved.

5.3. *The discrete surface tension force*

The calculation of the surface tension force is based on the gradient of a Heaviside function ∇H. A Finite Element formulation for the given force is achieved by considering the following scheme:

$$\frac{1}{We} \mathbf{M} \mathbf{f} = \frac{1}{We} \mathbf{\Sigma} \mathbf{G} H_\lambda \tag{51}$$

In the above equation, the symbol $\mathbf{\Sigma}$ represents a diagonal matrix with elements $\sigma\kappa_1, \sigma\kappa_2, \sigma\kappa_3, \cdots, \sigma\kappa_{NV}$, where NV is the total number of mesh nodes relative to the pressure field. The matrix $\mathbf{G}$ stands for the discrete form of the gradient operator ∇ and H_λ is the discrete Heaviside function.

6. Results

Many quantitative and qualitative discussions on the implementation of a 3D Arbitrary Lagrangian-Eulerian Finite Element method may be found in [**?**],[**?**],[**?**]. The reader is also referred to visit two available video libraries showing several test cases, benchmarks and latest results on two-phase flows in macro and micro scale configurations at *http://gustavo.rabello.org/videos* and *http://ltcm.epfl.ch/op/edit/page-70192.html*

7. Conclusions

A numerical method is presented to study two-phases flows for single and multiple bubbles. The governing equations are written in the Arbitrary Lagrangian-Eulerian formulation and discretized by the Finite Element Method. Using such a combination (ALE and FEM), the interface between the fluids is consequently defined by interconnected nodes, assuring a sharp transition in properties. Due to the constant motion of the mesh nodes, an adaptive remeshing process is proposed and suitable for large mesh deformations, including geometric operations to add and remove nodes and edges of the mesh. Moreover, the curvature is conveniently computed through the interface nodes and evaluated as a capillary pressure, leading to accurate results with moderate programming effort and computational cost. As result,

a modern and flexible tool is described to simulate a variety of physical problems in simple and complex geometries for a wide range of working fluid conditions.