An adaptive Fixed-Mesh ALE method for free surface flows

Joan Baiges[*,1,2], Ramon Codina[2,1], Arnau Pont[2,1] and Ernesto Castillo [3]

[1] Centre Internacional de Mètodes Numèrics a l'Enginyeria (CIMNE),
Edifici C1, Campus Nord UPC C/ Gran Capità S/N 08034 Barcelona, Spain
[2] Universitat Politècnica de Catalunya, Jordi Girona 1-3, Edifici C1, 08034 Barcelona, Spain
[3] Universidad de Santiago de Chile, USACH. Av. Bernardo O'Higgins 3363, Santiago de Chile, Chile
*jbaiges@cimne.upc.edu

September 23, 2016

Abstract

In this work we present a Fixed-Mesh ALE method for the numerical simulation of free surface flows capable of using an adaptive finite element mesh covering a background domain. This mesh is successively refined and unrefined at each time step in order to focus the computational effort on the spatial regions where it is required. Some of the main ingredients of the formulation are the use of an Arbitrary-Lagrangian-Eulerian formulation for computing temporal derivatives, the use of stabilization terms for stabilizing convection, stabilizing the lack of compatibility between velocity and pressure interpolation spaces, and stabilizing the ill-conditioning introduced by the cuts on the background finite element mesh, and the coupling of the algorithm with an adaptive mesh refinement procedure suitable for running on distributed memory environments. Algorithmic steps for the projection between meshes are presented together with the algebraic fractional step approach used for improving the condition number of the linear systems to be solved. The method is tested in several numerical examples. The expected convergence rates both in space and time are observed. Smooth solution fields for both velocity and pressure are obtained (as a result of the contribution of the stabilization terms). Finally, a good agreement between the numerical results and the reference experimental data is obtained.

1 Introduction

In the numerical simulation of physical phenomena many times one is faced with the need of simulating problems where the physical domain evolves in time. This is the case for instance in the numerical simulation of structural problems involving large deformations, multifluid flow problems or problems involving a free surface, amongst others. One of the main issues in such cases is to decide how to tackle with this movement. If a computational physics mesh is used, as for instance in finite elements simulations, a mesh which covers the physical domain is required (other possibilities which do not require a computational mesh exist such as meshless methods [56] or spectral methods [25], although these are perhaps not so common in day-to-day engineering practice).

This computational mesh needs to deal with the displacement of the physical domain. In many cases, the mesh can adapt to the shape of the domain by using Lagrangian formulations [17] which take into account large displacements and geometric non-linearities, or Arbitrary Lagrangian-Eulerian (ALE) formulations [38]. In most solid dynamics problems and even in fluid-structure interaction problems where the solid body is subject to large deformations but still has an anchor point [62], the simulation can be approached using these methods. In other cases, however, domain displace-

ments and deformations are so large that it is impossible for the mesh to adapt its shape to the new configuration without suffering from element distortion or element folding.

Several methodologies have been devised to address this situation. The most straightforward approach consists in, when an excessive mesh deformation is reached, computing a new mesh which covers the deformed configuration domain and project the results from the deformed mesh to the new mesh [5, 37, 54]. However, this step can require the use of external software for meshing the deformed domain, an operation which might be difficult to perform if simulations are run in a parallel environment. Moreover, in problems undergoing very large and quick deformations this step can be required to be repeated many times during the overall simulation and can become expensive in terms of computational time. Alternative approaches to remeshing strategies are for instance the use of meshless methods which do not require a computational mesh but rely instead on an approximation space of overlapping functions in the physical space [56]. Particle finite element methods [58] on the contrary, adopt a purely Lagrangian framework where each node of the finite element mesh is tracked through time. Although nodes are kept during the whole simulation, elements can be destroyed and regenerated at each time step if their deformation is too large.

Another approach (and the one in which this work is focused) is what is known as embedded or fixed mesh methods. In fixed mesh methods the boundary of the physical domain does not need to match with the boundary of the mesh. In this case boundary conditions need to be applied in a boundary which is immersed, and the variational finite element equations do only have physical meaning over the part of the mesh which is occupied by the physical domain. Families of fixed mesh methods are often classified precisely depending, firstly, on the way each method deals with the imposition of boundary conditions, and secondly, on the way they deal with time and space integration over the fixed mesh [55]. Regarding the imposition of Dirichlet boundary conditions, several approaches exist such as the use of penalty terms as in the original immersed boundary method [50, 59], the use of Lagrange multipliers [15, 42, 43, 13, 33, 20, 16, 3] which may require the use of additional unknowns accounting for the fluxes on the Dirichlet boundary, or the well-known Nitsche's method [57, 47, 44, 23], which yields symmetric, stable variational formulations through the use of a limited penalty term whose value needs to be estimated.

Regarding the way time and space integration is performed, several approaches can be adopted as well. In the original immersed boundary method [59] and in the fictitious domain method [43], the part of the computational mesh which is not covered by the physical domain is considered to be, and computed as if it was, made of the same material as the physical domain. Although this approach can be convenient and it has advantageous properties when dealing with fluid-structure interaction problems using a partitioned approach and facing the so called added-mass effect [41, 9, 8, 21, 32], it also introduces an error due to the fact that temporal derivatives in nodes close to the boundary are computed using velocity values from the non-physical, meaningless domain. This is also the case for other approaches like the ones presented in [61, 52, 51], where some kind of approximation needs to be done in order to compute the time derivative close to the interface. In order to cope with this issue, the Fixed-Mesh ALE method [34] was developed with the objective of accurately computing the temporal derivative at nodes close to the boundary by using an ALE strategy followed by a projection step back to the original background mesh. Other methodologies such as isogeometric analysis have also been applied to the problem of free surface flows recently (see [2, 4]).

In this paper we present the extension of the Fixed-Mesh ALE method to an adaptive framework for the simulation of free surface flows. The Fixed-Mesh ALE method was first used in [45] for the simulation of lost foam casting and in [35] in an application to free surface flows. The general formulation with details on the implementation and side ingredients like imposition of Dirichlet boundary conditions in immersed boundaries and a Lagrange Multiplier strategy for performing interpolations with restrictions can be found in [34], and its extension to fluid-structure interaction in [11, 12]. More recently, the Fixed-Mesh ALE method has been used in [60] in the context of rigid bodies-fluid

interaction in a parallel setting.

Here, we extend the method and we couple it with several ingredients which allow to use it in an adaptive setting. Firstly, the Fixed-Mesh ALE method is put together with the adaptive refinement library `RefficientLib [10]`, which makes the method capable of running on distributed memory parallel systems and handling meshes with hanging nodes. This is complemented with a refinement criteria that forces the mesh to improve its accuracy in the region close to the free surface interface.

Secondly, the stability properties of the finite element formulation are enhanced for the geometrical configurations in which the free surface cuts the mesh in an ill-conditioned manner. For this, we rely on ghost stabilization terms [19] which ensure that, independently on the geometry of the mesh and the free surface, the resulting system of equations is stable and has a good condition number. For the orthogonal projections involved in this type of non-consistent stabilization terms, we rely on L^2 projections which are evaluated after each non-linear iteration in order to optimize the sparsity pattern of the linear system matrix.

Finally, a splitting scheme for the Navier-Stokes equations is used which allows to separate the solution of the momentum and the pressure equations. Although the splitting scheme is a common one and has been used in previous works, its application to free surface problems has some particularities which are highlighted here. In particular, two types of splitting schemes for the incompressible Navier-Stokes equations are considered, namely a pressure Poisson equation based fractional step method and a purely algebraic fractional step method. In both cases, specific terms for the stabilization of ill-conditioned cuts in the finite element mesh are considered.

The paper is organized as follows. In Section 2 the Fixed-Mesh ALE formulation is presented. Emphasis is put in the finite element formulation, temporal discretization, computation of the temporal derivatives and the tracking of the free surface interface through a level set method. In Section 3, the algorithmic details of the projection step between meshes of the Fixed-Mesh ALE method are explained, and the particularities of its application in a parallel adaptive setting are detailed. In Section 4, two different approaches for the solution of the system of equations arising from the incompressible Navier-Stokes equations in its application to the solution of free surface flows are presented. In this section the focus is centered on the required additional stabilization terms when solving free surface problems and the required ingredients for the imposition of Dirichlet boundary conditions in the pressure Poisson equation. In Section 5, the adaptive approach for the Fixed-Mesh ALE method is described. Besides the adaptive algorithm, the refinement criteria implementation is also explained. Numerical examples which illustrate the performance of the method are presented in Section 6, and conclusions close the work in Section 7.

2 Fixed-Mesh ALE formulation

In this section we summarize the Fixed-Mesh ALE formulation and all the new numerical ingredients required in order to obtain a stable and accurate solution for the adaptively refined finite element method for free surface problems. This summary can be split into several parts, which cover the main ingredients of the method: stabilized finite element formulation, ALE mesh movement definition, level set function tracking and projection step.

2.1 Physical setting

Let us define the background domain $\Omega^0 \subset \mathbb{R}^d$ with $d = 2,3$ indicating the space dimensions. This is the domain which can be occupied by the fluid during the time interval $[0,T]$, and we denote by $\Omega(t)$ the part of Ω^0 which is effectively occupied by the fluid at each time instant $t \in [0,T]$. The moving boundary of $\Omega(t)$ is called the free surface, and it will be represented as $\Gamma_{\text{free}}(t) = \partial\Omega(t) \setminus \big(\partial\Omega^0 \cap \partial\Omega(t)\big)$. The movement of the fluid and the time evolution of the free surface causes the fluid

domain $\Omega(t)$ to change in time, and associated to this movement we define an ALE domain velocity $\boldsymbol{u}_{\text{dom}}(\boldsymbol{x},t) \in \mathbb{R}^d$, where $\boldsymbol{x} \in \Omega^0$ are the spatial coordinates. In order to do this let us define the bijective mapping $\boldsymbol{\chi}$ which for every point $\boldsymbol{X} \in \Omega(0)$ and time instant t returns a point $\boldsymbol{x} = \boldsymbol{\chi}(\boldsymbol{X},t) \in \Omega(t)$. The domain velocity can then be defined as

$$\boldsymbol{u}_{\text{dom}}(\boldsymbol{x},t) = \frac{\partial \boldsymbol{\chi}(\boldsymbol{X},t)}{\partial t}.$$

In the Fixed-Mesh ALE method, this domain velocity does not coincide in general with the velocity of the fluid in $\Omega(t)$. In fact, only the component of the velocity on Γ_{free} orthogonal to the free surface and the normal component of the velocity on the boundary of Ω^0, $\partial\Omega^0$, need to coincide with the velocity of the fluid $\boldsymbol{u}(\boldsymbol{x},t) \in \mathbb{R}^d$:

$$\boldsymbol{n} \cdot \boldsymbol{u}_{\text{dom}} = \boldsymbol{n} \cdot \boldsymbol{u} \quad \text{in } \Gamma_{\text{free}} \cup \partial\Omega^0, \tag{1}$$

where $\boldsymbol{n}$ denotes the outward pointing normal on Γ_{free} and $\partial\Omega^0$ and $\boldsymbol{u}$ is the fluid velocity. In the rest of Ω^0 the domain velocity will be defined once the spatial discretization is introduced in such a way that the bijective nature of $\boldsymbol{\chi}$ on $\Omega(t)$ is respected and the computational cost and the mesh distortion are minimized. We also define the ALE local temporal derivative of a function f as

$$\left.\frac{\partial f}{\partial t}\right|_{\boldsymbol{\chi}} = \frac{\partial f(\boldsymbol{\chi}(\boldsymbol{X},t),t)}{\partial t}.$$

Taking these definitions into account, the incompressible Navier-Stokes equations in the ALE frame of reference can be written as:

$$\rho \left.\frac{\partial \boldsymbol{u}}{\partial t}\right|_{\boldsymbol{\chi}} + \rho\left(\boldsymbol{u} - \boldsymbol{u}_{\text{dom}}\right) \cdot \nabla \boldsymbol{u} - \nabla \cdot \left(2\mu \nabla^S \boldsymbol{u}\right) + \nabla p = \rho \boldsymbol{f},$$
$$\nabla \cdot \boldsymbol{u} = 0,$$

where ρ is the fluid density, μ the viscosity, p the pressure, f the vector of body forces and $\nabla^s u$ is the symmetrical part of ∇u. Boundary and initial conditions need to be provided for this problem. We define the fluid stresses as

$$\boldsymbol{\sigma} = 2\mu \nabla^S \boldsymbol{u} - p\boldsymbol{I},$$

where $\boldsymbol{I}$ is the identity tensor. Usually boundary conditions can be subdivided into Neumann and Dirichlet boundary conditions:

$$\boldsymbol{u} = \bar{\boldsymbol{u}} \quad \text{on } \Gamma_D,$$
$$\boldsymbol{\sigma} \cdot \boldsymbol{n} = \boldsymbol{h} \quad \text{on } \Gamma_N,$$

where Γ_D is the Dirichlet boundary and Γ_N is the Neumann boundary. To simplify the exposition we will take $\bar{\boldsymbol{u}} = \boldsymbol{0}$. When considering free surface flows, the effect of any fluid outside the $\Omega(t)$ is neglected. Also, the effect of surface tension is neglected, which is a reasonable assumption given the dimensions and fluid properties of the engineering problems we are interested in. In cases where the effect of the surface tension is not negligible, it could be taken into account by adding the corresponding boundary forces terms. As a consequence, the boundary condition we enforce in the free surface is:

$$\boldsymbol{\sigma} \cdot \boldsymbol{n} = \boldsymbol{0} \quad \text{on } \Gamma_{\text{free}}.$$

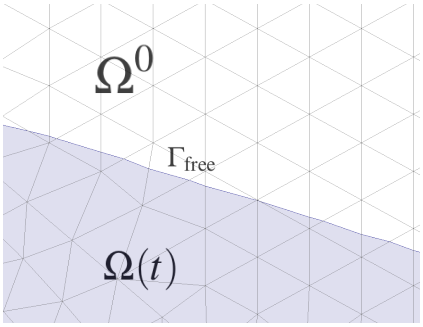


Figure 1: Background domain Ω^0 , and fixed background mesh. The domain covered by the fluid $\Omega(t)$ (in blue) evolves in time. The free surface boundary Γ_{free} cuts some elements of the mesh in an arbitrary manner. This introduces the need for integrating the finite element equations only on the region covered by $\Omega(t)$ in these elements, leading to subintegration.

2.2 Variational form

Let us consider $V = \left\{ v \in H^1(\Omega(t))^d | v = 0 \text{ on } \Gamma_D \right\}$ and $Q = L^2(\Omega(t))$. Let $L^2(0,T;V)$ be the set of functions whose $V-$norm in space is L^2 in time and $\mathscr{D}'(0,T;Q)$ the set of functions whose $Q-$norm in space is a distribution in time. Let us also define the forms:

$$\begin{aligned} B([\boldsymbol{v},q],[\boldsymbol{a};\boldsymbol{u},p]) &:= \rho \langle \boldsymbol{v}, \boldsymbol{a}\cdot\nabla\boldsymbol{u}\rangle + 2\mu \left(\nabla^S\boldsymbol{v}, \nabla^S\boldsymbol{u}\right) - (p, \nabla\cdot\boldsymbol{v}) + (q, \nabla\cdot\boldsymbol{u}) \\ L([\boldsymbol{v},q]) &:= \langle \rho\boldsymbol{v}, \boldsymbol{f}\rangle + \langle \boldsymbol{v}, \boldsymbol{h}\rangle_{\Gamma_N}, \end{aligned}$$

where $\boldsymbol{a} = \boldsymbol{u} - \boldsymbol{u}_{\text{dom}}$ is the advective velocity. Here, $(\cdot,\cdot)$ stands for the $L^2(\Omega(t))$ inner product, $\langle\cdot,\cdot\rangle$ for the integral in $\Omega(t)$ of the product of two functions, not necessarily in $L^2(\Omega(t))$, and $\langle\cdot,\cdot\rangle_{\Gamma_N}$for the integral over Γ_N of the product of two functions. The weak form of the problem consists in finding $[\boldsymbol{u},p] \in L^2(0,T;V)\times\mathscr{D}'(0,T;Q)$ such that:

$$\rho\left(\boldsymbol{v}, \left.\frac{\partial\boldsymbol{u}}{\partial t}\right|_{\boldsymbol{\chi}}\right) + B([\boldsymbol{v},q],[\boldsymbol{a};\boldsymbol{u},p]) = L([\boldsymbol{v},q]) \quad \forall \boldsymbol{v}\in V, \forall q \in Q. \tag{2}$$

Appropriate initial conditions need to be added to this variational form.

2.3 Finite element approximation

In order to approximate problem (2) numerically we aim to use a finite element method using a background mesh whose boundary does not necessarily match the boundary of $\Omega(t)$. This background mesh will cover Ω^0 and can be Cartesian or generally unstructured using general purpose mesh generators. Note that since the integrals in variational form (2) extend only over $\Omega(t)$, subintegration needs to be performed in elements cut by the fluid free surface Γ_{free}. This is illustrated in Fig. 1.

Let us now introduce the finite element partition $\mathscr{T}_h := \{K\}$ defined over domain Ω^0, h denoting the element size. From this partition the finite element spaces $V_h \subset V$ and $Q_h \subset Q$ are constructed. The straightforward introduction of these finite element spaces in equation (2) leads to unstable solutions of the incompressible Navier-Stokes equations due to the three following causes:

- Instabilities caused by the convective term, which lead to oscillating solutions if the cell Reynolds number is large.
- The requirement of compatibility conditions between the velocity and the pressure approximation spaces due to the LBB [18] inf-sup condition.

- The ill-conditioning and unstable behavior caused by the subintegration in the elements cut by the free surface Γ_{free}.

In order to overcome these instabilities, stabilization terms need to be added to variational form (2). The stabilization terms we use can be classified into two different groups, although they are in fact very similar and share the same stabilization mechanism.

The first group of stabilization terms corresponds to the Split Orthogonal Subgrid Scale stabilization (Split-OSS) presented in [30]. These terms allow to avoid the stabilization problems due to the convective term and the velocity-pressure inf-sup condition and can be derived from the Variational Multiscale framework [46]. Their form is the following:

$$S_{OSS}\left([\boldsymbol{v}_h, q_h], [\boldsymbol{a}_h; \boldsymbol{u}_h, p_h]\right) = \sum_K \tau_K \left\langle \boldsymbol{a}_h \cdot \nabla \boldsymbol{v}_h, \tilde{P}_V\left(\boldsymbol{a}_h \cdot \nabla \boldsymbol{u}_h\right)\right\rangle_{K(\Omega(t))} + \sum_K \tau_K \left\langle \nabla q_h, \tilde{P}_Q\left(\nabla p_h - \rho \boldsymbol{f}\right)\right\rangle_{K(\Omega(t))}.$$

Here $\langle \cdot, \cdot \rangle_{K(\Omega(t))}$ denotes the integral in element K of the mesh only in the part of the element covered by $\Omega(t)$. The stabilization parameter τ_K is defined at the element level as:

$$\tau_K = \left(c_1 \frac{\mu}{\rho h^2} + c_2 \frac{|\boldsymbol{a}_h|}{\rho h} \right)^{-1},$$

where c_1 and c_2 are algorithmic constants of the formulation (see for instance [28, 29] where the stabilized formulation for incompressible flows is derived). $c_1 = 4$ and $c_2 = 2$ are used in the numerical examples. $\tilde{P}_V$ and $\tilde{P}_Q$ denote a projection onto the space orthogonal to V_h and Q_h respectively, which can be computed as:

$$\begin{aligned} \tilde{P}_V\left(\boldsymbol{a}_h \cdot \nabla \boldsymbol{u}_h\right) &= \boldsymbol{a}_h \cdot \nabla \boldsymbol{u}_h - P_{V_h}(\boldsymbol{a}_h \cdot \nabla \boldsymbol{u}_h), \\ \tilde{P}_Q\left(\nabla p_h - \rho \boldsymbol{f}\right) &= \left(\nabla p_h - \rho \boldsymbol{f}\right) - P_{Q_h}\left(\nabla p_h - \rho \boldsymbol{f}\right), \end{aligned}$$

where P_{V_h} is a certain projection or interpolator operator onto V_h and P_{Q_h} is a certain projection or interpolator operator onto Q_h. Note that the added stabilization terms are not consistent, since they are not residual based. However the consistency error decreases fast enough with the element size h thanks to the approximation properties of the projectors P_{V_h} and P_{Q_h}, and the final rates of convergence of the proposed finite element method are not affected by this lack of consistency (see [30]). Also, it is important to remark that the integrals for this stabilization terms extend only over $\Omega(t)$. Many projection operators can be used which result in a stable formulation. In our formulation we opt for an L^2 projection, which involves the solution of a system of equations involving a mass matrix. In the numerical examples, where linear elements are used, we opt for using a lumped mass matrix projection, which yields an easy to invert projection matrix, and, most importantly, is not affected by ill-conditioned cuts because it fulfills the discrete maximum principle.

The second group of stabilization terms are in charge of avoiding the unstable behavior caused by the subintegration in elements cut by Γ_{free}. These instabilities and ill-conditioning appear when the free surface cuts the elements close to the interior fluid nodes and far away from the external nodes. This causes the contribution of the element integral in the external nodes to be small compared to contributions in other nodes, and instabilities appear. This is illustrated in Fig. 2. A solution for this issue was proposed in [19] for general finite element problems in what was called *ghost penalty stabilization*. The main feature of this method is that it allows to add stabilization terms which ensure the control on a given field in the cut elements without harming the convergence rate of the method, even if the added penalty terms are not consistent. This is so because the consistency error of the ghost terms depends on the difference between the considered field and its interpolation using a given projector: if the projector is chosen wisely, this consistency error decreases with the mesh size in such a way that the convergence of the method is not affected. See [39] and [19] for a more detailed elaboration on the properties of interpolators and the ghost-penalty method. The first applications

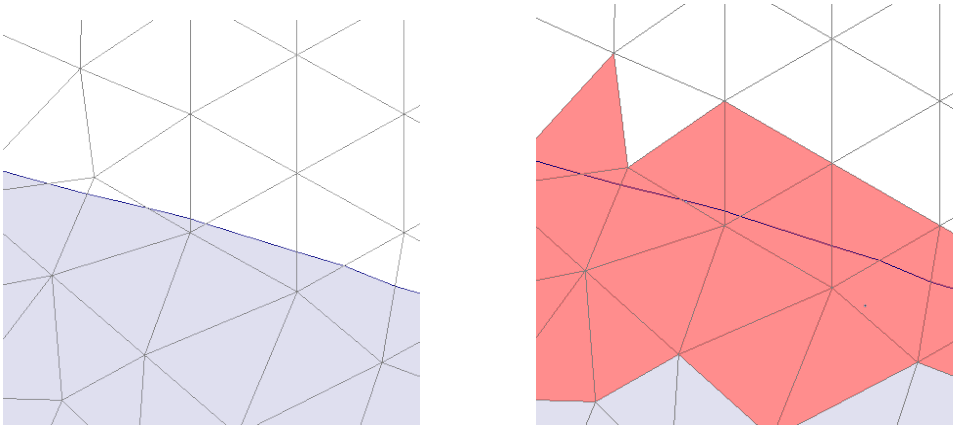


Figure 2: Left: free surface cuts causing ill-conditioning on the final linear system of equations. Right: Definition of the domain $\Omega_{\text{cut}}(t)$ (in orange) used to integrate the ghost penalty stabilization terms.

of the ghost-penalty method to flow problems can be found in [22, 24, 61]. In this work we have applied the ideas of the ghost penalty method to the incompressible Navier-Stokes equations in the context of an adaptive Fixed-Mesh ALE method. The stabilization terms we add are very similar to the ones used for stabilizing convection and the inf-sup condition, the main difference is that now the integrals do not extend over $\Omega(t)$ but over $\Omega_{\text{cut}}(t)$, which is composed by the region occupied by the elements cut by Γ_{free} and the first layer of neighbor elements inside the fluid. The definition of $\Omega_{\text{cut}}(t)$ is illustrated in Fig. 2.

The definition for these ghost penalty stabilization terms is the following:

$$\begin{aligned} S_{Ghost}\left([\boldsymbol{v}_h, q_h], [\boldsymbol{a}_h; \boldsymbol{u}_h, p_h]\right) = &\sum_K \gamma_{1K} \left\langle \nabla \boldsymbol{v}_h, \tilde{P}_{V_{\text{cut}}}\left(\nabla \boldsymbol{u}_h\right)\right\rangle_{K(\Omega_{\text{cut}}(t))} \\ &+ \sum_K \gamma_{2K} \left\langle \nabla q_h, \tilde{P}_{Q_{\text{cut}}}\left(\nabla p_h - \rho \boldsymbol{f}\right)\right\rangle_{K(\Omega_{\text{cut}}(t))}, \end{aligned}$$

where the orthogonal projections in the cut elements are defined as:

$$\begin{aligned} \tilde{P}_{V_{\text{cut}}}\left(\nabla \boldsymbol{u}_h\right) &= \nabla \boldsymbol{u}_h - P_{V_{h,\text{cut}}}(\nabla \boldsymbol{u}_h), \\ \tilde{P}_{Q_{\text{cut}}}\left(\nabla p_h - \rho \boldsymbol{f}\right) &= \nabla p_h - \rho \boldsymbol{f} - P_{Q_{h,\text{cut}}}\left(\nabla p_h - \rho \boldsymbol{f}\right). \end{aligned}$$

Note that the integrals for the calculation of the projection operators $P_{V_{h,\text{cut}}}$ and $P_{Q_{h,\text{cut}}}$ span over the $\Omega_{\text{cut}}(t)$ domain. As in the previous cases, in our formulation we opt for an L^2 projection for operators $P_{V_{h,\text{cut}}}$ and $P_{Q_{h,\text{cut}}}$. Again, a lumped mass matrix projector is used for these projectors in the numerical examples, where linear elements are used. The stabilization parameters γ_{1K} and γ_{2K} are defined elementwise as follows:

$$\begin{aligned} \gamma_{1K} &= c_3 h^2 \tau_K^{-1}, \\ \gamma_{2K} &= c_4 \tau_K. \end{aligned}$$

As it can be seen they have been defined as a function of the Split-OSS stabilization parameter. Ideally, $c_3 = 1$ and $c_4 = 1$, but in practical cases different values are required in order to obtain a stable solution. In the numerical examples presented in this work, $c_3 = 0.5$ and $c_4 = 0.5$ were chosen, since these values provided the best numerical results.

Finally, the stabilized formulation for the incompressible Navier-Stokes equations in the back-

ground non-boundary-matching mesh can be obtained as follows: Find $[\boldsymbol{u}_h, p_h]$ such that

$$\rho\left(\boldsymbol{v}_h, \left.\frac{\partial \boldsymbol{u}_h}{\partial t}\right|_{\boldsymbol{\chi}}\right) + B\left([\boldsymbol{v}_h, q_h], [\boldsymbol{a}_h; \boldsymbol{u}_h, p_h]\right)$$
$$+S_{OSS}\left([\boldsymbol{v}_h, q_h], [\boldsymbol{a}_h; \boldsymbol{u}_h, p_h]\right) + S_{Ghost}\left([\boldsymbol{v}_h, q_h], [\boldsymbol{a}_h; \boldsymbol{u}_h, p_h]\right) = L\left([\boldsymbol{v}_h, q_h]\right) \quad \forall \boldsymbol{v}_h \in V_h, \forall q_h \in Q_h. \quad (3)$$

2.4 Temporal discretization

For the temporal discretization we rely on simple differencing schemes. Consider a uniform partition of $[0,T]$ into N time intervals of lengh δt. The velocity field at time step $n+1$ is denoted as $\boldsymbol{u}^{n+1}$, whereas its discrete time derivative is denoted as $\delta_t \boldsymbol{u}^{n+1}$. We will consider the use of the second order backward differencing scheme, for which this derivative can be computed as:

$$\delta_t \boldsymbol{u}^{n+1} = \frac{\frac{3}{2}\boldsymbol{u}^{n+1} - 2\boldsymbol{u}^n + \frac{1}{2}\boldsymbol{u}^{n-1}}{\delta t}.$$

Let us now suppose that we are given a computational domain at time t^n with spatial coordinates $\boldsymbol{\chi}^n$, and $\boldsymbol{u}^n$ and p^n are known in this domain. The velocity at t^{n+1} can the obtained as the solution to:

$$\rho\left(\boldsymbol{v}_h, \delta_t \boldsymbol{u}_h^{n+1}\big|_{\boldsymbol{\chi}}\right) + B\left([\boldsymbol{v}_h, q_h], \left[\boldsymbol{a}_h^{n+1}; \boldsymbol{u}_h^{n+1}, p_h^{n+1}\right]\right)$$
$$+S_{OSS}\left([\boldsymbol{v}_h, q_h], \left[\boldsymbol{a}_h^{n+1}; \boldsymbol{u}_h^{n+1}, p_h^{n+1}\right]\right) + S_{Ghost}\left([\boldsymbol{v}_h, q_h], \left[\boldsymbol{a}_h^{n+1}; \boldsymbol{u}_h^{n+1}, p_h^{n+1}\right]\right) = L\left([\boldsymbol{v}_h, q_h]\right) \quad \forall \boldsymbol{v}_h \in V_h, \forall q_h \in Q_h. \quad (4)$$

2.5 Finite element solution in $\Omega^0 \backslash \Omega(t)$

Equation (4) provides the final variational form for the solution of $\boldsymbol{u}_h$ and p_h in $\Omega(t)$. However, something needs to be done for the values of the unknowns in $\Omega^0 \backslash \Omega(t)$, since in principle no equation has been defined so far for the solution of the problem unknowns in this region. Although in principle the solution is not needed in this region, it is convenient to have some kind of smooth extension which ensures that no numerical oscillations or sharp gradients are artificially introduced in the algorithm. For these unknowns we use a harmonic extension from the $\Omega(t)$ region to the $\Omega^0 \backslash \Omega(t)$ region. The velocity harmonic extension $\boldsymbol{u}^*$ can be found by solving, at each time step:

$$\begin{aligned} -\Delta \boldsymbol{u}^* &= 0 \quad \text{in } \Omega^*(t), \\ \boldsymbol{u}^* &= \boldsymbol{u} \quad \text{in } \Gamma^*(t). \end{aligned}$$

Here $\Omega^*(t)$ represents the domain limited by the last layer of nodes belonging to elements covering (at least partially) $\Omega(t)$, $\Gamma^*(t)$, see Fig. 3. Note that the Dirichlet boundary condition for this problem can be enforced exactly since $\Gamma^*(t)$ is formed by the external faces of elements.

Similarly, the harmonic extension for p^* can be found by solving, at each time step:

$$\begin{aligned} -\Delta p^* &= 0 \quad \text{in } \Omega^*(t), \\ p^* &= p \quad \text{in } \Gamma^*(t). \end{aligned}$$

We denote by $\boldsymbol{u}_h^*$ and p_h^* the standard finite element approximation to $\boldsymbol{u}^*$ and p^*, respectively. Note that no temporal derivative is solved for these harmonic extensions.

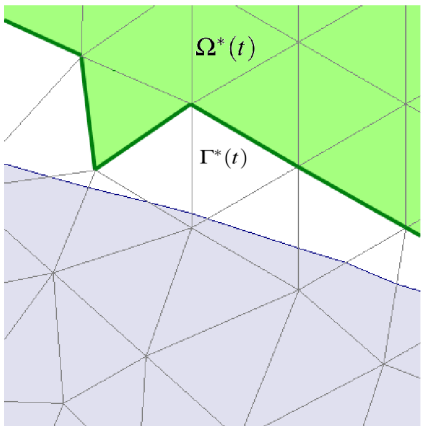


Figure 3: Definition of $\Omega^*(t)$ and $\Gamma^*(t)$ (in green) for the harmonic extension of $\boldsymbol{u}_h$ and p_h into $\Omega^0 \backslash \Omega(t)$.

2.6 ALE mesh movement

For the Arbitrary-Lagrangian-Eulerian mesh movement various possibilities for defining the mesh velocity exist. The essential requisite for the mesh movement is that the area of the mesh which covers $\Omega(t^n)$ at t^n is deformed in such a way that it covers $\Omega\left(t^{n+1}\right)$ at t^{n+1}. This translates into a boundary condition in the components of the mesh velocity normal to the external boundaries, as seen in equation (1).

For the movement in the rest of the domain, a smooth extension of the mesh displacement in the boundaries needs to be obtained. Several methods exist for this which minimize the element distortion, such as solving elastic problems with elementally varying stiffness coefficients, or using Laplace problems. In our case we opt for the last choice, which results in a good enough geometry of the deformed mesh. The problem we need to solve in the interior of Ω^0 is then:

$$\begin{aligned} \Delta \boldsymbol{u}_{\text{dom}} &= 0 \quad \text{in } \Omega^0, \\ \boldsymbol{n} \cdot \boldsymbol{u}_{\text{dom}} &= \boldsymbol{n} \cdot \boldsymbol{u} \quad \text{on } \Gamma_{\text{free}} \cup \partial\Omega^0. \end{aligned}$$

This problem is solved using a finite element approximation. Whereas the variational form of the problem is in principle standard, the application of the Dirichlet boundary conditions needs to take into account that Γ_{free} is immersed in Ω^0 and cuts the elements of the mesh in an arbitrary manner. In the $\partial\Omega^0$ boundary, on the contrary, this Dirichlet boundary condition can be easily applied, because the nodes of the mesh are on top of the domain boundary. In order to enforce the mesh velocity on Γ_{free} we rely on the symmetric method for enforcing boundary conditions in embedded grids presented in [13] and further extended and analyzed in [33, 49], which can be understood as a *Linked Lagrange Multiplier* method. This method can be related to Nitsche's method, its main advantage with respect to this classical method being that it does not require to estimate the value of the penalty parameter, and that it ensures that the resulting system is stable with respect to the imposition of the boundary conditions.

Let us consider that the same interpolation space is used for the approximation of the mesh velocities $\boldsymbol{u}_{\text{dom}}$ and the fluid velocities $\boldsymbol{u}$. If Poisson's problem is used to compute the smooth mesh velocity field in the domain interior, the final variational form of the problem including the terms for enforcing the boundary conditions in Γ_{free} is: for each t^n, find $\boldsymbol{u}_{h,\text{dom}} \in V_h$ such that:

$$(\nabla \boldsymbol{v}_h, \nabla \boldsymbol{u}_{h,\text{dom}})_{\Omega^0} + \langle \boldsymbol{v}_h, \boldsymbol{n} \cdot P_{\Sigma_h}(\boldsymbol{u}_{h,\text{dom}}) \rangle_{\Gamma_{\text{free}}} = \langle \boldsymbol{v}_h, \boldsymbol{n} \cdot P_{\Sigma_h}(\boldsymbol{u}_h) \rangle_{\Gamma_{\text{free}}} \quad \forall \boldsymbol{v}_h \in V_h, \tag{5}$$

where $P_{\Sigma}(\boldsymbol{u}_{h,\text{dom}})$ is a tensor defined only in the elements cut by Γ_{free}. It is obtained as the vol-

ume/boundary projection of $\boldsymbol{u}_{h,\mathrm{dom}}$ onto an elementwise discontinuous stress space Σ_h:

$$\frac{1}{2}\left(\tau_h, P_\Sigma\left(\boldsymbol{u}_{h,\mathrm{dom}}\right)\right)_{\Omega_{\Gamma_{\mathrm{free}}}} = \left\langle \boldsymbol{n}\cdot\tau_h, \boldsymbol{u}_{h,\mathrm{dom}}\right\rangle_{\Gamma_{\mathrm{free}}} \quad \forall \tau_h \in \Sigma_h, \tag{6}$$

where $\Omega_{\Gamma_{\mathrm{free}}}$ is the domain occupied by the elements cut by Γ_{free} (including both the part of the elements inside $\Omega(t)$ and the part outside of it). The stresses space Σ_h is usually taken to be an order of interpolation lower than the space for the velocities V_h, although equal order interpolation is also possible. Equations (5) and (6) are coupled and need to be solved simultaneously. However, taking into account that the τ_h functions are elementwise discontinuous and that they extend only over a single element, the expression of the projection can be computed at the element level and condensed prior to the matrix assembly. The main feature of the method is that it guarantees stability without the need of a penalty parameter. See [13] for a detailed description of the method and its properties.

2.7 Fixed-Mesh ALE

Equation (4) provides a stabilized finite element formulation for the incompressible Navier-Stokes equations in an ALE framework using a non-boundary-matching mesh. However, additional ingredients need to be introduced in order to deal with simulations where the computational domain undergoes very large deformations. In these cases, $\Omega(t)$ changes its shape greatly when time evolves, and the mesh becomes too distorted or can have folded elements. The solution to this issue in classical ALE methods is to build a new mesh with good element geometrical properties in the configuration at time t and then proceed to a projection step from the distorted mesh to the new mesh. However, building the new mesh can be computationally expensive and require to stop the simulation in order to call a mesh generator. The main advantage of the Fixed-Mesh ALE method is that since the mesh does not need to match the boundary of the physical domain, the original undistorted mesh covering Ω^0 can always be used as the new mesh when element distortion becomes too large. The algorithmic steps of the Fixed-Mesh ALE method are then the following:

- The simulation starts by using the original background mesh $\mathscr{M}(0)$ covering the background domain Ω^0, and integrating the corresponding equations in $\Omega(t)$, and the harmonic extension equations in $\Omega^*(t)$.
- When advancing a time step, the domain $\Omega(t)$ changes. The original mesh $\mathscr{M}(0)$ needs to be deformed following an ALE strategy: the mesh movement in Γ_{free} follows the normal velocity of the particles, whereas in some points on $\partial\Omega^0$ there is no mesh movement. The mesh at time instant t, $\mathscr{M}(t)$ is obtained by deforming $\mathscr{M}(0)$ following the ALE mesh movement criteria.
- At each step or after several time steps (both choices are possible), a remeshing step is performed, which consists in projecting the results from $\mathscr{M}(t)$ to $\mathscr{M}(0)$. After this, we redefine $\mathscr{M}(t) \leftarrow \mathscr{M}(0)$ and continue with the simulation.

In previous works on the Fixed-Mesh ALE, the number of steps between each projection from $\mathscr{M}(t)$ to $\mathscr{M}(0)$ was limited to one, meaning that at each time step the results were projected to the background mesh after what was called a *virtual mesh deformation*. This allowed the ALE mesh movement to be limited to only a few layers of elements surrounding Γ_{free}, because mesh distortion did not accumulate over time. In this work, on the contrary, we open the possibility to allow our formulation to deform the mesh during several time steps before projecting back to the original background mesh.

2.8 Level set function

For the tracking of the boundary defining the free surface, Γ_{free}, we rely on the classical level set method. The main feature of the level set method is that it allows for a sharp tracking of the free

surface Γ_{free} which results in an accurate definition of the fluid domain at time t, $\Omega(t)$. The level set function ψ is defined in such a way that, at each time instant t:

$$\begin{aligned}\psi(t) &= 0 \quad \text{in } \Gamma_{\text{free}},\\ \psi(t) &> 0 \quad \text{in } \Omega(t),\\ \psi(t) &< 0 \quad \text{in } \Omega^0 \backslash \Omega(t).\end{aligned}$$

It is important to remark, that the solution of the level set problem is done using the adapted mesh.

2.8.1 Initialization

Many strategies exist for the initialization of the level set function. Most of them rely in the computation of a positive (in the fluid part) and negative (in the part of the domain outside $\Omega(t)$) distance function from Γ_{free}. However, these approaches either rely in a search strategy which can be expensive if not implemented properly environment, or in enforcing the restriction $|\nabla\psi| = 1$, which is a non-linear restriction which needs to be solved iteratively. Our approach for the initialization of ψ does not rely on a distance function, but instead we solve the following diffusion-reaction problem (see [12, 26] for more details on this strategy):

$$-\Delta\psi + s\psi = g \quad \text{in } \Omega^0, \tag{7}$$

$$\psi = 0 \quad \text{in } \Gamma_{\text{free}}, \tag{8}$$

$$\boldsymbol{n}\cdot\nabla\psi = 0 \quad \text{in } \partial\Omega^0, \tag{9}$$

where $\boldsymbol{n}$ is the external normal in the boundary of Ω^0, and the source term g is defined as:

$$\begin{aligned}g &= 1 \qquad \text{in } \Omega(t),\\ g &= -1 \quad \text{in } \Omega^0 \backslash \Omega(t),\end{aligned}$$

and s is a reactive term which we typically take as $s = 1$, although other values are possible. This ensures a smooth ψ function and an accurate tracking of the free surface. Note that condition (8) enforces that the gradient of ψ in the the direction of the external boundary normal is null, except in those elements cuts by Γ_{free}, where the essential boundary condition prevails. This approach yields a smooth initial field, which is the main objective we pursue. Again, this equation is solved using a finite element approach. Since the Dirichlet boundary condition in (8) is embedded in the mesh, the *linked-Lagrange multiplier* method [13] is used again to enforce the boundary condition. Let us define the finite element space for the level set function $\Psi_h \subset H^1(\Omega)$. The variational problem to be solved to initialize the level set function is: find $\psi_h \in \Psi_h$ such that:

$$(\nabla\phi_h, \nabla\psi_h)_{\Omega^0} + \langle \phi_h, \boldsymbol{n}\cdot P_{\Sigma_h}(\psi_h)\rangle_{\Gamma_{\text{free}}} = \langle \phi_h, g\rangle_{\Omega^0} \quad \forall\phi_h \in \Psi_h,$$

where now Σ_h corresponds to the elementwise discontinuous space for the level set stresses and P_{Σ_h} is the projection operation defined in equation (6). This results in a smooth initial level set field which defines the $\Omega(t)$ domain. See [13] for more details on the *linked-Lagrange multiplier* method.

After several time steps of the simulation the level set function can loose smoothness and become difficult to advect. In this case, what is called a reinitialization step is required. For this, we rely in the same algorithm we use to initialize the level set function.

2.8.2 Transport

The level set function needs to evolve in time so that $\psi(t) = 0$ in $\Gamma_{\text{free}}(t)$. In Arbitrary-Lagrangian Eulerian methods, this is solved by advecting the level set function using the fluid velocity in $\Omega(t)$ as the advection velocity:

$$\partial_t\psi + \boldsymbol{a}_h\cdot\nabla\psi = 0, \tag{10}$$

with the advection velocity defined as $\boldsymbol{a}_h = \boldsymbol{u}_h - \boldsymbol{u}_{h,\text{dom}}$. If the method is purely Eulerian, $\boldsymbol{a}_h = \boldsymbol{u}_h$, whereas if the method is Lagrangian on Γ_{free} , then $\boldsymbol{a}_h = \boldsymbol{0}$ on Γ_{free} and equation (10) does not need to be solved (the relative position of Γ_{free} with respect to the nodes of the mesh does not change). In the case of the Fixed-Mesh ALE method we need to distinguish between two cases:

The first case corresponds to the situation in which several time steps of the ALE simulation are computed before the results are projected from the deformed to the background mesh. In this case, equation (10) is *not* solved, the nodal values for the level set function do not change during the simulation until the projection onto the background mesh is performed.

The second case corresponds to the situation in which the results are projected back onto the fixed background mesh *at each time step*. In this case we can opt by the following two options:

- Not solving equation (10), and projecting the level set function at the end of each time step after the mesh has been deformed.
- Solving equation (10) with $\boldsymbol{a}_h = \boldsymbol{u}_h$ and avoiding the projection step only for the computation of the level set function.

In this case we opt for the second option, because we have observed that it results in smoother level set fields in the long term.

For the finite element solution of equation (10), we rely on a strategy similar to the one applied to the Navier-Stokes equations, with the particularity that only convection needs to be stabilized.

2.8.3 Mass correction

The advection of the level set function does not guarantee that the amount of mass in the simulation volume after a time step has been solved is conserved. In practical cases, this usually results in a small amount of mass being lost at each time step. After hundreds or thousands of time steps this mass loss cumulates and can represent a significant fraction of the total volume. In order to avoid this we introduce a mass loss correction scheme which ensures that total mass is conserved at each time step. The mechanism consists in computing the amount of mass lost at the end of each time step and then moving the level set function in a uniform manner in the direction orthogonal to the free surface in order to correct this mass loss. The mechanism works in a global sense, and thus cannot ensure mass conservation at the local level. If S_{free} is the area (length in two dimensions) of the free surface, V_{lost} the lost volume and $\psi_0(t)$ the level set function solution field prior to the mass correction, then:

$$\psi(t) = \psi_0(t) + \frac{V_{\text{lost}}}{S_{\text{free}}} \cdot P_{\Omega_{\Gamma_{\text{free}}}}\left(\boldsymbol{n} \cdot \nabla \psi_0(t)\right),$$

with the following definition for $P_{\Omega_{\Gamma_{\text{free}}}}\left(\boldsymbol{n} \cdot \nabla \psi_0(t)\right) \in \Psi_h(\Omega_{\Gamma_{\text{free}}})$:

$$\left(\phi_h, P_{\Omega_{\Gamma_{\text{free}}}}\left(\boldsymbol{n} \cdot \nabla \psi_0(t)\right)\right)_{\Omega_{\Gamma_{\text{free}}}} = \langle \phi_h, \boldsymbol{n} \cdot \nabla \psi_0(t) \rangle_{\Omega_{\Gamma_{\text{free}}}} \quad \forall \phi_h \in \Psi_h(\Omega_{\Gamma_{\text{free}}}),$$

where $\Omega_{\Gamma_{\text{free}}}(t)$ is defined as the region of the domain occupied by elements cut by $\Gamma_{\text{free}}(t)$.

3 Algorithmic strategy for the projection step

Let us start this section by remarking that the method proposed in this work is implemented in a distributed memory parallel platform. Algorithmic parallelism is achieved by computing a partition of the finite element mesh by using graph partitioning algorithms, specifically ParMetis [48] is used for the partitioning of the mesh graph. The mesh partitioning algorithm in our approach is nodally based, which means that each node belongs to a single subdomain or processor, whereas elements can belong to multiple subdomains if they own nodes owned by different processors.

As explained in the previous section, the Fixed-Mesh ALE method involves a stage at each time step where the unknowns are projected onto the background fixed mesh. Independently of which type of projection is used, when dealing with this in a parallel framework the element in which a node or integration point lays after the deformation might belong to a different processor and parallel subdomain, which introduces the additional difficulty of a parallel spatial search across multiple threads. Several possible strategies exist for treating this issue.

The first strategy relies in taking a time step small enough related to the mesh size so that the free surface crosses at most one element in a single time step [60]. This allows to ensure that when looking for the position of a node or integration point in the deformed mesh, it will be found in the first-layer of elements adjacent to the node or integration point, which will be locally stored in the processor owning the node. However, this introduces a limitation in the time step size in the form of a *Courant–Friedrichs–Lewy condition (CFL)* [36], which may not be convenient when using implicit methods for solving the incompressible Navier-Stokes equations.

The second possible approach does not introduce a constraint in the time step size, but its efficiency still relies in the fact that the time step is not too large when compared to the *CFL* condition. It consists in starting the search in the elements adjacent to the node and progressively advancing the search to the neighbor elements in successive layers. When the search reaches the boundary of a processor subdomain before the sought element is found, a communication step is required in order to proceed with the search in the neighbor subdomain. This strategy can be used independently of the time step size and the number of subdomains. However, it requires a neighbor processor communication step after searching a layer of elements, which can limit the performance of the algorithm in high-performance computing environments. Moreover, when implemented in an adaptive framework where meshes can have hanging nodes, it requires to advance through neighbor elements in different refinement levels, a step which may not be straightforward to code.

The third approach consists in implementing an octree-based (quadtree in two dimensions) search algorithm whose efficiency does not depend on the time step size. This is precisely the approach we adopt. It is important in this case to implement the search algorithm so that it can be used in a parallel framework. The strategy we follow is to build first an octree search data structure for the processor corresponding to each subdomain. This is followed by the construction of a subdomain octree which allows to decide which, amongst the neighbor processors, contains the sought element when it is not found in the current processor. Since in the adaptive Fixed Mesh-ALE method the number of elements and the position of the nodes evolves in time, the octree needs to be rebuilt at each time step. The efficient parallel implementation we have pursued goes as follows.

Each processor (associated to a physical subdomain and a submesh), starts by computing its *coordinate range*, which is defined as the set of maximum and minimum nodal coordinates in each coordinate axis. This allows to subdivide the rectangular prism which encloses the local subdomain into eight subprisms, which we call the leafs of the tree. Elements are then assigned to each leaf, taking into account that an element can belong to multiple leafs if the coordinates of their nodes lay in different sides of the virtual boundaries that delimit the newly created leafs. Dynamical linked lists are used to store this information. As usual in octree based search algorithms, if the number of elements in a leaf is too large, the leaf is further subdivided and the elements of the leaf are reclassified in the new subleafs. The criteria used in order to decide when to stop the subdivision process needs to ensure that the classification and search process is efficient. If points were being classified instead of elements, a reasonable criteria would be to stop the subdivision process when at most one point per leaf is found. However, since elements occupy a volume, this can lead to an infinite recursion. In order to avoid this, three criteria for stopping the recursion process are applied:

- If the number of elements in the leaf is smaller than a number of elements threshold, stop the recursive process.

- If the leaf level is larger than a level threshold, stop the recursive process.
- If the the ratio between the size of the volume enclosed in the leaf and the size of the elements is smaller than a leaf volume threshold, stop the recursive process.

After building the octree structure following these criteria, the search at the local level can be carried out. If the time step is not too large, most of the searches for the projection will be successful by searching only in the elements of the current processor. However, in some cases, specially if the time step is large, the sought element can belong to a different processor. In this case it is convenient to have a *processor octree* which allows to decide in which processor should the element be looked for. The strategy is very similar to what has been explained for the element octree: first, every processor sends its *coordinates range* to all the other processors. This is an all-to-all communication process, but involves only the communication of two real variables per space dimension and needs to be done only once. With this information each processor is capable of building the global processor octree following the same steps as in the element octree. This processor octree is replicated in all the processors. Once each processor has built its own local element octree, and all the processors have built the global processor octree, the steps for the spatial search of the coordinates of a point or integration point in the projection step are:

- Try finding the element inside which the point is placed in the local subdomain using the local element octree. If the element is found, note down the element and the interpolation coefficients so that the search needs not to be performed again for multiple field interpolation.
- If the element is not found in the local subdomain, choose which are the candidate subdomains where the element needs to be looked for using the global processor octree. The coordinates of the point need to be sent to each of the candidate processors.
- Receive the coordinates of the points from other processors which need to be looked for in the current processor. Look for them using the local element octree and return a flag indicating whether the element was found or not to the processor owner of the node. If the element is found, note down the element and the interpolation coefficients so that the search needs not to be performed again for multiple field interpolation.

Once this preparatory stage is done, the interpolation of a field from the deformed to the background mesh can be efficiently done by:

- Interpolating the field for all the points which are local and whose corresponding element is also local.
- Interpolating the field for the points which were asked by other processors and sending them to these processors.
- Receiving the interpolated values from other processors for the local nodes whose corresponding element is not local.

Following these criteria an efficient global octree search algorithm for the projection step of arbitrary fields in the Fixed-Mesh ALE method is obtained.

4 Fractional step approach for free surface flows

The formulation described so far allows one to solve a free surface problem involving incompressible fluids using finite elements. However, at each time step several linear systems need to be solved involving the momentum and continuity equations. The problem at this point is that if the number of

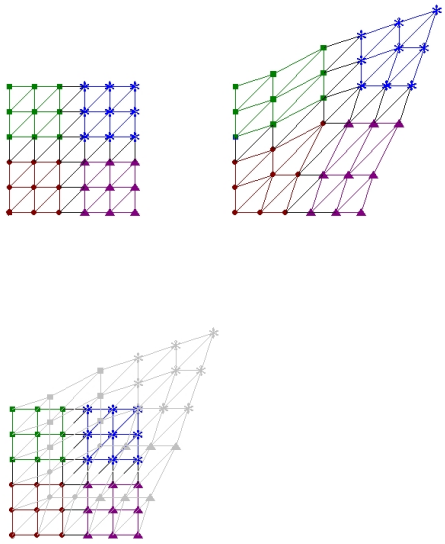


Figure 4: Projection from the deformed to the background mesh. Top left: original mesh. Top right: deformed mesh. Bottom left: projection process. Four different processors and subdomains exist in this case. In the bottom left image, the projection from the deformed to the background mesh is performed. Some nodes lay in the same processor for the deformed and the undeformed mesh (top left square green nodes, for instance). Other nodes however, need information from a different processor in order to do the projection step. This is the case of the bottom left central blue star node, which needs to be interpolated from the bottom-left red domain. Inter-processor communications are required in this case.

unknowns is large, these linear systems become ill-conditioned despite the stabilization methods employed. In these cases it is sometimes convenient to tackle this problem by splitting the global system into the iterative solution of subsystems involving only the velocity or the pressure unknowns, what are known as fractional step methods. In the following, two approaches for the fractional step splitting of the incompressible Navier-Stokes equations are presented. The first one is based on a modification of the monolithic incompressible Navier-Stokes equations, while the second one is associated to a purely algebraic approach.

4.1 Pressure Poisson equation based fractional step method

The type of fractional step schemes based on the solution of a Laplacian as an approximation to the pressure Schur complement were first proposed in [27]. In these schemes the solution of the monolithic system is decomposed into the solution of several simpler problems: a convection-diffusion equation for the velocity unknown, a pressure-Poisson equation and a projection step. Here we summarize the used fractional step method for a backward Euler time integration scheme, which can then be adapted to any other type of time integrator. We start by introducing an intermediate velocity $\hat{\boldsymbol{u}}_h$, for which we consider:

$$\hat{\boldsymbol{u}}_h^n = \boldsymbol{u}_h^n.$$

The steps of the fractional step scheme are then the following:

1. Convection-diffusion equation. We solve for the intermediate quantity $\hat{\boldsymbol{u}}_h^{n+1}$:

$$\rho\left(\boldsymbol{v}_h,\, \delta_t \hat{\boldsymbol{u}}_h^{n+1}\big|_{\boldsymbol{\chi}}\right) + B\left([\boldsymbol{v}_h,0]\,,\left[\hat{\boldsymbol{a}}_h^{n+1};\hat{\boldsymbol{u}}_h^{n+1},p_h^n\right]\right)$$
$$+S_{OSS}\left([\boldsymbol{v}_h,0]\,,\left[\hat{\boldsymbol{a}}_h^{n+1};\hat{\boldsymbol{u}}_h^{n+1},p_h^n\right]\right) + S_{Ghost}\left([\boldsymbol{v}_h,0]\,,\left[\hat{\boldsymbol{a}}_h^{n+1};\hat{\boldsymbol{u}}_h^{n+1},p_h^n\right]\right) = L\left([\boldsymbol{v}_h,q_h]\right), \tag{11}$$

where $\hat{\boldsymbol{a}}_h^{n+1} = \hat{\boldsymbol{u}}_h^{n+1} - \boldsymbol{u}_{h,}$

2. Pressure Poisson equation. We solve for p_h^{n+1} :

$$(q_h, \nabla\cdot\hat{\boldsymbol{u}}_h^{n+1}) + S_{OSS}\left([\boldsymbol{0},q_h]\,,\left[\hat{\boldsymbol{a}}_h^{n+1};\hat{\boldsymbol{u}}_h^{n+1},p_h^{n+1}\right]\right)$$
$$+S_{Ghost}\left([\boldsymbol{0},q_h]\,,\left[\hat{\boldsymbol{a}}_h^{n+1};\hat{\boldsymbol{u}}_h^{n+1},p_h^{n+1}\right]\right) = \delta t(\nabla(p_h^{n+1}-p_h^n),\nabla q_h), \tag{12}$$

3. Projection step: We solve for $\boldsymbol{u}_h^{n+1}$:

$$\rho\frac{1}{\delta t}(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1} - \hat{\boldsymbol{u}}_h^{n+1}) + (\boldsymbol{v}_h, \nabla p_h^{n+1} - \nabla p_h^n)$$
$$+S_{Ghost}\left([\boldsymbol{v}_h,0]\,,\left[\boldsymbol{0};\boldsymbol{u}_h^{n+1}-\hat{\boldsymbol{u}}_h^{n+1},p_h^n\right]\right) = 0. \tag{13}$$

The previous scheme provides a way to decouple the solution of the velocity and the solution of the pressure unknown. However, some errors are introduced which can affect the accuracy of the solution: the first error is due to the fact that the momentum equation (11) is solved for the intermediate velocity $\hat{\boldsymbol{u}}_h^{n+1}$instead of the end of step velocity $\boldsymbol{u}_h^{n+1}$. The second source of error is caused by the natural boundary condition of equation (12), which implicitly states that $\boldsymbol{n}\cdot\nabla p_h = 0$ on $\partial\Omega^0$. This causes a spurious pressure boundary layer to appear in the walls of the computational domain. Finally, and maybe the most important source of error in the case of free surface flows, a Dirichlet boundary condition needs to be provided for equation (12). The Dirichlet boundary condition we enforce is:

$$p = 0 \quad \text{on } \Gamma_{\text{free}}.$$

This condition is again applied using the *Linked-Lagrange multiplier* method. For low viscous flows this approximation is reasonable, since the effect of the viscosity is small. However, the proper boundary condition to be applied would be:

$$\boldsymbol{\sigma}\cdot\boldsymbol{n} = \boldsymbol{0} \quad \text{on } \Gamma_{\text{free}}.$$

As a consequence, an additional source of error is introduced by the fractional step method. Moreover, equation (12) allows to prescribe the end of step velocity values only in the direction normal to the boundary, $\boldsymbol{n}\cdot\boldsymbol{u}_h$, which is also a source of error.

4.2 Algebraic approach

Algebraic approaches for fractional step methods appear precisely to overcome some of the errors introduced by Laplacian based methods. At the algebraic level we can skip the discussion about the correct boundary conditions to be imposed. In fact, at the discrete level, the end-of-step velocity satisfies the good boundary conditions. Even more important, no boundary conditions have to be imposed for the pressure. See [7] for more details on algebraic pressure segregation methods for the incompressible Navier-Stokes equations. The numerical examples in Section 6 were run using the algebraic fractional step approach. We depart from the algebraic linear system for the monolithic problem, which is the algebraic counterpart of equation (4):

$$\begin{bmatrix} A_{uu} & A_{up} \\ A_{pu} & A_{pp} \end{bmatrix}\begin{bmatrix} U \\ P \end{bmatrix} = \begin{bmatrix} R_u \\ R_p \end{bmatrix},$$

with the straightforward identification of matrices and arrays. The algebraic approach we use for solving the incompressible Navier-Stokes equations consists in the following steps:

1. Solve for an intermediate velocity $\hat{U}^{n+1}$:

$$A_{uu}\hat{U}^{n+1} = R_u - A_{up}P^n. \tag{14}$$

2. Solve for an algebraic pressure Poisson equation. We solve for the pressure P^{n+1}:

$$\left(A_{pp} - A_{pu}\left[L(A_{uu})^{-1}A_{up}\right]\right)P^{n+1} = R_p - A_{pu}\hat{U}^{n+1} - A_{pu}\left[L(A_{uu})^{-1}A_{up}\right]P^n, \tag{15}$$

   where $L(A_{uu})^{-1}$ is the inverse of the lumping of matrix A_{uu}.

This purely algebraic approach for the fractional step solution of the incompressible Navier-Stokes equations works well for the split OSS stabilized formulation (4) using linear finite elements. In this case the lumping of the A_{uu} matrix, $L(A_{uu})$, does not contain any contribution from the stabilization terms, since all of them involve the spatial derivatives of $\boldsymbol{u}_h$, and these terms disappear due to the lumping. The viscous and convective terms also disappear since they contain spatial derivatives of $\boldsymbol{u}_h$. Also, A_{up} and A_{pu} do not contain any contribution from the stabilization terms. This causes that for linear elements, the resulting approximated Schur complement matrix:

$$S = A_{pp} - A_{pu}\left[L(A_{uu})^{-1}A_{up}\right], \tag{16}$$

is identical to explicitly built algebraic methods [31], where the Schur complement matrix is computed as:

$$S = A_{pp} - DM^{-1}G,$$

where matrix D is associated to the terms $(q_h, \nabla\cdot(\cdot))$, M is a lumped mass matrix arising from the temporal derivative terms $(\boldsymbol{v}_h, \partial_t(\cdot))$, and G is the gradient matrix associated to $(\nabla\cdot\boldsymbol{v}_h, \cdot)$. Note that the stabilization effect of the ghost-penalty terms remains in this formulation, since equation (14) continues to contain the original ghost-penalty terms for stabilizing the velocity gradients, and equation (16) continues to contain the original ghost-penalty terms for the stabilization of the pressure gradients. The lumping affects only the approximation of the Schur complement in equation (16).

For higher order elements, the purely algebraic approach cannot be directly built from the monolithic system for the incompressible Navier-Stokes equations, because the lumping of the mass matrix results in a singular matrix with zeroes on the diagonal. In these cases, a diagonal or easy to invert approximation to matrix A_{uu} should be devised. This is also the case for other stabilization methods like the residual based SUPG method, even in the case of linear elements: in this case there are stabilization terms in the A_{up} and A_{pu} matrices, and these stabilization terms affect the quality of the approximated Schur complement matrix. See [7] for a detailed review on algebraic fractional step methods.

Remark Taking into account the possibility of building the algebraic fractional step method directly from the matrices of the monolithic system, it is worth explaining how this has been implemented in an in-house parallel finite element computational code. Our finite element code relies on the PETSc library [14] for the solution of linear systems of equations. Amongst other features, the PETSc library allows to solve saddle point problems like the incompressible Navier-Stokes equations by building block preconditioners, in what are called *field split preconditioners*. In particular, the algebraic fractional step method described in equations (14)-(15) can be implemented by directly assembling the monolithic system (4) and then passing the following commandline options to the PETSc linear system solver:

```
-ksp_type richardson
-ksp_max_it 2

-pc_type fieldsplit
```

```
-pc_fieldsplit_type schur
-pc_fieldsplit_schur_factorization_type full
-pc_fieldsplit_0_fields 0,1
-pc_fieldsplit_1_fields 2
-pc_fieldsplit_schur_precondition selfp
-fieldsplit_1_mat_schur_complement_ainv_type lump
-fieldsplit_1_ksp_type preonly
-fieldsplit_1_pc_type ksp
```

Moreover, if the exact solution of the monolithic system is required, the algebraic fractional step method can be used as a preconditioner. In this case the Richardson driver can be replaced by a Krylov method, and the number of iterations can be incremented so that the monolithic solution is recovered:

```
-ksp_type gmres
-ksp_max_it 200
-ksp_tol 1e-8

-pc_type fieldsplit
-pc_fieldsplit_type schur
-pc_fieldsplit_schur_factorization_type full
-pc_fieldsplit_0_fields 0,1
-pc_fieldsplit_1_fields 2
-pc_fieldsplit_schur_precondition selfp
-fieldsplit_1_mat_schur_complement_ainv_type lump
-fieldsplit_1_ksp_type preonly
-fieldsplit_1_pc_type ksp
```

5 Adaptive approach

In the previous sections we have presented a stabilized variational formulation for the solution of incompressible free surface flows using the Fixed-Mesh ALE method and finite elements. This scheme needs now to be coupled with the adaptive refinement approach. For the adaptive refinement we rely in the algorithm presented in [10], which allows the distributed memory parallel finite element code to efficiently modify the mesh in a hierarchical manner (both refining and coarsening of the mesh are possible) so that it is refined only in the regions of the computational domain where it is required. In the following we summarize the main features of the adaptive refinement algorithm and its coupling with the Fixed-Mesh ALE method.

The adaptive refinement algorithm is based in a nodal parallel partition. This means that each node of the mesh belongs to a processor, whereas an element can belong to multiple processors if it owns nodes from more than one subdomain. This feature, which is strongly linked to the design of the parallel finite element code, introduces some particularities when designing the hierarchical mesh refiner, which are summarized next:

- The distributed data structure handles two and three dimensional unstructured meshes composed of triangular, quadrilateral, tetrahedral and hexahedral elements. This approach is capable of describing complex geometries, and doing non-uniform refinements.
- Each processor stores only the local information of the partitioned distributed mesh. This reduces the memory consumption, and allows scaling up to thousands of processors.
- The parallel refinement procedure is based on a hierarchical data structure for the refined elements of the mesh, that is used to efficiently search neighboring elements at the inter-processor level. A data structure containing parent and children pointers is implemented, where new refinement levels are successively added to or substracted from the computational mesh.

- Resulting meshes are non conforming with *hanging nodes* on sides where two levels of refinement meet. Contrary to other adaptive refinement methods, the algorithm proposed here does not enforce a *balancing* restriction in the refinement level of adjacent elements: the jump in the refinement level between neighbor elements can be arbitrarily large.

- For the parallel refinement process, the algorithm deals with element and node identification across processors by using a global element and global point identifier structure. This ensures that the global numbering structure and general nodal and elemental information can be transferred to all the neighboring processors in an efficient manner.

- To balance the processors load a dynamical parallel repartitioning framework is used that changes the ownership of the mesh nodes when load unbalance reaches a certain threshold, and then transfers the associated elements to the corresponding processors. Contrary to other algorithms for load rebalancing in hierarchical adaptive mesh refinement, our algorithm is independent from the renumbering strategy of the load rebalancing process. In particular, graph partitioning schemes and space-filling methods for load rebalancing can both be used.

Despite these interesting properties, from the Fixed-Mesh ALE point of view the adaptive refinement algorithm acts as a black box that allows to modify the mesh during runtime, with two particularities: First, a mesh refinement criteria needs to be provided so that the algorithm can decide which areas of the mesh need to be refined. Secondly, the finite element solver needs to deal with hanging nodes which, if a straightforward implementation is done, results in non-conforming discretizations.

With regards to the mesh refinement criteria, a very simple decision making algorithm has been used, which enforces that the elements occupied by the fluid are sufficiently fine. We define the refinement level as the number of hierarchical parents (in the refinement process) an element has. Based on this, the criteria reads as follows:

- If an element is occupied by the fluid and its refinement level is less than n_{Ref}, then refine it.

- Else if the element is less than n_{Lay} layers of elements away from the free surface Γ_{free} and its refinement level is less than $n_{\text{Ref}-\text{Lay}}$, then refine it.

- Else if the element is occupied by the fluid and its refinement level is greater than n_{Ref}, then unrefine it.

- Else, unrefine the element.

This ensures that the refinement level of the elements covered by the fluid is sufficiently fine and also that some additional refinement can be introduced in the region surrounding Γ_{free} in order to have a more accurate representation of the geometry of the free surface. The only care which needs to be taken is in the parallel implementation of this refinement criteria: as explained in [10], the parallel refinement algorithm requires that the refinement criteria passed to the refiner is the same in all the processors. The problem for this is that the n_{Lay} criteria is geometric and needs to propagate from the free surface through several layers of elements. When these layers cross from one subdomain (attached to a parallel processor) to another, communications are needed. The solution to this problem is to communicate the refinement criteria by marking the nodes of the elements to be refined and then proceeding to communicate this information through the processor boundaries as a usual nodal unknown communication in a nodally based partition finite element code. Obviously, an adaptive refinement strategy based on error estimators could also be used, although this has not been used in the numerical experiments presented in this work.

Regarding the way to deal with hanging nodes, the main problem is that the standard shape functions for hanging nodes result in non-conforming finite element approximations. It is possible to deal with non-conforming approximations in the variational form (see for instance [6], where

Discontinuous-Galerkin terms are used to deal with the discontinuities of shape functions in meshes with hanging nodes), but then a specifically designed variational formulation needs to be developed for each physical equation. The second possibility consists in modifying the shape functions so that they become conforming. For this, the adaptive refinement algorithm presented in [10] provides information on what are called hanging parents and their averaging coefficients, and then the finite element solver can proceed as if the considered mesh is flat, instead of hierarchical. The procedure consists in completely eliminating hanging nodes from the finite element equations, and the contribution of their test and shape functions is added to their hanging parents, whose test and shape functions are then modified. In general, the test and shape functions of a hanging parent node (hp), with several hanging children nodes (hc_i), are modified in the following manner:

$$N^*_{hp}(\boldsymbol{x}) = N_{hp}(\boldsymbol{x}) + \sum_{i=1}^{nchild} (N_{hp}(\boldsymbol{x}_i) N_{hc_i}(\boldsymbol{x}),$$

where N_{hp} and N_{hc_i} denote the original shape and test functions for the parent and children nodes respectively, $N^*_{hp}(\boldsymbol{x})$ denotes the modified hanging parent shape function, and $\boldsymbol{x}_i$ denotes the position in space of the ith hanging children. The test and shape functions for the hanging children nodes are completely eliminated from the final finite element problem.

Take for instance the example in Fig. 5. The shape and test function for node 1, N_1, is modified into N^*_1 so that:

$$N^*_1(\boldsymbol{x}) = N_1(\boldsymbol{x}) + \frac{1}{2} N_3(\boldsymbol{x}) + \frac{1}{4} N_4(\boldsymbol{x}).$$

Similarly, the test and shape functions for node 2, N_2, is modified into $N^*_2(\boldsymbol{x})$ so that:

$$N^*_2(\boldsymbol{x}) = N_2(\boldsymbol{x}) + \frac{1}{2} N_3(\boldsymbol{x}) + \frac{3}{4} N_4(\boldsymbol{x}).$$

The shape and test functions for nodes 3 and 4 are completely eliminated from the finite element equations, although their nodal value can be recovered once the finite element problem has been solved by saying:

$$u_4 = \frac{1}{4} u_1 + \frac{3}{4} u_2,$$
$$u_3 = \frac{1}{2} u_1 + \frac{1}{2} u_2.$$

Note that these relations are recursive, so an arbitrary difference in the refinement level of neighbor elements is possible.

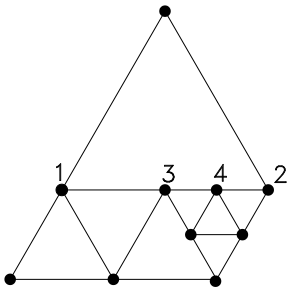


Figure 5: Hanging nodes. Node 4 is a hanging node, its hanging parents being nodes 2 and 3. Node 3 is in turn a hanging node, its hanging parents being nodes 1 and 2. The hanging node value for the unknown at node 4, u_4 is: $u_4 = \frac{1}{2}u_3 + \frac{1}{2}u_2 = \frac{1}{4}u_1 + \frac{3}{4}u_2$.

6 Numerical experiments

In this section we present some numerical experiments which illustrate the behavior of the proposed method.

6.1 2D sloshing in a rectangular tank

In this first numerical example we study the numerical performance of the proposed algorithm. For this, a bi-dimensional tank of dimensions 1.73×1.05 m^2 partially filled with water up to a height of 0.6 m is studied. The tank is then subject to an horizontal movement, which is characterized by an amplitude of 0.031 m and a period of 1.5 s:

$$x = 0.031 \cdot \sin\left(\frac{4\pi}{3}t\right). \tag{17}$$

The setting is similar to the one presented in section 6.2 and Fig. 12 , but the simulations are done in 2D so that the numerical method can be assessed more easily.

6.1.1 Spatial convergence

In this example we do not compare with experimental results, but with results obtained using a finer finite element mesh. The reference solution is obtained by solving the problem in a relatively fine and uniform mesh composed of 104960 linear elements. The first test we perform consists in studying the spatial convergence of the method. Only 10 time steps are simulated with a time step of 0.01 seconds. At each time step, the position of the free surface at 50 uniformly spaced height gauges is measured, and compared to the solution in the reference mesh. The relative error is computed as:

$$e = \sqrt{\int_0^{1.73}\int_0^{0.01} \left(h_{\text{coarse}}(x,t) - h_{\text{fine}}(x,t)\right)^2 \mathrm{d}x\mathrm{d}t},$$

where $h_{\text{coarse}}(x,t)$ and $h_{\text{fine}}(x,t)$ represent the height of the free surface at an abscissa x at a given time t, for the coarse and fine meshes respectively. Since very few time steps are simulated, the cumulative error caused by the time integration does not affect the results of the convergence study. In order to study, not only the convergence of the method, but also the effect of the adaptive mesh refinement, the convergence study is done in two different kinds of meshes: in the first kind of mesh,

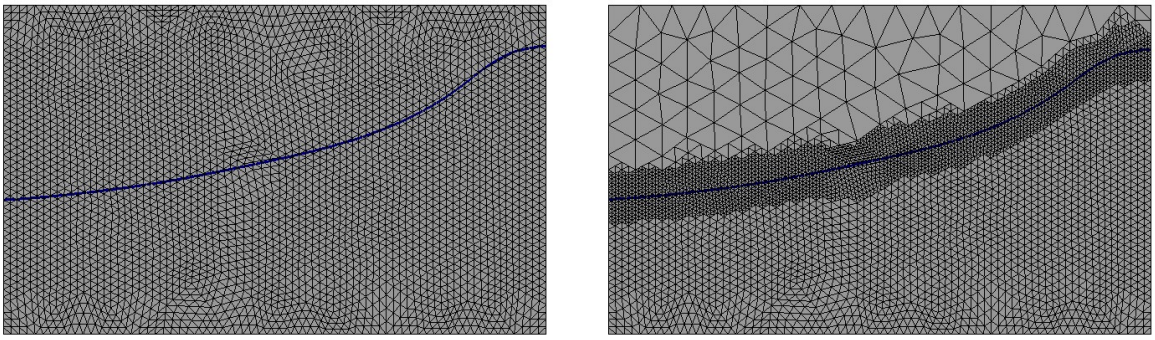


Figure 6: Uniformly refined mesh (left) and adaptively refined mesh (right) used for the convergence study in the 2D sloshing tank.

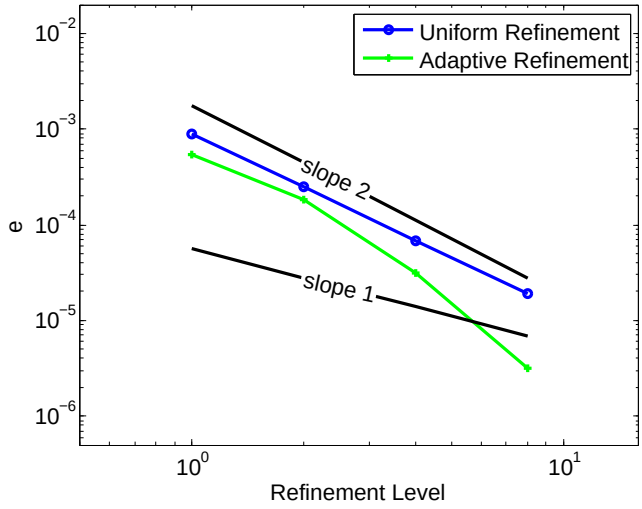


Figure 7: Spatial convergence for the 2D sloshing in a rectangular tank.

the refinement is uniform. In the second one, the criteria explained in section 5 is applied, with 1 additional refinement level in the 5 first layers of elements. Fig. 6 shows a uniformly refined mesh, and an adaptively refined mesh for a given refinement level. Note that the adaptively refined mesh does not add elements in the region of the flow which is not occupied by the fluid, and concentrates elements in the region surrounding the free surface.

Fig. 7 shows the spatial convergence results for both the uniformly refined and the adaptively refined cases. It can be observed that the spatial convergence ratio is quadratic (as expected, since linear elements are used), and that the results can be improved at a low increase in the computational cost by using the adaptive refinement technique.

6.1.2 Temporal convergence

The second test we perform consists in studying the convergence of the temporal discretization. For this, we consider a fixed element discretization size (with a uniformly refined mesh) which results in a mesh composed of approximately 7000 elements and we study the effect of the variation of the time

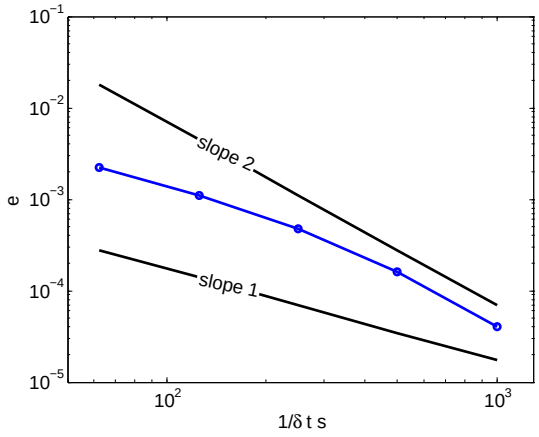


Figure 8: Temporal convergence for the 2D sloshing in a rectangular tank.

step size. Again, the reference solution is obtained by considering a small time step of 0.001 s. For this example, a viscosity of $\mu = 1\text{Pa}\cdot\text{s}$ is considered instead of the viscosity of water so that the tests do not require of a turbulence model. The relative error associated to the temporal discretization is then computed as:

$$e = \sqrt{\int_0^{1.73} \left(h_{\text{coarse}}(x,t_{\text{end}}) - h_{\text{fine}}(x,t_{\text{end}})\right)^2 \mathrm{d}x},$$

where now $h_{\text{coarse}}(x,t_{\text{end}})$ and $h_{\text{fine}}(x,t_{\text{end}})$ represent the position of the free surface at the end of the simulation computed with the coarse time step and with the reference fine time step respectively. A total simulation time of 0.32 s is computed. Fig. 8 shows the results obtained with the Fixed-Mesh ALE method. A second order time integrator (BDF2) is being used in all the equations (advection of the level set, integration of the Navier-Stokes equations), so a second order in time scheme is obtained for sufficiently refined time steps.

6.1.3 Effect of the ghost stabilization terms

It is also interesting to evaluate the effect of some of the numerical ingredients of the proposed formulation. We have already seen that the ghost penalty terms do not harm the spatial convergence of the solution, in the following test we deactivate the ghost penalty stabilization terms and observe the results if these terms are not considered. As expected, the solution explodes when a bad cut is found. This is shown in Fig. 9 where the velocity field and level set function for a bad cut configuration are shown with and without ghost penalty stabilization terms. The velocity without ghost stabilization terms explodes as expected, resulting in a non-physical position of the computed free surface. Also, the condition number of the linear system matrix explodes to $5.4 \cdot 10^{14}$. On the other hand, with the ghost stabilization terms, the velocity field is controlled (although with some oscillations in the nodes outside the physical domain, with little influence in the $\Omega(t)$ region), and the free surface position is correct for the corresponding physical configuration. The condition number for the linear system matrix in this very bad configuration is $1.75 \cdot 10^{11}$, which although large, is much better than the condition number without the ghost penalty stabilization terms.

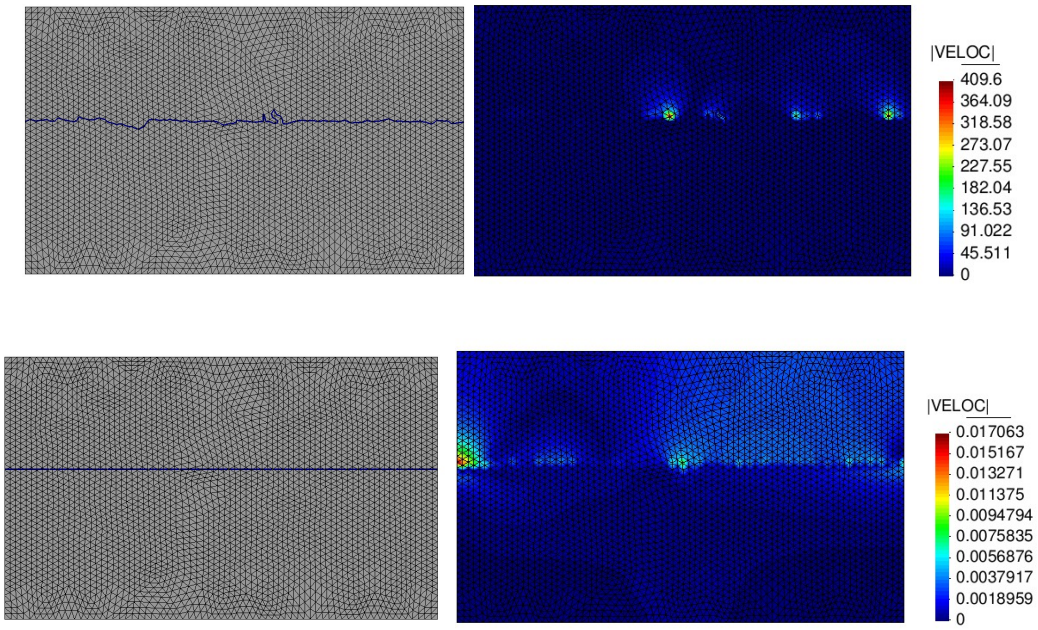


Figure 9: Effect of the ghost penalty stabilization terms. Top: Result without the ghost penalty stabilization terms for a bad cut configuration. Bottom: Results with the ghost penalty stabilization terms.

6.1.4 Effect of the mass loss correction scheme

Here we study the effect of the mass loss correction terms. For this a very coarse mesh (410 linear triangle elements) and a very coarse time step (0.2 s) are used, together with the very dissipative backward Euler scheme. Fig. 10 shows the initial volume of water, and the volume of water obtained with, and without mass correction scheme. As expected, if no correction is applied, a great amount of mass is lost in the process. On the other hand, the mass is conserved if the global mass correction scheme is used.

6.1.5 Effect of the level set reinitialization scheme

Another ingredient of the formulation is the level set reinitialization scheme. Here we evaluate the results for a mesh of 1640 linear triangles and a time step of 0.1 s after 200 steps have been simulated. Two cases are considered: in the first case no reinitialization is done at all, in the second one, a reinitialization step is done at each time step. In Fig. 11 the level set function and position of the free surface are presented at the end of the simulation. It can be observed that the level set function is different if the reinitialization scheme is used, but the position of the free surface is practically the same in both cases, even after 200 reinitialization steps. From this we conclude that the proposed reinitialization scheme is a good option for the transport of level set functions.

6.2 3D sloshing in a rectangular tank

This example was first studied in [40], where both numerical and physical experiments were performed and the elevation of the free surface of the problem was tracked. An almost bi-dimensional rectangular tank of dimensions $1.73 \times 1.05 \times 0.2$ m is partially filled with water up to a height of 0.6 m. The tank dimensions and initial configuration of the free surface are shown in Fig. 12.

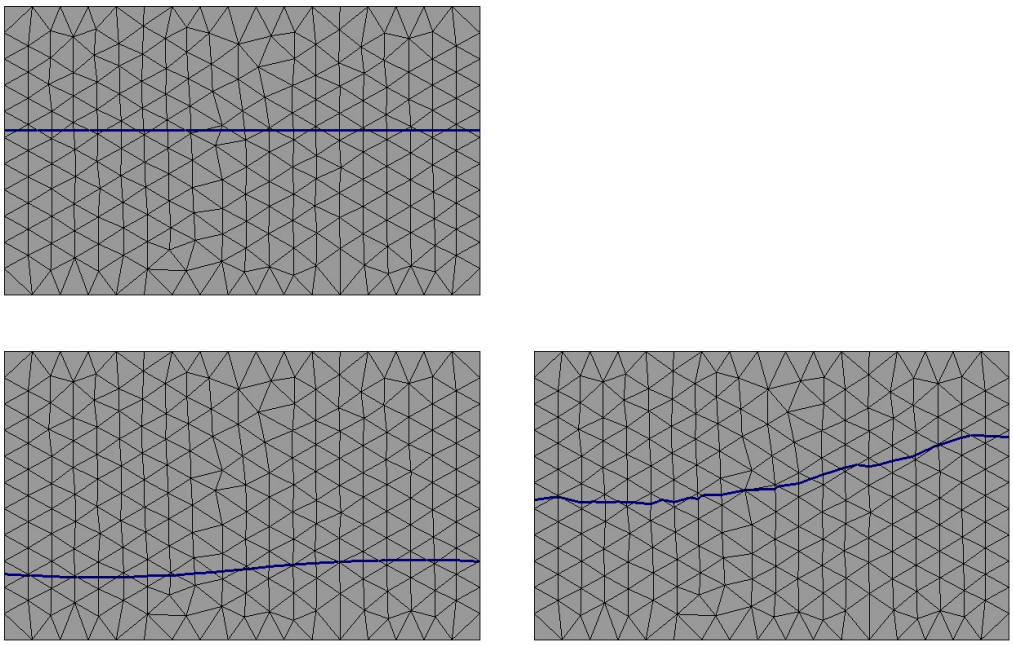


Figure 10: Effect of the mass loss correction scheme. Top: Initial level set configuration. Bottom: Level set position at the end of the simulation without (left) and with (right) mass loss correction scheme.

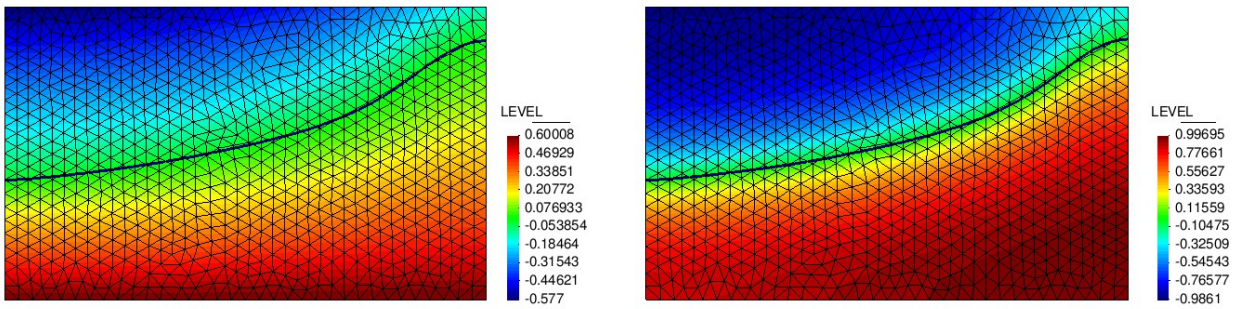


Figure 11: Effect of the level set reinitialization scheme. Left: Without reinitialization. Right: With reinitialization.

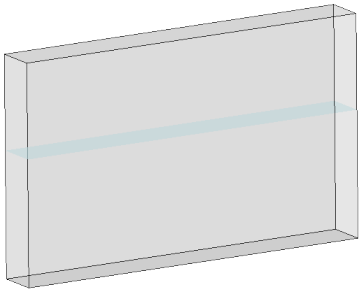


Figure 12: Initial configuration for the sloshing in a rectangular tank ($1.73 \times 1.05 \times 0.2$ m) problem.

The tank is then subject to an horizontal movement, which is characterized by an amplitude of 0.031 m and a period of 1.5 s, the same oscillation which was applied in the example in section 6.1 and in equation (17). When this displacement is applied to the tank, the interior fluid starts oscillating. The time history of the elevation (coordinate y) at a height gauge situated at coordinates $x = 0.05$ m, $z = 0.1$ m is then measured and compared to experimental results.

Experimental results show that the contribution of the depth coordinate to the solution of this problem is negligible, so a bidimensional mesh of dimensions 1.73×1.05 m is used. Slip boundary conditions which allow the free movement of the fluid in tangential directions are enforced on the tank boundary. A time step of 0.01 s is set and second order backward difference schemes are used for the time integration. An initial mesh of 4000 elements is used, from which all refined meshes are built. The mesh is refined around the section which is cut by the free surface, allowing for a better tracking of the fluid interface. In this case the refinement criteria is as follows: the mesh is refined only one level in the region surrounding the free surface and up to three layers of elements in each side. This results in a computational cost similar to the original problem without refinement, but it also allows a higher precision in the selected region. As it can be observed in Fig. 13, the mesh is successively refined and unrefined so that only the area surrounding the free surface is solved using the fine mesh at each time step. For the linear system of equations strategy, the algebraic fractional step approach presented in Section 4.2 is used, but we allow the solver to iterate until the monolithic solution is recovered.

Fig. 13 shows the velocity field at the free surface at a time shortly after the water surface impacts the tank ceiling. The obtained velocity and level set function fields are smooth, the adaptive mesh allows to accurately (and smoothly) capture the geometry of the free surface, as well as the velocity and pressure fields. Fig. 14 shows a smooth pressure field in one of the simulation steps.

Fig. 15 compares the water surface elevation at the height gauge in the experimental case presented in [40] with the results obtained with the proposed numerical method. As it can be observed, the natural oscillation period of the fluid in the tank does not coincide with the forced oscillation period, which is the cause for the beating effect observed, in which the amplitude of the fluid oscillation increases and mitigates successively.

A good agreement is obtained, both in the oscillation frequency, the beating frequency and the amplitude of the oscillations. For the first beating period the agreement is almost exact, despite the initial conditions being different for the experimental and numerical problems. After that, the error in the oscillation frequency and the effect of the initial conditions accumulates and the measured heights do not exactly overlap, although the beating period (of around 12 s) is also correctly captured by the numerical method.

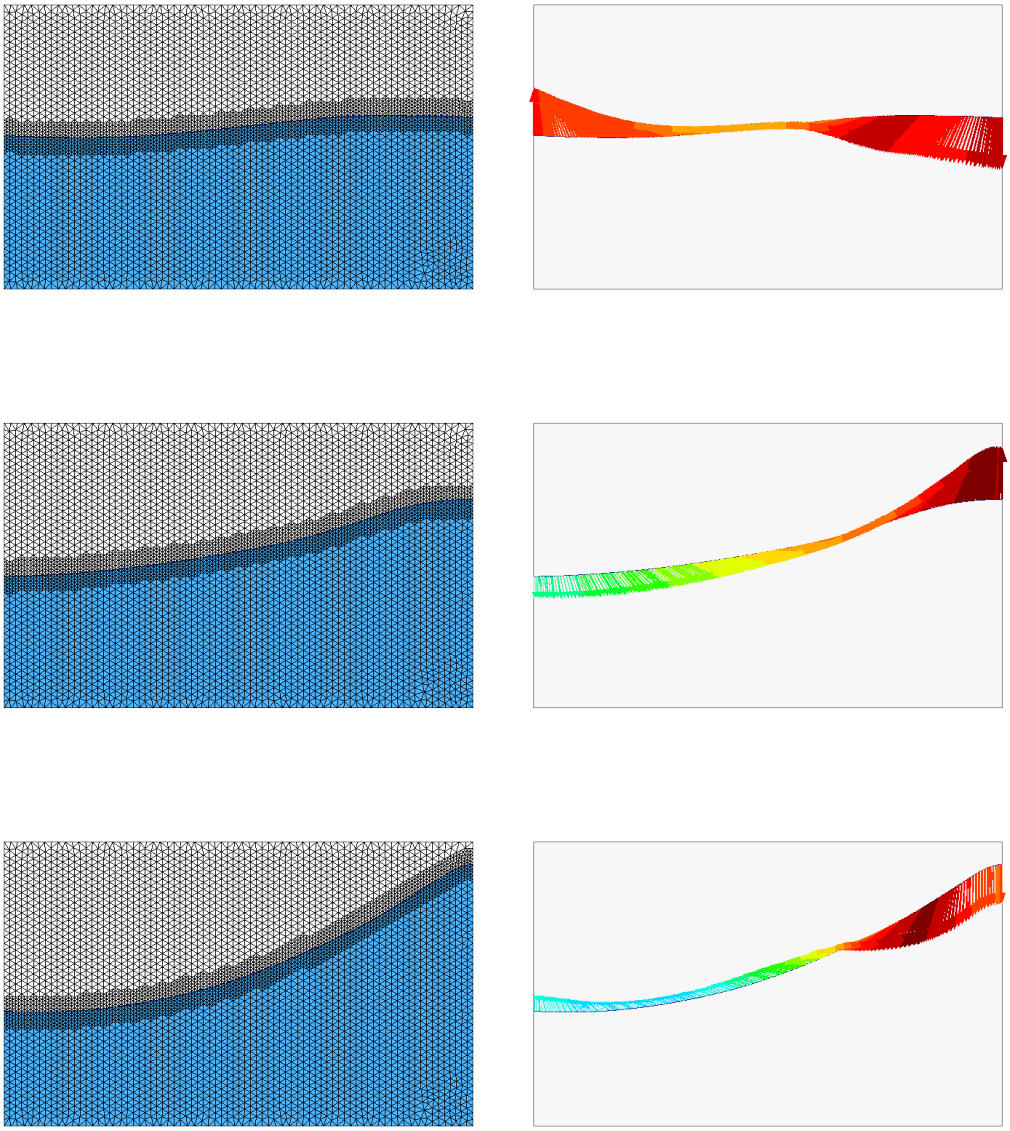


Figure 13: Mesh and position of the free surface (right) and velocity vectors (right) at several time instants of the simulation. Vector colors indicate velocity magnitudes, vector arrowheads indicate velocity direction.

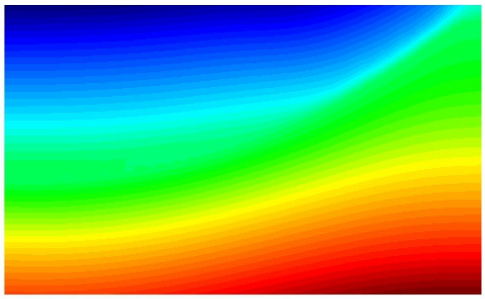


Figure 14: Smooth pressure field for the sloshing problem.

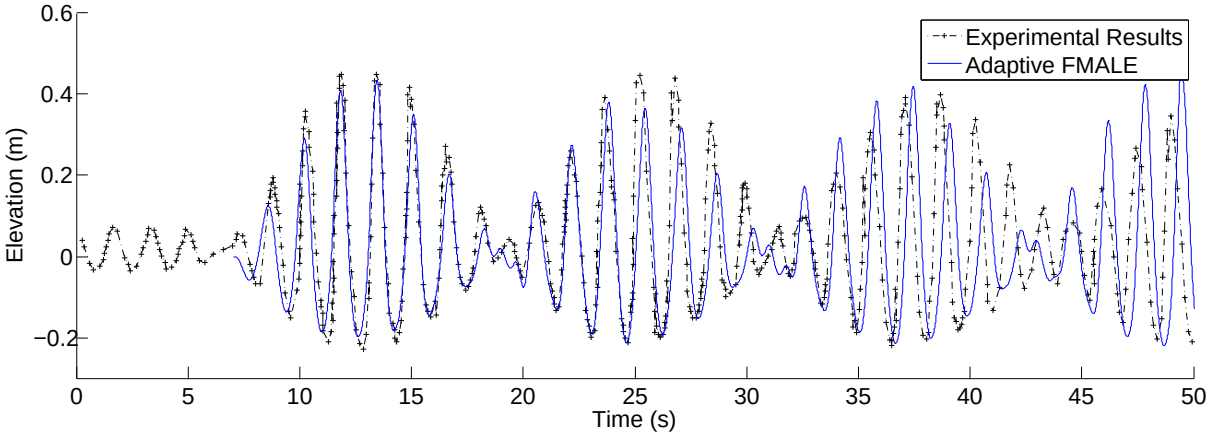


Figure 15: Water surface elevation at an elevation gauge situated at $x = 0.05$, $z = 0.1$.

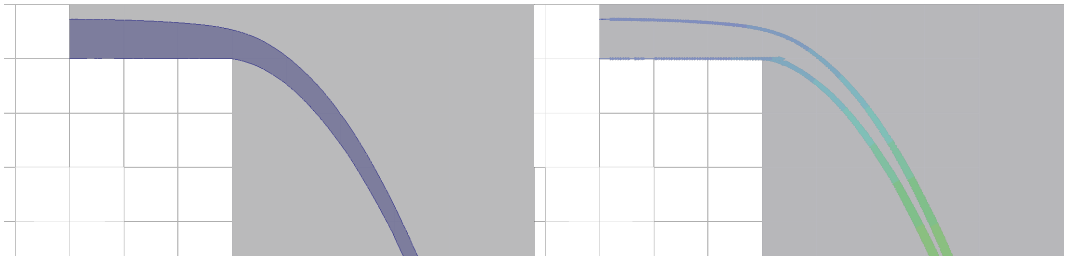


Figure 16: Free falling jet problem. Left: free surface plot. Right: Velocity vector field when the stationary state is reached. Vector colors indicate velocity magnitude, vector arrowheads indicate direction.

6.3 Free overfall experiment

In this numerical example we study the behavior of a free falling jet. The problem setting is depicted in Fig. 16. The free surface flow enters the problem domain in the top-left area with a uniform horizontal velocity of 0.857 m/s, and a free surface height of 0.075 m. Shortly after the entry point, the canal abruptly ends and the flow falls freely forming a jet. In this case the shape of the jet is studied and compared against the experimental results presented in [53].

In this case we depart from a finite element mesh composed of approximately 1000 triangular elements, and we adaptively refine it only in the region around the free surface region (4 hierarchical levels of refinement are considered) until a mesh of 33301 elements is obtained. The time step is set to 0.01 s, and a second order backward differences in time integration scheme is used. The simulation is run until a stationary state is reached. As it is shown in Fig. 16, the resulting free surface is smooth even in the point where the water jet separates from the canal wall, although some numerical oscillations appear which could be treated by using for instance a shock capturing technique for the advection of the level set function.

The equivalent uniform refinement mesh covering the whole computational domain region would be composed of 515690 , from where it is clear that an important gain in the required computation effort is obtained. The adaptively refined mesh when the stationary state is reached is depicted in Fig. 17. The mesh is heavily refined in the area surrounding the free surface, jumps of up to 4 refinement levels are obtained between neighbor elements. Finally, in Fig. 18 the results of the numerical simulation are compared against experimental results presented in [53]. The agreement is almost exact, thanks to the capability of the Adaptive Fixed-Mesh ALE method of concentrating the computational effort in the critical regions, which in this case are the free surface and specially the separation point where the jet departs from the canal and starts its free fall movement.

6.4 Green water flow experiment

This numerical example consists in the simulation of the water flow over the deck of a ship followed by the impact of the resulting wave against an obstacle. This is a well known benchmark for free surface problems and has been studied experimentally by the Maritime Research Institute Netherlands (MARIN). Experimental data are available in [1]. The setting of the problem is the following: a large tank with an open roof is considered. In one of the sides of the tank, a gate aisles a volume of water which lays at rest at a constant free surface height. In the other side of the gate there is a prismatic obstacle representing a container on the deck of a ship. This obstacle is static and no fluid-structure interaction effects are considered. The initial configuration is depicted in Fig. 19. The experiment starts when the gate quickly opens (the duration of the gate opening process is considered to be zero) and the resulting water wave impacts against the obstacle and the walls of the tank. The pressure

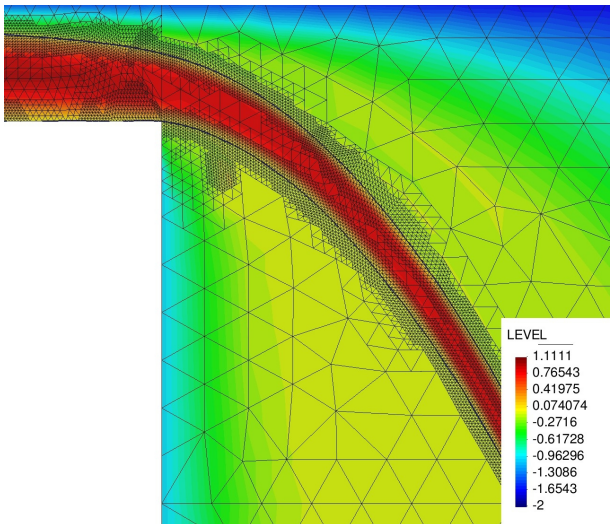


Figure 17: Free falling jet problem. Adaptively refined mesh and level set function field when the stationary state is reached.

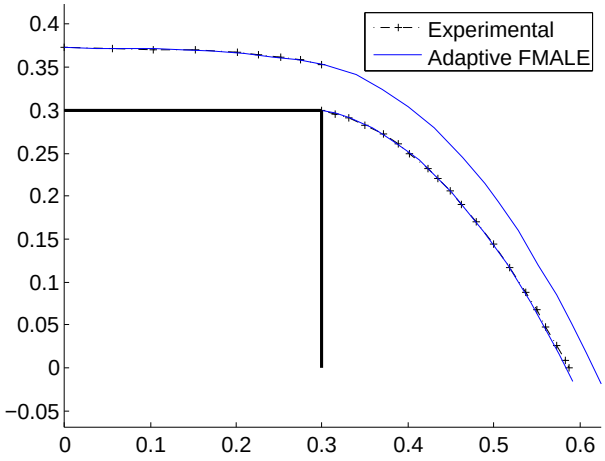


Figure 18: Comparison of experimental and numerical simulation results for the free falling jet problem.

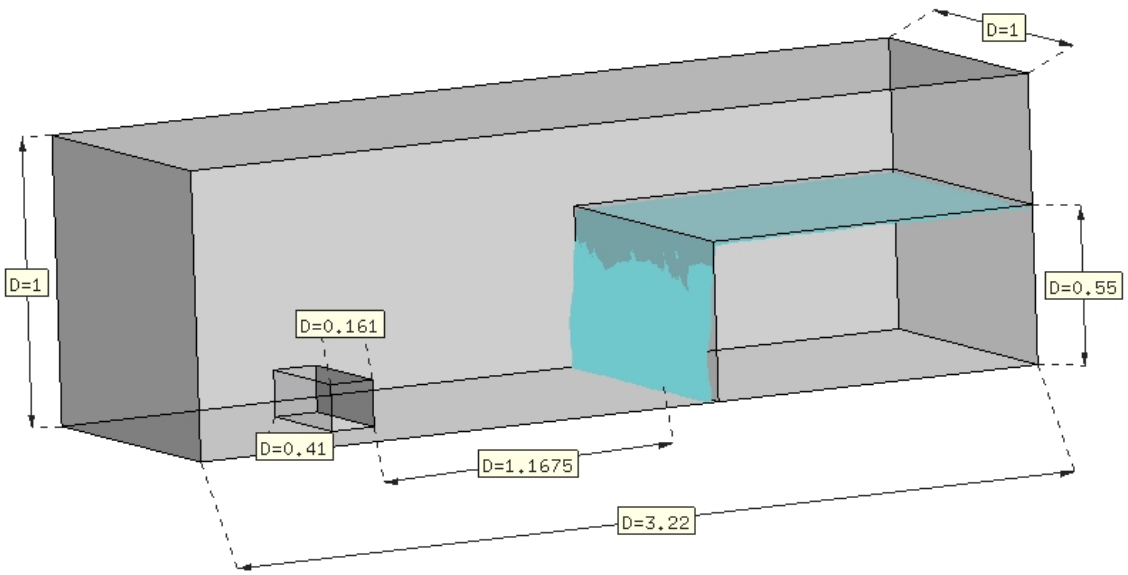


Figure 19: Kleefman's dambreaking experiment initial configuration

time history at several points on the surface of the obstacle is then measured and compared against experimental results.

An initial mesh composed of 3756 linear tetrahedral elements is used. Slip boundary conditions which allow the free movement of the fluid in tangential directions are enforced on the walls. These boundary conditions are enforced in a strong manner. At each time step the mesh is adapted so that two additional levels of refinement are applied to the region occupied by the fluid, and one additional level of refinement is applied in the three layers of elements closest to the free surface. The time step is set to 0.01 s and a total of 6 s are simulated. Again, second order backward difference schemes are used for the integration of the problem in time. The finite element mesh is then successively adapted until an average of half a million elements is obtained for the computational mesh at each time step. The mesh refinement is heavily localized around the free surface. The resulting adaptively refined mesh element sizes are equivalent to a uniform mesh of two and a half million elements if the element size used for the free surface area is uniformly applied to the whole computational domain, which leads to important savings in the computational effort.

In Fig. 20 the position of the free surface at several time instants is shown, together with the mesh at each of these time steps. As it can be observed, the mesh is heavily refined in the region surrounding the free surface, it is coarser in the fluid regions farther from the free surface, and extremely coarse in the region of the domain which is not occupied by the fluid, and thus of no interest for the simulation. Fig. 21 shows the comparison of the pressure time history obtained through numerical simulation with experimental results. Numerical and experimental results show a very good agreement both qualitatively and quantitatively. The most salient difference between experiments and numerical simulations is that the returning wave (which impacts the obstacle at $t \simeq 5$ s after the water front has reached first the left wall of the tank, then the right wall, and has returned to finally collide with the obstacle) is slightly delayed in the numerical experiments. Apart from this, the pressure levels through time match the pressure values of experimental results with a good accuracy. Fig. 22 shows the velocity vectors at the water surface. Thanks to the stabilization terms the velocity field

is smooth even at the free surface where bad cuts of the mesh elements can occur depending on the geometry of the free surface.

7 Conclusions

In this work we have presented an adaptive Fixed-Mesh ALE (Arbitrary-Lagrangian-Eulerian) method for free surface flows. The main ingredients of the proposed strategy are the use of stabilized formulations, including stabilization terms for the cut elements in the free surface and terms stabilizing convection and the classical LBB condition, the use of an ALE frame of reference for accurately computing the temporal derivatives and the contribution of the advective term, a method to prescribe boundary conditions on non-matching meshes, and the coupling with a set of adaptive mesh refinement algorithms. These adaptive mesh refinement algorithms allow for the solution of the problem in a distributed memory environment while reducing the overall computational cost by focusing the computational effort in the regions where it is needed at each time step. Other important components of the formulation are a simple mass correction procedure which ensures that mass is conserved at the global level throughout the simulation, an efficient algorithm for the element search in the projection step from the deformed to the background mesh, and the use of an algebraic fractional step approach for the incompressible Navier-Stokes equations which ensures that the resulting systems of equations is easy to solve while at the same time it converges to the solution of the original monolithic problem.

The performance of the method has been tested in three numerical examples representing the sloshing inside a tank subject to a horizontal acceleration, a free falling jet, and the impact of a wave against an obstacle after a quick gauge opening. Both spatial and temporal convergence have been assessed, and the behavior of each of the ingredients of the formulation has been successfully tested. The results obtained for the proposed adaptive Fixed-Mesh ALE method are in good agreement with experimental results, and the obtained velocity and pressure fields are smooth thanks to the terms stabilizing both the bad element cuts and the convective and inf-sup sources of instability.

Acknowledgements

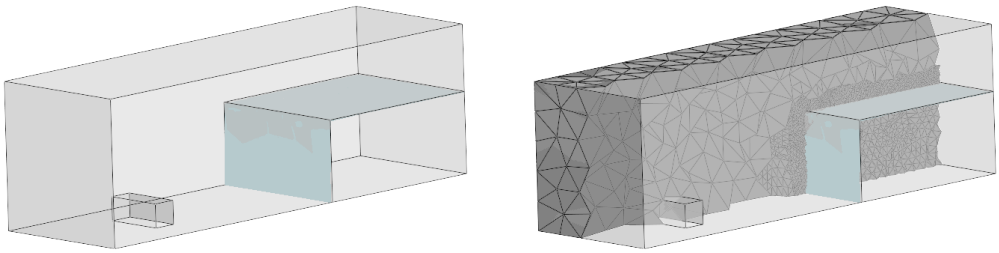
The authors would like to acknowledge Eric Burman for the fruitful discussions on the Ghost Penalty method. This work is partially funded through the ELASTIC-FLOW project, Ref. DPI2015-67857-R of the Spanish Government. J. Baiges acknowledges the support of the Spanish Government through the Ramón y Cajal grant RYC-2015-17367. R. Codina acknowledges the support received though the ICREA Acadèmia Research Program of the Catalan Government. E. Castillo gratefully acknowledges the support received from CONICYT (Programa de Formación de Capital o Humano Avanzado/Becas-Chile Folio 72120080).

References

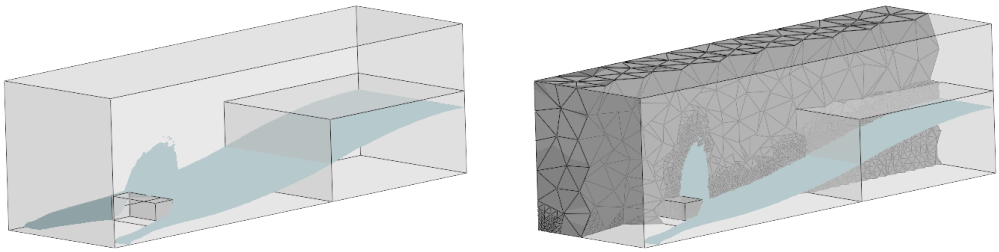
[1] S.E.R.I. community. `http://wiki.manchester.ac.uk/spheric/index.php`..

[2] I. Akkerman, Y. Bazilevs, C. Kees, and M. Farthing. Isogeometric analysis of free-surface flow. *Journal of Computational Physics*, 230(11):4137 – 4152, 2011. Special issue High Order Methods for {CFD} Problems.

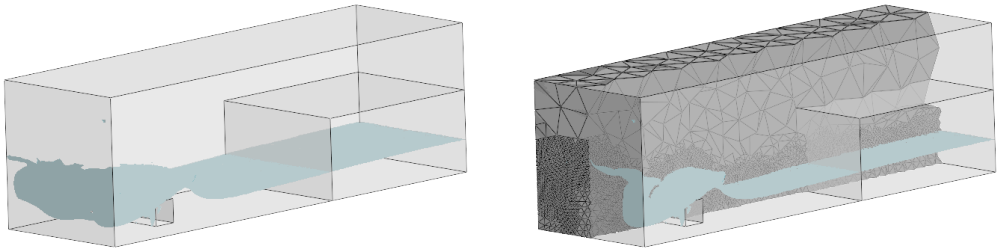
[3] S. Amdouni, M. Moakher, and Y. Renard. A local projection stabilization of fictitious domain method for elliptic boundary value problems. *Applied Numerical Mathematics*, 76:60–75, 2014.



(a) $t = 0.03$ s

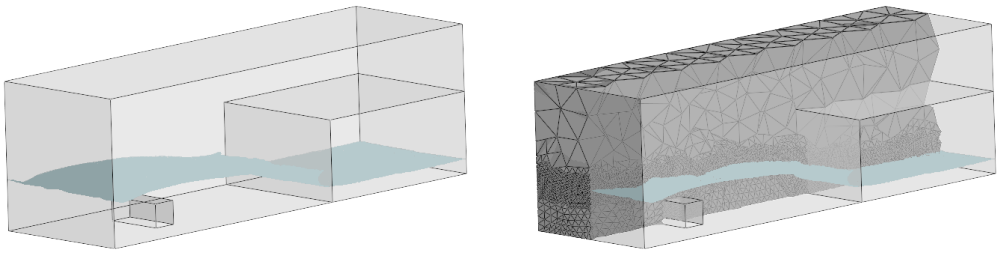


(b) $t = 0.6$ s

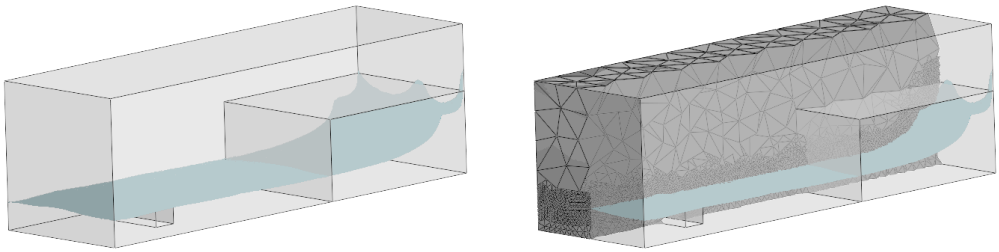


(c) $t = 1.29$s

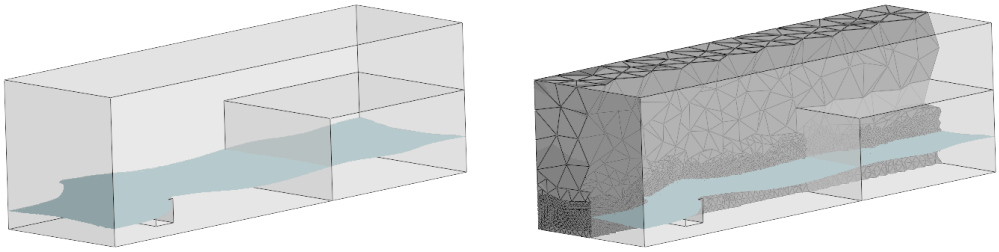
Figure 20: (a) Position of the free surface and computational mesh at several instants of the simulation for the green water flow case.



(d) $t = 2.61$ s



(e) $t = 3.87$ s



(f) $t = 5.49$ s

Figure 20: (b) Position of the free surface and computational mesh at several instants of the simulation for the green water flow case .

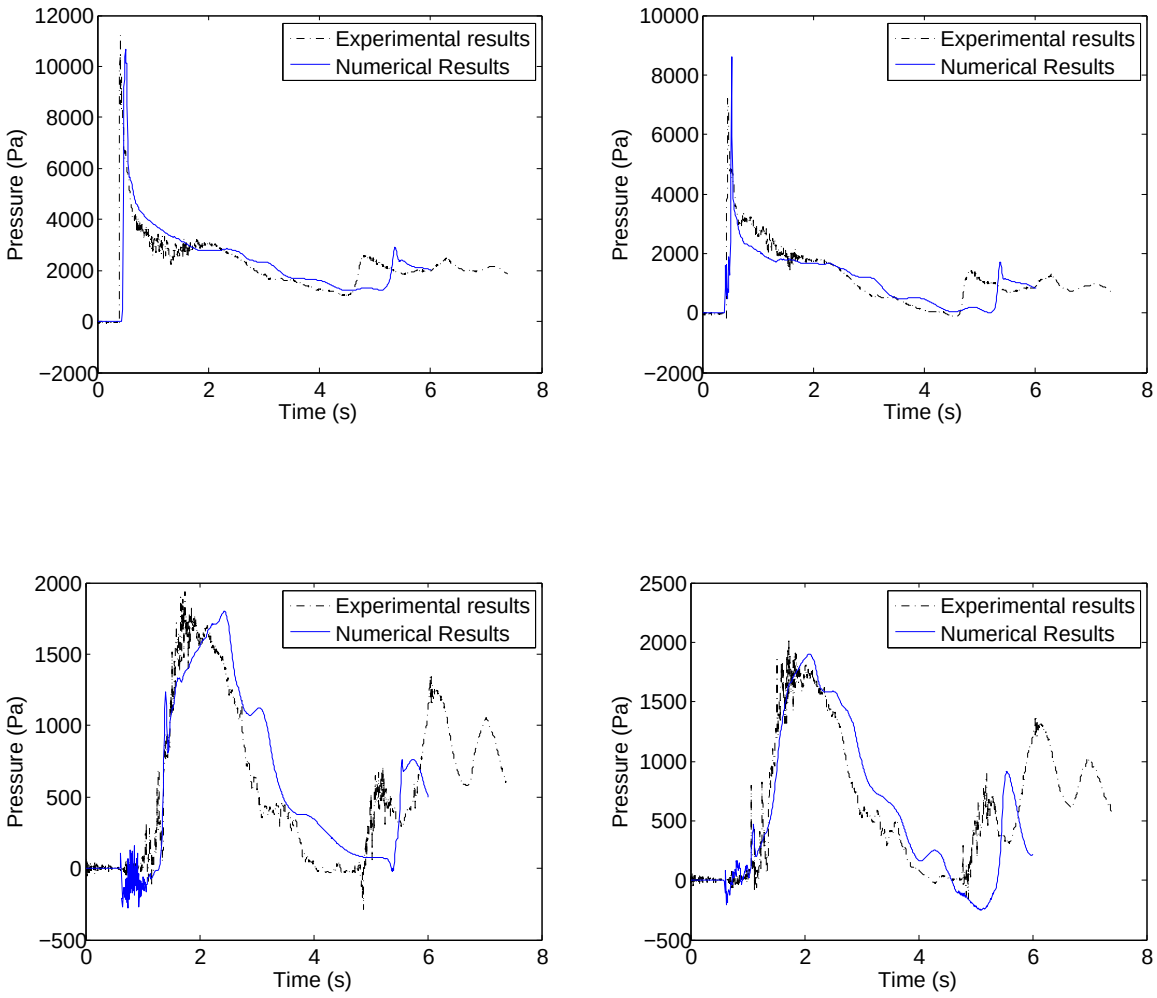
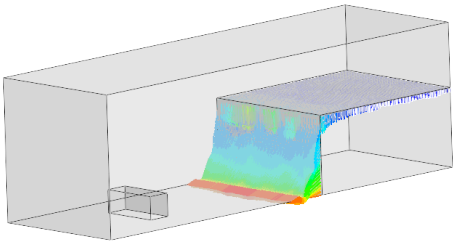
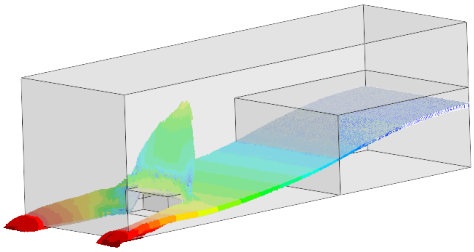


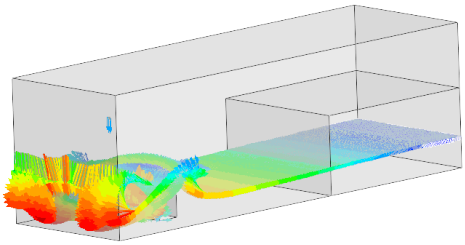
Figure 21: Comparison of experimental and numerical values for the pressure time history at several points of the obstacle surface (P1 to P4). From left to right and then top to bottom: P1: (0.8245, 0.021, 0.4965), P2: (0.8245, 0.141, 0.4965), P3: (0.7635, 0.161, 0.4965), P4: (0.7235, 0.161, 0.4965)



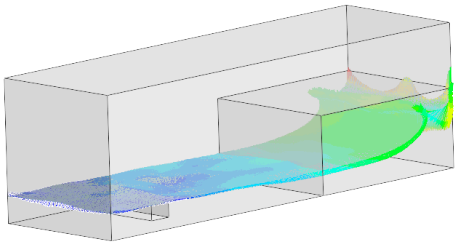
(a) $t = 0.03$ s, $v_{\max} = 1.06$ m/s.



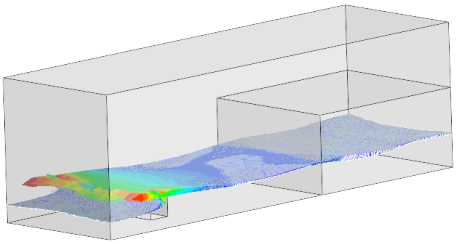
(b) $t = 0.6$ s, $v_{\max} = 1.73$ m/s.



(c) $t = 1.29$ s, $v_{\max} = 2.69$ m/s.



(d) $t = 3.87$ s, $v_{\max} = 1.86$ m/s.



(e) $t = 5.49$ s, $v_{\max} = 2.59$ m/s.

Figure 22: Velocity vectors at the free surface for the dam-break problem at several time steps. Vector colors indicate velocity magnitudes (with light red indicating the maximum velocity, dark blue indicating fluid at rest), vector arrowheads indicate flow direction. The maximum fluid nodal velocity at each time step is indicated in the captions.

[4] R. Amini, R. Maghsoodi, and N. Z. Moghaddam. Simulating free surface problem using isogeometric analysis. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 38(2):413–421, 2016.

[5] H. Askes and L. J. Sluys. Remeshing strategies for adaptive ale analysis of strain localisation. *European Journal of Mechanics-A/Solids*, 19(3):447–467, 2000.

[6] S. Badia and J. Baiges. Adaptive finite element simulation of incompressible flows by hybrid continuous-discontinuous galerkin formulations. *SIAM Journal on Scientific Computing*, 35(1):A491–A516, 2013.

[7] S. Badia and R. Codina. Algebraic pressure segregation methods for the incompressible Navier-Stokes equations. *Archives of Computational Methods in Engineering*, 15(3):1–52, 2007.

[8] S. Badia, F. Nobile, and C. Vergara. Robin-Robin preconditioned Krylov methods for fluid-structure interaction problems. *Computer Methods in Applied Mechanics and Engineering*, 198:2768–2784, 2009.

[9] S. Badia, A. Quaini, and A. Quarteroni. Modular vs non-modular preconditioners for fluid-structure systems with large added-mass effect. *Computer Methods in Applied Mechanics and Engineering*, 197:4216–4232, 2008.

[10] J. Baiges and C. Bayona. Refficientlib: An efficient load-rebalanced adaptive mesh refinement algorithm for high performance computational physics meshes. *Submitted*, 2015.

[11] J. Baiges and R. Codina. The Fixed Mesh-ALE approach applied to solid mechanics and fluid-structure interaction problems. *International Journal for Numerical Methods in Engineering*, 81:1529–1557, 2009.

[12] J. Baiges, R. Codina, and H. Coppola-Owen. The Fixed-Mesh ALE approach for the numerical simulation of floating solids. *International Journal for Numerical Methods in Fluids*, 67(8):1004–1023, 2011.

[13] J. Baiges, R. Codina, F. Henke, S. Shahmiri, and W. A. Wall. A symmetric method for weakly imposing dirichlet boundary conditions in embedded finite element meshes. *International Journal for Numerical Methods in Engineering*, 90(5):636–658, 2012.

[14] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, and H. Zhang. PETSc Web page. `http://www.mcs.anl.gov/petsc`, 2015.

[15] H. Barbosa and T. Hughes. The finite element method with Lagrangian multipliers on the boundary: circumventing the Babuška-Brezzi condition. *Computer Methods in Applied Mechanics and Engineering*, 85:109–128, 1991.

[16] G. R. Barrenechea and F. Chouly. A local projection stabilized method for fictitious domains. *Applied Mathematics Letters*, 25(12):2071 – 2076, 2012.

[17] T. Belytschko, W. Liu, and B. Moran. *Nonlinear finite elements for continua and structures.* John Wiley, 2001.

[18] F. Brezzi and M. Fortin. *Mixed and hybrid finite element methods*. Springer Verlag, 1991.

[19] E. Burman. Ghost penalty. *Comptes Rendus Mathematique*, 348(21):1217–1220, 2010.

[20] E. Burman. Projection stabilization of lagrange multipliers for the imposition of constraints on interfaces and boundaries. *Numerical Methods for Partial Differential Equations*, 30(2):567–592, 2014.

[21] E. Burman and M. Fernandez. Stabilization of explicit coupling in fluid-structure interaction involving fluid incompressibility. *Computer Methods in Applied Mechanics and Engineering*, 198:766–784, 2009.

[22] E. Burman and M. A. FernÃ¡ndez. An unfitted nitsche method for incompressible fluid-structure interaction using overlapping meshes. *Computer Methods in Applied Mechanics and Engineering*, 279:497 – 514, 2014.

[23] E. Burman and P. Hansbo. Fictitious domain finite element methods using cut elements: Ii. a stabilized nitsche method. *Applied Numerical Mathematics*, 62(4):328–341, 2012.

[24] E. Burman and P. Hansbo. Fictitious domain methods using cut elements: Iii. a stabilized nitsche method for stokes' problem. *ESAIM: Mathematical Modelling and Numerical Analysis*, 48(3):859–874, 2014.

[25] C. Canuto, M. Y. Hussaini, A. M. Quarteroni, A. Thomas Jr, et al. *Spectral methods in fluid dynamics*. Springer Science & Business Media, 2012.

[26] E. Castillo, J. Baiges, and R. Codina. Approximation of the two-fluid flow problem for viscoelastic fluids using the level set method and pressure enriched finite element shape functions. *Journal of Non-Newtonian Fluid Mechanics*, 225:37 – 53, 2015.

[27] A. Chorin. The numerical solution of the Navier–Stokes equations for an incompressible fluid. AEC Research and Development Report, NYO-1480-82. New York University, New York, 1967.

[28] R. Codina. A stabilized finite element method for generalized stationary incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 190:2681–2706, 2001.

[29] R. Codina. Stabilized finite element approximation of transient incompressible flows using orthogonal subscales. *Computer Methods in Applied Mechanics and Engineering*, 191:4295–4321, 2002.

[30] R. Codina. Analysis of a stabilized finite element approximation of the Oseen equations using orthogonal subscales. *Applied Numerical Mathematics*, 58:264–283, 2008.

[31] R. Codina and S. Badia. On some pressure segregation methods of fractional-step type for the finite element approximation of incompressible flow problems. *Computer Methods in Applied Mechanics and Engineering*, 195:2900–2918, 2006.

[32] R. Codina and J. Baiges. Finite element approximation of transmission conditions in fluids and solids introducing boundary subgrid scales. *International Journal for Numerical Methods in Engineering*, 87:386–411, 2011.

[33] R. Codina and J. Baiges. Weak imposition of essential boundary conditions in the finite element approximation of elliptic problems with non-matching meshes. *International Journal for Numerical Methods in Engineering*, 2014.

[34] R. Codina, G. Houzeaux, H. Coppola-Owen, and J. Baiges. The fixed-mesh ALE approach for the numerical approximation of flows in moving domains. *J. Comput. Phys.*, 228:1591–1611, 2009.

[35] H. Coppola-Owen and R. Codina. A finite element model for free surface flows on fixed meshes. *Int. J. Num. Meth. Fluids*, 54:1151–1171, 2007.

[36] R. Courant and K. Friedrichs. *Supersonic Flow and Shock Waves*. Springer–Verlag, 1948.

[37] P. Díez and A. Huerta. A unified approach to remeshing strategies for finite element h-adaptivity. *Computer Methods in Applied Mechanics and Engineering*, 176:215–229, 1999.

[38] J. Donea, A. Huerta, J.-P. Ponthot, and A. Rodriguez-Ferran. Encyclopedia of computational mechanics vol. 1: Fundamentals., chapter 14: Arbitrary lagrangian-eulerian methods, 2004.

[39] A. Ern and J.-L. Guermond. *Theory and Practice of Finite Element*. Springer-Verlag, 2004.

[40] O. M. Faltinsen, O. F. Rognebakke, I. A. Lukovsky, and A. N. Timokha. Multidimensional modal analysis of nonlinear sloshing in a rectangular tank with finite water depth. *Journal of Fluid Mechanics*, 407:201–234, 2000.

[41] C. Forster, W. Wall, and E. Ramm. Artificial added mass instabilities in sequential staggered coupling of nonlinear structures and incompressible viscous flows. *Computer Methods in Applied Mechanics and Engineering*, 196:1278–1293, 2007.

[42] A. Gerstenberger and W. Wall. An embedded Dirichlet formulation for 3D continua. *International Journal for Numerical Methods in Engineering*, 82:537–563, 2010.

[43] R. Glowinski, T. Pan, T. Hesla, D. Joseph, and J. Périaux. A distributed Lagrange multiplier/-fictitious domain method for flows around moving rigid bodies: application to particulate flow. *Int. J. Num. Meth. Fluids*, 30:1043–1066, 1999.

[44] P. Hansbo and M. Larson. Discontinuous Galerkin methods for incompressible and nearly incompressible elasticity by Nitsche's method. *Computer Methods in Applied Mechanics and Engineering*, 191:1895–1908, 2002.

[45] G. Houzeaux and R. Codina. A finite element model for the simulation of lost foam casting. *Int. J. Num. Meth. Fluids*, 46:203–226, 2004.

[46] T. Hughes, G. Feijóo, L. Mazzei, and J. Quincy. The variational multiscale method–a paradigm for computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 166:3–24, 1998.

[47] M. Juntunen and R. Stenberg. Nitsche's method for general boundary conditions. Helsinki University of Technology, Institute of Mathematics, Research Reports A530, 2007.

[48] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *International Conference on Parallel Processing*, pages 113–122, 1995.

[49] S. Kollmannsberger, A. Özcan, J. Baiges, M. Ruess, E. Rank, and A. Reali. Parameter-free, weak imposition of dirichlet boundary conditions and coupling of trimmed and non-conforming patches. *International Journal for Numerical Methods in Engineering*, 101(9):670–699, 2015.

[50] M.-C. Lai and C. Peskin. An immersed boundary method with formal second-order accuracy and reduced numerical viscosity. *J. Comput. Phys.*, 160:705–719, 2000.

[51] A. Larese, R. Rossi, and E. Oñate. Finite element modeling of free surface flow in variable porosity media. *Archives of Computational Methods in Engineering*, 22(4):637–653, 2015.

[52] R. Löhner, J. Cebral, F. Camelli, J. Baum, and E. Mestreau. Adaptive embedded/immersed unstructured grid techniques. *Archives of Computational Methods in Engineering*, 14:279–301, 2007.

[53] E. Marchi. On the free overfall. *Journal of Hydraulic Research*, 31(6):777–790, 1993.

[54] E. Mestreau, R. Lohner, and S. Aita. TGV tunnel entry simulations using a finite element code with automatic remeshing, 1993. AIAA Paper 93-0890.

[55] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annual Review of Fluid Mechanics*, 37:239–261, 2005.

[56] V. P. Nguyen, T. Rabczuk, S. Bordas, and M. Duflot. Meshless methods: a review and computer implementation aspects. *Mathematics and computers in simulation*, 79(3):763–813, 2008.

[57] J. Nitsche. Über ein Varationsprinzip zu Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind. *Abhandlungen aus dem Mathematisches Seminar der Universität*, 36:9–15, 1971.

[58] E. Oñate, S. R. Idelsohn, F. Del Pin, and R. Aubry. The particle finite element method-an overview. *International Journal of Computational Methods*, 1(02):267–307, 2004.

[59] C. Peskin. Flow patterns around heart valves: A numerical method. *J. Comput. Phys.*, 10:252–271, 1972.

[60] C. Samaniego, G. Houzeaux, E. Samaniego, and M. Vázquez. Parallel embedded boundary methods for fluid and rigid-body interaction. *Computer Methods in Applied Mechanics and Engineering*, 290:387 – 419, 2015.

[61] B. Schott and W. Wall. A new face-oriented stabilized xfem approach for 2d and 3d incompressible navier–stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 276:233–265, 2014.

[62] S. Turek and J. Hron. Proposal for numerical benchmarking of fluid–structure interaction between an elastic object and laminar incompressible flow. In H. J. Bungartz and M. Schäfer, editors, *Fluid–Structure Interaction – Modelling, Simulation, Optimization*, number 53 in Lecture Notes in Computational Science and Engineering, pages 371–385. Springer, Berlin, 2006. ISBN 3-540-34595-7.