Archive of Numerical Software
vol. 2, no. 1, pages 1–27

A Fully Coupled Immersed Finite Element Method for Fluid Structure Interaction via the Deal.II Library

Luca Heltai*[1], Saswati Roy[2], and Francesco Costanzo[2]

[1]Scuola Internazionale Superiore di Studi Avanzati, Via Bonomea 265, 34136 Trieste, Italy
[2]Center for Neural Engineering, Department of Engineering Science and Mechanics, The Pennsylvania State University, University Park PA 16802 USA

Received: Sep 13, 2012; **final revision:** May 13, 2014; **published:** Sep 3, 2014.

Abstract: We present the implementation of a solution scheme for fluid-structure interaction problems via the finite element software library `deal.II`. The solution scheme is an immersed finite element method in which two independent discretizations are used for the fluid and immersed deformable body. In this type of formulation the support of the equations of motion of the fluid is extended to cover the union of the solid and fluid domains. The equations of motion over the extended solution domain govern the flow of a fluid under the action of a body force field. This body force field informs the fluid of the presence of the immersed solid. The velocity field of the immersed solid is the restriction over the immersed domain of the velocity field in the extended equations of motion. The focus of this paper is to show how the determination of the motion of the immersed domain is carried out in practice. We show that our implementation is general, that is, it is not dependent on a specific choice of the finite element spaces over the immersed solid and the extended fluid domains. We present some preliminary results concerning the accuracy of the proposed method.

Keywords: Fluid Structure Interaction; Immersed Boundary Methods; Immersed Finite Element Method; Finite Element Immersed Boundary Method

1 Introduction

We discuss a C++ program implemented using the `deal.II` library (Bangerth et al., 2006, 2013) for the simulation of fluid-structure interaction (FSI) problems based on the immersed finite element method (IFEM). The latest stable version of the full source code is available at
`https://github.com/luca-heltai/ans-ifem/releases`
and as a public git repository at
`https://github.com/luca-heltai/ans-ifem`

Heltai and Costanzo (2012) have recently discussed a fully variational formulation for an immersed method to the solution of fluid-structure interaction (FSI) problems. Immersed methods, which deal with the motion of bodies immersed in fluids, allow one to choose the discretization for the fluid and solid domains independently from each other. As such, they stand in contrast to established methods like the arbitrary Lagrangian-Eulerian (ALE) ones (see, e.g., Hughes et al., 1981), where the topologies of the solution grids for the fluid and the solid are constrained.

Immersed methods have three main features:

*Corresponding Author. Email: Luca Heltai <luca.heltai@sissa.it>; Tel.: +39 040 3787 449; Fax: +39 040 3787528

1. The support of the equations of motion of the fluid is extended to the union of the physical fluid and solid domains.
2. The equations of motion of the fluid have terms that, from a continuum mechanics viewpoint, are body forces "informing" the fluid of its interaction with the solid.
3. The velocity field of the immersed solid is identified with the restriction to the solid domain of the velocity field in the equations of motion of the fluid.

A taxonomy of immersed methods can be based on how these three elements are treated theoretically and/or are implemented practically (see the discussion in Heltai and Costanzo (2012) and Roy et al., 2013). Here we employ the approach proposed in Heltai and Costanzo (2012) in which the entire solution scheme is developed within the general framework of the finite element method. Most importantly, the restriction mentioned at point 3 above is done via a fully variational approach. As such, the approach demonstrated herein stands in contrast to what is used in the immersed boundary methods stemming from the approach of Peskin and his co-workers (see, e.g., Peskin, 1977, 2002; Griffith and Luo, 2012; Griffith, 2012) or the finite element extension of Peskin's approach due to Liu and co-workers (see, e.g., Wang and Liu, 2004; Zhang et al., 2004; Liu et al., 2007), which is based on the implementation of the reproducing kernel particle method. As explained in detail in Heltai and Costanzo (2012), the method demonstrated herein stems from the approach by Boffi and Gastaldi (2003), Heltai (2006), and Boffi et al. (2008), and Heltai (2008).

In Section 2 we review the problem's governing equations. In Section 2.3 we present the variational reformulation of the governing equations and we will present their discrete counterparts in Section 3. The content of Sections 2 and 3 follows closely the exposition in Heltai and Costanzo (2012) and is reported here for completeness. In Section 4 we provide details about the code we have developed and instructions for compilation, execution, and generation of documentation. The entire code is based on the open source `deal.II` library (see Bangerth et al., 2007, 2006). We conclude the article with Section 5, where we present some numerical results.

2 Problem Formulation

2.1 Basic notation and governing equations

B_t in Fig. 1 represents the configuration of a regular body at time t. B_t is a (possibly multiply connected)

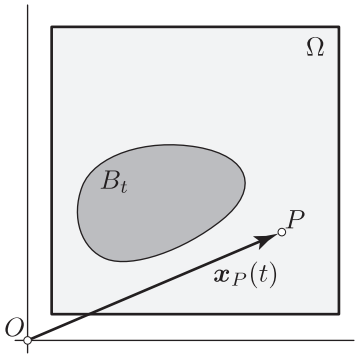


Figure 1: Current configuration B_t of a body $\mathscr{B}$ immersed in a fluid occupying the domain Ω.

proper subset of a fixed control volume Ω. The domain $\Omega \setminus B_t$ is filled by a fluid and we refer to B_t as the *immersed body*. $\partial\Omega$ and ∂B_t, with outer unit normals $\boldsymbol{m}$ and $\boldsymbol{n}$, respectively, are the boundaries of Ω and B_t. We denote by B the reference configuration of the immersed body. We denote the position of points of $\mathscr{B}$ in B by $\boldsymbol{s}$, whereas we denote the position at time t of a generic point $P \in \Omega$ by $\boldsymbol{x}_P(t)$. A motion of $\mathscr{B}$ is a diffeomorphism $\boldsymbol{\zeta} : B \to B_t$, $\boldsymbol{x} = \boldsymbol{\zeta}(\boldsymbol{s}, t)$, with $\boldsymbol{s} \in B$, $\boldsymbol{x} \in \Omega$, and $t \in [0, T)$, with T a positive real number.

The function $\rho(\boldsymbol{x}, t)$ describes the mass density in the entire domain Ω. The function ρ can be discontinuous across ∂B_t. The local form of the balance of mass requires that, $\forall t \in (0, T)$,

$$\dot{\rho} + \rho \operatorname{div} \boldsymbol{u} = 0, \quad \boldsymbol{x} \in \Omega \setminus (\partial\Omega \cup \partial B_t), \tag{1}$$

where $\boldsymbol{u}(\boldsymbol{x},t) = \partial\boldsymbol{\zeta}(\boldsymbol{s},t)/\partial t\big|_{\boldsymbol{s}=\boldsymbol{\zeta}^{-1}(\boldsymbol{x},t)}$ is the velocity field, a dot over a quantity denotes the material time derivative* of that quantity, and where 'div' represents the divergence operator with respect to $\boldsymbol{x}$. We note that Eq. (1) is not the only way to express the balance of mass. Other equivalent forms are discussed in Gurtin et al. (2010).

The local form of the momentum balance laws require that, $\forall t \in (0,T)$, $\mathsf{T} = \mathsf{T}^{\mathrm{T}}$ (the superscript T denotes the transpose) and

$$\operatorname{div}\mathsf{T} + \rho\boldsymbol{b} = \rho\dot{\boldsymbol{u}}, \quad \boldsymbol{x} \in \Omega \setminus (\partial\Omega \cup \partial B_t), \tag{2}$$

where $\mathsf{T}(\boldsymbol{x},t)$ is the Cauchy stress and $\boldsymbol{b}(\boldsymbol{x},t)$ is the external force density per unit mass acting on the system.

In addition to Eqs. (1) and (2), we demand that the velocity field be continuous (corresponding to a no slip condition between solid and fluid) and that the jump condition of the balance of linear momentum be satisfied across ∂B_t:

$$\boldsymbol{u}(\check{\boldsymbol{x}}^+,t) = \boldsymbol{u}(\check{\boldsymbol{x}}^-,t) \quad \text{and} \quad \mathsf{T}(\check{\boldsymbol{x}}^+,t)\boldsymbol{n} = \mathsf{T}(\check{\boldsymbol{x}}^-,t)\boldsymbol{n}, \quad \check{\boldsymbol{x}} \in \partial B_t, \tag{3}$$

where the superscripts $-$ and $+$ denote limits as $\boldsymbol{x} \to \check{\boldsymbol{x}}$ from within and without B_t, respectively.

We denote by $\partial\Omega_D$ and $\partial\Omega_N$ the subsets of $\partial\Omega$ where Dirichlet and Neumann boundary data are prescribed, respectively. The domains $\partial\Omega_D$ and $\partial\Omega_N$ are such that

$$\partial\Omega = \partial\Omega_D \cup \partial\Omega_N \quad \text{and} \quad \partial\Omega_D \cap \partial\Omega_N = \emptyset. \tag{4}$$

We denote by $\boldsymbol{u}_g(\boldsymbol{x},t)$, with $\boldsymbol{x} \in \partial\Omega_D$, and by $\boldsymbol{\tau}_g(\boldsymbol{x},t)$, with $\boldsymbol{x} \in \partial\Omega_N$, the prescribed values of velocity (Dirichlet data) and traction (Neumann data), respectively, i.e.,

$$\boldsymbol{u}(\boldsymbol{x},t) = \boldsymbol{u}_g(\boldsymbol{x},t), \quad \text{for } \boldsymbol{x} \in \partial\Omega_D, \quad \text{and} \quad \mathsf{T}(\boldsymbol{x},t)\boldsymbol{m}(\boldsymbol{x},t) = \boldsymbol{\tau}_g(\boldsymbol{x},t), \quad \text{for } \boldsymbol{x} \in \partial\Omega_N, \tag{5}$$

where the subscript g stands for 'given'.

2.2 Constitutive behavior

2.2.0.1 Constitutive response of the fluid. We assume that the fluid is linear viscous and incompressible with uniform mass density ρ. Denoting by $\hat{\mathsf{T}}_{\mathrm{f}}$ the constitutive response function of the Cauchy stress of the fluid, we have (see, e.g., Gurtin et al., 2010)

$$\hat{\mathsf{T}}_{\mathrm{f}} = -p\mathsf{I} + 2\mu\mathsf{D}, \quad \mathsf{D} = \tfrac{1}{2}\big(\mathsf{L} + \mathsf{L}^{\mathrm{T}}\big), \tag{6}$$

where p is the pressure of the fluid, I is the identity tensor, $\mu > 0$ is a given viscosity coefficient, and $\mathsf{L} = \operatorname{grad}\boldsymbol{u}$, and where a "hat" ($\hat{\mathsf{T}}$) is used to distinguish the constitutive response function for T from T itself. For convenience, we denote by $\hat{\mathsf{T}}_{\mathrm{f}}^{v}$ the viscous component of $\hat{\mathsf{T}}_{\mathrm{f}}$, i.e.,

$$\hat{\mathsf{T}}_{\mathrm{f}}^{v} = 2\mu\,\mathsf{D} = \mu\left(\mathsf{L} + \mathsf{L}^{\mathrm{T}}\right). \tag{7}$$

As already mentioned, the fluid is assumed to be incompressible. By definition, this means that the fluid's motions must always be locally volume-preserving. The definition of incompressibility implies that a material is incompressible if and only if $\dot{\rho} = 0$ (Gurtin et al., 2010), so that, as $\rho \neq 0$ always, Eq. (1) yields

$$\operatorname{div}\boldsymbol{u} = 0 \quad \text{for } \boldsymbol{x} \in \Omega \setminus B_t. \tag{8}$$

Under these conditions, p is a Lagrange multiplier that allows to enforce Eq. (8).

2.2.0.2 Constitutive response of the solid. The immersed body is taken to be incompressible and viscoelastic of differential type:

$$\hat{\mathsf{T}}_\mathrm{s} = -p\mathsf{I} + \hat{\mathsf{T}}^e_\mathrm{s} + \hat{\mathsf{T}}^v_\mathrm{s}, \tag{9}$$

where $\hat{\mathsf{T}}^e_\mathrm{s}$ and $\hat{\mathsf{T}}^v_\mathrm{s}$ denote the elastic and viscous parts of $\hat{\mathsf{T}}_\mathrm{s}$, respectively, and p is the Lagrange multiplier needed to enforce incompressibility. The viscous part of the behavior is assumed to be of the same type as that of the fluid, that is,

$$\hat{\mathsf{T}}^v_\mathrm{s} = 2\mu\,\mathsf{D} = \mu\left(\mathsf{L} + \mathsf{L}^\mathrm{T}\right), \tag{10}$$

where μ is the same constant viscosity coefficient of the fluid. We assume that $\hat{\mathsf{T}}^e_\mathrm{s}$ is obtained from a strain energy potential. To be precise, let the first Piola-Kirchhoff stress tensor be P. This tensor is related to T as follows (see, e.g., Gurtin et al., 2010):

$$\mathsf{P} = J\mathsf{T}\mathsf{F}^{-\mathrm{T}}, \tag{11}$$

where $J = \det\mathsf{F}$, and the tensor F, called the deformation gradient, is defined as

$$\mathsf{F} = \frac{\partial\boldsymbol{\zeta}(\boldsymbol{s},t)}{\partial\boldsymbol{s}}. \tag{12}$$

Letting $\hat{\mathsf{P}}^e_\mathrm{s} = J\hat{\mathsf{T}}^e_\mathrm{s}F^{-\mathrm{T}}$ denote the constitutive response function for the elastic part of the first Piola-Kirchhoff stress tensor, as is typical in elasticity, we assume the existence of a function $\hat{W}^e_\mathrm{s}(\mathsf{F})$ such that

$$\hat{\mathsf{P}}^e_\mathrm{s} = \frac{\partial\hat{W}^e_\mathrm{s}(\mathsf{F})}{\partial\mathsf{F}}, \tag{13}$$

where $\hat{W}^e_\mathrm{s}$ is the density of the elastic strain energy of the solid per unit volume. Invariance under changes of observer demands that $\hat{W}^e_\mathrm{s}$ be a function of an objective strain measure such as $\mathsf{C} = \mathsf{F}^\mathrm{T}\mathsf{F}$. If the solid is isotropic, $\hat{W}^e_\mathrm{s}$ must be a function of the principal invariants of C.

2.3 Reformulation of the governing equations

We now reformulate the governing equations in variational form. The motion of the solid will be described via the displacement field, denoted by $\boldsymbol{w}$ and defined as

$$\boldsymbol{w}(\boldsymbol{s},t) := \boldsymbol{\zeta}(\boldsymbol{s},t) - \boldsymbol{s}, \quad \boldsymbol{s}\in B. \tag{14}$$

The displacement gradient relative to the position in B is denoted by H:

$$\mathsf{H} := \frac{\partial\boldsymbol{w}}{\partial\boldsymbol{s}} \quad\Rightarrow\quad \mathsf{H} = \mathsf{F} - \mathsf{I}. \tag{15}$$

Equation (14) implies

$$\dot{\boldsymbol{w}}(\boldsymbol{s},t) = \boldsymbol{u}(\boldsymbol{x},t)\big|_{\boldsymbol{x}=\boldsymbol{\zeta}(\boldsymbol{s},t)}. \tag{16}$$

The principal unknowns of our fluid-structure interaction problem are then the fields

$$\boldsymbol{u}(\boldsymbol{x},t), \quad p(\boldsymbol{x},t), \quad \text{and} \quad \boldsymbol{w}(\boldsymbol{s},t), \quad \text{with } \boldsymbol{x}\in\Omega,\ \boldsymbol{s}\in B, \text{ and } t\in[0,T). \tag{17}$$

The functional spaces for the problem are

$$\boldsymbol{u}\in\mathscr{V} = H^1_D(\Omega)^d := \left\{\boldsymbol{u}\in L^2(\Omega)^d \,\middle|\, \nabla_{\boldsymbol{x}}\boldsymbol{u}\in L^2(\Omega)^{d\times d}, \boldsymbol{u}|_{\partial\Omega_D} = \boldsymbol{u}_g\right\}, \tag{18}$$

$$p\in\mathscr{Q} := L^2(\Omega), \tag{19}$$

$$\boldsymbol{w}\in\mathscr{Y} = H^1(B)^d := \left\{\boldsymbol{w}\in L^2(B)^d \,\middle|\, \nabla_{\boldsymbol{s}}\boldsymbol{w}\in L^2(B)^{d\times d}\right\}, \tag{20}$$

where $\nabla_{\boldsymbol{x}}$ and $\nabla_{\boldsymbol{s}}$ denote the gradient operators relative to $\boldsymbol{x}$ and $\boldsymbol{s}$, respectively. Also, referring to Eq. (18), the function space for the test functions for the velocity field is taken to be as follows:

$$\mathscr{V}_0 = H^1_0(\Omega)^d := \left\{\boldsymbol{v}\in L^2(\Omega)^d \,\middle|\, \nabla_{\boldsymbol{x}}\boldsymbol{v}\in L^2(\Omega)^{d\times d}, \boldsymbol{v}|_{\partial\Omega_D} = \boldsymbol{0}\right\}. \tag{21}$$

2.4 Variational restatement of the governing equations

When the solid is incompressible, the mass density of both the fluid and the solid are constant so that $\dot{\rho} = 0$ in Ω. Then, referring to Eqs. (5), Eqs. (18)–(20), and the constitutive response functions of both the fluid and the solid, the governing equations introduced so far can be expressed in weak form as follows:

$$\int_\Omega \rho(\dot{\boldsymbol{u}} - \boldsymbol{b}) \cdot \boldsymbol{v} \, \mathrm{d}v + \int_\Omega \hat{\mathsf{T}}_\mathrm{f} \cdot \nabla_{\boldsymbol{x}} \boldsymbol{v} \, \mathrm{d}v + \int_{B_t} \left(\hat{\mathsf{T}}_\mathrm{s} - \hat{\mathsf{T}}_\mathrm{f}\right) \cdot \nabla_{\boldsymbol{x}} \boldsymbol{v} \, \mathrm{d}v - \int_{\partial\Omega_N} \boldsymbol{\tau}_g \cdot \boldsymbol{v} \, \mathrm{d}a = 0 \quad \forall \boldsymbol{v} \in \mathscr{V}_0 \tag{22}$$

and

$$\int_\Omega q \, \mathrm{div}\, \boldsymbol{u} \, \mathrm{d}v = 0 \quad \forall q \in \mathscr{Q}. \tag{23}$$

A crucial aspect of our approach is the enforcement of Eq. (16). We enforce this relation weakly as follows:

$$\Phi_B \int_B \left[\dot{\boldsymbol{w}}(\boldsymbol{s}, t) - \boldsymbol{u}(\boldsymbol{x}, t)\big|_{\boldsymbol{x}=\boldsymbol{\zeta}(\boldsymbol{s},t)}\right] \cdot \boldsymbol{y}(\boldsymbol{s}) \, \mathrm{d}V = 0 \quad \forall \boldsymbol{y} \in \mathscr{Y}, \tag{24}$$

where $\mathrm{d}V$ is an infinitesimal volume element of B, and where Φ_B is a constant with dimensions of mass over time divided by length cubed, i.e., dimensions such that, in 3D, the volume integral of the quantity $\Phi_B \dot{\boldsymbol{w}}$ has the same dimensions as a force. We observe that, since we have assumed that the viscous part of the stress response of the solid is the same as that of the fluid (Heltai and Costanzo, 2012 discuss the most general of cases in which the immersed body and the surrounding fluid can have different constitutive response functions), the term $\left(\hat{\mathsf{T}}_\mathrm{s} - \hat{\mathsf{T}}_\mathrm{f}\right)$ in Eq. (22) is equal to the elastic response of the solid $\hat{\mathsf{T}}^e_\mathrm{s}$.

Our numerical approximation scheme for Eqs. (22)–(24) is based on the use of two independent triangulations, namely, one of Ω and one of B. The fields $\boldsymbol{u}$ and p, as well as their corresponding test functions, will be expressed via finite element spaces supported by the triangulation of Ω. By contrast, the field $\boldsymbol{w}$ will be expressed via a finite element space supported by the triangulation of B. Because of this, any term in Eq. (22) defined over B_t is now rewritten as an integral over B:

$$\int_\Omega \rho(\dot{\boldsymbol{u}} - \boldsymbol{b}) \cdot \boldsymbol{v} \, \mathrm{d}v - \int_\Omega p \, \mathrm{div}\, \boldsymbol{v} \, \mathrm{d}v + \int_\Omega \hat{\mathsf{T}}^v_\mathrm{f} \cdot \nabla_{\boldsymbol{x}} \boldsymbol{v} \, \mathrm{d}v - \int_{\partial\Omega_N} \boldsymbol{\tau}_g \cdot \boldsymbol{v} \, \mathrm{d}a + \int_B \hat{\mathsf{P}}^e_\mathrm{s} \, \mathsf{F}^\mathrm{T}(\boldsymbol{s}, t) \cdot \nabla_{\boldsymbol{x}} \boldsymbol{v}(\boldsymbol{x})\big|_{\boldsymbol{x}=\boldsymbol{\zeta}(\boldsymbol{s},t)} \, \mathrm{d}V = 0 \quad \forall \boldsymbol{v} \in \mathscr{V}_0. \tag{25}$$

We now define the operators we will use in our finite element formulation. In these definitions, we will use the following notation:

$$_{V^*}\left\langle \psi, \phi \right\rangle_{V}, \tag{26}$$

in which, given a vector space V and its dual V^*, ψ and ϕ are elements of the vector spaces V^* and V, respectively, and where $_{V^*}\left\langle \bullet, \bullet \right\rangle_{V}$ identifies the duality product between V^* and V. For convenience, we also introduce the following shorthand notation

$$\hat{\mathsf{T}}^v[\boldsymbol{u}] = \mu\left[\nabla_{\boldsymbol{x}} \boldsymbol{u}(\boldsymbol{x}, t) + (\nabla_{\boldsymbol{x}} \boldsymbol{u}(\boldsymbol{x}, t))^\mathrm{T}\right], \tag{27}$$

$$\mathsf{F}[\boldsymbol{w}] = \mathsf{I} + \nabla_s \boldsymbol{w}(\boldsymbol{s}, t), \tag{28}$$

$$\hat{\mathsf{P}}^e_\mathrm{s}[\boldsymbol{w}] = \left.\frac{\partial \hat{W}^e_\mathrm{s}(\mathsf{F})}{\partial F}\right|_{\mathsf{F}=\mathsf{F}[\boldsymbol{w}]}. \tag{29}$$

Finally, to help identify the domain and range of these operators, we establish the following convention. We will use the numbers 1, 2, and 3 to identify the spaces $\mathscr{V}$, $\mathscr{Q}$, and $\mathscr{Y}$, respectively. We will use the Greek letters α, β, and γ to identify the spaces $\mathscr{V}^*$, $\mathscr{Q}^*$, and $\mathscr{Y}^*$, respectively. Then, a Greek letter followed by a number will identify an operator whose domain is the space corresponding to the number, and whose co-domain is in the space corresponding to the Greek letter. For example, the notations

$$\mathcal{E}_{\alpha 2} \quad \text{and} \quad \mathcal{E}_{\alpha 2}\, p \tag{30}$$

will identify a map ($\mathcal{E}_{\alpha 2}$) from $\mathscr{Q}$ into $\mathscr{V}^*$ and the action of this map ($\mathcal{E}_{\alpha 2}\, p \in \mathscr{V}^*$) on the field $p \in \mathscr{Q}$, respectively. If an operators has only one subscript, that subscript identifies the space containing the range of the operator. With this in mind, let

$$\mathcal{M}_{\alpha 1} : \mathscr{V} \to \mathscr{V}^*, \qquad {}_{\mathscr{V}^*}\Big\langle \mathcal{M}_{\alpha 1}\boldsymbol{u}, \boldsymbol{v}\Big\rangle_{\mathscr{V}} := \int_\Omega \rho\, \boldsymbol{u}\cdot\boldsymbol{v}\,\mathrm{d}v \qquad \forall \boldsymbol{u}\in\mathscr{V}, \forall \boldsymbol{v}\in\mathscr{V}_0, \tag{31}$$

$$\mathcal{N}_{\alpha 1}(\boldsymbol{u}) : \mathscr{V} \to \mathscr{V}^*, \quad {}_{\mathscr{V}^*}\Big\langle \mathcal{N}_{\alpha 1}(\boldsymbol{u})\boldsymbol{w}, \boldsymbol{v}\Big\rangle_{\mathscr{V}} := \int_\Omega \rho(\nabla_x \boldsymbol{w})\boldsymbol{u}\cdot\boldsymbol{v}\,\mathrm{d}v \quad \forall \boldsymbol{u},\boldsymbol{w}\in\mathscr{V}, \forall \boldsymbol{v}\in\mathscr{V}_0, \tag{32}$$

$$\mathcal{D}_{\alpha 1} : \mathscr{V} \to \mathscr{V}^*, \qquad {}_{\mathscr{V}^*}\Big\langle \mathcal{D}_{\alpha 1}\boldsymbol{u}, \boldsymbol{v}\Big\rangle_{\mathscr{V}} := \int_\Omega \hat{\mathsf{T}}^v_{\mathrm{s}}[\boldsymbol{u}]\cdot\nabla_x\boldsymbol{v}\,\mathrm{d}v \quad \forall \boldsymbol{u}\in\mathscr{V}, \forall \boldsymbol{v}\in\mathscr{V}_0, \tag{33}$$

$$\mathcal{B}_{\beta 1} : \mathscr{V} \to \mathscr{Q}^*, \qquad {}_{\mathscr{Q}^*}\Big\langle \mathcal{B}_{\beta 1}\boldsymbol{u}, q\Big\rangle_{\mathscr{Q}} := -\int_\Omega q\,\mathrm{div}\,\boldsymbol{u}\,\mathrm{d}v \qquad \forall q\in\mathscr{Q}, \forall \boldsymbol{u}\in\mathscr{V}. \tag{34}$$

The operators defined in Eqs. (31)–(33) are found in traditional variational formulations of the Navier-Stokes equations and will be referred to as the Navier-Stokes component of the problem. As typical of other immersed methods, these operators have their support in Ω as a whole.

We now define the operator in our formulation that has its support over B but does not contain prescribed body forces or boundary terms.

$$\begin{aligned}&\mathcal{A}_\alpha(\boldsymbol{w},\boldsymbol{h}) \in \mathscr{V}^*,\ \forall \boldsymbol{w},\boldsymbol{h}\in\mathscr{Y}, \forall \boldsymbol{u}\in\mathscr{V}, \forall \boldsymbol{v}\in\mathscr{V}_0\\ &\qquad {}_{\mathscr{V}^*}\Big\langle \mathcal{A}_\alpha(\boldsymbol{w},\boldsymbol{h}), \boldsymbol{v}\Big\rangle_{\mathscr{V}} := \int_B \Big[\hat{\mathsf{P}}^e_{\mathrm{s}}[\boldsymbol{w}]\mathsf{F}^{\mathrm{T}}[\boldsymbol{h}]\cdot\nabla_x\boldsymbol{v}(\boldsymbol{x})\Big]_{\boldsymbol{x}=\boldsymbol{s}+\boldsymbol{h}(\boldsymbol{s},t)}\,\mathrm{d}V.\end{aligned} \tag{35}$$

We now define operators with support in B that express the coupling of the velocity fields defined over Ω and over B. Specifically, we have

$$\begin{aligned}&\mathcal{M}_{\gamma 3} : \mathscr{Y} \to \mathscr{Y}^*,\ \forall \boldsymbol{w},\boldsymbol{y}\in\mathscr{Y},\\ &\qquad {}_{\mathscr{Y}^*}\Big\langle \mathcal{M}_{\gamma 3}\boldsymbol{w}, \boldsymbol{y}\Big\rangle_{\mathscr{Y}} := \Phi_B \int_B \boldsymbol{w}\cdot\boldsymbol{y}(\boldsymbol{s})\,\mathrm{d}V,\end{aligned} \tag{36}$$

$$\begin{aligned}&\mathcal{M}_{\gamma 1}(\boldsymbol{w}) : \mathscr{V} \to \mathscr{Y}^*,\ \forall \boldsymbol{u}\in\mathscr{V}, \forall \boldsymbol{w},\boldsymbol{y}\in\mathscr{Y},\\ &\qquad {}_{\mathscr{Y}^*}\Big\langle \mathcal{M}_{\gamma 1}(\boldsymbol{w})\boldsymbol{u}, \boldsymbol{y}\Big\rangle_{\mathscr{Y}} := \Phi_B \int_B \boldsymbol{u}(\boldsymbol{x},t)\Big|_{\boldsymbol{x}=\boldsymbol{s}+\boldsymbol{w}(\boldsymbol{s},t)}\cdot\boldsymbol{y}(\boldsymbol{s})\,\mathrm{d}V,\end{aligned} \tag{37}$$

Finally, we define the operators that express the action of prescribed body and surface forces.

$$\begin{aligned}&\mathcal{F}_\alpha \in \mathscr{V}^*,\ \forall \boldsymbol{b}\in H^{-1}(\Omega), \forall \boldsymbol{\tau}_g \in H^{-\frac{1}{2}}(\partial\Omega_N), \forall \boldsymbol{v}\in\mathscr{V}_0\\ &\qquad {}_{\mathscr{V}^*}\Big\langle \mathcal{F}_\alpha, \boldsymbol{v}\Big\rangle_{\mathscr{V}} := \int_\Omega \rho\,\boldsymbol{b}\cdot\boldsymbol{v}\,\mathrm{d}v + \int_{\partial\Omega_N}\boldsymbol{\tau}_g\cdot\boldsymbol{v}\,\mathrm{d}a\end{aligned} \tag{38}$$

$$\begin{aligned}&\mathcal{G}_\alpha(\boldsymbol{w}) \in \mathscr{V}^*,\ \forall \boldsymbol{w}\in\mathscr{Y}, \forall \boldsymbol{b}\in H^{-1}(\Omega), \forall \boldsymbol{v}\in\mathscr{V}_0\\ &\qquad {}_{\mathscr{V}^*}\Big\langle \mathcal{G}_\alpha(\boldsymbol{w}), \boldsymbol{v}\Big\rangle_{\mathscr{V}} := \int_B \Big(\rho_{\mathrm{s}_0}(\boldsymbol{s}) - \rho J[\boldsymbol{w}]\Big)\boldsymbol{b}\cdot\boldsymbol{v}(\boldsymbol{x})\Big|_{\boldsymbol{x}=\boldsymbol{s}+\boldsymbol{w}(\boldsymbol{s},t)}\,\mathrm{d}v.\end{aligned} \tag{39}$$

In the definition of the operator $\mathcal{A}_\alpha$ in Eq. (35), the motion of the immersed solid plays a double role in that it affects the elastic response of the solid (through $\boldsymbol{w}$) as well as the map (through $\boldsymbol{h}$) functioning as a change of variables of integration. As discussed in Heltai and Costanzo (2012), it is important to separate these two roles and view $\mathcal{A}_\alpha$ as the composition of a *change of variable* operator and a Lagrangian elastic operator. To do so, we write

$$\begin{aligned}&\mathcal{S}_{\alpha\gamma}(\boldsymbol{h}) : \mathscr{H}^*_Y \to \mathscr{V}^*,\ \forall \boldsymbol{y}^*\in\mathscr{H}^*_Y, \forall \boldsymbol{h}\in\mathscr{Y}, \forall \boldsymbol{v}\in\mathscr{V}_0\\ &\qquad {}_{\mathscr{V}^*}\Big\langle \mathcal{S}_{\alpha\gamma}(\boldsymbol{h})\boldsymbol{y}^*, \boldsymbol{v}\Big\rangle_{\mathscr{V}} := {}_{\mathscr{H}^*_Y}\Big\langle \boldsymbol{y}^*, \boldsymbol{v}(\boldsymbol{x})\Big|_{\boldsymbol{x}=\boldsymbol{s}+\boldsymbol{h}(\boldsymbol{s})}\Big\rangle_{\mathscr{H}_Y},\end{aligned} \tag{40}$$

$$\begin{aligned}&\mathcal{A}_\gamma(\boldsymbol{w}) \in \mathscr{H}^*_Y,\ \forall \boldsymbol{w}\in\mathscr{Y}, \forall \boldsymbol{y}\in\mathscr{H}_Y\\ &\qquad {}_{\mathscr{H}^*_Y}\Big\langle \mathcal{A}_\gamma(\boldsymbol{w}), \boldsymbol{y}\Big\rangle_{\mathscr{H}_Y} := \int_B \hat{\mathsf{P}}^e_{\mathrm{s}}[\boldsymbol{w}]\cdot\nabla_s\boldsymbol{y}\,\mathrm{d}V.\end{aligned} \tag{41}$$

Once the operators $\mathcal{S}_{\alpha\gamma}(\boldsymbol{h})$ and $\mathcal{A}_\gamma(\boldsymbol{w})$ are defined, one can prove the following theorem (see Heltai and Costanzo, 2012):

Theorem 1 (Eulerian and Lagrangian stiffness operators of the immersed domain) *With reference to the definitions in Eqs.* (35), (40), *and* (41), *we have*

$$\mathcal{A}_\alpha(\boldsymbol{w},\boldsymbol{h}) = \mathcal{S}_{\alpha\gamma}(\boldsymbol{h})\mathcal{A}_\gamma(\boldsymbol{w}) \quad \textit{and} \quad \mathcal{S}_{\alpha\gamma}(\boldsymbol{h}) = \mathcal{M}^{\mathrm{T}}_{\gamma 1}(\boldsymbol{h})\mathcal{M}^{-1}_{\gamma 3}, \tag{42}$$

where $\mathcal{S}_{\alpha\gamma}(\boldsymbol{h})\mathcal{A}_\gamma(\boldsymbol{w})$ *and* $\mathcal{M}^{\mathrm{T}}_{\gamma 1}(\boldsymbol{h})\mathcal{M}^{-1}_{\gamma 3}$ *indicate the composition of the operators* $\mathcal{S}_{\alpha\gamma}(\boldsymbol{h})$ *and* $\mathcal{A}_\gamma(\boldsymbol{w})$ *and of the operators* $\mathcal{M}^{\mathrm{T}}_{\gamma 1}(\boldsymbol{h})$ *and* $\mathcal{M}^{-1}_{\gamma 3}$, *respectively.*

The operators defined above allow us to formally restate the overall problem described by Eqs. (25), (23), and (24) as follows:

Problem 1 (Dual formulation) *Given initial conditions* $\boldsymbol{u}_0 \in \mathscr{V}$ *and* $\boldsymbol{w}_0 \in \mathscr{Y}$, *for all* $t \in (0,T)$ *find* $\boldsymbol{u}(\boldsymbol{x},t) \in \mathscr{V}$, $p(\boldsymbol{x},t) \in \mathscr{Q}$, *and* $\boldsymbol{w}(\boldsymbol{s},t) \in \mathscr{Y}$ *such that*

$$\mathcal{M}_{\alpha 1}\boldsymbol{u}' + \mathcal{N}_{\alpha 1}(\boldsymbol{u})\boldsymbol{u} + \mathcal{D}_{\alpha 1}\boldsymbol{u} + (\mathcal{B}_{\beta 1})^{\mathrm{T}}p + \mathcal{S}_{\alpha\gamma}(\boldsymbol{w})\mathcal{A}_\gamma(\boldsymbol{w}) = \mathcal{F}_\alpha + \mathcal{G}_\alpha(\boldsymbol{w}), \tag{43}$$

$$\mathcal{B}_{\beta 1}\boldsymbol{u} = 0, \tag{44}$$

$$\mathcal{M}_{\gamma 3}\boldsymbol{w}' - \mathcal{M}_{\gamma 1}(\boldsymbol{w})\boldsymbol{u} = \boldsymbol{0}, \tag{45}$$

where $\boldsymbol{u}'(\boldsymbol{x},t) = \partial\boldsymbol{u}(\boldsymbol{x},t)/\partial t$ *and* $\boldsymbol{w}'(\boldsymbol{s},t) = \partial\boldsymbol{w}(\boldsymbol{s},t)/\partial t$.

Problem 1 can be formally presented in terms of the Hilbert space $\mathscr{Z} := \mathscr{V}\times\mathscr{Q}\times\mathscr{Y}$, and $\mathscr{Z}_0 := \mathscr{V}_0\times\mathscr{Q}\times\mathscr{Y}$ with inner product given by the sum of the inner products of the generating spaces. Defining $\mathscr{Z} \ni \xi := [\boldsymbol{u}, p, \boldsymbol{w}]^{\mathrm{T}}$ and $\mathscr{Z}_0 \ni \psi := [\boldsymbol{v}, q, \boldsymbol{y}]^{\mathrm{T}}$, then Problem 1 can be compactly stated as

Problem 2 (Grouped dual formulation) *Given an initial condition* $\xi_0 \in \mathscr{Z}$, *for all* $t \in (0,T)$ *find* $\xi(t) \in \mathscr{Z}$, *such that*

$$\langle \mathcal{F}(t,\xi,\xi'),\psi\rangle = 0, \quad \forall\psi \in \mathscr{Z}_0, \tag{46}$$

where the full expression of $\mathcal{F} : \mathscr{Z} \mapsto \mathscr{Z}_0^*$ *is defined as in Problem 1.*

The energy estimates concerning the above abstract formulation has been discussed in Heltai and Costanzo (2012) where it is shown that stability is obtained under the same assumptions that yield stability for the Navier-Stokes problems.

3 Discretization

3.1 Spatial discretization by finite elements

The fluid domain is discretized into the triangulation Ω_h and the immersed body into the triangulation B_h. Each of these triangulations consists of (closed) cells K (quadrilaterals in 2D, and hexahedra in 3D) such that:

1. $\overline{\Omega} = \cup\{K \in \Omega_h\}$, and $\overline{B} = \cup\{K \in B_h\}$;
2. Any two cells K, K' only intersect in common faces, edges, or vertices;
3. The decomposition Ω_h matches the decomposition $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$.

On Ω_h and B_h, we define the finite dimensional subspaces $\mathscr{V}_h \subset \mathscr{V}$, $\mathscr{Q}_h \subset \mathscr{Q}$, and $\mathscr{Y}_h \subset \mathscr{Y}$ as follows:

$$\mathscr{V}_h := \left\{\boldsymbol{u}_h \in \mathscr{V} \;\middle|\; \boldsymbol{u}_{h|K} \in \mathcal{P}_V(K),\ K \in \Omega_h\right\} \equiv \operatorname{span}\{\boldsymbol{v}_h^i\}_{i=1}^{N_V} \tag{47}$$

$$\mathscr{Q}_h := \left\{p_h \in \mathscr{Q} \;\middle|\; p_{h|K} \in \mathcal{P}_Q(K),\ K \in \Omega_h\right\} \equiv \operatorname{span}\{q_h^i\}_{i=1}^{N_Q} \tag{48}$$

$$\mathscr{Y}_h := \left\{\boldsymbol{w}_h \in \mathscr{Y} \;\middle|\; \boldsymbol{w}_{h|K} \in \mathcal{P}_Y(K),\ K \in B_h\right\} \equiv \operatorname{span}\{\boldsymbol{y}_h^i\}_{i=1}^{N_Y}, \tag{49}$$

where $\mathcal{P}_V(K)$, $\mathcal{P}_Q(K)$ and $\mathcal{P}_Y(K)$ are polynomial spaces of degree r_V, r_Q and r_Y respectively on the cells K, and N_V, N_Q and N_Y are the dimensions of each finite dimensional space. The pair $\mathscr{V}_h$ and $\mathscr{Q}_h$ are chosen

so that the inf-sup condition for the well-posedness of the Navier-Stokes problem (see, e.g., Brezzi and Fortin, 1991) is satisfied.

The discrete version of Problem 1 is now presented using a matrix notation. An element of a discrete space, say $\boldsymbol{u}_h \in \mathscr{V}_h$, is represented by a column vector of time dependent coefficients $u_h^j(t)$, $j = 1, \dots, N_V$, such that $\boldsymbol{u}_h(\boldsymbol{x}, t) = \sum u_h^j(t) \boldsymbol{v}_h^j(\boldsymbol{x})$, where $\boldsymbol{v}_h^j$ is the j^{th} base element of $\mathscr{V}_h$. We use the notation $M_{\alpha 1}\boldsymbol{u}_h$ to represent the multiplication of the column vector $\boldsymbol{u}_h$ by the matrix whose elements $M_{\alpha 1}^{ij}$ are

$$M_{\alpha 1}^{ij} := {}_{\mathscr{V}^*}\left\langle \mathcal{M}_{\alpha 1} \boldsymbol{v}_h^j, \boldsymbol{v}_h^i \right\rangle_{\mathscr{V}}, \tag{50}$$

where the operator in angle brackets is the one defined earlier. A similar notation is adopted for all other previously defined operators. With this notation, the duality products in the discrete spaces are indicated by simple scalar products in $\mathbb{R}^N$ (N depending on the dimension of the system at hand). Hence, using the matrix $M_{\alpha 1}$, we can write

$${}_{\mathscr{V}^*}\left\langle \mathcal{M}_{\alpha 1} \boldsymbol{u}_h, \boldsymbol{v}_h \right\rangle_{\mathscr{V}} = \boldsymbol{v}_h \cdot M_{\alpha 1} \boldsymbol{u}_h, \tag{51}$$

where the dot-product on the right hand side is the scalar product in $\mathbb{R}^{N_V}$.

Having chosen Ω_h and B_h along with $\mathscr{V}_h$, $\mathscr{Q}_h$, and $\mathscr{Y}_h$, Problem 1 is reformulated as follows:

Problem 3 *Given $\boldsymbol{u}_0 \in \mathscr{V}_h$, $\boldsymbol{w}_0 \in \mathscr{Y}_h$, for all $t \in (0, T)$, find $\boldsymbol{u}_h(t) \in \mathscr{V}_h$, $p_h(t) \in \mathscr{Q}_h$, and $\boldsymbol{w}_h(t) \in \mathscr{Y}_h$ such that*

$$M_{\alpha 1}\boldsymbol{u}_h' + N_{\alpha 1}(\boldsymbol{u}_h)\boldsymbol{u}_h + D_{\alpha 1}\boldsymbol{u}_h + (B_{\beta 1})^{\mathrm{T}} p_h + S_{\alpha\gamma}(\boldsymbol{w}_h) A_\gamma(\boldsymbol{w}_h) = F_\alpha + G_\alpha(\boldsymbol{w}_h), \tag{52}$$

$$B_{\beta 1}\boldsymbol{u}_h = 0, \tag{53}$$

$$M_{\gamma 3}\boldsymbol{w}_h' - M_{\gamma 1}(\boldsymbol{w}_h)\boldsymbol{u}_h = \boldsymbol{0}, \tag{54}$$

where $\boldsymbol{u}_h'(\boldsymbol{x}, t) = \sum [u_h^j(t)]' \boldsymbol{v}_h^j(\boldsymbol{x})$ and $\boldsymbol{w}_h'(\boldsymbol{s}, t) = \sum [w_h^j(t)]' \boldsymbol{y}_h^j(\boldsymbol{s})$, and where the prime denotes ordinary differentiation with respect to time.

In compact notation, Problem 3 can be casted as semi-discrete problems in the space $\mathscr{Z} \supset \mathscr{Z}_h := \mathscr{V}_h \times \mathscr{Q}_h \times \mathscr{Y}_h$ as

Problem 4 *Given an initial condition $\xi_0 \in \mathscr{Z}_h$, for all $t \in (0, T)$ find $\xi_h(t) \in \mathscr{Z}_h$, such that*

$$F(t, \xi_h, \xi_h') = 0, \tag{55}$$

where

$$F^i(t, \xi_h, \xi_h') := \langle \mathcal{F}(t, \xi_h, \xi_h'), \psi_h^i \rangle, \quad i = 0, \dots, N_V + N_Q + N_Y, \tag{56}$$

and $\mathcal{F}$ has the same meaning as in Eq. (46), with ψ_h^i being the basis function for the spaces $\mathscr{V}_h$, $\mathscr{Q}_h$, or $\mathscr{Y}_h$ corresponding to the given value of i.

3.2 Coupling of the fluid and immersed domains

The operators $M_{\alpha 1}$, $N_{\alpha 1}(\boldsymbol{u}_h)$, $D_{\alpha 1}$, $B_{\beta 1}$, and F_α in Problem 3 are common in variational formulations of the Navier-Stokes problem and were implemented in a standard fashion. The operator $M_{\gamma 3}$ was also implemented in a standard fashion since it is the mass matrix for $\mathscr{Y}_h$. Less common are the operators that depend nonlinearly on the motion of the immersed domain $\boldsymbol{w}$. Thus, we now discuss the practical implementation of such operators.

Let's consider, for example, the matrix $M_{\gamma 1}(\boldsymbol{w})$ contributing to the velocity coupling between the fluid and immersed domain:

$$M_{\gamma 1}^{ij}(\boldsymbol{w}_h) = {}_{\mathscr{Y}_h^*}\left\langle \mathcal{M}_{\gamma 1}(\boldsymbol{w}_h)\boldsymbol{v}_h^j, \boldsymbol{y}_h^i \right\rangle_{\mathscr{Y}_h} = \Phi_B \int_B \boldsymbol{v}_h^j(\boldsymbol{x})\big|_{\boldsymbol{x}=\boldsymbol{s}+\boldsymbol{w}_h(\boldsymbol{s},t)} \cdot \boldsymbol{y}_h^i(\boldsymbol{s})\, \mathrm{d}V. \tag{57}$$

The above integral is computed by summing contributions from each cell K of B_h. Each of these contributions is a sum over the N_Q quadrature points. We observe that the integrand $\boldsymbol{y}_h^i(\boldsymbol{s})$ is supported over the triangulation of B_h but the functions $\boldsymbol{v}_h^j(\boldsymbol{x})$ (with $\boldsymbol{x} = \boldsymbol{s} + \boldsymbol{w}_h(\boldsymbol{s}, t)$) are supported over the

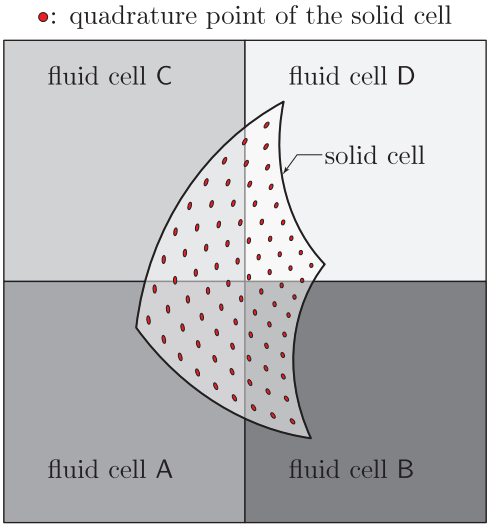


Figure 2: Cells denote as A–D represent a four-cell patch of the triangulation of the fluid domain. The cell denoted as "solid cell" represents a cell of the triangulation of the immersed solid domain that is contained in the union of cells A–D of the fluid domain. The filled dots represent the quadrature points of the quadrature rule adopted to carry out integration over the cells of the immersed domain.

triangulation Ω_h. Therefore, the construction of operators like $M^{ij}_{\gamma 1}(\boldsymbol{w}_h)$ draws information from two independent triangulations. In our code, we start by determining the position of the quadrature points of the immersed element, both relative to the reference unit element and relative to the global coordinate system adopted for the calculation, through the mappings:

$$\boldsymbol{s}_K : \hat{K} := [0,1]^d \mapsto K \in B_h, \tag{58}$$

$$I + \boldsymbol{w}_h : K \mapsto \text{solid cell}. \tag{59}$$

These maps allow us to determine the global coordinates of the quadrature points. These coordinates are then passed to a search algorithm that identifies the cells in Ω_h that contain the points in question. In turn, this identification allows us to evaluate the functions $\boldsymbol{v}^j_h$. The overall operation is illustrated in Fig. 2 where we show a cell of B_h straddling four cells of Ω_h denoted fluid cells A–D. The quadrature points over the solid cell are denoted by filled circles. The contribution to the integral in Eq. (57) due to the solid cell is then computed by summing the partial contributions corresponding to each of the fluid cells intersecting the solid cell in question:

$$\begin{aligned} M^{ij}_{\gamma 1}(\boldsymbol{w}_h) &= \sum_{K \in B_h} \int_K \boldsymbol{v}^j_h(\boldsymbol{x})\big|_{\boldsymbol{x}=\boldsymbol{s}+\boldsymbol{w}_h(\boldsymbol{s},t)} \cdot \boldsymbol{y}^i_h(\boldsymbol{s})\,\mathrm{d}V, \\ &\sim \sum_{K \in B_h} \sum_{q=1}^{N_{K,q}} \boldsymbol{v}^j_h(\boldsymbol{x})\big|_{\boldsymbol{x}=\boldsymbol{s}_{K,q}+\boldsymbol{w}_h(\boldsymbol{s}_{K,q},t)} \cdot \boldsymbol{y}^i_h(\boldsymbol{s}_{K,q})\omega_{K,q}, \end{aligned} \tag{60}$$

where $\boldsymbol{s}_{K,q}$ is the image of the q-th quadrature point under the mapping $\boldsymbol{s}_K$, and $\omega_{K,q}$ is the corresponding quadrature weight. The implementation of an efficient search algorithm responsible for identifying the fluid cells intersecting an individual solid cell is the only technically challenging part of the procedure. We use the built-in facilities of the `deal.II` library to perform this task. Once the fluid cells containing the quadrature points of a given solid cell are found, we determine the value of $\boldsymbol{v}^j_h$ at the quadrature points using the interpolation infrastructure inherent in the finite element representation of fields defined over Ω_h. The `deal.II` C++ class we use for this implementation is the `FEFieldFunction`.

3.3 Time discretization

Equation (55) represents a system of nonlinear differential algebraic equations (DAE), which we solve using a Newton iteration. In the code accompanying this paper, the time derivative ξ' is approximated

Table 1: Content of the provided `zip` archive.

`CMakeLists.txt`:	cmake file;
`source/*.cc`:	source files;
`include/*.h`:	header files;
`immersed_fem.prm`:	default parameter file;
`meshes/`:	directory containing a collection of input mesh files, in `UCD` format;
`prms/`:	examples parameter files;
`out/`:	empty directory, in which output files will be written;
`doc/`:	`Doxygen` documentation directory;
`Doxygen.in`:	source file to generate the `Doxygen` configuration file.

very simply via an implicit-Euler scheme:

$$\xi_n' = h^{-1}\Big(\xi_n - \xi_{n-1}\Big), \tag{61}$$

where ξ_n and ξ_n' are the computed approximations to $\xi(t_n)$ and $\xi'(t_n)$, respectively, and the step size $h = t_n - t_{n-1}$ is kept constant throughout the computation. Although not second order accurate, this time stepping scheme is asymptotically stable.

The application of the implicit-Euler scheme in Eq. (61) to the DAE system in Eq. (55) results in a nonlinear algebraic system to be solved at each step:

$$G(\xi_n) := F\Big(t_n, \xi_n, h^{-1}\Big(\xi_n - \xi_{n-1}\Big)\Big) = 0. \tag{62}$$

The nonlinear system in Eq. (62) is solved via Newton iterations. This leads to a linear system for each Newton correction, of the form

$$J[\xi_{n,m+1} - \xi_{n,m}] = -G(\xi_{n,m}), \tag{63}$$

where $\xi_{n,m}$ is the mth approximation to ξ_m. Here J is some approximation to the system's Jacobian

$$J = \frac{\partial G}{\partial \xi} = \frac{\partial F}{\partial \xi} + \alpha \frac{\partial F}{\partial \xi'}, \tag{64}$$

where $\alpha = 1/h$. In our finite element implementation, we assemble the residual $G(\xi_{n,m})$ at each Newton correction. The implementation of the residual vector is based on the formulation presented in Problem 3. However, this formulation makes the determination of the corresponding Jacobian rather involved due to the structure of the operator $\mathcal{S}_{\alpha\gamma}(\boldsymbol{w})$ (see Eq. (42)). Hence, we have implemented a Newton-Raphson iteration based on an approximate Jacobian. With reference to Theorem 1 and Eq. (43), the Jacobian we assemble is the exact Jacobian of a formulation in which the operator product $\mathcal{S}_{\alpha\gamma}(\boldsymbol{w})\mathcal{A}_{\gamma}(\boldsymbol{w})$ is replaced by the operator $\mathcal{A}_{\alpha}(\boldsymbol{w}, \boldsymbol{h})$ defined in Eq. (35). In the code accompanying this paper, the final system is solved using the direct solver provided by the `UMFPACK` package (see Davis, 2004).

4 Implementation

4.1 Source files and library requirements

The included source code is based on the `deal.II` library 8.0 and up (see Bangerth et al., 2006, 2013). In what follows, we assume that the user has installed the `deal.II` library in some directory and that the environment variable `DEAL_II_DIR` has been set pointing to the installation path. For the program to work properly, `deal.II` should be configured with UMFPACK support ((Davis, 2004)).

An additional `GIT` repository of the source code is availble at the address
`https://bitbucket.org/heltai/ans-ifem`.

Table 1 provides a summary of the distributed files and directories. Once the code is unzipped in a given directory, it can be compiled by simply typing `cmake .; make;` at the command line prompt, and run with

```
./ifem [optional_parameter_file.prm]
```


If the file `parameter_file.prm` does not exist, the program creates one with default values, which can then be suitably modified by the user. We distribute all the parameter files that were used to produce the results in Section 5 along with the needed mesh files. These can be found in the directories `prms` and `meshes`, respectively. If the program is run without arguments, it is assumed that the problem parameters are those in the file `parameter_file.prm`. As mentioned earlier, if the file in question does not already exist, a default copy will be created.

If `doxygen` is available, a complete and browsable documentation of the source code itself can be generated by enabling the `cmake` option `BUILD_DOCUMENTATION`, and typing `make docs` at the command line prompt. If you download the file `http://www.dealii.org/developer/doxygen/deal.tag` to the program directory, then the online `deal.II` documentation will be embedded in the `Doxygen` documentation of this program.

The program documentation is built in the `./doc/html/` subdirectory.

4.2 Parameter and input files

The behavior of the program is controlled by the `IFEMParameters<dim>` class, which are defined in `./include/ifem_parameters.h` and `./source/ifem_parameters.cc` which is derived from the `deal.II` class `ParameterHandler` and is used to define and to read from a file all the problem parameters that the user can set.

The following is a sample parameter file that can be used with our code.

Bash code

```
# Listing of Parameters
# ---------------------

# Time Stepping
set Final t                                 = 1
set Delta t                                 = .1
set Interval (of time-steps) between output = 1

# Non linear solver
set Force J update at step beginning        = false
set Update J cont                           = false
set Semi-implicit scheme                    = true
set Use spread operator                     = true

# Constitutive models available are: INH_0: incompressible Neo-Hookean with
# P^{e} = mu (F - F^{-T}); INH_1: incompressible neo-Hookean with P^{e} = mu F;
# CircumferentialFiberModel: incompressible with P^{e} = mu F
# (e_{\theta} \otimes e_{\theta}) F^{-T}; this is suitable for annular solid
# comprising inextensible circumferential fibers
set Solid constitutive model                = INH_0
set Density                                 = 1
set Viscosity                               = 1
set Elastic modulus                         = 1
# Dimensional constant for the velocity equation
set Phi_B                                   = 1

# Solid mesh information
set Solid mesh                              = meshes/solid_square.inp
set Solid refinement                        = 1

# Fluid mesh information
set Fluid mesh                              = meshes/fluid_square.inp
set Fluid refinement                        = 4
set All Dirichlet BC                        = true
set Dirichlet BC indicator                  = 1
set Velocity finite element degree          = 2
# Select between FE_Q (Lagrange finite element space of continuous, piecewise
# polynomials) or FE_DGP(Discontinuous finite elements based on Legendre
```

```
# polynomials) to approximate the pressure field
set Finite element for pressure             = FE_DGP
set Fix one dof of p                        = false

# Base name used for the output files
set Output base name                        = out/square

# This section is used only when the constitutive model is set to
# CircumferentialFiberModel
subsection Equilibrium Solution of Ring with Circumferential Fibers
  set Any edge length of the (square) control volume = 1.
  set Inner radius of the ring                       = 0.25
  set Width of the ring                              = 0.0625
  set x-coordinate of the center of the ring         = 0.5
  set y-coordinate of the center of the ring         = 0.5
end


subsection W0
  # Sometimes it is convenient to use symbolic constants in the expression
  # that describes the function, rather than having to use its numeric value
  # everywhere the constant appears. These values can be defined using this
  # parameter, in the form 'var1=value1, var2=value2, ...'.
  #
  # A typical example would be to set this runtime parameter to
  # 'pi=3.1415926536' and then use 'pi' in the expression of the actual
  # formula. (That said, for convenience this class actually defines both 'pi'
  # and 'Pi' by default, but you get the idea.)
  set Function constants  =

  # The formula that denotes the function you want to evaluate for particular
  # values of the independent variables. This expression may contain any of
  # the usual operations such as addition or multiplication, as well as all of
  # the common functions such as 'sin' or 'cos'. In addition, it may contain
  # expressions like 'if(x>0, 1, -1)' where the expression evaluates to the
  # second argument if the first argument is true, and to the third argument
  # otherwise. For a full overview of possible expressions accepted see the
  # documentation of the fparser library.
  #
  # If the function you are describing represents a vector-valued function
  # with multiple components, then separate the expressions for individual
  # components by a semicolon.
  set Function expression = 0; 0

  # The name of the variables as they will be used in the function, separated
  # by commas. By default, the names of variables at which the function will
  # be evaluated is 'x' (in 1d), 'x,y' (in 2d) or 'x,y,z' (in 3d) for spatial
  # coordinates and 't' for time. You can then use these variable names in
  # your function expression and they will be replaced by the values of these
  # variables at which the function is currently evaluated. However, you can
  # also choose a different set of names for the independent variables at
  # which to evaluate your function expression. For example, if you work in
  # spherical coordinates, you may wish to set this input parameter to
  # 'r,phi,theta,t' and then use these variable names in your function
  # expression.
  set Variable names      = x,y,t
end


subsection force
  set Function constants  =
  set Function expression = 0; 0; 0
```



```
  set Variable names      = x,y,t
end


subsection u0
  set Function constants  =
  set Function expression = 0; 0; 0
  set Variable names      = x,y,t
end


subsection ug
  set Function constants  =
  set Function expression = if(y>.99, 1, 0); 0; 0
  set Variable names      = x,y,t
end
```

At the beginning of the parameter file we find specifications for the time stepper and for the nonlinear solver. In addition, we find information on the constitutive behavior of both the fluid and the immersed solid.

The user can specify the names of the files containing the meshes for the control volume and the immersed solid, along with the initial global refinement level for each mesh, in the parameter file. In the section pertaining to the control volume, the user can also set the degree of the finite element spaces for the fluid velocity as well as the type of the finite element space for the fluid pressure. The type and degree of the finite element space for the displacement of the immersed domain are automatically set to be the same as those for the velocity of the fluid. A degree greater than or equal to two should be selected for the finite element space of the velocity so as to ensure proper inf-sup stability. The degree of the pressure space is then automatically set to be one less than that for the velocity.

In the second part of the parameter file, the user can specify the initial and boundary values of the solution as well as the external body forces. Here `W0` denotes the initial value of the displacement of the immersed domain, `force` denotes the external body force field, `u0` is the initial condition for the velocity and the pressure fields and `ug` is the Dirichlet boundary condition (here configured for a *lid-cavity* problem).

The above file, for example, generates the parameters for a *lid-cavity* problem inside a square control volume (read from `meshes/fluid_square.inp`), with an immersed solid whose mesh is given in `meshes/solid_square.inp`.

The full documentation of the class can be accessed through `Doxygen`.

4.3 Code structure

The structure of our program follows closely the structure of most tutorial programs in the `deal.II` library, to which we refer for further explanations and examples. The main class of the program is the class `ImmersedFEM<dim>`, in which all objects and methods to solve the problem at hand are defined (including an object of type `IFEMParameters<dim>`). This class is defined in `./include/immersed_fem.h` and implemented in `./source/immersed_fem.cc`.

Execution of the solution is triggered in the method `run()`, which starts the time stepping scheme of the DAE system described in Section 3.3, and controls the convergence of the Newton iteration scheme for the solution of system Eq. (63).

Detailed documentation of the code has been embodied in the code itself, and can be automatically generated with `Doxygen`. Here we only briefly overview the main ideas behind the use of `deal.II` for immersed methods.

Due to the nature of the method, two different sets of objects are needed to describe the triangulation, the degrees of freedom, etc., of both the fluid and the immersed domains. In the code, objects pertaining to the fluid have been denoted with the suffix `_f`, whereas objects pertaining to the immersed solid have been denoted with the suffix `_s`. For example, `tria_s` and `tria_f` are the two `Triangulation<dim>` objects of the solid and fluid domains, respectively.

In the code, solution vectors and residuals are constructed as
`BlockVector<double>` objects and the Jacobian matrix is constructed as a
`BlockSparseMatrix<double>` object. This has been done to reflect the logical splitting of these entities between the fluid and the solid, and to allow access to the individual blocks at the same time. We split the vectors and matrices into two and four parts, respectively. The block vectors storing the overall solutions at the current time step and at the previous time step are called `xi` and `previous_xi`, respectively. The first block of these block vectors pertains to the fluid and it is of size `n_dofs_up`, which is also equal to `dh_f.n_dofs()`. The second block pertains to the solid and has a size of `n_dofs_W`, which is also equal to `dh_s.n_dofs()`.

The various tutorial examples of the `deal.II` library describe in an exhaustive manner how to treat a single triangulation and a single degrees-of-freedom handler for both fluid-only problems (e.g., the example program `step-35`) and elasticity-only problems (e.g., the example program `step-44`). The most delicate part of immersed methods, however, requires the coupling between a fixed background mesh (the fluid), and a moving and deforming foreground mesh (the elastic solid). The deformation of the foreground mesh is achieved very effectively through the
`MappingQEulerian<dim,spacedim>` class, which uses the information stored in the displacement vector to automatically compute the deformed positions of the mesh and of the quadrature points in a Lagrangian way. Notice that while the name suggests an Eulerian description, this object in reality performs a Lagrangian iso-parametric transformation from the reference grid, stored in `tria_s`, to the current configuration of the solid via the deformation vector $\boldsymbol{w}$. Details on construction and use of this class are given in Section 4.3.1.

Evaluation of the quadrature points of the solid on the background fluid mesh is achieved through the class `FEFieldFunction<dim>`, which allows one to evaluate the values of finite element fields at *arbitrary* points. In particular, its method
`FEFieldFunction<dim>::compute_point_locations` is the one that returns the lists required to compute the coupling integrals (see Section 3.2) and is used both in the creation of the sparsity pattern that features the coupling between the degrees of freedom of the fluid and the immersed solid (see Section 4.3.2), as well as in the assembling of the residual vector and the Jacobian matrix of the DAE system (see Section 4.3.3).

4.3.1 Immersed map Whenever it is necessary to compute the deformed configuration of the solid, an iso-parametric displacement is superimposed on each node of the triangulation of the solid. This process is transparent to the user and is performed by the class `MappingQEulerian<dim>`. In our code, we pass an object of this class as an argument to all the standard `deal.II` classes which are involved in computing the finite element values and their gradients on the deformed cells of the triangulation of the solid. In the following code snippet we illustrate this process that takes place at the beginning of the computation of the residual and of the Jacobian:

C++ code

```
1  ...
2  MappingQEulerian<dim> * mapping;
3  ...
4
5  template <int dim>
6  void
7  ImmersedFEM<dim>::residual_and_or_Jacobian(...)
8  {
9    if(mapping != NULL) delete mapping;
10
11   if(par.semi_implicit == true)
12       mapping = new MappingQEulerian<dim, Vector<double>, dim>
13         (par.degree, previous_xi.block(1), dh_s);
14   else
15     mapping = new MappingQEulerian<dim, Vector<double>, dim>
16         (par.degree, xi.block(1), dh_s);
17
18   ...
19
```



```
  FEValues<dim,dim> fe_v_s_mapped (*mapping,
                                   fe_s,
                                   quad_s,
                                   update_quadrature_points);
  ...
}
```

This code snippet illustrates how to instantiate an iso-parametric mapping based on the current displacement solution, given by `xi.block(1)` or on the previous displacement solution `previous_xi.block(1)`. We refer to the `deal.II` documentation of the class `MappingQEulerian` for further details on the meaning of each of the arguments passed to the constructor of the class. Here it is important to notice that, once a mapping from the reference configuration to the deformed configuration is available, it is used in all instantiations of those classes which compute the values and the gradients of the basis functions on the deformed configuration (i.e., `FEValues<dim,dim>`).

Setting the parameter "`Semi-implicit scheme`" to `true` in the parameter file (see Section 4.2) will set the variable `par.semi_implicit` to `true` in the above snippet of code. The consequence of this choice is that, while the elastic response of the solid is computed at its current configuration, i.e., the Piola-Kirchhoff stress is still computed using `xi.block(1)`, the body force corresponding to this stress is applied to the fluid surrounding the body at the location `previous_xi.block(1)`, instead of `xi.block(1)`. In other words, the operator defined in Eq. (35), and later split in the change of variable operator and in the Lagrangian elastic operator in Theorem 1 (see Eq. (42)), will use `xi.block(1)` in place of the variable $\boldsymbol{w}$ and `previous_xi.block(1)` in place of the variable $\boldsymbol{h}$.

This splitting preserves the consistency of the method, and removes the nonlinearity due to the change of variable from the system at the cost of introducing a CFL condition on the time stepping scheme (for a more detailed discussion on this topic see Heltai, 2008; Boffi et al., 2007), which ceases to be asymptotically stable.

4.3.2 Sparsity pattern A `SparsityPattern` is a `deal.II` object which stores the nonzero entries of a sparse matrix. Since we are using a `BlockSparseMatrix<double>` class to store the Jacobian of the DAE system, we need a `SparsityPattern` for each of the sub-blocks of this block. The snippet of code that generates the coupling sparsity pattern is given by

C++ code

```
  FEFieldFunction<dim, DoFHandler<dim>, Vector<double> >
        up_field (dh_f, tmp_vec_n_dofs_up);

  vector< typename DoFHandler<dim>::active_cell_iterator > cells_f;
  vector< vector< Point< dim > > > qpoints_f;
  vector< vector< unsigned int> > maps;
  vector< unsigned int > dofs_f(fe_f.dofs_per_cell);
  vector< unsigned int > dofs_s(fe_s.dofs_per_cell);

  typename DoFHandler<dim,dim>::active_cell_iterator
    cell_s = dh_s.begin_active(),
    endc_s = dh_s.end();

  FEValues<dim,dim> fe_v_s(immersed_mapping, fe_s, quad_s,
                           update_quadrature_points);

  CompressedSimpleSparsityPattern sp1(n_dofs_up, n_dofs_W);
  CompressedSimpleSparsityPattern sp2(n_dofs_W , n_dofs_up);

  for(; cell_s != endc_s; ++cell_s)
    {
      fe_v_s.reinit(cell_s);
      cell_s->get_dof_indices(dofs_s);
      vector< Point< dim > >  &qpoints_s
                =  fe_v.get_quadrature_points();

```



```
27      up_field.compute_point_locations (qpoints_s,
28                                        cells_f, qpoints_f, maps);
29      for(unsigned int c=0; c<cells_f.size(); ++c)
30           {
31             cells_f[c]->get_dof_indices(dofs_f);
32             for(unsigned int i=0; i<dofs_f.size(); ++i)
33               for(unsigned int j=0; j<dofs_s.size(); ++j)
34                 {
35                   sp1.add(dofs_f[i],dofs_s[j]);
36                   sp2.add(dofs_s[j],dofs_f[i]);
37                 }
38           }
39    }
```

Here an `FEFieldFunction<dim>` object is constructed with a dummy finite element vector field (`tmp_vec_n_dofs_up`) to have access to its member function `FEFieldFunction<dim>::compute_point_locations`. This member function takes as input the location of the quadrature points in each solid cell `qpoints_s` (computed with the `FEValues<dim>` object `fe_v_s`, initialized with the mapping described in Section 4.3.1) and fills up a series of vectors, which allow the computation of the integrals as explained in Section 3.2.

These vectors are respectively:

- `cells_f`: the vector of all *fluid cells* containing at least one of the quadrature points of the immersed domain;
- `qpoints_f`: a vector of the same length as `cells_f`, containing the custom vector of quadrature points in the *fluid* reference (unit) cell, which gets transformed via the *fluid mapping* to the subset of solid quadrature points `qpoints_s` (that happen to be in the cell in question);
- `maps`: a vector of the same length as `cells` and `qpoints_f`, which contains vectors of indices of the solid quadrature points to which the fluid quadrature points refer to, i.e., `qpoints_f[i][j]` is mapped by the *fluid mapping* to the same physical location to which the point `qpoints_s[maps[i][j]]` is mapped by the *solid mapping*.

In the construction of the sparsity patterns, only the first vector, `cells_f`, is used since we only need to know which degrees of freedom are coupled. In particular, all degrees of freedom in the fluid cells contained in `cells_f` will couple with the solid cell identified with the cell iterator `cell_s`. These couplings are computed in the innermost for-loop.

4.3.3 Residual and Jacobian Similarly to what happens for the computation of the sparsity pattern, we use an object of type `FEFieldFunction<dim>` to compute the location of the quadrature points of the immersed solid within the fluid cells. Assembly of the coupling matrices is then possible by looping over all solid cells, and constructing custom quadrature formulas to use with the fluid cells in order to compute the integrals explained in Section 3.2. The following snippet of code explains the most relevant points:

C++ code

```
1   // Loop over solid cells
2   for(cell_s = dh_s.begin_active(); cell_s != endc_s; ++cell_s)
3   {
4     fe_v_s_mapped.reinit(cell_s);
5 ...
6     up_field.compute_point_locations (fe_v_s_mapped.get_quadrature_points(),
7                                       fluid_cells,
8                                       fluid_qpoints,
9                                       fluid_maps);
10 ...
11
12 // Cycle over all of the fluid cells that happen to contain some of
```



```
// the the quadrature points of the current solid cell.
      for(unsigned int c=0; c<fluid_cells.size(); ++c)
        {
          fluid_cells[c]->get_dof_indices (dofs_f);

// Local FEValues of the fluid
          Quadrature<dim> local_quad (fluid_qpoints[c]);
          FEValues<dim> local_fe_f_v (fe_f,
                                      local_quad,
                                      update_values |
                                      update_gradients |
                                      update_hessians);
          local_fe_f_v.reinit(fluid_cells[c]);
...

// Use the local_fe_f_v as you would normally do:
          for(unsigned int i=0; i<fe_s.dofs_per_cell; ++i)
            {
              unsigned int wi = i + fe_f.dofs_per_cell;
              comp_i = fe_s.system_to_component_index(i).first;
              for(unsigned int q=0; q<local_quad.size(); ++q)
                {
                  unsigned int &qs = fluid_maps[c][q];

...

                  local_res[wi] -= par.Phi_B
                                   * local_up[q](comp_i)
                                   * fe_v_s.shape_value(i,qs)
                                   * fe_v_s.JxW(qs);
...
```

In the snippet above, we show how the term $-\int_K \boldsymbol{u}(\boldsymbol{s}+\boldsymbol{w}(\boldsymbol{s},t),t)\cdot\boldsymbol{y}(\boldsymbol{s})d\boldsymbol{s}$ is assembled in practice. The point locations are computed by `up_field.compute_point_locations`. We loop over the filled vectors to compute the coupling between each of the fluid cells, `fluid_cells[c]`, and the solid cell `cell_s`. Since the computed quadrature points in the fluid reference cells are not standard (i.e., they are not located at Gauss quadrature points), we need to create a custom quadrature formula containing the points of interest (the object `local_quad`, initialized with `fluid_qpoints[c]`) as well as an `FEValues` object, `local_fe_f_v`, to calculate values and gradients of the fluid shape functions at the solid quadrature points.

These custom `FEValues` are then initialized with the fluid cell `fluid_cells[c]`. Notice that the correspondence between the indexing in the solid quadrature points and in the fluid custom quadrature is given by `fluid_maps[c][q]`. The rest follows the standard usage of the `deal.II` library, as can be found in any of the `deal.II` example programs.

4.3.4 Visualization During execution, the program will produce two output files for each time step, containing respectively the *fluid* solution and the *solid* solution at each time iteration.

The names of these files are decided using the `Output base name` option of the configuration file. For example, setting this value to `out/square` for the automatically generated parameter file, one obtains the following files in the `out` subdirectory:

```
square-fluid-00000.vtu
square-fluid-00001.vtu
square-fluid-00002.vtu
...
square_global.gpl
square_param.prm
square-solid-00000.vtu
```



```
square-solid-00001.vtu
square-solid-00002.vtu
...
```

The `vtu` files are *binary vtk* files, which can be opened, for example, with VisIt (see Childs et al., 2005). If "smart select" is checked during file opening, then VisIt recognises each group of files as belonging to one single simulation, with the ordinal number in the name identifying the "time" in the simulation.

The background fluid files should be opened first, by selecting the files `*-fluid-*.vtu` in the "Open Files" menu. The velocity data is saved as a vector variable named `v`, while the pressure is saved a scalar variable named `p`. Fluid data can be added at will (such as, for example, streamlines, isolines of the velocity magnitude, and so on).

Once the user is satisfied with the output of the velocity and pressure fields, it is possible to open at the same time the solid files `*-solid-*.vtu`. The best way to visualize the fluid structure interaction is to add a plot of the solid mesh, by selecting `Add->Mesh->mesh` when the solid files are active. As soon as we do this, VisIt should pop a dialog asking

```
Would you like to create a "Index"
database correlation for the following databases?

localhost:~/step-feibm/out/square-solid-*.vtu database
localhost:~/step-feibm/out/square-fluid-*.vtu database
```

Answering "yes" will create a correlation between the time steps in the two files, and they will be updated at the same time when using the navigation buttons in the files dialogs. If VisIt does not ask about the correlation, one should be created manually using the menu `Control->Database correlations`.

Once these operations are done, the user should be able to watch the evolution of the immersed body inside the fluid, just as in the plots in this paper, which were obtained using VisIt following the instructions in this section.

5 Numerics

We present in this section two numerical experiments that highlight the aspects of the accuracy and error convergence properties as well as the volume conservation feature of our numerical method.

5.1 Static equilibrium of an annular solid comprising circumferential fibers and immersed in a stationary fluid

This numerical test is motivated by the ones presented in Boffi et al. (2008); Griffith and Luo (2012). The objective of this test is to compute the equilibrium state of an initially undeformed thick annular cylinder submerged in a stationary incompressible fluid that is contained in a rigid prismatic box having a square cross-section. Our simulation is two-dimensional and comprises an annular solid with inner radius R and thickness w, and filled with a stationary fluid that is contained in a square box of edge length l (see Fig. 3). In this setting, the reference and the deformed configurations of the annular solid can be conveniently described using the polar coordinate systems, whose origins coincide with the center of the annulus and whose unit vectors are given by $(\hat{\boldsymbol{u}}_R, \hat{\boldsymbol{u}}_\Theta)$ and $(\hat{\boldsymbol{u}}_r, \hat{\boldsymbol{u}}_\theta)$, respectively. This ring is located coaxially with respect to that of the box and it is subjected to the hydrostatic pressure of the fluid p_i and p_o at its inner and outer walls, respectively. Negligible body forces act on the system and there is no inflow or outflow of fluid across the walls of the box. Since both the solid and the fluid are incompressible, it is expected that neither the annulus nor the fluid will move at all. Therefore, the problem reduces to determining the equilibrium solution for the Lagrange multiplier field p. The elastic behavior of the ring is governed by a continuous distribution of concentric fibers lying in the circumferential direction. The first Piola-Kirchhoff stress for the ring is then given by

$$\hat{\mathsf{P}} = -p_s \mathsf{F}^{-T} + \mu^e \mathsf{F}\hat{\boldsymbol{u}}_\Theta \otimes \hat{\boldsymbol{u}}_\Theta, \tag{65}$$

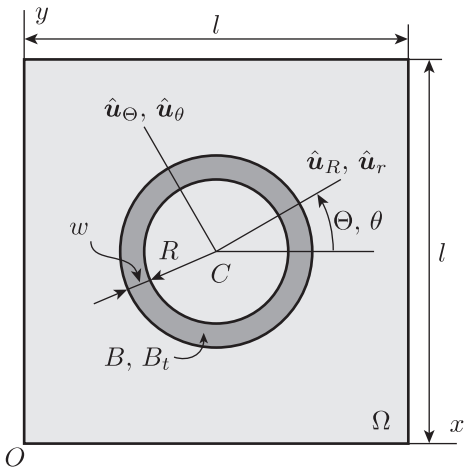


Figure 3: The reference and deformed configurations of a ring immersed in a square box filled with stationary fluid

where μ^e is a constant and p_s is the Lagrange multiplier that enforces incompressibility of the ring. As alluded to earlier, the reference configuration and the deformed configuration of the ring must coincide because of incompressibility, and the fact that the deformation of the ring must be axisymmetric in nature. For $\mathsf{F} = \mathsf{I}$ the constitutive response for the Cauchy stress can then be written as

$$\hat{\mathsf{T}}_\mathrm{s} = -p_\mathrm{s}\mathsf{I} + \mu^e \hat{\boldsymbol{u}}_\theta \otimes \hat{\boldsymbol{u}}_\theta, \tag{66}$$

where, for the deformation at hand, $\hat{\boldsymbol{u}}_\theta = \hat{\boldsymbol{u}}_\Theta$. The balance of linear momentum for the ring can be obtained from Eq. (2) as

$$-\operatorname{grad}(p_\mathrm{s}) + \mu^e \operatorname{div}(\hat{\boldsymbol{u}}_\theta \otimes \hat{\boldsymbol{u}}_\theta) = \boldsymbol{0}. \tag{67}$$

Noting that $\operatorname{grad}(p_\mathrm{s}) = (\partial p_s/\partial r)\,\hat{\boldsymbol{u}}_r + (1/r)(\partial p_s/\partial\theta)\hat{\boldsymbol{u}}_\theta$, and that $\operatorname{div}(\hat{\boldsymbol{u}}_\theta \otimes \hat{\boldsymbol{u}}_\theta) = -(1/r)\hat{\boldsymbol{u}}_r$, Eq. (67) can be rewritten as

$$-\frac{\partial p_\mathrm{s}}{\partial r} - \frac{\mu^e}{r} = 0 \qquad \text{and} \qquad \frac{\partial p_\mathrm{s}}{\partial \theta} = 0. \tag{68}$$

From Eq. (68), it can be concluded that the Lagrange multiplier enforcing incompressibility p_s is an axisymmetric function of the form

$$p_\mathrm{s} = c - \mu^e \ln\left(\frac{r}{R}\right), \tag{69}$$

where c is a constant. The satisfaction of the traction boundary conditions at the inner and outer walls of the ring demand that $p_\mathrm{s}|_{r=R} = p_i$ and $p_\mathrm{s}|_{r=R+w} = p_o$ and hence we can obtain that

$$p_\mathrm{s} = p_i - \mu^e \ln\left(\frac{r}{R}\right), \qquad p_o = p_i - \mu^e \ln\left(1 + \frac{w}{R}\right) \tag{70}$$

Note that Lagrange multiplier p defined over the control volume corresponds to p_s in the region occupied by the solid. By constraining the average value of p over the entire control volume to be zero we arrive at the following solution for the equilibrium problem:

$$p = \begin{cases} p_o = -\frac{\pi\mu^e}{2l^2}\left((R+w)^2 - R^2\right) & \text{for} \quad R + w \le r, \\ p_s = \mu^e \ln(\frac{R+w}{r}) - \frac{\pi\mu^e}{2l^2}\left((R+w)^2 - R^2\right) & \text{for} \quad R < r < R + w, \\ p_i = \mu^e \ln(1 + \frac{w}{R}) - \frac{\pi\mu^e}{2l^2}\left((R+w)^2 - R^2\right) & \text{for} \quad r \le R, \end{cases} \tag{71}$$

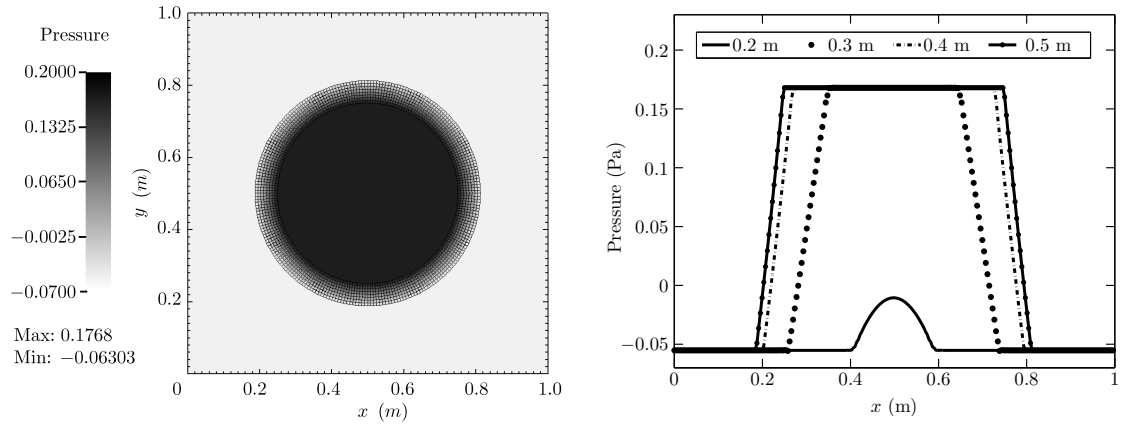
with velocity of fluid $\boldsymbol{u} = \boldsymbol{0}$ and the displacement of the solid $\boldsymbol{w} = \boldsymbol{0}$. Note that Eq. (71) is different from Eq. (69) of Boffi et al. (2008), where p varies linearly with r (we believe this to be in error).

For all our numerical simulations we have used $R = 0.25\,\mathrm{m}$, $w = 0.06250\,\mathrm{m}$, $l = 1.0\,\mathrm{m}$ and $\mu^e = 1\,\mathrm{Pa}$ and for these values we obtain $p_i = 0.16792\,\mathrm{Pa}$ and $p_o = -0.05522\,\mathrm{Pa}$ using Eq. (71). We have used

$\rho = 1.0\,\mathrm{kg/m^3}$, dynamic viscosity $\mu = 1.0\,\mathrm{Pa{\cdot}s}$, and time step size $h = 1{\times}10^{-3}\,\mathrm{s}$ in our tests. For all our numerical tests we have used $Q2$ elements to represent $\boldsymbol{w}$ of the solid, whereas we have used (i) $Q2/P1$ elements, and (ii) $Q2/Q1$ elements to represent $\boldsymbol{v}$ and p over the control volume. We present a sample profile of p over the entire control volume and its variation along different values of y, after one time step, in Fig. 4 and Fig. 5 for $Q2/P1$ and $Q2/Q1$ elements, respectively.

We assess the convergence property of our numerical scheme by obtaining the convergence rate of the error between the exact and the numerical solutions of this equilibrium problem. The order of the rate of convergence (see, Tables 2 and 3 for $Q2/P1$ and $Q2/Q1$ elements, respectively) is 2.5 for the L^2 norm of the velocity, 1.5 for the H^1 norm of the velocity and 1.5 for the L^2 norm of the pressure which matches the rates presented in Boffi et al. (2008). In all these numerical tests we have used 1856 cells with 15776 DoFs for the solid. These converge rates are optimal, since the exact pressure solution, although regular in each subdomain separately, can be shown to be globally in $H^{3/2}(\Omega)$, since its gradient has a jump discontinuity supported along a surface of co-dimension one.

The parameter files used for these tests can be found under the directory
`prms/RingEqm_XXX_fref_Y_param.prm`, where `XXX` is either `dgp` or `feq` and `Y` is 4, 5, 6 or 7, according to the type of pressure finite element and to the fluid refinement level. The tests can be run under the `step-feibm` directory, by typing
`./step-feibm prms/RingEqm_XXX_fref_Y_param.prm`



(a) Over the entire domain (b) At different values of y

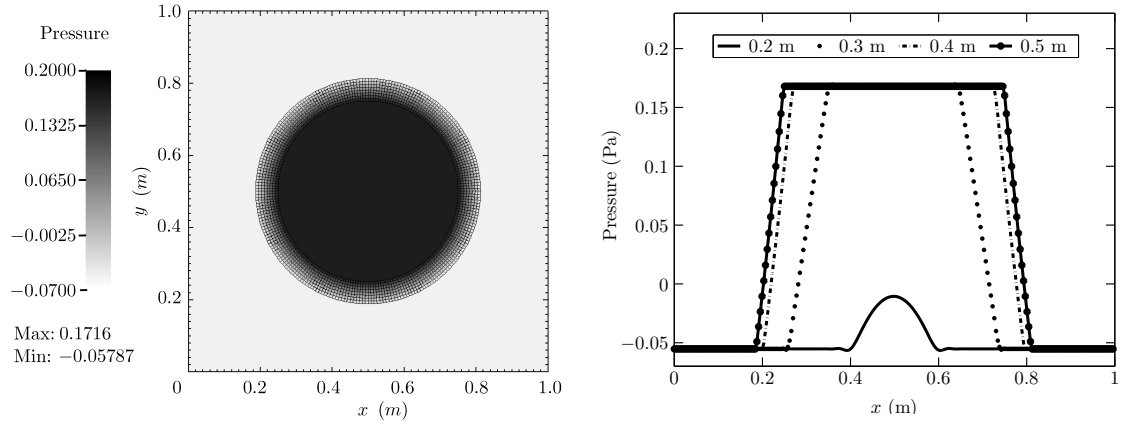
Figure 4: The values of p after one time step when using $P1$ elements for p

Table 2: Error convergence rate obtained when using $P1$ element for p after one time step

No. of cells	No. of DoFs	$\|\mathbf{u}_h - \mathbf{u}\|_0$		$\|\mathbf{u}_h - \mathbf{u}\|_1$		$\|p_h - p\|_0$	
256	2946	2.00605e-05	-	1.95854e-03	-	6.71603e-03	-
1024	11522	3.69389e-06	2.44	7.44696e-04	1.40	2.47476e-03	1.44
4096	45570	5.76710e-07	2.68	2.25134e-04	1.73	8.74728e-04	1.50
16384	181250	1.06127e-07	2.44	8.24609e-05	1.45	3.14028e-04	1.48

5.2 Disk entrained in a lid-driven cavity flow

We test the volume conservation of our numerical method by measuring the change in the area of a disk that is entrained in a lid-driven cavity flow of an incompressible, linearly viscous fluid. This test is motivated by similar ones presented in Wang and Zhang (2010); Griffith and Luo (2012). Referring to Fig. 6, the disk has a radius $R = 0.2\,\mathrm{m}$ and its center C is initially positioned at $x = 0.6\,\mathrm{m}$ and $y = 0.5\,\mathrm{m}$



(a) Over the entire domain (b) At different values of y

Figure 5: The values of p after one time step when using $Q1$ elements for p

Table 3: Error convergence rate obtained when using $Q1$ element for p after one time step

No. of cells	No. of DoFs	$\|\mathbf{u}_h - \mathbf{u}\|_0$		$\|\mathbf{u}_h - \mathbf{u}\|_1$		$\|p_h - p\|_0$	
256	2467	4.36912e-05	-	2.79237e-03	-	7.39310e-03	-
1024	9539	6.14959e-06	2.83	9.02397e-04	1.63	2.42394e-03	1.61
4096	37507	1.28224e-06	2.26	3.49329e-04	1.37	9.10608e-04	1.41
16384	148739	2.33819e-07	2.46	1.25626e-04	1.48	3.27256e-04	1.48

in the square cavity whose each edge has the length $l = 1.0\,\mathrm{m}$. Body forces on the system are negligible. The two different constitutive models for the elastic response of the disk which we have used for our simulations are as follows:

$$\text{case 1:} \quad \hat{\mathsf{P}} = -p_s \mathsf{I} + \mu^e \left(\mathsf{F} - \mathsf{F}^{-\mathrm{T}}\right), \tag{72}$$

$$\text{case 2:} \quad \hat{\mathsf{P}} = -p_s \mathsf{I} + \mu^e \mathsf{F}. \tag{73}$$

We have used the following parameters: $\rho = 1.0\,\mathrm{kg/m^3}$, dynamic viscosity $\mu = 0.01\,\mathrm{Pa{\cdot}s}$, shear modulus $\mu^e = 0.1\,\mathrm{Pa}$ and $U = 1.0\,\mathrm{m/s}$. For our numerical simulations we have used $Q2$ elements to represent $\boldsymbol{w}$ of the disk whereas we have used $Q2/P1$ element for the fluid. The disk is represented using 320 cells with 2626 DoFs and the control volume has 4096 cells and 45570 DoFs. The time step size $h = 1{\times}10^{-2}\,\mathrm{s}$. We consider the time interval $0 < t \leq 8\,\mathrm{s}$ during which the disk is lifted from its initial position along the left vertical wall, drawn along underneath the lid and finally dragged downwards along the right vertical wall of the cavity (see, Figs. 7 and 10). As the disk trails beneath the lid, it experiences large shearing deformations (see, Figs. 8 and 11). Ideally the disk should have retained its original area over the course of time because the incompressibility of the media and the nature of the motion require that the disk change its shape only and not its volume. However, from our numerical scheme we obtain an area change of the disk of about 6% for case 1 (see Fig. 9) and about 4% for case 2 (see Fig. 12).

The parameter files used for these two tests can be found under the directory
`deal.II/step-feibm/prms`, and are named `LDCFlow_Ball_DGP_INH0_param.prm` and
`LDCFlow_Ball_DGP_INH1_param.prm` respectively. The tests can be run under the
`deal.II/step-feibm` directory, by typing
`./step-feibm prms/LDCFlow_Ball_DGP_INH0_param.prm`
and
`./step-feibm prms/LDCFlow_Ball_DGP_INH1_param.prm`

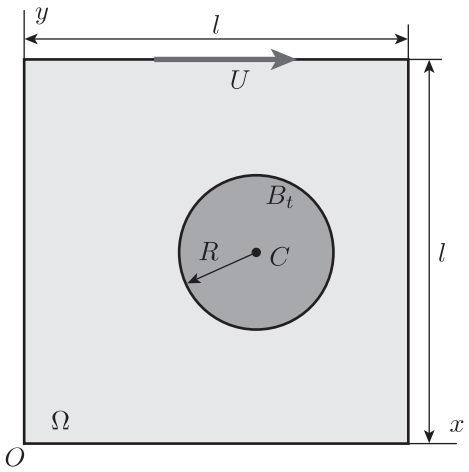


Figure 6: The initial configuration of an immersed disk entrained in a flow in a square cavity whose lid is driven with a velocity U towards the right

Acknowledgements

This work was partially supported by the "Young Scientist Grant" number ANFU.685, made available by the "Scuola Internazionale Superiore di Studi Avanzati" to the first author, and by the project OpenViewSHIP, "Sviluppo di un ecosistema computazionale per la progettazione idrodinamica del sistema elica-carena", supported by Regione FVG - PAR FSC 2007-2013, Fondo per lo Sviluppo e la Coesione.

References

Bangerth, W., R. Hartmann, and G. Kanschat (1998–2006) *deal.II Differential Equations Analysis Library—Technical Reference*.
URL `http://www.dealii.org`

——— (2007) "deal.II — a General Purpose Object Oriented Finite Element Library," *ACM Transactions on Mathematical Software (TOMS)*, **33**(4), pp. 24:1–24:27.

Bangerth, W., T. Heister, L. Heltai, G. Kanschat, M. Kronbichler, M. Maier, B. Turcksin, and T. D. Young (2013) "The `deal.II` Library, Version 8.1," *arXiv preprint* `http://arxiv.org/abs/1312.2266v4`.

Boffi, D. and L. Gastaldi (2003) "A Finite Element Approach for the Immersed Boundary Method," *Computers & Structures*, **81**(8–11), pp. 491–501.

Boffi, D., L. Gastaldi, and L. Heltai (2007) "On the CFL Condition for the Finite Element Immersed Boundary Method," *Computers & Structures*, **85**(11–14), pp. 775–783.

Boffi, D., L. Gastaldi, L. Heltai, and C. S. Peskin (2008) "On the Hyper-Elastic Formulation of the Immersed Boundary Method," *Computer Methods in Applied Mathematica and Mechanics*, **197**(25–28), pp. 2210–2231.

Brezzi, F. and M. Fortin (1991) *Mixed and Hybrid Finite Element Methods*, vol. 15 of *Springer Series in Computational Mathematics*, Springer-Verlag, New York.

Childs, H., E. S. Brugger, K. S. Bonnell, J. S. Meredith, M. Miller, B. J. Whitlock, and N. Max (2005) "A Contract-Based System for Large Data Visualization," in *Proceedings of IEEE Visualization 2005*, pp. 190–198.

Davis, T. A. (2004) "Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method," *ACM Trans. Math. Softw.*, **30**(2), pp. 196–199.

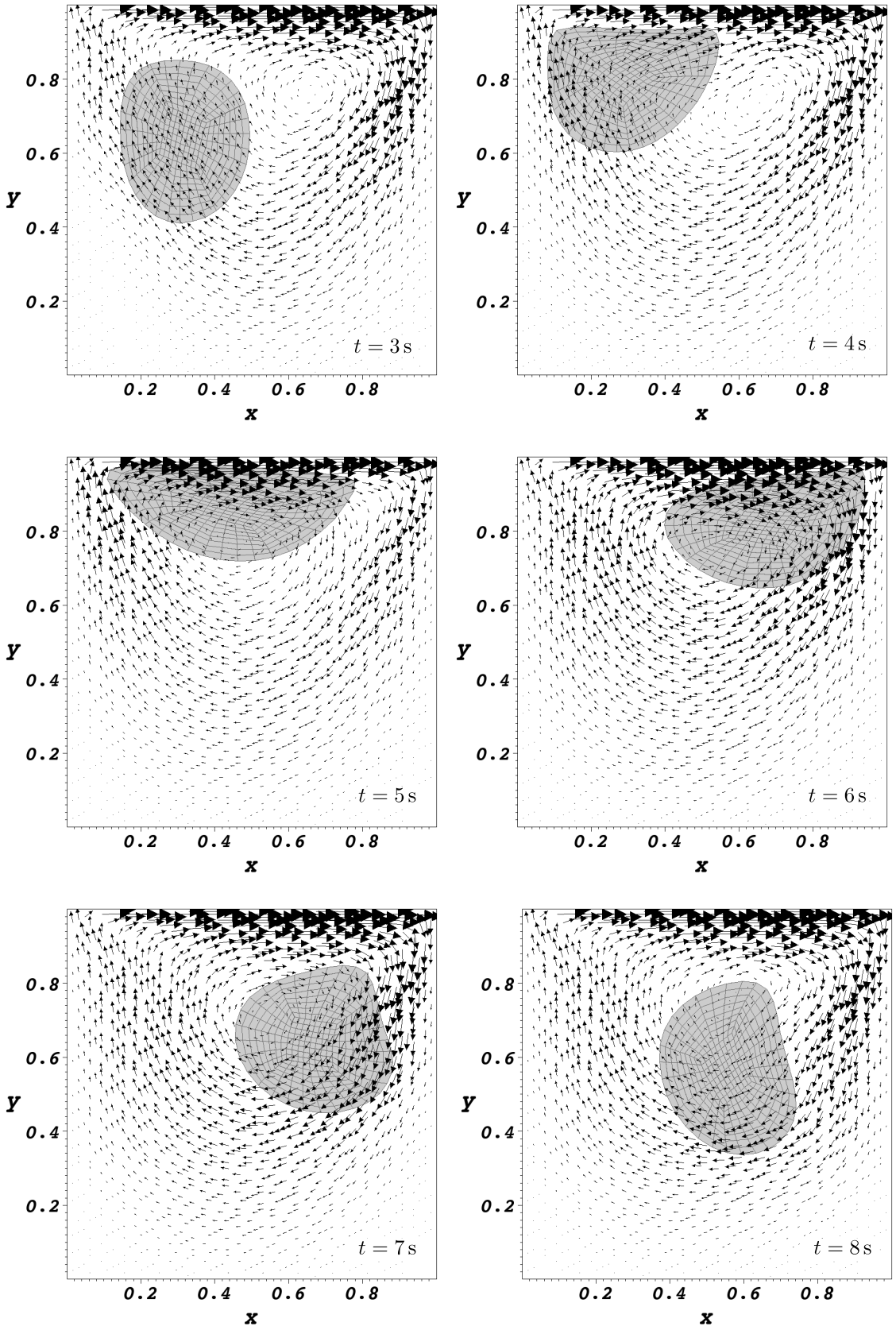


Figure 7: The motion of the disk for case 1 at different instants of time

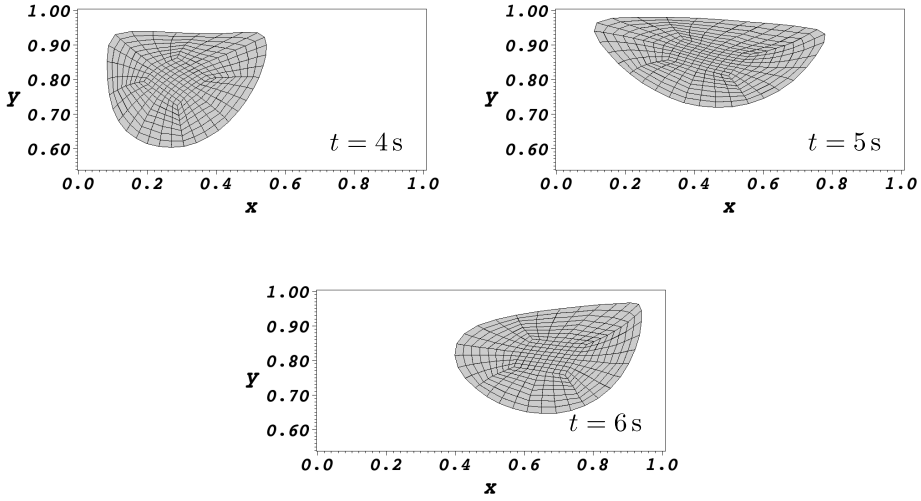


Figure 8: Enlarged view of the disk for case 1 depicting its shape and location at various instants of time

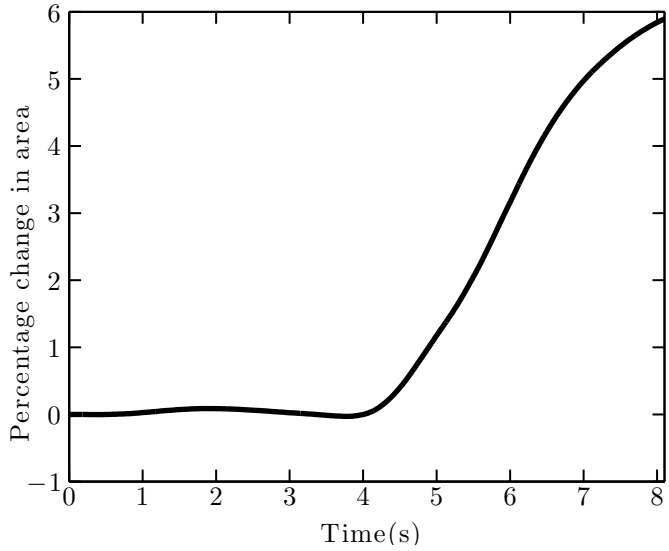


Figure 9: The percentage change in the area of the disk for case 1 over time

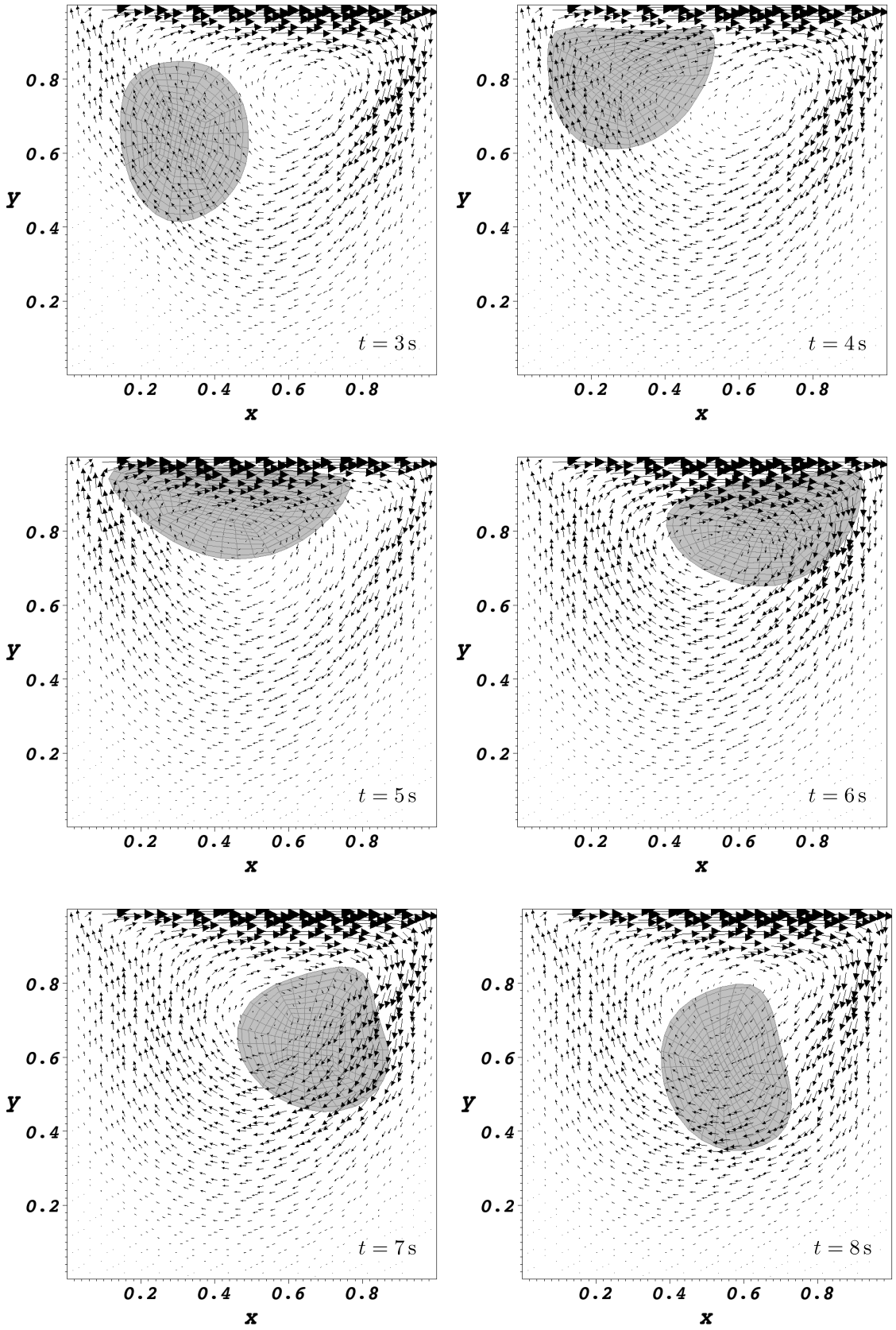


Figure 10: The motion of a disk for case 2 at different instants of time

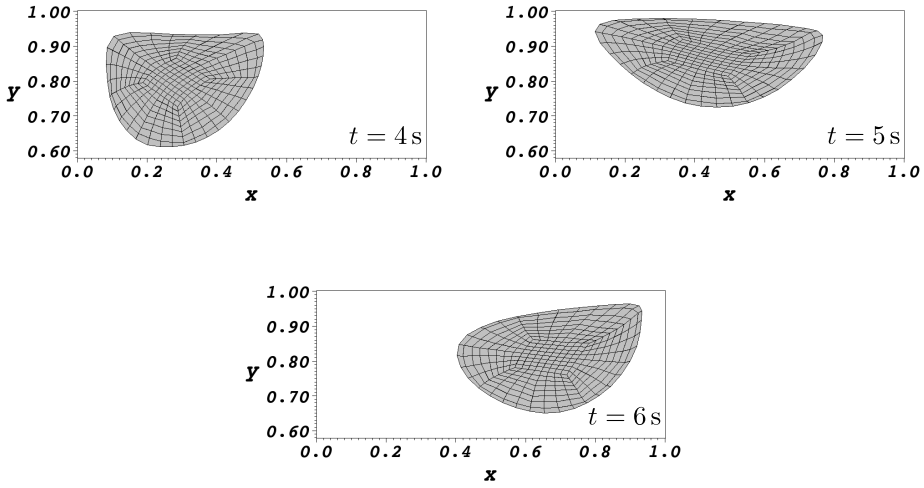


Figure 11: Enlarged view of the disk for case 2 depicting its shape and location at various instants of time

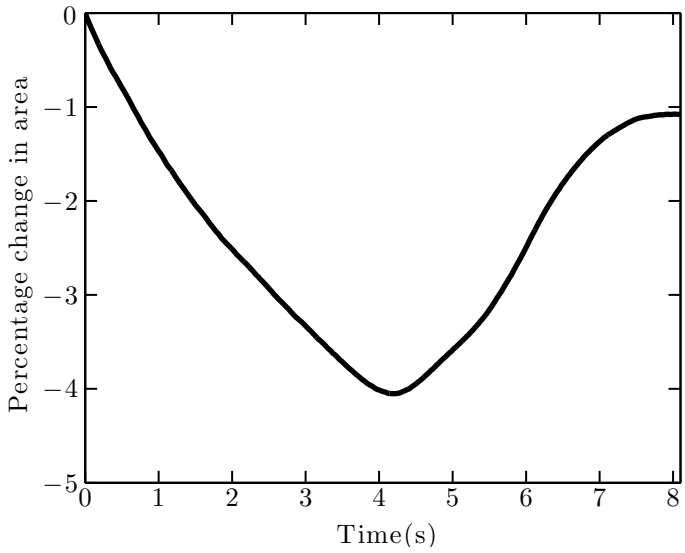


Figure 12: The percentage change in the area of the disk for case 2 over time

Griffith, B. E. (2012) "On the volume conservation of the immersed boundary method," *Communications in Computational Physics*.

Griffith, B. E. and X. Luo (2012) "Hybrid finite difference/finite element version of the immersed boundary method," Submitted in revised form.

Gurtin, M. E., E. Fried, and L. Anand (2010) *The Mechanics and Thermodynamics of Continua*, Cambridge University Press, New York.

Heltai, L. (2006) *The Finite Element Immersed Boundary Method*, Ph.D. thesis, Università di Pavia.

——— (2008) "On the Stability of the Finite Element Immersed Boundary Method," *Computers & Structures*, **86**(7–8), pp. 598–617.

Heltai, L. and F. Costanzo (2012) "Variational Implementation of Immersed Finite Element Methods," *Computer Methods in Applied Mechanics and Engineering*, **229–232**(0), pp. 110–127, accepted for publication.

Hughes, T. J. R., W. K. Liu, and Z. T. K. (1981) "Lagrangian-Eulerian Finite Element Formulations for Incompressible Viscous Flows," *Computer Methods in Applied Mechanics and Engineering*, **29**, pp. 329–349.

Liu, W. K., D. W. Kim, and S. Tang (2007) "Mathematical Foundations of the Immersed Finite Element Method," *Computational Mechanics*, **39**(3), pp. 211–222.

Peskin, C. S. (1977) "Numerical Analysis of Blood Flow in the Heart," *Journal of Computational Physics*, **25**(3), pp. 220–252.

——— (2002) "The Immersed Boundary Method," *Acta Numerica*, **11**, pp. 479–517.

Roy, S., L. Heltai, and F. Costanzo (2013) *Benchmarking the Immersed Finite Element Method for Fluid-Structure Interaction Problems*, *Tech. rep.*, SISSA.

Wang, X. and W. K. Liu (2004) "Extended Immersed Boundary Method using FEM and RKPM," *Computer Methods in Applied Mechanics and Engineering*, **193**(12–14), pp. 1305–1321.

Wang, X. and L. Zhang (2010) "Interpolation Functions in the Immersed Boundary and Finite Element Methods," *Computational Mechanics*, **45**, pp. 321–334.

Zhang, L., A. Gerstenberger, X. Wang, and W. K. Liu (2004) "Immersed Finite Element Method," *Computer Methods in Applied Mechanics and Engineering*, **193**(21–22), pp. 2051–2067.

