

Robust PFEM-FEM Partitioned Coupling Framework for the Simulation of Solids Under Complex Flow-Induced Loads

*A thesis submitted in fulfillment of the requirements for the
degree of Doctor of Philosophy (PhD) in Engineering Science*

Author : **Martin LACROIX**

Academic years 2021-2025

Candidate

Martin LACROIX

Supervisors

Prof. Jean-Philippe PONTHOT (*Supervisor*)

Dr. Romain BOMAN (*Co-supervisor*)

Prof. Jean-François REMACLE (*Co-tutelle supervisor*)

Jury Members

Prof. Thomas ANDRIANNE (*President*) University of Liège

Prof. Jean-Philippe PONTHOT University of Liège

Dr. Romain BOMAN University of Liège

Prof. Jean-François REMACLE Catholic University of Louvain

Prof. Joris DEGROOTE Ghent University

Prof. Harald VAN BRUMMELEN Eindhoven University of Technology

Prof. Marcela CRUCHAGA University of Santiago de Chile

Dr. Marco Lucio CERQUAGLIA Bureau Greisch

Abstract

Simulating solids under complex, flow-induced loads presents a significant challenge in fields like aerospace, turbomachinery, and biological systems. These applications often involve intricate Fluid-Structure Interactions (FSI), which include free surfaces, large deformations, and strong coupling effects, that traditional simulation methods struggle to capture effectively. The finite element code Metafor, developed by the MN2L lab of the University of Liège, can handle large deformations, plasticity, damage and fracture, thermo-mechanical interactions and solid-solid contact, making it ideal for advanced mechanical analysis. However, it cannot simulate complex FSI problems on its own. In applications where strong interactions between a liquid and solid occur, using simplified boundary conditions to represent fluid behavior on the solid surface is insufficient. This thesis addresses that limitation by introducing a framework that couples a Particle Finite Element Method (PFEM) fluid solver with Metafor using a fully partitioned approach. This coupling enables dynamic simulation of both domains and provides more accurate flow-induced boundary conditions for the solid structure.

The PFEM is a hybrid technique that combines the flexibility of particle-based methods with the precision of the Finite Element Method (FEM). The core idea of the PFEM is to apply the Lagrangian FEM framework for each time step, combined with a remeshing procedure between time steps that allows fluid parts to separate or merge. While PFEM has shown promising results in handling free-surface flows, splashing, and large deformations, challenges persist in mass conservation and mesh regularization, particularly in 3D configurations. Furthermore, the evolving and non-matching meshes between the fluid and solid solvers present a substantial difficulty in coupling the two domains.

This work advances the state of the art in FSI through a coupled PFEM-FEM framework applicable to 2D and 3D problems. The first set of contributions enhances the PFEM by introducing a novel mesh refinement strategy that hinders element quality degradation, specialized libraries to regularize mesh degeneracies, and a Level Set-based method to improve mass conservation (PFEM-LS). The second focus addresses numerical challenges related to the partitioned FSI coupling with evolving, non-matching meshes. These challenges are addressed by enforcing mechanical and thermal equilibrium between the solvers at each time step, supported by a flexible and efficient mesh interpolation technique. Particular attention has been given to the performance and scalability of the implementation. The proposed framework is validated against a wide range of experimental and numerical benchmarks from the literature. The variety of test cases and their agreement with published results confirm the robustness of the approach.

Acknowledgments

I would first like to thank all the members of the jury for the time they dedicated to reading and evaluating my work. In particular, I wish to express my sincere gratitude to my supervisor, Jean-Philippe Ponthot, for giving me the opportunity to pursue a PhD at the University of Liège and for his continuous support throughout the course of this thesis. I am also deeply thankful to my co-supervisor, Romain Boman, whose invaluable advice and guidance have greatly contributed to the quality of my research.

This work was carried out in collaboration with the research team of Jean-François Remacle from the University of Louvain. I am deeply grateful for this enriching collaboration, their invaluable support, stimulating discussions, and generous sharing of expertise throughout the course of this research.

I would also like to thank my colleagues and collaborators, without whom this work would not have been possible. Special thanks go to Luc Papeleux, Simon Février, and Eduardo Fernández for their expertise and insightful contributions to the technical aspects of my research. Last but not least, I am personally grateful to Thomas Leyssens, Romain Falla and Dorian Bogucki for the countless discussions we shared on numerical methods.

Finally, I would like to acknowledge the financial support provided by the Belgian National Fund for Scientific Research (F.R.S-FNRS) under the research project PFEM-FSI (PDR reference T.0183.21) and by the School of Engineering of the University of Liège. Their funding has been essential in allowing me to carry this research through to completion.

List of Abbreviations

AABB	Axis-Aligned Bounding Box
ALE	Arbitrary Lagrangian-Eulerian
BC	Boundary Condition
BGS	Block Gauss-Seidel
CAD	Computer Aided Design
CFL	Courant-Friedrichs-Lewy
CG	Conjugate Gradient
CPU	Central Processing Unit
CSC	Compressed Sparse Column
CSR	Compressed Sparse Row
DN	Dirichlet-Neumann
EDM	Element Deletion Method
FE	Finite Element
FEM	Finite Element Method
FFTB	Flux-Forward Temperature-Back
FS	Fluid-Structure
FSI	Fluid-Structure Interaction
FSPC	Fluid-Structure Partitioned Coupler
GETM	Geometric Element Transformation Method
GIL	Global Interpreter Lock
GMRES	Generalized Minimal Residual
GPU	Graphics Processing Unit
HFFB	Heat Transfer Coefficient-Forward Flux-Back
HFTB	Heat Transfer Coefficient-Forward Temperature-Back
IBM	Immersed Boundary Method
ICM	Interelement Crack Method
ILS	Inverse Least Squares
ILSM	Inverse Least-Squares and Surrogate Model
IQN	Interface Quasi-Newton
KD	K-Dimensional
LBB	Ladyzhenskaya Babuska Brezzi
LS	Level Set

MKL	Math Kernel Library
MM-ALE	Multi Material Arbitrary Lagrange Eulerian
MPI	Message Parsing Interface
MVJ	Multi-Vector Jacobian
ND	Neumann-Dirichlet
NR	Newton-Raphson
PDE	Partial Differential Equation
PFEM	Particle Finite Element Method
PFEM-LS	Particle Finite Element Method with Level Set
PSPG	Pressure-Stabilizing Petrov-Galerkin
RBF	Radial Basis Function
RCM	Reverse Cuthill-McKee
SPH	Smoothed-Particle Hydrodynamics
SPR	Superconvergent Patch Recovery
SRIPR	Selective Reduced Integration with Pressure Report
SVD	Singular Value Decomposition
TFFB	Temperature-Forward Flux-Back
TPS	Thin Plate Spline
VCPM	Volume-Compensated Particle Method
VOF	Volume of Fluid
XFEM	Extended Finite Element Method

Table of Contents

1	Introduction	8
1.1	Context and Motivations	9
1.2	State of the Art	10
1.3	Main Challenges	14
1.4	Outline and Objectives	16
1.5	Mathematical Notations	18
2	Particle Finite Element Method	19
2.1	Fluid Governing Equations	20
2.1.1	Time Integration	34
2.1.2	Fractional Step Method	37
2.1.3	Weak Compressibility	44
2.2	Building the Mesh	47
2.2.1	Remeshing with Edge Splitting	49
2.2.2	Numerical Examples	65
2.3	Conservation of Mass	81
2.3.1	Alpha-Shape with Level Set	84
2.3.2	Numerical Examples	91
3	Fluid-Structure Interactions	98
3.1	Introduction	99
3.1.1	The Structural Solver	99
3.1.2	Code Architecture	107
3.2	Partitioned Strategy	108
3.2.1	Interface Quasi-Newton Coupling	112
3.2.2	Numerical Examples	118
3.3	Non-Matching Meshes	130
3.3.1	Radial Basis Functions	130
3.3.2	Numerical Examples	137
3.3.3	Conclusion	149
3.4	Thermo-Mechanical Interaction	150
3.4.1	Partitioned Coupling Scheme	150
3.4.2	Numerical Examples	155

3.5	Computation Times	166
3.6	Material Rupture	184
3.6.1	Element Deletion Method	184
3.6.2	Numerical Examples	187
3.7	Example – Weak Compressibility	192
4	Conclusions	197
4.1	Overview	198
4.2	Conclusions of Chapter 2	199
4.3	Conclusions of Chapter 3	202
4.4	Perspectives	204
A	Elemental Matrices	208

Chapter 1

Introduction

1.1 Context and Motivations

Fluid-Structure Interactions (FSI) refer to the physical phenomena that occur when one or more solid structures interact with a fluid flow. FSI leads to complex, nonlinear, multi-physical interactions, playing a significant role in various scientific and engineering fields. It is particularly important in the design of aerospace vehicles, turbomachinery, wind turbines, flapping foil energy harvesters [1, 2] as well as in the stability analysis of bridge structures. Additionally, FSI is essential in biological fluid mechanics [3, 4] and biomechanics [5, 6]. It is also a major focus of research in the areas of aeroelasticity and hydroelasticity [7, 8]. Other examples of fluid-structure interactions include the simulation of particle sedimentation [9, 10], offshore and marine engineering [11, 12], electrostatics and electro-hydrodynamics [13, 14], or the analysis of bird-strike damage on airplanes [15, 16].

For most engineering problems involving FSI, developing analytical solutions is not feasible, and experiments can be expensive or unable to provide a complete understanding of the phenomena. As a result, numerical simulations have become the most effective approach. Significant progress in FSI modeling has been made over decades. Early computational studies of FSI, particularly in aeroelasticity, began in the 1970s [17, 18] and 1980s [19, 20], addressing structural vibration and fluid flow coupling through the Finite Element Method (FEM) for solids and simplified fluid models. By the 1990s, partitioned coupling methods, such as those developed by Farhat and Lesoinne [21, 22], allowed independent fluid and solid solvers to interact through a coupling interface. In 2004, to better manage free surface flows, topological changes, fragmentation, or droplet impacts, Idelsohn et al. [23, 24] introduced the Particle Finite Element Method (PFEM). The latter combines the flexibility of particle-based methods with the precision of finite elements, using a Lagrangian approach to naturally track evolving interfaces and automatic remeshing to prevent mesh distortion. At the same time, the Multi Material Arbitrary Lagrange Eulerian (MM-ALE) method [25, 26] has been developed to simulate air-blast and explosive detonation. The MM-ALE uses a coupled Eulerian-Lagrangian formulation in which the computational mesh can move independently of the material, and allow multiple materials to coexist within a single element.

Motivations

Metafor [27, 28] is an in-house Finite Element (FE) solver developed for simulating complex structural behaviors. It can handle large deformations, plasticity, damage and fracture, thermo-mechanical interactions and solid-solid contact, making it ideal for advanced mechanical analysis.

However, Metafor does not include any fluid modeling capabilities, meaning it cannot simulate realistic FSI problems on its own. In many applications, especially where strong interactions between a liquid and solid occur, using simplified Boundary Conditions (BC) to represent fluid behavior on the solid surface is insufficient. To address this limitation, Cerquaglia and Thomas [29, 30] developed a fully partitioned coupling framework between Metafor and a PFEM-based fluid solver, allowing for dynamic simulation of both domains, leading to more accurate flow-induced boundary conditions for the solid structure. While their work demonstrated proof-of-concept implementations, the use of PFEM within the FSI framework was limited to academic 2D benchmarks. This thesis takes the next step, extending the framework to 3D configurations and overcoming key challenges of the PFEM, which are discussed in a subsequent section. It enables the simulation of complex, dynamic loading conditions on the solid solver, including thermo-mechanical exchanges, material rupture, and flow through fractured zones. Essentially, our goal is to build on the work of Cerquaglia et al. [16, 31, 32] and move beyond the previous proof of concept, paving the way for industrial applications. In a recent development, Février [33] completely rewrote the PFEM code to support 3D fluid simulations. Thereby, the first step of this thesis is to revisit the FSI framework to enable strong partitioned coupling between Metafor and the newly developed fluid solver, called PFEM3D.

1.2 State of the Art

In the FEM, or in any alternative discretization technique, fluids and solids typically rely on distinct frameworks. The Lagrangian approach, commonly used in solid mechanics, is well-suited for problems involving free surfaces or large interface movements, as the computational mesh moves with the material. This eliminates the need for explicit boundary tracking, since the interface is naturally part of the model. However, deforming Lagrangian meshes can quickly become overly distorted, requiring frequent mesh refinement operations. In contrast, Eulerian frameworks, typically used in fluid mechanics, employ fixed meshes, thus avoiding the need for remeshing. However, accurately tracking moving boundaries in an Eulerian setup requires methods like Level Set (LS) [34, 35, 36] or Volume of Fluid (VOF) [37, 38], which significantly increase solver complexity. An alternative hybrid approach, known as Arbitrary Lagrangian-Eulerian (ALE) [39], can also be used. However, ALE is less effective at handling topological changes such as droplet separation, splashing, or crack propagation. Although these limitations can be addressed by the MM-ALE, it comes with higher complexity, and may require higher computational costs compared to purely Lagrangian or meshless methods such as Smoothed-Particle Hydrodynamics (SPH) in commercial software [40, 41, 42].

Immersed Boundary Method

State-of-the-art formulations for handling moving immersed or free interfaces in fluid dynamics simulations generally rely on an Eulerian framework, employing a fixed non-body-fitted mesh for the fluid domain. The solid domain is either treated as a rigid, fixed body or described using a Lagrangian formalism [43, 44, 45, 46]. Among these formulations, the Immersed Boundary Method (IBM) [47, 48] allows for the representation of complex geometries and moving interfaces without the need for mesh regeneration. IBM employs a composite of overlapping grids, where the background fluid mesh remains fixed, and the solid interface is embedded within it. The fundamental principle of IBM is to enforce the interaction between the fluid and the solid through additional forcing terms in the governing equations, rather than modifying the mesh topology. These forcing terms are introduced to satisfy kinematic constraints (velocity continuity) and dynamic conditions (force balance) at the interface. The method can be implemented in different ways, including the penalty approach, where an artificial force is added to penalize deviations from the interface conditions, or the projection method, where the velocity field is adjusted to enforce constraints more accurately.

Classical IBM formulations [48, 49] employs discrete delta functions to interpolate between the Eulerian fluid grid and the Lagrangian interface representation. These functions distribute the effects of the immersed structure onto the surrounding fluid nodes, ensuring a smooth transition between the solid and fluid phases. However, as the interface moves, cut-cells (fluid elements partially occupied by the solid) can become arbitrarily small, leading to numerical instabilities. To mitigate these issues, stabilizing penalty terms are often incorporated to improve the conditioning of the discrete system by maintaining consistency and stability in cut elements [50, 51].

Despite these enhancements, IBM formulations can still face challenges, particularly when dealing with highly dynamic interfaces, large deformations, or free surface interactions in 3D problems [49]. The requirement to accurately capture interface conditions while avoiding spurious oscillations often leads to intricate algorithms that demand careful tuning of numerical parameters. Alternatively, interface or free surface motion can be tracked using LS techniques. However, for both LS and IBM, the solid part covers an evolving part of the fixed Eulerian mesh. Thus, fluid degrees of freedom that are covered by the solid part have to be dynamically enabled, disabled and initialized as the interface evolves [52]. This is a source of complexity and potential inaccuracy that can be totally avoided while using a Lagrangian approach for the fluid, as the nodes naturally move with the interface and keep their physical properties.

Smoothed Particle Hydrodynamics

The SPH was originally introduced in 1977 for astrophysical simulations [53], and has since been widely used in computational fluid dynamics [54, 55]. It is a Lagrangian, meshless computational method using particles to represent the fluid, each carrying physical properties such as density, velocity, and pressure. Numerical integration involves summing contributions from neighboring particles weighted by a kernel function. Commonly used kernels are the Gaussian kernel, the cubic spline, or the Wendland C2 function with cut-off radius to limit the range of the neighborhood. This locality of interactions allows for efficient distribution of computations across multiple processors, making SPH particularly advantageous for parallel computing architectures [56, 57].

In SPH-based fluid-structure simulations, the structure is typically modeled as a moving boundary that influences fluid flow through boundary conditions and interaction forces. While SPH is well-suited for simulating free-surface flows and large deformations, accurately enforcing boundary conditions remains a technical challenge. Standard formulations often suffer from wall penetration issues. To mitigate this, techniques such as ghost particles, repulsive force models, and kernel corrections are commonly employed to enforce solid boundaries effectively [58, 59]. Additionally, SPH tends to exhibit numerical instability and lacks consistency when modeling elastic solid structures [54, 55, 60], which must be addressed for reliable simulations. Stability and convergence are also highly sensitive to particle distribution uniformity. As a result, most SPH implementations rely on uniform spatial resolution, limiting their efficiency in multiscale engineering applications, though recent research has focused on enabling non-uniform particle distributions [61].

Particle Finite Element Method

The PFEM is a recent discretization technique [23, 62] that holds both the features of particle methods to address mesh distortions, and of Lagrangian FEM for an accurate capturing of evolving interfaces as well as easy imposition of boundary conditions. A state-of-the-art review of the method can be found in [63]. The key idea of the method is the use of the classical FEM approach over each time step combined, in between different time steps, with a remeshing procedure that allows separation or merging of different parts of the fluid. The remeshing is based on the Delaunay tessellation [64] of the set of particles, which subdivides their convex hull into triangles in 2D or tetrahedra in 3D, ensuring that the circumsphere of each simplex* does not contain any other points. An α -shape [65] is then extracted from this tessellation to reconstruct the boundary of the fluid domain.

The PFEM was initially dedicated to fluid mechanics problems involving interfaces and free surfaces. Since then, it has been successfully used in a large panel of FSI problems [16, 66, 67]. However, it is important to note that the PFEM is not exempt from limitations. In particular, the shortcomings mostly stem from the remeshing which does not enforce the conservation of mass. Indeed, the α -shape alone is generally not sufficient to guarantee an accurate detection of the boundaries. This issue is even more significant in 3D, where obtaining a high-quality tessellation is challenging [68].

Fluid-Structure Interactions

When using body-fitted, Lagrangian finite element-based methods for FSI, a common technique is the monolithic (or unified) formulation, where the governing equations for the fluid and solid are solved simultaneously using the same solution scheme and variables in a coupled way [45, 66, 69, 70]. While robust and accurate in principle, the monolithic approach is inherently intrusive, as it requires solving a single system that directly incorporates both fluid and solid equations, which can increase implementation complexity and computational cost, especially for large systems. Furthermore, it inhibits the coupling of different software frameworks.

Alternatively, the loose coupling technique aims at solving the fluid and the solid equations sequentially, but may lead to instabilities [71, 72]. In the partitioned coupling, or staggered method, the flow and the displacement of the solid structure are computed separately and coupled thanks to a fluid-structure algorithm. The latter performs sub-iterations until convergence at the interface of the fluid and the solid parts, which guarantees the balance of action/reaction surface tractions [73, 74, 75]. This strategy is extremely versatile since two specialized software frameworks can be used together. For instance, it is possible to use an explicit solver for the fluid part and an implicit solver for the solid part, thus using the most appropriate technology for each domain. The main drawback of the partitioned coupling is that the number of iterations required to achieve convergence increases with the strength of the coupling and the complexity of the interaction, which can greatly slow down the simulation [31]. Moreover, communication delays or synchronization issues may arise if the two solvers run in parallel, especially in large distributed systems.

* A simplex represents the simplest possible polytope in any given dimension. Namely, it is a generalization of the concept of a triangle to higher dimensions, with a d -simplex having exactly $d + 1$ vertices.

1.3 Main Challenges

A major challenge in the PFEM arises from the remeshing process. Specifically, the boundary recognition algorithm, traditionally based on α -shape, focuses on the shape and size of each individual simplex, without considering the topological or physical characteristics of the fluid. As a result, this approach can alter the fluid boundary in a way that leads to spurious mass variations over successive remeshing steps. To address this, some researchers have turned to constrained Delaunay tessellations [76, 77], though its use has been primarily limited to plasticity problems, as it struggles with merging and separating bodies.

Falla et al. [78] proposed a technique based on a merging detection algorithm, enabling extreme (and local) mesh refinement near regions where fluid-fluid contact might occur. This approach is grounded in the idea that smaller finite elements lead to reduced mass variation by minimizing the scale at which the tessellation and the α -shape can alter the fluid boundary. However, this solution has two main drawbacks. First, it does not resolve the underlying issue of mass variation, as it only reduces the scale at which it occurs. Second, intensive mesh refinement can significantly increase the computational cost of solving the FE system, particularly in 3D, by raising the number of degrees of freedom. Lately, Fernández et al. [79] proposed a PFEM-LS that combines the LS technique with the α -shape, which shows promise for mass conservation without the drawbacks of previous methods. However, this approach has only been tested in 2D and introduces a non-negligible computational cost to the remeshing process.

Another major challenge directly related to remeshing is the automatic control of mesh resolution. Although the size of the FE mesh can be defined a priori to best suit the specific configuration being studied, the displacement of nodes due to the flow can randomize the resolution. This results in sparse regions where higher resolution is needed, or results in dense regions where it is not. Therefore, mesh adaptivity can only be achieved by frequently adding and removing particles.

Historically, new nodes are added at the centroids of finite elements that are too large with respect to the target mesh resolution [29, 80, 81]. Conversely, nodes are removed when they come too close from each other. While this approach may seem straightforward, centroid insertion tends to decrease the overall quality of the mesh. An alternative approach for refining the mesh is the Chew's Delaunay refinement [82]. The Delaunay refinement algorithm is guaranteed to improve the elements quality down to

$$B = \frac{r_c}{\min_e(L_e)} = 1 \quad (1.1)$$

where r_c is the circumradius of the considered simplex and $\min_e(L_e)$ is the length of its shortest edge. The only exception arises when the simplex contains constrained edges (such as those along a boundary) with an angle smaller than 60° . One of the primary drawbacks of Delaunay refinement is that it can generate very small elements relative to the target mesh resolution, which becomes a significant issue when precise control over mesh resolution is required. Additionally, this algorithm was originally proposed for 2D meshes only. More recently, Falla et al. [78] introduced an edge splitting approach that has been shown to produce good-quality elements on unstructured triangular meshes, while closely aligning with the target mesh resolution. However, this method was also proposed for 2D meshes only.

In addition to the above challenges, extending the remeshing process to 3D introduces new complexities. In 2D Delaunay tessellations, the empty sphere property [64] ensures that no other points from the point set lie inside the circle passing through the three vertices of any triangle. This property prevents the formation of skinny or degenerate triangles, as long as two nodes are not nearly coincident. In 3D, this property generalizes to a circumsphere, but it does not guarantee well-shaped tetrahedra and cannot prevent the formation of degenerate elements, often referred to as **slivers**. The presence of slivers in a FE mesh negatively affects the accuracy of the solution. In practice, slivers can prevent the linear solver from converging in implicit simulations and can reduce the critical time step in explicit simulations to nearly zero. For that reason, all slivers must be regularized. The regularization of slivers in the PFEM has been widely discussed in the literature [67, 83, 84, 85], yet it remains an open topic of ongoing research [86, 87, 88].

Another challenge, now related to the partitioned coupling with the PFEM for FSI, arises from the fact that the fluid mesh does not typically match the structural mesh. Specifically, the FS interface mesh on the fluid side undergoes frequent topological changes due to both the remeshing process and the evolution of the free surface. On the other hand, the FS interface on the solid side may also experience topological changes due to material rupture. This situation calls for a versatile and efficient interpolation algorithm that can facilitate the communication of nodal quantities between the two solvers in 3D. In a study from 2019 on the partitioned coupling of two FEM codes, Thomas et al. [89] demonstrated that the Radial Basis Function (RBF) interpolation is a promising candidate for this task, offering a fully meshless solution for transferring information between the fluid and solid domains.

1.4 Outline and Objectives

Chapter 2 details the PFEM and the weak formulation of the Navier-Stokes equations implemented in our in-house code PFEM3D [33]. Specifically, it introduces the governing equations and their discretization using the PSPG [90] and the fractional step [91] methods. Numerical examples, including liquid sloshing in a tank, dam breaks with obstacles, and droplet impacts, illustrate the ability of the method to simulate free surface flows. More importantly, the chapter focuses on addressing three key objectives. To the best of the author’s knowledge, the proposed solutions are original contributions of this work to the field of research:

1 – *Improved Mesh-Refinement in 2D/3D*

While the remeshing procedure effectively prevents mesh distortion due to large deformations, it requires the repeated addition and removal of nodes in the fluid domain to maintain element sizes aligned with the target mesh density. However, the implementation used prior to this thesis leads to a gradual decline in element quality over time and lacks support for local refinement based on physical or geometric criteria. This work introduces an alternative approach, inspired by the edge splitting method of Falla et al. [78], which is suitable for both 2D and 3D simulations.

2 – *Regularization of Mesh Degeneracies*

In 3D Delaunay tessellation, the empty sphere property [64] does not ensure well-shaped tetrahedra and cannot prevent the creation of slivers. Our objective is to integrate an open-source mesh optimization and tetrahedralization library, such as TetGen [92] or Gmsh [93, 94], into the PFEM code. This work is conducted in close collaboration with the Gmsh development team and the lab of Remacle at UCLouvain, which is designing an algorithm to enable efficient remeshing with theoretical quality guarantees [87, 95], specifically tailored for use in PFEM applications. Mesh optimization involve node relocation, known as vertex smoothing, node addition or deletion, or elementary edge and face flips to locally improve the quality of the elements and regularize slivers.

3 – *Improved Mass Conservation*

The main drawback of the PFEM is the impossibility to preserve mass exactly. When a particle comes into contact with a boundary, new elements are generated to connect the two domains, and typically leads to non-physical mass creation as these connecting elements are artificially introduced without a corresponding representation in the last-known configuration of the fluid domain.

Conversely, domain separation and particle detachment contribute to mass loss due to removal of fluid elements connecting the two domains. In this work, we propose a computationally efficient approach, inspired by the PFEM-LS of Fernandez et al. [79], to address mass variation issues in both 2D and 3D simulations.

Chapter 3 presents the partitioned strategy adopted for coupling PFEM3D with Metafor in order to simulate mechanical, thermal, and thermo-mechanical problems. This section validates the approach on different benchmarks from the literature. Numerical examples, including the flow-induced bending of a rubber gate, the flow-induced reconfiguration of buoyant and flexible aquatic vegetation, the natural convection in a cavity with a flexible beam, the floating and warming of a solid sphere in a water tank, and a test case of water-jet cutting, illustrate the robustness of the method to simulate FSI. More importantly, the chapter also focuses on addressing three key objectives:

4 – *Multi-Vector Jacobian Coupling*

The Interface Quasi-Newton (IQN) technique, with an approximation for the Inverse of the Jacobian by a Least-Squares model (IQN-ILS), is an advanced, strong coupling strategy that offers good convergence and stability when coupling black-box solvers [75]. Moreover, the convergence rate of the IQN can be further improved by reusing information from previous time steps into the ILS model. However, it requires a filtering algorithm to remove contradicting information or to enhance the solvability of the resulting linear system [96, 97]. Thereby, a manual fine-tuning of filtering tolerances and of the number of reuses is needed [31]. To circumvent this issue, the implementation of an Interface Quasi-Newton method with Multi-Vector Jacobian (IQN-MVJ) [98] is considered and compared to the IQN-ILS.

5 – *PFEM-FEM with Non-Matching Meshes*

While the previous implementation of the FSI coupling by Cerquaglia et al. [31] provided a foundational basis for our current approach, it did not account for dynamic changes in the interface mesh topology. Namely, the partitioned coupling with a PFEM solver was only tested with matching meshes or simplified 2D configurations where the PFEM solver does not modify the FS interface mesh during the simulation. Since remeshing and local refinement are a fundamental aspect of the PFEM, in more general scenarios, the interface mesh undergoes repeated topological changes. As a result, there is a need to enhance the interpolation algorithms to better handle material rupture, non-matching meshes, and evolving interfaces.

6 – Thermo-Mechanical FSI Coupling

The work of Cerquaglia focused essentially on isothermal mechanical FSI using the PFEM [29]. More recently, Bobach [99] extended the original PFEM code to include thermal effects for simulating phase changes in 2D. This capability was later expanded to 3D geometries by Février [100] within the new PFEM framework. However, the implementation of thermo-mechanical FSI coupling between Metafor and PFEM3D remained unaddressed. This integration requires thorough validation and verification against both experimental data and numerical benchmarks.

Chapter 4 concludes the thesis by summarizing the key findings from the previous chapters. It includes a dedicated section highlighting how the FSI coupling framework and PFEM3D, enhanced through the contributions of this work, have been adopted by other researchers. A curated list of publications and related projects is thus provided to showcase additional applications of the codes and to guide readers interested in exploring these extensions. Finally, the chapter outlines several perspectives for future work.

1.5 Mathematical Notations

In physics, a field is a physical quantity represented by a scalar, vector or tensor that is defined at each point in space and time. Throughout this document, scalar fields are denoted by italic symbols, vector fields and second-order tensor are represented using bold italic symbols, and fourth-order tensors are represented by blackboard bold symbols. For example, the pressure field is denoted as p , the velocity field is denoted as $\mathbf{v}$ and the Hooke tensor is denoted as $\mathbb{H}$. When expressing finite element formulations of partial differential equations, the discretized counterparts of field quantities on the finite element mesh, as well as generic vectors and matrices, are represented using bold roman symbols. For instance, $\mathbf{v}$ denotes the vector of nodal velocities, while $\mathbf{p}$ denotes the vector of nodal pressures.

Ideally, a rigorous use of capitalization distinguishes tensors and matrices, which are denoted by capital letters, from vectors and scalars, which are denoted by lowercase letters. However, the notations commonly used in the continuum mechanics literature do not always adhere to this distinction. For example, the Cauchy stress tensor is conventionally denoted by a small sigma σ despite being a second-order tensor rather than a vector. Consequently, we prioritize conventional notations in accordance with the literature, even at the expense of the capitalization rule stated above.

Chapter 2

Particle Finite Element Method

2.1 Fluid Governing Equations

Unlike the classical Lagrangian framework where $\mathbf{x}(\mathbf{X}, t)$ represents the time-dependent position of a particle initially located at $\mathbf{X}$, we interpret $\mathbf{x}$ as a spatial coordinate independent of time. The time-dependent aspect of the problem arises from the evolution of the computational domain $\Omega(t)$ which moves and deforms over time, as shown in Figure 1. This interpretation is particularly relevant in numerical formulations where equations are solved on a deforming mesh, and the computational grid follows the material motion without requiring an explicit reference to an initial particle positions.

This section provides an overview of the equations for Newtonian fluid motion with thermal effects in a PFEM framework. It is important to emphasize that the section does not present an original derivation developed within the scope of this thesis. Instead, its purpose is to introduce the reader to PSPG [90] stabilization, fractional step [91] methods, or more generally, the equations and integration schemes used in our PFEM code. These formulations are employed throughout all the test cases presented later in this thesis. Additionally, this section aims to make the document more self-contained, reducing the need for the reader to consult external references to understand the content of the work. Readers already familiar with these concepts may choose to skip this section and proceed to the next one, where the two first objectives of the thesis are addressed.

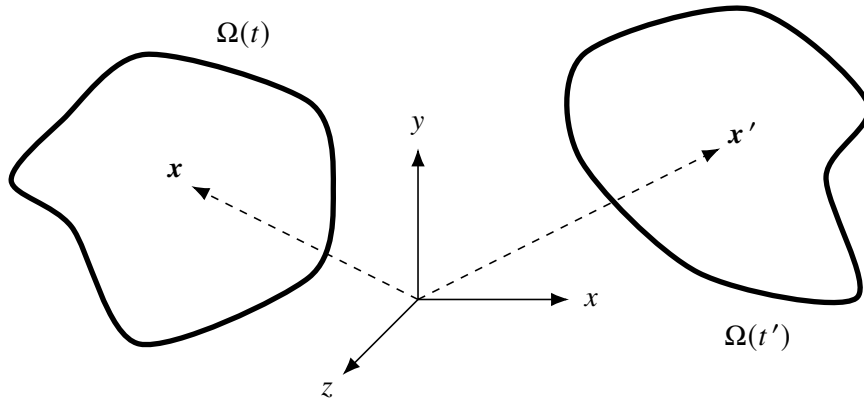


Figure 1: Current configuration of a continuum body in the Euclidean space. Material points acquire new spatial coordinates $\mathbf{x} = (x, y, z)$ as they move, but $\mathbf{x}$ itself remains a reference for describing physical quantities at any given instant.

The motion of a fluid is described by the Navier-Stokes equations, a set of coupled Partial Differential Equations (PDE) that represent momentum balance and incorporate mass conservation. To account for thermal effects in a fluid medium, these equations must be coupled with the energy conservation equation. In this work, we essentially consider single-phase, incompressible fluids without complex physical phenomena (such as combustion or radiation). Thereby, the density is independent of the pressure and the final set of governing equations is given by

$$\begin{cases} \nabla \cdot \boldsymbol{\sigma} + \rho \mathbf{g} = \rho \frac{d\mathbf{v}}{dt} & \text{in } \Omega \\ \nabla \cdot \mathbf{v} = 0 & \text{in } \Omega \\ \nabla \cdot \mathbf{q} + \rho c_p \frac{dT}{dt} = s & \text{in } \Omega \end{cases} \quad (2.1)$$

Here, $\boldsymbol{\sigma}$ is the Cauchy stress tensor, $\mathbf{v}$ is the velocity field, ρ is the density, $\mathbf{g}$ is the acceleration of gravity, T is the temperature field, $\mathbf{q}$ is the heat flux, c_p is the specific heat capacity at constant pressure and s is a volume heat generation rate. Due to the Lagrangian nature of the PFEM, the material points coincide with the grid nodes, and no convective terms appear in the governing equations. Consequently, material derivatives reduce to time derivatives. The constitutive equations for an incompressible Newtonian fluid following the Fourier's conduction law are

$$\begin{aligned} \boldsymbol{\sigma} &= \mu (\nabla \mathbf{v} + \nabla \mathbf{v}^T) - p \mathbf{I} \\ \mathbf{q} &= -k \nabla T \end{aligned} \quad (2.2)$$

where μ denotes the dynamic viscosity, p represents the pressure field, $\mathbf{I}$ stands for the identity matrix, and k refers to the thermal conductivity of the material, which is assumed to be isotropic. It is worth noting that the choice of an incompressible Newtonian fluid is not a limitation of the method. Indeed, this framework could be extended to encompass compressible, multiphase, or non-Newtonian fluids [100, 101, 102, 103].

Dirichlet and Neumann boundary conditions are critical for solving partial differential equations, as they ensure the problem is well-posed, yielding a unique and stable solution. Without appropriate boundary and initial conditions, PDE may have infinitely many solutions or none at all. The system of equations (2.1) and (2.2) combined with suitable initial and boundary conditions, allows modeling the entire flow field and solving for the unknown flow variables at every point in space and time. These boundary conditions are expressed as follows:

$$\begin{array}{l|l} \mathbf{q} \cdot \mathbf{n} = \phi & \text{on } \Gamma_Q \\ \boldsymbol{\sigma} \cdot \mathbf{n} = \mathbf{t} & \text{on } \Gamma_\Sigma \end{array} \quad \left| \quad \begin{array}{l} T = \bar{T} & \text{on } \Gamma_T \\ \mathbf{v} = \bar{\mathbf{v}} & \text{on } \Gamma_V \end{array} \right. \quad (2.3)$$

Here, $\mathbf{n}$ denotes the unit outer normal to the fluid domain. The boundary of the computational domain is given by $\Gamma = \Gamma_Q \cup \Gamma_T = \Gamma_\Sigma \cup \Gamma_V$ where Γ_Q is the Neumann boundary for prescribed heat flux, Γ_T is the Dirichlet boundary for prescribed temperature, Γ_Σ is the Neumann boundary for prescribed surface tractions and Γ_V is the Dirichlet boundary for prescribed velocity. As Neumann and Dirichlet BC cannot be applied on the same boundary for the same unknown, one has $\Gamma_Q \cup \Gamma_T = \emptyset$ and $\Gamma_\Sigma \cup \Gamma_V = \emptyset$. However, it is possible to combine velocity and temperature-related boundary conditions such that their respective boundaries overlap.

Conservation of Momentum

The weak formulation of the momentum conservation equation can be derived by first multiplying the momentum equation by a vector field $\mathbf{w}$, often referred to as test function. This function has the same dimension as the velocity field and belongs to an appropriate finite-dimensional subspace of the solution space. The test function is arbitrary within the domain, but kinematically admissible. This means it must be continuous, (at least piecewise) differentiable, and is required to vanish on the Dirichlet boundary. The resulting equation is then integrated over the fluid domain, leading to

$$\begin{aligned} \int_{\Omega} \mathbf{w}^\top (\nabla \cdot \boldsymbol{\sigma}) \, dx &= \int_{\Omega} \mathbf{w}^\top \rho \frac{d\mathbf{v}}{dt} \, dx - \int_{\Omega} \mathbf{w}^\top \rho \mathbf{g} \, dx \\ \Leftrightarrow \int_{\Omega} \nabla \cdot (\mathbf{w}^\top \boldsymbol{\sigma}) \, dx - \int_{\Omega} \boldsymbol{\sigma} : \nabla \mathbf{w} \, dx &= \int_{\Omega} \mathbf{w}^\top \rho \frac{d\mathbf{v}}{dt} \, dx - \int_{\Omega} \mathbf{w}^\top \rho \mathbf{g} \, dx \end{aligned} \quad (2.4)$$

for all $\mathbf{w}$ in Ω . The double-dot product is defined such that $\boldsymbol{\sigma} : \nabla \mathbf{w} = \text{tr}(\boldsymbol{\sigma} \nabla \mathbf{w}^\top)$ and appears in the second form of the equation as a result of an integration by part of the left-hand side. The next step involves applying the divergence theorem. This theorem states that the integral of a divergence term over a volume can be rewritten as a surface integral over the boundary of that volume. This yields:

$$\int_{\Gamma} \mathbf{w}^\top (\boldsymbol{\sigma} \cdot \mathbf{n}) \, dx - \int_{\Omega} \boldsymbol{\sigma} : \nabla \mathbf{w} \, dx = \int_{\Omega} \mathbf{w}^\top \rho \frac{d\mathbf{v}}{dt} \, dx - \int_{\Omega} \mathbf{w}^\top \rho \mathbf{g} \, dx \quad (2.5)$$

The surface integral in (2.5) is determined by the Neumann and Dirichlet boundary conditions, which apply to their respective portions of the boundary where the velocity and the surface tractions have been prescribed. Since Γ_v and Γ_Σ are mutually exclusive, this surface integral can be rewritten as follows:

$$\int_{\Gamma} \mathbf{w}^\top (\boldsymbol{\sigma} \cdot \mathbf{n}) d\mathbf{x} = \underbrace{\int_{\Gamma_v} \mathbf{w}^\top (\nabla \cdot \boldsymbol{\sigma}) d\mathbf{x}}_{\mathbf{w} = 0} + \int_{\Gamma_\Sigma} \mathbf{w}^\top \mathbf{t} d\mathbf{x} \quad (2.6)$$

Here, the first integral over Γ_v vanishes because the test function is defined to be zero on the Dirichlet boundary. Finally, by substituting (2.6) into (2.5) and rearranging the terms to isolate the unknowns on the left-hand side, the weak form of the momentum conservation equation is obtained as:

$$\int_{\Omega} \boldsymbol{\sigma} : \nabla \mathbf{w} d\mathbf{x} + \int_{\Omega} \mathbf{w}^\top \rho \frac{d\mathbf{v}}{dt} d\mathbf{x} = \int_{\Omega} \mathbf{w}^\top \rho \mathbf{g} d\mathbf{x} + \int_{\Gamma_\Sigma} \mathbf{w}^\top \mathbf{t} d\mathbf{x} \quad (2.7)$$

The next step is to rewrite the above equation in a form that is more suited for implementation. To achieve this, we adopt Voigt notation, which simplifies tensor representation by reducing dimensionality, making mathematical operations more efficient by reducing the number of operations while preserving all essential information. In this work, we use the following convention for the stress tensor and the gradient operator:

$$\hat{\boldsymbol{\sigma}} = \begin{pmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{31} \end{pmatrix} \quad \boldsymbol{\partial} = \begin{pmatrix} \partial/\partial x & 0 & 0 \\ 0 & \partial/\partial y & 0 \\ 0 & 0 & \partial/\partial z \\ \partial/\partial y & \partial/\partial x & 0 \\ 0 & \partial/\partial z & \partial/\partial y \\ \partial/\partial z & 0 & \partial/\partial x \end{pmatrix} \quad (2.8)$$

Here, the gradient operator $\boldsymbol{\partial}$ is defined such that $\boldsymbol{\sigma} : \nabla \mathbf{w} = (\boldsymbol{\partial} \mathbf{w})^\top \hat{\boldsymbol{\sigma}}$. It is important to note that Voigt notation conventions may vary across different fields of applications, and by extension, the formulation of finite element matrices depends on the specific convention adopted. Applying this Voigt notation to Equation (2.7) yields the following expression for the conservation of momentum:

$$\int_{\Omega} (\partial \mathbf{w})^{\top} \hat{\boldsymbol{\sigma}} \, d\mathbf{x} + \int_{\Omega} \mathbf{w}^{\top} \rho \frac{d\mathbf{v}}{dt} \, d\mathbf{x} = \int_{\Omega} \mathbf{w}^{\top} \rho \mathbf{g} \, d\mathbf{x} + \int_{\Gamma_{\Sigma}} \mathbf{w}^{\top} \mathbf{t} \, d\mathbf{x} \quad (2.9)$$

Spatial discretization is achieved by subdividing the integration domain into a set of finite elements, collectively forming the computational mesh. Further details regarding the construction of the mesh are provided in the next section. Within each element, fields are approximated using basis functions $N_j(\mathbf{x})$. In particular, Lagrange basis functions satisfy the property that their value is 1 at the generating node with index j and zero at all other nodes of the element. An example is provided in Figure 2. The approximated field is expressed as a weighted sum of those basis functions, with the weights corresponding to the nodal values. In each element of the FE mesh, one can write

$$\begin{aligned} \mathbf{v}(\mathbf{x}, t) &= \mathbf{N}_3(\mathbf{x})^{\top} \mathbf{v}(t) \\ \mathbf{w}(\mathbf{x}, t) &= \mathbf{N}_3(\mathbf{x})^{\top} \mathbf{w}(t) \end{aligned} \quad (2.10)$$

Here, $\mathbf{N}_k = \mathbf{I}_k \otimes \mathbf{N}$ is a rectangular matrix, where $\otimes$ denotes the Kronecker product between the identity matrix $\mathbf{I}_k$ of size $K \times K$ and the vector $\mathbf{N}$, whose components are the basis functions of the element. An explicit expression of $\mathbf{N}_3$ for a tetrahedral element is provided in the annex A as an example. The vectors $\mathbf{v}$ and $\mathbf{w}$ represent the nodal velocity and the nodal values of the test function, respectively. Additionally, we define the matrix containing the gradients of the basis functions as

$$\mathbf{B}_3 = \begin{pmatrix} \frac{\partial \mathbf{N}}{\partial x} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{N}}{\partial y} & \mathbf{0} & \frac{\partial \mathbf{N}}{\partial z} \\ \mathbf{0} & \frac{\partial \mathbf{N}}{\partial y} & \mathbf{0} & \frac{\partial \mathbf{N}}{\partial x} & \frac{\partial \mathbf{N}}{\partial z} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{N}}{\partial z} & \mathbf{0} & \frac{\partial \mathbf{N}}{\partial y} & \frac{\partial \mathbf{N}}{\partial x} \end{pmatrix} \quad (2.11)$$

Here, the gradient is defined such that $\partial \mathbf{w}(\mathbf{x}, t) = \mathbf{B}_3(\mathbf{x})^{\top} \mathbf{w}(t)$ in order to follow the notation convention established in (2.10). In order to keep the formulation general and independent of the type of material, the constitutive equations have not yet been introduced in the weak formulation. Injecting the discretizations (2.10) and (2.11) into the weak formulation (2.9) gives

$$\int_{\Omega} \mathbf{w}^T \mathbf{B}_3 \hat{\boldsymbol{\sigma}} d\mathbf{x} + \int_{\Omega} \rho \mathbf{w}^T \mathbf{N}_3 \mathbf{N}_3^T \frac{d\mathbf{v}}{dt} d\mathbf{x} = \int_{\Omega} \rho \mathbf{w}^T \mathbf{N}_3 \mathbf{g} d\mathbf{x} + \int_{\Gamma_{\Sigma}} \mathbf{w}^T \mathbf{N}_3 \mathbf{t} d\mathbf{x} \quad (2.12)$$

Since the vectors of nodal unknowns are independent of the spatial coordinates, they can be factored out of the integrals. Additionally, the weak form holds for any kinematically admissible test function, allowing $\mathbf{w}$ to be eliminated from the discrete formulation. This yields a system of equations that can be directly solved for the primary unknowns:

$$\int_{\Omega} \mathbf{B}_3 \hat{\boldsymbol{\sigma}} d\mathbf{x} + \int_{\Omega} \rho \mathbf{N}_3 \mathbf{N}_3^T d\mathbf{x} \frac{d\mathbf{v}}{dt} = \int_{\Omega} \rho \mathbf{N}_3 \mathbf{g} d\mathbf{x} + \int_{\Gamma_{\Sigma}} \mathbf{N}_3 \mathbf{t} d\mathbf{x} \quad (2.13)$$

In this thesis, we exclusively adopt the Stokes constitutive equation (2.2) for a Newtonian, incompressible fluid. Within this framework, the Voigt representation of the stress tensor and its FE discretization can be expressed as follows:

$$\begin{aligned} \hat{\boldsymbol{\sigma}} &= \mu \mathbf{F}(\partial \mathbf{v}) - \mathbf{k} p \\ &= \mu \mathbf{F} \mathbf{B}_3^T \mathbf{v} - \mathbf{k} \mathbf{N}_1^T \mathbf{p} \end{aligned} \quad (2.14)$$

Here, the velocity field has been discretized in space as shown in (2.10) and the pressure field has been discretized in space by $p = \mathbf{N}_1^T \mathbf{p}$. Note that since the pressure is a scalar field, its discretization simply reuses the vector of basis functions $\mathbf{N}_1 = \mathbf{N}$ because $\mathbf{I}_1 = 1$. The vector $\mathbf{k}$ and the matrix $\mathbf{F}$ are given by

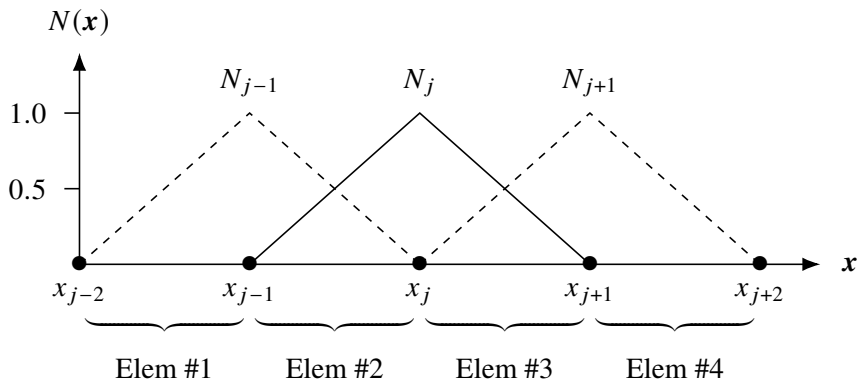


Figure 2: Example of 1D mesh with 4 finite elements and linear Lagrange basis functions. The Lagrange basis functions satisfy the property that their value is 1 at the generating node, and zero at all other nodes of the element.

$$\mathbf{k} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{F} = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.15)$$

In order to simplify the notation, we define the stiffness matrix $\mathbf{K}$ which represents the viscous term, the mass matrix $\mathbf{M}$ which accounts for inertial effects, the pressure matrix $\mathbf{D}$ which enforces the incompressibility condition, and the vector $\mathbf{f}$ which contains the applied loads. Injecting (2.14) into (2.13) yields

$$\begin{aligned} & \underbrace{\int_{\Omega} \mu \mathbf{B}_3 \mathbf{F} \mathbf{B}_3^{\top} d\mathbf{x}}_{\mathbf{K}} \mathbf{v} - \underbrace{\int_{\Omega} \mathbf{B}_3 \mathbf{k} \mathbf{N}_1^{\top} d\mathbf{x}}_{\mathbf{D}^{\top}} \mathbf{p} + \underbrace{\int_{\Omega} \rho \mathbf{N}_3 \mathbf{N}_3^{\top} d\mathbf{x}}_{\mathbf{M}} \frac{d\mathbf{v}}{dt} \\ &= \underbrace{\int_{\Omega} \rho \mathbf{N}_3 \mathbf{g} d\mathbf{x} + \int_{\Gamma_{\Sigma}} \mathbf{N}_3 \mathbf{t} d\mathbf{x}}_{\mathbf{f}} \end{aligned} \quad (2.16)$$

Surface Tension

Surface tension arises from the cohesive forces between molecules in a liquid. These forces are stronger at the surface because molecules there are not surrounded by the same liquid on all sides, unlike those in the bulk. Although surface tension may not always be the dominant physical factor in our simulations, it plays a crucial role in maintaining the cohesion of fluid particles and ensuring the smoothness of the free surface. For example, in water sloshing simulations, neglecting surface tension can lead to nonphysical instabilities in oscillation amplitudes. In practice, surface tractions experience a discontinuity due to the change in stress at the interface between the fluid and its surrounding medium. This discontinuity, associated with surface tension, is expressed as

$$\delta \mathbf{t} = \nabla_s \gamma - \gamma \kappa \mathbf{n} = \nabla_s \cdot \boldsymbol{\sigma}_s \quad (2.17)$$

where γ is the strain-independent surface tension, also called the surface energy, $\mathbf{n}$ is the outer unit normal to the interface and κ is the local curvature [104]. The local curvature at any point on the surface of a liquid is defined as the reciprocal of the radius of curvature of the surface at that point. Curvature can be interpreted as how much the surface deviates from being flat at a specific point. For a general 3D surface, which can be curved in two directions, the mean curvature is used in surface tension formulas, and is defined as:

$$\kappa = \frac{1}{R_1} + \frac{1}{R_2} \quad (2.18)$$

Here, R_1 and R_2 are the principal curvature radii. The surface gradient, denoted as ∇_s in (2.17), is a differential operator similar to the conventional gradient. The key difference is that it operates along a surface. In other words, it represents the orthogonal projection of the conventional gradient onto the surface. Let f be a differentiable scalar field on a surface S , its surface gradient is given by

$$\begin{aligned} \nabla_s f &= \nabla f - \mathbf{n}(\mathbf{n} \cdot \nabla f) \\ &= \underbrace{(\mathbf{I} - \mathbf{n} \otimes \mathbf{n})}_{\mathbf{P}} \cdot \nabla f \end{aligned} \quad (2.19)$$

Here, $\mathbf{P}$ is the surface-projection matrix [105]. The tensor σ_s in (2.17) is referred as the surface stress tensor in the literature because it is commonly used to describe the contribution of the surface stiffness to the overall material stiffness when simulating nanosolids. Indeed, the interface-to-volume ratio is so high that the interface energy, which is usually negligible with respect to the bulk energy in solid mechanics, can no longer be neglected. According to [106, 107], the surface stress is related to the surface-strain energy Ψ by Shuttleworth's equation:

$$\sigma_s = \gamma \mathbf{I}_s + \frac{\partial \Psi(\epsilon_s)}{\partial \epsilon_s} \quad (2.20)$$

Here, $\mathbf{I}_s$ is the unit tensor for surfaces and ϵ_s is the surface strain tensor [108, 109]. The first term of the right-hand-side accounts for the contribution of surface tension, while the second term accounts for the surface stiffness for nanosolids. In this work, we are interested in (macroscopic) fluids, thus the constitutive equation (2.20) reduces to the sole contribution of the surface tension:

$$\boldsymbol{\sigma}_s = \gamma \mathbf{P} \quad (2.21)$$

where $\mathbf{P}$ acts as the identity tensor for surface quantities. This formulation of surface tension offers the significant advantage of eliminating the need to compute curvature explicitly on the finite element mesh. Incorporating surface tension in the conservation of momentum introduces an additional boundary condition:

$$\delta \mathbf{t} = \nabla_s \cdot \boldsymbol{\sigma}_s \quad \text{on } \Gamma_s \quad (2.22)$$

Here, $\Gamma_s \subset \Gamma$ and represents the portion of the boundary where surface tension is applied, which in our implementation is limited to the free surface of the fluid, ignoring fluid-solid surface tension. With this boundary condition considered, the surface integral (2.6) is completed by the following additional term:

$$\begin{aligned} \int_{\Gamma_s} \mathbf{w}^\top \delta \mathbf{t} \, d\mathbf{x} &= \int_{\Gamma_s} \mathbf{w}^\top (\nabla_s \cdot \boldsymbol{\sigma}_s) \, d\mathbf{x} \\ &= \int_{\Gamma_s} \nabla_s \cdot (\mathbf{w}^\top \boldsymbol{\sigma}_s) \, d\mathbf{x} - \int_{\Gamma_s} \boldsymbol{\sigma}_s : \nabla_s \mathbf{w} \, d\mathbf{x} \end{aligned} \quad (2.23)$$

The next step consists in developing the first term of the right-hand side by applying the equivalent of the divergence theorem on a surface. To this end, let φ be the boundary of Γ_s (which is represented by a curve in 3D or by a point in 2D) and let $\mathbf{m}$ be the normal of φ tangent to Γ_s . According to Gurtin et al. [110, 111], the following alternative form of the surface divergence theorem:

$$\int_{\Gamma_s} \nabla_s \cdot \boldsymbol{\tau} \, d\mathbf{x} = \int_{\varphi} \boldsymbol{\tau} \cdot \mathbf{m} \, d\mathbf{x} \quad (2.24)$$

holds for any tangential vector field $\boldsymbol{\tau}$. In particular, the above relation is also valid for $\boldsymbol{\tau} = \boldsymbol{\gamma} \cdot \mathbf{w}$ where $\boldsymbol{\gamma}$ is a tangential tensor field. Since $\boldsymbol{\sigma}_s$ is by definition a tangential tensor field with respect to the free surface, we can inject (2.24) into (2.23) with $\boldsymbol{\tau} = \boldsymbol{\sigma}_s \cdot \mathbf{w}$ to obtain the following equality

$$\int_{\Gamma_s} \mathbf{w}^\top \delta \mathbf{t} \, d\mathbf{x} = \int_{\varphi} \mathbf{w}^\top (\boldsymbol{\sigma}_s \cdot \mathbf{m}) \, d\mathbf{x} - \int_{\Gamma_s} \boldsymbol{\sigma}_s : \nabla_s \mathbf{w} \, d\mathbf{x} \quad (2.25)$$

Here, φ represents the contact line between the liquid phase, the solid phase and the surrounding gas or void, as illustrated in Figure 3. When the fluid surface is closed, $\varphi = \emptyset$ and the corresponding integral vanishes. In other cases, neglecting this term is equivalent to prescribing a contact angle $\theta = 90^\circ$. This simplification is consistently adopted throughout this work to reduce computational complexity. The FE discretization of (2.25) is addressed by introducing the Voigt notation for the surface stress tensor:

$$\hat{\sigma}_s = \begin{pmatrix} \sigma_{11}^s \\ \sigma_{22}^s \\ \sigma_{33}^s \\ \sigma_{12}^s \\ \sigma_{23}^s \\ \sigma_{31}^s \end{pmatrix} = \gamma \begin{pmatrix} P_{11} \\ P_{22} \\ P_{33} \\ P_{12} \\ P_{23} \\ P_{31} \end{pmatrix} = \gamma \hat{\mathbf{P}} \quad (2.26)$$

Here, the constitutive relation (2.21) for σ_s has been employed, and $\hat{\mathbf{P}}$ is the Voigt notation of the surface projection operator. As in the previous developments, the spatial discretization of the test function on the finite element mesh is done accordingly to (2.10). Neglecting the contribution of the contact line integral in (2.25), the FE formulation of the surface tension term becomes

$$\int_{\Gamma_s} \mathbf{w}^\top \delta t \, dx = - \int_{\Gamma_s} \mathbf{w}^\top \mathbf{B}_3 \mathbf{U} \gamma \hat{\mathbf{P}} \, dx \quad (2.27)$$

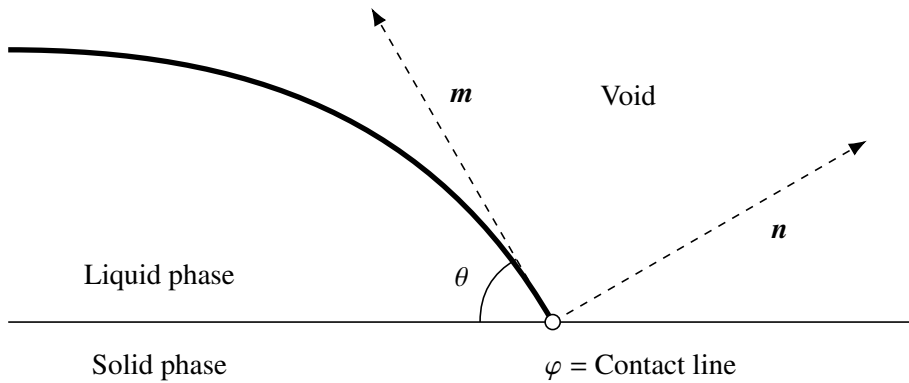


Figure 3: Liquid droplet placed on a flat solid surface in 2D. The variable θ is the contact angle between the liquid and the solid, φ is the contact line between the liquid, the solid and the void. The free surface is represented in bold.

Here, the matrix product $\mathbf{B}_3\mathbf{U}$ can be interpreted as the surface gradient of the basis functions in Voigt notation. The matrix $\mathbf{U}$ is a square, non-symmetric projection matrix that contains the components of the projection operator. It is worth noting that the expression of $\mathbf{U}$ depends on the convention used for the Voigt notation, thus it may vary across the literature. In this work, it can be shown that

$$\mathbf{U} = \begin{pmatrix} P_{11}^2 & P_{12}^2 & P_{31}^2 & 2P_{11}P_{12} & 2P_{31}P_{12} & 2P_{11}P_{31} \\ P_{21}^2 & P_{22}^2 & P_{23}^2 & 2P_{12}P_{11} & 2P_{23}P_{22} & 2P_{12}P_{23} \\ P_{31}^2 & P_{23}^2 & P_{33}^2 & 2P_{23}P_{31} & 2P_{23}P_{33} & 2P_{33}P_{31} \\ P_{11}P_{12} & P_{12}P_{22} & P_{31}P_{23} & P_{11}P_{22} + P_{12}^2 & P_{31}P_{22} + P_{23}P_{12} & P_{31}P_{12} + P_{11}P_{23} \\ P_{12}P_{31} & P_{23}P_{22} & P_{23}P_{33} & P_{22}P_{31} + P_{12}P_{23} & P_{22}P_{33} + P_{23}^2 & P_{32}P_{31} + P_{12}P_{33} \\ P_{11}P_{31} & P_{12}P_{23} & P_{33}P_{31} & P_{12}P_{31} + P_{11}P_{23} & P_{31}P_{23} + P_{33}P_{12} & P_{33}P_{11} + P_{31}^2 \end{pmatrix} \quad (2.28)$$

Finally, the contribution of the surface tension from the free surface is integrated into the right-hand side of the conservation of momentum (2.16). The vector of nodal values for the test function $\mathbf{w}$ in (2.27) can be removed from the discrete formulation, resulting in the following finite element formulation:

$$\mathbf{K}\mathbf{v} - \mathbf{D}^\top \mathbf{p} + \mathbf{M} \frac{d\mathbf{v}}{dt} = \underbrace{\int_{\Omega} \rho \mathbf{N}_3 \mathbf{g} \, d\mathbf{x} + \int_{\Gamma_\Sigma} \mathbf{N}_3 \mathbf{t} \, d\mathbf{x} - \int_{\Gamma_s} \mathbf{B}_3 \mathbf{U} \gamma \hat{\mathbf{P}} \, d\mathbf{x}}_{\mathbf{f}} \quad (2.29)$$

Conservation of Mass

The weak formulation of the conservation of mass, also known as the continuity equation, can be derived similarly to the conservation of momentum. First, both sides of the equation are multiplied by an arbitrary scalar field u which belongs to an appropriate finite-dimensional subspace of the solution space. The resulting expression is then integrated over the fluid domain, leading to

$$\int_{\Omega} u \nabla \cdot \mathbf{v} \, d\mathbf{x} = 0 \quad (2.30)$$

for all u in Ω . Since the PFEM ultimately relies on automatic remeshing procedures based on a Delaunay tessellation of a cloud of nodes, the finite elements used for spatial discretization are linear triangles or tetrahedra. The incompressible Navier-Stokes equations, as formulated in (2.1), inherently impose a constraint on the velocity and pressure spaces, known as the LBB condition [112, 113, 114]. This condition requires a certain compatibility between the finite element spaces used for velocity and pressure in order to ensure stability and uniqueness of the numerical solution [115, 116]. Specifically, when using linear finite elements for both velocity and pressure, the LBB condition is not satisfied, leading to pressure instabilities.

To circumvent this issue, Hughes and Tezduyar [90, 117] proposed the Pressure-Stabilizing Petrov-Galerkin (PSPG) formulation. The latter allows the use of equal-order interpolation for velocity and pressure by reformulating the continuity equation (2.30) to relax the strict incompressibility constraint. Mathematically, the PSPG introduces an element-weighted residual term $\mathcal{R}$ in the weak form which represents the imbalance in the momentum equation. This relaxation effectively stabilizes the pressure field while preserving the accuracy of the solution:

$$\int_{\Omega} u \nabla \cdot \mathbf{v} \, d\mathbf{x} - \underbrace{\int_{\Omega} \left(\nabla \cdot \boldsymbol{\sigma} + \rho \mathbf{g} - \rho \frac{d\mathbf{v}}{dt} \right) \cdot \nabla u \frac{\tau}{\rho} \, d\mathbf{x}}_{\mathcal{R}} = 0 \quad (2.31)$$

Here, τ is the PSPG coefficient obtained by a multidimensional generalization of the optimal coefficient for 1D space-time formulation. As a side remark, the value of τ proposed in [118] is slightly different from the one proposed in [90]. This is because the original definition reveals only first-order accuracy due to local truncation errors, while its variant in [118] has been shown to provide second-order accuracy:

$$\tau = \left[\left(\frac{2}{\Delta t} \right)^2 + \left(\frac{2\|\mathbf{v}\|}{h} \right)^2 + 9 \left(\frac{4\mu}{\rho h^2} \right)^2 \right]^{-1/2} \quad (2.32)$$

Here, Δt denotes the time step. In a finite element mesh, h represents the characteristic size of the current element, typically defined as the diameter of its circumsphere. The term $\|\mathbf{v}\|$ refers to the norm of the average fluid velocity within the element. The standard non-stabilized Galerkin formulation is recovered when $\tau = 0$.

Equation (2.31) can be further simplified by invoking (2.2) and noting that $\Delta \mathbf{v} = 0$ for linear elements. This is because the viscosity contribution in $\nabla \cdot \boldsymbol{\sigma}$ involves second-order derivatives of the velocity field, which vanish when linear basis functions are used for spatial discretization:

$$\int_{\Omega} u \nabla \cdot \mathbf{v} \, d\mathbf{x} + \int_{\Omega} \nabla p \cdot \nabla u \frac{\tau}{\rho} \, d\mathbf{x} + \int_{\Omega} \tau \frac{d\mathbf{v}}{dt} \cdot \nabla u \, d\mathbf{x} = \int_{\Omega} \tau \mathbf{g} \cdot \nabla u \, d\mathbf{x} \quad (2.33)$$

The finite element discretization is then performed in the same way as for the conservation of momentum. Specifically, the integration domain is discretized in finite elements using the same computational mesh. The pressure, velocity field, and test function are expressed using the linear basis functions associated with the elements. To this end, we define:

$$u(\mathbf{x}, t) = \mathbf{N}_1(\mathbf{x})^\top \mathbf{u}(t) \\ \mathbf{B}_1 = \begin{pmatrix} \frac{\partial \mathbf{N}}{\partial x} & \frac{\partial \mathbf{N}}{\partial y} & \frac{\partial \mathbf{N}}{\partial z} \end{pmatrix} \quad (2.34)$$

Here, $\mathbf{u}$ is the vector of nodal values for the test function and $\mathbf{B}_1$ is a matrix containing the gradients of the basis functions used to discretize the pressure gradient on the finite element mesh. Injecting the above definition into (2.33) results in

$$\int_{\Omega} \mathbf{u}^\top \mathbf{N}_1 \mathbf{k}^\top \mathbf{B}_3^\top \mathbf{v} \, d\mathbf{x} + \int_{\Omega} \mathbf{u}^\top \mathbf{B}_1 \mathbf{B}_1^\top \mathbf{p} \frac{\tau}{\rho} \, d\mathbf{x} \\ + \int_{\Omega} \tau \mathbf{u}^\top \mathbf{B}_1 \mathbf{B}_3^\top \frac{d\mathbf{v}}{dt} \, d\mathbf{x} = \int_{\Omega} \tau \mathbf{u}^\top \mathbf{B}_1 \mathbf{g} \, d\mathbf{x} \quad (2.35)$$

Since the vectors of nodal unknowns are not functions of the coordinates, they can be factored out of the integrals. Moreover, because the system of equations must hold for arbitrary test functions, the vectors of nodal values $\mathbf{u}$ can be eliminated from the discrete formulation. To simplify the notations, we define the matrix $\mathbf{C}$ associated with the inertia, the matrix $\mathbf{L}$ related to the pressure contribution of the stabilization, and the vector $\mathbf{h}$ representing the contribution of body forces:

$$\underbrace{\int_{\Omega} \mathbf{N}_1 \mathbf{k}^\top \mathbf{B}_3^\top \, d\mathbf{x}}_{\mathbf{D}} \mathbf{v} + \underbrace{\int_{\Omega} \frac{\tau}{\rho} \mathbf{B}_1 \mathbf{B}_1^\top \, d\mathbf{x}}_{\mathbf{L}} \mathbf{p} + \underbrace{\int_{\Omega} \tau \mathbf{B}_1 \mathbf{B}_3^\top \, d\mathbf{x}}_{\mathbf{C}} \frac{d\mathbf{v}}{dt} = \underbrace{\int_{\Omega} \tau \mathbf{B}_1 \mathbf{g} \, d\mathbf{x}}_{\mathbf{h}} \quad (2.36)$$

Conservation of Energy

Some test cases presented in this work involve thermo-mechanical interactions, where the thermal behavior is governed by the conservation of energy. The derivation of the finite element formulation for energy conservation follows the same steps as for the momentum and continuity equations, so only a brief outline is provided here. Starting with the strong form of the conservation of energy in (2.1), both sides are multiplied by a thermally admissible scalar test function u and integrated over the fluid domain:

$$\int_{\Omega} u \nabla \cdot \mathbf{q} \, d\mathbf{x} + \int_{\Omega} u \rho c_p \frac{dT}{dt} \, d\mathbf{x} = \int_{\Omega} u s \, d\mathbf{x} \quad (2.37)$$

for all u in Ω . As a reminder, $\mathbf{q}$ is the heat flux, c_p is the specific heat capacity at constant pressure, and s is a source term. Internal heat generation, such as chemical reactions or radiation absorption (for instance, laser heating), should be accounted for in the source term when applicable. Recently, Février et al. [100] used the PFEM to simulate melt pool dynamics, including vaporization, where s becomes particularly significant. By performing integration by parts on the left-hand side, applying the divergence theorem to the first integral, and incorporating the constitutive equation (2.2) along with the corresponding Neumann boundary condition, we obtain

$$\int_{\Omega} k \nabla u \cdot \nabla T \, d\mathbf{x} + \int_{\Omega} u \rho c_p \frac{dT}{dt} \, d\mathbf{x} = \int_{\Omega} u s \, d\mathbf{x} - \int_{\Gamma_q} u \phi \, d\mathbf{x} \quad (2.38)$$

Here, the prescribed heat flux in (2.3) is applied on Γ_q and u must vanish on the Dirichlet boundary. The computational domain is then discretized into finite elements using the same mesh as for the momentum and continuity equations. The temperature field, the heat sources, and the test function are approximated using linear basis functions associated with each element. Following the same procedure as before, the resulting FE formulation is expressed as

$$\underbrace{\int_{\Omega} k \mathbf{B}_1 \mathbf{B}_1^T \, d\mathbf{x}}_{\mathbf{S}} \mathbf{T} + \underbrace{\int_{\Omega} \rho c_p \mathbf{N}_1 \mathbf{N}_1^T \, d\mathbf{x}}_{\mathbf{R}} \frac{d\mathbf{T}}{dt} = \underbrace{\int_{\Omega} \mathbf{N}_1 \mathbf{s} \, d\mathbf{x} - \int_{\Gamma_q} \mathbf{N}_1 \phi \, d\mathbf{x}}_{\mathbf{b}} \quad (2.39)$$

2.1.1 Time Integration

The Navier-Stokes equations, previously discretized in space using the finite element method, are now discretized with respect to time. For this purpose, an implicit backward Euler scheme is applied. This choice is not a limitation of the PFEM, as more advanced time integration methods have also been explored in the literature [119]. In the backward Euler method, time derivatives are approximated using a finite difference formulation. The acceleration thus writes

$$\frac{d\mathbf{v}}{dt} \simeq \frac{\mathbf{v}_{n+1} - \mathbf{v}_n}{\Delta t} \quad (2.40)$$

Here, n denotes the current time step, and $n + 1$ refers to the unknown field at the next step. A key advantage of implicit schemes is that they remove the need to satisfy the Courant-Friedrichs-Lewy condition [120]. Instead, the time step Δt is selected based on geometric constraints and accuracy requirements. Although the backward Euler method introduces numerical dissipation, this effect decreases with smaller time steps [121, 122]. In the PFEM framework, the Lagrangian formulation imposes an additional constraint because node movement must remain small enough to avoid excessive mesh distortion or element flipping between successive remeshing steps. Applying the backward Euler scheme to the finite element formulations of mass and momentum conservation leads to the following system:

$$\underbrace{\begin{pmatrix} \mathbf{K}_{n+1} + \frac{1}{\Delta t} \mathbf{M}_{n+1} & -\mathbf{D}_{n+1}^\top \\ \frac{1}{\Delta t} \mathbf{C}_{n+1} + \mathbf{D}_{n+1} & \mathbf{L}_{n+1} \end{pmatrix}}_{\mathbf{p}_{n+1}} \begin{pmatrix} \mathbf{v}_{n+1} \\ \mathbf{p}_{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{f}_{n+1} + \frac{1}{\Delta t} \mathbf{M}_{n+1} \mathbf{v}_n \\ \frac{1}{\Delta t} \mathbf{C}_{n+1} \mathbf{v}_n + \mathbf{h}_{n+1} \end{pmatrix} \quad (2.41)$$

The temperature field is treated as constant during the mechanical solution phase. However, the matrix coefficients may depend on temperature. In such cases, a fully coupled iterative approach is used within each time step to achieve thermo-mechanical equilibrium. This process involves alternately solving the mechanical and thermal equations until both solutions converge [100].

Despite this, the system (2.41) remains inherently nonlinear, as the matrix coefficients also depend on the mechanical state, such as the current positions of the nodes. To solve this nonlinear system, iterative methods such as Newton-Raphson (NR) iterations are employed to ensure equilibrium at each time step:

$$\mathbf{p}_{k+1} = \mathbf{p}_k - \mathbf{J}_k^{-1} \mathbf{r}_k \quad (2.42)$$

Here, $\mathbf{p}$ represents the vector of unknown pressure and velocity, k is the index of the nonlinear iteration within the current time step, $\mathbf{J}$ denotes the system's Jacobian, also known as the tangent matrix, and $\mathbf{r}$ is the residual, which approaches zero as $\mathbf{p}$ converges to the actual solution. In practice, the Jacobian matrix in (2.42) is not inverted explicitly. Instead, each NR iteration involves solving a linear system:

$$\mathbf{p}_{k+1} = \mathbf{p}_k - \mathbf{J}_k \backslash \mathbf{r}_k \quad (2.43)$$

Here, we use the backslash notation $\mathbf{A} \backslash \mathbf{b}$ to denote the numerical solution to the linear system $\mathbf{Ax} = \mathbf{b}$ for the vector of unknowns $\mathbf{x}$. Thereby, the choice of the linear solver is critical and depends heavily on the properties of the Jacobian matrix. Further details on the linear solvers and their performance are provided in the next chapter. For the PSPG system (2.41), the residual reads

$$\mathbf{r}_k = \begin{pmatrix} \mathbf{r}_M^k \\ \mathbf{r}_C^k \end{pmatrix} \quad (2.44)$$

Here, $\mathbf{r}_M^k$ represents the momentum residual evaluated at the k -th NR iteration, while $\mathbf{r}_C^k$ corresponds to the mass conservation residual at the same iteration. Both are computed as the difference between the left and right-hand sides of equation (2.41), with $\mathbf{p}_k$ denoting the current solution at iteration k of the time t_{n+1} , and $\mathbf{p}_n$ representing the converged solution from the previous time step:

$$\begin{aligned} \mathbf{r}_M^k &= \left(\mathbf{K}_k + \frac{1}{\Delta t} \mathbf{M}_k \right) \mathbf{v}_k - \mathbf{D}_k^\top \mathbf{p}_k - \mathbf{f}_k - \frac{1}{\Delta t} \mathbf{M}_k \mathbf{v}_n \\ \mathbf{r}_C^k &= \left(\frac{1}{\Delta t} \mathbf{C}_k + \mathbf{D}_k \right) \mathbf{v}_k + \mathbf{L}_k \mathbf{p}_k - \frac{1}{\Delta t} \mathbf{C}_k \mathbf{v}_n - \mathbf{h}_k \end{aligned} \quad (2.45)$$

The subscript k given to matrices and vectors indicates that they are constructed using the updated position, velocity, and pressure values at the k -th iteration of the NR algorithm. An expression of the Jacobian matrix has been given in [100] as follows:

$$\mathbf{J}_k = \begin{pmatrix} \frac{\partial \mathbf{r}_M^k}{\partial \mathbf{v}_k} & \frac{\partial \mathbf{r}_M^k}{\partial \mathbf{p}_k} \\ \frac{\partial \mathbf{r}_C^k}{\partial \mathbf{v}_k} & \frac{\partial \mathbf{r}_C^k}{\partial \mathbf{p}_k} \end{pmatrix} = \begin{pmatrix} \frac{\partial \mathbf{r}_M^k}{\partial \mathbf{v}_k} & -\mathbf{D}_k^\top \\ \frac{\partial \mathbf{r}_C^k}{\partial \mathbf{v}_k} & \mathbf{L}_k \end{pmatrix} \quad (2.46)$$

The components involving a derivative with respect to the velocity can be computed by numerical perturbation. In theory, the solution to (2.41) is achieved when the residual reaches zero. In practice, however, the NR algorithm does not guarantee an exact solution in a finite number of iterations. Thereby, the stopping criterion for the NR process is a non-dimensional scalar expression of the residual, normalized by the right-hand side. The NR algorithm terminates when

$$\begin{aligned} \varepsilon_M^k &= \frac{\|\mathbf{r}_M^k\|}{\|\mathbf{f}_k\|} < tol \\ \varepsilon_C^k &= \frac{\|\mathbf{r}_C^k\|}{\|\mathbf{h}_k\|} < tol \end{aligned} \quad (2.47)$$

or when the maximum number of iterations is reached. In the second case, the current time step is restarted with a reduced time increment $\Delta t' = \Delta t/2$, and this process continues iteratively until (2.47) is satisfied. If convergence is achieved with $\Delta t' < \Delta t$, the time increment is progressively increased to restore the original time step size. The adaptive time-stepping strategy enhances the stability of the scheme when complex or rapid physical phenomena occur, while also allowing a return to larger time steps when possible to optimize computational efficiency.

It is worth noting that computing $\mathbf{J}_k$ at each iteration may become computationally expensive, especially with a perturbation approach. In this work, a modified NR approach has been used. The latter consists in reusing the initial Jacobian $\mathbf{J}_0$ from the first iteration, in all subsequent iterations of the corresponding time step. The modified NR has a lower convergence rate than the classical NR, but the time gained by skipping the recomputation of the Jacobian largely compensates for the increased number of iterations needed to reach the desired tolerance.

The thermal problem is solved similarly to the PSPG system. During the thermal step, the mechanical quantities, including the position of the nodes, are treated as constant. The time derivative of the temperature is approximated by a finite difference, and is injected into the FE formulation of the conservation of energy:

$$\left(\mathbf{S}_{n+1} + \frac{1}{\Delta t} \mathbf{R}_{n+1}\right) \mathbf{T}_{n+1} = \frac{1}{\Delta t} \mathbf{R}_{n+1} \mathbf{T}_n + \mathbf{b}_{n+1} \quad (2.48)$$

The residual $\mathbf{r}_l$ in the NR algorithm at iteration l is defined as the difference between the left and right-hand sides of Equation (2.48). Here, we used the index l rather than k to better distinguish thermal iterations from mechanical ones. The analytical expressions of the Jacobian matrix and the residual vector read

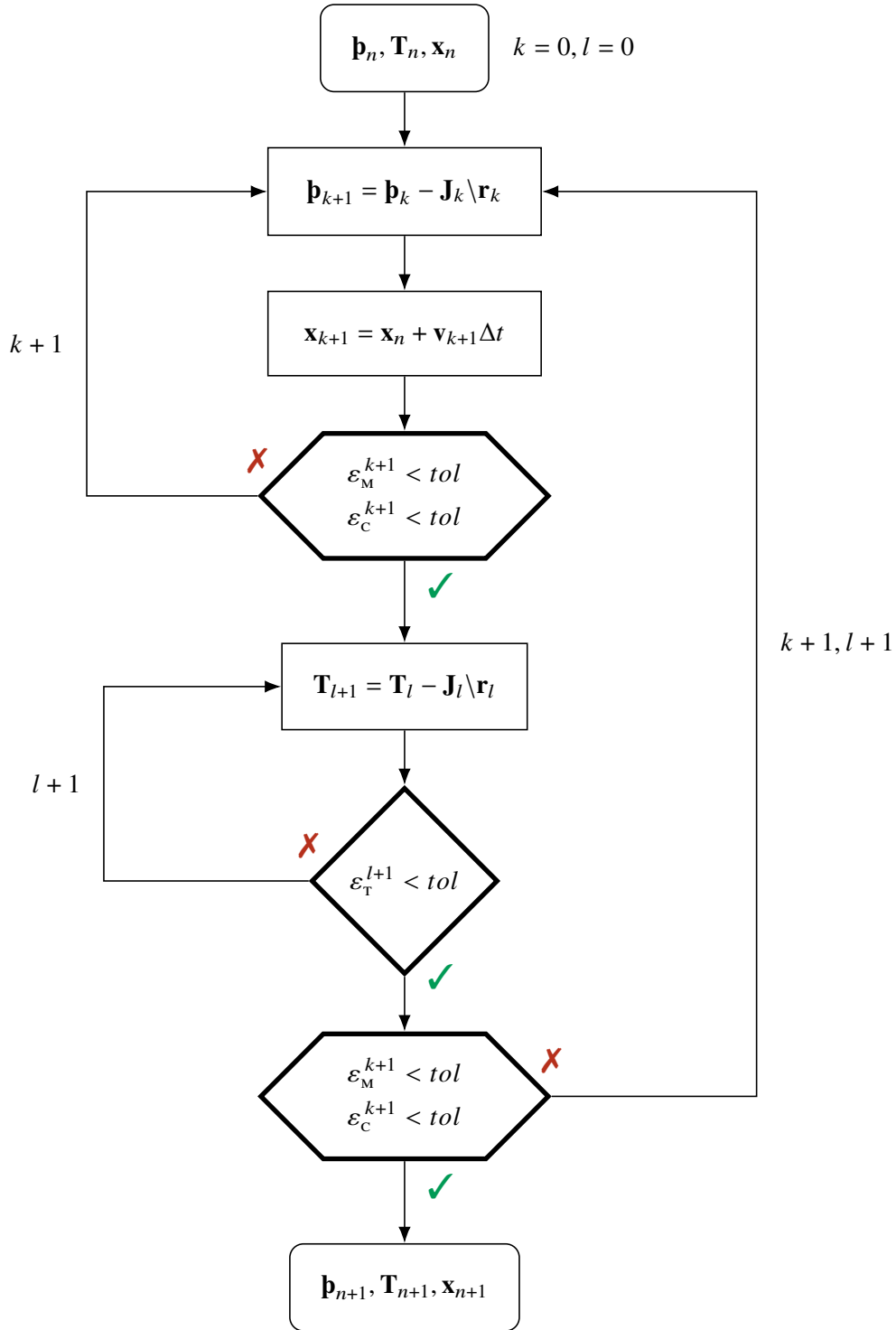
$$\begin{aligned} \mathbf{J}_l &= \frac{\partial \mathbf{r}_l}{\partial \mathbf{T}_l} = \frac{1}{\Delta t} \mathbf{R}_l + \mathbf{S}_l \\ \mathbf{r}_l &= \left(\mathbf{S}_l + \frac{1}{\Delta t} \mathbf{R}_l\right) \mathbf{T}_l - \frac{1}{\Delta t} \mathbf{R}_l \mathbf{T}_n - \mathbf{b}_l \end{aligned} \quad (2.49)$$

Similarly to the mechanical system, the Newton-Raphson algorithm is stopped when the residual reaches the desired tolerance, or when the maximum number of iterations has been exceeded. In the latter case, the time step is restarted with a smaller time increment. A summary of the coupling strategy is presented in Algorithm 1.

$$\varepsilon_{\tau}^l = \frac{\|\mathbf{r}_l\|}{\|\mathbf{b}_l\|} < tol \quad (2.50)$$

2.1.2 Fractional Step Method

While the PSPG offers a robust monolithic approach to solve the incompressible Navier-Stokes equations, it can become computationally expensive for large-scale 3D simulations. This is primarily due to the resulting non-symmetric, saddle-point-type linear system with poor conditioning, which require the use of general, less specialized solvers like LU decomposition or GMRES [123]. To address this problem, the fractional step method is considered. The fractional step scheme is a projection method designed to break down the original Navier-Stokes system into smaller sub-problems by decoupling the pressure and velocity variables.



Algorithm 1: Solution strategy for solving thermo-mechanical problems with the PSPG, between t_n and t_{n+1} . In order to simplify the analyses that follow, the same tolerance value is used for both pressure, velocity and temperature.

This decomposition introduces an implicit stabilization effect by separating the momentum and continuity equations [91, 124]. Specifically, the pressure correction step addresses the incompressibility constraint and projects the velocity field onto a divergence-free space, mitigating the instabilities that occur when using linear basis functions for both velocity and pressure. Starting from the strong form of the Navier-Stokes equations, the Chorin-Temam scheme [91, 125] introduces the pressure increment δp and discretizes the acceleration as follows:

$$\begin{aligned}\frac{d\mathbf{v}}{dt} &\simeq \frac{\hat{\mathbf{v}}_{n+1} - \mathbf{v}_n}{\Delta t} \\ \delta p_{n+1} &\simeq p_{n+1} - \lambda p_n\end{aligned}\tag{2.51}$$

where λ is a user-defined splitting parameter, and $\hat{\mathbf{v}}$ represents an intermediate velocity. It can be shown that the Chorin-Temam scheme is first-order accurate when $\lambda = 0$ and second-order accurate, but less stable, when $\lambda = 1$ [126]. The subscript n indicates the time step index. The first step of this projection scheme involves substituting the intermediate velocity into the conservation of momentum:

$$\begin{cases} \nabla \cdot \boldsymbol{\sigma}_{n+1} + \rho \mathbf{g} = \rho \frac{\hat{\mathbf{v}}_{n+1} - \mathbf{v}_n}{\Delta t} & \text{in } \Omega \\ \hat{\mathbf{v}}_{n+1} = \bar{\mathbf{v}} & \text{on } \Gamma_v \end{cases}\tag{2.52}$$

The stress tensor $\boldsymbol{\sigma}_{n+1}$ is expressed using the constitutive equation for an incompressible Newtonian fluid, based on the intermediate velocity and on the known pressure from the previous time step. The finite element formulation of (2.52) can then be derived by applying the same procedure as outlined earlier:

$$\left(\mathbf{K}_{n+1} + \frac{1}{\Delta t} \mathbf{M}_{n+1} \right) \hat{\mathbf{v}}_{n+1} = \mathbf{f}_{n+1} + \frac{1}{\Delta t} \mathbf{M}_{n+1} \mathbf{v}_n - \mathbf{D}_{n+1}^\top \lambda \mathbf{p}_n\tag{2.53}$$

The solution to the above system provides the velocity increment for the next time step based on the final pressure and velocity vectors at the previous time t_n . In the context of a Newton-Raphson iterative procedure, the Jacobian of the system and the residual vector are deduced from (2.53) as follows:

$$\begin{aligned}\hat{\mathbf{J}}_v^k &= \frac{\partial \hat{\mathbf{r}}_v^k}{\partial \hat{\mathbf{v}}_k} \\ \hat{\mathbf{r}}_v^k &= \left(\mathbf{K}_k + \frac{1}{\Delta t} \mathbf{M}_k \right) \hat{\mathbf{v}}_k - \mathbf{f}_k - \frac{\mathbf{M}_k}{\Delta t} \mathbf{v}_n + \mathbf{D}_k^\top \lambda \mathbf{p}_n\end{aligned}\tag{2.54}$$

Here, the subscript k denotes the index of the NR iteration relative to the current time, between t_n and t_{n+1} . The next phase of the Chorin-Temam scheme is called the projection step or the pressure correction. This projection step ensures that the updated velocity field satisfies the incompressibility condition by projecting the intermediate velocity onto a divergence-free space using the pressure field. This is achieved by solving

$$\begin{cases} \rho \frac{\mathbf{v}_{n+1} - \hat{\mathbf{v}}_{n+1}}{\Delta t} + \nabla \delta p = 0 & \text{in } \Omega \\ \nabla \cdot \mathbf{v}_{n+1} = 0 & \text{in } \Omega \\ \mathbf{v}_{n+1} \cdot \mathbf{n} = \bar{\mathbf{v}} \cdot \mathbf{n} & \text{on } \Gamma_v \end{cases}\tag{2.55}$$

An important consideration is that only the normal component of the final velocity can be explicitly controlled by the boundary conditions, leaving the tangential component unconstrained. Moreover, the pressure increment δp_{n+1} and the final velocity $\mathbf{v}_{n+1}$ remain coupled, as both appear in the first equation of the system (2.55). These issues are addressed by taking the divergence of the first equation and enforcing the divergence-free condition on $\mathbf{v}_{n+1}$ by injecting the second equation:

$$\begin{cases} \Delta(\delta p_{n+1}) = \frac{\rho}{\Delta t} \nabla \cdot \hat{\mathbf{v}}_{n+1} & \text{in } \Omega \\ \mathbf{n} \cdot \nabla \delta p_{n+1} = 0 & \text{on } \Gamma_v \end{cases}\tag{2.56}$$

This leads to a Poisson equation for the pressure increment. However, the system (2.56) imposes a non-physical boundary condition on δp , a topic that has been extensively discussed in the literature [127, 128]. Temam [127] interprets δp not as an approximation of the physical pressure, but as an auxiliary mathematical variable required to construct a projection operator within the functional framework of the Navier-Stokes equations. The FE formulation of (2.56) is given by

$$\underbrace{\int_{\Omega} \frac{1}{\rho} \mathbf{B}_1 \mathbf{B}_1^\top d\mathbf{x}}_{\mathbf{H}} \delta \mathbf{p}_{n+1} = \frac{1}{\Delta t} \underbrace{\int_{\Omega} \mathbf{N}_1 \mathbf{k}^\top \mathbf{B}_3^\top d\mathbf{x}}_{\mathbf{D}} \hat{\mathbf{v}}_{n+1}\tag{2.57}$$

It is worth noting that (2.56) lacks a Dirichlet condition and thus admits a family of solutions that differ by an arbitrary constant. To ensure the uniqueness of the solution, the pressure is typically prescribed on the free surface. However, this Dirichlet condition introduces several drawbacks. First, the continuity equation cannot be exactly satisfied on pressure-constrained elements, resulting in an end-of-step velocity $\mathbf{v}_{n+1}$ that is not strictly divergence-free [29, 129]. Second, the prescribed pressure must incorporate the natural pressure profile arising from the surface tension contribution, which is unknown a priori. Nevertheless, the Jacobian and residual vector for (2.57) are given by

$$\begin{aligned}\mathbf{J}_p^k &= \frac{\partial \mathbf{r}_p^k}{\partial \delta \mathbf{p}_k} = \mathbf{H}_k \\ \mathbf{r}_p^k &= \mathbf{H}_k \delta \mathbf{p}_k - \frac{1}{\Delta t} \mathbf{D}_k \hat{\mathbf{v}}_k\end{aligned}\tag{2.58}$$

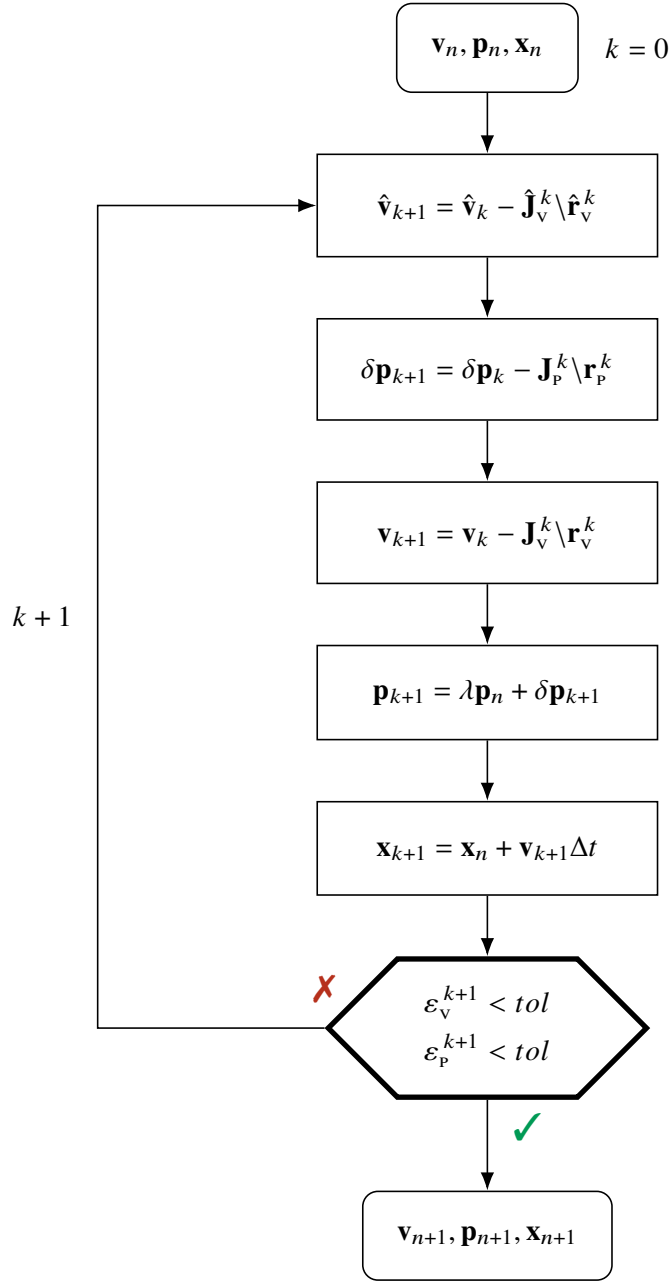
The last stage of the fractional step scheme involves using the first equation in (2.55) to recover the final velocity vector by correcting the intermediate solution for the velocity with the pressure increment, yielding

$$\underbrace{\frac{1}{\Delta t} \int_{\Omega} \rho \mathbf{N}_3 \mathbf{N}_3^T d\mathbf{x}}_{\mathbf{M}} \mathbf{v}_{n+1} = \underbrace{\frac{1}{\Delta t} \int_{\Omega} \rho \mathbf{N}_3 \mathbf{N}_3^T d\mathbf{x}}_{\mathbf{M}} \hat{\mathbf{v}}_{n+1} - \underbrace{\int_{\Omega} \mathbf{N}_3 \mathbf{B}_1^T d\mathbf{x}}_{\mathbf{S}} \delta \mathbf{p}_{n+1}\tag{2.59}$$

Here, the vector $\mathbf{v}_{n+1}$ is considered as the new equilibrium solution. Then, the new equilibrium pressure is obtained with $\mathbf{p}_{n+1} = \lambda \mathbf{p}_n + \delta \mathbf{p}_{n+1}$. In the Newton-Raphson iterative procedure, the Jacobian and the residual of (2.59) are given by

$$\begin{aligned}\mathbf{J}_v^k &= \frac{\partial \mathbf{r}_v^k}{\partial \mathbf{v}_k} \\ \mathbf{r}_v^k &= \mathbf{M}_k \mathbf{v}_k - \frac{1}{\Delta t} \mathbf{M}_k \hat{\mathbf{v}}_k + \mathbf{S}_k \delta \mathbf{p}_k\end{aligned}\tag{2.60}$$

A flowchart of the fractional step scheme is illustrated in Algorithm 2. As a side remark, the Picard method is a complementary approach to the NR algorithm, as the method is more robust and does not require computing a Jacobian, but provides a slower convergence rate. For a generic nonlinear system of the form $\mathbf{A}(\mathbf{x})\mathbf{x} = \mathbf{b}(\mathbf{x})$, the Picard iterations can be written as a simplification of the NR iterations where the Jacobian is approximated by $\mathbf{J}(\mathbf{x}) \simeq \mathbf{A}(\mathbf{x})$.



Algorithm 2: Solution strategy for solving mechanical problems with the fractional step, between time t_n and time t_{n+1} . In order to simplify the analyses that follow, the same tolerance value is used for both velocity and pressure.

In weakly nonlinear problems or when an analytical formulation of the Jacobian is not available, the Picard iterations can provide a more efficient alternative to the NR in terms of computation time because it avoids the derivation of the Jacobian. Nonlinear iterations are stopped once the solution has converged to a stable value:

$$\begin{aligned}\varepsilon_v^{k+1} &= \frac{\|\mathbf{v}_{k+1} - \mathbf{v}_k\|}{\|\mathbf{v}_n\|} < tol \\ \varepsilon_p^{k+1} &= \frac{\|\mathbf{p}_{k+1} - \mathbf{p}_k\|}{\|\mathbf{p}_n\|} < tol\end{aligned}\tag{2.61}$$

One major advantage of the fractional step method over the PSPG formulation is the symmetry of the sub-problems. Specifically, all three steps involve inverting a symmetric matrix, which can be efficiently solved using a CG method. This makes the fractional step significantly faster than PSPG, particularly in 3D simulations. However, this advantage comes with trade-offs. First, the Poisson equation (2.57) for pressure is typically ill-conditioned, posing challenges for most linear solvers. Second, it has been shown that the stability provided by the fractional step scheme is dependent on the time step size, and pressure oscillations may arise if the time step is too small [126].

Surface Tension

The fractional step procedure described so far does not explicitly account for surface tension on the free surface. To address this, Rabier and Medale [130] proposed a pressure lifting technique that corrects the pressure on the free surface once the projection scheme has converged. This correction involves estimating an intermediate free surface pressure p_s by solving the following equation:

$$\begin{aligned}\boldsymbol{\sigma}(p = p_s) \cdot \mathbf{n} &= \nabla_s \gamma - \gamma \boldsymbol{\kappa} \cdot \mathbf{n} - p_{\text{EXT}} \cdot \mathbf{n} \\ \Leftrightarrow \mathbf{n} \cdot \boldsymbol{\sigma}(p = p_s) \cdot \mathbf{n} &= -\gamma \boldsymbol{\kappa} - p_{\text{EXT}}\end{aligned}\tag{2.62}$$

where p_{EXT} is the external, ambient pressure of the surrounding medium. In our implementation of the PFEM, the pressure correction is integrated into the Poisson equation for the pressure increment as a Dirichlet BC. This approach seeks to impose a boundary-fitted condition that is representative of the surface stresses at the free boundary [129]. Equation (2.56) becomes:

$$\begin{cases} \Delta(\delta p_{n+1}) = \frac{\rho}{\Delta t} \nabla \cdot \hat{\mathbf{v}}_{n+1} & \text{in } \Omega \\ \mathbf{n} \cdot \nabla \delta p_{n+1} = 0 & \text{on } \Gamma_v \\ \delta p_{n+1} = p_s & \text{on } \Gamma_s \end{cases} \quad (2.63)$$

Note that while providing a better approximation of the free surface pressure, this pressure correction does not address the problem of mass conservation since $\mathbf{v}_{n+1}$ is still not divergence-free on pressure-constrained elements (i.e. at the free surface). Nevertheless, the nodal values of the free surface pressure correction $\mathbf{p}_s$ are obtained by solving the finite element formulation of (2.62):

$$\int_{\Gamma_s} \mathbf{N}_1 \mathbf{N}_1^\top d\mathbf{x} \mathbf{p}_s = \int_{\Gamma_s} \mathbf{N}_1 \left(\mathbf{p}_{\text{EXT}} + \hat{\mathbf{n}}^\top \mu \mathbf{F} \mathbf{B}_3^\top \mathbf{v}_n \right) d\mathbf{x} - \int_{\Gamma_s} \mathbf{N}_1 \gamma \kappa d\mathbf{x} \quad (2.64)$$

Recall that the normal is undefined at the nodes of linear finite elements, as this would require a local surface to which the normal can be perpendicular, a surface that does not exist at the vertices. However, it is still possible to approximate a normal at the nodes by taking the weighted sum of the normals of the connecting elements, allowing for a local estimation of κ for use in (2.64). The vector $\hat{\mathbf{n}}$ is the Voigt formulation of $\mathbf{n} \otimes \mathbf{n}$ at the interface, which is given by

$$\hat{\mathbf{n}} = \begin{pmatrix} n_1^2 \\ n_2^2 \\ n_3^2 \\ 2n_1n_2 \\ 2n_2n_3 \\ 2n_3n_1 \end{pmatrix} \quad (2.65)$$

2.1.3 Weak Compressibility

While the incompressible approximation is widely used for its simplicity and accuracy across many flow regimes, there are cases where it falls short. Certain physical phenomena, such as acoustic wave propagation or moderately high-speed flows, demand a slight allowance for compressibility, even when overall density variations are small. This gives rise to the concept of weak compressibility, where the fluid is not fully compressible, but the governing equations permit limited density variation.

This approach retains much of the computational efficiency and conceptual clarity of incompressible models, while offering better accuracy in regimes where pressure and density changes cannot be entirely ignored. When compressibility is accounted for, the governing equations become

$$\begin{cases} \nabla \cdot \boldsymbol{\sigma} + \rho \mathbf{g} = \rho \frac{d\mathbf{v}}{dt} & \text{in } \Omega \\ \frac{d\rho}{dt} + \rho \nabla \cdot \mathbf{v} = 0 & \text{in } \Omega \\ \nabla \cdot \mathbf{q} + \rho c_p \frac{dT}{dt} = s & \text{in } \Omega \end{cases} \quad (2.66)$$

The key difference with the system (2.1) is that the mass conservation equation now includes a time derivative of the density, which is an unknown of the system (2.66). Assuming the dependency $\rho = \rho(p, T)$ and using the chain rule of derivations, this additional term can be rewritten in terms of the time derivatives of pressure and temperature, which are the primary unknowns:

$$\frac{d\rho}{dt} = \frac{\partial \rho}{\partial p} \frac{dp}{dt} + \frac{\partial \rho}{\partial T} \frac{dT}{dt} = \frac{\rho}{K} \frac{dp}{dt} - \rho \alpha_v \frac{dT}{dt} \quad (2.67)$$

Here, K denotes the bulk modulus of the fluid, and α_v is the volumetric thermal expansion coefficient. Under the weak compressibility approximation, a linear equation of state relates pressure, temperature, and density. Typically, K is defined at constant temperature, while α_v is defined at constant pressure:

$$\begin{aligned} \frac{1}{K} &= \frac{1}{\rho} \left(\frac{\partial \rho}{\partial p} \right)_T \\ \alpha_v &= \frac{-1}{\rho} \left(\frac{\partial \rho}{\partial T} \right)_p \end{aligned} \quad (2.68)$$

Here, the subscripts T and P on the derivatives indicate that temperature and pressure are held constant, respectively. Substituting Equation (2.68) into the second equation of the system (2.66) and dividing both sides by the density yields the following strong form of the mass conservation equation:

$$\frac{1}{K} \frac{dp}{dt} - \alpha_v \frac{dT}{dt} + \nabla \cdot \mathbf{v} = 0 \quad (2.69)$$

The finite element formulation of equation (2.69) is derived in the same manner as for the incompressible case with PSPG stabilization. Namely, by multiplying both sides by a test function, integrating over the computational domain, and performing spatial discretization. We therefore proceed by directly recalling the final expression:

$$\mathbf{D} \mathbf{v} + \mathbf{L} \mathbf{p} + \mathbf{C} \frac{d\mathbf{v}}{dt} + \underbrace{\int_{\Omega} \frac{1}{K} \mathbf{N}_1 \mathbf{N}_1^{\top} d\mathbf{x}}_{\mathbf{H}} \frac{d\mathbf{p}}{dt} = \mathbf{h} + \underbrace{\int_{\Omega} \alpha_v \mathbf{N}_1 \mathbf{N}_1^{\top} d\mathbf{x}}_{\mathbf{Z}} \frac{d\mathbf{T}}{dt} \quad (2.70)$$

Here, $\mathbf{D}$, $\mathbf{L}$, $\mathbf{C}$ and $\mathbf{h}$ retain the same definitions as in Equation (2.36), which can be retrieved for $K \rightarrow \infty$ and $\alpha_v \rightarrow 0$. The resulting system is nonlinear with respect to the primary unknowns and is solved iteratively using a Newton-Raphson algorithm. In this context, the mechanical residual (2.44) at the k -th iteration is given by

$$\begin{aligned} \mathbf{r}_M^k &= \left(\mathbf{K}_k + \frac{1}{\Delta t} \mathbf{M}_k \right) \mathbf{v}_k - \mathbf{D}_k^{\top} \mathbf{p}_k - \mathbf{f}_k - \frac{1}{\Delta t} \mathbf{M}_k \mathbf{v}_n \\ \mathbf{r}_C^k &= \left(\frac{1}{\Delta t} \mathbf{C}_k + \mathbf{D}_k \right) \mathbf{v}_k + \left(\mathbf{L}_k + \mathbf{H}_k \right) \mathbf{p}_k - \mathbf{h}_k - \frac{1}{\Delta t} \mathbf{C}_k \mathbf{v}_n - \mathbf{Z} \frac{\Delta \mathbf{T}_n}{\Delta t} \end{aligned} \quad (2.71)$$

Here, $\Delta \mathbf{T}_n$ represents the temperature increment for the current time step. Recall that temperature is computed by solving the energy conservation equation separately from the mechanical system, using a staggered coupling approach. As a result, thermal quantities are treated as constant during the NR iterations of the mechanical system. The Jacobian is then computed as:

$$\mathbf{J}_k = \begin{pmatrix} \frac{\partial \mathbf{r}_M^k}{\partial \mathbf{v}_k} & \frac{\partial \mathbf{r}_M^k}{\partial \mathbf{p}_k} \\ \frac{\partial \mathbf{r}_C^k}{\partial \mathbf{v}_k} & \frac{\partial \mathbf{r}_C^k}{\partial \mathbf{p}_k} \end{pmatrix} \quad (2.72)$$

At each NR iteration, the fluid density must be updated to compute both the Jacobian and the residual. Let K_0 denote the initial bulk modulus at $p = p_0$ such that $K(p_0) = K_0$ and let ρ_0 be the corresponding initial density such that $\rho(p_0) = \rho_0$. Assuming the bulk modulus varies linearly with the pressure, where K' is the linear coefficient, the Tait-Murnaghan equation of state [131] is expressed as

$$\begin{aligned}\rho &= \rho_0 \left[1 + \frac{K'}{K_0} (p - p_0) \right]^{1/K'} \\ K &= K_0 + K' (p - p_0)\end{aligned}\tag{2.73}$$

2.2 Building the Mesh

In the finite element method, the domain is discretized in space using a mesh. Practically, the material is divided into an assembly of finite elements, each characterized by a specific number of nodes. Each element contributes to the calculation of local elemental matrices, which are then assembled into a global matrix $\mathbf{A}$ and a global right-hand side vector $\mathbf{b}$, representing the discretized system of equations. Let E denote the total number of elements in the mesh, and let $\mathbf{A}_e$ and $\mathbf{b}_e$ respectively represent the elemental matrices and right-hand side vectors related to the e -th finite element. The resulting system can then be expressed as

$$\mathbf{A} = \bigoplus_{e=1}^E \mathbf{A}_e \quad \mathbf{b} = \bigoplus_{e=1}^E \mathbf{b}_e \tag{2.74}$$

Here, the summation symbol $\oplus$ is used in place of Σ to indicate that the components of $\mathbf{A}_e$ are summed into $\mathbf{A}$ based on the global indices of the corresponding nodes in the mesh. Since the mesh is generated through a Delaunay tessellation, the PFEM employs isoparametric triangular and tetrahedral elements, as presented in Figure 4, whose basis functions are linear Lagrange polynomials outlined in Table 1.

	$N_1(\xi)$	$N_2(\xi)$	$N_3(\xi)$	$N_4(\xi)$
1D	$(1 - \xi)/2$	$(1 + \xi)/2$	–	–
2D	$1 - \xi - \eta$	ξ	η	–
3D	$1 - \xi - \eta - \zeta$	ξ	η	ζ

Table 1: Linear isoparametric basis functions used for the reference elements in Figure 4. The vector of local coordinates $\xi = (\xi, \eta, \zeta)$ is defined in the reference space of the element, and N_j is the basis function related to the j -th node.

This means that the basis functions are defined in the local space of a reference element, where each basis function evaluates to 1 on the generating node and linearly decreases to zero towards the other nodes. To numerically integrate a field quantity over isoparametric elements, a Gaussian quadrature rule is defined in the local space of the reference element. This means that a mapping $\mathbf{x}(\boldsymbol{\xi})$ between the global coordinates $\mathbf{x}$ of the fluid domain and the local coordinates $\boldsymbol{\xi}$ in the reference elements must be established to allow for an integration over the actual element's shape in the mesh. This transformation is generally represented by a Jacobian matrix, which is given by

$$\mathbf{J} = \begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{pmatrix} = \sum_{j=1}^M \begin{pmatrix} \frac{\partial N_j}{\partial \xi} x_j & \frac{\partial N_j}{\partial \xi} y_j & \frac{\partial N_j}{\partial \xi} z_j \\ \frac{\partial N_j}{\partial \eta} x_j & \frac{\partial N_j}{\partial \eta} y_j & \frac{\partial N_j}{\partial \eta} z_j \\ \frac{\partial N_j}{\partial \zeta} x_j & \frac{\partial N_j}{\partial \zeta} y_j & \frac{\partial N_j}{\partial \zeta} z_j \end{pmatrix} \quad (2.75)$$

In this context, the subscript j represents the index of the node within the element, and M denotes the total number of nodes in the element. Using the Jacobian, the partial derivatives of the basis functions with respect to the global coordinates, as defined in equations (2.11) and (2.34), are obtained by inverting the matrix $\mathbf{J}$ and applying the chain rule of differentiation:

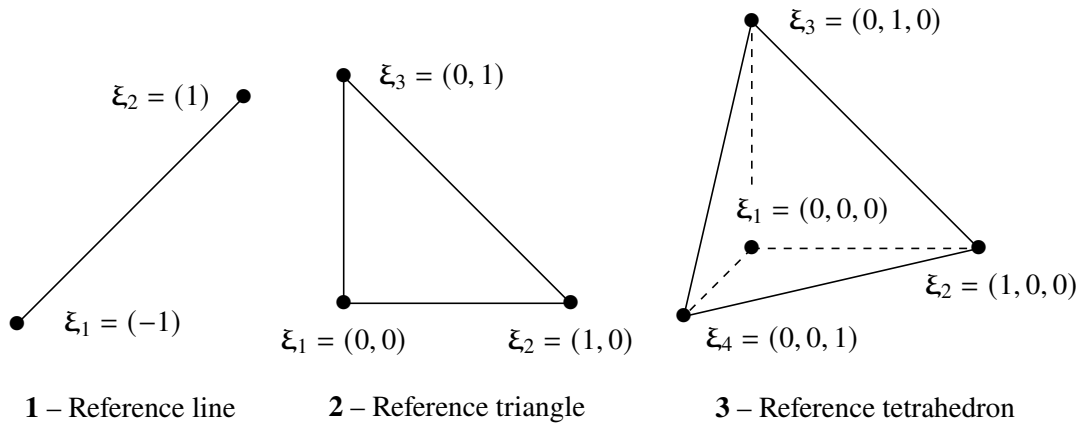


Figure 4: Reference geometry and vertices coordinates $\boldsymbol{\xi} = (\xi, \eta, \zeta)$ for isoparametric line, triangle and tetrahedron finite elements using the linear Lagrange polynomials from Table 1 as their basis functions.

$$\frac{\partial \mathbf{N}}{\partial x_k} = \frac{\partial \mathbf{N}}{\partial \xi} \mathbf{J}_{1k}^{-1} + \frac{\partial \mathbf{N}}{\partial \eta} \mathbf{J}_{2k}^{-1} + \frac{\partial \mathbf{N}}{\partial \zeta} \mathbf{J}_{3k}^{-1} \quad (2.76)$$

The numerical integration of the elemental matrices is performed using a quadrature rule applied to the reference space, with the results then mapped to the global space via the Jacobian. The quadrature rule approximates the integral of a function as a weighted sum of function values evaluated at specific points within the integration domain. With an appropriate choice of integration points and weights in 1D, the Gaussian quadrature constructed with K points provides an exact result for polynomials of degree $2K - 1$ or less. By default, PFEM3D uses a number of integration points equal to the number of nodes in the elements, although the user can opt for either lower or higher-order integration rules. Let $f : \Omega \rightarrow \mathbb{R}^3$ be an integrable function defined over Ω , where Ω represents the integration domain of an element in the global coordinate system:

$$\begin{aligned} \int_{\Omega} f(\mathbf{x}) d\mathbf{x} &= \int_{\Lambda} f\{\mathbf{x}(\boldsymbol{\xi})\} \det \mathbf{J}(\boldsymbol{\xi}) d\boldsymbol{\xi} \\ &= \sum_{j=1}^K w_j f\{\mathbf{x}(\boldsymbol{\xi}_j)\} \det \mathbf{J}(\boldsymbol{\xi}_j) \end{aligned} \quad (2.77)$$

Here, Λ represents the local space of the reference element, K denotes the total number of integration points in the Gaussian quadrature rule, $\boldsymbol{\xi}_j$ refers to the coordinates of the j -th integration point in the reference element, and w_j is the associated quadrature weight.

2.2.1 Remeshing with Edge Splitting

A defining feature of the PFEM is its frequent remeshing. In a Lagrangian framework, the motion of nodes leads to significant distortion of mesh elements after a few time steps, especially when simulating fluids. Such distortions can degrade the accuracy of the solution and eventually cause the solver to fail to converge. To address this, the mesh is periodically destroyed and regenerated based on the current configuration of the cloud of nodes, as illustrated in Figure 5. This approach treats the nodes and the information they carry as the only entities that persist through remeshing. The other mesh entities, such as elements and edges, are temporary constructs created solely to compute the solution at the current time step. Note that this method may pose additional challenges when deformation history is tied to the elements, as it is the case in solid mechanics with plasticity.

In such scenario, quantities defined at the Gauss points of the elements must be mapped to the updated mesh. Nevertheless, fluid nodes can be dynamically added or removed during remeshing based on specific criteria. The conditions governing these node operations are discussed in the remainder of this subsection.

The Alpha-Shape Criterion

Historically, the remeshing process is carried out in three fundamental steps [23]. First, the existing finite elements are destroyed. Second, a Delaunay tessellation is performed on the remaining cloud of nodes. Finally, the α -shape criterion is applied to the tessellation to extract the new FE mesh. Practically, the α -shape criterion $\mathcal{A}$ can be seen as a boolean function which, for each simplex e in the tessellation, checks whether the following condition is satisfied:

$$\mathcal{A}(e, \mathbf{x}) = \text{True} \Leftrightarrow r_e < \alpha h(\mathbf{x}) \quad (2.78)$$

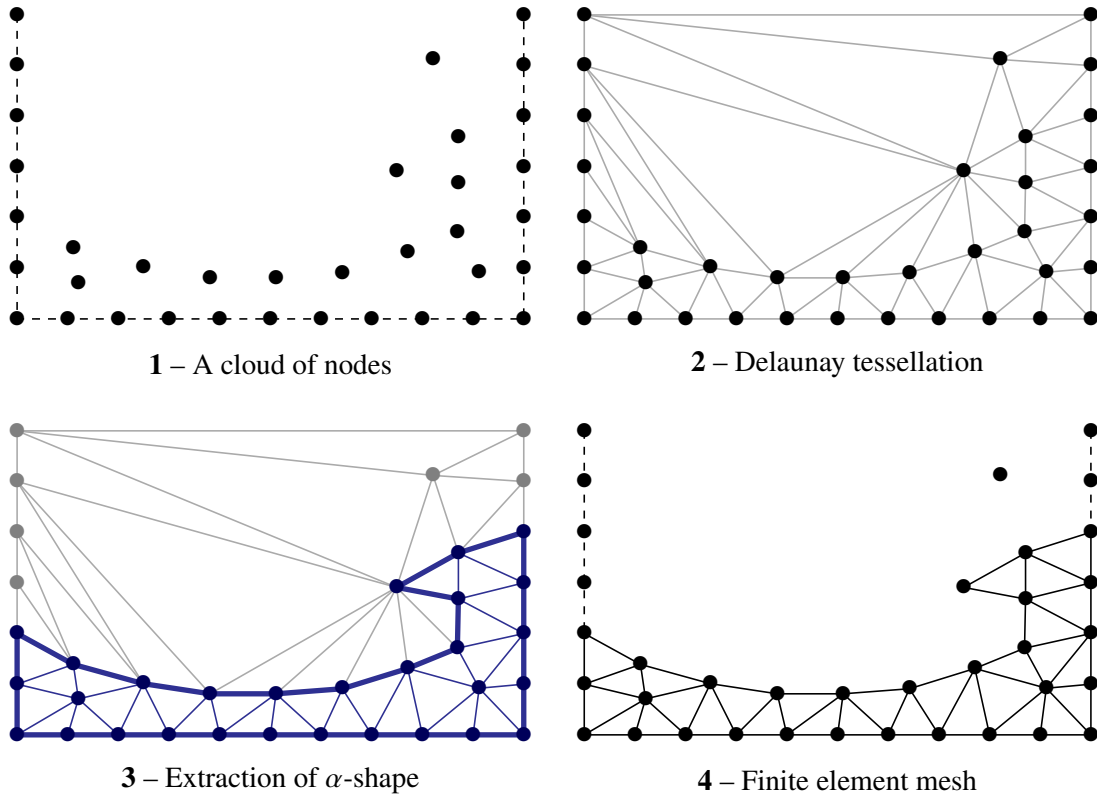


Figure 5: Standard steps of the remeshing procedure in the PFEM. The cloud of nodes **1** derives from the last equilibrium configuration obtained so far. The dashed lines represent the parts of the rigid container that are not meshed.

Here, r_e represents the radius of the circumcircle of a triangle e in 2D, or the circumsphere of a tetrahedron e in 3D, and α is a user-defined parameter [65]. The function $h(\mathbf{x})$ is called the **size-field**, and provides a characteristic size based on the coordinates $\mathbf{x}$, generally taken as the centroid of the element considered. It is important to note that additional user-defined parameters can also be incorporated when calculating the characteristic size. For example, one might prescribe a smaller mesh size in regions with large velocity gradients or high temperatures, as demonstrated in [78, 132]. For simplicity, we will describe the size-field as a function of $\mathbf{x}$ alone, but incorporating physical parameters into the characteristic size calculation does not invalidate the proposed methodology. In fact, our approach is independent of the specific expression of h . By evaluating the criterion $\mathcal{A}$ for each simplex in the tessellation, the method eliminates large or stretched elements, as shown in Figure 6, effectively recovering the shape of the fluid domain.

The α -shape method has been successfully applied in 2D simulations with a constant h in earlier studies, and later extended to variable size-fields, as explored in the thesis of Falla [78, 133]. However, evidence suggests that relying on the $\mathcal{A}$ alone is insufficient for handling complex cases, particularly in 3D [78, 79]. To overcome these limitations, we propose a six-step approach, building upon the advances achieved in prior works, designed to deliver good performance in 2D while extending the capability to handle 3D scenarios effectively.

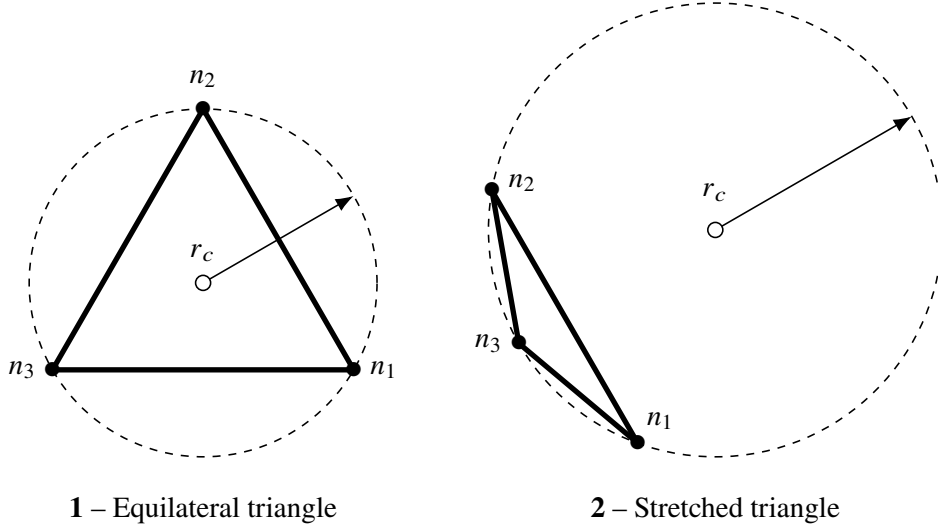


Figure 6: In Example 1, a triangle with good aspect ratio, whose circumradius is close to its characteristic size. In Example 2, a triangle with large aspect ratio, whose circumradius is larger than its characteristic size.

In particular, it incorporates strategies to regularize critical issues, such as tetrahedral degeneracy, which frequently arise in 3D Delaunay tessellations. Moreover, it remains relatively simple and leverages a consistent implementation framework for both 2D and 3D, making it practical and adaptable across dimensions.

Step 1 – Local Refinement

When the remeshing occurs, the mesh is destroyed and a new one is created on the current state of the cloud of nodes by a Delaunay tessellation. We seek the local resolution of the mesh to be in accordance with a size-field deduced dynamically depending on the geometry of the domain or the physical characteristics of the fluid. For instance, it is generally enforced that the size of the elements is smaller close to the free surface in order to accurately capture its geometry and limit mass error (the mass conservation is a major challenge in the PFEM which will be discussed later). To achieve this, we first evaluate for each node n the characteristic size h_n . The collection of nodal characteristic sizes defines a discrete approximation of $h(\mathbf{x})$ and is updated during the first step of the remeshing process. Let $\mathbf{x}_n$ be the coordinates of the n -th node in the mesh, one can write

$$h_n = h(\mathbf{x}_n) \quad (2.79)$$

A copy of the mesh before its deletion is stored in memory until the end of remeshing the procedure. Indeed, updating h_n as well as performing important computation during the remeshing may require the pre-existence of a finite element mesh. In the following, when we talk about elements, we are thus referring to the elements of the former mesh, which is temporarily stored in memory to allow these operations. Once the target mesh size has been evaluated at the nodes, we must ensure that the new tessellation will generate elements with the target size. In practice, this is achieved by creating and removing nodes wherever the actual node density at $\mathbf{x}$ does not match $h(\mathbf{x})$ up to some tolerance. The expected characteristic size $h_e(\mathbf{x})$ of an element e at location $\mathbf{x}$ is approximated by the average of the characteristic sizes of the nodes forming the element:

$$h_e(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^N h_n \quad \forall n \in e(\mathbf{x}) \quad (2.80)$$

Here, $e(\mathbf{x})$ denotes the finite element located at $\mathbf{x}$, and N is the number of nodes of this element. It is important to note that the definitions provided here are intended to describe the refinement in a simple and intuitive manner. In practice, the algorithm does not need to evaluate h_e at ambiguous locations, such as on vertices shared by multiple elements. Additionally, we define $S_e(\mathbf{x})$ as the actual size of the finite element at $\mathbf{x}$ in its corresponding dimensional space. For instance, $S_e(\mathbf{x})$ refers to the volume of a tetrahedron or to the area of a triangle, and $r_e(\mathbf{x})$ is its circumradius. New nodes are introduced into an element when its size is too large relative to the expected characteristic size. Mathematically, this happens when either of the following conditions is met:

$$\begin{aligned} r_e(\mathbf{x}) &> \alpha h_e(\mathbf{x}) \\ S_e(\mathbf{x}) &> \omega h_e(\mathbf{x})^d \end{aligned} \tag{2.81}$$

Here, α is the same α -shape parameter described in Equation (2.78), ω is a scalar parameter defined by the user (the optimal value of ω is computed in the next subsection), and d is the dimension of the problem. The first condition targets distorted elements, while the second condition focuses on large elements that may still have a good aspect ratio. For simplicity, the condition (2.81) can be seen as a boolean function which, for each element e in the former mesh, returns true if one or both inequalities are valid, and false otherwise:

$$\mathcal{E}\{e(\mathbf{x})\} = \text{True} \Leftrightarrow \vee \left\{ \begin{array}{l} r_e(\mathbf{x}) > \alpha h_e(\mathbf{x}) \\ S_e(\mathbf{x}) > \omega h_e(\mathbf{x})^d \end{array} \right. \tag{2.82}$$

Here, $\vee$ is the logical disjunction, also known as the logical OR. Several methods exist for introducing additional nodes within an element. A commonly used approach, as highlighted in previous studies [29, 80], involves placing a single node at the centroid of large elements. While this method is simple and efficient, it prevents boundary refinement, as nodes are only generated in the interior of the mesh. Additionally, this approach often results in new elements with higher aspect ratios, potentially degrading mesh quality. More recently, Falla et al. [78] introduced a 2D mesh refinement technique based primarily on edge splitting, which helps maintain overall mesh quality after refinement. The method proposed here draws inspiration from Falla's work, as it also utilizes an edge splitting technique, but aims to simplify and generalize it for 3D cases.

An element is eligible for **full refinement** when it satisfies $\mathcal{E}$ and when the center of its circumsphere lies inside the element. In this case, new nodes are added at the center of its edges, as illustrated in Figures 7 and 8. Special treatment is given to elements that satisfy $\mathcal{E}$ and have the center of their circumsphere located outside the element. This situation indicates that the original element is stretched, and in such cases, a single node is added at the center of its longest edge.

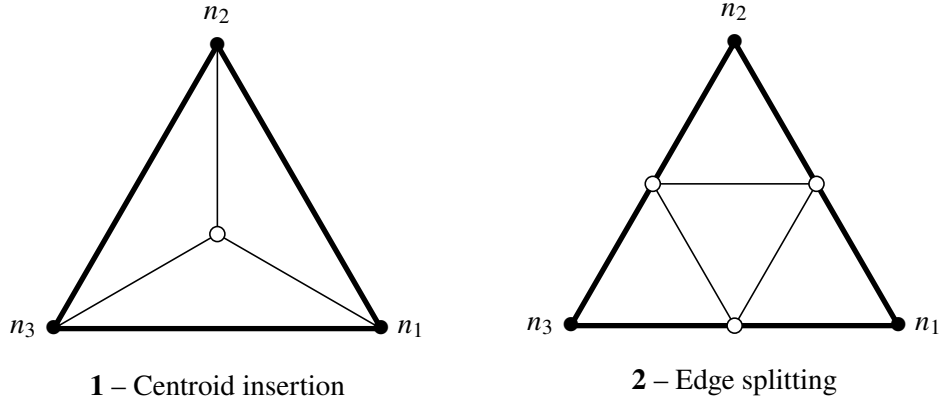


Figure 7: Example of **full refinement** of an element in 2D. In **1**, the triangle features mesh refinement based on the centroid insertion. In **2**, the triangle features edge splitting, generating 4 smaller elements with good aspect ratio.

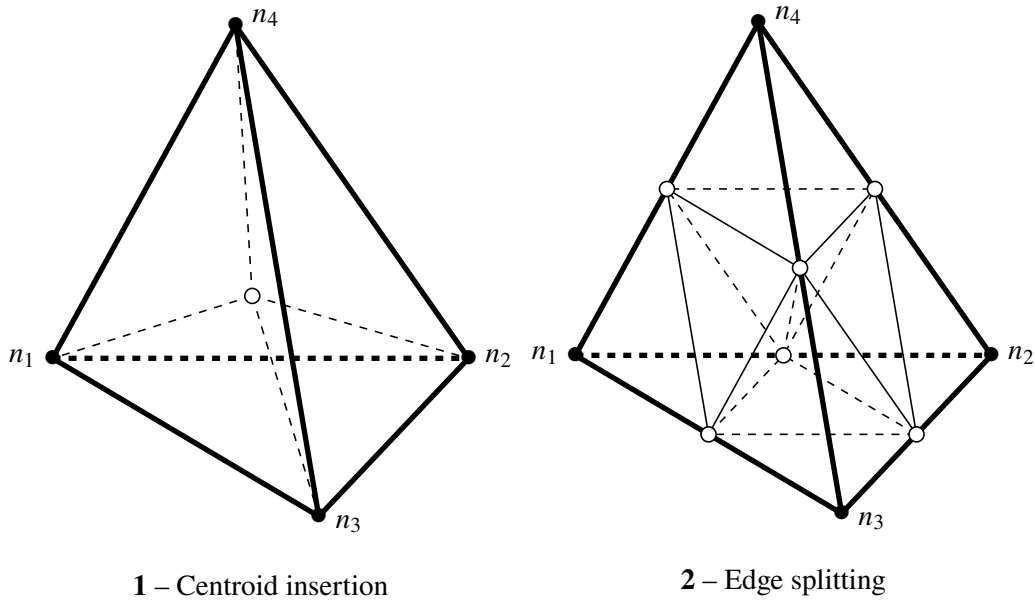


Figure 8: Example of **full refinement** of an element in 3D. In **1**, the tetrahedron features mesh refinement based on the centroid insertion. In **2**, the tetrahedron features edge splitting, generating 8 smaller elements with good aspect ratio.

It is important to note that large or distorted elements may also contain very small edges alongside larger ones. The expected length of an edge relative to $h(\mathbf{x})$ can be estimated using the condition (2.80), treating each edge as a 1D element with two nodes. Refinement of an edge is prevented if it satisfies

$$S_e(\mathbf{x}) < \phi h_e(\mathbf{x}) \quad (2.83)$$

Recall that $S_e(\mathbf{x})$ refers to the measure of the element in its corresponding dimensional space, which in this case is the length of the edge. Based on this condition, the refinement of 2D or 3D elements can be restricted by preventing the addition of nodes on small edges that satisfy the condition (2.83). This limitation is designed to avoid splitting distorted elements in a way that worsens their aspect ratio, thereby allowing for the potential improvement of the new elements aspect ratio. In this context, we refer to the process as **partial refinement** of an element.

Examples of partial refinements are presented in Figures 9 and 10. It is worth mentioning that an edge can be refined only once per remeshing. Once a node has been created, nodal quantities such as the velocity, the pressure or the characteristic size are computed and associated to the node by interpolation using the basis functions of the surrounding elements. Since the generated node is located at the center of an edge, the interpolation of nodal quantities on the new node n is trivially obtained as follows:

$$\mathbf{p}_n = \frac{\mathbf{p}_1 + \mathbf{p}_2}{2} \quad (2.84)$$

Here, $\mathbf{p}$ holds for arbitrary nodal quantities and the indices 1, 2 respectively denote the first and second node of the edge. In our PFEM code, each boundary is associated with a number of tags, except the free-surface. These **tags** are used to identify the boundary condition that must be applied. For instance, a boundary representing an inlet may be associated with the tag *inlet* for which Dirichlet BC on the velocity have been defined by the user. As an additional example, a boundary representing a solid wall may be associated with the tag *wall* having no-slip BC and a specified wall velocity if the latter moves or deforms over time. Since the boundaries are represented by a set of nodes, the tags are stored at the node level, and new nodes generated on a boundary must inherit from the related tags.

Step 2 – Local Coarsening

In some cases, the node density at a specific location may exceed the value expected by the size-field. Similarly to mesh refinement, the coarsening strategy involves removing excess nodes to align with $h(\mathbf{x})$ locally. The algorithm begins by looping over all nodes and removing those located very close to a boundary.

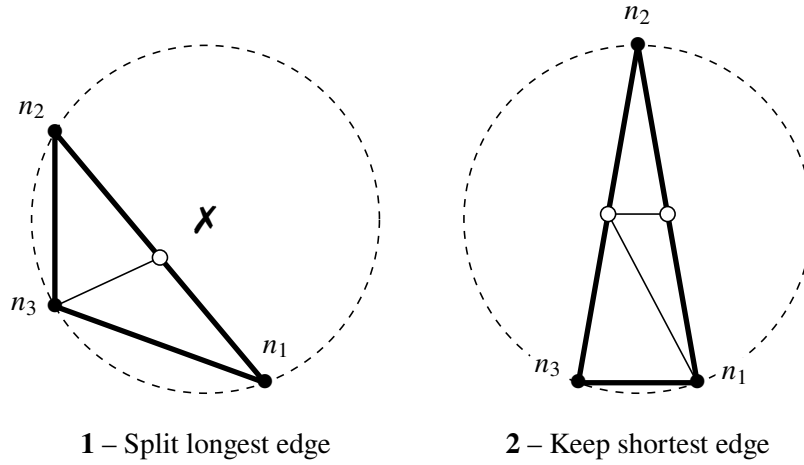


Figure 9: Example of **partial refinement** in 2D. In **1**, the simplex undergoes a splitting of its longest edge because its circumcenter lies outside the element. In **2**, the smallest edge is not refined because it satisfies the condition (2.83).

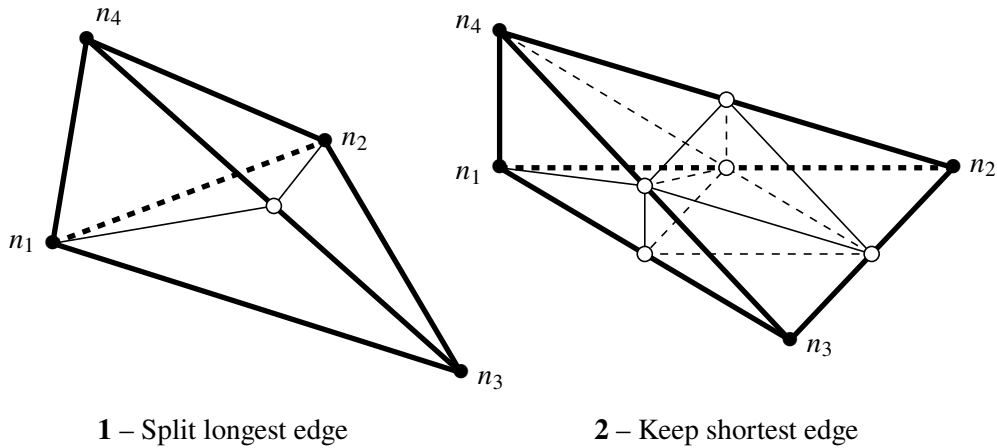


Figure 10: Example of **partial refinement** in 3D. In **1**, the simplex undergoes a splitting of its longest edge because its circumcenter lies outside the element. In **2**, the smallest edge is not refined because it satisfies the condition (2.83).

This is achieved by computing the Euclidean distance d_n between node n and the line (or the plane in 3D) passing through its two (or three in 3D) closest neighbors that share at least one common tag. Two nodes are considered neighbors if they share a common edge. A visual example in 2D is shown in Figure 11. Mathematically, a non-boundary node n is removed from the mesh if

$$d_n < \beta h_n \quad (2.85)$$

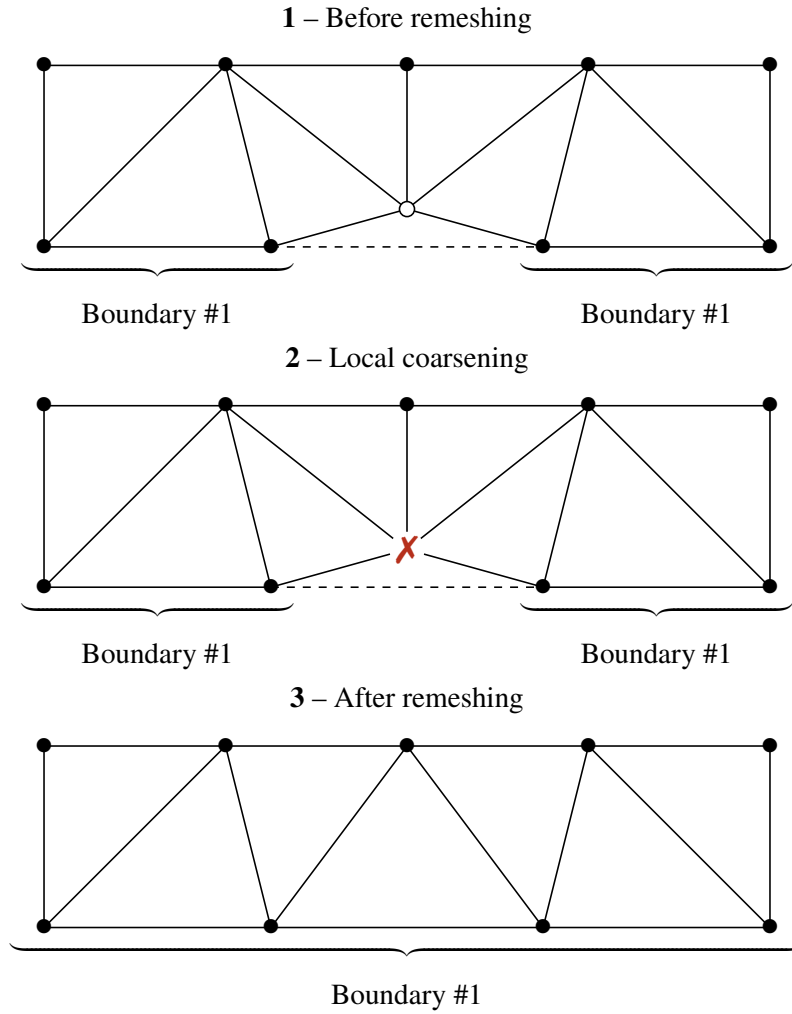


Figure 11: Example of **local coarsening** based on the distance between a node and the boundary. The nodes at the bottom share a common tag and are part of the same boundary, which is here the tag #1.

where β is a user-defined parameter discussed in the next subsection, and d_n is the distance between the white node and the dashed line representing the hypothetical wall. This step is designed to prevent pathological situations where a node comes very close to a boundary, leading to stretched simplices being created between this node and the boundary. Since stretched simplices typically do not satisfy the α -shape criterion, they are not accepted as part of the mesh, leaving an empty space through which the fluid may accidentally cross the boundary. This problem typically arises in 3D cases with curved walls. A second loop is then applied to all edges in the mesh, comparing their length to an approximation of the size-field at that location. Mathematically, an edge is eligible for coarsening if

$$S_e(\mathbf{x}) < \delta h_e(\mathbf{x}) \quad (2.86)$$

Here, δ is a user-defined parameter, whose optimal value is computed in the next subsection. The coarsening of an edge follows a rule based on the priority level associated with its nodes. A node's **priority level** is an integer, between 1 and 4, indicating the degree to which the node is protected against removal. If two nodes, which are part of the same edge satisfying (2.86), have different priority levels, the node with the lower priority is removed, and the other is retained. In contrast, when the two nodes have the same priority level, both are removed from the mesh and replaced by a single new node created at the center of the edge. The linear interpolation (2.84) is then applied to transfer the nodal data onto the newly created node. Priority levels are assigned as follows:

- The highest priority level **{4}** is assigned to tagged boundary nodes (such as walls, inlets or outlets) from the original mesh. These nodes together define the geometry of the fluid's container and must never be removed. A specific case occurs when an edge is composed solely of such boundary nodes and satisfies Equation (2.86). In this situation, the edge is never coarsened to ensure the preservation of the wall's geometry.
- The second-highest priority **{3}** is assigned to tagged boundary nodes that were added through local refinement during the simulation. These nodes are typically placed between two nodes of higher priority, and therefore can be removed without affecting the original discretization of the wall.
- The priority level **{2}** is assigned to nodes located on the free surface. Recall that free surface nodes refer to all nodes which are part of the fluid's current boundary, but which do not have a specified boundary tag. Such nodes, along with other boundary nodes with higher priority, define the geometry of the fluid's domain, which should remain unaltered when possible.

- Finally, the lowest priority level $\{1\}$ is assigned to all other nodes. Typically, those low-priority nodes are located in the interior of the fluid, and their removal does not affect the actual shape of the fluid domain.

A summary of the priority-based removal process is presented in Figure 12. An important remark is that the coarsening of multiple edges in the same region may result in the excessive removal of nodes compared to the expected size-field. To prevent this, an additional flag is applied to nodes marked for removal, ensuring that adjacent edges are not coarsened simultaneously. Furthermore, the chosen value of δ must align with the refinement criteria of the elements. For example, situations where a node is added to the edges of an element based on the condition $\mathcal{E}$ and then removed based on the condition (2.86) must be avoided. This is ensured by setting the value of ϕ such that:

$$\phi = 2\delta \quad (2.87)$$

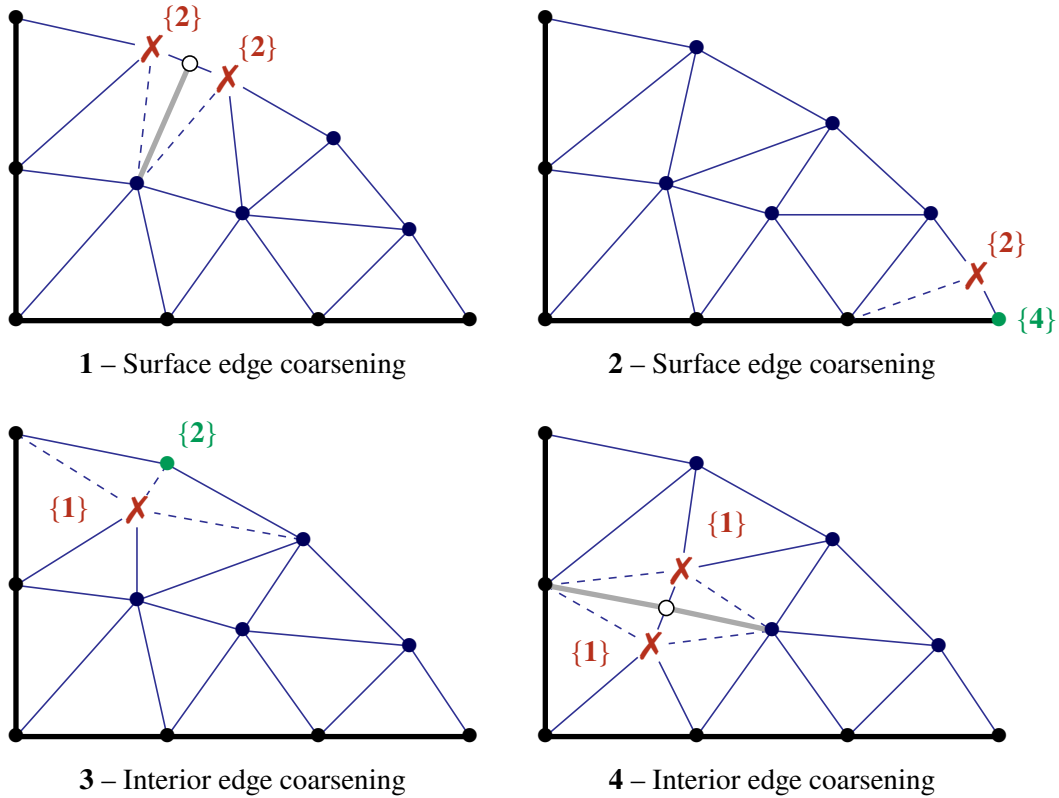


Figure 12: Miscellaneous examples of **edge coarsening** based on the priority level of their nodes. The bold line represents the rigid boundary. The cross mark denotes removed nodes and the white circles denote generated ones.

In other words, the relation (2.87) ensures that an edge whose length is smaller than 2δ is never refined, preventing any refined edges from becoming eligible for coarsening immediately after refinement. In all numerical examples of this thesis, the value of ϕ is not explicitly provided, as it can always be derived from δ using the relation (2.87).

Step 3 – Delaunay Tessellation

A tessellation refers to a division of space into a set of simplices without overlaps or gaps. The Voronoi diagram, also called Voronoi partition, forms the basis for understanding Delaunay tessellation [64]. Given a set of points $\mathcal{P}$ in a space, the Voronoi diagram partitions the space into regions called Voronoi cells. Each Voronoi cell $V(p_j)$ corresponds to a point $p_j \in \mathcal{P}$ and consists of all the locations in the space (also called the locus) that are closer to p_j than to any other point in $\mathcal{P}$. Mathematically, one has

$$V(p_j) = \left\{ \mathbf{x} \in \mathbb{R}^d \mid \|\mathbf{x} - \mathbf{p}_j\| \leq \|\mathbf{x} - \mathbf{p}_k\| \forall j \neq k \right\} \quad (2.88)$$

where d is the dimension of the space and $\mathbf{p}_j$ denotes the position of the j -th point. In 2D, the boundaries of the Voronoi cells are line segments, and the points where three or more Voronoi cells meet are called Voronoi vertices. In 3D, the boundaries are planar facets, and Voronoi vertices occur where four or more Voronoi cells intersect. The Delaunay tessellation is the simplicial decomposition obtained by connecting points whose Voronoi cells share a boundary, forming triangles in 2D and tetrahedra in 3D. More formally, the Delaunay tessellation of a set of points is the unique (in most cases) simplicial complex such that no point in $\mathcal{P}$ lies inside the circumsphere of any simplex in the tessellation. This property is called the empty sphere property and is illustrated in Figure 13.

The Delaunay tessellation can handle irregular domains and point distributions more effectively than grid-based or advancing front methods, which may struggle with highly non-uniform point sets or complex boundaries. Another alternative to the Delaunay tessellation is the octree partitioning [134]. However, the latter is rarely used in FEM because it often introduces hanging nodes that require special treatments. Additionally, octree meshing tends to generate irregular polyhedral elements at boundaries, posing further challenges for FE analysis [135]. In our implementation of the PFEM code, the tessellation of the cloud of nodes is performed using the Gmsh [93] library. Alternatively, TetGen [92] can be used for performing the tetrahedralization in 3D.

Step 4 – Extraction of the Boundary

The next step is the extraction of the new fluid boundary from the tessellation, which roughly corresponds to step 3 in Figure 5. Historically, this is achieved by applying the α -shape criterion $\mathcal{A}$ to the tessellation. In our implementation of the remeshing procedure, this step involves discarding spurious simplices from the tessellation based on a series of criteria, which α -shape is part of, but is clearly not the only criterion used. The different criteria are outlined below, with examples provided in Figure 14.

- The first criterion {1} targets the removal of simplices that are inevitably generated in dry (i.e. not in contact with the fluid) corners due to the natural proximity of the nodes forming the boundary at these locations. Although these simplices may have a good aspect ratio and characteristic size, they do not belong to the actual fluid domain. Specifically, we discard simplices whose nodes are all located on the boundary, and whose neighbors are all either located outside the last-known fluid domain, or composed exclusively of tagged boundary nodes.
- The second criterion {2} is based on the concept of exclusion zones and bounding boxes. An **exclusion zone** is defined by a closed surface formed by a set of segments in 2D or polygons in 3D. Any node or element that penetrates an exclusion zone is deleted during the remeshing process. Exclusion zones typically represent walls or solid structures through which the fluid cannot pass.

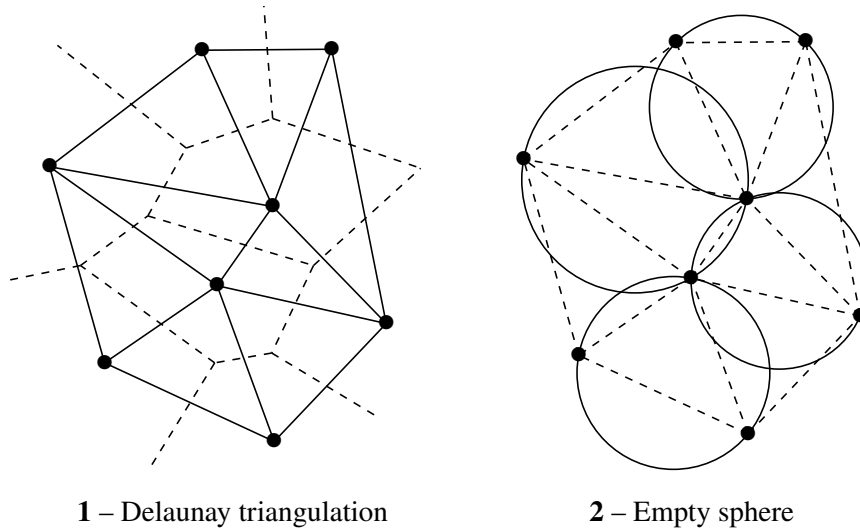


Figure 13: Example of Delaunay triangulation of a cloud of nodes in 2D. A Voronoi diagram is superimposed to the Delaunay triangulation in the first figure. The second figure illustrates the empty sphere property.

In some cases, the boundary of an exclusion zone may evolve over time, such as in fluid-structure interaction simulations. To check whether a point lies inside or outside the zone, a ray-casting algorithm is used, which is described in more detail in the next section. Conversely, a **bounding box** is the inverse of an exclusion zone. Nodes and elements outside the bounding box are deleted. Bounding boxes help confine the computational domain to regions of interest, thus improving simulation efficiency.

- The criterion $\{3\}$ aims to prevent the formation of holes in the fluid volume when particle movement causes node density to decrease faster than the refinement algorithm is able to match the size-field. The key idea is to retain elements composed of nodes that were previously located within the fluid's interior. This ensures that an element inside the fluid is always protected from deletion, even if it does not meet the other criteria, allowing for its refinement or regularization if necessary.
- The last criterion $\{4\}$ applies the α -shape (2.78) to remove distorted or large elements that have not already been discarded or protected by the previous criteria. It is important to note that this step is more complex than a simple application of the α -shape. Specifically, this step introduces an enhanced α -shape criterion, augmented by a Level Set (LS) function, and referred as $\mathcal{L}$ in subsequent sections. While $\mathcal{L}$ is a key contribution of this work, it will be omitted for simplicity until its introduction in the next section.

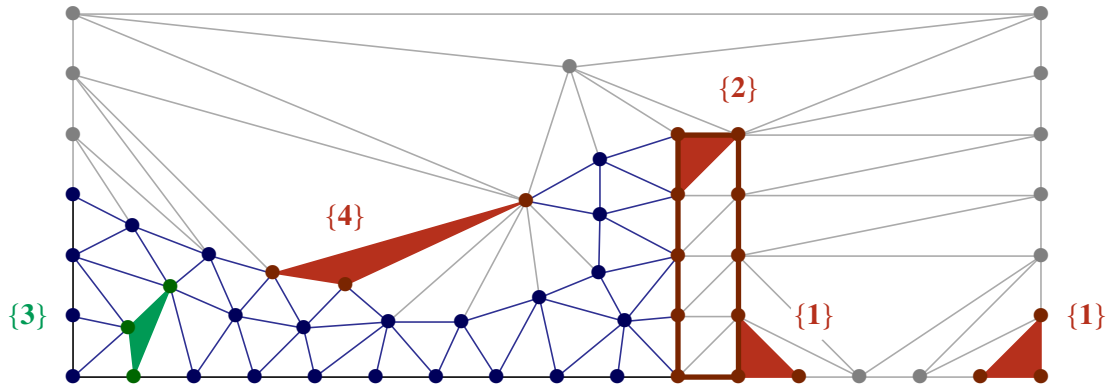


Figure 14: Example of spurious-element deletion in 2D. The bold line represents the closed boundary of a solid in which no fluid may penetrate (i.e. an exclusion zone). Red triangles are deleted while the green triangle is protected from deletion.

Step 5 – Slivers Removal

In Delaunay tetrahedralizations, a sliver is a type of degenerate tetrahedral element with poor geometric quality, as illustrated in Figure 15. A **sliver** typically has a very small or nearly zero volume (while none of its edges are particularly short), and correspondingly, its Jacobian determinant is close to zero. This can lead to numerical instabilities and inaccuracies in FE analysis. In 3D, the Delaunay condition ensures that no vertex of a tetrahedron lies inside the circumsphere of any other tetrahedron in the mesh. While this condition maximizes the minimum angle between faces and avoids skinny triangles in 2D, it does not guarantee well-shaped tetrahedra in 3D. Specifically, a sliver can satisfy the Delaunay criterion because no other vertices lie within its circumsphere, but its four vertices are nearly coplanar. In practice, the presence of a sliver in the finite element mesh can prevent the linear solver from converging in implicit simulations, and can reduce the time step in explicit simulations to nearly zero [80]. Several approaches have been proposed to address slivers in the literature. For example, a sliver can be eliminated by adding a node inside its circumsphere [84]. However, this method does not guarantee that new slivers will not be created elsewhere in the mesh.

A second approach is the pressure-based mesh smoothing technique proposed by Meduri et al. [67], where the sliver is inflated by solving a fictive FE elastic problem. This problem is governed by standard elasticity equations with modified volumetric compatibility. However, pressure-based smoothing is ineffective when a sliver appears near the free surface or any other boundary, as the nodes may be constrained to remain fixed in place.

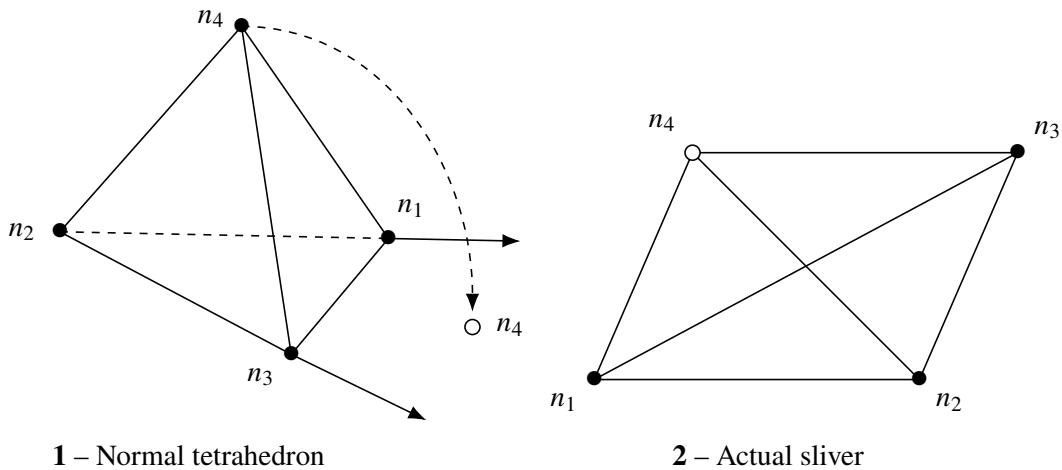


Figure 15: Comparison between a normal tetrahedron in Example 1, and a **sliver** when the node n_4 comes close to the plane formed by (n_1, n_2, n_3) in Example 2. The sliver displays 4 nearly coplanar vertices and a volume close to zero.

Alternatively, Vartziotis et al. [85] proposed the sequential Geometric Element Transformation Method (GETM), a smoothing technique based on simple geometric transformations. When applied iteratively, GETM helps to regularize degenerated elements. In this work, mesh optimization and sliver regularization are carried out using the TetGen library [92]. TetGen offers powerful tools for automatically eliminating slivers from an existing Delaunay tetrahedralization, as described below. Specifically, TetGen implements elementary edge and face flips, which can be used to locally alter the connectivity between adjacent elements and regularize slivers [83, 136].

The 2-to-3 flip, also known as a face-to-edge flip, removes a face and creates an edge. The 3-to-2 flip reverses the 2-to-3 flip and is commonly referred to as the edge-to-face flip. The indices indicate the number of tetrahedra before and after each flip operation. More complex local transformations can be achieved by combining these elementary flips. For example, the 4-to-4 flip interchanges two edges through a combination of a 2-to-3 and a 3-to-2 flip, as shown in Figure 16. Depending on the optimization level specified by the user, TetGen may propagate the flips to adjacent faces to explore improved mesh configurations. It is important to note that some faces or edges may not be flippable depending on their configuration, and the elementary flips are not always possible [92].

Similarly to the GETM approach, TetGen can also relocate vertices to maximize a given quality metric without altering the mesh connectivity, through a process known as vertex smoothing. This smoothing typically involves moving vertices away from flat configurations to reshape slivers into more well-formed tetrahedra. When vertices are moved, the nodal quantities associated with the affected nodes are re-interpolated to their new locations within the PFEM code. As with the GETM, vertex smoothing will not operate on nodes whose positions are restricted during remeshing, such as boundary nodes.

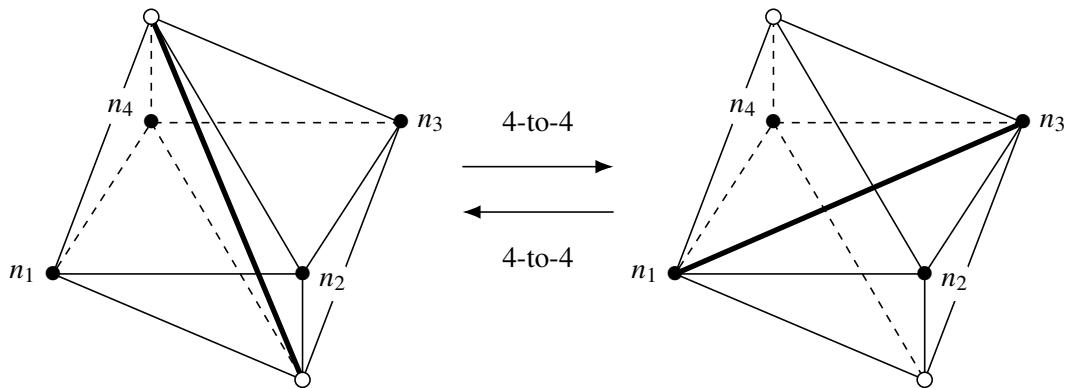


Figure 16: Example of a **4-to-4 flip** which interchanges two edges (drawn in bold) by a combination of a 2-to-3 and a 3-to-2 flips. The bold line denotes the edge that has been interchanged between the left and right picture.

Step 6 – Mesh Finalization

The final step in the remeshing process involves finalizing and adopting the optimized tessellation from the previous stage. At this point, the old mesh, which has been retained in memory, is permanently removed, and new finite elements are created. In the newly adopted mesh, some nodes that do not belong to any element, referred to as flying nodes, may appear. A flying node arises when it is ejected from the free surface and separated from the rest of the fluid by the α -shape, or when it belongs to a dry (i.e. not in contact with the fluid) wall. These flying nodes are treated as material points, with their motion governed by Newton's second law or by the Dirichlet BC imposed on the boundary they belong to. As a reminder, a node is considered part of a boundary if it has a tag associated with that boundary, regardless of whether it is connected to other nodes.

Free surfaces and wet (i.e. meshed) walls are automatically identified on any side (line elements in 2D or triangle elements in 3D) of a bulk element that lacks an adjacent neighboring bulk element. It is worth noting that any sliver remaining after the optimization process in Step 4 is simply deleted from the new mesh. This may create zero-volume holes within the fluid domain, which could lead to unwanted pressure singularities and negatively affect the velocity field. Additionally, if $h(\mathbf{x})$ is based on the distance to the free surface, the hole introduces a new free surface inside the domain, altering the size-field and potentially causing inconsistencies in the mesh. Therefore, it is crucial that all slivers are fully regularized to minimize these adverse effects.

2.2.2 Numerical Examples

In this subsection, we seek to evaluate the quality of the mesh generated through edge splitting refinement, focusing on a dam break case in both 2D and 3D. To this end, element quality factors are computed element-wise and collected into a single mesh quality score by an average of the element's quality. In order to be considered as a fair measure of elemental shape, the quality factor must possess the following attributes: First, it yields a zero-quality for degenerated elements. Second, it must be non-dimensional such that similar simplices of different sizes yield the same measurement. Third, it must take on positive values between zero and one [137]. From a theoretical perspective, indicators based on the Jacobian matrix have been demonstrated as relevant in the context of FEM because it is based on the matrix transformation from an ideal element to the physical element. Specifically, the definition of the mean ratio quality measure provided by Liu and Joe in [138, 139] is given by

$$Q = d \frac{\det(\mathbf{A}^\top \mathbf{A})^{1/d}}{\text{tr}(\mathbf{A}^\top \mathbf{A})} \quad (2.89)$$

Here, d is the dimension of the considered simplex, and $\mathbf{A}$ is the matrix involved in the transformation from a regular simplex (i.e. an equilateral triangle or tetrahedron) of the same volume to the simplex whose quality is measured. Mathematically, the transformation matrix is defined such that

$$\mathbf{x}_j = \mathbf{A} \boldsymbol{\xi}_j + \mathbf{b} \quad (2.90)$$

where $\mathbf{x}_j$ is the coordinate vector of the j -th node of the finite element, $\boldsymbol{\xi}_j$ is the coordinate vector of the j -th node of the regular simplex of same volume and $\mathbf{b}$ is a translation vector. In 3D, the components of the transformation matrix can be computed by writing the above definition in matrix form and solving the resulting set of 3 linear systems, one for each dimension of the space:

$$\begin{pmatrix} A_{11} & A_{21} & A_{31} \\ A_{12} & A_{22} & A_{32} \\ A_{13} & A_{23} & A_{33} \\ b_1 & b_2 & b_3 \end{pmatrix} = \begin{pmatrix} \xi_1 & \eta_1 & \zeta_1 & 1 \\ \xi_2 & \eta_2 & \zeta_2 & 1 \\ \xi_3 & \eta_3 & \zeta_3 & 1 \\ \xi_4 & \eta_4 & \zeta_4 & 1 \end{pmatrix}^{-1} \begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \end{pmatrix} \quad (2.91)$$

In 2D, a triangle with $Q = 1$ is perfectly equilateral and $Q = 0$ indicates perfectly flat elements. In 3D, a tetrahedron with $Q = 1$ is perfectly regular and $Q = 0$ indicates perfectly flat sliver. Some examples are provided in Figure 17. In addition to assessing the quality of the elements, it is crucial to ensure that the finite element sizes in the mesh closely match the expected size-field, as determined by the function $h(\mathbf{x})$. To achieve this, we define the difference between the average edge length of an element e and the characteristic length as follows:

$$\text{Error}(e) = \left\{ \frac{1}{N} \sum_{j=1}^N L_j - L'(h_e) \right\}^2 \left\{ L'(h_e) \right\}^{-2} \quad (2.92)$$

Here, L_j is the length of the j -th edge of the element, N represents the number of edges, h_e is the average characteristic size of e as defined in (2.80), and $L'(h_e)$ is a reference length. Specifically, $L'(h_e)$ is the expected edge length of a regular simplex of characteristic size h_e . In other words, $L'(h_e)$ is the average edge length of the simplex that would have been produced by the meshing software for a characteristic size h_e . Intuitively, one may expect $L'(h_e) = h_e$ such that an ideal element of size h_e would have edges of sizes h_e . However, the reference length depends on the meshing software used for constructing the initial mesh. In this work, all input meshes have been produced by Gmsh, which uses two reference sizes:

$$\begin{aligned} L'(h_e) &\simeq 0.95 h_e \quad \text{in 2D} \\ L'(h_e) &\simeq 1.29 h_e \quad \text{in 3D} \end{aligned} \tag{2.93}$$

The above relations have been empirically computed based on different input meshes produced by Gmsh [93] and will be used as reference length throughout this study. For 2D meshes, the default Frontal-Delaunay algorithm has been used by setting `Mesh.Algorithm = 6`. For 3D input meshes, the default Delaunay algorithm has been used by setting `Mesh.Algorithm3D = 1`. It is important to note that the optimal values of the remeshing parameters (specifically, ω and δ) that are established in the following study minimize (2.92). This means that we seek to maintain the relation between characteristic size and edge length (2.93) throughout the simulation. As a result, optimal values of ω and δ may be different if the input mesh is generated by a different software than Gmsh.

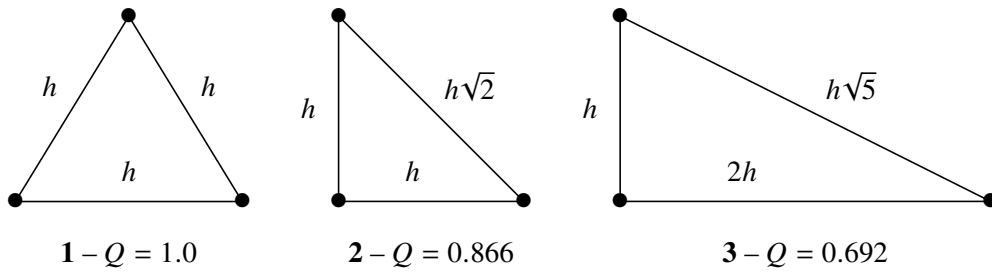


Figure 17: Examples of triangular elements with their quality. The quality measure Q is defined in (2.89) with respect to a regular simplex with the same area, meaning that the equilateral triangle has perfect quality.

Dam Break in 2D

The dam break test case is commonly used to validate numerical methods for free surface flows due to sudden water release, resulting in rapid, highly unsteady dynamics. Consequently, the free surface experiences violent deformations, such as breaking waves, splashing, and aeration. A schematic representation of the initial configuration is shown in Figure 18. The simulation parameters are listed in Table 2.

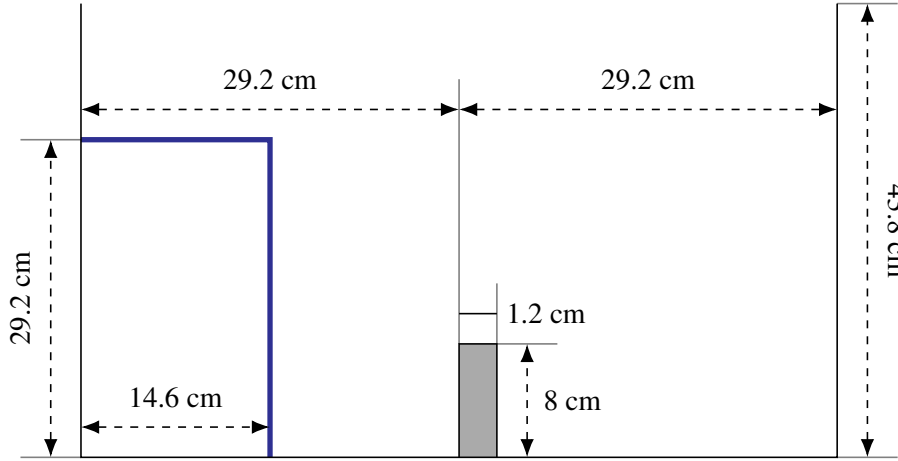


Figure 18: Initial configuration of the **dam break** test case whose simulation parameters are indicated in Table 2. No-slip BC are applied to all rigid boundaries and the initial free surface is outlined in bold.

<i>Remesh</i>	<i>Fluid</i>
$\delta = *$	$\rho = 10^3 \text{ kg/m}^3$
$\omega = *$	$g = -9.81 \text{ m/s}^2$
$\alpha = 1.2$	$\mu = 10^{-3} \text{ Pa s}$
$\beta = 0.2$	$\gamma = 0.072 \text{ N/m}$
	$tol = 10^{-8}$
	$\Delta t_{\text{MAX}} = 10^{-3} \text{ s}$

Table 2: Simulation parameters for the **dam break** whose initial configuration is given in Figure 18. The governing equations are solved with the PSPG stabilization and different remeshing techniques.

The symbol $*$ indicates that the corresponding parameter, specifically, δ which governs the coarsening criterion of an edge and ω which governs the refinement criterion of an element, are tested with varying values. The value of $\alpha = 1.2$ in Table 2 for the α -shape criterion is based on findings in the literature [29, 78, 79], where it has been shown that $\alpha = 1.2$ strikes a good balance between discarding large or distorted simplices and accepting good-quality elements in 2D meshes.

On the other hand, the purpose of β is to prevent nodes from moving too close to the boundary, as defined by the criterion in Equation (2.85). In practice, a value of β between 0.1 and 0.2 is sufficient to prevent nodes from crossing boundaries without significantly compromising the mesh quality or its alignment with the size-field. As a result, only δ and ω require careful tuning and analysis. To ensure a smooth variation in element sizes away from the free surface and walls, the mesh size-field is governed by the following function:

$$h(\mathbf{x}) = \min \left\{ h_0 + 0.1 d(\mathbf{x}), h_\infty \right\} \quad (2.94)$$

Here, $d(\mathbf{x})$ represents the smallest distance between the position $\mathbf{x}$ and the exterior of the fluid, $h_0 = 0.5$ cm is the minimum characteristic size of the mesh elements, and $h_\infty = 2$ cm is the maximum characteristic size. Figures 19 and 20 show the average error (2.92) in the element size and the average element quality Q over 1 second of simulation for the dam break problem. These averages are both spatial (across all elements in the fluid) and temporal (over the total simulation time). The centroid insertion method, as described in previous works [16, 80], involves inserting a node at the centroid of large or distorted elements. The edge splitting refers to the mesh refinement method implemented in this work. The results indicate that the edge splitting method enhances the mesh's compliance with the imposed size-field (2.94) and results in better element quality compared to the centroid insertion method.

An important point to note is that both the centroid insertion and edge splitting methods share a common set of parameters in our implementation. For instance, the parameter δ is consistently applied as the coarsening criterion for an edge, in accordance with (2.86), across both methods. Similarly, ω is used as defined in (2.82) to govern the refinement criterion for an element. However, the location and number of nodes added depend on the specific method chosen. The optimal values of ω and δ are selected to minimize the average error in the size-field.

The primary goal of local refinement and coarsening is not to maximize the element quality measure, but rather to closely match the imposed size-field, with mesh quality being a secondary (yet still important) consideration. In contrast, attempting to maximize the quality measure would require reducing ω below 0.4, which would overly stimulate element refinement. This would result in a gradual increase in the number of nodes in the mesh. Figure 21 illustrates the time evolution of both the average size error and element quality when ω and δ are set to their optimal values.

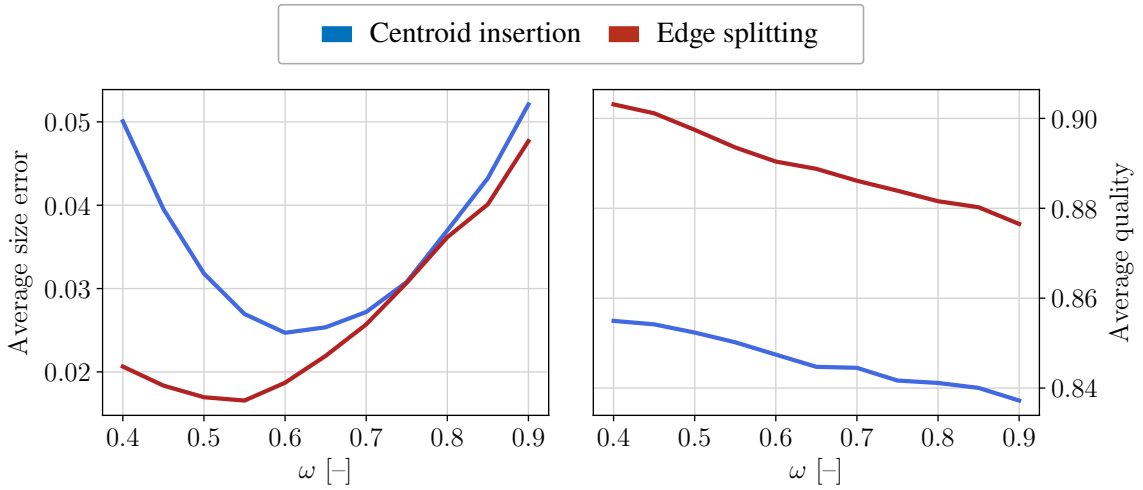


Figure 19: Comparison between the centroid insertion (with $\delta = 0.6$) and the edge splitting method (with $\delta = 0.6$) on the average size-error (2.92) and average element quality for the **dam break** test case in 2D, as a function of ω .

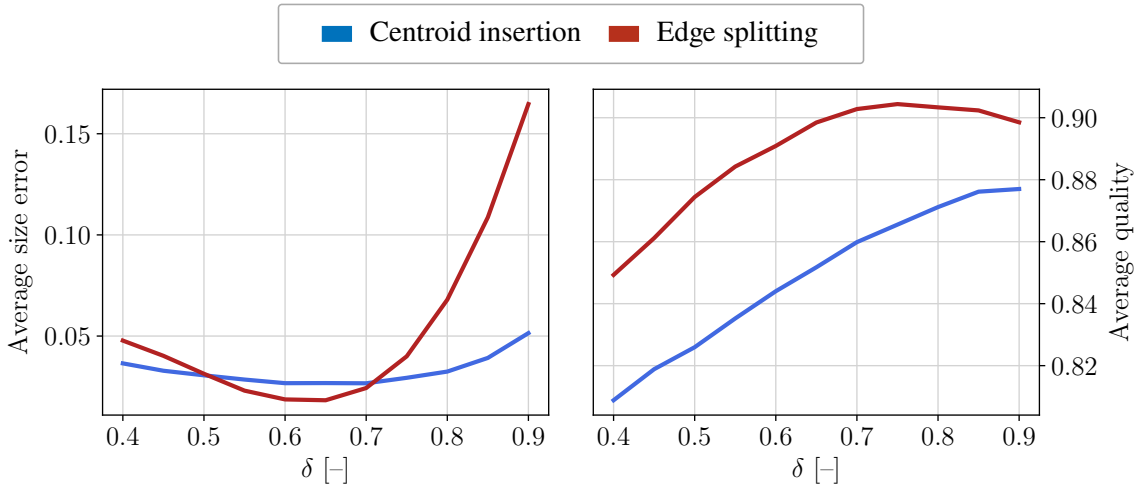


Figure 20: Comparison between the centroid insertion (with $\omega = 0.6$) and the edge splitting method (with $\omega = 0.6$) on the average size-error (2.92) and average element quality for the **dam break** test case in 2D, as a function of δ .

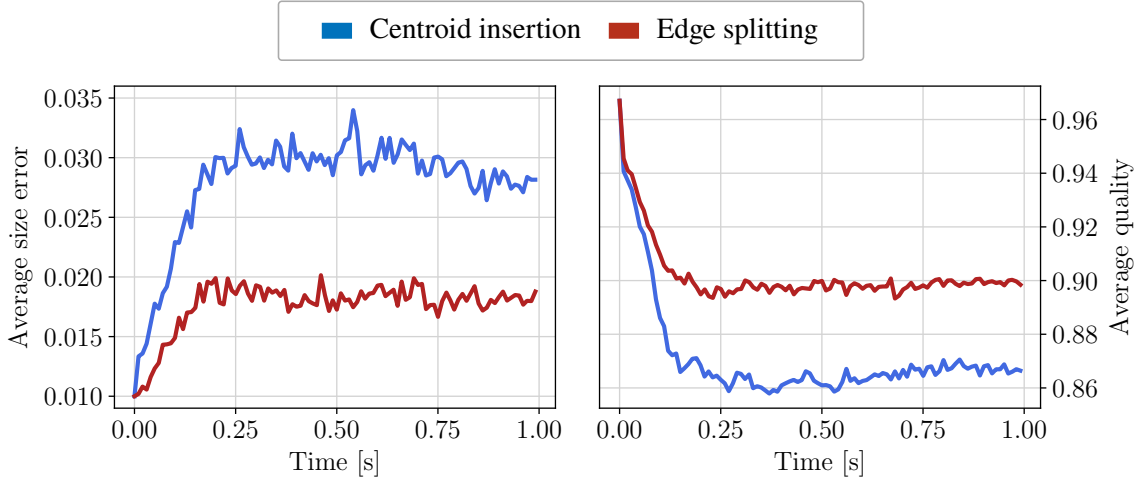


Figure 21: Comparison between the centroid insertion (with $\omega = 0.6$ and $\delta = 0.75$) and the edge splitting method (with $\omega = 0.6$ and $\delta = 0.6$) on the average size-error (2.92) and average element quality for the **dam break** with obstacle in 2D.

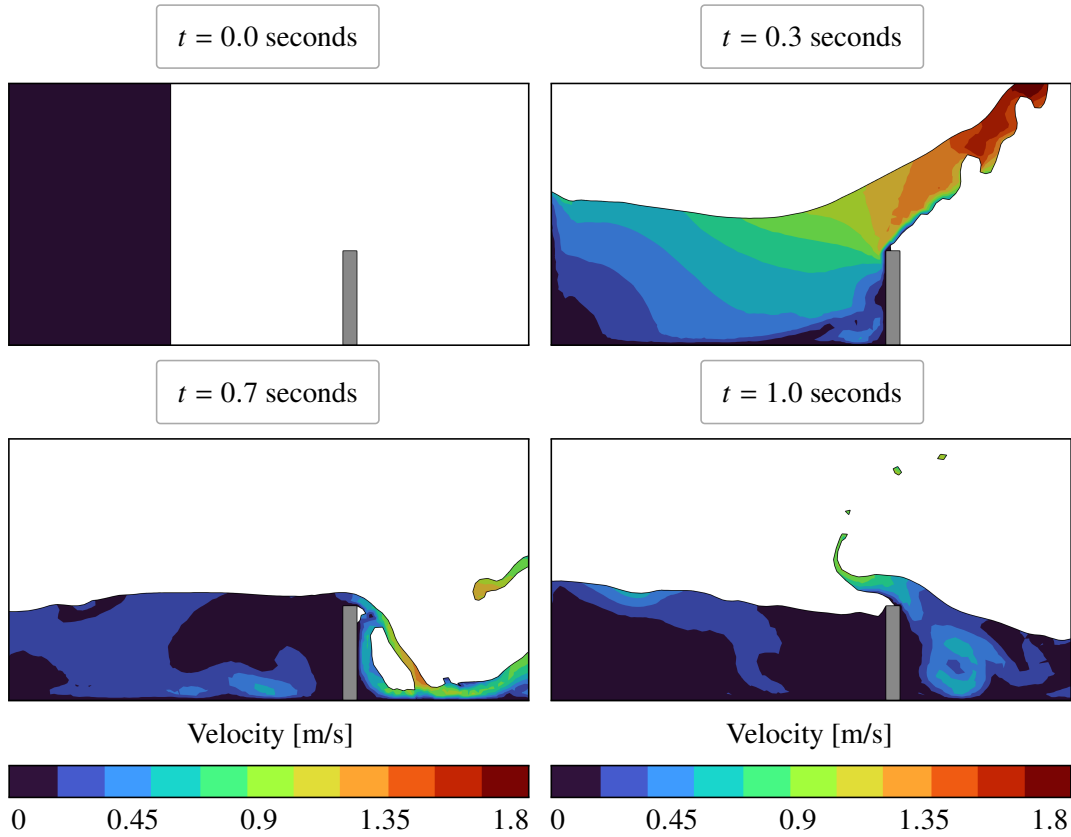


Figure 22: Snapshots of the particle finite element solution for the **dam break** test case whose initial configuration is given in Figure 18, at different time intervals. The parameters from Table 2 with $\omega = 0.6$ and $\delta = 0.6$ have been used.

As shown, the edge splitting method reduces the loss of element quality over time, ultimately achieving a slightly higher average quality compared to the centroid insertion method. It is important to note that at time zero, the initial mesh quality reflects that of the input mesh generated by Gmsh, while not perfect, is considered optimal for the simulation's purposes. Similarly, the average error in the size-field at time zero is calculated based on the input mesh and should be regarded as ideal, despite being non-zero. This small initial error is due to the fact that (2.93) is not an exact relation.

Snapshots of the solution with edge splitting remeshing at different time intervals are presented in Figure 22. A comparison of the FE mesh is shown in Figures 23 and 24, highlighting the greater smoothness of the free surface achieved with edge splitting refinement, compared to centroid insertion. This enhanced surface smoothness is crucial when surface tension effects are significant, as sharp angles can lead to instabilities and undesirable oscillations in the fluid. As a side note, a common strategy to further improve free surface representation is to allow edges along the free surface to be smaller than those in the bulk. This can be achieved by using a smaller value $\delta_{\text{FS}} < \delta$ for the coarsening criterion applied only to edges forming the free surface.

Dam Break in 3D

A comparison between centroid insertion and edge splitting refinement is carried out using an extension of the dam break test case in 3D, as shown in Figure 25. The choice of $\beta = 0.2$ in Table 3 remains the same as in the 2D case for similar reasons. However, the value of α must be adjusted in 3D. As a reminder, the α -shape criterion $\mathcal{A}$ compares the characteristic size of the element with its circumradius. Specifically, the circumradii r_2 and r_3 of a regular triangle and tetrahedron are respectively given by

$$\begin{aligned} r_2 &= L\sqrt{3}/3 \\ r_3 &= L\sqrt{6}/4 \end{aligned} \tag{2.95}$$

Here, L is the length of the edges of the triangle or tetrahedron. As mentioned earlier, the relationship between the characteristic size h_e of an element e and the length of its edges in a mesh generated by Gmsh is given by $L \simeq 0.95 h_e$ in 2D and by $L \simeq 1.29 h_e$ in 3D. Consequently, one can write

$$\frac{r_3}{r_2} = \frac{1.29 h_e (3\sqrt{6})}{0.95 h_e (4\sqrt{3})} \simeq 1.44 \tag{2.96}$$

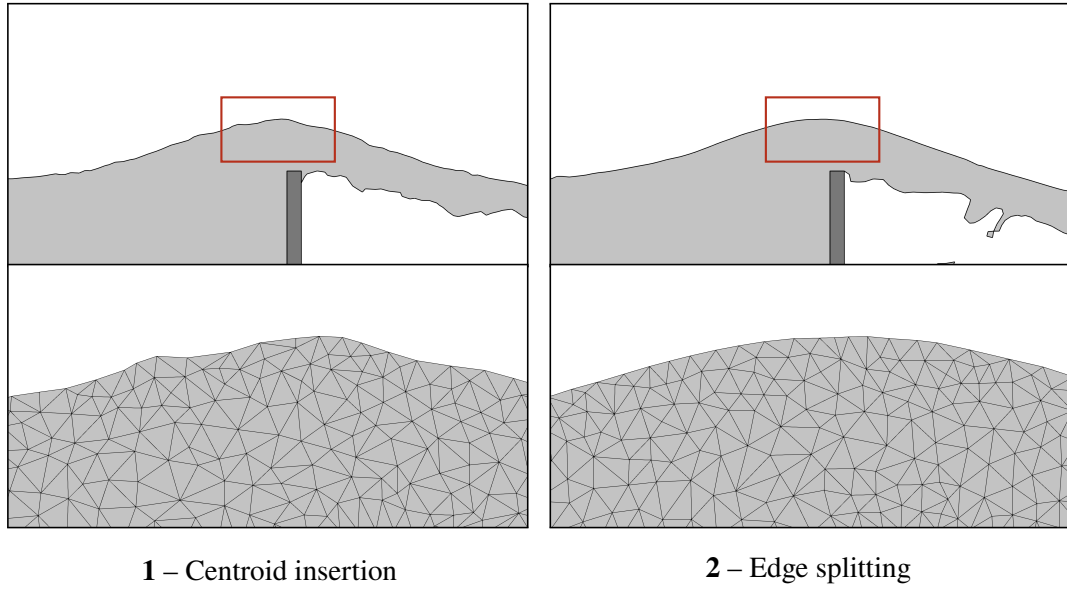


Figure 23: Outline of the free surface for the **dam break** test case whose initial configuration is given in Figure 18, at time $t = 0.5$ seconds. A zoom on the region outlined by the red rectangle is also presented below.

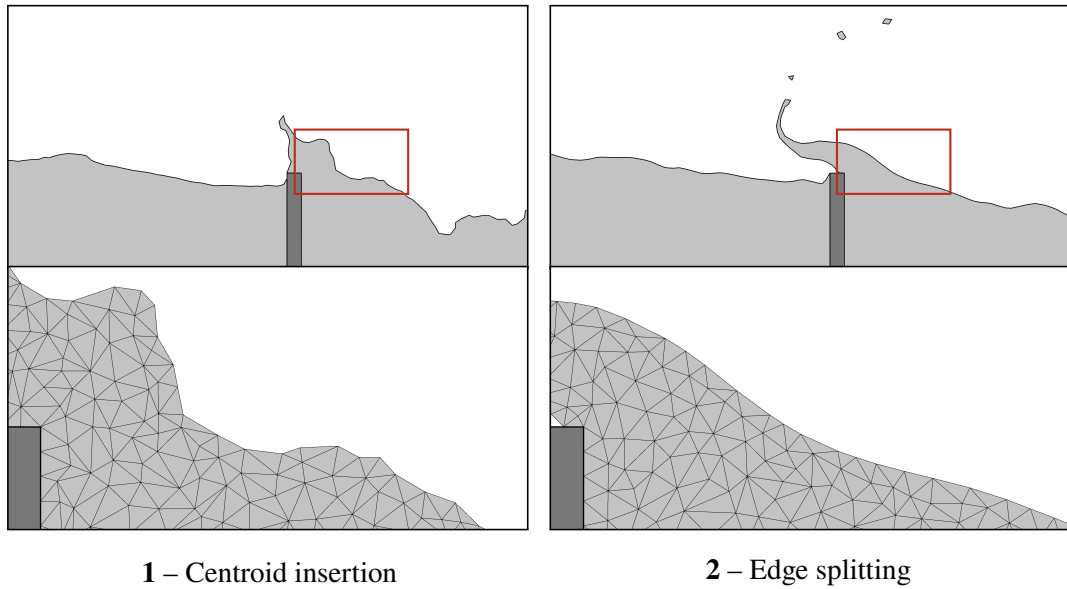


Figure 24: Outline of the free surface for the **dam break** test case whose initial configuration is given in Figure 18, at time $t = 1.0$ seconds. A zoom on the region outlined by the red rectangle is also presented below.

This implies that for the same characteristic size, the circumradius of a regular tetrahedron in Gmsh is approximately 44% larger than its 2D counterpart. Therefore, the value of $\alpha = 1.7$ is used in 3D to maintain the same level of tolerance in the α -shape criterion (2.78). The mesh size-field is governed by the distance to any boundary and is defined by the same function as in the 2D test case:

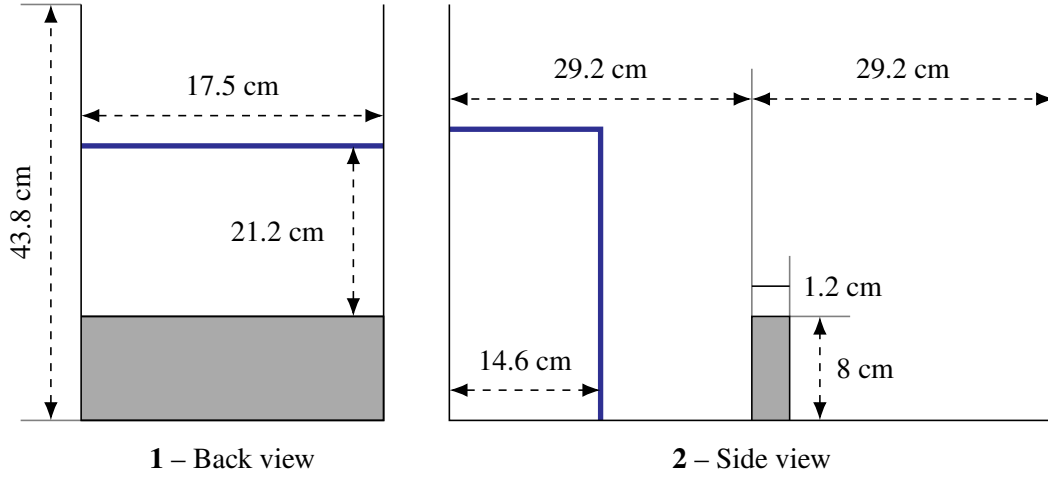


Figure 25: Initial configuration of the **dam break** test case whose simulation parameters are indicated in Table 3. No-slip BC are applied to all rigid boundaries and the initial free surface is outlined in bold.

<i>Remesh</i>	<i>Fluid</i>
$\delta = *$	$\rho = 10^3 \text{ kg/m}^3$
$\omega = *$	$g = -9.81 \text{ m/s}^2$
$\alpha = 1.7$	$\mu = 10^{-3} \text{ Pa s}$
$\beta = 0.2$	$\gamma = 0.072 \text{ N/m}$
	$tol = 10^{-8}$
	$\Delta t_{\text{MAX}} = 10^{-3} \text{ s}$
	$\lambda = 0.5$

Table 3: Simulation parameters for the **dam break** whose initial configuration is given in Figure 25. The governing equations are solved with the fractional step and different remeshing techniques.

$$h(\mathbf{x}) = \min \left\{ h_0 + 0.1 d(\mathbf{x}), h_\infty \right\} \quad (2.97)$$

with $h_0 = 1$ cm and $h_\infty = 5$ cm. Figures 26 and 27 show the average error (2.92) in the mesh element size and the average element quality Q over 1 second of simulation for the dam break. The results confirm the conclusions drawn from the 2D case, indicating that the edge splitting method improves the mesh compliance with the imposed size-field (2.97) and yields better element quality compared to the centroid insertion method.

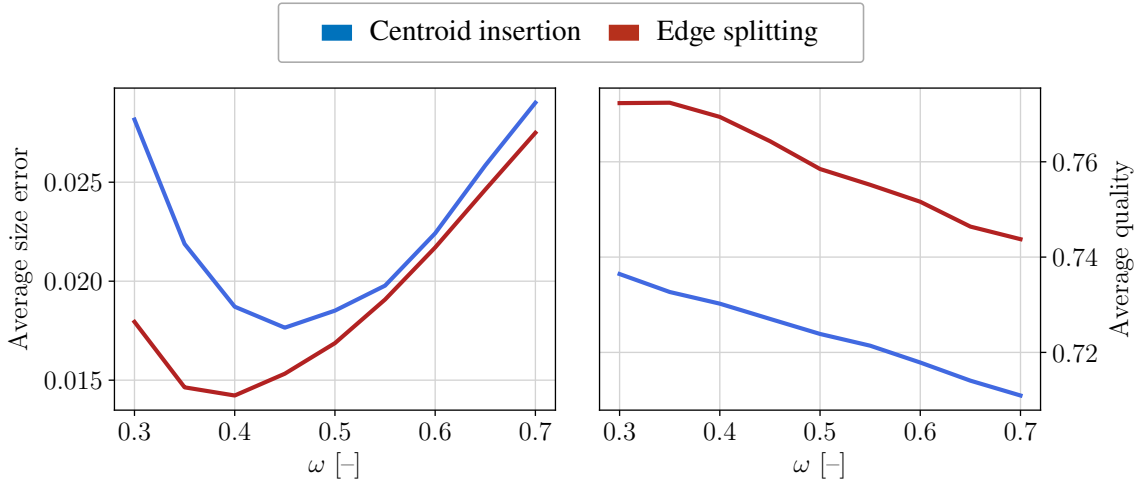


Figure 26: Comparison between the centroid insertion (with $\delta = 0.8$) and the edge splitting method (with $\delta = 0.8$) on the average size-error (2.92) and average element quality for the **dam break** test case in 3D, as a function of ω .

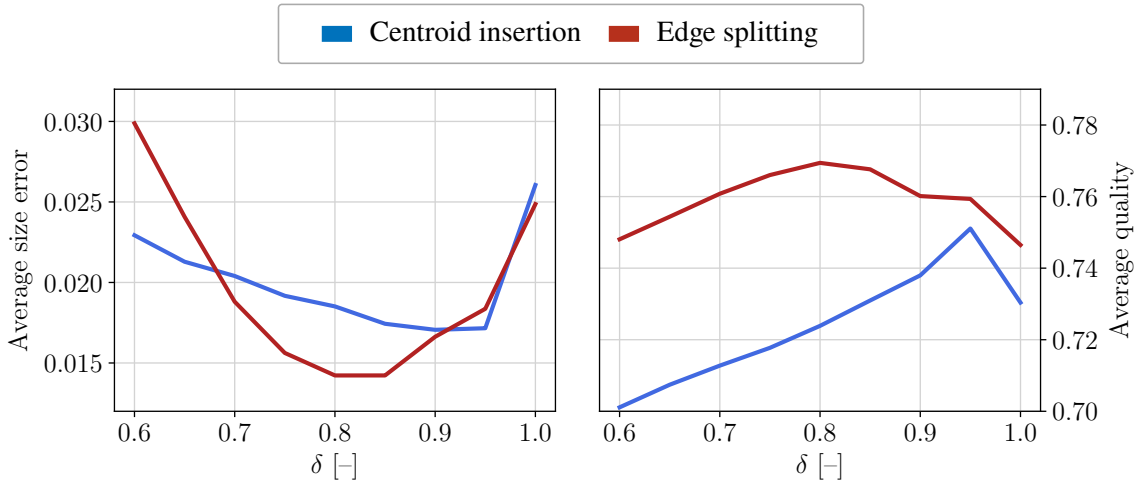


Figure 27: Comparison between the centroid insertion (with $\omega = 0.45$) and the edge splitting method (with $\omega = 0.4$) on the average size-error (2.92) and average element quality for the **dam break** test case in 3D, as a function of δ .

As in the 2D case, optimal values of ω and δ are selected to minimize the average error in the size-field. Figure 28 illustrates the time evolution of both the average size error and element quality when ω and δ are set to their optimal values. Snapshots of the solution with the edge splitting remeshing are presented in Figure 29. Additionally, a comparison of the FE mesh is provided in Figures 30 and 31, highlighting the smoother free surface achieved with edge splitting refinement compared to centroid insertion. However, some irregularities are still observed near the boundaries (and near the obstacle), even with edge splitting. These irregularities are primarily caused by the no-slip boundary condition and the fixed positions of boundary nodes, which prevent them from aligning properly with the free surface plane, resulting in a non-smooth surface.

It is crucial to emphasize that a mesh optimization step using TetGen was rigorously applied to both the edge splitting and centroid insertion approaches. This optimization is essential for two reasons. First, the presence of slivers can prevent the solver from completing the simulation successfully. Second, simply removing these slivers would create zero-volume gaps in the fluid mesh. These gaps introduce numerical singularities, disrupt the free surface, and interfere with the size-field function, which is distance-based relative to the free surface. TetGen offers several optimization levels for edge/face flips and vertex smoothing, which governs the extent at which the tessellation can be modified to achieve a better mesh quality. Here, the default optimization level associated with the `-O3` flag in version 1.6.0 of the library has been used. In our experience, applying edge/face flips at a moderate optimization level is sufficient to regularize all slivers for the test cases presented in this subsection.

Vertical Sloshing

A test case of vertical sloshing in a rectangular water tank has been experimentally studied by González et al. [140], focusing on the position of the free surface during a period of oscillation once a stable periodic regime has been established. In this work, we validate our implementation of the PFEM against those experimental results in both 2D and 3D. Figure 32 provides a schematic representation of the initial geometry of the water tank in 2D. A periodic vertical displacement with an amplitude of 0.15 cm and a frequency of 4.8 Hz is imposed on the tank. The characteristic element size at the free surface and the vertical walls is set to $h_0 = 0.15$ cm. The mesh size-field is defined as a function of the smallest distance $d(\mathbf{x})$ to the vertical walls or the free surface:

$$h(\mathbf{x}) = \min \left\{ h_0 + h_\infty \frac{d(\mathbf{x})}{H}, h_\infty \right\} \quad (2.98)$$

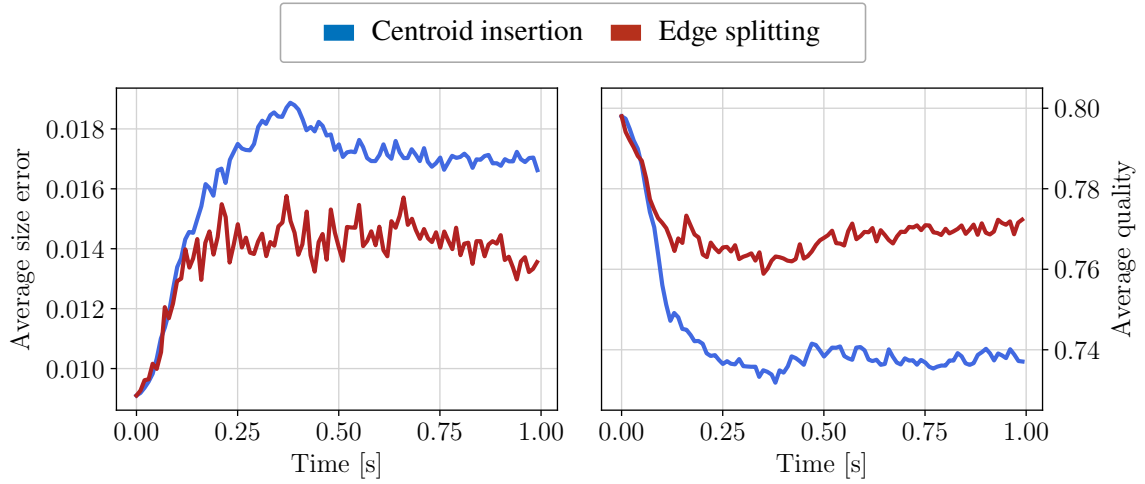


Figure 28: Comparison between the centroid insertion (with $\omega = 0.45$ and $\delta = 0.95$) and the edge splitting method (with $\omega = 0.4$ and $\delta = 0.8$) on the average size-error (2.92) and average element quality for the **dam break** with obstacle in 3D.

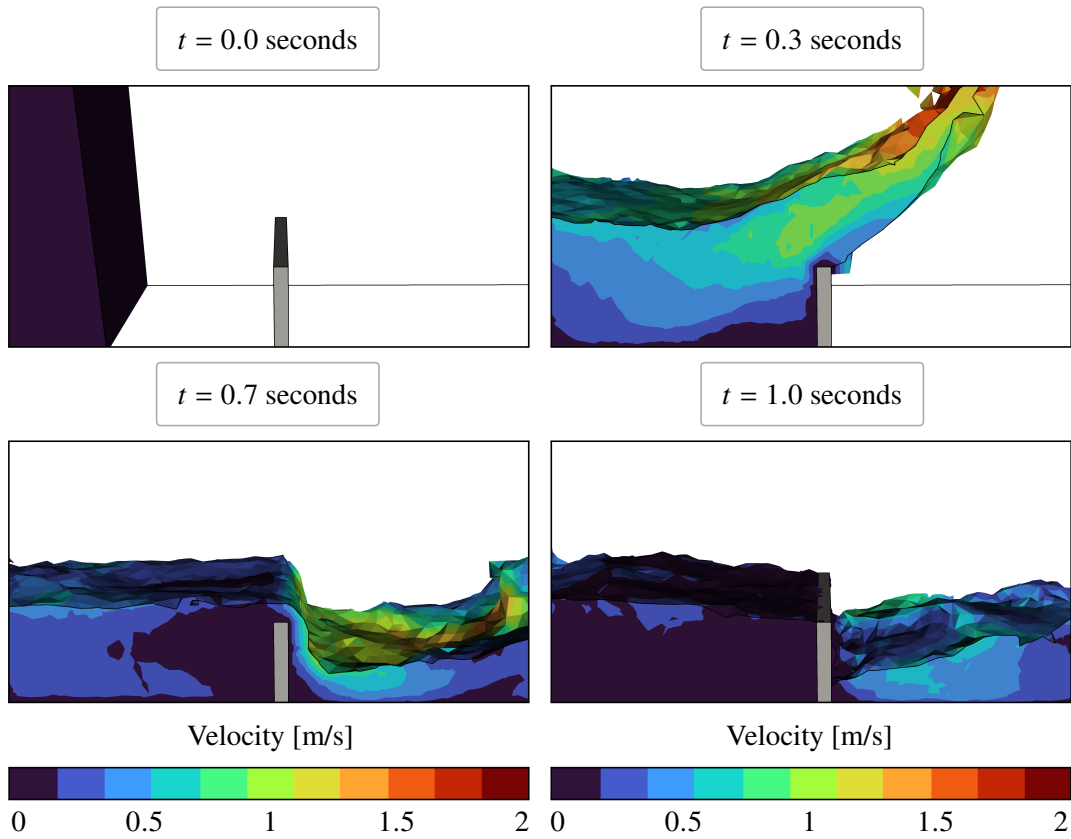


Figure 29: Snapshots of the particle finite element solution for the **dam break** test case whose initial configuration is given in Figure 25, at different time intervals. The parameters from Table 3 with $\omega = 0.4$ and $\delta = 0.8$ have been used.

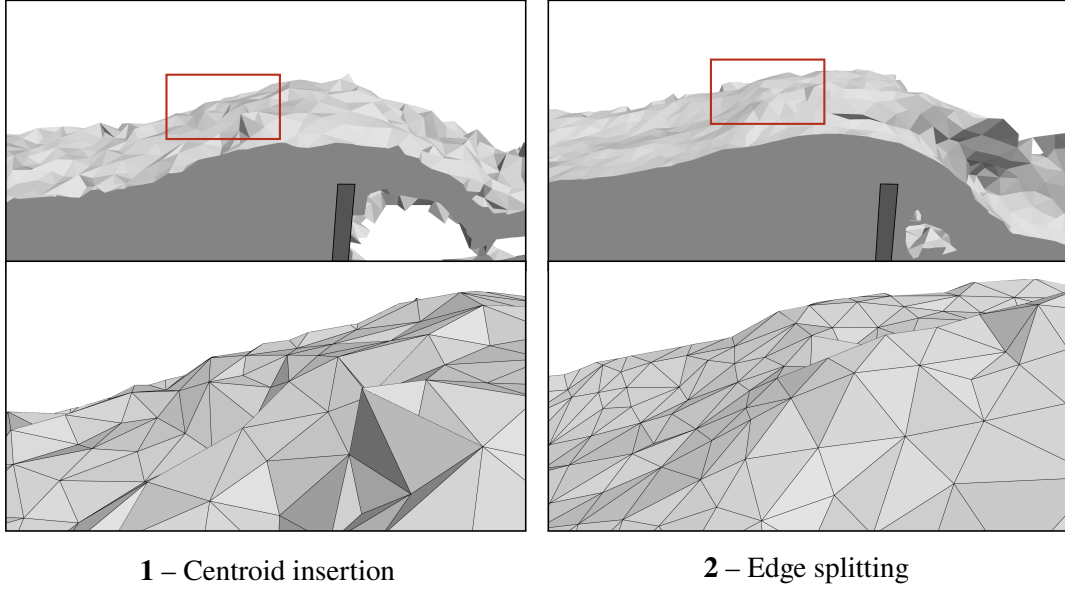


Figure 30: Outline of the free surface for the **dam break** test case whose initial configuration is given in Figure 25, at time $t = 0.5$ seconds. A zoom on the region outlined by the red rectangle is also presented below.

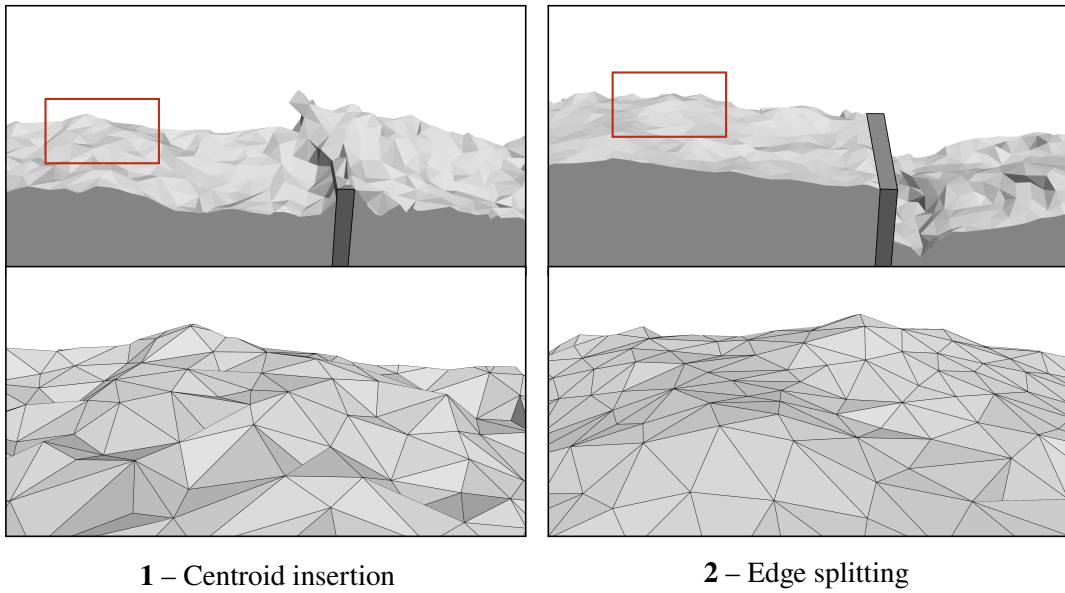


Figure 31: Outline of the free surface for the **dam break** test case whose initial configuration is given in Figure 25, at time $t = 1.0$ seconds. A zoom on the region outlined by the red rectangle is also presented below.

Here, $H = 10$ cm is the initial fluid level and $h_{\infty} = 0.5$ cm is the maximum allowed size for the finite elements. In 3D, the initial fluid level and material properties are consistent with those in 2D, with a tank depth of 20 cm. Figure 33 shows the position of the free surface at two stages within one oscillation period, highlighting the agreement between the experimental observations and the simulation results. An overview of the free surface in 3D is provided in Figures 34 and 35, demonstrating that the oscillation mode aligns with the length of the tank and remains invariant along the width, as expected.

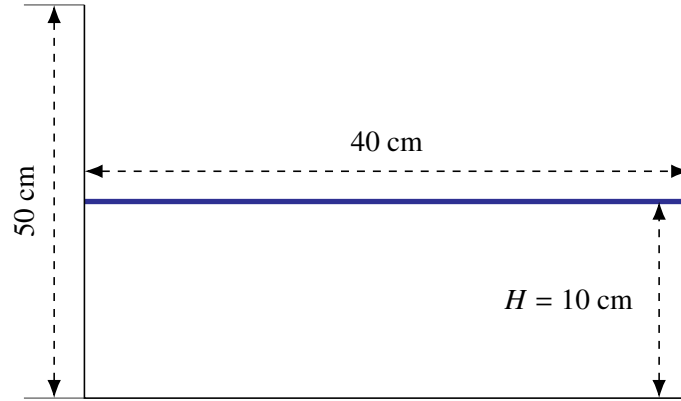


Figure 32: Initial configuration of the **vertical sloshing** test case whose simulation parameters are indicated in Table 4. No-slip BC are applied to all rigid boundaries and the initial free surface is outlined in bold.

<i>Remesh 2D</i>	<i>Remesh 3D</i>	<i>Fluid</i>
$\delta = 0.6$	$\delta = 0.8$	$\rho = 10^3 \text{ kg/m}^3$
$\omega = 0.6$	$\omega = 0.4$	$g = -9.81 \text{ m/s}^2$
$\alpha = 1.2$	$\alpha = 1.7$	$\mu = 10^{-3} \text{ Pa s}$
$\beta = 0.2$	$\beta = 0.2$	$\gamma = 0.072 \text{ N/m}$
		$tol = 10^{-8}$
		$\Delta t_{\text{MAX}} = 10^{-3} \text{ s}$

Table 4: Simulation parameters for the **vertical sloshing** whose initial configuration is given in Figure 32. The governing equations are solved with the PSPG stabilization and the edge splitting remeshing.

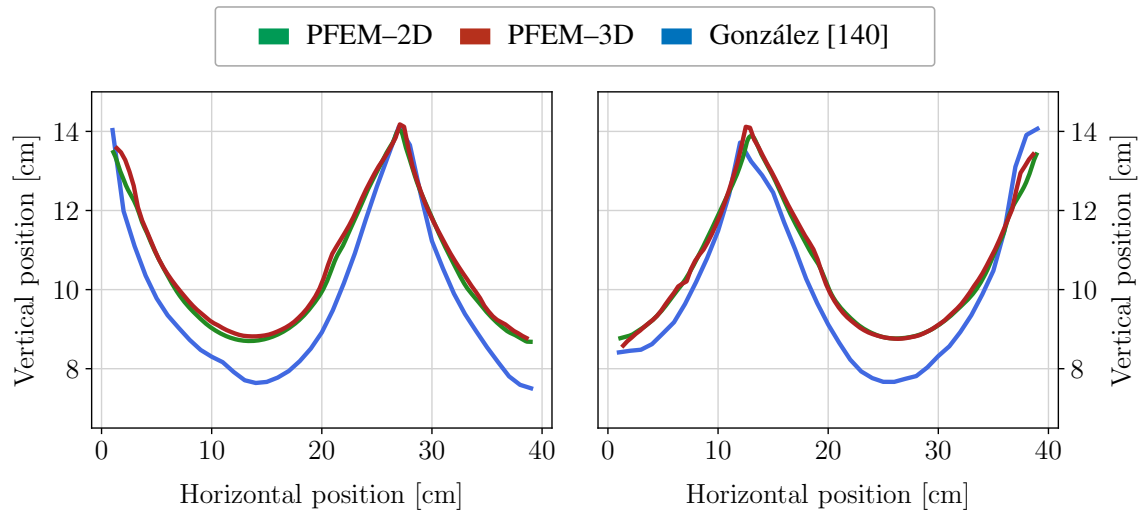


Figure 33: Position of the free surface for the **vertical sloshing** of a water tank respectively at 25% and 75% of an oscillation period of 0.4 seconds. The results are compared to experimental data obtained by González et al.

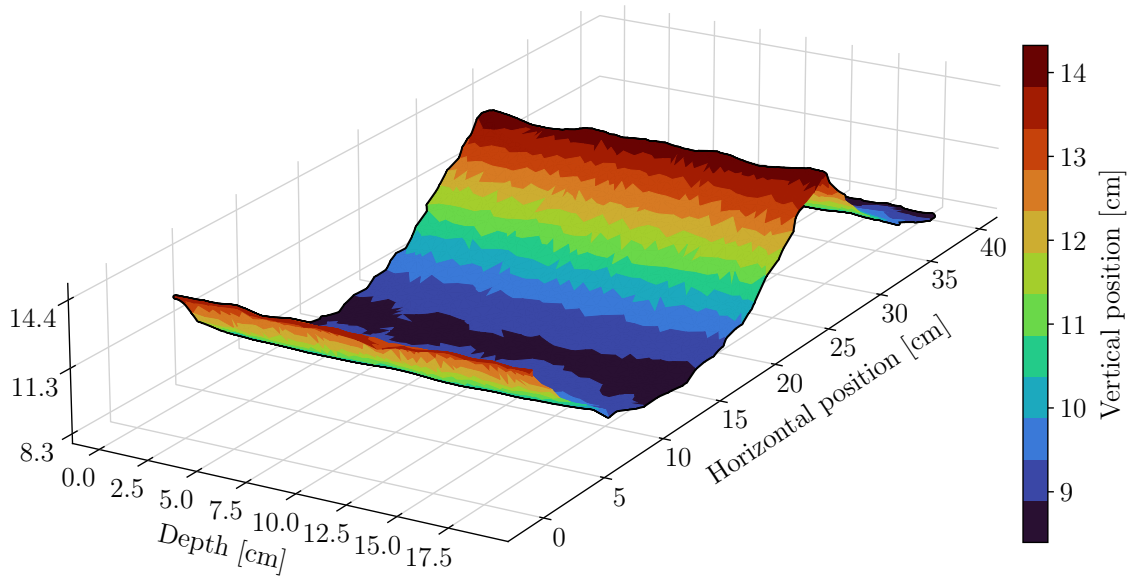


Figure 34: Vertical position and shape of the free surface for the **vertical sloshing** of a water tank in 3D. The figure shows the free surface of the fluid at 25% of an oscillation period of 0.4 seconds.

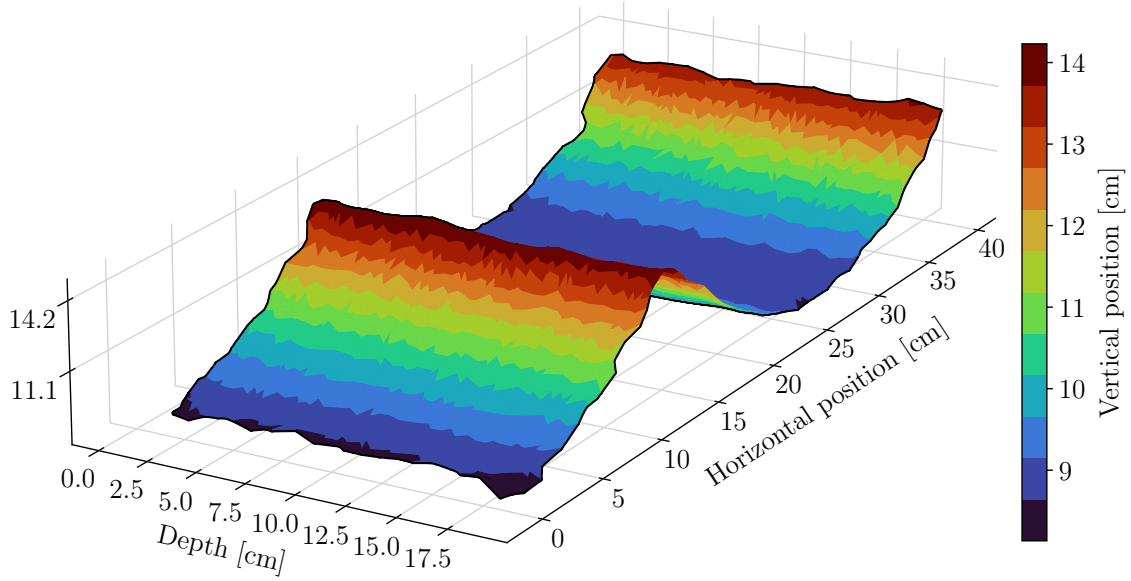


Figure 35: Vertical position and shape of the free surface for the **vertical sloshing** of a water tank in 3D. The figure shows the free surface of the fluid at 75% of an oscillation period of 0.4 seconds.

2.3 Conservation of Mass

When two free surfaces come sufficiently close, the connecting simplices generated by Delaunay tessellation typically satisfy the α -shape criterion and are thus accepted as fluid elements. This acceptance is crucial for enabling the merging of fluid bodies upon contact. However, it also leads to non-physical mass creation, as these connecting elements are artificially introduced without a corresponding representation in the last-known configuration of the fluid domain. In recent works, Falla et al. [78] addressed the problem of mass variation by a merging detection algorithm. The latter involves an additional condition along with the classical α -shape. A simplex is discarded when either of the two following conditions is met:

$$\begin{aligned} r_e &> \alpha h_e(\mathbf{x}) \\ A_e &> 2 \min_n (h_n)^2 \quad \forall n \in e(\mathbf{x}) \end{aligned} \quad (2.99)$$

Here, e represents an element, A_e is its area and n corresponds to the indices of the nodes of this element. Thereby, this criterion is only applicable to 2D meshes. The first condition is the α -shape criterion $\mathcal{A}$. The purpose of the second condition is to minimize the spurious creation of mass.

A new fluid element is only created if its size is comparable to that of its smallest neighbor. Additionally, when the simplex only satisfies the second condition, its largest neighbor is refined. More specifically, the target mesh size of the node lying on the coarser of the two boundaries is set to the minimum target mesh size. In other words, the approach of Falla et al. [78] consists in extensively refining the mesh in areas where two free surfaces come sufficiently close to each other, thus reducing the spurious mass creation by drastically reducing the size of connecting elements. While this approach effectively helps to conserve the total mass of the fluid, extensive refinement may lead to significant computational costs while solving the resulting FE system, especially when the free surface experiences violent deformations, such as breaking waves, splashing, and aeration. In our work, Falla's approach is also adopted to a lesser extent, as the size-field $h(\mathbf{x})$ is generally defined such that elements close to the free surface have a finer resolution than those far from it. However, the main contribution of this work to address the problem of mass creation lies in the implementation of a Level Set (LS) function.

The Level Set

The PFEM method incorporating a LS function (PFEM-LS) has been first proposed by Fernández et al. [79], and shares notable similarities with the algorithm proposed in this work. Both approaches use a signed distance d_Γ to the fluid's boundary as an additional filter to constrain the Delaunay tessellation based on the previously known fluid domain. The key difference lies in how the signed distance is computed. In the method presented in [79], a radial basis function is fitted to the fluid boundary. While this technique is both innovative and foundational of the PFEM-LS framework, it suffers from certain limitations. First, the evaluation of d_Γ may be inaccurate, since the radial basis function approximates rather than exactly represents the boundary. Second, the approach incurs significant computational costs, given that a highly dense linear system must be solved to obtain the level set polynomial function. In this way, the time required to construct the LS function in 2D is reported to be comparable to that of solving the finite element system at every time step. Moreover, this cost increases substantially as higher accuracy is required in the LS function.

In contrast, the PFEM-LS algorithm proposed in this work consists of simple, independent operations that are easily parallelized. It also computes the exact signed distance to the discretized boundary, rather than relying on an approximation. Additionally, the construction and evaluation of the distance function account for only a small fraction of the total remeshing time, even in large-scale 3D simulations. Further details on computation times are provided in the next chapter.

This subsection details the detection of such connecting simplices using a ray-casting algorithm, the core component of the level set approach implemented in this work. The α -shape with LS method seeks to minimize non-physical mass creation by accepting connecting elements only if they are sufficiently small. An element is accepted only if it meets the following set of criteria:

$$\mathcal{L}\{e(\mathbf{x})\} = \text{True} \Leftrightarrow \& \begin{cases} r_e < \alpha h_e(\mathbf{x}) \\ d_\Gamma < \varepsilon h_e(\mathbf{x}) \end{cases} \quad (2.100)$$

Here, $\&$ is the logical conjunction, also known as the logical AND, meaning that $\mathcal{L}$ returns true for an element e only if both inequalities are met simultaneously. The first condition corresponds to the classical α -shape criterion $\mathcal{A}$, while the second is based on the LS criterion. The scalar parameter $\varepsilon \geq 0$ is defined globally by the user. The variable d_Γ denotes the signed distance between the centroid of the element and the boundary Γ of the most recently known fluid mesh. The magnitude of this distance represents the Euclidean distance to the fluid boundary, as illustrated in Figure 36, while its sign indicates whether the centroid lies inside or outside the closed boundary defining the fluid domain. Specifically, the sign is positive when the centroid is outside the boundary and negative when it is inside. By combining these two conditions, the method ensures that any elements created outside the domain are sufficiently small or flat, thereby preventing significant non-physical mass increase in the system.

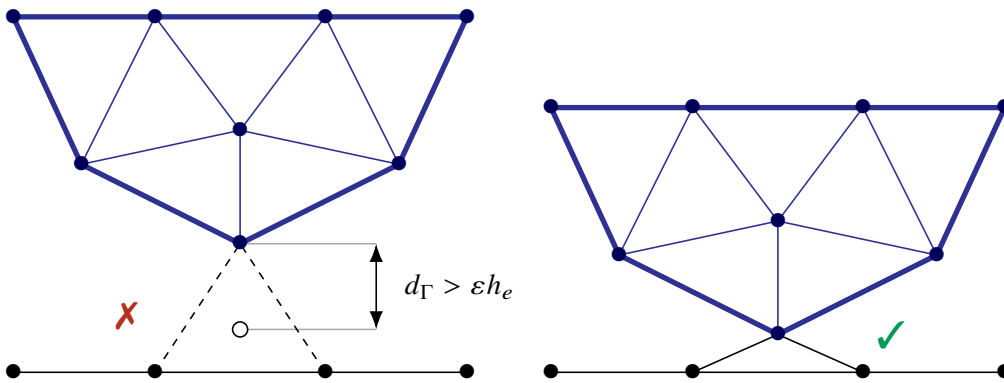


Figure 36: Example of connecting simplex between the free surface (outlined in bold) and a rigid boundary drawn by regular lines at the bottom. The signed distance $d_\Gamma > 0$ because the simplex's centroid lies outside the fluid domain.

2.3.1 Alpha-Shape with Level Set

Our implementation of the LS is based on the ray-casting algorithm, which is grounded in the theoretical framework of the Jordan theorem. A Jordan curve is the image of a continuous and injective mapping from a circle to the plane. According to the Jordan theorem, the complement of such a curve in the real plane consists of exactly two connected regions, one bounded and one unbounded. In other words, the curve divides the plane into an interior (bounded) region and an exterior (unbounded) region, as illustrated in Figure 37. Although this result may appear intuitive, its formal proof is nontrivial [141].

Let $\mathcal{S}$ be a d -dimensional topological hypersphere in the $(d + 1)$ -dimensional Euclidean space Ω . In other words, $\mathcal{S}$ is the image of an injective continuous mapping of a d -sphere into Ω . The Jordan-Brouwer theorem states that the complement of $\mathcal{S}$ in Ω consists of exactly two connected components [142, 143]. One of these components is bounded (the interior) and the other is unbounded (the exterior). Then the hypersphere $\mathcal{S}$ is their common boundary.

The ray-casting algorithm casts a ray $\mathcal{R}$ from a given point in an arbitrary direction and counts the number of times this ray intersects the simplices of a closed boundary Γ . If the number of intersections is odd, the point lies inside the region enclosed by the boundary, if it is even, the point lies outside. In our implementation of the PFEM-LS framework, Γ corresponds to the former mesh, which has been temporarily stored in memory during remeshing. This boundary is defined by a set of edges in 2D or triangles in 3D.

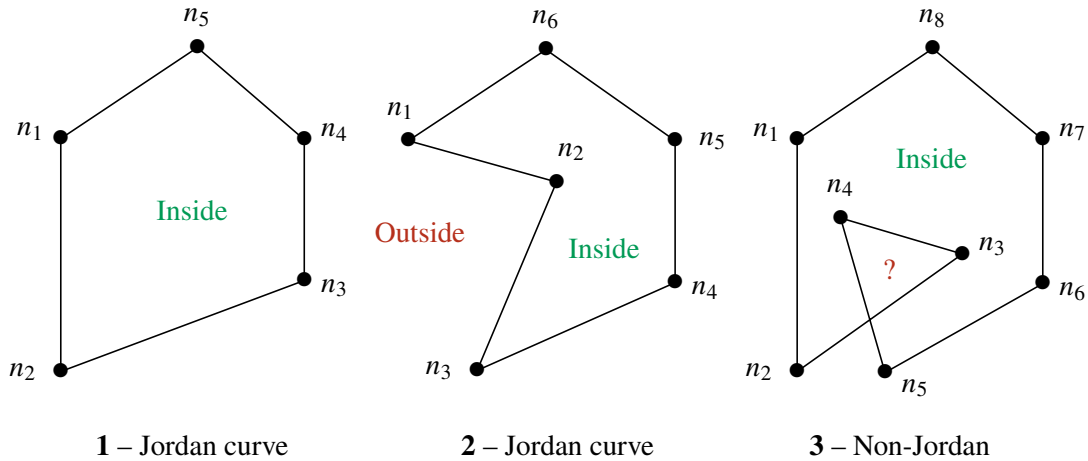


Figure 37: Examples of Jordan and non-Jordan curves formed by multiple segments and nodes in 2D. The third image is a closed loop that does not validate the properties of a Jordan curve because it intersects itself.

To determine whether a simplex lies inside or outside the fluid domain, $\mathcal{R}$ is cast from the centroid of the simplex, as shown in Figure 38. In some cases, the fluid domain may consist of multiple, non-connected subdomains, such as droplets. Ideally, the algorithm would count intersections with each subdomain separately to determine whether a point lies inside any of them. However, in practice, treating all edges or triangles as part of a single Γ does not compromise the ray-casting algorithm, as long as the subdomains do not interpenetrate. Such interpenetration is not expected in PFEM fluid simulations, as subdomains should merge upon contact. However, two ambiguous cases need special treatment:

- In computational geometry, floating-point precision can lead to incorrect results when the query point p lies very close to the boundary Γ . Rounding errors may cause misclassifications due to slight variations in intersection detection. To ensure correctness, the ray-casting algorithm incorporates a numerical tolerance and check whether p lies within a close neighborhood of Γ . If the point falls within this margin, the algorithm reports that p is on the boundary, setting $d_\Gamma = 0$ to avoid ambiguous classifications. It is worth noting that this case is not a large concern in most scenarios, as points located very close to Γ would result in $||d_\Gamma|| \simeq 0$ anyway, automatically satisfying the LS criterion $\mathcal{L}$ in (2.100) regardless of the sign of d_Γ provided ε is not set very close to zero either.

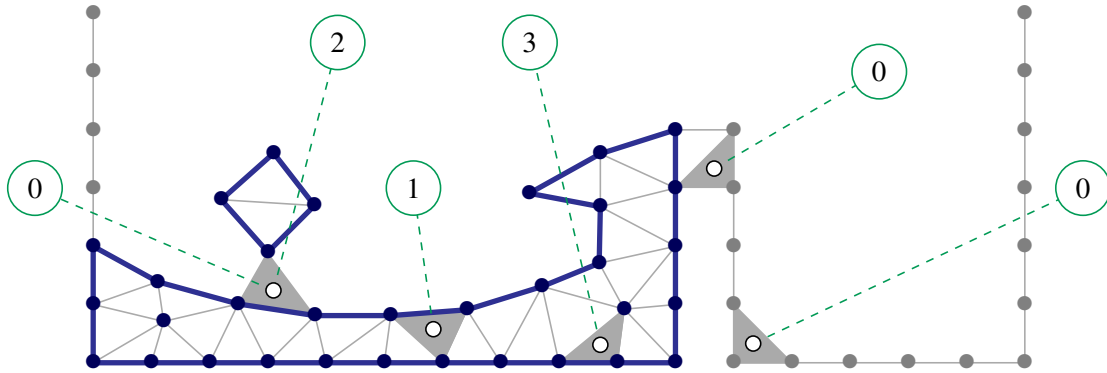


Figure 38: Example of **ray-casting** used to compute the sign of the level set function. The green dashed lines represent ray casts towards arbitrary directions from the centroid of the elements. The annotations indicate the number of intersections between the corresponding ray and the boundary of the fluid, drawn in bold.

- Another pathological case occurs when $\mathcal{R}$ exactly passes through a vertex of the boundary, leading to simplex intersections that might be counted twice erroneously. Similarly, horizontal or coplanar simplices that precisely align with $\mathcal{R}$ can cause miscounts. To address this, one can adopt a systematic rule. If the intersection occurs close to a vertex of Γ (and close to an edge in 3D), another ray $\mathcal{R}_\theta$ is cast with small angular perturbations θ in its direction until a non-ambiguous prediction is obtained.

As a side remark, another technique used to check if a point lies inside a closed surface is to compute the winding number of the query point with respect to the polytope [144, 145]. If the winding number is non-zero, the point lies inside the polytope, otherwise, the point lies outside. Intuitively, the winding number of a closed curve in the plane around a given point represents the total number of times that the curve travels counterclockwise around the point. Mathematically, let assume that one has a piecewise differentiable representation of Γ in 2D, the winding number of Γ around the point p is given by

$$r = \sqrt{(x - x_p)^2 + (y - y_p)^2}$$

$$\text{wind}(p, \Gamma) = \frac{1}{2\pi} \int_{\Gamma} \left(\frac{x - x_p}{r^2} dy - \frac{y - y_p}{r^2} dx \right) \quad (2.101)$$

The advantage of the winding number over the ray-casting is that the former is able to provide an answer for non-simple polytopes, such as the non-Jordan curve displayed in Figure 37. Here, any point located inside the area of self-intersection would be considered as inside the polygon. In 3D, the winding number of a closed oriented surface $\mathcal{S}$ around p is the number of times that the surface encloses p . This measure is generally known as the topological degree [142]. In this work, the boundary of the fluid is essentially simple and the ray-casting already provides an acceptable solution to the problem. Thereby, the approach based on the winding number has not been considered.

Once the sign of d_Γ has been determined using the ray-casting method detailed above, its magnitude must be computed by finding the shortest distance between the centroid of the considered simplex (i.e. the query point p) and the edges or triangles that form the boundary mesh. Since the PFEM-LS involves detecting ray-boundary intersections and calculating distances for numerous centroids and boundary elements, an efficient implementation is essential.

Implementation

Although individual intersection tests can be performed efficiently using established methods [146], a brute-force approach has a computational complexity of $O(nm)$, where n is the number of centroids and m is the number of boundary elements, making it impractical for large-scale simulations. To overcome this limitation, a spatial partitioning strategy is employed to accelerate both ray-casting and distance computation. The boundary elements are organized into an Axis-Aligned Bounding Box (AABB) tree, a hierarchical spatial structure provided by the CGAL library [147]. This tree recursively partitions the space into nested bounding boxes that tightly enclose subsets of boundary elements, allowing to quickly discard irrelevant elements. The AABB tree is commonly built following the top-down construction:

- Given a set $\mathcal{S} = \{s_1, s_2, \dots, s_n\}$ of objects, one first defines a corresponding set of axis-aligned bounding boxes, denoted by $\mathcal{B} = \{b_1, b_2, \dots, b_n\}$. Each bounding box b_j is computed based on the minimum and maximum coordinates of the object it encloses. Then, the bounding box b_0 of the entire set is created, the latter being considered as the root node of the AABB tree.
- During the recursive partitioning, A spatial partitioning heuristic is selected to split $\mathcal{B} = \mathcal{B}_L \cup \mathcal{B}_R$ into two subsets $\mathcal{B}_L$ and $\mathcal{B}_R$. A common example of partitioning heuristic is the median split, which divides the original set of objects by the median of their centroid positions.
- The axis aligned bonding box b_L of the left child set $\mathcal{B}_L$ is computed and associated to a child node of the master node (here, the root node). Similarly, one can compute the box b_R of the right child set $\mathcal{B}_R$. For each child set, the partitioning process is repeated. The recursion stops when a node contains only one object (i.e. is a leaf node), or a predefined depth limit is reached.

An illustration of the AABB structure is provided in Figure 39. During ray-casting, the AABB tree is traversed from the root to the leaves to identify the leaf boxes intersected by the ray. Only the boundary elements contained within these boxes are tested for exact intersections, significantly reducing the number of checks compared to a brute-force approach. Indeed, the time complexity of the brute-force approach for computing the intersection between a single ray and the boundary is

$$t = O(nc) \tag{2.102}$$

Here, n is the number of boundary elements in the mesh, and c is the time complexity for computing the intersection between the ray and the object (i.e. the element). This grows linearly with n and can become expensive even for small n for complex objects. On the other hand, traversing the AABB tree to reach a leaf node has a time complexity of

$$t = O \log_2 (nc_B) \quad (2.103)$$

where c_B is the time complexity of intersecting a ray with an axis-aligned box, which is typically cheap. Note that (2.103) is true only if the partitioning heuristic roughly halves the number of objects at each level of the hierarchy. Once a leaf node has been reached, a single intersection test with the boundary element it contains is computed. For distance computations, the AABB tree is augmented with a KD-tree structure to optimize nearest-neighbor searches. This dual-structure strategy efficiently computes the minimum distance between a centroid and the boundary mesh by restricting the search to a relevant subset of boundary elements [148, 149]. Like the AABB tree, the KD-tree is hierarchical, but instead of bounding boxes, it recursively subdivides space along specific coordinate axes. An illustration of the KD structure is provided in Figure 40.

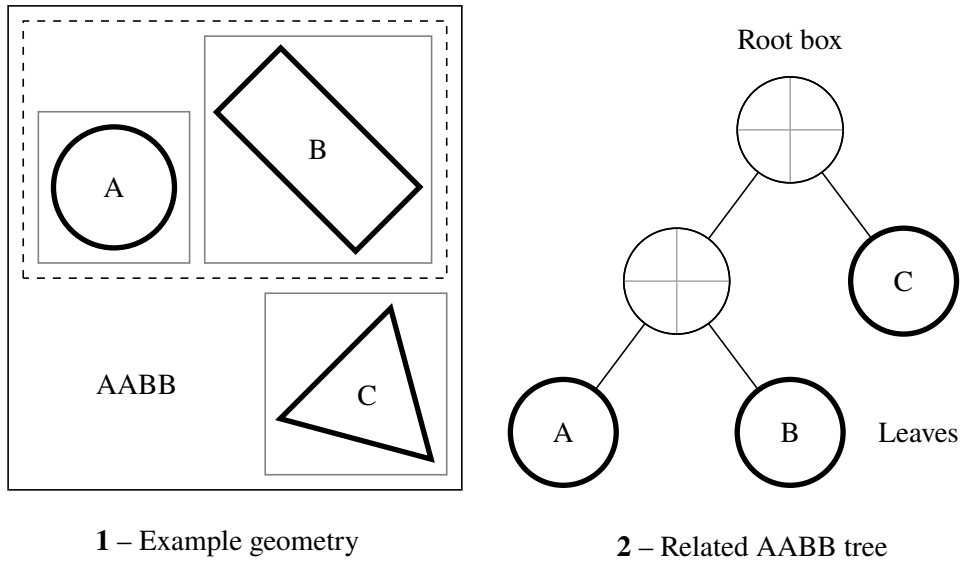
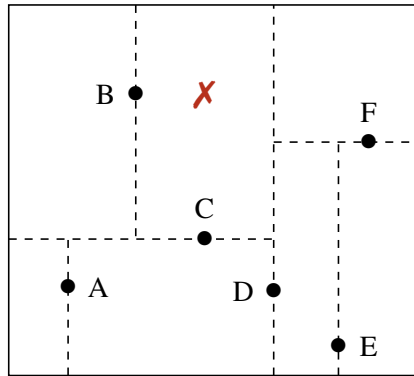
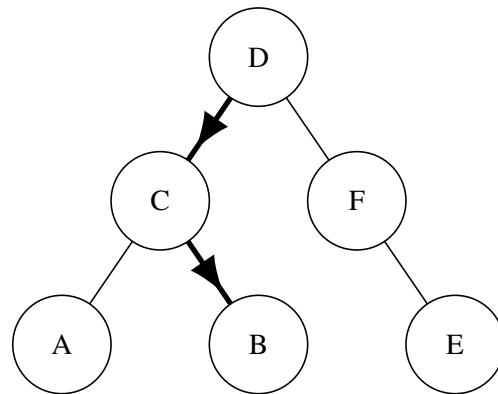


Figure 39: In Example 1, an arbitrary spatial configurations of objects, represented by the geometric entities named A, B and C in the first figure. The example 2 shows a visual representation of the corresponding AABB tree structure.

- Given a set of n points $\mathcal{P} = \{p_1, p_2, \dots, p_n\}$ in a 3D space, we aim to organize them into a binary tree where each node partitions the space. The most common strategy is cycling through dimensions, namely, splitting the space by x at depth 0, by y at depth 1, by z at depth 2, then cycle back.
- First, the points are sorted along each axis, and the median point p_m is selected as the split point to ensure balanced partition of the space $\mathcal{P} = \{p_m\} \cup \mathcal{P}_L \cup \mathcal{P}_R$ by the cutting plane P_m . This splitting point is assigned to a new node of the tree, or to the root node at the first iteration of the recursive splitting. For each subset $\mathcal{P}_L$ and $\mathcal{P}_R$ of the current node, the recursive splitting is repeated until one point remains (i.e. a leaf node), or until a depth threshold is reached.
- During the nearest neighbor search, the KD-tree is traversed from the root node to a leaf. At each tree node, the position $\mathbf{p}$ of the query point with respect to P_m is computed to decide whether to explore the left $\mathcal{P}_L$ or right $\mathcal{P}_R$ subtree. Once a leaf node j is reached, the distance to the query point $d_j = \|\mathbf{p} - \mathbf{x}_j\|$ is computed and recorded as the current smallest distance.
- As the algorithm backtracks, d_j is updated whenever a closer point is found. At each tree node, the algorithm checks whether the sphere $\mathcal{S}(\mathbf{p}, r = d_j)$ centered at the query point, with a radius equal to the current best distance, intersects the splitting plane. If no intersection is detected, the subtree $\mathcal{P}$ on the opposite side is pruned, and the search continues along the remaining branch. The search terminates once backtracking reaches the root node.



1 – Example geometry



2 – Related KD-tree

Figure 40: In Example 1, the KD-tree structure for a cloud of nodes in 2D, where the space is partitioned by axis-aligned planes. The example 2 shows the nearest neighbor search procedure with respect to $\mathbf{X}$ in the tree.

For a single query point, the KD-tree has a significantly lower complexity than the brute force approach. However, unlike the AABB tree, the time complexity of finding the nearest neighbor in the KD-tree heavily depends on the distribution of points in space, leading to best-case and worst-case scenarios in terms of search performance. If points are poorly distributed or all lie along a specific structure, the tree can become imbalanced, increasing traversal time toward $O(n)$. When points are well-distributed, each split reduces the number of remaining elements by roughly half, and the time complexity for finding the nearest neighbor tends to

$$t = O \log_2 (n) \quad (2.104)$$

Side Remark

Setting $\varepsilon = 0$ in (2.100), or using a very small value, causes any simplex generated outside the last-known configuration of the fluid domain to be rejected. As a result, separate fluid regions merge only upon physical contact, which helps to prevent the creation of non-physical mass. While this behavior is desirable for fluid-fluid interactions, it can hinder the formation of connecting elements between the fluid and a dry boundary during remeshing. In such cases, the fluid may pass through the boundary without interaction, allowing a small portion to enter an exclusion zone, a region where all nodes and elements are removed during remeshing. Any fluid entering the exclusion zone is subsequently deleted. Afterward, new connecting elements may form between the remaining fluid and the boundary, as long as their centroids fall within the former fluid domain. This process allows fluid to reestablish contact with an initially dry boundary, even when $\varepsilon = 0$, though it may result in slight mass loss. An example is shown in Figure 41. Whether a point lies inside or outside the exclusion zone is determined using the ray-casting algorithm described earlier.

An alternative approach to enable fluid-wall contact while using a very small ε is to relax the LS criterion $\mathcal{L}$ near the walls by introducing a larger tolerance $\varepsilon_w > \varepsilon$. This relaxed tolerance is applied only to connecting elements between the fluid and the walls. By doing so, these elements can be accepted before the fluid physically crosses the boundary, while their size remains constrained to limit artificial mass creation. Meanwhile, the standard LS criterion based on ε is retained for all other elements, including those connecting two free surfaces, ensuring improved mass conservation.

2.3.2 Numerical Examples

The test case described in this subsection involves a highly viscous fluid droplet falling into a tank filled with the same fluid. This setup is based on the work of Franci and Cremonesi [150] and has then been extended to 3D. The initial configuration of the fluid is illustrated in Figure 42. When the initial fluid level is $H = 7$ cm, the size-field depends on the distance to the free surface, as described by

$$h(\mathbf{x}) = \min \left\{ h_0 + h_\infty \frac{d(\mathbf{x})}{H}, h_\infty \right\} \quad (2.105)$$

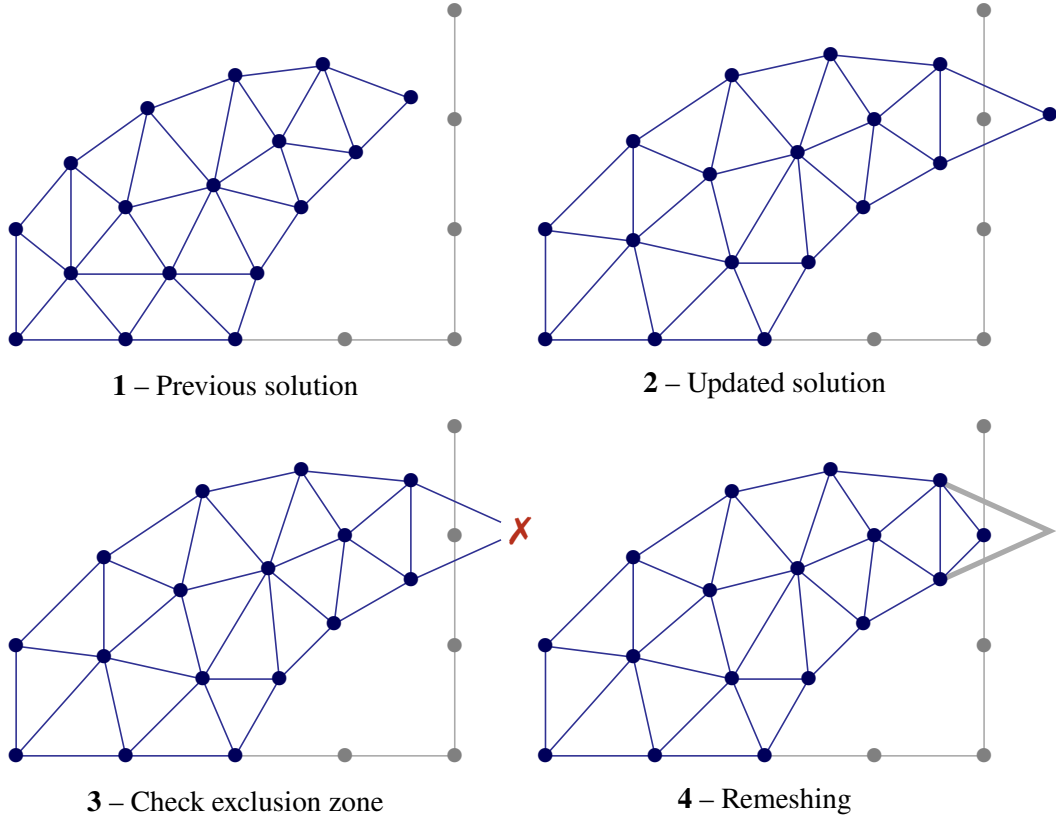


Figure 41: Example of reconnection between a fluid (continuous lines) and a wall (dashed lines) when $\varepsilon \simeq 0$. The nodes that enter into an exclusion zone are deleted, allowing the creation of connecting elements whose centroid is located inside the former mesh.

In 2D, two spatial discretizations are considered: A coarse, uniform mesh with $h_\infty = 0.5$ cm, $h_0 = 0.5$ cm, and a finer mesh with $h_\infty = 0.5$ cm, $h_0 = 0.15$ cm. The total mass variation of the fluid during the simulation is compared to other remeshing methods from the literature, as shown in Figure 43.

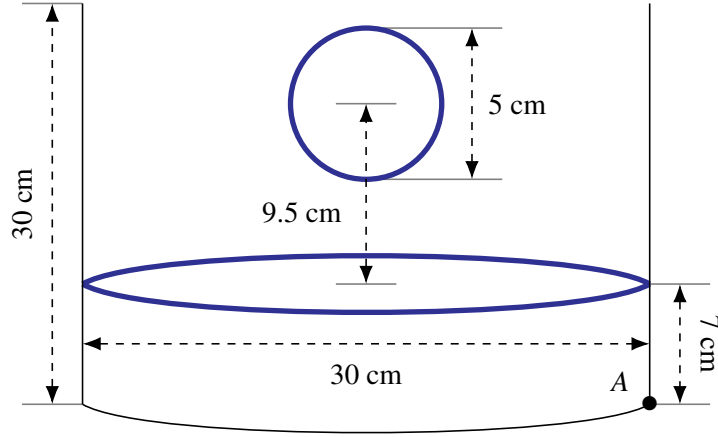


Figure 42: Initial configuration of the **droplet falling** test case whose simulation parameters are indicated in Table 5. No-slip BC are applied to all rigid boundaries and the initial free surface is outlined in bold.

<i>Remesh 2D</i>	<i>Remesh 3D</i>	<i>Fluid</i>
$\delta = 0.6$	$\delta = 0.8$	$\rho = 10^3 \text{ kg/m}^3$
$\omega = 0.6$	$\omega = 0.4$	$g = -9.81 \text{ m/s}^2$
$\alpha = 1.2$	$\alpha = 1.7$	$\mu = 0.1 \text{ Pa s}$
$\beta = 0.2$	$\beta = 0.2$	$\gamma = 0 \text{ N/m}$
$\varepsilon = 0.07$	$\varepsilon = 0.05$	
		$tol = 10^{-8}$
		$\Delta t_{\text{MAX}} = 10^{-4} \text{ s}$

Table 5: Simulation parameters for the **droplet falling** whose initial configuration is given in Figure 42. The governing equations are solved with the PSPG stabilization and different remeshing techniques.

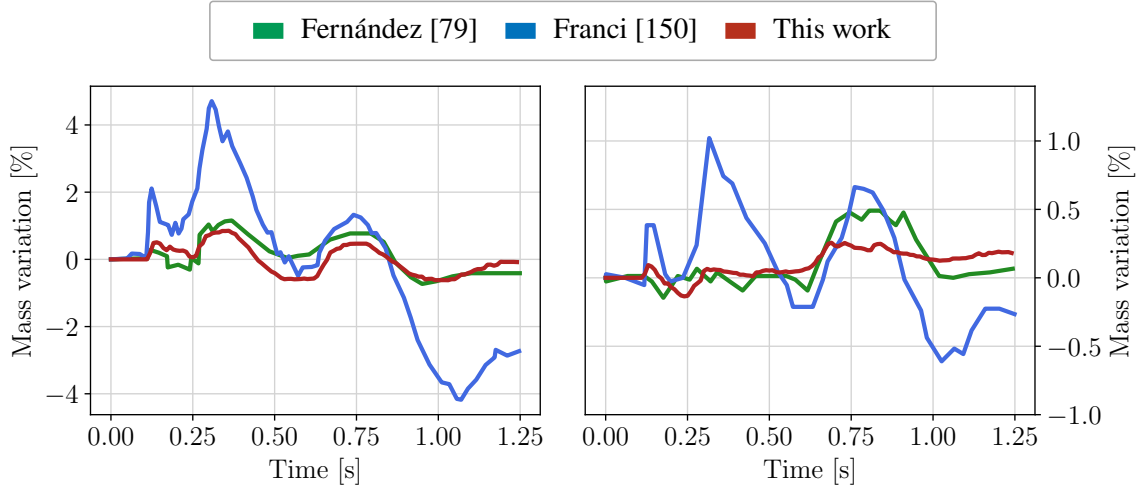


Figure 43: Mass variation of the **droplet falling** test case in 2D, respectively with the coarse and the fine mesh. The results from Fernández et al. have been obtained with an alternative implementation of the PFEM-LS.

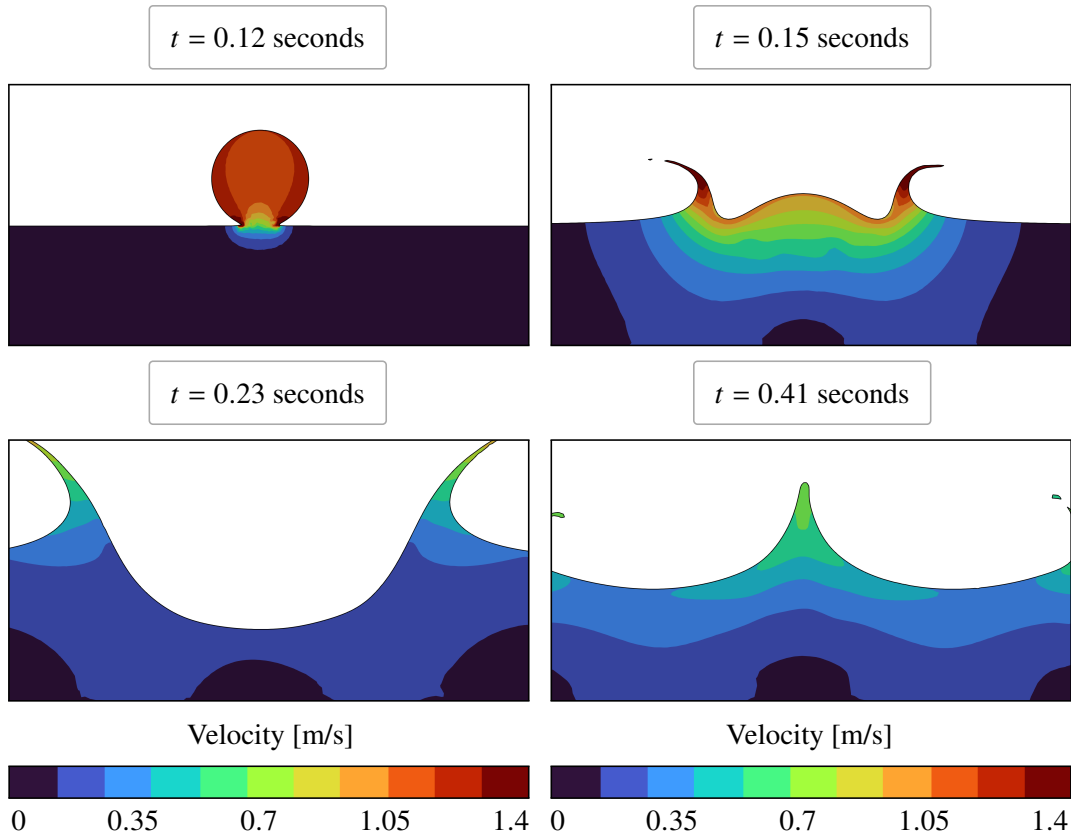


Figure 44: Snapshots of the particle finite element solution for the **droplet falling** test case whose initial configuration is given in Figure 42, at different time intervals. The parameters from Table 5 have been used.

Figure 44 shows snapshots of the solution at various time intervals. In addition, Figure 45 compares the classical α -shape criterion $\mathcal{A}$ with the LS criterion $\mathcal{L}$ during fluid-fluid contact. The figure clearly illustrates the formation of spurious connecting elements when the LS is disabled, which leads to significant mass creation as the two fluid surfaces merge. In contrast, using $\mathcal{L}$ delays the formation of connecting elements until the droplet actually reaches the bottom fluid surface, provided that ε is chosen sufficiently small. The results demonstrate that the PFEM-LS achieves improved mass conservation.

In 3D, the maximum and minimum allowed characteristic sizes are respectively set to $h_\infty = 0.5$ cm and $h_0 = 0.5$ cm, resulting in a uniform size-field. This size-field is designed to provide sufficient accuracy for the solution, although it is coarser than the mesh used by Avancini et al. [151], who employed a uniform characteristic size of $h = 0.3$ cm. The fluid's pressure at the bottom of the tank, along with the total mass variation over time, are shown in Figure 46. Snapshots of the solution are presented in Figure 47 at different time steps. Finally, Figure 48 compares the classical α -shape criterion with the LS criterion. The results confirm the findings from the 2D case, demonstrating that $\mathcal{L}$ helps to mitigate spurious mass creation.

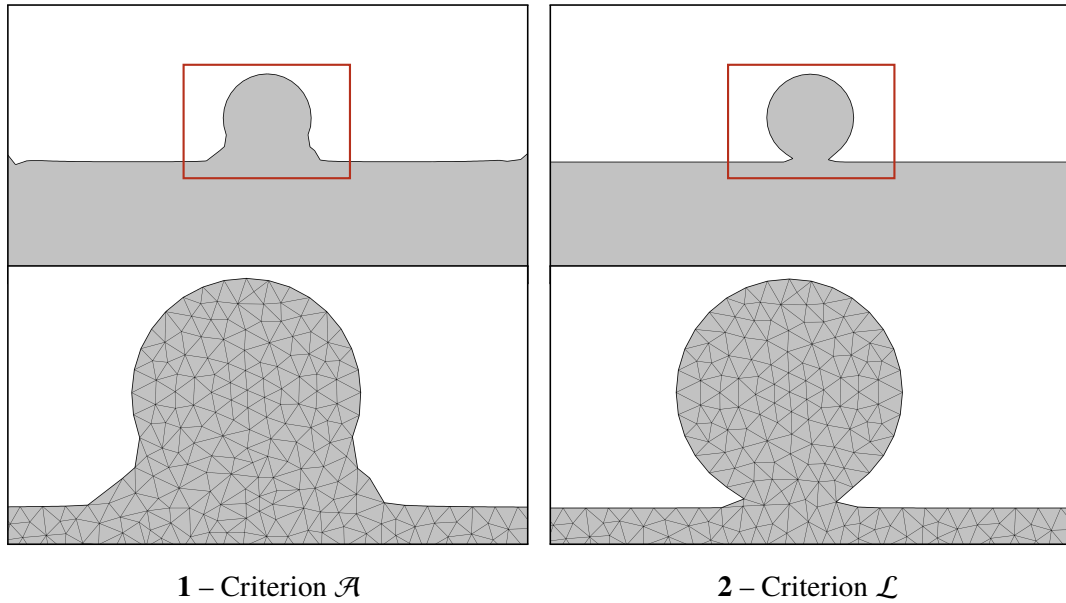


Figure 45: Outline of the free surface for the **droplet falling** test case whose initial configuration is given in Figure 42, at time $t = 0.12$ seconds. A zoom on the region outlined by the red rectangle is also presented below.

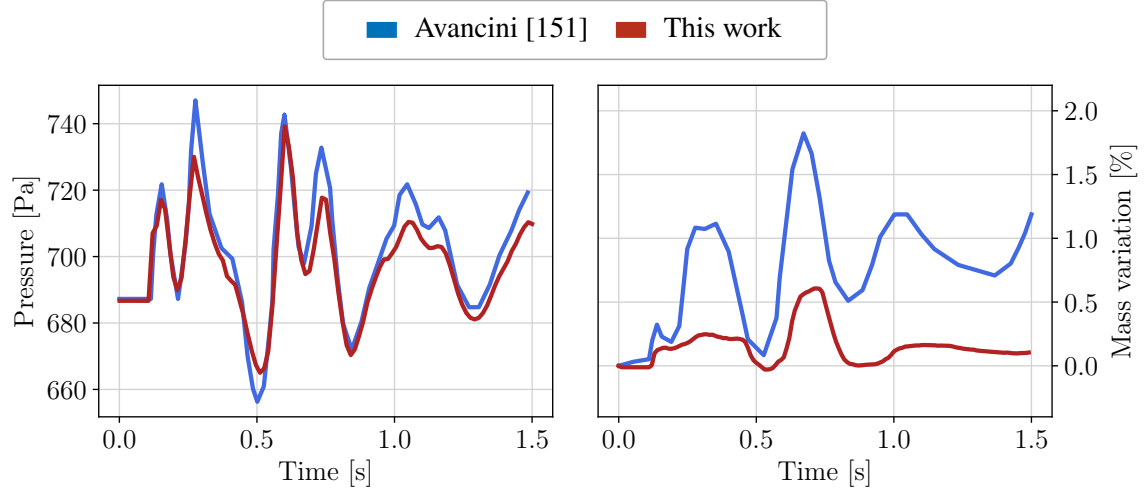


Figure 46: Evolution of the fluid pressure at the point A indicated in Figure 42 and total mass variation of the **droplet falling** test case in 3D. Numerical results from Avancini et al. are also displayed for comparison.

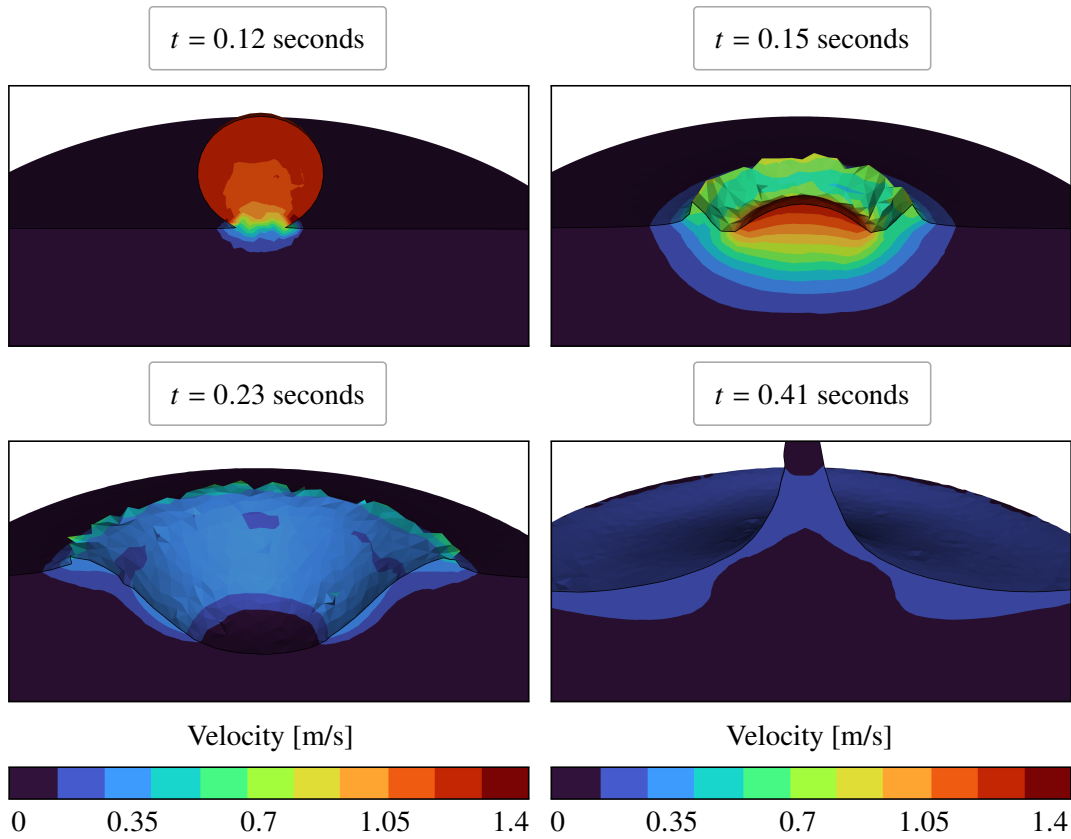


Figure 47: Snapshots of the particle finite element solution for the **droplet falling** test case whose initial configuration is given in Figure 42, at different time intervals. The parameters from Table 5 have been used.

It is important to note that this test case involves a highly viscous fluid, which naturally limits splashing and erratic free surface deformations, thereby reducing mass creation. Consequently, the mass variation observed both in the literature and in our results remains very small. Figure 49 shows the evolution of total fluid mass in the dam break scenario introduced earlier. Unlike the previous case, the dam break features violent free surface behavior, including breaking waves, splashing, and aeration, which pose greater challenges for mass conservation.

Although the $\mathcal{L}$ criterion allows keeping the mass variation close to zero when appropriately tuned, Figure 49 also reveals a limitation of the PFEM-LS, as the results are highly sensitive to the choice of ε . This parameter lacks a known optimal value and may need to be adjusted based on the specific physical conditions. The main drawback lies in the fact that, although the PFEM-LS approach successfully limits non-physical mass creation, it does not prevent non-physical mass loss caused by fluid separation. As a result, choosing an ε that is too small can lead to a gradual fluid loss over time, as the balance between mass creation and loss is disrupted.

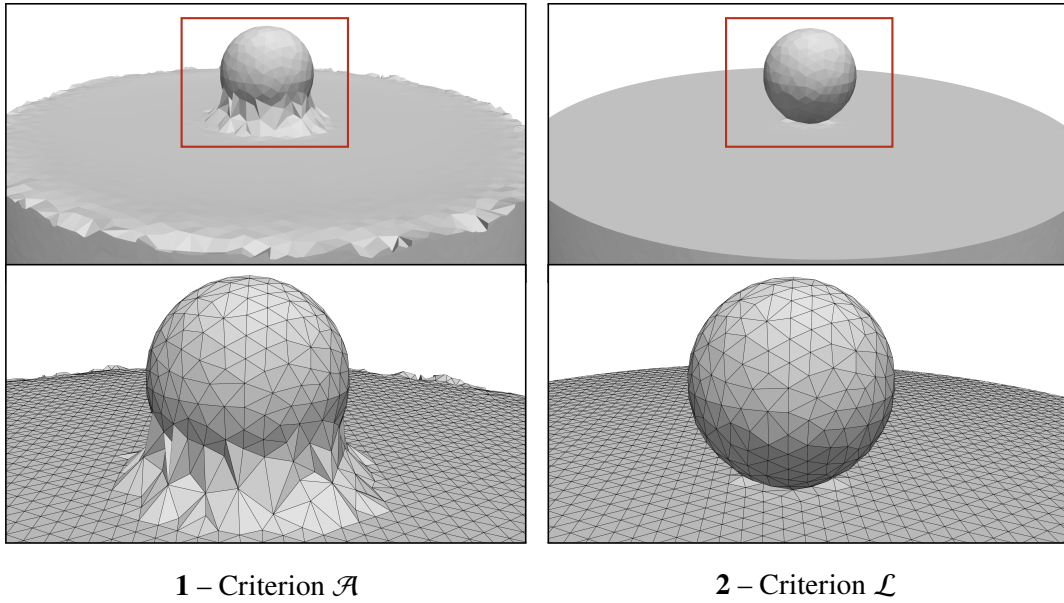


Figure 48: Outline of the free surface for the **droplet falling** test case whose initial configuration is given in Figure 42, at time $t = 0.12$ seconds. A zoom on the region outlined by the red rectangle is also presented below.

An interesting approach to mitigate this issue is the dynamic, automatic adjustment of ε based on observed mass variations, aiming to balance non-physical mass creation and loss. While this method has shown success in certain cases, it remains difficult to apply in scenarios involving unpredictable physical mass changes, such as those with inlets, outlets, or when the fluid naturally reaches the end of the bounding box. Additionally, this strategy does not fully resolve the root cause of non-physical mass variation, as it merely attempts to balance two undesirable effects rather than eliminate them altogether.

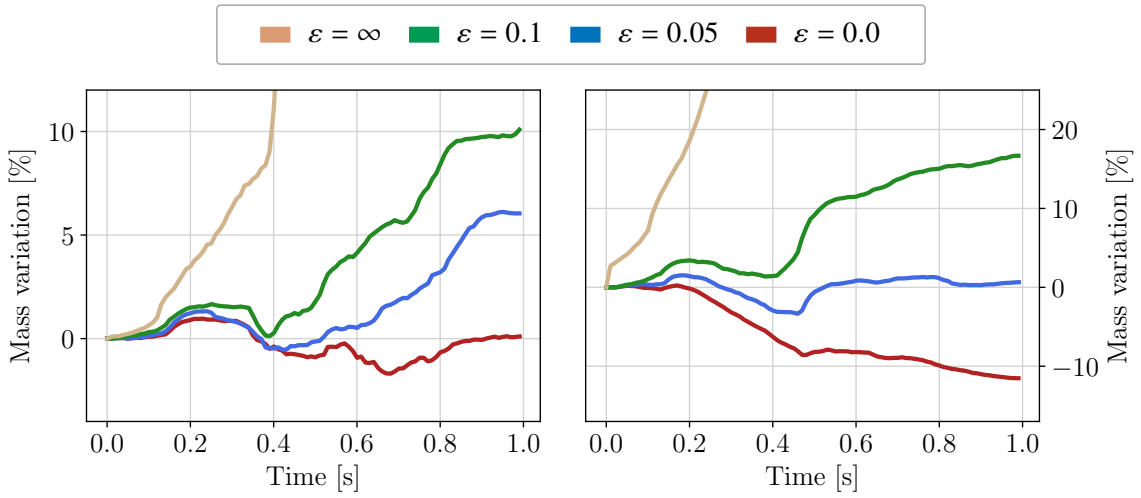


Figure 49: Mass variation of the **dam break** in 2D and 3D using the PSPG formulation, the edge splitting remeshing with $\mathcal{L}$ and different values of ε . Taking $\varepsilon = \infty$ is equivalent to using the classical α -shape without level set.

Chapter 3

Chapter

Fluid-Structure Interactions

3.1 Introduction

This chapter focuses on the simulation of Fluid-Structure Interaction (FSI) problems using a partitioned coupling strategy. The fluid is modeled using the PFEM-LS solver described in the previous chapter, while the solid is modeled with a FEM solver. Each solver operates separately on its respective domain, and the coupling algorithm ensures that the fluid's forces act on the structure, while the structural deformation influences the fluid boundary conditions. This approach is highly flexible, as it relies on specialized solvers for each domain, which can be independently optimized or modified. Historically, staggered methods employ black-box solvers, which operate without exposing their internal algorithms. As a result, the coupling algorithm interacts solely with the input and output data of these solvers [7, 21, 152]. While we aim to preserve this non-intrusive principle, the solvers used in this study are in-house codes [28, 100], which allow for greater control and potential modifications to the coupling process as needed. This adaptability provides the flexibility to refine and optimize the interaction between the fluid and structural solvers. The aim of this chapter is to present a comprehensive overview of the FSI framework, detailing both the implementation of the coupling strategy and the challenges encountered. Specifically, we address objectives 4, 5, and 6, as outlined in the introduction.

3.1.1 The Structural Solver

The structural solver used in this work is Metafor, a FE code based on an updated Lagrangian formulation, designed to simulate solids undergoing large deformations [27, 28]. While Metafor is primarily designed for metal forming, it also provides a wide range of constitutive models, including those for hyperelastic and biological materials. The solver is capable of modeling complex phenomena such as metal composite materials, damage and rupture, frictional contact between deformable bodies or CAD rigid surfaces, plasticity, and thermo-mechanical effects. Unlike commercial solvers, Metafor is research-oriented, emphasizing flexibility and customization to meet the needs of academic and experimental applications. It is important to note that this subsection does not introduce new derivations developed as part of this thesis. Rather, it provides a concise overview of the governing equations and time integration schemes employed by the structural solver, which are consistently applied across all the test cases presented later. Readers already familiar with these concepts may skip this subsection and proceed to the next one, where the architecture of the Fluid-Structure Partitioned Coupler (FSPC) is detailed.

Conservation of Momentum

In all generality, the deformation of a structure is governed by a Partial Differential Equation (PDE) expressing the conservation of momentum in a continuum. Let Ω represent the current configuration of the solid domain, and $\boldsymbol{\sigma}$ the Cauchy stress tensor. The momentum conservation equations are given by:

$$\nabla \cdot \boldsymbol{\sigma} + \rho \mathbf{g} = \rho \frac{d^2 \mathbf{u}}{dt^2} \quad \text{in } \Omega \quad (3.1)$$

Here, ρ denotes the solid's density, $\mathbf{g}$ the gravitational acceleration, and $\mathbf{u}$ the displacement field. These equations are supplemented with appropriate initial and boundary conditions. To complete the system, both kinematic and constitutive laws are required. The kinematic relationship is typically nonlinear to capture finite strains and large displacements, ensuring an accurate description of solid behavior under large deformation. For general material behavior that includes plasticity, the constitutive law can be expressed using a hypoelastic formulation, in which the current stress state depends on the material deformation history:

$$\overset{\nabla}{\boldsymbol{\sigma}} = \mathbb{H} : (\mathbf{D} - \mathbf{D}_{vp}) \quad (3.2)$$

Here, the superscript ∇ denotes the Jaumann (objective) derivative of the stress tensor, $\mathbb{H}$ is the Hooke tensor, $\mathbf{D}$ is the strain rate tensor and $\mathbf{D}_{vp}$ is the plastic strain rate. On the other hand, the class of hyperelastic materials is a special class of constitutive model for which the stress-strain relationship derives from a strain energy density function. Mathematically, the stress tensor of a hyperelastic model writes

$$\boldsymbol{\sigma} = \frac{\partial \Psi}{\partial \boldsymbol{\varepsilon}} \quad (3.3)$$

Here, Ψ is the strain-energy density function and $\boldsymbol{\varepsilon}$ is an associated strain tensor. While a detailed discussion of elastoplasticity and hyperelasticity models lies beyond the scope of this thesis, the reader is referred to [28, 153] for a comprehensive treatment of the subject. The semi-discretized finite element formulation of Equation (3.1) can be expressed in all generality by

$$\mathbf{M}\mathbf{a} + \underbrace{\int_{\Omega} \mathbf{B}_3 \hat{\boldsymbol{\sigma}} \, dx}_{\boldsymbol{\Sigma}(\mathbf{u})} = \mathbf{f} \quad (3.4)$$

Here, $\mathbf{u}$ is the vector of nodal displacements, $\mathbf{a}$ the vector of nodal accelerations, $\mathbf{M}$ the mass matrix, $\boldsymbol{\Sigma}$ the internal force vector which depends on the material model, and $\mathbf{f}$ is the external force vector. $\mathbf{B}_3$ is the gradient of the basis functions associated with the Voigt notation $\hat{\boldsymbol{\sigma}}$ of the stress tensor, as defined in the previous chapter. Time integration is carried out using the α -generalized family of methods. In this approach, the equilibrium equation (3.4) is reformulated by weighting the inertial forces at time t_n and t_{n+1} with the parameter α_M , and the internal and external forces with the parameter α_F such that

$$\begin{aligned} (1 - \alpha_M)\mathbf{M}\mathbf{a}_{n+1} + \alpha_M\mathbf{M}\mathbf{a}_n + (1 - \alpha_F)\boldsymbol{\Sigma}_{n+1} \\ + \alpha_F\boldsymbol{\Sigma}_n = (1 - \alpha_F)\mathbf{f}_{n+1} + \alpha_F\mathbf{f}_n \end{aligned} \quad (3.5)$$

The specific values of the weighting parameters lead to the Newmark scheme for $\alpha_M = \alpha_F = 0$, to the Hilber-Hughes-Taylor scheme [154] for $\alpha_M = 0$, to the Wood-Bossak-Zienkiewicz scheme [155] for $\alpha_F = 0$, or to the Chung-Hulbert scheme [156] for $\alpha_M \neq 0$ and $\alpha_F \neq 0$. The relationships between displacements, velocities and accelerations are given by the Newmark equations:

$$\begin{aligned} \mathbf{u}_{n+1} &= \mathbf{u}_n + \Delta t \, \mathbf{v}_n + \Delta t^2 \left[(1/2 - \beta)\mathbf{a}_n + \beta\mathbf{a}_{n+1} \right] \\ \mathbf{v}_{n+1} &= \mathbf{v}_n + \Delta t \left[(1 - \gamma)\mathbf{a}_n + \gamma\mathbf{a}_{n+1} \right] \end{aligned} \quad (3.6)$$

Here, β and γ are constant parameters of the Newmark scheme, and Δt is the current time step. The system of equations (3.5) is nonlinear with respect to $\mathbf{u}$ because both the internal forces and the integration domain depend on the nodal displacements. As a result, system is solved using a Newton-Raphson (NR) algorithm. At each time step, the following system is iteratively solved until convergence:

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \mathbf{J}_k \backslash \mathbf{r}_k \quad (3.7)$$

In this context, the backslash notation $\backslash$ denotes the numerical solution of a linear system. The subscript k indicates that the corresponding matrix or vector is evaluated using the nodal positions at the k -th iteration of the Newton-Raphson algorithm. The residual vector $\mathbf{r}_k$ represents the difference between the left-hand side and right-hand side of Equation (3.5) evaluated at the current solution estimate:

$$\begin{aligned} \mathbf{r}_k = & (1 - \alpha_M)\mathbf{M}\mathbf{a}_k + \alpha_M\mathbf{M}\mathbf{a}_n \\ & + (1 - \alpha_F)(\boldsymbol{\Sigma}_k - \mathbf{f}_k) + \alpha_F(\boldsymbol{\Sigma}_n - \mathbf{f}_n) \end{aligned} \quad (3.8)$$

The matrix $\mathbf{J}$ is the Jacobian, which can be obtained either analytically by deriving its closed-form expression, or numerically via perturbation methods. Whenever available, the analytical Jacobian is preferred for improved accuracy and efficiency. In the Chung-Hulbert scheme, the Jacobian is given by

$$\mathbf{J}_k = (1 - \alpha_F)\frac{\partial(\boldsymbol{\Sigma}_k - \mathbf{f}_k)}{\partial\mathbf{u}_k} + (1 - \alpha_M)\frac{1}{\beta\Delta t^2}\mathbf{M} \quad (3.9)$$

This predictor-corrector scheme must be applied iteratively until a zero residual is obtained within some user-defined tolerance. In Metafor, the stopping criterion for the NR process is a non-dimensional scalar expression of the residual, normalized by the external forces. The NR algorithm terminates when

$$\varepsilon_M^k = \frac{\|\mathbf{r}_k\|}{\|\mathbf{f}_k\|} < tol \quad (3.10)$$

or when the maximum number of iterations is reached. In the second case, the current time step is restarted with a reduced time increment $\Delta t' = \Delta t/2$, and this process continues iteratively until (3.10) is satisfied. If convergence is achieved with $\Delta t' < \Delta t$, the time increment is progressively increased to restore the original time step size. This adaptive time-stepping strategy enhances the stability of the scheme when complex or rapid physical phenomena occur, while also allowing a return to larger time steps when possible to optimize computational efficiency.

To simplify user input in the time integration scheme, α_M and α_F can be uniquely determined from a single parameter ρ_∞ known as the spectral radius at infinite frequency. In Metafor, the default value is $\rho_\infty = 0.01$ because it allows for damping high frequencies, which are associated to numerical noise, without affecting low frequencies. This spectral radius is related to the other parameters by

$$\begin{aligned}\alpha_M &= \frac{2\rho_\infty - 1}{\rho_\infty + 1} & \beta &= \frac{1}{(\rho_\infty + 1)^2} \\ \alpha_F &= \frac{\rho_\infty}{\rho_\infty + 1} & \gamma &= \frac{2}{\rho_\infty + 1} - \frac{1}{2}\end{aligned}\tag{3.11}$$

Conservation of Energy

In thermo-mechanical simulations, Metafor uses a loosely coupled framework, solving mechanical and thermal equilibrium sequentially at each time step. The mechanical equilibrium is computed first, using the temperature field from the previous equilibrium state. Then, the thermal equilibrium is updated based on the newly calculated mechanical state by solving the energy conservation equation:

$$\nabla \cdot \mathbf{q} + \rho c_p \frac{dT}{dt} = s\tag{3.12}$$

Here, T denotes the temperature field, c_p is the specific heat capacity at constant pressure, $\mathbf{q}$ is the heat flux, and s is a volumetric heat source. These sources may originate from external thermal loads, friction, or plastic deformation. Since the spatial discretization of Equation (3.12) was presented in the previous chapter, only the final form is recalled:

$$\mathbf{R} \frac{d\mathbf{T}}{dt} + \mathbf{S}\mathbf{T} = \mathbf{b}\tag{3.13}$$

Time discretization of the energy conservation equation in Metafor is carried out using a generalized trapezoidal scheme, controlled by the parameter ϑ . In this work, we adopt the default value $\vartheta = 1$ which corresponds to the Backward-Euler scheme. As outlined in the previous chapter, the Jacobian matrix and residual for the thermal system at the l -th Newton-Raphson iteration are given by

$$\begin{aligned}\mathbf{J}_l &= \frac{1}{\Delta t} \mathbf{S}_l + \mathbf{L}_l \\ \mathbf{r}_l &= \left(\mathbf{S}_l + \frac{1}{\Delta t} \mathbf{R}_l \right) \mathbf{T}_l - \frac{1}{\Delta t} \mathbf{R}_l \mathbf{T}_n - \mathbf{b}_l\end{aligned}\tag{3.14}$$

Here, the subscript l is used to distinguish NR iterations for the thermal system from those of the mechanical system. The NR algorithm terminates when the residual falls below a specified tolerance or when the maximum number of iterations is reached. In the latter case, the time step is restarted with a reduced time increment. The stopping criterion is identical to that used in the fluid solver:

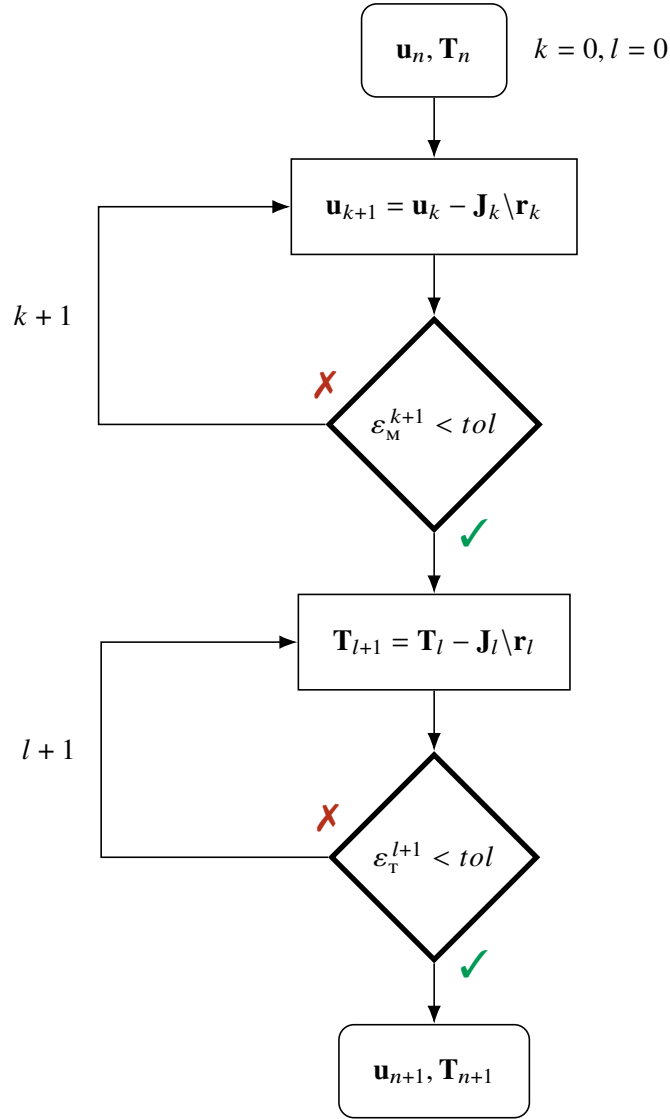
$$\varepsilon_{\text{T}}^l = \frac{\|\mathbf{r}_l\|}{\|\mathbf{b}_l\|} < tol\tag{3.15}$$

Recall that Metafor employs a loose coupling strategy, meaning it does not enforce the simultaneous validation of $\varepsilon_{\text{M}} < tol$ and $\varepsilon_{\text{T}} < tol$ within a single time step. Although the two conservation equations are not resolved simultaneously within the same time step, they are implicitly coupled over successive time steps. Specifically, the mechanical equilibrium at the start of each time step depends on the thermal state computed at the end of the previous one, ensuring temporal consistency between the two fields throughout the simulation. A summary of this coupling strategy is provided in Algorithm 3. This loosely coupled approach contrasts with the fluid solver, which uses a fully coupled iterative scheme within each time step to enforce thermo-mechanical equilibrium.

Contact Interaction

Contact interactions between two surfaces are handled using a penalty method [157, 158]. This approach introduces an external force vector on the right-hand side of the FE system, applied to the slave nodes in contact with the master surface to prevent normal penetration. Additionally, a tangential force is applied at the slave nodes to account for frictional contact. Let d_{Γ} denote the signed Euclidean distance between the penetrating (slave) node and the master surface, as shown in Figure 50. If $d_{\Gamma} > 0$, no contact occurs between the slave and master. If $d_{\Gamma} < 0$, the slave node is considered in contact. Sticking contact is achieved by applying following forces on the node:

$$\begin{aligned}\mathbf{F}_{\text{N}} &= -k_{\text{N}} A d_{\Gamma} \mathbf{n} \\ \mathbf{F}_{\text{T}} &= -k_{\text{T}} A (\delta_1 \mathbf{m}_1 + \delta_2 \mathbf{m}_2)\end{aligned}\tag{3.16}$$



Algorithm 3: Solution strategy for solving thermo-mechanical problems with Metafor, between time t_n and time t_{n+1} . In order to simplify the analyses that follow, the same tolerance value is used for both systems of equations.

Here, k_N and k_T are user-defined penalty coefficients in the normal and tangential directions, respectively. The vector $\mathbf{n}$ is the unit outward normal to the master surface, while $\mathbf{m}_1$ and $\mathbf{m}_2$ are the tangential unit vectors in 3D. The quantities δ_1 and δ_2 represent the projected tangential slips along $\mathbf{m}_1$ and $\mathbf{m}_2$ respectively, and $A = A_1 + A_2$ denotes the equivalent contact area. A detailed description of the contact algorithm is provided by Wautelet and Ponthot in [159]. Sticking contact refers to a state in which no relative sliding occurs between the contacting bodies. For frictionless contact, only the normal force $\mathbf{F}_N$ is included in the system. In this work, frictional contact between solid bodies is governed by Coulomb law of friction, which involves regularizing the tangential force such that

$$\|\mathbf{F}_T\| \leq \mu_c \|\mathbf{F}_N\| \quad (3.17)$$

Here, the parameter μ_c is the coefficient of friction, and $\|\mathbf{F}_N\|$ is the amplitude of the normal force exerted by each surface on the other. The amplitude of the frictional force can range between zero and $\mu_c \|\mathbf{F}_N\|$ to prevent motion between the two surfaces until the maximum value is reached, beyond which sliding will begin. In all simulations featuring frictional contact, the normal penalty coefficient k_N is set by the user, and the tangential penalty coefficient is set to

$$k_T = \mu_c k_N \quad (3.18)$$

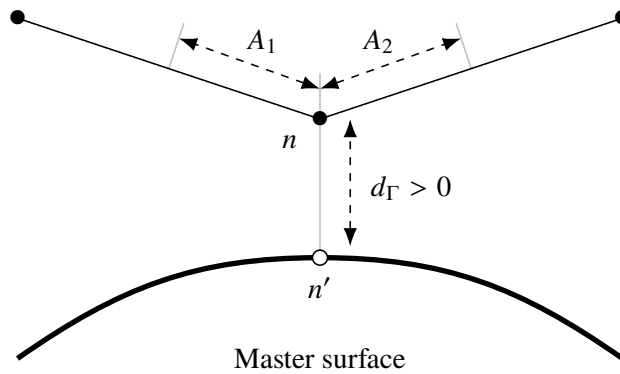


Figure 50: Example of signed distance between a slave node n and its projection n' onto the master surface for contact interaction. Here, the signed distance d_Γ is positive, meaning that the two surfaces are not in contact.

3.1.2 Code Architecture

The architecture of FSPC developed in this work shares many conceptual similarities with the framework created by Thomas and Cerquaglia [29, 31, 89] for their research, but differs notably in its implementation. Specifically, FSPC is designed to couple Metafor with the fluid solver PFEM3D, introduced in the previous chapter. Since neither Metafor nor PFEM3D uses Message Parsing Interfaces (MPI) in their internal operations, FSPC employs MPI to isolate each solver within separate environments and processors. This approach enhances control over computational resource allocation while facilitating efficient communication between their respective interfaces.

FSPC is designed to operate on multicore processors or when using two separate processors, with each solver running on a dedicated machine. Optimized performances are achieved by leveraging Python vectorization to boost computational efficiency. A more detailed analysis of the numerical performance will be provided in a subsequent section. It is worth noting that FSPC is not designed for use with processor clusters, as the fluid and structural solvers do not support such configurations. However, the architecture is flexible and could be extended to work with MPI-based solvers in future works. A scheme of the architecture is illustrated in Figure 51 and consists of three main components:

- Both the fluid and solid solvers are implemented in C++ and compiled using the Simplified Wrapper and Interface Generator (SWIG) [160], allowing direct access to their optimized C++ routines from Python scripts. The wrappers $\mathcal{S}$ and $\mathcal{F}$ shown in Figure 51 are Python classes that facilitate communication between FSPC and the solvers. Specifically, they handle tasks such as imposing boundary conditions, triggering remeshing, loading solutions, and advancing time steps. Each wrapper, along with its associated solver, operates within separate processes and does not communicate with the other one directly.
- The **surface interpolator** class manages the transfer of nodal quantities between the fluid and solid wrappers. For instance, when PFEM3D computes nodal loads at the FS interface, the wrapper $\mathcal{F}$ extracts these loads and passes them to the interpolator. The interpolator then applies an interpolation function, if necessary, to map the nodal loads from the fluid mesh to the solid mesh and sends the result to the wrapper $\mathcal{S}$, which imposes it as a boundary condition in Metafor. Additional details on the interpolation process and handling of non-matching meshes are provided in a subsequent section.

- The **coupling algorithm** class manages time-stepping loops and implements iterative solvers to enforce fluid-structure compatibility at each time step. In practice, this class communicates directly with the solver wrappers, and operates mostly on the computed and predicted solid displacement or temperature provided by $\mathcal{S}$. As a result, most computations within this class are performed on the same processor as the structural solver to maximize efficiency. Further details on the coupling algorithms are provided in the next section.

3.2 Partitioned Strategy

Let $\mathbf{t}_s$ denote the surface tractions applied to the solid body at the FS interface, defined as the product of the stress tensor at the interface and the outward unit normal vector $\mathbf{n}_s$ of the solid domain. Let $\mathbf{u}_s$ represent the displacement at the solid's FS boundary. Similarly, $\mathbf{t}_f$ and $\mathbf{u}_f$ denote the surface tractions and displacements of the fluid at the FS interface. The coupling conditions enforce continuity of both displacement and surface tractions across the interface, based on the action-reaction principle, ensuring consistent interaction between the fluid and solid domains.

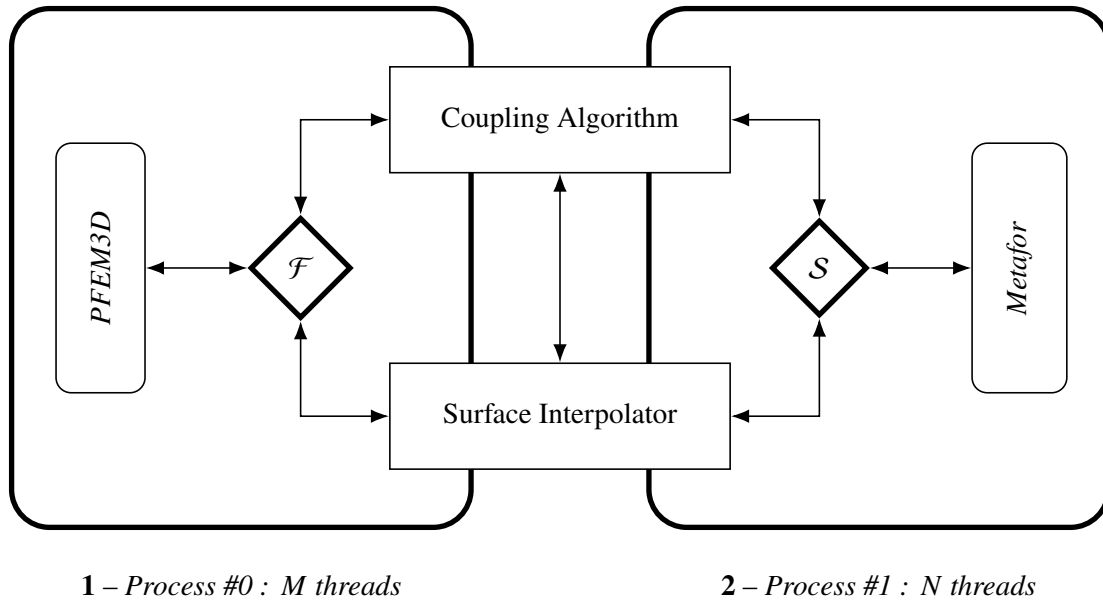


Figure 51: Schematic representation of the architecture of FSPC. Each block represents a major portion of code and the arrows indicate communication between those blocks. The region outlined in bold reside on separate processes.

$$\begin{aligned} \mathbf{u}_S &= \mathbf{u}_F \\ \mathbf{t}_S &= -\mathbf{t}_F \end{aligned} \tag{3.19}$$

The negative sign in the second condition arises because the unit normals $\mathbf{n}_S$ and $\mathbf{n}_F$ point in opposite directions at the interface. In the context of partitioned coupling with independent solvers, the coupling condition (3.19) is reformulated as a **Dirichlet-Neumann** (DN) partitioning. In the DN approach, the nodal displacements at the interface, obtained from the structural solver, are provided to the fluid solver as Dirichlet boundary conditions. The fluid solver then computes the load at the interface using these boundary conditions, which are passed to the structural solver. The structural solver then applies those fluid loads as Neumann boundary conditions to update the interface displacement [74, 161].

It is worth mentioning that the reverse procedure is known as the **Neumann-Dirichlet** (ND) coupling scheme. Here, the fluid solver imposes the nodal displacement as Dirichlet boundary conditions on the solid interface, while the structural solver applies the load as Neumann boundary conditions on the fluid interface. This scheme is rarely used for mechanical coupling due to stability issues, particularly when the fluid density is comparable to the solid density [71]. As a result, the DN coupling scheme is preferred. In order to represent the solid side of the DN scheme, we define the Neumann operator S such that

$$\mathbf{u}_S = S(\mathbf{t}_S) \tag{3.20}$$

This nonlinear operator may be seen as a black-box abstraction of a generic, independent structural solver (which in this case holds for Metafor), which computes the deformation of the structure with respect to a given load applied on its interface. Similarly, we define the Dirichlet operator $\mathcal{F}$ associated with the fluid solver. This operator returns the fluid loads at the interface based on the Dirichlet condition of displacement applied to its interface:

$$\mathbf{t}_F = \mathcal{F}(\mathbf{u}_F) \tag{3.21}$$

The coupling is then formulated as a root-finding problem by combining (3.20) and (3.21) with the continuity conditions (3.19). In the following developments, the indices S and F associated to $\mathbf{u}$ and $\mathbf{t}$ will be omitted for simplicity, as the continuity condition imposes a direct correspondence between the applied loads on both sides of the FS interface, and between the interface displacement:

$$\begin{aligned}\mathbf{r}_F &= \mathcal{F}(\mathbf{u}) - \mathbf{t} = \mathbf{0} \\ \mathbf{r}_S &= \mathcal{S}(\mathbf{t}) - \mathbf{u} = \mathbf{0}\end{aligned}\tag{3.22}$$

Here, $\mathbf{r}_F$ is the residual load for the fluid solver and $\mathbf{r}_S$ is the residual displacement for the solid solver. This coupled system can be solved at a given time following a standard Newton-Raphson iterative procedure by computing the system's Jacobian and solving for an update in the Newton direction:

$$\begin{pmatrix} \mathbf{u}_{k+1} \\ \mathbf{t}_{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{u}_k \\ \mathbf{t}_k \end{pmatrix} - \mathbf{J}_k^{-1} \begin{pmatrix} \mathbf{r}_F^k \\ \mathbf{r}_S^k \end{pmatrix}\tag{3.23}$$

Here, the index k indicates that the quantity is evaluated at the k -th iteration of the NR algorithm. Recalling the expressions of the residuals in Equation (3.22), the final form of the Jacobian derived by Bogaers et al. [162] reads

$$\mathbf{J} = \begin{pmatrix} \frac{\partial \mathbf{r}_F}{\partial \mathbf{u}} & \frac{\partial \mathbf{r}_F}{\partial \mathbf{t}} \\ \frac{\partial \mathbf{r}_S}{\partial \mathbf{u}} & \frac{\partial \mathbf{r}_S}{\partial \mathbf{t}} \end{pmatrix} = \begin{pmatrix} \frac{\partial \mathcal{F}}{\partial \mathbf{u}} & \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{t}} - \mathbf{I} \\ \frac{\partial \mathcal{S}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{u}} - \mathbf{I} & \frac{\partial \mathcal{S}}{\partial \mathbf{t}} \end{pmatrix}\tag{3.24}$$

In a partitioned approach using black-box solvers, the above Jacobian is never available or would need specifically enhanced differential solvers. Vierendeels et al. [163] adopted a quasi-Newton approach where the Jacobian is approximated by reduced order models based on the resolution to the least squares problem that combines input and output variables of each coupled solver. A major drawback of the approach (3.23) is that it is solved in a staggered way, where reduced models must be developed for each block. Namely, two linear systems must be solved at each coupling iteration:

$$\begin{aligned}
\mathbf{J}_{11}\delta\mathbf{t} &= -\left[\mathcal{F}(\mathbf{u}) - \mathbf{t}\right] - \mathbf{J}_{12}\delta\mathbf{u} \\
\mathbf{J}_{21}\delta\mathbf{u} &= -\left[\mathcal{S}(\mathbf{t}) - \mathbf{u}\right] - \mathbf{J}_{22}\delta\mathbf{t}
\end{aligned} \tag{3.25}$$

Here, $\delta\mathbf{t} = \mathbf{t}_{k+1} - \mathbf{t}_k$ is the load increment and $\delta\mathbf{u} = \mathbf{u}_{k+1} - \mathbf{u}_k$ is the displacement increment. In order to circumvent this drawback, one can alternately write a Newton linearized system for the residual equation defined by the difference in interface displacement. The root-finding problem thus becomes

$$\mathbf{r} = \mathcal{S}\left\{\mathcal{F}(\mathbf{u})\right\} - \mathbf{u} = \mathbf{0} \tag{3.26}$$

The benefit of the above approach is that the Jacobian is derived as $\mathbf{J} = \partial\mathbf{r}/\partial\mathbf{u}$ and the approximation of its inverse can be constructed in a single step. It has been shown in the literature [162] that the general performance of (3.26) in terms of number of coupling iterations is comparable to (3.22), while being fairly less robust. The iterative Newton procedure now writes

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \mathbf{J}_k^{-1}\mathbf{r}_k \tag{3.27}$$

Implementation

In practice, the algorithm implementing the NR update, along with the computation of the Jacobian, proceeds mostly on the same MPI process as the solid solver. The main reason for this choice of design is that the computation of the residual displacement in (3.26) is naturally obtained on the solid side, since the final operator is $\mathcal{S}$ and returns the displacement computed by the solid solver. At the discrete level of the coupled problem, let $\mathbf{u}$ denote the predicted interface displacement provided by the coupling algorithm to the fluid solver, and let $\hat{\mathbf{u}}$ represent the updated solid displacement returned by the solid solver based on the nodal loads computed by the fluid solver. The coupling condition is satisfied when $\hat{\mathbf{u}} = \mathbf{u}$ at a given time. Mathematically, this results in the following root-finding problem:

$$\begin{aligned}
\mathbf{r} &= \mathcal{S}\left\{\mathcal{F}(\mathbf{u})\right\} - \mathbf{u} = \mathbf{0} \\
&= \hat{\mathbf{u}} - \mathbf{u} = \mathbf{0}
\end{aligned} \tag{3.28}$$

If the interaction between the fluid and the structure is weak, it may be sufficient to simply communicate the loads and displacements between the solvers once per time step, without actually solving (3.28). This approach, known as loose or explicit coupling, is commonly used in aeroelasticity problems, where the solid is much denser than the fluid. Although efficient in principle, loose coupling does not enforce equilibrium conditions at the FS interface and becomes unstable in cases of strong interaction, regardless of the time step chosen [164]. This instability, inherent to the coupling scheme, is referred to as the artificial added-mass effect because significant portions of the fluid act as additional mass on the structure [72, 165]. In this work, strong coupling interactions are considered, meaning that NR iterations are performed at each time step until convergence:

$$\begin{aligned}\mathbf{u}_{k+1} &= \mathbf{u}_k - \mathbf{J}_k^{-1} \mathbf{r}_k \\ \Leftrightarrow \mathbf{u}_{k+1} &= \mathbf{u}_k - \mathcal{J}(\hat{\mathbf{u}}_k - \mathbf{u}_k)\end{aligned}\tag{3.29}$$

Here, the function $\mathcal{J}$, later referred as the update function, denotes the application of the inverse of the Jacobian to the residual displacement $\mathbf{r}_k = \hat{\mathbf{u}}_k - \mathbf{u}_k$ of the fluid-structure interface. This choice of notation is motivated by the fact that an explicit computation of the Jacobian is not always required to compute the displacement increment. Ultimately, only the product $\mathbf{J}_k^{-1} \mathbf{r}_k$ is needed in the calculations. In order to further simplify the subsequent developments, we introduce a modified Jacobian $\hat{\mathbf{J}}$ which is related to the actual Jacobian by

$$\hat{\mathbf{J}}^{-1} = \left(\frac{\partial \mathbf{r}}{\partial \hat{\mathbf{u}}} \right)^{-1} = \frac{\partial \hat{\mathbf{u}}}{\partial \mathbf{r}} = \frac{\partial (\mathbf{u} + \mathbf{r})}{\partial \mathbf{r}} = \frac{\partial \mathbf{u}}{\partial \mathbf{r}} + \mathbf{I} = \mathbf{J}^{-1} + \mathbf{I}\tag{3.30}$$

3.2.1 Interface Quasi-Newton Coupling

Quasi-Newton methods form a class of algorithms that approximate the Jacobian rather than computing it exactly. In this section, we derive the mathematical expression for the update function $\mathcal{J}$ which computes the displacement increment by approximating the product of the Jacobian's inverse with the residual vector. Within the context of partitioned FSI coupling between two black-box solvers, Degroote et al. [75, 166] demonstrated that the Interface Quasi-Newton (IQN) technique, specifically the variant using a Least-Squares approximation of the Inverse Jacobian (IQN-ILS), achieved a superior convergence rate than its common alternatives for the cases studied.

Specifically, it has been reported that the IQN-ILS required fewer iterations than coupling algorithms based on the Block Gauss-Seidel (BGS) method [74, 167] or on the Interface-GMRES method [73]. A subsequent study by Cerquaglia et al. [31] further validated the improved convergence rate and stability of the IQN-ILS method compared to the BGS for the considered set of test cases.

Inverse Least Squares

As previously mentioned, the IQN-ILS method approximates the inverse of the Jacobian matrix using a least-squares approach. This quasi-Newton iteration can be interpreted as a secant method, meaning the approximated Jacobian must satisfy a set of k relations known as the secant equations, also referred to as the fundamental equations of quasi-Newton methods [168, 169, 170]. These equations yield an estimate of the Jacobian based solely on solver outputs from the current time step:

$$\delta \hat{\mathbf{u}}_l = \hat{\mathbf{J}}_k^{-1} \delta \mathbf{r}_l \quad \forall l \in \{1, \dots, k\} \quad (3.31)$$

Here, $\hat{\mathbf{J}}_k^{-1}$ is the approximate inverse of the modified Jacobian at the k -th NR iteration, $\delta \hat{\mathbf{u}}_l$ holds for the difference between the interface displacement prediction at the beginning of the NR iteration l and the previous iteration $l - 1$. Similarly, we denote $\delta \mathbf{r}_l$ as the difference between the interface residual obtained at the end of the NR iteration l and the previous iteration:

$$\begin{aligned} \delta \hat{\mathbf{u}}_l &= \hat{\mathbf{u}}_l - \hat{\mathbf{u}}_{l-1} \\ \delta \mathbf{r}_l &= \mathbf{r}_l - \mathbf{r}_{l-1} \end{aligned} \quad (3.32)$$

It is important to note that these two quantities cannot be computed at iteration zero of each time step, as there is no previous residual or displacement available yet. Consequently, an alternative method must be used to obtain the first predictor. Nevertheless, we define the matrix of predicted displacement differences $\mathbf{W}_k$ and the matrix of residual differences $\mathbf{V}_k$ at iteration k , as follows:

$$\begin{aligned} \mathbf{W}_k &= \begin{pmatrix} \delta \hat{\mathbf{u}}_k & \delta \hat{\mathbf{u}}_{k-1} & \dots & \delta \hat{\mathbf{u}}_1 \end{pmatrix} \\ \mathbf{V}_k &= \begin{pmatrix} \delta \mathbf{r}_k & \delta \mathbf{r}_{k-1} & \dots & \delta \mathbf{r}_1 \end{pmatrix} \end{aligned} \quad (3.33)$$

These are rectangular matrices with k columns and $N_s \times d$ rows, where N_s is the number of interface nodes in the solid mesh and d is the dimension of the displacement field. This structure arises because the vectors of residuals and predicted displacements have been flattened into a single column, each corresponding to a specific iteration. Substituting (3.33) into (3.31) and (3.30) yields

$$\begin{aligned}\hat{\mathbf{J}}_k^{-1} &= \mathbf{W}_k \mathbf{V}_k^\dagger \\ \mathbf{J}_k^{-1} &= \mathbf{W}_k \mathbf{V}_k^\dagger - \mathbf{I}\end{aligned}\tag{3.34}$$

Here, $\mathbf{V}_k^\dagger$ is the generalization of the inverse, known as the pseudo-inverse or Moore-Penrose inverse, for rectangular matrices [171]. Injecting this least-square approximation into the definition of $\mathcal{J}$ results in the closed-form expression for the update function used in the so-called IQN-ILS coupling scheme:

$$\begin{aligned}\mathcal{J}(\mathbf{r}_k) &= (\mathbf{W}_k \mathbf{V}_k^\dagger) \mathbf{r}_k - \mathbf{r}_k \\ &= \mathbf{W}_k \mathbf{V}_k \backslash \mathbf{r}_k - \mathbf{r}_k\end{aligned}\tag{3.35}$$

To improve the convergence rate of the IQN-ILS, information from previous time steps can be incorporated into the system's matrices. Specifically, the matrices $\mathbf{V}$ and $\mathbf{W}$ may store not only the columns $\delta \hat{\mathbf{u}}$ and $\delta \mathbf{r}$ from the previous iterations of the current time step, but also those computed in prior time steps. However, because the vectors used to construct $\mathbf{V}$ and $\mathbf{W}$ do not form an orthogonal basis, some columns may become nearly linear combinations of others, especially when many columns from previous time steps are included. This can degrade the convergence properties of the method. To address this issue, filtering techniques are typically employed [96], although these can introduce additional computational costs or parameters. Furthermore, the optimal number of previous time steps to retain is problem-dependent and generally unknown a priori. Using too much historical data can negatively impact both the convergence rate and the computational cost of each iteration.

Multi-Vector Jacobian

The Interface Quasi-Newton with Multi-Vector Jacobian (IQN-MVJ) method [98] builds upon the IQN-ILS approach, but introduces a key modification in how the Jacobian is approximated.

While the IQN-ILS constructs the inverse Jacobian by solving an underdetermined least-squares problem, minimizing residuals over all past iterations, the IQN-MVJ updates an initial Jacobian approximation to the current configuration. In practice, this initial Jacobian is denoted $\mathbf{J}_{\text{PRV}}$ and represents the final Jacobian obtained at the end of the previous time step. Mathematically, this approach introduces an additional constraint, ensuring that the current Jacobian is approximated by

$$\hat{\mathbf{J}}_k^{-1} = \hat{\mathbf{J}}_{\text{PRV}}^{-1} + \Delta\hat{\mathbf{J}}_k^{-1} \quad (3.36)$$

Here, $\hat{\mathbf{J}}_{\text{PRV}}^{-1}$ is known from the previous time step, and $\Delta\hat{\mathbf{J}}_k^{-1}$ is the increment that must be computed for the k -th iteration of the current time step. As a side remark, the IQN-MVJ becomes mathematically equivalent to the IQN-ILS at the very first time step, because $\mathbf{J}_{\text{PRV}}$ does not exist yet and is treated as zero. This leads to $\hat{\mathbf{J}}_k^{-1}$ being computed using the same approximation as in the IQN-ILS approach. Nevertheless, injecting (3.36) into (3.31) results in

$$\begin{aligned} \Delta\hat{\mathbf{J}}_k^{-1} &= (\mathbf{W}_k - \hat{\mathbf{J}}_{\text{PRV}}^{-1} \mathbf{V}_k) \mathbf{V}_k^\dagger \\ \Leftrightarrow \hat{\mathbf{J}}_k^{-1} &= \hat{\mathbf{J}}_{\text{PRV}}^{-1} + (\mathbf{W}_k - \hat{\mathbf{J}}_{\text{PRV}}^{-1} \mathbf{V}_k) \mathbf{V}_k^\dagger \end{aligned} \quad (3.37)$$

This approach offers two key advantages. First, it implicitly incorporates information from previous time steps, which enhances convergence speed compared to methods that rely solely on the current time step's data. Second, the number of columns in $\mathbf{V}_k$ and $\mathbf{W}_k$ is limited to the number of iterations within the current time step, reducing the need for filtering. The main disadvantage of the IQN-MVJ is the explicit computation of the inverse Jacobian, which can increase memory requirements and computational costs. However, $\hat{\mathbf{J}}_k^{-1}$ is a dense matrix of size $N_s \times N_s$, where N_s is typically much smaller than the total number of nodes in both the fluid and solid domains. As a result, the computational effort needed to compute and store the inverse Jacobian is often relatively small compared to the time required to solve the conservation of momentum by the fluid and solid solvers. By injecting (3.37) into (3.30), the following relation is derived:

$$\mathcal{J}(\mathbf{r}_k) = \hat{\mathbf{J}}_{\text{PRV}}^{-1} \mathbf{r}_k + (\mathbf{W}_k - \hat{\mathbf{J}}_{\text{PRV}}^{-1} \mathbf{V}_k) \mathbf{V}_k \setminus \mathbf{r}_k - \mathbf{r}_k \quad (3.38)$$

Similarly to the IQN-ILS, the matrices $\mathbf{V}_k$ and $\mathbf{W}_k$ cannot be computed at the first iteration of each time step since there is no previous iteration available. In this case, the previous Jacobian is simply reused without modification. It is also possible to encounter a situation where neither the previous Jacobian nor previous iterations are available. In such cases, a BGS iteration is performed [74, 167]. The BGS uses a scalar relaxation parameter ω in place of a Jacobian. This parameter can take values between 0 and 1 to avoid potential divergence, and is set to 0.5 by default. The update function thus simplifies to

$$\mathcal{J}(\mathbf{r}_k) = -\omega \mathbf{r}_k \quad (3.39)$$

To assess the convergence of the fluid-structure coupling at each time step, we define a scalar quantity that measures the order of magnitude of the residual displacement. This scalar residual is typically non-dimensional to ensure that it is independent of the size of the interface, the units of the physical quantities being evaluated, and the number of nodes used in its discretization:

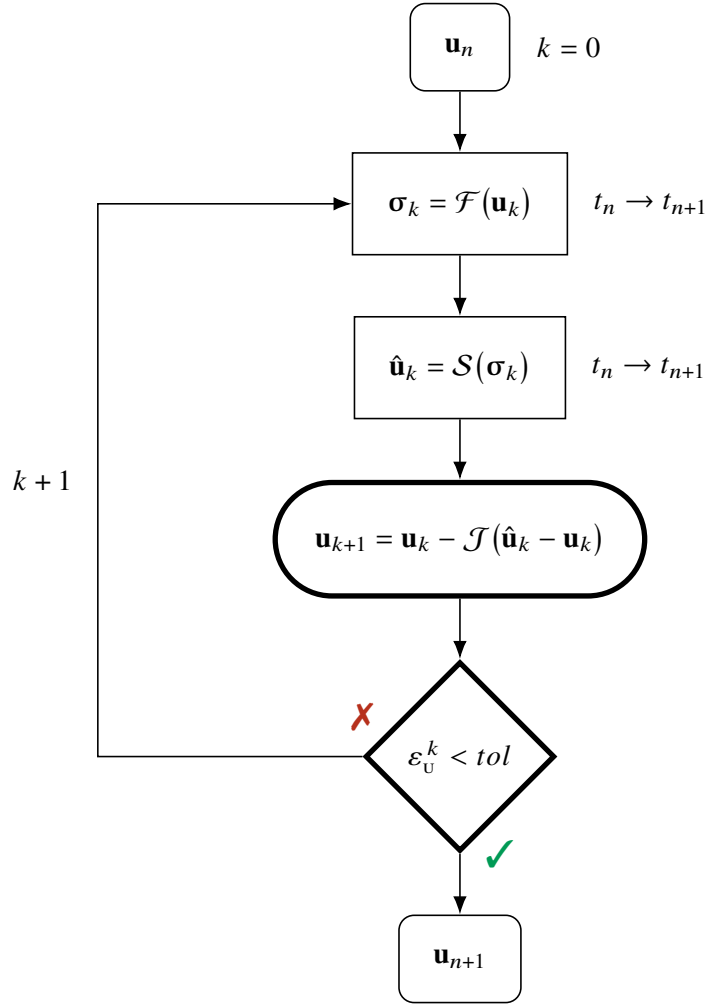
$$\varepsilon_U^k = \frac{\|\mathbf{r}_k\|}{\|tol + \hat{\mathbf{u}}_k\|} < tol \quad (3.40)$$

Here, tol is a user-defined tolerance. Note that the inclusion of tol in the denominator is a regularization factor. Regularization of the convergence criterion aims to prevent divergence when the interface displacement approaches zero. If the maximum number of iterations is reached before $\varepsilon_U < tol$, or if either the structural or fluid solvers fail to find an equilibrium solution, the current time step is restarted with a reduced time increment $\Delta t' = \Delta t/2$, and this process continues iteratively until (3.40) is satisfied. If convergence is achieved with $\Delta t' < \Delta t$, the time increment is progressively increased to restore the original time step size.

Side Remarks

The continuity condition in (3.19) explicitly mentions the continuity of surface tractions at the interface. Consequently, the theoretical framework suggests that surface tractions are computed by the fluid solver and then passed to the structural solver. In our implementation, only the nodal components of the stress tensor are communicated to the structural solver. Practically, smoothed nodal stresses can be recovered by extrapolation from the Gauss points or from superconvergent sampling points to the nodes, as better accuracy is obtainable for gradients and stresses at those points [172, 173].

The latter approach is referred as Superconvergent Patch Recovery (SPR) and is described with more details in [174, 175, 176]. Metafor then automatically projects these received stresses onto its own finite elements and integrates the resulting surface tractions as Neumann boundary conditions. This choice to communicate stresses rather than surface tractions is driven by the reliance on nodal quantities for the communication. Communicating surface tractions would require estimating nodal normals on the fluid side. However, normals are not defined at the nodes of linear finite elements, as this would require a local surface perpendicular to the normal, which does not exist at the vertices. Moreover, the normal on the fluid mesh may not align with that of the solid mesh when non-matching. The issue of non-matching meshes is addressed in the next section.



Algorithm 4: Strategy for solving partitioned fluid-structure coupling problems with a DN scheme, between time t_n and time t_{n+1} . Recall that each call to $\mathcal{S}$ and $\mathcal{F}$ involves the solving of a FE system within the current time step.

A summary of the coupling procedure is provided in Algorithm 4, using nodal values of the stress tensor σ rather than surface tractions. Recall that the fluid solver uses the PFEM-LS. This means that the mesh topology of the FS interface (i.e. the number of nodes and their connectivity in the mesh) may evolve due to the remeshing procedure. While fluid remeshing during the NR iterations does not pose fundamental issue to the IQN, as the Jacobian is computed based on nodal displacement of the solid interface, it may still slow-down the convergence due to small perturbations of the pressure field in the fluid between successive remeshing, caused by node connectivity changes, and due to the computational cost of the remeshing itself. Thereby, remeshing in the PFEM-LS is prevented during IQN iterations, and triggered once the new equilibrium has been reached at the end of the time step.

3.2.2 Numerical Examples

In this subsection, we compare the convergence rates of the IQN-ILS with the IQN-MVJ across various test cases. The first test case involves a dam break against a solid and deformable obstacle, a problem previously studied numerically by Zhu and Scott [177] using a semi-partitioned PFEM-FEM approach with a corotational mesh of beam elements for the obstacle, and later by Cerquaglia et al. [31] who employed quadrilateral elements for the solid and the PFEM for the fluid. In this study, the obstacle is modeled using an elastoplastic constitutive law with linear isotropic hardening:

$$\sigma_Y = \sigma_Y^0 + H\bar{\epsilon}_{VP} \quad (3.41)$$

Here, σ_Y is the current yield stress, σ_Y^0 is the initial yield stress, H is the isotropic hardening coefficient and $\bar{\epsilon}_{VP}$ is the effective plastic strain, which serves as a scalar measure of the accumulated plastic deformation. The initial configuration is shown in Figure 52. Two mesh refinements are considered, a reference mesh with a constant size-field of $h = 3$ mm, and a fine mesh with a constant size-field of $h = 1.5$ mm.

The structural solver employs linear quadrilateral finite elements for meshing the obstacle. This choice is especially important due to the inherent incompressibility of the material during plastic deformation. Indeed, quadrilateral elements help mitigate locking effects in the plastic region, ensuring a more accurate representation of the material's response under highly nonlinear conditions.

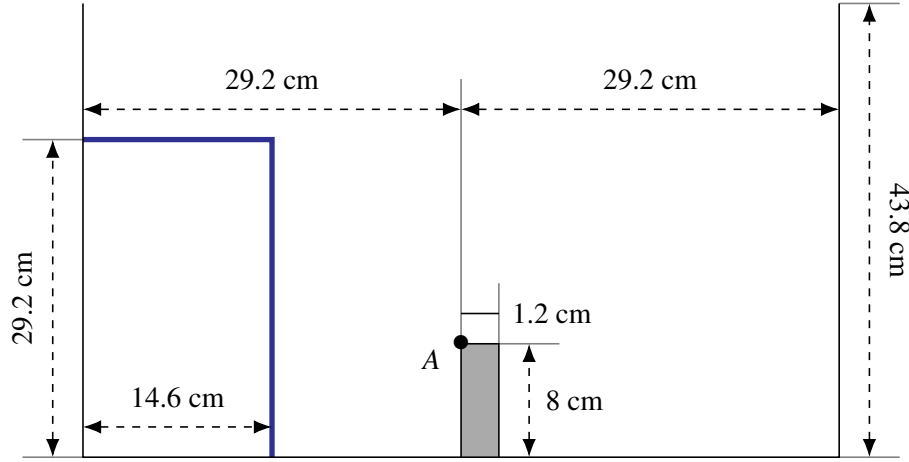


Figure 52: Initial configuration of the **dam break** test case whose simulation parameters are indicated in Table 6. No-slip BC are applied to rigid and deformable boundaries, and the initial free surface is outlined in bold.

<i>Remesh</i>	<i>Fluid</i>	<i>Solid</i>	<i>Coupling</i>
$\delta = 0.6$ $\omega = 0.6$ $\alpha = 1.2$ $\beta = 0.2$ $\varepsilon = 0$	$\rho = 10^3 \text{ kg/m}^3$ $g = -9.81 \text{ m/s}^2$ $\mu = 10^{-3} \text{ Pa s}$ $\gamma = 0.07 \text{ N/m}$	$\rho = 2500 \text{ kg/m}^3$ $g = 0 \text{ m/s}^2$ $\nu = 0$ $E = 10^6 \text{ Pa}$ $H = 2 \times 10^4 \text{ Pa}$ $\sigma_Y^0 = 5 \times 10^4 \text{ Pa}$	$\Delta t_{\text{MAX}} = 10^{-4} \text{ s}$ $tol = 10^{-8}$
	$tol = 10^{-8}$	$tol = 10^{-7}$ $\rho_{\infty} = 0.01$	

Table 6: Simulation parameters for the **dam break** whose initial configuration is given in Figure 52. The governing equations are solved with the PSPG stabilization and the edge splitting remeshing with level set.

To further mitigate volumetric locking in the incompressible regimes, Selective Reduced Integration with Pressure Report (SRIPR) is used in the structural solver. SRIPR estimates the pressure at the element center [178]. The pressure is then reported at each deviatoric integration point, and the pressure integral is computed with 4 points in 2D (or 8 in 3D) instead of just one, improving precision and numerical stability. Furthermore, a small and constant time step is employed to ensure a fair comparison between the different algorithms, as variations caused by adaptive time stepping are eliminated.

The average number of iterations per time step is presented in Table 7. As shown, the MVJ technique achieves a convergence rate comparable to a well-tuned ILS for the cases studied, without the need for filtering or user-defined parameters. Recall that the filtering algorithm is designed to detect and remove linearly dependent columns from $\mathbf{V}$ and $\mathbf{W}$ when their column count becomes large. This typically occurs when multiple columns from previous time steps are retained in the current matrices to improve the convergence rate of IQN-ILS. Cerquaglia et al. [31] demonstrated that the Haelterman filtering algorithm [96] outperforms other methods within our framework, it has thus been employed in the present comparison.

The deformation of the solid structure is analyzed by tracking the displacement of the point *A* indicated in Figure 52. This displacement is compared with results from the literature in Figure 53, demonstrating that our simulations align with numerical results from other authors. However, the final displacement predicted by Zhu and Scott [177] for the obstacle differs from our simulations.

<i>Method</i>	<i>Filter</i>	<i>Steps</i>	<i>Avg it #</i>
ILS	—	0	7.126
ILS	Haelterman	5	4.090
ILS	Haelterman	10	3.462
ILS	Haelterman	20	3.051
ILS	Haelterman	30	2.947
MVJ	—	—	2.640

Table 7: Average number of iterations per time step for the **dam break** test case. The column *Steps* refers to the number of iterations from previous time steps that have been retained in the IQN-ILS.

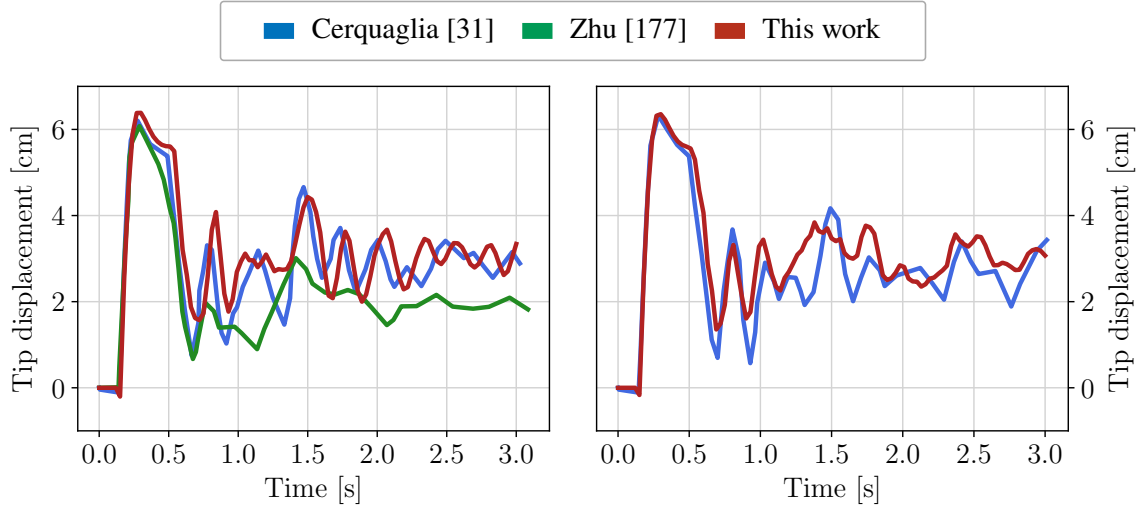


Figure 53: Tip x -displacement for the **dam break** with obstacle in 2D, first using the reference mesh, and second using the fine mesh. The initial position of the tip is indicated by the point A in Figure 52.

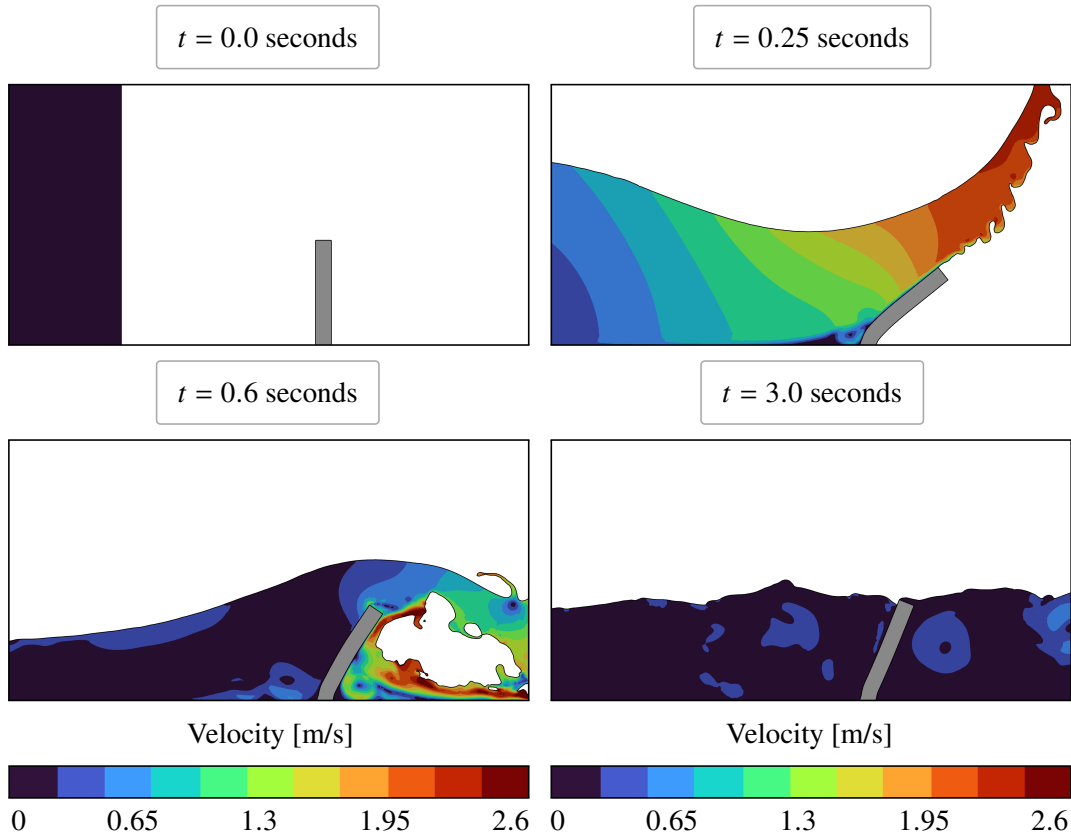


Figure 54: Snapshots of the particle finite element solution for the **dam break** test case whose initial configuration is given in Figure 52, at different time intervals. The parameters from Table 6 have been used.

This discrepancy arises because Zhu and Scott did not account for the thickness of the obstacle on the fluid side, which introduces a significant difference in the geometry of the problem compared to the FSI scenario presented in this work, especially once the fluid has passed the obstacle. To further illustrate the solution, snapshots from the simulation at various time intervals are shown in Figure 54.

Rubber Gate

The second test case involves water flow interacting with a rubber gate, a problem previously studied both experimentally and numerically by Antoci et al. [179] using a monolithic fluid-structure SPH model. The fluid pressure causes the elastic gate to deform, allowing fluid particles to pass beneath it. For this study, the structural deformation is modeled using a Mooney-Rivlin material, with the following two-parameter strain energy function:

$$\Psi = C_1(I_1 - 3) + C_2(I_2 - 3) + \frac{K_0}{2} \left[(J - 1)^2 + \ln J \right] \quad (3.42)$$

Here, K_0 denotes the initial bulk modulus, $J = \det(\mathbf{F})$ is the determinant of the deformation gradient tensor $\mathbf{F}$, while I_1 and I_2 are respectively the first and second principal invariants of the modified left Cauchy-Green deformation tensor $\bar{\mathbf{B}}$. As a reminder, the modified left Cauchy-Green deformation tensor has unit determinant, which excludes the effect of volume change in the material:

$$\begin{aligned} \bar{\mathbf{F}} &= J^{-1/3} \mathbf{F} \\ \bar{\mathbf{B}} &= \bar{\mathbf{F}} \bar{\mathbf{F}}^\top \end{aligned} \quad (3.43)$$

The material coefficients C_1 and C_2 were determined by fitting the experimental tension test data provided in [179]. A schematic representation of the initial configuration is shown in Figure 55. The 3D version of the test case has exactly the same geometry as the 2D test case, with a tank depth of 10 cm. The characteristic size of the elements is governed by a function $h(\mathbf{x})$ of the form

$$h(\mathbf{x}) = \min \left\{ h_0 + 4 d(\mathbf{x})^2, h_\infty \right\} \quad (3.44)$$

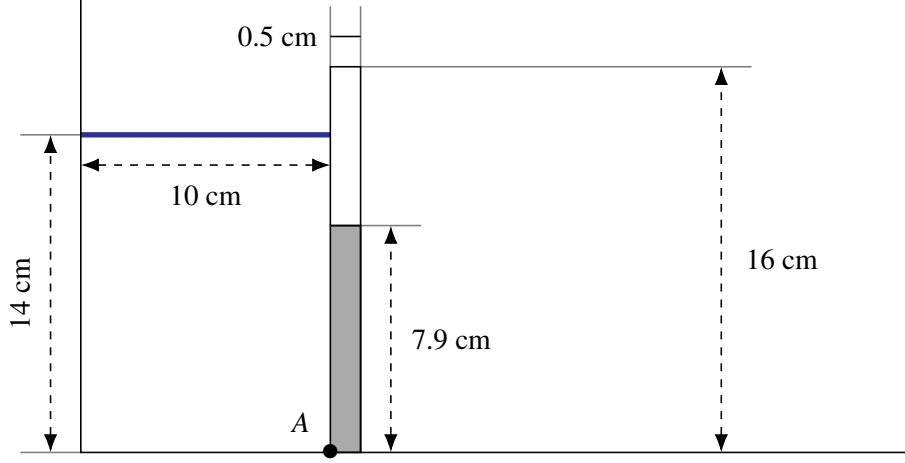


Figure 55: Initial configuration of the **rubber gate** test case whose simulation parameters are indicated in Table 8. No-slip BC are applied to rigid and deformable boundaries, and the initial free surface is outlined in bold.

<i>Remesh</i>	<i>Fluid</i>	<i>Solid</i>	<i>Coupling</i>
$\delta = 0.6$ $\omega = 0.6$ $\alpha = 1.2$ $\beta = 0.2$ $\varepsilon = 0.15$	$\rho = 10^3 \text{ kg/m}^3$ $g = -9.81 \text{ m/s}^2$ $\mu = 10^{-3} \text{ Pa s}$ $\gamma = 0 \text{ N/m}$	$\rho = 1100 \text{ kg/m}^3$ $g = -9.81 \text{ m/s}^2$ $C_1 = -1.2 \times 10^6 \text{ Pa}$ $C_2 = 2.4 \times 10^6 \text{ Pa}$ $K_0 = 1.3 \times 10^6 \text{ Pa}$	$\Delta t_{\text{MAX}} = 10^{-4} \text{ s}$ $tol = 10^{-6}$
	$tol = 10^{-6}$ $\lambda = 0.5$	$tol = 10^{-6}$ $\rho_{\infty} = 0.01$	

Table 8: Simulation parameters for the **rubber gate** in 2D whose initial configuration is given in Figure 55. The governing equations are solved with the fractional step and the edge splitting remeshing with level set.

Here, $d(\mathbf{x})$ represents the minimal distance between the position $\mathbf{x}$ and the deformable solid, the free surface or the vertical walls. The minimal characteristic size in 2D is $h_0 = 1$ mm, while the maximum allowed characteristic size is $h_\infty = 5$ mm. In order to compensate for higher computation times, the 3D variant uses a coarser mesh with $h_0 = 2$ mm and $h_\infty = 10$ mm. To mitigate volumetric locking in the nearly incompressible Mooney-Rivlin material, SRIPR hexahedral elements have been used in the structural solver.

The average number of iterations per time step is presented in Table 9, confirming the results from the dam break test case. Specifically, the MVJ technique achieves a convergence rate that is either comparable to or even superior to that of a well-tuned ILS for the 2D and 3D cases studied. The deformation of the solid structure is analyzed by tracking the displacement of the point *A* indicated in Figure 55. This displacement is compared to literature results in Figure 56, showing that our simulations are consistent with both numerical and experimental studies. Additionally, snapshots from the simulation at different time intervals are provided in Figures 57 and 58.

			2D	3D
<i>Method</i>	<i>Filter</i>	<i>Steps</i>	<i>Avg iteration #</i>	
ILS	–	0	3.576	3.900
ILS	Haelterman	5	2.212	2.858
ILS	Haelterman	10	1.940	2.713
ILS	Haelterman	20	1.852	2.685
ILS	Haelterman	30	1.868	2.683
MVJ	–	–	1.827	2.039

Table 9: Average number of iterations per time step for the **rubber gate** test case. The column *Steps* refers to the number of iterations from previous time steps that have been retained in the IQN-ILS.

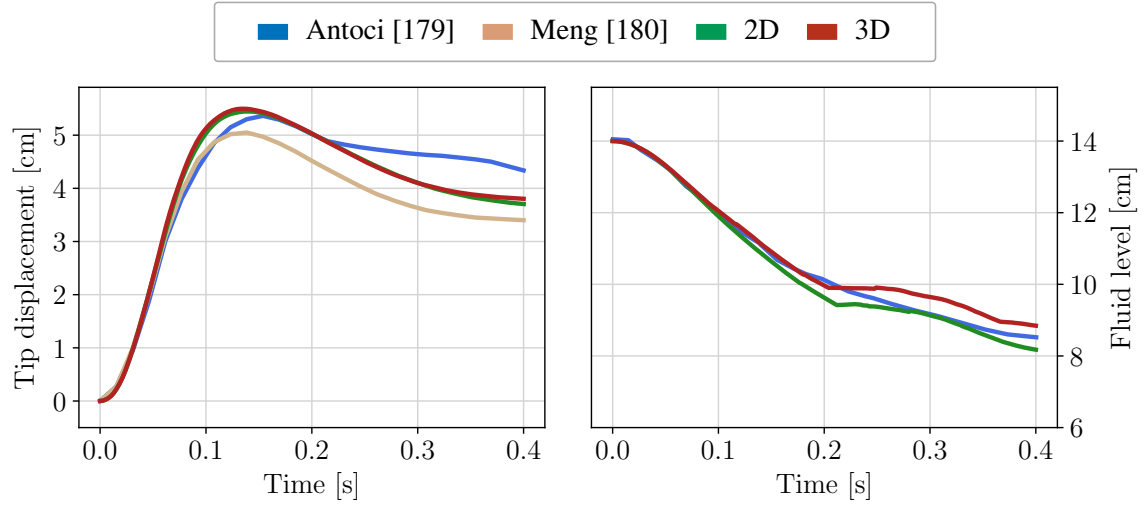


Figure 56: Tip displacement magnitude and fluid level for the **rubber gate** test case in 2D. The initial position of the tip is indicated by the point A in Figure 55. The fluid level is measured on the left of the gate.

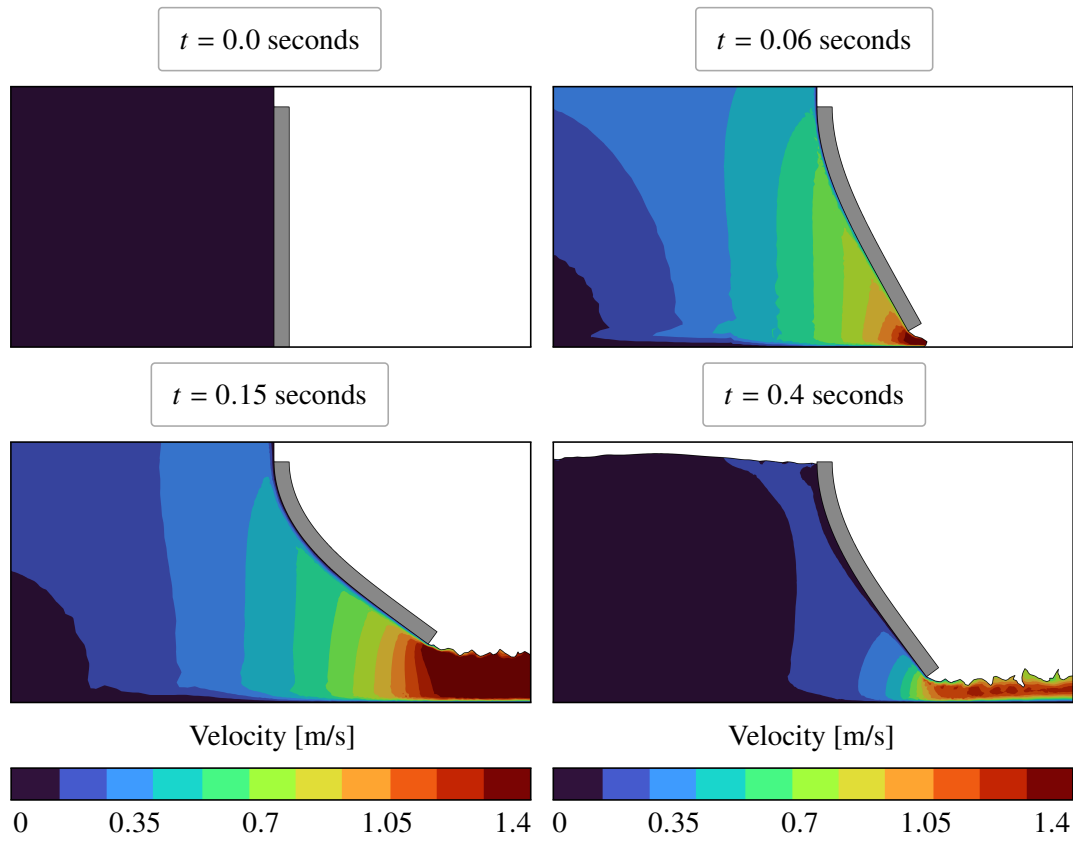


Figure 57: Snapshots of the particle finite element solution for the **rubber gate** test case in 2D, whose initial configuration is given in Figure 55, at different time intervals. The parameters from Table 8 have been used.

Aquatic Vegetation

The last test case investigates the flow-induced reconfiguration of buoyant and flexible aquatic vegetation, originally studied experimentally by Luhar and Nepf [181]. This test case was later analyzed numerically by Bano et al. [182], who used a partitioned coupling between the commercial software Ansys Mechanical for the solid and Ansys Fluent for the fluid [183]. The solid is modeled using a finite strain linear elastic law, meaning that the stress-strain relationship for the vegetation remains linear and follows Hooke's law, while geometric nonlinearities arising from large deformations of the structure are accounted for:

$$\overset{\nabla}{\boldsymbol{\sigma}} = \mathbb{H}(E, \nu) : \mathbf{D} \quad (3.45)$$

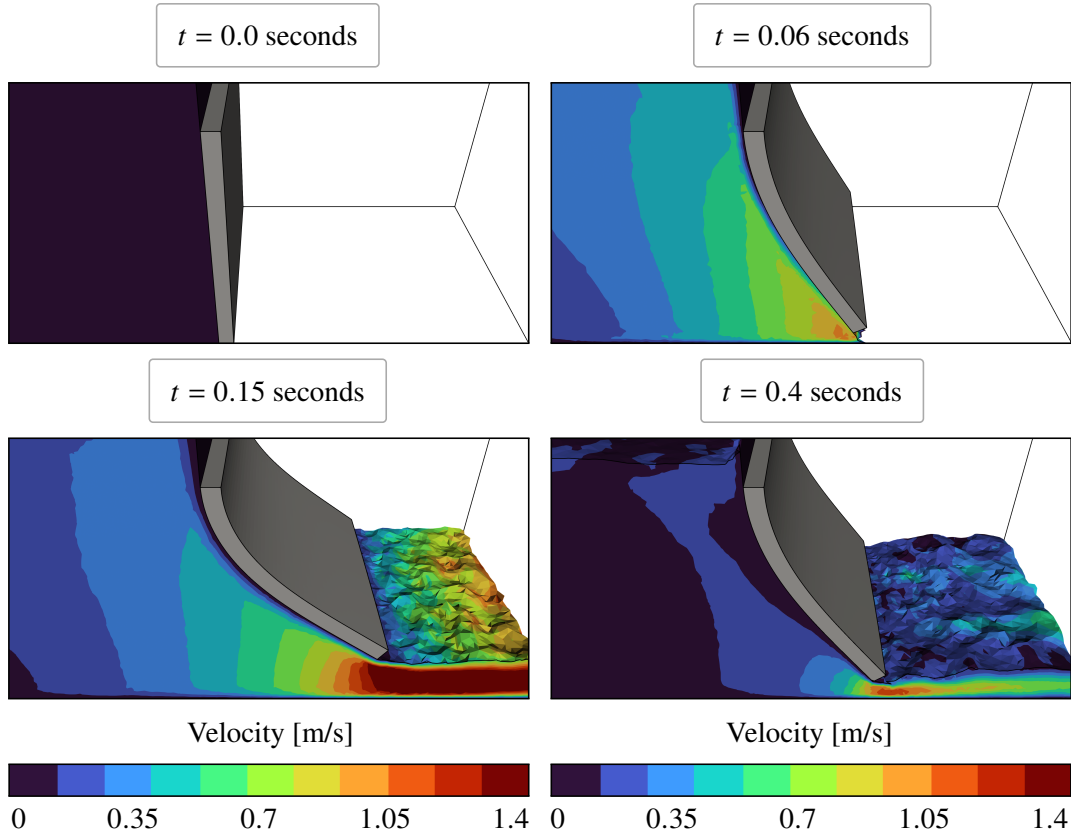


Figure 58: Snapshots of the particle finite element solution for the **rubber gate** test case in 3D, whose initial configuration is given in Figure 55, at different time intervals. The parameters from Table 8 have been used.

Here, $\mathbb{H}$ is a constant stiffness tensor, expressed as a function of the elastic modulus E and Poisson's ratio ν . The choice of this simple model for the solid structure ensures consistency with the reference paper [182], where linear elasticity was employed. The initial configuration of the problem is shown in Figure 59.

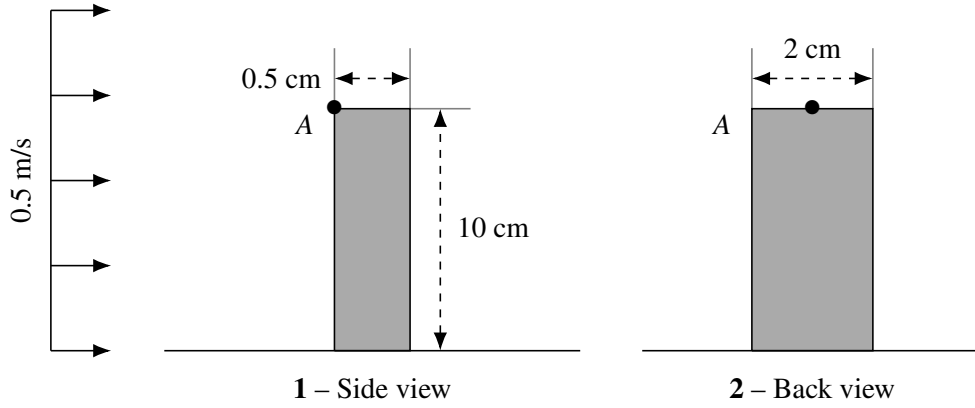


Figure 59: Initial configuration of the **aquatic vegetation** test case whose simulation parameters are indicated in Table 10. No-slip BC are applied to rigid and deformable boundaries, and the fluid domain is closed without free surface.

<i>Remesh</i>	<i>Fluid</i>	<i>Solid</i>	<i>Coupling</i>
$\delta = 0.8$ $\omega = 0.4$ $\alpha = 1.7$ $\beta = 0.2$ $\varepsilon = \infty$	$\rho = 1220 \text{ kg/m}^3$ $g = 0 \text{ m/s}^2$ $\mu = 1 \text{ Pa s}$ $\gamma = 0 \text{ N/m}$	$\rho = 1030 \text{ kg/m}^3$ $g = -9.81 \text{ m/s}^2$ $\nu = 0.3$ $E = 1.23 \times 10^6 \text{ Pa}$	$\Delta t_{\text{MAX}} = 10^{-4} \text{ s}$ $tol = 10^{-6}$
	$tol = 10^{-6}$ $\lambda = 0.5$	$tol = 10^{-6}$ $\rho_{\infty} = 0.01$	

Table 10: Simulation parameters for the **aquatic vegetation** whose initial configuration is given in Figure 59. The governing equations are solved with the fractional step and the edge splitting remeshing with level set.

The solid is meshed with regular hexahedral elements. An important remark is that while it is still possible to match the fluid and solid nodes at the interface, this configuration can degrade the tessellation quality. Regularly arranged nodes on 3D interfaces are prone to generating slivers and poor quality simplices in the PFEM mesh. To mitigate this issue, the FS interface on the fluid side uses an unstructured mesh, ensuring better-quality tetrahedral elements at the cost of node mismatching with the structured solid mesh. An interpolation technique is then employed to facilitate communication between the solid and fluid meshes. A detailed analysis of this interpolation method is provided in the next section. Nevertheless, the mesh size-field is given by:

$$h(\mathbf{x}) = \min \left\{ h_0 + 0.1 d(\mathbf{x}), h_\infty \right\} \quad (3.46)$$

Here, $d(\mathbf{x})$ represents the minimum distance to the solid structure, $h_0 = 2$ cm is the minimum characteristic element size, and $h_\infty = 3$ cm is the maximum element size. The domain is assumed to be semi-infinite, with the ground acting as a boundary that influences the structure (since it is clamped), while the top, sides, and outlet are sufficiently distant from the region of interest to minimize boundary effects, effectively simulating an infinite domain in those directions. The inlet imposes a velocity of 0.5 m/s along the x -axis. Similarly to the approach proposed by Cerquaglia and Meduri [29, 184], inlet boundary conditions are applied using a single layer of Eulerian nodes with prescribed velocity. These nodes remain fixed in space, effectively simulating fluid entering the computational domain without requiring physical movement of the inlet boundary.

<i>Case</i>	<i>Method</i>	<i>Filter</i>	<i>Steps</i>	<i>Avg it #</i>
3D	ILS	—	0	3.629
	ILS	Haelterman	5	2.744
	ILS	Haelterman	10	2.582
	ILS	Haelterman	20	2.308
	ILS	Haelterman	30	2.213
	MVJ	—	—	1.709

Table 11: Average number of iterations per time step for the **aquatic vegetation** test case. The column *Steps* refers to the number of iterations from previous time steps that have been retained in the IQN-ILS.

The analysis follows the same procedure as the previous cases, using a small and constant time step to eliminate variations caused by adaptive time stepping. The average number of iterations per time step is presented in Table 11. The deformation of the solid structure is analyzed by tracking the displacement of the point A indicated in Figure 59. A comparison of this displacement with results from the literature is presented in Figure 60, demonstrating that our simulations match both numerical and experimental studies. To provide further insight, snapshots from the simulation are shown in Figure 61.

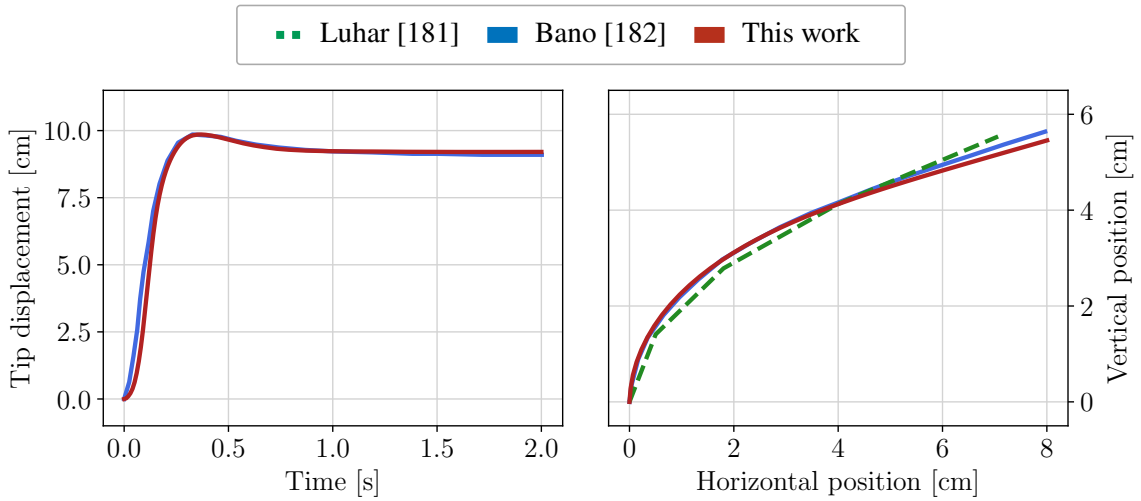


Figure 60: Tip displacement magnitude and bending profile for the **aquatic vegetation** test case in 3D. The initial position of the tip is indicated by the point A in Figure 59. The dashed line features experimental results.

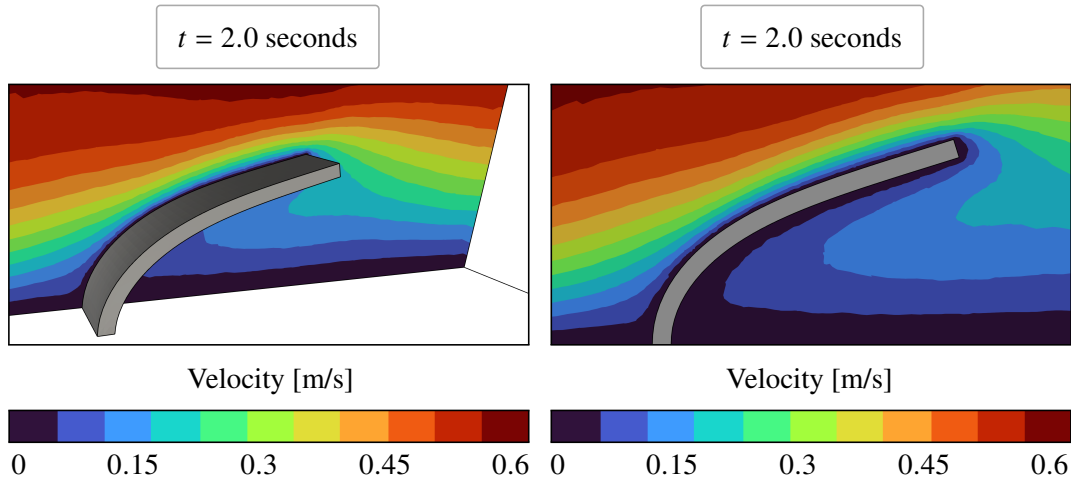


Figure 61: Snapshots of the particle finite element solution for the **aquatic vegetation** test case whose initial configuration is given in Figure 59, at time $t = 2.0$ seconds. The parameters from Table 10 have been used.

3.3 Non-Matching Meshes

When the solid and fluid meshes are conformal at the FS interface, each interface node viewed by the solid solver has its equivalent on the fluid side, and the meshes are said to be matching. However, in many situations, one may want to use different levels of mesh refinement, or different type of elements in the two solvers. Consequently, an interpolation technique is required to transfer the nodal data from one solver to another. In this context, the discretized counterpart of the coupling condition (3.19) can be viewed as follows:

$$\begin{aligned}\mathcal{V}_{\text{SF}}(\mathbf{u}_s) &= \mathbf{u}_f \\ \mathbf{t}_s &= -\mathcal{V}_{\text{FS}}(\mathbf{t}_f)\end{aligned}\tag{3.47}$$

Here, $\mathcal{V}_{\text{SF}}$ is the solid-to-fluid interpolation operator that maps the nodal displacement of the solid mesh onto the FE space of the fluid solver. Similarly, $\mathcal{V}_{\text{FS}}$ is the fluid-to-solid interpolation operator that maps the nodal loads of the fluid mesh onto the FE space of the solid solver. Recall that $\mathbf{t}_f$ designates the fluid loads by surface tractions, but in practice, we may opt for communicating nodal stresses rather than nodal tractions directly. The root-finding problem (3.28) writes

$$\mathbf{r} = \mathcal{S}\mathcal{V}_{\text{FS}}\left\{\mathcal{F}\mathcal{V}_{\text{SF}}(\mathbf{u}_s)\right\} - \mathbf{u}_s = \mathbf{0}\tag{3.48}$$

Here, we note by $\mathcal{F}\mathcal{V}_{\text{SF}}(\mathbf{u}_s)$ the successive application of $\mathcal{V}_{\text{SF}}$ and $\mathcal{F}$ to the nodal displacements defined on the solid mesh. This choice of notation aims to highlight the fact that the fluid solver uses the nodal displacements communicated by the solid solver, interpolated onto its own FE mesh as a Dirichlet condition, without overloading the notations with parentheses. The same explanation holds for $\mathcal{S}\mathcal{V}_{\text{FS}}(\mathbf{t}_f)$ implying that the solid solver operates with nodal loads, provided by the fluid solver, interpolated onto its own FE mesh as Neumann BC.

3.3.1 Radial Basis Functions

General theory of Radial Basis Function (RBF) interpolation can be found in [185, 186], and examples of applications of the method are commonly found in the literature [187, 188, 189]. This section summarizes the findings of our study conducted during the preparation of this thesis [190].

The RBF interpolation offers significant advantages, particularly in terms of computational efficiency, compared to other approaches such as free-form deformation [188]. Since the remeshing procedure frequently updates the mesh structure, the interpolation matrices must be updated frequently, and the computation time becomes crucial in the choice of method. Moreover, this technique is highly versatile as it requires minimal topological information, limited to the relative distances between nodes, making it independent of an explicit interface mesh.

Indeed, the RBF contrasts with methods such as Mortar [191] or finite volume transfer techniques [192] where mesh connectivity data is required. This characteristic is especially valuable in the context of the PFEM, where the FS interface is defined by the boundary where nodal data are exchanged between the fluid and the solid solvers. In particular, the fluid-structure contact region, also referred to as the wetted boundary, represents only the portion of the FS interface that is in contact with the fluid at a given time. On free-surface flows, this contact region is likely to evolve with time. In the PFEM, finite elements are generated only for nodes within the current computational domain, including those at its boundaries. Consequently, the fluid solver only generates a mesh on wetted boundaries of the FS interface, while the dry boundaries are represented by free nodes that track the interface's position dynamically. While this may be challenging for mesh-based interpolation methods, it does not introduce any issues for the RBF interpolation, as it is a meshless method.

Solid Displacement Interpolation

This subsection describes the construction of the interpolation operator $\mathcal{V}_{\text{SF}}$, which transfers the displacement field $\mathbf{u}_s$ discretized as a vector of nodal values $\mathbf{u}_s$ on the solid mesh (also called the source mesh), into a vector of nodal values $\mathbf{u}_f$ on the fluid mesh (also called the target mesh). The RBF interpolation method computes the interpolated displacement as a weighted sum of basis functions:

$$\mathbf{u}_f(\mathbf{x}) = \sum_{j=1}^{N_s} \alpha_j \varphi\{\varepsilon_j(\mathbf{x})\} + \beta^\top \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} \quad (3.49)$$

Here, φ is the basis function, evaluated for each normalized radius $\varepsilon_j(\mathbf{x}) = r_j(\mathbf{x})/R$ where $r_j(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_s^j\|$ is the Euclidean distance between the coordinates $\mathbf{x}$ and the j -th node in the solid mesh. R is a user-defined characteristic radius whose influence is discussed in the next subsection.

The choice of the basis function is user-dependent and significantly affects the interpolation's behavior. A selection of commonly used basis functions is provided in Table 12. The second term on the right-hand side introduces constant and linear polynomial terms, with coefficients stored in β , ensuring that constant functions are exactly reproduced by the interpolation. Lastly, the coefficients stored in α , which will be computed later, are associated with the basis function. At the discrete level of the coupled problem, $\mathbf{u}_F$ is evaluated only at the fluid nodes, and the RBF interpolation yields:

$$\mathbf{u}_F = \underbrace{\begin{pmatrix} \varphi\{\varepsilon_1(\mathbf{x}_F^1)\} & \dots & \varphi\{\varepsilon_{N_s}(\mathbf{x}_F^1)\} & 1 & [\mathbf{x}_F^1]^\top \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \varphi\{\varepsilon_1(\mathbf{x}_F^{N_F})\} & \dots & \varphi\{\varepsilon_{N_s}(\mathbf{x}_F^{N_F})\} & 1 & [\mathbf{x}_F^{N_F}]^\top \end{pmatrix}}_{\mathbf{B}} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \quad (3.50)$$

Here, $\mathbf{u}_F$ is an $N_F \times 3$ matrix in 3D, where each row corresponds to the coordinates of a node in the fluid interface mesh. The vector $\mathbf{x}_F^j$ represents the coordinates of the j -th interface node in the fluid mesh, while $\mathbf{x}_s^j$ corresponds to the j -th interface node in the solid mesh. In 3D, $\mathbf{B}$ is a rectangular matrix of size $N_F \times (N_s + 4)$, where N_F is the number of nodes in the fluid interface mesh and N_s is the number of nodes in the solid interface mesh. To construct the interpolation, the coefficients α and β must be computed by satisfying a set of constraints. The first constraint ensures that the interpolated field passes exactly through the source data. The second constraint stipulates that if the source field can be exactly represented by the polynomial terms alone, the interpolation should reduce to those terms, making $\alpha = \mathbf{0}$ in this case. These constraints lead to the following linear system, which, once solved, determines the values of the coefficients:

$$\underbrace{\begin{pmatrix} \mathbf{U} & \mathbf{P} \\ \mathbf{P}^\top & \mathbf{0} \end{pmatrix}}_{\mathbf{A}} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \mathbf{u}_s \\ \mathbf{0} \end{pmatrix} \quad (3.51)$$

The matrix $\mathbf{A}$ of the system is symmetric, with dimensions $(N_s + 4) \times (N_s + 4)$ in 3D. The first N_s equations correspond to the constraint of exact matching at the source nodes, while the last 4 equations (or the last 3 equations in the 2D case) enforce the second constraint on the α coefficients. The sub-matrices $\mathbf{U}$ and $\mathbf{P}$ are defined as follows:

$$\mathbf{U} = \begin{pmatrix} \varphi\{\varepsilon_1(\mathbf{x}_s^1)\} & \dots & \varphi\{\varepsilon_{N_s}(\mathbf{x}_s^1)\} \\ \vdots & \ddots & \vdots \\ \varphi\{\varepsilon_1(\mathbf{x}_s^{N_s})\} & \dots & \varphi\{\varepsilon_{N_s}(\mathbf{x}_s^{N_s})\} \end{pmatrix} \quad \mathbf{P} = \begin{pmatrix} 1 & [\mathbf{x}_s^1]^\top \\ \vdots & \vdots \\ 1 & [\mathbf{x}_s^{N_s}]^\top \end{pmatrix} \quad (3.52)$$

Note that $\mathbf{U}$ has zero diagonal elements if the basis function is chosen such that $\varphi(0) = 0$, which may require special considerations regarding the existence and uniqueness of a solution to the system (3.51). Nevertheless, by substituting the solution of (3.51) into (3.50), the RBF interpolation on the target mesh, and so the solid-to-fluid interpolation operator, can be expressed as:

$$\mathbf{u}_F = \mathbf{B}\mathbf{A}^{-1} \begin{pmatrix} \mathbf{u}_s \\ \mathbf{0} \end{pmatrix} = \mathcal{V}_{\text{SF}}(\mathbf{u}_s) \quad (3.53)$$

In the literature, this method is referred to as **consistent** interpolation because it ensures the exact interpolation of a constant field [193]. An important note is that in standard FEM, without remeshing or material rupture, the inversion of $\mathbf{A}$ could be constructed once, based on the initial configuration, and the simple matrix product (3.53) performed whenever interpolation is needed.

<i>Support</i>	<i>Acronym</i>	<i>Full name</i>	$\varphi(\varepsilon)$
$\varepsilon = \frac{r}{R} \leq 1$	C0	Wendland function	$(1 - \varepsilon)^2$
	C2	Wendland function	$(1 - \varepsilon)^4(4\varepsilon + 1)$
$\varepsilon = \frac{r}{R}$	GS	Gaussian	$\exp(-\varepsilon^2)$
	IMQ	Inverse multiquadratic	$1/\sqrt{1 + \varepsilon^2}$
	MQ	Multiquadratic	$\sqrt{1 + \varepsilon^2}$
	TPS	Thin plate spline	$\varepsilon^2 \ln \varepsilon$

Table 12: List of common compactly and globally supported **basis functions** for RBF interpolation. The variable r is the distance to the center of the function and R is a characteristic radius. Compactly supported functions return zero when $r \geq R$.

However, in a PFEM framework, the number of nodes at the FS interface can change over time due to remeshing and the adaptive refinement of wetted boundaries. As a result, the interpolation matrices must be recomputed whenever the FS interface mesh has been modified by the remeshing. Therefore, it is preferable to avoid the explicit computation of $\mathbf{A}^{-1}$ as directly inverting a matrix is generally expensive. Moreover, if the original matrix is sparse, it typically leads to a loss of sparsity due to fill-in effects. Instead, only the intermediate matrices $\mathbf{A}$ and $\mathbf{B}$ (or the factorization of $\mathbf{A}$ if a direct solver is used) are actually stored in memory, and each call to the interpolation operator results in solving the following linear system:

$$\mathcal{V}_{\text{SF}}(\mathbf{u}_s) = \mathbf{B} \mathbf{A} \setminus \begin{pmatrix} \mathbf{u}_s \\ \mathbf{0} \end{pmatrix} \quad (3.54)$$

As shown in [190], the linear system in (3.54) can be highly ill-conditioned, with the matrix $\mathbf{A}$ potentially approaching singularity. In previous work, Thomas et al. [89] used a GMRES solver. However, the accuracy of the GMRES deteriorates for systems with high condition numbers. Moreover, its performance heavily depends on the choice of the preconditioner, which adds a layer of complexity and may still fail to produce a solution. In contrast, least-squares solvers based on the SVD decomposition inherently handle ill-conditioned systems by minimizing the residual norm and truncating small singular values. While this approach may increase computation time, it ensures more robust and consistent results across various configurations. Thereby, SVD is the default solver unless otherwise specified. It is worth reminding that the remeshing procedure in the PFEM-LS solver is triggered only at the end of each coupling time step, meaning that the interpolation matrices are constant during IQN iterations. The key advantage of using a direct solver for (3.54) rather than an iterative solver is that the factorization of $\mathbf{A}$ can be computed only once per interface update. This pre-factored matrix can then be reused for multiple interpolations (i.e. for each IQN iteration) at minimal cost, significantly reducing the overall execution time. More details about computation times are provided in a subsequent section.

Another important remark is that, in theory, the interpolation matrices need to be updated only when the remeshing makes topological changes to the FS interface (such as relocating, adding or deleting nodes), which may not occur at every time step. However, the indices of the nodes in the PFEM3D's internal database are not guaranteed to remain the same following remeshing.

Indeed, PFEM3D stores the nodes contiguously in memory, and uses a Reverse Cuthill-McKee (RCM) algorithm [194] on the nodes list to optimize memory traffic. This means that the interpolation matrices and their factorizations may become obsolete even if the topology of the FS interface remains unchanged. As a result, FSPC automatically detects whenever the FS interface nodes undergo modifications (removal, addition or reordering) following remeshing, and triggers the re-computation/factorization of interpolation matrices when needed.

Fluid Load Interpolation

Consistent interpolation of the nodal loads from the fluid to the solid meshes involves constructing $\mathbf{A}$ and $\mathbf{B}$ using the same process outlined above, by inverting the solid and fluid quantities and interpolating the components of the stress tensor at the interface, rather than the nodal displacement. The projection $\mathbf{t}_s = \boldsymbol{\sigma}_s \cdot \mathbf{n}_s$ of these interpolated nodal stresses onto the surface, along with their numerical integration over the solid elements, is then managed by the structural solver. This approach eliminates the need to approximate nodal tractions on the fluid side. Consequently, the interpolation becomes

$$\boldsymbol{\sigma}_s = \mathcal{V}_{\text{FS}}(\boldsymbol{\sigma}_f) \quad (3.55)$$

On the other hand, the **conservative** approach is based on the global conservation of virtual work across the FS interface [22, 193]. This approach is rooted in the general principle that energy should be conserved over the interface. In the limit of small time steps, energy is globally conserved over the interface when

$$\int_{\Gamma_s} \mathbf{u}_s^\top \mathbf{t}_s d\mathbf{x} = - \int_{\Gamma_f} \mathbf{u}_f^\top \mathbf{t}_f d\mathbf{x} \quad (3.56)$$

Here, $\mathbf{u}_s$ is a virtual displacement, namely, an arbitrary infinitesimal variation, and Γ_s is the FS interface on the solid side. Similarly, $\mathbf{u}_f$ is the same virtual displacement of the fluid interface Γ_f . In the context of discretized fields, the conservation of energy at the interface is equivalent to

$$\begin{aligned} \mathbf{u}_s^\top \mathbf{F}_s &= -\mathbf{u}_f^\top \mathbf{F}_f \\ &= -\mathcal{V}_{\text{SF}}(\mathbf{u}_s)^\top \mathbf{F}_f \end{aligned} \quad (3.57)$$

Here, $\mathbf{F}_s$ represents the nodal forces applied on the solid interface, and $\mathbf{F}_f$ represents the nodal forces on the fluid mesh. The second equality holds if we have access to an interpolation operator mapping the nodal displacement of the solid interface to nodal displacements of the fluid interface. Assuming the latter has been computed by the consistent RBF interpolation (3.53), one has

$$\mathbf{u}_s^\top \mathbf{F}_s = - \left\{ \mathbf{B} \mathbf{A}^{-1} \begin{pmatrix} \mathbf{u}_s \\ \mathbf{0} \end{pmatrix} \right\}^\top \mathbf{F}_f = - (\mathbf{H} \mathbf{u}_s)^\top \mathbf{F}_f \quad (3.58)$$

Here, $\mathbf{H}$ is defined as the sub-matrix corresponding to the first $N_f \times N_s$ block of $\mathbf{B} \mathbf{A}^{-1}$. Since this equality is valid for any kinematically admissible virtual displacements, the vectors of nodal displacement can be eliminated from the equation. As a result, one can show that energy is globally conserved when

$$\begin{aligned} \mathbf{u}_s^\top \mathbf{F}_s &= - (\mathbf{H} \mathbf{u}_s)^\top \mathbf{F}_f \\ \Leftrightarrow \mathbf{F}_s &= - \mathbf{H}^\top \mathbf{F}_f \end{aligned} \quad (3.59)$$

Consequently, the transposed of the transformation matrix $\mathbf{H}$ for the displacement can be used to exchange forces to obtain global conservation of energy over the interface. While this approach may seem advantageous at first glance, it requires the computation of nodal forces, and thus nodal normals, on the fluid side. These quantities are not naturally defined on geometric vertices, and their approximation in 3D meshes is both non-trivial and often inaccurate. Moreover, conservative interpolation has been shown to yield nonphysical results in various configurations. A comparison between conservative and consistent interpolations, and between different basis functions, is provided throughout this section. Additionally, the reader is referred to the work of de Boer et al. [193] for more information.

Side Remark

As a first remark, de Boer et al. [193] has shown that even if the consistent interpolation of the load is not strictly globally conservative of the energy transfer over the FS interface, the error on the transferred work decreases consistently with the characteristic size of the target mesh. In contrast, when the conservative approach is used, the resulting interpolation is never consistent.

As a second side note, it is worth observing from Table 12 that some basis functions decrease with respect to the normalized distance ε from their center, while others, such as the MQ function, increase as ε increases. The interpolation in (3.53) can be viewed as a weighted sum of the source field values at the j -th node in the source mesh, for each target node in the target mesh. The interpolation weights are implicitly determined by computing $\mathbf{B}\mathbf{A}^{-1}$ where the basis function is used to assemble both $\mathbf{B}$ and $\mathbf{A}$. However, using an increasing $\varphi(\varepsilon)$ does not mean that nodes farther from the target location receive higher interpolation weights. Instead, the solution to the linear system determines the interpolation weights, and their absolute values always decrease as the distance from the target location increases.

3.3.2 Numerical Examples

In this subsection, we evaluate the accuracy of the RBF methods by comparing the basis functions across various configurations. The first test case features the flow-driven deformation of a rubber gate in 2D, as described in the previous section. The second test case focuses on the flow-induced reconfiguration of buoyant, flexible aquatic vegetation in 3D, also introduced earlier. To assess the accuracy of the consistent interpolation, we project the nodal stresses onto the corresponding surface elements of the target mesh and compute the average error in pressure and shear relative to reference values obtained from the source mesh. For the conservative interpolation, which operates on nodal forces instead of nodal stresses, we first convert the forces into equivalent nodal tractions. These tractions, whose magnitudes are independent of mesh density, are then projected onto pressure and shear components for comparison with the reference field. The equivalent traction vector at the j -th interface node is calculated by

$$\mathbf{t}_j \simeq \mathbf{F}_j \left(\frac{1}{E_j} \sum_{e=1}^{E_j} \frac{S_{ej}}{N_e} \right)^{-1} \quad (3.60)$$

Here, S_{ej} denotes the size of the e -th boundary element connected to the j -th interface node, measured as area in 3D or length in 2D. N_e is the number of nodes associated with that element, and E_j is the total number of boundary elements connected to the j -th node. It is important to emphasize that these equivalent tractions represent an averaged form of the actual surface tractions. As such, they are less accurate and are intended only for estimating interpolation error across varying mesh densities. The average error relative to a reference solution $f(\mathbf{x}, t)$ is then defined by

$$\text{Error}(t) = \left\{ \int_{\Gamma} [f(\mathbf{x}, t) - g(\mathbf{x}, t)]^2 d\mathbf{x} \right\} \left\{ \int_{\Gamma} f(\mathbf{x}, t)^2 d\mathbf{x} \right\}^{-1} \quad (3.61)$$

where $\mathbf{x}$ is the parametric coordinate of the fluid-structure interface Γ . The function g represents the output of the interpolation, expressed as a continuous field on the target mesh. It is evaluated at any position on the interface using the nodal values and the basis functions of the finite elements. To assess the accuracy of an interpolation method during a simulation, the error (3.61) is averaged over time.

Rubber Gate

At the FS interface, the characteristic element size is set to $h_F = 1$ mm on the fluid side. On the solid side, three levels of mesh refinement are considered (see Figure 62). The fine mesh has $h_s = 0.4$ mm, the medium mesh has $h_s = 1.34$ mm, and the coarse mesh has $h_s = 5.27$ mm, resulting in three non-conforming discretizations. For each mesh, the basis functions listed in Table 12 are tested. Since the basis functions depend on the characteristic radius R , a wide range of values is included in the study. Specifically, 40 values of R are considered, covering the range $0.01 \leq R/h_F \leq 20$ for each RBF interpolation. The results are stored at each time step, and the average interpolation error relative to the reference solution on the source mesh is computed. To facilitate analysis, a non-dimensional characteristic radius is defined as follows:

$$\text{Normalized radius} = \frac{R}{h_F} \quad (3.62)$$

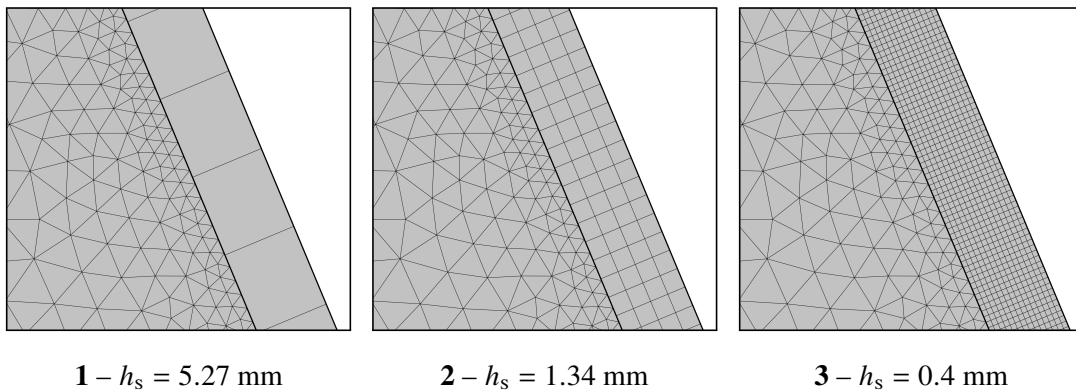


Figure 62: Zoom on the fluid-structure interface mesh for the **rubber gate** test case whose initial configuration is given in Figure 55, with different refinement levels. The mesh on the fluid side remains identical in all cases.

When $h_s = 5.27$ mm, the fluid interface mesh is significantly finer than the solid interface mesh. The time-averaged errors for the pressure field p and the shear field τ are presented in Figure 63. The normalized radii that minimize the error are listed in Table 13. Finally, Figure 64 shows an overview of the pressure field on the target mesh at the end of the simulation. These results illustrate that, with a low target mesh resolution, the conservative interpolation helps to maintain the conservation of total work on the coarse interface by redistributing additional loads to neighboring nodes. In contrast, local variations in the fluid load may be lost during consistent interpolation due to the insufficient mesh resolution, leading to a higher error on the interpolated pressure.

As an important remark, a higher error bound is observed for the shear field compared to the pressure field. This discrepancy stems from the non-matching meshes, which cause slight differences in the orientations of normal and tangent vectors between fluid-side elements and their corresponding solid-side elements. In the conservative interpolation approach, this mismatch is further amplified by the need to approximate nodal normals on the fluid side. In this simulation, the fluid loads are dominated by pressure, with the average pressure exerted by the fluid being 3 orders of magnitude higher than the average shear. As a result, the shear stress on the solid mesh is highly sensitive to small changes in orientation, but its impact on the deflection of the obstacle remains negligible.

<i>Approach</i>	<i>Solid</i>	<i>C0</i>	<i>C2</i>	<i>GS</i>	<i>IMQ</i>	<i>MQ</i>	<i>TPS</i>
	h_s [mm]	R/h_F					
Conservative	5.27	20	15	5	5	2	9
	1.34	7	13	2	2	1	0.15
	0.4	0.01	0.01	0.01	0.01	0.01	0.2
Consistent	5.27	6	5	2	5	3	0.01
	1.34	7	20	2	5	0.2	0.01
	0.4	20	4	0.7	0.55	0.1	8

Table 13: Optimal choice of normalized radius R/h_F leading to the minimal interpolation error for the **rubber gate** test case, for different refinement levels of the solid mesh. The optimal radii have been deduced from Figures 63, 65 and 67 presented hereafter.

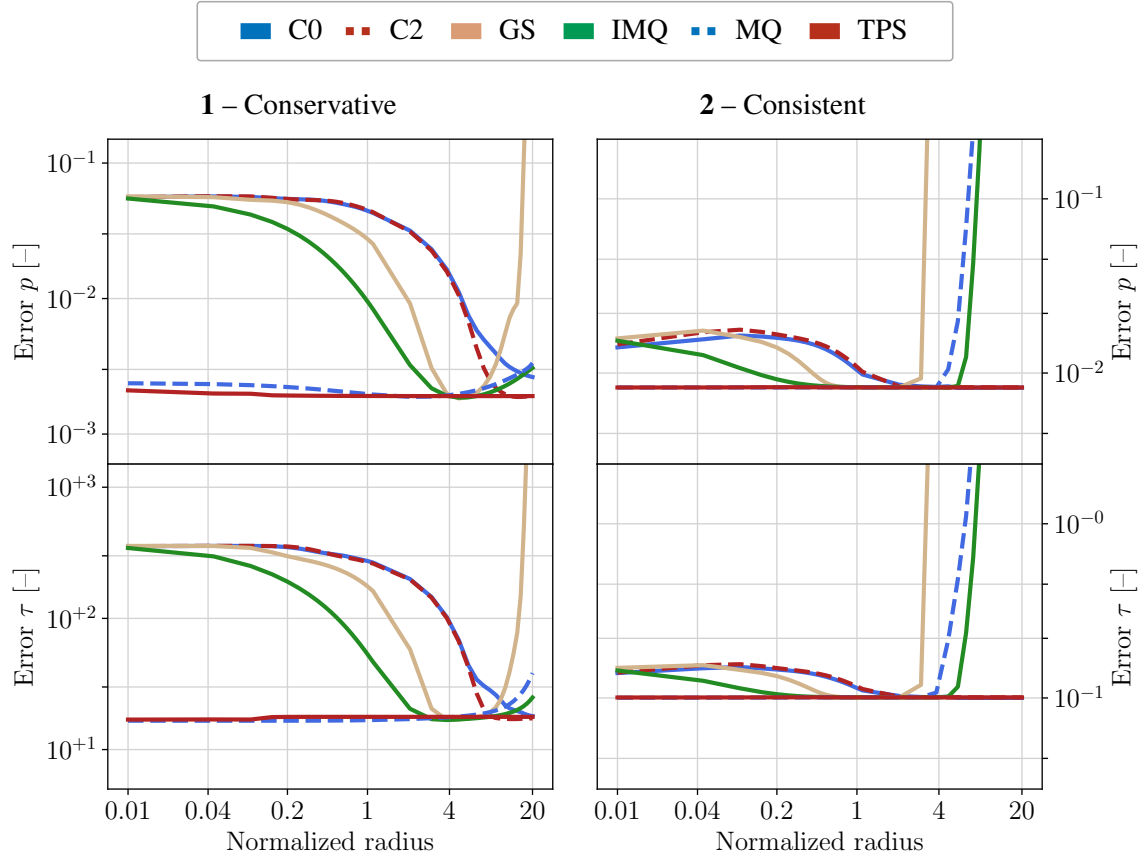


Figure 63: Average interpolation error for the **rubber gate** test case, as a function of the RBF characteristic radius for different basis functions. The solid structure is discretized with $h_s = 5.27$ mm.

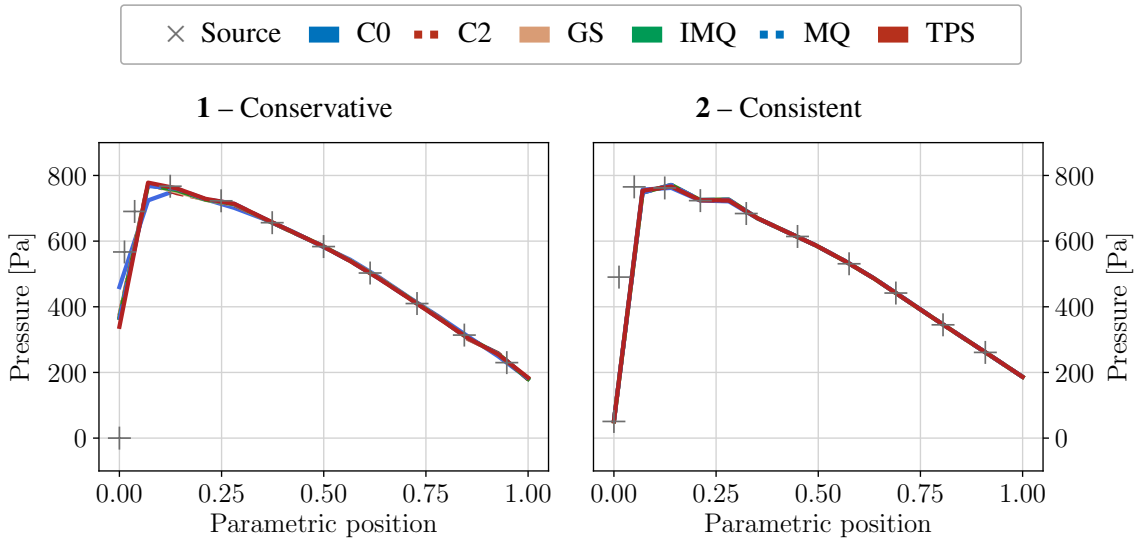


Figure 64: Interpolated pressure profile on the FS interface of the **rubber gate** at time $t = 0.4$ seconds. The optimal R from Table 13 have been used and the solid structure is discretized with $h_s = 5.27$ mm.

Nevertheless, all basis functions show comparable performance when the optimal characteristic radius is selected. Notably, the TPS is nearly insensitive to R and consistently produces optimal results. This robustness makes the TPS especially advantageous in practice, as it eliminates the need for precise tuning of the characteristic radius across different configurations. Additionally, it maintains reliable performance during dynamic mesh refinement (i.e. changes in characteristic size) due to the remeshing process. In contrast, the C2 function offers a viable alternative to the TPS, achieving optimal performance with a sufficiently large R , rather than requiring a limited range of values. Its main advantage lies in its compact support, which makes the matrix $\mathbf{A}$ sparse when the ratio between the total size of the interface and the characteristic size of the elements increases.

When $h_s = 1.34$ mm, the difference between the characteristic sizes of the solid and fluid meshes is small, meaning the solid mesh is sufficiently refined to capture local variations in the fluid loads. The time-averaged errors for the interpolated pressure and shear fields are shown in Figure 65. A slightly higher error on the pressure is observed for the conservative interpolation. The latter arises from nonphysical oscillations of the load around the reference solution, as illustrated in Figure 66. On the other hand, the consistent interpolation displays a stable, non-oscillating pressure field across the interface. Similar results have been reported in [193] for quasi-1D problems.

When $h_s = 0.4$ mm, the solid mesh is more refined than the fluid mesh at the interface. The average interpolation errors are shown in Figure 67. The nonphysical oscillations previously observed for $h_s = 1.34$ mm with conservative interpolation are amplified, as seen in Figure 68. This occurs because the conservative interpolation tends to produce concentrated loads on solid nodes near fluid nodes at the interface, and let the other nodes unloaded. With smaller solid elements, the magnitude of these concentrated loads increases to conserve the total energy of the interface. Evidence of this behavior is observed in Figure 68. On the other hand, the consistent interpolation still displays a stable, non-oscillating pressure field across the interface.

It is important to note that the seemingly smaller error for small R observed with the conservative interpolation in Figure 67 is misleading. In fact, the interpolation result at $R/h_F = 0.01$ is unrepresentative of the source field. However, in this specific scenario, the solution appears, on average, closer to the reference solution than the concentrated loads shown in Figure 68. As a result, the error appears deceptively smaller when evaluated using the measure (3.61). This clearly highlights a limitation of the chosen error metric, suggesting that a more sophisticated measure is needed to better discriminate such cases.

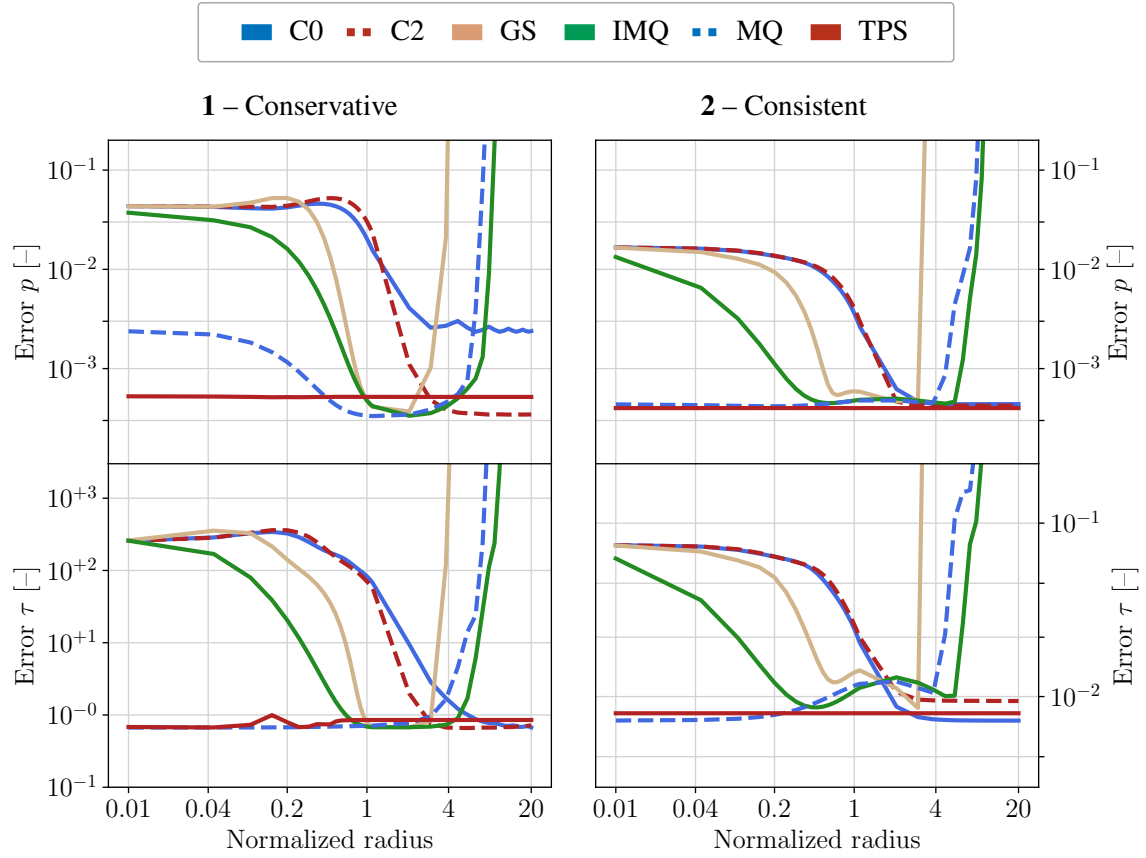


Figure 65: Average interpolation error for the **rubber gate** test case, as a function of the RBF characteristic radius for different basis functions. The solid structure is discretized with $h_s = 1.34$ mm.

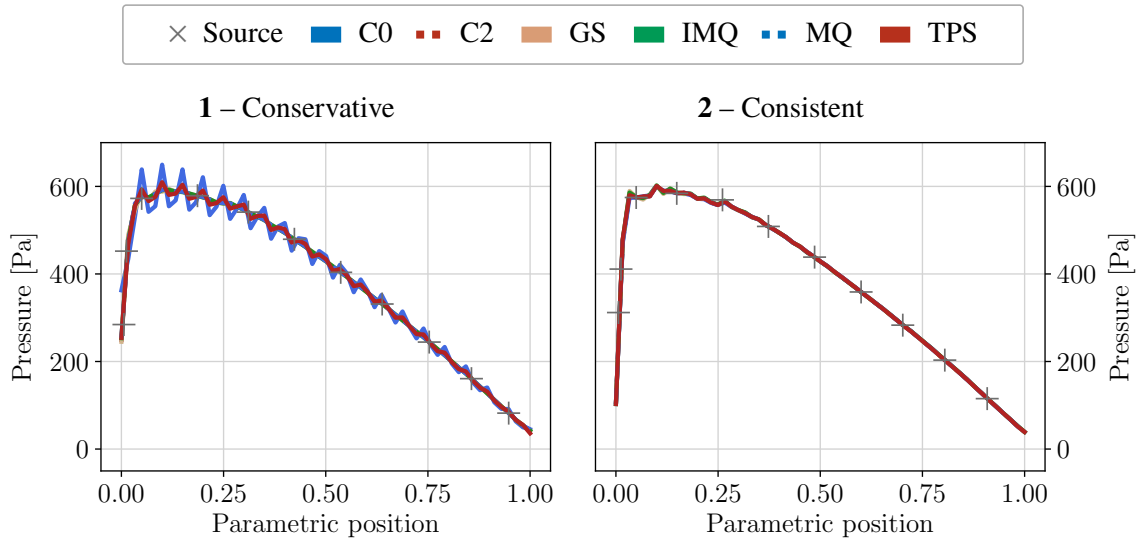


Figure 66: Interpolated pressure profile on the FS interface of the **rubber gate** at time $t = 0.4$ seconds. The optimal R from Table 13 have been used and the solid structure is discretized with $h_s = 1.34$ mm.

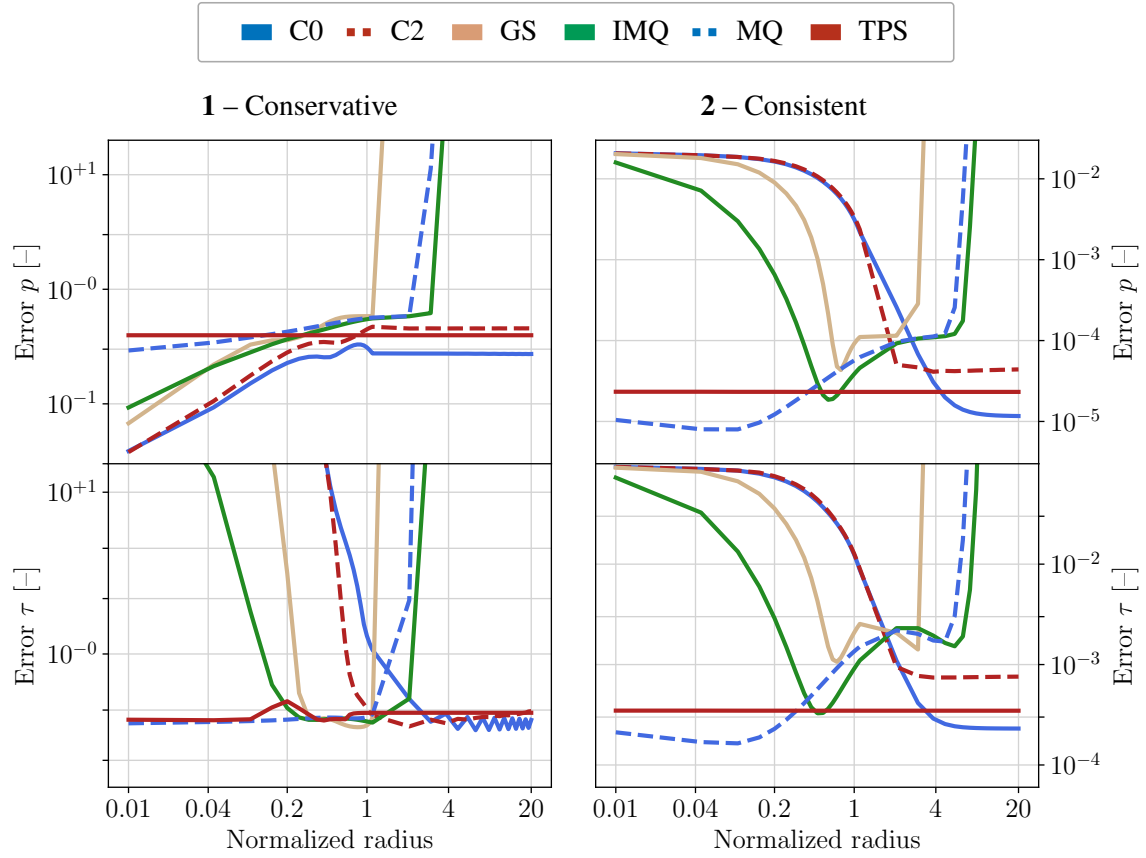


Figure 67: Average interpolation error for the **rubber gate** test case, as a function of the RBF characteristic radius for different basis functions. The solid structure is discretized with $h_s = 0.4$ mm.

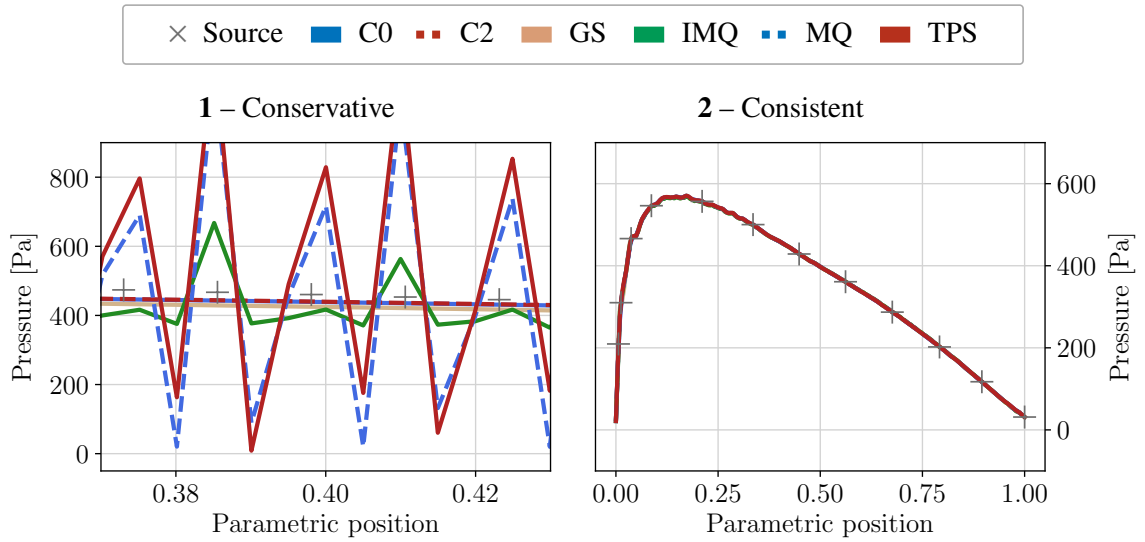


Figure 68: Interpolated pressure profile on the FS interface of the **rubber gate** at time $t = 0.4$ seconds. The optimal R from Table 13 have been used and the solid structure is discretized with $h_s = 0.4$ mm.

As a side remark, the accuracy of the interpolation (as well as the computation time) can also be influenced by the condition number of the system. As shown in Figure 69, the condition number of $\mathbf{A}$ is generally high and varies by several orders of magnitude depending on the characteristic radius used. Specifically, for the TPS basis function, it has been demonstrated that R does not significantly affect the accuracy of the interpolation within the range $0.01 \leq R/h_F \leq 20$, provided the solution to Equation (3.54) is computed using a robust least squares solver. However, the condition number of $\mathbf{A}$ increases drastically for smaller values of R , which may pose a challenge to most linear solvers. A more detailed analysis of computation times is presented in a subsequent section.

Aquatic Vegetation

This test case revisits the flow-induced reconfiguration of buoyant and flexible aquatic vegetation, modeled by the 3D bending of an elastic flap in cross flow, as presented in the previous subsection (Figure 59). The test follows the same methodology as before. Specifically, interpolation errors for the pressure and shear fields on the leading face are evaluated for each basis function across the range $0.01 \leq R/h_F \leq 20$. To simplify the analysis of shear stresses, only the error for the shear component perpendicular to the y -axis is considered in the figures.

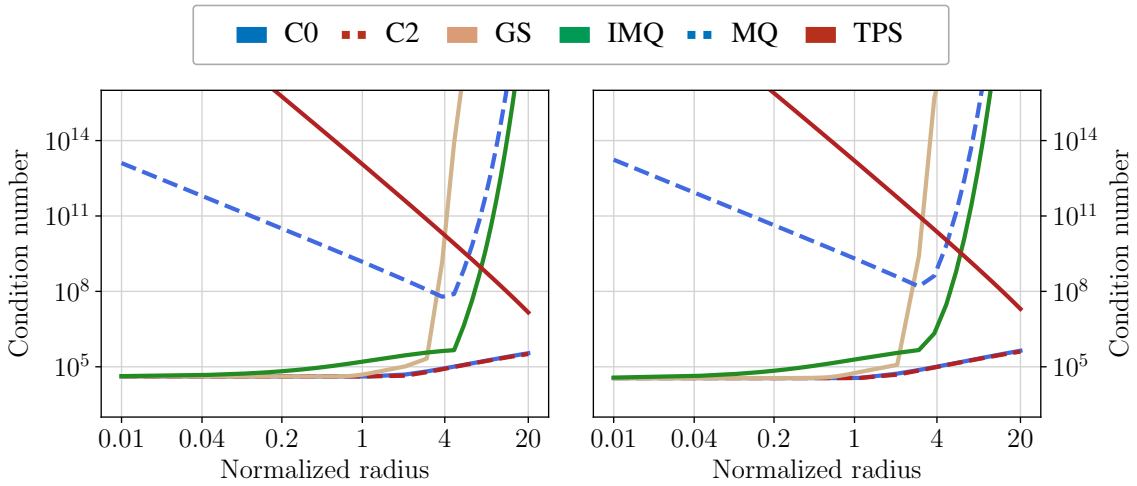


Figure 69: Condition number of $\mathbf{A}$ for the **rubber gate** test case in 2D. The first figure holds for the interpolation matrix $\mathbf{A}_{FS}$ and the second figure holds for the interpolation matrix $\mathbf{A}_{SF}$ with $h_s = 1.34$ mm at the end of the simulation.

In this analysis, the fluid mesh consists of tetrahedral elements with a characteristic size of $h_F = 2$ mm at the FS interface. The solid mesh is composed of regular hexahedral elements with two levels of refinement. The medium-resolution mesh uses $h_s = 2.5$ mm, resulting in comparable characteristic sizes for the fluid and solid meshes at the interface, although the solid elements are slightly larger, as shown in Figure 70. The interpolation error as a function of the characteristic radius is shown in Figure 71, with optimal parameters listed in Table 14.

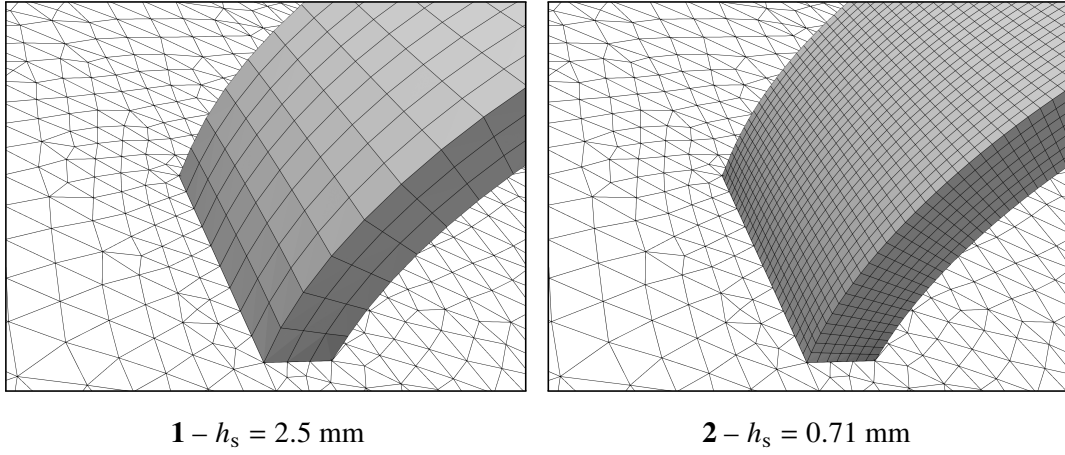


Figure 70: Zoom on the fluid-structure interface mesh for the **aquatic vegetation** test case whose initial configuration is given in Figure 59, with different refinement levels. The mesh on the fluid side remains identical in all cases.

<i>Approach</i>	<i>Solid</i>	<i>C0</i>	<i>C2</i>	<i>GS</i>	<i>IMQ</i>	<i>MQ</i>	<i>TPS</i>
	h_s [mm]	R/h_F					
Conservative	2.5	19	3	1	1	0.8	0.7
	0.71	0.01	0.01	18	0.01	17	0.25
Consistent	2.5	20	5	0.9	1	0.95	0.75
	0.71	8	2	0.7	0.55	0.1	0.15

Table 14: Optimal choice of normalized radius R/h_F leading to the minimal interpolation error for the **aquatic vegetation** test case, for different refinement levels of the solid mesh. The optimal radii have been deduced from Figures 71 and 73 presented hereafter.

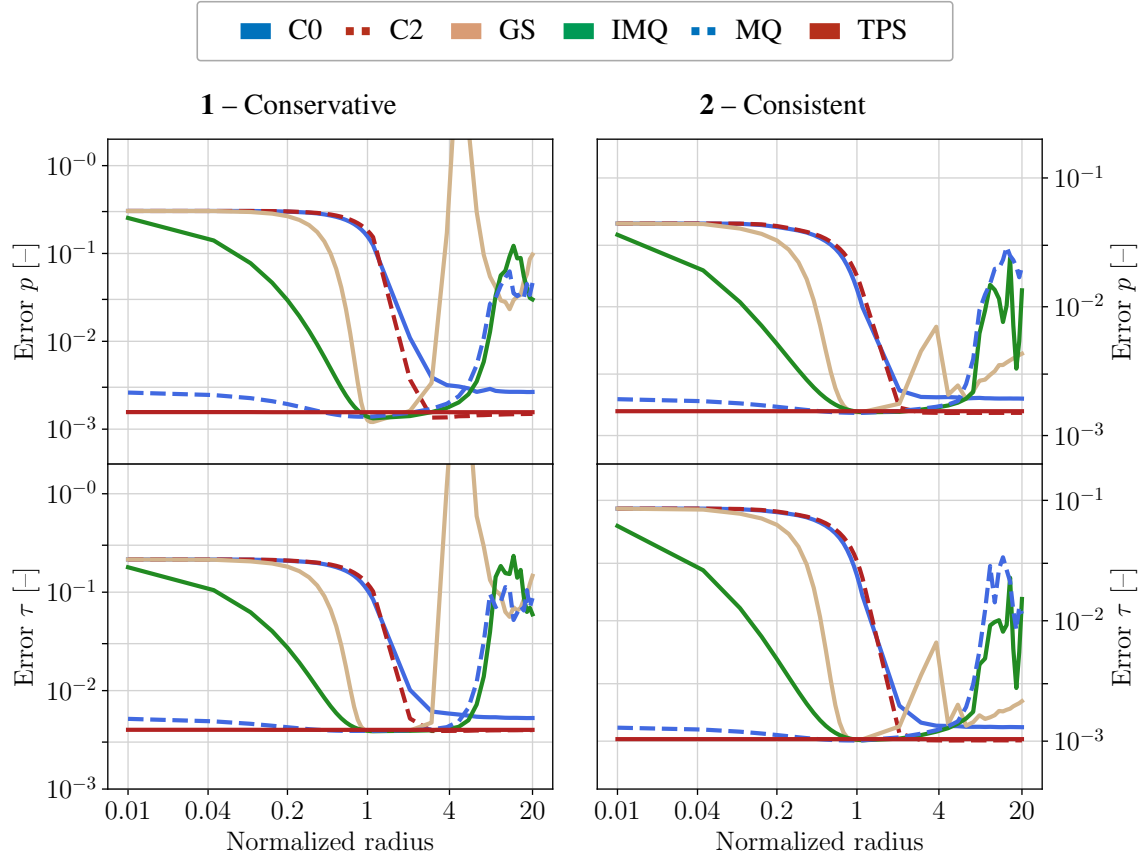


Figure 71: Average interpolation error for the **aquatic vegetation** test case, as a function of the RBF characteristic radius for different basis functions. The solid structure is discretized with $h_s = 2.5$ mm.

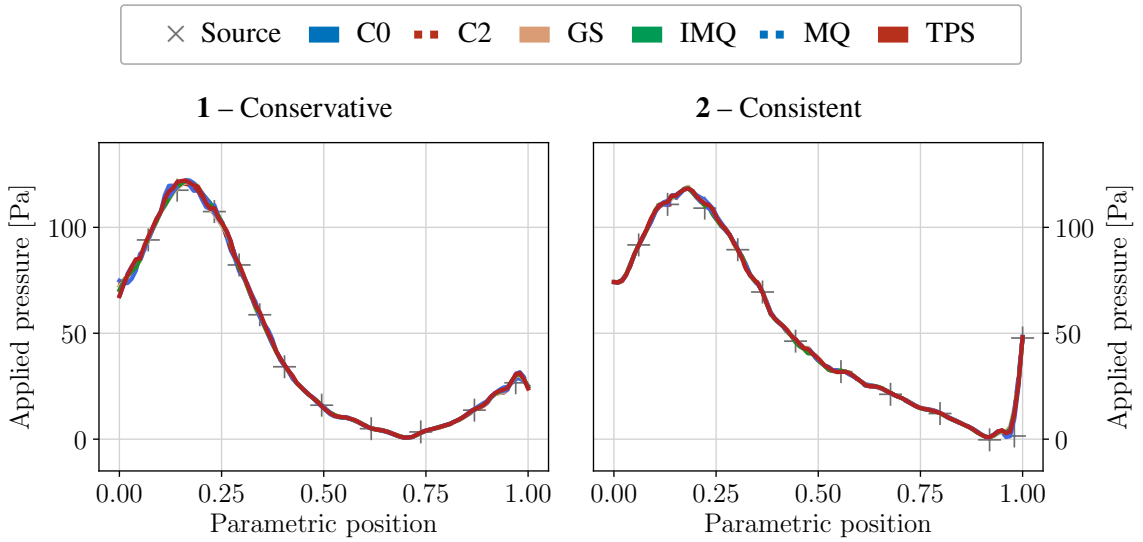


Figure 72: Interpolated pressure profile on the FS interface of the **aquatic vegetation** at time $t = 2.0$ seconds. The optimal R from Table 14 have been used and the solid structure is discretized with $h_s = 2.5$ mm.

With the medium solid mesh, both consistent and conservative interpolations perform well across all basis functions, provided the characteristic radius R is appropriately selected. As in the 2D case, the TPS yields optimal results regardless of R , while the C2 basis function achieves satisfactory accuracy when $R/h_F > 6$. Recall that this test case involves a highly viscous fluid, resulting in average shear and pressure fields of similar magnitude applied to the structure. As a result, the interpolation errors for both the shear and pressure fields are closely comparable. An overview of the interpolated pressure on the solid interface at the end of the simulation is shown in Figure 72. As a reminder, in the conservative case, the pressure and shear fields on the solid interface are approximated from the nodal forces (for the purpose of estimating the interpolation error), whereas in the consistent case, they are directly computed from the stress tensor components. Consequently, the minimal error bounds for each framework may differ.

With a fine solid mesh ($h_s = 0.71$ mm), the configuration becomes susceptible to non-physical oscillations in the nodal load when using conservative interpolation. The interpolation error as a function of the characteristic radius is shown in Figure 73. This simulation extends the validation performed in the 2D case to a 3D context. As expected, and confirmed, the conservative interpolation fails to accurately capture fluid loads on the solid mesh when the solid is equally or more finely resolved than the fluid mesh, even when the characteristic radius is optimized, as shown in Figure 74. In contrast, the consistent interpolation method delivers accurate and reliable results.

The interpolation matrices in the 3D case exhibit characteristics similar to those observed in the 2D test. Specifically, global support basis functions produce a dense linear system, whereas compact support functions can yield a sparse system. For instance, using the C2 basis function with $R/h_F = 6$, the sparsity of $\mathbf{A}_{FS}$ reaches 87%. Additionally, the condition number of $\mathbf{A}$ is, on average, one order of magnitude lower than in the 2D case, as shown in Figure 75. This improvement arises because the condition number of $\mathbf{A}$ increases significantly when the nodes defining the FS interface are regularly aligned on both sides. Such alignment is common in 2D when the characteristic element size is uniform across the interface, resulting in evenly spaced nodes, at least initially. In contrast, unstructured tetrahedral meshes in 3D produce a less regular node distribution at the interface, which helps to reduce the condition number of $\mathbf{A}$.

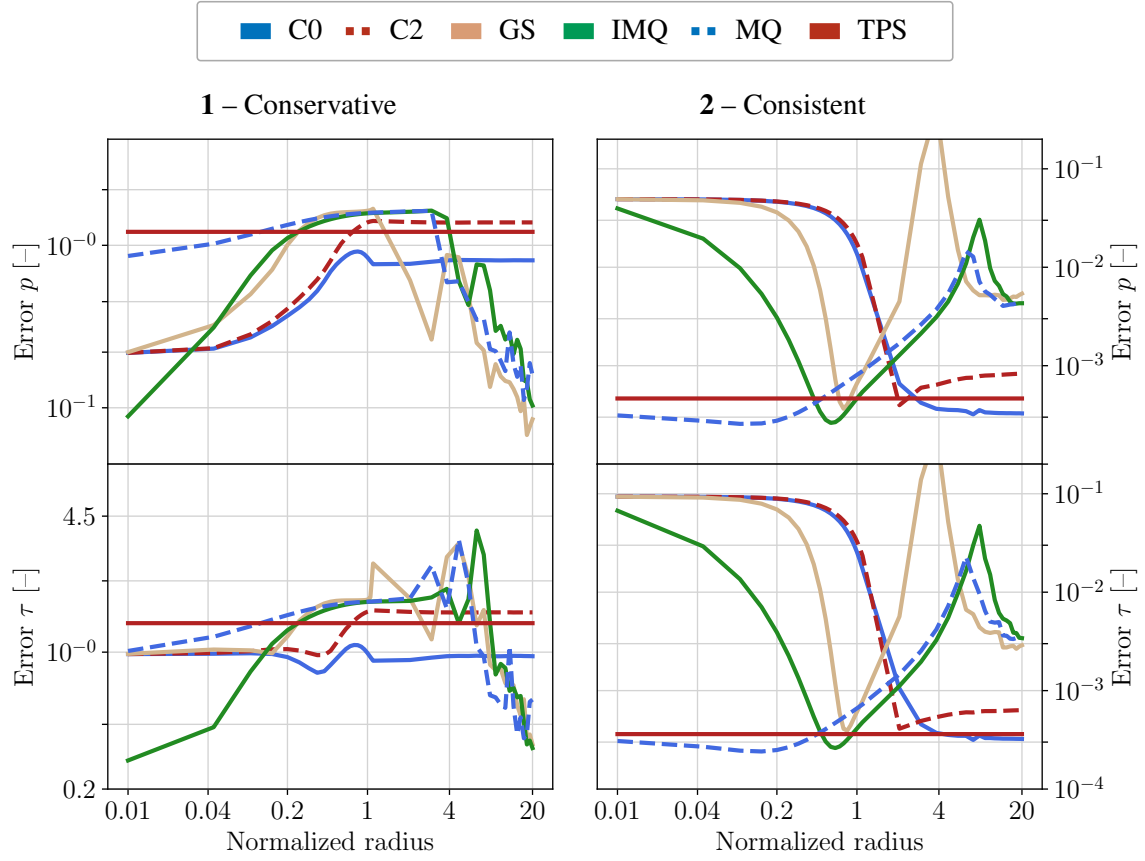


Figure 73: Average interpolation error for the **aquatic vegetation** test case, as a function of the RBF characteristic radius for different basis functions. The solid structure is discretized with $h_s = 0.71$ mm.

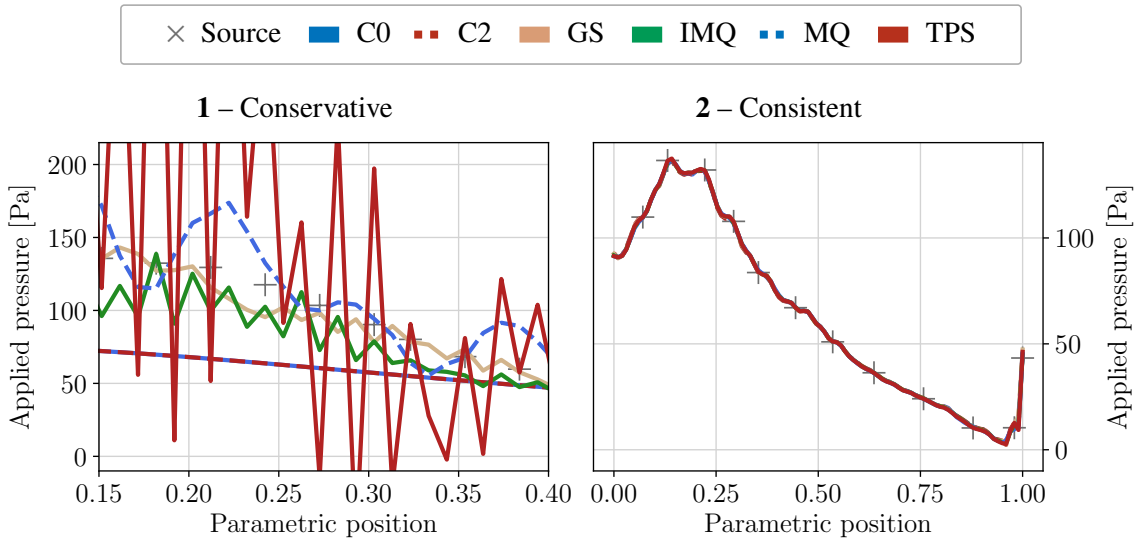


Figure 74: Interpolated pressure profile on the FS interface of the **aquatic vegetation** at time $t = 2.0$ seconds. The optimal R from Table 14 have been used and the solid structure is discretized with $h_s = 0.71$ mm.

3.3.3 Conclusion

The first major conclusion is that the consistent approach should generally be preferred, as it avoids the nonphysical oscillations in interpolated nodal forces that can arise when the target mesh is finer than, or equally refined to, the source mesh. The only notable exception occurs when the solid mesh is insufficiently refined to capture the local variations of the applied fluid load. In such cases, the conservative interpolation can help mitigate the loss of information by redistributing additional loads to nearby nodes, ensuring the conservation of total work.

The second major conclusion concerns the choice of the basis function. Our results show that the TPS function, particularly when the resulting linear system is solved using a solver able to handle ill-conditioned matrices, exhibits the most robust behavior. In particular, the TPS consistently delivers accurate results, regardless of the characteristic radius R/h_F chosen within the range 0.01 to 20. Additionally, increasing R tends to improve the condition number of $\mathbf{A}$ for the TPS. On the other hand, the C2 function offers a compelling alternative in large 3D scenarios. Its compact support enables the construction of sparse linear systems for large meshes, unlike globally supported RBF. The test cases demonstrated that the C2 function achieves acceptable accuracy when $R/h_F > 6$.

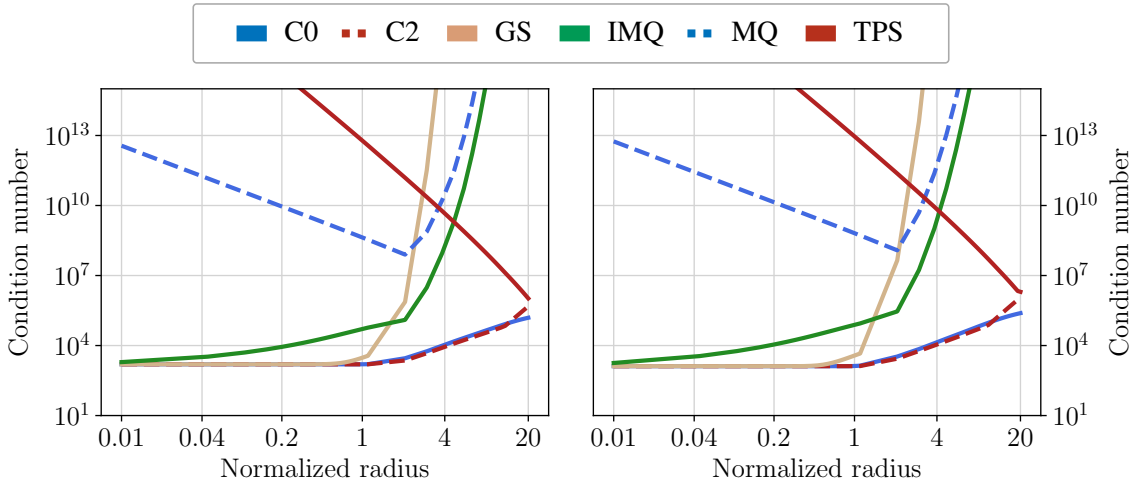


Figure 75: Condition number of $\mathbf{A}$ for the **aquatic vegetation** test case in 3D. The first figure holds for the interpolation matrix $\mathbf{A}_{FS}$ and the second figure holds for the interpolation matrix $\mathbf{A}_{SF}$ with $h_S = 2.5$ mm at the end of the simulation.

3.4 Thermo-Mechanical Interaction

So far, we have discussed purely mechanical cases of FSI coupling. However, both Metafor and PFEM3D are also capable of simulating thermal effects. In this section, we extend the coupling algorithms to include thermo-mechanical interactions. Thermo-mechanical FSI generally involves coupling the mechanical displacement with the temperature at the FS interface between the structural and fluid solvers at each time step. Alternatively, pure thermal coupling can be used when the interface remains stationary while temperature-related quantities are exchanged between the solvers through the Python interface. In this scenario, motion may still occur within the fluid domain or within the solid domains, but the FS interface itself remains fixed. FSPC supports both coupling options, allowing the user to independently activate or deactivate the thermal and mechanical coupling processes.

By analogy with the mechanical coupling, we define T_F as the temperature field seen by the fluid solver at the FS interface, and T_S as the temperature field seen by the solid solver at the same interface. Accordingly, $\mathbf{q}_F = -k_F \nabla T_F$ represents the heat flux computed by the fluid solver at the FS interface, while $\mathbf{q}_S$ is the heat flux on the solid side. The coupling conditions enforce the continuity of both temperature and heat flux at the interface between the fluid and solid domains:

$$\begin{aligned} T_S &= T_F \\ \phi_S &= -\phi_F \end{aligned} \tag{3.63}$$

The normal heat flux is denoted by $\phi = \mathbf{q} \cdot \mathbf{n}$, and is defined as the dot product between the heat flux vector $\mathbf{q}$ and the outward unit normal vector $\mathbf{n}$ to the domain. The negative sign in the second equality arises because the unit normals $\mathbf{n}_S$ and $\mathbf{n}_F$ point in opposite directions at the interface, similarly to the mechanical coupling.

3.4.1 Partitioned Coupling Scheme

In the **Temperature-Forward Flux-Back** (TFFB) scheme, the fluid solver imposes the nodal temperature as Dirichlet boundary conditions on the solid interface, while the structural solver imposes the heat flux on the fluid interface as Neumann boundary conditions. This scheme is analogous to the ND coupling in mechanical problems, where the fluid solver controls the interface displacement.

On the other hand, the **Flux-Forward Temperature-Back** (FFTB) scheme is the thermal counterpart of the DN mechanical coupling. In this scheme, the structural solver provides the nodal temperatures at the interface as Dirichlet boundary conditions to the fluid solver. Using these conditions, the fluid solver computes the heat flux at the interface and passes it back to the structural solver as Neumann boundary conditions. The structural solver then uses this heat flux to update the interface temperatures, and the process repeats until system convergence. The FFTB scheme results in the following root-finding problem:

$$\begin{aligned} \mathbf{r} &= \mathcal{S}\mathcal{V}_{\text{FS}}\left\{\mathcal{F}\mathcal{V}_{\text{SF}}(\mathbf{T}_s)\right\} - \mathbf{T}_s = \mathbf{0} \\ &= \hat{\mathbf{T}}_s - \mathbf{T}_s = \mathbf{0} \end{aligned} \quad (3.64)$$

Here, $\mathbf{T}_s$ is the vector of predicted nodal temperatures provided by the coupling algorithm, and $\hat{\mathbf{T}}_s$ is the updated solid temperature returned by the solid solver based on the nodal heat flux computed by the fluid solver with $\mathbf{T}_s$ as Dirichlet BC. It is important to note that instead of directly transferring the normal heat flux ϕ from the fluid mesh to the solid mesh, we transfer the total heat flux. This approach avoids the need to estimate nodal normals, which are not clearly defined at the nodes of linear finite elements. The transferred heat flux $\mathbf{q}_s$ is then non-ambiguously projected onto the integration points of the solid's elements. A summary of different approaches is provided in Figure 76.

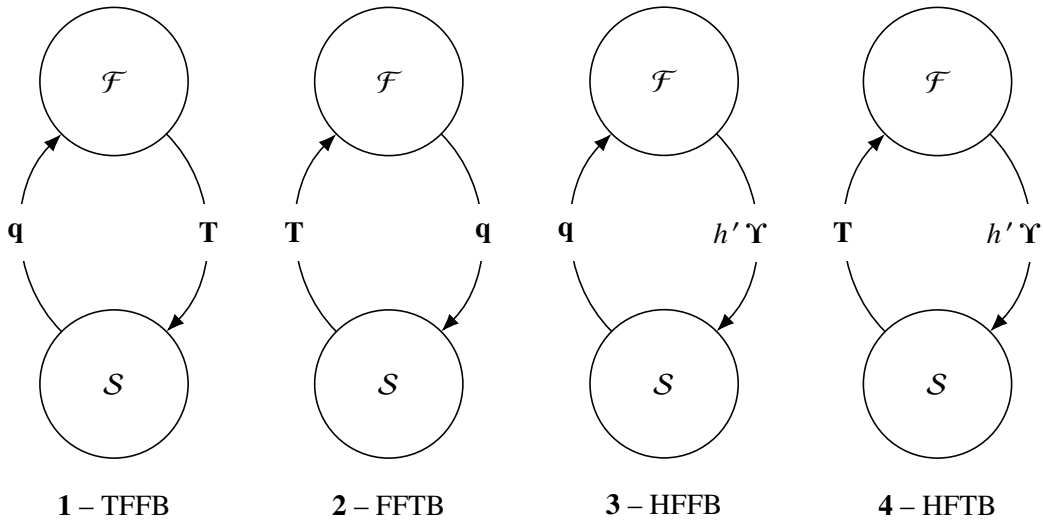


Figure 76: Schematic representation of different partitioned thermal coupling approaches. The HFFB and HFTB schemes indicated on the right, which are described hereafter for the sake of completeness, are not used in this work.

In this work, only the FFTB scheme has been implemented, as it is a straightforward adaptation of Algorithm 4, originally developed for mechanical coupling. The implementation involves substituting displacement with temperature and replacing stresses with heat flux. As a result, the problem (3.64) is solved iteratively at each time step using the same IQN algorithm as for the mechanical problem:

$$\begin{aligned}\mathbf{T}_{k+1} &= \mathbf{T}_k - \mathcal{J}(\mathbf{r}_k) \\ &= \mathbf{T}_k - \hat{\mathbf{J}}_k^{-1} \mathbf{r}_k + \mathbf{r}_k\end{aligned}\tag{3.65}$$

Here, the subscript S has been dropped for simplicity. The index k refers to the iteration of the IQN algorithm, and $\mathbf{r}_k$ represents the residual interface temperature at the k -th iteration. In the IQN-MVJ approach, the inverse of the modified Jacobian matrix $\hat{\mathbf{J}}_k^{-1}$ at each iteration is approximated by updating the inverse Jacobian $\hat{\mathbf{J}}_{\text{PRV}}^{-1}$ obtained at the end of the previous time step:

$$\hat{\mathbf{J}}_k^{-1} = \hat{\mathbf{J}}_{\text{PRV}}^{-1} + (\mathbf{W}_k - \hat{\mathbf{J}}_{\text{PRV}}^{-1} \mathbf{V}_k) \mathbf{V}_k^\dagger\tag{3.66}$$

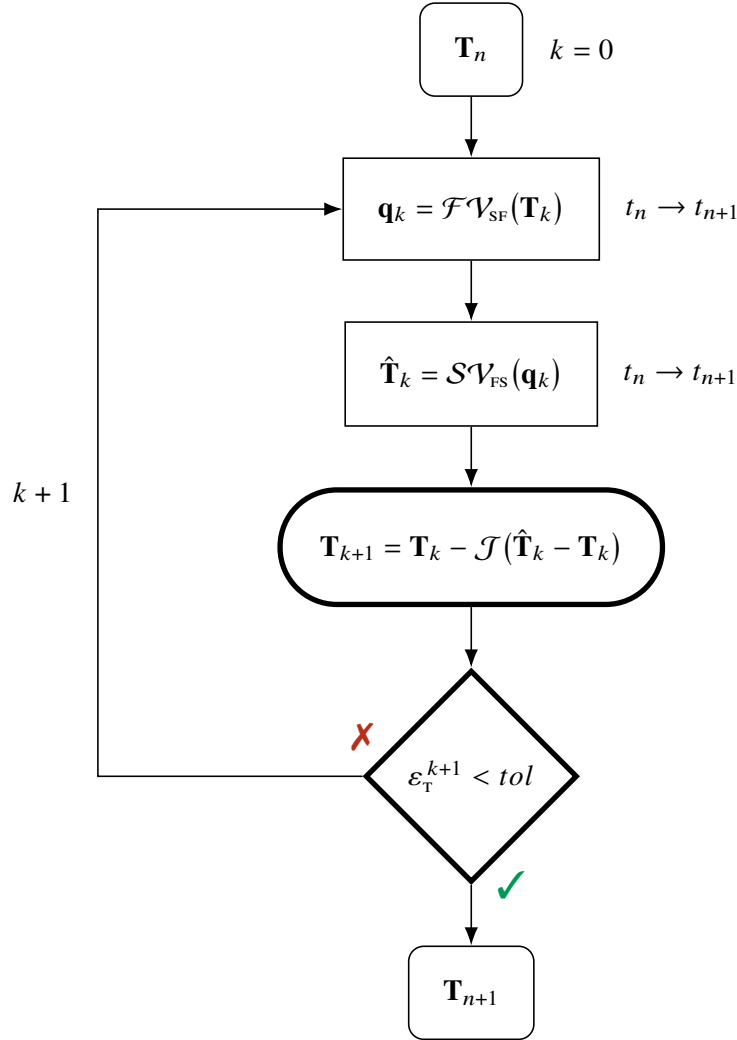
with $\text{col}_l(\mathbf{V}_k) = \mathbf{r}_l - \mathbf{r}_{l-1}$ and $\text{col}_l(\mathbf{W}_k) = \hat{\mathbf{T}}_l - \hat{\mathbf{T}}_{l-1}$ for $l = 1, \dots, k$ where the index l corresponds to the l -th IQN iteration of the current time step. If no previous Jacobian is available, it is set to zero and the IQN-ILS formulation is recovered. Since the temperature is a scalar field, there is only one degree of freedom per node. Therefore, $\mathbf{W}_k$ and $\mathbf{V}_k$ are rectangular matrices with k columns and N_s rows, where N_s is the number of interface nodes in the solid solver. This results in a significantly smaller system compared to the mechanical problem. The IQN-MVJ iterations are stopped once the following convergence criterion is satisfied:

$$\varepsilon_T^k = \frac{\|\mathbf{r}_k\|}{\|\text{tol} + \hat{\mathbf{T}}_k\|} < \text{tol}\tag{3.67}$$

Verstraete and Scholl [195] demonstrated that the FFTB scheme is unconditionally stable if and only if the Biot number satisfies $\text{Bi} < 1$. Conversely, the TFFB scheme is unconditionally stable if and only if $\text{Bi} > 1$. However, it is important to understand that these stability criteria were derived for a simple analytic 1D case. Previous works [30] have shown that convergence can still be achieved for both the FFTB and TFFB schemes outside their respective stability limits, when using the BGS coupling method with Aitken relaxation [196]. Recall that the Biot number is defined as

$$\text{Bi} = \frac{h}{k_s} L \quad (3.68)$$

Here, k_s is the thermal conductivity of the solid, h is the convective heat transfer coefficient, and L is the characteristic length of the geometry under consideration. Within the proposed coupling framework, heat fluxes and temperatures are directly exchanged at the FS interface, eliminating the need for an explicit definition of the heat transfer coefficient. Instead, it naturally emerges as an effective quantity based on the interface interactions, as resolved by the solvers. In the next subsection, we demonstrate that convergence can be achieved even beyond this stability limit using the IQN-MVJ method for unsteady regimes. A summary of the thermal coupling strategy is provided in Algorithm 5.



Algorithm 5: Strategy for solving partitioned fluid-structure coupling problems with a FFTB scheme, between time t_n and time t_{n+1} . Recall that each call to $\mathcal{S}$ and $\mathcal{F}$ involves the solving of a FE system within the current time step.

The HFFB and HFTB Schemes

As a side remark, two alternative approaches can be found in the literature, the Heat transfer coefficient-Forward Flux-Back (HFFB) and Heat transfer coefficient-Forward Temperature-Back (HFTB) schemes. Both are based on the same principle [197, 198]. At the discrete level of the coupled problem, these schemes impose a convective heat transfer relationship through a Robin boundary condition that combines heat flux and temperature. In these approaches, an equivalent temperature Υ is computed on the fluid side of the interface. The Newton's law of cooling is then applied on the structural side to compute a new heat flux, which is used as input for the structural solver:

$$\begin{aligned}\Upsilon &= T_F - \frac{\mathbf{q}_F \cdot \mathbf{n}_F}{h'} \\ \mathbf{q}_s \cdot \mathbf{n}_s &= (T_s - \Upsilon) h'\end{aligned}\tag{3.69}$$

Here, h' represents a numerical heat transfer coefficient, typically estimated using the fluid temperature from the first cells normal to the interface. Alternatively, a simpler approach involves imposing a constant h' whose optimal value is not known a priori and must be adapted to the specific test case. In practice, the value of h' primarily affects the convergence rate of the coupling algorithm, without influencing the final result [195]. The resulting scheme is classified as either HFFB or HFTB, depending on the quantity provided by the solid solver and subsequently used as input by the fluid solver, as shown in Figure 76. If we define the numerical Biot number $\text{Bi}' = \text{Bi}(h = h')$ associated with the numerical heat flux, Verstraete and Gimenez [195, 199] demonstrated that convergence is guaranteed if

$$\begin{aligned}\text{HFFB} : \left| \frac{1}{\text{Bi}} \frac{\text{Bi}' - \text{Bi}}{\text{Bi}' + 1} \right| &< 1 \\ \text{HFTB} : \left| \frac{\text{Bi}' - \text{Bi}}{\text{Bi}' + 1} \right| &< 1\end{aligned}\tag{3.70}$$

In practice, Thomas [30] has shown that the HFFB and HFTB schemes do not offer significant advantages over the classical ND and DN approaches, even when the coefficient h' is well-tuned. In fact, the average number of iterations required to achieve the desired coupling tolerance with the HFFB and HFTB schemes is, at best, comparable to that of the TFFB and FFTB schemes, respectively.

3.4.2 Numerical Examples

This section presents the study of three different test cases. The first case involves conjugate heat transfer in a hollow cylinder immersed in a laminar cross-flow. The cylinder is treated as a rigid, fixed body, so the FSI is limited to thermal exchange with the surrounding fluid. The main goal of this simulation is to assess the convergence and stability of the FFTB coupling with IQN-MVJ. This is done by varying the solid conductivity for two different Biot numbers, while keeping all other parameters constant. The remaining two test cases are used to validate the results against existing literature in both 2D and 3D.

Hollow Cylinder

An overview of the initial configuration for the hollow cylinder in a cross-flow is shown in Figure 77. The solid conductivity is marked with an asterisk in Table 15 because it can take different values in this example. To determine the conductivity that results in the desired Biot number, one can substitute the known parameters from Table 15 into the classical definition of the Biot number (3.68):

$$\text{Bi} = \frac{hL}{k_s} \quad \Leftrightarrow \quad k_s = \frac{hL}{\text{Bi}} \quad (3.71)$$

The characteristic length L in most problems is defined as the ratio between the solid volume and the cooled surface area [200]. For a 2D cylinder with radius R , the characteristic length is given by $L = \pi R^2 / (2\pi R) = R/2$. The convective heat transfer coefficient h arises naturally as an effective quantity, determined by the interface interaction resolved by the solvers, and is therefore not known a priori. To estimate the value of h , one can use the definition of the Nusselt number:

$$\text{Nu} = \frac{hD}{k_F} \quad \Leftrightarrow \quad h = \frac{\text{Nu} k_F}{D} \quad (3.72)$$

Here, D is the outer diameter of the hollow cylinder. The Nusselt number represents the ratio between the total heat transfer and the conductive heat transfer at a boundary in a fluid, but it is not known a priori either. However, Churchill and Bernstein [201] proposed an empirical relation to calculate the average Nusselt number for forced convection over a cylinder in cross-flow:

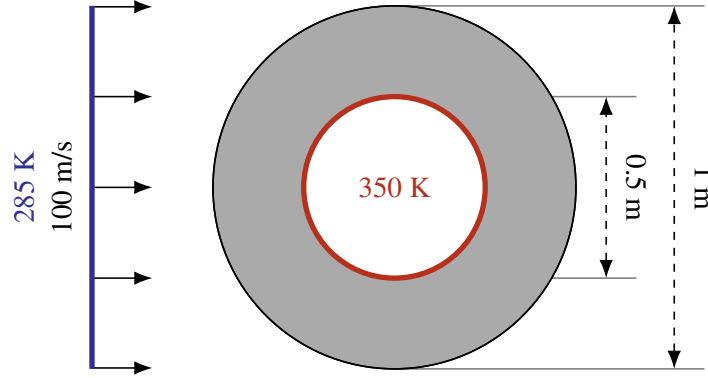


Figure 77: Initial configuration of the **heated cylinder** test case whose simulation parameters are indicated in Table 15. No-slip BC are applied to rigid and deformable boundaries, and the fluid domain is closed without free surface.

<i>Remesh</i>	<i>Fluid</i>	<i>Solid</i>	<i>Coupling</i>
$\delta = 0.6$	$\rho = 1 \text{ kg/m}^3$	$c_p = 1 \text{ J/K}$	$\varphi = \text{TPS}$
$\omega = 0.6$	$c_p = 1 \text{ J/K}$	$k = * \text{ W/mK}$	$R = 0.2 \text{ m}$
$\alpha = 1.2$	$k = 2.5 \text{ W/mK}$		$\Delta t_{\text{MAX}} = 10^{-4} \text{ s}$
$\beta = 0.2$	$\mu = 2.5 \text{ Pa s}$		$tol = 10^{-6}$
$\varepsilon = \infty$	$\gamma = 0 \text{ N/m}$		
	$tol = 10^{-8}$	$tol = 10^{-6}$	
		$\rho_{\infty} = 0.01$	

Table 15: Simulation parameters for the **heated cylinder** whose initial configuration is given in Figure 77. The governing equations are solved with the PSPG stabilization and the edge splitting remeshing with level set.

$$\text{Nu} = 0.3 \frac{0.62 \text{Re}^{1/2} \text{Pr}^{1/3}}{\left[1 + (0.4/\text{Pr})^{2/3}\right]^{1/4}} \left[1 + \left(\frac{\text{Re}}{282000}\right)^{5/8}\right]^{4/5} \quad (3.73)$$

This correlation is claimed to be valid for $\text{Pr Re} > 0.2$. Here, Re is the Reynolds number, which represents the ratio of inertial forces to viscous forces, and Pr is the Prandtl number, defined as the ratio of momentum diffusivity to thermal diffusivity. The interest of equation (3.73) is that it allows for the estimation of the Nusselt number without directly knowing the heat transfer coefficient. Using the parameters from Figure 77, we have $\text{Re} = 40$ and $\text{Pr} = 1$, which fall within the range of validity for this correlation. By substituting the values of Re and Pr into Equation (3.73), and then injecting the resulting Nusselt number into Equation (3.72), one has

$$\begin{aligned} \text{Nu} &= 3.83 \\ h &= 9.573 \text{ W/m}^2\text{K} \end{aligned} \quad (3.74)$$

The first simulation is conducted with $\text{Bi} = 0.2$, where the solid conductivity is set to $k_s = 11.967 \text{ W/mK}$. The second simulation corresponds to $\text{Bi} = 5$, with the solid conductivity set to $k_s = 0.478 \text{ W/mK}$, which lies outside the theoretical stability limit of the FFTB scheme. The simulation duration is set to 1 second, ensuring that the steady state is reached before the simulation ends. Lastly, the mesh size-field on the fluid side is determined by the function

$$h(\mathbf{x}) = \min \left\{ h_0 + 0.05 d(\mathbf{x})^2, h_\infty \right\} \quad (3.75)$$

where $h_0 = 2 \text{ cm}$, $h_\infty = 20 \text{ cm}$ and $d(\mathbf{x})$ is the distance to the FS boundary. The solid body is meshed using regular quadrilateral elements with a characteristic size of $h_s = 2 \text{ cm}$ at the exterior boundary. Initially, the two meshes are matching, but the edge splitting remeshing may trigger refinement of the fluid elements at the FS interface and disrupt the mesh alignment. Consequently, an interpolation method is required to map nodal quantities between the fluid and solid interfaces. A comparison of the steady-state temperature distribution over the wetted surface of the cylinder with numerical results obtained using Ansys Fluent [183] is shown in Figure 78.

These results demonstrate that our coupling framework converges and provides accurate results for both $Bi < 1$ and $Bi > 1$ in the cases presented. On average, 2 and 2.8 coupling iterations per time step are needed for $Bi = 0.2$ and $Bi = 5$, respectively, to reach the coupling tolerance. Snapshots of the velocity profile in the fluid are shown in Figure 79, while the temperature distribution in the fluid and inside the hollow cylinder is depicted in Figure 80.

Flexible Beam

A case of convection in a 2D square cavity with a flexible beam is used to verify the accuracy of the thermo-mechanical coupling and compare it with pure mechanical coupling. This test case was first introduced and simulated by Long and Yang [202] using a Smoothed Finite Element Method (S-FEM) combined with SPH. The initial configuration is shown in Figure 81. The cavity is filled with fluid, and its density variation is modeled using the Boussinesq approximation. The deformable beam is modeled using the finite strain linear elastic law (3.45). It is important to note that gravity acts on the fluid but not on the solid, in line with the setup presented in the literature. The mesh size-field for the fluid is governed by

$$h(\mathbf{x}) = \min \left\{ h_0 + 0.05 d(\mathbf{x}), h_\infty \right\} \quad (3.76)$$

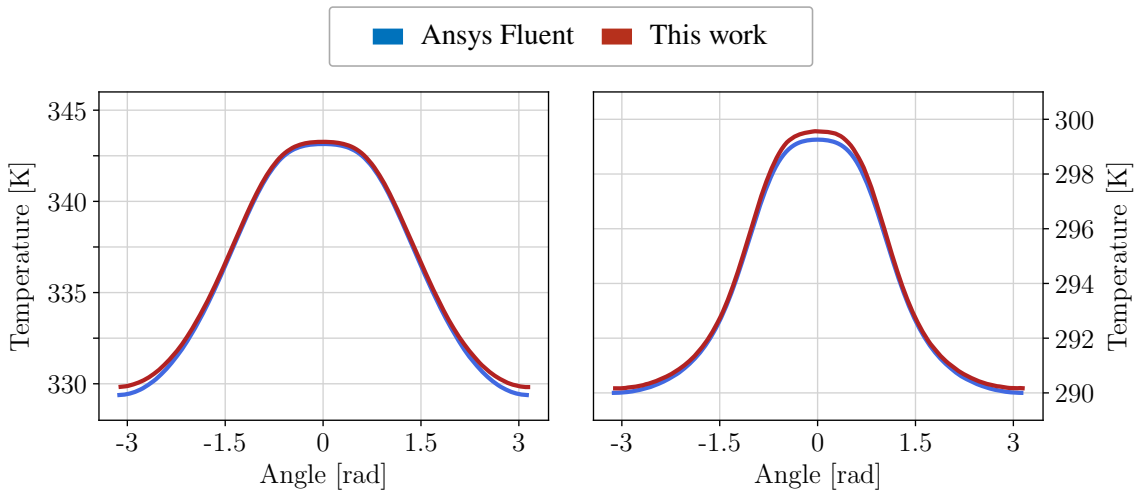


Figure 78: Temperature of the fluid-structure interface for the **heated cylinder** test case in 2D, as a function of the angular coordinate. The first figure holds for $Bi = 0.2$ and the second figure holds for $Bi = 5$.

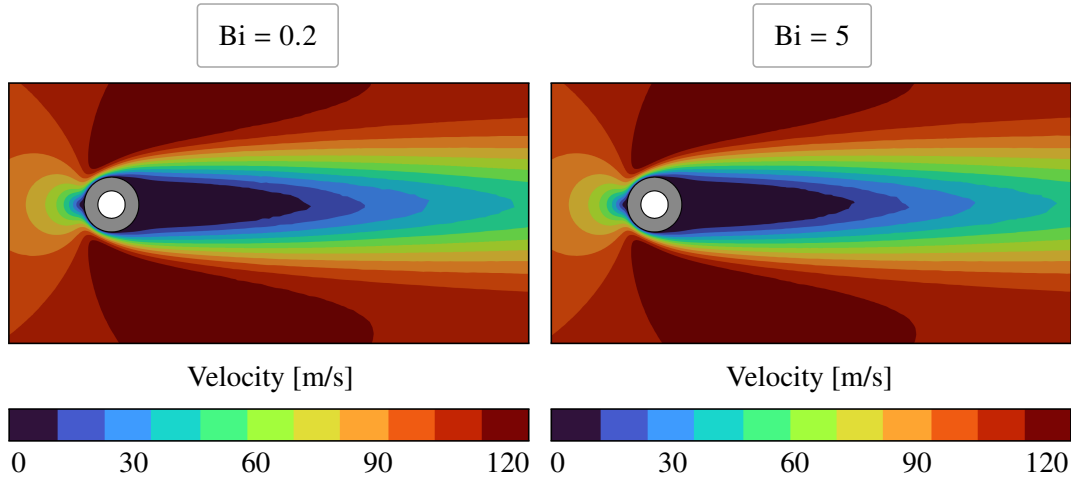


Figure 79: Snapshots of the particle finite element solution for the **heated cylinder** test case whose initial configuration is given in Figure 77, at time $t = 1.0$ seconds. The parameters from Table 15 have been used.

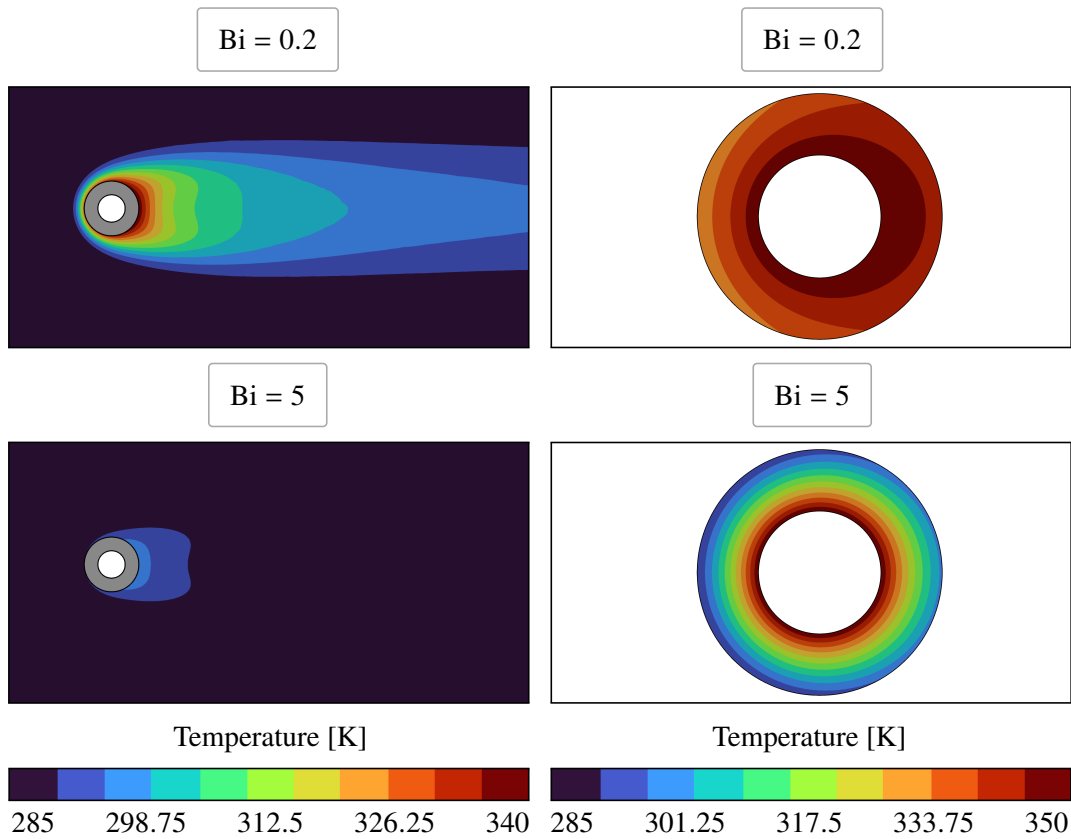


Figure 80: Snapshots of the particle finite element solution for the **heated cylinder** test case whose initial configuration is given in Figure 77, at time $t = 1.0$ seconds. The parameters from Table 15 have been used.

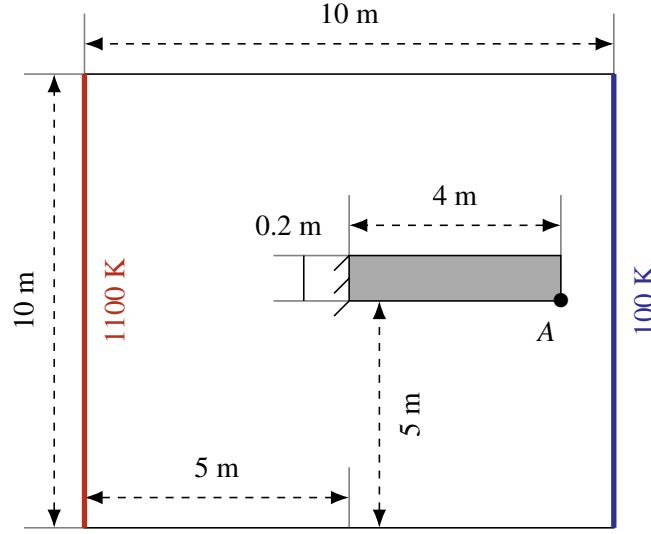


Figure 81: Initial configuration of the **flexible beam** test case whose simulation parameters are indicated in Table 16. No-slip BC are applied to rigid and deformable boundaries, and the fluid domain is closed without free surface.

<i>Remesh</i>	<i>Fluid</i>	<i>Solid</i>	<i>Coupling</i>
$\delta = 0.6$ $\omega = 0.6$ $\alpha = 1.2$ $\beta = 0.2$ $\varepsilon = \infty$	$\rho = 1 \text{ kg/m}^3$ $c_p = 1 \text{ J/K}$ $g = -10 \text{ m/s}^2$ $k = 1 \text{ W/mK}$ $\mu = 0.71 \text{ Pa s}$ $\gamma = 0 \text{ N/m}$ $\alpha = 0.071 \text{ K}^{-1}$	$\rho = 10^3 \text{ kg/m}^3$ $c_p = 1 \text{ J/K}$ $g = 0 \text{ m/s}^2$ $k = 1 \text{ W/mK}$ $\nu = 0.3$ $E = 4 \times 10^6 \text{ Pa}$ $\alpha = 5 \times 10^{-4} \text{ K}^{-1}$	$\varphi = \text{TPS}$ $R = 0.1 \text{ m}$ $\Delta t_{\text{MAX}} = 10^{-3} \text{ s}$ $tol = 10^{-6}$
	$tol = 10^{-6}$ $\lambda = 0.5$	$tol = 10^{-6}$ $\rho_{\infty} = 0.01$	

Table 16: Simulation parameters for the **flexible beam** whose initial configuration is given in Figure 81. The governing equations are solved with the fractional step and the edge splitting remeshing with level set.

Here, $h_0 = 5$ cm, $h_\infty = 20$ cm and $d(\mathbf{x})$ is the minimum distance between $\mathbf{x}$ and the solid boundary. The flexible beam is discretized using regular quadrilateral elements with a characteristic size of $h_s = 5$ cm. Both the fluid and structure are initially at rest and isothermal, with a temperature of 600 K, while the top and bottom walls are adiabatic. The beam deforms due to the thermal-induced motion of the fluid and due to its own thermal expansion, as the temperature field inside the beam becomes non-homogeneous.

Figure 82 shows the time history of the vertical displacement at point A (see Figure 81) for two coupling scenarios. The first scenario disables thermal coupling, treating the FS interface as adiabatic, but still synchronizes the displacement and surface tractions between the two solvers. The second scenario enables both thermal and mechanical coupling through FSPC, ensuring that the beam displacement, surface tractions, interface temperature, and heat fluxes are synchronized at the FS interface. In both cases, our results align with those in the literature. An overview of the temperature and velocity fields for the solid and fluid are presented in Figures 83 and 84, respectively.

Floating Sphere

The 3D test case is adapted from the work of Oñate et al. [203], which employs a Lagrangian formulation where the governing equations for the fluid and the structure are expressed in a unified manner using the PFEM. The simulation involves a solid sphere, initially at 270 K, falling and warming in a cylindrical tank initially at rest, filled with hot water at 340 K.

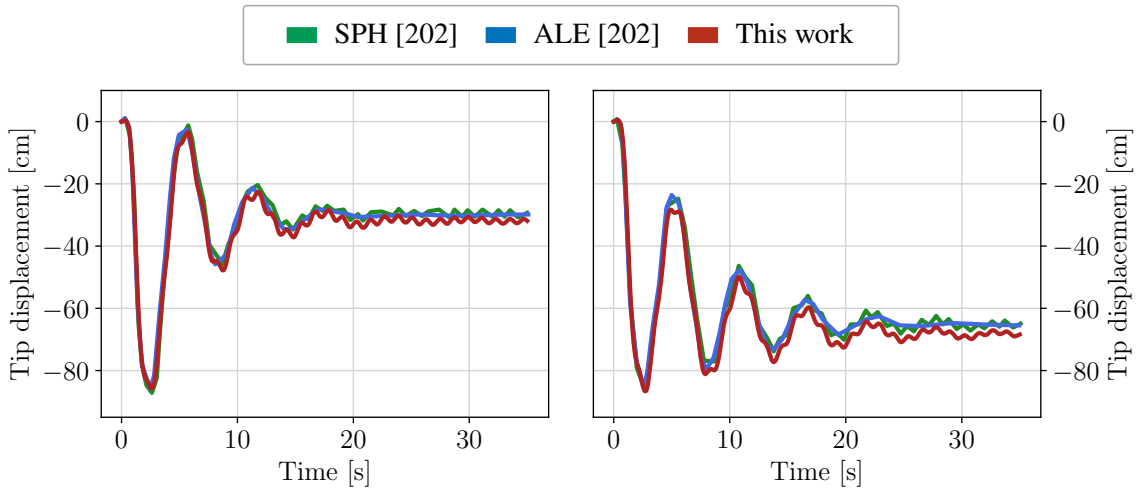


Figure 82: Tip y-displacement for the **flexible beam** test case in 2D. The initial position of the tip is indicated by the point A in Figure 81. The first figure enables mechanical coupling. The second figure enables thermo-mechanical coupling.

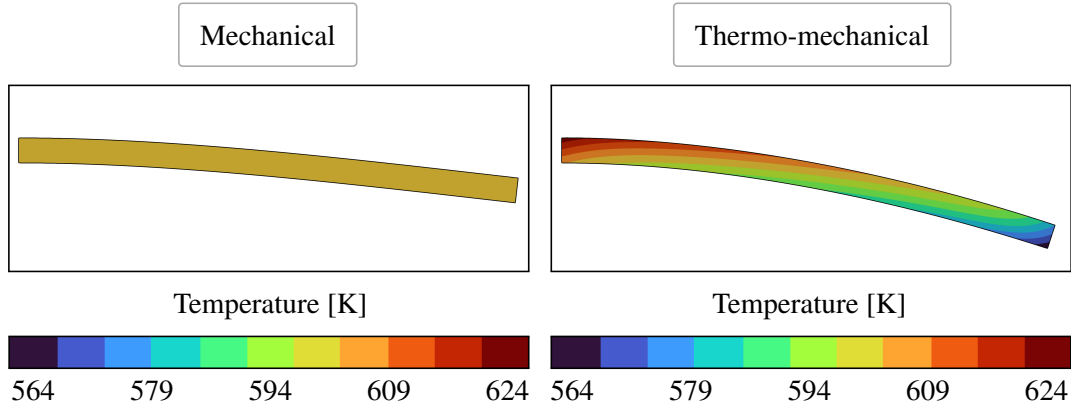


Figure 83: Snapshots of the particle finite element solution for the **flexible beam** test case whose initial configuration is given in Figure 81, at time $t = 35$ seconds. The parameters from Table 16 have been used.

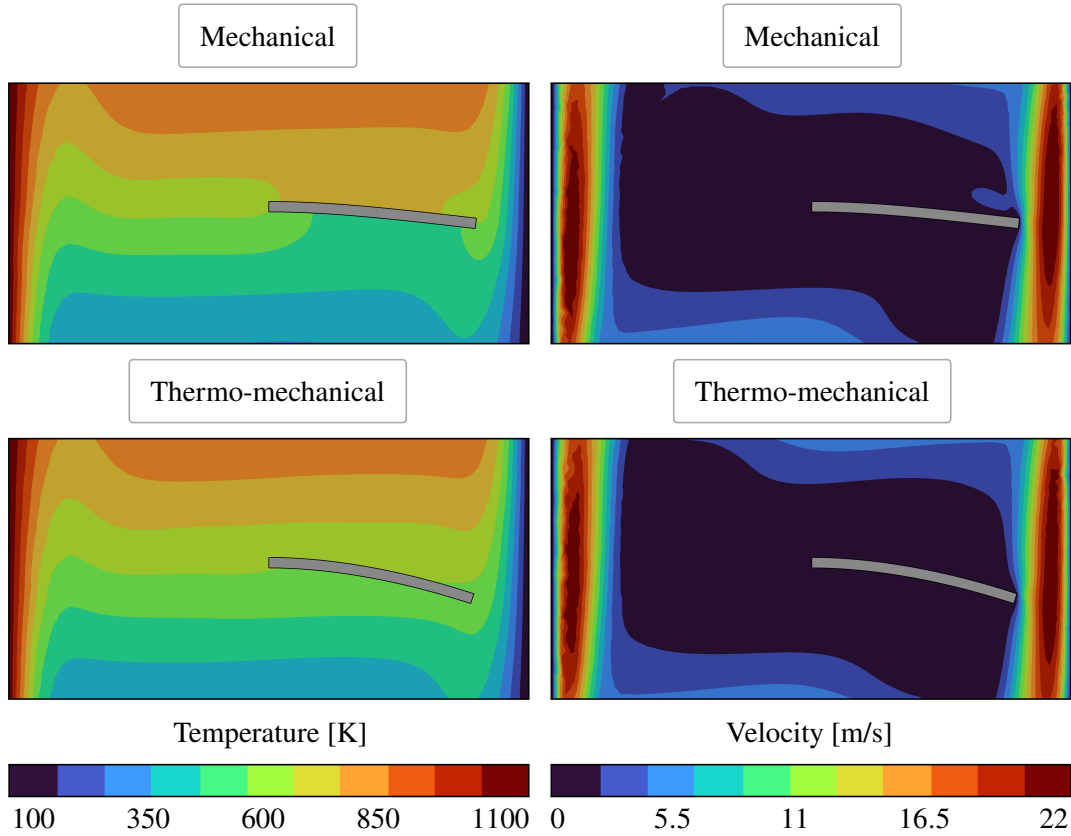


Figure 84: Snapshots of the particle finite element solution for the **flexible beam** test case whose initial configuration is given in Figure 81, at time $t = 35$ seconds. The parameters from Table 16 have been used.

Mechanical coupling occurs primarily through the Archimedes force applied to the sphere, which is sufficiently stiff to render strains negligible. The initial configuration is shown in Figure 85. The solid sphere is modeled as a linear elastic material and discretized using tetrahedral elements with standard integration and uniform characteristic size. The mesh size-field in the fluid domain is defined by

$$h(\mathbf{x}) = \min \left\{ h_0 + 0.1 d(\mathbf{x}), h_\infty \right\} \quad (3.77)$$

Here, $d(\mathbf{x})$ represents the minimum distance between the point $\mathbf{x}$ and either the free surface or the solid body. Two mesh discretizations are used: A coarse mesh with $h_0 = 0.3$ cm, $h_\infty = 1$ cm, $h_s = 0.2$ cm and a fine mesh with $h_0 = 0.2$ cm, $h_\infty = 0.5$ cm, $h_s = 0.1$ cm. Figure 86 compares the temperature evolution at the central node of the sphere, along with its vertical coordinate, to reference results from the literature. The simulation results show close agreement with the data reported in [203], with only minor discrepancies depending on the mesh resolution. Figure 87 displays the mesh configuration of the free surface and solid sphere at different time intervals, while Figure 88 illustrates the corresponding temperature field over time.

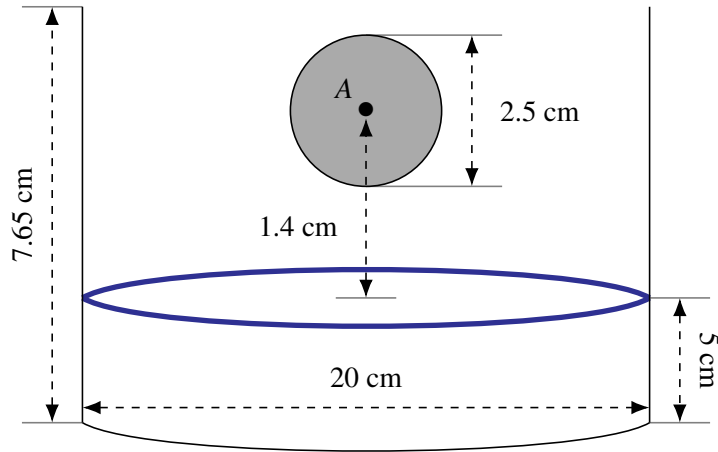


Figure 85: Initial configuration of the **floating sphere** test case whose simulation parameters are indicated in Table 17. No-slip BC are applied to rigid and deformable boundaries, and the initial free surface is outlined in bold.

<i>Remesh</i>	<i>Fluid</i>	<i>Solid</i>	<i>Coupling</i>
$\delta = 0.8$ $\omega = 0.4$ $\alpha = 1.7$ $\beta = 0.2$ $\varepsilon = 0.1$	$\rho = 10^3 \text{ kg/m}^3$ $c_p = 10^3 \text{ J/K}$ $g = -9.81 \text{ m/s}^2$ $k = 50 \text{ W/mK}$ $\mu = 0.01 \text{ Pa s}$ $\gamma = 0 \text{ N/m}$	$\rho = 600 \text{ kg/m}^3$ $c_p = 400 \text{ J/K}$ $g = -9.81 \text{ m/s}^2$ $k = 500 \text{ W/mK}$ $\nu = 0$ $E = 10^7 \text{ Pa}$	$\varphi = \text{TPS}$ $R = 0.01 \text{ m}$ $\Delta t_{\text{MAX}} = 10^{-3} \text{ s}$ $\text{tol} = 10^{-6}$
	$\text{tol} = 10^{-6}$ $\lambda = 0.5$	$\text{tol} = 10^{-6}$ $\rho_\infty = 0.01$	

Table 17: Simulation parameters for the **floating sphere** whose initial configuration is given in Figure 85. The governing equations are solved with the fractional step and the edge splitting remeshing with level set.

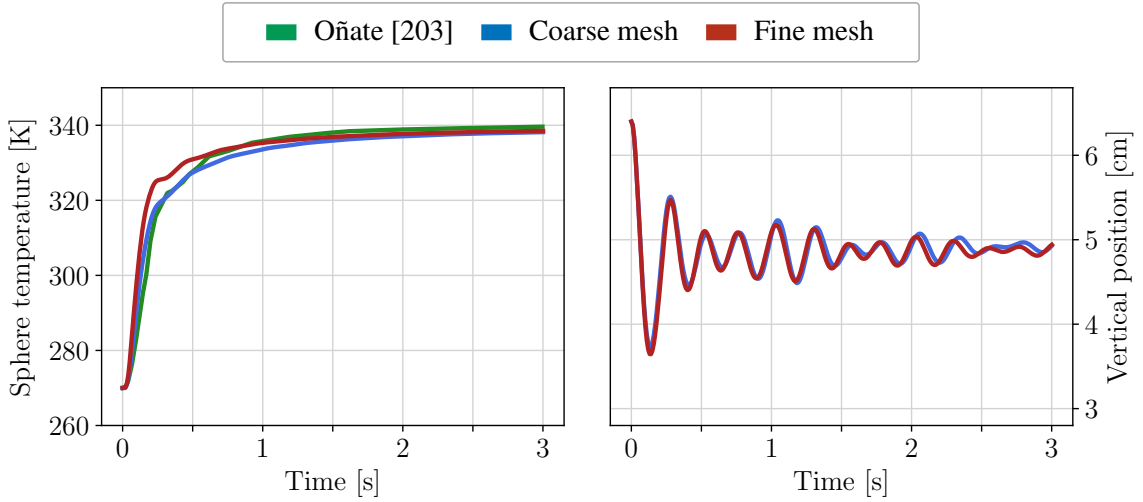


Figure 86: Temperature and y-coordinate of the **floating sphere** test case in 3D. The initial position of the tracked node is indicated by the point A in Figure 85. Note that the position of the sphere was not available in [203] for comparison.

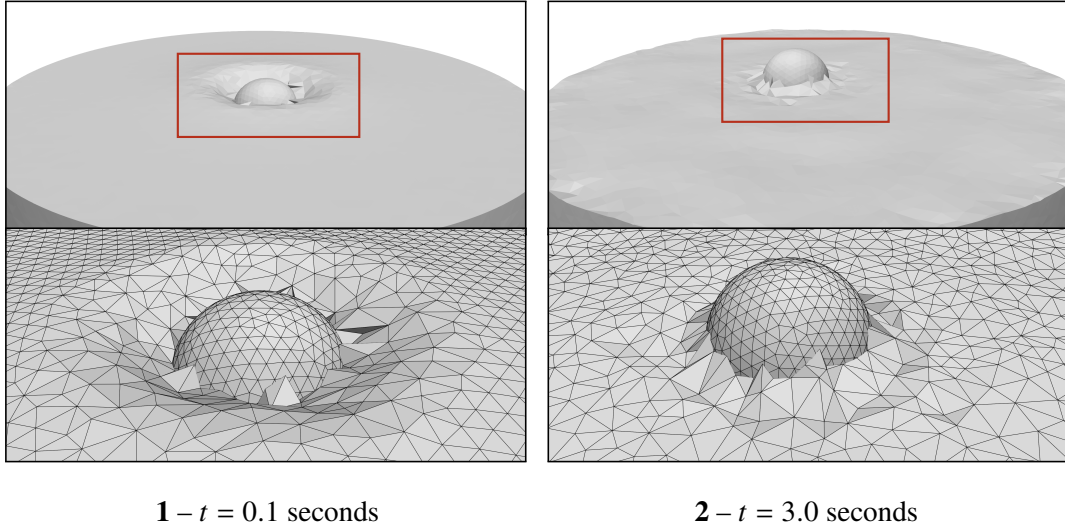


Figure 87: Outline of the free surface for the **floating sphere** test case whose initial configuration is given in Figure 85, at different time intervals. A zoom on the region outlined by the red rectangle is also presented below.

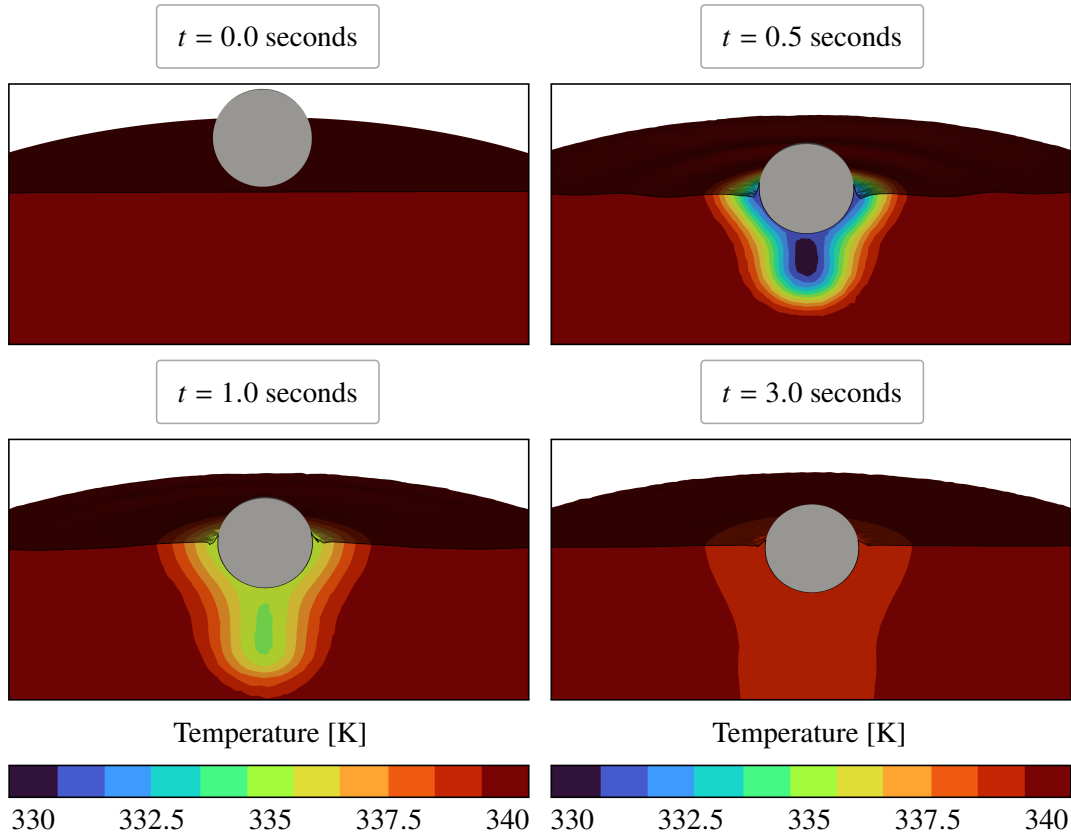


Figure 88: Snapshots of the particle finite element solution for the **floating sphere** test case whose initial configuration is given in Figure 85, at different time intervals. The parameters from Table 17 have been used.

3.5 Computation Times

Computation time is an important performance metric used to assess whether an algorithm can run within an acceptable time frame. This section presents an overview of the computation times for FSPC and PFEM3D across representative 3D test cases. The tests were conducted on an AMD Ryzen Threadripper 3970X processor with 32 cores, running Debian 12. PFEM3D relies on Eigen [204] for linear algebra operations and is compiled using Intel MKL [205]. Specifically, the parallel direct sparse solver interface from MKL is employed to solve the PSPG finite element system via LU factorization. When using the fractional step scheme, the finite element matrices are symmetric and positive definite, allowing the system to be solved with a Conjugate Gradient (CG) solver. This typically offers better performance than a general LU factorization, particularly for large 3D meshes. Parallel operations, such as assembling the global FE matrices, are handled using OpenMP [206]. On the other hand, Metafor uses Intel TBB [207] for parallelization and MKL [205] with LU factorization for the linear solver.

As a reminder, FSPC is a Python code that relies on efficient C++ and Fortran routines provided by the Scipy library [208]. In this study, Scipy is linked with MKL, which provides multithreaded, optimized implementations of BLAS and Lapack routines. Specifically, the computation of the inverse Jacobian (3.37) in the IQN-MVJ is performed using the SVD decomposition algorithm DGESDD from Lapack [209]. The predictor update (3.37) is computed with the least squares solver DGELSY, which finds the minimum-norm solution by performing a complete orthogonal factorization of the matrix. This least squares approach is required because $\mathbf{V}$ is rectangular.

The RBF interpolation system (3.54) is symmetric and solved using the Bunch-Kaufman LDLT factorization [210] with diagonal pivoting, implemented by the DSYTRF and DSYTRS routines from Lapack. The key advantage of using a direct solver for RBF interpolation, rather than an iterative solver, is that the factorization of $\mathbf{A}$ is computed once per remeshing. This factored matrix can then be reused for multiple interpolations (i.e. for each IQN iteration) at minimal cost, significantly reducing the overall execution time. From our experience, the DSYTRF routine offers comparable accuracy and faster computation than other Lapack routines, as long as the linear system is not nearly singular. This is typically the case for most 3D configurations when the PFEM mesh at the interface is unstructured. However, when $\mathbf{A}$ is singular or nearly singular, the DGELSD routine can provide the least squares solution and handle rank-deficient matrices, albeit at the cost of significantly higher computation time.

Aortic Valve

This first study illustrates the dynamics of blood ejection from the left ventricle into the aorta. We consider a simplified 2D plane strain model of the aortic valve, as described by Delhez in [101]. The initial configuration of the problem is shown in Figure 89. The fluid domain includes the left ventricle, two sinuses, and the initial section of the aorta. The aortic valve is represented by two flexible leaflets, modeled using a linear elastic law (3.45). It is important to note that this setup is a significant simplification of real heart valves. In reality, aortic valves have three leaflets in a 3D geometry, each opening into its respective sinus cavity. Furthermore, realistic leaflet models often incorporate fiber-reinforced materials to capture the behavior of collagen fibers embedded in the tissue. The primary objective of this test case is to replicate the fluid-structure interaction and valve closure that separates the fluid domain into two distinct regions. This represents a non-trivial configuration for evaluating the performance of the numerical codes. For a more comprehensive analysis of blood flow and its interaction with arterial walls, the reader is referred to the study [101], also conducted using FSPC.

The fluid is initially discretized with 3.4×10^4 triangles of characteristic sizes $h_F = 0.5$ mm and 1.7×10^4 nodes of which $N_F = 244$ are located on the FS interface. The solid is discretized with 3039 quadrangle elements and 3464 nodes of which $N_s = 820$ are located on the FS interface. The blood flow is driven by imposing the temporal pressure variation shown in Figure 90 on both the inlet and the outlet. This BC has been obtained from the work of Nobari et al. [211] who measured clinical values of average coronary flow [212]. Inlet and outlet boundary conditions are applied using a single layer of Eulerian nodes with prescribed pressure. These nodes remain fixed in space, effectively simulating fluid entering and exiting the computational domain without requiring physical movement of the inlet and outlet boundaries.

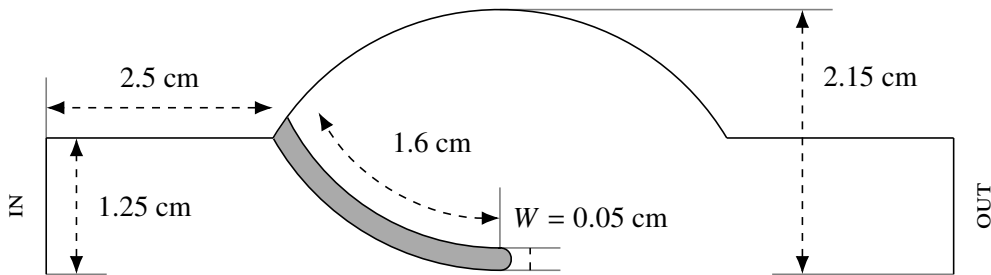


Figure 89: Initial configuration of the **aortic valve** test case whose simulation parameters are indicated in Table 18. No-slip BC are applied to rigid and deformable boundaries, and the fluid domain is closed without free surface.

<i>Remesh</i>	<i>Fluid</i>	<i>Solid</i>	<i>Coupling</i>
$\delta = 0.6$ $\omega = 0.6$ $\alpha = 1.2$ $\beta = 0.2$ $\varepsilon = \infty$	$\rho = 1050 \text{ kg/m}^3$ $g = 0 \text{ m/s}^2$ $\mu = 3.8 \times 10^{-3} \text{ Pa s}$ $\gamma = 0 \text{ N/m}$	$\rho = 1060 \text{ kg/m}^3$ $g = 0 \text{ m/s}^2$ $\nu = 0.3$ $E = 10^7 \text{ Pa}$	$\varphi = \text{TPS}$ $R = 0.01 \text{ m}$ $\Delta t_{\text{MAX}} = 10^{-4} \text{ s}$ $\text{tol} = 10^{-6}$
	$\text{tol} = 10^{-8}$	$\text{tol} = 10^{-6}$ $\rho_{\infty} = 0.01$ $k_N = 10^8 \text{ N/m}^2$	

Table 18: Simulation parameters for the **aortic valve** whose initial configuration is given in Figure 89. The governing equations are solved with the PSPG stabilization and the edge splitting remeshing with level set.

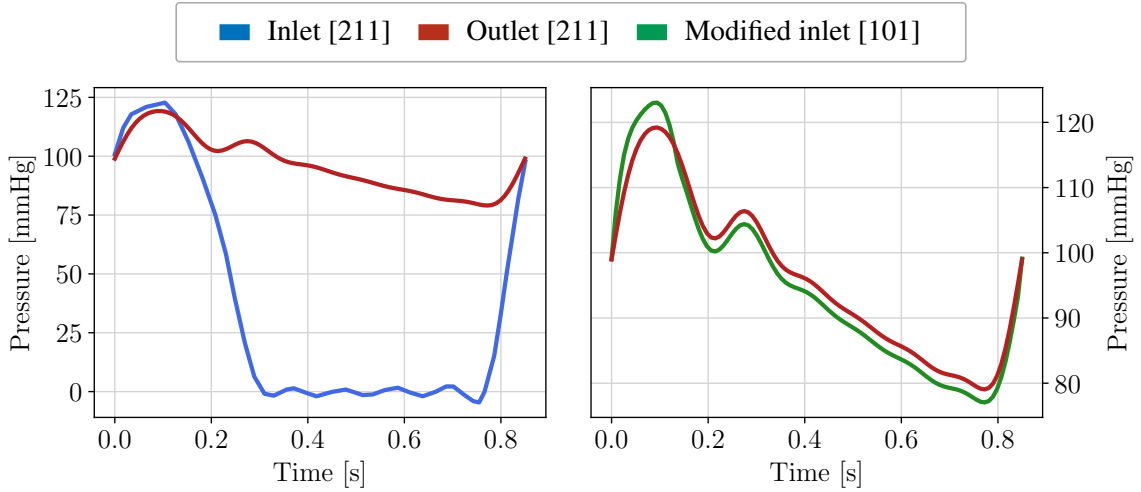


Figure 90: Time series of the inlet (ventricle) and outlet (aorta) pressures during a cardiac cycle. To compensate for the impact of the 2D geometry, a modified inlet pressure is applied to ensure that the valve resists the applied pressure difference.

Mathematically, pressure BC are not enforced as Dirichlet conditions on the primary unknowns but are instead introduced as Neumann conditions in the weak formulation of the PSPG, by adding the following surface traction contribution to the right-hand side of the finite element system:

$$\int_{\Gamma_\Sigma} \mathbf{N}_3 \mathbf{t} d\mathbf{x} = - \int_{\Gamma_\Sigma} \mathbf{N}_3 p(t) \mathbf{n} d\mathbf{x} \quad (3.78)$$

Here, Γ_Σ denotes the Neumann boundary of the fluid domain, which holds for the inlet and the outlet surfaces in Figure 89, $p(t)$ is the prescribed pressure defined Figure 90 at a given time, and $\mathbf{n}$ is the outer unit normal to the fluid domain. As a reminder, $\mathbf{N}_3$ is the matrix of shape functions associated with a 3D vector field introduced in the previous chapter, and whose full expression is given in the Annex A. Note that, in the present test case, the fields are defined in 2D.

Due to the simplified 2D geometry of the problem, the valve's ability to block reverse flow is significantly reduced compared to real-life configurations. To compensate for this geometric limitation, a modified inlet condition is applied to ensure the valve can withstand the imposed pressure difference. Additionally, a sticking contact condition is enforced between the two leaflets to prevent sliding, keeping the valve closed as long as the net normal force acts in the opposite direction to their outer normal vector.

Snapshots of the solution at various time intervals during a cardiac cycle are shown in Figure 91. Figure 92 breaks down the total simulation time into its main computational components. In these figures, *TPS* refers to the time spent on interpolation using the Thin Plate Spline basis function, including both matrix assembly and interpolation. *Solver* represents the time required to solve the fluid and structural equations at each time step. *MVJ* denotes the time attributed to the coupling algorithm, while *Update* accounts for additional operations needed to prepare the FE solvers for the next time step, such as remeshing in the PFEM-LS, reinitializing boundary condition vectors, or storing the equilibrium solution in memory.

Note that the MPI processes shown in Figure 92 run in parallel using a combination of blocking and non-blocking operations. This parallelism complicates precise attribution of execution time to individual components. For instance, the solution of the FE system, grouped under the *Solver* category, requires coordinated execution and cannot be performed concurrently by both processes. In contrast, both solvers can perform the *Update* step in parallel, meaning the total time spent by FSPC on this step is less than the sum of times reported by each process.

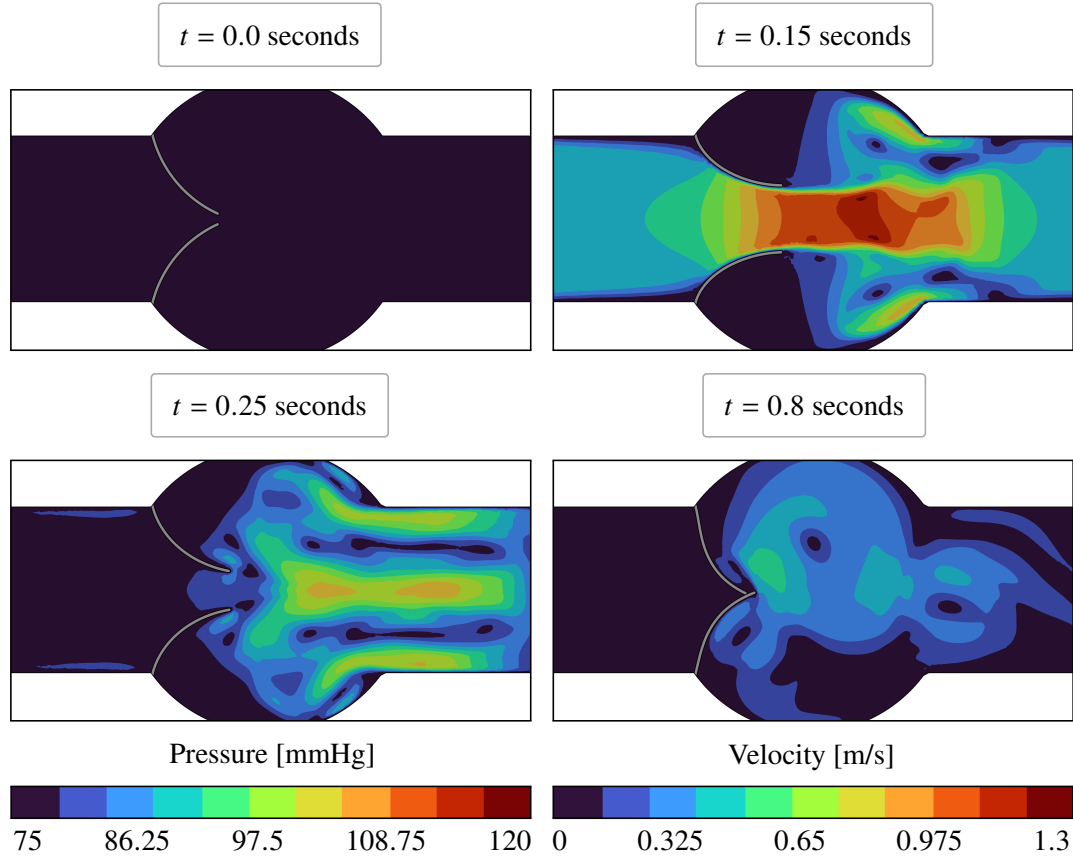


Figure 91: Snapshots of the particle finite element solution for the **aortic valve** test case whose initial configuration is given in Figure 89, at different time intervals. The parameters from Table 18 have been used.

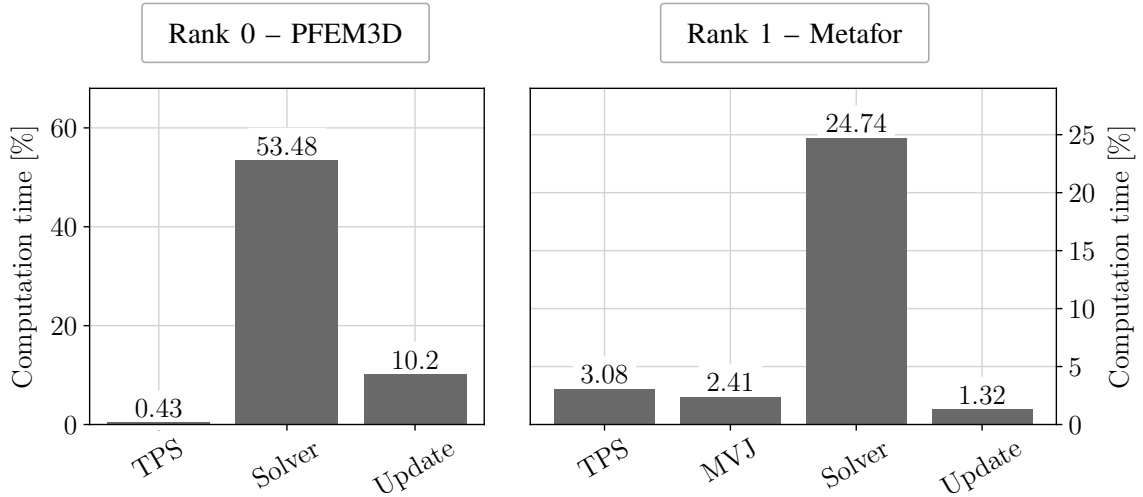


Figure 92: Shares of the total computation time (1.6 hours) for the **aortic valve** test case using the **PSPG** formulation for the fluid. The initial configuration is given in Figure 89 and the simulation parameters are given in Table 18.

Similarly, interpolation matrices can be assembled and factorized independently on each process. As a reminder, the interpolation operators $\mathcal{V}_{\text{FS}}$ and $\mathcal{V}_{\text{SF}}$ are frequently updated to reflect mesh configuration changes following remeshing. In practice, the majority of the interpolation time is spent reconstructing the interpolation matrices $\mathbf{A}$ and $\mathbf{B}$, as defined in Equations (3.50) and (3.51), and performing the LDLT factorization of $\mathbf{A}$. These tasks collectively account for approximately 3% of the total execution time. Since the interpolation matrices are assembled in parallel across both processes, this cost remains relatively low compared to the overall simulation time.

A more detailed breakdown of computation times within the PFEM-LS solver is presented in Figure 93. The figure shows that, when using the PSPG stabilization, most of the execution time is spent on the linear solver, followed by the global matrix assembly. In contrast, remeshing, which includes Delaunay tessellation and finite element mesh initialization, accounts for approximately 5% of the total execution time. These operations roughly correspond to Step 3 in the remeshing procedure described in the previous chapter. More importantly, edge splitting refinement using the LS criterion, represented by the categories *Edge split* and *Level set*, and corresponding to Steps 1, 2, and 4, accounts for only 2% of the total computation time. Notably, our implementation of the LS criterion using a ray-casting algorithm has a negligible impact on remeshing time, representing a significant improvement over earlier approaches in the literature [79].

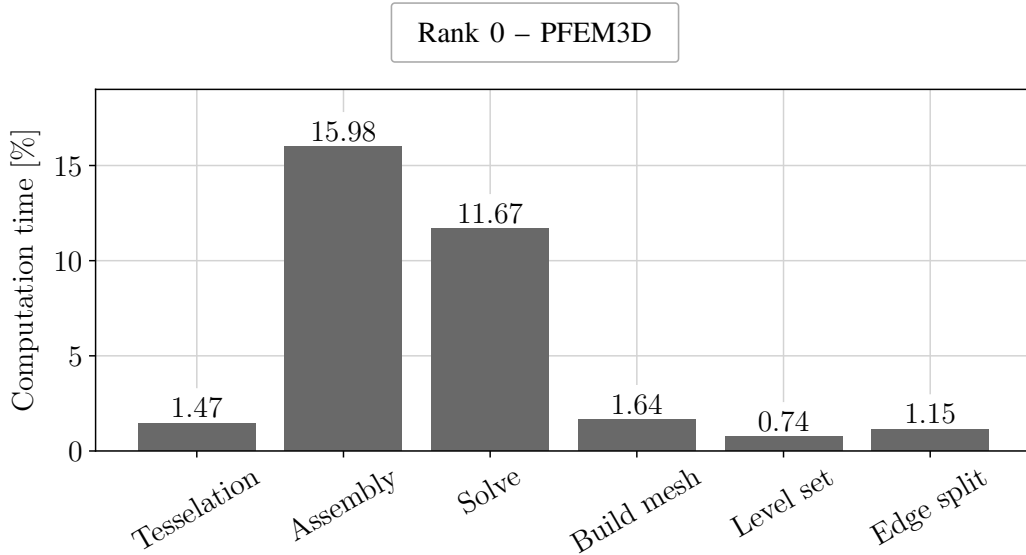


Figure 93: Shares of the total computation time (1.6 hours) for the **aortic valve** test case using the **PSPG** formulation for the fluid. The initial configuration is given in Figure 89 and the simulation parameters are given in Table 18.

The *Build mesh* category corresponds to Step 6 in the remeshing process, where the tessellation is converted into a finite element mesh. The *Assembly* category includes the computation of element matrices and their insertion into the global FE matrix. Finally, *Solve* refers to the LU solver, including both matrix factorization and substitution steps.

Rolling Tank

The second study investigates the execution times for the simulation of a hanging plate in a rolling tank in 3D, originally proposed by Souto-Iglesias et al. [213] in an experimental setup. The benchmark has later been analyzed numerically in [214] with the SPH method for the fluid, and a Volume-Compensated Particle Method (VCPM) for the solid. The initial configuration of the problem is shown in Figure 94. The tank is subjected to a smooth periodic rotation $\theta(t)$ around the axis O given by the following function:

$$\theta(t) = 4 \sin \left\{ \frac{2\pi t}{Q(1 + e^{S-Kt}) + \frac{R-Q}{1+e^t} + \frac{(R-Q)e^{S-Kt}}{1+e^t}} \right\} \quad (3.79)$$

The parameters in Equation (3.79) were obtained by fitting experimental measurements of the roll angle reported by Souto-Iglesias et al. [213], resulting in $K = 4.9$, $S = 2.144$, $R = 2.278$, and $Q = 1.278$. The fluid domain is initially discretized with 1.5×10^5 tetrahedral elements of characteristic size $h_F = 0.5$ cm, and consists of approximately 4.4×10^4 nodes, of which $N_F = 493$ are located on the FS interface. The solid domain is discretized using 780 regular hexahedral SRIPR elements and 1320 nodes, with $N_s = 969$ nodes lying on the FS interface. In this study, structural deformation is modeled using a Neo-Hookean material, characterized by the following strain-energy function:

$$\Psi = \frac{G_0}{2} (\text{tr } \bar{\mathbf{C}} - 3) + \frac{K_0}{2} (\ln J)^2 \quad (3.80)$$

Here, K_0 denotes the initial bulk modulus, G_0 is the initial shear modulus, $J = \det(\mathbf{F})$ is the determinant of the deformation gradient tensor, and $\bar{\mathbf{C}}$ is the modified right Cauchy-Green deformation tensor with unit determinant. The deformation of the plate is analyzed by tracking the horizontal displacement of the point A indicated in Figure 94. A comparison with results from the literature is presented in Figure 95 along with the mass variation of the fluid. Snapshots of the mesh close to the FS interface and of the full solution are presented in Figures 96 and 97.

Figure 98 shows the fraction of total simulation time allocated to different components of the software. The results confirm the earlier conclusion drawn from the 2D case. Both the interpolation and the IQN-MVJ coupling consume only a small portion of the total computation time, even when frequent remeshing operations are involved.

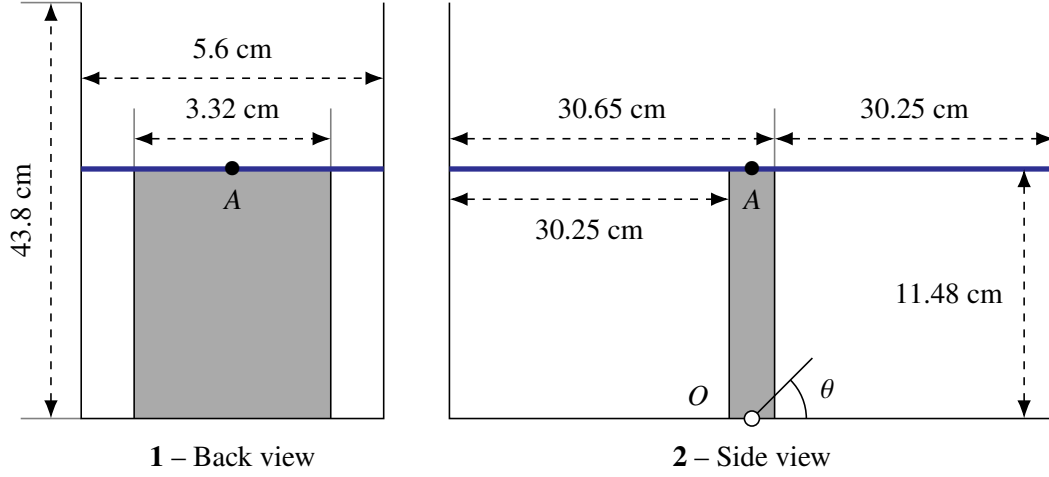


Figure 94: Initial configuration of the **rolling tank** test case whose simulation parameters are indicated in Table 19. No-slip BC are applied to rigid and deformable boundaries, and the initial free surface is outlined in bold.

<i>Remesh</i>	<i>Fluid</i>	<i>Solid</i>	<i>Coupling</i>
$\delta = 0.8$	$\rho = 917 \text{ kg/m}^3$	$\rho = 1100 \text{ kg/m}^3$	$\varphi = \text{TPS}$
$\omega = 0.4$	$g = -9.81 \text{ m/s}^2$	$g = -9.81 \text{ m/s}^2$	$R = 0.01 \text{ m}$
$\alpha = 1.7$	$\mu = 5 \times 10^{-2} \text{ Pa s}$	$G_0 = 2.2 \times 10^6 \text{ Pa}$	$\Delta t_{\text{MAX}} = 10^{-3} \text{ s}$
$\beta = 0.2$	$\gamma = 0 \text{ N/m}$	$K_0 = 10^7 \text{ Pa}$	$tol = 10^{-6}$
$\varepsilon = 0.02$			
	$tol = 10^{-6}$	$tol = 10^{-6}$	
	$\lambda = 0.5$	$\rho_{\infty} = 0.01$	

Table 19: Simulation parameters for the **rolling tank** whose initial configuration is given in Figure 94. The governing equations are solved with the fractional step and the edge splitting remeshing with level set.

This finding also supports the observations of Thomas [30], who reported negligible interpolation times during steady aeroelastic simulations of AGARD wings using SU2 [215]. In that test case, the mesh consisted of 2.48×10^5 fluid elements, with $N_s = 797$ and $N_F = 6281$ interface nodes. It is important to note, however, that SU2 is not a PFEM solver. Therefore, no remeshing is performed during the simulation, and the interpolation matrices remain unchanged. This significantly reduces the relative computational cost of interpolation compared to total execution time. A more detailed breakdown of computation times within the PFEM-LS solver is provided in Figure 99. In the fractional step scheme, most of the execution time is spent on global matrix assembly, followed by the linear solver.

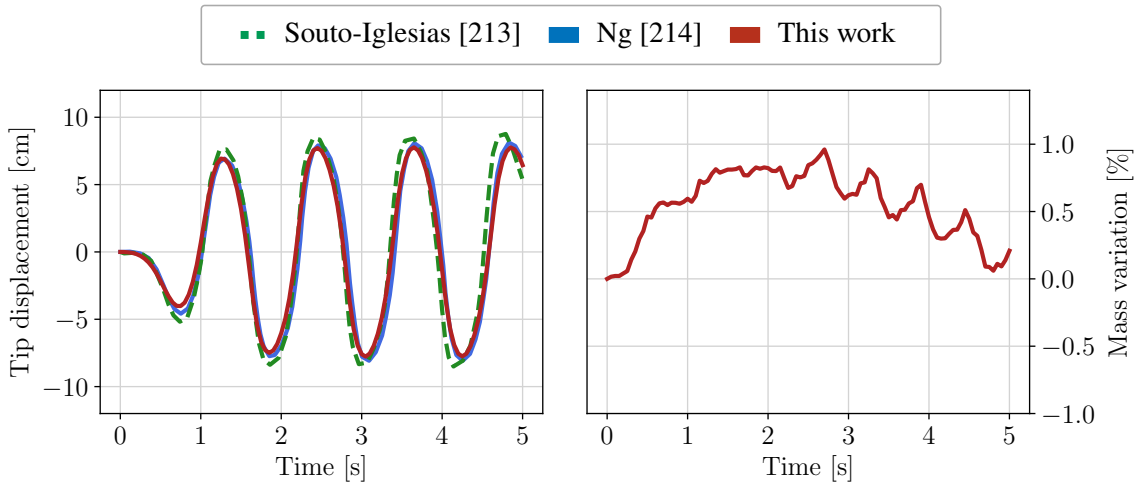


Figure 95: Horizontal displacement of the point A indicated in Figure 94 and mass variation of the fluid for the **rolling tank** test case in 3D. The dashed line refers to experimental measurements by Souto-Iglesias.

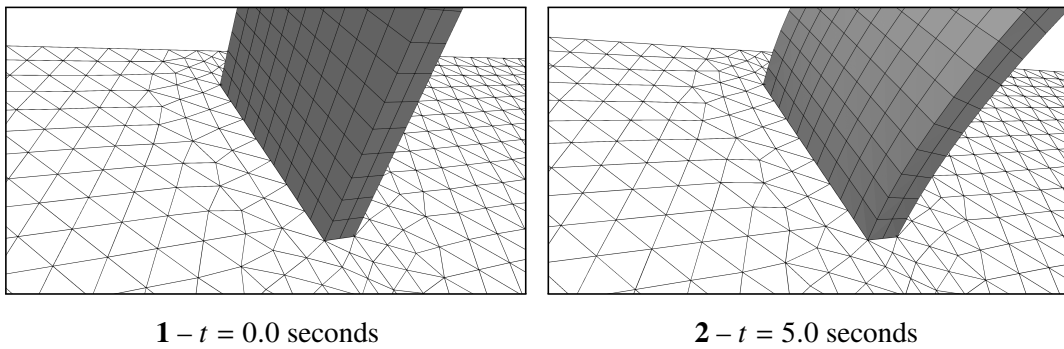


Figure 96: Zoom on the fluid-structure interface mesh for the **rolling tank** test case whose initial configuration is given in Figure 19, at different time intervals. Only the bottom boundary of the fluid mesh is displayed.

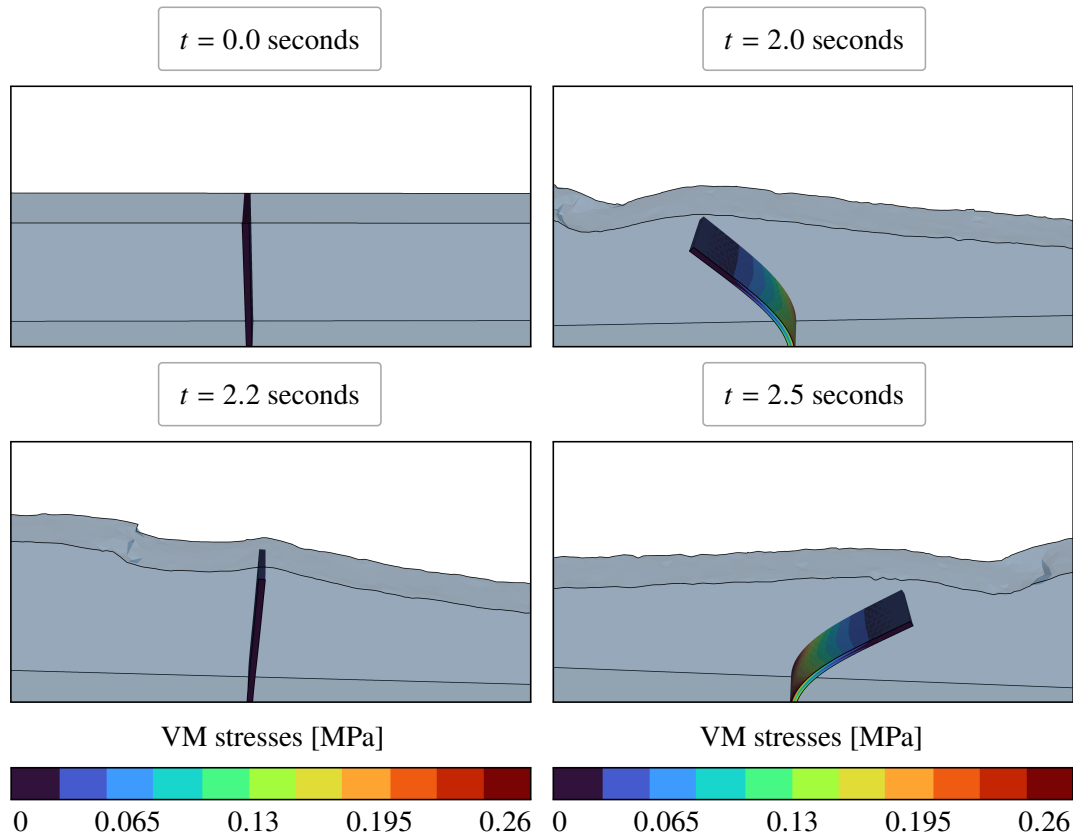


Figure 97: Snapshots of the particle finite element solution for the **rolling tank** test case whose initial configuration is given in Figure 94, at different time intervals. The parameters from Table 19 have been used.

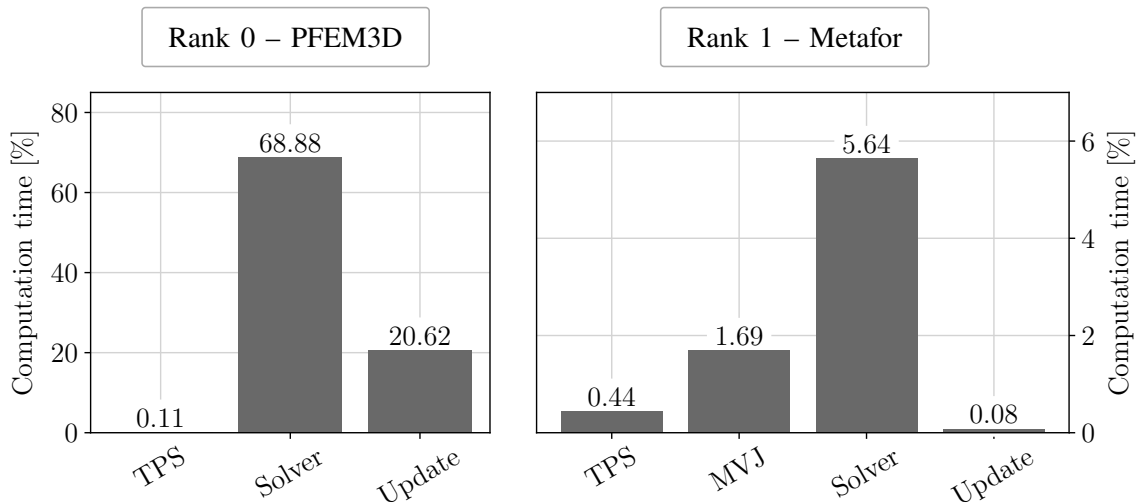


Figure 98: Shares of the total computation time (8.5 hours) for the **rolling tank** test case using the **fractional step** for the fluid. The initial configuration is given in Figure 94 and the simulation parameters are given in Table 19.

In contrast, less than 10% of the total execution time is consumed by the remeshing library, which manages Delaunay tessellation and mesh optimization (primarily to regularize slivers). These tasks roughly correspond to Steps 3 and 5 of the remeshing procedure described in the previous chapter. Edge splitting refinement using the LS criterion accounts for less than 3% of the total computation time, further demonstrating the method's efficiency in 3D.

In Chapter 2, two stabilizations have been presented. The PSPG stabilizes the system by reformulating the continuity equation to relax the strict incompressibility constraint, and leads to a monolithic formulation where the velocity and the pressure of the fluid are solved together in a unified system. On the other hand, the fractional step scheme is a projection method designed to break down the original Navier-Stokes system into smaller sub-problems by decoupling the pressure and velocity variables, which introduces an implicit stabilization effect by separating the momentum and continuity equations. While the PSPG tends to produce more stable and accurate results, the fractional step has the major advantage of being less computationally expensive due to the smaller size and the symmetry of the sub-problems, allowing for a CG solver to be used. Evidences of this are presented in Figures 100 and 101, where the PSPG has been used for simulating the hanging plate in a rolling tank. Notably, the fractional step approach is roughly twice faster than the PSPG for the present test case, and this difference is expected to increase with the number of degrees of freedom in the fluid mesh.

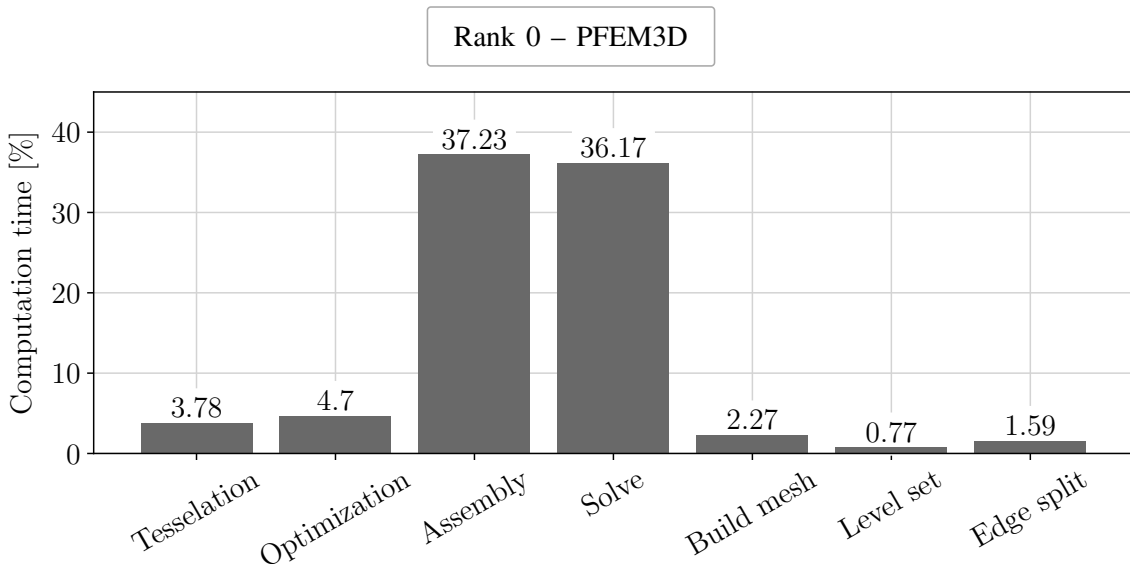


Figure 99: Shares of the total computation time (8.5 hours) for the **rolling tank** test case using the **fractional step** for the fluid. The initial configuration is given in Figure 94 and the simulation parameters are given in Table 19.

Indeed, LU solvers typically exhibit $O(n^3)$ complexity, where n is the size of the matrix. In contrast, the CG method has a typical complexity of $O(nk)$, where k is the number of iterations required for convergence, and is generally much smaller than n . While these estimates are characteristic of dense (i.e. non-sparse) matrices, they still illustrate the superior scalability of the CG method as the system size increases.

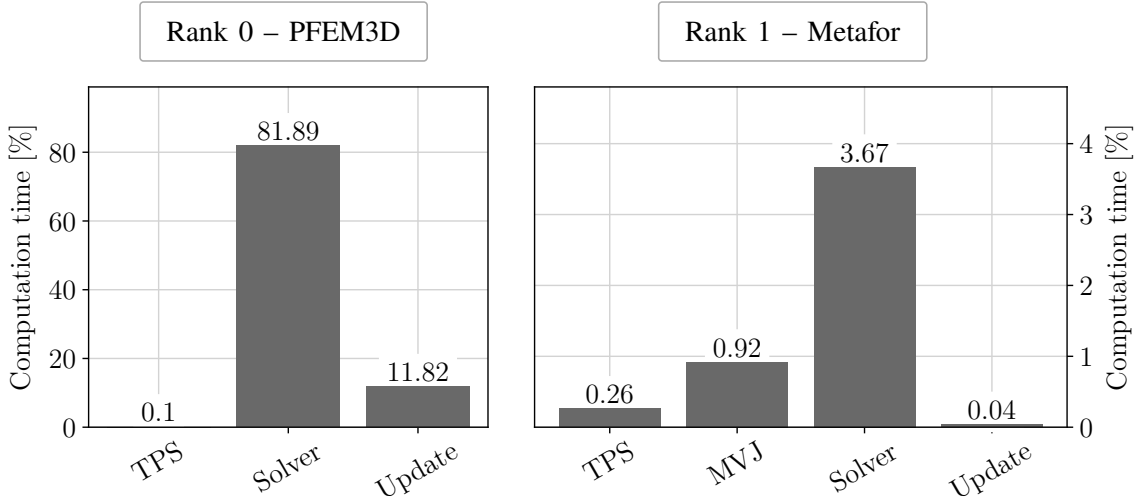


Figure 100: Shares of the total computation time (18.5 hours) for the **rolling tank** test case using the **PSPG** formulation for the fluid. The initial configuration is given in Figure 94 and the simulation parameters are given in Table 19.

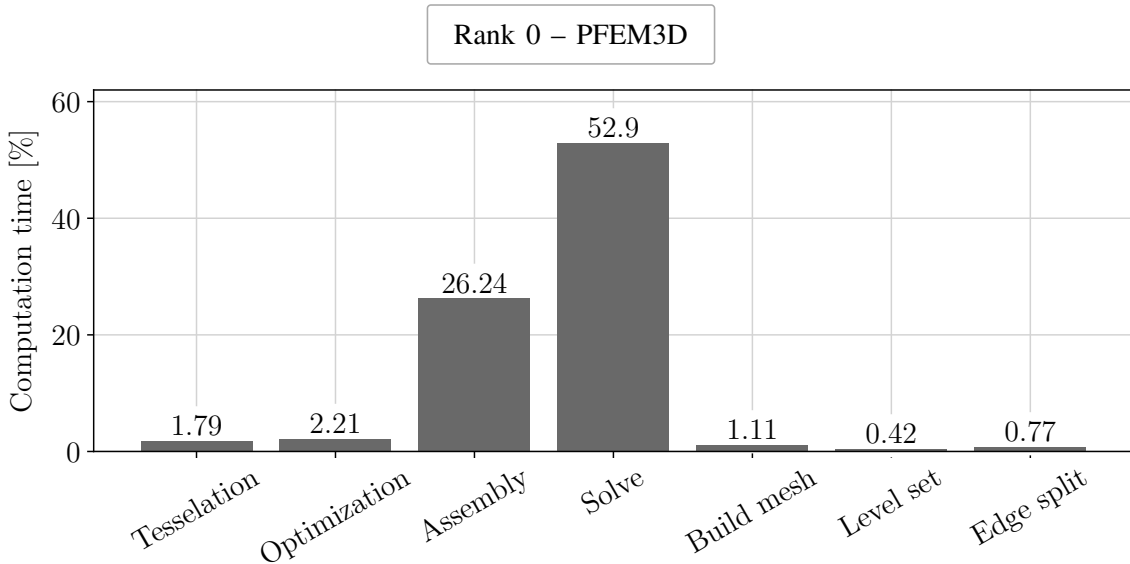


Figure 101: Shares of the total computation time (18.5 hours) for the **rolling tank** test case using the **PSPG** formulation for the fluid. The initial configuration is given in Figure 94 and the simulation parameters are given in Table 19.

Obstructed Pipe

Until now, the test cases have featured solids with a relatively small number of nodes compared to the fluid mesh, as well as a limited number of FS interface nodes on the solid side. For example, in the rolling tank test case, the ratio between the number of FSI nodes on the solid side and those in the fluid volume was around 0.65%. Given this, it is worth exploring a case where this disparity is less pronounced. To this end, we analyze the deformation of an obstacle obstructing a pipe. A time-dependent flow rate is imposed at the pipe's inlet, and the incoming fluid progressively fills the pipe, pushing the solid obstacle out of the cavity. The initial configuration of the problem is shown in Figure 102. The solid structure is modeled using the hyperelastic strain-energy function (3.80). The contact between the obstacle and the rigid pipe is governed by the Coulomb law of friction.

The solid is discretized using a structured mesh of 14420 hexahedral SRIPR elements and $N_s = 7772$ nodes on the FS interface. The fluid mesh uses a uniform characteristic size of $h_f = 1$ cm and includes $N_f = 3888$ nodes on the FS interface. As the simulation advances, more fluid enters the pipe cavity, increasing the number of degrees of freedom on the fluid side. The inlet velocity is initially zero and a constant acceleration of $a = 20$ m/s² is imposed. This profile enables the calculation of the mass flow rate into the pipe and, in turn, the evolution of the total fluid mass:

$$m(t) = \frac{\rho \pi R^2 a t^2}{2} \quad (3.81)$$

Here, $R = 5$ cm is the inlet radius. Figure 103 shows the time evolution of both the total fluid mass and the number of nodes in the fluid mesh, illustrating the good mass conservation achieved by the PFEM-LS solver. In this simulation, the total number of nodes in the fluid mesh remains lower than the number of nodes on the solid interface until 0.3 seconds. As a result, the relative computational cost of the coupling algorithm across the FS interface is expected to be more significant. Snapshots of the solution at various time intervals are shown in Figures 104 and 105. Note that the rigid pipe boundary is omitted from these visualizations. The total computation time for the obstructed pipe test case was approximately 23 hours.

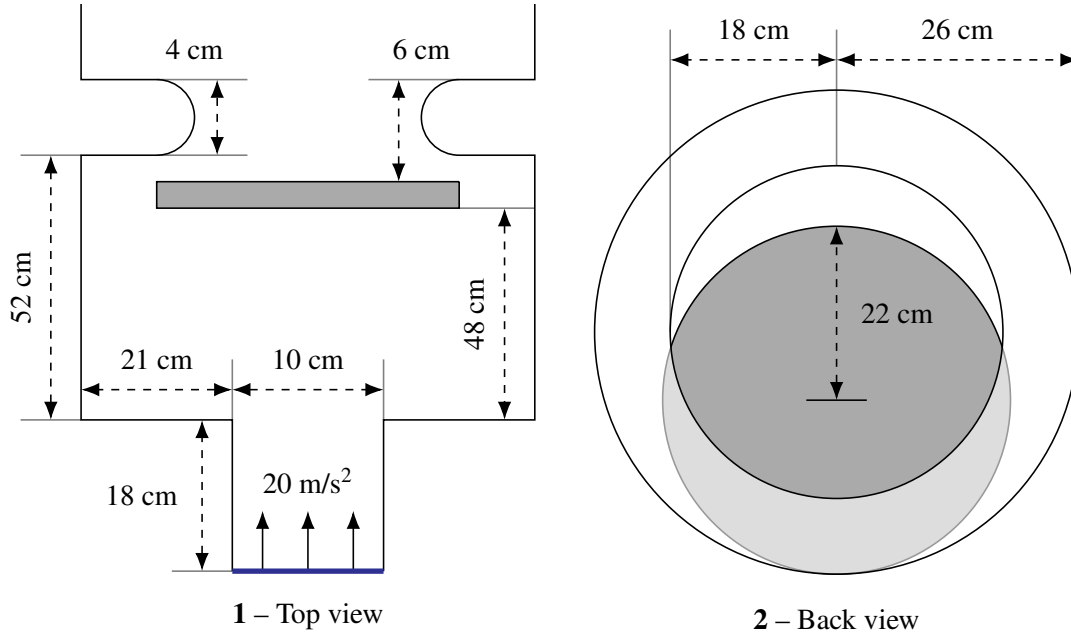


Figure 102: Initial configuration of the **obstructed pipe** test case whose simulation parameters are indicated in Table 20. No-slip BC are applied to rigid and deformable boundaries, and the initial free surface is outlined in bold.

<i>Remesh</i>	<i>Fluid</i>	<i>Solid</i>	<i>Coupling</i>
$\delta = 0.8$	$\rho = 10^3 \text{ kg/m}^3$	$\rho = 600 \text{ kg/m}^3$	$\varphi = \text{TPS}$
$\omega = 0.4$	$g = -9.81 \text{ m/s}^2$	$g = -9.81 \text{ m/s}^2$	$R = 0.1 \text{ m}$
$\alpha = 1.7$	$\mu = 10^{-3} \text{ Pa s}$	$G_0 = 2.2 \times 10^6 \text{ Pa}$	$\Delta t_{\text{MAX}} = 10^{-4} \text{ s}$
$\beta = 0.2$	$\gamma = 0 \text{ N/m}$	$K_0 = 10^7 \text{ Pa}$	$tol = 10^{-7}$
$\varepsilon = 0.05$		$\mu_c = 0.15$	
	$tol = 10^{-8}$	$tol = 10^{-6}$	
	$\lambda = 0.5$	$\rho_\infty = 0.01$	
		$k_N = 10^7 \text{ N/m}^3$	

Table 20: Simulation parameters for the **obstructed pipe** whose initial configuration is given in Figure 102. The governing equations are solved with the fractional step and the edge splitting remeshing with level set.

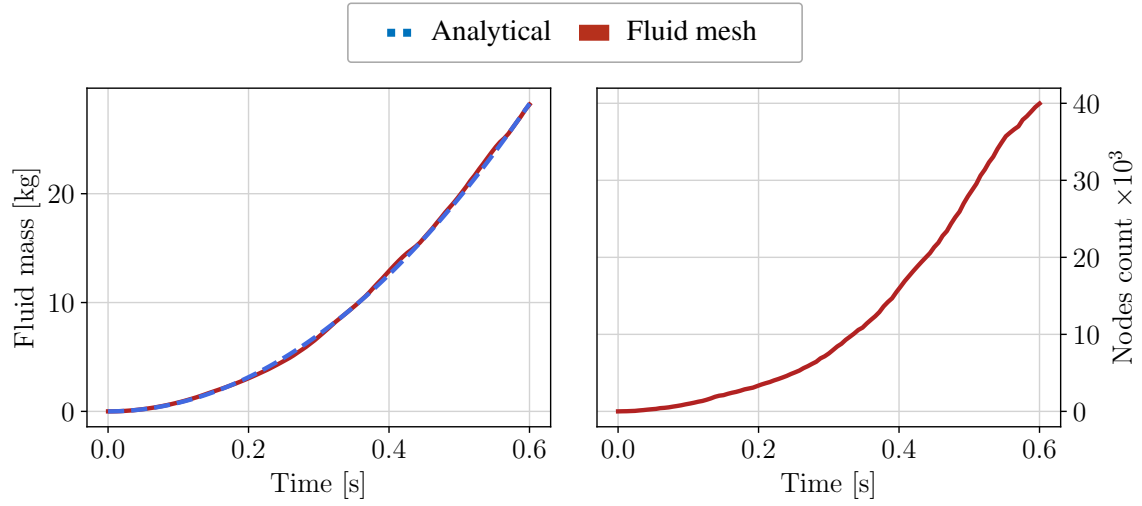


Figure 103: Evolution of the total mass of the fluid in the **obstructed pipe** and total number of nodes in the fluid mesh. The numerical results are compared to the analytical expectation from Equation (3.81).

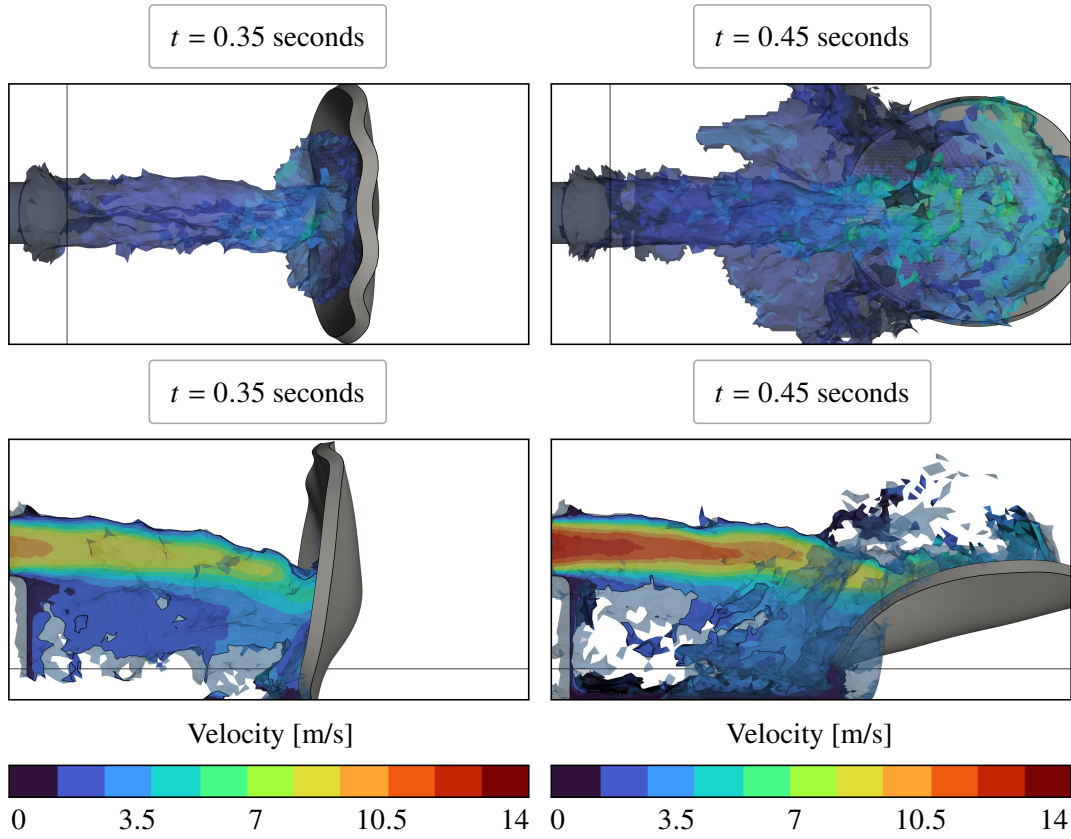


Figure 104: Snapshots of the particle finite element solution for the **obstructed pipe** test case whose initial configuration is given in Figure 102, at different time intervals. The parameters from Table 20 have been used.

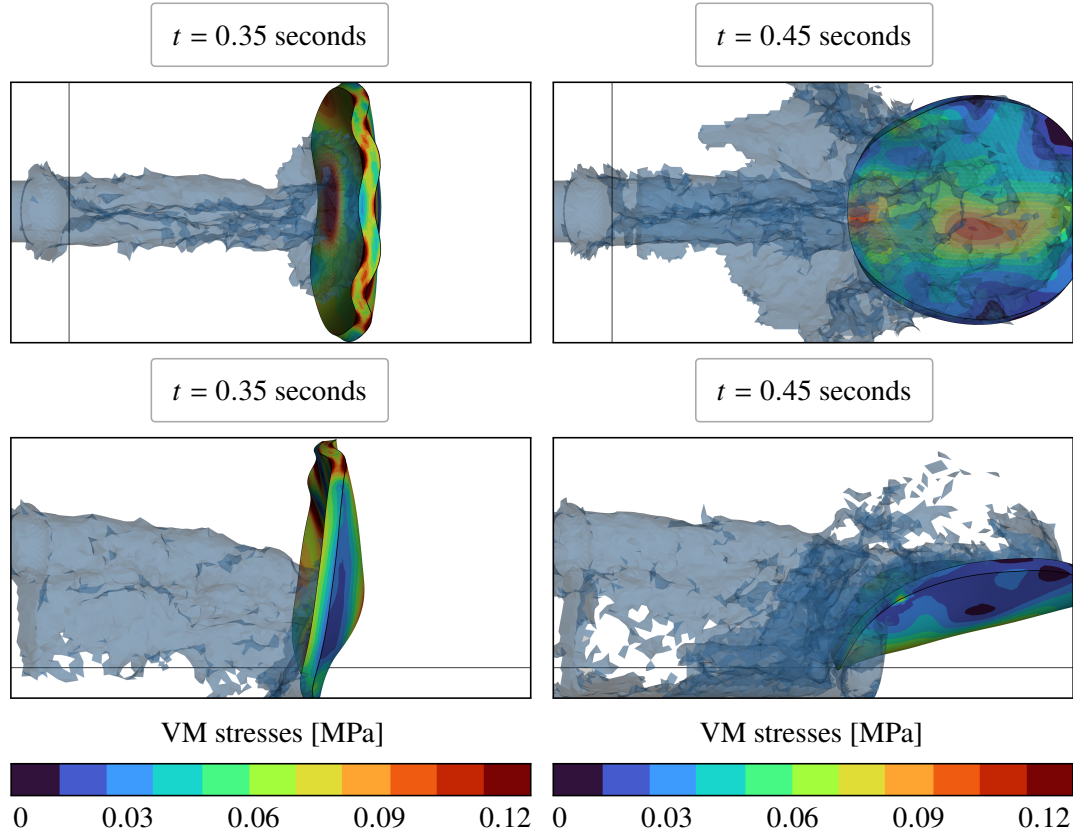


Figure 105: Snapshots of the particle finite element solution for the **obstructed pipe** test case whose initial configuration is given in Figure 102, at different time intervals. The parameters from Table 20 have been used.

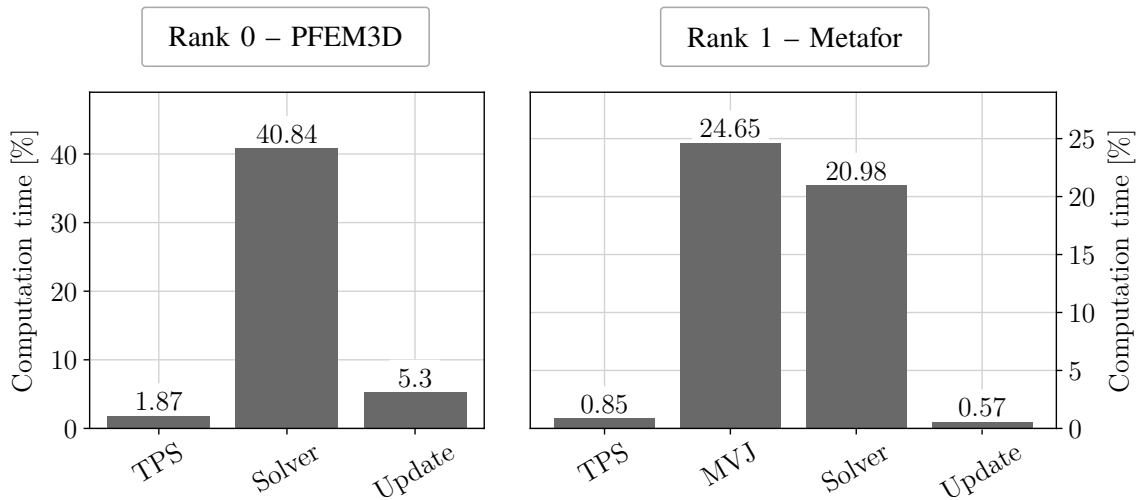


Figure 106: Shares of the total computation time (22.8 hours) for the **obstructed pipe** test case using the **IQN-MVJ** coupling. The initial configuration is given in Figure 102 and the simulation parameters are given in Table 20.

Figure 106 shows the distribution of simulation time across different software components for each MPI process. Notably, a significant portion of the runtime is spent on IQN steps, indicating potential for further optimization. In the IQN-MVJ method, most of this time is devoted to computing and operating on the inverse Jacobian in Equation (3.37). One potential strategy to reduce this cost is to skip the update of the previous Jacobian and instead rely solely on current time-step information, effectively reducing the method to IQN-ILS iterations. However, as shown in Figure 107, the total execution time for the IQN-ILS simulation is approximately 20% longer than for IQN-MVJ, primarily because the latter averages 3.6 coupling iterations per time step, whereas the IQN-ILS requires an average of 6.6 iterations to reach the same coupling tolerance, offsetting the time saved by omitting the explicit Jacobian computation. To address this issue, matrix-free variants of the IQN-MVJ, or IQN-ILS augmented by Surrogate Models (IQN-ILSM), have been proposed in the literature [97, 216] with the aim to combine the no-tuning property of the IQN-MVJ with the linear time complexity of the IQN-ILS method, while reusing information from previous time steps to improve convergence.

Nevertheless, the execution time for the structural solver increased significantly compared to the rolling tank test case, primarily due to the higher number of degrees of freedom and the contact interaction with the pipe's rigid boundary. Indeed, interaction between solid bodies is enforced by introducing a stiffness-like term that penalizes interpenetration. This method increases numerical cost because it modifies the system's stiffness matrix, making it more ill-conditioned and harder to solve.

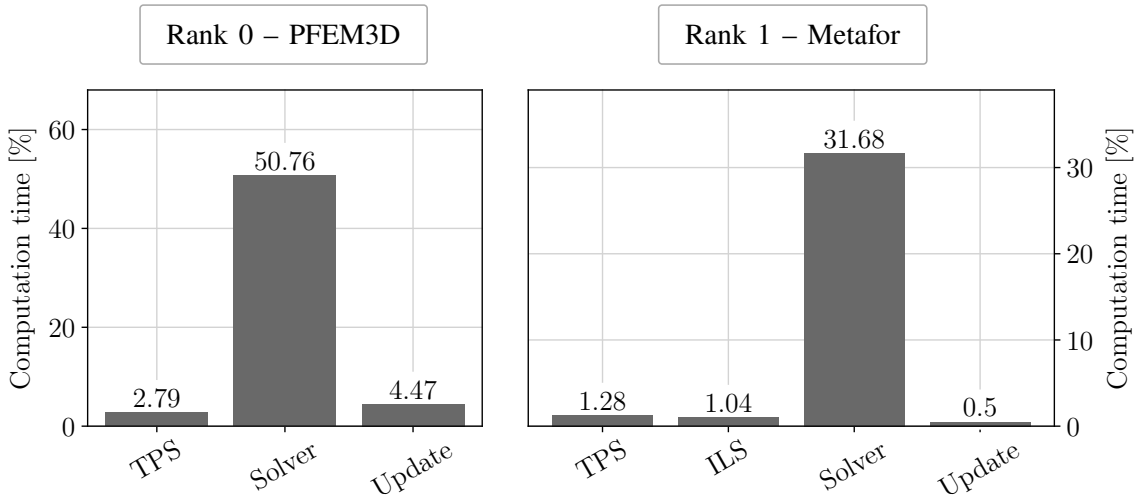


Figure 107: Shares of the total computation time (27.3 hours) for the **obstructed pipe** test case using the **IQN-ILS** coupling. The initial configuration is given in Figure 102 and the simulation parameters are given in Table 20.

A more detailed breakdown of the execution times is provided in Figures 108 and 109. In the second figure, the *Contact* category refers to the detection of contact between the deformable body and the rigid boundaries, the *Forces* category accounts for the computation of internal, external, and inertial forces, and *Restart* represents the time required to save and load the last equilibrium solution into the current model.

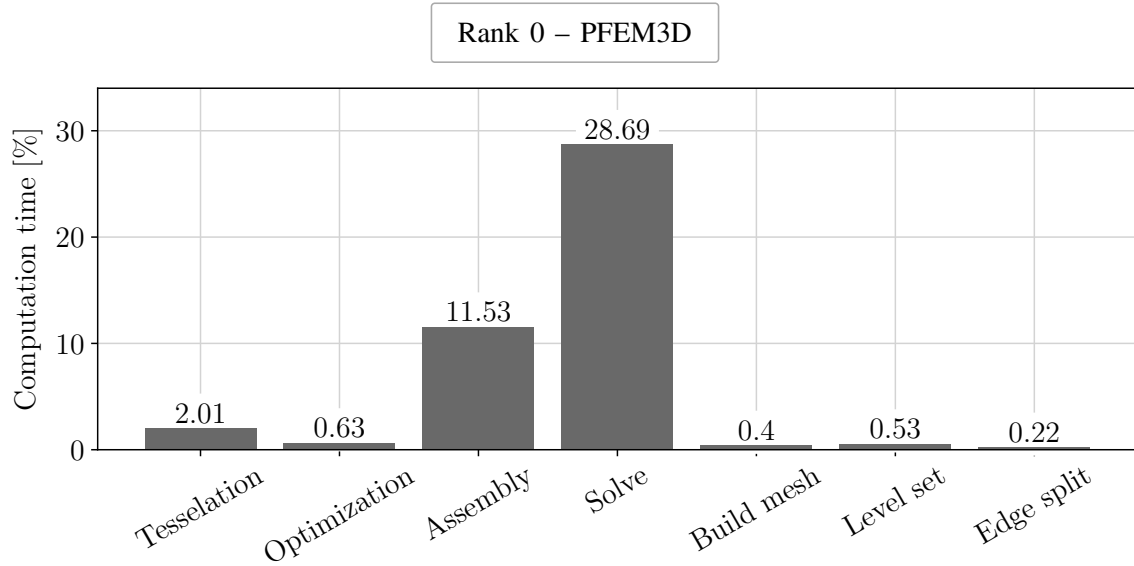


Figure 108: Shares of the total computation time (22.8 hours) for the **obstructed pipe** test case using the **IQN-MVJ** coupling. The initial configuration is given in Figure 102 and the simulation parameters are given in Table 20.

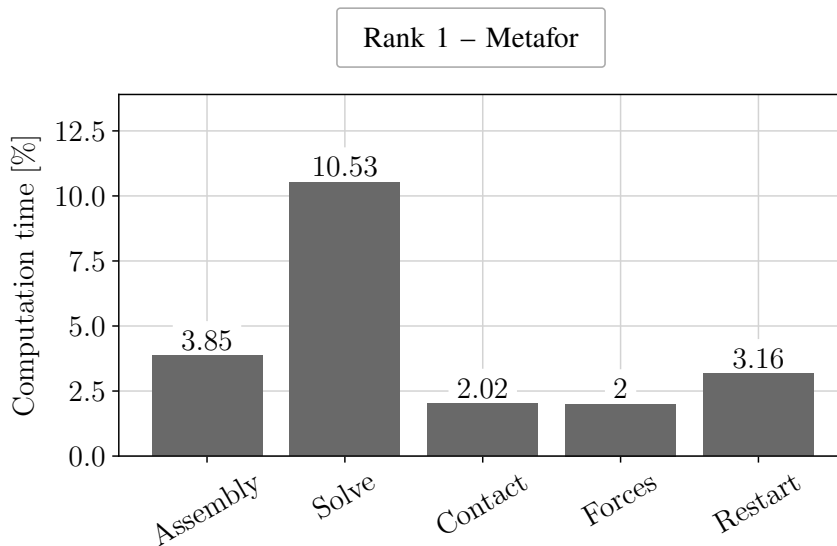


Figure 109: Shares of the total computation time (22.8 hours) for the **obstructed pipe** test case using the **IQN-MVJ** coupling. The initial configuration is given in Figure 102 and the simulation parameters are given in Table 20.

3.6 Material Rupture

Several finite element methods have been developed to model the complex physics of fracture. One example is the Element Deletion Method (EDM), which simulates material failure by removing elements that meet a predefined fracture criterion, such as the Hancock and Mackenzie criterion [217], the Johnson and Cook criterion [218] or the Bao-Wierzbicki criterion [219]. This method is easy to implement and computationally efficient, making it ideal for large-scale simulations. Additionally, EDM can effectively simulate material ejection or fragmentation resulting from impact and failure. However, the method's reliance on element removal introduces issues like mesh dependency, artificial mass loss, and challenges in accurately capturing crack paths and branching. Notably, crack branching occurs when a single crack divides into multiple paths, creating distinct fracture surfaces. In the following, we note by $\mathcal{R}$ the boolean function which indicates whether the finite element meets the given rupture criterion. In other words, the finite element e is removed from the structural mesh if

$$\mathcal{R}(e) = \text{True} \quad (3.82)$$

In contrast, the Interelement Crack Method (ICM) [220, 221] confines crack propagation to predefined element edges, with crack separation governed by a cohesive law. ICM is especially effective for modeling crack-face interactions, representing the interaction between material parts that were once connected, and for capturing fracture energy dissipation. While it offers a structured approach to crack evolution, its accuracy is highly dependent on how well the mesh aligns with the crack path, which is unknown a priori. Other examples of approaches for modelling material rupture include the Extended Finite Element Method (XFEM), which incorporate special enrichment functions that allow the modeling of cracks and discontinuities without requiring mesh refinement or remeshing [222, 223]. In this work, we will use the EDM already implemented in Metafor.

3.6.1 Element Deletion Method

Within the partitioned FSI framework of FSPC, the rupture of the solid material can only be triggered at the end of a coupling time step. This means that no solid elements are deleted from the mesh until the IQN method has fully converged to an equilibrium solution for the current time step.

This limitation is primarily technical, since changes in the number of solid elements, and consequently in the number of solid interface nodes, during the Newton-Raphson iterations can cause a dimension mismatch between the matrices used in the algorithm. For example, the IQN method depends on the least squares approximation of the Jacobian matrix, which is constructed by assembling intermediate matrices, $\mathbf{V}$ and $\mathbf{W}$, from the difference in nodal quantities between successive iterations. If material rupture occurs between iterations, the number of interface nodes may change, preventing the proper computation of columns in $\mathbf{V}$ and $\mathbf{W}$. Additionally, material rupture requires updating the RBF interpolation matrices to reflect the new interface topology, which can become computationally expensive if done repeatedly. At each FSPC time step, the following operations are performed sequentially:

- First, the root-finding problem (3.28) is solved iteratively using either the IQN-MVJ or the IQN-ILS procedure, depending on the availability of an initial Jacobian, to determine the new fluid-structure equilibrium at time t_{n+1} . During the nonlinear iterations, the solid deforms due to its interaction with the fluid, but material rupture is not triggered by the structural solver.
- Once the equilibrium configuration at t_{n+1} has been obtained, Metafor loops on its finite elements. For each element e in the solid mesh, the rupture criterion $\mathcal{R}(e)$ is evaluated. Broken elements are removed from the structural mesh, and the FS interface on the solid side is updated to reflect the new mesh topology. This is done by retrieving the set of surface elements $\mathcal{E}$ located on the new solid boundary (i.e. after material rupture), and update the vector of nodal displacements $\mathbf{u}_s$ accordingly.
- Then, the updated set of surface elements $\mathcal{E}$ is communicated to the fluid solver, which uses this information to rebuild its own FS interface based on the solid mesh topology. Remeshing of the fluid domain is then performed using the edge splitting technique, and the next time step is started.
- If topological changes in the FS interface are detected by FSPC, the mesh interpolation operators $\mathcal{V}_{\text{FS}}$ and $\mathcal{V}_{\text{SF}}$ are recomputed based on the current configuration of the FS interface nodes. Furthermore, if the detected changes are due to material rupture on the solid side, the initial Jacobian $\mathbf{J}_{\text{PRV}}$ used in the IQN-MVJ is reinitialized to zero in order to avoid dimension mismatch in (3.36).

Fluid Interface Update

The reconstruction of the FS interface on the fluid side, following material rupture on the solid side, is performed using the set of solid surface elements $\mathcal{E}$ which is communicated to the PFEM-LS solver. In practice, FSPC iterates over the fluid interface nodes and, for each node, computes its projection onto the solid surface. If the distance between the projected point and the original node exceeds a specified tolerance, the node is removed from the FS interface. It is important to clarify that removing a node from the FS interface does not mean removing it from the fluid mesh. Instead, the node's FS interface tag is cleared, effectively reclassifying it as a regular PFEM node. Such nodes typically become part of the free surface, where their displacement is no longer prescribed by the solid interface.

The second step involves iterating over the solid elements in $\mathcal{E}$. For each of these elements, including surfaces newly exposed to the fluid due to the element deletion algorithm, FSPC generates new FS interface nodes on the surface defined by these elements, provided such nodes do not already exist.

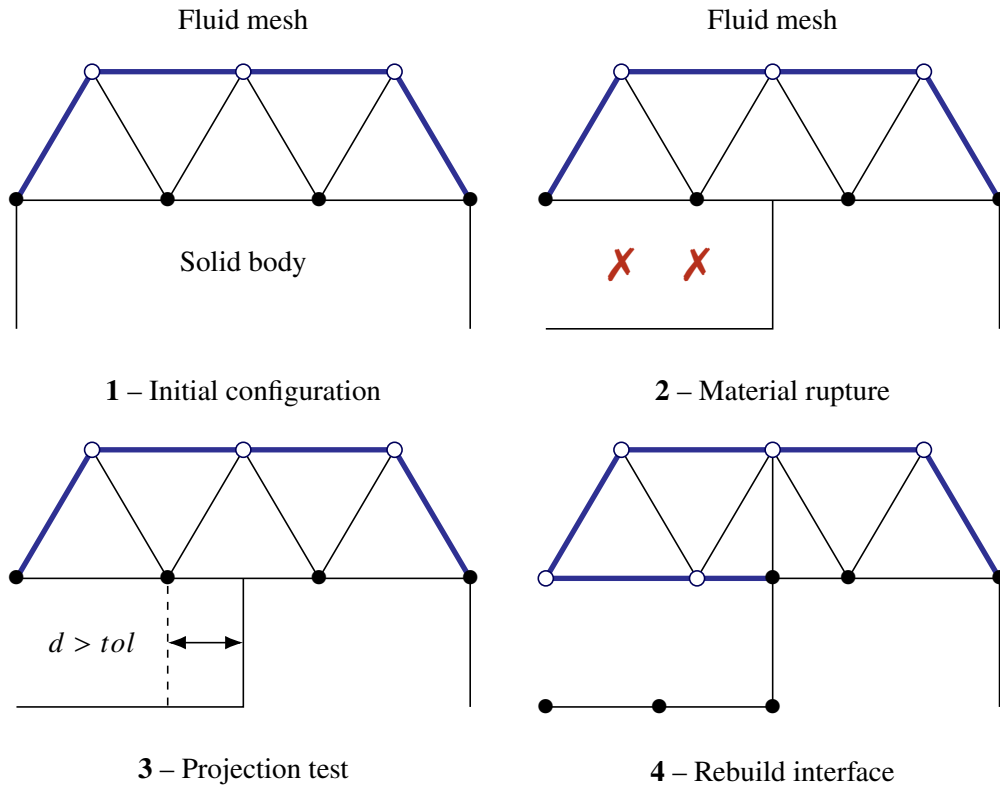


Figure 110: Example of FS interface reconstruction on the fluid side following material rupture on the solid side. The free surface is drawn in bold, boundary nodes are drawn in black and free surface nodes are drawn in white.

This step effectively reconstructs the missing portion of the FS interface on the fluid side following material rupture. If the characteristic mesh sizes of the fluid and solid differ at the interface, the new fluid nodes must be created to conform to the target size field of the PFEM mesh. As a result, the number of interface nodes generated per element in $\mathcal{E}$ depends on the mismatch between the solid element size and the desired density of fluid nodes at the FS interface. An illustration of this reconstruction process is provided in Figure 110.

Before discussing the applications, it is important to revisit the concept of exclusion zones introduced in the previous chapter. An **exclusion zone** is defined as a closed loop discretized by segments in 2D, or a closed surface discretized by polygons in 3D. In the PFEM-LS code, nodes and finite elements are not permitted within an exclusion zone. As a result, the remeshing algorithm automatically deletes any simplex generated inside this closed surface during tessellation. The primary purpose of exclusion zones is to prevent unintended fluid penetration into the solid domain, or more generally, to stop the fluid from crossing walls. This becomes especially important when dealing with thin solid structures, where tessellation may inadvertently create simplices inside the structure by connecting boundary nodes. Without exclusion zones, distinguishing these invalid simplices from legitimate fluid elements can be challenging.

In FSI simulations involving material rupture, the fluid may propagate into cracks. However, the specific geometry of these cracks, particularly the staircase artifacts introduced by the EDM method on regular hexahedral meshes, can increase the likelihood of remeshing generating spurious elements within regions of the solid that have not fractured. These unintended elements can be effectively eliminated using exclusion zones. To achieve this, the set of elements $\mathcal{E}$, which defines the updated solid boundary after rupture, is also used to update the PFEM exclusion zone. This ensures that the exclusion zone accurately reflects the current geometry of the solid, thereby preventing the creation of spurious fluid elements within intact material regions. This mechanism relies on an efficient implementation of ray-casting and AABB trees, as discussed in the previous chapter.

3.6.2 Numerical Examples

As an introductory example, we revisit the 2D dam break with obstacle presented earlier. The initial geometry is shown in Figure 52, and the simulation uses the same parameters listed in Table 6. However, a rupture criterion is now introduced into the structural model. This criterion is based on a critical threshold of equivalent plastic strain within each element. Specifically, a solid element is removed from the mesh at time t_{n+1} when the equivalent plastic strain $\bar{\epsilon}_{vp}$ exceeds a user-defined limit:

$$\bar{\varepsilon}_{VP}(t_n) + \int_{t_n}^{t_{n+1}} \sqrt{\frac{2}{3}} \mathbf{D}_{VP} : \mathbf{D}_{VP} dt > 0.1 \quad (3.83)$$

Here, $\mathbf{D}_{VP}$ denotes the viscoplastic strain rate tensor. The equivalent plastic strain is computed at the integration points of each element, with the element-averaged value used in the rupture criterion. It is important to note that equation (3.83) is not intended to model a realistic material rupture, but rather to demonstrate the capability of our FSI framework to handle significant topological changes in the solid structure. Snapshots of the solution at various time intervals are shown in Figure 111. As observed, rupture initiates at the base of the obstacle near the clamped region, where stress concentration and plastic deformation are the highest due to constraints and fluid-induced reaction forces. After rupture, the detached portion of the obstacle is carried by the fluid and eventually contacts the ground. This contact is modeled using the Coulomb law (3.17) with $\mu_c = 0.15$.

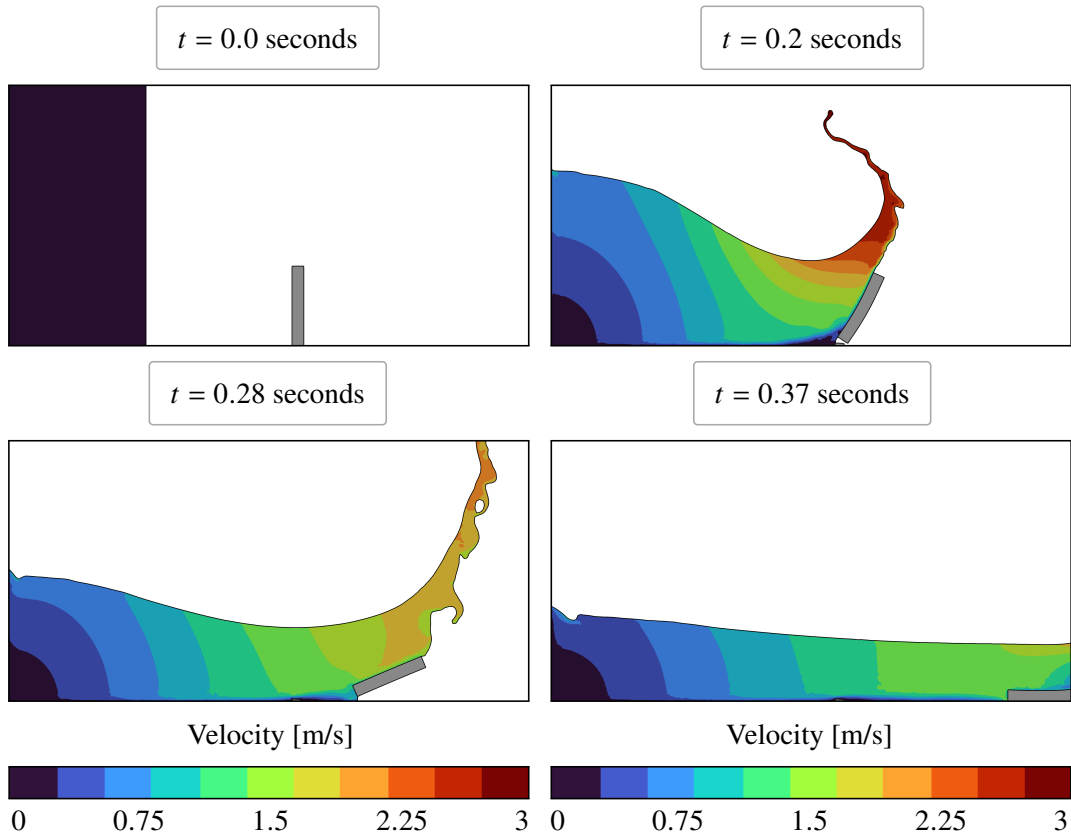


Figure 111: Snapshots of the particle finite element solution for the **dam break** test case whose initial configuration is given in Figure 52, at different time intervals. The parameters from Table 6 have been used.

Water-Jet Cutting

The capability of FSPC to handle material rupture using EDM is further evaluated through a water-jet cutting test case. Water-jet cutting is a manufacturing process that uses a high-pressure stream of water, often mixed with abrasives, to erode and cut through materials with precision. It is especially valued for producing no heat, making it ideal for cutting temperature-sensitive materials. In this simulation, the solid undergoes extreme erosion, leading to rapid degradation of the contact region between the fluid and the solid. This scenario presents a challenging test case for FSI frameworks, as it involves significant and continuous topological changes in the solid surface.

However, water-jet cutting typically operates in a hypersonic regime, where the assumption of incompressible or weakly compressible Newtonian flow no longer holds. Consequently, the governing equations presented in Chapter 2 are not applicable in this scenario. Nonetheless, a fictitious test case can still be created as a proof of concept. Specifically, the following example highlights our framework's ability to simulate the progressive erosion of material through its strong interaction with the fluid flow, but does not aim to replicate real-life water-jet cutting scenarios.

The initial configuration of the problem is shown in Figure 112. The simulation is carried out using an axisymmetric formulation, with the axis of symmetry aligned along the vertical axis. Additionally, rigid-body motion and rotations of the structure are constrained. The prescribed fluid velocity at the inlet is initially zero and a constant acceleration of 1000 m/s^2 is imposed. The solid structure is modeled as an elastoplastic material, whose isotropic hardening is governed by the Swift law:

$$\sigma_Y = \sigma_Y^0 + H \bar{\epsilon}_{vp}^n \quad (3.84)$$

Here, σ_Y denotes the current yield stress, σ_Y^0 is the initial yield stress, H and n are user-defined material parameters, and $\bar{\epsilon}_{vp}$ is the effective plastic strain. Two uniform mesh refinement levels are considered. The coarse mesh uses a characteristic size of $h_F = 1.5 \text{ cm}$ for the fluid and $h_s = 1 \text{ cm}$ for the solid. The fine mesh adopts a uniform size of $h_F = h_s = 0.5 \text{ cm}$. The solid domain is discretized using a structured grid of regular SRIPR quadrilateral elements. For simplicity, the rupture criterion applied is the same as defined in Equation (3.83).

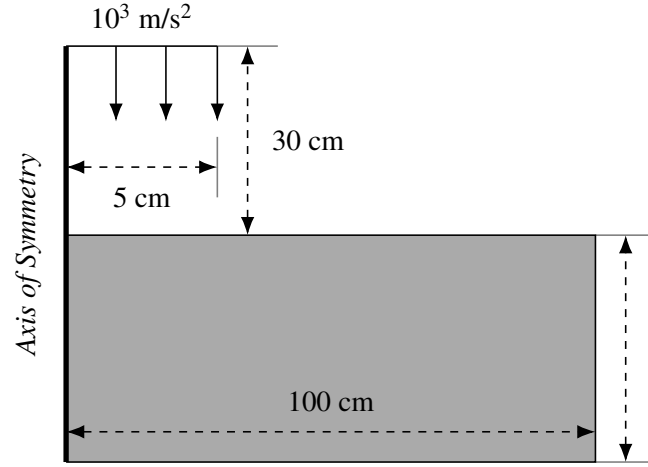


Figure 112: Initial configuration of the **water-jet cutting** test case whose simulation parameters are indicated in Table 21. No-slip BC are applied to rigid and deformable boundaries, and the axis of symmetry is outlined in bold.

<i>Remesh</i>	<i>Fluid</i>	<i>Solid</i>	<i>Coupling</i>
$\delta = 0.6$ $\omega = 0.6$ $\alpha = 1.2$ $\beta = 0.2$ $\varepsilon = 0.1$	$\rho = 1000 \text{ kg/m}^3$ $g = -9.81 \text{ m/s}^2$ $\mu = 10^{-3} \text{ Pa s}$ $\gamma = 0.07 \text{ N/m}$	$\rho = 2700 \text{ kg/m}^3$ $g = -9.81 \text{ m/s}^2$ $\nu = 0$ $E = 7 \times 10^{10} \text{ Pa}$ $H = 375 \text{ Pa}$ $\sigma_Y^0 = 10^6 \text{ Pa}$ $n = 0.2$	$\varphi = \text{TPS}$ $R = 0.1 \text{ m}$ $\Delta t_{\text{MAX}} = 10^{-4} \text{ s}$ $tol = 10^{-7}$
	$tol = 10^{-6}$	$tol = 10^{-5}$ $\rho_{\infty} = 0.01$	

Table 21: Simulation parameters for the **water-jet cutting** whose initial configuration is given in Figure 112. The governing equations are solved with the PSPG stabilization and the edge splitting remeshing with level set.

Snapshots of the FS contact region obtained using the fine mesh are shown in Figure 113. These images demonstrate that the fluid interface update procedure, outlined in Figure 110, effectively tracks the evolving solid boundary as Metafor removes elements that satisfy the rupture criterion. In real-world water-jet cutting, the cut width is not perfectly uniform but typically stays within a predictable range. This behavior is captured in the numerical results shown in Figure 113. Pure water jets, which rely solely on high-pressure water, tend to produce narrower cuts.

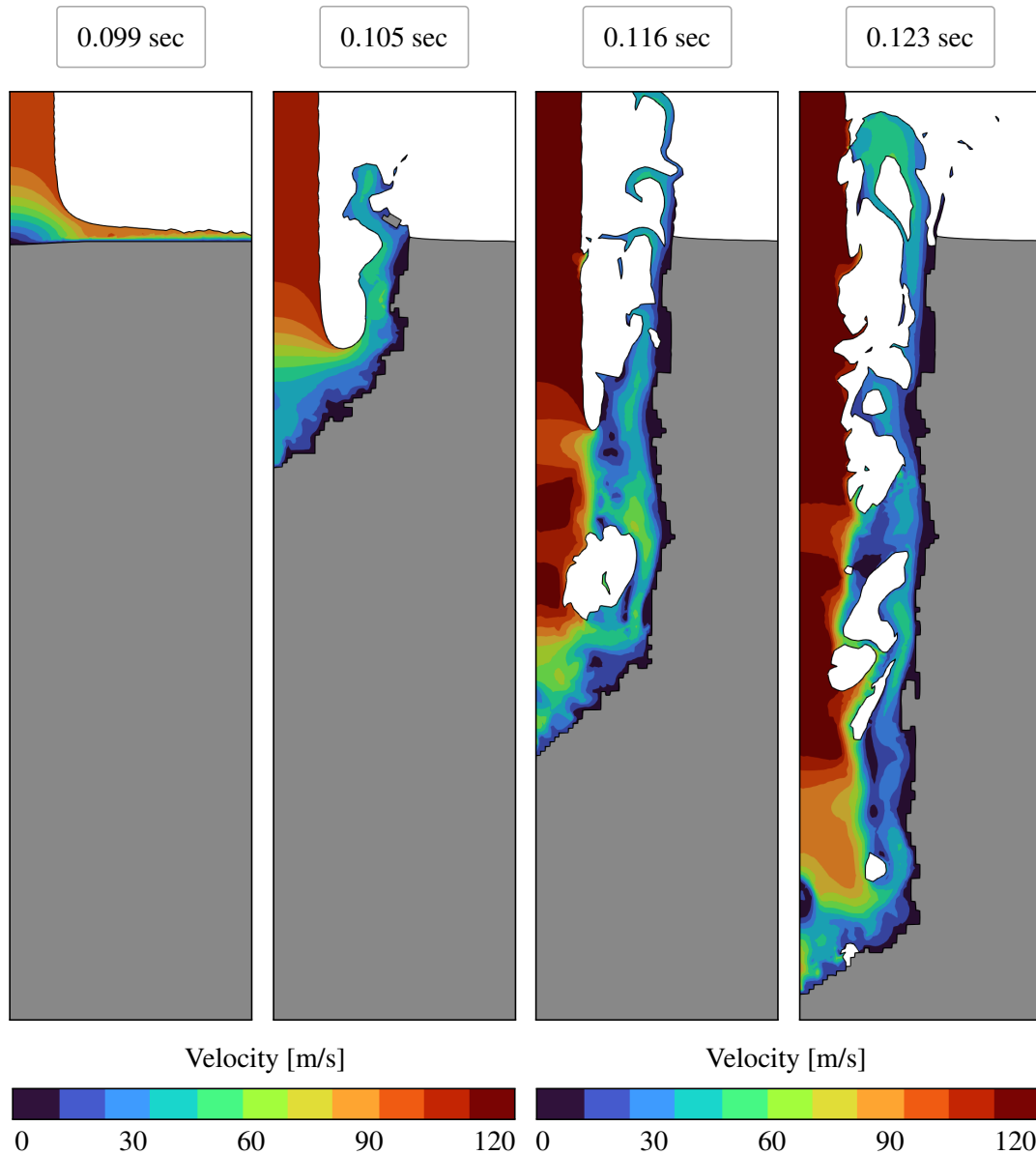


Figure 113: Snapshots of the particle finite element solution for the **water-jet cutting** test case whose initial configuration is given in Figure 112, at different time intervals. The parameters from Table 21 have been used.

In contrast, abrasive jets, enhanced with abrasive particles, can cut through harder or thicker materials and distribute force over a wider area, resulting in broader cuts. Figure 114 illustrates the time evolution of the hole's depth and volume within the solid material. Those results show that the depth of the hole is highly dependent on the mesh density. Indeed, EDM leads to strain localization into the smallest admissible region allowed by the mesh, typically a single or a few elements. As the mesh is refined, the effective region over which failure localizes decreases, and so does the total energy dissipated, unless special regularization is introduced. This results in mesh-dependent fracture energy, with finer meshes producing more brittle responses and coarser meshes overestimating ductility or toughness.

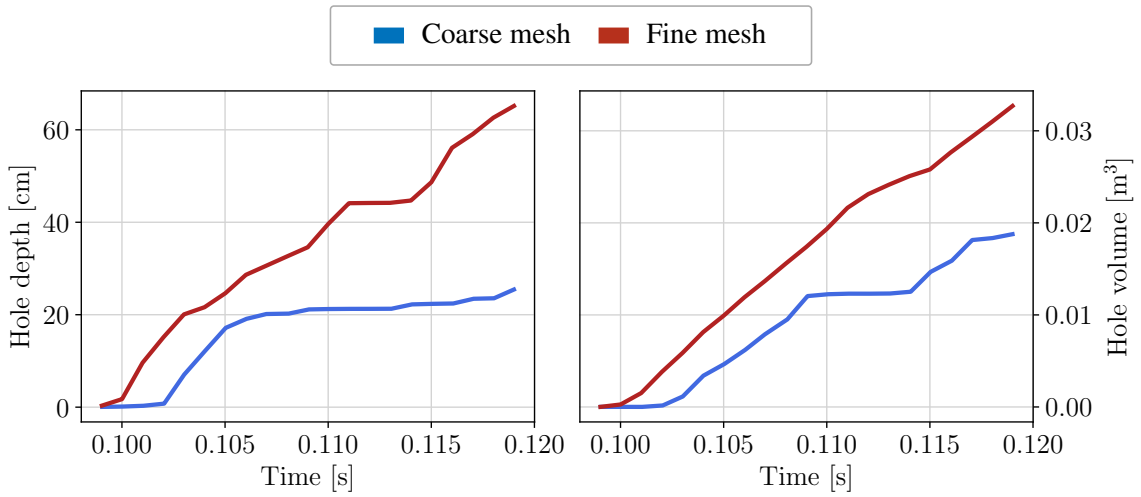


Figure 114: Time evolution of the depth and volume of the hole excavated by the fluid into the solid structure, for different mesh sizes. The volume is computed by a rotational extrusion of the hole's area around the axis of symmetry.

3.7 Example – Weak Compressibility

In a test case simulated by Anger et al. [224], an initially cylindrical hyperelastic body in 2D plane strain is compressed through a hyperelastic ridge by the force of the surrounding fluid flow. In [224], this setup was designed to evaluate the formulation's ability, using a monolithic non-interface-fitted cut finite element method, to handle strong interactions between the FS interface and the contact interface, including large contact areas and significant topological changes. The initial configuration is shown in Figure 115. Following [224], all physical quantities are nondimensional. The imposed normal velocity $\bar{v}_N$ at the inlet boundary follows a parabolic profile:

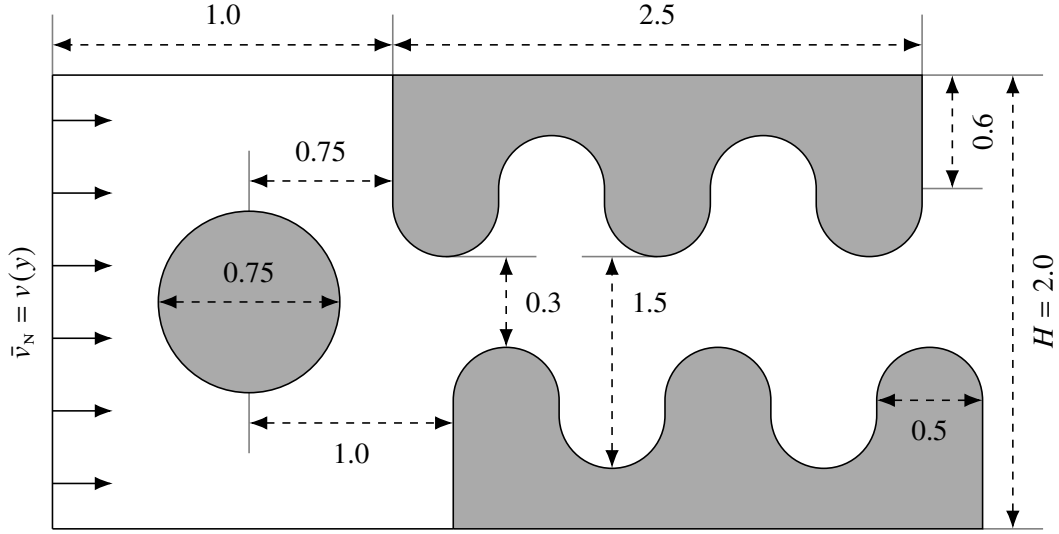


Figure 115: Initial configuration of the **squeezed disk** test case whose simulation parameters are indicated in Table 22. No-slip BC are applied to rigid and deformable boundaries, and the fluid domain is closed without free surface.

<i>Remesh</i>	<i>Fluid</i>	<i>Solid</i>	<i>Coupling</i>
$\delta = 0.6$	$\rho_0 = 10^{-6}$ [-]	$\rho = 10^{-6}$ [-]	$\varphi = \text{TPS}$
$\omega = 0.6$	$g = 0$ [-]	$g = 0$ [-]	$R = 0.1$ [-]
$\alpha = 1.2$	$\mu = 10^{-5}$ [-]	$E_D = 100$ [-]	$\Delta t_{\text{MAX}} = 10^{-4}$ [-]
$\beta = 0.2$	$\gamma = 0$ [-]	$E_W = 200$ [-]	$tol = 10^{-6}$
$\varepsilon = \infty$	$K_0 = 100$ [-] $K' = 1$ [-]	$\nu = 0.3$ $\mu_c = 0.15$	
	$tol = 10^{-6}$	$tol = 10^{-4}$ $\rho_\infty = 0.01$ $k_N = 10^4$ [-]	

Table 22: Simulation parameters for the **squeezed disk** whose initial configuration is given in Figure 115. The governing equations are solved with the PSPG stabilization and the edge splitting remeshing with level set.

$$v(y) = v_{\text{MAX}} \left[1 - \frac{(y - H/2)^2}{(H/2)^2} \right] \quad (3.85)$$

Here, y denotes the vertical coordinate, H is the height of the pipe, and $v_{\text{MAX}} = 100$. The primary challenge in this test case, within our partitioned framework, arises from the interplay between the boundary conditions and the fluid's constitutive model. Specifically, the setup includes an imposed velocity at the inlet, no-slip boundary conditions on both rigid and deformable walls, and a solid body obstructing the outlet. Under these conditions, the fluid between the inlet and the obstructed outlet becomes fully constrained by Dirichlet BC on velocity, as illustrated in Figure 116. For an incompressible fluid, this configuration demands exact compatibility between the prescribed inlet velocity and the structure's predicted velocity. Any mismatch would result in fluid compression or expansion, leading to infinite pressure due to the incompressibility condition. In our partitioned FSI framework, such compatibility may not be achieved until the fluid-structure system reaches equilibrium. As a result, the coupling scheme can become unstable as soon as the disk begins to obstruct the outlet [225].

One way to address this issue is to use the weakly compressible fluid formulation introduced in the previous chapter. Allowing for slight compressibility relaxes the compatibility constraint between the inlet velocity and the structural displacement, enabling the IQN iterative coupling to reach an equilibrium state more effectively. The solid structure is modeled using the Neo-Hookean strain-energy function (3.80). Although Anger et al. [224] do not provide the values of G_0 and K_0 directly, they can be derived from the elastic modulus E and Poisson's ratio ν given in the study, using the following relations:

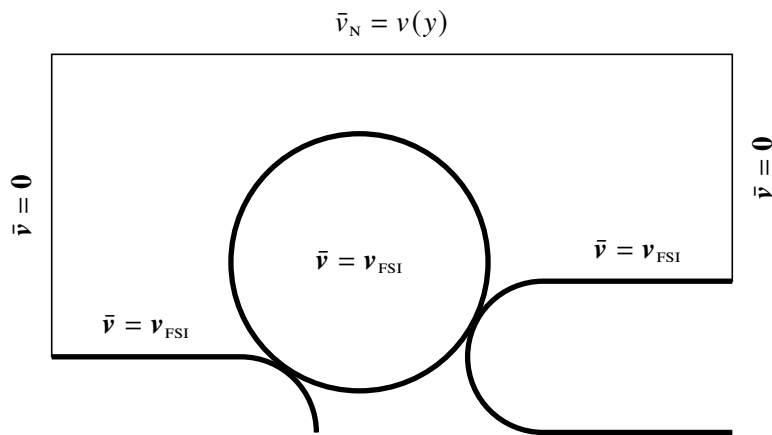


Figure 116: Fluid domain located between the inlet of the pipe, the rigid walls and the obstructed outlet, achieved once the disk enters into contact with the ridge. In such case, all boundaries of the fluid are constrained by Dirichlet BC for the velocity.

$$G_0 = \frac{E}{2(1 + \nu)}$$

$$K_0 = \frac{E}{3(1 - 2\nu)}$$
(3.86)

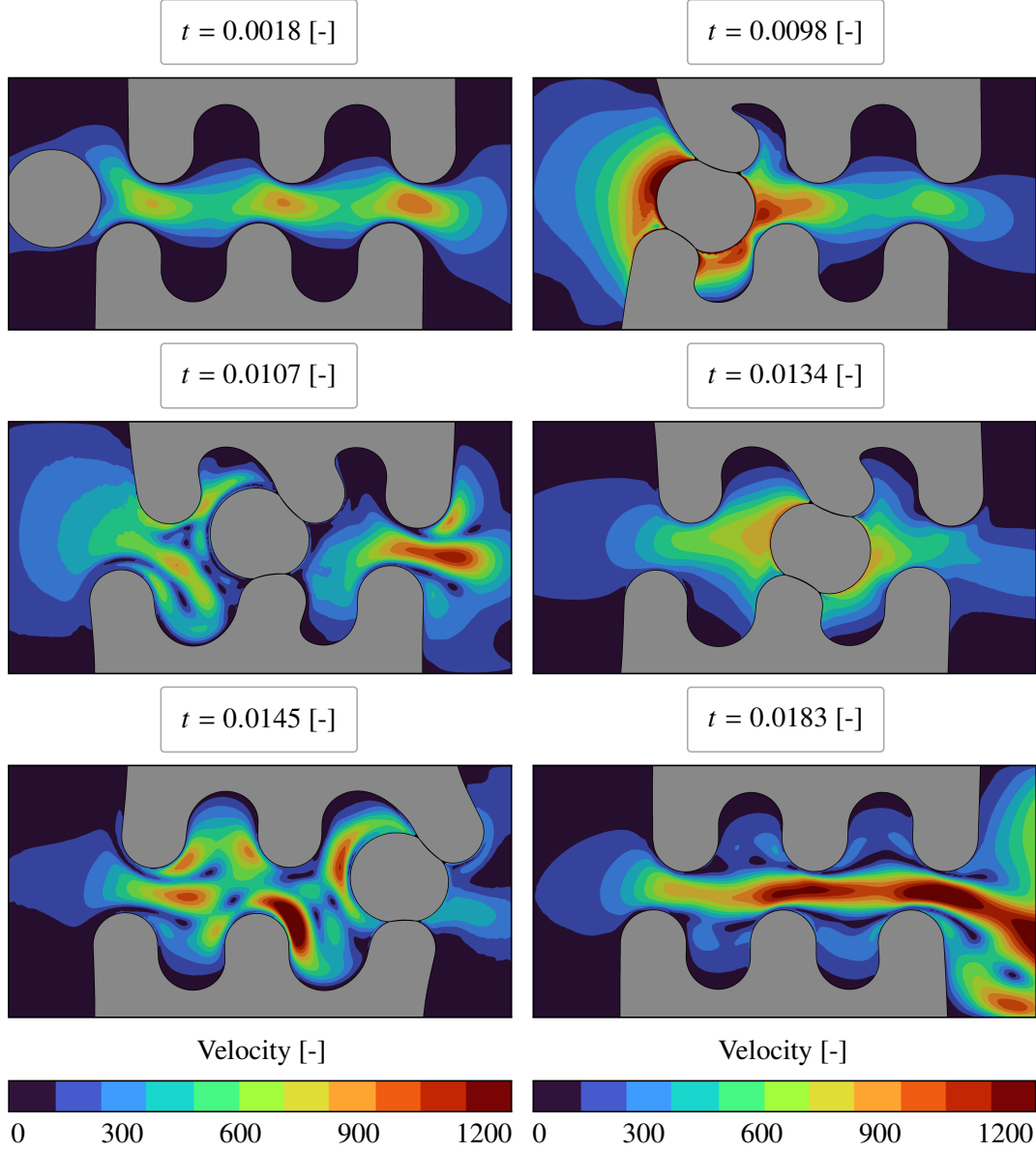


Figure 117: Snapshots of the particle finite element solution for the **squeezed disk** test case whose initial configuration is given in Figure 115, at different time intervals. The parameters from Table 22 have been used.

The simulation parameters for the weakly compressible fluid and the hyperelastic structure are summarized in Table 22. Here, E_D and E_w represent the elastic moduli of the disk and the flexible walls, respectively, while the Poisson's ratio $\nu = 0.3$ is common to both structures. Snapshots of the simulation at various time intervals are shown in Figure 117. As in the aortic valve test case, and as previously noted by Anger et al. in [224], the topology of the fluid domain evolves from a single connected region to two separate fluid regions once the structures come into contact. The fluid domain comes back to a single-connected region once the disk exits the ridge structure.

Chapter 4

Conclusions

4.1 Overview

In this thesis, the PFEM-FEM partitioned FSI framework was extended to support 3D configurations, addressing key limitations of the PFEM and enabling simulations of complex, dynamic loading conditions on the solid structure, including thermo-mechanical interactions. Particular attention has been given to the performance and scalability of the implementation. The proposed framework has been validated against a wide range of experimental and numerical benchmarks from the literature. These benchmarks include the sloshing of a water tank, splashing and droplet separation, a dam break with an elasto-viscoplastic deformable obstacle, the flow-induced deformation of hyperelastic structures, the floating and warming of a solid sphere in a water tank, the dynamics of blood ejection from the ventricle into the aorta, the large deformation of obstructing structures into a pipe, including solid-solid contact interactions, and a test case of water-jet cutting featuring extreme topological changes induced by material rupture. The variety of test cases and their agreement with published results in both 2D and 3D confirm the robustness of the approach.

Related Works

To further illustrate the applicability of the method on real-life test cases, this subsection highlighting how the FSI coupling framework and PFEM3D, enhanced through the contributions of this work, have been adopted by other researchers. A curated list of publications and related projects is thus provided to showcase additional applications of the codes and to guide readers interested in exploring these extensions. A first study by Février et al. [100] presents a simulation of melt pool dynamics in the laser powder bed fusion process, including phase changes and vaporization using PFEM3D alone. A second publication by Fernández et al. [132] models melt pool dynamics in aluminum-to-steel welds via friction melt bonding. Examples of experimental setup and of the melt pool dynamics are presented in Figures 118 and 119. The simulations have been performed by coupling PFEM3D and Metafor through FSPC. More recently, Delhez et al. [101, 226] investigated the interaction between non-Newtonian blood flow and arterial walls, including aneurysm rupture and blood ejection, using the present FSI framework. An example of rupture of cerebral aneurysm, taken from the study, is presented in Figure 120. Finally, Claeskens [227] studied thermal battery charging involving a phase-change material, also achieved by coupling PFEM3D and Metafor through FSPC.

4.2 Conclusions of Chapter 2

Chapter 2 introduced the PFEM-LS as a robust framework for solving fluid dynamics problems involving large deformations and free surface flows. The methodology was presented in detail, beginning with the discretization of the Navier-Stokes equations using the PSPG and fractional step methods. The second part of the chapter focused on the remeshing procedure, addressing three primary objectives. To the best of the author's knowledge, the proposed solutions represent original contributions to the PFEM field. Each has been validated through a series of 2D and 3D benchmark tests drawn from the existing literature. The conclusions for each objective are summarized below:

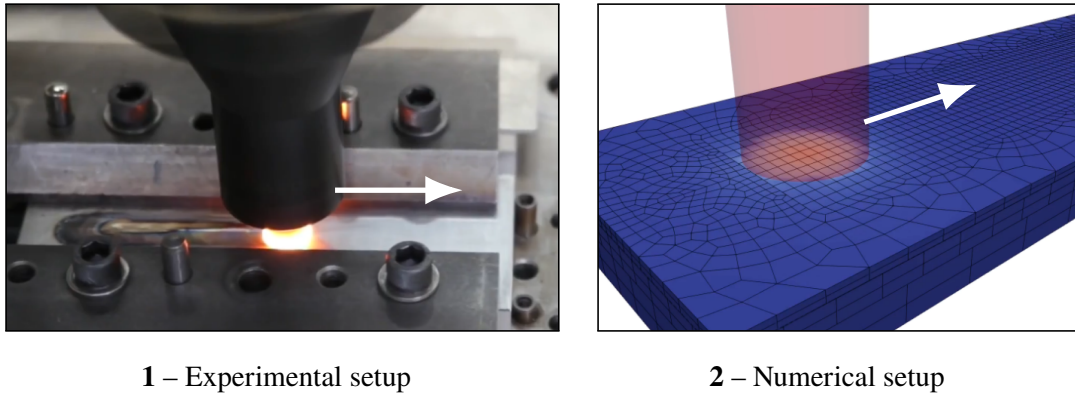


Figure 118: Example of experimental setup where a rotating tool is used to weld two metallic plates. On the right, the 3D numerical setup presented in the study of Fernández et al. [132]. The arrow indicates the direction of the tool displacement.

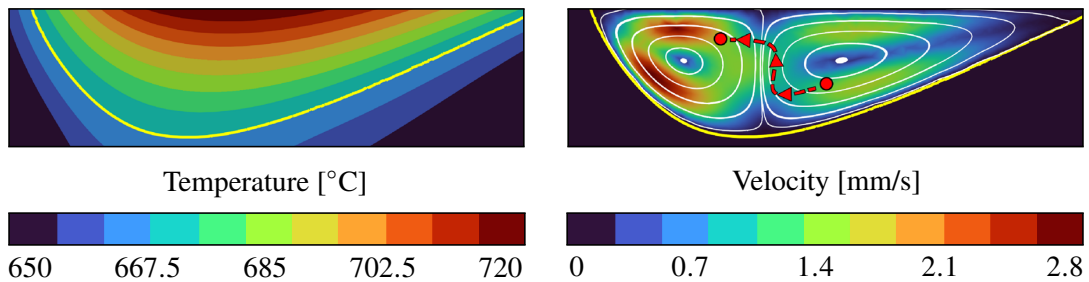


Figure 119: Example of temperature and velocity profiles of the fluid inside the melt pool of the aluminum plate located between the two steel plates. The figures have been taken from the work of Fernández et al. [132].

1 – Improved Mesh-Refinement in 2D/3D

The first objective focused on developing an automatic mesh refinement and coarsening procedure between successive remeshing steps, applicable to both 2D and 3D geometries. As nodes are displaced by the flow, mesh resolution within the fluid domain becomes uneven, creating sparse regions where finer resolution is needed, and dense areas where it is not. The proposed method, based on edge splitting, generates higher-quality elements than the traditional centroid insertion approach and produces smoother free surfaces. It also ensures that node distribution, and therefore mesh resolution, follows a user-defined size-field $h(\mathbf{x})$. This size-field is typically designed to provide finer resolution near critical regions, such as free surfaces, and coarser resolution elsewhere. In many industrial applications, especially in 3D, the use of $h(\mathbf{x})$ is crucial for reducing the number of degrees of freedom, and thus execution time, without sacrificing solution accuracy.

The edge splitting refinement method relies on several user-defined parameters. The parameter β sets the minimum allowable distance between a fluid node and a solid boundary. While it has little effect on mesh quality or density within the fluid, using $\beta = 0.2$ helps prevent nodes from inadvertently crossing into solid regions. In contrast, γ and ω are key to controlling node density. Results show that $\gamma = 0.6$ and $\omega = 0.6$ in 2D, and $\gamma = 0.8$ and $\omega = 0.4$ in 3D, yield the closest match between the user-defined size field and the actual element sizes in the mesh. These optimal values were identified based on meshes generated using the Gmsh software [93].

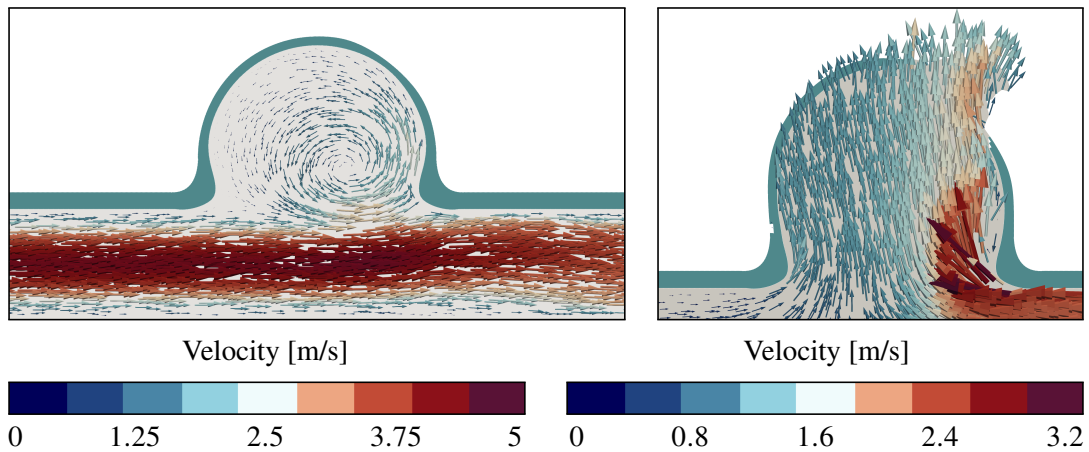


Figure 120: Example of velocity field in a cerebral aneurysm obtained with the present PFEM-FEM coupling framework. The second figure shows the rupture of the vessel's wall due to internal pressure. The figures have been taken from the work of Delhez [101].

2 – Regularization of Mesh Degeneracies

The second objective addressed the regularization of degenerate tetrahedra (slivers) that arise during Delaunay tetrahedralization of fluid nodes in 3D. Delaunay tessellation is a key step in the remeshing process, as it updates mesh connectivity to maintain well-shaped elements throughout the simulation. Regularization is achieved through local edge and face flips, which adjust connectivity between neighboring elements, and vertex smoothing, which repositions vertices away from flat configurations to reshape slivers into better-formed tetrahedra. In this work, the open-source libraries Gmsh [93] and TetGen [136] was used to successfully perform Delaunay tessellation and mesh regularization.

3 – Improved Mass Conservation

The third objective addressed the issue of mass variation introduced by the remeshing process. When two free surfaces come into proximity, the Delaunay tessellation may generate connecting simplices, allowing separate fluid bodies to merge upon contact. However, this can lead to artificial mass creation, as these newly formed elements do not correspond to any actual volume in the last-known configuration of the fluid. To mitigate this issue, an efficient implementation of the PFEM-LS approach was proposed and extended to 3D geometries. This method introduces a signed distance function as an additional filtering criterion to constrain the Delaunay tessellation, using the previous fluid domain as a reference. As a result, it delays the merging of fluid bodies until they are sufficiently close, significantly reducing spurious mass generation compared to the classical PFEM formulation.

The PFEM-LS introduces two user-defined parameters, referred to as α and ε . The first corresponds to the conventional α -shape criterion and is typically set to $\alpha = 1.2$ in 2D, as supported by the literature. By analogy, this work shows that $\alpha = 1.7$ offers a comparable level of tolerance in 3D. The second parameter is associated with the LS criterion and plays a critical role in controlling mass variation. Its optimal value generally falls between $\varepsilon = 0$ and $\varepsilon = 0.1$ for free surface flows, depending on the specific test case, and cannot be determined in advance. In theory, setting $\varepsilon = 0$ is expected to yield the most physically accurate results, as it prevents fluid domains from merging until they are in direct contact. This is achieved by disallowing the creation of elements whose centroids lie outside the most recently identified fluid domain. However, while the PFEM-LS approach effectively reduces artificial mass gain, it does not address non-physical mass loss resulting from fluid separation. Consequently, an excessively small ε may lead to gradual fluid loss over time due to an imbalance between mass gain and loss.

4.3 Conclusions of Chapter 3

Chapter 3 introduced the Dirichlet-Neumann (DN) partitioned coupling strategy as an effective framework for simulating complex and dynamic loading conditions on a 3D solid structure. These include thermo-mechanical interactions, free surface flows, material rupture, and fluid flow through fractured zones. The methodology was presented in a structured manner, beginning with an overview of the C++/Python multi-code coupling software developed in this work, followed by a detailed explanation of the IQN method in its ILS and MVJ variants. Each contribution to the FSI framework was validated using a series of 2D and 3D benchmark tests from the literature. The conclusions for each objective are summarized below:

4 – *Multi-Vector Jacobian Coupling*

A Python implementation of the IQN-MVJ coupling algorithm was evaluated against the IQN-ILS method with Haelterman filtering, using benchmark tests from the literature. Although the IQN-MVJ method itself is not original to this thesis, its implementation and evaluation within a 3D PFEM-FEM partitioned coupling framework represent a meaningful contribution to FSI research. For the benchmark cases considered, the IQN-MVJ method showed a convergence rate comparable to, or better than, a well-tuned IQN-ILS, without requiring filtering or manual tuning of input parameters. The main drawback of the MVJ approach is the need to explicitly compute the Jacobian matrix. When the number of structural interface nodes N_s is small compared to the total number of nodes in the fluid and solid domains, this overhead is negligible. However, as N_s increases and becomes comparable to the overall node count, the cost of computing the approximation of the inverse Jacobian becomes significant. Ultimately, the choice between IQN-MVJ and IQN-ILS involves a trade-off between the cost of computing the inverse Jacobian, and the need for filtering and manual tuning. Nevertheless, the IQN-MVJ remains advantageous in most cases. As a side remark, alternative IQN methods have been recently proposed in the literature [97, 216] with the aim to combine the no-tuning property of the IQN-MVJ with the linear time complexity of the IQN-ILS method.

5 – *PFEM-FEM with Non-Matching Meshes*

This contribution focused on simulating FSI with non-matching meshes, accounting for frequent topological changes at the interface. These changes primarily stem from two sources: The remeshing process, which may add, remove, or reorder nodes, and topological modifications in the solid mesh due to material rupture.

Although the Radial Basis Function (RBF) interpolation method is not original to this thesis, its implementation and evaluation within the 3D PFEM-FEM partitioned coupling framework represent a meaningful advancement in FSI research. A key conclusion from this work is that consistent interpolation is generally preferable to conservative interpolation, an observation that aligns with findings reported by de Boer et al. [193] in quasi-1D settings. The consistent approach avoids non-physical oscillations in interpolated nodal loads, which often arise when the target mesh is as fine as, or finer than, the source mesh. This behavior occurs because conservative interpolation tends to concentrate loads on solid nodes adjacent to fluid nodes, leaving others unloaded. When solid elements are smaller, the magnitude of these concentrated loads increases to conserve total energy at the interface. The sole exception to this guideline arises when the solid mesh lacks sufficient resolution to capture local variations in fluid loads at the FS interface. In such cases, conservative interpolation can be beneficial, as it redistributes excess loads to neighboring nodes, thereby improving the conservation of total work.

The second major conclusion concerns the choice of the RBF. Among those tested, the Thin Plate Spline (TPS) demonstrated the most robust performance, consistently delivering accurate results across a wide range of characteristic radius values ($R/h_F = 0.01$ to 20). Increasing R generally improves the condition number of the TPS interpolation matrix. In comparison, the C2 function achieves acceptable accuracy only when $R/h_F > 6$ and can enable the construction of sparse matrices. The level of sparsity is proportional to N_R/N , where N_R is the number of nodes within the cut-off radius, and N is the total number of interface nodes. In the context of PFEM-FEM coupling, TPS is particularly advantageous, as its accuracy remains unaffected by variations in R , making it well-suited for non-uniform and evolving node distributions at the FS interface. In contrast, the C2 function would require dynamically adapting R to local node densities or using a globally large R , the latter of which diminishes the primary benefit of C2 by reducing the sparsity of the resulting matrices.

The execution time for the interpolation remains negligible when the number of nodes on the FS interface is small relative to the total number of nodes in the fluid and solid meshes. However, it accounts for a larger percentage of the total simulation time as N_s and N_F approach the overall mesh size. Frequent changes in mesh topology can further increase the cost of updating the interpolation matrices and their factorizations. Nonetheless, interpolation time remained low across all tested configurations when using the DSYTRF and DSYTRS routines from Lapack [209].

6 – *Thermo-Mechanical FSI Coupling*

Thermo-mechanical PFEM-FEM coupling has been achieved by applying the same DN strategy used in the mechanical interface, substituting nodal temperatures for displacements and nodal heat fluxes for tractions. This approach, referred to as the FFTB scheme, has been implemented in FSPC throughout this thesis to extend and validate the proposed framework across a wider range of applications. Thermo-mechanical equilibrium is reached iteratively using the IQN-MVJ method. A key advantage of this unified coupling strategy is that thermal and mechanical quantities, temperatures and displacements, heat fluxes and loads, can be exchanged simultaneously between the fluid and solid solvers with minimal additional computational cost. This enables seamless and efficient thermo-mechanical simulations. The FFTB scheme has been successfully validated against several benchmark cases from the literature.

4.4 Perspectives

Despite the progress in FSI with the PFEM described above, numerous opportunities for further improvement remain, and the PFEM continues to be an active area of research with known limitations. One such limitation concerns mass conservation. While the proposed PFEM-LS approach offers improvements, it does not fully resolve mass variation during remeshing. Another challenge lies in computational efficiency. Although the PFEM is based on the standard Lagrangian finite element formulation, enabling the use of dedicated, optimized numerical solvers, the need for automatic remeshing, the frequent changes in mesh connectivity and node count reduce the method's scalability on computing clusters. These changes require frequent updates to the finite element matrix pattern and the domain decomposition, which limits performance gains from parallelization. Nevertheless, research is currently conducted on massively parallelized remeshing on GPU [228, 229, 230]. This final section outlines a non-exhaustive list of potential developments that, to the best of the author's knowledge, warrant further investigation and deserve priority in future research.

Boundary Representation

In the PFEM-LS code, boundaries are represented as sets of nodes that are part of the fluid mesh only when they are wet (i.e. in contact with the fluid). When the boundary is dry (i.e. not in contact with the fluid), these nodes behave as free nodes, disconnected from any mesh element, and their motion is governed either by Newton's second law or by prescribed Dirichlet BC on the associated boundary.

This boundary modeling strategy introduces mass variation, as elements must be dynamically created to connect the boundary to the fluid domain, even though these elements are absent in the last-known configuration of the fluid. Additionally, it can lead to irregularities and small non-physical oscillations at the free surface, particularly in 3D simulations. These artifacts arise primarily from the initial position of boundary nodes, which often do not align smoothly with the evolving free surface, resulting in a non-smooth interface.

In a recent study, Leyssens et al. [87] employed an external representation for the wet boundary, independent of the fluid mesh. In this approach, the boundary is treated as an analytical surface with which fluid nodes enter into contact, eliminating problems related to node misalignment and to the creation of connecting elements with the walls. For complex wall geometries, the analytical surface can be replaced by an external, independent surface mesh $\mathcal{M}$ that serves as the fluid's contact interface. This method provides an elegant and effective, albeit more complex, way to model FS contact. The main challenge with using an independent surface mesh is enforcing free-slip BC, as fluid nodes must be constrained to move along an arbitrary, potentially non-smooth, discrete surface. By contrast, implementing no-slip BC in PFEM3D with an external surface mesh $\mathcal{M}$ is relatively straightforward. In practice, one can simply loop over the fluid nodes at each remeshing or time step, and check if their smallest projection distance onto $\mathcal{M}$ falls within a specified tolerance. If yes, the fluid node is projected on $\mathcal{M}$, and becomes a boundary node, assigned to the corresponding boundary tag. Whenever a boundary-projected node detaches from the rest of the fluid, it is removed from the database. The projection step can be optimized by using a KD-tree, as discussed in a previous chapter. When performing FSI with a flexible solid boundary, the interface displacement computed by the solid solver in response to the applied fluid load is interpolated onto the vertices of $\mathcal{M}$ using the RBF method. These interpolated displacements then define the motion of the fluid nodes in contact with the interface mesh.

Delaunay Refinement

In parallel with the work conducted during this thesis on the PFEM-FEM partitioned coupling, Leyssens et al. [87] developed a Delaunay refinement algorithm with theoretical quality guarantees, which has proven effective in maintaining stable and smooth free surface geometries in 2D. Building on Chew's method [82], the algorithm introduces a size-field constraint to control element sizes. Although still under development, it has been extended to 3D [95] and integrated into the Gmsh [93] library. A key advantage of this implementation over other remeshing tools is that it is specifically designed and optimized for use with the PFEM.

Moreover, the development teams at UCLouvain (Gmsh) and ULiège (PFEM3D) are already working in close collaboration. In our PFEM-LS code, Gmsh is already used to generate the initial fluid mesh and to perform the tessellation of the cloud of nodes. Therefore, once the new algorithm is officially released in Gmsh for 3D cases, integrating it into PFEM3D should require minimal additional effort.

Explicit Coupling Strategy

To the best of the author's knowledge, the IQN-MVJ coupling strategy remains the most advanced implicit, fully partitioned technique available in the literature. However, Meduri et al. [184] successfully applied the so-called Gravouil-Combescure [231, 232] explicit coupling strategy for fluid-structure interactions. This method couples the solid and fluid solvers using Lagrange multipliers, which help avoid stability issues caused by the added-mass effect. The Lagrange multipliers can be physically interpreted as the interaction forces at the FS interface. The main limitation of this approach is that both the fluid and solid solvers must be modified to incorporate the Lagrange multipliers, preventing the method from being fully partitioned. Nevertheless, when both solvers are in-house codes, this modification is feasible and can potentially deliver better performance than the IQN approach.

Decreasing CPU Times

There are many instances where computation time is the critical factor that determines whether a simulation is feasible. Moreover, faster computations may indirectly enhance accuracy, as they allow for the use of more refined meshes within the same time constraints. For example, refining the mesh at the free surface of the fluid is a strategy to improve mass conservation in the PFEM [78]. In this work, a NR iterative method is used to solve the discretized formulation of the Navier-Stokes equation at each time step. The resulting linear system is solved using an LU solver from Intel MKL when the monolithic PSPG stabilization is employed. However, direct solvers are known to be computationally expensive for large 3D systems. The fractional step method significantly speeds up computations, as discussed in Chapter 3. However, it is generally less robust and accurate. A promising direction for improving the fluid solver is to explore alternatives to the LU solver, such as implementing (or interfacing with) an efficient, highly parallel GMRES [233]. If the hardware supports it, utilizing GPU-accelerated linear solvers could further enhance performance [234].

Beyond the computational efficiency of the PFEM-LS, we have demonstrated that the FSI coupling in FSPC can become computationally expensive in certain scenarios. Alternative linear solver libraries could be explored to address this issue. However, it is important to note that FSPC is written in Python. While Python is a highly versatile language for scripting, pure Python loops and operations tend to be slow, as shown in Figure 121. Additionally, the Global Interpreter Lock (GIL) limits Python to executing bytecode with a single thread at a time, which prevents true parallel execution of CPU-bound tasks. The limitations of Python’s multithreading can be overcome through multiprocessing, as provided by the MPI interface for Python, a method successfully implemented in [89]. However, its reliance on pure Python loops for parallel array assembly introduce performance bottlenecks.

In this work, we made extensive use of Numpy vectorization [235] to avoid explicit loops and improve computational efficiency. In contrast, languages such as C/C++ or Cython offer more direct control over execution, enabling explicit loop implementation while preserving high performance. Re-implementing FSPC in C/C++ or Cython could allow for more natural optimization and parallelization, reducing the reliance on forced vectorization. Specifically, Cython provides a compelling alternative by combining Python’s flexibility with performance levels close to C [236]. This approach could significantly enhance efficiency, especially for iterative computations requiring fine-grained performance tuning.

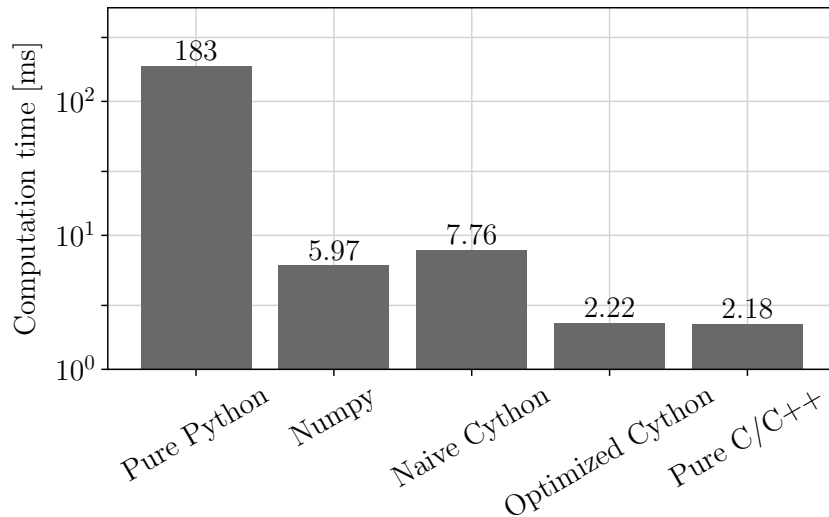


Figure 121: Comparison of the time needed by pure Python, Numpy, and Cython implementations for computing the standard deviation of a million floats. A typical C/C++ implementations with `-O2` optimization is displayed for reference. The bar plot has been reproduced from <https://notes-on-cython.readthedocs.io>.

Annex A – Elemental Matrices

$$\begin{aligned}
 \mathbf{N}_3 = & \begin{pmatrix} N_1 & 0 & 0 \\ N_2 & 0 & 0 \\ N_3 & 0 & 0 \\ N_4 & 0 & 0 \\ 0 & N_1 & 0 \\ 0 & N_2 & 0 \\ 0 & N_3 & 0 \\ 0 & N_4 & 0 \\ 0 & 0 & N_1 \\ 0 & 0 & N_2 \\ 0 & 0 & N_3 \\ 0 & 0 & N_4 \end{pmatrix} & \mathbf{B}_3 = & \begin{pmatrix} \frac{\partial N_1}{\partial x} & 0 & 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_1}{\partial z} \\ \frac{\partial N_2}{\partial x} & 0 & 0 & \frac{\partial N_2}{\partial y} & 0 & \frac{\partial N_2}{\partial z} \\ \frac{\partial N_3}{\partial x} & 0 & 0 & \frac{\partial N_3}{\partial y} & 0 & \frac{\partial N_3}{\partial z} \\ \frac{\partial N_4}{\partial x} & 0 & 0 & \frac{\partial N_4}{\partial y} & 0 & \frac{\partial N_4}{\partial z} \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_1}{\partial x} & \frac{\partial N_1}{\partial z} & 0 \\ 0 & \frac{\partial N_2}{\partial y} & 0 & \frac{\partial N_2}{\partial x} & \frac{\partial N_2}{\partial z} & 0 \\ 0 & \frac{\partial N_3}{\partial y} & 0 & \frac{\partial N_3}{\partial x} & \frac{\partial N_3}{\partial z} & 0 \\ 0 & \frac{\partial N_4}{\partial y} & 0 & \frac{\partial N_4}{\partial x} & \frac{\partial N_4}{\partial z} & 0 \\ 0 & 0 & \frac{\partial N_1}{\partial z} & 0 & \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} \\ 0 & 0 & \frac{\partial N_2}{\partial z} & 0 & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} \\ 0 & 0 & \frac{\partial N_3}{\partial z} & 0 & \frac{\partial N_3}{\partial y} & \frac{\partial N_3}{\partial x} \\ 0 & 0 & \frac{\partial N_4}{\partial z} & 0 & \frac{\partial N_4}{\partial y} & \frac{\partial N_4}{\partial x} \end{pmatrix} & (5.1) \\
 \mathbf{B}_1 = & \begin{pmatrix} \frac{\partial N_1}{\partial x} & \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial z} \\ \frac{\partial N_2}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial z} \\ \frac{\partial N_3}{\partial x} & \frac{\partial N_3}{\partial y} & \frac{\partial N_3}{\partial z} \\ \frac{\partial N_4}{\partial x} & \frac{\partial N_4}{\partial y} & \frac{\partial N_4}{\partial z} \end{pmatrix}
 \end{aligned}$$

$$\begin{aligned}
 \mathbf{N}_1 = & \begin{pmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{pmatrix} & \boldsymbol{\sigma} = & \begin{pmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{pmatrix} & \mathbf{P} = & \begin{pmatrix} 1 - n_1^2 & n_1 n_2 & n_1 n_3 \\ n_2 n_1 & 1 - n_2^2 & n_2 n_3 \\ n_3 n_1 & n_3 n_2 & 1 - n_3^2 \end{pmatrix} & (5.2)
 \end{aligned}$$

References

- [1] A. Korobenko, J. Yan, S. Gohari, S. Sarkar, and Y. Bazilevs. “FSI Simulation of Two Back-to-Back Wind Turbines in Atmospheric Boundary Layer Flow”. In: *Computers and Fluids* **158** (2017), pp. 167–175. doi: <https://doi.org/10.1016/j.compfluid.2017.05.010>.
- [2] M. Sayed, T. Lutz, E. Krämer, S. Shayegan, and R. Wüchner. “Aeroelastic Analysis of 10 MW Wind Turbine Using CFD-CSD Explicit FSI-Coupling Approach”. In: *Journal of Fluids and Structures* **87** (2019), pp. 354–377. doi: <https://doi.org/10.1016/j.jfluidstructs.2019.03.023>.
- [3] G. Liu, B. Geng, X. Zheng, and Q. Xue. “An Integrated High-Fidelity Approach for Modeling Flow-Structure Interaction in Biological Propulsion and its Strong Validation”. In: *AIAA Aerospace Sciences Meeting* (2018), p. 1543. doi: <https://doi.org/10.2514/6.2018-1543>.
- [4] A.M. Bavo, G. Rocatello, F. Iannaccone, J. Degroote, J. Vierendeels, and P. Segers. “Fluid-Structure Interaction Simulation of Prosthetic Aortic Valves: Comparison between Immersed Boundary and Arbitrary Lagrangian-Eulerian Techniques for the Mesh Representation”. In: *PLOS One* **11** (2016), pp. 1–17. doi: <https://doi.org/10.1371/journal.pone.0154517>.
- [5] J.C. Vardakis, L. Guo, and T.W. Peach et al. “Fluid-Structure Interaction for Highly Complex, Statistically Defined, Biological Media: Homogenisation and a 3D Multi-Compartmental Poroelastic Model for Brain Biomechanics”. In: *Journal of Fluids and Structures* **91** (2019), p. 102641. doi: <https://doi.org/10.1016/j.jfluidstructs.2019.04.008>.
- [6] E. Sundström, R. Jonnagiri, I. Gutmark-Little, E. Gutmark, P. Critser, M.D. Taylor, and J.T. Tretter. “Effects of Normal Variation in the Rotational Position of the Aortic Root on Hemodynamics and Tissue Biomechanics of the Thoracic Aorta”. In: *Cardiovascular Engineering and Technology* **11** (2020), pp. 47–58. doi: <https://doi.org/10.1007/s13239-019-00441-2>.
- [7] S. Piperno, C. Farhat, and B. Larrouturou. “Partitioned Procedures for the Transient Solution of Coupled Aeroelastic Problems”. In: *Computer Methods in Applied Mechanics and Engineering* **124** (1995), pp. 79–112. doi: [https://doi.org/10.1016/0045-7825\(95\)92707-9](https://doi.org/10.1016/0045-7825(95)92707-9).

- [8] E.H. Dowell and K.C. Hall. “Modeling of Fluid-Structure Interaction”. In: *Annual Review of Fluid Mechanics* **33** (2001), pp. 445–490. doi: <https://doi.org/10.1146/annurev.fluid.33.1.445>.
- [9] W. Fornari, F. Picano, and L. Brandt. “Sedimentation of Finite-Size Spheres in Quiescent and Turbulent Environments”. In: *Journal of Fluid Mechanics* **788** (2016), pp. 640–669. doi: <https://doi.org/10.1017/jfm.2015.698>.
- [10] A. Franci, I. de-Pouplana, G. Casas, M. Celigueta, J. González-Usúa, and E. Oñate. “PFEM-DEM for Particle-Laden Flows with Free Surface”. In: *Computational Particle Mechanics* **7** (2020), pp. 101–120. doi: <https://doi.org/10.1007/s40571-019-00244-1>.
- [11] K. Hughes, R. Vignjevic, J. Campbell, T. De Vuyst, N. Djordjevic, and L. Papiannidis. “From Aerospace to Offshore: Bridging the Numerical Simulation Gaps – Simulation Advancements for Fluid-Structure Interaction Problems”. In: *International Journal of Impact Engineering* **61** (2013), pp. 48–63. doi: <https://doi.org/10.1016/j.ijimpeng.2013.05.001>.
- [12] H. Gotoh and A. Khayyer. “Current Achievements and Future Perspectives for Projection-Based Particle Methods with Applications in Ocean Engineering”. In: *Journal of Ocean Engineering and Marine Energy* **2** (2016), pp. 251–278. doi: <https://doi.org/10.1007/s40722-016-0049-3>.
- [13] A. Taleei and M. Dehghan. “Direct Meshless Local Petrov-Galerkin Method for Elliptic Interface Problems with Applications in Electrostatic and Elastostatic”. In: *Computer Methods in Applied Mechanics and Engineering* **278** (2014), pp. 479–498. doi: <https://doi.org/10.1016/j.cma.2014.05.016>.
- [14] F. Almasi, M.S. Shadloo, A. Hadjadj, M. Ozbulut, N. Tofghi, and M. Yildiz. “Numerical Simulations of Multi-Phase Electro-Hydrodynamics Flows Using a Simple Incompressible Smoothed Particle Hydrodynamics Method”. In: *Computers and Mathematics with Applications* **81** (2021), pp. 772–785. doi: <https://doi.org/10.1016/j.camwa.2019.10.029>.
- [15] R.E. McCarty. “Finite Element Analysis of F-16 Aircraft Canopy Dynamic Response to Bird Impact Loading”. In: *Proceedings of the Structures, Structural Dynamics, and Materials Conference, Seattle, USA* **21** (1980), pp. 841–852. doi: <https://doi.org/10.2514/6.1980-804>.

- [16] M.L. Cerquaglia, G. Deliège, R. Boman, L. Papeleux, and J.P. Ponthot. “The Particle Finite Element Method for the Numerical Simulation of Bird Strike”. In: *International Journal of Impact Engineering* **109** (2017), pp. 1–13. doi: <https://doi.org/10.1016/j.ijimpeng.2017.05.014>.
- [17] L. Pinson and C. Brown. “Finite Element Method for Nonaxisymmetric Vibrations of Pressurized Shells of Revolution Partially Filled with Liquid”. In: *Structures, Structural Dynamics, and Materials Conference* **14** (1973), pp. 1–31. doi: <https://doi.org/10.2514/6.1973-399>.
- [18] L. Kiefling and G.C. Feng. “Fluid-Structure Finite Element Vibrational Analysis”. In: *AIAA Journal* **14** (1976), pp. 199–203. doi: <https://doi.org/10.2514/3.61357>.
- [19] C.A. Felippa and K.C. Park. “Staggered Transient Analysis Procedures for Coupled Mechanical Systems: Formulation”. In: *Computer Methods in Applied Mechanics and Engineering* **24** (1980), pp. 61–111. doi: [https://doi.org/10.1016/0045-7825\(80\)90040-7](https://doi.org/10.1016/0045-7825(80)90040-7).
- [20] R. Ohayon. “Variational Analysis of a Slender Fluid-Structure System: The Elasto-Acoustic Beam - A New Symmetric Formulation”. In: *International Journal for Numerical Methods in Engineering* **22** (1986), pp. 637–647. doi: <https://doi.org/10.1002/nme.1620220309>.
- [21] C. Farhat and M. Lesoinne. “Two Efficient Staggered Algorithms for the Serial and Parallel Solution of Three-Dimensional Nonlinear Transient Aeroelastic Problems”. In: *Computer Methods in Applied Mechanics and Engineering* **182** (2000), pp. 499–515. doi: [https://doi.org/10.1016/S0045-7825\(99\)00206-6](https://doi.org/10.1016/S0045-7825(99)00206-6).
- [22] C. Farhat, M. Lesoinne, and P. Le Tallec. “Load and Motion Transfer Algorithms for Fluid/Structure Interaction Problems with Non-Matching Discrete Interfaces: Momentum and Energy Conservation, Optimal Discretization and Application to Aeroelasticity”. In: *Computer Methods in Applied Mechanics and Engineering* **157** (1998), pp. 95–114. doi: [https://doi.org/10.1016/S0045-7825\(97\)00216-8](https://doi.org/10.1016/S0045-7825(97)00216-8).
- [23] S.R. Idelsohn, E. Oñate, and F. Del Pin. “The Particle Finite Element Method: A Powerful Tool to Solve Incompressible Flows with Free-Surfaces and Breaking Waves”. In: *International Journal for Numerical Methods in Engineering* **61** (2004), pp. 964–989. doi: <https://doi.org/10.1002/nme.1096>.

- [24] S.R. Idelsohn, E. Oñate, F. Del Pin, and N. Calvo. “Fluid-Structure Interaction Using the Particle Finite Element Method”. In: *Computer Methods in Applied Mechanics and Engineering* **195** (2006), pp. 2100–2123. doi: <https://doi.org/10.1016/j.cma.2005.02.026>.
- [25] M. Souli, A. Ouahsine, and L. Lewin. “ALE Formulation for Fluid-Structure Interaction Problems”. In: *Computer Methods in Applied Mechanics and Engineering* **190** (2000), pp. 659–675. doi: [https://doi.org/10.1016/S0045-7825\(99\)00432-6](https://doi.org/10.1016/S0045-7825(99)00432-6).
- [26] A. Alia and M. Souli. “High Explosive Simulation Using Multi-Material Formulations”. In: *Applied Thermal Engineering* **26** (2006), pp. 1032–1042. doi: <https://doi.org/10.1016/j.applthermaleng.2005.10.018>.
- [27] R. Boman, L. Papeleux, and J.P. Ponthot. *Metafor*. University of Liège, Aerospace & Mechanical Engineering, 2025. URL: <http://metafor.ltas.ulg.ac.be>.
- [28] P.P. Jeunechamps and J.P. Ponthot. “An Efficient 3D Implicit Approach for the Thermomechanical Simulation of Elastic-Viscoplastic Materials Submitted to High-Strain Rate and Damage”. In: *International Journal for Numerical Methods in Engineering* **94** (2013), pp. 920–960. doi: <https://doi.org/10.1002/nme.4489>.
- [29] M.L. Cerquaglia. “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”. PhD thesis. The University of Liège, 2019. URL: <https://hdl.handle.net/2268/233166>.
- [30] D. Thomas. “Efficient and Flexible Implementation of an Interfacing Python-Based Tool for Numerical Simulations of Fluid-Structure Interaction Problems”. PhD thesis. The University of Liège, 2020. URL: <https://hdl.handle.net/2268/252830>.
- [31] M.L. Cerquaglia, D. Thomas, R. Boman, V. Terrapon, and J.P. Ponthot. “A Fully Partitioned Lagrangian Framework for FSI Problems Characterized by Free Surfaces, Large Solid Deformations and Displacements, and Strong Added-Mass Effects”. In: *Computer Methods in Applied Mechanics and Engineering* **348** (2019), pp. 409–442. doi: <https://doi.org/10.1016/j.cma.2019.01.021>.

- [32] M.L. Cerquaglia, G. Delière, R. Boman, V. Terrapon, and J.P. Ponthot. “Free-Slip Boundary Conditions for Simulating Free-Surface Incompressible Flows Through the Particle Finite Element Method”. In: *International Journal for Numerical Methods in Engineering* **110** (2016), pp. 921–946. doi: <https://doi.org/10.1002/nme.5439>.
- [33] S. Février. “Development of a 3D Compressible Flow Solver for PFEM Fluid Simulation”. MA thesis. The University of Liège, 2020. url: <https://hdl.handle.net/2268.2/9010>.
- [34] S. Osher and J.A. Sethian. “Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations”. In: *Journal of Computational Physics* **79** (1988), pp. 12–49. doi: [https://doi.org/10.1016/0021-9991\(88\)90002-2](https://doi.org/10.1016/0021-9991(88)90002-2).
- [35] M. Sussman, P. Smereka, and S. Osher. “A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow”. In: *Journal of Computational Physics* **114** (1994), pp. 146–159. doi: <https://doi.org/10.1006/jcph.1994.1155>.
- [36] B. Schott and W.A. Wall. “A New Face-Oriented Stabilized XFEM Approach for 2D and 3D Incompressible Navier-Stokes Equations”. In: *Computer Methods in Applied Mechanics and Engineering* **276** (2014), pp. 233–265. doi: <https://doi.org/10.1016/j.cma.2014.02.014>.
- [37] C.W. Hirt and B.D. Nichols. “Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries”. In: *Journal of Computational Physics* **39** (1981), pp. 201–225. doi: [https://doi.org/10.1016/0021-9991\(81\)90145-5](https://doi.org/10.1016/0021-9991(81)90145-5).
- [38] R. Scardovelli and S. Zaleski. “Direct Numerical Simulation Of Free-surface and Interfacial Flow”. In: *Annual Review of Fluid Mechanics* **31** (1999), pp. 567–603. doi: <https://doi.org/10.1146/annurev.fluid.31.1.567>.
- [39] J. Donea, A. Huerta, J.P. Ponthot, and A. Rodríguez-Ferran. *Arbitrary Lagrangian-Eulerian Methods*. Vol. 2. Encyclopedia of Computational Mechanics. John Wiley & Sons, 2017. ISBN: 978-1-119-00379-3. doi: <https://doi.org/10.1002/9781119176817.ecm2009>.
- [40] C. Bojanowski and R.F. Kulak. “Comparison of Lagrangian, SPH and MM-ALE Approaches for Modeling Large Deformations in Soil”. In: *International LS-DYNA Users Conference, United States* **11** (2010), pp. 11–45. url: <https://doi.org/10.1002/lsdyna.1101>.

//www.dynalook.com/conferences/international-conf-2010.

- [41] C. Bojanowski. “Numerical Modeling of Large Deformations in Soil Structure Interaction Problems Using FE, EFG, SPH, and MM-ALE Formulations”. In: *Archive of Applied Mechanics* **84** (2014), pp. 743–755. doi: <https://doi.org/10.1007/s00419-014-0830-5>.
- [42] Jovan Trajkovski. “Comparison of MM-ALE and SPH Methods for Modelling Blast Wave Reflections of Flat and Shaped Surfaces”. In: *European LS-DYNA Conference, Austria* **11** (2017), pp. 1–15. URL: <https://api.semanticscholar.org/CorpusID:201786622>.
- [43] S.D. Costarelli, L. Garelli, M. Cruchaga, M.A. Storti, R. Ausensi, and S.R. Idelsohn. “An Embedded Strategy for the Analysis of Fluid Structure Interaction Problems”. In: *Computer Methods in Applied Mechanics and Engineering* **300** (2016), pp. 106–128. doi: <https://doi.org/10.1016/j.cma.2015.11.001>.
- [44] C. Kadapa, W.G. Dettmer, and D. Perić. “A Stabilised Immersed Framework on Hierarchical B-Spline Grids for Fluid-Flexible Structure Interaction with Solid-Solid Contact”. In: *Computer Methods in Applied Mechanics and Engineering* **335** (2018), pp. 472–489. doi: <https://doi.org/10.1016/j.cma.2018.02.021>.
- [45] B. Schott, C. Ager, and W.A. Wall. “Monolithic Cut Finite Element-Based Approaches for Fluid-Structure Interaction”. In: *International Journal for Numerical Methods in Engineering* **119** (2019), pp. 757–796. doi: <https://doi.org/10.1002/nme.6072>.
- [46] R. Zorrilla, R. Rossi, R. Wüchner, and E. Oñate. “An Embedded Finite Element Framework for the Resolution of Strongly Coupled Fluid-Structure Interaction Problems. Application to Volumetric and Membrane-Like Structures”. In: *Computer Methods in Applied Mechanics and Engineering* **368** (2020), p. 113179. doi: <https://doi.org/10.1016/j.cma.2020.113179>.
- [47] C.S. Peskin. “Flow Patterns Around Heart Valves: A Numerical Method”. In: *Journal of Computational Physics* **10** (1972), pp. 252–271. doi: [https://doi.org/10.1016/0021-9991\(72\)90065-4](https://doi.org/10.1016/0021-9991(72)90065-4).
- [48] C.S. Peskin. “The Immersed Boundary Method”. In: *Acta Numerica* **11** (2002), pp. 479–517. doi: <https://doi.org/10.1017/S0962492902000077>.

- [49] R. Mittal and G. Iaccarino. “Immersed Boundary Methods”. In: *Annual Review of Fluid Mechanics* **37** (2005), pp. 239–261. doi: <https://doi.org/10.1146/annurev.fluid.37.061903.175743>.
- [50] E. Burman and P. Hansbo. “Fictitious Domain Finite Element Methods Using Cut Elements: A Stabilized Nitsche Method”. In: *Applied Numerical Mathematics* **62** (2012), pp. 328–341. doi: <https://doi.org/10.1016/j.apnum.2011.01.008>.
- [51] E. Burman and M.A. Fernández. “An Unfitted Nitsche Method for Incompressible Fluid-Structure Interaction Using Overlapping Meshes”. In: *Computer Methods in Applied Mechanics and Engineering* **279** (2014), pp. 497–514. doi: <https://doi.org/10.1016/j.cma.2014.07.007>.
- [52] E. Burman and M.A. Fernández. “Stabilized Explicit Coupling for Fluid-Structure Interaction Using Nitsche Method”. In: *Proceedings of the Academy of Sciences* **345** (2007), pp. 467–472. doi: <https://doi.org/10.1016/j.crma.2007.09.010>.
- [53] R.A. Gingold and J.J. Monaghan. “Smoothed Particle Hydrodynamics: Theory and Application to Non-Spherical Stars”. In: *Monthly Notices of the Royal Astronomical Society* **37** (1977), pp. 375–389. doi: <https://doi.org/10.1093/mnras/181.3.375>.
- [54] G.R. Liu and M.B. Liu. *Smoothed Particle Hydrodynamics: A Meshfree Particle Method*. World Scientific Publishing, 2003. ISBN: 978-9-812-38456-0. doi: <https://doi.org/10.1142/5340>.
- [55] D. Violeau. *Fluid Mechanics and the SPH Method: Theory and Applications*. Oxford University Press, 2012. ISBN: 978-0-191-74122-7. doi: <https://doi.org/10.1093/acprof:oso/9780199655526.001.0001>.
- [56] J. Cercos-Pita. “Aquagpusph, a New Free 3D SPH Solver Accelerated with Opencl”. In: *Computer Physics Communications* **192** (2015), pp. 295–312. doi: <https://doi.org/10.1016/j.cpc.2015.01.026>.
- [57] G. Oger, D.L. Touzé, D. Guibert, H. de Leffe, J. Biddiscombe, J. Soumagne, and J.G. Piccinalli. “On Distributed Memory MPI-Based Parallelization of SPH Codes in Massive HPC Context”. In: *Computer Physics Communications* **200** (2016), pp. 1–14. doi: <https://doi.org/10.1016/j.cpc.2015.08.021>.

- [58] J. Campbell, R. Vignjevic, and L. Libersky. “A Contact Algorithm for Smoothed Particle Hydrodynamics”. In: *Computer Methods in Applied Mechanics and Engineering* **184** (2000), pp. 49–65. doi: [https://doi.org/10.1016/S0045-7825\(99\)00442-9](https://doi.org/10.1016/S0045-7825(99)00442-9).
- [59] S. Adami, X. Hu, and N. Adams. “A Generalized Wall Boundary Condition for Smoothed Particle Hydrodynamics”. In: *Journal of Computational Physics* **231** (2012), pp. 7057–7075. doi: <https://doi.org/10.1016/j.jcp.2012.05.005>.
- [60] J.W. Swegle, D.L. Hicks, and S.W. Attaway. “Smoothed Particle Hydrodynamics Stability Analysis”. In: *Journal of Computational Physics* **116** (1995), pp. 123–134. doi: <https://doi.org/10.1006/jcph.1995.1010>.
- [61] R. Vacondio, B.D. Rogers, P.K. Stansby, and P. Mignosa. “Variable Resolution for SPH in Three Dimensions: Towards Optimal Splitting and Coalescing for Dynamic Adaptivity”. In: *Computer Methods in Applied Mechanics and Engineering* **300** (2016), pp. 442–460. doi: <https://doi.org/10.1016/j.cma.2015.11.021>.
- [62] E. Oñate, S.R. Idelsohn, F. Del Pin, and R. Aubry. “The Particle Finite Element Method - An Overview”. In: *International Journal of Computational Methods* **1** (2004), pp. 267–307. doi: <https://doi.org/10.1142/S0219876204000204>.
- [63] M. Cremonesi, A. Franci, S. Idelsohn, and E. Oñate. “A State of the Art Review of the Particle Finite Element Method (PFEM)”. In: *Archives of Computational Methods in Engineering* **27** (2020), pp. 1709–1735. doi: <https://doi.org/10.1007/s11831-020-09468-4>.
- [64] B. Delaunay. “On the Empty Sphere”. In: *Academy of Sciences of the Union of Soviet Socialist Republics* **6** (1934), pp. 793–800. URL: <http://mi.mathnet.ru/eng/im4937>.
- [65] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. “On the Shape of a Set of Points in the Plane”. In: *IEEE Transactions on Information Theory* **29** (1983), pp. 551–559. doi: <https://doi.org/10.1109/TIT.1983.1056714>.
- [66] A. Franci, E. Oñate, and J.M. Carbonell. “Unified Lagrangian Formulation for Solid and Fluid Mechanics and FSI Problems”. In: *Computer Methods in Applied*

- Mechanics and Engineering* **298** (2016), pp. 520–547. doi: <https://doi.org/10.1016/j.cma.2015.09.023>.
- [67] S. Meduri, M. Cremonesi, and U. Perego. “An Efficient Runtime Mesh Smoothing Technique for 3D Explicit Lagrangian Free-Surface Fluid Flow Simulations”. In: *International Journal for Numerical Methods in Engineering* **117** (2018), pp. 430–452. doi: <https://doi.org/10.1002/nme.5962>.
 - [68] Z. Chen, W. Wang, B. Lévy, L. Liu, and F. Sun. “Revisiting Optimal Delaunay Triangulation for 3D Graded Mesh Generation”. In: *Methods and Algorithms for Scientific Computing* **36** (2014), pp. 930–954. doi: <https://doi.org/10.1137/120875132>.
 - [69] C. Michler, E.H. van Brummelen, and R. De Borst. “A Monolithic Approach to Fluid-Structure Interaction”. In: *Computers and Fluids* **33** (2004), pp. 839–848. doi: <https://doi.org/10.1016/j.compfluid.2003.06.006>.
 - [70] P. Becker, S.R. Idelsohn, and E. Oñate. “A Unified Monolithic Approach for Multi-Fluid Flows and Fluid-Structure Interaction Using the Particle Finite Element Method with Fixed Mesh”. In: *Computational Mechanics* **55** (2015), pp. 1091–1104. doi: <https://doi.org/10.1007/s00466-014-1107-0>.
 - [71] S. Badia, F. Nobile, and C. Vergara. “Fluid-Structure Partitioned Procedures Based on Robin Transmission Conditions”. In: *Journal of Computational Physics* **227** (2008), pp. 7027–7051. doi: <https://doi.org/10.1016/j.jcp.2008.04.006>.
 - [72] E.H. Van Brummelen. “Added Mass Effects of Compressible and Incompressible Flows in Fluid-Structure Interaction”. In: *Journal of Applied Mechanics* **76** (2009), p. 021206. doi: <https://doi.org/10.1115/1.3059565>.
 - [73] C. Michler, E.H. van Brummelen, and R. De Borst. “An interface Newton-Krylov Solver for Fluid-Structure Interaction”. In: *Numerical Methods in Fluids* **47** (2004), pp. 1189–1195. doi: <https://doi.org/10.1002/flid.850>.
 - [74] U. Küttler and W.A. Wall. “Fixed-Point Fluid-Structure Interaction Solvers with Dynamic Relaxation”. In: *Computational Mechanics* **43** (2008), pp. 61–72. doi: <https://doi.org/10.1007/s00466-008-0255-5>.
 - [75] J. Degroote. “Partitioned Simulation of Fluid-Structure Interaction”. In: *Archives of Computational Methods in Engineering* **20** (2013), pp. 185–238. doi: <https://doi.org/10.1007/s11831-013-9085-5>.

- [76] J.M. Carbonell, J.M. Rodríguez, and E. Oñate. “Modelling 3D Metal Cutting Problems with the Particle Finite Element Method”. In: *Computational Mechanics* **66** (2020), pp. 603–624. doi: <https://doi.org/10.1007/s00466-020-01867-5>.
- [77] J.M. Rodriguez, J.M. Carbonell, J.C. Cante, and J. Oliver. “Continuous Chip Formation in Metal Cutting Processes Using the Particle Finite Element Method (PFEM)”. In: *International Journal of Solids and Structures* **120** (2017), pp. 81–102. doi: <https://doi.org/10.1016/j.ijsolstr.2017.04.030>.
- [78] R. Falla, B.J. Bobach, R. Boman, J.P. Ponthot, and V. Terrapon. “Mesh Adaption for Two-Dimensional Bounded and Free-Surface Flows with the Particle Finite Element Method”. In: *Computational Particle Mechanics* **10** (2023), pp. 1049–1076. doi: <https://doi.org/10.1007/s40571-022-00541-2>.
- [79] E. Fernández, S. Février, M. Lacroix, R. Boman, L. Papeleux, and J.P. Ponthot. “A Particle Finite Element Method Based on Level-Set Functions”. In: *Journal of Computational Physics* **487** (2023), p. 112187. doi: <https://doi.org/10.1016/j.jcp.2023.112187>.
- [80] S. Meduri. “A Fully Explicit Lagrangian Finite Element Method for Highly Non-linear Fluid-Structure Interaction Problems”. PhD thesis. Polytechnic University of Milan, 2019. URL: <https://hdl.handle.net/10589/145705>.
- [81] A. Franci. *Unified Lagrangian Formulation for Fluid and Solid Mechanics, Fluid-Structure Interaction and Coupled Thermal Problems Using the PFEM*. Springer Theses. Springer Cham, 2017. ISBN: 978-3-319-45662-1. doi: <https://doi.org/10.1007/978-3-319-45662-1>.
- [82] L.P. Chew. “Guaranteed-Quality Mesh Generation for Curved Surfaces”. In: *Proceedings of the Ninth Annual Symposium on Computational Geometry* **125** (1993), pp. 274–280. doi: <https://doi.org/10.1145/160985.16115>.
- [83] J. Radon. “Mengen Konvexer Körper, die Einen Gemeinsamen Punkt Enthalten”. In: *Mathematische Annalen* **83** (1921), pp. 113–115. doi: <https://doi.org/10.1007/BF01464231>.
- [84] X.Y. Li. “Sliver-Free Three Dimensional Delaunay Mesh Generation”. PhD thesis. University of Illinois at Urbana-Champaign, 2001. URL: <https://hdl.handle.net/2142/81994>.

- [85] D. Vartziotis, J. Wipper, and B. Schwald. “The Geometric Element Transformation Method for Tetrahedral Mesh Smoothing”. In: *Computer Methods in Applied Mechanics and Engineering* **199** (2009), pp. 169–182. doi: <https://doi.org/10.1016/j.cma.2009.09.027>.
- [86] C. Marot and J.F. Remacle. *Quality Tetrahedral Mesh Generation with HXT*. 2020. arXiv: 2008.08508. URL: <https://arxiv.org/abs/2008.08508>.
- [87] T. Leyssens, M. Henry, J. Lambrechts, and J.F. Remacle. “A Delaunay Refinement Algorithm for the Particle Finite Element Method Applied to Free Surface Flows”. In: *International Journal for Numerical Methods in Engineering* **125** (2024), e7554. doi: <https://doi.org/10.1002/nme.7554>.
- [88] A. Bawin, A. Garon, and J.F. Remacle. *Isometric Simplices for High-Order Anisotropic Mesh Adaptation. Part I: Definition and Existence of Isometric Triangulations*. 2025. arXiv: 2501.06328. URL: <https://arxiv.org/abs/2501.06328>.
- [89] D. Thomas, M.L. Cerquaglia, R. Boman, T.D. Economon, J.J. Alonso, G. Dimitriadis, and V. Terrapon. “CUPyDO - An Integrated Python Environment for Coupled Fluid-Structure Simulations”. In: *Advances in Engineering Software* **128** (2019), pp. 69–85. doi: <https://doi.org/10.1016/j.advengsoft.2018.05.007>.
- [90] T.E. Tezduyar. “Stabilized Finite Element Formulations for Incompressible Flow Computations”. In: *Advances in Applied Mechanics* **28** (1991), pp. 1–44. doi: [https://doi.org/10.1016/0045-7825\(92\)90141-6](https://doi.org/10.1016/0045-7825(92)90141-6).
- [91] R. Temam. “Sur l’Approximation de la Solution des Equations de Navier-Stokes par la Méthode des Pas Fractionnaires”. In: *Archive for Rational Mechanics and Analysis* **32** (1969), pp. 135–153. doi: <https://doi.org/10.1007/BF00247678>.
- [92] H. Si. “TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator”. In: *ACM Transactions on Mathematical Software* **41** (2015), pp. 1–36. doi: <https://doi.org/10.1145/2629697>.
- [93] C. Geuzaine and J.F. Remacle. “Gmsh: A Three-dimensional Finite Element Mesh Generator with Built-In Pre- and Post-Processing Facilities”. In: *International Journal for Numerical Methods in Engineering* **79** (2009), pp. 1309–1331. doi: <https://doi.org/10.3390/app112411893>.

- [94] C. Marot, J. Pellerin, and J.F. Remacle. “One Machine, One Minute, Three Billion Tetrahedra”. In: *International Journal for Numerical Methods in Engineering* **117** (2019), pp. 967–990. doi: <https://doi.org/10.1002/nme.5987>.
- [95] T. Leyssens, J. Lambrechts, and J.F. Remacle. “Un Algorithme d’Adaptation de Maillage pour la Simulation d’Ecoulements à Surface Libre dans la Méthode des Éléments Finis-Particules”. In: *Computational Structural Mechanics Association* (2024), pp. 1–8. URL: <https://hal.science/hal-04611053>.
- [96] R. Haelterman, A.E.J. Bogaers, K. Scheufele, B. Uekermann, and M. Mehl. “Improving the Performance of the Partitioned QN-ILS Procedure for Fluid-Structure Interaction Problems: Filtering”. In: *Computers and Structures* **171** (2016), pp. 9–17. doi: <https://doi.org/10.1016/j.compstruc.2016.04.001>.
- [97] N. Delaissé, T. Demeester, D. Fauconnier, and J. Degroote. “Surrogate-Based Acceleration of Quasi-Newton Techniques for Fluid-Structure Interaction Simulations”. In: *Computers and Structures* **260** (2022), p. 106720. doi: <https://doi.org/10.1016/j.compstruc.2021.106720>.
- [98] K. Scheufele and M. Mehl. “Robust Multisecant Quasi-Newton Variants for Parallel Fluid-Structure Simulations and Other Multiphysics Applications”. In: *SIAM Journal on Scientific Computing* **39** (2017), S404–S433. doi: <https://doi.org/10.1137/16M1082020>.
- [99] B.J. Bobach. “Towards a Unified Formulation for the Simulation of Thermo-Fluid-Solid Problems with Phase Change”. PhD thesis. The University of Liège, 2023. URL: <https://hdl.handle.net/2268/304838>.
- [100] S. Février, E. Fernández, M. Lacroix, R. Boman, and J.P. Ponthot. “Simulation of Melt Pool Dynamics Including Vaporization Using the Particle Finite Element Method”. In: *Computational Mechanics* (2024), pp. 1–29. doi: <https://doi.org/10.1007/s00466-024-02571-4>.
- [101] J. Delhez. “Application of the PFEM to the Study of Blood Flows and their Interactions with Artery Walls”. MA thesis. The University of Liège, 2024. URL: <http://hdl.handle.net/2268.2/20394>.
- [102] M. Cremonesi, S. Meduri, U. Perego, and A. Frangi. “An Explicit Lagrangian Finite Element Method for Free-Surface Weakly Compressible Flows”. In: *Com-*

- putational Particle Mechanics* **4** (2017), pp. 357–369. doi: <https://doi.org/10.1007/s40571-016-0122-7>.
- [103] K. Kamran, R. Rossi, E. Oñate, and S.R. Idelsohn. “A Compressible Lagrangian Framework for the Simulation of the Underwater Implosion of Large Air Bubbles”. In: *Computer Methods in Applied Mechanics and Engineering* **255** (2013), pp. 210–225. doi: <https://doi.org/10.1016/j.cma.2012.11.018>.
 - [104] J.W.M. Bush. *Lecture 5: Stress Boundary Conditions*. Interfacial Phenomena, MIT OpenCourseWare. 2013. URL: <https://ocw.mit.edu/courses/mathematics>.
 - [105] E. Michaelides, C.T. Crowe, and J.D. Schwarzkopf. *Multiphase Flow Handbook*. Vol. 2. Mechanical and Aerospace Engineering Series. CRC Press, 2016. doi: <https://doi.org/10.1201/9781315371924>.
 - [106] R.C. Cammarata. “Surface and Interface Stress Effects in Thin Films”. In: *Progress in Surface Science* **46** (1994), pp. 1–38. doi: [https://doi.org/10.1016/0079-6816\(94\)90005-1](https://doi.org/10.1016/0079-6816(94)90005-1).
 - [107] R. Shuttleworth. “The Surface Tension of Solids”. In: *Proceedings of the Physical Society* **63** (1950), pp. 444–457. doi: <https://doi.org/10.1088/0370-1298/63/5/302>.
 - [108] J. Yvonnet, H.L. Quang, and Q.C. He. “An XFEM/Level Set Approach to Modelling Surface/Interface Effects and Computing the Size-Dependent Effective Properties of Nanocomposites”. In: *Computational Mechanics* **42** (2008), pp. 119–131. doi: <https://doi.org/10.1007/s00466-008-0241-y>.
 - [109] B.E. Hachi, A.E. Benkhechiba, M.R. Kired, D. Hachi, and M. Haboussi. “Some Investigations on 3D Homogenization of NanoComposite/Nano-Porous Materials with Surface Effect by FEM/XFEM Methods Combined with Level-Set Technique”. In: *Computer Methods in Applied Mechanics and Engineering* **371** (2020), p. 113319. doi: <https://doi.org/10.1016/j.cma.2020.113319>.
 - [110] M.E. Gurtin and A. Ian Murdoch. “A Continuum Theory of Elastic Material Surfaces”. In: *Archive for Rational Mechanics and Analysis* **57** (1975), pp. 291–323. doi: <https://doi.org/10.1007/BF00261375>.
 - [111] M.E. Gurtin, J. Weissmüller, and F. Larché. “A General Theory of Curved Deformable Interfaces in Solids at Equilibrium”. In: *International Journal for*

- Numerical Methods in Engineering* **78** (1998), pp. 1093–1109. doi: <https://doi.org/10.1080/01418619808239977>.
- [112] O.A. Ladyzhenskaya. *The Mathematical Theory of Viscous Incompressible Flows*. Vol. 2. Mathematics and its Applications. Gordon and Breach, 1969. ISBN: 978-1-614-27671-5.
 - [113] I. Babuska. “The Finite Element Method for Elliptic Equations with Discontinuous Coefficients”. In: *Computing* **5** (1970), pp. 207–213. doi: <https://doi.org/10.1007/BF02248021>.
 - [114] F. Brezzi. “On the Existence, Uniqueness and Approximation of Saddle-Point Problems Arising from Lagrangian Multipliers”. In: *RAIRO Numerical Analysis* **8** (1974), pp. 129–151. doi: <https://doi.org/10.1051/m2an/197408R201291>.
 - [115] J. Donea and A. Huerta. *Finite Element Methods for Flow Problems*. John Wiley & Sons, 2003. ISBN: 978-0-471-49666-3. doi: <https://doi.org/10.1002/0470013826>.
 - [116] R.L. Sani, P.M. Gresho, R.L. Lee, and D.F. Griffiths. “The Cause and Cure of the Spurious Pressures Generated by Certain FEM Solutions of the Incompressible Navier-Stokes Equations”. In: *International Journal for Numerical Methods in Fluids* **1** (1981), pp. 17–43. doi: <https://doi.org/10.1002/flid.1650010104>.
 - [117] T.J.R. Hughes and L.P. Franca. “A New Finite Element Formulation for Computational Fluid Dynamics: The Stokes Problem with Various Well-Posed Boundary Conditions: Symmetric Formulations that Converge for All Velocity/Pressure Spaces”. In: *Computer Methods in Applied Mechanics and Engineering* **65** (1987), pp. 85–96. doi: [https://doi.org/10.1016/0045-7825\(87\)90184-8](https://doi.org/10.1016/0045-7825(87)90184-8).
 - [118] T.E. Tezduyar, S. Mittal, S.E. Ray, and R. Shih. “Incompressible Flow Computations with Stabilized Bilinear and Linear Equal-Order-Interpolation Velocity-Pressure Elements”. In: *Computer Methods in Applied Mechanics and Engineering* **95** (1992), pp. 221–242. doi: [https://doi.org/10.1016/0045-7825\(92\)90141-6](https://doi.org/10.1016/0045-7825(92)90141-6).
 - [119] E. Fernández, S. Février, M. Lacroix, R. Boman, and J.P. Ponthot. “Generalized- α Scheme in the PFEM for Velocity-Pressure and Displacement-Pressure Formula-

- tions of the Incompressible Navier-Stokes Equations”. In: *International Journal for Numerical Methods in Engineering* **124** (2022), pp. 40–79. DOI: <https://doi.org/10.1002/nme.7101>.
- [120] R. Courant, K. Friedrichs, and H. Lewy. “Über die Partiellen Differenzengleichungen der Mathematischen Physik”. In: *Mathematische Annalen* **100** (1928), pp. 132–174. DOI: <https://doi.org/10.1007/BF01448839>.
- [121] J.C. Butcher. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, 2008. ISBN: 978-0-470-75376-7. DOI: <https://doi.org/10.1002/9780470753767>.
- [122] C. Hirsch. *Numerical Computation of Internal and External Flows: Fundamentals of Numerical Discretization*. Vol. 1. Wiley Series in Numerical Methods in Engineering. John Wiley & Sons, 1991. ISBN: 978-0-471-92385-5.
- [123] Y. Saad Y and M.H. Schultz. “GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems”. In: *SIAM Journal on Scientific and Statistical Computing* **7** (1986), pp. 856–869. DOI: <https://doi.org/10.1137/0907058>.
- [124] J.L. Guermond and L. Quartapelle. “On Stability and Convergence of Projection Methods Based on Pressure Poisson Equation”. In: *International Journal for Numerical Methods in Fluids* **26** (1998), pp. 1039–1053. DOI: [https://doi.org/10.1002/\(SICI\)1097-0363\(19980515\)26:9<1039::AID-FLD675>3.0.CO;2-U](https://doi.org/10.1002/(SICI)1097-0363(19980515)26:9<1039::AID-FLD675>3.0.CO;2-U).
- [125] A.J. Chorin. “The Numerical Solution of the Navier-Stokes Equations for an Incompressible Fluid”. In: *Bulletin of the American Mathematical Society* **73** (1967), pp. 928–931. DOI: <https://doi.org/10.1090%2FS0002-9904-1967-11853-6>.
- [126] R. Codina. “Pressure Stability in Fractional Step Finite Element Methods for Incompressible Flows”. In: *Journal of Computational Physics* **170** (2001), pp. 112–140. DOI: <https://doi.org/10.1006/jcph.2001.6725>.
- [127] R. Temam. “Remark on the Pressure Boundary Condition for the Projection Method”. In: *Theoretical and Computational Fluid Dynamics* **3** (1991), pp. 181–184. DOI: <https://doi.org/10.1007/bf00271801>.
- [128] P.M. Gresho and R.L. Sani. “On pressure Boundary Conditions for the Incompressible Navier-Stokes Equations”. In: *International Journal for Numerical Meth-*

- ods in Fluids* **7** (1987), pp. 1111–1145. doi: <https://doi.org/10.1006/jcph.2001.6725>.
- [129] B. Yang and A. Prosperetti. “A second-Order Boundary-Fitted Projection Method for Free-Surface Flow Computations”. In: *Journal of Computational Physics* **213** (2006), pp. 574–590. doi: <https://doi.org/10.1016/j.jcp.2005.08.025>.
 - [130] S. Rabier and M. Medale. “Computation of Free Surface Flows with a Projection FEM in a Moving Mesh Framework”. In: *Computer Methods in Applied Mechanics and Engineering* **192** (2003), pp. 4703–4721. doi: [https://doi.org/10.1016/S0045-7825\(03\)00456-0](https://doi.org/10.1016/S0045-7825(03)00456-0).
 - [131] J.R. Macdonald. “Some Simple Isothermal Equations of State”. In: *Reviews of Modern Physics* **38** (1966), p. 669. doi: <https://doi.org/10.1103/RevModPhys.38.669>.
 - [132] E. Fernández, M. Lacroix, S. Février, T. Zhang, L. Papeleux, B.J. Bobach, R. Boman, S. Ryelandt, A. Simar, and J.P. Ponthot. “Modelling Melt Pool Dynamics in Aluminium-to-Steel Welds Performed by Friction Melt Bonding: A Challenge Addressed with the Particle Finite Element Method”. In: *Computational Particle Mechanics* (2024), pp. 1–17. doi: <https://doi.org/10.1007/s40571-024-00852-6>.
 - [133] B.J. Bobach, R. Boman, D. Celentano, V. Terrapon, and J.P. Ponthot. “Simulation of the Marangoni Effect and Phase Change Using the Particle Finite Element Method”. In: *Applied Sciences* **11** (2021), p. 11893. doi: <https://doi.org/10.3390/app112411893>.
 - [134] M.A. Yerry and M.S. Shephard. “Automatic Three-Dimensional Mesh Generation by the Modified-Octree Technique”. In: *International Journal for Numerical Methods in Engineering* **20** (1984), pp. 1965–1990. doi: <https://doi.org/10.1002/nme.1620201103>.
 - [135] S.H. Huoa, G.R. Liub, J.Q. Zhangc, and C.M. Song. “A Smoothed Finite Element Method for Octree-Based Polyhedral Meshes with Large Number of Hanging Nodes and Irregular Elements”. In: *Computer Methods in Applied Mechanics and Engineering* **359** (2020), p. 112646. doi: <https://doi.org/10.1016/j.cma.2019.112646>.

- [136] H. Si. “TetGen, Towards a Quality Tetrahedral Mesh Generator”. In: *Weierstraß-Institut für Angewandte Analysis und Stochastik* **1762** (2013), pp. 1–32. doi: <https://doi.org/10.20347/WIAS.PREPRINT.1762>.
- [137] D.A. Field. “Qualitative Measures for Initial Meshes”. In: *International Journal for Numerical Methods in Engineering* **47** (2000), pp. 887–906. doi: [https://doi.org/10.1002/\(SICI\)1097-0207\(20000210\)47:4<887::AID-NME804>3.0.CO;2-H](https://doi.org/10.1002/(SICI)1097-0207(20000210)47:4<887::AID-NME804>3.0.CO;2-H).
- [138] A. Liu and B. Joe. “On the Shape of Tetrahedra From Bisection”. In: *Mathematics of Computation* **63** (1994), pp. 141–154. doi: <https://doi.org/10.1090/s0025-5718-1994-1240660-4>.
- [139] A. Liu and B. Joe. “Relationship Between Tetrahedron Shape Measures”. In: *BIT Numerical Mathematics* **34** (1994), pp. 268–287. doi: <https://doi.org/10.1007/BF01955874>.
- [140] O. González, M. Cruchaga, M. Lacroix, E. Fernández, J.P. Ponthot, and D. Celen-tano. “Numerical and Experimental Analysis of Vertical Sloshing in Rectangular Tanks Through the Use of PFEM”. In: *XL Argentine Congress of Computational Mechanics* (2024). URL: <https://amcaonline.org.ar/ocs/index.php/mecom2024/mecom2024/paper/view/7940>.
- [141] R. Maehara. “The Jordan Curve Theorem via the Brouwer Fixed-Point Theorem”. In: *The American Mathematical Monthly* **91** (1984), pp. 641–643. doi: <https://doi.org/10.1080/00029890.1984.11971517>.
- [142] L.E.J. Brouwer. “On the Mapping of Manifolds”. In: *Mathematische Annalen* **71** (1912), pp. 97–115. doi: <https://doi.org/10.1007/BF01456931>.
- [143] E.L. Lima. “The Jordan-Brouwer Separation Theorem for Smooth Hypersur-faces”. In: *The American Mathematical Monthly* **95** (1988), pp. 39–42. doi: <https://doi.org/10.2307/2323445>.
- [144] J.W. Alexander. “Topological Invariants of Knots and Links”. In: *Transactions of the American Mathematical Society* **30** (1928), pp. 275–306. doi: <https://doi.org/10.1090/S0002-9947-1928-1501429-1>.
- [145] K. Hormann and A. Agathos. “The Point in Polygon Problem for Arbitrary Poly-gons”. In: *Computational Geometry* **20** (2001), pp. 131–144. doi: [https://doi.org/10.1016/S0925-7721\(01\)00012-8](https://doi.org/10.1016/S0925-7721(01)00012-8).

- [146] T. Möller and B. Trumbore. “Fast, Minimum Storage Ray-Triangle Intersection”. In: *Journal of Graphics Tools* **2** (1977), pp. 21–28. doi: <https://doi.org/10.1080/10867651.1997.10487468>.
- [147] S. Hert and M. Seel. *CGAL User and Reference Manual: Convex Hulls and Delaunay Triangulations*. CGAL Editorial Board, Version 6.0. 2024. URL: <https://doc.cgal.org/6.0/Manual/packages.html>.
- [148] D. Meagher. “Geometric Modeling Using Octree-Encoding”. In: *Computer Graphics and Image Processing* **19** (1982), pp. 129–147. doi: [https://doi.org/10.1016/0146-664X\(82\)90104-6](https://doi.org/10.1016/0146-664X(82)90104-6).
- [149] J.L. Bentley. “Multidimensional Binary Search Trees Used for Associative Searching”. In: *Communications of the ACM* **18** (1975), pp. 509–517. doi: <https://doi.org/10.1145/361002.361000>.
- [150] A. Franci and M. Cremonesi. “On the Effect of Standard PFEM Remeshing on Volume Conservation in Free-Surface Fluid Flow Problems”. In: *Computational Particle Mechanics* **4** (2016), pp. 331–343. doi: <https://doi.org/10.1007/s40571-016-0124-5>.
- [151] G. Avancini, A. Franci, S. Idelsohn, and R.A.K. Sanches. “A Particle-Position-Based Finite Element Formulation for Free-Surface Flows with Topological Changes”. In: *Computer Methods in Applied Mechanics and Engineering* **429** (2024), p. 117118. doi: <https://doi.org/10.1016/j.cma.2024.117118>.
- [152] J. Degroote, K.J. Bathe, and J. Vierendeels. “Performance of a New Partitioned Procedure Versus a Monolithic Procedure in Fluid-Structure Interaction”. In: *Computer and Structures* **87** (2009), pp. 793–801. doi: <https://doi.org/10.1016/j.compstruc.2008.11.013>.
- [153] J.P. Ponthot. “Unified Stress Update Algorithms for the Numerical Simulation of Large Deformation Elasto-Plastic and Elasto-Viscoplastic Processes”. In: *International Journal of Plasticity* **18** (2002), pp. 91–126. doi: [https://doi.org/10.1016/S0749-6419\(00\)00097-8](https://doi.org/10.1016/S0749-6419(00)00097-8).
- [154] H.M. Hilber, T.J.R. Hughes, and R.L. Taylor. “Improved Numerical Dissipation for Time Integration Algorithms in Structural Dynamics”. In: *Earthquake Engineering and Structural Dynamics* **5** (1977), pp. 283–292. doi: <https://doi.org/10.1002/eqe.4290050306>.

- [155] W.L. Wood, M. Bossak, and O.C. Zienkiewicz. “An Alpha Modification of Newmark’s Method”. In: *International Journal of Numerical Methods in Engineering* **15** (1980), pp. 1562–1566. doi: <https://doi.org/10.1002/nme.1620151011>.
- [156] J. Chung and G.M. Hulbert. “A Time Integration Algorithm for Structural Dynamics With Improved Numerical Dissipation: The Generalized- α Method”. In: *Journal of Applied Mechanics* **60** (1993), pp. 371–375. doi: <https://doi.org/10.1115/1.2900803>.
- [157] P. Wriggers. *Computational Contact Mechanics*. John Wiley & Sons, 2002. ISBN: 978-0-471-49680-9.
- [158] K.J. Bathe and P.A. Bouzinov. “On the Constraint Function Method for Contact Problems”. In: *Computers and Structures* **64** (1997), pp. 1069–1085. doi: [https://doi.org/10.1016/S0045-7949\(97\)00036-9](https://doi.org/10.1016/S0045-7949(97)00036-9).
- [159] G. Wautelet and J.P. Ponthot. “The Influence of Equivalent Contact Area Computation in Extended Node to Surface Contact Elements”. In: *Key Engineering Materials* **618** (2014), pp. 1–22. doi: <https://doi.org/10.4028/www.scientific.net/KEM.618.1>.
- [160] D.M. Beazley. “SWIG: An Easy-to-Use Tool for Integrating Scripting Languages with C and C++”. In: *Proceedings of the 4th Conference on USENIX Tcl/Tk Workshop* **4** (1996), p. 15. URL: <https://dl.acm.org/doi/10.5555/1267498.1267513>.
- [161] J. Degroote, A. Souto-Iglesias, W. van Paepegem, S. Annerel, P. Bruggeman, and J. Vierendeels. “Partitioned Simulation of the Interaction Between an Elastic Structure and Free Surface Flow”. In: *Computer Methods in Applied Mechanics and Engineering* **199** (2010), pp. 2085–2098. doi: <https://doi.org/10.1016/j.cma.2010.02.019>.
- [162] A.E.J. Bogaers, S. Kok, B.D. Reddy, and T. Franz. “Quasi-Newton Methods for Implicit Black-Box FSI Coupling”. In: *Computer Methods in Applied Mechanics and Engineering* **279** (2014), pp. 113–132. doi: <https://doi.org/10.1016/j.cma.2014.06.033>.
- [163] J. Vierendeels, L. Lanoye, J. Degroote, and P. Verdonck. “Implicit Coupling of Partitioned Fluid-Structure Interaction Problems with Reduced Order Models”.

- In: *Computers and Structures* **85** (2007), pp. 11–14. doi: <https://doi.org/10.1016/j.compstruc.2006.11.006>.
- [164] P. Causin, J. Gerbeau, and F. Nobile. “Added-Mass Effect in the Design of Partitioned Algorithms for Fluid-Structure Problems”. In: *Computer Methods in Applied Mechanics and Engineering* **194** (2005), pp. 4506–4527. doi: <https://doi.org/10.1016/j.cma.2004.12.005>.
 - [165] C. Förster, W.A. Wall, and E. Ramm. “Artificial Added Mass Instabilities in Sequential Staggered Coupling of Nonlinear Structures and Incompressible Viscous Flows”. In: *Computer Methods in Applied Mechanics and Engineering* **196** (2007), pp. 1278–1293. doi: <https://doi.org/10.1016/j.cma.2006.09.002>.
 - [166] J. Degroote, R. Haelterman, S. Annerel, A. Swillens, P. Segers, and J. Vierendeels. “An Interface Quasi-Newton Algorithm for Partitioned Simulation of Fluid-Structure Interaction”. In: *Fluid-Structure Interaction : Theory, Numerics and Applications* (2009), pp. 55–64. URL: <http://hdl.handle.net/1854/LU-533424>.
 - [167] C. Wood, A.J. Gil, O. Hassan, and J. Bonet. “Partitioned Block-Gauss-Seidel Coupling for Dynamic Fluid-Structure Interaction”. In: *Mathematics of Computation* **88** (2010), pp. 1367–1382. doi: <https://doi.org/10.1016/j.compstruc.2008.08.005>.
 - [168] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Vol. **28**. Classics in Applied Mathematics. SIAM Review, 1986. doi: <https://doi.org/10.1137/1028131>.
 - [169] C.G. Broyden. “Quasi-Newton Methods and their Application to Function Minimisation”. In: *Mathematics of Computation* **21** (1967), pp. 368–381. doi: <https://doi.org/10.1090/S0025-5718-1967-0224273-2>.
 - [170] R. Haelterman, J. Degroote, D. Van Heule, and J. Vierendeels. “The Quasi-Newton Least Squares Method: A New and Fast Secant Method Analyzed for Linear Systems”. In: *SIAM Journal on Numerical Analysis* **47** (2009), pp. 2347–2368. doi: <https://doi.org/10.1137/070710469>.
 - [171] R. Penrose. “A Generalized Inverse for Matrices”. In: *Mathematical Proceedings of the Cambridge Philosophical Society* **51** (1955), pp. 406–413. doi: <https://doi.org/10.1017/S0305004100030401>.

- [172] L.R. Herrmann. “Interpretation of Finite Element Procedure as Stress Error Minimization Procedure”. In: *Journal of the Engineering Mechanics Division* **98** (1972), pp. 1330–1336. doi: <https://doi.org/10.1061/JMCEA3.0001684>.
- [173] J. Barlow. “Optimal Stress Locations in finite Element Models”. In: *Numerical Methods in Engineering* **10** (1976), pp. 243–251. doi: <https://doi.org/10.1002/nme.1620100202>.
- [174] O.C. Zienkiewicz and J.Z. Zhu. “The Superconvergent Patch Recovery and a Posteriori Error Estimates. Part 1: The Recovery Technique”. In: *Numerical Methods in Engineering* **33** (1992), pp. 1331–1364. doi: <https://doi.org/10.1002/nme.1620330702>.
- [175] O.C. Zienkiewicz and J.Z. Zhu. “The Superconvergent Patch Recovery (SPR) and Adaptive Finite Element Refinement”. In: *Computer Methods in Applied Mechanics and Engineering* **101** (1992), pp. 207–224. doi: [https://doi.org/10.1016/0045-7825\(92\)90023-D](https://doi.org/10.1016/0045-7825(92)90023-D).
- [176] O.C. Zienkiewicz and R.L. Taylor. *The Finite Element Method*. Vol. **1**. Finite Element Method Series. Butterworth-Heinemann, 2000. ISBN: 978-0-750-65049-6.
- [177] M. Zhu and M.H. Scott. “Improved Fractional Step Method for Simulating Fluid-Structure Interaction Using the PFEM”. In: *International Journal for Numerical Methods in Engineering* **99** (2014), pp. 925–944. doi: <https://doi.org/10.1002/nme.4727>.
- [178] T. Belytschko, W.K. Liu, B. Moran, and K. Elkhodary. *Nonlinear Finite Elements for Continua and Structures*. Vol. **2**. Wiley and Sons, 2014. ISBN: 978-1-118-63270-3.
- [179] C. Antoci, M. Gallati, and S. Sibilla. “Numerical Simulation of Fluid-Structure Interaction by SPH”. In: *Computers and Structures* **85** (2007), pp. 879–890. doi: <https://doi.org/10.1016/j.compstruc.2007.01.002>.
- [180] Z.F. Meng, A.M. Zhang, J.L. Yan, P.P. Wang, and A. Khayyer. “A Hydroelastic Fluid-Structure Interaction Solver Based on the Riemann-SPH Method”. In: *Computer Methods in Applied Mechanics and Engineering* **390** (2022), p. 114522. doi: <https://doi.org/10.1016/j.cma.2021.114522>.

- [181] M. Luhar and H.M. Nepf. “Flow-Induced Reconfiguration of Buoyant and Flexible Aquatic Vegetation”. In: *Applied Sciences* **56** (2011), pp. 2003–2017. doi: <https://doi.org/10.4319/lo.2011.56.6.2003>.
- [182] T. Bano, F. Hegner, M. Heinrich, and R. Schwarze. “Investigation of Fluid-Structure Interaction Induced Bending for Elastic Flaps in a Cross Flow”. In: *Applied Sciences* **10** (2020), p. 6177. doi: <https://doi.org/10.3390/app10186177>.
- [183] Ansys Software. *Ansys – Engineering Simulation Software*. Canonsburg, Pennsylvania. 2025. URL: <https://www.ansys.com>.
- [184] S. Meduri, M. Cremonesi, U. Perego, O. Bettinotti, A. Kurkchubasche, and V. Oancea. “A Partitioned Fully Explicit Lagrangian Finite Element Method for Highly Nonlinear Fluid-Structure Interaction Problems”. In: *International Journal for Numerical Methods in Engineering* **113** (2017), pp. 43–64. doi: <https://doi.org/10.1002/nme.5602>.
- [185] M.D. Buhmann. *Radial Basis Functions, Theory and Implementations*. Vol. **12**. Applied and Computational Mathematics. Cambridge University Press, 2009. ISBN: 978-0-511-54324-1. doi: <https://doi.org/10.1017/CBO9780511543241>.
- [186] H. Wendland. *Scattered Data Approximation*. Vol. **17**. Applied and Computational Mathematics. Cambridge University Press, 2004. ISBN: 978-0-511-61753-9. doi: <https://doi.org/10.1017/CBO9780511617539>.
- [187] A. Beckert and H. Wendland. “Multivariate Interpolation for Fluid-Structure-Interaction Problems Using Radial Basis Functions”. In: *Aerospace Science and Technology* **2** (2001), pp. 125–134. doi: [https://doi.org/10.1016/S1270-9638\(00\)01087-7](https://doi.org/10.1016/S1270-9638(00)01087-7).
- [188] C. Groth, S. Porziani, and M.E. Biancolini. “Radial Basis Functions Vector Fields Interpolation for Complex Fluid Structure Interaction Problems”. In: *Fluids* **6** (2021), p. 314. doi: <https://doi.org/10.3390/fluids6090314>.
- [189] L. Kedward, C.B. Allen, and T.C.S. Rendall. “Efficient and Exact Mesh Deformation Using Multiscale RBF Interpolation”. In: *Journal of Computational Physics* **345** (2017), pp. 732–751. doi: <https://doi.org/10.1016/j.jcp.2017.05.042>.

- [190] M. Lacroix, S. Février, E. Fernández, L. Papeleux, R. Boman, and J.P. Ponthot. “A Comparative Study of Interpolation Algorithms on Non-Matching Meshes for PFEM-FEM Fluid-Structure Interactions”. In: *Computers and Mathematics with Applications* **155** (2024), pp. 51–65. doi: <https://doi.org/10.1016/j.camwa.2023.11.045>.
- [191] F.B. Belgacem and Y. Maday. “The Mortar Element Method for Three Dimensional Finite Elements”. In: *Mathematical Modelling and Numerical Analysis* **31** (1997), pp. 289–302. doi: <https://doi.org/10.1051/m2an/1997310202891>.
- [192] P. Bussetta, R. Boman, and J.P. Ponthot. “Efficient 3D Data Transfer Operators Based on Numerical Integration”. In: *International Journal for Numerical Methods in Engineering* **102** (2014), pp. 892–929. doi: <https://doi.org/10.1002/nme.4821>.
- [193] A. de Boer, A.H. van Zuijlen, and H. Bijl. “Comparison of Conservative and Consistent Approaches for the Coupling of Non-Matching Meshes”. In: *Computer Methods in Applied Mechanics and Engineering* **197** (2008), pp. 4284–4297. doi: <https://doi.org/10.1016/j.cma.2008.05.001>.
- [194] J.D. Trotter, X. Cai, and S.W. Funke. “On Memory Traffic and Optimisations for Low-order Finite Element Assembly Algorithms on Multi-core CPUs”. In: *ACM Transactions on Mathematical Software* **48** (2022), pp. 1–31. doi: <https://doi.org/10.1145/3503925>.
- [195] T. Verstraete and S. Scholl. “Stability Analysis of Partitioned Methods for Predicting Conjugate Heat Transfer”. In: *International Journal of Heat and Mass Transfer* **101** (2016), pp. 852–869. doi: <https://doi.org/10.1016/j.ijheatmasstransfer.2016.05.041>.
- [196] B.M. Irons and R.C. Turck. “A version of the Aitken Accelerator for Computer Iteration”. In: *International journal for Numerical Methods in Engineering* **1** (1969), pp. 275–277. doi: <https://doi.org/10.1002/nme.1620010306>.
- [197] E. Divo, E. Steinthorsson, A.J. Kassab, and R. Bialecki. “An Iterative BEM/FVM Protocol for Steady-State Multi-Dimensional Conjugate Heat Transfer in Compressible Flows”. In: *Engineering Analysis with Boundary Elements* **26** (2002), pp. 447–454. doi: [https://doi.org/10.1016/S0955-7997\(01\)00106-0](https://doi.org/10.1016/S0955-7997(01)00106-0).

- [198] J.D. Heidmann, A.J. Kassab, E.A. Divo, F. Rodriguez, and E. Steinthorsson. “Conjugate Heat Transfer Effects on a Realistic Film-Cooled Turbine Vane”. In: *Proceedings of the ASME Turbo Expo and International Joint Power Generation Conference* **5** (2003), pp. 361–371. doi: <https://doi.org/10.1115/GT2003-38553>.
- [199] G. Gimenez, M.P. Errera, D. Baillis, Y. Smith, and F. Pardo. “A coupling Numerical Methodology for Weakly Transient Conjugate Heat Transfer Problems”. In: *International Journal of Heat and Mass Transfer* **97** (2016), pp. 975–989. doi: <https://doi.org/10.1016/j.ijheatmasstransfer.2016.02.037>.
- [200] F.P. Incropera, D.P. Dewitt, T.L. Bergman, and A.S. Lavine. *Fundamentals of Heat and Mass Transfer*. Vol. **6**. John Wiley & Sons, 2006. ISBN: 978-0-471-45728-2.
- [201] S.W. Churchill and M. Bernstein. “A Correlating Equation for Forced Convection From Gases and Liquids to a Circular Cylinder in Crossflow”. In: *Journal of Heat Transfer* **99** (1977), pp. 300–306. doi: <https://doi.org/10.1115/1.3450685>.
- [202] T. Long, P. Yang, and M. Liu. “A Novel Coupling Approach of Smoothed Finite Element Method with SPH for Thermal Fluid Structure Interaction Problems”. In: *International Journal of Mechanical Sciences* **174** (2020), p. 105558. doi: <https://doi.org/10.1016/j.ijmecsci.2020.105558>.
- [203] E. Oñate, A. Franci, and J.M. Carbonell. “A Particle Finite Element Method for Analysis of Industrial Forming Processes”. In: *Computational Mechanics* **54** (2014), pp. 85–107. doi: <https://doi.org/10.1007/s00466-014-1016-2>.
- [204] B. Jacob and G. Guennebaud. *Eigen*. C++ Template Library for Linear Algebra. 2025. URL: <https://eigen.tuxfamily.org/index.php>.
- [205] Intel Corporation. *OneMKL*. Intel OneAPI Math Kernel Library. 2025. URL: <https://www.intel.com/content/www/us/en/homepage.html>.
- [206] OpenMP Architecture Review Board. *OpenMP*. The OpenMP API Specification for Parallel Programming. 2025. URL: <https://www.openmp.org>.
- [207] Intel Corporation. *OneTBB*. Intel OneAPI Threading Building Blocks. 2025. URL: <https://www.intel.com/content/www/us/en/homepage.html>.

- [208] P. Virtanen, R. Gommers, and T.E. Oliphant et al. “Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* **17** (2020), pp. 261–272. doi: <https://doi.org/10.1038/s41592-019-0686-2>.
- [209] E. Anderson, Z. Bai, C. Bischof, S.Blackford, J. Demme, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *Lapack Users Guide*. Vol. **3**. Society for Industrial and Applied Mathematics, 1999. ISBN: 978-0-898-71447-0.
- [210] J.R. Bunch and L. Kaufman. “Some Stable Methods for Calculating Inertia and Solving Symmetric Linear Systems”. In: *Mathematics of Computation* **31** (1977), pp. 163–179. doi: <https://doi.org/10.1090/S0025-5718-1977-0428694-0>.
- [211] S. Nobari, R. Mongrain, R. Leask, and R. Cartier. “The Effect of Aortic Wall and Aortic Leaflet Stiffening on Coronary Hemodynamic: A Fluid-Structure Interaction Study”. In: *Medical & Biological Engineering & Computing* **51** (2013), pp. 923–936. doi: <https://doi.org/10.1007/s11517-013-1066-1>.
- [212] M.N. Levy and A.J. Pappano. *Cardiovascular Physiology*. Vol. **9**. Physiology Monograph Series. Mosby, 2006. ISBN: 978-8-131-20459-7.
- [213] A. Souto-Iglesias, S.R. Idelsohn, J. Martí, R. Zamora-Rodríguez, and E. Oñate. “Modeling of Free Surface Flows with Elastic Bodies Interactions”. In: *Symposium on Naval Hydrodynamics* (2008), pp. 1–17. URL: <https://oa.upm.es/3462>.
- [214] K.C. Ng, W.C. Low, H. Chen, A. Tafuni, and A. Nakayama. “A Three-Dimensional Fluid-Structure Interaction Model Based on SPH and Lattice-Spring Method for Simulating Complex Hydroelastic Problems”. In: *Ocean Engineering* **260** (2022), p. 112026. doi: <https://doi.org/10.1016/j.oceaneng.2022.112026>.
- [215] T.D. Economon, F. Palacios, S.R. Copeland, T.W. Lukaczyk, and J.J. Alonso. “SU2: An Open-Source Suite for Multiphysics Simulation and Design”. In: *AIAA Journal* **54** (2015), pp. 828–846. doi: <https://doi.org/10.2514/1.J053813>.
- [216] N. Delaissé, T. Demeester, R. Haelterman, and J. Degroote. “Quasi-Newton Methods for Partitioned Simulation of Fluid-Structure Interaction Reviewed in the Generalized Broyden Framework”. In: *Archives of Computational Methods in*

- Engineering* **30** (2023), pp. 3271–3300. DOI: <https://doi.org/10.1007/s11831-023-09907-y>.
- [217] J.W. Hancock and A.C. Mackenzie. “On the Mechanisms of Ductile Failure in High-Strength Steels Subjected to Multi-Axial Stress-States”. In: *Journal of the Mechanics and Physics of Solids* **24** (1976), pp. 147–160. DOI: [https://doi.org/10.1016/0022-5096\(76\)90024-7](https://doi.org/10.1016/0022-5096(76)90024-7).
 - [218] G.R. Johnson and W.H. Cook. “A Constitutive Model and Data for Metals Subjected to Large Strains, High Strain Rates, and High Temperatures”. In: *Proceedings 7th International Symposium on Ballistics* (1983), pp. 541–547. URL: <https://wbladb.lievers.net/10134084.html>.
 - [219] Y. Bao and T. Wierzbicki. “On Fracture Locus in the Equivalent Strain and Stress Triaxiality Space”. In: *International Journal of Mechanical Sciences* **46** (2004), pp. 81–98. DOI: <https://doi.org/10.1016/j.ijmecsci.2004.02.006>.
 - [220] X.P. Xu and A. Needleman. “Numerical Simulation of Fast Crack Growth in Brittle Solids”. In: *Journal of the Mechanics and Physics of Solids* **42** (1994), pp. 1397–1434. DOI: [https://doi.org/10.1016/0022-5096\(94\)90003-5](https://doi.org/10.1016/0022-5096(94)90003-5).
 - [221] G.T. Camacho and M. Ortiz. “Computational Modelling of Impact Damage in Brittle Materials”. In: *International Journal of Solids and Structures* **33** (1996), pp. 2899–2938. DOI: [https://doi.org/10.1016/0020-7683\(95\)00255-3](https://doi.org/10.1016/0020-7683(95)00255-3).
 - [222] T. Belytschko and T. Black. “Elastic Crack Growth in Finite Elements with Minimal Remeshing”. In: *Numerical Methods in Engineering* **45** (1999), pp. 601–620. DOI: [https://doi.org/10.1002/\(SICI\)1097-0207\(19990620\)45:5<601::AID-NME598>3.0.CO;2-S](https://doi.org/10.1002/(SICI)1097-0207(19990620)45:5<601::AID-NME598>3.0.CO;2-S).
 - [223] N. Moës, J. Dolbow, and T. Belytschko. “A Finite Element Method for Crack Growth Without Remeshing”. In: *Numerical Methods in Engineering* **46** (1999), pp. 131–150. DOI: [https://doi.org/10.1002/\(SICI\)1097-0207\(19990910\)46:1<131::AID-NME726>3.0.CO;2-J](https://doi.org/10.1002/(SICI)1097-0207(19990910)46:1<131::AID-NME726>3.0.CO;2-J).
 - [224] C. Ager, A. Seitz, and W.A. Wall. “A Consistent and Versatile Computational Approach for General Fluid-Structure-Contact Interaction Problems”. In: *International Journal for Numerical Methods in Engineering* **122** (2020), pp. 5279–5312. DOI: <https://doi.org/10.1002/nme.6556>.

- [225] U. Küttler, C. Förster, and W.A. Wall. “A Solution for the Incompressibility Dilemma in Partitioned Fluid-Structure Interaction with pure Dirichlet Fluid Domains”. In: *Computational Mechanics* **38** (2006), pp. 417–429. doi: <https://doi.org/10.1007/s00466-006-0066-5>.
- [226] J. Delhez, M. Lacroix, and J.P. Ponthot. “Application of the PFEM to the Study of Blood Flows and their Interactions with Highly Deformable Artery Walls, Including Rupture”. In: *Computational Particle Mechanics* (2025). doi: <https://doi.org/10.1007/s40571-025-01042-8>.
- [227] M. Claeskens. “Performance Simulation of a Latent Heat Storage Device Using the Particle Finite Element Method”. MA thesis. The University of Liège, 2025. URL: <http://hdl.handle.net/2268.2/23360>.
- [228] N. Tihon, J. Lambrecht, and J.F. Remacle. “Direct α -Shape on the GPU”. In: *SIAM International Meshing Roundtable, Texas, USA* (2025), pp. 1–5. URL: <https://internationalmeshingroundtable.com/assets/research-notes/imr33/2009-compressed.pdf>.
- [229] T.B. Masood, T. Ray, and V. Natarajan. “Parallel Computation of Alpha Complex for Biomolecules”. In: *Leibniz International Proceedings in Informatics* **164** (2020), pp. 1–17. doi: <https://doi.org/10.4230/LIPIcs.SoCG.2020.17>.
- [230] N. Ray, D. Sokolov, S. Lefebvre, and B. Lévy. “Meshless Voronoi on the GPU”. In: *ACM Transactions on Graphics* **37** (2018), pp. 1–12. doi: <https://doi.org/10.1145/3272127.3275092>.
- [231] A. Combescure and A. Gravouil. “A Time-Space Multi-Scale Algorithm for Transient Structural Nonlinear Problems”. In: *Mechanics and Industries* **2** (2001), pp. 43–55. doi: [https://doi.org/10.1016/S1296-2139\(00\)01077-0](https://doi.org/10.1016/S1296-2139(00)01077-0).
- [232] A. Gravouil and A. Combescure. “Multi-Time-Step and Two-Scale Domain Decomposition Method for Non-Linear Structural Dynamics”. In: *International Journal for Numerical Methods in Engineering* **58** (2003), pp. 1545–1569. doi: <https://doi.org/10.1002/nme.826>.
- [233] S. Balay, W.D. Gropp, L.C. McInnes, and B.F. Smith. “Efficient Management of Parallelism in Object-Oriented Numerical Software”. In: *Modern Software Tools*

- in Scientific Computing* (1997), pp. 163–202. doi: https://doi.org/10.1007/978-1-4612-1986-6_8.
- [234] Nvidia Corporation. *NVIDIA cuDSS: A High-Performance CUDA Library for Direct Sparse Solvers*. Santa Clara, California. 2025. URL: <https://docs.nvidia.com/cuda/cudss>.
- [235] C.R. Harris, K.J. Millman, and S.J. Van der Walt et al. “Array Programming with Numpy”. In: *Nature* **585** (2020), pp. 357–362. doi: <https://doi.org/10.1038/s41586-020-2649-2>.
- [236] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D.S. Seljebotn, and K. Smith. “Cython: The Best of Both Worlds”. In: *Institute of Electrical and Electronics Engineers* **13** (2010), pp. 31–39. doi: <https://doi.org/10.1109/MCSE.2010.118>.