

A Delaunay refinement algorithm for the particle finite element method applied to free surface flows

Thomas Leyssens¹ | Michel Henry | Jonathan Lambrechts | Jean-François Remacle

Institute of Mechanics, Materials and Civil Engineering, Université Catholique de Louvain, Louvain-la-Neuve, Belgium

Correspondence

Thomas Leyssens, Institute of Mechanics, Materials and Civil Engineering, Université catholique de Louvain, Louvain-la-Neuve, Belgium.

Email: thomas.leyssens@uclouvain.be

Abstract

This article proposes two contributions to the calculation of free-surface flows using the particle finite element method (PFEM). The PFEM is based upon a Lagrangian approach: a set of particles defines the fluid and each particle is associated with a velocity vector. Then, unlike a pure Lagrangian method, all the particles are connected by a triangular mesh. The difficulty lies in locating the free surface from this mesh. It is a matter of deciding which of the elements in the mesh are part of the fluid domain, and to define a boundary—the free surface. Then, the incompressible Navier–Stokes equations are solved on the fluid domain and the particle position is updated using the velocity vector from the finite element solver. *Our first contribution is to propose an approach to adapt the mesh with theoretical guarantees of quality:* the mesh generation community has acquired a lot of experience and understanding about mesh adaptation approaches with guarantees of quality on the final mesh. The approach we use here is based on a *Delaunay refinement strategy*, allowing to insert and remove nodes while gradually improving mesh quality. We show that what is proposed allows to create stable and smooth free surface geometries. One characteristic of the PFEM is that only one fluid domain is modeled, even if its shape and topology change. It is nevertheless necessary to apply conditions on the domain boundaries. When a boundary is a free surface, the flow on the other side is not modeled, it is represented by an external pressure. On the external free surface boundary, atmospheric pressure can be imposed. Nevertheless, there may be internal free surfaces: the fluid can fully encapsulate cavities to form bubbles. The pressure required to maintain the volume of those bubbles is a priori unknown. For example, the atmospheric pressure would not be sufficient to prevent the bubbles from deflating and eventually disappearing. *Our second contribution is to propose a multi-point constraint approach to enforce global incompressibility of those empty bubbles.* We show that this approach allows to accurately model bubbly flows that involve two fluids with large density differences, for instance water and air, while only modeling the heavier fluid.

KEYWORDS

finite element methods, mesh generation, particle methods

1 | INTRODUCTION

A considerable number of flows involve *free surfaces* that may evolve in complex geometries. In many cases, these flows are highly transient. A wide range of applications of free surface flows can be found in various fields of engineering: hydraulics (flows over weirs, around bridge piers, dam breaks, wetting and drying), naval engineering (flows around ship hulls), additive manufacturing (a melt pool is produced as a result of laser radiation on the surface of the powder), combustion and so forth.

The main difficulty in numerically computing free surface flows is directly linked to modeling the geometry of the free surface. Indeed, the free surface evolves in time, sometimes rapidly, changing its scales, its geometry and its topology. The computational fluid dynamics community has long been interested in modeling strongly deforming free surface flows. Two classes of methods have emerged for solving them: the evolving free surface can be numerically *tracked* or *captured*.

Interface capturing methods represent the interface on a fixed mesh and let it evolve by means of an advection equation. By contrast, in interface tracking approaches, the mesh is conforming to the interface at all times. This is of course a challenge: consider, for example, a seemingly simple fluid mechanics problem of a container partially filled with water. When this container is shaken, the dynamics of the water become relatively complex: interactions with the container walls, mixing of the water with the surrounding air, droplets splashing and so forth.

The particle finite element method, or PFEM, is a tracking approach designed to tackle this complexity. Initially developed for fluid flows,¹ the PFEM has been successfully applied to complex industrial applications such as fluid structure interactions,² landslides,³ or even solid mechanics problems such as metal cutting.⁴

The PFEM starts from a Lagrangian point of view: the fluid domain is represented by a set of particles that follow the motion of the fluid. These particles carry all the relevant fields (velocity, pressure, temperature, etc.), but they have no prescribed mass. Lagrangian methods are typically separated into two different categories: meshless methods and mesh-based methods. The smoothed particle hydrodynamics (SPH)⁵ and material point method (MPM)⁶ belong to the meshless group. These methods have the advantage of not requiring any remeshing. However, the absence of a mesh leads to additional difficulties. Computing gradients of fields can be tricky and properly imposing boundary conditions is challenging. In mesh-based Lagrangian methods, particles are connected through a mesh that is used to solve the equations describing the evolution of the continuum (e.g., with finite elements). If the connectivity of the mesh is fixed, large deformations of the front are not always permitted: arbitrary Lagrangian–Eulerian (ALE)⁷ methods are extremely accurate methods with great properties of conservation but they are restricted to small deformations of the free surface and have difficulties with topological changes of the fluid domain.

Though the PFEM^{8,9} is mesh-based, this method actually assumes that computing the Delaunay triangulation/tetrahedrization of a large set of points is a task that can be carried out in an efficient manner. In highly transient flows, changes in the domain geometry may happen very often. Since the PFEM relies on re-meshing to detect topological changes, it is necessary to compute the triangulation at each time step. Recomputing the triangulation at every time step is cheaper than solving the Navier–Stokes equations.¹⁰ One remaining difficulty is to determine the boundary of the fluid domain—the free surface. This challenge is linked to a well-known problem of computational geometry: surface reconstruction from point clouds. We need an indicator that defines the boundary of the domain. A Delaunay triangulation covers exactly the convex hull of the set of points: this is the simplest indicator. More complex indicators based on α -shapes have been proposed in the PFEM. Using α -shapes roughly means eliminating poor-quality elements from the triangulation (too large or too distorted). If we consider a set of points that is “ordered” (like in a finite element mesh), the α -shape approach works pretty well. Let’s now imagine that our set of points is initially well distributed. After a few time steps, the points will evolve with the fluid velocity into a more disordered state, with some regions highly concentrated in particles, and others suffering from low particles density. This leads to a significant degradation in mesh quality—even inside the fluid domain. Figure 1 illustrates this phenomenon.

The fact that the mesh is of poor quality has two consequences. First, as with any other finite element simulation, solving the equations on a quality mesh is important for obtaining a good accuracy. Second, even more crucially, in the context of the PFEM, the detection of the fluid domain actually depends on mesh quality. Hence, degrading the mesh has the well-known consequence of causing abrupt, unphysical variations in the shape of the fluid domain. Solutions have been proposed¹¹ in the literature for adapting the mesh during simulation: adding nodes when the local density is too low, or removing nodes when they are too close.

In this article, our first contribution is to *propose an approach to adapt the mesh with theoretical guarantees of quality*: the mesh generation community has acquired a lot of experience and understanding about mesh adaptation approaches with quality guarantees on the final mesh. The approach we use here is based on a *Delaunay refinement*

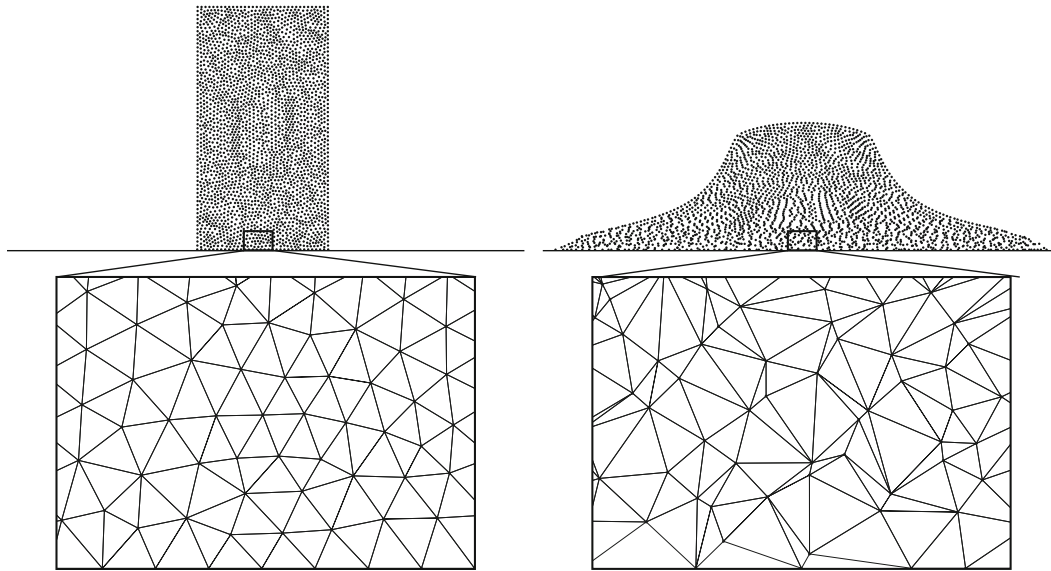


FIGURE 1 A typical PFEM simulation of a water column collapsing. After a number of time steps, the initially well-distributed particles move according to the velocity field, resulting in a disordered state. Without any adaptation, the mesh quality suffers greatly from this motion of the particles, eventually leading to poor quality simulations.

strategy, more specifically Chew's first algorithm,¹² allowing to insert and remove nodes while gradually improving mesh quality. The philosophy consists of enforcing a high quality Delaunay mesh at all times, and refining it based on a user-defined size field. We base this on the works of Chew¹² and Shewchuck¹³ who showed that node insertion at the circumcenter of poorly shaped or sized elements will only create elements of better quality. Other approaches for local mesh refinement have been proposed recently, for instance in Reference 14, who refine through edge-splitting. The main improvement we suggest is to guarantee mesh quality improvement while ensuring the size field criterion is met throughout the mesh. Instead of performing node insertion through edge-splitting or, arguably worse, at the center of mass of bad quality elements, we show that node insertion at the circumcenter will guarantee mesh quality improvement. Moreover, as will be presented in the paper, the criterion used for node insertion corresponds exactly to the one used to define the fluid domain, the α -shape. The proposed method also has the advantages of being easy to implement, efficient, and can be used with any user-defined size field. This greatly improves the robustness of the method, as it removes the arbitrariness in the detection of the free surface. Moreover, the use of a size field is very useful since it allows greater accuracy in regions of interest through mesh refinement, and saves computational costs thanks to coarsening.

One characteristic of the PFEM is that only one fluid domain is modeled, even if its shape and topology change. It is nevertheless necessary to apply conditions on the domain boundaries. When a boundary is a free surface, the flow on the other side is not modeled, it is represented by an external pressure. On the external free surface boundary, atmospheric pressure can be imposed. Nevertheless, there may be internal free surfaces: the fluid can fully encapsulate cavities to form bubbles. The pressure required to maintain the volume of those bubbles is a priori unknown. For example, the atmospheric pressure would not be sufficient to prevent the bubbles from deflating and eventually disappearing. *Our second contribution is to propose a multi-point constraint approach to enforce global incompressibility of those empty bubbles.* We show that this approach allows to accurately model bubbly flows that involve two fluids with large density differences, for instance water and air, while only modeling the heavier fluid.

The structure of this article goes as follows. Our methodology is presented in Section 2, in which the overall algorithm of the particle finite element method is briefly presented first in Section 2.1. The physical concepts of free surface flows are described in Section 2.2, followed by the domain detection algorithm in Section 2.3. Next, Section 2.4 describes our approach to adapt and refine the mesh. We then focus on different boundary conditions in Section 2.5, one of which is the incompressibility condition of empty bubbles. Our solver is finally verified on a few benchmark tests in Section 3, and a few additional results are shown in Section 4.

2 | METHODOLOGY

The need for a method that seamlessly incorporates strong topological changes during a simulation is what led to the birth of the particle finite element method. Whenever there exists a discontinuity in the parameters of a model (Young's modulus, latent heat, density, viscosity, etc.), the solution of the problem or its derivatives may also present discontinuities that are called interfaces or fronts. These fronts are mobile, they can merge, split, nucleate or disappear.

An accurate method that captures these moving fronts requires two main ingredients. The first is a solver that numerically solves the physical equations of the model. The second is an approach to track the fronts, in our case the free surface. These two ingredients are presented in this section, after briefly describing the overall PFEM algorithm.

2.1 | The particle finite element method

As its name suggests, the PFEM follows the evolution of a fluid by tracking the motion of particles. Therefore, an initial particles distribution is generated. This can be obtained from a mesh that represents the initial state of the fluid. From this mesh, the nodes are taken as the particles of the simulation.

Next, a geometry that defines the full computational domain is created. Geometrical points, defined as control nodes, are needed for the representation of the domain. These control nodes are then connected to create the solid boundaries of the geometrical domain.

Having finished the preliminary steps, the main loop starts. The steps refer to those depicted in Figure 2. First, the control nodes and fluid particles are triangulated together, and the solid boundary edges are constrained. A complete triangulation of the computational domain is thereby obtained (step 1). Next, the α -shape algorithm is applied to determine which elements belong to the fluid domain, also allowing to detect the solid and free surface boundaries (step 2). This step is detailed in Section 2.3. From the full geometrical domain, we determine which are the boundaries of the fluid along solid walls. Then, the quality of the fluid elements is improved using the Delaunay refinement algorithm (step 3). The algorithm also ensures that a size field is respected for all the elements in the fluid domain. The mesh refinement algorithm is detailed in Section 2.4. The definition of the mesh is now complete: Elements are guaranteed to meet a certain quality and to respect the size field, and the boundaries are also clearly defined.

Since some particles may have been added during the mesh adaptation process, it is necessary to project the velocity of the previous time step onto these particles. This is achieved using the deformed mesh of the previous time step using piecewise interpolations. The equations of motion can now be solved using the procedure described in Section 2.2 (step

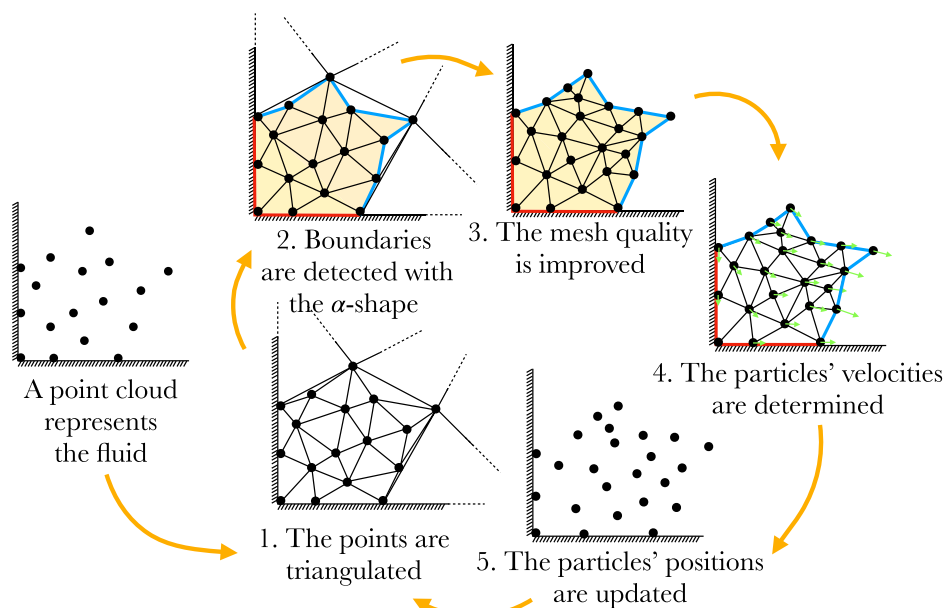


FIGURE 2 The main steps in the PFEM algorithm.

4). The material velocity is thereby obtained throughout the whole fluid domain, and in particular on the particles. Using this velocity, the particles are displaced (step 5).

2.2 | Solving free surface flows

The PFEM solves the equations of motion using a finite element formulation. In this paper, we focus on incompressible free surface flows, hence the Navier–Stokes equations are used in their incompressible form. The strong form can be expressed as follows:

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla p + \mu \nabla^2 \mathbf{u} + \rho \mathbf{g}, \quad (1)$$

$$\nabla \cdot \mathbf{u} = 0. \quad (2)$$

Here, ρ is the density of the fluid, $\mathbf{u}$ its velocity, p the pressure, μ the dynamic viscosity, and $\mathbf{g}$ the gravitational acceleration. The momentum Equation (1) is expressed with the material derivative of velocity. Indeed, recall that the PFEM is a Lagrangian method: the velocity obtained at each node is the velocity of a material point within the fluid. The nonlinearity of the Eulerian form of the momentum equation caused by the convective acceleration is not present in the PFEM. This is a great advantage when solving the system of equations.

Equations (1) and (2) need to be complemented with boundary conditions. Along solid walls, a slip-wall condition is applied, meaning that the velocity's normal component to the wall is zero:

$$\mathbf{u} \cdot \hat{\mathbf{n}} = 0 \quad \text{for } \mathbf{x} \in \Gamma_{\text{wall}}. \quad (3)$$

For a detailed explanation of the slip-wall condition, we refer to Section 2.5.1. Along the free surface, the stress normal to the boundary is subject to an exterior pressure:

$$\boldsymbol{\sigma} \cdot \hat{\mathbf{n}} = p_{\text{ext}} \hat{\mathbf{n}} \quad \mathbf{x} \in \Gamma_{\text{f.s.}}, \quad (4)$$

where $\boldsymbol{\sigma}$ is the fluid stress tensor. Finally, for cavities within the fluid, a pressure is applied around the bubbles to emulate the presence of an incompressible fluid within these cavities:

$$\boldsymbol{\sigma} \cdot \hat{\mathbf{n}} = p_{\text{cavity}} \hat{\mathbf{n}} \quad \mathbf{x} \in \Gamma_{\text{bubble}}. \quad (5)$$

The derivation of this pressure p_{cavity} is presented in Section 2.5.2. An illustration of the boundary conditions is presented in Figure 3.

Equations (1) and (2) are solved with the fluid component of the Migflow solver,¹⁵ which implements the standard piecewise pressure stabilized linear finite element formulation.¹⁶

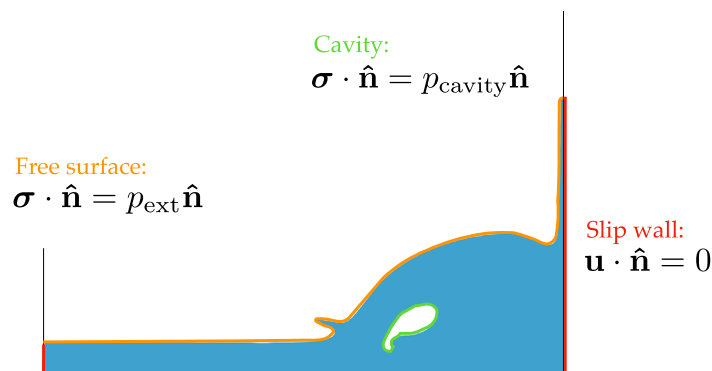


FIGURE 3 Boundary conditions along the fluid.

The solver is monolithic in velocity and pressure. A first order implicit integration time scheme is used to obtain the velocity field throughout the domain. In particular, the velocity vector at the nodes corresponds to the material velocity $\mathbf{U}_i^t$ of the particles. The position $\mathbf{X}_i^t$ of the particles is then updated as follows:

$$\mathbf{X}_i^{t+1} = \mathbf{X}_i^t + \mathbf{U}_i^t \Delta t, \quad i = 1, \dots, n$$

for each particle i .

2.3 | Using α -shapes to capture the domain changes

Due to the displacement of the particles, the domain is constantly undergoing geometrical and topological changes. The shape of the fluid domain therefore needs to be detected at every time step. To this end, the approach proposed in the particle finite element method considers a shape detection algorithm closely related to the α -shape of a set of points. Initially proposed by Edelsbrunner,¹⁷ the α -shape is a generalization of the convex hull of a set of points S . Consider α a positive real number whose dimension is a length. An open disk of radius α is said to be empty if it contains no point of S . The α -hull of S is the region that does not intersect any open disk of radius α . The α -shape has the same topology as the α -hull but every circle arc of radius α of the α -hull of S is replaced by a straight line in the α -shape of S (see Figure 4).

The Delaunay triangulation $\mathcal{T}(S)$ of S is the easiest way to build the α -shape of S . We remove from $\mathcal{T}(S)$ triangles that have a circumcircle greater than α and the boundary of that domain is the α -shape of S . In this work, we assume that α is a field. In other words, $\alpha(\mathbf{x})$ depends on the position $\mathbf{x}$. Assuming that $\mathbf{x}_e$ is the circumcenter of element e , we define an adimensional parameter

$$\alpha_e = \frac{R_e}{\alpha(\mathbf{x}_e)},$$

where R_e is the circumradius of element e . Triangles having a value of α_e greater than a certain threshold value (typically, a value of 1.2) are removed from the fluid domain (see Figure 5). The definition of $\alpha(\mathbf{x})$ is detailed in Section 2.4.

We combine the α -shape approach with a finite element method to solve fluid flows. The initial set of points S_0 at $t = 0$ corresponds to the user-defined finite element mesh of a given fluid domain Ω . The elements of this initial mesh must have circumcircles that respect the size field $\alpha(\mathbf{x})$. To generate this initial mesh, we use a *Delaunay-based* mesh generator (Gmsh¹⁸). In a classical mesh generation process, points are inserted as part of a *Delaunay refinement* process with the aim of robustly respecting the mesh size, such that the circumradius of the triangles is controlled. The Delaunay triangulation does not necessarily minimize the length of the edges. In other words, the Delaunay triangulation does not always connect points that are close to each other. The only result in that regard is that, in a Delaunay triangulation, the graph distance between two points is never more than 2 times the Euclidian distance between those points. Thus, if points are inserted randomly, even in a controlled process such as a Poisson process, there is no guarantee that large

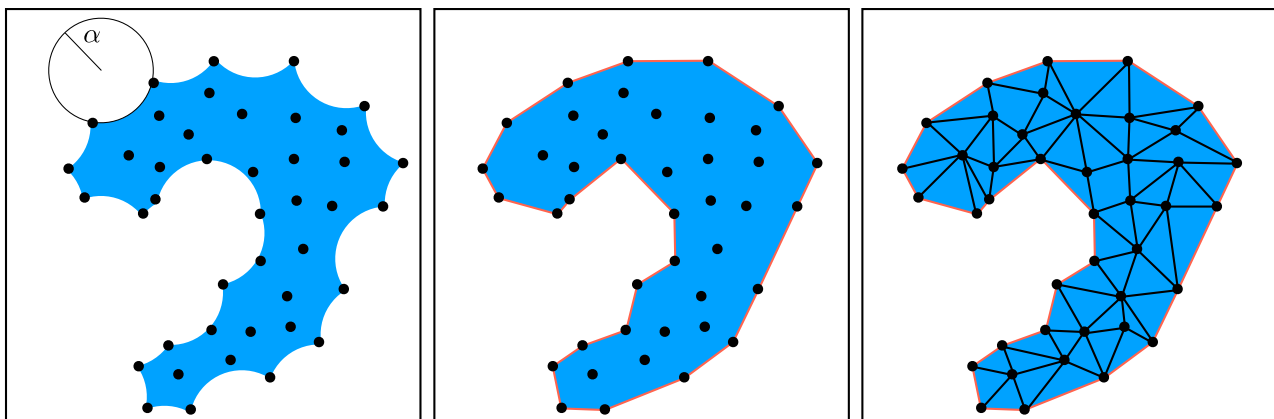


FIGURE 4 Left: α -hull of a set of points. Center: the corresponding α -shape. Right: The Delaunay triangulation of the α -shape.

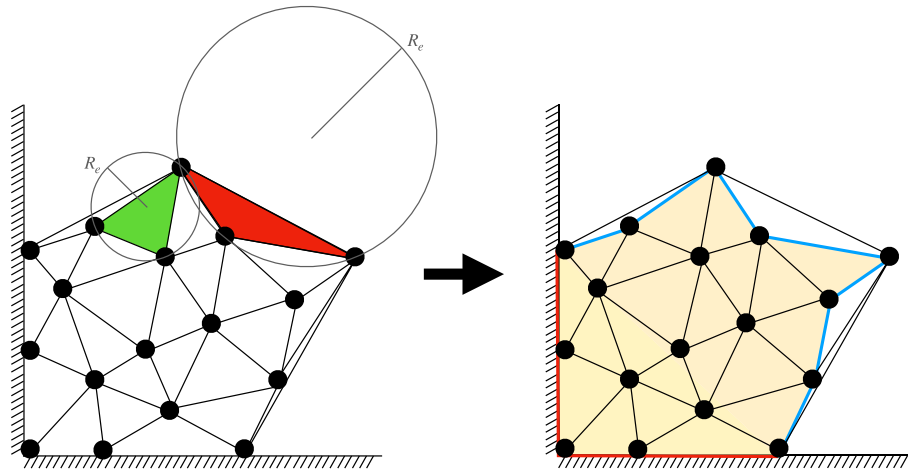


FIGURE 5 To build the α -shape of the set of fluid particles, we build the Delaunay triangulation. Only elements of the triangulation whose circumradii are smaller than a prescribed value are considered part of the fluid. The green element is added to the fluid, while the red element is not. Boundaries are naturally detected by the algorithm.

triangles will not appear in $\mathcal{T}(S)$. Slightly disturbing the set of points by a distance locally smaller than $\alpha(\mathbf{x})$ is not really a concern: the Voronoi diagram (dual of the Delaunay triangulation) is a very stable structure and the topology of the α -shape will not be disturbed unless the topology of the fluid domain changes.¹⁷ However, if nothing is done for a large number of time steps and the points are simply moved using the fluid velocity, then the initial structure proposed by the mesh generator will be broken and major disturbances in the α -shape will appear: holes in the fluid domain, oscillatory and Chainsaw-like structures along the free surface. In Figure 6, elements in blue are those recognized by the α -shape as fluid. As the simulation advances, the initial distribution of the particles is disturbed by their velocity. Without any mesh adaptation, as presented in the left column, holes appear within the fluid domain. Elements which should have remained part of the fluid have been removed, since they no longer respect the α criterion described in the previous section. In the PFEM, it is therefore crucial to adapt the mesh through particles insertion and removal.

The philosophy behind our approach is as follows. We assume that at time step t the set of points S_t has a structure resulting from a clean Delaunay refinement. We therefore determine the fluid domain's shape through the α -shape process described above. Next, in order to ensure that all elements within the fluid domain have sufficient quality, the set of points S_t is modified by a Delaunay refinement algorithm with a structure of proof to restore a clean point distribution. Then, we calculate the velocities $\mathbf{U}_i^t$ of the points by finite elements on the domain bounded by the α -shape of S_t . We then move the points according to the fluid velocity to create $S_{t+\Delta t}$. To ensure sufficient precision of the solution, a time step is selected to respect the following condition:

$$\Delta t < \inf_{\mathbf{x}} \frac{\alpha(\mathbf{x})}{\|\mathbf{U}^t(\mathbf{x})\|}.$$

This condition, similar to a Courant–Friedrichs–Lewy condition, corresponds to an approximate upper bound on the time step to avoid excessive element deformations and to be sure the desired physical scales are captured within the mesh. In practice, a fixed time step is selected based on an approximation of the maximum velocity and the minimum element size over the course of the simulation.

This is perfectly acceptable since we assume that the mesh size chosen corresponds to the characteristic length of the given phenomena.

2.4 | Mesh adaptation

As shown in the right column of Figure 6, in contrast to the left column, quality-based mesh adaptation allows to accurately represent the evolution of the fluid domain. In this section, we describe the mesh adaptation algorithm which, along with the modified version of the α -shape algorithm described above, ensures an accurate representation of the domain at

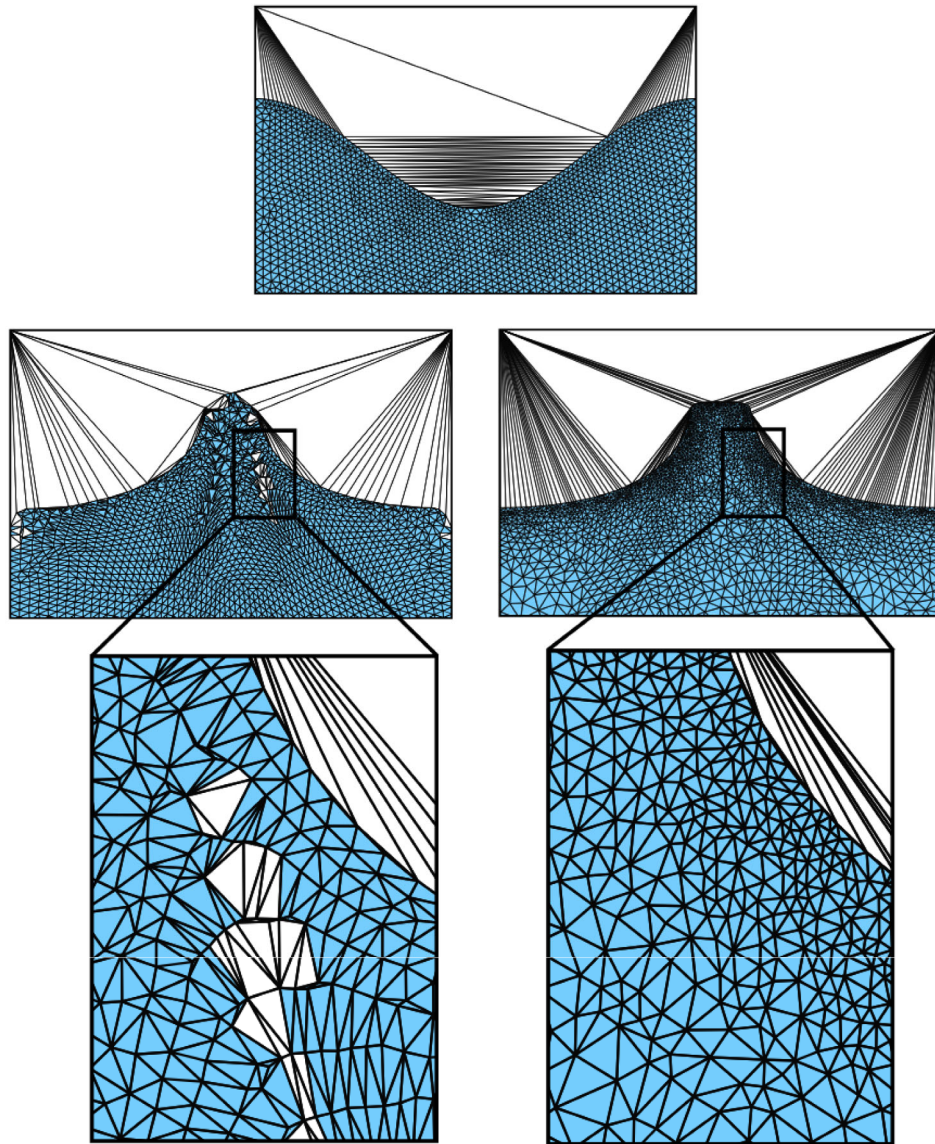


FIGURE 6 Comparison of a sloshing simulation with (right) and without (left) mesh adaptation. Holes appear inside the mesh due the α -shape algorithm that removes elements that no longer respect the quality and size constraints.

each time step. An example using a spatially variable analytic incompressible velocity field is then used to demonstrate the effectiveness of the proposed approach.

2.4.1 | Adaptive meshing algorithm

First, a size field is defined to allow variations in the size of the mesh. For instance, a higher resolution may be desired near the free surface to accurately simulate the strong deformations and topological changes. A KD-tree structure is used to compute the distance to the free surface to derive this size field. The position of the free surface at time $t + \Delta t$ is determined by advecting the free surface at time t with the velocity field. An example is presented in Figure 7.

To adapt the mesh based on this size field, nodes are inserted into the triangulation. To this end, an algorithm greatly inspired by Chew's algorithm for Delaunay mesh refinement is employed.¹² Chew's algorithm ensures that all triangles within the triangulation are well-sized and well-shaped and that internal and external boundaries are maintained. Shewchuck has since demonstrated that using this algorithm, no angles smaller than 26.5° will be generated, confirming the well-shapedness of the mesh resulting from this algorithm.¹³ The user-defined size field dictates the size of the elements.

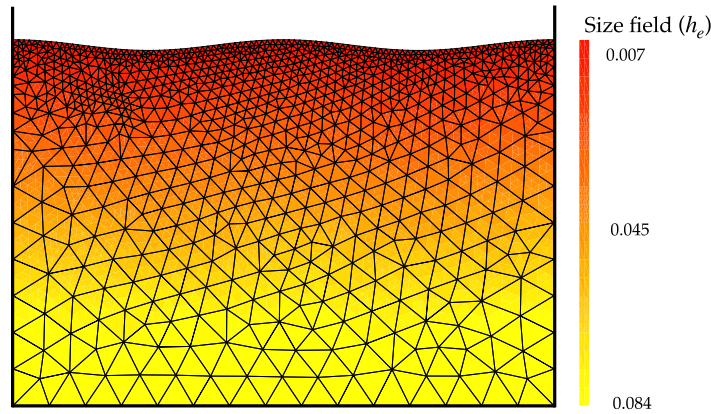


FIGURE 7 The size of the elements is dictated by their distance to the free surface.

Note that the algorithm is performed only after the fluid domain has been detected. Only elements which have been defined as being part of the fluid by the α -shape will potentially be adapted. Hence, only the part of the mesh defined as the fluid domain will be improved in terms of quality from one time step to another.

The algorithm takes a constrained Delaunay triangulation $\mathcal{T}_c(S)$ as input. $\mathcal{T}_c(S)$ is a Delaunay triangulation in which certain edges are constrained, meaning that they must remain in the mesh and therefore cannot be flipped.

$\mathcal{T}_c(S)$ is obtained from the α -shape algorithm, which defines the shape of the domain. A set of boundary edges are thereby obtained, those along solid walls and those belonging to the free surface. These edges are the constrained edges of $\mathcal{T}_c(S)$. It is important to note that the α -shape has the role of defining the shape of the domain, in order to capture the topological changes. With $\mathcal{T}_c(S)$ as input, the algorithm then seeks to meet the imposed quality and size criteria.

The essence of the approach consists of inserting nodes in the triangulation at the circumcenter of poorly shaped or poorly sized triangles. Edges are flipped after insertion to ensure the mesh remains Delaunay. In case a circumcenter falls outside the meshing domain, then the bounding edge that crosses the line-of-sight of the triangle to its circumcenter is split. Algorithm 1 summarizes the process. Figure 8 illustrates the main steps of the algorithm on a small sample of elements.

Node insertion at the circumcenter of poor-quality elements guarantees overall mesh improvement. Indeed, the only new edges created by an insertion of a node are edges connected to this node (see e.g., Figure 8, step 2 to 3). This node has been inserted at the circumcenter of a triangle which respects the Delaunay criterion. In other words, there are no other nodes within the circumcircle of this triangle. Therefore, none of the new edges can be shorter than the circumradius of this triangle. This avoids the appearance of triangles with a small inner radius to circumradius ratio. This property of the algorithm is essential to the PFEM mesh adaptation step.

The quality criterion for an element e is defined as $\gamma_e = r_e/R_e$, where r_e is the inscribed radius and R_e is the circumradius. The size criterion compares the circumradius of the element to the prescribed size field. Before the mesh refinement algorithm runs, a coarsening step is performed. This allows to avoid excessively small elements to remain inside the mesh, where the size field actually allows a coarser mesh. If two particles are too close to each other, their mutual edge is collapsed and $\mathcal{T}_c(S)$ is updated accordingly. Finally, to ensure termination of the algorithm, no points are inserted for elements whose circumradius is smaller than a limit value.

Using the mesh adaptation algorithm in the PFEM greatly improves the quality of the mesh. This step turns out to be crucial, as it removes uncertainties linked to the α -shape's decision of the shape of the fluid.

2.4.2 | Adaptive meshing illustration: vortex-in-a-box

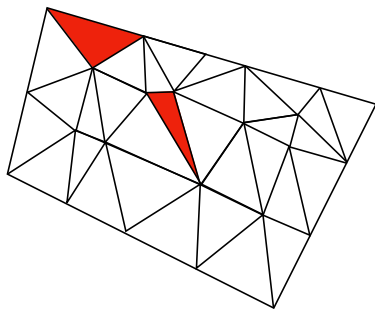
To illustrate the ability of the mesh adaptation algorithm to capture flow features down to the size of the mesh, the vortex-in-a-box simulation is presented below. In this flow, a disk is inserted into a circular flow inside a box with the following stream function:

Algorithm 1. Mesh adaptation

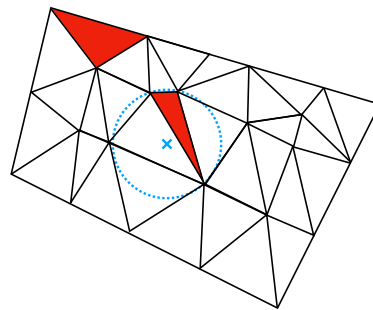
```

Get the  $\alpha$ -shape of the point cloud and define the fluid domain;
Build  $\mathcal{T}_c(S)$  of the fluid domain;
Collapse internal edges that are too small with respect to the size field;
Get all the elements in the  $\alpha$ -shape that do not meet the quality or size criterion;
while There are still poorly shaped or sized elements do
    Choose the largest element that does not respect the quality criterion;
    Determine its circumcenter;
    if The circumcenter is inside the fluid domain then
        Insert a new node at the circumcenter;
    end if
    if The circumcenter is outside of the fluid domain then
        Insert a node at the middle of the constrained edge that blocks the triangle from seeing its circumcenter;
        Delete any node that is closer to the newly created node than the length of the new edge;
    end if
    Do edge swaps to recover a Delaunay mesh;
end while

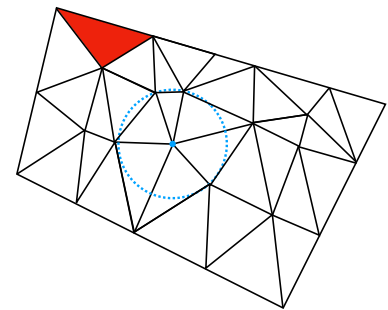
```



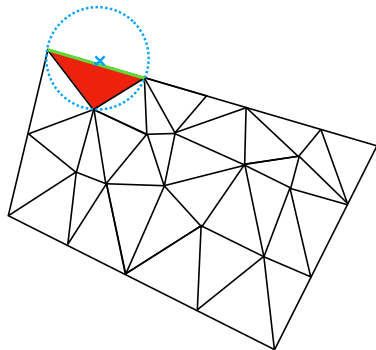
1. Find all elements which do not respect the quality/size criterion



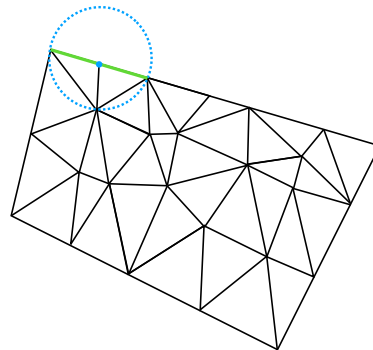
2. Insert a node at the circumcenter if it is inside the fluid domain.



3. Swap edges to recover the Delaunay triangulation



4. If the circumcenter is not inside the fluid domain, split the boundary edge.



5. Swap edges to recover the Delaunay triangulation

FIGURE 8 Main steps of the mesh adaptation algorithm.

$$\psi = \frac{1}{\pi} \sin^2(\pi x) \sin^2(\pi y) \sin(2\pi t/T).$$

The resulting velocity field is obtained as follows:

$$u_x = \frac{\partial \psi}{\partial y} = \sin(2\pi y) \sin^2(\pi x) \sin(2\pi t/T),$$

$$u_y = -\frac{\partial \psi}{\partial x} = -\sin(2\pi x) \sin^2(\pi y) \sin(2\pi t/T).$$

The domain is a box of size $[0, 1] \times [0, 1]$, and the initial disk is located at $(0.5, 0.75)$. Gradually, as the disk follows the rotation of the flow, its shape evolves into a thin filament. The filament can become infinitely long and thin, and the challenge of the numerical solver is to capture this filament down to the size of the grid. As a means of validation, a time-dependent term, $\sin(2\pi t/T)$, is added to the velocity field that reverses the direction of the flow after a half period $T/2$. Hence, if the initial disk is recovered after a full period T , this shows that there is no area loss. Two different test cases are presented.

In the first, the fluid domain is considered as the inside of the disk (i.e., a bubble of fluid). The second, its counter-part, considers the region around the disk as fluid (i.e., a hole in the fluid). Time integration, for this particular example, is performed using a fourth order Runge Kutta scheme. This scheme is chosen in order to avoid as much as possible volume conservation errors caused by the time integration scheme, and hence focus almost entirely on the re-meshing procedure itself. Figures 9 and 10 present a few timeshots of the two different simulations. The pink lines correspond to the actual position of the interface.

Figure 11 presents the full meshed domain, for both cases. Clearly, mesh adaptation is only performed inside the region that is considered to be the fluid region. A comparison is also done with a simulation without any mesh adaptation. The number of elements is about the same (≈ 25000), but a uniform mesh size is considered throughout the domain. Figure 12 shows the volume variation of the disk over time, for all three cases presented above. For an objective comparison, the volume of the inside of the bubble is computed. The greater volume variation for the hole case is explained by the fact that the end of the filament becomes infinitely thin as the vortex rotates. Hence, capturing this thin filament with the α -shape would also require infinitely small elements in this region. A portion of the volume of the disk is therefore lost as time is reversed. The third image (at $t = 2.5$ s) of Figure 10 illustrates this phenomenon. This situation does not happen when simulating the inside of the disk since there are no nonfluid elements of small, α -acceptable size in this region. Only the small elements which become too distorted end up being removed by the α -shape, which explains the very small variation of volume in this case. Overall, it is clear that mesh adaptation is crucial to correctly track the evolution of the fluid domain.

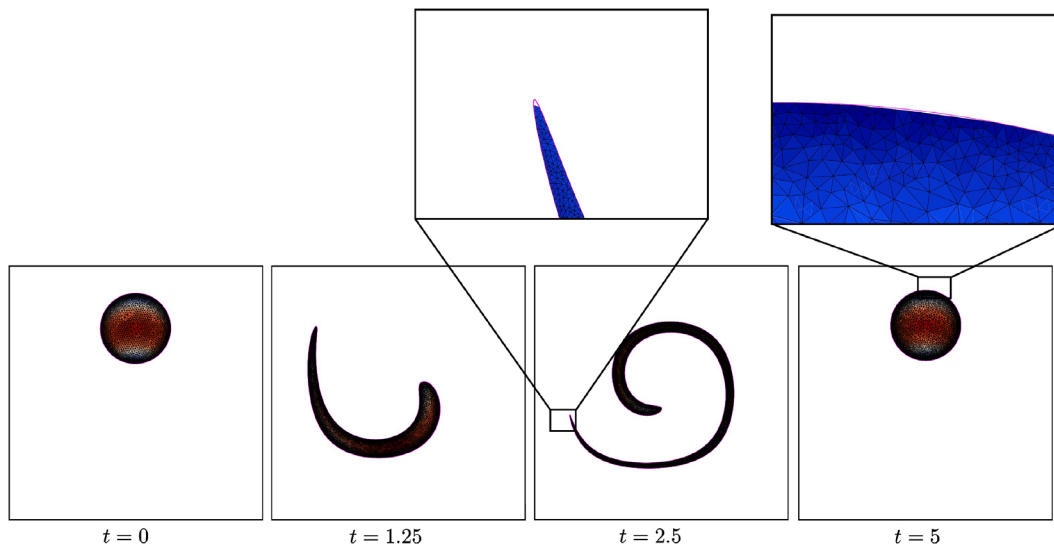


FIGURE 9 Snapshots of the vortex-in-a-box simulation for the first case (a bubble of fluid).

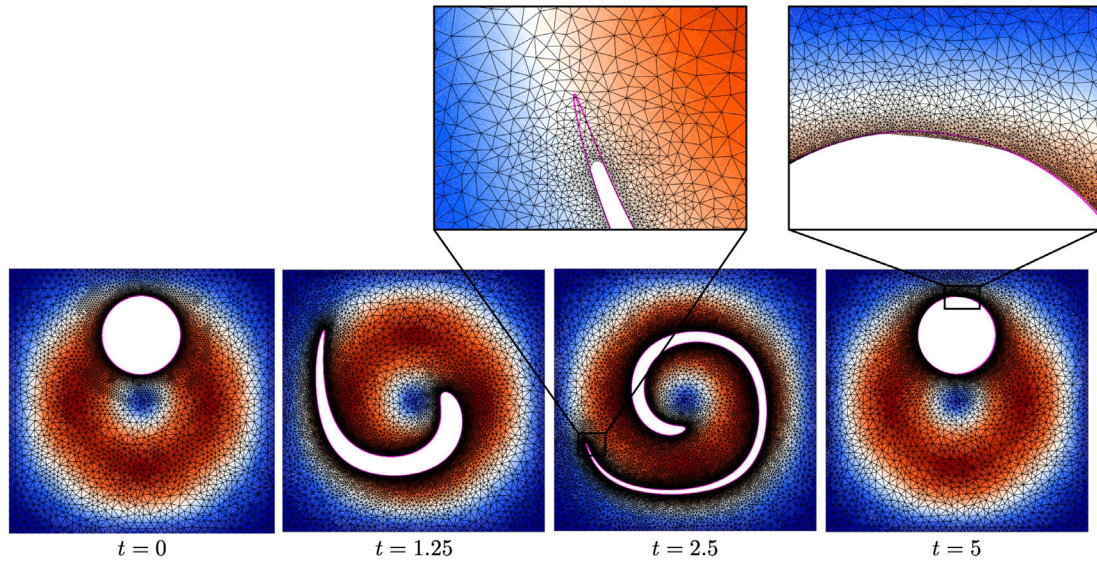


FIGURE 10 Snapshots of the vortex-in-a-box simulation for the second case (a hole in the fluid).

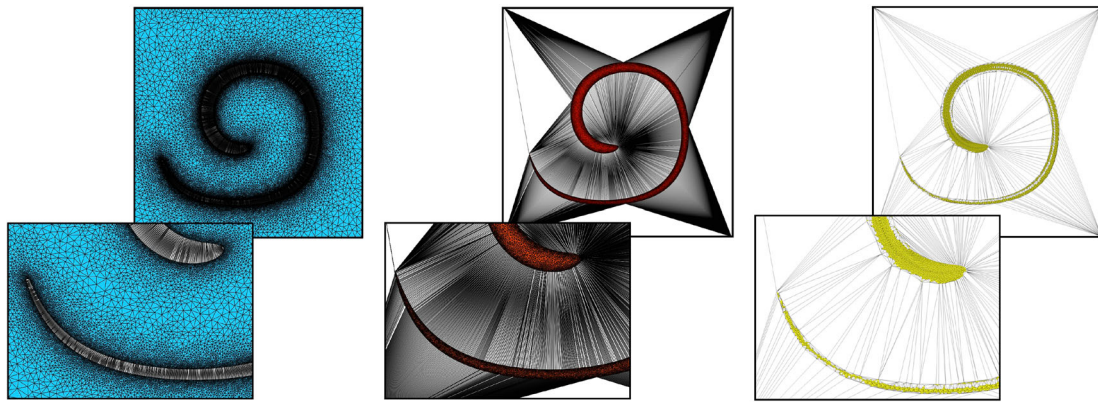


FIGURE 11 Difference between simulating the outside (left) or inside (middle) of the disk. Comparison with a simulation without any mesh adaptation (right).

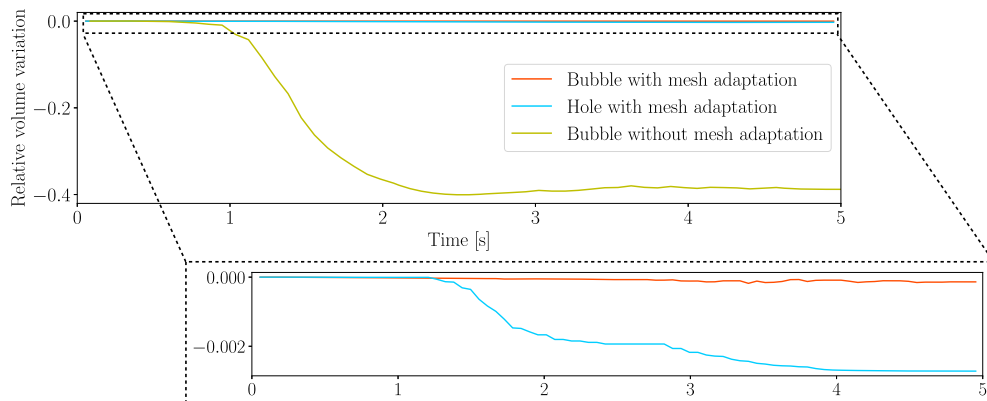


FIGURE 12 Volume variation during one full period of duration $T = 5$ s, when simulating the inside of the disk (bubble) and the outside of the disk (hole), compared to a simulation without any mesh adaptation. For the sake of comparison, the relative volume variation of the *inside* of the disk is computed for all three cases.

2.5 | Boundary conditions

Two boundary condition formulations are presented next. The first concerns the slip boundary condition along solid walls. The second addresses empty cavities within the fluid.

2.5.1 | Slip boundary conditions

Accurately capturing the boundary layer along a wall requires strong refinement along these walls, resulting in expensive computational efforts. To circumvent this, it is common to allow a relative motion between the fluid and the solid wall, resulting in a slip-wall boundary condition. Different approaches for slip boundaries in the PFEM have been proposed. Cerquaglia et al.¹⁹ use one layer of elements along the solid wall to represent a boundary layer. Others, for instance Cremonesi et al.²⁰ use an Eulerian-Lagrangian formulation to account for slip walls.

In this work, we suggest to avoid the use of fixed particles attached to the walls. The boundaries of the computational domain are only represented geometrically. To implement the slip boundary condition, the motion of particles situated along a solid wall is restricted to the direction tangential to this wall. In other words, the fluid particles slide along the wall and cannot reenter the bulk of the domain. Any slip boundary condition, ranging from no-slip to free-slip, can then be applied with wall models.

A special case must be considered for particles that are still in the bulk, and cross a wall after their explicit position update, as illustrated in Figure 13. Indeed, in such cases, the particle does not know a priori that it will hit a wall. To correct this, these particles are projected again onto the wall by backtracking their trajectory, and their velocity component normal to the wall is set to zero. The slip-boundary condition is thereby recovered. An error on mass conservation of the order of the time step is conceded in these situations since the incompressibility condition is locally disregarded. A future improvement involves using an implicit update on the position in order to predict collisions between the fluid and solid walls.

2.5.2 | Empty bubble boundary condition

In some simulations, for example those driven by high levels of inertia, the fluid can encapsulate cavities to form bubbles, as illustrated in Figure 14. Although the presence of these bubbles is natural, if no special attention is paid to them, they disappear. If we ignore the fluid inside the bubbles, the model considers those regions as a void spaces, and fills them up. Global mass conservation is imposed on the fluid, but not on the bubbles.

Let us assume that the dynamics inside each bubble occur on a time scale that is much faster than that of the fluid. In other words, a variation of the fluid is instantaneously compensated inside the bubble. This is the case for a sufficiently high density ratio between the fluid and the bubble. Between water and air, for instance, this assumption is reasonable. In this case, without modeling explicitly its flow, we can still consider the presence of another incompressible fluid inside the bubble.

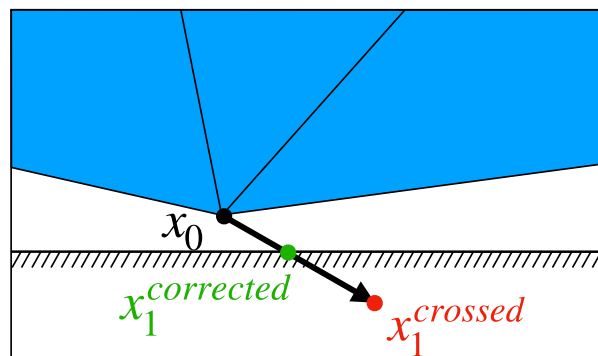


FIGURE 13 When a fluid particle crosses a solid wall, it is reprojected back onto the wall, and its velocity component normal to the wall is set to zero.

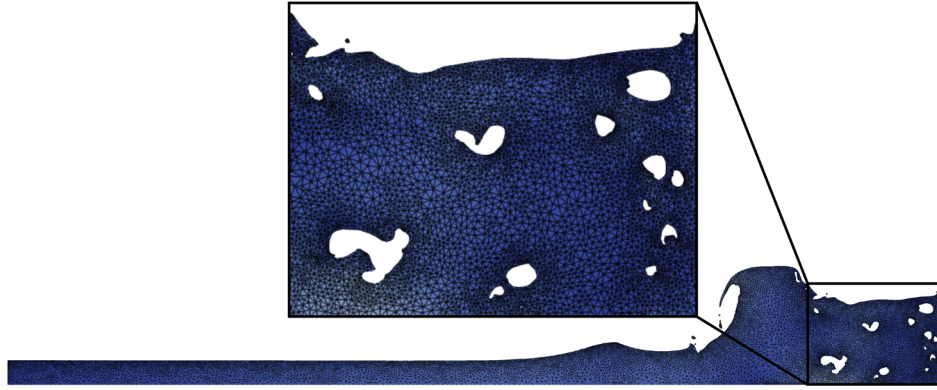


FIGURE 14 This highly transient flow generates the appearance of bubbles of air within the fluid.

We apply a boundary condition that exerts a pressure enforcing the incompressibility of the bubble, while taking into account its weight and the surface stresses on the interface. The pressure term applied on this boundary is composed of three terms: a vertical hydrostatic pressure gradient that accounts for buoyancy, surface tension, and a constant term that enforces incompressibility:

$$p_{\text{cavity}}(\mathbf{x}) = \underbrace{\rho_c g z}_{\text{buoyancy}} + \underbrace{\kappa(\mathbf{x})\sigma}_{\text{surface tension}} + \underbrace{p_i}_{\text{incompressibility}}. \quad (6)$$

Here, ρ_c is the density of the fluid inside the bubble, g the gravitational acceleration, z the vertical component of $\mathbf{x}$, κ is the curvature, and σ the surface tension coefficient. These first two terms control the gradient of p_{cavity} and can be explicitly computed.

To control the average pressure p_i , we add an incompressibility constraint for each bubble:

$$\int_{\Omega_c} \nabla \cdot \mathbf{u}_c \, dV = 0 \Leftrightarrow \int_{\Gamma_c} \mathbf{u}_c \cdot \hat{\mathbf{n}} \, dS = 0,$$

where $\mathbf{u}_c$ is the velocity of the fluid inside the bubble. By continuity, the velocity of both fluids is the same at the interface. Hence, we can express this condition only with the velocity of the modeled fluid,

$$\int_{\Gamma_c} \mathbf{u} \cdot \hat{\mathbf{n}} \, dS = 0. \quad (7)$$

For each bubble we have thus inserted one new equation, the incompressibility constraint (7), and one new degree of freedom p_i , into the linear system. Those can be seen as Lagrange multipliers used to enforce the constraints linking the nodes on the boundary of each bubble. A validation test case of this boundary condition is presented in Section 3.2.

3 | VERIFICATIONS

This section aims at verifying the different features of our PFEM solver. Two test cases are considered. First, we consider a low amplitude fluid sloshing in a tank. This configuration is endowed with an analytical solution so we can verify analytically the overall approach. The second example is a rising bubble simulation. Other authors have used that test case so we can compare our results with references and verify the accuracy of our incompressibility condition around cavities.

3.1 | Low amplitude sloshing

The sloshing experiment consists in analyzing the evolution of a fluid in a container, with an initial perturbation of the height of the fluid. This initial height takes the form of a sinusoidal wave:

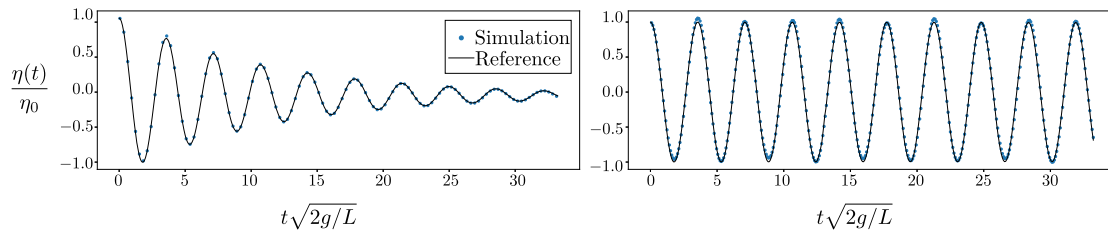


FIGURE 15 Evolution of wave height over time in a low amplitude sloshing experiment. Left: dampening due to viscous effects. Right: no viscous effects.

$$h(x, 0) = h_0 + \eta(x, 0) = h_0 + \eta_0 \cos(2\pi x/L),$$

where h_0 is the mean height of the fluid in the container, η_0 is the amplitude of the perturbation, and L is the length of the container. For sufficiently low values of η_0 , we can assume a linearization of the Navier–Stokes equations to obtain an analytical solution. Indeed, the evolution of the perturbation is given by²¹:

$$\frac{\eta(t)}{\eta_0} = 1 - \frac{1}{1 + 4\nu^2 k^3/g} \left[1 - e^{-2\nu k^2 t} \left(\cos \sqrt{kg} t + 2\nu k^2 \frac{\sin \sqrt{kg} t}{\sqrt{kg}} \right) \right],$$

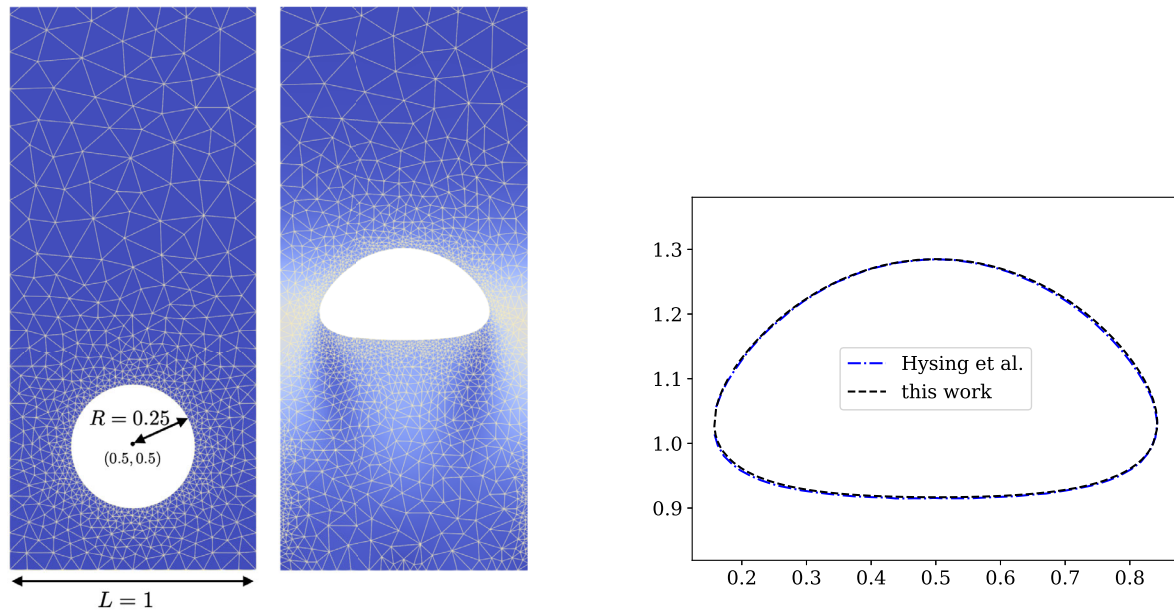
where ν is the kinematic viscosity of the fluid and $k = \pi/L$ is the wave number. This result is obtained under the approximation that $g \gg \nu^2 k^3$, i.e., viscous effects are sufficiently small with respect to the gravitational effects. Moreover, we assume that the initial perturbation is sufficiently small compared to the mean height. Surface tension is not considered. We consider an initial perturbation height $\eta_0/h_0 = 1\%$. The two plots presented in Figure 15 show two different cases. The first, on the left, is a viscous fluid, with $\nu = 0.005$. Hence, there is a dampening effect on the perturbation, and the amplitude of the wave reduces to 0. The second, on the right is a simulation without viscosity. In this case, there is no dampening of the wave and the oscillation amplitude remains constant. Both simulations show good agreement with the analytical reference solution.

3.2 | Rising bubble

The following simulation aims at verifying the boundary condition that imposes incompressibility of a cavity fully surrounded by fluid. The study of rising bubbles in a fluid is well established, and many approaches exist for the simulation of such two-phase flows. To name a few, the volume of fluid approach²² and levelset methods²³ have been used for these simulations. The use of a one-phase fluid model for the simulation of such rising bubbles may not be the most adequate choice, since the dynamics inside the bubble are being neglected. Nevertheless, in order to validate the incompressibility boundary condition, this can be, under certain assumptions, an interesting problem to investigate. Indeed, considering that the dynamics of the bubble happen over a much shorter time period than that of the fluid, it is acceptable to neglect the evolution of the fluid within the bubble.

In the test case presented below, results have been compared with a 2-dimensional simulation of a rising bubble performed by Hysing et al.²² The same experimental configurations have been implemented, and the following physical parameters were considered:

$$\begin{aligned} \rho_{\text{fluid}} &= 1000 \text{ kg m}^{-3}, \\ \rho_{\text{bubble}} &= 100 \text{ kg m}^{-3}, \\ \mu_{\text{fluid}} &= 10 \text{ Pa s}, \\ g &= 0.98 \text{ m s}^{-2}, \\ \sigma &= 24.5 \text{ N m}^{-1}. \end{aligned}$$



(a) Evolution of the rising bubble between initial time ($t = 0$ s) and final time ($t = 3$ s). The colormap corresponds to the norm of the velocity.

(b) Comparison between the reference solution [18] and our PFEM approach at $t = 3$ s.

FIGURE 16 Rising bubble experiment. (A) Evolution of the rising bubble between initial time ($t = 0$ s) and final time ($t = 3$ s). The colormap corresponds to the norm of the velocity. (B) Comparison between the reference solution²² and our PFEM approach at $t = 3$ s.

Here, ρ_{fluid} and μ_{fluid} are the density and viscosity of the fluid outside the bubble, ρ_{bubble} the density of the fluid inside the bubble, g the gravitational acceleration and σ the surface tension. The nondimensional Reynolds and Eotvos numbers are defined as follows:

$$Re = \frac{\rho_{\text{fluid}} U_g L}{\mu_{\text{fluid}}} = 35, \quad Eo = \frac{\rho_{\text{fluid}} U_g^2 L}{\sigma} = 10.$$

The characteristic length scale $L = 2R$, and the gravitational velocity is given by $U_g = \sqrt{g2R}$, with R the initial radius of the bubble. The final state of the bubble is considered to be at $t = 3$ s, time at which the shape of the bubble is compared. Figure 16A presents the initial and final shapes of the bubble as it rises in the fluid, with the color field referring to the velocity. Figure 16B presents the comparison between the two-phase volume of fluid approach performed by Hysing et al., and the result obtained with our model, with mesh adaptation and with the boundary condition on the bubble as described in Section 2.5.2. The results confirm that the incompressibility boundary condition correctly captures the physics of the bubble, with only minimal differences in terms of shape compared to a full, two-phase flow simulation.

4 | RESULTS

To further demonstrate the abilities of our PFEM solver, three simulations are presented in this section. The first is a drop of fluid falling in a bulk, followed by a dam break simulation over a dry bed, and finally a dam break over a wet bed.

4.1 | Falling drop

The situation of a drop of fluid falling into a bulk is an interesting test case to study mass variation effects due to remeshing and topological changes. Indeed, due to the impact of the drop, splashes occur and the domain changes are challenging

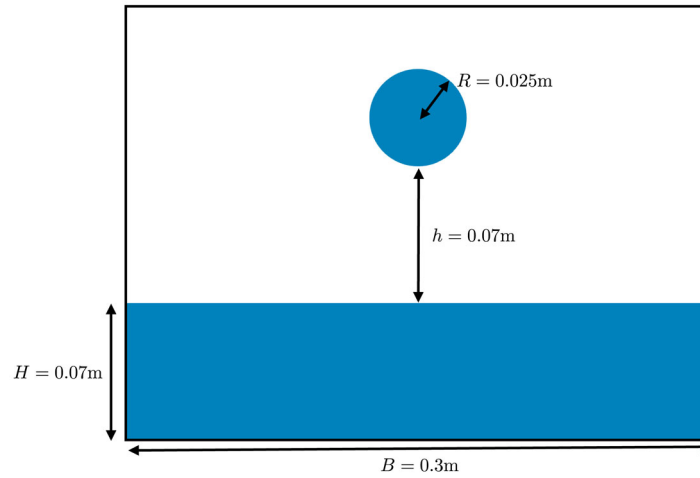


FIGURE 17 Initial configuration of the falling drop test case.

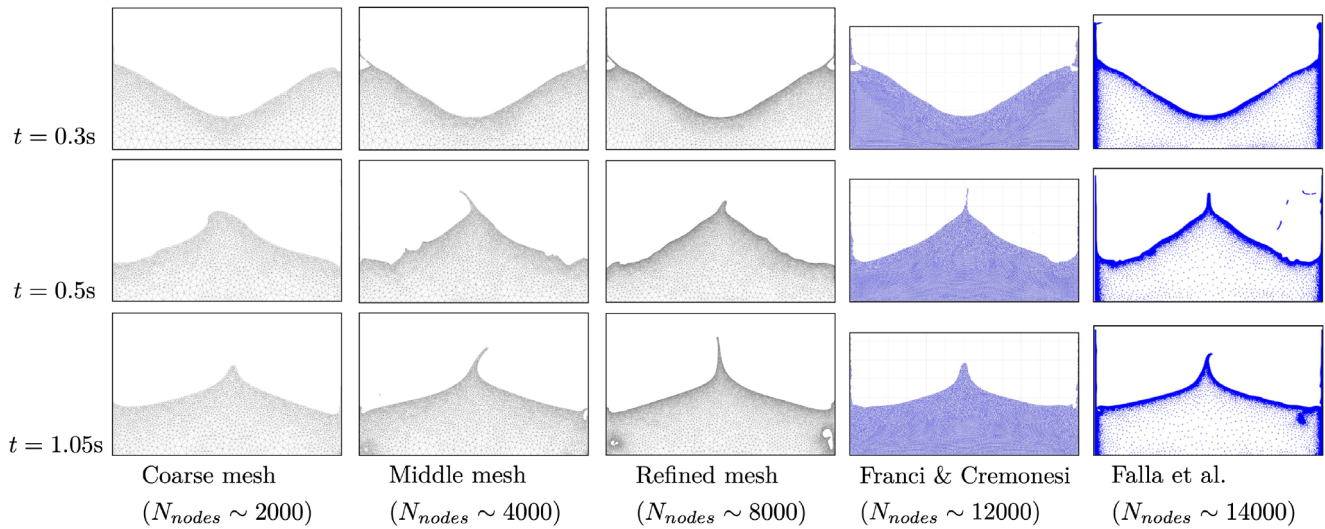


FIGURE 18 Falling drop simulation. Different time shots for different levels of refinement, and comparison with Franci and Cremonesi²⁴ and Falla et al.¹⁴

to capture. Franci and Cremonesi²⁴ have studied this effect with a PFEM approach, and they were able to show that, as the mesh is refined, mass variation effects are reduced which shows the consistency of the model. With our proposed algorithm for mesh adaptation, the nonuniform mesh allows for more refinement near the free surface, allowing to reach higher accuracy with a smaller number of particles. The geometry of the initial setup is presented in Figure 17. To compare with other results from the literature, the density and viscosity of the fluid are respectively 1000 kg m^{-3} and 10^{-1} Pa s . We define a Reynolds number based on the initial diameter of the drop and the velocity as it impacts the bulk:

$$\text{Re} = \frac{v_{\text{impact}} \rho D}{\mu} = 586.$$

Figure 18 presents a few snap shots of the falling drop simulation. These results are compared to simulations performed by Franci and Cremonesi²⁴ and Falla et al.,¹⁴ who performed the same simulations with identical parameters. The former used a uniform grid (with a number of particles estimated around 12,000), while the latter considered a locally refined mesh. The obtained results compare relatively well with the other results from the literature, and similar levels of detail can be captured with smaller amounts of particles.

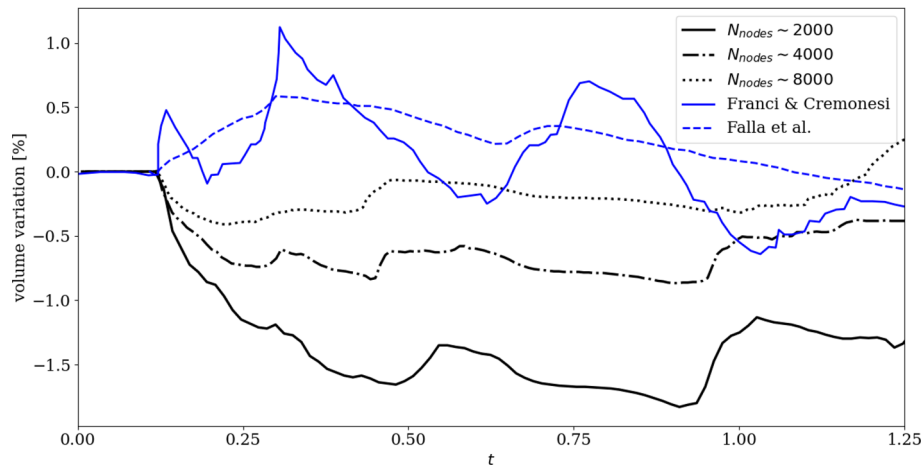


FIGURE 19 Volume variation over time for the falling drop simulation for different levels of refinement, and comparison with Franci and Cremonesi, and Falla et al.

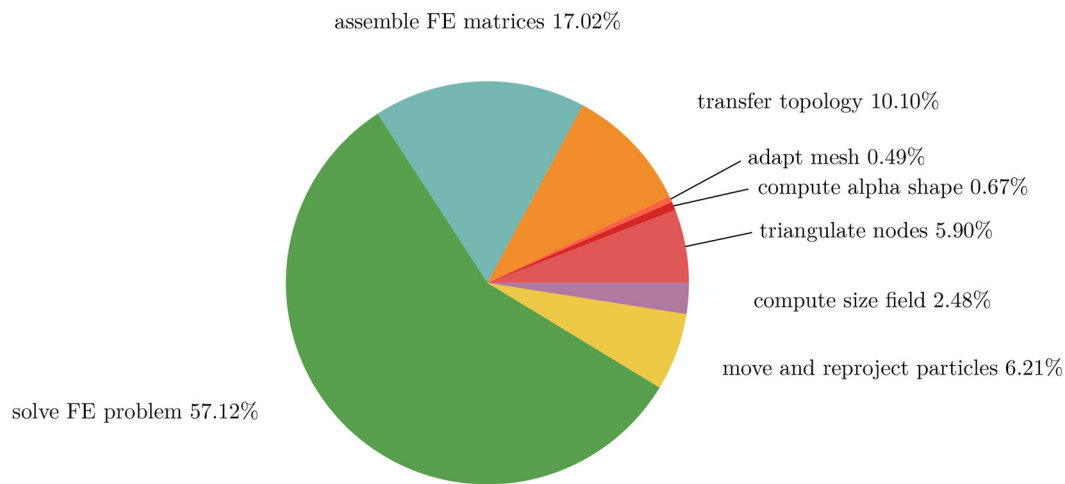


FIGURE 20 Average computational cost of the different steps of the algorithm.

Figure 19 presents the evolution of the total volume of the fluid over time. The variation reduces as the mesh is refined, which shows consistency of the approach. Moreover, our most refined mesh, with a number of particles around 8000, shows smaller variations compared to Franci and Cremonesi's most refined result. Compared to Falla et al., our approach shows smaller amounts of volume variation as well with a lesser amount of particles. Another point worth mentioning is the appearance of air bubbles within the fluid, especially visible in the most refined simulation. These bubbles appear because of trapped air coming from the free surface, which then drops due to the motion inside the fluid. This again highlights the interest of the incompressibility condition around the bubbles. It also shows the capacity the PFEM offers in capturing lower and lower scales in the flow as the mesh is refined.

Finally, to give a general idea of computational costs, Figure 20 presents a pie chart of the average time for the different steps within the algorithm. These results show that the computational cost of the full re-meshing procedure, which includes triangulating the nodes, computing the α -shape, and refining the mesh, represents about 7% of the total cost. Since the refinement is performed at each time step, not many node insertions need to be performed each time, allowing to keep the computational cost very low. As a word of caution, work is ongoing to optimize the different blocks of the solver, in particular regarding the fluid solver. The chart is only meant to give a rough indication on timings. In particular, mesh adaptation, which includes coarsening and refinement based on the size field, represents only a marginal cost. The topology transfer, which takes about 10% of the cost, includes transferring the mesh to the fluid solver, but also interpolating

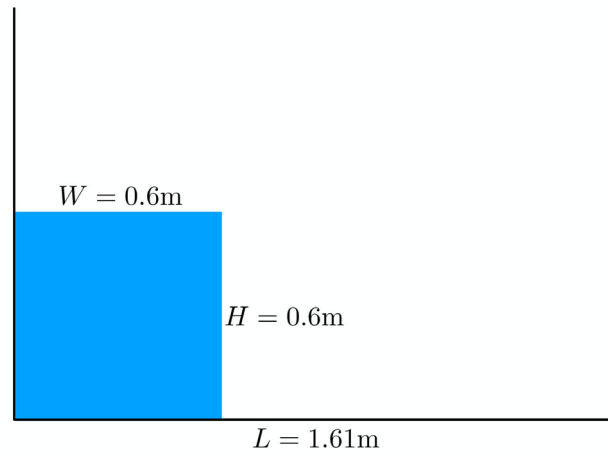


FIGURE 21 Geometrical configuration of the dam break experiment.

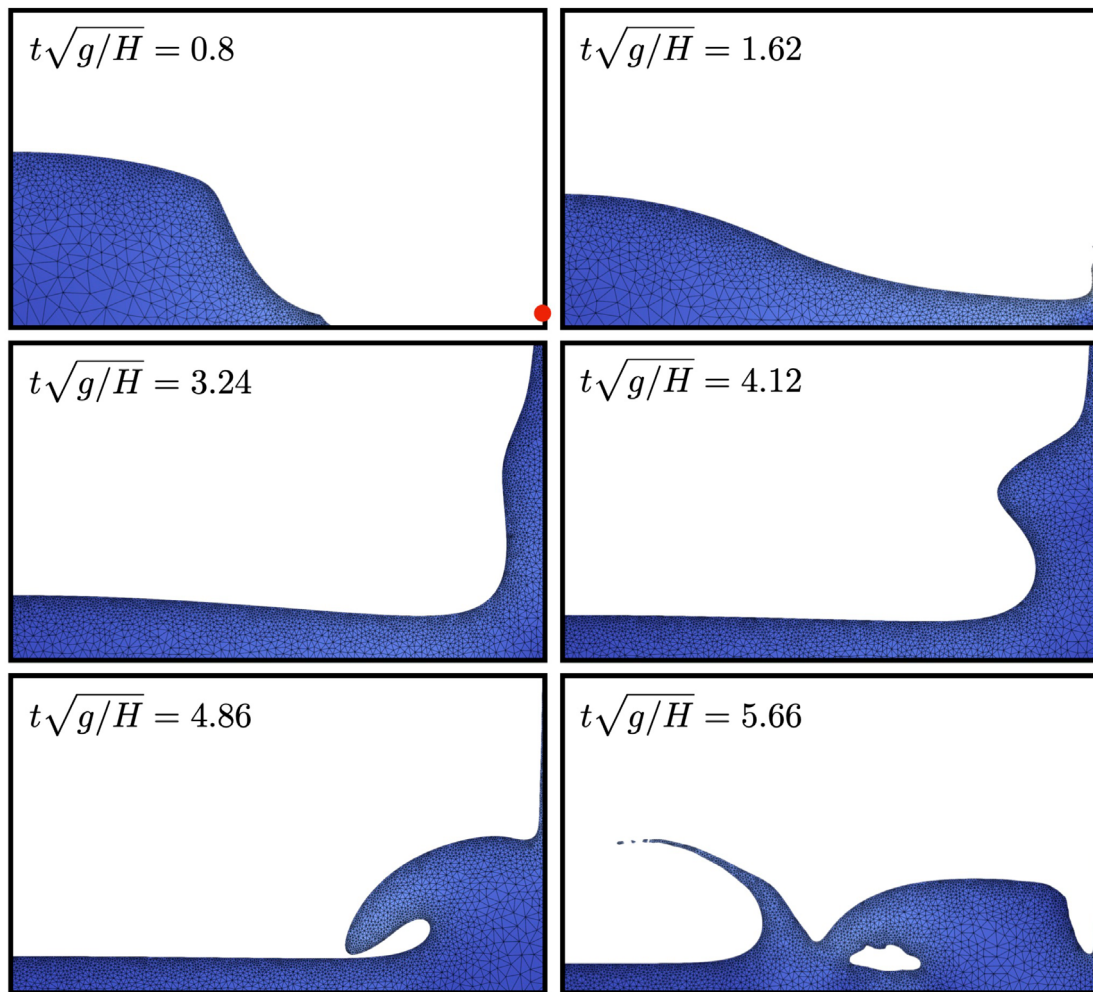


FIGURE 22 Different time steps of the dam break over a dry bed simulation.

the solution on the nodes and coloring the boundaries. The main cost for the meshing step is the triangulation of the nodes, which itself has a low cost compared to the fluid solver.

4.2 | Dam break over a dry bed

In the dam break simulation, a column of fluid initially at rest between three walls is set in motion due to the sudden removal of one of the side walls. A flow then develops along the bottom surface, until it hits an outer vertical wall. The flow quickly becomes turbulent as a wave collapses back onto the fluid and propagates in the other direction. A high amplitude sloshing regime develops, and the fluid returns to a stable state as it is slowed down by its own viscosity. The initial regime has been well documented, and experimental data exist to validate the numerical solvers.

Non dimensional times are given by $t^* = t\sqrt{g/H}$, with $H = 0.6$ m the initial height of the water column. The fluid considered in this case is water, with density $\rho = 1000 \text{ kg m}^{-3}$ and viscosity $\mu = 855 \times 10^{-6} \text{ Pa s}$. The initial dimensional values of the dam break over a dry bed are summarized in Figure 21, and Figure 22 presents a few time steps of the simulation. The number of particles in this simulation is around 3000.

To validate the obtained results, the pressure is probed on the right wall, at a height of 0.03 and 0.08 m, as illustrated by the red dot on the first time step in Figure 22. The evolution of the pressure is presented in Figure 23 and Figure 24.

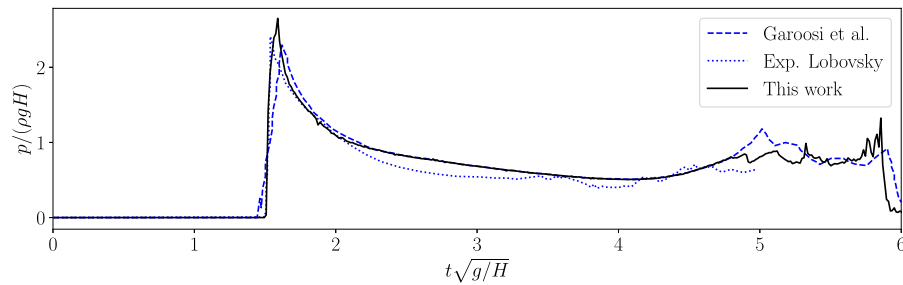


FIGURE 23 Pressure probed at a height of 0.03 m along the right wall.

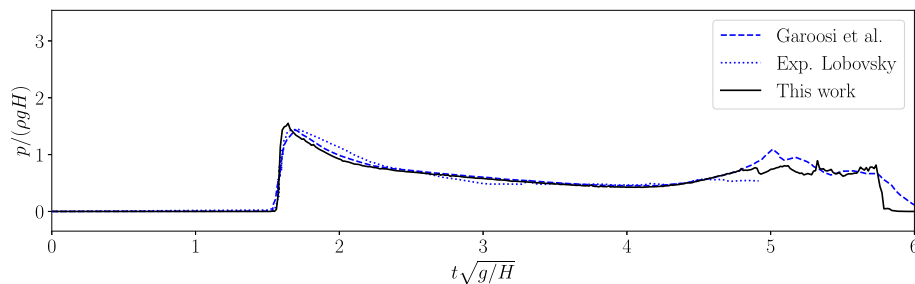


FIGURE 24 Pressure probed at a height of 0.08 m along the right wall.

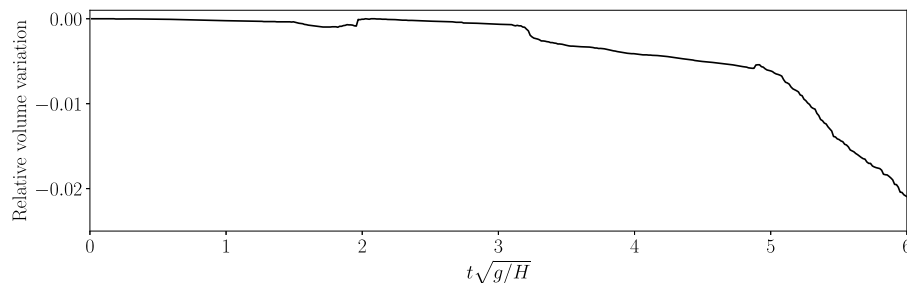


FIGURE 25 Relative volume variation of the fluid.

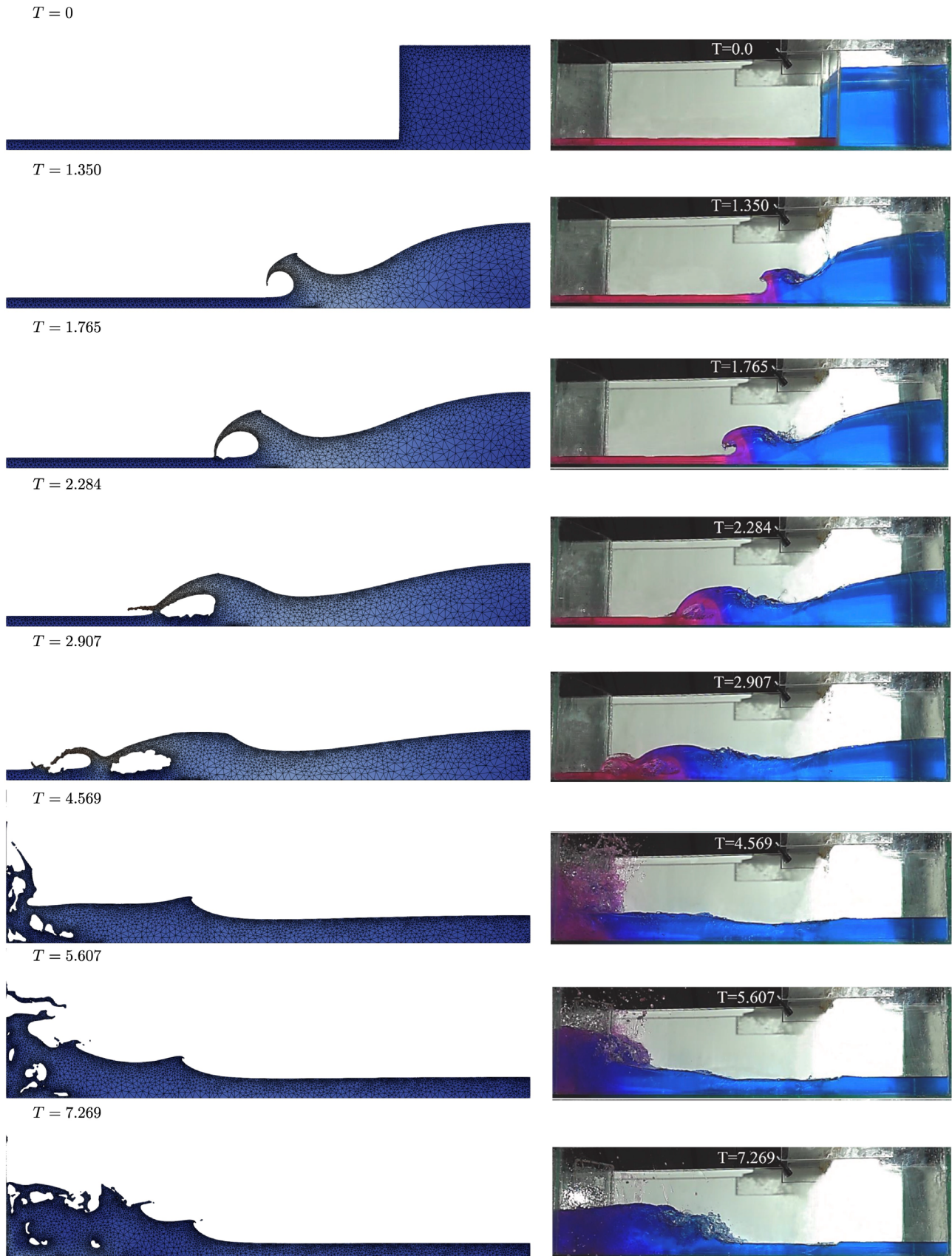


FIGURE 26 Different time shots of the dam break over a wet bed simulation. Qualitative comparison with experiments from Reference 27.

We compare these results with two references. The first is a two-phase volume of fluid approach, performed by Garoosi et al.,²⁵ and the second is experimental data obtained by Lobovsky et al.²⁶ It is important to note that the experimental data available stop relatively soon after the first wave hits the right wall, but this event nevertheless matches well with our numerical simulation. Finally, in Figure 25 we show that, over the course of the simulation, the relative volume variation does not exceed 2%.

4.3 | Dam break over a wet bed

This final simulation presents a water column collapse on a bed that has an initial height of water, as is usually the case on a river bed. The dynamics of this flow are considerably different to the dry bed case. Indeed, due to the presence of water downstream, the initial wave is slowed down, and a mushroom-like jet structure forms and evolves. This wave front then collapses onto the bed, and subsequent jets rebound on the surface and propagate, creating entrapped air cushions. These air cushions are an important part of the flow, because they are responsible for a recirculating vortex to form and cause mixing within the flow. This again highlights the interest of the incompressibility boundary condition around bubbles presented in Section 2.5.2. This phenomenon has been experimentally observed, for example in References 27 and 28.

Figure 26 presents a few time shots of the simulation, and are compared with experimental results from Reference 27. Although the flow is in reality three-dimensional, the 2D simulation performed with the PFEM solver matches quite nicely the experimental results. The formation of the air cushions is clearly visible, and these bubbles remain in the flow until hitting the wall, which generates high levels of splashing.

5 | CONCLUSION

This paper presented two new contributions to the particle finite element method for the simulation of free surface flows. The first, focusing on the re-meshing step, is an algorithm that adapts the mesh with a guarantee of quality improvement for better representation of domain geometry changes. The second is a boundary condition around cavities within the fluid for the simulation of bubbly flows in a single-phase flow setting.

In the present formulation of the PFEM, due to constant remeshing of the particles, the shape of the domain must be redefined at each time step. The α -shape of a set of points, commonly used in PFEM solvers, can by itself lead to domain detection errors. Hence, to ensure continuity of the shape of the fluid between time steps, and to avoid artificial variations of the volume, it is essential to adapt the mesh, and ensure high quality of the elements within the fluid domain. The strategy we have proposed in this work for mesh adaptation, by inserting particles at element circumcenters, guarantees an improvement of the quality of the elements. The algorithm, based on Chew's approach for adaptive Delaunay refinement, not only guarantees the elements within the domain meet a certain quality criterion, but also allows to refine the elements based on a predefined size field. This adaptive meshing approach thus allows to have an efficient resolution of the problem, while having sufficient accuracy in the regions of interest. Essentially, the proposed refinement strategy is an improvement to classical node insertions performed in PFEM simulations by giving a guarantee to improve the mesh quality, with little or no additional costs. Local mesh adaptations are efficient in 2D, the total re-meshing procedure, which includes the triangulation, computing the α -shape, and improving the mesh quality taking roughly 7% of computational costs over a time step.

The boundary condition around cavities allows to simulate bubbly flows while only considering one fluid. The appearance of bubbles is almost systematic in highly turbulent free surface flows, and standard one-phase approaches do not take these features of the flow into account. By imposing a pressure along this surface that preserves the incompressibility, the buoyancy and the surface tension of the bubble, we have shown that it is possible to simulate the presence of bubbles within a single-phase model.

Simulations show good agreement with experimental, analytical and simulation results available in the literature. In the two dam break simulations, the first over a dry bed and the second over an initially wet bed, our PFEM approach accurately captures the main features of these particularly transient free surface flows.

Future prospects include an update to the size field, by including, for instance, solution-based error estimations. This will allow to better capture the main features of the flows by refining the solution in regions that include high velocity gradients. Furthermore, analyzing the sensitivity of numerical parameters such as the choice of the α -shape limit value

will add more consistency to the simulations. Finally, the main topic of interest will be the generalization of the mesh adaptation algorithm to three dimensions.

ACKNOWLEDGMENTS

Thomas Leyssens is a Research Fellow of the F.R.S.-FNRS.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Thomas Leyssens  <https://orcid.org/0009-0002-9954-9062>

REFERENCES

- Idelsohn SR, Oñate E, Del Pin F. The particle finite element method: a powerful tool to solve incompressible flows with free-surfaces and breaking waves. *Int J Numer Methods Eng*. 2004;61(7):964-989.
- Cerquaglia M-L, Thomas D, Boman R, Terrapon V, Ponthot J-P. A fully partitioned Lagrangian framework for FSI problems characterized by free surfaces, large solid deformations and displacements, and strong added-mass effects. *Comput Methods Appl Mech Eng*. 2019;348:409-442.
- Franci A, Cremonesi M, Perego U, Crosta G, Oñate E. 3D simulation of Vajont disaster. Part 1: Numerical formulation and validation. *Eng Geol*. 2020;279:105854.
- Oñate E, Franci A, Carbonell JM. A particle finite element method for analysis of industrial forming processes. *Comput Mech*. 2014;54:85-107.
- Monaghan JJ. Smoothed particle hydrodynamics. *Rep Prog Phys*. 2005;68(8):1703.
- De Vaucorbeil A, Nguyen VP, Sinaie S, Jian Ying W. Material point method after 25 years: theory, implementation, and applications. *Adv Appl Mech*. 2020;53:185-398.
- Hughes TJR, Liu WK, Zimmermann TK. Lagrangian-Eulerian finite element formulation for incompressible viscous flows. *Comput Methods Appl Mech Eng*. 1981;29(3):329-349.
- Cremonesi M, Franci A, Idelsohn S, Oñate E. A state of the art review of the particle finite element method (PFEM). *Arch Comput Methods Eng*. 2020;27:1709-1735.
- Oñate E, Idelsohn SR, Del Pin F, Aubry R. The particle finite element method—an overview. *Int J Comput Methods*. 2004;1(2):267-307.
- Marot C, Pellerin J, Remacle J-F. One machine, one minute, three billion tetrahedra. *Int J Numer Methods Eng*. 2019;117(9):967-990.
- Cremonesi M, Ferrara L, Frangi A, Perego U. Simulation of the flow of fresh cement suspensions by a Lagrangian finite element approach. *J Non-Newtonian Fluid Mech*. 2010;165(23-24):1555-1563.
- Chew LP. Guaranteed-quality mesh generation for curved surfaces. *Proceedings of the Ninth Annual Symposium on Computational Geometry*. ACM; 1993:274-280.
- Shewchuk JR. Delaunay refinement algorithms for triangular mesh generation. *Comput Geom*. 2002;22(1-3):21-74.
- Falla R, Bobach B-J, Boman R, Ponthot J-P, Terrapon VE. Mesh adaption for two-dimensional bounded and free-surface flows with the particle finite element method. *Comput Part Mech*. 2023;10(5):1049-1076.
- Constant M, Dubois F, Lambrechts J, Legat V. Implementation of an unresolved stabilised FEM–DEM model to solve immersed granular flows. *Comput Part Mech*. 2019;6:213-226.
- Hughes TJR, Franca LP, Balestra M. A new finite element formulation for computational fluid dynamics: V. Circumventing the Babuška-Brezzi condition: A stable Petrov-Galerkin formulation of the stokes problem accommodating equal-order interpolations. *Comput Methods Appl Mech Eng*. 1986;59(1):85-99.
- Edelsbrunner H, Kirkpatrick D, Seidel R. On the shape of a set of points in the plane. *IEEE Trans Inf Theory*. 1983;29(4):551-559.
- Geuzaine C, Remacle J-F. GMSH: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *Int J Numer Methods Eng*. 2009;79(11):1309-1331.
- Cerquaglia M-L, Deliége G, Boman R, Terrapon V, Ponthot J-P. Free-slip boundary conditions for simulating free-surface incompressible flows through the particle finite element method. *Int J Numer Methods Eng*. 2017;110(10):921-946.
- Cremonesi M, Meduri S, Perego U. Lagrangian–Eulerian enforcement of non-homogeneous boundary conditions in the particle finite element method. *Comput Part Mech*. 2020;7:41-56.
- Wu GX, Eatock Taylor R, Greaves DM. The effect of viscosity on the transient free-surface waves in a two-dimensional tank. *J Eng Math*. 2001;40:77-90.
- Hysing S, Turek S, Kuzmin D, et al. Quantitative benchmark computations of two-dimensional bubble dynamics. *Int J Numer Methods Fluids*. 2009;60(11):1259-1288.
- Marchandise E, Geuzaine P, Chevaugeon N, Remacle J-F. A stabilized finite element method using a discontinuous level set approach for the computation of bubble dynamics. *J Comput Phys*. 2007;225(1):949-974.
- Franci A, Cremonesi M. On the effect of standard PFEM remeshing on volume conservation in free-surface fluid flow problems. *Comput Part Mech*. 2017;4(3):331-343.

25. Garoosi F, Merabtene T, Mahdi T-F. Numerical simulation of merging of two rising bubbles with different densities and diameters using an enhanced volume-of-fluid (VoF) model. *Ocean Eng.* 2022;247:110711.
26. Lobovsky L, Botia-Vera E, Castellana F, Mas-Soler J, Souto-Iglesias A. Experimental investigation of dynamic pressure loads during dam break. *J Fluids Struct.* 2014;48:407-434.
27. Garoosi F, Mellado-Cusicañhua AN, Shademani M, Shakibaeinia A. Experimental and numerical investigations of dam break flow over dry and wet beds. *Int J Mech Sci.* 2022;215:106946.
28. Jánosi IM, Dominique Jan K, Szabó G, Tél T. Turbulent drag reduction in dam-break flows. *Exp Fluids.* 2004;37:219-229.

How to cite this article: Leyssens T, Henry M, Lambrechts J, Remacle J-F. A Delaunay refinement algorithm for the particle finite element method applied to free surface flows. *Int J Numer Methods Eng.* 2024;e7554. doi: 10.1002/nme.7554