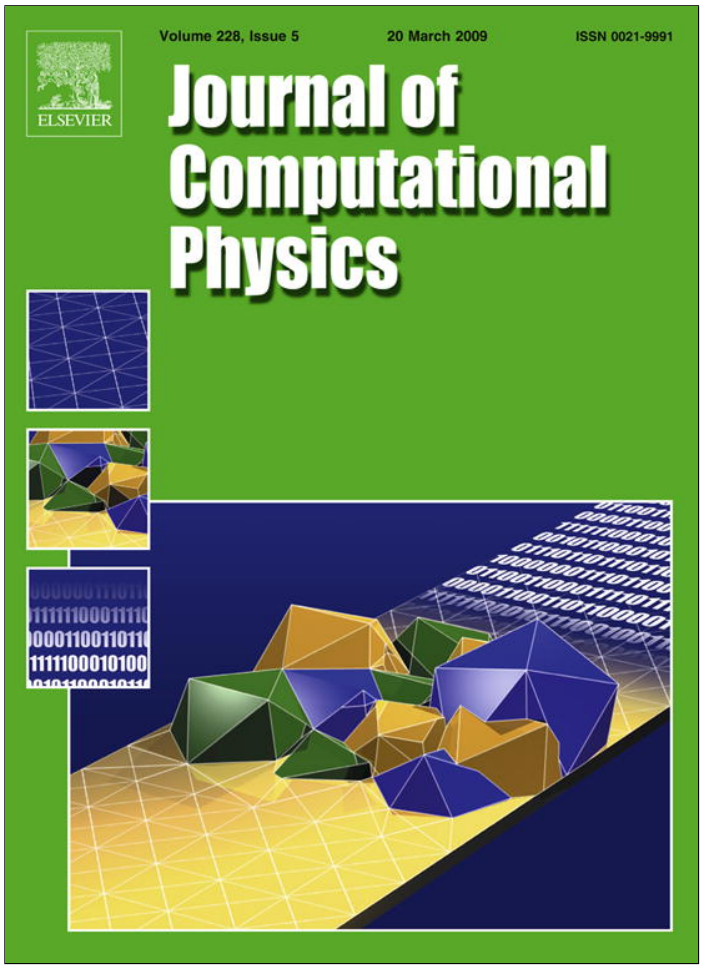

Journal of Computational Physics 228 (2009) 1591–1611

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

The fixed-mesh ALE approach for the numerical approximation of flows in moving domains

Ramon Codina [a,*], Guillaume Houzeaux [b], Herbert Coppola-Owen [a], Joan Baiges [a]

[a] International Center for Numerical Methods in Engineering (CIMNE), Universitat Politècnica de Catalunya, Jordi Girona 1-3, Edifici C1, 08034 Barcelona, Spain
[b] Barcelona Supercomputing Center, Jordi Girona 29, Edifici Nexus II, 08034 Barcelona, Spain

ARTICLE INFO

Article history:
Received 30 July 2008
Received in revised form 2 October 2008
Accepted 2 November 2008
Available online 20 November 2008

Keywords:
ALE
Immersed boundary methods
Approximate boundary conditions
Transmission conditions
Level set

ABSTRACT

In this paper we propose a method to approximate flow problems in moving domains using always a given grid for the spatial discretization, and therefore the formulation to be presented falls within the category of fixed-grid methods. Even though the imposition of boundary conditions is a key ingredient that is very often used to classify the fixed-grid method, our approach can be applied together with any technique to impose approximately boundary conditions, although we also describe the one we actually favor. Our main concern is to properly account for the advection of information as the domain boundary evolves. To achieve this, we use an arbitrary Lagrangian–Eulerian framework, the distinctive feature being that at each time step results are projected onto a fixed, background mesh, that is where the problem is actually solved.

1. Introduction

In many coupled problems of practical interest the domain of at least one of the problems evolves in time. The Arbitrary Eulerian Lagrangian (ALE) approach is a tool very often employed to cope with this domain motion. In this work we aim at describing a particular version of the ALE formulation that can be used in different coupled problems, and which we will apply to two problems in fluid mechanics.

In the classical ALE approach to solve problems in computational fluid dynamics, the mesh in which the computational domain is discretized is deformed (see for example [12,28,26]). This is done according to a prescribed motion of part of its boundary, which is transmitted to the interior nodes in a way as smooth as possible so as to avoid mesh distortion. In this work we present an ALE-type strategy with a different motivation. Instead of assuming that the computational domain is defined by the mesh boundary, we assume that there is a function that defines the boundary of the domain where the flow takes place. We will refer to it as the *boundary function*. It may be given, for example, by the shape of a body that moves within the fluid, or it may need to be computed, as in the case of level set functions described in the paper. It may be also defined discretely, by a set of points. When this boundary function moves, the flow domain changes, and that must be taken into account at the moment of writing the conservation equations that govern the flow, which need to be cast in the ALE format. However, our purpose here is to explain how to use always a background *fixed* mesh. That requires a virtual motion of the mesh nodes followed by a projection of the new node positions onto the fixed mesh.

The basic numerical formulation we will use consists of a stabilized finite element method to solve the ALE flow equations and finite difference time integration schemes. However, other discretization techniques could be applied, since the idea we

* Corresponding author.
E-mail address: ramon.codina@upc.edu (R. Codina).

0021-9991/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2008.11.004

want to expose is independent of the numerical method being used. This idea consists in projecting the results of the ALE deformed mesh onto a fixed background mesh *at each time step*, prior to solving the flow equations. It will be shown that at the end all the calculations can be performed on the fixed mesh, and in fact the ALE deformed mesh does not need to be explicitly built.

We want to stress that this idea is independent on the way to impose boundary conditions on the moving boundary. The way to impose this prescription is often used to classify a particular fixed-mesh method. Since the physical boundary is contained in the domain actually discretized, these methods are often called *immersed boundary methods*. Moreover, since the fixed-grid used is often Cartesian, these formulations can be found under the keywords *Cartesian grid methods* (see for example the reviews [42,36,35]). These methods are developed for constant-in-time domains, and then extended in a more or less ad-hoc way to time dependent domains. In spite of the fact that we want to distinguish between the way to deal with moving domains from the way of approximately imposing the boundary conditions on the moving boundary, we will briefly describe the particular approach we use.

The paper is organized as follows. A general overview of the FM–ALE method is presented in Section 2, starting with the discretization of the classical ALE formulation and then describing the algorithmic steps of the FM–ALE alternative. These steps are further elaborated in Section 3. Even though they are not intrinsic to the main idea of the method, there are three numerical ingredients that are essential for the success of the formulation. These are the definition and updating of the moving boundary, the approximate imposition of boundary conditions and the projection of data between two different finite element meshes. These "side ingredients" have been published before [9,7,23], but are here particularized to the FM–ALE method. They are described in Section 4. A simple numerical example, but containing all the features of the formulation, is presented in Section 5. We discuss then the application of the FM–ALE idea to two coupled problems of practical interest in Section 6. One is the simulation of lost foam casting [25]. In this case the flow is coupled to the heat equation because of the interface evolution, which is governed by the advance of the burning front of the molten metal used in the casting. Therefore, the boundary velocity is given and the normal stress on the fluid is unknown. The second problem considered is a classical free surface problem in which, once more, the free surface position is modeled by a level set function [11]. In this case, the velocity on the free surface is unknown, but the normal stress can be prescribed to zero (or to the atmospheric pressure). Some conclusions close the paper in Section 7.

2. The fixed-mesh ALE method

In this section we describe the essential idea of the FM–ALE method. However, we start with the classical ALE formulation of the incompressible Navier–Stokes equations and their numerical approximation.

2.1. The classical ALE method and its finite element approximation

2.1.1. Problem statement

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 $\Gamma_{\text{free}}(t) = \partial\Omega(t) \backslash \partial\Omega^0 \cap \partial\Omega(t)$. This moving part of $\partial\Omega(t)$ may correspond to the boundary of a moving solid immersed in the fluid or can be determined by a level set function, as we will see in the applications.

In order to cope with the time-dependency of $\Omega(t)$, we use the ALE approach, with the particular feature of considering a variable definition of the domain velocity. Let $\boldsymbol{\mathcal{X}}_t$ be a family of invertible mappings, which for all $t \in [0,T]$ map a point $\boldsymbol{X} \in \Omega(0)$ to a point $\boldsymbol{x} = \boldsymbol{\mathcal{X}}_t(\boldsymbol{X}) \in \Omega(t)$, with $\boldsymbol{\mathcal{X}}_0 = \boldsymbol{I}$, the identity. If $\boldsymbol{\mathcal{X}}_t$ is given by the motion of the particles, the resulting formulation would be Lagrangian, whereas if $\boldsymbol{\mathcal{X}}_t = \boldsymbol{I}$ for all t, $\Omega(t) = \Omega(0)$ and the formulation would be Eulerian.

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

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

Given a function $f : \Omega(t) \times (0,T) \to \mathbb{R}$ we define

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

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 $\boldsymbol{u} : \Omega(t) \times (0,T) \to \mathbb{R}^d$ and a pressure $p : \Omega(t) \times (0,T) \to \mathbb{R}$ such that

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

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

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

Initial and boundary conditions have to be appended to problem (2) and (3). In the applications we have considered in Section 6, the boundary conditions on $\Gamma_{\mathrm{free}}(t)$ can be of two different types: (a) Free surface flows: p (or the normal stress) given, $\boldsymbol{u}$ unknown on Γ_{free}; (b) Lost foam casting: $\boldsymbol{u}$ given, p (or the normal stress) unknown on Γ_{free}. On the rest of the boundary of $\Omega(t)$ the usual boundary conditions can be considered. In general, we consider these boundary conditions of the form

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

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 $\Gamma_D\cup\Gamma_N=\partial\Omega$, and therefore time dependent.

2.1.2. The time-discrete problem

Let us start introducing some notation. 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

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

Even though other options are obviously possible, we will use the simple trapezoidal rule to discretize problem (2) and (3) in time. 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

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

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

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

$$\boldsymbol{u}^{n+\theta}_{\mathrm{dom}}=\frac{1}{\theta\delta t}(\mathcal{X}_{t^{n+\theta},t^n}(\boldsymbol{x}^n)-\boldsymbol{x}^n). \tag{6}$$

Note that the order of accuracy of this approximation is consistent with the order of accuracy of (4) and (5), that is to say, it is 2 for $\theta=1/2$ and 1 otherwise. We are interested only in the cases $\theta=1/2$ and $\theta=1$ (implicit schemes are required).

Remark 1. The trapezoidal rule considered for the time integration, with a single mesh, satisfies the so called *geometric conservation law* (GCL) condition (see, e.g. [3,15,32]). However, there are second order accurate schemes based on multi-step time discretizations that do not satisfy it. The price to be paid is that these schemes are usually only conditionally stable, although stability conditions are often very mild and not encountered in practice (see for example the analyses in [1,3,15,16,38]). We will use one of such schemes in Section 5.

2.1.3. The fully discrete problem

The next step is to consider the spatial discretization of problem (4) and (5). As for the time discretization, different options are possible. Here we simply describe the stabilized finite element formulation employed in our numerical simulations.

Let $\{\Omega^e\}^{n+1}$ be a finite element partition of the domain $\Omega(t^{n+1})$, with index e ranging from 1 to the number of elements n_{el} (which may be different at different time steps). We denote with a subscript h the finite element approximation to the unknown functions, and by $\boldsymbol{v}_h$ and q_h the velocity and pressure test functions associated to $\{\Omega^e\}^{n+1}$, respectively.

An important point is that we are interested in using equal interpolation for the velocity and the pressure. Therefore, the corresponding finite element spaces are assumed to be built up using the standard continuous interpolation functions.

In order to overcome the numerical problems of the standard Galerkin method, a stabilized finite element formulation is applied. This formulation is presented in [5]. It is based on the subgrid scale concept introduced in [27], although when linear elements are used it reduces to the Galerkin/least-squares method described for example in [17]. We apply this stabilized formulation together with the finite difference approximation in time (4) and (5).

The bottom line of the method is to test the continuous equations by the standard Galerkin test functions plus perturbations that depend on the operator representing the differential equation being solved. In our case, this operator corresponds

to the linearized form of the time discrete Navier–Stokes Eqs. (4) and (5). In this case, the method consists of finding $\boldsymbol{u}_h^{n+1}$ and p_h^{n+1} such that

$$m_1^{n+\theta}(\delta_t \boldsymbol{u}_h^{n+1}|_{\boldsymbol{x}^n}, \boldsymbol{v}_h) + a^{n+\theta}(\boldsymbol{u}_h, \boldsymbol{v}_h) + c^{n+\theta}(\boldsymbol{u}_h - \boldsymbol{u}_{\mathrm{dom}}; \boldsymbol{u}_h, \boldsymbol{v}_h) + b_1^{n+\theta}(p_h, \boldsymbol{v}_h) = l_1^{n+\theta}(\boldsymbol{v}_h), \tag{7}$$

$$m_2^{n+\theta}(q_h, \delta_t \boldsymbol{u}_h^{n+1}|_{\boldsymbol{x}^n}) + b_2^{n+\theta}(q_h, \boldsymbol{u}_h) + s^{n+\theta}(q_h, p_h) = l_2^{n+\theta}(q_h), \tag{8}$$

for all test functions $\boldsymbol{v}_h$ and q_h, the former vanishing on the Dirichlet part of the boundary Γ_D. The different forms appearing in these equations are given by

$$m_1(\delta_t \boldsymbol{u}_h, \boldsymbol{v}_h) = \int_\Omega \boldsymbol{v}_h \cdot \rho \delta_t \boldsymbol{u}_h + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \boldsymbol{\zeta}_{u1} \cdot \rho \delta_t \boldsymbol{u}_h,$$

$$a(\boldsymbol{u}_h, \boldsymbol{v}_h) = \int_\Omega 2\nabla^S \boldsymbol{v}_h : \mu \nabla^S \boldsymbol{u}_h + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \boldsymbol{\zeta}_{u1} \cdot (-2\nabla \cdot (\mu \nabla^S \boldsymbol{u}_h)) + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \zeta_{u2} \nabla \cdot \boldsymbol{u}_h,$$

$$c(\boldsymbol{a}; \boldsymbol{u}_h, \boldsymbol{v}_h) = \int_\Omega \boldsymbol{v}_h \cdot (\rho \boldsymbol{a} \cdot \nabla \boldsymbol{u}_h) + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \boldsymbol{\zeta}_{u1} \cdot (\rho \boldsymbol{a} \cdot \nabla \boldsymbol{u}_h),$$

$$b_1(p_h, \boldsymbol{v}_h) = -\int_\Omega p_h \nabla \cdot \boldsymbol{v}_h + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \boldsymbol{\zeta}_{u1} \cdot \nabla p_h,$$

$$m_2(q_h, \delta_t \boldsymbol{u}_h) = \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \zeta_p \cdot \rho \delta_t \boldsymbol{u}_h,$$

$$b_2(q_h, \boldsymbol{u}_h) = \int_\Omega q_h \nabla \cdot \boldsymbol{u}_h + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \zeta_p \cdot (\rho \boldsymbol{a} \cdot \nabla \boldsymbol{u}_h - 2\nabla \cdot (\mu \nabla^S \boldsymbol{u}_h)),$$

$$s(q_h, p_h) = \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \zeta_p \cdot \nabla p_h,$$

$$l_1(\boldsymbol{v}_h) = \int_\Omega \boldsymbol{v}_h \cdot \boldsymbol{f} + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \boldsymbol{\zeta}_{u1} \cdot \boldsymbol{f} + \int_{\Gamma_{\mathrm{N}}} \boldsymbol{v}_h \cdot \bar{\boldsymbol{t}},$$

$$l_2(q_h) = \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \zeta_p \cdot \boldsymbol{f},$$

where the functions $\boldsymbol{\zeta}_{u1}$, ζ_{u2} and ζ_p are computed within each element as

$$\boldsymbol{\zeta}_{u1} = \tau_u [\rho(\boldsymbol{u}_h - \boldsymbol{u}_{\mathrm{dom}}) \cdot \nabla \boldsymbol{v}_h + 2\nabla \cdot (\mu \nabla^S \boldsymbol{v}_h)], \tag{9}$$

$$\zeta_{u2} = \tau_p \nabla \cdot \boldsymbol{v}_h, \tag{10}$$

$$\boldsymbol{\zeta}_p = \tau_u \nabla q_h, \tag{11}$$

and the parameters τ_u and τ_p are also computed element-wise as (see [6])

$$\tau_u = \left[\frac{4\mu}{h^2} + \frac{2\rho \mid \boldsymbol{u}_h - \boldsymbol{u}_{\mathrm{dom}} \mid}{h}\right]^{-1}, \quad \tau_p = 4\mu + 2\rho \mid \boldsymbol{u}_h - \boldsymbol{u}_{\mathrm{dom}} \mid h,$$

where h is the element size for linear elements and half of it for quadratics.

Remark 2.

- The superscript $n+\theta$ in all the terms in (7) and (8) indicates that all the forms are evaluated with the unknowns at $n+\theta$, except for the term coming from the temporal derivative, whose superscript is explicitly indicated. Likewise, the integrals are evaluated at $\Omega(t^{n+\theta})$.
- The dependency on the advection velocity $\boldsymbol{a} = \boldsymbol{u}_h - \boldsymbol{u}_{\mathrm{dom}}$ has been only indicated in the form coming directly from the convective term of the equations, namely, $c(\boldsymbol{a}; \boldsymbol{u}_h, \boldsymbol{v}_h)$. However, it has to be noted that all the forms listed above depend on the stabilization parameters, and therefore depend on $\boldsymbol{a}$ as well. Moreover, the dependency of $b_2(q_h, \boldsymbol{u}_h)$ on $\boldsymbol{a}$ is even more explicit. However, in order to keep the notation more concise only the above mentioned dependency of $c(\boldsymbol{a}; \boldsymbol{u}_h, \boldsymbol{v}_h)$ has been left.
- As usual, the mesh of $\Omega(t^{n+1})$ is assumed to be obtained from the mesh of $\Omega(t^n)$ by moving the nodes of the latter with the domain velocity $\boldsymbol{u}_{\mathrm{dom}}$ (often referred to as mesh velocity). This greatly simplifies the implementation of the ALE method, since in this case the nodal values of $\boldsymbol{u}^{n+1}(\boldsymbol{x})$ and those of $\boldsymbol{u}^n(\boldsymbol{x}^n)$ correspond to the same nodes (at time steps $n+1$ and n, respectively).
- If $\theta = 1/2$, the unknowns of the problem can be taken as $\boldsymbol{u}^{n+1/2}$ and $p^{n+1/2}$, since $\delta_t \boldsymbol{u}_h^{n+1}|_{\boldsymbol{x}^n} = 2\delta t^{-1}(\boldsymbol{u}^{n+1/2}(\boldsymbol{x}) - \boldsymbol{u}^n(\boldsymbol{x}^n))$. All the calculations to be performed are the same as for $\theta = 1$, with the only modification that once $\boldsymbol{u}^{n+1/2}$ is computed $\boldsymbol{u}^{n+1}$ has to be updated to go to the next time step. This analogy includes the updating of the computational domain. When $\theta = 1/2$ we need to update this domain from $n - 1/2$ to $n + 1/2$ to compute $\boldsymbol{u}^{n+1/2}$ and $p^{n+1/2}$, whereas when $\theta = 1$ we need to update it from n to $n+1$ to compute $\boldsymbol{u}^{n+1}$ and p^{n+1}. For conciseness, the latter situation is considered in the following.

- From (9) and (11) it is observed that these terms are precisely the adjoints of the (linearized) operators of the differential equations to be solved applied to the test functions (observe the sign of the viscous term in (9)). This method corresponds to the algebraic version of the subgrid scale approach [27] and circumvents the stability problems of the Galerkin method. In particular, in this case it is possible to use equal velocity pressure interpolations, that is, we are not tied to the satisfaction of the inf-sup stability condition. For more details about this formulation, see for example [27,5].
- In order to simplify a little bit the statement of the problem, we will consider that $m_2(q_h,\delta_t \boldsymbol{u}_h) = 0$. This can be justified by using a space-time finite element method with constant-in-time interpolation (in the case $\theta = 1$) or by using the orthogonal-subscale stabilization (OSS) method (see [6]).

2.2. The fixed-mesh ALE approach: algorithmic steps

The purpose of this subsection is to give an overview of the FM–ALE method and to describe the main idea, leaving for the next section a more detailed description of the different steps involved.

Suppose Ω^0 is meshed with a finite element mesh M^0 and that at time level t^n the domain $\Omega(t^n)$ is meshed with a finite element mesh M^n (as we will see, close to M^0). Let $\boldsymbol{u}^n$ be the velocity already computed on $\Omega(t^n)$. The purpose is to obtain the fluid region $\Omega(t^{n+1})$ and the velocity field $\boldsymbol{u}^{n+1}$. The former may move according to a prescribed kinematics, for example due to the motion of a solid, or can be an unknown of the problem, as in the two applications we will describe in Section 6. If the classical ALE method is used, M^n would deform to another mesh defined at t^{n+1}. The key idea is not to use this mesh to compute $\boldsymbol{u}^{n+1}$ and p^{n+1}, but to re-mesh in such a way that the new mesh is, essentially, M^0 once again.

The steps of the algorithm to achieve the goal described are the following:

1. Define $\Gamma^{n+1}_{\text{free}}$ by updating the function that defines it.
2. Deform *virtually* the mesh M^n to M^{n+1}_{virt} using the classical ALE concepts and compute the mesh velocity $\boldsymbol{u}^{n+1}_{dom}$.
3. Write down the ALE Navier–Stokes equations on M^{n+1}_{virt}.
4. *Split the elements* of M^0 cut by $\Gamma^{n+1}_{\text{free}}$ to define a mesh on $\Omega(t^{n+1})$, M^{n+1}.
5. *Project* the ALE Navier–Stokes equations from M^{n+1}_{virt} to M^{n+1}.
6. Solve the equations on M^{n+1} to compute $\boldsymbol{u}^{n+1}$ and p^{n+1}.

In Section 3 we describe all these steps in detail. A global idea of the meshes involved in the process is represented in Fig. 1. Note in particular that at each time steps two sets of nodes have to be appropriately dealt with, namely, the so called newly created nodes and the boundary nodes. Contrary to other fixed-grid methods, some of which are described in the next subsection, newly created nodes are treated in a completely natural way using the FM–ALE approach: the value of the velocity there is directly given by the projection step from M^{n+1}_{virt} to M^{n+1}. Boundary nodes require either additional unknowns with respect to those of mesh M^0 or an appropriate imposition of boundary conditions. This issue is treated in Section 4.

2.3. Other fixed-grid methods

Other possibilities to use a single grid in the whole simulation can be found in the literature, each one having advantages and drawbacks. As the method presented in this paper, they were designed as an alternative to body fitted meshes and are sometimes referred to as *Embedded Mesh Methods*. They can be divided into two main groups [8], corresponding in fact to two ways of prescribing the boundary conditions on Γ_{free}:

- *Force term methods*. The interaction of the fluid and the solid is taken into account through a force term, which appears either in the strong or in the weak form of the flow equations. Therefore, the boundary conditions on Γ_{free} are neither imposed as Dirichlet nor as Neumann boundary conditions. Among this type of methods, let us cite for example the Immersed Boundary method as a variant of the Penalty method, where punctual forces are added to the momentum equation, and the Fictitious Domain method, where the solid boundary conditions are imposed through a Lagrange multiplier.
- *Approximate boundary conditions*. Instead of adding a force term, these methods impose the boundary conditions in an approximate way once the discretization has been carried out, either by modifying the differential operators near the interface (in finite differences) or by modifying the unknowns near the interface.

The *Immersed Boundary Method* in its original form [40] consists in adding punctual penalty forces in the domain boundary so that the boundary conditions are fulfilled. The forces are computed from a fluid-structure (elastic) interaction problem at the interface. The method is first order accurate even if second order approximation schemes are used, although *formal second order accuracy* has been reported in [31]. The more recent *Immersed Interface Method* achieves higher order accuracy by avoiding the use of the Dirac delta distribution to define the forcing terms (see [33,34,44]).

The *Penalty method* is similar to the previous one in the sense that a force term is added to the momentum equations. The difference raises in the fact that the penalty parameter is not computed from a fluid-structure interaction as in the original immersed boundary method, but it is simply required to be large enough to enforce the boundary conditions approximately. The force terms can be of two types, depending on whether they are imposed as boundary or as volume forces [43].

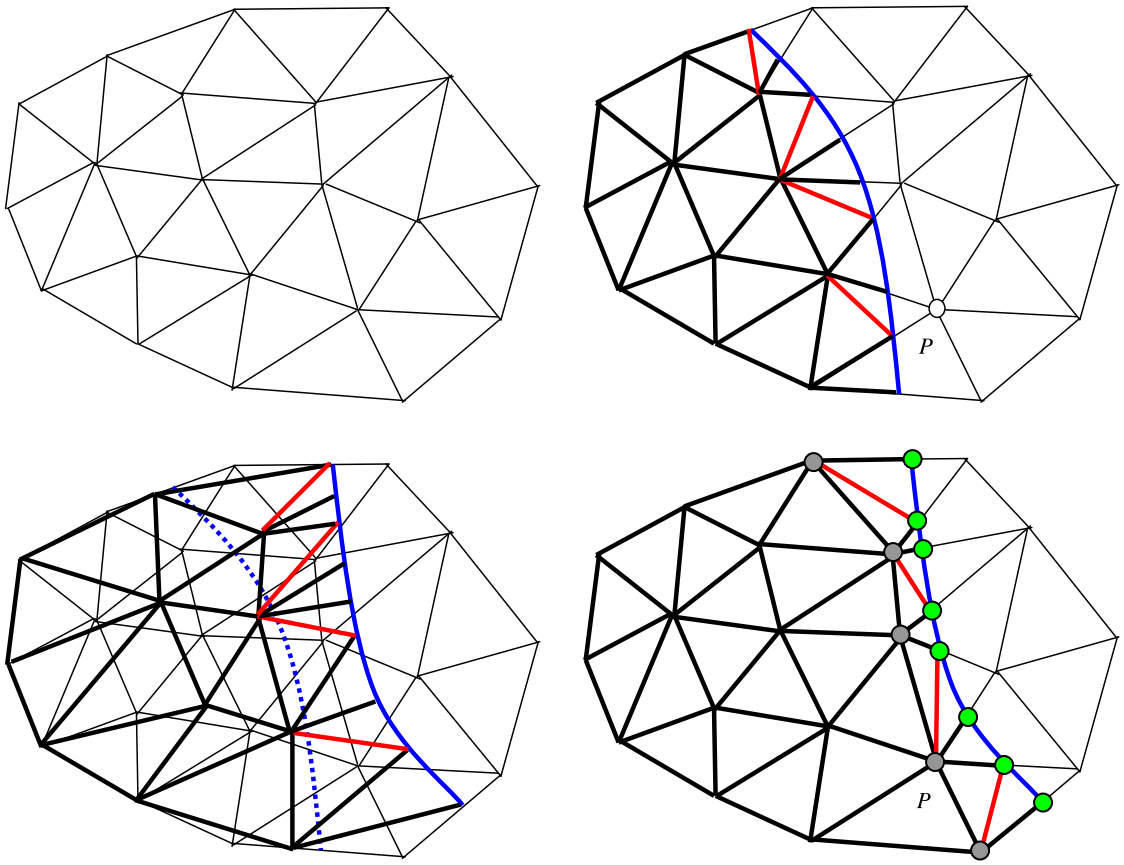

Fig. 1. Two dimensional FM–ALE schematic. Top-left: original finite element mesh M^0 of Ω^0. Top-right: finite element mesh M^n of $\Omega(t^n)$, with the elements represented by a thick line and the elements of M^0 represented by thin line. The blue line represents Γ^n_{free} and the red edges indicate the splitting of M^0 to obtain M^n. Bottom-left: updating of M^n to M^{n+1}_{virt} using the classical ALE strategy. The position of $\Gamma^{n+1}_{\text{free}}$ is again shown using a solid blue line and the previous position Γ^n_{free} using a dotted blue line. Bottom-right: Mesh M^{n+1} of $\Omega(t^{n+1})$, represented by a thick line. The edges that split elements of M^0 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 gray.

Another approach is the use of Lagrange multipliers to enforce the boundary conditions. However, the finite element subspaces for the bulk and Lagrange multiplier fields must satisfy the classical inf-sup condition, which usually leads to the need for stabilization (see [22,2,29]). Moreover, additional degrees of freedom must be added to the problem. The use of Lagrange multipliers is the basis of the *Fictitious Domain Method* [19,20].

Recently, *hybrid Cartesian/immersed boundary methods* have been developed for Cartesian grids, which use the grid nodes closest to the boundary to enforce boundary conditions [18,45,37]. The method is second order accurate.

Most of these methods have been well tested in the literature for both steady and moving interfaces. Generally, the last case is treated by applying directly the former at each time step. However, very few authors have described the full formulation for moving interfaces, sometimes simply by ignoring the problem. The fact that the boundary moves and the subsequent advection of unknowns is often not taken into account.

To explain an obvious consequence of the boundary motion, let us discuss the treatment of the newly created nodes. To explain the problem, let us consider point P in Fig. 1. Suppose that the boundary Γ_{free} corresponds in this case to the rigid boundary of a moving object. Physically, it is clear that the solution in the fluid cannot depend on what happens inside the solid. Mathematically, this means that the values of the unknowns at the fluid nodes are uncoupled from those at the solid nodes. Therefore, the velocity and the pressure at the solid nodes (apart from those participating to the enforcing of the boundary conditions) can be *whatever* at a certain time step n, in particular their value at node P (see Fig. 1, top-right). Now we move on to the next time step $n+1$ as the solid moves. Some solid nodes can therefore become fluid nodes, such as node P (see Fig. 1, bottom-right). The velocity at this node at time step n is in fact needed in the temporal term of the momentum equations and cannot be *whatever*. In the case of fractional step techniques, the situation can even be worse as the previous time step pressure could also be needed at these nodes.

A special treatment is needed for the newly created fluid nodes. In many publications, the previous time step values are computed using ad-hoc arguments, that sometimes lead to good approximations from the practical point of view when small time steps are used. As an example, in [35] the authors extrapolate the velocity and pressure from the nearest fluid nodes at the previous time step. It is worth to note that if the solid is deformable and has been solved together with the fluid in a coupled way (as in the original immersed boundary method [40] or in the fluid–solid approach in [46]), this velocity is phys-

ically meaningful. This is not the case, however, in the case of rigid bodies or bodies with rigid boundaries. A possibility to deal with this situation is to write the Navier–Stokes equations in a non-inertial frame of reference attached to the body, as in [24] in the context of Chimera meshes or in [30], where an immersed boundary method is used.

We explain in the following what we believe is a consistent way of treating moving interfaces based on a fixed-mesh ALE approach.

3. Developing the fixed-mesh ALE method

In this section we describe the steps enumerated previously, concentrating on those specific of the FM–ALE method and leaving for Section 4 those that can be considered side numerical ingredients.

3.1. Step 1. Boundary function update

This step is completely problem dependent. The motion of $\Gamma_{\text{free}}(t)$ may be determined by different ways. In a typical fluid-structure interaction problem, $\Gamma_{\text{free}}(t)$ will be part of the solid boundary, and therefore its kinematics will be determined by the dynamics of the solid under the action exerted by the fluid. As a particular case, the motion of the solid boundary may be directly prescribed. This is the simplest situation and the one corresponding to the validating numerical example presented in Section 5.

In a wide variety of applications, $\Gamma_{\text{free}}(t)$ may be represented by a level set function. In Section 6 we will describe two of these applications. The peculiarities of the level set function update in the context of the FM–ALE approach are described in Section 4.

3.2. Step 2. Mesh velocity

Updating the boundary function defines the deformation of the domain from $\Omega(t^n)$ to $\Omega(t^{n+1})$ (recall that we are considering the case $\theta = 1$, see Remark 2). Consequently, the mesh M^n used at time step n has to be deformed to adapt to the domain $\Omega(t^{n+1})$. This mesh deformation has to be defined by means of a mesh velocity.

The mesh velocity at the boundary points can be computed from their position $\boldsymbol{x}_b^{n+1}$ and $\boldsymbol{x}_b^n$, where subscript b refers to points on Γ_{free}. Using approximation (6), this mesh velocity would be $\boldsymbol{u}_{\text{dom},b}^{n+1} = (\boldsymbol{x}_b^{n+1} - \boldsymbol{x}_b^n)/\delta t$. Once the velocity at the nodes of Γ_{free} is known, it has to be extended to the rest of the nodes. A classical possibility is to solve the Laplace problem $\Delta \boldsymbol{u}_{\text{dom}} = \boldsymbol{0}$ using $\boldsymbol{u}_{\text{dom},b}^{n+1}$ as Dirichlet boundary conditions. However, it is also possible to restrict $\boldsymbol{u}_{\text{dom}} \neq \boldsymbol{0}$ to the nodes next to $\Gamma_{\text{free}}^{n+1}$, since in our approach mesh distortion does not accumulate from one time step to another (see Fig. 1 for a schematic of the mesh deformation). This is in practice what we do. Details about this point are provided in the numerical examples of Sections 5 and 6.

3.3. Step 3. Solving the flow equations I: Equations on the deformed mesh

The previous procedure defines the domain $\Omega(t^{n+1})$ and a mesh that we call M_{virt}^{n+1}, obtained from a deformation of the mesh M^n. The equations to be solved there are (see (7,8)):

$$m_1^{n+1}\left(\frac{1}{\delta t}(\boldsymbol{u}_{h,\text{virt}}^{n+1}(\boldsymbol{x}) - \boldsymbol{u}_{h,\text{virt}}^{n}(\boldsymbol{x}^n)), \boldsymbol{v}_h\right) + a^{n+1}(\boldsymbol{u}_{h,\text{virt}}, \boldsymbol{v}_h) + c^{n+1}(\boldsymbol{u}_{h,\text{virt}} - \boldsymbol{u}_{\text{dom,virt}}; \boldsymbol{u}_{h,\text{virt}}, \boldsymbol{v}_h) + b_1^{n+1}(p_{h,\text{virt}}, \boldsymbol{v}_h) = l_1^{n+1}(\boldsymbol{v}_h), \tag{12}$$

$$b_2^{n+1}(q_h, \boldsymbol{u}_{h,\text{virt}}) + s^{n+1}(q_h, p_{h,\text{virt}}) = l_2^{n+1}(q_h), \tag{13}$$

where subscript "virt" refers to the mesh M_{virt}^{n+1} on which these equations should now be solved using the space discretization described in Subsection 2.1.3. Let us stress once again that, as it is well known in the classical ALE approach, $\boldsymbol{u}^n(\boldsymbol{x}^n)$ is known on M_{virt}^{n+1} because the nodes of this mesh are obtained from the motion of the nodes of M^n with the mesh velocity $\boldsymbol{u}_{\text{dom,virt}}^{n+1}$.

3.4. Step 4. Splitting of elements

The key idea of the FM–ALE method is *not to use* M_{virt}^{n+1} to solve the flow equations at time t^{n+1}, but to use instead another mesh M^{n+1} that will be a *a minor modification of the background mesh* M^0. This mesh M^{n+1} is obtained by splitting the elements of M^0 cut by $\Gamma_{\text{free}}^{n+1}$, as shown in Fig. 1. Meshes M^{n+1} and M^0 only differ in the subelements created after the splitting just mentioned.

Mesh M^{n+1} could be thought as a local refinement of mesh M^0 to make it conform the boundary $\Gamma_{\text{free}}^{n+1}$. This is certainly a possibility that can be implemented as such. Let us note however that this requires the introduction of boundary nodes at each step, as shown in Fig. 1, and the subsequent change in the mesh graph and in the sparsity pattern of the matrix of the final algebraic system to be solved for the arrays of nodal unknowns. As in other fixed grid methods, this computational complication can be avoided *by prescribing boundary conditions on* $\Gamma_{\text{free}}^{n+1}$ *in an approximate way*. Nevertheless, this issue, in spite of its major practical importance, is not an essential concept of the FM–ALE method, and we defer its description to Section 4.

The local refinement from M^0 to M^{n+1} is needed also to perform the numerical integration of the different terms appearing in (7) and (8). Obviously, the impact of this in the computational cost of the overall calculation is minimum.

The splitting of elements is a strictly algorithmic step that shall not be discussed here. In the case of 2D linear elements, Fig. 2 shows how the splitting can be done and the numerical integration points (red points) required in each triangle resulting from this splitting.

3.5. Step 5. Solving the flow equations II: Equations on the background mesh

Let P^{n+1} be the projection of finite element functions defined on M^{n+1}_{virt} to M^{n+1}. To define it, for each node of M^{n+1} the element in M^{n+1}_{virt} where it is placed has to be identified. Once this is done, the value of any unknown at this node can be obtained through interpolation, possibly with restrictions. The way to construct this projection operator is a problem common to different situations in which transfer of information between finite element meshes is required. We describe our approach in Section 4.

The velocity $\boldsymbol{u}^n$ in M^{n+1}_{virt} is known because its nodal values correspond to those of mesh M^n. However, its nodal values on M^{n+1} have to be computed using the projection just described. The same happens with the mesh velocity $\boldsymbol{u}_{\text{dom}}$.

If now we define

$$\boldsymbol{u}_h^{n+1} := P^{n+1}\left(\boldsymbol{u}_{h,\text{virt}}^{n+1}\right),$$

the problem to be solved at time step $n+1$ is to find a velocity $\boldsymbol{u}_h^{n+1}$ and a pressure p_h^{n+1} such that

$$m_1^{n+1}(\delta t^{-1}(\boldsymbol{u}_h^{n+1}(\boldsymbol{x}) - P^{n+1}(\boldsymbol{u}_{h,\text{virt}}^n(\boldsymbol{x}^n))), \boldsymbol{v}_h) + a^{n+1}(\boldsymbol{u}_h, \boldsymbol{v}_h) + c^{n+1}(\boldsymbol{u}_h - P^{n+1}(\boldsymbol{u}_{\text{dom,virt}}); \boldsymbol{u}_h, \boldsymbol{v}_h) + b_1^{n+1}(p_h, \boldsymbol{v}_h) = l_1^{n+1}(\boldsymbol{v}_h), \tag{14}$$

$$b_2^{n+1}(q_h, \boldsymbol{u}_h) + s^{n+1}(q_h, p_h) = l_2^{n+1}(q_h), \tag{15}$$

which again must hold for all velocity test functions $\boldsymbol{v}_h$ and pressure test functions q_h.

Note that $p_h^{n+1} \neq P^{n+1}\left(p_{h,\text{virt}}^{n+1}\right)$. Pressure p_h^{n+1} is determined by imposing that $\boldsymbol{u}_h^{n+1}$ is divergence free, which at the discrete level is not equivalent to impose that $\boldsymbol{u}_{h,\text{virt}}^{n+1}$ is divergence free.

Problem (14) and (15) is posed on M^{n+1} which, as it has been said, coincides with M^0 except for the splitting of the elements crossed by the interface. Even this difference can be avoided if instead of prescribing exactly the boundary conditions an approximation is performed, for example using Nitsche's method, Lagrange multipliers or the strategy described in Section 4. Therefore, the goal of using a fixed mesh during the whole simulation has been achieved.

It is observed that the projection P^{n+1} has to be applied to

- $P^{n+1}(\boldsymbol{u}_{h,\text{virt}}^n(\boldsymbol{x}^n))$. This clarifies the effect of the mesh motion in the context of fixed-mesh methods. In particular, there is no doubt about the velocity at previous time steps of newly created nodes.
- $P^{n+1}\left(\boldsymbol{u}_{\text{dom,virt}}^{n+1}\right)$. The mesh velocity is computed on M^{n+1}_{virt}, and therefore needs to be projected to compute on M^{n+1}.

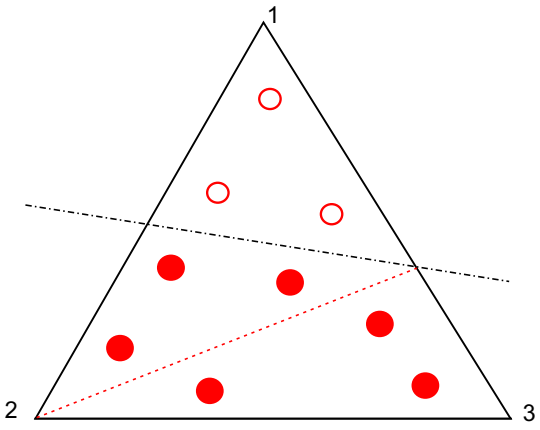

Fig. 2. Splitting of elements.

3.6. Comparison with the classical ALE approach

To conclude this section, it is important to highlight the differences between our FM–ALE approach and a classical ALE formulation:

- Given a position of the fluid front on the fixed mesh, elements cut by the front are split into subelements (only for integration purposes), so that the front coincides with the edges of the subelements.
- After deforming the mesh from one time step to the other using classical ALE procedures, results are projected back to the original mesh.
- The front is represented by a boundary function, and not by the position of the material points at Γ_{free} as in a classical ALE method.

4. Side numerical ingredients

In this section we describe some numerical ingredients that, in spite of being essential in the development of the FM–ALE method, are not inherent to its main concept. In other words, these ingredients may be changed without altering the main concept of the method.

4.1. Level set function update

In the applications, there are several ways to define Γ_{free}. In general, we assume that this part of the boundary of the flow domain is defined by what we have called generically a *boundary function*. This function may be defined analytically or by discrete means, for example through interpolation from some nodes that define the location of Γ_{free}. That would be a natural way to deal with fluid-structure interaction problems.

In some applications, as those described in Section 6, it is convenient to represent Γ_{free} by a *level set function* (see [39] for an overview of these methods). This function, say ψ, will be the solution of the problem

$$\begin{aligned} &\partial_t \psi + \boldsymbol{u}\cdot\nabla\psi = 0 \quad \text{in } \Omega^0\times(0,T),\\ &\psi = \overline{\psi} \quad \text{on } \Gamma_{\text{inf}}\times(0,T),\\ &\psi(\boldsymbol{x},0) = \psi_0(\boldsymbol{x}) \quad \text{in } \Omega^0, \end{aligned} \tag{16}$$

where $\Gamma_{\text{inf}} := \{\boldsymbol{x}\in\partial\Omega^0 | \boldsymbol{u}\cdot\boldsymbol{n} < 0\}$ is the inflow part of the domain boundary. In free surface simulations, the initial condition ψ_0 is chosen in order to define the initial position of the fluid front to be analyzed. The boundary condition $\bar{\psi}$ determines whether fluid enters or not through a certain point of the inflow boundary.

Due to the pure convective type of the equation for ψ, we use the SUPG technique for the spatial discretization. Again, the temporal evolution is treated via the standard trapezoidal rule.

If ψ is taken as a step function, numerical problems may be encountered when it is transported. It is known that small oscillations in the vicinity of sharp gradients still remain using the SUPG formulation. These oscillations may propagate and yield to distorted front shapes, specially near corners. Compared to similar methods, such as the volume-of-fluid (VOF) method [21], one particularity of the level set method is that it uses a smooth function ψ. As the smoothness can be lost as the simulation evolves, the level set function must be redefined for each mesh node as explained for example in [9].

Once ψ is computed, $\Gamma_{\text{free}}(t)$ is defined as

$$\Gamma_{\text{free}}(t) = \{\boldsymbol{x}\in\Omega^0 \mid \psi(\boldsymbol{x},t) = 0\}.$$

Thus, $\Gamma_{\text{free}}(t)$ is simply updated by solving the problem for $\psi(\boldsymbol{x},t)$.

The important point to be noted is that the system is solved on the whole domain Ω^0. As mentioned earlier, we approximate this problem using a stabilized finite element method. For the discrete problem it is necessary to extrapolate the velocity defined on $\Omega(t)$ to the rest of Ω^0. The question is how to perform this extrapolation. In principle, the advection velocity $\boldsymbol{u}$ in (16) is only needed in the neighborhood of $\Gamma_{\text{free}}(t)$, since the precise transport of ψ is not needed, except for the transport of the isovalue that defines $\Gamma_{\text{free}}(t)$. In our calculations, we have found useful to extrapolate $\boldsymbol{u}$ by solving a Stokes problem on $\Omega(t)^c = \Omega^0\backslash\Omega(t)$. This has two main advantages with respect to a simpler extrapolation procedure, namely, the extrapolated velocity is weakly divergence free in $\Omega(t)^c$ and we can impose the correct boundary conditions for it.

4.2. Approximate imposition of boundary conditions

Even though we have not formulated it as such, the FM–ALE method can be considered an immersed boundary method, in the sense that $\Gamma_{\text{free}}(t)$ is a boundary that moves within a fixed domain Ω^0. From the conceptual point of view, there is no problem in imposing exactly Dirichlet boundary conditions on this part of the boundary. However, this requires the dynamic addition of mesh nodes (see Fig. 1, where these nodes are drawn in green), with the associated change in the sparsity of the

matrix of the algebraic system to be solved mentioned earlier. This is why it is very convenient from the implementation standpoint to avoid the explicit introduction of such nodes and to prescribe boundary conditions *approximately*. We summarize next the strategy proposed in [7] to prescribe Dirichlet boundary conditions on a generic immersed boundary, that we denote by Γ.

Let u_h be the unknown solution of a problem posed in $\Omega \subset \Omega^0$ for which we want to prescribe a condition on Γ. Let Ω_Γ be the set of elements cut by Γ, which is split as $\Omega_\Gamma = \Omega_{\Gamma,\mathrm{in}} \cup \Omega_{\Gamma,\mathrm{out}}$, where $\Omega_{\Gamma,\mathrm{in}} = \Omega \cap \Omega_\Gamma$ and $\Omega_{\Gamma,\mathrm{out}}$ is the interior of $\Omega_\Gamma \backslash \Omega_{\Gamma,\mathrm{in}}$. Let also Ω_{in} be such that $\Omega = \Omega_{\mathrm{in}} \cup \Omega_{\Gamma,\mathrm{in}}$. For simplicity, we will assume that the intersection of Γ with the element domains can be exactly represented by the classical isoparametric mapping. For the notation to be used, see Fig. 3 (left).

Suppose that the unknown u_h is interpolated as

$$u_h(\boldsymbol{x}) = \sum_{a=1}^{n_{\mathrm{in}}} I^a_{\mathrm{in}}(\boldsymbol{x}) U^a_{\mathrm{in}} + \sum_{b=1}^{n_{\mathrm{out}}} I^b_{\mathrm{out}}(\boldsymbol{x}) U^b_{\mathrm{out}} = \boldsymbol{I}_{\mathrm{in}}(\boldsymbol{x})\boldsymbol{U}_{\mathrm{in}} + \boldsymbol{I}_{\mathrm{out}}(\boldsymbol{x})\boldsymbol{U}_{\mathrm{out}},$$

where $I^a_{\mathrm{in}}(\boldsymbol{x})$ and $I^b_{\mathrm{out}}(\boldsymbol{x})$ are the standard interpolation functions, n_{in} is the number of nodes in Ω_{in}, the domain where the problem needs to be solved (including layer L_0) and n_{out} the number of nodes in layer L_{-1} (see Fig. 3).

The objective is to compute $\boldsymbol{U}_{\mathrm{out}}$. Suppose that u_h needs to be prescribed to a given function $\bar{\boldsymbol{u}}$ on Γ. The main idea is to compute $\boldsymbol{U}_{\mathrm{out}}$ by minimizing the functional

$$J_2(\boldsymbol{U}_{\mathrm{in}}, \boldsymbol{U}_{\mathrm{out}}) = \int_\Gamma (u_h(\boldsymbol{x}) - \bar{u}(\boldsymbol{x}))^2 = \int_\Gamma (\boldsymbol{I}_{\mathrm{in}}(\boldsymbol{x})\boldsymbol{U}_{\mathrm{in}} + \boldsymbol{I}_{\mathrm{out}}(\boldsymbol{x})\boldsymbol{U}_{\mathrm{out}} - \bar{u}(\boldsymbol{x}))^2. \tag{17}$$

Suppose now that the problem for u_h in Ω_{in} leads to an algebraic equation of the form

$$\boldsymbol{K}_{\mathrm{in,in}}\boldsymbol{U}_{\mathrm{in}} + \boldsymbol{K}_{\mathrm{in,out}}\boldsymbol{U}_{\mathrm{out}} = \boldsymbol{F}_{\mathrm{in}}. \tag{18}$$

The domain integrals in matrices $\boldsymbol{K}_{\mathrm{in,in}}$ and $\boldsymbol{K}_{\mathrm{in,out}}$ extend only over Ω. The nodal values $\boldsymbol{U}_{\mathrm{out}}$ are merely used as degrees of freedom to interpolate u_h in the domain Ω. If (18) is supplemented with the equation resulting from the minimization of functional (17), the system to be solved is finally

$$\begin{bmatrix} \boldsymbol{K}_{\mathrm{in,in}} & \boldsymbol{K}_{\mathrm{in,out}} \\ \boldsymbol{N}_\Gamma & \boldsymbol{M}_\Gamma \end{bmatrix} \begin{bmatrix} \boldsymbol{U}_{\mathrm{in}} \\ \boldsymbol{U}_{\mathrm{out}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_{\mathrm{in}} \\ \boldsymbol{f}_\Gamma \end{bmatrix}, \tag{19}$$

where

$$\boldsymbol{M}_\Gamma = \int_\Gamma \boldsymbol{I}^{\mathrm{t}}_{\mathrm{out}}(\boldsymbol{x})\boldsymbol{I}_{\mathrm{out}}(\boldsymbol{x}), \quad \boldsymbol{f}_\Gamma = \int_\Gamma \boldsymbol{I}^{\mathrm{t}}_{\mathrm{out}}(\boldsymbol{x})\bar{u}(\boldsymbol{x}), \quad \boldsymbol{N}_\Gamma = \int_\Gamma \boldsymbol{I}^{\mathrm{t}}_{\mathrm{out}}(\boldsymbol{x})\boldsymbol{I}_{\mathrm{in}}(\boldsymbol{x}).$$

It is important to note that this implementation maintains the connectivity of the background mesh.

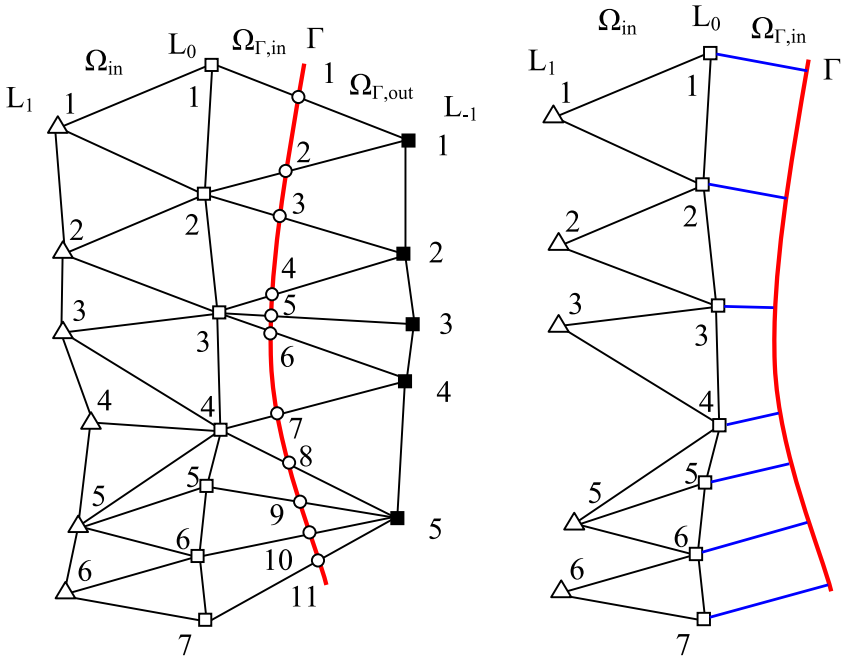

Fig. 3. Immersed boundary sketch (left) and domain of extrapolation (right) in a 2D example.

As it is explained in detail in [7], this method works well if the boundary Γ is not too close to layer L_0 in Fig. 3 (left). If this is not the case, the idea is to use the nodes of this layer to prescribe the boundary conditions in an approximate way, and to impose the equation to be solved at the rest of nodes of $\Omega_{\rm in}$.

Let E be the *extrapolation operator* of functions defined on the elements in contact with $\partial\Omega_{\rm in}$ to $\Omega_{\Gamma,{\rm in}}$. The first point to consider is how to choose the extrapolation region of this operator E. There are several possibilities, but the one we have found most accurate is the following. Let K be an element with an edge (in 2D) or face (in 3D) F on $\partial\Omega_{\rm in}$. Let K_Γ be the cylinder obtained from projecting F onto Γ in a way orthogonal to Γ. Then, E is defined as the extension from functions defined on K to functions defined on $K\cup K_\Gamma$. The extrapolation regions obtained this way in 2D using triangular elements are shown in Fig. 3 (right).

Suppose now that in $\Omega_{\rm in}$ the unknown u_h is interpolated as

$$u_h(\boldsymbol{x}) = \sum_{a=1}^{n_1} I_1^a(\boldsymbol{x}) U_1^a + \sum_{b=1}^{n_{00}} I_{00}^b(\boldsymbol{x}) U_{00}^b = \boldsymbol{I}_1(\boldsymbol{x})\boldsymbol{U}_1 + \boldsymbol{I}_{00}(\boldsymbol{x})\boldsymbol{U}_{00},$$

where $I_1^a(\boldsymbol{x})$ and $I_{00}^b(\boldsymbol{x})$ are the standard interpolation functions, n_1 is the number of nodes interior to $\Omega_{\rm in}$ (up to layer L_1) and n_{00} the number of nodes in layer L_0 (see Fig. 3).

The objective is to compute $\boldsymbol{U}_{00}$. We propose to obtain it by minimizing the functional

$$J_2'(\boldsymbol{U}_1,\boldsymbol{U}_{00}) = \int_\Gamma (Eu_h(\boldsymbol{x}) - \bar{u}(\boldsymbol{x}))^2 = \int_\Gamma (E\boldsymbol{I}_1(\boldsymbol{x})\boldsymbol{U}_1 + E\boldsymbol{I}_{00}(\boldsymbol{x})\boldsymbol{U}_{00} - \bar{u}(\boldsymbol{x}))^2,$$

which leads to

$$\boldsymbol{M}_{00}\boldsymbol{U}_{00} = \boldsymbol{f}_{00} - \boldsymbol{N}_{00}\boldsymbol{U}_1, \tag{20}$$

where

$$\boldsymbol{M}_{00} = \int_\Gamma E\boldsymbol{I}_{00}^{\rm t}(\boldsymbol{x})E\boldsymbol{I}_{00}(\boldsymbol{x}), \quad \boldsymbol{f}_{00} = \int_\Gamma E\boldsymbol{I}_{00}^{\rm t}(\boldsymbol{x})\bar{u}(\boldsymbol{x}), \quad \boldsymbol{N}_{00} = \int_\Gamma E\boldsymbol{I}_{00}^{\rm t}(\boldsymbol{x})E\boldsymbol{I}_1(\boldsymbol{x}).$$

If the matrix form of the problem for u_h posed in $\Omega_{\rm in}$ is

$$\boldsymbol{K}_{1,1}\boldsymbol{U}_1 + \boldsymbol{K}_{1,00}\boldsymbol{U}_{00} = \boldsymbol{F}_1,$$

the combination of this equation with (20) leads to the final system to be solved:

$$\begin{bmatrix} \boldsymbol{K}_{1,1} & \boldsymbol{K}_{1,00} \\ \boldsymbol{N}_{00} & \boldsymbol{M}_{00} \end{bmatrix} \begin{bmatrix} \boldsymbol{U}_1 \\ \boldsymbol{U}_{00} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_1 \\ \boldsymbol{f}_{00} \end{bmatrix}. \tag{21}$$

To conclude this subsection, let us explain how to combine methods (19) and (21). Let us first write problem (19) as

$$\begin{bmatrix} \boldsymbol{K}_{1,1} & \boldsymbol{K}_{1,00} & \boldsymbol{0} \\ \boldsymbol{K}_{00,1} & \boldsymbol{K}_{00,00} & \boldsymbol{K}_{00,{\rm out}} \\ \boldsymbol{0} & \boldsymbol{N}_{\Gamma,00} & \boldsymbol{M}_\Gamma \end{bmatrix} \begin{bmatrix} \boldsymbol{U}_1 \\ \boldsymbol{U}_{00} \\ \boldsymbol{U}_{\rm out} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_1 \\ \boldsymbol{F}_{00} \\ \boldsymbol{f}_\Gamma \end{bmatrix}, \tag{22}$$

where the splitting of the matrices corresponds to the splitting of $\boldsymbol{U}_{\rm in}$ into $\boldsymbol{U}_1$ and $\boldsymbol{U}_{00}$.

Problem (21) is obtained by considering the degrees of freedom of *all* nodes in layer L_0 as parameters to prescribe the boundary conditions, but of course the last equation in (22) can be kept, case in which the system to be solved is

$$\begin{bmatrix} \boldsymbol{K}_{1,1} & \boldsymbol{K}_{1,00} & \boldsymbol{0} \\ \boldsymbol{N}_{00} & \boldsymbol{M}_{00} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{N}_{\Gamma,00} & \boldsymbol{M}_\Gamma \end{bmatrix} \begin{bmatrix} \boldsymbol{U}_1 \\ \boldsymbol{U}_{00} \\ \boldsymbol{U}_{\rm out} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_1 \\ \boldsymbol{f}_{00} \\ \boldsymbol{f}_\Gamma \end{bmatrix}. \tag{23}$$

Clearly, $\boldsymbol{U}_{\rm out}$ depends on $\boldsymbol{U}_{00}$, but not the other way around. If Γ is very close to $\partial\Omega_{\rm in}$, the coefficients in $\boldsymbol{M}_\Gamma$ can be very small, but this does not affect the unknowns in the interior of the computational domain and, in fact, $\boldsymbol{M}_\Gamma$ can be replaced by any matrix without altering $\boldsymbol{U}_1$ and $\boldsymbol{U}_{00}$.

Method (22) is more accurate than method (23) (even though the order of accuracy is the same; see [7]). In order to use (22) in all situations except when instability problems may appear, we have implemented a blending of methods (22) and (23). The idea is simple. When a node in layer L_0 is detected to be very close to Γ, its degree of freedom is used to prescribe the boundary conditions, that is to say, the row in the equation for $\boldsymbol{U}_{00}$ in (22) is replaced by the corresponding row in (23). This strategy has proved robust and effective. Since usually only a few equations need to be changed, the overall accuracy obtained is very close to that of method (22). However, there is also the possibility of moving the mesh nodes when a node in layer L_0 is detected to be very close to Γ in order to avoid the use of (23). This, of course, will affect the domain velocity.

4.3. Data transfer between finite element meshes

The last crucial ingredient in the FM–ALE approach is the transfer of information between meshes $M_{\rm virt}^{n+1}$ and M^{n+1} for each time step n (see Fig. 1). In principle, it would be possible to use a simple interpolation operator. However, it is well known that

this interpolation, for example when it is of Lagrangian type, may suffer from overdiffusivity, in the sense that results on the new mesh may be damped from those of the original one. Another possibility could be to use the L^2 projection as transfer operator. We explain here how to incorporate *restrictions* to the projection between meshes. The idea described in the following was introduced in [23] in the context of transmission of information through boundaries in domain decomposition methods. For a method particularly designed in the context of immersed boundary methods for the transfer of forces, see [46].

Let us consider two meshes, M_1 and M_2, of a domain Ω. For simplicity, we assume that both are conforming (matching $\partial\Omega$). Let n_i ($i = 1,2$) be the number of nodes in M_i and let $\boldsymbol{\Phi}_i \in \mathbb{R}^{n_i}$ be the array of nodal values of a scalar variable ϕ. Suppose that $\boldsymbol{\Phi}_1$ is known and we want to project it onto M_2 to obtain $\boldsymbol{\Phi}_2$. If $\boldsymbol{P}_{21} \in \mathrm{Mat}_{\mathbb{R}}(n_2, n_1)$ is the transfer operator from M_1 to M_2 (for example the standard interpolation or the L^2 projection), a simple choice would be $\boldsymbol{\Phi}_2 = \boldsymbol{P}_{21}\boldsymbol{\Phi}_1$. However, suppose that we require $\boldsymbol{\Phi}_2$ to inherit a set of properties from $\boldsymbol{\Phi}_1$, written in the form

$$\boldsymbol{R}_2\boldsymbol{\Phi}_2 = \boldsymbol{R}_1\boldsymbol{\Phi}_1, \quad \boldsymbol{R}_i \in \mathrm{Mat}_{\mathbb{R}}(n_r, n_i), \tag{24}$$

where n_r is the number of restrictions to be imposed. The idea we propose is to take $\boldsymbol{\Phi}_2$ as close as possible to $\boldsymbol{P}_{21}\boldsymbol{\Phi}_1$ but satisfying (24). A possibility is to solve the optimization problem

$$\begin{aligned} &\text{minimize } \frac{1}{2}\,|\,\boldsymbol{\Phi}_2 - \boldsymbol{P}_{21}\boldsymbol{\Phi}_1|^2, \\ &\text{under the constraint } \boldsymbol{R}_2\boldsymbol{\Phi}_2 = \boldsymbol{R}_1\boldsymbol{\Phi}_1. \end{aligned}$$

This problem can be solved by optimizing the Lagrangian $L(\boldsymbol{\Phi}_2,\boldsymbol{\lambda})$, where $\boldsymbol{\lambda} \in \mathbb{R}^{n_r}$, given by

$$L(\boldsymbol{\Phi}_2, \boldsymbol{\lambda}) = \frac{1}{2}\,|\,\boldsymbol{\Phi}_2 - \boldsymbol{P}_{21}\boldsymbol{\Phi}_1|^2 - \boldsymbol{\lambda}^{\mathrm{t}}(\boldsymbol{R}_2\boldsymbol{\Phi}_2 - \boldsymbol{R}_1\boldsymbol{\Phi}_1).$$

This leads to the system

$$\begin{aligned} &\boldsymbol{\Phi}_2 - \boldsymbol{R}_2^{\mathrm{t}}\boldsymbol{\lambda} = \boldsymbol{P}_{21}\boldsymbol{\Phi}_1, \\ &\boldsymbol{R}_2\boldsymbol{\Phi}_2 = \boldsymbol{R}_1\boldsymbol{\Phi}_1, \end{aligned}$$

which after solving for $\boldsymbol{\Phi}_2$ yields

$$\boldsymbol{\Phi}_2 = \boldsymbol{P}_{21}\boldsymbol{\Phi}_1 + \boldsymbol{R}_2^{\mathrm{t}}(\boldsymbol{R}_2\boldsymbol{R}_2^{\mathrm{t}})^{-1}(\boldsymbol{R}_1 - \boldsymbol{R}_2\boldsymbol{P}_{21})\boldsymbol{\Phi}_1.$$

In the applications, the number of restrictions n_r is small, so that inverting $\boldsymbol{R}_2\boldsymbol{R}_2^{\mathrm{t}} \in \mathrm{Mat}_{\mathbb{R}}(n_r, n_r)$ is computationally affordable. In the case of the FM–ALE method, a typical restriction would be for example to impose global conservation of momentum and of mass when projecting velocities from mesh M_{virt}^{n+1} to M^{n+1} for each n. In this case, $n_r = d + 1$.

5. A numerical example

In this section we will solve the flow over a moving cylinder with the proposed FM–ALE strategy. The objective is to apply this methodology to this simple validating example, before showing more complex applications in the next section.

The corresponding flow equations are those described in Section 2, although in this case a multi-step time discretization will be used. In particular, we will use the second order backward differentiation scheme (BDF2), in which the time derivative at time $n + 1$ is approximated as:

$$\left.\frac{\partial u}{\partial t}\right|^{n+1} \approx \frac{1}{\delta t}\left(\frac{3}{2}u^{n+1} - 2u^n + \frac{1}{2}u^{n-1}\right).$$

The strategy described in Subsection 4.2 and [7] will be used to prescribe Dirichlet type boundary conditions on the surface of the moving solid, in this case the cylinder.

The *hold-all domain* is the rectangle $B = [0,2.2] \times [0,0.44]$. A background mesh of 9000 linear triangles has been used. The considered solid is a cylinder of diameter $D = 0.2$, its trajectory being defined by the position of its center:

$$\begin{aligned} &x_c(t) = 1.1 + 0.8\sin\left(\frac{2\pi}{3}(t - 0.75)\right), \\ &y_c(t) = 0.22. \end{aligned}$$

The velocity is prescribed to (0,0) on the walls of the rectangular domain, except for the wall corresponding to $x = 2.2$, where it is left free, whereas it matches the cylinder velocity on the cylinder surface. Note that the flow is due only to the cylinder movement. Viscosity is set to 0.001, so that the maximum Reynolds number is $\mathrm{Re} \approx 300$ based on the cylinder diameter and the (maximum) velocity when the cylinder is located at the central section of the rectangle. The time step size has been set to $\delta t = 0.05$, and 60 time steps (a full period) have been performed, after which the flow is considered to be fully developed.

Fig. 4 shows the results obtained at time $t = 3$. We would like to remark the smoothness of the velocity field close to the cylinder surface.

It is also interesting to see which are the differences between the treatment of the newly created nodes in the proposed FM–ALE approach and other usual procedures. To this end we compare nodal values for newly created nodes at time t^n (in the time step which goes from t^n to t^{n+1}) for the FM–ALE approach (information is convected and projected) and for the more usual procedure of extrapolating values from neighboring nodes mentioned earlier.

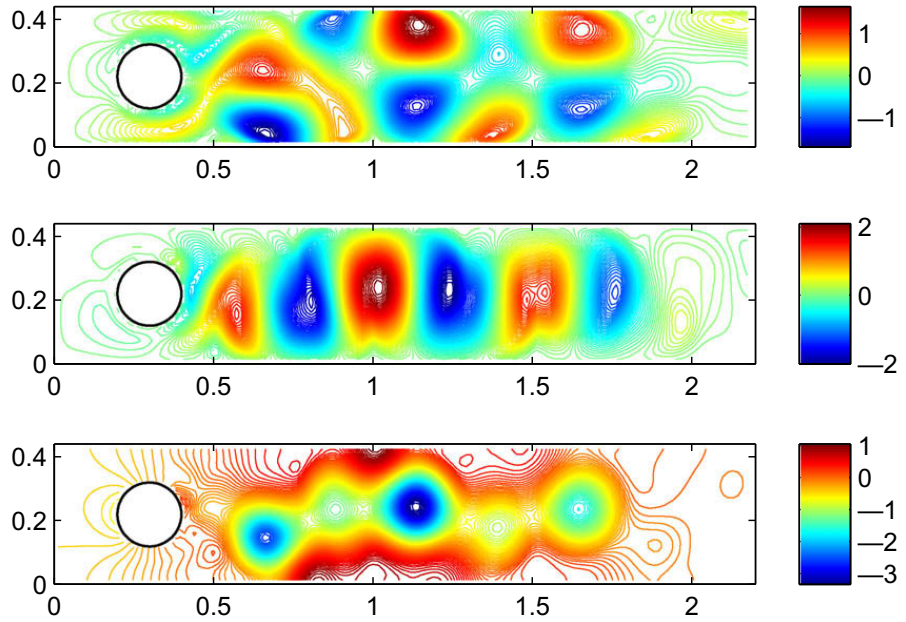

Fig. 4. Solution at $t = 3$. From top to bottom: x-velocity, y-velocity, pressure.

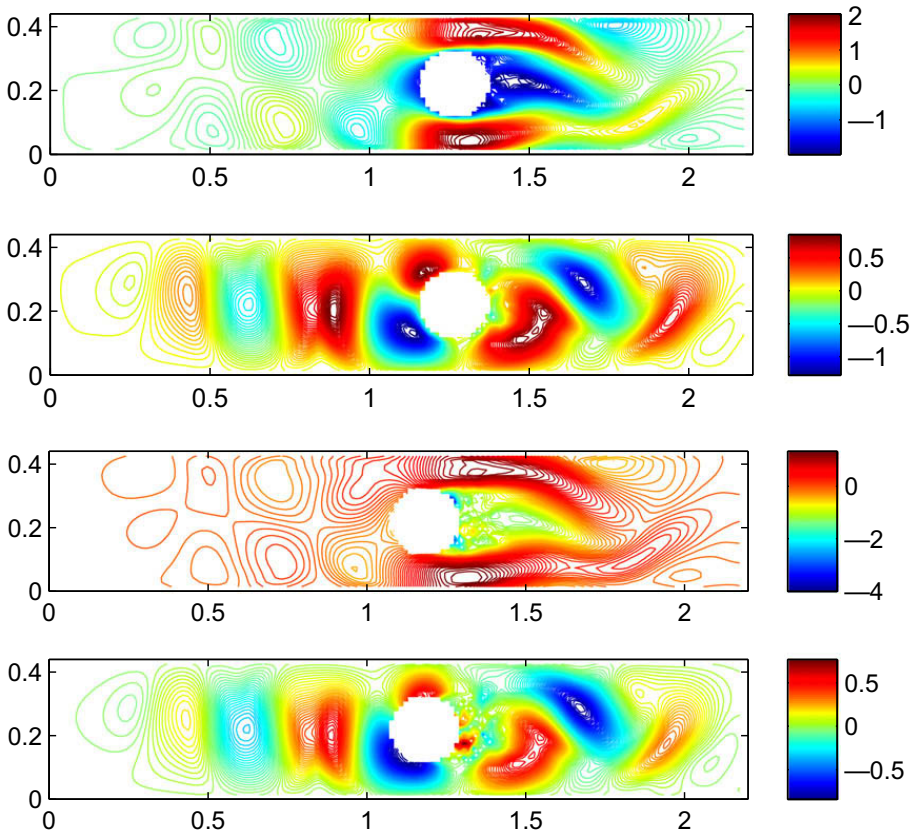

Fig. 5. Solution at $t = 2.25$, extrapolation procedure. From the top to the bottom: x-velocity before extrapolating, y-velocity before extrapolating, x-velocity after extrapolating, y-velocity after extrapolating.

Figs. 5 and 6 show velocity values (before and after the convection-projection or the extrapolation procedures) at $t^n = 2.25$. It can be seen that for large incremental displacements, as those of the time step we are considering, extrapolated values differ significantly from convected-projected values, and are much less smooth. Also, the values before the convection-projection or extrapolation procedure are smoother for the FM–ALE approach. We would like to stress that, contrary to the convection-projection of the FM–ALE method, the extrapolation procedure lacks physical grounds.

6. Two applications

The purpose of this section is to describe briefly two applications that led us to the development of the FM–ALE method. It is not our intention here to enter into the details of the problems, but rather to formulate them to stress how to apply the methodology in these two examples. For details, the reader is referred to [25] for Subsection 6.1 and to [11] for Subsection 6.2.

6.1. Lost foam casting

Lost foam casting is a casting technique in which the mold to be filled with molten metal is previously filled with a solid foam. The melt burns the foam when it contacts it, creating a residue that partly escapes through the mold walls (usually made of sand) and is partly trapped next to the boundaries of the mold. See [41] for a description of this technique.

Let Ω_m be the domain that is filled by the molten metal, Ω_f be the domain occupied by the foam and Ω^0 be the total domain (metal and foam). They are schematically shown in Fig. 7. Obviously, both Ω_m and Ω_f depend on time.

In this problem, apart from the Navier–Stokes equations (2,3), also the heat equation needs to be solved. Let ϑ be the temperature, C_p the specific heat at constant pressure, κ the thermal conduction coefficient and α_{ij} the heat transfer coefficient between materials "i" and "j". The subscripts "m", "f" and "o" will be used to refer to the physical properties of the molten metal, foam and mold, respectively. Likewise, Γ_{ij} will be used to denote the interface between materials "i" and "j", and the subscript inf will refer to values at the inflow of the domain; see Fig. 7.

Let us denote by $\boldsymbol{u}_{mf}$ the velocity at which the front of molten metal advances through the foam. In this problem, it turns out that this velocity *can be computed from an energy budget*. If u_{mf} is its norm, it turns out that (see [25])

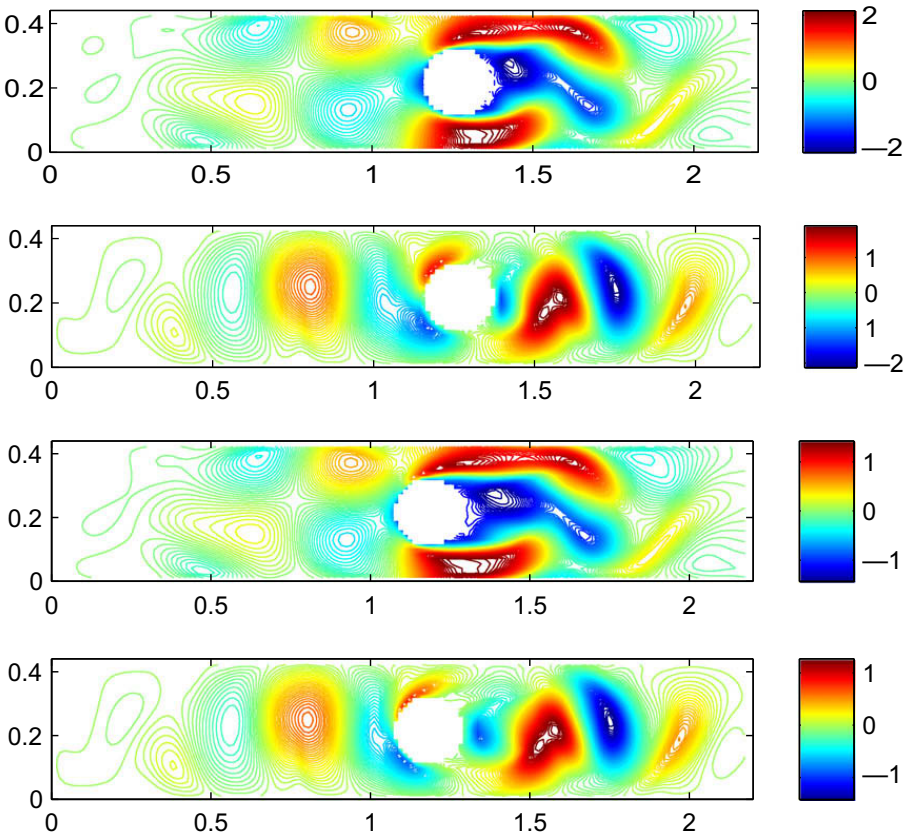

Fig. 6. Solution at $t = 2.25$, FM–ALE procedure. From the top to the bottom: x-velocity before convection-projection, y-velocity before convection-projection, x-velocity after convection-projection, y-velocity after convection-projection.

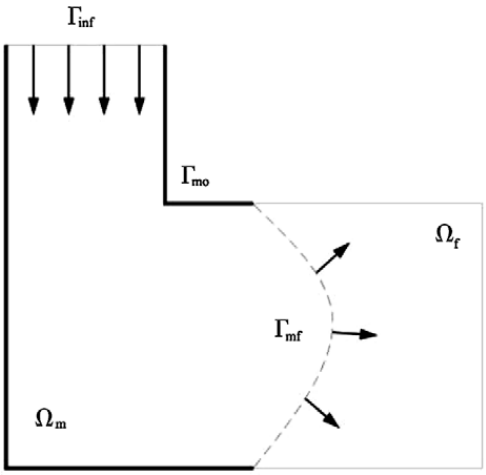

Fig. 7. Lost foam casting: problem setting.

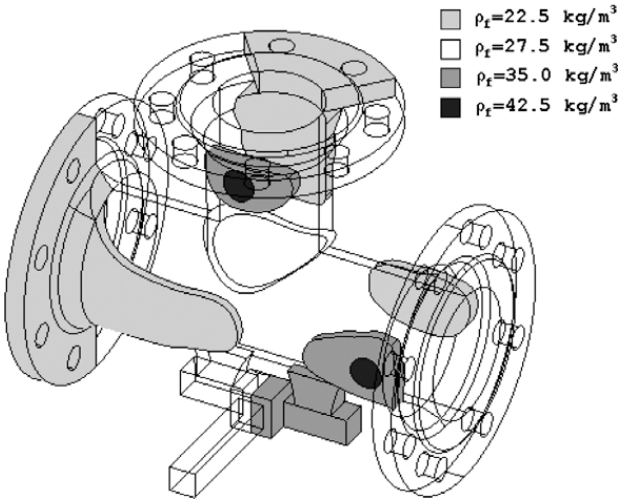

Fig. 8. Geometry and foam density for the lost foam casting example.

$$u_{\mathrm{mf}} = \frac{\alpha_{\mathrm{mf}}(\vartheta_{\mathrm{m}} - \vartheta_{\mathrm{f}})}{\rho_{\mathrm{f}}(c_{\mathrm{pf}}(\vartheta_{\mathrm{m}} - \vartheta_{\mathrm{f}}) + E_{\mathrm{mel}} + E_{\mathrm{vap}})}, \tag{25}$$

where E_{mel} and E_{vap} are the melting and vaporization energies, respectively, which must be determined from experiments.

The direction and orientation of $\boldsymbol{u}_{\mathrm{mf}}$ is determined by imposing this velocity to be normal to the advancing front. The key idea is to represent this front by a level set function ψ, the approximation of which has been described earlier. This leads to

$$\boldsymbol{u}_{\mathrm{mf}} = -\frac{u_{\mathrm{mf}}}{|\nabla\psi|}\nabla\psi.$$

If we consider now $\Omega(t) = \Omega_{\mathrm{m}}$, the problem to be solved consists in solving (2) and (3) together with the energy balance equation

$$\rho C_p\left[\left.\frac{\partial\vartheta}{\partial t}\right|_{\boldsymbol{x}}(x,t) + (u - u_{\mathrm{dom}})\cdot\nabla\vartheta\right] - \kappa\Delta\vartheta = 0, \tag{26}$$

and the boundary conditions on the interface Γ_{mf}

$$u = u_{\mathrm{mf}}, \tag{27}$$

$$-\kappa_{\mathrm{m}}\frac{\partial\vartheta}{\partial n} = \alpha_{\mathrm{mf}}(\vartheta - \vartheta_{\mathrm{f}}), \tag{28}$$

and the appropriate boundary conditions on the rest of the boundary.

There are several remarks to be made concerning the application of the FM–ALE method to this problem:

Remark 3

- The finite element approximation in space and the time integration of (26) is performed using the same formulation as for the Navier–Stokes equations.
- In view of (28), temperature is needed also in the foam domain Ω_f, which is also time dependent. Thus, an equation analogous to (26) has to be solved there, with $\boldsymbol{u} = \boldsymbol{0}$ but with a mesh velocity computed as for Ω_m.

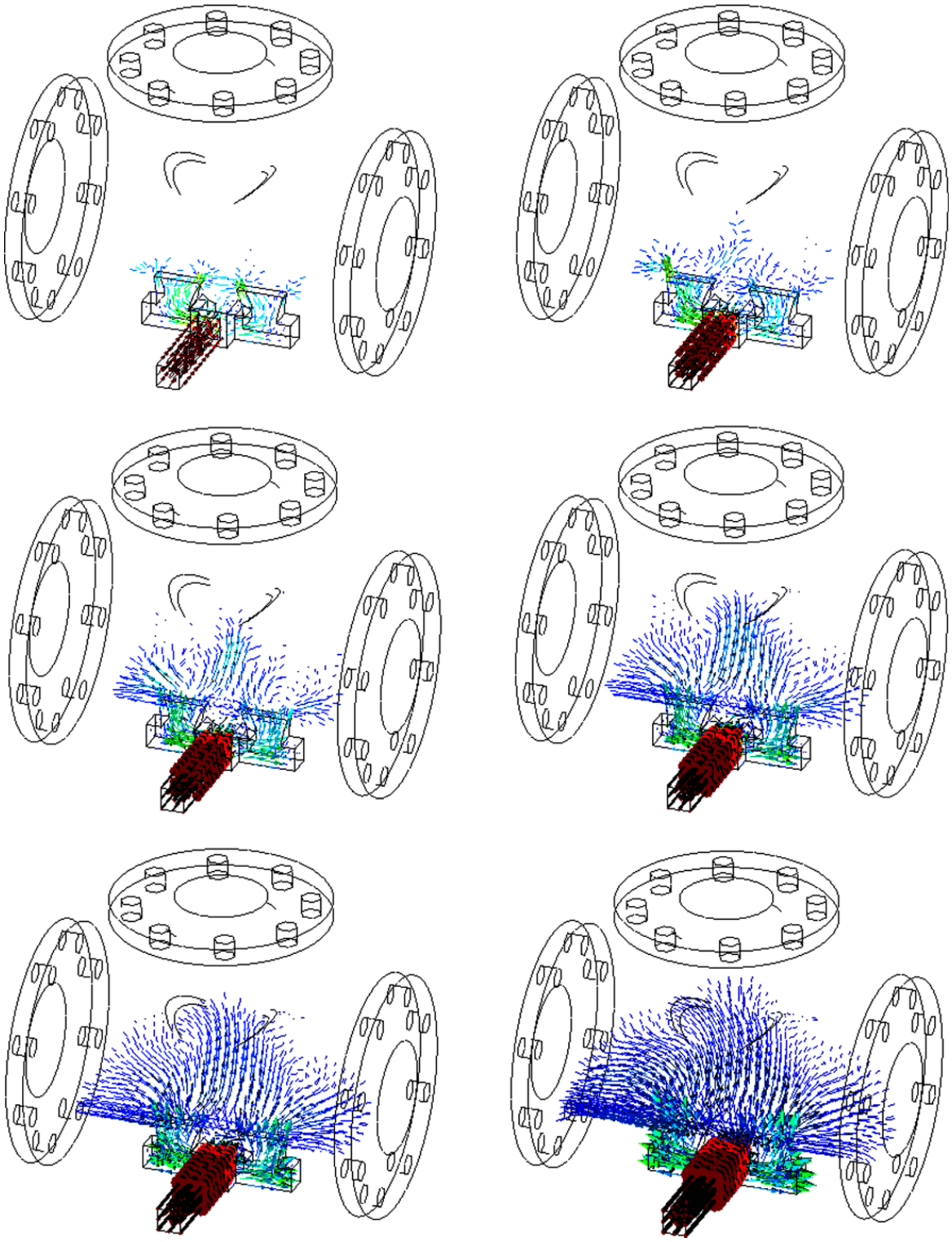

Fig. 9. Velocity vectors for the lost foam casting example.

- For the discrete problem, the transport of the level set function that determines Γ_{mf} requires the velocity $\boldsymbol{u}$ to be extrapolated from Ω_m to Ω_f. As mentioned earlier, we do this extrapolation by solving a Stokes problem in Ω_f using $\boldsymbol{u}_{mf}$ as boundary conditions on Γ_{mf}.
- In order to avoid introducing new nodes (apart from those of the background mesh), Dirichlet boundary conditions (27) need to be approximately imposed, for example using the strategy described in Subsection 4.2. Eq. (28) does not require any special treatment, as it is prescribed weakly and evaluating surface integrals on an immersed boundary does not pose any particular problem.

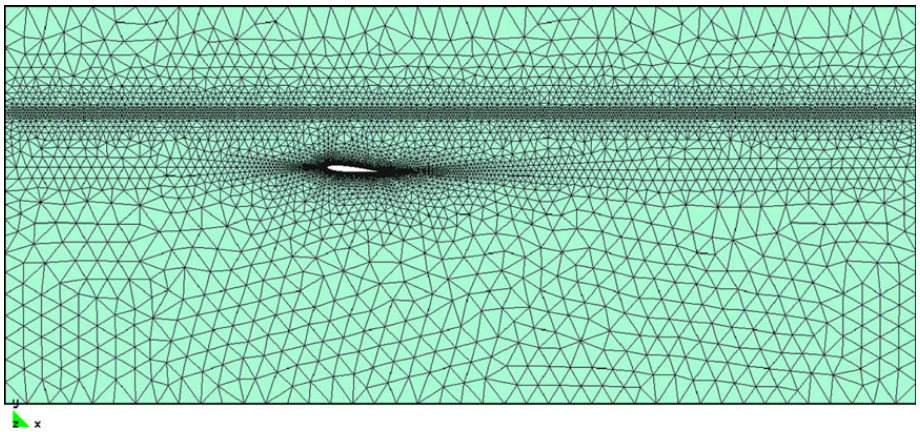

Fig. 10. Mesh around the NACA profile.

Fig. 11. Free surface model at $t = 20$ (top) and two phase model at $t = 6$ (bottom), when the solution starts to deteriorate.

To conclude this subsection, we present the results of a simulation of a three-dimensional tee-shaped casting (see [25] for more details). The only purpose of this example is to show that the methodology proposed is feasible in real applications.

The inner diameter of the vertical cylinder is 0.08 m while the inner diameter of the others is 0.10 m. The geometry is symmetrical with respect to the ingate and therefore the filling should also be symmetric. However, we want to observe the effects of variable foam density. In fact, the foam density is likely to be non-uniform, especially near the injection points. The front velocity model given by (25) is expected to take into account these effects, as the foam density appears in the denominator. Different zones of foam density are shown in Fig. 8.

The time evolution of the velocity vectors in the molten metal region is shown in Fig. 9.

6.2. Free surface flows

This example can be considered a prototypical problem for the application of the FM–ALE method. The domain $\Omega(t)$ is the region occupied by the fluid, separated from a region without fluid by a *free surface* Γ_{free}. The problem to be solved is exactly (2,3) with $\boldsymbol{f} = \boldsymbol{g}$, the gravity acceleration, and the boundary condition

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

Contrary to the previous example, now the velocity is unknown on Γ_{free} but the stress is known. This simplifies very much the imposition of boundary conditions since, as it has been mentioned in the previous subsection for the heat equation, boundary conditions imposed weakly on immersed boundaries do not represent any particular computational problem.

The alternative to the free surface treatment of many problems is a two-fluid coupling, assuming that the effect of one of the fluids over the other is negligible. In this case, Γ_{free} plays the role of an *interface*, rather than a *free* surface. In fact, in the applications the two-fluid approach is often more realistic, as in the case of water–air interfaces. However, from the numerical point of view the free surface approach is usually more robust. We will show this in a particular example presented next. Before this, let us comment that the major difference between the free surface and the two-fluids approach is that the former requires solving the flow equations on a moving domain, whereas the latter consists in solving on the whole domain where

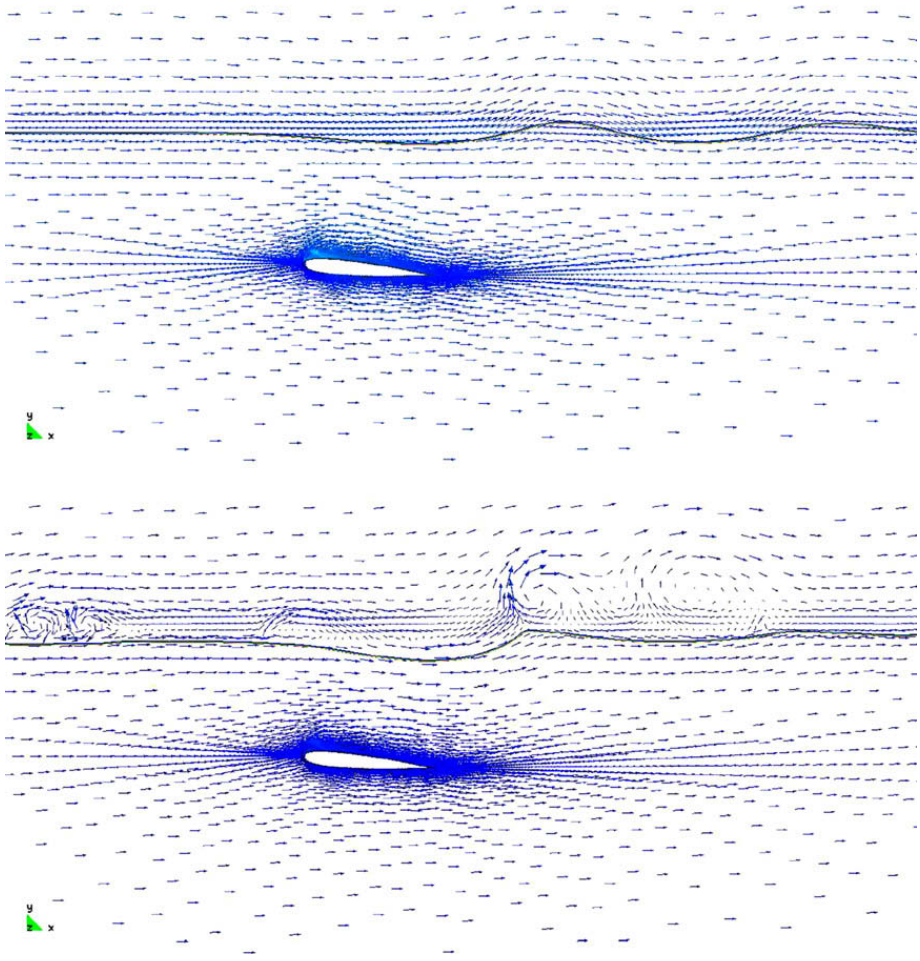

Fig. 12. Velocity field and free surface shape at $t = 6$. Free surface model (top) and two phase model (bottom).

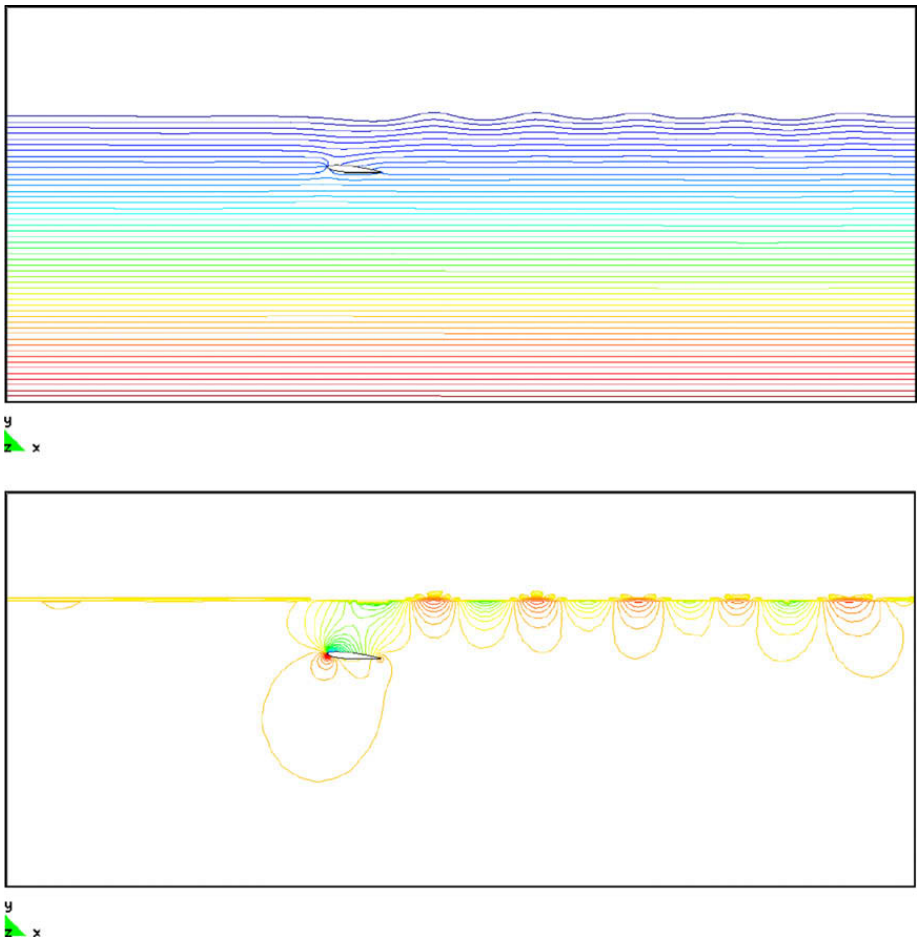

Fig. 13. Pressure contours at $t = 20$ with (top) and without (bottom) hydrostatic component (free surface model).

the flow takes place, which is constant-in-time, with different fluid properties in the regions occupied by the two fluids. In this second case, the stress (and the velocity) will be continuous at the moving interface.

As an example of application, let us consider a 2D flow over a submerged hydrofoil. The hydrofoil section considered is a NACA0012 with an angle of attack of 5° traveling at a speed of 1.776 m/s. It is an example that has been studied in the laboratory by Duncan [13] and used as a benchmark for numerical results by several authors [14,4].

For the two-fluid simulation, the material properties used (SI units) are $\rho_1 = 1000$, $\mu_1 = 0.001$ for the bottom fluid (water), and $\rho_2 = 1.2$, $\mu_2 = 0.000018$ for the top one (air). The simulations where run for 30 seconds with a 0.04 second time step size. The acceleration of gravity is $g = 10$. The Reynolds number using water properties is $\mathrm{Re} = 1.776 \times 10^6$, and the Froude number is $\mathrm{Fr} = 0.5673$.

A 2D unstructured mesh that covers a rectangle sized $[-6.0,11.0] \times [-4.4,3.0]$ around the hydrofoil was used. It is formed by 7415 nodes and 14364 linear triangular elements refined close to the hydrofoil and to the initial position of the interface as can be seen in Fig. 10. The boundary conditions used are prescribed inlet velocity at the left side of the rectangle, free slip at the bottom wall and at the hydrofoil, an open boundary on the top side, and the normal traction equal to minus the hydrostatic pressure corresponding to the initial water height at the right side of the rectangle. The level set function that determines the interface position is only prescribed at the left side of the domain. A constant 1.776 horizontal velocity is used as initial condition on the whole mesh except on the NACA hydrofoil, where it is zero. The interface is initially flat and positioned at $y = 0.9904$.

We have compared the results obtained with the free surface model [11] with those a two-fluid approach proposed in [10], in which air is also simulated. The former turns out to be more robust than the latter with the physical properties we have chosen. In Fig. 11 results for the position of the free surface/interface are shown. The interface position starts to deteriorate at $t = 6$ in the two-fluid approach, whereas for the free surface treatment the wave length and height match the experimental results satisfactorily. This is better observed in Fig. 12, where velocity vectors are plotted.

Finally in Fig. 13 we show the pressure and the pressure without the hydrostatic component corresponding to the initial interface height, in both cases at $t = 30$ and using the free surface formulation.

7. Conclusions

In this paper we have introduced in detail the concept of the FM–ALE approach. Succinctly, it consists in using the standard ALE method but "remeshing" at each time step so as to use always the same given mesh, which discretizes the whole region where the flow takes place.

The first benefit is conceptual. Ad-hoc approximations to account for the advection of information that can be found in several fixed-grid methods are avoided. This is in particular reflected by the treatment of the so called newly created nodes. When a node "dry" in one time step becomes part of the flow region in the next time step, the value of the flow variables to be assigned there to approximate (local) time derivatives is perfectly determined.

It has been our intention to clearly distinguish the main concept of the formulation from other related issues, and in particular from the approximate imposition of boundary conditions. Nevertheless, the way to carry out this imposition is essential for the success of the method. We have described our particular approach. Some remarks concerning the transfer of information between meshes have also been made, and the possibility to model the moving surface by level set functions has been explained.

Precisely the use of level set functions is crucial in the two applications shown, which we have included to demonstrate the potential of the method to deal with problems of different nature. Another natural application of the FM–ALE approach is the numerical approximation of fluid-structure interaction problems, a subject of a forthcoming work.

References

[1] S. Badia, R. Codina, Analysis of a stabilized finite element approximation of the transient convection-diffusion equation using an ALE framework, SIAM Journal on Numerical Analysis 44 (2006) 2159–2197.
[2] H.J. C Barbosa, T.J. R Hughes, The finite element method with Lagrangian multipliers on the boundary: circumventing the Babuška–Brezzi condition, Computer Methods in Applied Mechanics and Engineering 85 (1991) 109–128.
[3] D. Boffi, L. Gastaldi, Stability and geometric conservation laws for ALE formulation, Computer Methods in Applied Mechanics and Engineering 193 (2004) 4717–4739.
[4] C.H. Chan, K. Anastasiou, Solution of incompressible flows with or without a free surface using the finite volume method on unstructured triangular meshes, International Journal for Numerical Methods in Fluids 29 (1999) 35–57.
[5] R. Codina, A stabilized finite element method for generalized stationary incompressible flows, Computer Methods in Applied Mechanics and Engineering 190 (2001) 2681–2706.
[6] R. Codina, Stabilized finite element approximation of transient incompressible flows using orthogonal subscales, Computer Methods in Applied Mechanics and Engineering 191 (2002) 4295–4321.
[7] R. Codina, J. Baiges, Approximate imposition of boundary conditions in immersed boundary methods, submitted for publication.
[8] R. Codina, G. Houzeaux, Implementation aspects of coupled problems in CFD involving time dependent domains, in: G. Bugeda, J. C Courty, A. Guilliot, R. Höld, M. Marini, T. Nguyen, K. Papailiou, J. Périaux, D. Schwamborn (Eds.), Verification and Validation Methods for Challenging Multiphysics Problems, CIMNE, Barcelona, 2006, pp. 99–123.
[9] R. Codina, O. Soto, A numerical model to track two-fluid interfaces based on a stabilized finite element method and the level set technique, International Journal for Numerical Methods in Fluids 40 (2002) 293–301.
[10] H. Coppola-Owen, R. Codina, Improving Eulerian two-phase flow finite element approximation with discontinuous gradient pressure shape functions, International Journal for Numerical Methods in Fluids 49 (2005) 1287–1304.
[11] H. Coppola-Owen, R. Codina, A finite element model for free surface flows on fixed meshes, International Journal for Numerical Methods in Fluids 54 (2007) 1151–1171.
[12] J. Donea, P. Fasoli-Stella, S. Giuliani, Lagrangian and Eulerian finite element techniques for transient fluid structure interaction problems. Transactions Fourth SMIRT, 1977, p. B1/2.
[13] J.H. Duncan, The breaking and non-breaking wave reinstance of a two dimensional hydrofoil, Journal of Fluid Mechanics 126 (1983) 507–520.
[14] T. Duncan, L. Martinelli, A. Jameson, A finite-volume method with unstructured grid for free surface flow simulations, in: Proc. 26th Int. Conf. Num. ship Hydro-dyn. Iowa City, IA, 1993, pp. 173–193.
[15] L. Formaggia, F. Nobile, A stability analysis for the Arbitrary Lagrangian Eulerian formulation with finite elements, East–West Journal of Numerical Mathematics 7 (1999) 105–132.
[16] L. Formaggia, F. Nobile, Stability analysis of second-order time accurate schemes for ALE–FEM, Computer Methods in Applied Mechanics and Engineering 193 (2004) 4097–4116.
[17] L.P. Franca, S.L. Frey, Stabilized finite element methods: II. The incompressible Navier–Stokes equations, Computer Methods in Applied Mechanics and Engineering 99 (1992) 209–233.
[18] A. Gilmanov, F. Sotiropoulos, A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies, Journal of Computational Physics 207 (2005) 457–492.
[19] R. Glowinski, T.-W. Pan, J. Périaux, A fictitious domain method for Dirichlet problems and applications, Computer Methods in Applied Mechanics and Engineering 111 (1994) 203–303.
[20] R. Glowinski, T.W. Pan, T.I. Hesla, D.D. Joseph, 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 30 (1999) 1043–1066.
[21] C.W. Hirt, B.D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, Journal of Computational Physics 39 (1981) 201–225.
[22] J. Dolbow H.M. Mourad, I. Harari, A bubble-stabilized finite element method for Dirichlet constraints on embedded interfaces, International Journal for Numerical Methods in Engineering 69 (2007) 772–793.
[23] G. Houzeaux, R. Codina, Transmission conditions with constraints in finite element domain decomposition methods for flow problems, Communications in Numerical methods in Engineering 17 (2001) 179–190.
[24] G. Houzeaux, R. Codina, A Chimera method based on a Dirichlet/Neumann (Robin) coupling for the Navier–Stokes equations, Computer Methods in Applied Mechanics and Engineering 192 (2003) 3343–3377.
[25] G. Houzeaux, R. Codina, A finite element model for the simulation of lost foam casting, International Journal for Numerical Methods in Fluids 46 (2004) 203–226.
[26] A. Huerta, W.K. Liu, Viscous flow with large free surface motion, Computer Methods in Applied Mechanics and Engineering 69 (1988) 277–324.
[27] T.J.R. 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 127 (1995) 387–401.
[28] T.J.R. Hughes, W.K. Liu, T.K. Zimmerman, Lagrangian–Eulerian finite element formulation for incompressible viscous flows, Computer Methods in Applied Mechanics and Engineering 29 (1981) 329–349.

[29] H. Ji, J.E. Dolbow, On strategies for enforcing interfacial constraints and evaluating jump conditions with the extended finite element method, International Journal for Numerical Methods in Engineering 61 (2004) 2508–2535.
[30] D. Kima, H. Choi, Immersed boundary method for flow around an arbitrarily moving body, Journal of Computational Physics 212 (2006) 662–680.
[31] Ming-Chih Lai, C.S. Peskin, An immersed boundary method with formal second-order accuracy and reduced numerical viscosity, Journal of Computational Physics 160 (2000) 705–719.
[32] M. Lesoinne, C. Farhat, Geometric conservation laws for flow problems with moving boundaries deformable meshes and their impact on aerolastic computations, Computer Methods in Applied Mechanics and Engineering 134 (1996) 71–90.
[33] R.J. LeVeque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM Journal on Numerical Analysis 31 (4) (1994) 1019–1044.
[34] R.J. LeVeque, Z. Li, Immersed interface method for incompressible Navier–Stokes equations SIAM, Journal on Scientific and Statistical Computing 18 (3) (1997) 709–735.
[35] R. Löhner, J.R. Cebral, F.F. Camelli, J.D. Baum, E.L. Mestreau, Adaptive embedded/immersed unstructured grid techniques, Archives of Computational Methods in Engineering 14 (2007) 279–301.
[36] R. Mittal, G. Iaccarino, Immersed boundary methods, Annual Review of Fluid Mechanics 37 (2005) 239–261.
[37] J. Mohd-Yusof, Combined immersed boundaries/B-splines methods for simulations of flows in complex geometries, CTR Annual Research Briefs, Stanford University, NASA Ames, 1997.
[38] F. Nobile, Numerical Approximation of Fluid-Structure Interaction problems with application to Haemodynamics. PhD Thesis, École Polytechnique Fédérale de Lausanne, 2001.
[39] S. Osher, R.P. Fedkiw, Level set methods: and overview and some recent results, Journal of Computational Physics 169 (2001) 463–502.
[40] C.S. Peskin, Flow patterns around heart valves: a numerical method, Journal of Computational Physics 10 (1972) 252–271.
[41] T.S. Piwonka, A comparison of lost pattern casting processes, Material Designing 11 (6) (1990) 283–290.
[42] T.E. Tezduyar, Finite element methods for flow problems with moving boundaries and interfaces, Archives of Computational Methods in Engineering 8 (2) (2001) 83–130.
[43] J.V. Voorde, J. Vierendeels, E. Dick, Flow simulations in rotary volumetric pumps and compressors with the fictitious domain method, Journal of Computational and Applied Mathematics 168 (2004) 91–499.
[44] S. Xu, Z.J. Wang, An immersed interface method for simulating the interaction of a fluid with moving boundaries, Journal of Computational Physics 216 (2006) 454–493.
[45] J.H. Ferziger, Y.H. Tseng, A ghost-cell immersed boundary method for flow in complex geometry, Journal of Computational Physics 192 (2003) 593–623.
[46] H. Zhao, J.B. Freund, R.D. Moser, A fixed-mesh method for incompressible flow-structure systems with finite solid deformations, Journal of Computational Physics 227 (2008) 3114–3140.