An adaptive Level-Set Method with enhanced volume conservation for simulations in multiphase domains

Modesar Shakoor, Pierre-Olivier Bouchard, Marc Bernacki

An adaptive Level-Set Method with enhanced volume conservation for simulations in multiphase domains

M. Shakoor* P.-O. Bouchard
M. Bernacki
MINES ParisTech, PSL - Research University,
CEMEF - Centre de mise en forme des matériaux, CNRS UMR 7635,
CS 10207 rue Claude Daunesse 06904 Sophia Antipolis Cedex, France

May 4, 2016

Abstract

Mechanical computations in multiphase domains raise numerous difficulties from the generation of the initial mesh to its adaptation throughout the simulation. All alternatives to mesh adaptation, such as Level-Set methods, have the well-known drawback of inducing volume conservation issues. In this paper, a moving mesh method is coupled to a topological mesh adaptation technique in order to track moving and deforming interfaces in multiphase simulations, with a robust control of mesh quality. Level-Set functions are used as intermediaries to enhance the mesh adaptation technique with a volume conservation constraint which is compatible both with implicit and body-fitted interfaces. Results show that this method has the same advantage of permitting important displacements, deformations and topological changes (coalescence of interfaces for example) as a standard Level-Set method, while volume diffusion is drastically reduced.

1 Introduction

Finite Element (FE) simulation in multiphase domains arises in many applications of computational solid and fluid mechanics. At a macroscopic scale, fluid-structure interaction involves an heterogeneity which is due to the presence of a solid, usually in rigid body motion, structure surrounded by a fluid, usually in turbulent flow [1, 2, 3]. At a smaller scale, all materials are heterogeneous. For example, one may cite the microstructure of dual phase steels or particle reinforced composites, which contain a main phase, the matrix, second phase particles, and defects [4]. Metal phases can themselves be decomposed into a heterogeneous grain structure [5].
Once the initial mesh is generated, a challenging problem arises in transient simulations: tracking moving and deforming interfaces. Depending on the magnitude of these displacements and deformations from one time increment to another, various methods may be used. A brief and non-exhaustive comparison is presented in Tab. 1. Lagrangian methods can be based on explicit meshes of interfaces or implicit representations [6], where the interfaces are carried by FE fields stored at mesh nodes (Level-Set [7], Phase-Field [8, 9]), or at mesh elements (Volume Of Fluid [10]). Advanced representations have also been proposed in the literature, for example by coupling two of these methods [11], or enhancing an Eulerian Level-Set method with Lagrangian particles [12, 13], or with remeshing techniques [14, 15, 5, 16]. In all cases, large displacements and deformations are difficultly handled by such techniques because of the risk of element flipping. While many commercial codes implement remeshing techniques to prevent element flipping at domain boundaries, it is

*Corresponding author: modesar.shakoor@mines-paristech.fr

Method	Large displacements	Large deformations
Lagrangian	☹	☹
Arbitrary-Lagrangian-Eulerian	☺	☹
Eulerian	☺	☺

Table 1: A comparison of some common techniques for interface tracking.

less common to remesh internal boundaries. Even with implicit interfaces, remeshing is difficult because of the risk of diffusing the FE fields carrying these interfaces. When the interfaces move or deform too fast as in fluid mechanics, or when their topology becomes too complex to be followed as in recrystallization [5], it is common to resort to fixed mesh methods. In this last category of methods, convection is not solved by mesh motion as in Lagrangian methods but through convection equations which involve the FE fields carrying the interfaces. The resolution of such equations raises volume conservation and stabilization issues. Note that although these issues are sometimes addressed using topological mesh adaptation in order to improve the interfaces description [5, 15], the term "fixed mesh methods" will be used here to refer to all Eulerian methods, regardless of whether they are coupled to topological mesh adaptation or not. Finally, an interesting compromise between Lagrangian and Eulerian methods is offered by Arbitrary-Lagrangian-Eulerian methods. The aim of these methods is to avoid as much as possible the volume conservation and stabilization issues raised by Eulerian methods, by activating Eulerian convection only in areas where the deformations are too important to follow by pure mesh motion. Regarding interfaces, explicit representations require the interfaces to remain purely Lagrangian throughout the simulation (otherwise the mesh of the interface would be delayed with respect to the interface itself). This choice was made for example in [17] and also in [18], with the major consequence that it restricted the range of applications to simulations where no large stretching or topological change occurred.

In this paper, a new mesh adaptation technique that combines explicit meshing of interfaces and implicit Level-Set (LS) representation is presented (see Sec. 2). Compared to a standard LS method with moving or fixed mesh, it is shown that this method reduces drastically the diffusion of interfaces during convection and remeshing. Though this approach relies on Lagrangian mesh motion (see Sec. 3), it is proven that it remains competitive for large displacements and deformations, especially regarding volume conservation. An advantage of the whole procedure is that it is purely topological, and may be applied in all dimensions, including space-time frameworks. To meet this requirement, this method is restricted to first order simplex meshes, a simplex being a line segment in 1D, a triangle in 2D, a tetrahedron in 3D, and a pentachoron in 4D. Numerical experiments are proposed with interfaces that undergo large displacements and deformations. Topological changes are also addressed, including a test case where new interfaces are captured on-the-fly during the simulation, such as in fracture mechanics applications.

2 Volume-conserving mesh adaptation in Lagrangian simulations

Depending on the application, various definitions can be used to represent the interface in implicit methods. One may cite the Level-Set method, or the Volume Of Fluid method. Both methods are commonly used in multiphase fluid dynamics, where they are discretized on a fixed mesh, and convected through a convection equation. Though this allows to reduce the dependence on mesh adaptation, it raises a major issue regarding volume conservation. In solid mechanics, this issue may be circumvented by using a Lagrangian mesh, which naturally convects interfaces through mesh motion. Since this process may deteriorate element quality, it is usually combined with remeshing. Mesh adaptation may also be employed together with appropriate error estimators, both in moving and fixed mesh methods. The aim of the method proposed hereafter is to reduce volume loss and interface diffusion caused by the remeshing process itself. Though it is presented in the case where the interface is carried both by a LS function and by the mesh, it can easily be generalized to pure

LS representations (see Subsec. 2.4). In the present case, the domain Ω of dimension d is the union of N subdomains Ω_i corresponding to the N phases, each having its own LS function ϕ_i defined as:

$$\forall t, \forall x \in \Omega(t), \phi_i(x,t) = \begin{cases} +d(x, \partial\Omega_i(t)), & x \in \Omega_i(t), \\ -d(x, \partial\Omega_i(t)), & x \notin \Omega_i(t). \end{cases}$$

2.1 Anisotropic mesh adaptation

Mesh adaptation is an open research problem that has motivated various approaches. Most techniques consist in defining a certain set of operators, and applying them one by one in order to improve mesh quality. Here, the quality is defined for any simplex T as:

$$Q(T) = \min\left(c_0 \frac{|T|_M}{h_M^d}, h_M^d, \frac{1}{h_M^d}\right), \tag{1}$$

where:

- c_0 is a normalization factor so that a regular simplex would have a quality of 1, $c_0 = \frac{d!}{\sqrt{d+1}} 2^{d/2}$,
- $|T|$ is the volume of T and $|T|_M = |T|\sqrt{\det(M(T))}$,
- h_M is the Euclidean norm of edge lengths $h_M = \sqrt{\left(\frac{1}{c_1}\sum_{(i,j)\in\mathcal{N}(\{T\}), i<j} ||S^j - S^i||^2_{M(T)}\right)}$, with $\mathcal{N}(\{T\})$ being the set of the nodes of T, and S^i (resp. S^j) being the point corresponding to node i (resp. j),
- c_1 is the number of edges in a simplex, $c_1 = \frac{d(d+1)}{2}$,
- $||v||_A = \sqrt{v^t A v}, v \in \mathbb{R}^d, A \in \mathbb{R}^d \times \mathbb{R}^d$.

In these formulae, M is a metric field defined at mesh nodes which drives anisotropic mesh adaptation, and $M(T)$ is the interpolation of M at the center of T. Though the presented algorithms enable anisotropic meshes, only uniform isotropic meshes will be used here, namely, $M = diag(\frac{1}{h^2})$, where h is a prescribed uniform mesh size. Summing up Eq. 1, an over-sized, under-sized or ill-shaped element has a quality close to 0, and an appropriately sized and shaped element has a quality close to 1.

Regarding the set of operators used to adapt the mesh, many propositions can be found for example in [17, 19]: node smoothing, edge swapping, edge splitting, edge collapsing. As detailed in [20], edge swap itself can be declined into many different versions in 3D. Most techniques consist in considering local patches of elements in the mesh, and trying to improve them with each one of these operators. These techniques often distinguish the optimization of mesh size from the optimization of element quality by first applying edge split and collapse to reach an optimal size on every edge of the mesh, and then trying to improve element shape using node smoothing and edge swap [17, 19].

Here, the technique used in [21] and [22] is preferred. This technique is presented in Alg. 1. All nodes and edges of the mesh are considered one by one. Note that though considering nodes and edges proves to be sufficient in practice, this method can also deal with the faces of the mesh. The patch of elements to improve is defined as all elements that contain the given node or edge (line 12). The external faces of this patch are extracted easily by searching for the faces that belong to only one element of the patch. Then, for each node of the patch, the generic operator consists in connecting all the external faces to this node (line 16). The qualities of the elements built by this procedure are computed for each tested candidate, and the winner is the one having its worst element with the highest quality (line 17). This generic operator, named *star-connection* operator, has the advantage of considering all swapping, splitting and collapsing operations at once. Node insertion and smoothing are implemented by adding the center of the patch to the list of candidates (line 15). Though no comparison has ever been performed with classical approaches, a possible drawback of considering all operations altogether is that some operations may be redundantly tested.

An example of patch for a nodal target is described in Fig. 1a. After extraction of the external faces, as

shown in Fig. 1b, several candidates are tested (Figs. 1c - 1h), with swapping operations that cover a larger set of topological possibilities than a simple edge swap. Finally, the node smoothing candidate in Fig. 1h seems to be preferable (in the isotropic case).
A similar example is described in Fig. 2, but with the target being an edge of the mesh. While the candidate 2e is equivalent to an edge split, the candidates in Figs. 2c and 2d are equivalent to a simple edge swap. Moreover, these two candidates consist in the same topological modification, which is an example of the redundant operations performed by Alg. 1. However, this algorithm has certain advantages compared to classical approaches.

- When edge length optimization is uncoupled from element shape optimization, there might be a risk of infinite loop [17, 19], as element shape improvement may require to collapse edges that were split during edge length optimization. Uncoupling edge coarsening and refining also raises the same risk [17, 19]. Here, since the quality criterion takes simultaneously into account both objectives, such issues are avoided and convergence is obtained without restricting the maximum number of iterations in any way.

- The proposed method is purely topological. This point is essential for the implementation in order to avoid classical programmer mistakes inherent to the implementation of geometric procedures. In particular, Alg. 1 is far easier to implement than classical remeshing algorithms. Another major consequence of avoiding geometric operations is the extension to space-time methods, and, in particular, to remeshing in four dimensions (see the appendix A of [23]).

- Any other local remeshing operation can be introduced without significant programming effort. The only modification to make in order to add new operations is to change the patch definition at line 12 of Alg. 1. Indeed, the algorithm works for any patch of connected elements (the domain formed by the patch has to consist in one connected component).

Regarding implementation details, they are hidden behind the undefined structures and functions used in Alg. 1. In Fitz [22], the C (as of [24]) implementation of this algorithm developed in the present work, two arrays are used to store the mesh, one containing each node's coordinates and the other each element's connectivity. The elements neighboring each node are stored in an additional two-dimensional array, which enables to increase the number of neighbors for one node without modifying the whole structure. The *Mesh* structure contains these three arrays. Though using arrays may seem inappropriate for a problem like remeshing which implies important changes in the number of nodes and elements, they are preferred here because they avoid repetitive memory allocations. When at any point of the algorithm, it is tried to add a new item to an array which has already reached maximum capacity, this array is reallocated with twice its current capacity. This functioning is similar to the C++ class vector, and the number of reallocations is proven to be negligible and independent from the number of nodes or elements.
The *Node* structure is just an integer referring to a node's position in the coordinates array. The *Element* structure is a small connectivity array used to store an element. It can also be used to store faces, edges, or nodes. This enables the *ElementList* structure used for queuing to contain faces, edges or nodes. This structure itself is a linked list.
Based on these structures, the functions *addAll()* and *nodes()* at line 7 of Alg. 1 are easy to implement. The function *edges()* at line 8 consists in collecting all edges of the mesh and eliminating duplicates. This last operation requires to sort the edges. Sorting is used repetitively in Fitz, and the quick-sort algorithm has been chosen to perform this task. Function *notEmpty()* at line 10 is obvious, together with *poll()* at line 11, which removes the last item from the queue and returns it. Operation *neighbors()* is solved directly by the neighbor array, and the intersection at line 12 is based on sorting and dichotomic search. Note that at the same line, *patch* is not actually a complete mesh, since neighbors do not have to be computed.
The barycenter of the patch is computed by function *barycenter()* at line 15, and *starConnect()* at line 16 has already be detailed. Note that the external faces of the patch only have to be extracted once for all candidates. At line 17, the comparison criterion has already been explained, and the function *isConform()* is a key point of the proposed algorithm which is detailed in the next subsection. Finally, line 23 implies

Algorithm 1 Quality maximization through local topological operations, the lines in bold contain key operations that are enhanced in the present work

```
1:  function MeshAdapt(Mesh mesh)
2:      Node node
3:      Element element, target
4:      ElementList queue
5:      Mesh patch, candidate, winner
6:
7:      queue.addAll(mesh.nodes())
8:      queue.addAll(mesh.edges())
9:
10:     while queue.notEmpty() do
11:         target ← queue.poll()
12:         patch ← ⋂_{node ∈ target.nodes()} { node.neighbors() }
13:
14:         winner ← patch
15:         for node ∈ ({ patch.nodes() } ∪ { patch.barycenter() })\{ target.nodes() } do
16:             candidate ← patch.starConnect(node)
17:             if candidate.isConform() and candidate > winner then
18:                 winner ← candidate
19:             end if
20:             end for
21:
22:             if winner ≠ patch then
23:                 mesh ← (mesh \ patch) ∪ winner
24:                 queue.addAll(winner.nodes())
25:                 queue.addAll(winner.edges())
26:             end if
27:         end while
28:         return mesh
29:     end function
```

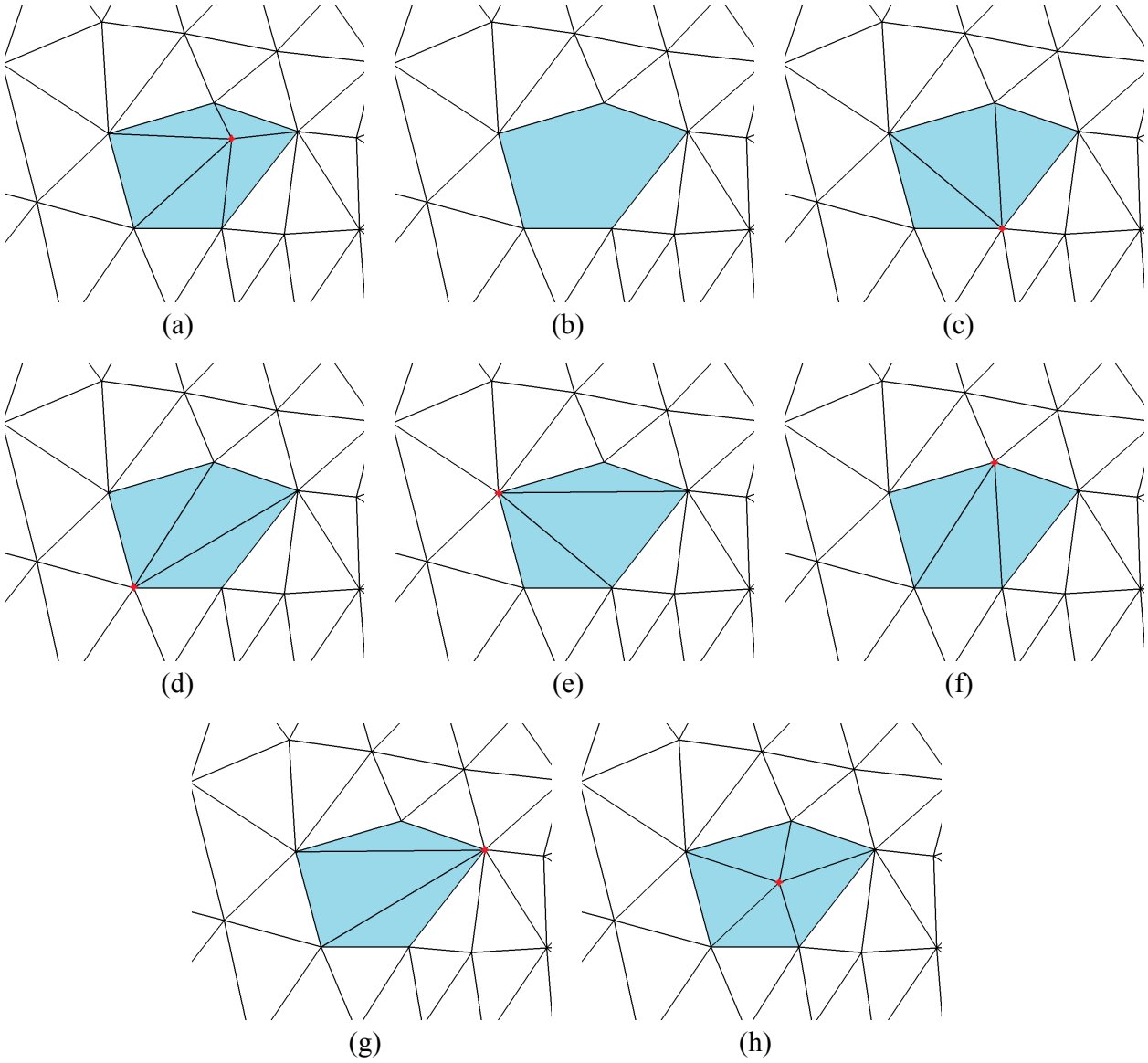


Figure 1: Example of patch at the neighboring of a node: (a) initial patch, (b) external faces extraction, (c-h) candidates.

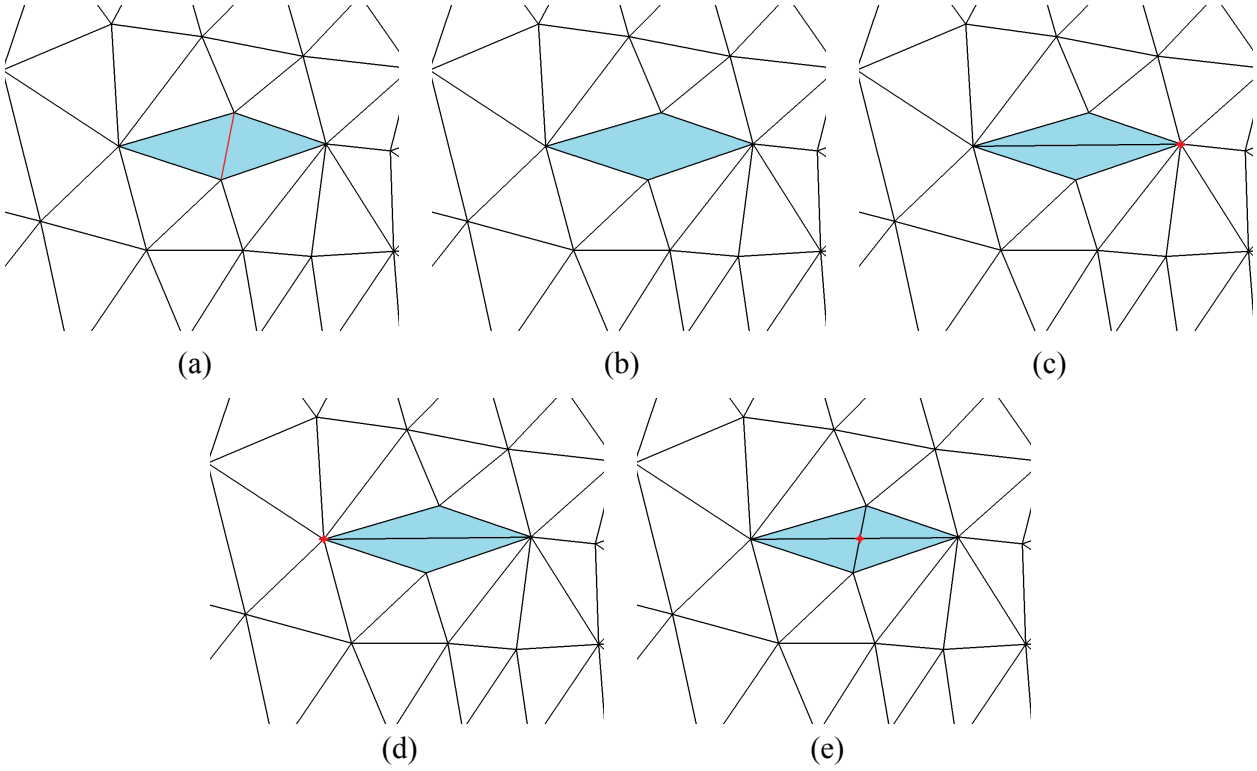


Figure 2: Example of patch at the neighboring of an edge: (a) initial patch, (b) external faces extraction, (c-e) candidates.

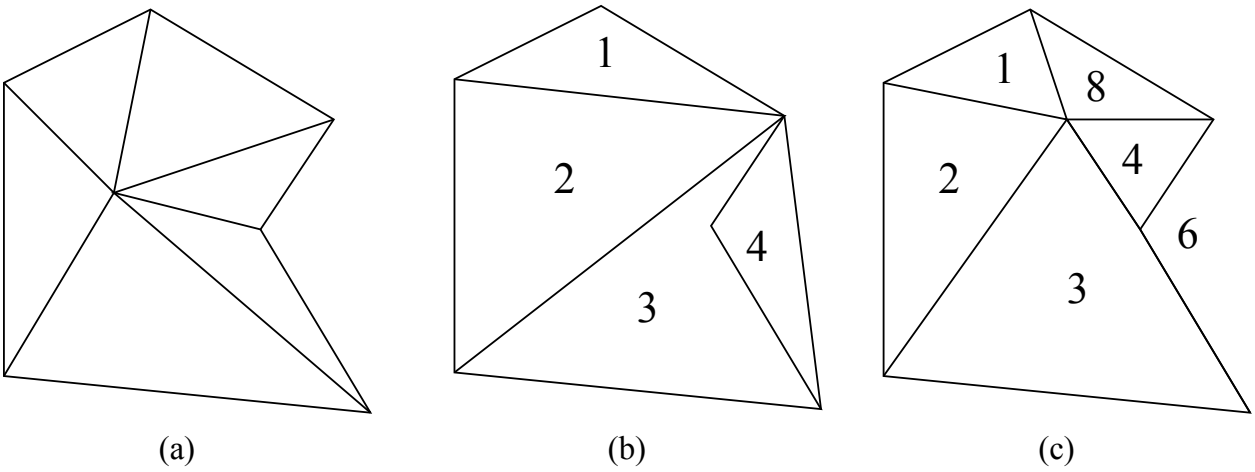


Figure 3: Initial patch (a) and forbidden remeshing operations: (b) element 4 has negative volume, (c) element 6 has a quality that is lower than the initial patch's worst quality.

replacing obsolete elements with new ones in the connectivity array, and possibly inserting a new node in the coordinates array.

2.2 Mesh adaptation with internal interfaces

The extension of the previous algorithm to conserve preexisting internal boundaries is implemented in the function *isConform()* at line 17 of Alg. 1. In the standard definition, this function only has to verify that the new candidate is a conform (in a FE sense) mesh of the domain defined by the previous patch. By construction, this candidate is a well-defined topology: it has same external faces as the previous patch, and none of its faces has more than two neighbors. For this topology to be conform, two constraints have to be added: the new elements have to be correctly oriented (*i.e.* positive volume), and the total volume of the candidate patch has to match the volume of the previous patch. This constraint prevents element overlapping and flipping, while element degeneration is prevented by the quality criterion. In Fig. 3, two variants of an initial 2D topology 3a are presented. Both variants have the same total volume as the initial patch, but they will not be considered because 3b has a negative volume element (flipped element) and 3c does not maximize the quality criterion (degenerated element).
To address the conservation of internal interfaces, LS functions come into play. Inside the initial patch, an element is attributed to a phase if the associated LS function is positive or null at its barycenter. If multiple LS functions are null for the same element (pathological case), this element is attributed to the phase which has lower index. In Figs. 4a and 5a, two 2D examples (respectively with 2 and 3 phases) are shown where the colors indicate which phase an element belongs to. Based on this attribution, the interface faces corresponding to a phase are the faces that separate an element of this phase from elements belonging to other phases. Note that in Fig. 5a, interface faces are pointed out by markers, but there is no marker for the blue phase because it has no LS function.
The first constraint added to the operation *isConform()* is: if there are any interface faces inside the patch, the candidate chosen at line 16 has to lie on all of them. For example, the operation illustrated in Fig. 4b is not conform with this new definition. This constraint was already used in [22] and promising results were obtained in 2D with two phases. The fact that the candidate has to belong to all interfaces extends this constraint to any number of phases, as illustrated in Fig. 5b.
For the 3D applications addressed in the present work, this constraint proved to be insufficient because it authorized aggressive interface remeshing, leading to important diffusion. Therefore, another constraint is added: the distribution of phase volumes has to be conserved in the new patch. This constraint is really restrictive and nearly blocks interface remeshing for shapes with high curvature. In the case where the worst

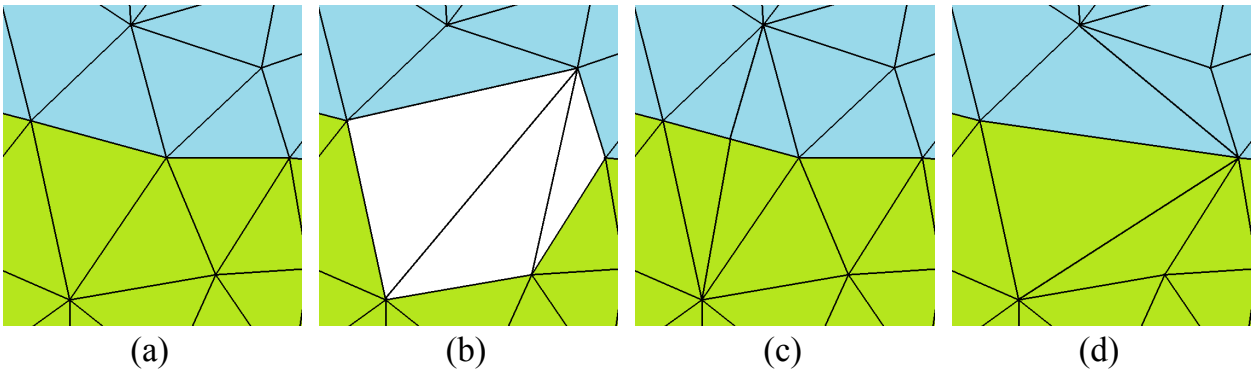


Figure 4: Mesh adaptation with two phases: (a) initial patch, (b) illegal operation, (c) conform operation, (d) relaxation of the phase volumes conservation constraint.

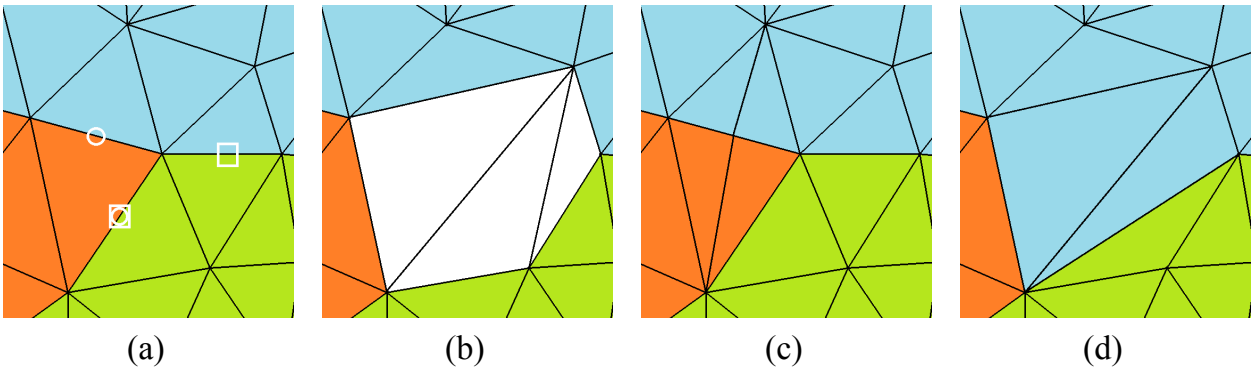


Figure 5: Mesh adaptation with three phases: (a) initial patch with the interface between orange and other phases pointed out by white circles, and the interface between green and other phases pointed out by white squares, (b) illegal operation, (c) conform operation, (d) relaxation of the phase volumes conservation constraint.

quality of the initial patch is below a certain quality threshold Q_y, this constraint is relaxed to prevent issues during FE resolution. For example, the operations in Figs. 4c and 5c are always conform, while only relaxation can enable the operations in Figs. 5d and 5d. In practice, the value of Q_y is chosen empirically as the minimum value for which the influence on the conditioning of the FE problems is negligible, which is in general between 0.05 and 0.1. For higher values, relaxation would be too important and volume conservation would be severely affected. For lower values, convergence of the FE solvers would be affected, depending on whether metric anisotropy is important. An exception is presented in Subsec. 4.0.1 in the case of rigid body motions, where a value of 0 can be used (no relaxation).

2.3 Robustness improvement

Ideally, thanks to the strategy defined in the preceding section, no element should have a quality lower than Q_y after remeshing. In practice, numerical experiments with domains involving more than two phases revealed that this situation could occur at intersections between multiple phases. Improving the robustness of the method is essential to avoid the presence of very low quality elements, that may reduce the conditioning of the FE resolution and hence increase computation time. It is also essential to the remeshing strategy itself: if the remeshing algorithm is efficient enough, relaxation of the volume preservation constraint should happen very rarely.

In this aim, new operations were added to the remeshing process. As stated in Subsec. 2.1, this is easily done here by adding a new possibility, called wide gather, to the patch definition at line 12 of Alg. 1. This definition is activated only when the following conditions are met at the end of an iteration of the remeshing loop: the target was an edge, no modification was performed (the condition at line 22 was not verified), and the patch contains an element with quality under Q_y. Under these conditions, at the next iteration, the target is kept to the same edge, but the intersection operator $\bigcap$ at line 12 is replaced by a union operator $\bigcup$. This simple change adds a wide range of new topological modifications, including edge collapse. Since, by construction, the patch still consists in one connected component, the FE conformity and volume conservation criteria remain valid. An example of patch constructed using this new definition is pictured in Figs. 6a and 6b. Among all candidates, the patch pictured in Fig. 6i is aesthetically interesting, but is not reachable directly using the standard patch definitions of Figs. 1 and 2.

2.4 Concluding remarks

In the present paper, the mesh explicitly represents the interfaces, as in Fig. 4. This is essential for the accuracy of the FE resolution at the interface. A comparable accuracy could also be obtained using the eXtended Finite Element Method (X-FEM), as described in [6]. Though only explicit meshes will be used here, it is important to precise that since the volume conservation condition defined in Subsec. 2.2 is purely based on the distribution of phase volumes, it is readily applicable to implicit methods. If interfaces are implicit, these volumes can be accurately computed by subdividing the elements crossed by the interfaces, as already required by X-FEM [6]. Regarding the interface preservation case, it is dropped since there are no interface faces. Therefore, in relaxation mode, all operations are allowed. Hence, implicit methods may ease the remeshing process, with the advantage of volume conservation.
Additionally, element subdivision can be performed by pure edge splitting, as defined in the following section. Thus, it can be observed that the implementation of the volume conservation constraint is purely topological, and in particular, the whole algorithm remains dimension-independent.

3 Mesh motion and transport

In this section, a simple iterative technique is described in order to move mesh nodes, based on a given displacement field, without triggering element flipping. This technique is no different from usual procedures implemented in commercial codes or used for example in [17]. However, its coupling with the relaxation method defined in the preceding section gives a powerful tool that can handle large deformations and complex entanglements of interfaces. The problem of transporting history variables after each remeshing process is also addressed here.

3.1 Mesh motion

Since convection is modeled by mesh motion in Lagrangian methods, there is a risk of element flipping locally if an element is misaligned with respect to the direction of the displacement increment, or if an element is small with respect to the magnitude of the displacement increment. This is illustrated in Figs. 7a and 7b, where the displacement field is voluntarily singular (only one node is moving). The best way to avoid such situation is to maintain every element as close as possible to a regular simplex [20]. This is the reason why mesh anisotropy is generally avoided in Lagrangian methods, or at least controlled carefully in the regions where the displacement field is known *a priori* to be oscillatory.
In Alg. 2, a method is proposed to operate mesh motion progressively, with a constant checking of element flipping and quality decrease. This checking is based on the operations *coordinates()*, *volumes()* and *qualities()*, which are directly solved from the coordinates and connectivity arrays. Under the assumption that the initial mesh does not contain any flipped element, the condition at line 15 of Alg. 2 ensures that no element flipping occurs during mesh motion. It is avoided by rolling back when it occurs, and starting again with a smaller displacement increment, as shown respectively in Figs. 7c and 7d. Of course, the same

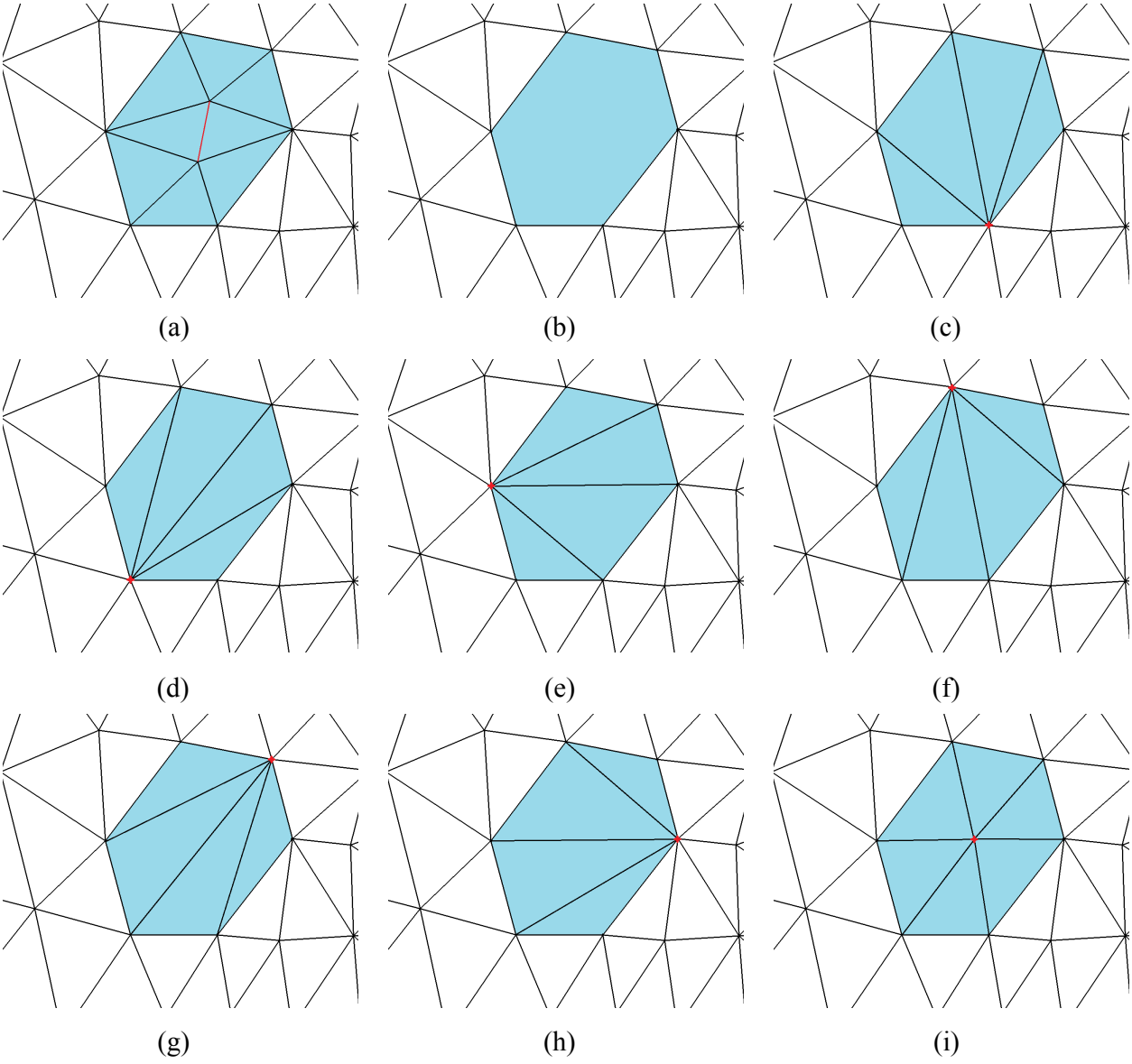


Figure 6: Example of patch at the wide gathered neighboring of an edge: (a) initial patch, (b) external faces extraction, (c-e) candidates.

problem would appear even if small increments are used, hence the mesh has to be adapted at some point. This is triggered at line 17 if at least one of the two following conditions were met during mesh motion: element flipping was detected, or the worst quality of the mesh decreased too importantly. For example, the remeshing operation in Fig. 7e solves the problem in Fig. 7b, as shown in Fig. 7f. The procedure *MeshTransport* at line 20 is a standard operation which consists in transporting all fields stored at nodes and elements from the old mesh to the new mesh after remeshing. This operation includes the correction for multiphase domains detailed in Subsec. 3.2.

Algorithm 2 Mesh motion with element flipping prevention, β is a user-defined tolerance on quality decrease, Q_{old} is a user-defined initial quality, or the quality obtained after the last remeshing, the lines in bold contain key operations that are enhanced in the present work

```
1: function MeshMove(Mesh mesh, NodeField X_obj, Real β, Real Q_old)
2:     Real α, Q_new
3:     NodeField X_old, X_new
4:     Mesh mesh^old
5:
6:     repeat
7:         X_old ← mesh.nodes().coordinates()
8:         X_new ← X_obj
9:         α ← 2
10:        repeat
11:            mesh.nodes().coordinates() ← X_new
12:            X_new ← (X_old + X_new)/2
13:            α ← α/2
14:        until min(mesh.elements().volumes()) > 0
15:        Q_new ← min(mesh.elements().qualities())
16:        if α ≠ 1 or Q_new < γ Q_old then
17:            mesh^old ← mesh
18:            mesh ← MeshAdapt(mesh^old)
19:            MeshTransport(mesh, mesh^old)
20:            Q_old ← Q_new
21:        end if
22:    until α = 1
23:    return mesh
24: end function
```

3.2 Transport of history variables

History variables are mainly present in solid mechanics, where updated Lagrangian formulations require the knowledge of stress and strain states element-wise, throughout the simulation. To describe the issue addressed here, the following element-wise Heaviside functions are defined for each phase i:

$$H_i(x) = \begin{cases} 1 & \text{where } \phi_i(x) \geq 0, \\ 0 & \text{elsewhere.} \end{cases}$$

These H_i functions are computed on the elements of the initial mesh, and will be transported by a basic transport method after remeshing: for each element of the new mesh, the value of any element-wise field is taken from the closest element of the initial mesh (distance is computed from center to center). Though the present method has the aim of conserving internal interfaces, some local optimizations of these interfaces may change their position, for example in the aggressive case illustrated in Fig. 5d. Thus, for some elements,

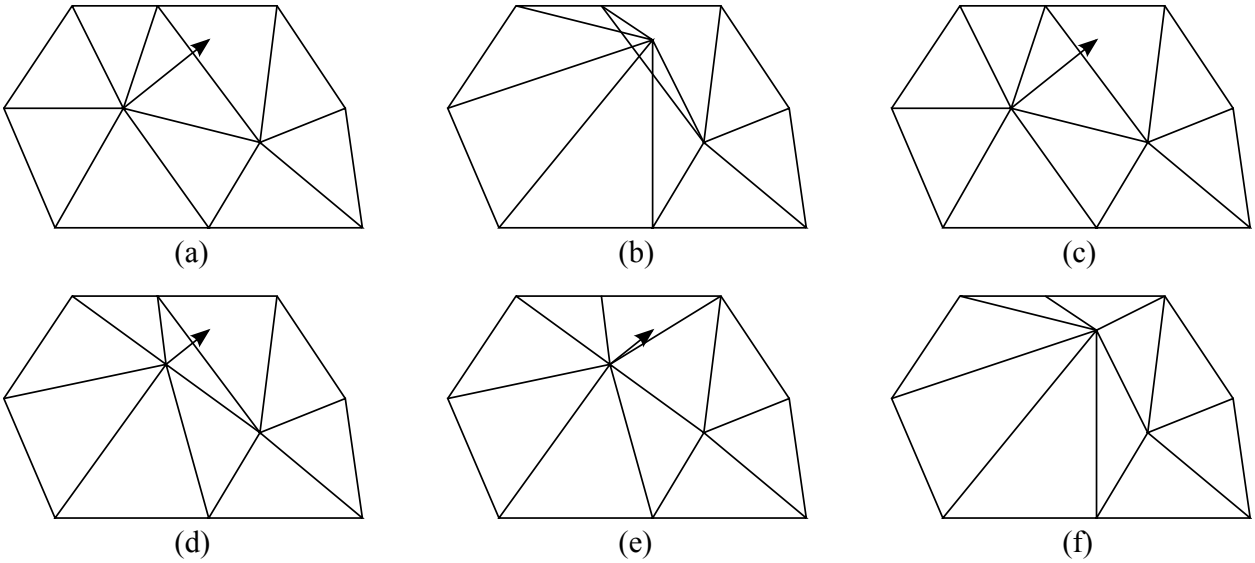


Figure 7: Example of node motion (black arrow) provoking element flipping: (a) initial mesh, (b) mesh after motion, (c) rollback, (d) new try with half motion, (e) remeshing by edge swap, (f) final motion.

recomputing Heaviside functions $\tilde{H}_i$ from the LS functions modified by the remeshing process may yield a result different from H_i. As a consequence, an element may be attributed history variables from a phase it does not belong to, which is wrong from a thermomechanical point of view. This is illustrated in Fig. 8, where an expected blue element of the new mesh has its barycenter (marked by a white star in Fig. 8b) located in an element of the initial green phase (highlighted in white in Fig. 8b), which makes it wrongfully considered as a green element.
To correct this mistake, two strategies seem possible: moving the interfaces in order to agree with the transported history variables, or correct the transport method in order to agree with the interfaces. In the present framework, it seems appropriate to leave interface tracking to the mesh motion and adaptation algorithms, and modify the transport of history variables. To correct this method, for each phase i, history variables of elements where transported H_i is different from recomputed $\tilde{H}_i$ are corrected by projecting the history variable of the closest element which has $\tilde{H}_i = H_i = 1$. To perform this operation efficiently, the barycenters of all elements where $\tilde{H}_i = H_i = 1$ are stored in a specific structure described in [25, 26]. For each element where $\tilde{H}_i$ is different from recomputed H_i, the cost of finding the closest element in this structure is logarithmic on the number of elements in the structure. Consequently, the cost of the present treatment is negligible. An example of corrected result is illustrated in Fig. 8c.

4 Results

In this section, the proposed method is applied to track interfaces in multiphase simulations. Though these test cases are academic, they correspond to real and challenging computational solid and fluid mechanics problems. The efficiency of the algorithms presented in this paper is investigated in terms of mesh quality and computation time. All computations were performed on a 1.2 GHz Intel Xeon processor. Unless otherwise mentioned, the volume conservation parameter is fixed to $Q_y = 0.05$.
In an attempt to establish a panel of applications where the method presented in Secs. 2 and 3 could be competitive and preferable compared to standard approaches listed in Tab. 1, several test cases are proposed. A particular attention will be given to volume conservation and computation time for small and large displacements and deformations. Topological changes such as coalescence of objects will also be addressed. Unless otherwise mentioned, the domain Ω is the 3D box $[0, 1]^3$, an unstructured uniform isotropic

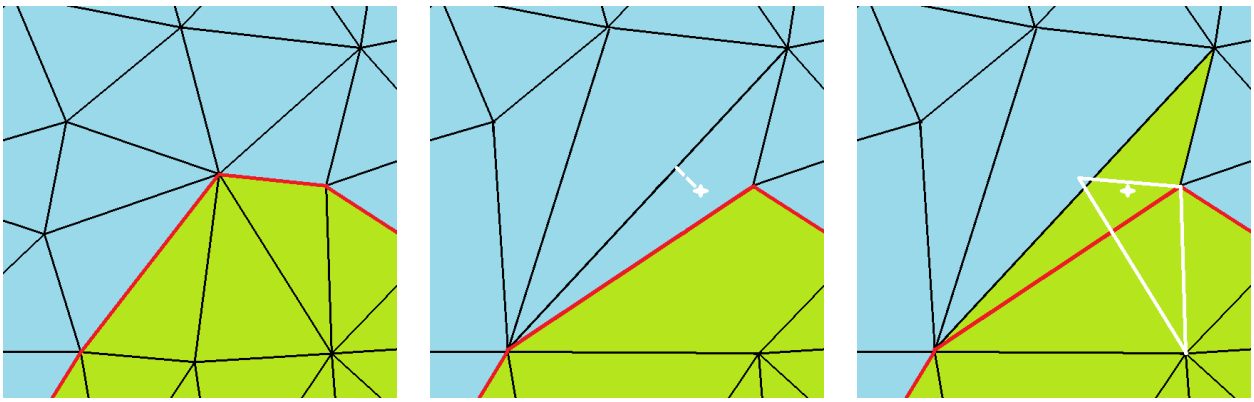


Figure 8: Example of an incorrect transport of history variables: (a) initial mesh, (b) adapted mesh, (c) corrected transport. The two phases are colored in green and blue according to the H_i functions, and the interface is highlighted in red according to the ϕ_i functions.

mesh is used, and the time step is fixed to 0.01, all spatial and temporal data being nondimensionalized.
In the following comparisons, Implicit EULerian (IEUL) refers to an Eulerian LS method where all interfaces are implicitly described by LS functions, the mesh remains fixed during the whole simulation, and convection is operated by solving $\frac{\partial \phi}{\partial t} + v\nabla\phi = 0$, with a first order implicit Euler scheme, a first order Lagrange FE and SUPG stabilization. More advanced and recent stabilization methods, such as [27], could be used, but it is reasonable to suppose that the used stabilization method would not change the conclusions raised here. Implicit LAGrangian (ILAG) refers to a Lagrangian LS method where all interfaces are implicitly described by LS functions, and convection is operated by mesh motion using Alg. 2 with $\beta = 0.5$, but regardless of the interfaces and their conservation. Finally, Fitted LAGrangian with constraint (FLAGc) refers to the method proposed in this paper, which is the same as ILAG, but with explicit interface meshing and enhanced volume conservation.
For the comparison to be fair, the initial mesh is the same for all methods, but in the case of FLAGc an explicit interface is obtained using the interface fitting procedure described in [22]. This procedure consists in introducing the points of intersection between the interfaces described by the LS functions and the FE mesh. As mesh quality is likely to be deteriorated after such operation, it is followed by a mesh adaptation step using Alg. 1. This ensures that the initial interpolation error is the same for all methods. The measures of computation time consider only the resolution of the convection equation in the IEUL case, and the whole cost of Alg. 2 in the ILAG and FLAGc cases, which includes mesh motion, remeshing, and transport of FE fields. For comparisons between these two last methods, the Number of Remeshing Operations (NRO) is also reported.

4.0.1 Large displacements: sphere rotation

Though here no material behavior is defined, this first problem has to be seen as a rigid sphere rotating inside a fluid in turbulent flow. Hence, the displacements and the deformations inside the fluid cannot be modeled by a Lagrangian mesh. However, an Arbitrary-Lagrangian-Eulerian (ALE) could be used to track the rotation of the sphere by mesh motion [17, 18]. The presently proposed method falls in this category. In this test, only the IEUL and the FLAGc methods are compared, and in the case of the FLAGc method mesh motion is applied only where $\phi \geq 0$.
The set-up is directly inspired from [12], where an enhanced LS Method was proposed, which consisted in adding Lagrangian markers on an Eulerian LS, also to reduce LS diffusion during convection. The sphere is

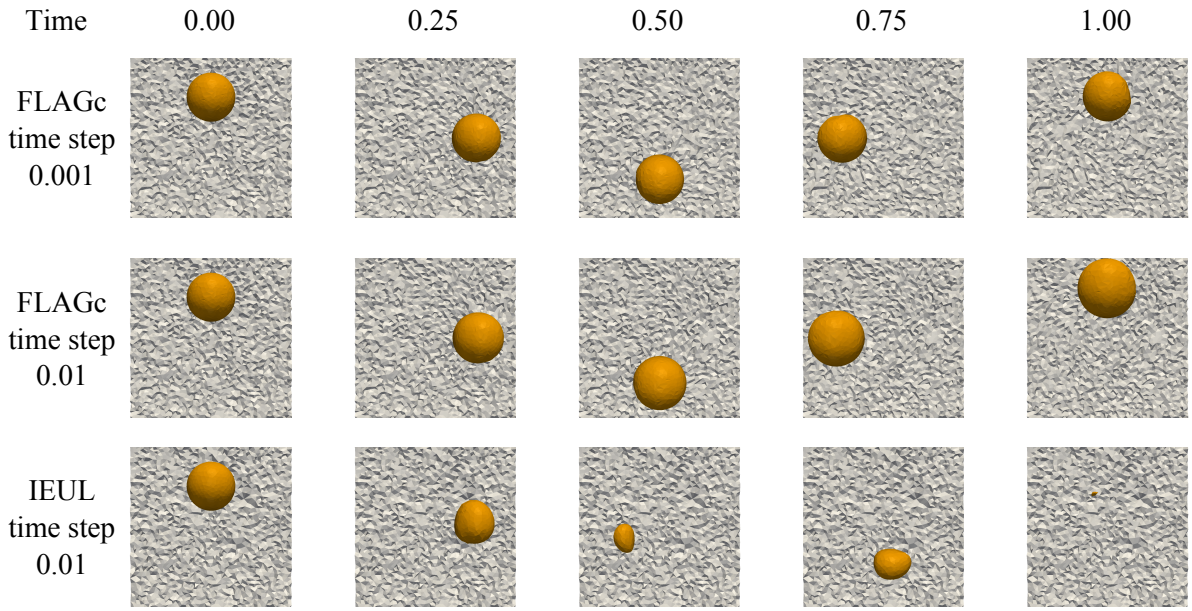
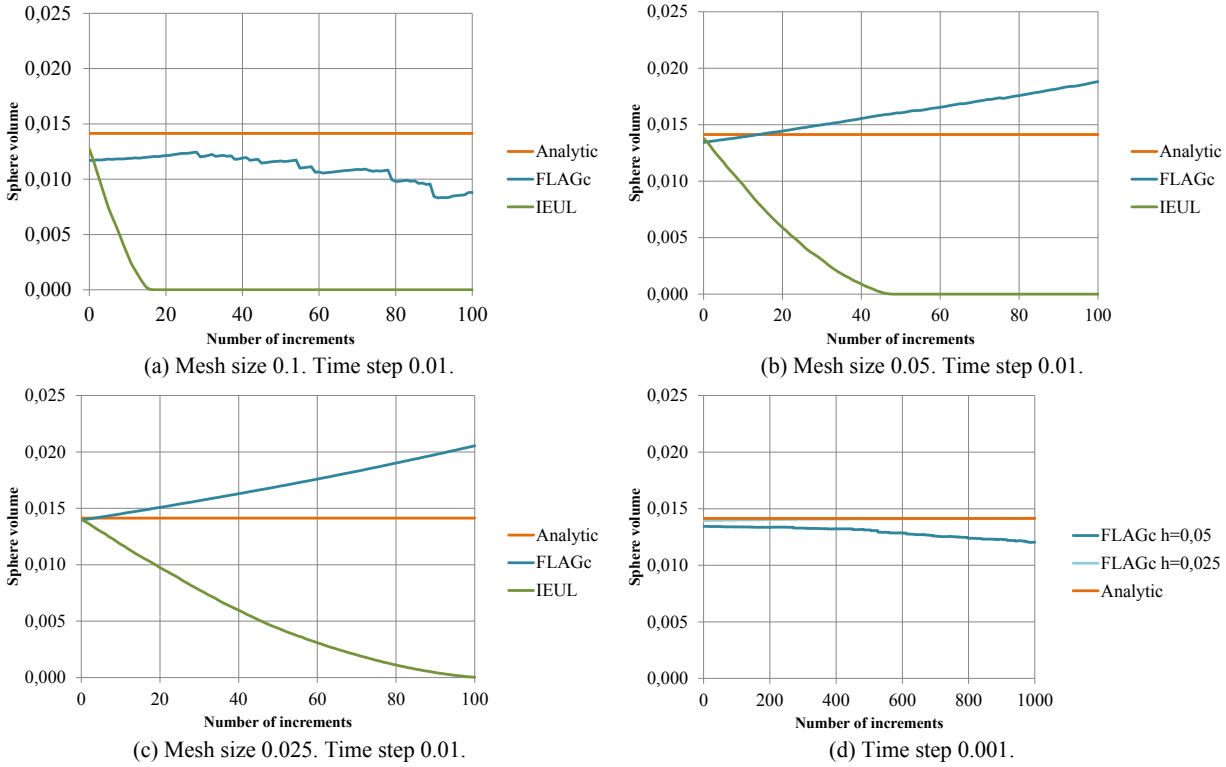


Figure 9: Sphere rotation with a mesh size of 0.025 using the FLAGc method with two different time steps and the IEUL method.

placed at (0.5, 0.75, 0.5), has a radius of 0.15, and its rotation is ruled by the velocity field

$$v(x, y, z) = (-2\pi(0.5 - y), 2\pi(0.5 - x), 0)\,.$$

This case is illustrated in Fig. 9.
It can already be observed in Fig. 9 that diffusion is way more important for the IEUL method, while it is controlled using the FLAGc method. The same tendency appears in Fig. 10a. On the one hand, this poor performance for the IEUL method was expected, as the meshes used here are not as fine as the finite difference grids used in [12] for example, or the structured 2D FE meshes used in [27]. On the other hand, the performance of the FLAGc method could be improved by starting with an explicit mesh of the interface directly built from a proper meshing tool such as [28], and fixing Q_y to 0 since there is no interface deformation. Volume conservation would then be optimal, but it seems obvious that other moving mesh techniques such as [17] or [18] would certainly yield the same result with far less computation time. Indeed, these techniques consist in defining artificial velocity fields for the nodes inside the fluid phase, in order to maintain a good quality as long as possible, and delay the use of computationally expensive algorithms such as Alg. 1. Another possibility for rigid body motions would be embedded mesh techniques [29], where the interfaces are implicit, and carried not by FE fields on the computational mesh, but by separate surface meshes. Nevertheless, these approaches are restricted to rigid body motions or small deformations, while the LS methods (FLAGc, ILAG and IEUL) considered here cover a wider range of applications.
In Figs. 10b and 10c, the IEUL method shows first order convergence, but the results regarding the FLAGc method are surprising, as the volume goes increasing, and so does the error. These startling results are due to the fact that between Fig. 10a and Fig. 10b, there is a major change in the composition of the volume error: the temporal error becomes dominant over the spatial one. This is confirmed by Fig. 10d, which is the same numerical experiment, but with a time step 10 times smaller. Additionally, in Fig. 10a, increments where remeshing was operated can be clearly recognized as the points where sphere volume drops abruptly. Indeed, a larger mesh size reduces the number of calls to the remeshing procedure. On the contrary, in Figs. 10b and 10c, the drops of the sphere volume are reduced both by the fact that the mesh is finer, and that remeshing is more frequent, and are hence less visible.
More precise information is summarized in Tab. 2, which also reports the computation time for each



(a) Mesh size 0.1. Time step 0.01.

(b) Mesh size 0.05. Time step 0.01.

(c) Mesh size 0.025. Time step 0.01.

(d) Time step 0.001.

Figure 10: Evolution of sphere volume during a full rotation using the reference time step and a mesh size of: (a) 0.1, (b) 0.05, (c) 0.025. (d) Evolution of sphere volume during a full rotation using a time step of 0.001.

Method	Mesh size	Time step	Computation time (s)	Error (%)
FLAGc	0.1	0.01	62	15.8
	0.05	0.01	570	10.0
	0.025	0.01	8935	15.3
	0.05	0.001	580	6.5
	0.025	0.001	6909	0.3
IEUL	0.1	0.01	9	100
	0.05	0.01	98	93.6
	0.025	0.01	808	57.8

Table 2: Computation time and relative error on the volume of the sphere after 40% of rotation for the two methods, the three used mesh sizes, and the two used time steps.

simulation. The relative error is computed based on the sphere volume, and the measures are performed after 20 increments (resp. 200 increments for the cases of Fig. 10d). While the FLAGc method is approximately 10 times more costly, it is also more accurate. In particular, it is robust and gives a satisfying solution, even for the coarsest mesh size, while the sphere did not survive 20 increments for the IEUL method. Using a mesh size of 0.1 for the FLAGc method and 0.025 for the IEUL one, the computation time is inverted and the IEUL method is approximately 10 times more costly, for an error that is more than 3 times bigger than the one obtained with the FLAGc method. Note that since in this case, there are no continuum mechanics equations to solve, the computation time for the finer meshes is underestimated. In practice, if one had to use finer meshes to avoid diffusion, the cost of all other operations (Navier-Stokes resolution, output writing, etc.) would increase the advantage of the FLAGc method. Therefore, for coarse meshes, or when the interfaces show multiple scales of variations, the FLAGc method would be significantly more interesting than the IEUL one in preserving all variations during the simulation, including the smallest ones.

4.0.2 Large deformations: sphere stretching

This case is representative of an heterogeneous material with multiple phases that are likely to deform at comparable rates. A sphere of radius 0.15 is placed at the center of the box, and an incompressible vertical stretching is applied by defining the velocity field as

$$v(x,y,z) = \left(-\frac{1}{2}(x-0.5), 1(y-0.5), -\frac{1}{2}(z-0.5)\right),$$

where the factors 1 and $-\frac{1}{2}$ are updated in Lagrangian so that the stretching remains the same as in Eulerian and incompressibility is verified at each increment. With such stretching of the interface, pure moving mesh techniques such as [17] or [18] do not apply anymore, since interfaces need to be remeshed. This case is illustrated in Fig. 11.

The evolution of the volume of the sphere during stretching is presented in Fig. 12 for three different mesh sizes. There is a clear difference of accuracy between the IEUL method, which diffuses at each increment, and the ILAG method, which diffuses only at remeshing. The diffusion at remeshing is then drastically reduced with the FLAGc method (note that remeshing is not triggered at the same increments for the ILAG and FLAGc methods).

Additionally, convergence can be observed on these graphs for the three methods, as diffusion is reduced by mesh refinement. This is confirmed in Tab. 3, where the relative error is computed based on the sphere volume after 100 increments. For the FLAGc and ILAG methods, the gap between the error using the coarsest mesh size and the intermediary one is big due simply to the fact that the first mesh size is too coarse for such stretching. Then, a first order convergence is obtained. The same observation can be made for the IEUL method, with the difference that the sphere is totally diffused using the coarsest mesh, as illustrated in Fig. 12a.

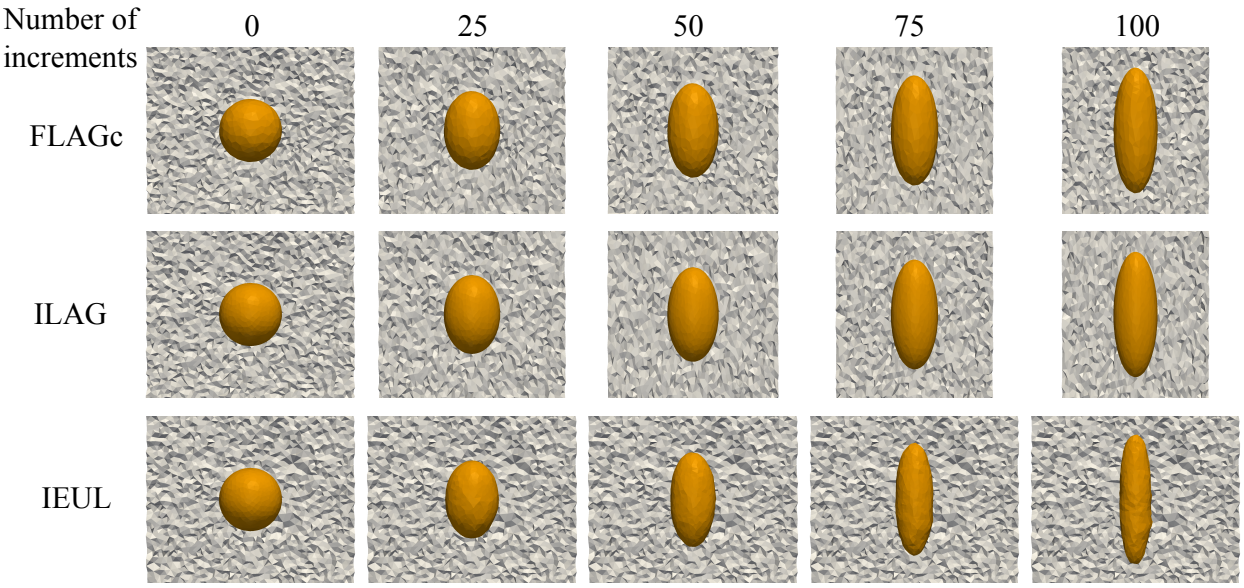
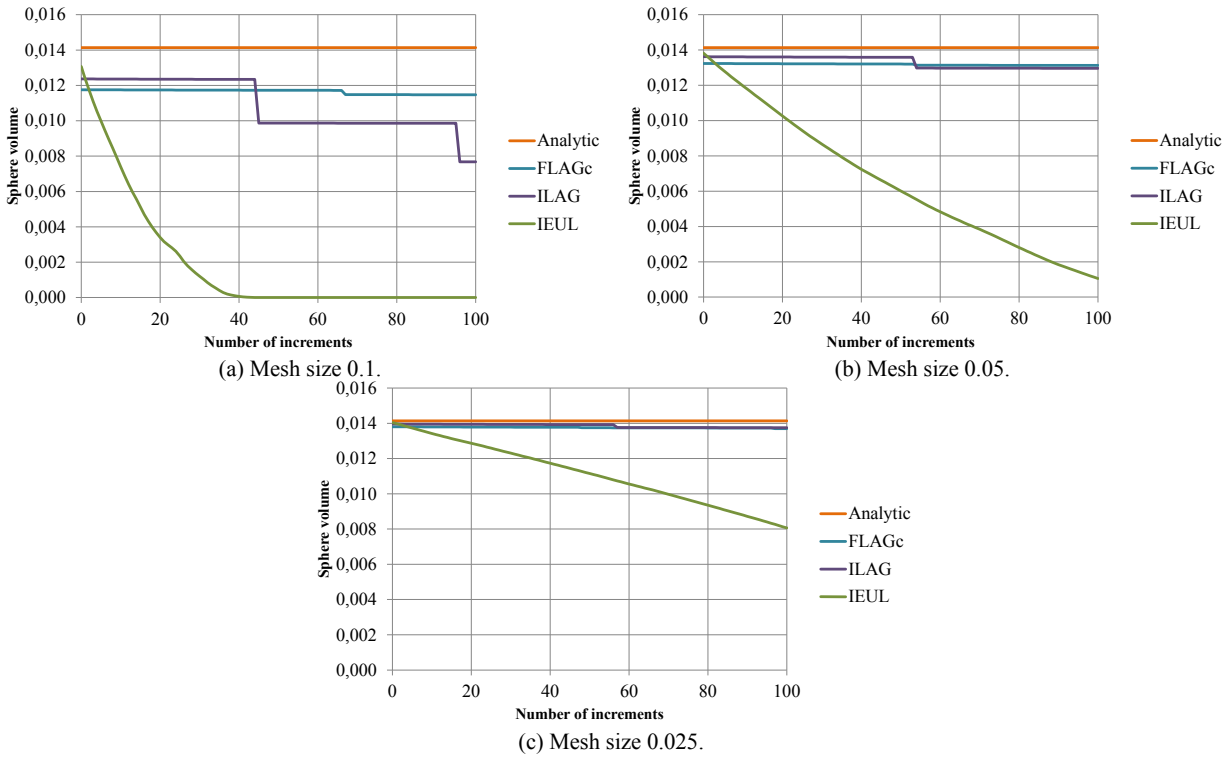


Figure 11: Sphere stretching with a mesh size of 0.025 using the three different methods.



(a) Mesh size 0.1.

(b) Mesh size 0.05.

(c) Mesh size 0.025.

Figure 12: Evolution of the volume of the sphere during its stretching. After 100 increments, the major diameter is doubled, while the minor one is divided by $\sqrt{2}$. The mesh size is: (a) 0.1, (b) 0.05, (c) 0.025.

Method	Mesh size	Computation time (s)	Error (%)	NRO
FLAGc	0.1	6	18.9	2
	0.05	49	7.2	2
	0.025	820	3.1	3
ILAG	0.1	10	45.7	3
	0.05	46	8.3	2
	0.025	422	2.7	2
IEUL	0.1	23	100	
	0.05	252	92.5	
	0.025	2101	43	

Table 3: Computation time and relative error on the volume of the sphere after full stretching for the three methods and the three used mesh sizes.

Finally, measures of computation time in Tab. 3 must be considered carefully since they highly depend on the number of remeshings triggered for the ILAG and FLAGc methods. Though this number is low in the present case, it could be more important depending on the magnitude of the deformation the mesh undergoes at each time step. The main conclusion that can be raised from these measures is that the volume preservation constraint does not induce a significant cost increase, as at equal NRO, the computation time of the FLAGc method is comparable to the one of the ILAG method.

4.0.3 Topological changes: sphere fracture

Computational fracture mechanics is an open research topic and multiple strategies have been proposed to model the initiation and propagation of a crack in homogeneous or heterogeneous materials. This crack may be modeled by explicitly splitting faces and therefore inserting a discontinuity in the mesh [30], or by inserting zero-thickness elements as in a Cohesive Zone Model (CZM) [31]. In the case of small deformations, one may prefer to avoid remeshing by using the X-FEM [6], which can also be enhanced with a CZM. A last and simpler method is the kill-element method, which consists in removing elements along the crack direction.
In [4, 22], the void phase was meshed and cracks were initiated and propagated by dynamically modifying the LS function to the void phase and then meshing it. While standard kill-element and face-splitting methods (including standard CZM methods) have the well-known drawback of generating mesh dependent solutions due to the shape of the removed elements, the proposed method enabled to remove arbitrary shapes. In [4], the behavior of the void phase was defined through penalization, in order to accurately represent a free surface.
This method was already used in 2D in [22] for a homogeneous material containing holes. Here, it is applied in the case of a 3D two-phase material. As stated in the introduction, examples of such materials are dual phase steels and composites. At the microscale, fracture of these materials occurs by debonding between the two-phases, or fragmentation inside one of the two phases. The same micromechanisms can be found in the grain structure of a metal alloy, and are then named respectively inter-granular and intra-granular cracking. To illustrate both situations, the computational set-up of Subsec. 4.0.2 is used, with the difference that at increment 50, a nucleus of thickness twice the mesh size is inserted in the simulation. In practice, this arbitrary instant of fracture should be replaced by appropriate fracture criteria associated to the material behavior of each phase. These criteria can be stress-based, as in [4], or linked to a continuum damage model, as in [22]. As the goal here is to demonstrate the possibility of capturing new interfaces on-the-fly during the simulation, only the FLAGc method is used, with the two configurations of uniform mesh size 0.05 and 0.025. Two sets of computations are performed: one corresponding to an instantaneous failure of the interface between the sphere and the rest of the domain, and one corresponding to an instantaneous failure of the sphere itself. Both situations are pictured in Fig. 13. The fact that the thickness of both nuclei is equal to twice the mesh size is important to prevent its diffusion when volume relaxation occurs. It also guarantees

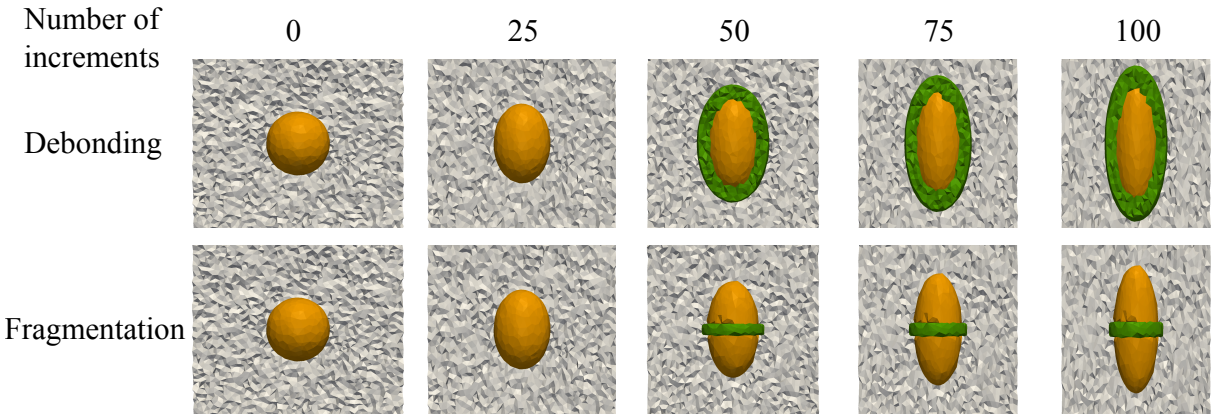


Figure 13: Sphere debonding and fragmentation using the FLAGc method and a mesh size of 0.025. In the debonding case, the nuclei is only partially displayed for sake of representation.

Problem	Mesh size	Computation time (s)	NRO
Debonding	0.05	52	3
	0.025	408	3
Fragmentation	0.05	61	3
	0.025	489	3

Table 4: Total computation time for the sphere debonding and fragmentation cases and the two used mesh sizes.

that the LS function of the nuclei will be accurately described by the mesh. Using mesh refinement, the size of the nucleus could be reduced in order to control the loss of mass induced by its insertion, as done in 2D in [22].

Computation time is reported in Tab. 4. The fact that the cost of the computations with a mesh size of 0.025 is smaller than the time obtained for the pure stretching simulation (see Tab. 3) is due to the randomness of mesh adaptation triggering. This effect is not observed for the simulations with a mesh size of 0.05. The main conclusion that can be drawn from the comparison of these two tables is that computation time does not seem to be significantly influenced by the presence of a new phase. This can be explained by the fact that the used meshes are uniform.

4.0.4 Topological changes: sphere stacking

Applications where topological changes occur do not restrict to crack propagation, where crack branching and merging are difficult to handle from a meshing point of view [30], but easily modeled using an implicit interface. Other examples are recrystallization [5], where grains may appear or disappear during the simulation, or polymer injection, where bubbles may nucleate and coalesce [32, 33]. Generally, in all these applications, explicit interface meshing becomes very difficult and is generally avoided [32, 33].
The present test aims at showing that the present method can handle important topological changes and, more importantly, maintain a good element quality when they occur. It can be seen as the opposite of the fracture calculations where new interfaces appeared during the simulation, as here portions of the initial interfaces are going to disappear.
The domain contains three spheres of identical radius 0.15 which are placed at $(0.3, 0.2, 0.5)$, $(0.7, 0.2, 0.5)$ and $(0.5, 0.5, 0.5)$, and the lower face of the box is of the same material as the spheres (ϕ is zero on this face). In this test, a Stokes problem is solved using a mixed P1+/P1 FE method at each time step, with homogeneous Dirichlet boundary conditions for the velocity, and homogeneous Neumann boundary conditions for

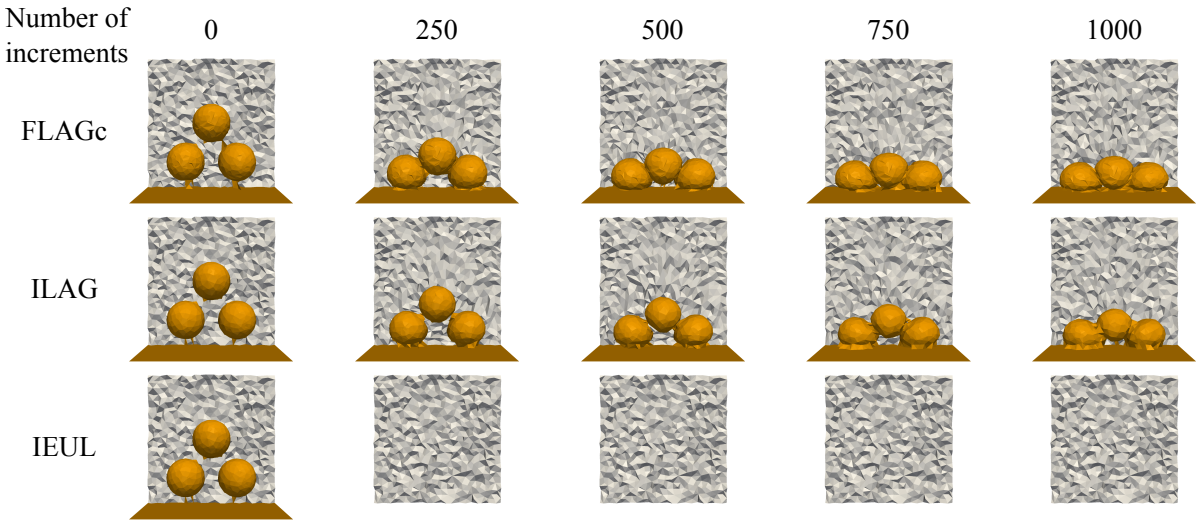


Figure 14: Spheres stacking with a mesh size of 0.05 using the three different methods.

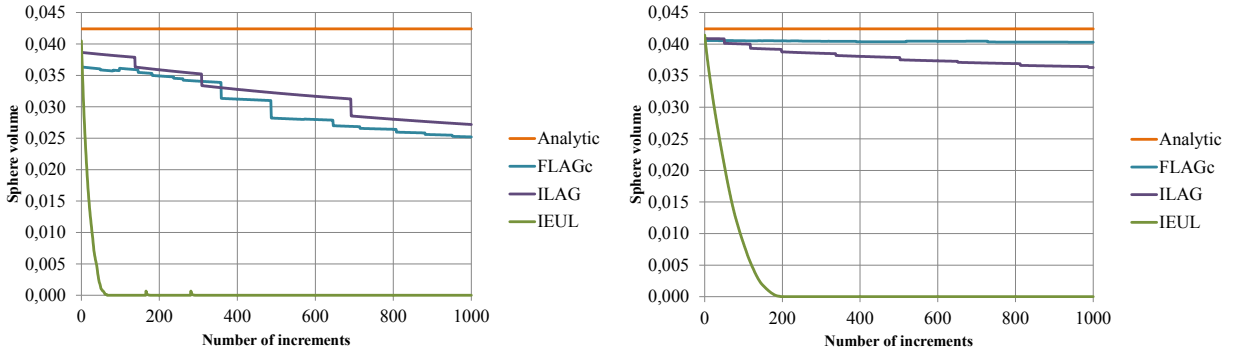
the pressure. The parameters of this problem are the viscosity and the gravity, which are set respectively to 1000 and $(0, -1000, 0)$ inside the spheres, and 10 and $(0, -10, 0)$ in the rest of the domain. As illustrated in Fig. 14, the spheres and the plane are expected to coalesce altogether (as in polymer injection for example). Due to the important computational cost of the Stokes solver (not reported here), the configuration with a mesh size of 0.025 is not considered for this test.

For the coarsest mesh, the results in Fig. 15a show a poor performance of the FLAGc method. Maintaining a body-fitted mesh raises complicate issues when the spheres start to touch, consequently the NRO is more important than with the ILAG method (see Tab. 5), and volume relaxation is omnipresent. Then, when the spheres have coalesced (after ≈ 500 increments), mesh quality seems to have been restored, as volume is better conserved. The main difference between the FLAGc method and the ILAG one is that the ILAG method diffuses approximately the same amount of volume whatever the encountered deformation or topological change, while the FLAGc method controls this diffusion. In fact, if the FLAGc method was used with an implicit interface, one could assume that remeshing would be less frequent, and a better performance could be recovered. Though such experiment is not performed here, the results in Fig. 15b with a finer mesh prove that the FLAGc method controls indeed volume diffusion, while the ILAG method keeps releasing the same amount of volume at each remeshing.

Regarding the IEUL method, the conclusions are the same as in the preceding sections: convergence is of first order, but the difference with the other methods is huge. An interesting point is that the IEUL method is way more costly in this case, as reported in Tab. 5. This is due to the fact that the cost of the ILAG and FLAGc methods is mainly due to Alg. 1. Decreasing the time step has no reason to influence importantly the NRO, and hence to change significantly computation time. However, the IEUL method relies on the resolution of a convection equation at every time step. Though here a preconditioned conjugate residual iterative method is used and the number of iterations decreases with the time step, there is still an irreducible cost at each resolution.

4.0.5 Robustness investigation

For all preceding numerical experiments, a particular attention was given to volume conservation and computation time. Since the method proposed in this paper is based on a mesh adaptation algorithm, it is also important to verify that this algorithm is efficient. In that purpose, element quality, as defined in Eq. 1, is computed at each increment and integrated over time for all tests performed with the FLAGc method. This quality distribution is expected to be concentrated around two peaks corresponding respectively to the



(a) Mesh size 0.1. (b) Mesh size 0.05.

Figure 15: Evolution of the volume of the spheres during stacking for a mesh size of: (a) 0.1, (b) 0.05.

Method	Mesh size	Computation time (s)	Error (%)	NRO
FLAGc	0.1	103	40.6	16
	0.05	889	5.0	22
ILAG	0.1	62	35.9	4
	0.05	600	14.4	9
IEUL	0.1	251	100	
	0.05	2940	100	

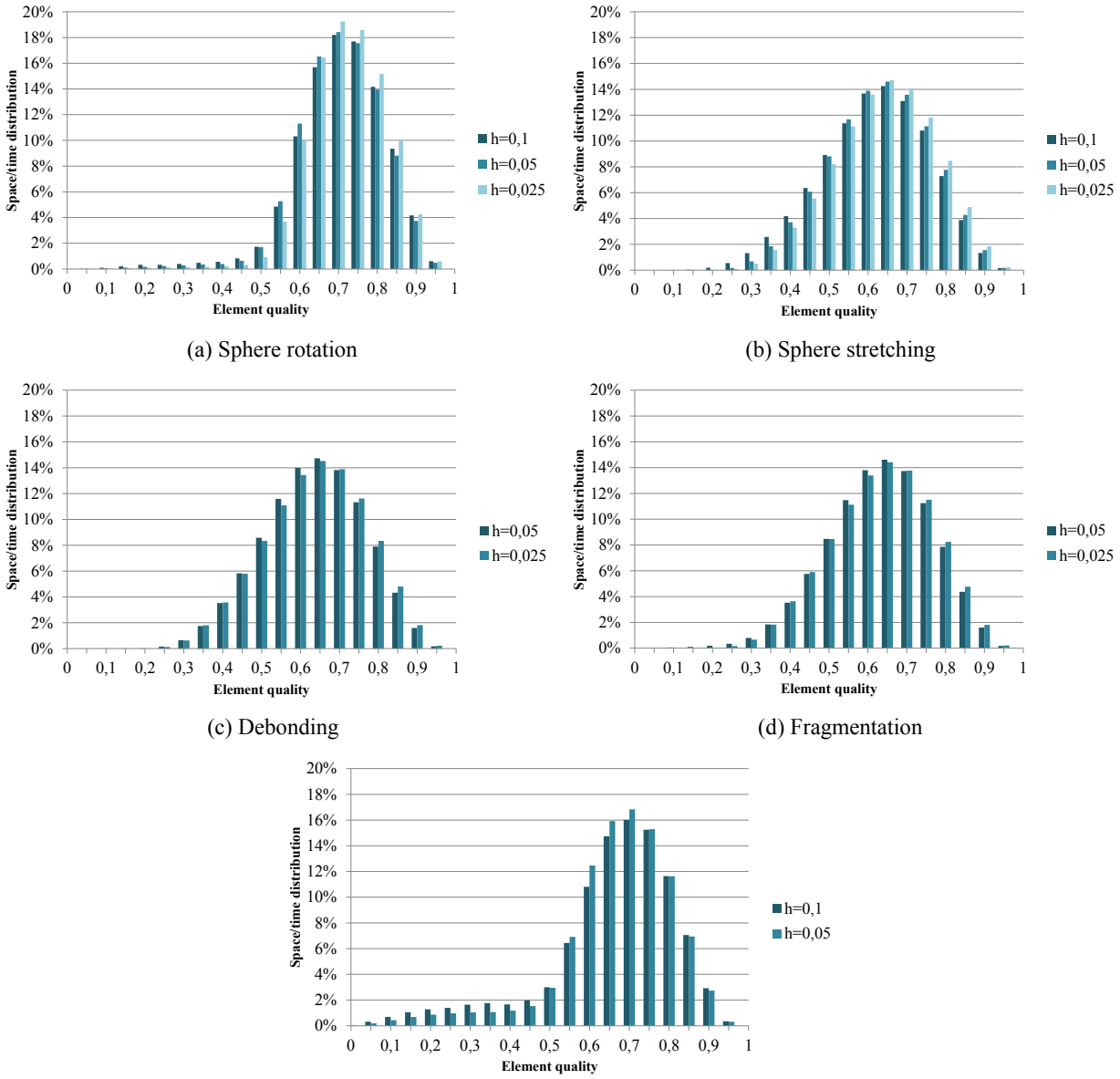
Table 5: Sphere stacking test: computation time and relative error on the volume of the spheres after 1000 increments for the three methods and the two used mesh sizes.

elements of the domain far from any interface, and the elements having their quality deteriorated by the presence of a neighboring interface. As most histograms presented in Fig. 16 feature only one peak, a first qualitative conclusion is that, in most cases, element quality is not deteriorated significantly by the volume conservation constraint.

First, in the sphere rotation case in Fig. 16a, it can be seen that element quality is efficiently restored after the passing of the sphere, as element quality is concentrated around 0.7 for all used mesh sizes. Regarding the sphere stretching case in Fig. 16b, the histogram is slightly shifted to the left because the whole domain was stretched and remeshing was not operated frequently enough to maintain higher quality, hence the quality of some elements was allowed to decrease. The fact that there are very few ill-shaped elements is nevertheless satisfying and remeshing based on a quality decrease criterion seems appropriate.

In the two fracture cases, major changes occur regarding the morphology of the interfaces, especially when new edges appear due to the fragmentation of the sphere in two parts. Comparing Figs. 16c and 16d with the sphere stretching histogram in Fig. 16b, no major difference can be seen, whichever the used mesh size. This shows the robustness of the mesh adaptation method, as all poor quality elements produced by the fitting procedure are removed. Indeed, there are no elements in the first bar between 0 and 0.05, which means that whenever remeshing was activated, all elements with quality below 0.05 were improved, probably using volume relaxation.

Finally, element quality distribution for the spheres stacking case is the only distribution that clearly exhibits two distinct concentration points: one around 0.3 and one around 0.7 (see Fig. 16e). This suggests that a whole part of the domain was represented with elements of inferior quality, and that these elements were kept in the mesh for a long time. This is most likely due to the small space between the spheres that progressively narrowed down and then disappeared, while a prescribed nucleus thickness prevented such situation in the fracture calculations. The numerous contact areas between the spheres and the plane and between the spheres themselves could also explain this decrease of mesh quality.



(a) Sphere rotation

(b) Sphere stretching

(c) Debonding

(d) Fragmentation

(e) Sphere stacking

Figure 16: Distribution of element quality integrated on all increments for: (a) the sphere rotation case, (b) the sphere stretching case, (c) the sphere debonding case, (d) the sphere fragmentation case, (e) the spheres stacking case.

5 Conclusion

Coupling a mesh motion algorithm with a quality and element flipping criterion (see Sec. 3), it was proved in this paper that remeshing can be delayed enough so that a Lagrangian Level-Set method becomes more than interesting compared to an Eulerian Level-Set method, even when large deformations and topological changes occur (see Subsecs. 4.0.2 and 4.0.4). The main defect of the Lagrangian method was the tendency of significantly diffusing the interfaces and the volume at each remeshing operation. The new remeshing method with volume conservation constraint presented in this paper (see Sec. 2) was then proved to control and reduce this diffusion, and promising results were obtained for large displacements (see Subsec. 4.0.1). Compared to the two first approaches, this new method also added explicit interface meshing, which usually raises difficulties when large deformations and topological changes occur. Thanks to the intermediary use of Level-Set functions, the method was proved to be able to handle such events without a significant loss of accuracy or increase of computation time.
Consequently, this adaptive Level-Set Method with enhanced volume conservation is a robust tool that can be used to track interfaces in simulations where all type of events occur (large displacements, large deformations, topological changes). This robustness also allows to use coarser meshes, or to improve the conservation of all scales of variations of the interfaces, including the smallest ones.
Coupled with an interface fitting procedure, this technique was used to generate body-fitted meshes from Level-Set functions, enabling fracture mechanics applications where new interfaces are captured on-the-fly during the simulation. Regarding this last point, a point of improvement is the small loss of volume induced by the fact that cracks are approximated by volumetric voids. Future work includes the dynamic meshing of actual cracks, and the handling of the contact issues that may occur during the simulation. Generalization of this fitting procedure to the treatment of multiphase junctions would also be interesting for application of the present method to polycrystals and other massively multiphase materials.

Acknowledgments

The supports of the Carnot M.I.N.E.S institute and the French Agence Nationale de la Recherche (ANR) through the COMINSIDE project are gratefully acknowledged.

References

[1] Alauzet F, Frey P, George P, Mohammadi B. 3D transient fixed point mesh adaptation for time-dependent problems: Application to CFD simulations. Journal of Computational Physics Mar 2007; **222**(2):592–623, doi:10.1016/j.jcp.2006.08.012.

[2] Quan DL, Toulorge T, Marchandise E, Remacle JF, Bricteux G. Anisotropic mesh adaptation with optimal convergence for finite elements using embedded geometries. Computer Methods in Applied Mechanics and Engineering Jan 2014; **268**:65–81, doi:10.1016/j.cma.2013.09.007.

[3] Hachem E, Feghali S, Coupez T, Codina R. A three-field stabilized finite element method for fluid-structure interaction: elastic solid and rigid body limit. International Journal for Numerical Methods in Engineering nov 2015; **104**(7):566–584, doi:10.1002/nme.4972.

[4] Roux E, Shakoor M, Bernacki M, Bouchard PO. A new finite element approach for modelling ductile damage void nucleation and growth—analysis of loading path effect on damage mechanisms. Modelling and Simulation in Materials Science and Engineering Oct 2014; **22**(7):075 001, doi:10.1088/0965-0393/22/7/075001.

[5] Bernacki M, Resk H, Coupez T, Logé RE. Finite element model of primary recrystallization in polycrystalline aggregates using a level set framework. Modelling and Simulation in Materials Science and Engineering Sep 2009; **17**(6):064 006, doi:10.1088/0965-0393/17/6/064006.

[6] Sukumar N, Chopp D, Moës N, Belytschko T. Modeling holes and inclusions by level sets in the extended finite-element method. Computer Methods in Applied Mechanics and Engineering Sep 2001; **190**(46-47):6183–6200, doi:10.1016/S0045-7825(01)00215-8.

[7] Osher S, Sethian JA. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. Journal of Computational Physics Nov 1988; **79**(1):12–49, doi:10.1016/0021-9991(88)90002-2.

[8] Boettinger WJ, Warren JA, Beckermann C, Karma A. Phase-field simulation of solidification. Annual Review of Materials Research Aug 2002; **32**(1):163–194, doi:10.1146/annurev.matsci.32.101901.155803.

[9] Chen LQ. Phase-field models for microstructure evolution. Annual Review of Materials Research Aug 2002; **32**(1):113–140, doi:10.1146/annurev.matsci.32.112001.132041.

[10] Hirt C, Nichols B. Volume of fluid (VOF) method for the dynamics of free boundaries. Journal of Computational Physics Jan 1981; **39**(1):201–225, doi:10.1016/0021-9991(81)90145-5.

[11] Sussman M. A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles. Journal of Computational Physics may 2003; **187**(1):110–136, doi:10.1016/S0021-9991(03)00087-1.

[12] Enright DP. Use of the Particle Level Set Method for enhanced resolution of free surface flows. PhD Thesis, Stanford University 2002.

[13] Hieber SE, Koumoutsakos P. A Lagrangian particle level set method. Journal of Computational Physics nov 2005; **210**(1):342–367, doi:10.1016/j.jcp.2005.04.013.

[14] Zheng X, Lowengrub J, Anderson A, Cristini V. Adaptive unstructured volume remeshing – II: Application to two- and three-dimensional level-set simulations of multiphase flow. Journal of Computational Physics sep 2005; **208**(2):626–650, doi:10.1016/j.jcp.2005.02.024.

[15] Sussman M. A parallelized, adaptive algorithm for multiphase flows in general geometries. Computers & Structures Feb 2005; **83**(6-7):435–444, doi:10.1016/j.compstruc.2004.06.006.

[16] Hallberg H. A modified level set approach to 2D modeling of dynamic recrystallization. Modelling and Simulation in Materials Science and Engineering dec 2013; **21**(8):085 012, doi:10.1088/0965-0393/21/8/085012.

[17] Compère G, Remacle JF, Marchandise E. Proceedings of the 17th International Meshing Roundtable. Springer Berlin Heidelberg: Berlin, Heidelberg, 2008, doi:10.1007/978-3-540-87921-3.

[18] Alauzet F. A changing-topology moving mesh technique for large displacements. Engineering with Computers Oct 2013; **30**(2):175–200, doi:10.1007/s00366-013-0340-z.

[19] Dobrzynski C, Frey P. Anisotropic Delaunay Mesh Adaptation for Unsteady Simulations. Proceedings of the 17th International Meshing Roundtable, Garimella RV (ed.). Springer Berlin Heidelberg: Berlin, Heidelberg, 2008; 177–194, doi:10.1007/978-3-540-87921-3.

[20] Klingner BM, Shewchuk JR. Aggressive Tetrahedral Mesh Improvement. Proceedings of the 16th International Meshing Roundtable. chap. Session 1A, Springer Berlin Heidelberg, 2008; 3–23, doi:10.1007/978-3-540-75103-8_1.

[21] Gruau C, Coupez T. 3D tetrahedral, unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric. Computer Methods in Applied Mechanics and Engineering Nov 2005; **194**(48-49):4951–4976, doi:10.1016/j.cma.2004.11.020.

[22] Shakoor M, Bernacki M, Bouchard PO. A new body-fitted immersed volume method for the modeling of ductile fracture at the microscale: Analysis of void clusters and stress state effects on coalescence. Engineering Fracture Mechanics oct 2015; **147**:398–417, doi:10.1016/j.engfracmech.2015.06.057.

[23] Gruau C. Génération de métriques pour adaptation anisotrope de maillages, applications à la mise en forme des matériaux. Phd, Ecole Nationale Supérieure des Mines de Paris 2004.

[24] ISO. Iso/iec 9899:1999. Technical Report, International Organization for Standardization, Geneva, Switzerland 1999.

[25] Shakoor M, Scholtes B, Bouchard PO, Bernacki M. An efficient and parallel level set reinitialization method – Application to micromechanics and microstructural evolutions. Applied Mathematical Modelling dec 2015; **39**(23-24):7291–7302, doi:10.1016/j.apm.2015.03.014.

[26] Bentley JL. Multidimensional binary search trees used for associative searching. Communications of the ACM Sep 1975; **18**(9):509–517, doi:10.1145/361002.361007.

[27] Di Pietro DA, Lo Forte S, Parolini N. Mass preserving finite element implementations of the level set method. Applied Numerical Mathematics Sep 2006; **56**(9):1179–1195, doi:10.1016/j.apnum.2006.03.003.

[28] Geuzaine C, Remacle JF. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. International Journal for Numerical Methods in Engineering Sep 2009; **79**(11):1309–1331, doi:10.1002/nme.2579.

[29] Löhner R, Cebral JR, Camelli FE, Appanaboyina S, Baum JD, Mestreau EL, Soto OA. Adaptive embedded and immersed unstructured grid techniques. Computer Methods in Applied Mechanics and Engineering Apr 2008; **197**(25-28):2173–2197, doi:10.1016/j.cma.2007.09.010.

[30] Bouchard PO, Bay F, Chastel Y, Tovena I. Crack propagation modelling using an advanced remeshing technique. Computer Methods in Applied Mechanics and Engineering Sep 2000; **189**(3):723–742, doi:10.1016/S0045-7825(99)00324-2.

[31] Ortiz M, Pandolfi A. Finite-deformation irreversible cohesive elements for three-dimensional crack-propagation analysis. International Journal for Numerical Methods in Engineering Mar 1999; **44**(9):1267–1282, doi:10.1002/(SICI)1097-0207(19990330)44:9¡1267::AID-NME486¿3.0.CO;2-7.

[32] Bruchon J, Coupez T. A numerical strategy for the direct 3D simulation of the expansion of bubbles into a molten polymer during a foaming process. International Journal for Numerical Methods in Fluids Jul 2008; **57**(8):977–1003, doi:10.1002/fld.1660.

[33] Silva L, Valette R, Laure P, Coupez T. A new three-dimensional mixed finite element for direct numerical simulation of compressible viscoelastic flows with moving free surfaces. International Journal of Material Forming feb 2011; **5**(1):55–72, doi:10.1007/s12289-011-1030-2.