AN EMBEDDED STRATEGY FOR LARGE SCALE INCOMPRESSIBLE FLOW SIMULATIONS IN MOVING DOMAINS

RAMON CODINA[1,2], JOAN BAIGES[1,2], INOCENCIO CASTAÑAR[1],
IGNACIO MARTÍNEZ-SUÁREZ[2], LAURA MORENO[2] AND SAMUEL PARADA[1]

Abstract. In this work we describe a methodology to approximate the incompressible Navier-Stokes equations in time dependent domains. To deal with the motion of the domain, we employ a fixed mesh method that we call fixed-mesh ALE. It consists of writing the equations in a moving ALE reference system but then projecting them onto a fixed background mesh. This implies that the boundaries of the elements do not necessarily coincide with the physical boundaries, and thus there is the possibility of badly cut elements. We use a Nitsche's type formulation to prescribe the boundary conditions and stabilise the bad cuts by introducing a term that penalises the gradient of the unknown orthogonal to the finite element space in a patch that contains the badly cut element. The flow formulation is a stabilised finite element method that allows one to treat convection dominated flows and to use equal velocity–pressure interpolation. Furthermore, this formulation can be shown to behave as an implicit large eddy simulation approach. A key issue is that the sub-grid scales on which the formulation depends are allowed to be time dependent; this fact has proved to be crucial for the robustness of the approach. The calculation of the velocity and the pressure is segregated by using a fractional step scheme designed at the pure algebraic level, and the conditioning of the pressure equation is improved by using an artificial compressibility technique. Finally, an adaptive mesh refinement strategy is described. All the algorithms are implemented in a parallel environment. The strategy described can be applied to complex flow problems, and in particular here we show the simulation of the air flow generated by a train moving inside a tunnel.

Keywords: Embedded methods; Fixed-mesh ALE; Small cut instability; Stabilised finite element methods; Fractional step schemes

1. Introduction

The finite element (FE) simulation of incompressible flows in moving domains involves several difficulties. The objective of this paper is to explain which is the methodology we employ to tackle them. Most of the techniques to be described are well known or proposed earlier by our group, and thus our contribution in the sense is merely their combination and adaption to the problem at hand. However, some other ingredients, such as the stabilisation of cut elements or the tailoring of fractional step schemes to the problem we consider, are original contributions. All these ingredients, known or new, constitute our proposal to deal with flows in time dependent domains.

We are interested in problems in which the computational domain changes very significantly with time. In particular, we are motivated by the problem of simulating the air flow caused by a train moving in a tunnel, without the possibility of considering this tunnel cylindrical; that would permit to consider the train fixed and the air flowing against it, but

Date: March 1, 2023.

[1] Universitat Politècnica de Catalunya, Barcelona Tech, Jordi Girona 1-3, Edifici C1, 08034 Barcelona, Spain.

[2] Centre Internacional de Mètodes Numèrics en Enginyeria (CIMNE), Edifici C1, Campus Nord UPC, Gran Capitán S/N, 08034 Barcelona, Spain.

E-mails: ramon.codina@upc.edu (RC), joan.baiges@upc.edu (JB), icastanar@cimne.upc.edu (IC), imartinezs@cimne.upc.edu (IM), lmoreno@cimne.upc.edu (LM), sparada@cimne.upc.edu (SP) .

this is not the case we wish to consider. Thus, the type of problems we wish to study is that in which an object moves in a fluid domain. We shall see that our approach extends easily to several moving objects.

The first decision to take is which strategy to use to model moving domains. We do not consider the possibility of purely Lagrangian approaches, and this leads us to some sort of arbitrary Lagrangian-Eulerian (ALE) method [1, 2]. The way classical ALE methods proceed consists of deforming the domain boundary according to some predetermined velocity and then deform the interior points with a domain velocity field as smooth as possible. In our context, this would mean to move the boundaries of the moving object with its velocity and then obtain an extension of this velocity to the whole domain. In the FE context in which we are interested, this amounts to obtain the mesh velocity, i.e., the velocity of the nodes of the FE mesh. Inevitably, this causes mesh distortion, and brings the need for re-meshing every certain number of time steps. Since in the problem in which we are interested mesh distortion shows up very soon, we will not use a classical ALE formulation.

We will deal with the treatment of moving domains using a fixed mesh method. This means that the FE mesh employed to solve the problem will not be modified in time. Obviously, this implies that when the object moves, its boundary will not coincide with a union of element boundaries, but it will cut the elements. Therefore, what we propose is an *embedded mesh* method, and this opens several questions, such as how to take into account the motion of the object, how to prescribe boundary conditions and how to stabilise the ill-conditioning that is found when very small portions of an element belong to the fluid domain.

The way we propose to deal with the domain motion in a fixed mesh strategy is what we call *fixed-mesh ALE* formulation [3, 4]. We proposed it a few years ago, but we summarise it here for completeness. Suppose we have the flow configuration at a certain time step of a discretisation in time and we wish to move to the next time step. The idea is to write the flow equations in an ALE frame of reference, using the velocity of the moving object in its contact with the fluid and extending it somehow to the rest of the flow domain. Then, instead of solving the flow equations in the moving ALE mesh, we project them back to the fixed mesh, and solve the problem there. The implementation of this technique is briefly explained in Section 2, already considering the equations discretised in time.

Once the flow equations are written using the fixed-mesh ALE approach, we may proceed to their FE discretisation. Here we employ a strategy that we have developed over the years, consisting of a stabilised FE model based on the Variational Multi-Scale (VMS) concept [5]. The unknowns, velocity and pressure, are split into their FE component and a sub-grid scale (SGS). The latter is approximated in terms of the former, ending up with a problem in terms of the FE component only. Two particular ingredients of our approach are crucial for the methodology to be described, namely, the SGSs are considered orthogonal to the FE space and *time dependent* [6, 7]. The first feature is critical for the design of the fractional step scheme described below, whereas the second provides the scheme with much more robustness. In our applications, we have found many times absolutely necessary to consider time dependent SGSs to obtain converged solutions within each time step. The formulation we use is summarised in Section 3.

Since we employ an embedded strategy and the boundary of the flow domain in contact with the moving object is not made of element boundaries, a technique to prescribe boundary conditions needs to be chosen. Neumann-type boundary conditions offer no difficulty, as we integrate only in the flow domain and they appear as natural boundary conditions when the viscous term and the pressure gradient term tested with the velocity test function are integrated by parts. However, Dirichlet-type boundary conditions require a special treatment. Several approaches exist in the literature, such as the use of penalty

terms as in the original immersed boundary method [8, 9], the use of Lagrange multipliers [10, 11, 12, 13, 14, 15, 16, 17], which may require the use of additional unknowns accounting for the fluxes on the Dirichlet boundary, or the well-known Nitsche's method [18, 19, 20], which yields symmetric, stable variational formulations through the use of a limited penalty term whose value needs to be estimated. This last option, sometimes termed as a method to enforce essential conditions in a weak manner, is the one that we will use in this work. Its implementation in conjunction with the stabilised FE method employed is described in Section 4. A critical issue regarding the weak imposition of Dirichlet boundary conditions is the ill-conditioning that results when the elements that are cut by the boundary (in our case the boundary of the moving body) have a very small portion in the fluid domain; these are what are called badly cut elements. In this case, little errors in the prescription of the boundary values produce significant errors in the solution. There exist several methods designed to suppress or alleviate this instability [20, 21, 22]. Here we propose a novel technique based on penalising the difference between the gradient of the velocity and this gradient projected onto the FE space. This method is also explained in Section 4.

The next ingredient of the proposed methodology is the fractional step scheme described in Section 5 designed at the pure algebraic level [23], together with the artificial compressibility method used to improve the conditioning of the pressure equation [24, 25]. It is a classical second order pressure correction scheme that allows one to uncouple the calculation of the velocity and the pressure degrees of freedom, being this much more efficient than a monolithic coupling. This scheme is tailored here to account for its application to the fixed-mesh ALE method with a weak imposition of Dirichlet boundary conditions and with a stabilisation term to account for badly cut elements.

The last ingredient of the overall formulation is an adaptive mesh refinement. Mesh refinement has been extensively used in the literature to diminish computational cost and concentrate effort in the regions of the domain where it is necessary. Multiple strategies exist such as the moving interface meshes presented in [26, 27, 28], and remeshing strategies such as the one presented in [29]. The strategy we use in the present in this paper is based on hierarchical element subdivision [30]. It is based on an octree strategy, splitting the elements in which accuracy needs to be improved in a nested way. This introduces the need for handling hanging nodes but at the same time allows to easily treat interpolation between meshes and parallel implementation. We explain this scheme and how elements cut by the boundary need to be treated. These issues are detailed in Section 6.

The adaptive mesh refinement concludes the design of our methodology. In Section 7 we present the numerical simulation of a train circulating through a tunnel. Several remarks concerning the mesh refinement employed and the parallel implementation of the scheme and its application to the problem being solved are also included.

Conclusions close the paper in Section 8.

2. The Fixed-mesh ALE method

2.1. **ALE formulation of the flow equations.** Let us consider a region $\Omega^0 \subset \mathbb{R}^d$ ($d = 2, 3$) where a flow will take place during a time interval $[0, T]$. However, we consider the case in which the fluid at time t occupies only a subdomain $\Omega(t) \subset \Omega^0$ (note in particular that $\Omega(0) \subset \Omega^0$). Suppose also that the boundary of $\Omega(t)$ is defined by part of $\partial\Omega^0$ and a moving boundary that we call $\Gamma_{\rm mov}(t) = \partial\Omega(t) \setminus \partial\Omega^0 \cap \partial\Omega(t)$. This moving part of $\partial\Omega(t)$ may correspond to the boundary of a moving solid immersed in the fluid. This boundary does not need to be connected, as it is the case if several solids move in the fluid. We call $\Gamma_{\rm fix}$ the fixed part of $\partial\Omega(t)$, i.e., $\Gamma_{\rm fix} = \partial\Omega(t) \setminus \Gamma_{\rm mov}(t)$ (see Fig. 1).

In order to cope with the time-dependency of $\Omega(t)$, we may use the ALE approach. Let $\boldsymbol{\chi}_t$ be a family of invertible mappings, which for all $t \in [0, T]$ map a point $\boldsymbol{X} \in \Omega(0)$ to

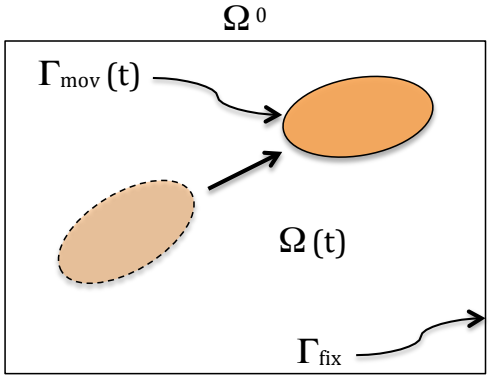


FIGURE 1. Domain setting

a point $\boldsymbol{x} = \boldsymbol{\chi}_t(\boldsymbol{X}) \in \Omega(t)$, with $\boldsymbol{\chi}_0 = \boldsymbol{I}$, the identity. If $\boldsymbol{\chi}_t$ is given by the motion of the particles, the resulting formulation would be Lagrangian, whereas if $\boldsymbol{\chi}_t = \boldsymbol{I}$ for all t, $\Omega(t) = \Omega(0)$ and the formulation would be Eulerian.

Let now $t' \in [0, T]$, with $t' \leq t$, and consider the mapping

$$\begin{aligned} \boldsymbol{\chi}_{t,t'} : \Omega(t') &\longrightarrow \Omega(t) \\ \boldsymbol{x}' &\mapsto \boldsymbol{x} = \boldsymbol{\chi}_t \circ \boldsymbol{\chi}_{t'}^{-1}(\boldsymbol{x}'). \end{aligned}$$

Let $\mathfrak{D} = \{(\boldsymbol{x}, t) \mid \boldsymbol{x} \in \Omega(t),\ 0 < t < T\}$ be the space-time domain where the problem is defined. Given a function $f : \mathfrak{D} \longrightarrow \mathbb{R}$ we define

$$\left.\frac{\partial f}{\partial t}\right|_{\boldsymbol{x}'} (\boldsymbol{x}, t) := \frac{\partial (f \circ \boldsymbol{\chi}_{t,t'})}{\partial t}(\boldsymbol{x}', t), \quad \boldsymbol{x} \in \Omega(t),\ \boldsymbol{x}' \in \Omega(t').$$

In particular, the domain velocity taking as a reference the coordinates of $\Omega(t')$ is given by

$$\boldsymbol{u}_{\text{dom}} := \left.\frac{\partial \boldsymbol{x}}{\partial t}\right|_{\boldsymbol{x}'} (\boldsymbol{x}, t). \tag{1}$$

The incompressible Navier-Stokes formulated in $\Omega(t)$, accounting also for the motion of this domain, can be written as follows: find a velocity $\boldsymbol{u} : \mathfrak{D} \longrightarrow \mathbb{R}^d$ and a pressure $p : \mathfrak{D} \longrightarrow \mathbb{R}$ such that

$$\rho \left[\left.\frac{\partial \boldsymbol{u}}{\partial t}\right|_{\boldsymbol{x}'} (\boldsymbol{x}, t) + (\boldsymbol{u} - \boldsymbol{u}_{\text{dom}}) \cdot \nabla \boldsymbol{u} \right] - \nabla \cdot (2\mu \nabla^S \boldsymbol{u}) + \nabla p = \rho \boldsymbol{f}, \tag{2}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{3}$$

where $\nabla^S \boldsymbol{u}$ is the symmetrical part of the velocity gradient, ρ is the fluid density, μ is the viscosity and $\rho \boldsymbol{f}$ is the vector of body forces.

Initial and boundary conditions have to be appended to problem (2)-(3). The boundary conditions on $\Gamma_{\text{mov}}(t)$ can be of two different types: a) p (or the normal stress) given, $\boldsymbol{u}$ unknown on Γ_{mov}; b) $\boldsymbol{u}$ given, p (or the normal stress) unknown on Γ_{mov}. The first option would correspond to an object whose motion is determined by the flow dynamics, whereas the second would apply in the case of an object moving with a given velocity. This is the situation we consider in what follows, i.e., Γ_{mov} will be part of the boundary where velocities are prescribed. On the rest of the boundary Γ_{fix} the usual boundary conditions can be considered. In general, we consider these boundary conditions of the form

$$\begin{aligned} \boldsymbol{u} &= \bar{\boldsymbol{u}} \quad \text{on } \Gamma_D, \\ \boldsymbol{n} \cdot \boldsymbol{\sigma} &= \bar{\boldsymbol{t}} \quad \text{on } \Gamma_N, \end{aligned}$$

where $\boldsymbol{n}$ is the external normal to the boundary, $\boldsymbol{\sigma} = -p\boldsymbol{I} + 2\mu\nabla^S\boldsymbol{u}$ is the Cauchy stress tensor and $\bar{\boldsymbol{u}}$ and $\bar{\boldsymbol{t}}$ are the given boundary data. The components of the boundary Γ_D and Γ_N are obviously disjoint and such that $\Gamma_D \cup \Gamma_N = \Gamma_{\rm fix}$. The Dirichlet part of the boundary will be $\Gamma_D \cup \Gamma_{\rm mov}$.

2.2. **Time discretisation.** We will consider now the discretisation in time of the flow equations using a simple finite difference scheme, namely, the trapezoidal rule. Obviously, any other finite difference time discretisation could be adopted, and in fact the results in Section 7 have been obtained using the backward difference (BDF) scheme of second order (BDF2). Furthermore, both the Galerkin FE approximation and the stabilisation we will describe commute with the time discretisation, and thus we could start either discretising in time or in space.

Consider a uniform partition of $[0,T]$ into N time intervals of length δt. Let us denote by f^n the approximation of a time dependent function f at time level $t^n = n\delta t$. We will also denote

$$\begin{aligned} \delta f^{n+1} &= f^{n+1} - f^n, \\ \delta_t f^{n+1} &= \frac{f^{n+1} - f^n}{\delta t}, \\ f^{n+\theta} &= \theta f^{n+1} + (1-\theta) f^n, \quad \theta \in [1/2, 1]. \end{aligned}$$

The last notation will be used for the convex combination of any object between its value at t^n and at t^{n+1}, including spaces of functions or FE meshes.

Suppose we are given a computational domain at time t^n, with spatial coordinates labeled $\boldsymbol{x}^n$, and $\boldsymbol{u}^n$ and p^n are known in this domain. The velocity $\boldsymbol{u}^{n+1}$ and the pressure p^{n+1} can then be found as the solution to the problem

$$\rho\left[\delta_t \boldsymbol{u}^{n+1}\big|_{\boldsymbol{x}^n} + (\boldsymbol{u}^{n+\theta} - \boldsymbol{u}^{n+\theta}_{\rm dom})\cdot\nabla\boldsymbol{u}^{n+\theta}\right] - \nabla\cdot(2\mu\nabla^S\boldsymbol{u}^{n+\theta}) + \nabla p^{n+\theta} = \rho\boldsymbol{f}^{n+\theta}, \tag{4}$$

$$\nabla\cdot\boldsymbol{u}^{n+\theta} = 0, \tag{5}$$

where now $\delta_t\boldsymbol{u}^{n+1}\big|_{\boldsymbol{x}^n} = (\boldsymbol{u}^{n+1}(\boldsymbol{x}) - \boldsymbol{u}^n(\boldsymbol{x}^n))/\delta t$, being $\boldsymbol{x} = \boldsymbol{\chi}_{t^{n+\theta},t^n}(\boldsymbol{x}^n)$ the spatial coordinates in $\Omega(t^{n+\theta})$. The domain velocity given by (1), with $\boldsymbol{x}' = \boldsymbol{x}^n$, is approximated as

$$\boldsymbol{u}^{n+\theta}_{\rm dom} = \frac{1}{\theta\delta t}\left(\boldsymbol{\chi}_{t^{n+\theta},t^n}(\boldsymbol{x}^n) - \boldsymbol{x}^n\right). \tag{6}$$

2.3. **Galerkin finite element approximation.** The next step is to consider the spatial discretisation of problem (4)-(5). Even though we use a stabilised FE formulation, let us start stating the standard Galerkin approximation of the problem. This will allow us to describe the fixed-mesh ALE method we employ. The terms arising from the VMS formulation that provide enhanced stability will be described later on.

Given the flow domain $\Omega(t^n)$, $n = 0, 1, \ldots, N$, let V^n and Q^n be the spaces where the velocity and the pressure must belong at time t^n, i.e., $\boldsymbol{u}^n \in V^n$ and $p^n \in Q^n$. Space V^n is the subspace of $H^1(\Omega(t^n))^d$ of vector functions satisfying the Dirichlet conditions, whereas the space of test functions (which vanish where the velocity is prescribed) will be denoted as V_0^n. The pressure space is $Q^n = L^2(\Omega(t^n))$, and this space modulo constants if all boundary conditions are of Dirichlet type for the velocity.

Let $\mathcal{T}_h^n = \{K^n\}$ be a FE partition of the domain $\Omega(t^n)$, $n = 0, 1, \ldots, N$. The number of subdomains may in principle change from one time instant to the other, although the fixed-mesh ALE approach to be described will make use only of the partition $\mathcal{T}_h$ of Ω^0. Note that this partition in general does not coincide with $\mathcal{T}_h^0$, the FE partition of $\Omega(0)$. Note also that the diameter of the FE partition h may depend on n, although we have not introduced any additional superscript to avoid overloading the notation. All FE functions

will be identified by the subscript h. To avoid dealing with the error arising from the geometrical representation of $\Omega(t)$, we will assume it is always polyhedral.

From $\mathcal{T}_h^n$ we may construct FE spaces for the velocity and the pressure, V_h^n and Q_h^n, respectively. Only conforming approximations will be considered, so that $V_h^n \subset V^n$ and $Q_h^n \subset Q^h$. For the test functions, whose spaces for velocity and pressure will respectively be $V_{0,h}^n$ and Q_h^n, we will not use any superscript indicating the time level, being understood that it is the same as for the velocity and pressure unknowns.

The Galerkin FE approximation of the problem is nothing but the weak form of the problem restricting both unknowns and test functions to their corresponding FE spaces. It can now be written as follows: given $\boldsymbol{u}_h^n \in V_h^n$ and $p_h^n \in Q_h^n$, find $\boldsymbol{u}_h^{n+1} \in V_h^{n+1}$ and $p_h^{n+1} \in Q_h^{n+1}$ such that

$$
\begin{aligned}
&m^{n+\theta}(\boldsymbol{v}_h, \delta_t \boldsymbol{u}_h^{n+1}\big|_{\boldsymbol{x}^n}) + a^{n+\theta}(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+\theta}) \\
&\qquad + c^{n+\theta}(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+\theta} - \boldsymbol{u}_{\mathrm{dom}}^{n+\theta}, \boldsymbol{u}_h^{n+\theta}) - b^{n+\theta}(p_h^{n+\theta}, \boldsymbol{v}_h) = l^{n+\theta}(\boldsymbol{v}_h), \qquad (7)\\
&b^{n+\theta}(q_h, \boldsymbol{u}_h^{n+\theta}) = 0, \qquad (8)
\end{aligned}
$$

for all test functions $\boldsymbol{v}_h \in V_{0,h}^{n+1}$ and $q_h \in Q_h^{n+1}$, where

$$
\begin{aligned}
m^{n+\theta}(\boldsymbol{v}, \boldsymbol{u}) &= \int_{\Omega(t^{n+\theta})} \boldsymbol{v} \cdot \rho\, \boldsymbol{u},\\
a^{n+\theta}(\boldsymbol{v}, \boldsymbol{u}) &= \int_{\Omega(t^{n+\theta})} 2\nabla^S \boldsymbol{v} : \mu \nabla^S \boldsymbol{u},\\
c^{n+\theta}(\boldsymbol{v}, \boldsymbol{w}, \boldsymbol{u}) &= \int_{\Omega(t^{n+\theta})} \boldsymbol{v} \cdot [\rho\, \boldsymbol{w} \cdot \nabla \boldsymbol{u}],\\
b^{n+\theta}(p, \boldsymbol{v}) &= \int_{\Omega(t^{n+\theta})} p \nabla \cdot \boldsymbol{v},\\
l^{n+\theta}(\boldsymbol{v}) &= \int_{\Omega(t^{n+\theta})} \boldsymbol{v} \cdot \rho \boldsymbol{f}^{n+\theta} + \int_{\Gamma_N} \boldsymbol{v} \cdot \bar{\boldsymbol{t}}^{n+\theta}.
\end{aligned}
$$

Recall that Γ_{mov} has been considered part of the Dirichlet boundary.

It is observed that all terms in Eqs. (7)-(8) are evaluated at $\Omega(t^{n+\theta})$. If $\theta = 1/2$ this guarantees satisfaction of the so called geometric conservation law (GCL) (see, e.g. [31, 32]). This ensures that the time integration scheme is stable if the underlying finite difference approximation is also stable. In our case, we have considered the trapezoidal rule, and therefore that the different terms of the equation are also evaluated at $t^{n+\theta}$, yielding an unconditionally stable scheme for $\theta \in [1/2, 1]$. In principle, if the variational equations are evaluated at any other time instant between t^n and t^{n+1}, the GCL is not satisfied and the resulting discrete problem is only conditionally stable. Nevertheless, we often consider the variational equations evaluated at t^{n+1} without finding in the numerical experiments any time step limitation. This is particularly useful if a BDF scheme in time is adopted, with all the terms in the differential equation evaluated at t^{n+1}. In particular, in the numerical simulation to be presented we have adopted a second order BDF scheme and evaluated the variational equations at t^{n+1}, without observing any time step limitation. This scheme is analysed for the linear convection-diffusion equation in [2].

2.4. **Fixed-mesh ALE implementation.** The key point of the fixed-mesh ALE method is that, even if we are using a fixed-mesh method, domain movement cannot be obviated, and it is necessary to take it into account with an ALE method. However, since we are interested in using a fixed mesh, we develop a strategy which allows us to always work with the background fixed mesh and, at the same time, includes the ALE terms which appear due to the domain movement. We have developed this strategy in [3, 4, 33, 34].

Suppose Ω^0 is meshed with a FE mesh $\mathcal{T}_h$ and that at time level t^n the domain $\Omega(t^n)$ is meshed with a FE mesh $\mathcal{T}_h^n$ (as we will see, close to $\mathcal{T}_h$). Let $\boldsymbol{u}_h^n$ be the velocity already computed on $\Omega(t^n)$. The purpose is to obtain the fluid region $\Omega(t^{n+\theta})$ and the velocity field $\boldsymbol{u}_h^{n+\theta}$. The former may move according to prescribed kinematics, for example due to the motion of a solid as we have considered, or can be an unknown of the problem. If the classical ALE method is used, $\mathcal{T}_h^n$ would deform to another mesh defined at $t^{n+\theta}$. The key idea is not to use this mesh to compute $\boldsymbol{u}_h^{n+\theta}$ and $p_h^{n+\theta}$, but to re-mesh in such a way that the new mesh is, essentially, $\mathcal{T}_h$ once again. Since $\boldsymbol{u}_h^{n+1}$ and p_h^{n+1} may be directly computed from $\boldsymbol{u}_h^{n+\theta}$ and $p_h^{n+\theta}$, respectively, we will take $\theta = 1$ in the following description of the method.

The steps of the algorithm to achieve the goal described are schematically represented in Fig. 2 and are the following:

(1) Update the position of the moving boundary to $\Gamma_{\text{mov}}(t^{n+1})$. In our problem, the motion of $\Gamma_{\text{mov}}(t)$ is directly given by the velocity of a moving object, assumed to be prescribed.
(2) Deform *virtually* the mesh $\mathcal{T}_h^n$ to $\mathcal{T}_{h,\text{virt}}^{n+1}$ using the classical ALE concepts and compute the mesh velocity $\boldsymbol{u}_{\text{dom}}^{n+1}$. Here we have introduced the term *virtually* because in fact mesh $\mathcal{T}_{h,\text{virt}}^{n+1}$ will never be used, as it follows from the next steps.
(3) Write down the ALE Navier-Stokes equations on $\mathcal{T}_{h,\text{virt}}^{n+1}$. This exactly corresponds to Eqs. (7)-(8) (for $\theta = 1$) considering that the FE partition of $\Omega(t^{n+1})$ is $\mathcal{T}_{h,\text{virt}}^{n+1}$.
(4) *Split the elements* of $\mathcal{T}_h$ cut by $\Gamma_{\text{mov}}(t^{n+1})$ to define a mesh on $\Omega(t^{n+1})$, $\mathcal{T}_h^{n+1}$.
(5) *Project* the ALE Navier-Stokes equations from $\mathcal{T}_{h,\text{virt}}^{n+1}$ to $\mathcal{T}_h^{n+1}$. This means that we again consider Eqs. (7)-(8), but now with $\Omega(t^{n+1})$ meshed with $\mathcal{T}_h^{n+1}$ instead of being meshed with $\mathcal{T}_{h,\text{virt}}^{n+1}$. The unknowns $\boldsymbol{u}_h^{n+1}$ and p_h^{n+1} can be computed with the nodal values of $\mathcal{T}_h^{n+1}$, and thus they do not require any additional operation. However, Eq. (7) involves $\boldsymbol{u}_h^n$ and $\boldsymbol{u}_{\text{dom}}^{n+1}$, and both fields are known in $\mathcal{T}_{h,\text{virt}}^{n+1}$, i.e., computed with the nodal values in this virtual mesh. Therefore, the only new operation required by this algorithm consists in *computing the nodal values of* $\boldsymbol{u}_h^n$ *and* $\boldsymbol{u}_{\text{dom}}^{n+1}$ *in* $\mathcal{T}_h^{n+1}$. This amounts to a search operation: one has to determine the position in the elements of $\mathcal{T}_h^{n+1}$ of the nodes of $\mathcal{T}_{h,\text{virt}}^{n+1}$. However, since the change from $\mathcal{T}_h^n$ to $\mathcal{T}_h^{n+1}$ can be considered small, in practice one only needs to move the nodes of the elements in contact with Γ_{mov} (or perhaps one additional layer of elements) in step (2). Thus, the search operation is inexpensive from the computational standpoint. Note that this step corresponds to a classical re-meshing of an ALE implementation when the elements get too distorted, now with the peculiarity that this re-meshing needs to be performed at each time step but it is restricted to the elements in contact with Γ_{mov}.
(6) Solve the equations on $\mathcal{T}_h^{n+1}$ to compute $\boldsymbol{u}_h^{n+1}$ and p_h^{n+1}.

Note that at each time step two sets of nodes have to be appropriately dealt with, namely, the so called newly created nodes (nodes that are 'dry' in one time step and 'wet' in the following) and the boundary nodes. Contrary to other fixed grid methods, newly created nodes are treated in a completely natural way using the fixed-mesh ALE approach: the value of the velocity there is directly given by the projection step from $\mathcal{T}_{h,\text{virt}}^{n+1}$ to $\mathcal{T}_h^{n+1}$. Boundary nodes require either additional unknowns with respect to those of mesh $\mathcal{T}_h$ or an appropriate imposition of boundary conditions, as it will be explained in Section 4.

Let us also note that the splitting of the elements is mainly needed for integration purposes. Once the cut elements are split into subelements, one can use the same numerical integration rule there than for the rest of uncut elements (see, e.g., [4]).

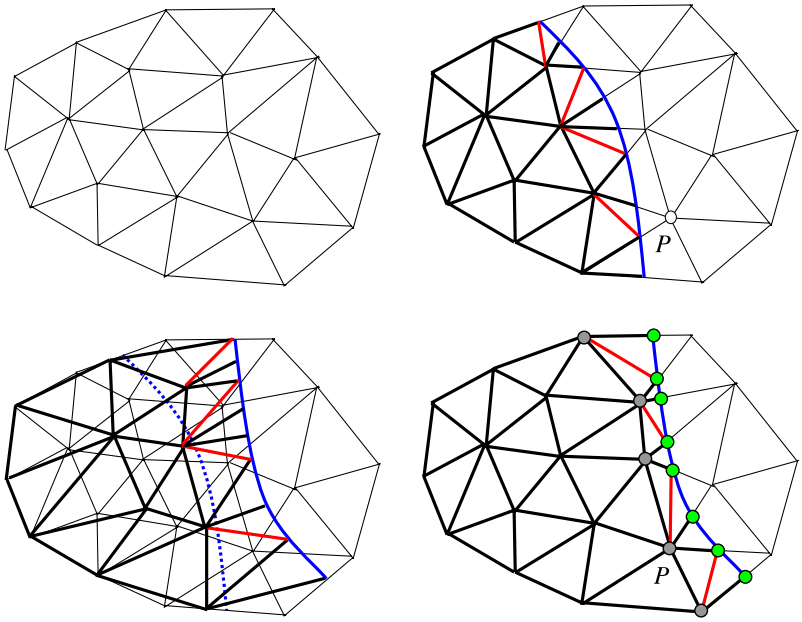


FIGURE 2. Two dimensional FM-ALE schematic. Top-left: original finite element mesh $\mathcal{T}_h$ of Ω^0. Top-right: finite element mesh $\mathcal{T}_h^n$ of $\Omega(t^n)$, with the elements represented by a thick line and the elements of $\mathcal{T}_h$ represented by thin line. The blue line represents $\Gamma_{\rm mov}(t^n)$ and the red edges indicate the splitting of $\mathcal{T}_h$ to obtain $\mathcal{T}_h^n$. Bottom-left: updating of $\mathcal{T}_h^n$ to $\mathcal{T}_{h,\rm virt}^{n+1}$ using the classical ALE strategy. The position of $\Gamma_{\rm mov}(t^{n+1})$ is again shown using a solid blue line and the previous position $\Gamma_{\rm mov}(t^n)$ using a dotted blue line. Bottom-right: Mesh $\mathcal{T}_h^{n+1}$ of $\Omega(t^{n+1})$, represented by a thick line. The edges that split elements of $\mathcal{T}_h$ are again indicated in red. Boundary nodes, where approximate boundary conditions need to be imposed, are drawn in green, whereas newly created nodes are drawn in grey.

3. Stabilised finite element approximation

The Galerkin FE approximation given by Eqs. (7)-(8) may be unstable because of two reasons. First, because the FE spaces for the velocity and the pressure do not satisfy the appropriate inf-sup condition and, second, because viscosity is small and convection effects dominate viscous ones. Both instabilities can be overcome by resorting to the stabilised FE formulation described next, which was proposed in [35] and is based on the VMS approach introduced in [5]. Details can be found in [36]. Here we will just write the resulting numerical formulation.

The starting point of the formulation is the introduction of a SGS component of the velocity and the pressure. The two main features of our approach are that we allow these SGSs to be time dependent, and that we seek them in the space orthogonal to the corresponding FE space in the L^2-sense. If $\tilde{\boldsymbol{u}}$ is the SGS component of the velocity and $\tilde{p}$ that of the pressure, the problem to be solved at each time step instead of (7)-(8) is the following: given $\boldsymbol{u}_h^n \in V_h^n$, $\tilde{\boldsymbol{u}}^n \in V_h^{n,\perp}$ (space orthogonal to V_h^n in the L^2-sense) and $p_h^n \in Q_h^n$, find $\boldsymbol{u}_h^{n+1} \in V_h^{n+1}$ and $p_h^{n+1} \in Q_h^{n+1}$ such that

$$
\begin{aligned}
&m^{n+\theta}(\boldsymbol{v}_h, \delta_t \boldsymbol{u}_h^{n+1}\big|_{\boldsymbol{x}^n}) + a^{n+\theta}(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+\theta}) \\
&\quad + c^{n+\theta}(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+\theta} - \boldsymbol{u}_{\rm dom}^{n+\theta}, \boldsymbol{u}_h^{n+\theta}) - b^{n+\theta}(p_h^{n+\theta}, \boldsymbol{v}_h)
\end{aligned}
$$

$$
\begin{aligned}
&- \sum_{K^{n+\theta}} \int_{K^{n+\theta}} \tilde{\boldsymbol{u}}^{n+\theta} \cdot \Big[\rho(\boldsymbol{u}_h^{n+\theta} - \boldsymbol{u}_{\rm dom}^{n+\theta}) \cdot \nabla \boldsymbol{v}_h + \nabla \cdot (2\mu \nabla^S \boldsymbol{v}_h) \Big] \\
&- \sum_{K^{n+\theta}} \int_{K^{n+\theta}} \tilde{p}^{n+\theta} \nabla \cdot \boldsymbol{v}_h = l^{n+\theta}(\boldsymbol{v}_h), \qquad (9) \\
& b^{n+\theta}(q_h, \boldsymbol{u}_h^{n+\theta}) - \sum_{K^{n+\theta}} \int_{K^{n+\theta}} \tilde{\boldsymbol{u}}^{n+\theta} \cdot \nabla q_h = 0, \qquad (10)
\end{aligned}
$$

for all test functions $\boldsymbol{v}_h \in V_{0,h}^{n+1}$ and $q_h \in Q_h^{n+1}$, where $\tilde{\boldsymbol{u}}^{n+\theta}$ and $\tilde{p}^{n+\theta}$ are obtained from

$$
\rho \delta_t \tilde{\boldsymbol{u}}^{n+1}\big|_{\boldsymbol{x}^n} + (\tau_u^{n+\theta})^{-1} \tilde{\boldsymbol{u}}^{n+\theta} = P_u^{\perp}[\boldsymbol{r}_h^{n+\theta}], \qquad (11)
$$

$$
\boldsymbol{r}_h^{n+\theta} := \rho \boldsymbol{f}^{n+\theta} - \rho\, \delta_t \boldsymbol{u}_h^{n+1}\big|_{\boldsymbol{x}^n} - \rho(\boldsymbol{u}_h^{n+\theta} - \boldsymbol{u}_{\rm dom}^{n+\theta}) \cdot \nabla \boldsymbol{u}_h^{n+\theta} + \nabla \cdot (2\mu \nabla^S \boldsymbol{u}_h^{n+\theta}) - \nabla p_h^{n+\theta},
$$

$$
\tilde{p}^{n+\theta} = -\tau_p^{n+\theta} P_p^{\perp}[\nabla \cdot \boldsymbol{u}_h^{n+\theta}], \qquad (12)
$$

where τ_u and τ_p are the stabilisation parameters on which the formulation depends, computed element-wise at time level $n+\theta$ as:

$$
\tau_u^{n+\theta}|_K = \left(\frac{c_1}{k^4} \frac{\mu}{h_K^2} + \frac{c_2}{k} \frac{\rho \|\boldsymbol{u}_h^{n+\theta}\|_{L^\infty(K)}}{h_K} \right)^{-1},
$$

$$
\tau_p^{n+\theta}|_K = (\tau_u^{n+\theta}|_K)^{-1} h_K^2.
$$

Here and below, $\|\cdot\|_{L^\infty(R)}$ stands for the maximum norm in a region R.

Even though we have included $\rho \delta_t \boldsymbol{u}_h^{n+1}$ in the expression of the FE residual, this term belongs to the velocity FE space, and therefore its orthogonal projection is zero. It will not contribute to the mass matrix of the final algebraic system written below. This is particularly important, as the fractional step scheme described in Section 5 makes use of the standard mass matrix of the problem to approximate the equation for the pressure. Similarly, the body force has been included in the FE residual, but its orthogonal projection can be neglected, as it does not contribute neither to stability nor to the order of accuracy.

In the expressions above, P_u is the $L^2(\Omega)$ projection onto $V_h^{n+\theta}$, $P_u^{\perp} = I - P_u$ is the orthogonal projection onto $V_h^{n+\theta,\perp}$, P_p is the $L^2(\Omega)$ projection onto $Q_h^{n+\theta}$, and $P_p^{\perp} = I - P_p$ is the orthogonal projection onto $Q_h^{n+\theta,\perp}$. We have not used a superscript to indicate the time level of all these projections, as this is determined by their arguments. In any case, to avoid enlarging the stencil of the matrices of the final algebraic system, we usually compute the projection onto the finite element space explicitly. Thus, if f is an arbitrary unknown and P any of these projections, we approximate $P^{\perp}(f^{n,k}) \approx f^{n,k} - P(f^{n,k-1})$ or $P^{\perp}(f^{n,k}) \approx f^{n,k} - P(f^{n-1})$, where n is the time step counter and k the iteration counter of the iterative procedure employed to linearise the problem.

The stabilised formulation proposed is stable for all velocity-pressure pairs of FE spaces V_h^{n+1} and Q_h^{n+1}, as soon as the pressure interpolation is continuous. In the case of discontinuous pressure interpolations, additional terms involving the jumps of pressures between elements need to be introduced (see [37]). In the numerical example of Section 7, linear interpolations have been used for both velocity and pressure (i.e., the P_1-P_1 pair).

Since it is understood that in the fixed-mesh ALE method what is kept constant when taking the time derivative is the position $\boldsymbol{x}^n$, we shall omit this subscript in what follows.

In the expression of the stabilisation parameters, h_K is the size of element K (at $n+\theta$) and k is the order of the FE interpolation. The algorithmic constants c_1 and c_2 are independent of the mesh size and the interpolation order.

The right-hand-side (RHS) in Eq. (11) is the projection of the FE residual $\boldsymbol{r}_h^{n+\theta}$ onto the space orthogonal to the velocity FE space. From this equation we can write

$$
\tilde{\boldsymbol{u}}^{n+\theta} = \tau_{\rm eff}^{n+\theta} \rho (\theta \delta t)^{-1} \tilde{\boldsymbol{u}}^n + \tau_{\rm eff}^{n+\theta} P_u^{\perp}[\boldsymbol{r}_h^{n+\theta}], \qquad (13)
$$

$$\tau_{\rm eff}^{n+\theta} := \left[\rho(\theta\delta t)^{-1} + (\tau_u^{n+\theta})^{-1}\right]^{-1}.$$

It is observed that the orthogonal projection of the residual is multiplied by the 'effective' stabilisation parameter $\tau_{\rm eff}$. However, one cannot simply replace τ_u by $\tau_{\rm eff}$, as the term $\tau_{\rm eff}^{n+\theta}(\theta\delta t)^{-1}\tilde{\boldsymbol{u}}^n$ needs to be taken into account. If a steady state is reached, it is observed from Eq. (11) that

$$\tilde{\boldsymbol{u}} = \tau_u P_u^\perp[\boldsymbol{r}_h].$$

Thus, the steady state does not depend on the time step size δt used to reach it.

The component $\tau_{\rm eff}^{n+\theta}\rho(\theta\delta t)^{-1}\tilde{\boldsymbol{u}}^n$ of $\tilde{\boldsymbol{u}}^{n+\theta}$ needs to be stored at the numerical integration points to evaluate its contribution to Eq. (9). This is the only relevant computational cost of considering the velocity SGSs time dependent.

From the comments above, it follows that the stabilising terms in the momentum equation (9) and the continuity equation (10) are:

$$\begin{aligned}
&\sum_{K^{n+\theta}}\int_{K^{n+\theta}} \tau_{\rm eff}^{n+\theta} P_u^\perp\left[\rho(\boldsymbol{u}_h^{n+\theta}-\boldsymbol{u}_{\rm dom}^{n+\theta})\cdot\nabla\boldsymbol{u}_h^{n+\theta} - \nabla\cdot(2\mu\nabla^S\boldsymbol{u}_h^{n+\theta}) + \nabla p_h^{n+\theta}\right]\\
&\qquad\cdot P_u^\perp\left[\rho(\boldsymbol{u}_h^{n+\theta}-\boldsymbol{u}_{\rm dom}^{n+\theta})\cdot\nabla\boldsymbol{v}_h + \nabla\cdot(2\mu\nabla^S\boldsymbol{v}_h)\right]\\
&\qquad+\sum_{K^{n+\theta}}\int_{K^{n+\theta}} \tau_p^{n+\theta} P_p^\perp[\nabla\cdot\boldsymbol{u}_h^{n+\theta}] P_p^\perp[\nabla\cdot\boldsymbol{v}_h],
\end{aligned}\tag{14}$$

$$\sum_{K^{n+\theta}}\int_{K^{n+\theta}} \tau_{\rm eff}^{n+\theta} P_u^\perp\left[\rho(\boldsymbol{u}_h^{n+\theta}-\boldsymbol{u}_{\rm dom}^{n+\theta})\cdot\nabla\boldsymbol{u}_h^{n+\theta} - \nabla\cdot(2\mu\nabla^S\boldsymbol{u}_h^{n+\theta}) + \nabla p_h^{n+\theta}\right]\cdot P_u^\perp[\nabla q_h],\tag{15}$$

respectively. Note that we have introduced the orthogonal projection in the operators that multiply the test functions. This highlights the symmetry of the formulation when there is no advection (Stokes problem). If the stabilisation parameters were constants, the introduction of these projections has no effect, and in general our experience is that for variable stabilisation parameters (as it is the case), these projections should correspond to the $L^2(\Omega(t^{n+\theta}))$-inner product weighted by these parameters (see [38]). In any case, our experience is that there are no significant differences in the numerical results putting or not the orthogonal projections applied to the operators evaluated with the test functions. Let us also note that all terms are evaluated element-wise; thus, for constant viscosities and linear elements (as those employed in Section 7), the contribution of the viscous term to the stabilisation is zero.

At this point, it is convenient to write the algebraic version of problem (9)-(10). It reads:

$$\begin{bmatrix}\mathsf{M}_{\rm G} & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\delta_t\mathsf{U}\\ \delta_t\mathsf{P}\end{bmatrix} + \begin{bmatrix}\mathsf{A}_{11} & \mathsf{A}_{12}\\ \mathsf{A}_{21} & \mathsf{A}_{22}\end{bmatrix}\begin{bmatrix}\mathsf{U}\\ \mathsf{P}\end{bmatrix} = \begin{bmatrix}\mathsf{F}_{\rm G}(\mathsf{U}^n)+\mathsf{F}_{\rm S}(\tilde{\mathsf{U}}^n)\\ 0\end{bmatrix},$$

where

$$\begin{bmatrix}\mathsf{A}_{11} & \mathsf{A}_{12}\\ \mathsf{A}_{21} & \mathsf{A}_{22}\end{bmatrix} := \begin{bmatrix}\mathsf{K}_{\rm G}(\mathsf{U})+\mathsf{K}_{\rm S}(\mathsf{U}) & -\mathsf{B}_{\rm G}+\mathsf{B}_{1,\rm S}(\mathsf{U})+\mathsf{B}_{2,\rm S}\\ \mathsf{B}_{\rm G}^T+\mathsf{B}_{1,\rm S}^T(\mathsf{U})-\mathsf{B}_{2,\rm S}^T & \mathsf{L}_{\rm S}\end{bmatrix}.\tag{16}$$

The identification of the different matrices and arrays with those of problem (9)-(10) is straightforward. Subscript G has been used to indicate terms arising from the Galerkin FE approximation, whereas subscript S has been used for those arising from the stabilisation terms. This equation is understood to correspond to the time interval $[t^n, t^{n+1}]$, and thus being evaluated at $n+\theta$. Nonlinearities, apart from the dependence of the stabilisation parameters on the velocity, have been explicitly displayed. The RHS depends on the unknowns at t^n, both the array of FE nodal velocities U and that of velocity SGSs $\tilde{\mathsf{U}}$ (recall that these are in fact known at the numerical integration points). The contribution

from the stabilisation terms to the A_{12} block of the stiffness matrix has been split as $\mathsf{B}_{1,\mathrm{S}}(\mathsf{U}) + \mathsf{B}_{2,\mathrm{S}}$; the first term comes from the convective operator applied to the test function that multiplies the pressure gradient, whereas the second comes from the viscous operator applied to the test function. Note that in the A_{21} block the first matrix keeps the sign and the second one changes it, both being transposed.

It is worth to remark that the presented formulation can be simplified. As the orthogonal projection converges to zero when the mesh is refined, not all terms coming from the stabilisation need to be accounted for. In particular, one can design a term-by-term stabilisation that has less terms to compute with the same accuracy [39]. In this case, we would have $\mathsf{B}_{1,\mathrm{S}} = \mathsf{B}_{2,\mathrm{S}} = \mathsf{0}$, and matrix K_{S} would only have contributions from the convective term.

The use of dynamic SGSs has been found to be crucial in the performance of the stabilised formulation proposed. Both linearisation schemes and iterative algebraic solvers perform much better in this case. This is partly due to the presence of δt in τ_{eff}. In fact, in residual based stabilisation schemes it is often seen that the stabilisation parameter is taken directly as τ_{eff}, without considering dynamic SGSs. But if τ_{eff} is used without coming from a temporal tracking of the SGSs, i.e., quasi-static SGSs are employed, the amount of stabilisation introduced in the formulation depends on δt and, in fact, if the steady state solution exists it depends on this time step size (see [7] for further discussion).

Let us close this section mentioning that the formulation presented acts as an implicit large eddy simulation problem, as it is shown in [40] and tested for example in [41]. Most real life simulations are turbulent, but we do not need to add any additional turbulence model to the formulation presented. In the numerical simulation presented we shall demonstrate that the features of turbulent flows are indeed captured.

4. Approximate imposition of boundary conditions

In the previous section we have considered that the Dirichlet velocity boundary conditions on $\Gamma_{\mathrm{mov}}(t)$ are imposed exactly. This is possible if nodes on this boundary are effectively created, as in this case mesh $\mathcal{T}_h^{n+1}$ can be considered a local re-meshing of the background mesh (see Fig. 2, bottom-right). However, activating and deactivating nodes is a cumbersome process from the numerical standpoint, as the profile of the stiffness matrix keeps changing from one time step to the other, thus implying continuous allocation and deallocation of computer memory. In this sense, it is much simpler to prescribe boundary conditions approximately, using a Nitsche's approach as described below. In this case, the degrees of freedom that need to be activated always belong to the background mesh. Note, however, that the splitting of the elements cut by $\Gamma_{\mathrm{mov}}(t)$ is still needed for integration purposes. The fixed-mesh ALE formulation, together with the approximate imposition of Dirichlet boundary conditions, are the two key features of the embedded approach we propose. The stabilised FE formulation presented in the previous section and the fractional step scheme described in the following one are simply adapted to this embedded approach.

4.1. Imposition of Dirichlet boundary conditions.

There are different methods to impose approximately the condition

$$\boldsymbol{u} = \boldsymbol{u}_{\mathrm{mov}} \quad \text{on } \Gamma_{\mathrm{mov}},\ t \in (0,T). \tag{17}$$

A pure algebraic approach is presented in [14], whereas a method based on a predetermined form of the Lagrange multiplier to enforce this condition is proposed in [13]. However, in this work we shall restrict ourselves to the classical Nitsche's method (see, e.g. [18]).

Let $\mathcal{E}_h^n = \{E^n\}$ be the collection of edges of $\mathcal{T}_h^n$, $n = 1, 2, \ldots$, including the edges generated by the intersection of $\Gamma_{\mathrm{mov}}(t^n)$ with $\mathcal{T}_h$. For simplicity, we assume that $\Gamma_{\mathrm{mov}}(t^n)$ is made by the union of edges in $\mathcal{E}_h^n$. In contrast with the exact imposition of condition (17), in Nitsche's method the test functions $\boldsymbol{v}_h \in V_h^n$ do not vanish on $\Gamma_{\mathrm{mov}}(t^n)$, and when the

differential equations are integrated by parts there is a contribution from this boundary. Let $\{E^n_{\rm mov}\}$ be the collection of edges contained in $\Gamma_{\rm mov}(t^n)$. Nitsche's method consists in adding to the discrete variational form of the problem the terms

$$\sum_{E^{n+\theta}_{\rm mov}} \int_{E^{n+\theta}_{\rm mov}} \boldsymbol{v}_h \cdot (p_h^{n+\theta}\boldsymbol{n} - 2\mu\, \boldsymbol{n}\cdot\nabla^S \boldsymbol{u}_h^{n+\theta})$$
$$+ \sum_{E^{n+\theta}_{\rm mov}} \int_{E^{n+\theta}_{\rm mov}} (\boldsymbol{u}_h^{n+\theta} - \boldsymbol{u}_{\rm mov}^{n+\theta}) \cdot (-q_h\boldsymbol{n} - 2\mu\, \boldsymbol{n}\cdot\nabla^S \boldsymbol{v}_h)$$
$$+ \sum_{E^{n+\theta}_{\rm mov}} \int_{E^{n+\theta}_{\rm mov}} \frac{\mu_{\rm N}}{h_E}\boldsymbol{v}_h \cdot (\boldsymbol{u}_h^{n+\theta} - \boldsymbol{u}_{\rm mov}^{n+\theta}). \tag{18}$$

Let us discuss the motivation for these three terms. The first term comes from the fact that the test function $\boldsymbol{v}_h$ does not vanish on $\Gamma_{\rm mov}(t^{n+\theta})$, and does not deserve further explanation. Note that should there be Neumann boundary conditions prescribed on part of $\Gamma_{\rm mov}(t^n)$, they would appear in the first integral of Eq. (18), and the resulting known term treated as an external force. This would not affect the design of the fractional step scheme proposed in Section 5.

The second term is added for symmetry reasons, and it is sometimes referred to as the adjoint consistency term. If there is no convection (i.e, an updated Lagrangian approach is used), it would lead to a symmetric contribution to the stiffness matrix, as the sign of the continuity equation could be changed by taking $-q_h$ as pressure test function. However, we use a negative sign for the viscous contribution of the second term, and this makes the contribution non-symmetric for non-symmetric problems (with convection). This unusual sign can be shown to have some enhanced stability and easy motivation in the context of discontinuous Galerkin approximations [42].

Finally, the third term in Eq. (18) is the so called stabilisation term, although in fact it is the term that penalises the restriction given by the boundary condition (17). There, h_E is the characteristic length of $E^{n+\theta}_{\rm mov}$ and $\mu_{\rm N}$ is an algorithmic parameter with units of viscosity, that we compute for each edge as

$$\mu_{\rm N} = \mu + \rho h_E \|\boldsymbol{u}_h^{n+\theta}\|_{L^\infty(E^{n+\theta}_{\rm mov})}.$$

This completes the description of the version of Nitsche's method that we employ.

4.2. **Stabilisation of badly cut elements.** It is known that this approach described leads to unstable results because of the presence of 'badly cut' elements, i.e., elements cut by $\Gamma_{\rm mov}(t^{n+\theta})$ of which only one small portion belongs to the flow domain $\Omega(t^{n+\theta})$. Recall that the integrals involved in the discrete variational form of the problem are performed over this domain. There are different strategies to avoid this instability, which in fact can be related to the algebraic ill-conditioning introduced by the badly cut elements in the stiffness matrix. Among these techniques, let us mention the ghost-penalty method proposed in [20], the shifted boundary method introduced in [21] or the aggregated unfitted FE method proposed in [22]. Here we propose a new alternative. There are several reasons for using the technique we propose next and not any of the methods just mentioned. First, it fits well with the stabilisation technique we use for the flow equations, since now we will also use projections onto FE spaces that we already have available, thus saving computing time. Second, and perhaps the most important, we do not need any additional data structure, such as the edge connectivities that are required for example in the implementation of the method described in [20]; the new term to be introduced can be computed looping over the elements, and only integrals over edges defining the boundary of the domain need to be computed. This leads to a third benefit, which is the simplification in the parallelisation of the method. Finally, we have found our approach quite effective

and robust in our computations; a simple numerical test is presented at the end of this section.

Let $\mathcal{T}^n_{h,\mathrm{cut}} = \{K^n_{\mathrm{cut}}\}$ be the collection of elements of the background mesh $\mathcal{T}_h$ cut by $\Gamma_{\mathrm{mov}}(t^n)$ together with their neighbours in the interior of the flow domain $\Omega(t^n)$. Let also $\Omega^n_{\mathrm{cut}} = \bigcup K^n_{\mathrm{cut}}$. In fact, Ω^n_{cut} could be defined with an arbitrary number of layers inside $\Omega(t^n)$. Note that the elements of Ω^n_{cut} cut by $\Gamma_{\mathrm{mov}}(t^n)$ will have a portion of them outside $\Omega(t^n)$. As degrees of freedom for the velocity and pressure interpolation we will use all those of the sub-mesh $\mathcal{T}^n_{h,\mathrm{cut}}$. As stabilising term to account for instability due to badly cut elements, we introduce

$$\sum_{K^{n+\theta}_{\mathrm{cut}}} \int_{K^{n+\theta}_{\mathrm{cut}}} \mu_{\mathrm{C}} \nabla \boldsymbol{v}_h : (\nabla \boldsymbol{u}_h^{n+\theta} - P_{\mathrm{cut}}[\nabla \boldsymbol{u}_h^{n+\theta}]) + \sum_{K^{n+\theta}_{\mathrm{cut}}} \int_{K^{n+\theta}_{\mathrm{cut}}} \frac{h_K^2}{\mu_{\mathrm{C}}} \nabla q_h : (\nabla p_h^{n+\theta} - P_{\mathrm{cut}}[\nabla p_h^{n+\theta}]), \tag{19}$$

where P_{cut} is the $L^2(\Omega^n_{\mathrm{cut}})$-projection, i.e., for any function f (a scalar, a vector or a tensor) defined in Ω^n_{cut}, in general discontinuous, there holds

$$\sum_{K^{n+\theta}_{\mathrm{cut}}} \int_{K^{n+\theta}_{\mathrm{cut}}} v_h P_{\mathrm{cut}}[f] = \sum_{K^{n+\theta}_{\mathrm{cut}}} \int_{K^{n+\theta}_{\mathrm{cut}}} v_h f,$$

for all continuous FE functions v_h in Ω^n_{cut} of the same tensorial order as f. The parameter μ_{C} has units of viscosity, and we compute it as

$$\mu_{\mathrm{C}} = \mu + \rho h_K \|\boldsymbol{u}_h^{n+\theta}\|_{L^\infty(K^{n+\theta}_{\mathrm{cut}})},$$

where h_K is the diameter of $K^{n+\theta}_{\mathrm{cut}}$. Since P_{cut} is the $L^2(\Omega^n_{\mathrm{cut}})$-projection, we could also write (19) as

$$\sum_{K^{n+\theta}_{\mathrm{cut}}} \int_{K^{n+\theta}_{\mathrm{cut}}} \mu_{\mathrm{C}} P^\perp_{\mathrm{cut}}[\nabla \boldsymbol{v}_h] : P^\perp_{\mathrm{cut}}[\nabla \boldsymbol{u}_h^{n+\theta}] + \sum_{K^{n+\theta}_{\mathrm{cut}}} \int_{K^{n+\theta}_{\mathrm{cut}}} \frac{h_K^2}{\mu_{\mathrm{C}}} P^\perp_{\mathrm{cut}}[\nabla q_h] : P^\perp_{\mathrm{cut}}[\nabla p_h^{n+\theta}], \tag{20}$$

with $P^\perp_{\mathrm{cut}} = I - P_{\mathrm{cut}}$, emphasising the symmetric nature of these terms.

It is observed that the terms in Eq. (20) are not consistent, in the sense that if we replace $\boldsymbol{u}_h$ by the exact velocity $\boldsymbol{u}$ and p_h by the exact pressure p they do not vanish. However, the difference $|\nabla \boldsymbol{u} - P_{\mathrm{cut}}[\nabla \boldsymbol{u}]|$ is expected to convergence to zero as the mesh is refined with the optimal power of the mesh size that can be provided by the FE interpolation, and likewise for $|\nabla p - P_{\mathrm{cut}}[\nabla p]|$. Thus, there is a consistency error, but it is of optimal order. A version of this method applied to free surface flows was already used in [43].

Combining Eqs. (9)-(10)-(14)-(15)-(18)-(19), we can finally write the fully discrete variational form of the formulation we propose. It reads as follows: given $\boldsymbol{u}_h^n \in V_h^n$, $\tilde{\boldsymbol{u}}^n \in V_h^{n,\perp}$ and $p_h^n \in Q_h^n$, find $\boldsymbol{u}_h^{n+1} \in V_h^{n+1}$ and $p_h^{n+1} \in Q_h^{n+1}$ such that

$$\begin{aligned}
&m^{n+\theta}(\boldsymbol{v}_h, \delta_t \boldsymbol{u}_h^{n+1}\big|_{\boldsymbol{x}^n}) + a^{n+\theta}(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+\theta}) \\
&\quad + c^{n+\theta}(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+\theta} - \boldsymbol{u}_{\mathrm{dom}}^{n+\theta}, \boldsymbol{u}_h^{n+\theta}) - b^{n+\theta}(p_h^{n+\theta}, \boldsymbol{v}_h) \\
&\quad + \sum_{K^{n+\theta}} \int_{K^{n+\theta}} \tau_{\mathrm{eff}}^{n+\theta} P_u^\perp \left[\rho(\boldsymbol{u}_h^{n+\theta} - \boldsymbol{u}_{\mathrm{dom}}^{n+\theta}) \cdot \nabla \boldsymbol{v}_h + \nabla \cdot (2\mu \nabla^S \boldsymbol{v}_h)\right] \\
&\qquad \cdot P_u^\perp \left[\rho(\boldsymbol{u}_h^{n+\theta} - \boldsymbol{u}_{\mathrm{dom}}^{n+\theta}) \cdot \nabla \boldsymbol{u}_h^{n+\theta} - \nabla \cdot (2\mu \nabla^S \boldsymbol{u}_h^{n+\theta}) + \nabla p_h^{n+\theta}\right] \\
&\quad + \sum_{E^{n+\theta}_{\mathrm{mov}}} \int_{E^{n+\theta}_{\mathrm{mov}}} \boldsymbol{v}_h \cdot (p_h^{n+\theta} \boldsymbol{n} - 2\mu\, \boldsymbol{n} \cdot \nabla^S \boldsymbol{u}_h^{n+\theta}) - \sum_{E^{n+\theta}_{\mathrm{mov}}} \int_{E^{n+\theta}_{\mathrm{mov}}} \boldsymbol{u}_h^{n+\theta} \cdot 2\mu\, \boldsymbol{n} \cdot \nabla^S \boldsymbol{v}_h
\end{aligned}$$

$$
\begin{aligned}
&+ \sum_{E^{n+\theta}_{\mathrm{mov}}} \int_{E^{n+\theta}_{\mathrm{mov}}} \frac{\mu_{\mathrm{N}}}{h_E} \boldsymbol{v}_h \cdot \boldsymbol{u}_h^{n+\theta} + \sum_{K^{n+\theta}} \int_{K^{n+\theta}} \tau_p^{n+\theta} P_p^{\perp}[\nabla \cdot \boldsymbol{u}_h^{n+\theta}] P_p^{\perp}[\nabla \cdot \boldsymbol{v}_h] \\
&+ \sum_{K^{n+\theta}_{\mathrm{cut}}} \int_{K^{n+\theta}_{\mathrm{cut}}} \mu_{\mathrm{C}} P_{\mathrm{cut}}^{\perp}[\nabla \boldsymbol{v}_h] : P_{\mathrm{cut}}^{\perp}[\nabla \boldsymbol{u}_h^{n+\theta}] \\
&= l(\boldsymbol{v}_h) + \sum_{K^{n+\theta}} \int_{K^{n+\theta}} \tau_{\mathrm{eff}}^{n+\theta} \rho (\theta \delta t)^{-1} \tilde{\boldsymbol{u}}^n \cdot P_u^{\perp} \left[\rho (\boldsymbol{u}_h^{n+\theta} - \boldsymbol{u}_{\mathrm{dom}}^{n+\theta}) \cdot \nabla \boldsymbol{v}_h + \nabla \cdot (2\mu \nabla^S \boldsymbol{v}_h) \right] \\
&- \sum_{E^{n+\theta}_{\mathrm{mov}}} \int_{E^{n+\theta}_{\mathrm{mov}}} \boldsymbol{u}_{\mathrm{mov}}^{n+\theta} \cdot 2\mu\, \boldsymbol{n} \cdot \nabla^S \boldsymbol{v}_h + \sum_{E^{n+\theta}_{\mathrm{mov}}} \int_{E^{n+\theta}_{\mathrm{mov}}} \frac{\mu_{\mathrm{N}}}{h_E} \boldsymbol{v}_h \cdot \boldsymbol{u}_{\mathrm{mov}}^{n+\theta} \\
&b^{n+\theta}(q_h, \boldsymbol{u}_h^{n+\theta}) + \sum_{K^{n+\theta}} \int_{K^{n+\theta}} \tau_{\mathrm{eff}}^{n+\theta} P_u^{\perp}[\nabla q_h] \\
&\qquad \cdot P_u^{\perp} \left[\rho (\boldsymbol{u}_h^{n+\theta} - \boldsymbol{u}_{\mathrm{dom}}^{n+\theta}) \cdot \nabla \boldsymbol{u}_h^{n+\theta} - \nabla \cdot (2\mu \nabla^S \boldsymbol{u}_h^{n+\theta}) + \nabla p_h^{n+\theta} \right] \\
&- \sum_{E^{n+\theta}_{\mathrm{mov}}} \int_{E^{n+\theta}_{\mathrm{mov}}} \boldsymbol{u}_h^{n+\theta} \cdot q_h \boldsymbol{n} + \sum_{K^{n+\theta}_{\mathrm{cut}}} \int_{K^{n+\theta}_{\mathrm{cut}}} \frac{h_K^2}{\mu_{\mathrm{C}}} P_{\mathrm{cut}}^{\perp}[\nabla q_h] : P_{\mathrm{cut}}^{\perp}[\nabla p_h^{n+\theta}] \\
&= - \sum_{E^{n+\theta}_{\mathrm{mov}}} \int_{E^{n+\theta}_{\mathrm{mov}}} \boldsymbol{u}_{\mathrm{mov}}^{n+\theta} \cdot q_h \boldsymbol{n},
\end{aligned}
$$

for all test functions $\boldsymbol{v}_h \in V_{0,h}^{n+1}$ and $q_h \in Q_h^{n+1}$. Once $\boldsymbol{u}_h^{n+\theta}$ is known, $\tilde{\boldsymbol{u}}^{n+\theta}$ can be computed using Eq. (13).

At this point it is convenient to write the structure of the stiffness matrix of the system. Instead of Eq. (16), now we have

$$
\begin{bmatrix} \mathsf{A}_{11} & \mathsf{A}_{12} \\ \mathsf{A}_{21} & \mathsf{A}_{22} \end{bmatrix} = \begin{bmatrix} \mathsf{K}_{\mathrm{G}}(\mathsf{U}) + \mathsf{K}_{\mathrm{S}}(\mathsf{U}) + \mathsf{K}_{\mathrm{N}} + \mathsf{K}_{\mathrm{C}} & -\mathsf{B}_{\mathrm{G}} + \mathsf{B}_{1,\mathrm{S}}(\mathsf{U}) + \mathsf{B}_{2,\mathrm{S}} + \mathsf{B}_{\mathrm{N}} \\ \mathsf{B}_{\mathrm{G}}^T + \mathsf{B}_{1,\mathrm{S}}^T(\mathsf{U}) - \mathsf{B}_{2,\mathrm{S}}^T - \mathsf{B}_{\mathrm{N}}^T & \mathsf{L}_{\mathrm{S}} + \mathsf{L}_{\mathrm{C}} \end{bmatrix}, \tag{21}
$$

where the contributions from Nitsche's method have been identified with subscript N and those from the stabilisation of the cuts with subscript C. Note that, although not explicitly displayed, these matrices depend also on the velocity through the calculation of their algorithmic parameters μ_{N} and μ_{C}.

Let us finally remark that when the orthogonal projection of the pressure gradient is treated explicitly, as explained above, it is convenient to consider only the non-hydrostatic pressure gradient, i.e., to subtract $\rho \boldsymbol{f}$ from the total pressure gradient (assuming $\boldsymbol{f}$ to be a potential field), for improving the behaviour of the iterative scheme.

4.3. **A numerical test for the embedded formulation.** Let us conclude this section with a numerical test to check that the proposed technique to impose Dirichlet boundary conditions and the stabilisation of badly cut elements provides an optimally accurate formulation in $L^2(\Omega)$. To this end, let us consider Ω as the unit square $\Pi = (0,1) \times (0,1) \subset \mathbb{R}^2$ to which a circle Λ centred at $(1/2, 1/2)$ and radius 0.3 is subtracted, i.e., $\Omega = \Pi \setminus \Lambda$. On the boundaries of Ω the velocity is prescribed and a force vector is introduced so that the exact solution to the problem is

$$
\boldsymbol{u}_0(x,y) = 100(f(x)f'(y), -f'(x)f(y)), \quad \text{with} \quad f(z) = z^2(1-z)^2, \tag{22}
$$

$$
p(x,y) = 100x^2. \tag{23}
$$

The viscosity is taken as $\mu = 0.1$ and the density as $\rho = 1$. A structured uniform mesh of triangles of size $h = 1/n$, n being the number of subdivisions per side of Π, is employed to discretise Π. As a reference, the solution computed in the whole unit square Π, but with the error computed only in Ω, is obtained; we refer to this as the solution obtained with

no cuts. Then the problem is solved prescribing weakly the boundary conditions on $\partial\Lambda$ (note that $\boldsymbol{u} = \boldsymbol{0}$ on $\partial\Pi$). Linear interpolation is used for both velocity and pressure.

The $L^2(\Omega)$-norm of the velocity is shown in Fig. 3. It is observed that convergence is virtually identical for the solution with no cuts and using the embedded formulation with the stabilisation we propose.

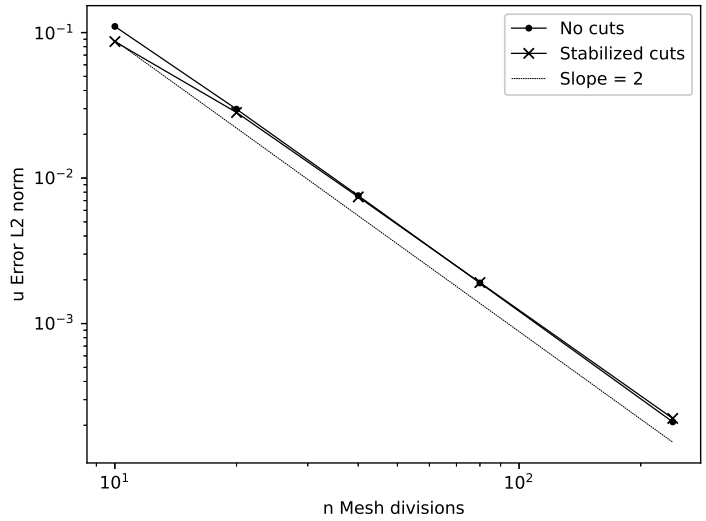


FIGURE 3. Convergence test to check the performance of the embedded formulation proposed in a stationary problem.

5. Artificial compressibility and fractional step scheme

5.1. **Artificial compressibility technique.** We have found convenient to use an artificial compressibility approach for efficiency purposes, but controlling the amount of added compressibility to keep a good approximation to the original incompressibility condition [24, 25]. The reason is simply that this leads to a better conditioning of the final pressure equation obtained below, and iterative schemes to solve this equation converge significantly better. This model can be motivated from the equations of isentropic compressible flows neglecting some terms.

The idea is to replace the incompressibility condition given in Eq. (3) by

$$\frac{1}{\rho\, c^2}\frac{\partial p}{\partial t} + \nabla\cdot\boldsymbol{u} = 0,$$

where c is a parameter with units of velocity that we compute as $c = \beta U_{\max}$, with $U_{\max}$ the maximum velocity norm in the domain and β a parameter to be adjusted. Using the same scheme to approximate the pressure time derivative as the velocity time derivative, equations Eq. (5) and Eq. (8) need to be respectively replaced by:

$$\frac{1}{\rho\, c^2}\delta_t p_h^{n+1}|_{\boldsymbol{x}^n} + \nabla\cdot\boldsymbol{u}^{n+\theta} = 0 \quad \text{and} \quad \frac{1}{\rho\, c^2}(q_h, \delta_t p_h^{n+1}|_{\boldsymbol{x}^n}) + b^{n+\theta}(q_h, \boldsymbol{u}_h^{n+\theta}) = 0.$$

As $\delta_t p_h^{n+1}$ belongs to the pressure FE space, the stabilisation has no effect on the artificial compressibility term. Thus, it is readily seen that the algebraic system to be solved at each time step is

$$\begin{bmatrix} \mathsf{M}_{\mathrm{G}} & 0 \\ 0 & \mathsf{M}_{\epsilon} \end{bmatrix} \begin{bmatrix} \delta_t \mathsf{U} \\ \delta_t \mathsf{P} \end{bmatrix} + \begin{bmatrix} \mathsf{A}_{11} & \mathsf{A}_{12} \\ \mathsf{A}_{21} & \mathsf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathsf{U} \\ \mathsf{P} \end{bmatrix} = \begin{bmatrix} \mathsf{F}_u \\ \mathsf{F}_p \end{bmatrix}, \tag{24}$$

where M_ϵ is the mass matrix computed with pressure shape functions multiplied by $\epsilon := \rho^{-1}c^{-2}$. The RHS accounts for all contributions to the discrete momentum and the discrete continuity equations, including now non-homogeneous velocities, and the stiffness matrix is given by Eq. (21). Of course the approximation of the temporal derivatives has terms evaluated at t^n that can be moved to the RHS, but it is convenient to keep them in the left-hand-side (LHS) to design the fractional step scheme presented next. Recall that the time step at which the unknowns are evaluated has been omitted, but it corresponds to $t^{n+\theta}$ for the unknown multiplying the stiffness matrix and the incremental quotient $\delta_t\mathsf{U}$ is computed with the difference between U^{n+1} and U^n, and likewise for $\delta_t\mathsf{P}$.

5.2. **Fractional step strategy.** We present now a fractional step scheme designed at the pure discrete level. As explained in [23], there are two ways to attempt the design of algebraic fractional step schemes. The first is to consider an incomplete factorisation of the matrix of the final algebraic system, and the second to extrapolate one of the variables, compute the rest, and then correct the effect of the extrapolation. We pursue this second path in this work. The plan for each time step is as follows: we consider first an extrapolation of the pressure in the momentum equation that allows us to compute an intermediate velocity, then we obtain a pressure equation that, after an additional modification to make it easily solvable, allows us to compute the pressure and, with this pressure at our disposal, we compute the final velocity to be used in the next time step. The key point is that the modification of the pressure equation needs some care, as the terms introduced by the weak imposition of boundary conditions complicates this. The way to circumvent this issue is to use an extrapolation for these terms of one order higher than for the pressure gradient in the first step and then skip its correction. This procedure is described next and, since the order of the time integration scheme is at most 2 (for $\theta = 1/2$) the splitting scheme to be presented has to have also an additional error limited to order $\mathcal{O}(\delta t^2)$.

Let us consider the system, fully equivalent to (24), given by

$$\frac{1}{\theta\delta t}\mathsf{M}_{\mathrm{G}}(\hat{\mathsf{U}}-\mathsf{U}^n)+\mathsf{A}_{11}\mathsf{U}-\mathsf{B}_{\mathrm{G}}\mathsf{P}^n+\mathsf{A}_{12,\mathrm{R}}((1+\theta)\mathsf{P}^n-\theta\mathsf{P}^{n-1})=\mathsf{F}_u, \tag{25}$$

$$\frac{1}{\theta\delta t}\mathsf{M}_{\mathrm{G}}(\mathsf{U}-\hat{\mathsf{U}})-\mathsf{B}_{\mathrm{G}}(\mathsf{P}-\mathsf{P}^n)+\mathsf{A}_{12,\mathrm{R}}[\mathsf{P}-((1+\theta)\mathsf{P}^n-\theta\mathsf{P}^{n-1})]=0, \tag{26}$$

$$\frac{1}{\theta\delta t}\mathsf{M}_{\epsilon}(\mathsf{P}-\mathsf{P}^n)+\mathsf{A}_{21}\mathsf{U}+\mathsf{A}_{22}\mathsf{P}=\mathsf{F}_p. \tag{27}$$

We have introduced a new variable $\hat{\mathsf{U}}$ and we have written $\mathsf{A}_{12}=-\mathsf{B}_{\mathrm{G}}+\mathsf{A}_{12,\mathrm{R}}$. Clearly, adding up (25) and (26) we recover the original discrete momentum equation. Since P^n is an approximation of order $\mathcal{O}(\delta t)$ to P and $(1+\theta)\mathsf{P}^n-\theta\mathsf{P}^{n-1}$ an extrapolation of order $\mathcal{O}(\delta t^2)$, we may expect the difference $\mathsf{U}-\hat{\mathsf{U}}$ to be of order $\mathcal{O}(\delta t^2)$. Therefore, if in (25) we replace U by $\hat{\mathsf{U}}$ and in (26) we neglect $\mathsf{P}-((1+\theta)\mathsf{P}^n-\theta\mathsf{P}^{n-1})$, the expected error will be of order $\mathcal{O}(\delta t^2)$. The resulting problem is:

$$\frac{1}{\theta\delta t}\mathsf{M}_{\mathrm{G}}(\hat{\mathsf{U}}-\mathsf{U}^n)+\mathsf{A}_{11}\hat{\mathsf{U}}-\mathsf{B}_{\mathrm{G}}\mathsf{P}^n+\mathsf{A}_{12,\mathrm{R}}((1+\theta)\mathsf{P}^n-\theta\mathsf{P}^{n-1})=\mathsf{F}_u, \tag{28}$$

$$\frac{1}{\theta\delta t}\mathsf{M}_{\mathrm{G}}(\mathsf{U}-\hat{\mathsf{U}})-\mathsf{B}_{\mathrm{G}}(\mathsf{P}-\mathsf{P}^n)=0, \tag{29}$$

$$\frac{1}{\theta\delta t}\mathsf{M}_{\epsilon}(\mathsf{P}-\mathsf{P}^n)+\mathsf{A}_{21}\mathsf{U}+\mathsf{A}_{22}\mathsf{P}=\mathsf{F}_p. \tag{30}$$

The main advantage of this system with respect to (25)-(27) is that the first equation is uncoupled, i.e., it allows us to solve directly for $\hat{\mathsf{U}}$ (being understood that A_{11} is also computed with this velocity). Furthermore, we can perform an additional approximation to uncouple the calculation of P. To this end, let us also split $\mathsf{A}_{21}=\mathsf{B}_{\mathrm{G}}^T+\mathsf{A}_{21,\mathrm{R}}$. If we

isolate U in terms of $\hat{\mathsf{U}}$ from Eq. (29) and insert the result in Eq. (30) we get:

$$\frac{1}{\theta\delta t}\mathsf{M}_\epsilon(\mathsf{P}-\mathsf{P}^n)+\theta\delta t\mathsf{B}_{\mathrm{G}}^T\mathsf{M}_{\mathrm{G}}^{-1}\mathsf{B}_{\mathrm{G}}(\mathsf{P}-\mathsf{P}^n)+\mathsf{A}_{21,\mathrm{R}}\theta\delta t\mathsf{M}_{\mathrm{G}}^{-1}\mathsf{B}_{\mathrm{G}}(\mathsf{P}-\mathsf{P}^n)+(\mathsf{L}_{\mathrm{S}}+\mathsf{L}_{\mathrm{C}})\mathsf{P} = -\mathsf{A}_{21}\hat{\mathsf{U}}+\mathsf{F}_p. \tag{31}$$

Since $\mathsf{P}=(1+\theta)\mathsf{P}^n-\theta\mathsf{P}^{n-1}+\mathcal{O}(\delta t^2)$, we have that

$$\mathsf{P}-\mathsf{P}^n=\theta(\mathsf{P}^n-\mathsf{P}^{n-1})+\mathcal{O}(\delta t^2).$$

We cannot approximate $\mathsf{P}-\mathsf{P}^n$ by $\mathsf{P}^n-\mathsf{P}^{n-1}$ in all the LHS of Eq. (31), as there would be no equation for P, but we can do it in the third term of the LHS. This implies an explicit treatment of some terms, but in our calculations we have never observed a time step limitation because of this approximation. Furthermore, matrix $\mathsf{B}_{\mathrm{G}}^T\mathsf{M}_{\mathrm{G}}^{-1}\mathsf{B}_{\mathrm{G}}$ represents at the discrete level an approximation to the Laplacian operator, and can be replaced by the standard Laplacian matrix L_{G} (see [44]); this approximation is of optimal order in space. Note that to be able to do this approximation it is crucial not to change the mass matrix of the Galerkin problem, M_{G}; we achieve this condition because of the use of velocity SGSs orthogonal to the velocity FE space.

With the two approximations described, Eq. (31) leads to:

$$\frac{1}{\theta\delta t}\mathsf{M}_\epsilon(\mathsf{P}-\mathsf{P}^n)+(\theta\delta t\mathsf{L}_{\mathrm{G}}+\mathsf{L}_{\mathrm{S}}+\mathsf{L}_{\mathrm{C}})\mathsf{P} = \theta\delta t\mathsf{L}_{\mathrm{G}}\mathsf{P}^n-\mathsf{A}_{21,\mathrm{R}}\theta^2\delta t\mathsf{M}_{\mathrm{G}}^{-1}\mathsf{B}_{\mathrm{G}}(\mathsf{P}^n-\mathsf{P}^{n-1})-\mathsf{A}_{21}\hat{\mathsf{U}}+\mathsf{F}_p.$$

This is an equation that can be solved for P once we have computed $\hat{\mathsf{U}}$. It is understood that matrix A_{21} is computed with this velocity. Therefore, we can solve the three following equations sequentially:

$$\frac{1}{\theta\delta t}\mathsf{M}_{\mathrm{G}}(\hat{\mathsf{U}}-\mathsf{U}^n)+\mathsf{A}_{11}\hat{\mathsf{U}}-\mathsf{B}_{\mathrm{G}}\mathsf{P}^n+\mathsf{A}_{12,\mathrm{R}}((1+\theta)\mathsf{P}^n-\theta\mathsf{P}^{n-1})=\mathsf{F}_u, \tag{32}$$

$$\frac{1}{\theta\delta t}\mathsf{M}_\epsilon(\mathsf{P}-\mathsf{P}^n)+(\theta\delta t\mathsf{L}_{\mathrm{G}}+\mathsf{L}_{\mathrm{S}}+\mathsf{L}_{\mathrm{C}})\mathsf{P} = \theta\delta t\mathsf{L}_{\mathrm{G}}\mathsf{P}^n-\mathsf{A}_{21,\mathrm{R}}\theta^2\delta t\mathsf{M}_{\mathrm{G}}^{-1}\mathsf{B}_{\mathrm{G}}(\mathsf{P}^n-\mathsf{P}^{n-1})-\mathsf{A}_{21}\hat{\mathsf{U}}+\mathsf{F}_p, \tag{33}$$

$$\frac{1}{\theta\delta t}\mathsf{M}_{\mathrm{G}}(\mathsf{U}-\hat{\mathsf{U}})-\mathsf{B}_{\mathrm{G}}(\mathsf{P}-\mathsf{P}^n)=0. \tag{34}$$

This is the fractional step scheme we propose. The splitting error is of second order, and for $\theta=1/2$ it is a globally second order time integration scheme. Of course, we could have started from other monolithic time integration schemes, such as the second order BDF method.

From Eq. (33) the need for approximating $\mathsf{P}-\mathsf{P}^n$ by $\theta(\mathsf{P}^n-\mathsf{P}^{n-1})$ in the second term of the LHS of Eq. (31) becomes apparent. First, we can approximate $\mathsf{B}_{\mathrm{G}}^T\mathsf{M}_{\mathrm{G}}^{-1}\mathsf{B}_{\mathrm{G}}$ by matrix L_{G} of narrower stencil, but we do not have any means (at least to our knowledge) to approximate matrix $\mathsf{A}_{21,\mathrm{R}}\theta^2\delta t\mathsf{M}_{\mathrm{G}}^{-1}\mathsf{B}_{\mathrm{G}}$ by a more compact one. Second, the terms treated explicitly do not contribute at all to spatial stability, neither those coming from the stabilised formulation ($\mathsf{B}_{1,\mathrm{S}}^T(\hat{\mathsf{U}})-\mathsf{B}_{2,\mathrm{S}}^T$) nor that coming from Nitsche's method ($\mathsf{B}_{\mathrm{N}}^T$).

The final scheme (32)-(34) has some nonlinear terms that need to be linearised. This can be done in an iterative loop, using either a fixed point or a Newton-Raphson linearisation, or directly taking the second order extrapolated velocity from previous time steps. The same applies to the various orthogonal projections involved in the formulation, which can be treated explicitly as explained earlier.

5.3. **A numerical test for the fractional step scheme.** To close this section, we extend the test of subsection 4.3 to a transient test using the fractional step method proposed.

Boundary conditions, initial conditions and forcing term are chosen so that the exact solution to the problem is

$$\boldsymbol{u}(x,y,t) = \cos(\pi t)(\exp(-t)+1)\boldsymbol{u}_0,$$

where $\boldsymbol{u}_0$ is given in (22) and the pressure is again given by (23). A rigid body horizontal displacement $x(t) = 0.1\sin(\pi t)$ is prescribed to the cylinder Λ, so that the problem corresponds to an incompressible flow in a moving domain. The finest mesh used in subsection 4.3 is now employed for the space approximation. Only the embedded strategy is considered. For the time approximation, both BDF1 and BDF2 schemes are considered, combined with the fractional step scheme we have proposed.

Velocity convergence in the $L^2(\Omega)$ norm at time $t = 1$ with the time step size is shown in Fig. 4. It is seen that for BDF1 the error is driven by the temporal approximation, and thus a perfect first order rate of convergence is observed. However, for BDF2 the error saturates at the spatial error observed in Fig. 3 for the finest mesh. Thus, the convergence curve moves from a steep straight line of slope around 2 to a horizontal asymptote. Nevertheless, this test demonstrates that the fractional step scheme proposed does not spoil the convergence rate expected for the BDF2 scheme. The saturation error can be obviously decreased using higher order elements, in particular quadratic triangles.

Fig. 4 also compares the results obtained with the fractional step scheme and those found with a monolithic integration in time. For BDF1 the errors are very similar, but for BDF2 the errors computed using the fractional step scheme are about 5 times higher than using a monolithic scheme. This is a well known fact of fractional step methods, not to be attributed to our formulation tailored for moving domains. In any case, convergence is quadratic for BDF2, which is the best we can expect.

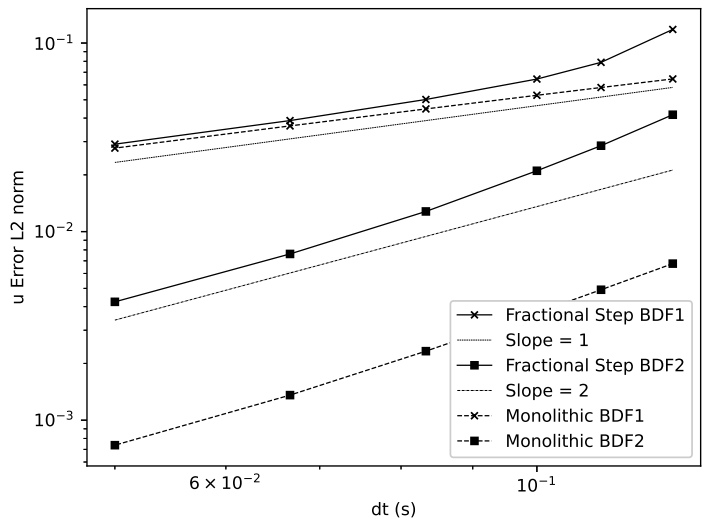


FIGURE 4. Convergence test to check the performance of the fractional step scheme proposed.

6. Adaptive mesh refinement and parallel implementation

.

6.1. **Level set function for the boundary definition.** For the tracking of the moving boundary, a standard option is to rely on the classical level set method. The main feature of this method is that it allows for a sharp representation of the interface, which results in an accurate definition of the fluid domain at time t, $\Omega(t)$. The level set function ψ is defined on the whole landscape domain Ω^0, in such a way that, at each time instant t:

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

For the definition of such boundary, contrary to other problems such as free surface or multi-fluid flows, the level set function can be described directly from the geometry of the moving boundary.

An alternative to the use of a classical level set function is to describe the moving object by another auxiliary FE mesh that includes the geometry of the moving body. This mesh moves in a rigid body (Lagrangian) way, following the movement of the moving obstacle. Thus, this can be considered as a *foreground* mesh that moves on top of the background mesh $\mathcal{T}_h$ that covers the whole computational domain. One can then define a fixed level set on this Lagrangian mesh, and then, at each time step, this level set function is *projected* onto the background mesh. In turn, the way the level set function is constructed depends on whether the foreground mesh is body fitted or not. If the foreground mesh is fitted to the moving object, i.e., element boundaries of the foreground mesh define the object, then ψ is easily constructed (for example by computing node-to-node distances). On the contrary, if the foreground mesh is not body fitted and the object is defined from a surface representation (in any format used in computational geometry), then this surface has to be intersected with the foreground mesh to compute ψ. In the numerical simulation of Section 7 we have considered a body fitted foreground mesh to ease the calculation of the level set function, but there are of course other options. The key point is that the level set function needs to be computed only once, as a preprocess, and then projected at each time step to the background mesh.

6.2. **Algorithmic strategy for the projection step.** Let us start this subsection 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 FE mesh by using graph partitioning algorithms, specifically ParMetis [45] in our case, 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. Also, at each time step we require a projection of the level set function from the Lagrangian mesh to the background Eulerian mesh. Independently of the case of projection considered, 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 after the projection, which introduces the additional difficulty of a parallel spatial search across multiple threads. Several possible strategies exist for treating this issue.

The approach we favour for these projections consists in implementing an octree-based (quadtree in two dimensions) search algorithm whose efficiency does not depend on the time step size. 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 neighbour

processors, contains the sought element when it is not found in the current processor. Since in the adaptive fixed-mesh ALE method described next the number of elements and the position of the nodes evolves in time, the octree needs to be rebuilt at each time step.

6.3. **Adaptive approach.** For the adaptive refinement we rely on the algorithm presented in [30], which allows the distributed memory parallel FE 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 summarise 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. As said above, this means that each node of the mesh belongs to a processor. This feature, which is strongly linked to the design of the parallel FE code, introduces some particularities when designing the hierarchical mesh refiner. 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 criterion needs to be provided so that the algorithm can decide which areas of the mesh need to be refined; secondly, the FE solver needs to deal with hanging nodes which, if a straightforward implementation is done, results in non-conforming discretisations. The way we deal with hanging nodes is to eliminate their associated degrees of freedom and impose that they are obtained from the interpolation of the nodes of the parent element, i.e., the element that has been refined but whose neighbour has not. This results in a conforming approximation. In particular, in the case of linear elements, the unknowns at the hanging nodes are prescribed to be the linear interpolation from the nodes of the parent element (see [43]).

With regards to the mesh refinement criterion, 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 criterion at t^n reads as follows:

- If an element is occupied by the fluid and its refinement level is less than n_{Ref} (given), then refine it.
- Else if the element is less than n_{Lay} (given) layers of elements away from the interface $\Gamma_{\mathrm{mov}}(t^n)$ and its refinement level is less than $n_{\mathrm{Ref}} - n_{\mathrm{Lay}}$, 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 $\Gamma_{\mathrm{mov}}(t^n)$ in order to have a more accurate representation of the geometry of the moving obstacle. The only care which needs to be taken is in the parallel implementation of this refinement criterion: as explained in [30], the parallel refinement algorithm requires that the refinement criterion passed to the refiner is the same in all the processors. The problem for this is that the n_{Lay} criterion is geometric and needs to propagate from the interface 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 criterion 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 FE 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.

7. Numerical example

In this section we present an application of the methodology described. This methodology is particularly well suited for large scale calculations, although the numerical example we will present has been chosen solely for illustrative purposes.

We do not pretend to give details about the data of the problem and the quality of the numerical results, but only to demonstrate qualitatively the application of the formulation we propose. The accuracy and robustness of its different components has been proved elsewhere. Concerning the new ingredients developed in this paper, we show that results are completely smooth close to boundaries, demonstrating that the weak imposition of boundary conditions and its stabilisation work properly, and give details of the savings in computing time provided by the fractional step scheme proposed. We also show the type of mesh refinement employed in this simulation.

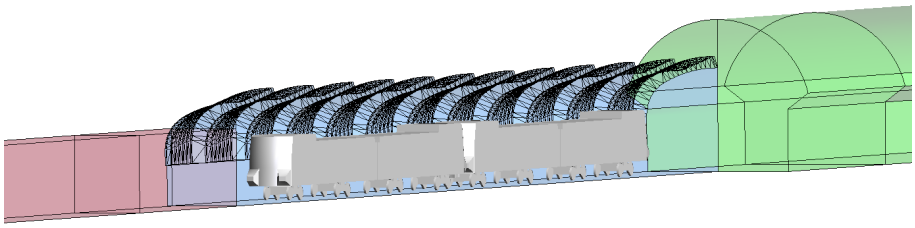


Figure 5. Train travelling through the middle section of the tunnel.

7.1. **Problem statement.** The problem we consider is the circulation of a train through a tunnel, with the particularity that there exists a change in the cross section of this tunnel and the transition between tunnel sections (see Fig. 5). The objective of the simulation was to evaluate the pressure peaks generated by the train circulation with the aim of structurally designing fire-proof panels in the tunnel wall. This problem fits perfectly in the type of problems described in this paper, as the only possibility to model it is to consider the motion of a solid, the train, inside the tunnel.

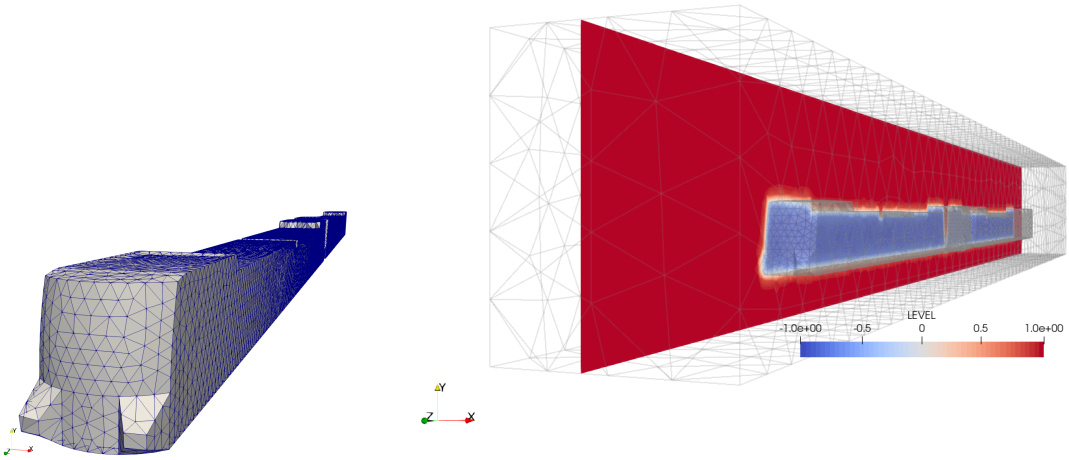
The tunnel is composed of three sections: a narrow constant 20 m^2 cross-section tunnel, a 95 m long curvy tunnel with average section of 26 m^2, and another constant section tunnel of 83 m^2. The section of greatest interest is the curvy region in the middle as well as the transition when the train enters this section from the two adjacent ones. The other sections are extended to reduce the impact of boundary conditions at the ends. As a result, the total tunnel length of the computational model is 590 m.

In the studied case, the train is initially placed in the narrow tunnel section moving at 100 km/h. It travels through the middle curvy section and continues through the wide tunnel. From now on, all physical units are assumed to be given in the SI system. The values of viscosity and density employed are those corresponding to air.

The surfaces of the train and the tunnel walls are modelled as no-slip boundaries. The ends of the computational domain are in reality sections where the tunnel continues but where the flow is expected to be negligible. Therefore, they are modelled as zero velocity Dirichlet surfaces. This assumption ensures mass conservation and avoids boundary instabilities and it is an approximation to considering a tunnel of infinite length. Furthermore, the length of the computational domain is large enough to ensure that boundary conditions do not impact the flow around the train. The train surface is considered a moving no-slip wall.

7.2. **Numerical strategy.** The problem is modelled by embedding a foreground and a background mesh. The foreground mesh contains the geometry of the train and tracks

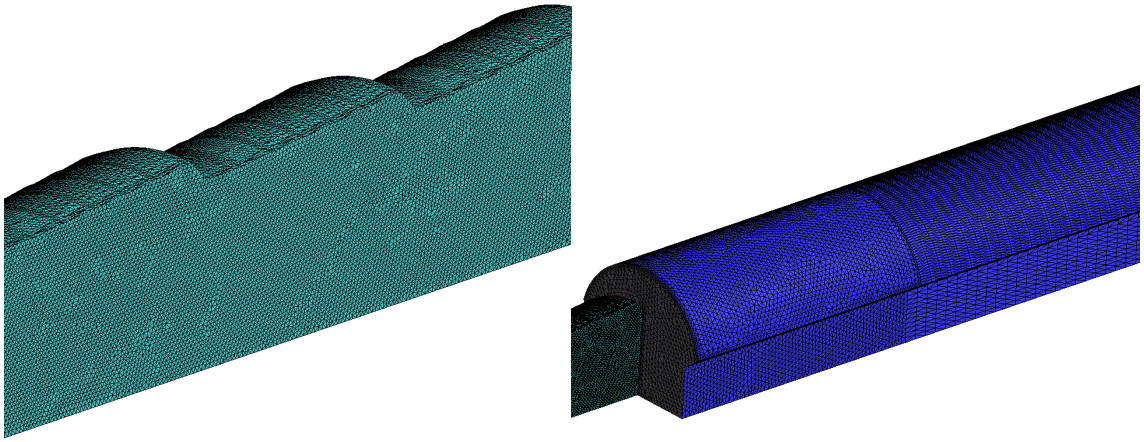
its displacement. The background mesh has the geometry of the air volume inside the tunnel and it is where the computation of the incompressible Navier-Stokes problem takes place. The position and velocity of the train in the foreground mesh is interpolated to the background mesh every time step, where it is imposed as an embedded boundary condition of the aerodynamic problem. Thus, in this case the definition of the solid surface is done through an auxiliary FE mesh.



(A) Foreground surface mesh

(B) Foreground volume mesh

FIGURE 6. Foreground mesh.



(A) Background mesh, middle curvy tunnel

(B) Background mesh, wide tunnel section

FIGURE 7. Unrefined background mesh.

Two meshes are constructed to model the foreground and the background domains. On the one hand, the foreground mesh focuses on accurately representing the details of the geometry of the train, but this is the only requirement, and thus it can be relatively coarse. The foreground mesh has 32702 nodes and 189335 tetrahedral elements. Its restriction to the surface of the train is shown in Fig. 6a, whereas Fig. 6b shows the whole volume of the

foreground mesh in which the train is immersed; it is in this foreground mesh where the level set is defined and projected onto the background mesh to obtain the position of the train. The cost of generating the foreground mesh is very low (a few minutes, at most).

On the other hand, the background mesh contains the geometry of the tunnel and must comply with the small grid size required by a high-fidelity Navier-Stokes computation. This is achieved by constructing an initial coarse mesh to which adaptive mesh refinement is applied during the computation. While the two tunnel regions with constant section have semi-structured grids generated by extrusion of a surface mesh (see Fig. 7b), the curvy middle section is fully unstructured (Fig. 7a). Smaller grid size is applied near the tunnel walls to better capture the flow close to the train surface, although it is not our intention to fully capture the turbulent boundary layer. As stated in [46], Nitsche's method when applied to turbulent flows acts as a wall law model. In total, the unrefined background mesh has 2742050 tetrahedral elements and 528493 nodes. In the final computation, adaptive mesh refinement is used on the background mesh. The refinement strategy is based on layers around the surface of the train. In each time step, the 15 layers of elements around the interpolated position of the surface of the train are refined. An additional uniform refinement is also applied. Because the train moves along the tunnel, the refined mesh changes every time step. The refined mesh reached over 60M elements and over 8M nodes.

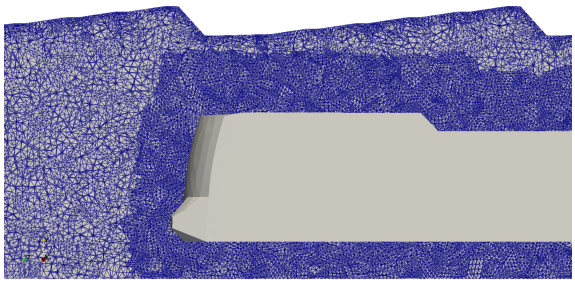


FIGURE 8. Refined background mesh with adaptive refinement around the position of the train.

The computation of the Navier-Stokes problem follows the methodology explained in the previous sections. The background mesh is cut according to the current position of the train surface in the foreground mesh. The velocity of the train is imposed as Dirichlet condition at the cuts. This condition is weakly imposed using Nitsche's method and stabilisation of the cuts. The scheme is stabilised with the following algorithmic parameters for the VMS method: $c_1 = 4.0$ and $c_2 = 2.0$.

The incompressible flow is computed with the fractional step algorithm described in Section 5. In the resolution of the velocity step, 4 Picard (fixed point) iterations are employed. A relaxation factor of 0.8 is used to improve the convergence of the non-linearity.

The artificial compressibility technique is employed, with a parameter of $\beta = 30$, resulting in $c^2 \approx 10^6$. The artificial compressibility does not help stabilising the pressure, since it is already stable using the stabilisation scheme employed, but improves the conditioning of the pressure equation resulting from the fractional step scheme without altering significantly the incompressibility approximation. In this particular example we have checked for a few time steps that the norm of the velocity, which is not zero because of the interpolation (not divergence free) and the stabilisation (see Eq. (10)), is very similar with and without this artificial compressibility, but much less iterations are required by the iterative solver to converge. We cannot claim $\beta = 30$ is a universal value to be used, and choosing it for each case may require running a few time steps of the simulation with different values

of β before launching the complete one. Nevertheless, $\beta = 30$ could be a reasonable initial guess.

The transient movement of the train takes 7 seconds of physical time. Time integration is done with a second order BDF2 scheme. The problem is computed along 1000 time steps. The chosen time step of $\delta t = 0.007$ is found to be small enough to ensure the nonlinear stability of the implicit integration and to enter the inertial range of the turbulent spectrum.

The computation was carried out on a HPC cluster using 16 Intel Xeon E5-2690v4 processors with a total of 224 CPUs. This represents mesh partitions of around 250000 elements per MPI process, which entails great parallel efficiency according to previous scalability tests. Load rebalancing is also applied to ensure small parallelisation overhead.

Note that the methodology introduced here also works for smaller computations, but the high mesh resolution and the turbulence capturing aimed for this application required the use of a HPC cluster.

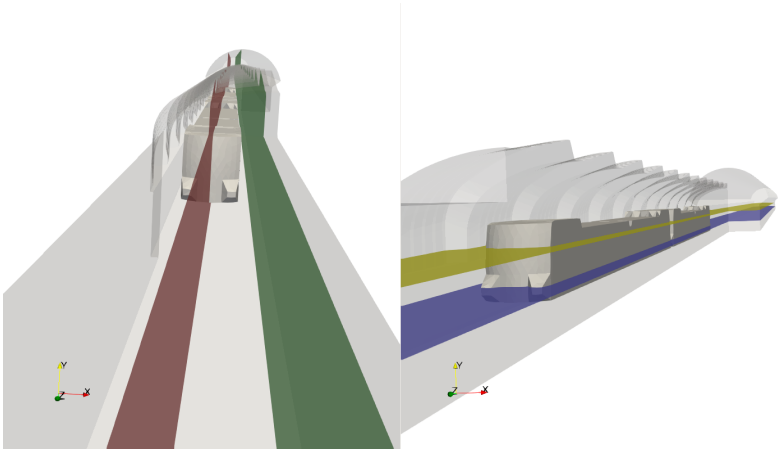


FIGURE 9. Geometry slices where results are presented.

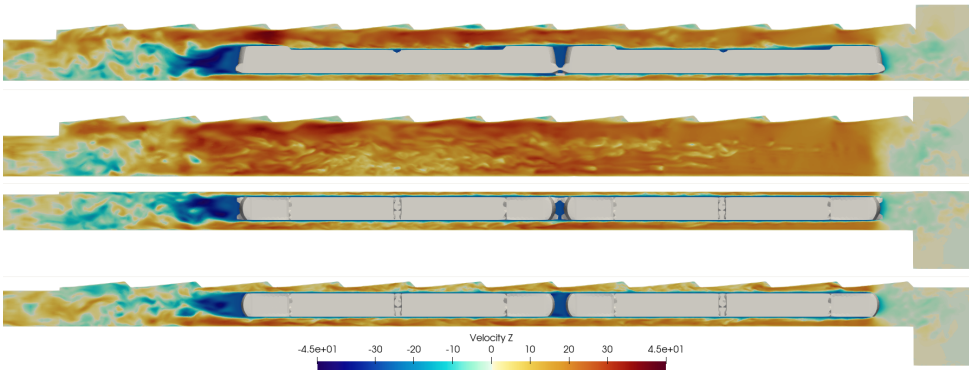


FIGURE 10. Slices of lengthwise velocity.

7.3. **Numerical results.** The results presented in this section are taken from an instant when the train is half way through the middle tunnel. The slicing planes used for the representation of results are shown in Fig. 9 in the seek of clarity.

As the train moves through the tunnel, it drags along a layer of air through shear stresses (see Fig. 10). The displaced air from ahead travels in the opposite direction in the gap between the train and the tunnel walls (Fig. 11). The problem is characterised by strong turbulent flow (Fig. 12). The flow detaches at the nose of the train creating vortices

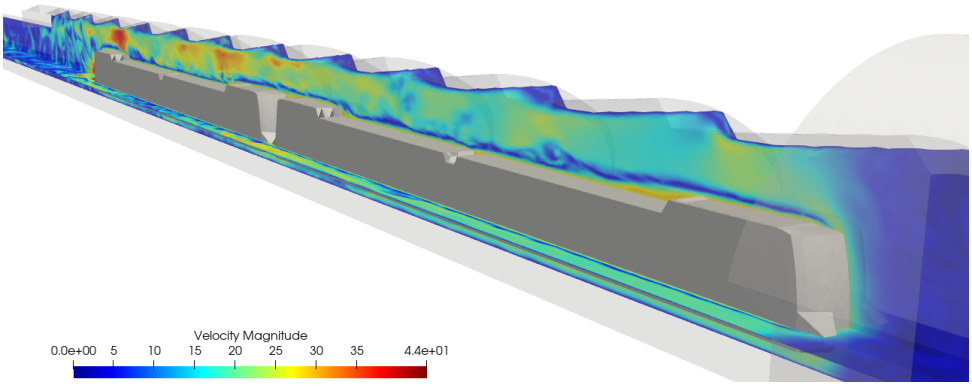


FIGURE 11. Velocity Magnitude in the tunnel.

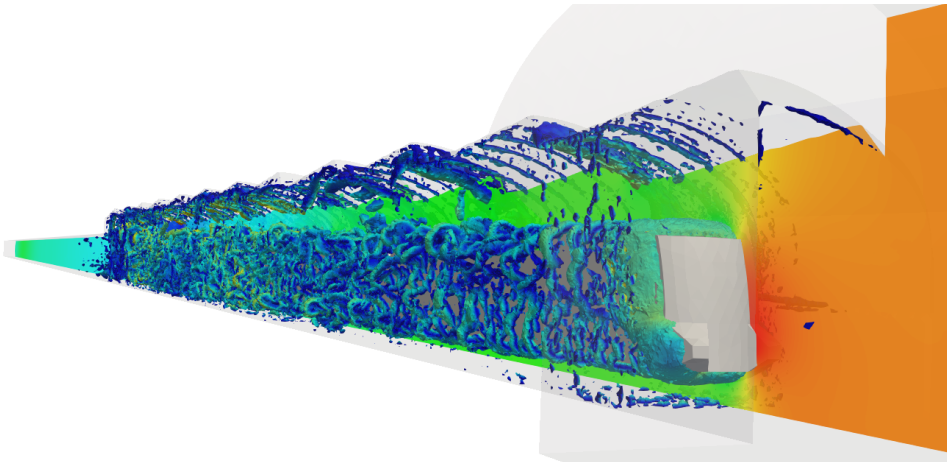


FIGURE 12. Vortical structures represented with Iso Q-Criterion volumes.

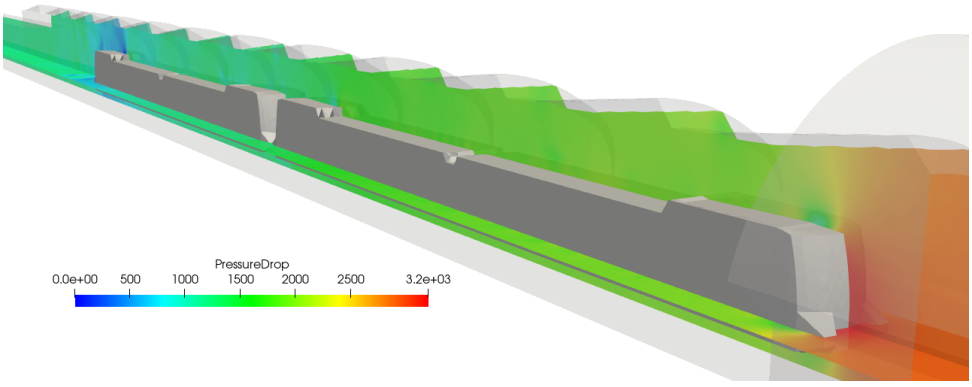


FIGURE 13. Pressure distribution in the tunnel.

on both sides. Secondary vortices develop all along the sides of the train. Additional vortices are generated when the flow hits the sharp edges of the tunnel walls. Turbulence is greatest at the tail of the train, where vortices are alternatively detached as air from the sides of the train is sucked. As a result, the train leaves behind a wake region which gradually dissipates downstream. As expected, the maximum pressure is found at the nose of the train and the strongest suction appears at the tail (see Fig. 13 and Fig. 14). There is a pressure drop along the sides of the train.

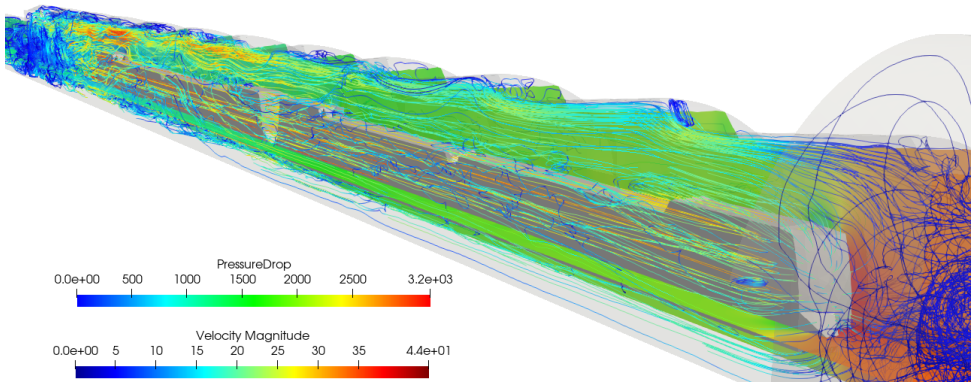


FIGURE 14. Flow streamlines around the train.

TABLE 1. Performace of the solver of the fractional step algorithm for a representative time step.

Problem [Picard iteration]	**Solver iterations**	**Solver Time**
Velocity [1]	140	18.3 s
Velocity [2]	135	17.5 s
Velocity [3]	136	17.6 s
Velocity [4]	141	18.1 s
Pressure Correction	574	31.5 s

The results show a fluid problem computation with a high level of detail. The fine discretisation allows for the resolution of a large range of turbulent scales, leading to an accurate capturing of the eddies and complex features of the flow. The implemented embedded methodology achieved the task of transferring the position and velocity of the train, enabling the resolution of a fully transient and complicated problem which would have been very difficult to solve with a single fixed mesh or with a classical ALE approach.

The adaptive mesh refinement (Fig. 8) also plays a significant role in the numerical computation. Firstly, it provides the background mesh with fine element sizes in the region close to the surface of the train. This optimises the imposition of boundary conditions, thus enhancing the precision of the embedded strategy. Secondly, it refines the region around the train where the most complex flow features appear. This guarantees a mesh resolution capable of capturing sharp gradients and turbulent phenomena, even if we have not aimed at fully capturing the boundary layer, while the regions further from the train where the flow is smooth and easily computed remain unrefined. As a result, the refinement strategy proves to be successful in increasing the accuracy of the numerical scheme and the computational efficiency.

7.4. **Numerical performance.** The computation of the whole problem with 224 CPUs took 35 h. 90% of the computation time was spent on the assembly and solving of the linear systems of the different steps of the incompressible Navier-Stokes fractional step algorithm. The velocity step non-linearity converged to the order of 10^{-3} with 4 Picard iterations.

In terms of solver of the linear systems, the parallel library PETSc was employed [47]. The iterative BCGs (Biconjugate Gradients Stabilized) [48] solver with ILU preconditioner was chosen for this problem. The convergence of the solver was stable, typically reaching a tolerance of the order of 10^{-6}. The linear solver of the fluid problem used a 73% of the total CPU time. The solver information for a characteristic step is found in Table 1.

It is interesting to notice the high number of iterations and time required for the pressure correction step compared to a velocity step. Had a monolithic algorithm been used instead of the proposed fractional step, the solver of the monolithic system would have taken at least the iterations required by the pressure correction step, creating a significant computation overhead. Considering that each nonlinear iteration would involve four variables per node instead of three (pressure plus three velocity components), and that the number iterations would be driven by the pressure (of the order of 574 instead of something between 135 and 141), the expected factor of CPU increase would be about 5.5.

It is interesting to point out that the adaptive refinement uses 1.8% of the computation time and the embedding operations take 1%. The computation overhead of both is negligible compared to solving of the incompressible Navier-Stokes systems. This justifies the use of the different ingredients of the methodology and proves its numerical efficiency.

8. Conclusions

In this paper we have presented a methodology to deal with large scale fluid flow problems in which the flow domain changes because of the motion of a solid moving inside a fluid. The novelties of the paper consist of the tailoring of different numerical techniques to this problem and also two specific new ingredients. The former include a stabilised FE method that can be used as implicit LES, the fixed-mesh ALE strategy, the weak imposition of boundary conditions in non-matching meshes, the artificial compressibility technique and the adaptive mesh refinement, whereas the latter are the stabilisation of badly cut elements and the design of the fractional step scheme. In all cases, the choice of these techniques is motivated by the physical needs and, above all, by efficiency considerations. This is why we favour for example fractional step schemes rather than monolithic ones and fixed mesh methods rather than moving meshes.

The resulting formulation is very robust and efficient. We have presented a numerical simulation with the objective of highlighting the convenience of the choices adopted and the good numerical performance of this formulation, both from the point of view of the quality of the results and from the computational efficiency.

Acknowledgements

R. Codina gratefully acknowledges the support received through the ICREA Acadèmia Research Program of the Catalan Government. I. Castañar gratefully acknowledges the support received from the Agència de Gestió d'Ajuts i de Recerca through the predoctoral FI grant 2019-FI-B-00649. CIMNE is a recipient of a "Severo Ochoa Programme for Centers of Excellence in R&D" grant (CEX2018-000797-S) by the Spanish Ministry of Economy and Competitiveness.

References

[1] J. Donea, A. Huerta, J.-P. Ponthot, and A. Rodríguez-Ferran, *Arbitrary Lagrangian–Eulerian Methods,* in Encyclopedia of Computational Mechanics (eds E. Stein, R. Borst and T.J.R. Hughes), pp. 1–25. John Wiley & Sons Ltd., 2004.

[2] S. Badia and R. Codina, "Analysis of a stabilized finite element approximation of the transient convection-diffusion equation using an ALE framework," *SIAM Journal on Numerical Analysis*, vol. 44, pp. 2159–2197, 2006.

[3] G. Houzeaux and R. Codina, "A finite element model for the simulation of lost foam casting," *International Journal for Numerical Methods in Fluids*, vol. 46, pp. 203–226, 2004.

[4] R. Codina, J. Houzeaux, H. Coppola-Owen, and J. Baiges, "The fixed-mesh ALE approach for the numerical approximation of flows in moving domains," *Journal of Computational Physics*, vol. 228, pp. 1591–1611, 2009.

[5] T. Hughes, "Multiscale phenomena: Green's function, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized formulations," *Computer Methods in Applied Mechanics and Engineering*, vol. 127, pp. 387–401, 1995.

[6] R. Codina, "Stabilized finite element approximation of transient incompressible flows using orthogonal subscales," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, pp. 4295–4321, 2002.
[7] R. Codina, J. Principe, O. Guasch, and S. Badia, "Time dependent subscales in the stabilized finite element approximation of incompressible flow problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 196, pp. 2413–2430, 2007.
[8] M.-C. Lai and C. Peskin, "An immersed boundary method with formal second-order accuracy and reduced numerical viscosity," *Journal of Computational Physics*, vol. 160, pp. 705–719, 2000.
[9] C. Peskin, "Flow patterns around heart valves: A numerical method," *Journal of Computational Physics*, vol. 10, pp. 252–271, 1972.
[10] 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*, vol. 85, pp. 109–128, 1991.
[11] A. Gerstenberger and W. Wall, "An embedded Dirichlet formulation for 3D continua," *International Journal for Numerical Methods in Engineering*, vol. 82, pp. 537–563, 2010.
[12] 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," *International Journal for Numerical Methods in Fluids*, vol. 30, pp. 1043–1066, 1999.
[13] J. Baiges, R. Codina, F. Henke, S. Shahmiri, and W. Wall, "A symmetric method for weakly imposing Dirichlet boundary conditions in embedded finite element meshes," *International Journal for Numerical Methods in Engineering*, vol. 90, pp. 636–658, 2012.
[14] 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*, vol. 104, pp. 624–654, 2015.
[15] E. Burman, "Projection stabilization of lagrange multipliers for the imposition of constraints on interfaces and boundaries," *Numerical Methods for Partial Differential Equations*, vol. 30, pp. 567–592, 2014.
[16] G. Barrenechea and F. Chouly, "A local projection stabilized method for fictitious domains," *Applied Mathematical Letters*, vol. 25, pp. 2071–2076, 2012.
[17] S. Amdouni, M. Moakher, and Y. Renard, "A local projection stabilization of fictitious domain method for elliptic boundary value problems," *Applied Numerical Mathematics*, vol. 76, pp. 60–75, 2014.
[18] M. Juntunen and R. Stenberg, "Nitsche's method for general boundary conditions," *Mathematics of Computation*, vol. 78, pp. 1353–1374, 2009.
[19] 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*, vol. 191, pp. 1895–1908, 2002.
[20] E. Burman and P. Hansbo, "Fictitious domain finite element methods using cut elements: II. A stabilized Nitsche method," *Applied Numerical Mathematics*, vol. 62, pp. 328–341, 2012.
[21] A. Main and G. Scovazzi, "The shifted boundary method for embedded domain computations. Part I: Poisson and Stokes problems," *Journal of Computational Physics*, vol. 372, pp. 972–995, 2018.
[22] S. Badia, A. F. Martín, E. Neiva, and F. Verdugo, "The aggregated unfitted finite element method on parallel tree-based adaptive meshes," *SIAM Journal on Scientific Computing*, vol. 43, pp. C203–C234, 2021.
[23] S. Badia and R. Codina, "Algebraic pressure segregation methods for the incompressible Navier–Stokes equations," *Archives of Computational Methods in Engineering*, vol. 15, pp. 1–52, 2007.
[24] A. Chorin, "Numerical solution of the Navier–Stokes equations," *Mathematics of Computation*, vol. 22, pp. 745–762, 1968.
[25] J.-L. Guermond and P. D. Minev, "High-order time stepping for the incompressible Navier–Stokes equations," *SIAM Journal on Scientific Computing*, vol. 37, pp. A2656 –A2681, 2015.
[26] E. Hachem, S. Feghali, R. Codina, and T. Coupez, "Anisotropic adaptive meshing and monolithic variational multiscale method for fluidstructure interaction," *Computers & Structures*, vol. 122, pp. 88–100, 2013. Computational Fluid and Solid Mechanics 2013.
[27] P. J. Frey and F. Alauzet, "Anisotropic mesh adaptation for cfd computations," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 48, pp. 5068–5082, 2005. Unstructured Mesh Generation.
[28] C. Dobrzynski and P. Frey, "Anisotropic delaunay mesh adaptation for unsteady simulations," in *Proceedings of the 17th International Meshing Roundtable* (R. V. Garimella, ed.), (Berlin, Heidelberg), pp. 177–194, Springer Berlin Heidelberg, 2008.
[29] R. Löhner, "Adaptive remeshing for transient problems with moving bodies," in *11th International Conference on Numerical Methods in Fluid Dynamics* (D. L. Dwoyer, M. Y. Hussaini, and R. G. Voigt, eds.), (Berlin, Heidelberg), pp. 379–383, Springer Berlin Heidelberg, 1989.

[30] J. Baiges and C. Bayona, "Refficientlib: An Efficient Load-Rebalanced Adaptive Mesh Refinement Algorithm for High-Performance Computational Physics Meshes," *SIAM Journal on Scientific Computing*, vol. 39, no. 2, pp. 65–95, 2017.
[31] D. Boffi and L. Gastaldi, "Stability and geometric conservation laws for ALE formulation," *Computer Methods in Applied Mechanics and Engineering*, vol. 193, pp. 4717–4739, 2004.
[32] M. Lesoinne and C. Farhat, "Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aerolastic computations," *Computer Methods in Applied Mechanics and Engineering*, vol. 134, pp. 71–90, 1996.
[33] H. Coppola-Owen and R. Codina, "A finite element model for free surface flows on fixed meshes," *International Journal for Numerical Methods in Fluids*, vol. 54, pp. 1151–1171, 2007.
[34] 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*, vol. 81, pp. 1529–1557, 2010.
[35] R. Codina, "A stabilized finite element method for generalized stationary incompressible flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 190, pp. 2681–2706, 2001.
[36] R. Codina, S. Badia, J. Baiges, and J. Principe, *Variational Multiscale Methods in Computational Fluid Dynamics,* in Encyclopedia of Computational Mechanics (eds E. Stein, R. Borst and T.J.R. Hughes), pp. 1–28. John Wiley & Sons Ltd., 2017.
[37] R. Codina, J. Principe, and J. Baiges, "Subscales on the element boundaries in the variational two-scale finite element method," *Computer Methods in Applied Mechanics and Engineering*, vol. 198, pp. 838–852, 2009.
[38] R. Codina, "Analysis of a stabilized finite element approximation of the Oseen equations using orthogonal subscales," *Applied Numerical Mathematics*, vol. 58, pp. 264–283, 2008.
[39] E. Castillo and R. Codina, "Dynamic term-by-term stabilized finite element formulation using orthogonal subgrid-scales for the incompressible Navier-Stokes problem," *Computer Methods in Applied Mechanics and Engineering*, vol. 349, pp. 701–721, 2019.
[40] O. Guasch and R. Codina, "Statistical behavior of the orthogonal subgrid scale stabilization terms in the finite element large eddy simulation of turbulent flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 261-262, pp. 154–166, 2013.
[41] O. Colomés, S. Badia, R. Codina, and J. Principe, "Assessment of variational multiscale models for the large eddy simulation of turbulent incompressible flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 285, pp. 32–63, 2015.
[42] R. Codina and S. Badia, "On the design of discontinuous Galerkin methods for elliptic problems based on hybrid formulations," *Computer Methods in Applied Mechanics and Engineering*, vol. 263, pp. 158–168, 2013.
[43] J. Baiges, R. Codina, A. Pont, and E. Castillo, "An adaptive fixed-mesh ALE method for free surface flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 313, pp. 159–188, 2017.
[44] R. Codina, "Pressure stability in fractional step finite element methods for incompressible flows," *Journal of Computational Physics*, vol. 170, pp. 112–140, 2001.
[45] G. Karypis, K. Schloegel, and V. Kumar, "Parmetis parallel graph partitioning and sparse matrix ordering library," 1997.
[46] Y. Bazilevs, C. Michler, V. Calo, and T. Hughes, "Weak Dirichlet boundary conditions for wall-bounded turbulent flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 196, pp. 4853–4862, 2007.
[47] 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, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, "PETSc Web page, http://www.mcs.anl.gov/petsc," 2015.
[48] H. A. V. der Vorst, "Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems," *SIAM, Journal of Scientific and Statistical Computing*, vol. 13, no. 2, pp. 631 – 644, 1992.