Computational Particle Mechanics manuscript No.
(will be inserted by the editor)

Mesh adaption for two-dimensional bounded and free-surface flows with the Particle Finite Element Method

Romain Falla · Billy-Joe Bobach · Romain Boman · Jean-Philippe Ponthot · Vincent E. Terrapon

Received: date Accepted: date

Abstract The particle finite element method (PFEM) is a Lagrangian method that avoids large mesh distortion through automatic remeshing when the computational grid becomes too distorted. The method is well adapted for flows with deforming interfaces and moving boundaries. However, the α-shape technique used to identify these boundaries presupposes a mesh of approximately uniform size. Moreover, the α-shape criterion is purely geometric and, thus, leads to violations of mass conservation at boundaries. We propose a new algorithm for mesh refinement and adaptation in two dimensions to improve the ratio accuracy to computational cost of the PFEM. A local target mesh size is prescribed according to geometric and/or physics-based criteria and particles are added or removed to approximately enforce this target mesh size. Additionally, the new boundary recognition algorithm relies on the tagging of boundary nodes and a local α-shape criterion that depends on the target mesh size. The method allows thereby reducing mass conservation errors at free surfaces and improving the local accuracy through mesh refinement, and simultaneously offers a new boundary tracking algorithm. The new algorithm is tested on four two-dimensional validation cases. The first two cases, i.e., the lid-driven cavity flow at Reynolds number 400 and the flow around a static cylinder at Reynolds numbers below 200, do not feature a free surface and mainly illustrate the mesh refinement capability. The last two test cases consist in the sloshing problem in a reservoir subjected to forced oscillations and the fall of a 2D liquid drop into a tank filled with the same viscous fluid. These last two cases demonstrate the more accurate representation of the free surface and a corresponding reduction of the error in mass conservation.

Keywords particle finite element method (PFEM) · free-surface flow · mesh adaptation · boundary recognition · α-shape technique · mass conservation

Aerospace & Mechanical engineering, University of Liège, Belgium
E-mail: romain.falla@uliege.be
E-mail: vincent.terrapon@uliege.be

1 Introduction

The Particle Finite Element Method (PFEM) [15,47] is a numerical approach that relies on a mesh-based finite-element formulation of the governing equations for the simulation of multi-physics problems in deforming domains. In contrast to classical Eulerian finite-element methods (FEM) used in computational fluid dynamics (CFD), the mesh nodes behave as particles and move with the material. A new computational grid is generated when the mesh becomes too distorted using a fast Delaunay triangulation [19,41,49]. Additionally, the domain boundaries are identified from the full convex triangulation by means of geometric criteria using the α-shape technique [25,26,47].

Owing to its Lagrangian nature, the PFEM is well suited for the simulation of flows involving deforming interfaces, moving bodies and fluid structure interaction. For instance, Oñate et al. studied bed erosion [46], Cerquaglia et al. focused on fluid structure interaction problems with strong added mass effects [10], Franci et al. studied free-surface Bingham fluid flows interacting with structures [29]. Other examples include the study of landslides by Cremonesi et al. [14] and Zhang et al. [67] or the numerical analysis of fluid-saturated porous media by Monforte et al. [44]. The PFEM has also been applied to solid mechanics problems, such as the simulation of metal cutting [7, 50] and the numerical modeling of granular flows [23], but the present work focuses on the simulation of incompressible fluid flows.

In the specific context of fluid mechanics, the PFEM combines the robustness of the finite-element formulation and the versatility of a Lagrangian description to treat moving boundaries. However, the numerical solution of such complex problems always requires a trade-off between accuracy and computational cost. It is thus important to rely on an efficient implementation. In this context, mesh adaptivity represents a powerful strategy to increase accuracy for a given computational cost, or equivalently, to reduce the cost for a given accuracy.

Mesh adaptivity essentially refers to the use of a discretization based on mesh elements of different sizes, such that high resolution is used in regions where the solution features require it, while coarser elements are used where the solution is smooth. This strategy has been successfully used for decades in Eulerian mesh-based methods for either fixed meshes or meshes that evolve during the simulation. At the most basic level, adaptivity can simply be achieved by defining *a priori* a fixed but non-uniform mesh. In this case, the characteristics of the mesh are usually based on user expertise and/or some mesh convergence study. At a more advanced level, iterative or time-dependent solution-based mesh adaptation relies on the local solution to define the local mesh density. This is often referred to as Adaptive Mesh Refinement (AMR) and is typically used with unstructured meshes or octree (quadtree in 2D) grids [3,30,39,61].

AMR consists of two steps: i) the identification of the mesh elements that need to be adapted, and ii) the corresponding dynamic adaption of these elements, typically through splitting or grouping. The identification of the elements to refine or coarsen is usually based on some solution-dependent quantity and corresponding threshold values. As suggested by Bansch et al. [24], such mesh adaptation techniques can be divided into two categories. The error-based methods use an estimate of the interpolation error (see [18,22,24] for some examples). The other category encompasses the heuristic methods in which the elements are refined following physical criteria [6]. For instance, the refinement can be applied to areas of large gradients [17,42]. This second category of methods is well suited for highly nonlinear problems, such as computational fluid dynamics (CFD) and fluid-structure interaction (FSI), where error estimates are either unavailable or expensive to compute [2]. Additionally, geometric criteria can also be used, for instance based on the distance to solid surfaces when bodies are moving.

While splitting or grouping cells is relatively straightforward in the context of octree grids, refining and coarsening an unstructured mesh without hanging nodes often requires to modify not only the targeted element itself but also its neighbors. This adaptation can be isotropic or in specific directions. Beyond the geometric aspects of defining new, coarser or finer, elements, the main challenge is to assign them a solution value. In particular, interpolation of new values should ensure that conservation principles are satisfied and that no large error is introduced.

Despite the vast amount of literature on mesh adaptation and its common use in Eulerian methods, particle methods still mostly rely on meshes with a uniform resolution. A few recent exceptions can be found in the context of the Smooth Particle Hydrodynamics (SPH) method, such as Vacondio et al. [60], Yang et al. [65] or Sun et al. [56]. Regarding the PFEM, the recent review of Cremonesi et al. [15] only reports two examples of mesh adaptation, both using error-based refinement but only applied to solid mechanics problems [7,50]. To the authors' best knowledge, the PFEM with physics-based mesh adaption has yet to be applied to fluid mechanics

problems. The ability to adapt, possibly dynamically, the mesh resolution to the solution is particularly crucial to simulate flows with viscous boundary layers, recirculation zones, interfaces, free-surface, etc. The current lack of robust and efficient mesh adaptivity capability thus considerably limits the range of possible applications that can be simulated with the PFEM.

This shortcoming of the PFEM can be explained by two major challenges originating in the fundamental nature of the method. First, unlike Eulerian methods, the PFEM, and more generally particle methods, are characterized by a set of nodes that constantly evolve along the simulation. More precisely, the nodes are material points and thus move according to the equations of motion in a Lagrangian fashion. In the context of fluid flows, each node moves following the flow, even when the velocity field itself is steady. In other words, although the resolution could be defined *a priori* such that it is initially best adapted for the specific configuration studied, the displacement of the nodes by the flow would completely randomize the resolution, leaving sparse regions where high resolution is required, and conversely. Adaptivity can thus only be achieved by constantly creating and deleting particles.

The second reason is very specific to the PFEM as it is directly related to the boundary recognition algorithm. In particular, the classical version of the α-shape technique typically used in the PFEM presupposes a uniform mesh [16,47], as discussed in more detail in the next section.

Our objective is therefore to develop an algorithm for mesh adaptivity and to implement it into an existing PFEM solver for incompressible flows. Note that only two-dimensional meshes are considered here. The goal is to efficiently capture important flow features found around moving bodies.

The proposed approach consists first in identifying the elements to adapt by comparing the actual mesh size to a target mesh size defined at each node. This target mesh size is determined based on a combination of heuristic solution-based and geometric criteria. The adaptation is then carried out through the addition / deletion of one or more nodes inside, and possibly around, the corresponding mesh elements. The solution value at the newly defined nodes is subsequently obtained from interpolation with neighbor values using the shape functions associated with the element's nodes. Finally, the domain boundary identification relies on both node tracking and the α-shape technique, but based on *local* parameters. Both the identification and the refinement of the moving boundaries are considered simultaneously in order to minimize the error in mass (volume) conservation stemming from remeshing and time integration.

The present article is organized as follows. After this introduction, the next section provides a more detailed description of the PFEM. In particular, it illustrates some of the limitations of the general method and highlights the challenges linked to mesh adaptivity. Section 3 describes in detail the proposed mesh refinement approach, considering successively the definition of the target mesh size, the adaptation of the mesh and the boundary recognition algorithm. The mesh refinement algorithm and its implementation are then assessed in section 4 through several two-dimensional test cases, including the lid-driven cavity flow, the uniform flow around a static cylinder, the forced sloshing of a reservoir and the fall of a liquid drop into a pool of the same liquid. This is followed by a brief discussion on the extension of the algorithm to three-dimensional tetrahedral meshes in section 5. Finally, the work and the main results are summarized in the conclusion.

2 The Particle Finite Element Method

The Particle Finite Element Method has been originally proposed by Oñate & Idelsohn [47] and more recent developments can be found in the review article of Cremonesi et al. [15]. The present work is based on the PFEM implementation of Cerquaglia [8,9,10,11,12]. This section first introduces the finite element formulation of the basic equations and the α-shape technique. The challenge of mass conservation is then briefly discussed.

2.1 Basic equations and finite-element formulation

The two-dimensional Navier-Stokes equations for an isothermal incompressible fluid are discretized using a finite element formulation. After integration by parts of second derivatives, the momentum and continuity equations respectively read

$$\mathbf{M}\frac{D\mathbf{v}}{Dt} + \mathbf{K}\mathbf{v} + \mathbf{D}^T\mathbf{p} = \mathbf{f}\,, \tag{1}$$

$$\mathbf{D}\mathbf{v} = 0\,, \tag{2}$$

where $\mathbf{v}$ and $\mathbf{p}$ are the vectors containing the two velocity components and the pressure fields, $\mathbf{M}$ is the mass matrix, $\mathbf{K}$ contains the viscous terms, $\mathbf{D}$ is the discrete version of the divergence operator, $(\cdot)^T$ is the transpose operator, D/Dt is the material derivative and $\mathbf{f}$ is a vector containing the contribution of body forces

(typically gravity) and surface tractions (e.g., applied forces, surface tension, etc.).

The discretization relies on linear shape functions for both the pressure and velocity fields, thus violating the Ladyzhenskaya-Babuška-Brezzi (LBB) condition [5,51]. To avoid pressure instabilities, a monolithic Pressure-Stabilizing Petrov-Galerkin (PSPG) approach is used, which consists in adding stabilization terms to the mass conservation equation [38,58]. This leads to

$$\mathbf{M}\frac{D\mathbf{v}}{Dt} + \mathbf{K}\mathbf{v} + \mathbf{D}^T\mathbf{p} = \mathbf{f}\,, \tag{3}$$

$$\mathbf{C}\frac{D\mathbf{v}}{Dt} - \mathbf{D}\mathbf{v} + \mathbf{L}_\tau\mathbf{p} = \mathbf{h}\,, \tag{4}$$

where $\mathbf{C}$ is a dynamic stabilization term, $\mathbf{L}_\tau$ is the discretized version of a stabilization Laplacian operator and $\mathbf{h}$ results from the presence of the body forces in the stabilization term [11]. Note that the stabilization term vanishes as the iterations converge towards the correct solution.

Finally, the system is integrated in time using an implicit Euler scheme to obtain the velocity field $\mathbf{v}^{n+1}$ at the next time step. This velocity is then used to determine the new particle locations $\mathbf{x}^{n+1} = \mathbf{x}^n + \mathbf{v}^{n+1}\Delta t$ at time step $t^{n+1} = t^n + \Delta t$. The iteration over the system nonlinearity relies on a Picard (or fixed-point) algorithm.

2.2 Meshing and α-shape technique

Because the particles move with the flow, the initial mesh is rapidly distorted. In the PFEM, the distorted mesh used at the previous time step is thus discarded and a new mesh is constructed from the cloud of particles at their new position. The process is illustrated in Fig. 1: Given a cloud of particles (a), a classical Delaunay triangulation is performed to define the new triangular mesh (b). Nonetheless, the entire convex hull defined by the particles is triangulated. In order to identify the actual boundary of the fluid, some elements must be eliminated (c). In the PFEM, this is traditionally achieved through a geometric criterion based on the α-shape technique [47].

Originally, the α-shape technique has been developed as a tool to separate subgroups of simplices contained in the convex hull of a point set [25,26]. It simply consists in removing the triangles having a circumscribed circle whose radius r_c is larger than a threshold α. Because otherwise the threshold α would be problem-dependent, it is customary in the PFEM to use this criterion in its non-dimensional form

$$\frac{r_c}{h} > \alpha\,, \tag{5}$$

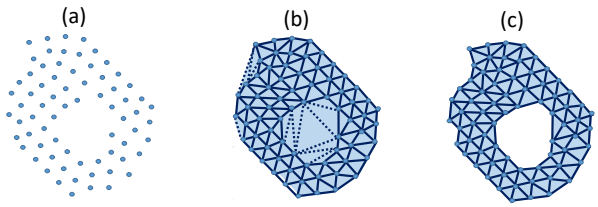


Fig. 1: Illustration of the meshing procedure and α-shape technique. (a) Cloud of particles obtained from the previous time step, (b) mesh over the convex hull of the particle cloud after Delaunay triangulation, (c) final mesh after elimination of the elements that are too elongated (i.e., with an α-value larger than the threshold). The deleted triangles are considered as empty space.

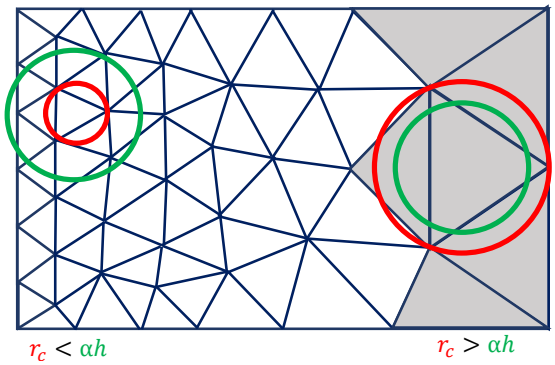


Fig. 2: Non-uniform mesh with a size progression from left to right. The red circles represent schematically the circumscribed circle of the corresponding triangle with radius r_c and the green circles the threshold αh of Eq. (5) using a constant value of h and α. Applying the criterion would wrongfully eliminate all triangles that are "large", i.e., the grey triangles in this figure, despite their regular shape.

where h is a characteristic length scale of the mesh. Typically, h is defined as the average length of the smallest element edge over the mesh [11,16]. Choosing a global value α between 1.2 and 1.5 allows discarding triangles that are too badly shaped or too large.

The drawback of the above method is that it presupposes some uniformity of the mesh size, preventing thus its direct use with mesh adaptation. More specifically, applying the above α-shape technique to a non-uniform mesh would eliminate all triangles that are too large even if they are equilateral and belonging to the fluid domain, as illustrated in Fig. 2. Increasing the value of h would remedy it, but then smaller badly-shaped elements that should be eliminated would be incorrectly kept.

To circumvent this issue, one could replace the global length scale h by a local value h_e attached to the element e, e.g., the shortest length of the corresponding triangle. Although this approach would eliminate badly-shaped triangles, some larger elements

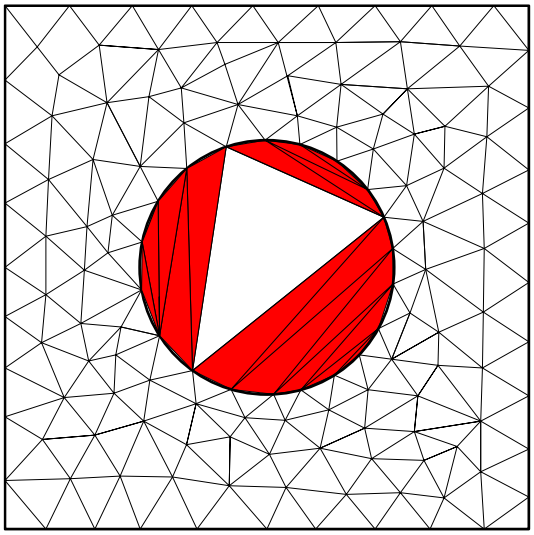


Fig. 3: Mesh obtained after Delaunay triangulation for a cylindrical solid body immersed in a fluid. The use of a local scaling length h_e instead of a global measure h of the mesh size in the α-shape technique would correctly eliminate the badly-shaped elements (in red) but would fail to discard the large regular triangle inside the solid body.

would be wrongly kept, as illustrated in Fig. 3. In this case, the large triangle inside the cylinder is not discarded and the algorithm fails to identify the cylinder immersed in the fluid as a solid body. Nonetheless, such an approach could be considered if the local value h_e is understood as an average over some local region rather than directly related to the corresponding element, and if only smooth spatial variations of h_e are permitted. A related method is proposed in the present work, as described in Section 3.

Another option is to rely on a generalization of the α-shape technique, the so-called weighted α-shape method [4,26,48], in conjunction with weighted Delaunay triangulation (also referred to as regular triangulation) [27]. In this case, large triangles or small badly-shaped triangles can be simultaneously discriminated by introducing user-defined weights at each node. Note that it can be shown that the aforementioned approach based on a local length scale h_e is equivalent to the weighted α-shape method under the assumption that the weights vary smoothly in space. Nevertheless, the weighted α-shape method is still a geometric criterion and special treatment is required in the vicinity of the free-surface to minimize the mass creation or destruction that is inherent to the PFEM.

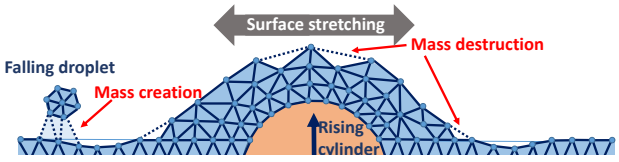


Fig. 4: Typical examples of error in mass conservation at a free surface: mass creation due to two free surfaces approaching each other (falling droplet on the left) and mass destruction due to the stretching of the free surface (above the rising cylinder in the middle).

2.3 Mass conservation errors

Because the α-shape technique used to identify the boundary of the fluid domain is not based on physics, mass conservation cannot be rigorously enforced in the PFEM, as it is also the case with some other methods used for simulating free-surface or multiphase flows. For incompressible flows, the total mass is directly proportional to the volume of the fluid. Local volume creation or destruction typically occurs at a free-surface, both during numerical time integration (displacement of boundary nodes) and remeshing (deletion of existing or addition of new elements). The change in total volume can thus be decomposed into two contributions, $\Delta V = \Delta V_{\text{num}} + \Delta V_{\text{rem}}$, respectively.

As an example, volume creation/destruction due to remeshing can take place when two free surfaces, or two parts of the same free surface, approach each other, as illustrated by the falling droplet merging with the bulk of the fluid in Fig. 4. When the distance between the two boundaries becomes of the order of the mesh size, the elements between them are not discarded by the boundary identification algorithm, thus creating mass. On the other hand, mass can be destroyed along a free surface that is stretched, as also illustrated in Fig. 4. In this case, the stretching pulls boundary nodes apart leading to obtuse triangles that are eliminated by the α-shape criterion. Similar errors in mass conservation can also be observed around the contact point of a free surface and a solid wall (see Fig. 5), where mass is created at the front of the moving fluid wetting the surface and destroyed at its back.

Two approaches can be considered to limit mass creation/destruction inherent to the PFEM. On the one hand, one can locally refine the mesh in the critical regions of the free surface, which provides another motivation for mesh adaptation. On the other hand, a different treatment can be applied to the nodes on the free surface during the boundary recognition step. Nevertheless, this requires some kind of tracking of the free surface, e.g., by tagging the nodes belonging to it.

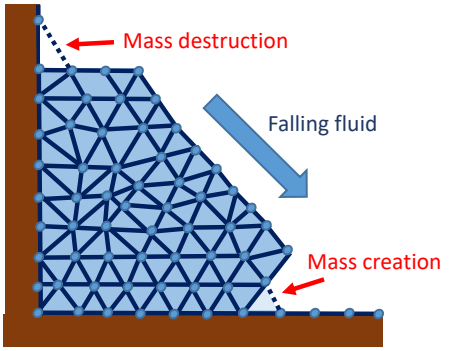


Fig. 5: Typical examples of errors in mass conservation at a solid wall: mass creation at the fluid front along the horizontal surface and mass destruction behind the falling fluid on the vertical wall.

Both approaches are considered in the present work, as explained in the following section.

3 Mesh adaptation algorithm

The objective of the proposed algorithm is to adapt the mesh resolution so as to accurately capture the flow features and decrease mass creation/destruction, while minimizing the computational cost. In particular, we want to dynamically refine the mesh in the viscous layer close to solid surfaces, in the wake of solid bodies, at free surfaces undergoing deformation and in regions of possible flow separation or reattachment.

The main idea is to define a local target mesh size L^* that varies smoothly between a minimum and maximum value across the fluid domain, and to approximately enforce it through creation or destruction of particles where the actual mesh size differs from the target value. This step occurs at the end of each time step, after the nodes have been moved according to their computed velocity. Once new particles have been added and some existing particles have been potentially removed, the old mesh is discarded and a new one is created using Delaunay triangulation. This is followed by the boundary identification step that is based on a boundary tracking approach in conjunction with a local α-shape technique.

This section describes the definition of the target mesh size and the process of particle creation/destruction. Finally, the tracking of the free surface and the special treatment of elements at the boundary is explained.

3.1 Definition of the target mesh size

The local target mesh size $L^* = \sqrt{A^*}$ is a measure of the desired surface area A^* of the mesh elements. Two approaches are combined to define it. First, geometric mesh refinement is based on the absolute position of the particles in the computational domain and/or on their relative position with respect to solid surfaces. While this type of criterion is adequate in many cases, the actual location of viscous layers is not always known a priori. For instance, the trajectory of a buoyant object is a result of the simulation, so that the position of its wake cannot be easily defined purely geometrically. Therefore, a solution-based mesh refinement is also considered, which relies on a measure of the local velocity gradients.

Note that, because the information about elements is lost at each time step when the mesh is discarded, the local target mesh size L^* is calculated and stored at each node n. It is also recomputed at each time step to accommodate for the displacement of the nodes.

3.1.1 Geometric criteria

Viscous layers adjacent to walls are characterized by large velocity gradients. A finer mesh is thus required close to solid boundaries to accurately capture these gradients. The target mesh size at a given location can thus be defined as a continuous function $L^*(d)$ of the shortest distance d to solid walls, whether fixed boundaries or moving bodies. Different analytical functions can be considered. For instance, $L^*(d)$ could correspond to the smallest mesh size L^*_{min} in some region adjacent to a wall, then increase linearly with d to the maximum mesh size L^*_{max} over some prescribed distance and finally remain constant at L^*_{max} further away. The wake region behind a body also typically requires a finer mesh. If the location of the wake is known beforehand, the target mesh size can be prescribed through pseudo geometrical entities defined in either an absolute frame of reference or relative to the body, for instance. Such pseudo-entities could be a rectangular zone or a center line behind the body. Another option could be to define the target mesh size as an analytical function of both the distance and angle with respect to the body.

For complex geometries the calculation of the exact minimal distance to a wall is not trivial and potentially expensive from a computational point of view. In practice it is therefore more convenient to consider each (pseudo-) geometric entity k separately. This leads to several criteria L^*_k. Note that each criterion can use different values for L^*_{min} and L^*_{max}. The actual target size can then be defined as the minimum over the different criteria:

$$L^* = \min_k L^*_k \,. \tag{6}$$

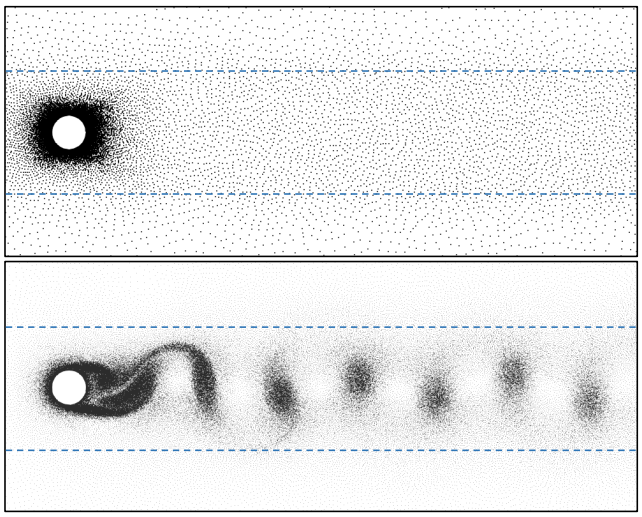


Fig. 6: Non-uniform meshes for the simulation of the flow around a fixed cylinder at $\mathrm{Re}_D = 200$. Only the nodes are shown, and each image corresponds to a specific instant in time after a limit cycle oscillation has been reached. The first mesh (top) uses only geometrical mesh refinement based on the distances from the cylinder and from the center line. The second mesh (bottom) also uses this refinement but, in addition, includes a solution−based refinement where the target mesh size is given by Eq. (7) with the parameter $\beta = \frac{1}{3}$. The dashed blue lines represent the rectangular area for geometric refinement. Note that for better visualization of the most refined regions, the point size is twice smaller in the bottom image.

An example is shown in the top image of Fig. 6 for the simulation of the flow around a fixed cylinder. In this case, the target mesh size is set based on the distance to the cylinder and a rectangular zone downstream of it.

Additionally, complex boundaries or bodies can be approximated through simpler geometric shapes (e.g., line, circle, rectangle) to further simplify the definition of the target mesh size. This also provides the opportunity to implement generic criteria that can be used for different configurations. Nevertheless, an adaptation for each specific case is usually still required. The complexity also increases with the number of geometrical elements considered. An additional drawback is that regions of large gradients sometimes result from the flow dynamics and are thus unknown before the computation.

3.1.2 Solution-based refinement

To avoid the need for an a priori knowledge of the flow features, it is helpful to combine geometric criteria with solution-based mesh refinement. While different metrics can be considered, the present work relies on the Froebenius norm of the local velocity gradient tensor, $||\nabla \mathbf{u}||$. The goal is to prescribe a small mesh size in regions with large velocity gradients, and vice versa.

In the proposed approach, the target mesh size is based on a linear interpolation of a negative power of $||\nabla \mathbf{u}||$,

$$L^* = L^*_{\min} + (L^*_{\max} - L^*_{\min}) \times \max\left(\min\left(\frac{||\nabla \mathbf{u}||^{-\beta} - ||\nabla \mathbf{u}||^{-\beta}_{\max}}{||\nabla \mathbf{u}||^{-\beta}_{\min} - ||\nabla \mathbf{u}||^{-\beta}_{\max}}, 1\right), 0\right) \quad (7)$$

where β, $||\nabla \mathbf{u}||_{\min}$ and $||\nabla \mathbf{u}||_{\max}$ are user-defined parameters. The parameter β, usually between zero and one, controls the rate of increase of the mesh size, while the two other parameters are threshold values that can be estimated beforehand based on characteristic length and velocity scales of the problem considered or known solutions of similar problems. The rationale behind Eq. (7) is to retrieve a classical grid stretching for a wall boundary layer, but other functional forms could also be considered.

An example of this technique is shown in the bottom image of Fig. 6, where, in addition to the previously mentioned geometrical refinement, a solution-based refinement is used. In this case, a small target mesh size is imposed, in accordance with Eq. (7), in regions of large gradients such as in the wake of the cylinder. In particular, a high grid resolution can be seen in and around the von Karman vortices and shear layers. Such grid refinement allows a more accurate prediction of drag, lift and Strouhal number, as discussed in more details in section 4.

In flows where features of interest have widely different levels of velocity gradient magnitude, Eq. (7) might not provide an adequate target mesh size. For instance corner vortices might be much weaker than other flow features but still important to capture. In this case, it is better to replace $||\nabla \mathbf{u}||$ in the above expression by a normalized measure of the velocity gradient magnitude, e.g.,

$$||\widetilde{\nabla \mathbf{u}}|| = \frac{||\nabla \mathbf{u}||}{||\mathbf{u} - \mathbf{U}_\infty|| + U_\epsilon}, \quad (8)$$

where U_ϵ is a small user-chosen velocity to avoid division by 0, and $\mathbf{U}_\infty$ is the constant free-stream velocity. Note that other options can be considered depending on the case of interest.

For problems involving free-surface deformations, a finer mesh is also needed to accurately capture the dynamics and location of free surfaces. Therefore, we also rely on a solution-based measure related to the free-surface deformation. In particular, the local target mesh size is prescribed as a linear function of the free-surface

radius of curvature r_s to ensure an approximately constant angular mesh size:

$$L^* = \min\left(\max\left(\frac{\pi r_s}{m}, L^*_{\min}\right), L^*_{\max}\right), \quad (9)$$

where m is a user-defined parameter that controls the angular mesh resolution, i.e., the number of edges over a half-circle. The radius of curvature itself is obtained as the radius of the circle passing through the corresponding boundary node and its two direct neighbors on the free surface. The target mesh size of neighbor nodes in the fluid bulk can be prescribed in a second step using the smoothing algorithm introduced in the next subsection.

Finally, it should be mentioned that such solution-based approaches, or similar ones, can partly replace and/or complement some of the aforementioned geometric criteria. Equation (6) is then used to prescribe the actual local target mesh size. This is illustrated in Fig. 6, where the geometric criteria are combined with the solution-based refinement approach of Eq. (7) to better resolve the von Karman vortices.

3.1.3 Smoothing of the target mesh size

For numerical accuracy of the discretization, but also for applying the α-shape technique locally, it is important to ensure a certain smoothness of the mesh size. In particular, small and large elements should not be directly adjacent to each other. This implies that the target mesh size should also be sufficiently smooth. While rapid variations of L^* can be easily avoided for geometric criteria, this is less trivial for solution-based approaches. If variations in space and/or time of velocity gradients are too rapid, discontinuities in the target mesh size can occur. The refinement of the free surface is another example because the curvature-based target mesh size can only be prescribed for the boundary nodes, and not for their direct and more distant neighbors.

A smoothing algorithm is thus subsequently applied to the target mesh size, so as to enforce a maximum target mesh size ratio p_r between two neighbors. More specifically, if the condition

$$\frac{1}{p_r} \leq \frac{L^*_m}{L^*_n} \leq p_r \quad (10)$$

is not satisfied for a node n and its neighbor node m, then the larger target mesh size is reduced to satisfy Eq. (10). Starting from nodes with the smallest target mesh size, the neighbors are, if needed, gradually updated. The process is repeated recursively for all nodes that have been themselves updated. Note that the complexity of the algorithm is of order N, as the above condition is tested for each mesh node a finite number of times that depends only on the number of direct neighbour nodes.

This smoothing step also allows refining the mesh around solid boundaries with a complex shape, in a similar manner as it is done with the curvature-based refinement of the free surface. In this case, the target mesh size can be imposed on the boundary itself and the smoothing algorithm can be leveraged to "propagate", with some progression factor, the target mesh size into the fluid domain.

3.1.4 Delaying mesh coarsening

After a mesh refinement, it might be sometimes advantageous to delay any potential subsequent coarsening. This could be to avoid a repeating cycle of node creation and destruction that would introduce unnecessary numerical dissipation.

Another situation in which delaying mesh coarsening might be required is when the need for a fine mesh is anticipated but the local target mesh size provided by the refinement criteria is larger than required. This is for instance the case during the merging of two fluid regions, as illustrated by the falling drop in Fig. 4. In such a case, one of the regions (e.g., the droplet) might be much more refined than the other (e.g., the liquid bath). As predicted by the α-shape criterion, merging typically occurs when the distance between the two regions is of the same order as the mesh size of the coarser fluid region. Therefore, an anticipatory local mesh refinement around the liquid bath free surface, when the droplet approaches, allows reducing mass creation. To ensure that the mesh resolution remains sufficient before the actual merging, despite a larger calculated target mesh size, the coarsening that would otherwise take place should be delayed for a duration Δt_{delay} until the merging process has started.

This is achieved by assigning to each node a time counter t^* that is incremented at each time step, but reset to zero every time the new calculated target mesh size is smaller than the current one. If the new calculated target mesh size is larger and if $t^* < \Delta t_{\text{delay}}$, then the current target mesh size at the node is not updated and its old value is kept, thus preventing a potential coarsening during the time step. The delay time Δt_{delay} can be imposed globally or locally as a multiple of the time step size depending on the case considered. For instance, just before the merging of two fluid regions, their relative velocity can be used to estimate a local Δt_{delay}. A practical illustration of

the process and the benefit of delaying coarsening is discussed in Section 4.4.

3.2 Creation and destruction of particles

Once a smooth target mesh size has been defined everywhere, it must be enforced up to some tolerance by adapting the actual mesh. In practice, this is achieved by creating/destroying nodes wherever the actual mesh size L is outside a user-defined range around its target value.

As explained in the following section, the comparison between target and actual mesh size is usually performed at the edge or, for elements at the boundary, at the element level. Because the target mesh size L_n^* is initially defined at the nodes n, we also define a target mesh size L_{elem}^* associated with an element and L_{edge}^* with an edge using a simple arithmetic average over the corresponding nodal values,

$$L_{\text{elem}}^* = \frac{1}{3} \sum_{n_{\text{elem}}} L_{n_{\text{elem}}}^* \,, \tag{11}$$

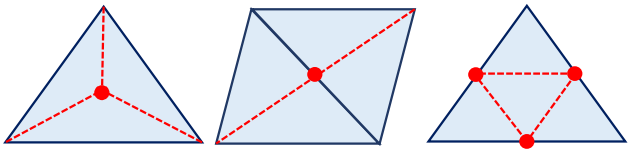
$$L_{\text{edge}}^* = \frac{1}{2} \sum_{n_{\text{edge}}} L_{n_{\text{edge}}}^* \,. \tag{12}$$

where n_{elem} and n_{edge} indicate the corresponding nodes of the element and edge, respectively.

3.2.1 Adding particles

In general, new nodes can be added either within elements (Fig. 7(a)) or on their edges (Fig. 7(b,c)). In the present algorithm, nodes are only added at the mid-point of selected edges to avoid new elements that are too obtuse [33] and a too agressive mesh refinement, i.e., a too large increase in nodal density[1], from one time step to the next. Nonetheless, the addition of nodes in the center of elements could in principle also be considered.

The main idea is to divide elements in the bulk that are too large by a factor of two through the addition of a new node on only one of their two largest edges (Fig. 7(b)), and elements at the boundary by a factor of four through the addition of new nodes on each of their edges (Fig. 7(c)). The special treatment of boundary elements is dictated by the boundary recognition step. In order to correctly identify the boundary using the α-shape technique, and thus reduce the mass conservation error, it is crucial to ensure high-quality boundary elements. This is discussed in more detail in section 3.3. It is also important to emphasize that the new edges (red dashed lines in Fig. 7) are only virtual because the mesh is discarded and a new triangulation is performed right after the node creation/destruction step).



(a) × 3 **(b)** × 2 **(c)** × 4

Fig. 7: Different strategies to add nodes and corresponding increase of the nodal density: one node added at the center of the element (left), one node added at the mid-point of a single edge (middle), nodes added at the mid-point of each edge of an element (right). The new nodes are indicated by red circles and the new virtual edges by red dashed lines.

Concretely, an edge in the bulk is refined by adding a node at its mid-point if the three following conditions are all satisfied:

- the average area of the two elements sharing this edge is larger than the threshold value:

$$\frac{1}{2}(A_{\text{elem}_1} + A_{\text{elem}_2}) > \frac{4}{3} L_{\text{edge}}^{*2} \,, \tag{13}$$

- the edge is not the shortest edge of any of the two elements sharing this edge, and
- the edge has not been previously tagged to prevent its refinement.

The addition of a node on an edge impacts the two adjacent elements sharing this edge. To avoid the further division of these elements through the potential addition of new nodes on their other edges, these other edges are then tagged. This tag prevents their refinement, thereby ensuring that elements in the bulk are at most divided by two. Additionally, if the edge that is refined belongs to a boundary element (i.e., an element with an edge on the boundary), this boundary element is automatically refined by adding nodes on all its edges irrespective of its size (Fig. 7(c)). The rationale is to ensure that the shape of the boundary elements is as optimal as possible to improve the boundary detection in the subsequent step.

On the other hand, if the edge is on the boundary and if the area of the associated boundary element is larger than the threshold value,

$$A_{\text{elem}} > \frac{4}{3} L_{\text{elem}}^{*2} \,, \tag{14}$$

[1] In 2D, the nodal density is defined as the number of nodes per unit area. Because a triangular mesh has on average twice more elements than nodes (away from boundaries), the nodal density can be locally approximated as $1/(2A_{\text{elem}})$, where A_{elem} is the surface area of the element. Therefore, the more new elements are created by the addition of a new node, the smaller their area is and the more the local nodal density increases. As illustrated in Fig. 7, adding one node on a single edge (case b) leads to the smallest increase in nodal density.

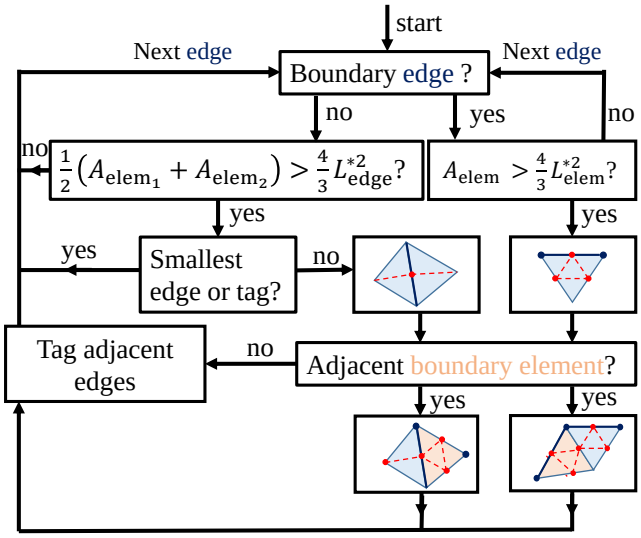


Fig. 8: Algorithm for node addition. The edge considered is indicated by the bold blue line. The blue dots represent boundary nodes while the red dots and the red dashed lines are respectively the new nodes and the new virtual edges if the edge is refined.

then this boundary element is divided by four. This is achieved by adding a new node on each of the edges of that element (Fig. 7(c)). Furthermore, if this boundary element has a direct neighbor (i.e., sharing a common edge) that is also a boundary element (i.e., with an edge on the boundary), this neighbor element is also refined irrespective of its size.

The algorithm is illustrated in Fig. 8. It should be noted that, with this algorithm, the refinement has some dependence on the order in which the edges are considered. This is however not deemed to be a problem, especially since there are anyway many different strategies to refine the mesh. The threshold coefficient 4/3 in Eqs. (13) and (14) is chosen so that, on average, the elements that are refined have afterwards an area larger than $\frac{2}{3}L^{*2}$ in the bulk and $\frac{1}{3}L^{*2}$ at the boundaries, and the unrefined elements an area smaller than $\frac{4}{3}L^{*2}$. Increasing this coefficient reduces the number of nodes added but induces a larger discrepancy between actual and target mesh size, and conversely.

Once nodes have been added, the associated nodal values of physical quantities (e.g., velocity, pressure, etc.) must be prescribed. This initialization uses the interpolation shape function of the finite element discretization. In practice, it implies that each variable is simply obtained using the mean value of the corresponding variable at the nodes of the refined edge. This ensures that the local fields remain unchanged during the process.

3.2.2 Removing particles

For the same reason as mentioned above, node removal is implemented at boundaries differently than in the bulk. In the bulk, the main idea is to "collapse" an element that is too small into a single node at its center. In other words, the nodes of a triangle whose area is smaller than the threshold,

$$A_{\text{elem}} < \gamma L^{*2}_{\text{elem}}\,, \tag{15}$$

are deleted and a new node at the element center is created. The new (virtual) elements have thus all more or less the same size and are larger than the original elements, which ensures a smoother transition between coarse and fine mesh.

This new node is assigned nodal values corresponding to the average over the three deleted nodes. Deleting the nodes of the element has a direct impact on its neighbors that share these nodes. More specifically, it is important to realize that collapsing all elements of a mesh region would increase the nodal density rather than decrease it[2]. Therefore, the direct neighbor elements and their own direct neighbors are prevented to be themselves collapsed, irrespective of their size. An example is shown in Fig. 9 where the direct neighbors of the collapsed element 10 and their respective neighbors, indicated by green circles, cannot be collapsed. It can be shown that, for a uniform mesh of equilateral triangles, this coarsening process corresponds to increasing the area of the elements by a factor 4/3.

The special coarsening algorithm at the boundary is applied to all elements with at least one node (and not only an edge as for node addition) on the boundary. First, the actual element area A_{elem} used in the criterion for coarsening, Eq. (15), is replaced by $\frac{1}{2}h^2_{\text{min}}$,

$$\frac{1}{2}h^2_{\text{min}} < \gamma L^{*2}_{\text{elem}}\,, \tag{16}$$

where h_{min} is the length of the smallest edge of the element. This is to avoid too skewed boundary elements, as illustrated in Fig. 10. Then, the smallest edge of the element rather than the element itself is collapsed. More specifically, if either both nodes of the smallest edge are on the boundary or none of them is, then the two nodes are deleted and replaced by a single node at their mid-point. The average value of the physical quantities at the two deleted nodes is assigned to the new node. If the shortest edge involves a boundary node and an interior

[2] As already mentioned, a triangular mesh has more or less twice more elements than nodes so that collapsing each element into a node at its center would approximately double the number of nodes, and thus the nodal density.

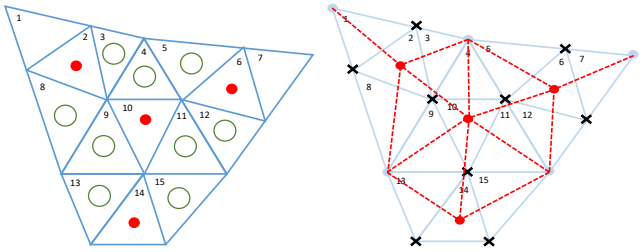


Fig. 9: Illustration of the coarsening process of a mesh region in the bulk, i.e., without boundary elements, where the triangular elements are numbered from 1 to 15. (Left) If element 10 is collapsed, then all its neighbors and their respective neighbors (green circles) are tagged to prevent them from being collapsed. On the other hand, elements 2, 6 and 14 can be collapsed. Note that collapsing all elements would not decrease the nodal density. (Right) Resulting nodes and virtual mesh after coarsening where the red dots indicate the new nodes, the black crosses the nodes that have been deleted and the red dashed lines the corresponding new virtual triangulation.

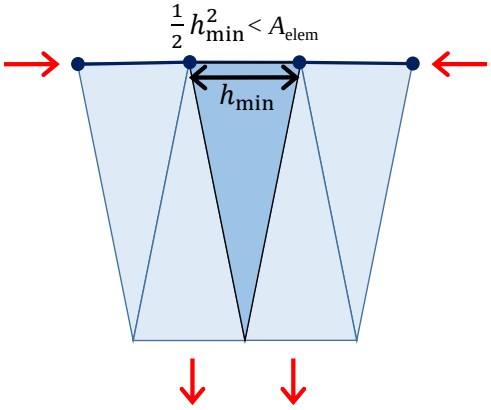


Fig. 10: Illustration of a squeezed boundary (e.g., free-slip boundary) where the boundary nodes are represented by the dark blue dots and the boundary edges by the thick dark blue lines. If the free surface is squeezed, boundary elements are compressed along, and extended perpendicularly to, the free surface, as indicated by the red arrows. This would lead to highly skewed elements with a surface area A_{elem} sufficiently large that the coarsening criterion, Eq. (15), is not satisfied. The modified criterion, Eq. (16), reduces the threshold for coarsening and thus avoids that boundary elements become too skewed.

node, then this interior node is simply deleted. This minimizes the impact of the coarsening on the shape of the boundary.

The choice of the threshold coefficient γ in Eq. (15) is important to avoid a continuously alternating mesh refinement and coarsening in the same region. In particular, if γ is too large, new nodes created at the previous iteration might be deleted because their associated element is now too small, or conversely, elements that have been coarsened might be refined at the next time step. Moreover, as the coarsening step averages locally the solution, it introduces unwanted numerical diffusion that should be minimized. It is thus suggested to impose an upper bound, $\gamma < 1/2$, such that the coarsened mesh in the bulk is, on average, smaller than $\frac{2}{3}L^{*2}$ to avoid overlapping of refined and coarsened elements in the same region. In practice the parameter γ is chosen between $1/3$ and $1/2$ in the bulk. Reducing its value leads to more variability in the element size, potentially slightly more deformed elements but fewer node eliminations, and conversely. At boundaries the increase of nodal density, if particles are added, is twice that in the bulk. Therefore, the parameter γ in Eq. (15) is there divided by two and a smooth transition between $\gamma/2$ at the boundary and γ in the bulk is imposed over a few cells.

3.3 Boundary recognition algorithm

Once the node addition/destruction step has been completed, the old mesh can be discarded and a new Delaunay triangulation is performed. Because the full convex hull of the domain is triangulated, mesh elements that do not belong to the fluid must be discarded. This boundary recognition step relies both on boundary tracking and a local α-shape technique. Concretely, nodes belonging to the boundary at the previous time step are tagged and the global characteristic length scale h in Eq. (5) is replaced by the local target mesh size. In particular, the α-criterion for triangle removal is modified to

$$r_c > \alpha L^*_{\text{elem}}\,, \tag{17}$$

where the user-defined constant α should take a value between 1.2 and 1.8 depending on the case. Note that, because of the difference in the definition of h and L^*_{elem}, the typical value $\alpha = 1.2$ in Eq. (5) corresponds approximately to $\alpha = 1.8$ in Eq. (17) for an equilateral triangle. However, the boundary tracking allows using a much lower value of α, so that in practice $\alpha = 1.2$ in Eq. (17) provides good results. The influence of α is further discussed in section 4.

As already illustrated in Figs. 4 and 5, the mass conservation error is directly linked to the boundary identification algorithm. To minimize this error, three important aspects must be ensured: 1) a sufficiently refined mesh in the boundary vicinity, 2) well-shaped elements, and 3) the knowledge of which nodes belonged to the boundary prior to remeshing (boundary tracking). These requirements explain the aforementioned special treatment of boundary elements during node addition/destruction.

Because elements with at most one node on the boundary can be safely considered as being part of the fluid domain, the α-shape criterion is only applied to

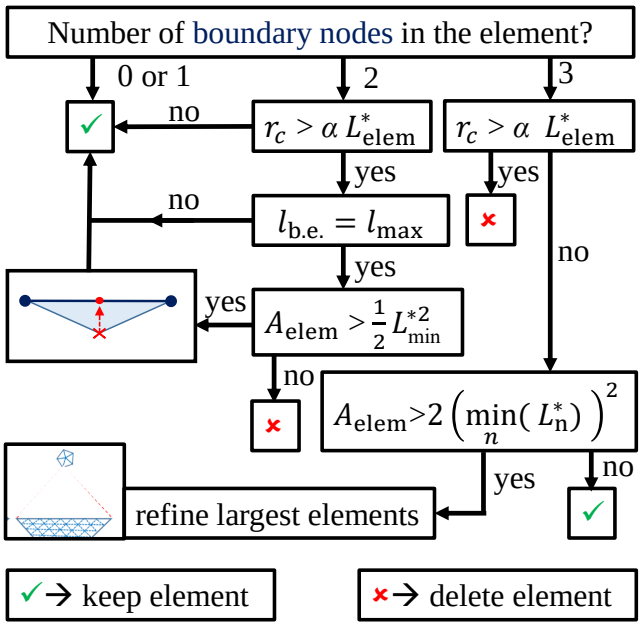


Fig. 11: Algorithm for boundary recognition combining a local α-shape criterion and boundary tracking.

elements with two or three nodes tagged during the previous time step as belonging to the boundary. This reduces the computational cost and simultaneously prevents the unphysical destruction of fluid elements in the fluid domain.

An element with two nodes on the boundary is only eliminated if the following three conditions are all met (see also Fig. 11):

- the local α-criterion for triangle removal, Eq. (17), is satisfied,
- the element edge on the boundary is the longest edge of its element ($l_{\text{b.e.}} = l_{\max}$), and
- the element area is smaller than a minimum value imposed by the global minimal target mesh size: $A_{\text{elem}} < \frac{1}{2} L^{*2}_{\min}$.

The third requirement ensures that the element is removed from the triangulation only if the associated error on mass conservation is of the order of the finest mesh resolution. On the other hand, if only the first two conditions are satisfied, the boundary is locally stretched. In this case, the node that is not on the boundary is tagged to be removed and the boundary edge is tagged to be refined at the next time step, as illustrated in Fig. 12. With this approach the number of elements that would otherwise be deleted is greatly reduced, which allows the use of a lower value of α, as discussed in section 4.

Elements with three nodes on the boundary should a priori be empty elements and thus discarded. Notable exceptions are elements at domain "corners" (see Fig. 5)

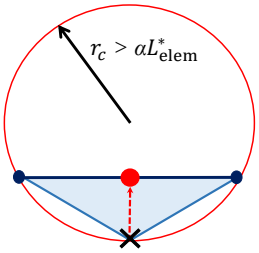


Fig. 12: Illustration of the addition of a new node (red dot) at the mid-point of the longest edge of an obtuse element when this edge (thick blue line) is located on the boundary and the corresponding element satisfies the α-shape criterion for element removal, Eq. (17). The black cross indicates the node that is tagged for removal.

and elements in a thin film whose thickness corresponds to the element size (i.e., only one element thick). Such elements also play a key role during the merging phase of two fluid regions, as already shown in Fig. 4. They are thus discarded of the mesh as soon as one of the following two conditions is satisfied:

- the local α-criterion for triangle removal, Eq. (17), is satisfied, or
- their area is significantly larger than that of one of their direct neighbors. In practice, this condition is expressed as

$$A_{\text{elem}} > 2 \left(\min_n(L^*_n)\right)^2 , \tag{18}$$

where the index n corresponds to the three nodes of the element and the factor 2 has been chosen arbitrarily to ensure that the element size is sufficiently different from that of the neighbors.

The purpose of this second condition is to minimize the spurious creation of mass, as illustrated in Fig. 13 for a free surface folding on itself. A new fluid element is only created if its size is comparable to that of its smallest neighbor. Note that, in practice, the curvature-based refinement of the free-surface ensures that this new element has a size of the order of the global minimum mesh size.

The merging of two boundaries with a significant difference in their corresponding mesh resolution is another example to illustrate the use for this second condition, as shown in Fig. 14. If a significant difference is identified among the target mesh sizes L^*_n at the nodes of the element connecting the two boundaries, this element is discarded. Additionally, the largest neighbor(s) of this element is/are refined. More specifically, the target mesh size of the node(s) lying on the coarser of the two boundaries is assigned the minimum target mesh size. This new target mesh size

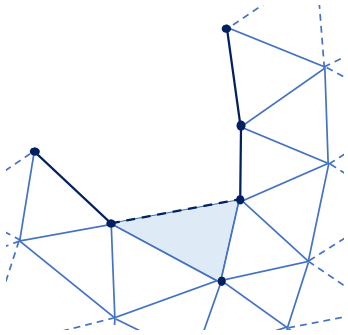


Fig. 13: Illustration of mass creation when a free surface folds on itself. At the previous time iteration, the blue shaded triangle had been discarded by the boundary recognition algorithm and has thus three nodes on the boundary (blue dots). After remeshing, this element would be kept if it does not satisfy the α-shape criterion for removal. To minimize the resulting mass creation, the element is only kept if, additionally, it is of the same size as its direct neighbors. The dark blue dashed line indicates in this case the new boundary edge.

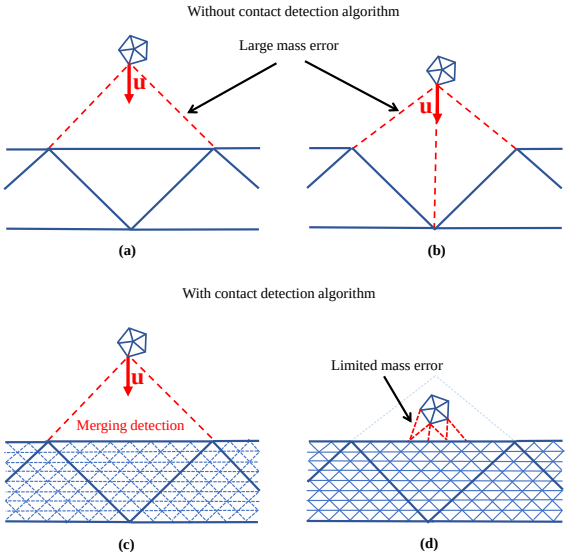


Fig. 14: Merging of two fluid regions without (a,b) and with (c,d) merging detection algorithm. (a) When the more refined region is sufficiently close to the other boundary, the connecting element (red dashed lines) is not discarded by the α-shape criterion, leading to significant mass creation. Considering the second condition based on Eq. (18) eliminates this connecting element from the fluid region and, thus, prevents this artificial mass creation. (b) However, when the more refined region becomes closer to the coarse boundary, Delaunay triangulation might yield a mesh with a different topology, e.g., with two elements (red dashed lines) connecting the two fluid regions. In this case, it is unclear whether these two elements should be kept or discarded. Moreover, in both cases the error in mass conservation is of the same order as the size of the coarser mesh. (c) Refining the coarser boundary and its neighborhood after discarding the connecting element (red dashed lines) delays the merging of the two fluid regions. (d) Finally, delaying the coarsening of the newly refined boundary region ensures that the merging takes only place when the two boundaries are within a distance of the order of the minimum mesh size and, thereby, significantly reduces the mass conservation error.

is then propagated into the bulk through smoothing, as described in sections 3.1.3 and 3.1.4.

Once all the elements that are considered non-fluid elements have been removed, the new boundary can be identified. In particular, all the nodes that define edges belonging to a single element are tagged as boundary nodes for the next time iteration.

Finally, it should be emphasized that, with the proposed algorithm, the separation of a fluid region into two can only take place in a fluid filament that is stretched to a thickness of a single element, i.e., an element with three nodes of the free surface. Fluid detachment is thus delayed compared to the classical PFEM.

4 Validation cases

Four different test cases are now considered to illustrate and validate the proposed mesh refinement algorithm. The first two cases, i.e., the lid-driven cavity flow and the flow around a cylinder at low Reynolds number, do not feature a free surface and are mostly used to assess the mesh refinement part (target mesh size definition and node creation/destruction). If an Eulerian frame of reference were used, these two flows would be either steady or periodic in time. The third test case corresponds to the sloshing motion of a liquid pool in a reservoir that undergoes a rotating motion with a period close to the sloshing resonance. Finally, the last test case involves a two-dimensional liquid drop falling into a bath of the same liquid. For the latter two examples, that are fundamentally unsteady, the focus is on the free-surface motion and mass conservation.

4.1 The lid-driven cavity

The first test case corresponds to a square cavity whose bottom and side walls are fixed and whose top wall (lid) moves from left to right at constant velocity. After a transient phase, the velocity field reaches a steady state characterized by a primary rotational structure and smaller vortices in the lower corners of the cavity, as illustrated in Fig. 15. All reported quantities are made non-dimensional using the fluid density ρ, the cavity side length L and the lid velocity U. A Reynolds number $\mathrm{Re} = \rho U L/\mu = 400$, with μ the dynamic viscosity, is considered here but similar results have been obtained at $\mathrm{Re} = 100$. First, a mesh convergence analysis is performed using a uniform mesh and results are compared with the literature. Then, several strategies to define the target mesh size are analyzed. Finally, results obtained with uniform and non-uniform meshes

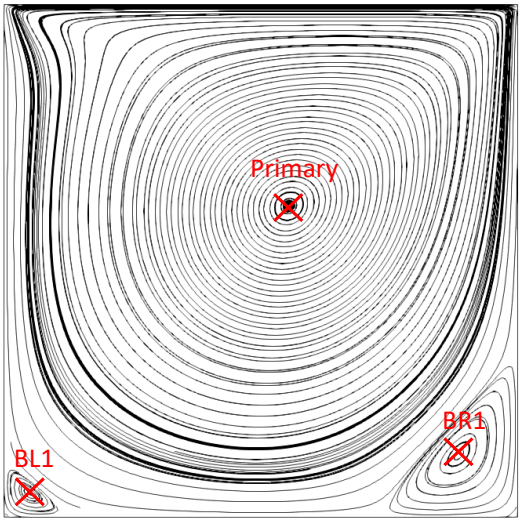


Fig. 15: Streamlines at steady state obtained for the lid-driven cavity flow on a uniform mesh with 80×80 subdivisions along the horizontal and vertical walls. The location of the three main vortices is indicated by the red crosses.

are compared. These results are mainly assessed in terms of the position of the primary and secondary vortices (BL1 and BR1 in Fig. 15) and the vorticity at these locations.

4.1.1 Mesh convergence analysis

First, the convergence of the results on a uniform mesh[3] is analyzed using a series of refined meshes with $N \times N$ subdivisions along the vertical and horizontal walls, ranging from 20×20 to 160×160. The corresponding total number of nodes is more or less $4N^2/3$, as triangular elements are used. For comparison, this would correspond to $L^*_{\text{min}} = L^*_{\text{max}} \approx 0.6/N$. Note that, in the Lagrangian framework, the nodes on the moving lid are moved back to their original position at each time step before remeshing. The time step size is defined based on the minimum target mesh size as $\Delta t = 5L^*_{\text{min}}/12$ for this mesh convergence study and all subsequent simulations. The results obtained with these meshes are also compared to several references from the literature [31,37,52,62].

Figure 15 shows the streamlines of the solution obtained with the 80×80 mesh. As summarized in Table 1, it is found that the mesh with a resolution of 80×80 is sufficient to get results close to those of the literature. The most resolved mesh (160×160) does not lead to results that are significantly different (the difference with respect to the coarser mesh is of the same order of magnitude as the variability across the reference data), but increases noticeably the computational time. Because the intermediate 80×80 mesh provides sufficiently accurate results while keeping the computational cost limited, all following analyzes will be based on meshes of a similar size.

[3] The mesh is not perfectly uniform since it deforms at each time step, but it has approximately a uniform element size.

4.1.2 Definition of the target mesh size

As discussed in Section 3.1, several criteria can be used to define the target mesh size. For this test case, a geometric and two physics-based criteria are investigated. The geometric criterion (denoted GEO) defines a target mesh size that increases linearly from L^*_{min} at the top wall to L^*_{max} at the bottom wall. The finer mesh at the lid is motivated by the larger velocity gradients and the thinner boundary layer in this region, as indicated by the closely packed streamlines in Fig. 15. Moreover, a finer mesh at the cavity upper corners reduces the effect of the velocity discontinuity induced by the moving lid.

It can be expected that this geometric criterion is not sufficient to accurately capture the vortical structures. Alternatively, a refinement based on the velocity gradients using Eq. (7) (denoted SOL1) is thus also considered. Nonetheless, because the corner vortices are much weaker than the primary vortical structure, their associated velocity gradients are much smaller, which leads to a rather large target mesh size there. A better targeted refinement of the corner vortices can be achieved with a solution-based criterion (denoted SOL2) that uses rescaled velocity gradients according to Eq. (8). The mesh size is thus prescribed based on the shape of the vortices rather than on their intensity. The different target mesh size parameters for the three criteria are summarized in Table 2. The mesh resulting from these three criteria are compared for $L^*_{\text{min}} = 0.6/80 = 0.0075$ and $L^*_{\text{max}} = 8L^*_{\text{min}}$.

Typical meshes obtained at steady state using the above mesh refinement criteria are illustrated in Fig. 16. The GEO criterion produces a mesh with an element size progressively increasing in the direction normal to the lid (Fig. 16a), while the SOL1 criterion leads to a fine mesh along the lid with additional refinement at the edge of the primary vortex, in particular below the top right corner (Fig. 16b). However, the resulting mesh at the two bottom corners is very coarse, preventing an accurate representation of the corner vortices there. The SOL2 criterion, on the other hand, yields a fine mesh along the entire edge of the primary vortex and at its core (Fig. 16c). Additionally, a better refinement at the bottom corners is also achieved. For the chosen mesh parameters, none of the two bottom vortices can

		PFEM simulations				Ghia et al.	Schreiber et al.	Vanka	Hou et al.
Resolution →		20×20	40×40	80×80	160×160	257×257	180×180	321×321	256×256
Num. methods		PFEM				CSI-MG	"Homemade"	BLIMM	LBM
Formulated var.		u, v, p				$\psi - \omega$	ψ	u, v, p	BGK model
Primary	x_c	0.5522	0.5527	0.5562	0.5543	0.5547	0.5571	0.5563	0.5608
	y_c	0.6039	0.6080	0.6074	0.6059	0.6055	0.6071	0.6	0.6078
	ω	2.446	2.3155	2.292	2.289	2.2947	2.281	–	–
BR1	x_c	0.8910	0.8857	0.8856	0.8854	0.8906	0.8857	0.8875	0.8902
	y_c	0.1173	0.1213	0.1223	0.1224	0.1250	0.1143	0.1188	0.1255
	ω	0.3749	0.4007	0.4104	0.4161	0.433 52	0.394	–	–
BL1	x_c	–	0.0493	0.05	0.050 83	0.0508	0.05	0.05	0.0549
	y_c	–	0.0457	0.0467	0.047 07	0.0469	0.0429	0.05	0.0510
	ω	–	0.0675	0.044 57	0.060 37	0.056 97	0.0471	–	–

Table 1: Mesh convergence study with uniform mesh for the lid-driven cavity test case: position (x_c, y_c) and associated vorticity ω of the primary and secondary (BR1 and BL1, see Fig. 15 for definition) vortices for different mesh sizes and comparison with the results of Ghia et al. (finite difference method, vorticity-streamfunction formulation) [31], Schreiber et al. (fourth-order finite difference method, vorticity-streamfunction formulation) [52], Vanka (finite difference method on staggered grid) [62] and Hou et al. (Lattice Boltzmann method) [37].

	GEO	SOL1	SOL2
$\|\|\nabla \mathbf{u}\|\|_{\min}$	–	2/9	2
$\|\|\nabla \mathbf{u}\|\|_{\max}$	–	6	50
β	–	1/3	1
$(\mathbf{U}_\infty, U_\epsilon)$	–	–	$((0,0)\,,\,0.01)$

Table 2: Target mesh size parameters of Eqs. (7) and (8) for the mesh refinement of the lid-driven cavity test case.

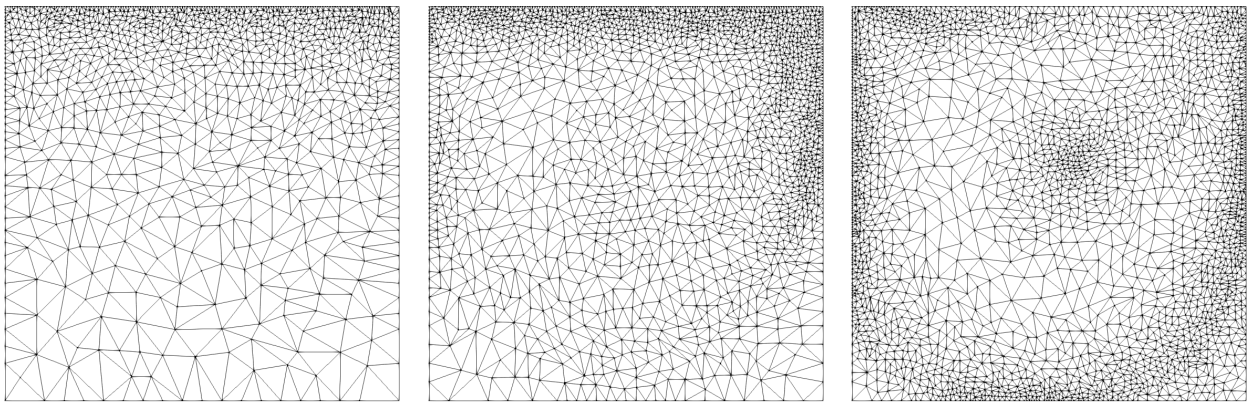
be captured using the GEO refinement, and the SOL1 criterion fails to predict the vortex BL1.

4.1.3 Uniform vs. non-uniform mesh

One of the main motivation for the present mesh refinement algorithm is to be able to refine locally the mesh in order to improve the solution accuracy in specific regions while keeping the computational cost limited. Alternatively, this also provides the ability to use a coarser mesh in regions where the solution is smooth, thus reducing the computational cost while keeping the same accuracy of the solution. However, it should be emphasized that even a very local and spatially limited mesh refinement has an impact on the time step. More specifically, the Courant-Friedrichs-Lewy (CFL) condition imposes a corresponding time step reduction when the element size is reduced. This is compounded by the fundamentally unsteady nature of the Lagrangian formulation that prevents the use of large time steps as typically employed to solve steady problems. In other words, even when the decrease of $L^*_{\min}$ induces only a small increase in the total number of elements, a non-negligible increase in the overall computational cost due to a larger number of time steps is unavoidable.

Three non-uniform meshes based on the SOL2 refinement with the same $L^*_{\min}$ (corresponding to that of a 80×80 uniform mesh) and a decreasing $L^*_{\max}$ are compared to uniform meshes in Table 3. One can observe that increasing $L^*_{\max}$ deteriorates slightly the accuracy of the simulation, but also significantly reduces the number of nodes. For instance, Mesh 2 in Table 3 has half as many nodes as the 80×80 uniform mesh for virtually the same accuracy. Comparing Mesh 1 and the 40×40 uniform mesh that both have a similar number of nodes, one observes that, although the position of the primary vortex is surprisingly slightly less accurate for Mesh 1, the corresponding vorticity is much better predicted.

Finally, it should be emphasized that these results, and in particular the position of the different vortices, are very sensitive to many factors. This is clearly demonstrated by the variability of the results in the literature (see Table 1) despite very fine meshes. This sensitivity is most likely due to the presence of additional smaller and smaller vortices that emerge in the corners when increasing the grid resolution. The abrupt appearance of such a vortex impacts the overall flow, and correspondingly the other vortices. This phenomenon thus complicates the grid convergence analysis. Furthermore, because the lid-driven cavity flow is fully confined and involves boundary layers on all four walls, the gain achieved by using a non-uniform mesh is somehow limited. Most Eulerian studies in the literature are actually based on uniform meshes. Nonetheless, this test case provides the opportunity to test different strategies of mesh refinement, as shown above.



(a) GEO: $N_{\mathrm{st}} = 1184$ **(b)** SOL1: $N_{\mathrm{st}} = 3566$ **(c)** SOL2: $N_{\mathrm{st}} = 2469$

Fig. 16: Meshes at steady state obtained using three different mesh refinement strategies for the lid driven cavity. For all three cases, $L^*_{\mathrm{min}} = 0.6/80 = 0.0075$ and $L^*_{\mathrm{max}} = 8L^*_{\mathrm{min}}$. The approximate total number of nodes at steady state, N_{st}, is also reported. The other target mesh size parameters are summarized in Table 2.

Mesh type →			Similar L^*_{min}			Uniform		
			Mesh 1	Mesh 2	Mesh 3	(40 × 40)	(80 × 80)	(160 × 160)
Parameters	L^*_{min}		0.0075	0.0075	0.0075	0.015	0.0075	0.003 25
	$L^*_{\mathrm{max}}/L^*_{\mathrm{min}}$		8	4	2	1	1	1
	$\Delta t \times 10^3$		3.125	3.125	3.125	6.25	3.125	1.5625
	N_{init}		∽ 500	∽ 1200	∽ 2900	∽ 2200	∽ 8500	∽ 34000
	N_{steady}		∽ 3300	∽ 5200	∽ 7900	∽ 2700	∽ 10500	∽ 42000
Results	Primary	x_c	0.5594	0.5563	0.5551	0.5527	0.5562	0.5543
		y_c	0.6106	0.6083	0.6072	0.6080	0.6074	0.6059
		ω	2.281	2.2895	2.292	2.3155	2.292	2.289

Table 3: Comparison of different mesh adaptation strategies for the lid-driven cavity flow: SOL2 mesh refinement for three meshes with same L^*_{min} and decreasing L^*_{max} and three uniform meshes with decreasing L^*_{min}. Mesh parameters, time step, initial and final number of nodes and position (x_c, y_c) and associated vorticity ω of the primary vortex.

4.2 Flow around a circular cylinder at low Reynolds number

The second test case corresponds to the classical two-dimensional flow around a circular cylinder of diameter D. Several Reynolds numbers are considered up to $\mathrm{Re}_D = 200$, so that the flow remains laminar and two-dimensional but features either steady or periodic dynamics.

4.2.1 Numerical setup

All physical quantities reported below are made non-dimensional using the cylinder diameter D, the free-stream velocity U_∞ and the fluid density ρ. The computational domain is rectangular. Unlike the previous test case, this one features a flow that enters and leaves the computational domain at all far field boundaries. Because inlet/outlet boundary conditions are not straightforward in Lagrangian methods, the upper and lower boundaries are here modeled as slip walls, i.e., with zero normal velocity, to prevent an in- or outflow at these two boundaries. The left and right boundaries are on the other hand inlet and outlet boundaries, respectively. A uniform velocity profile is imposed at the inlet and constant pressure at the outlet. A description of the actual implementation of such boundary conditions can be found in Cerquaglia [11].

A relatively large computational domain is chosen, with $-4.5 \leq x \leq 75.5$ in the streamwise direction and $-15 \leq y \leq 15$ in the flow normal direction, where the origin is located at the cylinder center. This allows minimizing the effect of boundary conditions and the blockage effect induced by the slip walls (the blockage is here $B = 1/30$). Furthermore, it provides the opportunity to clearly demonstrate the advantage of a non-uniform mesh, as relatively large elements can be used away from the cylinder and its wake.

Overall, the non-dimensional target mesh size ranges from $L^*_{\mathrm{min}} = 0.01$ to $L^*_{\mathrm{max}} = 1.1$, corresponding to a ratio of 110 between smallest and largest elements. Two geometric refinement criteria are combined. First,

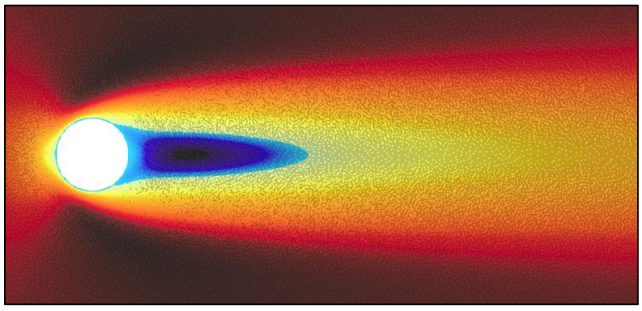
L_1^* is increased linearly from $L_{\min}^*$ at the cylinder surface to $L_{\max}^*$ at a distance $d = 15$ from the cylinder center. Then, the second geometric criterion imposes a target mesh size L_2^* with an intermediate value of 0.11 for $|y| \leq 2$ (i.e., in the wake region). For $|y| \geq 2$, L_2^* increases linearly in y to $L_{\max}^*$ at the upper and lower boundary. Additionally, a solution-based criterion provides a target mesh size L_3^* that varies between $L_{\min}^*$ where non-dimensional gradients are larger than 4 and 0.14 where gradients are smaller than 0.012 according to Eq. (7) with $\beta = 1/3$. This criterion is however only applied in the region $|y| \leq 2$ to ensure a good resolution of the Karman vortex street. Finally, the prescribed target mesh size is obtained as the minimum over the three criteria: $L^* = \min\{L_1^*, L_2^*, L_3^*\}$. Examples of meshes using only the two geometric or all three criteria are shown in Fig. 6. It is interesting to note that a uniform mesh with a mesh size of 0.01 everywhere would increase the number of elements by a factor between 10 and 60. At last, the non-dimensional time step is $\Delta t = L_{\min}^*$.

4.2.2 Results

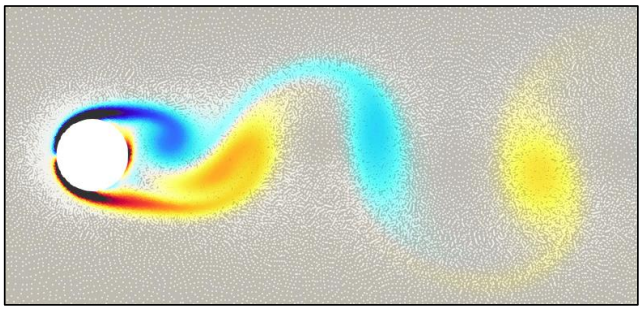
It is well-known that, for $\mathrm{Re}_D \lessapprox 40$, the flow around a static cylinder is steady with a symmetric recirculation bubble downstream of the cylinder (see Fig. 17a), while at higher Reynolds number the flow becomes unsteady with periodic vortex shedding and a corresponding Karman vortex street (Fig. 17b). The wake remains two-dimensional and laminar for flows at $\mathrm{Re}_D \lessapprox 200$.

The steady or mean (for unsteady cases) drag coefficient C_D is compared for different values of Re_D ranging between 20 and 200 with references from the literature in Fig. 18. Because the steady or mean lift vanishes by symmetry, the root-mean-square (rms) of the lift coefficient is reported in Fig. 19 and compared with the numerical results of Norberg [45]. A very good match is observed in both cases between the present results and the reference data. This excellent agreement is further confirmed by the comparison of the recirculation bubble length L_{rec} for steady cases with similar blockage B in Fig. 20, and the Strouhal number, $\mathrm{St} = fD/U_\infty$ with f the shedding frequency, for unsteady cases in Fig. 21.

The simulation at $\mathrm{Re}_D = 200$ has also been repeated using only the geometric refinement criteria, L_1^* and L_2^*. The same aerodynamic coefficients are found as for the simulation using the solution-based criterion, but a 5% lower shedding frequency is observed, owing to the coarser mesh in the wake.



(a) $\mathrm{Re}_D = 40$



(b) $\mathrm{Re}_D = 200$

Fig. 17: Two-dimensional flow around a static cylinder. (a) Contour of the streamwise velocity component for the the steady case at $\mathrm{Re}_D = 40$. The blue zone indicates the region where the streamwise velocity is negative and illustrates the extent of the recirculation bubble. (b) Contour of the vorticity for the unsteady case at $\mathrm{Re}_D = 200$ showing the periodic Karman vortex street. In both cases, only a short portion of the entire computational domain is shown.

4.3 Forced sloshing reservoir

The previous two test cases are classical examples for which an Eulerian approach would be better adapted. On the other hand, the PFEM can be of advantage when simulating flows with a free surface. To highlight the added value of adaptive mesh refinement and to evaluate the use of the proposed algorithm for free-surface flows, the sloshing in a tank undergoing a forced roll motion is now studied. In particular, the objective is to reproduce the experimental and numerical results of Delorme et al. [20] and Souto-Iglesias et al. [55] for the case of water in the large tank with lateral impacts. A schematic of the setup and its dimensions is shown in Fig. 22 and the key physical parameters are listed in Table 4. The Reynolds number is defined with the initial water height H and the characteristic velocity $\sqrt{gH}$, where g is the gravity acceleration, so that the Froude

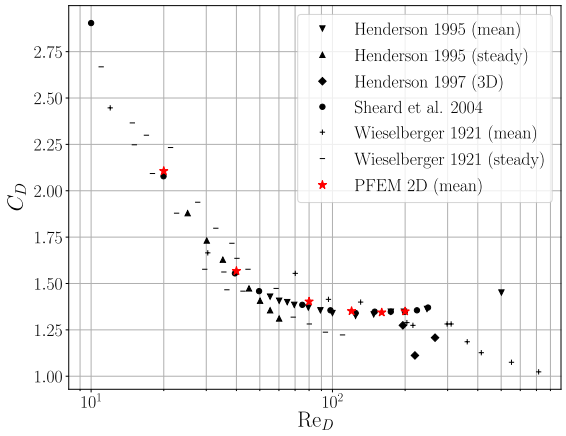


Fig. 18: Steady or mean (if unsteady) drag coefficient as a function of the Reynolds number for the laminar flow around a static cylinder. The present results (red stars) are compared with the experimental results of Wieselberger [63] and the numerical results of Henderson [35,36] and Sheard et al. [54].

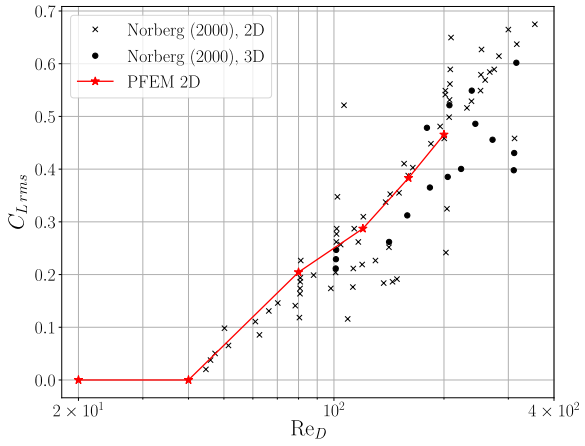


Fig. 19: Root-mean-square (rms) lift coefficient as a function of the Reynolds number for the laminar flow around a static cylinder. The present results (red stars) are compared with the numerical results of Norberg [45]. The first two PFEM data points correspond to the steady cases without vortex shedding.

number $\mathrm{Fr} = 1$. Using the shallow water dispersion relation, Souto−Iglesias et al. [55] estimate the resulting sloshing period as

$$T_s = 2\pi \left(\frac{\pi\, g}{L} \tanh\left(\frac{\pi\, H}{L} \right) \right)^{-1/2} = 1.9191\,\mathrm{s}\,. \qquad (19)$$

The imposed rolling motion of the tank is an oscillatory rotational motion around the horizontal with an amplitude $\phi_{\max} = \pm 4°$ and a period T_f that is 85% of the natural sloshing period T_s. The

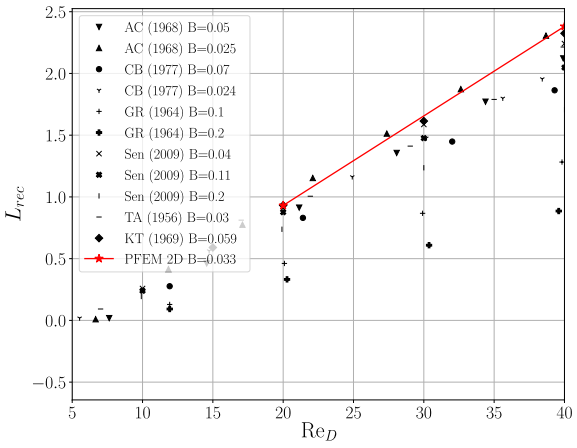


Fig. 20: Non-dimensional length of the recirculation bubble behind the static cylinder for $\mathrm{Re}_D \lessapprox 40$. The present results (red stars, $B = 0.033$) are compared with the experimental results of Taneda (TA) [57], Grove et al. (GR) [32], Coutanceau and Bouard (CB) [13] and Acrivos et al. (AC) [1], and to the numerical results of Sen et al. [53] and Keller and Takami (KT) [40], for different values of the blockage B.

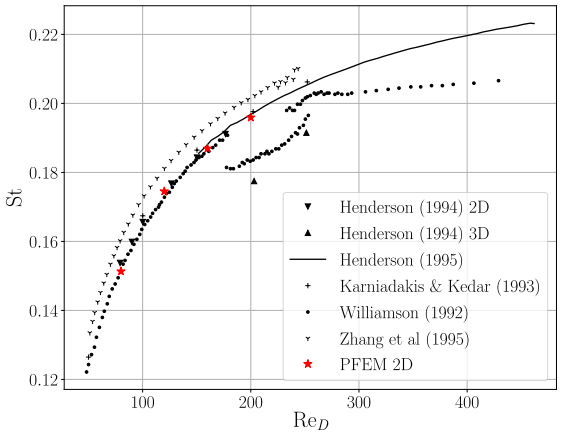


Fig. 21: Strouhal number of the vortex shedding in the wake of a static cylinder for different Reynolds numbers. The present results (red stars) are compared with experimental results of Zhang et al. [66] and Williamson [64] and numerical results of Henderson (2D and 3D) [34] and Karniadakis and Kedar (taken from Williamson [64]).

corresponding tilt angle $\phi(t)$ with respect to the horizontal line has been taken from the experiment and is shown in Fig. 23.

4.3.1 Mesh adaptation

For this case, the mesh adaptation strategy relies on the combination of one geometric and two solution-based

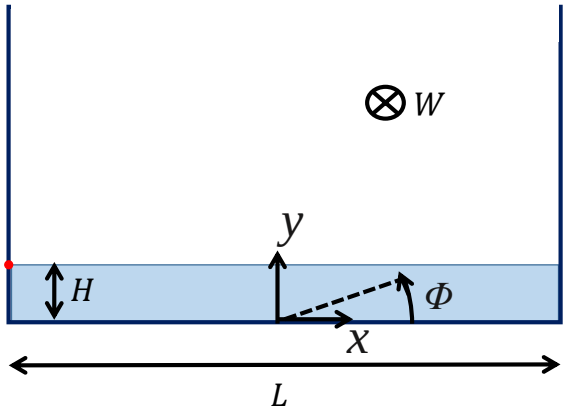


Fig. 22: Schematic of the setup for the sloshing experiment of Delorme et al. [20] and Souto-Iglesias et al. [55]. Only two-dimensional simulations are performed, neglecting the presence of the front and back walls and the resulting friction. The red dot on the left wall indicates the position of the pressure sensor located at a height corresponding to the initial water height H. The respective numerical values are summarized in Table 4.

Density	ρ	998 kg/m^3
Dynamic viscosity	μ	$8.94 \cdot 10^{-4}$ Pa·s
Surface tension	σ	0.0728 N/m
Initial water height	H	93 mm
Tank length	L	900 mm
Tank width	W	62 mm
Reynolds number	Re	10^5
Froude number	Fr	1
Sloshing period	T_s	1.9191 s
Rolling period	T_f	1.6312 s
Rolling amplitude	$\phi_{\max}$	$\pm 4°$

Table 4: Key physical parameters for the sloshing experiment of Delorme et al. [20] and Souto-Iglesias et al. [55]. The geometrical dimensions are defined in Fig. 22.

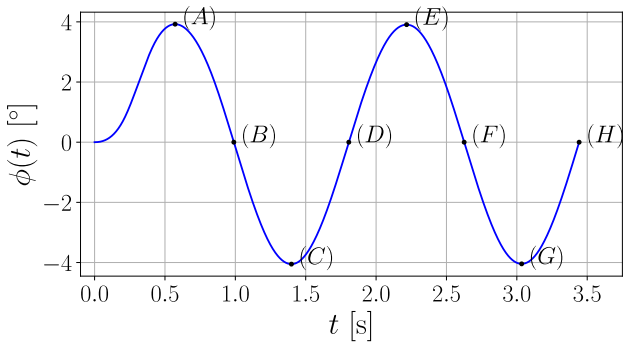


Fig. 23: Tilt angle $\phi(t)$ of the tank as a function of the time taken from the sloshing experiment of Delorme et al. [20] and Souto-Iglesias et al. [55]. The points labeled (A) to (H) correspond to maximum, zero and minimum tilt.

$L^*_{\min}$	1.4 mm
$L^*_{\max}$	13 mm
$\|\|\widetilde{\nabla \mathbf{u}}\|\|_{\min}$	21.5 m^{-1}
$\|\|\widetilde{\nabla \mathbf{u}}\|\|_{\max}$	107.5 m^{-1}
β	1
U_ϵ	0.01 m/s
m	120

Table 5: Parameters used to define the target mesh size for the solution-based mesh refinement of the sloshing test case.

criteria. First, a mesh stretching from the minimum target mesh size $L^*_{\min} = 1.4\,\text{mm}$ at the walls to $L^*_{\max} = 9.28 L^*_{\min}$ at a distance H from the walls is imposed through a geometric progression. Then, regions of large velocity gradients are refined according to Eqs. (7) and (8), while the free-surface is refined using the criterion of Eq. (9). The corresponding parameters are summarized in Table 5. The results are also compared to those obtained on a uniform mesh with an element size $L^*_{\min}$ using the classical PFEM algorithm. Despite the slight differences in Eqs. (5) and (17), the same α is used for both mesh types. Moreover, two different values, $\alpha = 1.2$ and $\alpha = 1.4$, are considered to assess the impact of α on mass conservation. As shown in the following, the best results are obtained with $\alpha = 1.2$ for the non-uniform mesh and $\alpha = 1.4$ for the uniform mesh. These values are used when illustrating the results. Finally, the same time step size $\Delta t = 0.0005$ s is considered for all simulations.

4.3.2 Results

The deformation of the free surface obtained with the non-uniform mesh ($\alpha = 1.2$) is compared to experimental measurements in Fig. 24 at the time instants (A) to (H) defined in Fig. 23, i.e., at zero, maximum and minimum tilt angles. The white dots correspond to the mesh nodes and are superposed to photographs of the experiment. The effect of the curvature-based free-surface mesh adaption is clearly visible with a progressive refinement around the high-curvature zones, which allows a better description of the wave crest. Very good agreement can be observed for the free-surface motion during the first two sloshing periods ((A) to (F) in Fig. 24). Following the subsequent impact of the wave on the left wall, the flow becomes more chaotic, with splashes and individual vortices, and discrepancies become more apparent. The wave front seems to break slightly earlier in the numerical simulations (instant (G)). Additionally, the free-surface reaches a higher height when impinging onto the right wall (instant (H)). Nevertheless, although discrepancies become more

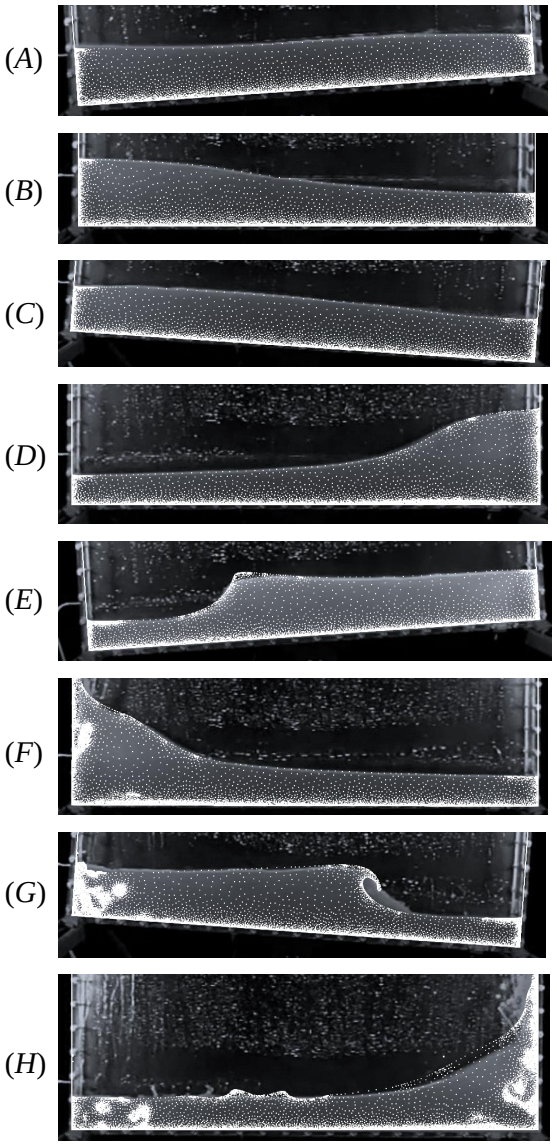


Fig. 24: Free-surface deformation for the sloshing test case: computational nodes (white dots) of the PFEM on a non-uniform mesh with $\alpha = 1.2$ superimposed to photographs of the experiment [20,55] at different time instants corresponding to the highlighted points in Fig. 23.

significant over time, the new algorithm provides a very good approximation of the flow and free-surface motion in this initial phase.

The experiment also includes pressure measurements that provide a more quantitative basis for comparison [20,55]. The pressure sensor is located on the left wall at a height corresponding to the initial water height H (red dot in Fig. 22). The time evolution of the pressure is shown in Fig. 25. Very good agreement is found between the PFEM simulation on a non-uniform mesh (red line) and the

	Uniform	Non-uniform
N_{init}	~ 29000	~ 4700
$N_{\text{imp.2}}$	~ 30000	~ 5500
$N_{\text{imp.3}}$	~ 29000	~ 8000
$N_{\text{imp.4}}$	~ 26000	~ 8000

Table 6: Total number of nodes at the beginning of the simulation and after the second, third and fourth impacts on the left wall, for both the uniform ($\alpha = 1.4$) and the non-uniform ($\alpha = 1.2$) meshes. The decrease of the total number of nodes of the uniform mesh is due to mass destruction linked to nodes that are eliminated when too close to each other.

experiment (green line) for the first wave reflection at $t \approx 1$ s and when the wave impacts the left wall at $t \approx 2.3$ s. The agreement remains good for the two subsequent impacts, in particular regarding the impact time, but a somewhat larger pressure level is predicted by the simulation in the post-impact phase (at $t \approx 4.3$ s and $t \approx 6$ s). Furthermore, the fourth impact seems to be predicted slightly earlier. The slower motion of the wave in the experiment is possibly due to the friction forces on the front and back walls of the tank, as the tank width is about 15 times smaller than its length. The presence of these front and back walls, and the corresponding friction, are not accounted for in the present two-dimensional simulations.

Figure 25 also includes the results obtained with the classical PFEM on a uniform mesh with $\alpha = 1.4$ (blue line). In this case, although the first two impacts are well predicted, the long-time evolution of the pressure clearly shows discrepancies. In particular, a growing time lag is observed for the third, and mostly fourth impact. This time-lag is even more significant with $\alpha = 1.2$ (not shown here). Moreover, the post-impact pressure level continuously decreases at each impact. To further analyze the differences between the two PFEM simulations and the experiment in the later phase, the free-surface deformation at the fourth impact ($t = 5.65$ s) is shown in Fig. 26. For the non-uniform mesh, the figure illustrates well the mesh refinement at the free surface, close to the walls and in vortical regions. The overall free surface is well captured, despite some expected discrepancies in the impact region on the left wall. For the uniform mesh on the other hand, the aforementioned large time lag is clearly visible. Even more striking in this case is the lower height of the free surface, which suggests non-negligible mass loss. Such mass destruction, and the resulting lowering of the average liquid height H, likely explains the time lag (larger sloshing period T_s, Eq. (19)) and possibly the decreasing pressure level over time that are observed with the classical PFEM on the uniform mesh.

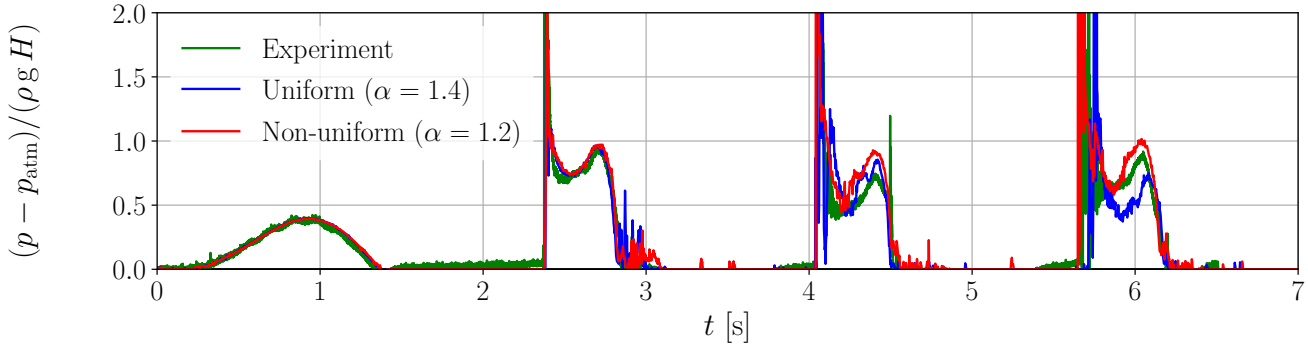


Fig. 25: Non-dimensionalized pressure at the pressure sensor on the left wall of the sloshing tank as a function of the time. PFEM results with a uniform mesh and $\alpha = 1.4$ (blue) and a non-uniform mesh and $\alpha = 1.2$ (red) are compared to the experimental measurements (green) [20,55].

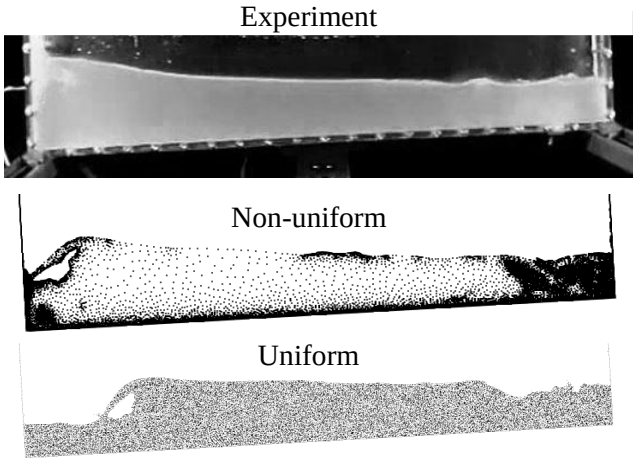


Fig. 26: Free-surface deformation for the sloshing case at the fourth impact ($t = 5.65$ s): photograph from the experiment [20, 55] (top) and mesh nodes from the PFEM simulations on the non-uniform mesh with $\alpha = 1.2$ (middle) and uniform mesh with $\alpha = 1.4$ (bottom). The wave break is barely distinguishable in the experiment. Note that the size of the dots for the uniform mesh (bottom) is twice smaller for better visualization.

4.3.3 Mass conservation

To further investigate the problem of mass conservation, Fig. 27 depicts the time variation of the total volume due to remeshing alone, ΔV_{rem}, and to both remeshing and time integration, $\Delta V_{\text{tot}} = \Delta V_{\text{rem}} + \Delta V_{\text{num}}$. The first observation is that the value of α has a non-negligible impact on mass conservation. In particular, a lower mass error is achieved with $\alpha = 1.4$ on a uniform mesh while $\alpha = 1.2$ yields better results on the non-uniform mesh.

The remeshing in the new algorithm leads to some mass creation (see Fig. 27(a)), but which remains limited compared to the larger mass destruction (about two times in magnitude when comparing the better results) in the classical algorithm. As already mentioned in section 3.3, the new algorithm strongly reduces unwanted mass destruction at the free-surface; this allows using a low value of α that limits mass creation. Note that most mass creation takes place at the vertical walls when wetted by splashes. As shown by the red dashed lines in Fig. 27, increasing α to 1.4 leads to a larger mass creation. On the other hand, a larger α reduces mass destruction with the classical PFEM on a uniform mesh. This mass destruction mostly results from nodes deleted on purpose to prevent a bad mesh quality when they get too close to each other. This is further supported by Table 6 that summarizes the total number of nodes at the beginning of the simulation and at the second, third and fourth impacts on the left wall. Because the uniform mesh has elements of size L^*_{min}, its initial number of nodes is approximately six times that of the non-uniform mesh. However, its mesh size decreases between the third and fourth impact. On the other hand, the total number of nodes of the non-uniform mesh expectedly increases over time due to mesh adaptation. Nevertheless, towards the end of the simulations, the number of nodes of the non-uniform mesh remains more than three times smaller than that of the uniform mesh. Finally, the numerical time integration induces in both cases some mass destruction (see Fig. 27(b)). For the new algorithm the two contributions, ΔV_{rem} and ΔV_{num}, cancel each other almost perfectly (probably by chance here) so that the total error in mass conservation is almost zero, while it amounts to more than 25% after 8 s with the classical PFEM using $\alpha = 1.4$.

Overall, the results obtained with the new algorithm are very good despite some discrepancies at later times, mostly due to the inherently chaotic nature of splashes, two-dimensional approximation and finite mesh resolution. Moreover, the use of mesh adaptation and boundary tracking reduces the error in mass

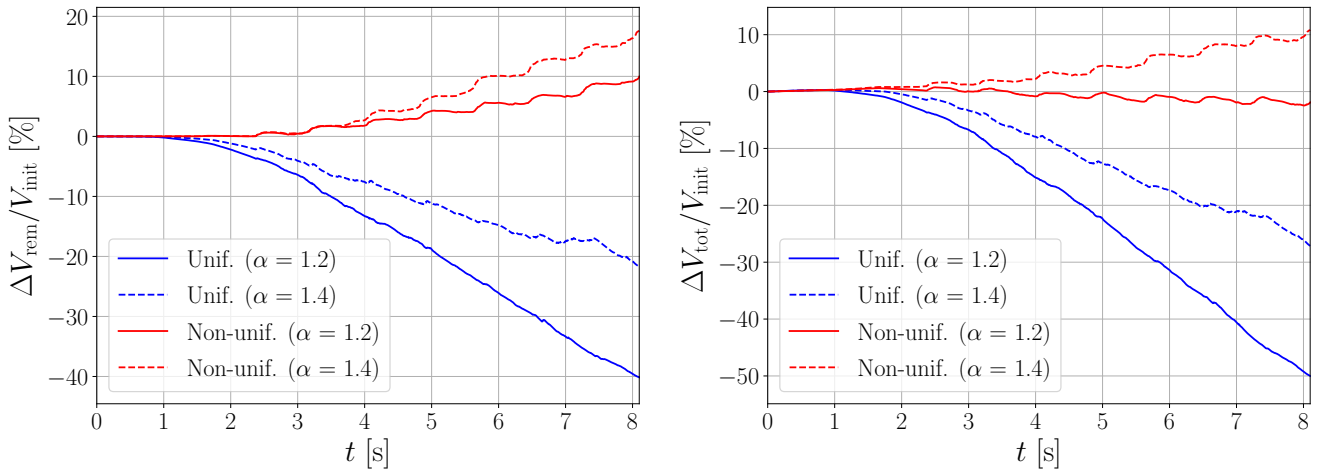


Fig. 27: Variation of the total volume (mass) as a function of the time for the classical PFEM on a uniform mesh and for the new algorithm on a non-uniform mesh for both $\alpha = 1.2$ and 1.4 in Eqs. (5) and (17), respectively. (a) Error ΔV_{rem} due to the remeshing only, and (b) overall error including the error due to the time integration, $\Delta V_{\text{tot}} = \Delta V_{\text{rem}} + \Delta V_{\text{num}}$.

conservation. In particular, the new algorithm allows choosing a lower value of α (more stringent criterion), which decreases mass destruction. Additionally, the value of α has been found to have an important impact on mass conservation. This value could potentially be further optimized for this specific case, but mass conservation would still remain a major challenge [28]. This issue can be mitigated with the new algorithm. Moreover, for the same maximum mesh resolution, the non-uniform mesh has a lower total number of nodes, so that the computational cost is significantly reduced. Finally, it should be mentioned that the significant error in mass conservation due to the time integration is most likely due to the accumulation of errors following the relatively long simulation time and repeated impacts at the side walls.

4.4 Two-dimensional drop falling into a liquid bath

The last test case considered is taken from Franci et al. [28] and corresponds to a two-dimensional drop falling into a bath of the same liquid. A schematic of the problem is shown in Fig. 28 and the key parameters are summarized in Table 7.

To reduce the computational time, the simulations begin with the drop closer to the free surface, as the initial phase only involves the drop's uniform acceleration. In practice, the simulation is started at $t_0 = 108.3$ ms, i.e., when the drop is at a distance $h_0 = 0.5R$ above the free-surface, with the initial drop velocity $U_0 = \sqrt{2g(H - h_0)}$.

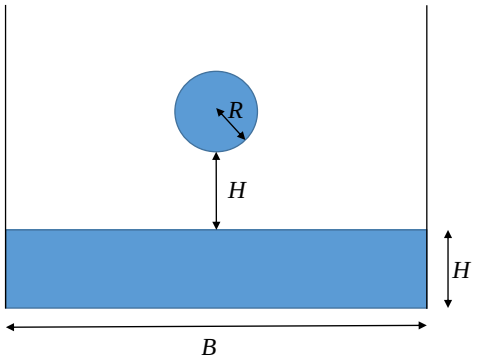


Fig. 28: Schematics of the initial configuration of the test case where a two-dimensional drop falls into a tank full of the same viscous fluid, as proposed by Franci et al. [28]. The numerical values of the different geometric parameters are summarized in Table 7.

Initial liquid height	H	0.07 m
Length of the tank	B	0.3 m
Initial drop position	H	0.07 m
Initial drop radius	R	0.025 m
Liquid dynamic viscosity	μ	0.1 Pa·s
Liquid density	ρ	10^3 kg/m^3

Table 7: Geometric and physical parameters for the 2D drop falling into a liquid bath, as proposed by Franci et al. [28]. The definition of the geometric parameters is given in Fig. 28.

4.4.1 Mesh adaptation

Two different meshes are considered. The first one is uniform and the classical α-shape technique with $\alpha = 1.2$ in Eq. (5) is used, while the second simulation relies on a non-uniform mesh using the new refinement

	Uniform	Non-uniform
α	1.2	1.2
h [mm]	1.5	-
L^*_{min} [mm]	~ 1	0.18
L^*_{max} [mm]	~ 1	7.3
N_{init}	~ 14000	~ 8000
N_{post}	$\sim 14000 - 15000$	$\sim 14000 - 16500$
$\|\|\nabla \mathbf{u}\|\|_{\text{min}}$ [s^{-1}]	-	85
$\|\|\nabla \mathbf{u}\|\|_{\text{max}}$ [s^{-1}]	-	3.15
β	-	1/3

Table 8: Mesh parameters for the falling liquid drop test case. Both a uniform mesh similar to that of Franci et al. [28] and a non-uniform mesh using the solution-based criterion given by Eq. 7 are considered. The definition of α is given by Eq. (5) for the uniform mesh and by Eq. (17) for the non-uniform mesh. The number of nodes at the beginning of the simulation and after the drop impact is also reported.

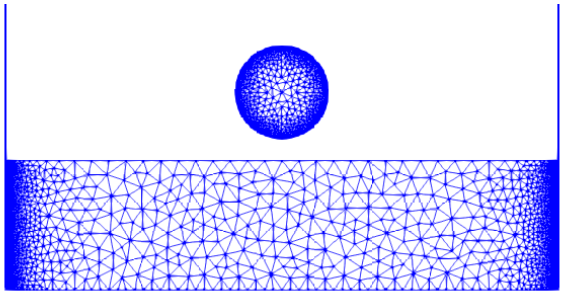


Fig. 29: Initial non-uniform mesh for the falling drop. The refined mesh in the vicinity of the side walls, as imposed by the geometric criterion, and at the drop surface are clearly visible.

and boundary recognition algorithm with $\alpha = 1.2$ in Eq. (17). These results are then compared to the most resolved case of Franci et al. [28].

Again, the target size of the non-uniform mesh relies on a combination of geometric and solution-based criteria. First, the tank's side walls are discretized using the finest mesh resolution L^*_{min} in order to minimize the mass error during their wetting. Then, a geometric criterion imposes a linear increase of the target mesh size from L^*_{min} at the walls to L^*_{max} at a distance $0.75\,D$ away from these walls. No special refinement is imposed on the bottom no-slip wall as the impact of the boundary layer in this region is considered small. Then, the curvature-based criterion, Eq. (9), is used to refine the free-surface with $m = 270$, provided that the resulting target mesh size is not smaller than L^*_{min}. This is combined with a solution-based refinement relying on Eq. (7). On the other hand, the drop is initially discretized at its surface using a mesh size of L^*_{min}, which is slightly smaller (62%) than the value that would be imposed by the curvature-based criterion. Finally, this test case relies additionally on the contact detection algorithm to minimize mass conservation errors during the impact of the drop onto the bath's free surface and merging of the two fluid subdomains.

The mesh parameters for both the uniform and non-uniform meshes are summarized in Table 8 and the initial non-uniform mesh is shown in Fig. 29. The uniform mesh is similar to that of Franci et al. [28]. It is interesting to observe that the two meshes have approximately the same final number of nodes N_{post}. At last, the time step size is set to $\Delta t = 0.157$ ms for both simulations.

4.4.2 Results

A key feature of this test case is the merging of two fluid domains. As discussed above, the PFEM is well adapted to deal with such cases but it also suffers from mass conservation errors. The mesh refinement technique and the contact detection algorithm proposed here attempt to mitigate this issue. The application of the contact algorithm is illustrated in Fig. 30, which shows the mesh of the drop and bath free surface at different instants in time shortly before and during the merging process for both mesh types. The mesh of the falling drop is initially much finer than that of the flat liquid pool. Without contact detection algorithm, the merging of the drop with the bath occurs when the drop is within a distance of the bath mesh size of the free surface, with a thereby large associated mass error. The contact detection allows anticipating the merging and refining accordingly the liquid bath free-surface region around the expected impact point. The mass conservation error is thus lower, of the order of the smallest element size.

The motion of the bath free surface after the drop impact is shown in Fig. 31 and should be compared to Fig. 25 of Franci et al. [28]. The results with the uniform mesh are very similar to those of Franci et al., with small discrepancies, particularly close to the walls. These discrepancies can be explained by the chaotic nature of the problem. This high sensitivity to small perturbations is highlighted by the asymmetry of the solution at later times. The mesh resolution has thus also a clear impact, as demonstrated by the results of Franci et al. and the present comparison between uniform and non-uniform mesh. The more accurate representation of the free surface through local mesh refinement is further illustrated by the close view on a splash next to the right wall in Fig. 32. The non-uniform mesh yields a much more refined description of the liquid filament because mesh refinement ensures that small elements are used

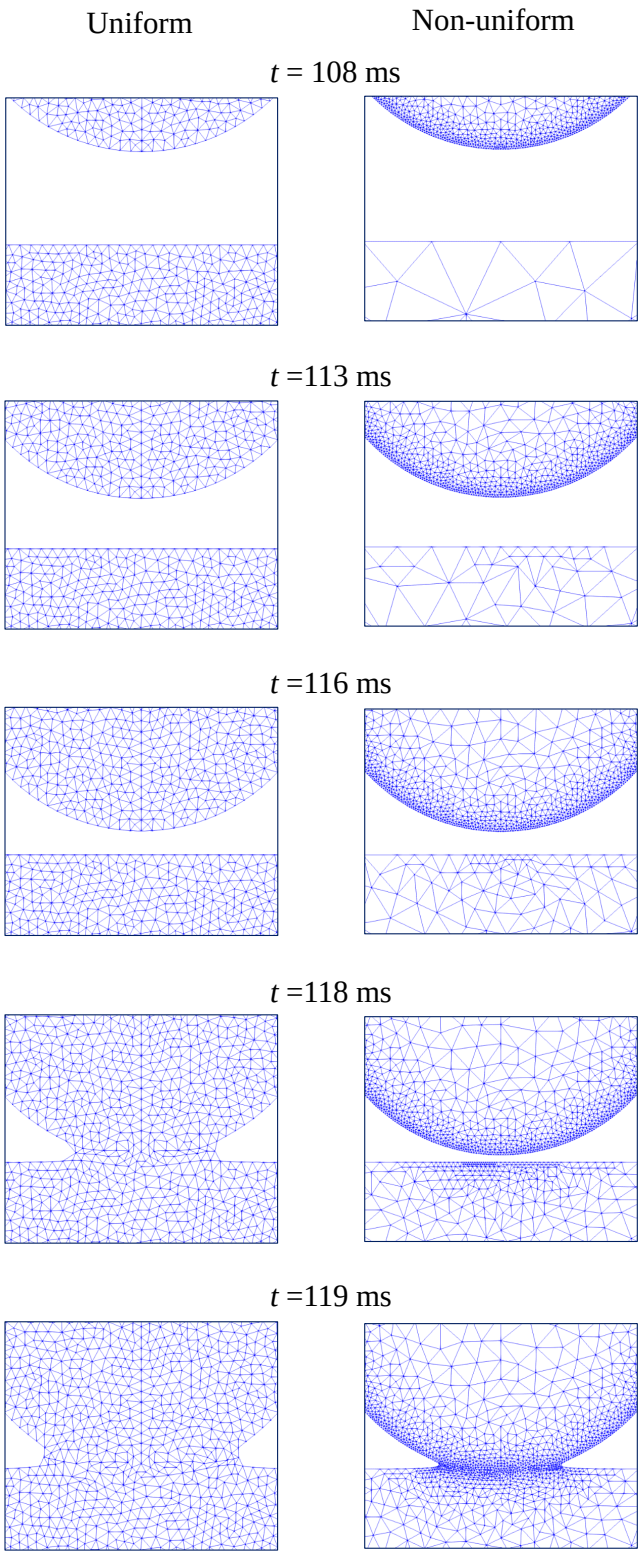


Fig. 30: Comparison between the uniform mesh (left) and non-uniform mesh with contact detection algorithm (right) at several time instants before the initial drop impact and during merging with the bath. The non-uniform mesh at the free surface of the liquid bath is initially much coarser than the respective mesh of the drop. By identifying the upcoming merging process, the algorithm can refine the liquid bath mesh in the expected impinging region. As shown by the last two time instants, the size of the added elements during the merging process is thus proportional to the smallest mesh size so as to minimize the mass conservation error.

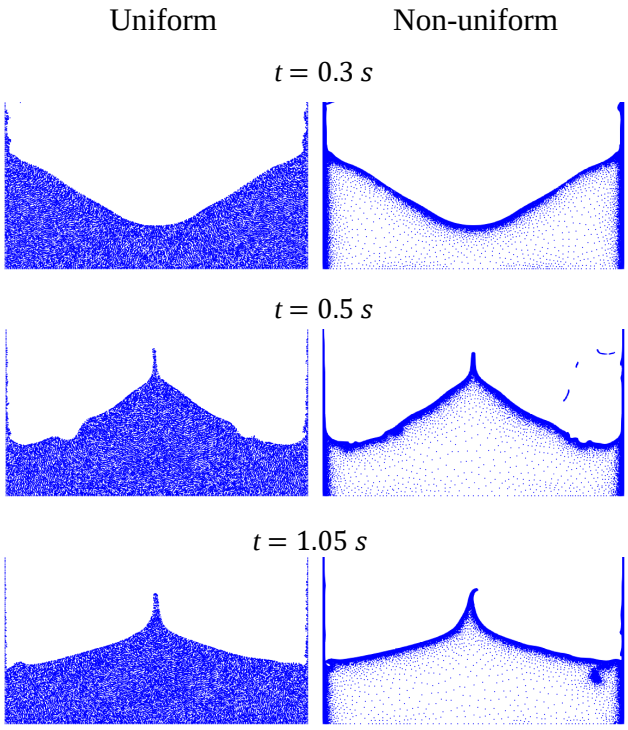


Fig. 31: Free-surface deformation at three instants in time after the impact of the falling drop for the uniform (left) and non-uniform (right) mesh. The snapshots correspond to those in Fig. 25 of Franci et al. [28].

there and the boundary recognition algorithm prevents elements to be removed unless they are smaller than the highest grid resolution or have their three nodes on the free surface. Nonetheless, the total number of nodes is kept limited by using a coarser mesh away from the free surface, which greatly reduces the overall computational time.

4.4.3 Mass conservation

Unlike the previous test case, the error in mass conservation due to remeshing is here of the same order of magnitude for the classical PFEM on the uniform mesh and for the new algorithm, as illustrated in Fig. 33. For the uniform mesh this error oscillates between periods with predominantly mass destruction and periods of mass creation, such that the error seems to remain bounded over time. On the other hand, a monotone mass increase is observed with the new algorithm. This increase seems to take place during certain events between phases where the mass remains mostly constant. This is again due to the interface tracking algorithm which strongly reduces mass destruction. Despite the relative small value of α, mass is created at the vertical walls during wetting events (see for example Fig. 32). A larger

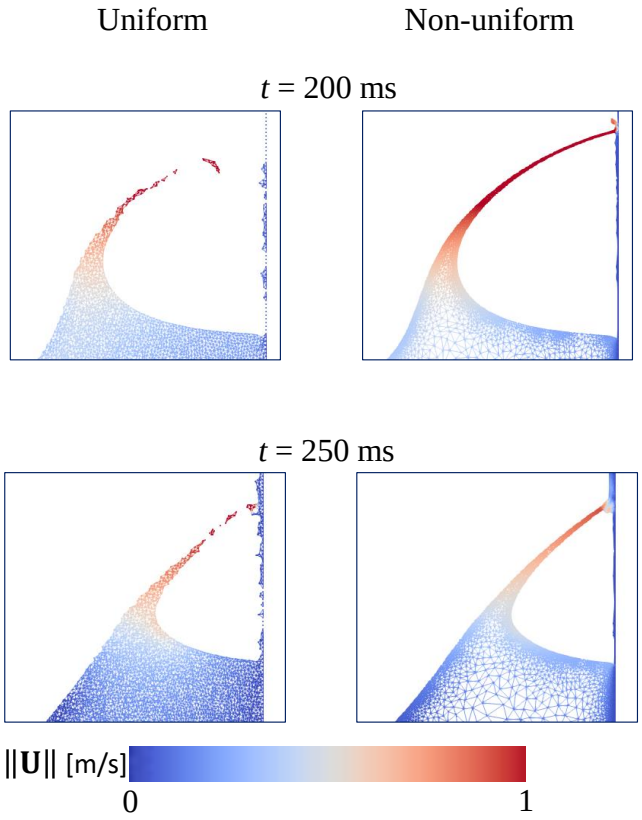


Fig. 32: Free-surface deformation and splash at the right wall after the fall of a drop into a liquid bath. Two different time instants are shown for the uniform (left) and non-uniform (right) meshes. Contour of the velocity magnitude.

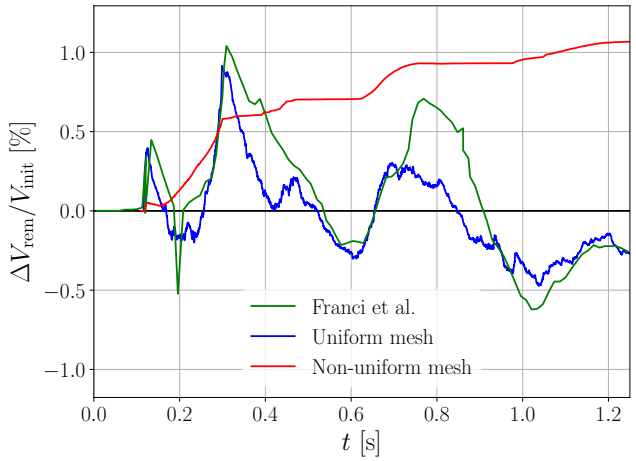


Fig. 33: Variation of the total volume (mass) due to remeshing as a function of the time for the falling drop test case: most resolved results of Franci et al. [28] (green), classical PFEM algorithm on the uniform mesh (blue) and new algorithm with contact detection on the non-uniform mesh (red). The parameters used for the simulations are summarized in Table 8.

value of α would further exacerbate this issue. Owing to the contact detection algorithm, a much lower

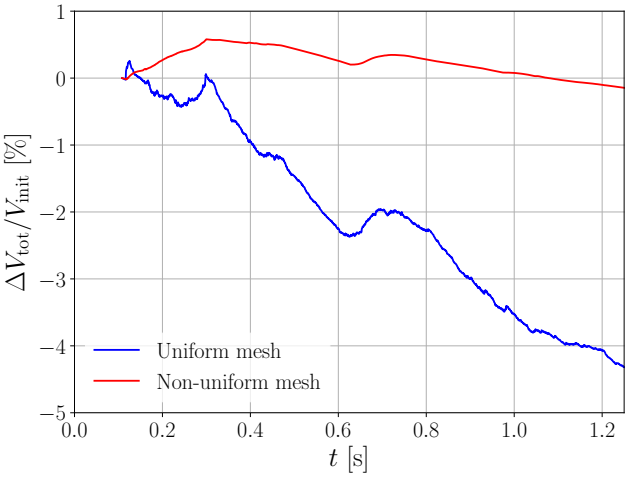


Fig. 34: Variation of the total volume (mass) due to both time integration and remeshing as a function of the time for the falling drop test case: classical PFEM algorithm on the uniform mesh (blue) and new algorithm with contact detection on the non-uniform mesh (red).

mass increase is also observed at the drop impact ($t \approx 0.1$ ms). Despite the similar mass conservation error due to remeshing, the new algorithm allows a much better representation of the free surface, and, thereby, a more accurate time integration. This is shown by Fig. 34, which represents the variation of the total volume (mass) due to both remeshing and time integration. Because the time integration leads to mass destruction, the two contributions for the new algorithm again almost cancel each other, such that the net error remains very small. On the other hand, the classical PFEM shows a continuous mass decrease over time. This again illustrates the accuracy improvement brought by the new algorithm.

5 Extension to 3D

Although the present work has focused on two-dimensional meshes, it is interesting to briefly discuss the extension of the proposed mesh adaptation and tracking technique to three dimensions. Three-dimensional simulations are computationally very expensive. The use of a non-uniform mesh can thus provide an even larger reduction of the computational cost than in 2D. For some cases, it might even be the only feasible approach to perform simulations with a mesh that is sufficiently refined to ensure the required solution accuracy.

Even if this extension is for many steps of the algorithm rather trivial, several challenges arise, mostly because some of the optimal properties of Delaunay

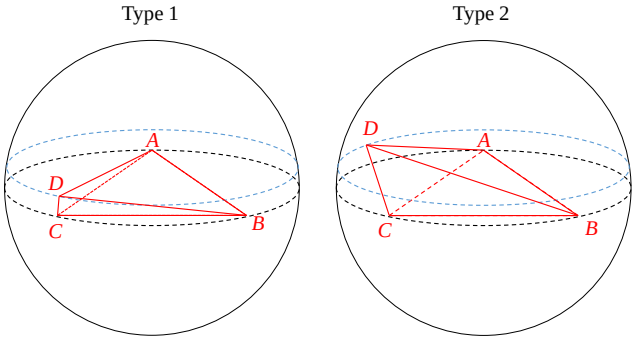


Fig. 35: Two types of badly-shaped tetrahedra called slivers, characterized by a small volume V_{elem} and at least one small dihedral angle θ_i. In type 1 at least two points (C and D here) are very close to each other, such that the sum of the solid angles, $\sum_i \Omega_i$, is not particularly small. In type 2 all nodes are well separated from each other, which results in a small sum of the solid angles.

triangulation in 2D do not carry over in 3D. In particular, the 3D triangulation can lead to the creation of badly-shaped elements called slivers. Slivers merely consist in tetrahedra having all their nodes close to the equator of their circumscribed sphere, such that they have a flat shape, as illustrated in Fig. 35. They are characterized by at least one small dihedral angle and a small volume, which can be used to identify them. Slivers should be avoided as much as possible, as they typically lead to inaccurate results and stability issues in the case of artificially compressible schemes [43].

Overall, the idea of enforcing a target mesh size through addition/elimination of nodes and of tracking the interface can be reused. First, the definition of the target mesh size L^* based on geometric and/or solution-based criteria can be easily extended to three dimensions. However, determining whether elements need to be refined or coarsened based on this target mesh size is less obvious in 3D. In particular, while the local nodal density depends only on the element surface area in 2D, it depends on the tetrahedra volume and shape in 3D. Specifically, it can be shown that the local nodal density σ_{elem} away from boundaries is given by[4]

$$\sigma_{\text{elem,3D}} = \frac{f_{N,\text{3D}}}{V_{\text{elem}}} = \frac{1}{V_{\text{elem}}} \frac{\sum_i \Omega_i}{4\pi}, \tag{20}$$

$$\sigma_{\text{elem,2D}} = \frac{f_{N,\text{2D}}}{A_{\text{elem}}} = \frac{1}{A_{\text{elem}}} \frac{\sum_i \Omega_i}{2\pi}, \tag{21}$$

where f_N is the nodal fraction, V_{elem} the volume, A_{elem} the surface area and Ω_i the solid angles of an element[5]. The nodal fraction is the fraction of the element nodes that can be assigned to this specific element. Because the three angles of a triangle sum to π, the nodal fraction $f_{N,\text{2D}} = \pi/(2\pi) = 1/2$ is constant in 2D and the nodal density can simply be calculated using the element surface area A_{elem}. This is not the case in 3D ($f_{N,\text{3D}}$ depends on the specific tetrahedron shape), so that it is useful to define a target nodal density σ^* from the element target mesh size L^*_{elem}:

$$\sigma^*_{\text{elem}} = \frac{f^*_N}{V^*_{\text{elem}}} = \frac{f^*_N}{L^{*3}_{\text{elem}}}. \tag{22}$$

Based on well-shaped space-filling tetrahedra, a value $f^*_N \approx 1/5$ can be estimated. It is thus suggested to replace the refinement/coarsening criterion based on surface area or volume by a criterion based on the actual nodal density and bounds around its target value.

An advantage of the proposed approach is that ensuring a smoothly varying nodal density can prevent the appearance of slivers of type 1 because these elements are characterized by a local nodal density that is larger than their neighbors (small volume but not so small solid angle sum, see Fig. 35). On the other hand, slivers of type 2 have not only a small volume, but also a small $f_{N,\text{3D}}$, resulting in a nodal density that is not particularly large with respects to their neighbors. A mesh adaptation criterion only based on the volume would consequently lead to a mesh coarsening, even if the mesh nodes were uniformly distributed and the mesh size were adequate, without solving the problem of slivers. In other words, slivers of type 2 should not be eliminated by adding or removing nodes (unless the actual nodal density differs from its target value) but rather by rearranging locally the node distribution. Several methods have been proposed in the literature [43,59] that could potentially be integrated into the present mesh adaptation algorithm.

More degrees of freedom are available for refinement in 3D. In particular, a new node can be added at the center of the tetrahedron (dividing the element by four), at the center of a face (dividing the element by three) or in the middle of an edge (dividing the element by two). In order to avoid large variations in local nodal density, it is thus better to add a new node on an edge [21]. Moreover, adding a node in the center of a face would lead to four nodes in a plane, increasing the risk of slivers. This latter option could however be used to refine a boundary face. On the other hand, mesh coarsening could follow the same algorithm as in 2D: elements with a too large nodal density would be collapsed into a single node in their center and their direct and second neighbors would be tagged to prevent their own collapse.

[4] Note that the sum of the solid angles can be related to the sum of the dihedral angles θ_i through Gram-Euler theorem : $\sum_i \Omega_i/(4\pi) = \sum_i \theta_i/(2\pi) - 1$.

[5] In 2D, Ω_i are simply the internal angles.

Finally, the boundary tracking algorithm in Fig. 11 can be directly extended to 3D. Tetrahedra with less than three nodes on the boundary would be automatically kept. Tetrahedra with three nodes, i.e., a face, on the boundary would be considered a priori as fluid elements but would be eliminated if they do not satisfy the α-shape criterion in Eq. (17), if their largest face is on the boundary and if they reach the minimal mesh resolution. If the last condition is not satisfied, a new node would be added in the middle of the boundary face. At last, elements with four nodes on the boundary would only be kept if their size is similar to that of their neighbors.

In summary, the extension of the proposed algorithm to 3D would provide a powerful approach to reduce the computational cost of three-dimensional simulations. Most of the steps could be easily adapted in 3D. The most important change would be the use of the nodal density rather than the volume to determine the elements that need to be refined or coarsened. This proposed approach would provide a useful solution to slivers of type 1. Nonetheless, the algorithm would need to be further extended, potentially leveraging existing methods, to address the issue of slivers of type 2.

6 Conclusion

A novel mesh adaptation technique has been proposed in the context of the particle finite element method to improve the method's computational efficiency and mitigate its known deficiency regarding mass conservation. The mesh adaptation algorithm relies on the definition of a local target mesh size. This target mesh size can be prescribed according to geometric considerations, such as a distance from a solid surface, or based on the solution itself, using for instance velocity gradients, surface curvature or other physical quantities. Both geometric and physics-based criteria can be combined and customized to the specific case considered. A smoothing of the target mesh size has also been proposed to avoid large jumps in mesh size across neighbor elements and to propagate the mesh refinement from the boundary to the domain interior. The target mesh size is then enforced approximately by adding nodes if the actual mesh size is much larger than its target value, or deleting nodes in regions where the mesh is finer than it ought to be. Depending on whether the corresponding elements are in the interior of the domain or at a boundary, different strategies have been proposed for adding or removing nodes. Finally, the boundary recognition step relies on a local adaptation of the α-shape technique and boundary node tracking. In this context, a contact detection algorithm with delayed coarsening has also been developed to improve the description of the merging process between two fluid regions.

The added-value of the method and its correct implementation have then been illustrated through several test cases without and with a free-surface, i.e., the lid-driven cavity flow, the laminar flow around a cylinder, the sloshing dynamics in a oscillating tank and the fall of a drop into a liquid bath. The corresponding results have shown that the algorithm ensures a non-uniform mesh with higher resolution in the targeted regions. In many cases, this non-uniform mesh allows reducing the computational cost by constructing a coarser mesh in parts of the domain away from the region of interest, or improving the solution accuracy by increasing the mesh resolution where needed.

Importantly, a better overall conservation of mass has been observed with the new algorithm. Because of boundary tracking, less mass is destroyed so that the error associated with remeshing corresponds for all test cases to mass creation, while a net mass destruction has been observed with the classical PFEM. This mass creation takes mostly place when a fluid region approaches another one or wets a wall, but it can be mitigated by using a small value of α. A concrete consequence of the reduced mass destruction is a delayed detachment of parts of the fluid. Because with the classical PFEM mass creation and destruction at different locations can partly cancel each other, the error in mass conservation due to remeshing is not always smaller with the new algorithm. However, the local refinement of the free surface enabled by the new algorithm provides an efficient mean to decrease this problem, and the overall error (from both remeshing and time integration) has been found to always be smaller than for the classical PFEM. Nevertheless, the α-shape technique, although local here, remains a purely geometric criterion. As such, it is not possible to fully eliminate mass creation/destruction at boundaries. The only possible cure would be to consider a criterion that relies on the physics and involves nodal physical quantities.

It should be emphasized that, in general, geometric and physics-based criteria are not restricted to the few examples considered here. The proposed approach allows easily incorporating new criteria that might be better adapted for a specific case. In that respect, the criteria proposed here are most likely not optimal but they have the advantage of being flexible and versatile. On the other hand, tailoring the refinement criteria to a specific case reduces also their generality and presupposes an a priori knowledge of the flow. Furthermore, any new criterion requires additional

implementation effort. The proposed criteria offer thus a good trade-off between generality and optimality, but many others can be imagined. Similarly, the choice of the mesh adaptation parameters (e.g., in Eqs. (7)-(10)) provides flexibility and user-control.

The mesh adaptation algorithm evidently adds to the work load at each time step, but the algorithm complexity remains $\mathcal{O}(N)$, where N is the total number of nodes. The different test cases have shown that the new algorithm increases by at most a factor of two the cost of the remeshing step, which itself corresponds to only a limited part of the overall simulation time. Moreover, this additional cost can be by far compensated by the reduced size of the mesh for a given accuracy. It should nonetheless be emphasized that the refinement of even very small portions of the domain should be accompanied by a reduction of the time step size to keep a constant CFL number, which can have a non-negligible impact on the overall computational cost.

Finally, it has been shown that the present algorithm can be quite straightforwardly extended to 3D. It has been suggested to use the local nodal density rather than the element volume to determine the elements to be coarsened or refined. This should eliminate slivers of type 1. However, the algorithm should be further adapted, possibly by integrating existing methods, to address slivers of type 2. Overall, the significantly larger computational cost of three-dimensional simulations renders any mesh size reduction even more critical than in 2D. The present mesh adaptation technique thus represents an important contribution in this regard and opens the way to the application of the PFEM to real cases of interest.

7 Acknowledgments

The financial support of the Belgian Fund for Scientific Research under research project WOLFLOW (F.R.S.-FNRS, PDR T.0021.18) is gratefully acknowledged.

8 Conflict of interest

On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Acrivos, A., Leal, L.G., Snowden, D.D., Pan, F.: Further experiments on steady separated flows past bluff objects. Journal of Fluid Mechanics **34**(1), 25–48 (1968). DOI https://doi.org/10.1017/S0022112068001758
2. Bathe, K., Zhang, H.: A mesh adaptivity procedure for CFD and fluid-structure interactions. Computer and Structures **87**(11-12), 604–617 (2009). DOI https://doi.org/10.1016/j.compstruc.2009.01.017
3. Berger, M., Colella, P.: Local adaptive mesh refinement for shock hydrodynamics. Journal of Computational Physics **82**, 64–84 (1989). DOI https://doi.org/10.1016/0021-9991(89)90035-1
4. Bernadini, F., Bajaj, C.L.: Sampling and Reconstructing Manifolds using Alpha-Shapes. Tech. Rep. 97-013, Department of computer science technical reports (1997)
5. Brezzi, F.: On the existence, uniqueness and approximation of saddle-point problems arising from lagrangian multipliers. ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique **8**(R2), 129–151 (1974)
6. Bristeau, M., Glowinski, R., Periaux, J.: Numerical methods for the Navier-Stokes equations. Applications to the simulation of compressible and incompressible viscous flows. Computer Physics Reports **6**, 73–187 (1987)
7. Carbonell, J., Rodríguez, J., Oñate, E.: Modelling 3D metal cutting problems with the particle finite element method. Computational Mechanics **66**, 603–624 (2020). DOI https://doi.org/10.1007/s00466-020-01867-5
8. Cerquaglia, M., Deliége, G., Boman, R., Papeleux, L., Ponthot, J.: The particle finite element method for the numerical simulation of bird strike. International Journal of Impact Engineering **109**, 1–13 (2017). DOI https://doi.org/10.1016/j.ijimpeng.2017.05.014
9. Cerquaglia, M., Deliége, G., Boman, R., Terrapon, V., Ponthot, J.: Free-slip boundary conditions for simulating free-surface incompressible flows through the particle finite element method. International Journal for Numerical Methods in Engineering **110**(10), 921–946 (2017). DOI https://doi.org/10.1002/nme.5439
10. Cerquaglia, M., Thomas, D., Boman, R., Terrapon, V., Ponthot, J.: A fully partitioned lagrangian framework for FSI problems characterized by free surfaces, large solid deformations and displacements, and strong added-mass effects. Computer Methods in Applied Mechanics and Engineering **348**, 409–442 (2019). DOI https://doi.org/10.1016/j.cma.2019.01.021
11. Cerquaglia, M.L.: Development of a fully-partitioned PFEM-FEM approach for fluid-structure interaction problems characterized by free surfaces, large solid deformations, and strong added-mass effects. Ph.D. thesis, university of Liege (2019)
12. Cerquaglia, M.L., Deliége, G., Boman, R., Ponthot, J.P.: Preliminary assessment of the possibilities of the particle finite element method in the numerical simulation of bird impact on aeronautical structures. pp. 101–108 (2017). DOI https://doi.org/10.1016/j.proeng.2016.12.043
13. Coutanceau, M., Bouard, R.: Experimental determination of the main features of the viscous flow in the wake of a circular cylinder in uniform translation. part 1. Steady flow. Journal of Fluid Mechanics **79**(2), 231–256 (1977). DOI https://doi.org/10.1017/S0022112077000135
14. Cremonesi, M., Ferri, F., Perego, U.: A Lagrangian PFEM approach to the numerical simulation of 3D large scale landslides impinging in water reservoirs. In: ECCOMAS Congress 2016: VII European Congress on Computational Methods in Applied Sciences and Engineering (2016). DOI https://doi.org/10.7712/100016.1840.7927
15. Cremonesi, M., Franci, A., Idelsohn, S.: A state of the art review of the particle finite element method (PFEM). Archives of Computational Methods in Engineering **27**, 1709–1735 (2020). DOI https://doi.org/10.1007/s11831-020-09468-4
16. Cremonesi, M., Frangi, A., Perego, U.: A Lagrangian finite element approach for the analysis of fuid–structure interaction problems. International Journal of Numerical Methods in Engineering **184**(5), 610–630 (2010). DOI https://doi.org/10.1002/nme.2911
17. Dannenhofer, F., Baron, J.: Grid adaptation for the 2D Euler equations (1985). DOI https://doi.org/10.2514/6.1985-484
18. De Sterck, H., Manteuffel, T., McCormick, S., Nolting, J., Ruge, J., Tang, L.: Efficiency-based h- and hp-refinement strategies for finite element methods. Numerical Linear Algebra with Applications **00**, 1–25 (2008). DOI https://doi.org/10.1002/nla.567
19. Delaunay, B.: Sur la sphère vide, à la mémoire de Georges Voronoï. Bulletin de l'Académie des sciences de l'URSS. Classe des sciences mathématiques et naturelles **6**, 793–800 (1934)
20. Delorme, L., Colagrossi, A., Souto-Iglesias, A., et al.: A set of canonical problems in sloshing, part 1: Pressure field in forced roll-comparison between experimental results and SPH. Ocean Engineering **36**(2), 168–178 (2009). DOI https://doi.org/10.1016/j.oceaneng.2008.09.014
21. Douglas, N.A., Arup, M., , Luc, P.: On global and local mesh refinements by a generalized conforming bisection algorithm. SIAM Journal on Scientific Computing **22**(2), 431–448 (2000). DOI https://doi.org/10.1137/S1064827597323373
22. Duval, M., Lozinskic, A., Passieuxb, J., Salaunb, M.: Residual error based adaptive mesh refinement with the non-intrusive patch algorithm. Computer Methods in Applied Mechanics and Engineering **329**, 118–143 (2018). DOI https://doi.org/10.1016/j.cma.2017.09.032
23. Dávalos, C., Cante, J., Hernández, J., Oliver, J.: On the numerical modeling of granular material flows via the Particle Finite Element Method (PFEM). International journal of solids and structures **71**, 99–125 (2015). DOI https://doi.org/10.1016/j.ijsolstr.2015.06.013
24. Eberhard, B.: An adaptive finite-element strategy for the three-dimensional time-dependent Navier-Stokes equations. Journal of Computational and Applied Mathematics **36**, 3–28 (1991). DOI https://doi.org/10.1016/0377-0427(91)90224-8
25. Edelsbrunner, H., Kirkpatrick, D.G., Seidel, R.: On the shape of a set of points in the plane. IEEE Transactions on information theory **29**(4), 551 – 559 (1983). DOI https://doi.org/10.1109/TIT.1983.1056714
26. Edelsbrunner, H., Mücke, E.P.: Three-dimensional alpha shapes. ACM Transactions on Graphics **13**(1), 43 – 72 (1994). DOI https://doi.org/10.1145/147130.147153
27. Edelsbrunner, H., Shah, N.: Incremental Topological Flipping Works for Regular Triangulations. Algorithmica **15**, 223–241 (1996). DOI https://doi.org/10.1007/BF01975867
28. Franci, A., Cremonesi, M.: On the effect of standard PFEM remeshing on volume conservation in free-surface fluid flow problems. Computational Particle Mechanics **4**(3), 331–343 (2017). DOI https://doi.org/10.1007/s40571-016-0124-5
29. Franci, A., Zhang, X.: 3D numerical simulation of free-surface Bingham fluids interacting with structures using the PFEM. Journal of Non-Newtonian Fluid Mechanics **259**, 1–15 (2018). DOI https://doi.org/10.1016/j.jnnfm.2018.05.001
30. Friedel, H., Grauer, R., Marliani, C.: Adaptive Mesh Refinement for Singular Current Sheets in Incompressible Magnetohydrodynamic Flows. Journal of Computational Physics **134**(1), 190–198 (1997). DOI https://doi.org/10.1006/jcph.1997.5683
31. Ghia, U., Ghia, K., Shin, C.T.: High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. Journal of Computational Physics **48**, 387–411 (1982). DOI https://doi.org/10.1016/0021-9991(82)90058-4

32. Grove, A.S., Shair, F.H., Peterson, E.: An experimental investigation of the steady separated flow past a circular cylinder. Journal of Fluid Mechanics **19**(1), 60–80 (1964). DOI https://doi.org/10.1017/S0022112064000544
33. Hannukainen, A., Korotov, S., Křížek, M.: On global and local mesh refinements by a generalized conforming bisection algorithm. Journal of Computational and Applied Mathematics **235**(2), 419–436 (2010). DOI https://doi.org/10.1016/j.cam.2010.05.046
34. Henderson, R.D.: Adaptive Spectral Element Methods, parallel algorithms and simulations. Ph.D. thesis, Princeton University (1994)
35. Henderson, R.D.: Details of the drag curve near the onset of vortex shedding. Physics of Fluids **7**(9), 2102–2104 (1995). DOI https://doi.org/10.1063/1.868459
36. Henderson, R.D.: Non-linear dynamics and pattern formation in turbulent wake transition. Journal of Fluid Mechanics **352**, 65–112 (1997). DOI https://doi.org/10.1017/S0022112097007465
37. Hou, S., Zou, Q., Chen, S., Doolen, G., Cogley, A.: Simulation of cavity flow by the lattice Boltzmann method. Journal of Computational Physics **118**, 329–347 (1995). DOI https://doi.org/10.1006/jcph.1995.1103
38. Hughes, T.J.R., Franca, L.P., 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. Computer Methods in Applied Mechanics **54**(3), 85–99 (1986). DOI https://doi.org/10.1016/0045-7825(86)90025-3
39. John, B., Marsha, B., Jeff, S., Welcome, M.: Three-Dimensional Adaptive Mesh Refinement for Hyperbolic Conservation Laws. SIAM Journal on Scientific Computing **15**, 127–138 (1994). DOI https://doi.org/10.1137/0915008
40. Keller, H.B., Takami, H.: Numerical studies of viscous flow about cylinders, p. 115 (1966)
41. Lee, D.T., Schachter, B.J.: Two algorithms for constructing a Delaunay triangulation. International Journal of Computer and Information Sciences **9**(3), 219–242 (1980). DOI https://doi.org/10.1007/BF00977785
42. Lohner, R., Morgan, K.: Improved adaptive refinement strategies for finite element aerodynamic computations (1986). DOI https://doi.org/10.2514/6.1986-499
43. Meduri, S., Cremonesi, M., Perego, U.: An efficient runtime mesh smoothing technique for 3D explicit Lagrangian free-surface fluid flow simulations. International Journal for Numerical Methods in Engineering **117**(4), 430–452 (2018). DOI https://doi.org/10.1002/nme.5962
44. Monforte, L., Carbonell, J., Arroyo, M., Gens, A.: Performance of mixed formulations for the particle finite element method in soil mechanics problems. Computational Particle Mechanics **4**(3), 269–284 (2017). DOI https://doi.org/10.1007/s40571-016-0145-0
45. Norberg, C.: Flow around a circular cylinder: aspects of fluctuating lift. Journal of Fluids and Structures **15**, 459–469 (2001). DOI https://doi.org/10.1006/jfls.2000.0367
46. Oñate, E., Celigueta, M., Idelsohn, S.R.: Modeling bed erosion in free surface flows by the particle finite element method. Acta Geotechnica **1**(4), 237–252 (2006). DOI https://doi.org/10.1142/S0219876204000204
47. Oñate, E., Idelsohn, S.R.: The Particle Finite Element Method - An overview. International Journal of Computational Methods **1**(2), 267–307 (2004). DOI https://doi.org/10.1142/S0219876204000204
48. Park, S., Lee, S., Kim, J.: A surface reconstruction algorithm using weighted alpha shapes. In: Fuzzy Systems and Knowledge Discovery, Second International Conference, FSKD, p. 1141–1150 (2005). DOI https://doi.org/10.1007/11539506_143
49. Rajan, V.T.: Optimality of the Delaunay triangulation in $\mathbb{R}^d$. Discrete Comput. Geom. **12**, 189–202 (1994). DOI https://doi.org/10.1007/BF02574375
50. Rodríguez, J., Carbonell, J., Cante, J.: Continuous chip formation in metal cutting processes using the particle finite element method (PFEM). International Journal of Solids and Structures **120**, 81–102 (2017). DOI https://doi.org/10.1016/j.ijsolstr.2017.04.030
51. Sani, R.L., Gresho, P.M., Lee, R.L., Griffiths, D.F.: The cause and cure (?) of the spurious pressures generated by certain FEM solutions of the incompressible Navier-Stokes equations: Part 1. International Journal for Numerical Methods in Fluids **1**(1), 171 – 204 (1981). DOI https://doi.org/10.1002/fld.1650010104
52. Schreiber, R., Keller, H.B.: Driven cavity flows by efficient numerical techniques. Journal of Computational Physics **49**(2), 310–333 (1983). DOI https://doi.org/10.1016/0021-9991(83)90129-8
53. Sen, S., Mittal, S., Biswas, G.: Steady separated flow past a circular cylinder at low Reynolds numbers. Journal of Fluid Mechanics **620**, 89–119 (2009). DOI https://doi.org/10.1017/S0022112008004904
54. Sheard, G., Hourigan, K., Thompson, M.: Computations of the drag coefficients for low-Reynolds-number flow past rings. Journal of Fluid Mechanics **526**, 257–275 (2005). DOI https://doi.org/10.1017/S0022112004002836
55. Souto-Iglesias, A., Bulian, G., Botia-Vera, E.: A set of canonical problems in sloshing. part 2: Influence of tank width on impact pressure statistics in regular forced angular motion. Ocean Engineering **38**(16), 1823–1830 (2015). DOI https://doi.org/10.1016/j.oceaneng.2011.09.008
56. Sun, W.K., Zhang, L.W., Liew, K.M.: Adaptive particle refinement strategies in smoothed particle hydrodynamics. Computer Methods in Applied Mechanics and Engineering **389** (2022). DOI https://doi.org/10.1016/j.cma.2021.114276
57. Taneda, S.: Experimental investigation of the wake behind Cylinders and Plates at low Reynolds number. Journal of the Physical Society of Japan **11**(3), 302–307 (1956). DOI https://doi.org/10.1143/JPSJ.11.302
58. Tezduyar, S., Mittal, S., Ray, E., Shih, R.: Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity pressure elements. Computer Methods in Applied Mechanics and Engineering **95**(2), 221–242 (1992). DOI https://doi.org/10.1016/0045-7825(92)90141-6
59. Tournois, J., Srinivasan, R., Alliez, P.: Perturbing slivers in 3d delaunay meshes. pp. 157–173 (2009). DOI https://doi.org/10.1007/978-3-642-04319-2_10
60. Vacondio, R., Altomare, C., De Leffe, M., et al.: Grand challenges for Smoothed Particle Hydrodynamics numerical schemes. Computational Particle Mechanics **8**, 575–588 (2020). DOI https://doi.org/10.1007/s40571-020-00354-1
61. Vanella, M., Rabenold, P., Balaras, E.: A direct-forcing embedded-boundary method with adaptive mesh refinement for fluid-structure interaction problems. Journal of Computational Physics **229**(18), 6427–6449 (2010). DOI https://doi.org/10.1016/j.jcp.2010.05.003
62. Vanka, S.: Block-implicit multigrid solution of Navier-Stokes equations in primitive variables. Journal of Computational Physics **65**, 138–158 (1986). DOI https://doi.org/10.1016/0021-9991(86)90008-2
63. Wieselberger, C.v.: Neuere Feststellungen über die Gestze des Flussigkeits - und Luftwiderstands. Phys. Z. **22**(11), 321–328 (1921)
64. Williamson, C.H.K.: Vortex dynamics in the cylinder wake. Annual Review of Fluid Mechanics **28**(1), 477–539 (1996). DOI https://doi.org/10.1146/annurev.fluid.28.1.477

65. Yang, X., Kong, S.C.: Adaptive resolution for multiphase smoothed particle hydrodynamics. Computer Physics Communications **239**, 112–125 (2019). DOI https://doi.org/10.1016/j.cpc.2019.01.002
66. Zhang, H.Q., Uwe, F., Noack, B.R., et al.: On the transition of the cylinder wake. Physics of Fluids **7**(4), 779–794 (1995). DOI https://doi.org/10.1063/1.868601
67. Zhang, X., Krabbenhoft, K., Sheng, D., Li, W.: Numerical simulation of a flow-like landslide using the particle finite element method. Computational Mechanics **55**(1), 167–177 (2015). DOI https://doi.org/10.1007/s00466-014-1088-z