Engineering with Computers (2023) 39:1019–1040
https://doi.org/10.1007/s00366-022-01754-y

ORIGINAL ARTICLE

Leveraging code generation for transparent immersogeometric fluid–structure interaction analysis on deforming domains

Grant E. Neighbor[1] · Han Zhao[2] · Mehdi Saraeian[1] · Ming-Chen Hsu[1] · David Kamensky[2]

Received: 13 August 2022 / Accepted: 17 October 2022 / Published online: 16 November 2022

Abstract
Code generation technology has been transformative to the field of numerical partial differential equations (PDEs), allowing domain scientists and engineers to automatically compile high-performance solver routines from abstract mathematical descriptions of PDE systems. However, this often assumes a rigid code structure, which is only appropriate to a subset of applications and numerical methods, such as the traditional finite element methods used by the FEniCS code generation system. The present contribution demonstrates how to productively integrate FEniCS into a custom implementation of immersogeometric analysis (IMGA) of thin shell structures interacting with incompressible fluid flows on deforming domains. IMGA is an emerging paradigm for numerical PDEs with complex domain geometries, where non-watertight geometry descriptions are used directly as computational meshes. In particular, we generalize past related work by leveraging code generation to concisely pull back the deforming-domain Navier–Stokes problem to a stationary reference mesh. We also show how code generation enables rapid implementation of different material models for the structure subproblem. We verify our implementation using several benchmark problems, demonstrate its robustness and flexibility by simulating a prosthetic heart valve immersed in a flexible artery, and distribute the full source code online, to be used and modified by the community. Impact of the last item is amplified by the transparent nature of our code-generation-based implementation.

Keywords Isogeometric analysis · Fluid–structure interaction · Immersed boundary · Open-source software · FEniCS · Heart valve

1 Introduction

Traditionally, software to compute numerical approximations to solutions of partial differential equation (PDE) systems has been developed separately for different PDE systems. Developing a new PDE solver is often a daunting task, as many applications demand that software be optimized carefully for parallel high-performance computing (HPC) systems. However, over the past several decades, researchers from applied math have demonstrated that effective numerical methods for a wide range of PDE systems can be represented as variational problems posed over finite-dimensional spaces of solution functions, called finite element (FE) methods [1]. This shared structure suggests that a common set of software abstractions can be used to implement different numerical schemes for different PDEs, spanning a broad range of application domains. In particular, the translation of variational forms into computational kernels for FE assembly is a systematic enough process that optimized code for these kernels can be generated automatically from the mathematical definition of a variational form. A leading example of code generation for numerical PDEs is the FEniCS Project [2]. In its canonical workflow, users write the variational forms of their PDEs in the Python-based Unified Form Language (UFL) [3], and these UFL problem statements are then automatically compiled into efficient C++ kernels [4] that plug into a general high-performance FE solver called DOLFIN [5].

FEniCS has proven highly popular with scientists, mathematicians, and engineers across a wide range of disciplines. Providing a comprehensive picture of its applications to date would require a dedicated review article, but a cursory

✉ David Kamensky
dmkamensky@eng.ucsd.edu

[1] Department of Mechanical Engineering, Iowa State University, 2043 Black Engineering, Ames, IA 50011, USA

[2] Department of Mechanical and Aerospace Engineering, University of California San Diego, 9500 Gilman Drive, Mail Code 0411, La Jolla, CA 92093, USA

glance at citation data reveals FEniCS-based work ranging from basic research in numerical methods [6, 7] and exotic physical phenomena [8, 9] to industrial-scale engineering analysis [10–13]. The concise and readable nature of UFL problem statements also encourages using code as a form of scientific communication. While voices within the computational math and science communities have increasingly called for reproducibility of numerical results [14, 15], code generation enables the stronger condition of *transparency*, where code shared by researchers is essentially as easy to understand (and thus also to modify and extend) as the notation from published articles. This attribute of code generation has also made it popular in pedagogical works [16, 17].

Despite the promise code generation shows for streamlining the process of conducting and disseminating research, challenges remain. In particular, systems like FEniCS assume that the target problem consists of a single PDE system defined on a given domain, over which a good quality FE mesh can be generated. In engineering problems, it is often the case that multiple coupled subproblems are posed on different subdomains, whose *upstream* geometrical descriptions are separately parameterized and often even mathematically ill-defined, such as computer-aided design (CAD) models specifying volumes through non-watertight descriptions of boundary surfaces. Geometric complexity can also occur *downstream* of engineering analysis, as an emergent feature of a PDE system's solution. Examples of this include large deformations of soft materials and topological changes due to fracture and/or contact mechanics. Numerical schemes to couple non-matching parameterizations of geometry or contend with topological changes—e.g., immersed boundary [18, 19] or meshfree [20] methods—cannot be reduced to the self-contained element-level kernels assumed by FEniCS's code generation architecture. An active area of research is then: How can code generation capabilities be gainfully applied in conjunction with such methods?

To motivate the present contribution, we introduce a problem which exhibits both upstream and downstream geometric complexity: fluid–structure interaction (FSI) analysis of prosthetic heart valves. In summary, the most popular class of prosthetic valves mimic the structure of native valves, in which thin flexible leaflets are passively opened and closed by periodic blood flow. The large deformation and especially topology change (during valve closure) of the fluid subproblem's domain are obvious examples of downstream geometric complexity. However, applications like optimizing prosthetic valve leaflets and patient-specific modeling also benefit from seamless integration of analysis with upstream CAD geometry and medical imaging data. Kamensky, Hsu, and various other collaborators have recently published a series of articles on heart valve FSI analysis [21–25].[1] In particular, these authors directly use a CAD model of the valve as an analysis mesh, which is an example of isogeometric analysis (IGA) [29, 30], and couple it to an unfitted mesh of the artery lumen, which is an instance of an immersed method.[2] They refer to the synergy between IGA and immersed methods as immersogeometric analysis (IMGA).

The cited works implemented these methods in custom research codes, without regard for transparency. The recent paper [32] implemented a subset of this immersogeometric FSI functionality as the open-source library CouDALFISh [33], which used FEniCS for classical FE analysis of the fluid and the FEniCS-based IGA library tIGAr [34] for IGA of the valve leaflets. However, the functionality of CouDALFISh documented in [32] was quite limited. In particular, there was no option to use boundary-fitted arbitrary Lagrangian–Eulerian (ALE) [35] FSI where feasible, such as for modeling the mild deformations of an elastic artery's lumen. Reference [22] demonstrated that artery deformation is crucial to accurate analysis, but the heart valve example of [32] simply treated the artery as rigid. The present contribution builds upon [32], introducing the following novelties:

- Formulating ALE FSI on a static reference domain, leveraging abstractions from FEniCS UFL to maintain a clear interpretation (whereas previous work in [32] was limited to an Eulerian description of the fluid, and did not include a solid subproblem).
- Interfacing this formulation with the immersogeometric fluid–thin structure interaction method of [21], which entails some changes to the algorithm from [32].
- Demonstrating the potential for rapid prototyping of soft tissue constitutive models, expressed in symbolic form using FEniCS UFL, which was technically possible with only the technology from [32], but not implemented or tested until the present work.

In the remainder of this paper, we first introduce the mathematical problem under consideration (Sect. 2), then recall the particular immersed technique that we use for the fluid–thin structure interaction analysis (Sect. 3). Section 4 then discusses the implementation of the above-itemized contributions, and how it leverages code generation technology. Numerical results for several problems are then

[1] We shall not attempt to review all recent work on heart valve FSI here, as this (substantial) task is already accomplished by dedicated review articles [26–28]; our introduction is limited to direct ancestors of the present contribution.

[2] We interpret the adjective "immersed" rather liberally here, using it to refer to any numerical method in which a computational mesh is not fitted to a boundary or interface; this is at odds with some references, which reserve it for a specific method introduced by [31].

presented, including various benchmarks (Sect. 5) and a simulation of a prosthetic heart valve in a flexible artery (Sect. 6). Finally, Sect. draws conclusions and discusses potential future developments.

2 Combined fluid–solid and fluid–shell interaction

This paper considers the problem of a thin shell structure (modeled geometrically as a surface of co-dimension one) immersed in a "background" 3D fluid–solid continuum. The two subproblems are constrained to have the same velocity on the topologically 2D midsurface of the shell structure, with the 3D continuum's traction jump across the surface equal to a normal force acting on the shell. Mathematically, we can write this problem as: Find shell structure midsurface velocity $\mathbf{v}_{\text{sh}} \in \mathcal{S}_{\text{sh}}$, background fluid–solid velocity $\mathbf{v}_{\text{fs}} \in \mathcal{S}_{\text{fs}}$, fluid pressure $p \in \mathcal{S}_p$, and interface Lagrange multiplier $\lambda \in \mathcal{S}_\lambda$ such that for all test functions $(\mathbf{w}_{\text{sh}}, \mathbf{w}_{\text{fs}}, q, \delta\lambda) \in \mathcal{V}_{\text{sh}} \times \mathcal{V}_{\text{fs}} \times \mathcal{V}_p \times \mathcal{V}_\lambda$,

$$\begin{aligned} &R_{\text{sh}}(\mathbf{v}_{\text{sh}}, \mathbf{w}_{\text{sh}}) + R_{\text{fs}}(\{\mathbf{v}_{\text{fs}}, p\}, \{\mathbf{w}_{\text{fs}}, q\}) \\ &\quad + C_\lambda(\mathbf{v}_{\text{sh}}, \mathbf{v}_{\text{fs}}, \delta\lambda) + C_\lambda(\mathbf{w}_{\text{sh}}, \mathbf{w}_{\text{fs}}, \lambda) = 0\,, \end{aligned} \tag{1}$$

where $\mathcal{S}_{(\cdot)}$ are spaces of trial functions satisfying Dirichlet boundary conditions (if any) and $\mathcal{V}_{(\cdot)}$ are corresponding test function spaces. The variational forms R_{sh} and R_{fs} define the shell structure and fluid–solid continuum subproblems, while C_λ defines the FSI kinematic constraint. We now elaborate on these definitions in the following sections.

2.1 Fluid–solid continuum

Following the kinematic framework established by [36], we refer momentum balance for the fluid–solid continuum to a static reference domain Ω_y. This domain is mapped to the physical domain Ω_x by a motion $\hat{\boldsymbol{\phi}}$, which can be defined by a displacement field, $\hat{\mathbf{u}}$, such that the spatial point $\mathbf{x}$ corresponding to a reference point $\mathbf{y}$ is given by

$$\mathbf{x} = \hat{\boldsymbol{\phi}}(\mathbf{y}) = \mathbf{y} + \hat{\mathbf{u}}(\mathbf{y})\,. \tag{2}$$

The mapping $\hat{\boldsymbol{\phi}}$ and displacement $\hat{\mathbf{u}}$ depend, in general, on time, but this dependence has been suppressed above for notational simplicity. When differentiating with respect to time, we must be precise about what spatial coordinate is held fixed. In this paper, we shall use

$$\left.\frac{\partial f}{\partial t}\right|_{\mathbf{y}} \quad \text{and} \quad \left.\frac{\partial f}{\partial t}\right|_{\mathbf{x}}, \tag{3}$$

to denote partial time derivatives of some field f at a fixed points in the reference domain and spatial domains, respectively. In particular, the velocity of the mapping from the reference to physical domain is

$$\hat{\mathbf{v}} = \left.\frac{\partial \hat{\mathbf{u}}}{\partial t}\right|_{\mathbf{y}}\,. \tag{4}$$

We must similarly disambiguate spatial derivatives. Spatial derivatives in Ω_x are denoted using ∇_x and spatial derivatives in Ω_y are denoted using ∇_y. We divide the reference domain Ω_y into two subsets Ω_y^{so} and Ω_y^{f} such that

$$\overline{\Omega_y} = \overline{\Omega_y^{\text{so}}} \cup \overline{\Omega_y^{\text{f}}}\,. \tag{5}$$

The images Ω_x^{so} and Ω_x^{f} of these subsets under the mapping $\hat{\boldsymbol{\phi}}$ are occupied by solid and fluid material respectively in physical space. In the solid subdomain, we identify each reference point $\mathbf{y} \in \Omega_y^{\text{so}}$ with a material point $\mathbf{X}$, for consistency with prevailing conventions from the literature on nonlinear solid mechanics. Following the convention established by (3), the partial time derivative at a fixed material point is denoted $\partial/\partial t|_{\mathbf{X}}$ (corresponding to the notation "D/Dt" in many references), and spatial derivatives in Ω_y^{so} are denoted with ∇_X. The fluid–solid continuum residual R_{fs} can be split into two forms integrated over these subdomains:

$$R_{\text{fs}}(\{\mathbf{v}, p\}, \{\mathbf{w}, q\}) = R_{\text{f}}(\{\mathbf{v}, p\}, \{\mathbf{w}, q\}) + R_{\text{so}}(\mathbf{v}, \mathbf{w})\,. \tag{6}$$

We now define these forms for an incompressible Newtonian fluid (Sect. 2.1.1) and a hyperelastic solid (Sect. 2.1.2).

2.1.1 Incompressible Newtonian fluid

The weak formulation of the incompressible Navier–Stokes equations on a deforming domain is frequently given as

$$\begin{aligned} R_{\text{f}}(\{\mathbf{v}, p\}, \{\mathbf{w}, q\}) = &\int_{\Omega_x^{\text{f}}} \rho_{\text{f}} \left(\left.\frac{\partial \mathbf{v}}{\partial t}\right|_{\mathbf{y}} + (\mathbf{v} - \hat{\mathbf{v}}) \cdot \nabla_x \mathbf{v} \right) \cdot \mathbf{w} - \mathbf{f} \cdot \mathbf{w}\, d\Omega_x \\ &+ \int_{\Omega_x^{\text{f}}} \boldsymbol{\sigma} : \nabla_x \mathbf{w} + q \nabla_x \cdot \mathbf{v}\, d\Omega_x \\ &+ \int_{\partial\Omega_x \cap \partial\Omega_x^{\text{f}}} \left(\mathbf{h} - \gamma \rho_{\text{f}} \{(\mathbf{v} - \hat{\mathbf{v}}) \cdot \mathbf{n}_x\}_- \mathbf{v} \right) \cdot \mathbf{w}\, d\partial\Omega_x\,, \end{aligned} \tag{7}$$

where ρ_{f} is the fluid mass density,

$$\boldsymbol{\sigma} = 2\mu_{\text{f}} \text{sym} \nabla_x \mathbf{v} - p\mathbf{I} \tag{8}$$

is the Cauchy stress for dynamic viscosity μ_{f}, $\mathbf{I}$ is the identity tensor, ∇_x differentiates with respect to spatial position, $\mathbf{f}$ is a body force per unit volume, $\mathbf{h}$ is an applied momentum flux on $\partial\Omega_x \cap \partial\Omega_x^{\text{f}}$, $\mathbf{n}_x$ is the outward-facing unit normal to $\partial\Omega_x$, $\{\cdot\}_-$ takes the negative part of its argument, i.e.,

$$\{f\}_- = \frac{1}{2}(f - |f|)\,, \tag{9}$$

and $0 \le \gamma \le 1$ is a dimensionless scalar parameter.

Remark 1 The term involving $\{\cdot\}_-$ stabilizes the problem when flow enters the domain through a Neumann boundary part. When no flow is entering the boundary, $\mathbf{h}$ has the interpretation of a traction vector, whereas, when flow is entering the boundary, it becomes a combination of traction and advective flux. The parameter γ controls the strength of this stabilizing effect. See [37] for details. Note that this term is included in the continuous problem, to ensure that it is well-posed.

The formulation (7) contains the awkward combination of a temporal partial derivative taken on the static reference domain and spatial partial derivatives taken on the deforming spatial domain. Our discretization will be based on referring the entire formulation back to the reference domain, which may be accomplished by the following substitutions:

$$\int_{\Omega_x^{\mathrm{f}}} (\cdots)\, d\Omega_x \;\rightarrow\; \int_{\Omega_y^{\mathrm{f}}} (\cdots)\, \hat{J} d\Omega_y \,, \tag{10}$$

$$\int_{\partial\Omega_x \cap \partial\Omega_x^{\mathrm{f}}} (\cdots)\, d\partial\Omega_x \;\rightarrow\; \int_{\partial\Omega_y \cap \partial\Omega_y^{\mathrm{f}}} (\cdots) \left|\hat{J}\hat{\mathbf{F}}^{-T} \cdot \mathbf{n}_y\right| d\partial\Omega_y \,, \tag{11}$$

$$\nabla_x(\cdots) \;\rightarrow\; \nabla_y(\cdots) \cdot \hat{\mathbf{F}}^{-1} \,, \tag{12}$$

$$\mathbf{n}_x \;\rightarrow\; \frac{\hat{J}\hat{\mathbf{F}}^{-T} \cdot \mathbf{n}_y}{\left|\hat{J}\hat{\mathbf{F}}^{-T} \cdot \mathbf{n}_y\right|} \,, \tag{13}$$

where

$$\hat{\mathbf{F}} = \frac{\partial \hat{\boldsymbol{\phi}}}{\partial \mathbf{y}}, \tag{14}$$

is the deformation gradient of the motion $\hat{\boldsymbol{\phi}}$,

$$\hat{J} = \det\hat{\mathbf{F}}, \tag{15}$$

is its determinant, and $\mathbf{n}_y$ is the outward-facing unit normal to Ω_y.

When discretizing the fluid in space, i.e., posing the fluid subproblem on finite-dimensional subspaces of $\mathcal{S}_{\mathrm{fs}}$ and $\mathcal{V}_{\mathrm{fs}}$, we augment R_{f} with additional terms, following the variational multiscale (VMS) theory utilized by Bazilevs and collaborators in [36, 38–45]. This maintains the stability of the formulation even when the exact solution contains features too small to resolve with a given spatial discretization. A common source of such solution features is flow turbulence, and VMS can correspondingly be understood as an implicit large-eddy simulation filter, as discussed in [38], with favorable comparisons to direct numerical simulation.

2.1.2 Hyperelastic solid

Within the solid subdomain, we assume that the background continuum velocity field can be expressed as the time derivative of a displacement field, viz.,

$$\mathbf{v}_{\mathrm{fs}}\big|_{\Omega_y^{\mathrm{so}}} = \left.\frac{\partial \mathbf{u}_{\mathrm{so}}}{\partial t}\right|_{\mathbf{X}}, \tag{16}$$

where $\mathbf{u}_{\mathrm{so}}$ is the solid material's displacement field. We further assume that

$$\hat{\mathbf{u}} = \mathbf{u}_{\mathrm{so}}, \tag{17}$$

in Ω_y^{so}, i.e., that the reference domain deforms with the material. An obvious implication is that $\hat{\mathbf{v}} = \mathbf{v}_{\mathrm{fs}}$ within the solid subdomain. Accordingly, we also drop the $\hat{(\cdot)}$ from the deformation gradient (14) and its determinant (15), which now correspond to the material motion:

$$\mathbf{F} = \mathbf{I} + \nabla_X \mathbf{u}_{\mathrm{so}} \quad \text{and} \quad J = \det\mathbf{F} \,. \tag{18}$$

With these assumptions, the solid residual form R_{so} can be written as

$$\begin{aligned} R_{\mathrm{so}}(\mathbf{v}, \mathbf{w}) = &\int_{\Omega_y^{\mathrm{so}}} \rho_0^{\mathrm{so}} \left.\frac{\partial \mathbf{v}}{\partial t}\right|_{\mathbf{X}} \cdot \mathbf{w} - \mathbf{f}_0 \cdot \mathbf{w} \\ &+ (\mathbf{FS}){:}\nabla_X \mathbf{w} + C_{\mathrm{d}} \rho_0^{\mathrm{so}} \mathbf{v} \cdot \mathbf{w}\, d\Omega_y \\ &- \int_{\partial\Omega_y^{\mathrm{so}} \cap \partial\Omega_y} \mathbf{h}_0 \cdot \mathbf{w}\, d\partial\Omega_y \,, \end{aligned} \tag{19}$$

where $\mathbf{h}_0$ is an applied traction per unit reference boundary area, $\mathbf{f}_0$ is a body force per unit reference volume, ρ_0^{so} is mass density per unit reference volume, $\mathbf{S}$ is the 2nd Piola–Kirchhoff stress, and C_{d} is a mass damping coefficient. The defining feature of a hyperelastic solid is that $\mathbf{S}$ can be obtained by differentiating a scalar energy density:

$$\mathbf{S} = \frac{\partial \psi}{\partial \mathbf{E}} \,, \tag{20}$$

where ψ is an elastic energy density per unit reference volume and

$$\mathbf{E} = \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right), \tag{21}$$

is the Green–Lagrange strain tensor. These definitions imply that

$$(\mathbf{FS}) : \nabla_X \mathbf{w} = \mathbf{S} : D_{\mathbf{w}}\mathbf{E} = D_{\mathbf{w}}\psi \,, \tag{22}$$

where $D_{\mathbf{w}}$ is a Gateaux derivative with respect to displacement in the direction $\mathbf{w}$.

In this work, we consider only moderately compressible solids.[3] In that case, an appropriate discretization is the Bubnov–Galerkin method, i.e., simply restricting the form R_{so} to finite-dimensional subsets of $\mathcal{S}_{\text{fs}}$ and $\mathcal{V}_{\text{fs}}$.

2.2 Kirchhoff–Love shell structure

The thin immersed structure is a simplification of the hyperelastic solid, where the reference domain is instead a thin region of thickness h_{th}, extending symmetrically in the normal direction from a 2D midsurface manifold Γ_0. As in the case of the solid, we assume that the shell structure velocity is derived from a displacement field,

$$\mathbf{v}_{\text{sh}} = \frac{\partial \mathbf{u}_{\text{sh}}}{\partial t} , \tag{23}$$

where the partial time derivative is taken at a fixed point in Γ_0. If the through-thickness direction is parameterized by an arc-length coordinate $\xi^3 \in (-h_{\text{th}}/2, h_{\text{th}}/2)$, then the residual form can be written as

$$R_{\text{sh}}(\mathbf{v}, \mathbf{w}) = h_{\text{th}} \int_{\Gamma_0} \rho_0^{\text{sh}} \frac{\partial \mathbf{v}}{\partial t} \, d\Gamma_0 + \int_{-h_{\text{th}}/2}^{h_{\text{th}}/2} \int_{\Gamma_0} \mathbf{S} : D_{\mathbf{w}} \mathbf{E} \, d\Gamma_0 d\xi^3 , \tag{24}$$

where ρ_0^{sh} is the shell structure's mass density per unit reference volume, and $\mathbf{S}$ and $\mathbf{E}$ are again the 2nd Piola–Kirchhoff stress and Green–Lagrange strain. However, in the Kirchhoff–Love thin shell theory, kinematic assumptions are invoked to simplify the definition of $\mathbf{E}$ so that it depends entirely on derivatives of the midsurface displacement, and $\mathbf{S}$ is assumed to satisfy a plane stress condition in the tangent plane to Γ_0. For the case of hyperelastic material models, the complete shell structure formulation used in this work can be found in [46].

A notable feature of the Kirchhoff–Love shell model is that the approximate expression for $\mathbf{E}$ involves second derivatives of the midsurface displacement, as does $\mathbf{S}$ (through its dependence on $\mathbf{E}$). This means that the displacement function spaces $\mathcal{S}_{\text{sh}}$ and $\mathcal{V}_{\text{sh}}$ must be at least contained in the Sobolev space $H^2(\Gamma_0)$ for R_{sh} to be well defined. In practical terms, this means that, if one wishes to use a conforming Bubnov–Galerkin discretization with piecewise polynomial displacements, the discrete displacements must be at least C^1 between elements. As mentioned in [29, 47], we accomplish this in the present work using IGA with smooth B-spline function spaces for the midsurface displacement.

2.3 FSI kinematic constraint

The fluid–thin-structure constraint form C_λ in (1) is defined generally as

$$C_\lambda(\mathbf{a}_1, \mathbf{a}_2, \mathbf{b}) = \int_{\Gamma_0} (\mathbf{a}_1 - \mathbf{a}_2) \cdot \mathbf{b} \, d\Gamma_0 . \tag{25}$$

This corresponds to enforcement of the constraint $\mathbf{v}_{\text{fs}} = \mathbf{v}_{\text{sh}}$ on the shell midsurface, enforced by the Lagrange multiplier $\boldsymbol{\lambda}$, in (1).

Note that the integral over the *reference* shell midsurface Γ_0 differs from many formulations in the literature, implying that $\boldsymbol{\lambda}$ is a traction per unit reference midsurface area. This does not affect the velocity solutions to the continuous problem, but it simplifies the implementation of some numerical schemes. In particular, if the integral is discretized using a collection of quadrature points on Γ_0, their corresponding weights will not depend on spatial derivatives of the shell midsurface displacement $\mathbf{u}_{\text{sh}}$.

2.4 Shell structure self-contact

In principle, the FSI kinematic constraint should prevent initially separated parts of an immersed shell structure from interpenetrating (since they are constrained to follow the motion of a continuous velocity field).[4] However, when FSI kinematics are enforced approximately in the discrete setting, it is usually necessary to augment the shell structure formulation with some treatment of contact mechanics. In this work, we use a nonlocal regularization of contact, fitting into the general framework of volumetric potentials introduced by [49]. In summary, the residual (24) of the shell structure subproblem is augmented with a term of the form

$$+D_{\mathbf{w}} E_{\text{c}} , \tag{26}$$

where[5]

$$E_{\text{c}} = \frac{1}{2} \int_{\Gamma_0 \setminus B_{R_{\text{self}}}(\mathbf{X}_1)} \int_{\Gamma_0} \phi\big(r_{1\to 2}\big) \, d\mathbf{X}_1 d\mathbf{X}_2, \tag{27}$$

[3] This is in contrast with the shell structure subproblem, where we consider incompressible material in some problems, to model biological soft tissue in heart valve leaflets. One might (correctly) point out that a high-fidelity model of arterial soft tissue should also be incompressible, but the scope of our solid artery modeling is limited to capturing its effect on valve dynamics; this does not depend critically on details of the artery's constitutive model.

[4] This mathematical result of course contradicts everyday experience; the principled solution is to consider the interaction of flow with surface roughness, as detailed in [48], but we proceed with a practical *ad hoc* approach in the present work.

[5] A factor of h_{th}^2 has been absorbed into the potential density ϕ, to account for the difference between area integrals here and volume integrals in [49, (17)].

is a potential energy associated with contact forces, in which $B_{R_{\text{self}}}(\mathbf{X}_1)$ is a ball of radius $R_{\text{self}} > 0$ about $\mathbf{X}_1$ (to exclude "contact" between points that are within R_{self} of each other in the reference configuration),

$$r_{1\to 2} = \left\|(\mathbf{X}_2 + \mathbf{u}_{\text{sh}}(\mathbf{X}_2)) - (\mathbf{X}_1 + \mathbf{u}_{\text{sh}}(\mathbf{X}_1))\right\|_{\ell^2}, \tag{28}$$

is the distance between the deformed images of material points $\mathbf{X}_1, \mathbf{X}_2 \in \Gamma_0$, and the potential density ϕ governs the repulsion between material points. In this work, ϕ is selected such that its derivative (i.e., the magnitude of repulsive force density) is

$$\phi'(r) = -k_{\text{c}} \begin{cases} 0 & , \quad r \geq r_{\max}, \\ (r_{\max} - r)^2/(2 s_{\text{c}} r_{\max}) & , \quad (1-s_{\text{c}}) r_{\max} < r < r_{\max}, \\ r_{\max}(1 - s_{\text{c}}/2) - r & , \quad r \leq (1-s_{\text{c}}) r_{\max}, \end{cases} \tag{29}$$

where k_{c} modulates the stiffness of contact forces, $0 < r_{\max} < R_{\text{self}}$ is the maximum range of nonlocal contact, and $0 \leq s_{\text{c}} \leq 1$ is the fraction of $r_{\max}$ over which activation of forces is smoothed. While it does not preclude penetration of the shell surface through itself at finite k_{c}, it provides more reliable nonlinear convergence with a fixed time step than the singular potential given in [49], which required a specialized time integrator and line search algorithm.

When the integrals of (27) are discretized using a finite set of quadrature points, the method becomes computationally similar to the "pinball" method of [50], as discussed in [49, Section 3.2]. In this work, we use as quadrature points the nodes of the finite element function space to which tIGAr extracts the isogeometric midsurface displacement space. An extension of the model given by (27) to frictional contact can be found in [51], while [52] reviews nonlocal regularizations of contact mechanics more generally.

2.5 Deformation of the fluid mesh

While we have now fully specified the physical problem being solved, we have yet to specify how the reference-to-physical displacement $\hat{\mathbf{u}}$ is defined in the fluid subdomain. In this work, we define it to satisfy an artificial elasticity-like problem with 2nd Piola–Kirchhoff stress

$$\hat{\mathbf{S}} = \hat{K}\left(\text{tr}\hat{\mathbf{E}}\right)\mathbf{I} + 2\hat{\mu}\left(\hat{\mathbf{E}} - \frac{\text{tr}\hat{\mathbf{E}}}{3}\mathbf{I}\right), \tag{30}$$

where $\hat{\mathbf{E}}$ is the Green–Lagrange strain associated with the motion $\hat{\boldsymbol{\phi}}$ and $\hat{\mu}$ and $\hat{K}$ are artificial shear and bulk moduli. These artificial material parameters are scaled inversely by a power of $\hat{J}$, i.e.,

$$\hat{\mu} = \hat{K} = \hat{K}_0 \hat{J}^{-p}, \tag{31}$$

where $\hat{K}_0 > 0$ is a reference stiffness value for the fictitious problem and larger values of $p \geq 0$ penalize deformations that locally flatten volume elements. This scaling of artificial material parameters is related to earlier work on Jacobian-based mesh stiffening [36, 53–57], and has the goal of preserving physical-domain mesh element quality in the fluid subproblem. The reference-to-physical displacement field is subject to the boundary condition

$$\hat{\mathbf{u}} = \mathbf{u}_{\text{so}} \quad \text{on} \quad \overline{\Omega_y^{\text{so}}} \cap \overline{\Omega_y^{\text{f}}}. \tag{32}$$

It may also be subject to other Dirichlet boundary conditions on $\partial\Omega_y \cap \partial\Omega_y^{\text{f}}$, which depend on the particular problem being solved.

3 FSI with the dynamic augmented Lagrangian method and explicit geometry

Our discretizations of the shell, solid, and fluid subproblems use well-known formulations, as mentioned in Sects. 2.2 and 2.1. The discretization of the kinematic constraint from Sect. 2.3 uses a method introduced in [21], which we now refer to as the dynamic augmented Lagrangian (DAL) scheme [58, 59], along with a time-explicit treatment of the computational geometry problem of computing the shell–mesh intersection, which increases robustness without incurring any of the "added mass" instabilities [60] that often plague explicit fluid–structure coupling.

3.1 The dynamic augmented Lagrangian method

DAL is formulated in the discrete-in-time setting, where we want to enforce the constraint

$$C_\lambda(\mathbf{v}_{\text{fs}}^{n+\alpha}, \mathbf{v}_{\text{sh}}^{n+\alpha}, \delta\lambda) = \int_{\Gamma_0} \left(\mathbf{v}_{\text{fs}}^{n+\alpha} - \mathbf{v}_{\text{sh}}^{n+\alpha}\right) \cdot \delta\lambda \, d\Gamma_0 = 0 \quad \forall \delta\lambda \in \mathcal{V}_\lambda, \tag{33}$$

where $n + \alpha$ is an intermediate time level between steps n and $n + 1$, where the constraint enforcement is collocated. Typically, the choice of α here is tied to some initial baseline time integration scheme used to discretize time derivatives in the fluid–solid and shell subproblems. For instance, if the subproblems are discretized with backward Euler, we would use $\alpha = 1$, if they were discretized with the implicit midpoint rule, we would use $\alpha = 1/2$, and if they were discretized with generalized-α [61, 62], we would use $\alpha = \alpha_{\text{f}}$, where α_{f} is interpreted following the notation conventions of [36] rather than the original reference.

Emphasizing constraint enforcement and suppressing unnecessary functional arguments for visual clarity, we write our discrete-in-time formulation as: Find $\mathbf{v}_{\text{fs}}^{n+1}$, p^{n+1}, $\mathbf{v}_{\text{sh}}^{n+1}$, and λ^{n+1} such that for all $\mathbf{w}_{\text{fs}}$, q, $\mathbf{w}_{\text{sh}}$, and $\delta\lambda$,

$$R_{\text{fs}} + R_{\text{sh}} + \int_{\Gamma_0} \left(\mathbf{w}_{\text{fs}} - \mathbf{w}_{\text{sh}}\right) \cdot \lambda^{n+1}\, d\Gamma_0 + \int_{\Gamma_0} \left(\mathbf{v}_{\text{fs}}^{n+\alpha} - \mathbf{v}_{\text{sh}}^{n+\alpha}\right) \cdot \delta\lambda\, d\Gamma_0 + \int_{\Gamma_0} \beta\left(\mathbf{v}_{\text{fs}}^{n+\alpha} - \mathbf{v}_{\text{sh}}^{n+\alpha}\right) \cdot \left(\mathbf{w}_{\text{fs}} - \mathbf{w}_{\text{sh}}\right) d\Gamma_0 = 0\,, \tag{34}$$

where we have added a penalty term, with penalty parameter $\beta \geq 0$, to augment the constraint enforcement of the Lagrange multiplier λ^{n+1}. In the spatially continuous setting, this penalty term is technically redundant in light of the Lagrange multiplier terms, but it will play a crucial role in the derivation of the DAL method.

To obtain the DAL method, we first regularize the formulation (34) using only a *scalar* Lagrange multiplier associated with the normal part of the kinematic constraint:

$$R_{\text{fs}} + R_{\text{sh}} + \int_{\Gamma_0} \lambda^{n+1}\left(\mathbf{w}_{\text{fs}} - \mathbf{w}_{\text{sh}}\right) \cdot \mathbf{n}_{\text{sh}}^{n+\alpha}\, d\Gamma_0 + \int_{\Gamma_0} \delta\lambda\left(\mathbf{v}_{\text{fs}}^{n+\alpha} - \mathbf{v}_{\text{sh}}^{n+\alpha}\right) \cdot \mathbf{n}_{\text{sh}}^{n+\alpha}\, d\Gamma_0 + \int_{\Gamma_0} \beta\left(\mathbf{v}_{\text{fs}}^{n+\alpha} - \mathbf{v}_{\text{sh}}^{n+\alpha}\right) \cdot \left(\mathbf{w}_{\text{fs}} - \mathbf{w}_{\text{sh}}\right) d\Gamma_0 = 0\,, \tag{35}$$

where $\mathbf{n}_{\text{sh}}^{n+\alpha}$ is the unit normal vector to the deformed shell structure midsurface at time level $n+\alpha$. In this modified problem, the tangential portion of the constraint is regularized, and enforced entirely through the penalty term. If the penalty β scales inversely with a refinement parameter $h \to 0$, this still enforces the tangential constraint exactly in fully-converged analyses.

The essential feature of the DAL method is to update the Lagrange multiplier explicitly, using the forces computed from an implicit treatment of the penalty term. This results in a two-step solution procedure for each time step: Find $\mathbf{v}_{\text{fs}}^{n+1}$, p^{n+1}, and $\mathbf{v}_{\text{sh}}^{n+1}$ such that for all $\mathbf{w}_{\text{fs}}$, q, and $\mathbf{w}_{\text{sh}}$,

$$R_{\text{fs}} + R_{\text{sh}} + \int_{\Gamma_0} \lambda^{n}\left(\mathbf{w}_{\text{fs}} - \mathbf{w}_{\text{sh}}\right) \cdot \mathbf{n}_{\text{sh}}^{n+\alpha}\, d\Gamma_0 + \int_{\Gamma_0} \beta\left(\mathbf{v}_{\text{fs}}^{n+\alpha} - \mathbf{v}_{\text{sh}}^{n+\alpha}\right) \cdot \left(\mathbf{w}_{\text{fs}} - \mathbf{w}_{\text{sh}}\right) d\Gamma_0 = 0\,, \tag{36}$$

then find λ^{n+1} such that for all $\delta\lambda$

$$\int_{\Gamma_0} \left(\lambda^{n+1} - \left(\lambda^{n} + \beta\left(\mathbf{v}_{\text{fs}}^{n+\alpha} - \mathbf{v}_{\text{sh}}^{n+\alpha}\right) \cdot \mathbf{n}_{\text{sh}}^{n+\alpha}\right)\right) \delta\lambda\, d\Gamma_0 = 0\,. \tag{37}$$

If $\mathcal{V}_\lambda$ is a piecewise-constant space defined on Γ_0 and the integral $\int_{\Gamma_0}$ is approximated using one quadrature point per element on this space, then a static condensation procedure provides the following explicit formula for updating the Lagrange multiplier field:

$$\lambda^{n+1} = \lambda^{n} + \beta\left(\mathbf{v}_{\text{fs}}^{n+\alpha} - \mathbf{v}_{\text{sh}}^{n+\alpha}\right) \cdot \mathbf{n}_{\text{sh}}^{n+\alpha}\,. \tag{38}$$

In this way, the DAL approach eliminates the need to solve a discrete saddle point problem, while also allowing for strong constraint enforcement in the steady limit with only moderate values of the penalty value β, which do not harm the conditioning of the implicit algebraic problem solved at each time step. Extensive discussion of DAL and comparison with related methods can be found in [59, Section 4.4].

Remark 2 References [58, 63] introduced improvements to the basic DAL method described above. These improvements allow for *a priori* error estimation in the context of a simplified model problem [64], and can greatly improve the quality of the approximate Lagrange multiplier. However, their impact on the velocity solutions of the fluid and shell structure are minimal in most cases, and they are conceptually orthogonal to the main contributions of the present work.

Remark 3 An important question about DAL is: How well does it prevent flow through immersed structures? A naive investigation into this question might focus on the constraint residual, $\mathbf{v}_{\text{fs}}^{n+\alpha} - \mathbf{v}_{\text{sh}}^{n+\alpha}$. However, the effect of the concentrated multiplier and penalty forces on the accuracy of the surrounding fluid solution is of far greater significance than constraint satisfaction to apparent flow through the structure. The numerical tests of [21, Section 4.4] show how solutions with accurate satisfaction of the constraint can still have large apparent leakage through the structure. This is fundamentally due to stabilized FE formulations not being "pressure-robust" in the sense defined by [65]. The solution of [21] was to locally modify stabilization parameters near the immersed boundary. This solution is analyzed mathematically in [24, Appendix A] and found effective by [32, 66–68] across a variety of immersed-boundary methods and software frameworks. The present work also uses this local modification of stabilization parameters. An alternative solution is to use a pressure-robust formulation for the background fluid, as done in [24, 69, 70]. A pressure-robust stabilized IGA formulation was recently implemented with FEniCS/tIGAr in [71], but CouDALFISh is designed around a classical FE discretization with the standard VMS formulation cited earlier.

3.2 Explicit treatment of geometry

When resolving the problem (36) implicitly, the location of deformed shell midsurface points pulled back to Ω_y by $\hat{\boldsymbol{\phi}}$ depends on the unknown shell structure displacement

$\mathbf{u}_{\text{sh}}^{n+1}$ and the displacement $\hat{\mathbf{u}}^{n+1}$ of Ω_y to Ω_x, which depends, through the boundary condition (32), on the unknown solid displacement $\mathbf{u}_{\text{so}}$. When a time-independent finite element mesh for the background fluid–solid subproblem is defined on Ω_y, this means that material points on the shell midsurface may jump between elements of the background mesh during an iterative nonlinear solution process, making convergence difficult, due to non-smoothness of the residual.

Reference [32] circumvented this difficulty by treating the geometry of the shell–mesh intersection explicitly. In that work, the background mesh was static, so this reduced to using an explicit predictor for $\mathbf{u}_{\text{sh}}^{n+1}$ to locate the $(n+\alpha)$-level shell–mesh intersection throughout the nonlinear iteration, while still updating $\mathbf{u}_{\text{sh}}^{n+1}$ implicitly in the nonlinear residual (36) to avoid added mass instabilities.

We now generalize this explicit treatment of geometry to the deforming-mesh setting of the present work. To maintain a fixed shell–mesh intersection throughout the nonlinear iteration to solve (36), we also use an explicit predictor for $\hat{\mathbf{u}}^{n+1}$ throughout the nonlinear iteration. We then solve the mesh deformation problem described in Sect. 2.5 once at the end of each time step to obtain $\hat{\mathbf{u}}^{n+1}$, using the converged solution of $\mathbf{u}_{\text{so}}^{n+1}$ in the boundary condition (32).

4 Implementation leveraging FEniCS and tIGAr

As discussed in Sect. 1, the primary contribution of the present work is to implement the formulation and discretization described in Sects. 2 and 3 in an open-source library that makes effective use of code generation technology. Our starting point is the library CouDALFISh (pronounced "cuttlefish" and standing for Coupling via DAL of Fluids with Isogeometric Shells), described in [32]. The original version of this library described in the cited reference coupled an isogeometric discretization of a thin shell with a stabilized finite element discretization of incompressible Newtonian fluid on a *static* domain. The isogeometric shell analysis was implemented using the library tIGAr [34], which extends FEniCS to IGA, while the fluid subproblem was implemented using pure FEniCS. To make the shell and fluid discretizations modular and individually reusable, they were abstracted behind the concise modules ShNAPr (Shell Nonlinear Analysis Programs) [72] and VarMINT (Variational Multiscale Incompressible Navier–Stokes Technology) [73] respectively.

In the language and notation of Sect. 2, the problem considered in [32] would correspond to the special case in which $\Omega_x = \Omega_y = \Omega_y^{\text{f}}$, implying that $\hat{\mathbf{u}} = \mathbf{0}$, $\Omega_y^{\text{so}} = \emptyset$, the background solid subproblem can be neglected entirely, and there is no need to formulate (much less solve) a fictitious problem for $\hat{\mathbf{u}}$ in Ω_y^{f}. Restriction to this special case greatly simplified the implementation of CouDALFISh. We refer the reader to [32] for a detailed discussion of the initial version of CouDALFISh, and focus the present section on what has been modified since then, to generalize it to the problem setting described in Sect. 2. In particular, we discuss how code generation simplifies treatment of the deforming fluid domain (Sect. 4.1) and how we integrate this treatment into the overall solution algorithm of CouDALFISh (Sect. 4.2).

4.1 Transparent pullbacks to the reference domain

The pullback of the fluid subproblem (7) to Ω_y, summarized as the substitutions (10)–(13), would be tedious to implement directly in a finite element solver without leveraging code generation. In many traditional finite element implementations, this change of variables is simply offloaded to existing shape function routines, under the assumption that the reference-to-physical displacement $\hat{\mathbf{u}}$ is approximated using isoparametric elements. In such implementations, the coordinates of nodes in the mesh data structure are updated using $\hat{\mathbf{u}}$, and (7) is assembled over the deformed mesh. However, this approach can be limiting for various reasons:

- One might wish to use an analytical or superparametric deformation of the reference domain.
- The hyperelastic background solid subproblem and the artificial problem for $\hat{\mathbf{u}}$ in the fluid subdomain are more naturally posed on Ω_y.
- Expressing the pullback symbolically as part of the problem residual enables automated differentiation of the residual with respect to $\hat{\mathbf{u}}$. This functional derivative may be needed in some types of implicit solvers or shape optimization schemes.

As such, to maximize the generality of our implementation (which we may wish to use in contexts other than immersed FSI), we prefer to pull the fluid subproblem back to the reference domain. Fortunately, this choice adds no significant complexity to the implementation when taking advantage of modern code generation capabilities.

The value of UFL for simplifying the implementation of the fluid subproblem on Ω_y lies in its extensibility. Because UFL is embedded within the Python programming language, it can be augmented with new auxiliary operators by simply defining Python functions. For example, the spatial gradient operator defined by the substitution (12) can be implemented as

```
def gradx(f,x):
    return dot(grad(f),inv(grad(x)))
```

where `x` is a symbolic UFL expression of spatial position $\mathbf{x} = \hat{\boldsymbol{\phi}}(\mathbf{y}) = \mathbf{y} + \hat{\mathbf{u}}(\mathbf{y})$, the native UFL `grad` function acts as ∇_y when the finite element mesh is defined on Ω_y (so that `grad(x)` is effectively $\hat{\mathbf{F}}$), `dot` contracts over the last index of its first argument and first index of its second, and `inv` symbolically inverts a UFL matrix. This applies naturally to a tensor `f` of arbitrary rank, with the unambiguous index-notation interpretation

$$\hat{F}_{kA} = \frac{\partial \hat{u}_k}{\partial y_A} + \delta_{kA} \quad \text{and} \quad (\nabla_x \mathbf{f})_{i\cdots jk} = \frac{\partial f_{i\cdots j}}{\partial y_A} F^{-1}_{Ak} \,. \tag{39}$$

In this work, we augment functions from the original implementation of VarMINT used by [32] to include optional keyword arguments for the spatial position $\mathbf{x}$, which are then used to apply the corresponding domain pullback, via easily understandable constructions like the `gradx` function listed out explicitly above.

Remark 4 As explained in [34], a similar change-of-variables implementation is used for IGA by tIGAr, wherein DOLFIN operates directly on a mesh in the spline parameter space.

4.2 Temporary mesh motion and solution algorithm

In the specific context of immersed FSI, updating the mesh to be in Ω_x is overwhelmingly advantageous for one specific task: locating the background elements containing deformed images of quadrature points on Γ_0. DOLFIN already implements robust computational geometry routines for locating points in finite element meshes, but these are only applicable to immersed FSI if the mesh is in Ω_x. Thus, we follow the approach of *temporarily* updating the mesh by adding $\hat{\mathbf{u}}$ to its vertices, performing this computational geometry operation, then reverting the mesh to its original position in Ω_y, over which the fluid, solid, and reference-to-physical mapping problems are posed. Thus, the overall algorithm for computing the $(n+1)$-level solution within a single time step can be summarized as follows:

1. Compute explicit predictors for the $(n+1)$-level fluid–solid velocity, shell structure displacement, and mesh displacement.
2. Compute the $(n+\alpha)$-level deformed positions of fluid–shell coupling points using the predicted $(n+1)$-level shell structure displacement.
3. Repeat the following steps until reaching a converged[6] solution to the nonlinear problem (36). Each pass of these steps is referred to as a "block iteration", following the terminology of [74]. (This is a generalization of the block iteration algorithm analyzed in [24, Section 4] for the case of a static fluid domain.)
   (a) Assemble the vector and matrix corresponding to the monolithic fluid–solid residual (6) and its derivative with respect to $\mathbf{v}_{\text{fs}}^{n+1}$.
   (b) Temporarily move the fluid–solid mesh of Ω_y to the current configuration (using the explicit predictor for $\hat{\mathbf{u}}^{n+1}$, as discussed in Sect. 3.2).
   (c) Add contributions from the Γ_0 integrals in (36) to the assembled fluid–solid system.
   (d) Apply Dirichlet boundary conditions to the assembled fluid–solid residual vector and Jacobian matrix.
   (e) Solve for an increment of the fluid–solid solution and update the current values.
   (f) Assemble the vector and matrix corresponding to the shell subproblem residual (24) and its derivative with respect to $\mathbf{u}_{\text{sh}}^{n+1}$.
   (g) Add contributions to the assembled shell structure system corresponding to the contact residual (26) and its Jacobian.
   (h) Add contributions from the Γ_0 integrals in (36) to the assembled shell structure system, using the updated fluid solution from step 3e.
   (i) Apply Dirichlet boundary conditions to the assembled shell structure system.
   (j) Solve for an increment of the shell displacement solution and update the current values.
   (k) Move the mesh back to its static reference configuration.
4. Solve the fictitious hyperelastic mesh deformation problem of Sect. 2.5 to obtain $\hat{\mathbf{u}}^{n+1}$, using the converged solution for $\mathbf{u}_{\text{so}}^{n+1}$ as a boundary condition.

Remark 5 Our reliance on temporary modification of the mesh data structure for applying point sources in Ω_x effectively limits the mesh displacement $\hat{\mathbf{u}}$ to being piecewise linear, due to many of the computational geometry routines in FEniCS being restricted to affine simplicial elements.

[6] By default, convergence is tested by computing ℓ^2 norms of assembled residual vectors for both the fluid–solid and shell subproblems, normalizing these against residual norms computed at the start of the iteration, and checking whether both fall below a given relative tolerance.

Because $\hat{\mathbf{u}}$ must coincide with the solid displacement in Ω_y^{so} and we assume a single continuous FE space on Ω_y for $\mathbf{v}_{\text{fs}}$, the piecewise-linear restriction on $\hat{\mathbf{u}}$ propagates to the fluid velocity. This limitation on the polynomial degree of velocity elements imposes high resolution requirements for fully resolving turbulent flow features. Section 6 discusses the implications of this for heart valve FSI.

5 Benchmark tests

This section summarizes the results of benchmark testing to verify the implementations of pullbacks to Ω_y and computation of immersed boundary forces on a deforming Ω_x^{f}.

5.1 Navier–Stokes on a deforming domain

To test the deforming-domain fluid formulation in the updated library VarMINT, we use the classic 2D Taylor–Green vortex problem [75], a special case of the Navier–Stokes equations for incompressible fluids that has an analytic solution. This problem allows us to verify the implementation of the fluid ALE-VMS formulation from Sect. 2.1.1 and the pullbacks to the reference configuration described in Sect. 4.1. The exact two-dimensional velocity solution is

$$\mathbf{v}(\mathbf{x},t) = \big(\sin(x_1)\cos(x_2)\mathbf{e}_1 - \cos(x_1)\sin(x_2)\mathbf{e}_2\big)e^{-2\nu_{\text{f}}t}, \quad (40)$$

where $\nu_{\text{f}} = \mu_{\text{f}}/\rho_{\text{f}}$ is the kinematic viscosity and $\{\mathbf{e}_i\}$ are standard Cartesian basis vectors in physical space. We choose the spatial domain $\Omega_x = [-\pi,\pi]\times[-\pi,\pi]$. The problem is solved on the static reference domain Ω_y, which deforms to Ω_x via a nontrivial time-dependent deformation with the prescribed displacement field

$$\hat{\mathbf{u}}(\mathbf{y},t) = \frac{1}{5}\sin\big(y_1\big)\sin\big(y_2\big)\sin\left(\frac{2\pi t}{T}\right)\big(\mathbf{e}_1+\mathbf{e}_2\big), \quad (41)$$

where T is the simulation duration. To simplify the enforcement of boundary conditions, the displacement field is chosen to specifically maintain $\partial\Omega_y = \partial\Omega_x$.

The computations presented here use continuous degree-one finite element spaces for the pressure and all velocity components with $\nu_{\text{f}} = 0.01$ and $T = 1$. Each mesh is generated from a uniform $N\times N$ structured quadrilateral mesh divided into triangles. The computation uses an implicit midpoint time-stepping scheme and takes N time steps to uniformly cover the interval $[0, T]$. The exact velocity solution is strongly enforced across the domain boundary and the exact pressure is enforced at a single corner of the domain. For further details, the reader is referred to the published GitHub repository [73] that contains the script needed to reproduce the results shown here.

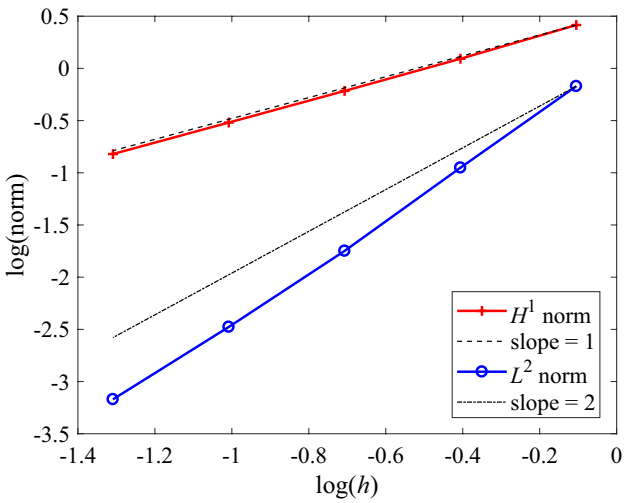


Fig. 1 Optimal convergence of Taylor–Green vortex problem to the analytical solution for given element sizes, h

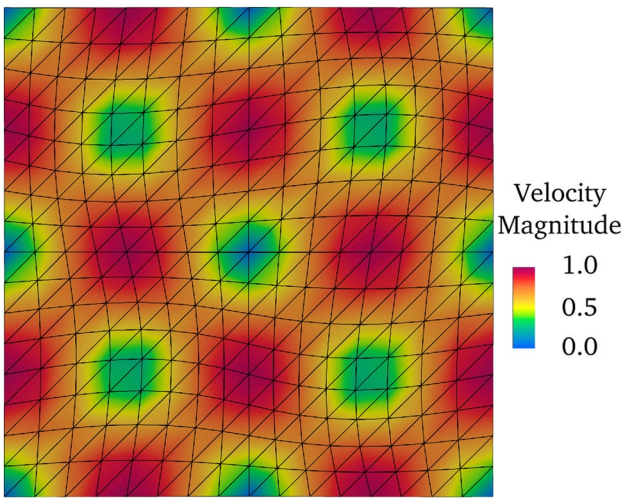


Fig. 2 Representative solution snapshot of the decaying Taylor–Green vortex on a deformed domain. The solution presented here was computed on a 16×16 ($\log(h) = -0.406$) uniform quadrilateral mesh split into triangles and is shown at $t = 0.21875$

Figure 1 shows the convergence of solutions to this problem under mesh refinement. We see optimal first- and second-order convergence of the H^1 and L^2 norms of the error, respectively. The reported error norms are calculated at $t = T$ and plotted against a mesh-size parameter, $h = \frac{2\pi}{N}$. A representative snapshot of a numerical solution is also shown in Fig. 2, illustrating how the qualitative structure of the solution in Ω_x is preserved, despite the motion of the mesh.

Table 1 Setup cases for the Turek–Hron FSI3 benchmark where N is the total number of triangular elements and h is the mesh size at the interface

Setup	N	h (mm)
M1	17,327	8
M2	34,684	4
M3	77,985	2
M4	192,251	1

5.2 Conforming FSI: The Turek and Hron benchmark

We next consider a test of the coupled fluid–solid problem without any immersed shell structure to validate the boundary-fitted coupling and mesh motion implementations behind (6). For this, we use the 2D benchmark proposed by Turek and Hron [76] and studied extensively by other authors (e.g., [77–82]). The benchmark involves incompressible channel flow over a rectangular block of solid material, which is fixed to a cylinder at its leading edge and deflects periodically due to vortex shedding. The FSI3 case of [76] is adopted here. Detailed information on the problem setup and a set of converged reference results are provided by [76]. Here we give a brief summary. The length and height of the channel domain are 2.5 m and 0.41 m, respectively. The center of the cylinder is positioned at $(0.2\,\text{m}, 0.2\,\text{m})$ measured from the bottom-left corner of the channel. The radius of the cylinder is 0.05 m. The elastic structure bar has a length and height of 0.35 m and 0.02 m, respectively. The bottom-right corner of the structure is positioned at $(0.6\,\text{m}, 0.19\,\text{m})$ and the left end is fully attached to the fixed cylinder. The setting is intentionally non-symmetric in the vertical direction to ensure that the onset of a vortex street does not rely on an initial small perturbation.

A parabolic inflow velocity profile [76, Eq. (10)] is prescribed at the left channel boundary with a mean inflow velocity of 2.0 m/s. A smooth increase of the velocity profile in time [76, Eq. (11)] is applied as the initial inflow condition. The outflow boundary condition is traction-free, and the no-slip condition is applied on the remaining boundaries, including the fluid–structure interface. The structure is elastic and compressible, modeled as a St. Venant–Kirchhoff material. The density of the structure is $\rho_0^{\text{so}} = 1.0 \times 10^3$ kg/m^3, the Poisson's ratio is $\nu = 0.4$, and the shear modulus is $\mu_s = 2.0$ MPa. The fluid is incompressible Newtonian. The density is $\rho_f = 1.0 \times 10^3$ kg/m^3 and the kinematic viscosity is $\nu_f = 1.0 \times 10^{-3}$ m^2/s.

Each finite element mesh of the domain is generated with Gmsh [83]. Table 1 describes the meshes size, h, near the fluid–object interface (both the cylinder and the structure). The maximum element size for all of the mesh levels is $h_{\max} = 15$ mm. The interface mesh size h is maintained for any elements within 25 mm of the interface. Outside of the uniformly-refined region, the elements are sized according to a linear interpolation between h and $h_{\max}$ over 525 mm. In the absence of refinement, the elements maintain an average size of $h_{\max}$ and conform to the outer boundary.

We discretize this 2D problem using the updated version of VarMINT for the deforming fluid subdomain and a standard Bubnov–Galerkin formulation of hyperelasticity in the solid subdomain. The problem utilizes a fully implicit backward-Euler time integration scheme and a consistent time step size of 1.0×10^{-3} s between different mesh refinement levels. The solid displacement at the fluid–structure interface and the channel domain boundaries are used as boundary conditions for the mesh problem, solved as a hyperelasticity problem for the internal mesh displacement per (30). The strongly coupled fluid–solid problems and mesh motion problem are solved iteratively until both converge in a typical block-iterative fashion [74] to validate the fluid–solid coupling. That is, this benchmark does *not* make use of the CouDALFISh solution algorithm described in Sect. 4.2 so that the two coupling types (conforming fluid–solid and immersed fluid–shell) could be independently validated. The sparse, parallel, direct solver MUMPS [84] was used as the linear solver for both the fluid–solid and mesh problems. Again, the reader is referred to the GitHub repository [73] that contains the scripts used to produce the results here.

A representative snapshot of the results on a coarse mesh (M1) is shown in Fig. 3. The snapshot displays the velocity field and highlights the boundary-fitted nature of the mesh. Table 2 shows the comparison quantity results over several mesh resolutions, which compare well with the published benchmark results in [76]. The displacements in Table 2 are evaluated at the center of the tip of the structure bar, initially located at $(0.6\,\text{m}, 0.2\,\text{m})$. The displacements are presented here in terms of their mean, amplitude, and frequency in accordance with [76]. The lift and drag forces in Table 2 are computed over the entire fluid–object boundary using the conservative definition of traction [85, Eq. (37)], which accounts for any contributions from the stabilization terms at the boundary.

5.3 2D valve immersed in a deforming domain

We now move on to a more complicated verification test, which can be viewed as a 2D model problem sharing some features with a heart valve. Minor variants of this problem have been thoroughly studied in numerous references [21, 58, 86–88]. We refer to [21, Section 4.7] for a complete statement of the domain dimensions, boundary conditions, material properties, etc. that are reused in the present work. However, we briefly summarize the problem here. A

Table 2 Numerical results for the Turek–Hron FSI3 benchmark. u_1 and u_2 are the horizontal and vertical displacements of the structure bar recorded at the center of its tip

Setup	u_1 (mm)	u_2 (mm)	Drag (N)	Lift (N)	f_1 (Hz)	f_2 (Hz)
M1	-1.56 ± 1.53	1.74 ± 24.90	436.80 ± 20.25	-4.12 ± 190.83	11.1	5.6
M2	-2.39 ± 2.29	1.35 ± 31.41	454.24 ± 24.96	1.85 ± 171.09	11.1	5.6
M3	-2.53 ± 2.41	1.45 ± 32.51	458.19 ± 25.54	2.06 ± 156.78	11.1	5.5
M4	-2.55 ± 2.43	1.49 ± 32.66	457.74 ± 25.65	2.39 ± 154.30	11.0	5.5
REF	-2.69 ± 2.53	1.48 ± 34.38	457.30 ± 22.66	2.22 ± 149.78	10.9	5.3

Drag and lift forces are evaluated over the entire fluid–object interface

horizontal channel is modeled by a rectangular domain of 8.0×1.61 with no-slip boundary conditions on the top and bottom, a homogeneous Neumann boundary condition on the right end, and a parabolic inflow profile [21, Eq. (55)] on the left, whose amplitude varies periodically in time. Two thin beams, each with a height of 0.7 and a thickness of 0.0212, are located 2.0 from the inlet and are immersed in this channel, one with a cantilever support on the top side and the other with a cantilever support on the bottom. These beams deflect periodically, in response to the inflow profile. Although they never contact each other in the standard version of the benchmark,[7] the beams are still referred to as "leaflets", in a loose analogy to heart valve mechanics. The fluid and structure have equal densities of $\rho_f = \rho_0^{sh} = 100$. The dynamic viscosity of the fluid is $\mu_f = 10$. The beam

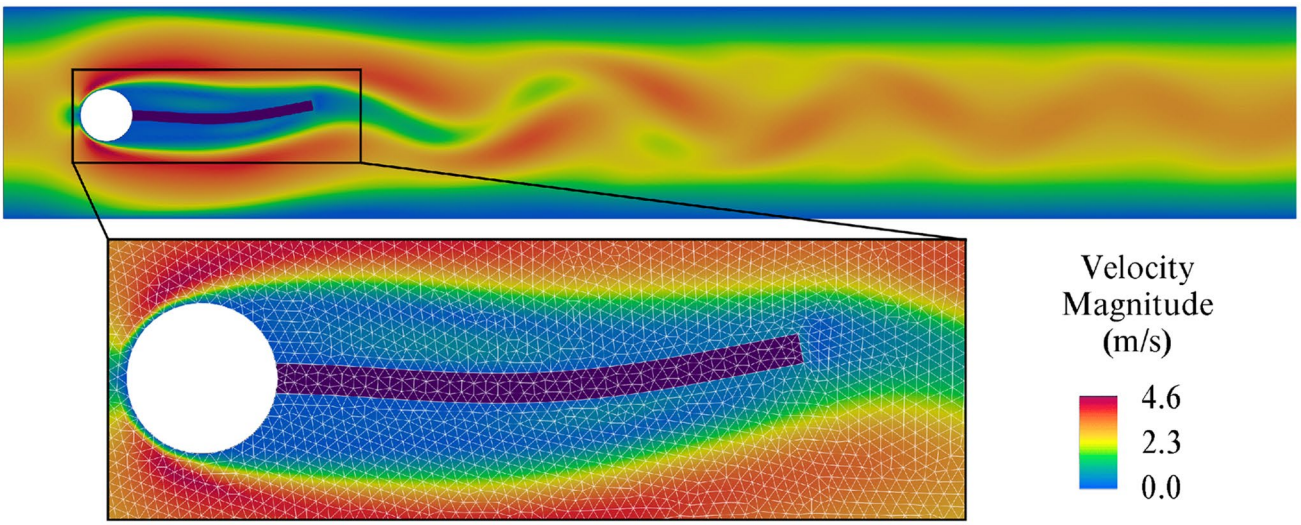
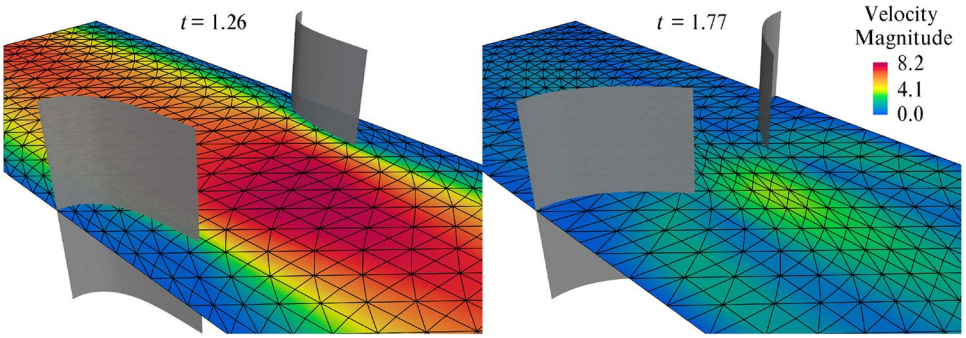


Fig. 3 A sample solution of the Turek–Hron FSI3 benchmark on mesh M1 showing the boundary-fitted nature of the mesh

Fig. 4 Sample solutions at two different time steps on a coarse mesh deformed in two different configurations



[7] A variant with leaflet coaptation is documented in [58].

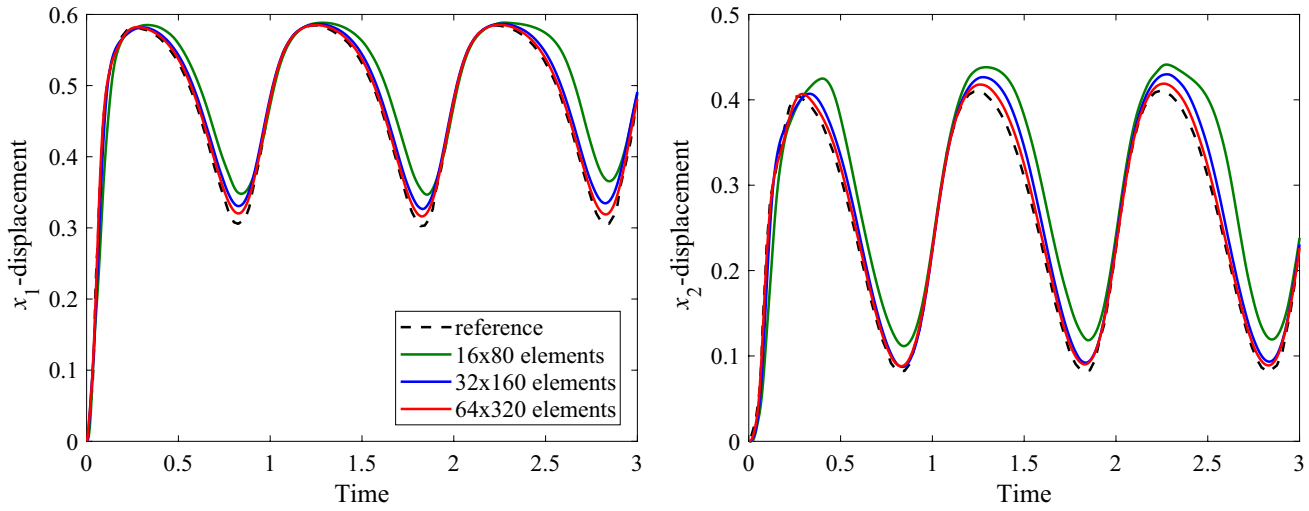


Fig. 5 Plots showing the convergence of both the x_1- and x_2-displacement of the top-leaflet tip to the boundary-fitted reference

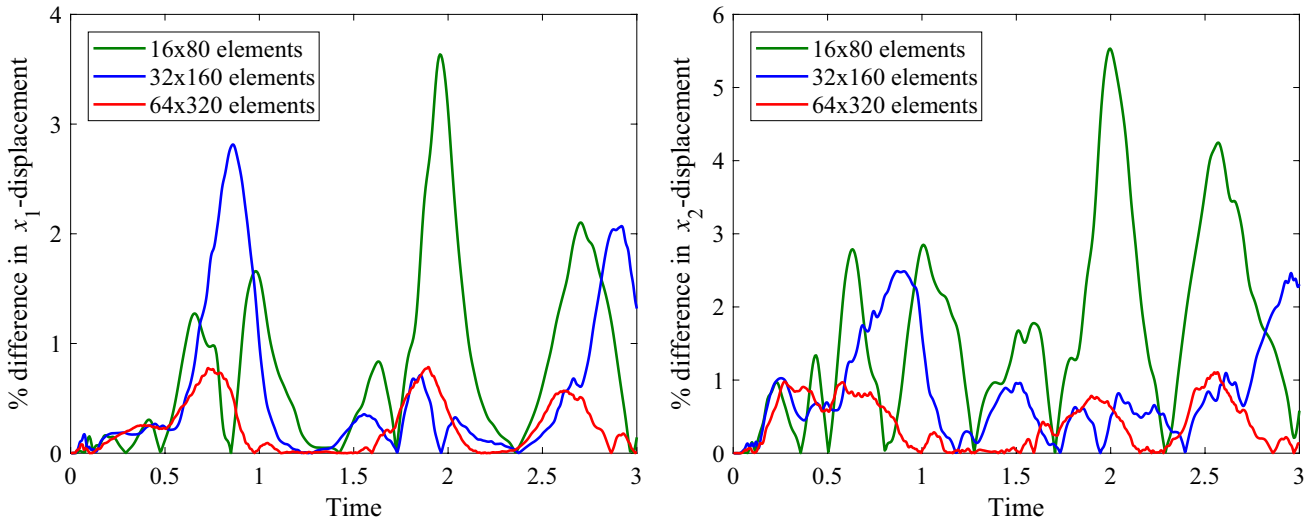


Fig. 6 Each plot shows the difference in displacement between the two tips, one plot for each independent direction of displacement (x_1 and x_2 directions). The differences are presented as percentages of the maxima with respect to time of the corresponding tip displacement components. The convergence towards zero signifies a symmetric displacement between the two leaflet tips under refinement of the fluid mesh

is modeled using a St. Venant–Kirchhoff material with $E = 5.6 \times 10^7$ and Poisson's ratio of $\nu = 0.4$.

Because immersed shells are fundamentally a 3D problem, this 2D problem was solved on a finite element mesh with a layer of two elements. To maintain the 2D nature of the problem, the out-of-plane velocity component in both the fluid and shell was strongly constrained to zero. The final discrete mesh contains $N \times 5N \times 2$ tetrahedral elements where N is the number of elements in the x_2 direction. The default structured tetrahedral mesh generated in FEniCS (`mesh = BoxMesh(N,5N,2)`) is not symmetric. We found that the asymmetric nature of the background fluid mesh introduced asymmetry in the velocity solution. To remedy this, and avoid using an external mesh generator, we discovered that the uniform refinement algorithm in FEniCS (`mesh = refine(BoxMesh(N/2,5N/2,1))`) produces a symmetric mesh of $N \times 5N \times 2$ elements. The temporal discretization is refined with the spatial discretization such that the time step size $\Delta t = T/(32N)$, where $T = 3.0$ is the total simulation time. In this study, three meshes, $N = 16$, 32, and 64, are considered. The interested reader is referred to the published CouDALFISh repository [33] for the complete code (geometry, mesh, boundary conditions, solver algorithms, problem script, etc.) to reproduce the exact problem setup and results shown here.

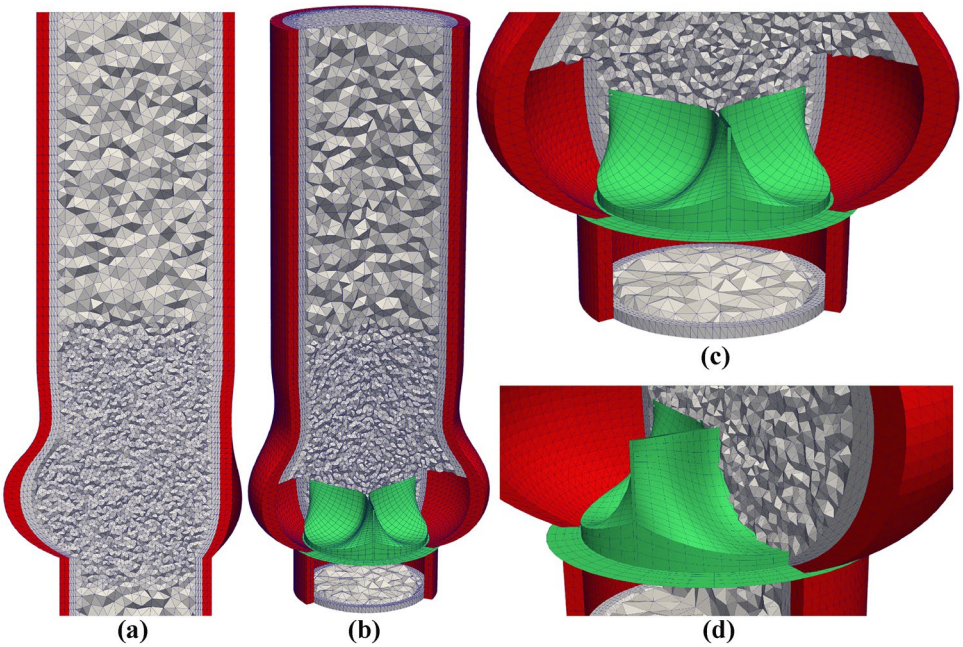


Fig. 7 Subfigure **a** shows a cross-section of the computational mesh for the heart valve problem. The flow "inlet" is on the bottom and the flow "outlet" is on the top. The solid domain (112, 752 structured tetrahedral elements) is shown in red and fluid domain (629, 964 elements) in gray. Subfigure **b** shows the relative positioning of the valve and the background mesh. The valve stent base is aligned with the root of the sinus, shown in more detail in **c**. Subfigure **d** shows the relative sizes of fluid elements and shell elements along the valve's belly region

This problem was previously used to verify the initial implementation of CouDALFISh in [32], with the stationary-domain restriction of $\Omega_x = \Omega_y = \Omega_x^{\mathrm{f}}$. As with the 2D Taylor–Green problem in Sect. 5.1, we now solve with a nontrivial time-dependent deformation of Ω_y, given by displacement

$$\hat{\mathbf{u}}(\mathbf{y}, t) = A \sin\left(2\pi \frac{y_1}{L}\right) \sin\left(2\pi \frac{y_2}{H}\right) \sin\left(f\pi \frac{t}{T}\right)(\mathbf{e}_1 + \mathbf{e}_2), \tag{42}$$

where $A = 0.15$ is a chosen displacement amplitude, $f = 8$ is a chosen oscillation frequency, $T = 3$ is the total simulation time, and $L = 8$ and $H = 1.61$ are the domain length and height, respectively.

Several representative snapshots of the solution are shown in Fig. 4, demonstrating that the overall symmetry of the solution is preserved, despite a mesh deformation that does respect that symmetry. We also compare the results with a converged reference solution from [21, Section 4.7.2]. In particular, we look at the deflection of the top leaflet as a function of time. The time histories plotted in Fig. 5 show a clear convergence toward the reference solution as the immersed discretizations are refined. Figure 6 also plots the difference between the tip deflections of the top and bottom leaflets for each immersed discretization. The asymmetry clearly converges to zero, despite the asymmetric mesh deformation.

6 Application to FSI of a bioprosthetic heart valve

Having verified the accuracy of the deforming-domain fluid solver, using benchmarks for both pure Navier–Stokes and FSI, we move on to a complex application, namely, FSI simulation of a bioprosthetic heart valve immersed in a flexible artery. We summarize the problem setup in Sect. 6.1, followed by a presentation and discussion of the simulation results in Sect. 6.2. For all results in this section, the geometry definitions, mesh creation scripts, boundary and initial condition files, solver algorithms, problem scripts, and basic post-processing tools are included in the CouDALFISh repository on GitHub [33] for simulation reproducibility and transparency.

6.1 Geometry, boundary conditions, and parameters

Many of the main modeling assumptions for the background artery FSI (Sect. 6.1.1) and immersed leaflet shell structure (Sect. 6.1.2) are adopted from prior work, e.g. [21–25, 89–92], but we reiterate them here for completeness.

6.1.1 Fluid–solid background problem

The background fluid–solid domain is defined in its reference configuration, Ω_y, to model the aortic root, sinuses, and a short portion of the ascending aorta. Our finite element mesh of this geometry is shown in Fig. 7, consisting of 742, 716 tetrahedral elements. The solid domain is

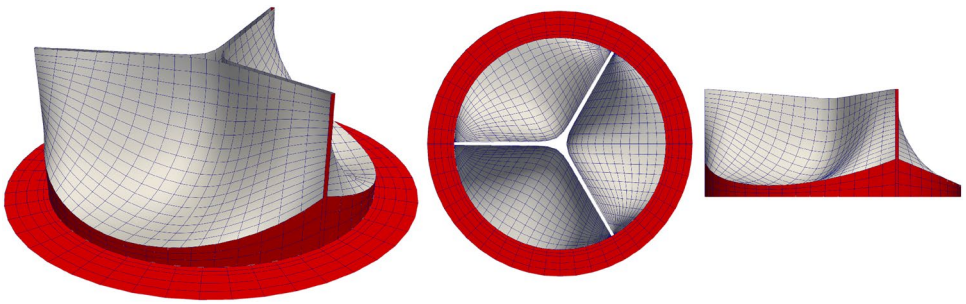


Fig. 8 Isometric, top, and side views of the valve geometry and discretization with the leaflet patches shown in white and the stent patches shown in red

comprised of 112, 752 structured elements in two layers. The fluid domain is 629, 964 elements split between three layers of boundary-layer structured elements and a center unstructured region that is refined near the valve. The mesh was generated with the open-source meshing tool Gmsh [83]. We designate the ventricular end as the "inflow" and the aortic end as the "outflow", based on the nominal flow direction permitted by the valve. The artery wall thickness is 14% of the lumen radius at a given axial cross-section, while the overall length of the domain from inflow to outflow is 10.5 cm. The diameter of the artery lumen's cross section is 2.3 cm at the inflow and 3.0 cm at the outflow, following the aorta geometry from [90]. The bottom of the sinus (aortic annulus) is located 1.0 cm from the inlet.

Following [22], we use a neo-Hookean hyperelastic solid with mass damping and a dilational penalty as an effective model for the elasticity of the artery and dissipation due to interaction with surrounding tissue. (We do not claim that this provides accurate stresses within the artery wall, as it neglects the layered structure.) Specifically, the second Piola–Kirchhoff stress in the material is given by

$$\mathbf{S} = \mu_s J^{-2/3}\left(\mathbf{I} - \frac{1}{3}(\mathrm{tr}\mathbf{C})\mathbf{C}^{-1}\right) + \frac{1}{2}\kappa\left(J^2 - 1\right)\mathbf{C}^{-1}\,, \tag{43}$$

where we choose a Young's modulus $E = 1.0 \times 10^7$ dyn/cm^2 and the Poisson's ratio $\nu = 0.45$. The shear modulus and bulk modulus, respectively, are found from the following equations: $\mu_s = \frac{E}{2(1+\nu)}$ and $\kappa = \frac{E}{3(1-2\nu)}$. For the mass-proportional damping term, we choose $C_d = 1.0 \times 10^{-4}$ s^{-1}. The density of the solid is $\rho_0^{so} = 1.0$ g/cm^3. The solid subdomain is subject to sliding boundary conditions [93] at the inflow and outflow, and is fixed at its intersection with the prosthetic valve's stent. The outer boundaries of the artery are treated as traction-free (where, again, the effects of interaction with surrounding tissue are modeled through the choice of effective stiffness and mass damping).

[8] Although blood flow has well-documented non-Newtonian features, the Newtonian model remains appropriate in large arteries [94].

For the fluid subproblem posed within the artery lumen, we assume a Newtonian[8] dynamic viscosity of $\mu_f = 3.0 \times 10^{-2}$ g/(cm s) and a mass density of $\rho_f = 1.0$ g/cm^3. The fluid subproblem is driven by applied fluxes at the inflow and outflow. The inflow flux is given by the pressure profile

$$\mathbf{h}|_{\text{inflow}} = -p_{\text{in}}\mathbf{n}_x\,, \tag{44}$$

where the scalar pressure p_{in} varies periodically in time, following the profile of [90, Figure 18] (varying between approximately 128 mmHg during systole and −2 mmHg during diastole in a cardiac cycle of 0.86 s). The flux at the outflow is determined based on the volumetric flow rate, viz.,

$$\mathbf{h}|_{\text{outflow}} = -(p_0 + RQ)\mathbf{n}_x\,, \tag{45}$$

where $p_0 = 80$ mmHg is a baseline pressure value, $R = 70$ dyn s/cm^5 is a resistance coefficient, and

$$Q = \int_{\partial\Omega_x^f|_{\text{outflow}}} \mathbf{v}\cdot\mathbf{n}_x\, d\partial\Omega_x, \tag{46}$$

is the volumetric flow rate through the outflow. The backflow stabilization coefficient γ in (7) is set to 1 on both the inflow and outflow.

Remark 6 The choice of $\gamma = 1$ is motivated by mathematical stability analysis of model problems (e.g., the coercivity analysis of [95, Section 2.1] for advection–diffusion with Neumann boundary conditions). The coefficient γ was not considered a free parameter in the first reference [96] introducing this form of backflow stabilization. It was later introduced in [37] with the the possibility of $\gamma < 1$. Reducing γ may lessen the impact of stabilization on the flow solution while remaining stable in practice. In this example, we opt for the original choice of $\gamma = 1$. In any case, it is important to note that the value of γ, especially at the inflow, affects the calibration of resistance boundary conditions, because backflow stabilization adds a net resistance to flow when applied to both ends of a tube. Thus, resistance boundary conditions must be calibrated for specific choices of γ. (However, in the present example, we simply choose a

value of R that provides a volumetric flow rate within the physiological range, and do not attempt to calibrate it using patient-specific data.)

The velocity and pressure fields are initialized to zero throughout the whole domain. To ease convergence at the start of the simulation, the simulation starts at $t = 0.6$ s (during the diastole of the cardiac cycle) and the pressures p_{in} and p_0 are linearly increased over 0.03 s (300 steps when $\Delta t = 1 \times 10^{-4}$) until p_{in} matches the inlet pressure profile in [90, Figure 18] and p_0 matches the outlet baseline pressure. The results presented in this section correspond to the next complete cardiac cycle, which starts at $t = 0.86$ s. Based on previous experience with these problems, we observe this "first" cycle to have a well-settled solution in the fluid domain and relatively periodic valve kinematics between cardiac cycles.

6.1.2 Modeling the valve

We model a bioprosthetic valve consisting of thin leaflets clamped into a rigid stent. The valve and stent are modeled as a multi-patch B-spline surface, designed in the CAD software Rhinoceros [97]. The valve has an overall height of 0.94 cm and diameter of 2.235 cm (not including the suture ring). The outer diameter of the suture ring is 2.8 cm. The spline geometry is shown in Fig. 8. Following the IGA paradigm, this spline model also serves as the analysis mesh. The mesh has a total of 1386 cubic B-spline elements, 960 of which make up the three leaflets and 426 elements make up the stent patches. The valve is located such that the suture ring is level with the aortic annulus and centered in the aortic root, shown in Fig. 7. The stent patches are entirely fixed, while the clamped attachments of leaflets are modeled by fixing two rows of B-spline control points at each attachment edge.

Constitutive modeling of the chemically treated soft tissue used in bioprosthetic valves remains a subject of ongoing research [98–100]. It is thus useful to develop a software framework that permits rapid prototyping of different hyperelastic potentials. The present contribution achieves this through the flexible isogeometric shell analysis module ShNAPr. While an initial version of ShNAPr was introduced in [32], the present study looks deeper into its implications for studying material models in heart valve FSI. In particular, the submodule ShNAPr.hyperelastic provides a universal interface to automate the implementation of incompressible hyperelastic constitutive models, which uses computer algebra within FEniCS UFL to circumvent the manual calculation of tensor derivatives spelled out in [89]. The general form of a hyperelastic energy density for an incompressible material is

Table 3 Material properties for the different models compared in this work

Model	c_0 (dyn/cm^2)	c_1 (dyn/cm^2)	c_2
NH	3,670,000	–	–
LSI	676,080	132,848	38.1878

$$\psi(\mathbf{E}) = \psi_{\text{el}}(\mathbf{E}) - p(J-1)\,, \tag{47}$$

where p is a Lagrange multiplier to enforce the constraint that $J = 1$. To perform simulations with a given hyperelastic potential, one must only implement a single Python function `psi_el(E)` corresponding to $\psi_{\text{el}}(\mathbf{E})$, given a UFL representation `E` of the 3D Green–Lagrange strain in a local Cartesian coordinate system whose third basis vector is orthogonal to the shell midsurface. For example, to use an incompressible neo-Hookean model, one would simply define

```
def psi_el(E):
    mu = Constant(1e4)
    C = 2.0*E + Identity(3)
    I1 = tr(C)
    return 0.5*mu*(I1-3.0)
```

where the UFL scalar `mu` is assumed to be the shear modulus. Note that this user-defined function corresponds to the 3D constitutive model. Following the formulation [46, Section 5.1], (47) is then implemented generically within the library as

```
psi_el(E) - p*(J-1.0)
```

where

```
C = 2.0*E + Identity(3)
J = sqrt(det(C))
```

defines the Jacobian determinant `J` in terms of `E`, and `p` is the return value of the function

```
def incompressiblePressureKL(psi_el,E):
    E = variable(E)
    dpsi_el_dC = 0.5*diff(psi_el(E),E)
    C22 = 2.0*E[2,2] + 1.0
    return 2.0*dpsi_el_dC[2,2]*C22
```

which uses a plane-stress criterion on the second Piola–Kirchhoff stress to statically condense the Lagrange multiplier for an arbitrary user-defined choice of ψ_{el}. This procedure is not limited to the neo-Hookean model defined as an example earlier.

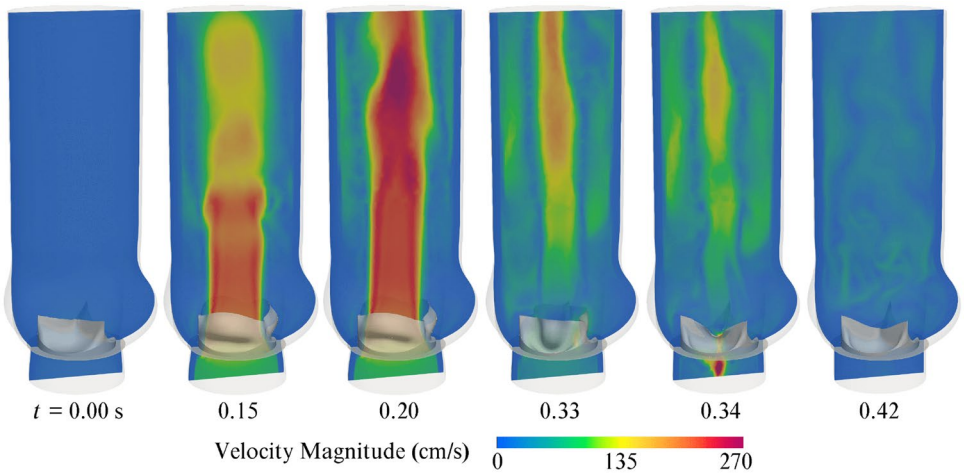


Fig. 9 A sample of the flow-field solutions at selected time steps showing the development of the jet upon opening and the stopping of flow at the valve closing. The duration of a single cardiac cycle is 0.86 s

Remark 7 The indices of `E` and `dpsi_el_dC` (corresponding to $\partial\psi_{\text{el}}/\partial\mathbf{C}$) and the variable name `C22` for the out-of-plane component of $\mathbf{C}$ follow the convention of indices starting from zero, not one, which FEniCS UFL inherits from the Python programming language it is embedded in.

As a demonstration of the value of code generation to heart valve FSI, we compare simulation results using two different material models for the valve leaflets. Specifically, we consider the isotropic neo-Hookean (NH) potential,

$$\psi_{\text{el}} = \frac{1}{2}c_0\big(I_1 - 3\big)\,, \tag{48}$$

and the isotropic Lee–Sacks (LSI) potential [89],

$$\psi_{\text{el}} = \frac{1}{2}c_0\big(I_1 - 3\big) + \frac{1}{2}c_1\Big(e^{c_2(I_1-3)^2} - 1\Big)\,, \tag{49}$$

where $I_1 = \operatorname{tr}\mathbf{C}$ and $\{c_i\}$ are material parameters that are given in Table 3.

The parameters of the Lee–Sacks models are chosen according to the calibration of [89] and the shear modulus for the neo-Hookean model is derived from the Young's modulus, E, of the St. Venant–Kirchhoff model in [23, Section 3.1] assuming a perfectly incompressible medium. That is, assuming a Poisson's ratio of 0.5, $c_0 = \mu_{\text{s}} = E/3$. Additionally, the density of the valve is $\rho_0^{\text{sh}} = 1.0\ \text{g/cm}^3$ and the leaflet thickness is 0.0386 cm in all cases, following the data reported in [89].

6.1.3 Contact parameters

The contact parameter R_{self} must be selected to be less than the initial distance between leaflets in the reference configuration, to permit contact forces between the leaflets. In this work, it is chosen as $R_{\text{self}} = 0.0308$ cm. The range of contact forces must be less than this, as mentioned earlier, and $r_{\max}$ is correspondingly set to $r_{\max} = R_{\text{self}}/1.3 \approx 0.0237$ cm. The stiffness and smoothing parameters are set to $k_{\text{c}} = 1.0 \times 10^{11}\,\text{g cm}^{-4}\text{s}^{-2}$ and $s_{\text{c}} = 0.2$, based on experience with this problem class.

6.1.4 Time-integration scheme

All heart valve FSI results here were computed with a time step size of $\Delta t = 1 \times 10^{-4}$ s and a generalized-α time integration scheme for the baseline time integrator for subproblems, on top of which DAL is applied (cf. Section 3.1). The spectral radius of the amplification matrix in the limit of $\Delta t \to \infty$ is set to $\rho_\infty = 0.0$, to maximize numerical damping of high-frequency modes, which improves robustness in complicated calculations (while maintaining formal second-order accuracy of the generalized-α scheme).

6.1.5 Nonlinear solution procedure

Section 4.2 presents the nonlinear solution procedure for the valve without any mention of the nonlinear tolerances or special considerations, so we include those here for completeness. For a full simulation of a cardiac cycle, the computation is limited to three block iterations or a relative tolerance of 1.0×10^{-3}, whichever is reached first. The size of the background mesh problem necessitates an iterative solver, but the scaling of the stability parameters near the immersed boundary [21, Section 4.4] makes it difficult to converge the linear solver within each Newton iteration. Following the original heart valve example published with CouDALFISh, we again fix the number of GMRES [101] iterations for the fluid linear solver to 300 iterations, which is usually enough to still allow the outer nonlinear iteration to converge. The shell structure problem uses a direct solver (UMFPACK [102]).

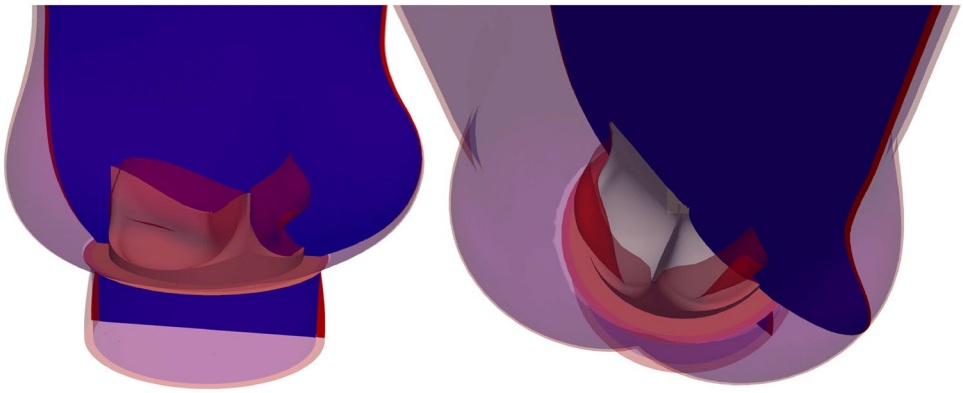


Fig. 10 Two section views showing the difference in the aorta shape between peak systole ($t = 0.25$ s, shown in red) and diastole ($t = 0.52$ s, shown in blue)

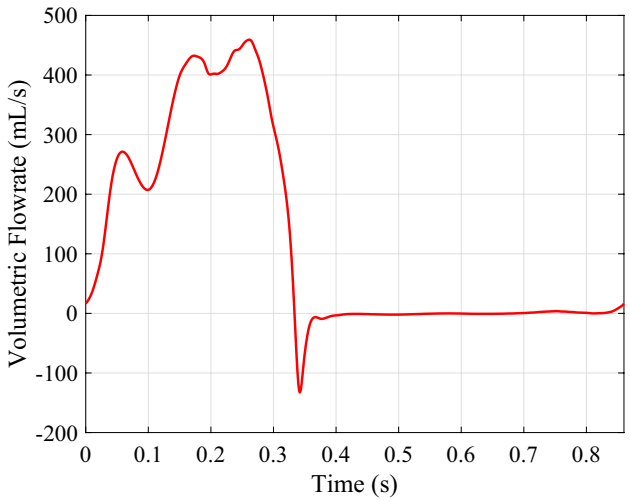


Fig. 11 The volumetric flow rate at the outlet of the domain throughout the cardiac cycle. The flow rate is quantitatively comparable to [23] and results in an overall cardiac output of 6.97 L/min

6.2 Results

Figure 9 shows slice-intersection renderings of several snapshots of the FSI analysis results for the Lee–Sacks model. The flow-field solutions show the development of the jet upon opening of the valve and the stopping of flow at the valve closing. The element size and polynomial degree in the fluid mesh downstream of the valve is not sufficient to resolve the detailed vortex dynamics expected at the flow's peak Reynolds number, especially as it encounters an adverse pressure gradient during the onset of diastole. However, the primary quantity of interest in many bioprosthetic valve analyses is the stress field within the leaflets, which is relatively insensitive to downstream vortex dynamics. Fully-resolved fluid dynamics may be relevant to other questions, though (e.g., studies on hemolysis, wall shear stress, or noise), and we refer the reader to [103–106] for examples of highly-resolved simulations of valvular hemodynamics and in-depth discussion of the turbulent flow features.

Figure 10 shows slice-intersection renderings comparing the aorta displacement at systole and diastole ($t = 0.25$ s and $t = 0.52$ s, respectively). The deformation solutions show the significant displacement of the domain, especially near the inlet during systole. The significance of arterial wall deformation for valvular FSI analysis was clearly demonstrated by [22], where it was shown to provide a mechanism for dissipating the kinetic energy of the diastolic blood hammer, damping the oscillation of the heart valve, which is clearly visible in plots of volumetric flow rate over time [22, Figure 8].[9] This important damping effect is reproduced in the present work, where the outflow flow rate in Fig. 11 exhibits no significant oscillation after valve closure. Additionally, the flow rate in Fig. 11 implies a cardiac output of 6.97 L/min and is quantitatively consistent with the results of [22, 23].

The deformation and flow field for the neo-Hookean model (48) are qualitatively similar to the Lee–Sacks results. However, a closer examination in Fig. 12 reveals significant differences in the strain (and therefore stress) distributions, which can have major implications for the long-term durability of valve leaflets [107]. Figure 12 shows snapshots of the valve leaflet deformations, with the maximum in-plane eigenvalue of $\mathbf{E}$ (MIPE) plotted over the surface, evaluated on the aortic side of the leaflets. In particular, we see that the neo-Hookean model has large concentrations of strain near the commissure points of each leaflet, while the strain is more evenly distributed with the Lee–Sacks model. This is in agreement with the earlier observations of [23, Figure 5], where a Fung-type constitutive model with similar exponential stiffening properties also led to more even strain distributions,

[9] For a detailed explanation of the mechanism behind this flow rate oscillation, see the electrical circuit analogy used to discuss results in [21, Section 5.4.4].

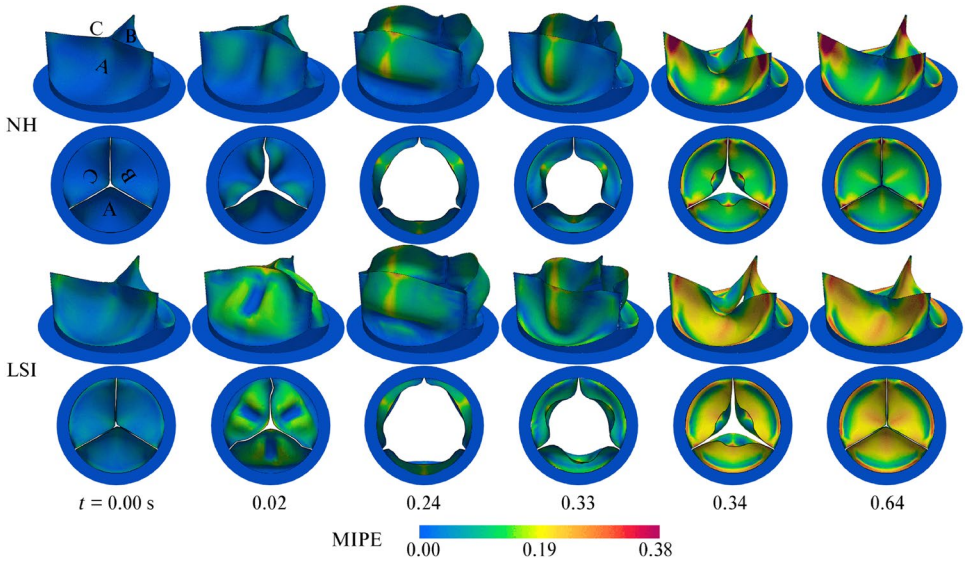


Fig. 12 Selected snapshots comparing MIPE for the different constitutive models

by more strongly penalizing strain concentrations in the energy functional.

7 Conclusion

This paper has introduced a modern, open-source implementation of the immersogeometric FSI analysis techniques of [21, 22], emphasizing the role of code generation technology in ensuring transparency and versatility. In particular, it generalizes the work of [32] by immersing thin structures into deforming fluid domains. In doing so, we showed how code generation can be used to design clear abstractions for referring problems on deforming domains back to static reference configurations. The suitability of this implementation for complex applications was demonstrated using it to test the effects of different material models on FSI-induced strain of prosthetic heart valve leaflets in a deforming artery. In the context of this application, our use of code generation was shown to greatly simplify the process of implementing new material models in FSI analysis. The code for this project will be maintained in the publicly-accessible Git repository [33] and its dependencies [72, 73, 108]. Any discussion of code structure in this document refers to the state of the repository at time of submission,[10] and some results shown in the paper were computed using earlier versions. The comparison between material models in Sect. 6.2 served primarily to demonstrate methodology and software capabilities, rather than to answer a question of scientific interest. However, we believe that the present contribution will lower the human resource cost and increase the reproducibility and transparency of studies like [91, 109], which derive non-trivial physical insights from FSI simulations using DAL-based IMGA.

[10] This repository state can always be recovered using Git, but we expect post-publication changes to improve the software, and recommend against reverting to previous states for most practical purposes.

Acknowledgements G. E. Neighbor and M. Saraeian were partially supported by the Presbyterian Health Foundation Team Science Grant No. C5122401, and M.-C. Hsu was partially supported by the National Heart, Lung, and Blood Institute of the National Institutes of Health under Award No. R01HL142504. H. Zhao was partially supported by National Aeronautics and Space Administration Grant No. 80NSSC21M0070 and D. Kamensky was partially supported by National Science Foundation Grant No. 2103939. This support is gratefully acknowledged. We also thank the Texas Advanced Computing Center (TACC) at the University of Texas at Austin for providing high-performance computing resources that contributed to the results presented in this paper.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

1. Johnson C (2012) Numerical solution of partial differential equations by the finite element method. Dover books on mathematics series. Dover Publications, Sweden
2. Logg A, Mardal K-A, Wells GN et al (2012) Automated solution of differential equations by the finite element method. Springer, Switzerland
3. Alnæs MS, Logg A, Ølgaard KB, Rognes ME, Wells GN (2014) Unified form language: a domain-specific language for weak formulations of partial differential equations. ACM Trans Math Softw 40(2):9–1937
4. Kirby RC, Logg A (2006) A compiler for variational forms. ACM Trans Math Softw 32(3):417–444
5. Logg A, Wells GN (2010) DOLFIN: automated finite element computing. ACM Trans Math Softw 37(2):1–28

6. Evans JA, Kamensky D, Bazilevs Y (2020) Variational multiscale modeling with discretely divergence-free subscales. Comput Math Appl 80(11):2517–2537
7. Calo VM, Ern A, Muga I, Rojas S (2020) An adaptive stabilized conforming finite element method via residual minimization on dual discontinuous Galerkin norms. Comput Methods Appl Mech Eng 363:112891
8. Medina E, Farrell PE, Bertoldi K, Rycroft CH (2020) Navigating the landscape of nonlinear mechanical metamaterials for advanced programmability. Phys Rev B 101:064101
9. Carlson J, Pack A, Transtrum MK, Lee J, Seidman DN, Liarte DB, Sitaraman NS, Senanian A, Kelley MM, Sethna JP, Arias T, Posen S (2021) Analysis of magnetic vortex dissipation in Sn-segregated boundaries in Nb_3Sn superconducting RF cavities. Phys Rev B 103:024516
10. Hoffman J, Jansson J, Johnson C (2016) New theory of flight. J Math Fluid Mech 18(2):219–241
11. Jansson J, Krishnasamy E, Leoni M, Jansson N, Hoffman J (2018). In: López Mejia OD, Escobar Gomez JA (eds) Time-resolved adaptive direct FEM simulation of high-lift aircraft configurations, pp 67–92. Springer, Switzerland
12. Petras A, Leoni M, Guerra JM, Jansson J, Gerardo-Giorda L (2018) Effect of tissue elasticity in cardiac radiofrequency catheter ablation models. 2018 Comput Cardiol Conf (CinC) 45:1–4
13. Richardson CN, Sime N, Wells GN (2019) Scalable computation of thermomechanical turbomachinery problems. Finite Elem Anal Des 155:32–42
14. LeVeque RJ, Mitchell IM, Stodden V (2012) Reproducible research for scientific computing: tools and strategies for changing the culture. Comput Sci Eng 14(4):13–17
15. Ivie P, Thain D (2018) Reproducibility in scientific computing. ACM Comput Surv 51(3)
16. Scott LR (2018) Introduction to automated modeling with FEniCS. Computational Modeling Initiative LLC, Chicago
17. Angoshtari A, Matin AG (2020) Finite element methods in civil and mechanical engineering. CRC Press, Boca Raton
18. Peskin CS (2002) The immersed boundary method. Acta Numer 11:479–517
19. Mittal R, Iaccarino G (2005) Immersed boundary methods. Annu Rev Fluid Mech 37:239–261
20. Chen J-S, Hillman M, Chi S-W (2017) Meshfree methods: progress made after 20 years. J Eng Mech 143(4):04017001
21. Kamensky D, Hsu M-C, Schillinger D, Evans JA, Aggarwal A, Bazilevs Y, Sacks MS, Hughes TJR (2015) An immersogeometric variational framework for fluid-structure interaction: application to bioprosthetic heart valves. Comput Methods Appl Mech Eng 284:1005–1053
22. Hsu M-C, Kamensky D, Bazilevs Y, Sacks MS, Hughes TJR (2014) Fluid-structure interaction analysis of bioprosthetic heart valves: significance of arterial wall deformation. Comput Mech 54:1055–1071
23. Hsu M-C, Kamensky D, Xu F, Kiendl J, Wang C, Wu MCH, Mineroff J, Reali A, Bazilevs Y, Sacks MS (2015) Dynamic and fluid-structure interaction simulations of bioprosthetic heart valves using parametric design with T-splines and Fung-type material models. Comput Mech 55:1211–1225
24. Kamensky D, Hsu M-C, Yu Y, Evans JA, Sacks MS, Hughes TJR (2017) Immersogeometric cardiovascular fluid-structure interaction analysis with divergence-conforming B-splines. Comput Methods Appl Mech Eng 314:408–472
25. Xu F, Morganti S, Zakerzadeh R, Kamensky D, Auricchio F, Reali A, Hughes TJR, Sacks MS, Hsu M-C (2018) A framework for designing patient-specific bioprosthetic heart valves using immersogeometric fluid-structure interaction analysis. Int J Numer Methods Biomed Eng 34(4):2938
26. Borazjani I (2015) A review of fluid-structure interaction simulations of prosthetic heart valves. J Long Term Eff Med Implants 25(1–2):75–93
27. Hirschhorn M, Tchantchaleishvili V, Stevens R, Rossano J, Throckmorton A (2020) Fluid-structure interaction modeling in cardiovascular medicine—a systematic review 2017–2019. Med Eng Phys 78:1–13
28. Abbas SS, Nasif MS, Al-Waked R (2022) State-of-the-art numerical fluid-structure interaction methods for aortic and mitral heart valves simulations: a review. Simulation 98(1):3–34
29. Hughes TJR, Cottrell JA, Bazilevs Y (2005) Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Comput Methods Appl Mech Eng 194:4135–4195
30. Cottrell JA, Hughes TJR, Bazilevs Y (2009) Isogeometric analysis: toward integration of CAD and FEA. Wiley, Chichester
31. Peskin CS (1972) Flow patterns around heart valves: a numerical method. J Comput Phys 10(2):252–271
32. Kamensky D (2021) Open-source immersogeometric analysis of fluid-structure interaction using FEniCS and tIGAr. Comput Math Appl 81:634–648
33. https://github.com/david-kamensky/CouDALFISh: CouDALFISh source code
34. Kamensky D, Bazilevs Y (2019) tIGAr: automating isogeometric analysis with FEniCS. Comput Methods Appl Mech Eng 344:477–498
35. Donea J, Huerta A, Ponthot J-P, Rodriguez-Ferran A (2004) Arbitrary Lagrangian–Eulerian methods. In: Encyclopedia of Computational Mechanics. Volume 3: Fluids. John Wiley & Sons, Hoboken, New Jersey. Chap. 14
36. Bazilevs Y, Calo VM, Hughes TJR, Zhang Y (2008) Isogeometric fluid-structure interaction: theory, algorithms, and computations. Comput Mech 43:3–37
37. Esmaily-Moghadam M, Bazilevs Y, Hsia T-Y, Vignon-Clementel IE, Marsden AL, of Congenital Hearts Alliance (MOCHA), M (2011) A comparison of outlet boundary treatments for prevention of backflow divergence with relevance to blood flow simulations. Comput Mech 48:277–291
38. Bazilevs Y, Calo VM, Cottrel JA, Hughes TJR, Reali A, Scovazzi G (2007) Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. Comput Methods Appl Mech Eng 197:173–201
39. Akkerman I, Bazilevs Y, Calo VM, Hughes TJR, Hulshoff S (2008) The role of continuity in residual-based variational multiscale modeling of turbulence. Comput Mech 41:371–378
40. Bazilevs Y, Michler C, Calo VM, Hughes TJR (2010) Isogeometric variational multiscale modeling of wall-bounded turbulent flows with weakly enforced boundary conditions on unstretched meshes. Comput Methods Appl Mech Eng 199:780–790
41. Takizawa K, Bazilevs Y, Tezduyar TE (2012) Space-time and ALE-VMS techniques for patient-specific cardiovascular fluid-structure interaction modeling. Arch Comput Methods Eng 19:171–225
42. Bazilevs Y, Hsu M-C, Takizawa K, Tezduyar TE (2012) ALE-VMS and ST-VMS methods for computer modeling of wind-turbine rotor aerodynamics and fluid-structure interaction. Math Models Methods Appl Sci 22:1230002
43. Hsu M-C, Akkerman I, Bazilevs Y (2014) Finite element simulation of wind turbine aerodynamics: validation study using NREL Phase VI experiment. Wind Energy
44. Korobenko A, Hsu M-C, Akkerman I, Bazilevs Y (2014) Aerodynamic simulation of vertical-axis wind turbines. J Appl Mech 81:021011
45. Takizawa K, Bazilevs Y, Tezduyar TE, Long CC, Marsden AL, Schjodt K (2014) ST and ALE-VMS methods for patient-specific cardiovascular fluid mechanics modeling. Math Models Methods Appl Sci 24:2437–2486

46. Kiendl J, Hsu M-C, Wu MCH, Reali A (2015) Isogeometric Kirchhoff-Love shell formulations for general hyperelastic materials. Comput Methods Appl Mech Eng 291:280–303
47. Kiendl J, Bletzinger K-U, Linhard J, Wüchner R (2009) Isogeometric shell analysis with Kirchhoff-Love elements. Comput Methods Appl Mech Eng 198:3902–3914
48. Ager C, Schott B, Vuong A-T, Popp A, Wall WA (2019) A consistent approach for fluid-structure-contact interaction based on a porous flow model for rough surface contact. Int J Numer Meth Eng 119(13):1345–1378
49. Kamensky D, Xu F, Lee C-H, Yan J, Bazilevs Y, Hsu M-C (2018) A contact formulation based on a volumetric potential: application to isogeometric simulations of atrioventricular valves. Comput Methods Appl Mech Eng 330:522–546
50. Belytschko T, Neal MO (1991) Contact-impact by the pinball algorithm with penalty and Lagrangian methods. Int J Numer Meth Eng 31(3):547–572
51. Kamensky D, Behzadinasab M, Foster JT, Bazilevs Y (2019) Peridynamic modeling of frictional contact. J Peridyn Nonlocal Model 1(2):107–121
52. Kamensky D, Alaydin MD, Bazilevs Y (2022) A review of nonlocality in computational contact mechanics. In: Aldakheel F, Hudobivnik B, Soleimani M, Wessels H, Weißenfels C, Marino M (eds) Current trends and open problems in computational mechanics. Springer, Cham, pp 239–246
53. Bazilevs Y, Takizawa K, Tezduyar TE (2013) Computational fluid-structure interaction: methods and applications. Wiley, Chichester
54. Tezduyar T, Aliabadi S, Behr M, Johnson A, Mittal S (1993) Parallel finite-element computation of 3D flows. Computer 26(10):27–36
55. Johnson AA, Tezduyar TE (1994) Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces. Comput Methods Appl Mech Eng 119:73–94
56. Stein K, Tezduyar TE, Benney R (2004) Automatic mesh update with the solid-extension mesh moving technique. Comput Methods Appl Mech Eng 193:2019–2032
57. Stein K, Tezduyar T, Benney R (2003) Mesh moving techniques for fluid-structure interactions with large displacements. J Appl Mech 70:58–63
58. Kamensky D, Evans JA, Hsu M-C (2015) Stability and conservation properties of collocated constraints in immersogeometric fluid-thin structure interaction analysis. Commun Comput Phys 18:1147–1180
59. Hsu M-C, Kamensky D (2018) Immersogeometric analysis of bioprosthetic heart valves, using the dynamic augmented Lagrangian method. In: Tezduyar TE (ed) Frontiers in computational fluid-structure interaction and flow simulation. Springer, Cham, pp 167–212
60. van Brummelen EH (2009) Added mass effects of compressible and incompressible flows in fluid-structure interaction. J Appl Mech 76:021206
61. Chung J, Hulbert GM (1993) A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized-α method. J Appl Mech 60:371–75
62. Jansen KE, Whiting CH, Hulbert GM (2000) A generalized-α method for integrating the filtered Navier-Stokes equations with a stabilized finite element method. Comput Methods Appl Mech Eng 190:305–319
63. Kamensky D, Evans JA, Hsu M-C, Bazilevs Y (2017) Projection-based stabilization of interface Lagrange multipliers in immersogeometric fluid-thin structure interaction analysis, with application to heart valve modeling. Comput Math Appl 74(9):2068–2088
64. Yu Y, Kamensky D, Hsu M-C, Lu XY, Bazilevs Y, Hughes TJR (2018) Error estimates for projection-based dynamic augmented Lagrangian boundary condition enforcement, with application to fluid-structure interaction. Math Models Methods Appl Sci 28(12):2457–2509
65. John V, Linke A, Merdon C, Neilan M, Rebholz L (2017) On the divergence constraint in mixed finite element methods for incompressible flows. SIAM Rev 59(3):492–544
66. Casquero H, Bona-Casas C, Gomez H (2017) NURBS-based numerical proxies for red blood cells and circulating tumor cells in microscale blood flow. Comput Methods Appl Mech Eng 316:646–667
67. Boilevin-Kayl L, Fernández MA, Gerbeau J-F (2019) Numerical methods for immersed FSI with thin-walled structures. Comput Fluids 179:744–763
68. Boilevin-Kayl L, Fernández M, Gerbeau J-F (2019) A loosely coupled scheme for fictitious domain approximations of fluid-structure interaction problems with immersed thin-walled structures. SIAM J Sci Comput 41(2):351–374
69. Casquero H, Zhang YJ, Bona-Casas C, Dalcin L, Gomez H (2018) Non-body-fitted fluid-structure interaction: divergence-conforming B-splines, fully-implicit dynamics, and variational formulation. J Comput Phys 374:625–653
70. Casquero H, Bona-Casas C, Toshniwal D, Hughes TJR, Gomez H, Zhang YJ (2021) The divergence-conforming immersed boundary method: application to vesicle and capsule dynamics. J Comput Phys 425:109872
71. Tong GG, Kamensky D, Evans JA (2022) Skeleton-stabilized divergence-conforming B-spline discretizations for incompressible flow problems of high Reynolds number. Comput Fluids 248:105667
72. https://github.com/david-kamensky/ShNAPr: ShNAPr source code
73. https://github.com/david-kamensky/VarMINT: VarMINT source code
74. Tezduyar TE, Sathe S (2007) Modelling of fluid-structure interactions with the space-time finite elements: solution techniques. Int J Numer Meth Fluids 54(6–8):855–900
75. Taylor GI (1923) On the decay of vortices in a viscous fluid. Lond Edinburgh Dublin Philos Magazine J Sci 46(274):671–674
76. Turek S, Hron J (2006) Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow. In: Bungartz H-J, Schäfer M (eds) Fluid-structure interaction. Springer, Berlin, pp 371–385
77. Turek S, Hron J, Razzaq M, Wobker H, Schäfer M (2010) Numerical benchmarking of fluid-structure interaction: a comparison of different discretization and solution approaches. In: Bungartz H-J, Mehl M, Schäfer M (eds) Fluid structure interaction II. Springer, Berlin, pp 413–424
78. Tian F-B, Dai H, Luo H, Doyle JF, Rousseau B (2014) Fluid-structure interaction involving large deformations: 3D simulations and applications to biological systems. J Comput Phys 258:451–469
79. Mehl M, Uekermann B, Bijl H, Blom D, Gatzhammer B, van Zuijlen A (2016) Parallel coupling numerics for partitioned fluid-structure interaction simulations. Comput Math Appl 71(4):869–891
80. Bungartz H-J, Lindner F, Gatzhammer B, Mehl M, Scheufele K, Shukaev A, Uekermann B (2016) Precice—a fully parallel library for multi-physics surface coupling. Comput Fluids 141:250–258
81. Heil M, Hazel AL, Boyle J (2008) Solvers for large-displacement fluid-structure interaction problems: segregated versus monolithic approaches. Comput Mech 43:91–101

82. Breuer M, De Nayer G, Münsch M, Gallinger T, Wüchner R (2012) Fluid-structure interaction using a partitioned semi-implicit predictor-corrector coupling scheme for the application of large-eddy simulation. J Fluids Struct 29:107–130
83. Geuzaine C, Remacle J-F (2009) Gmsh: a 3-D finite element mesh generator with built-in pre- and post-processing facilities. Int J Numer Meth Eng 79(11):1309–1331
84. a MUltifrontal Massively Parallel sparse direct Solver, M.: http://mumps.enseeiht.fr/. Accessed 24 Apr 2016
85. Bazilevs Y, Hsu M-C, Scott MA (2012) Isogeometric fluid-structure interaction analysis with emphasis on non-matching discretizations, and with application to wind turbines. Comput Methods Appl Mech Eng 249–252:28–41
86. Hesch C, Gil AJ, Arranz Carreño A, Bonet J (2012) On continuum immersed strategies for fluid-structure interaction. Comput Methods Appl Mech Eng 247–248:51–64
87. Gil AJ, Carreño AA, Bonet J, Hassan O (2013) An enhanced immersed structural potential method for fluid-structure interaction. J Comput Phys 250:178–205
88. Wick T (2014) Flapping and contact FSI computations with the fluid-solid interface-tracking/interface-capturing technique and mesh adaptivity. Comput Mech 53(1):29–43
89. Wu MCH, Zakerzadeh R, Kamensky D, Kiendl J, Sacks MS, Hsu M-C (2018) An anisotropic constitutive model for immersogeometric fluid-structure interaction analysis of bioprosthetic heart valves. J Biomech 74:23–31
90. Wu MCH, Muchowski HM, Johnson EL, Rajanna MR, Hsu M-C (2019) Immersogeometric fluid-structure interaction modeling and simulation of transcatheter aortic valve replacement. Comput Methods Appl Mech Eng 357:112556
91. Johnson EL, Wu MCH, Xu F, Wiese NM, Rajanna MR, Herrema AJ, Ganapathysubramanian B, Hughes TJR, Sacks MS, Hsu M-C (2020) Thinner biological tissues induce leaflet flutter in aortic heart valve replacements. Proc Natl Acad Sci 117(32):19007–19016
92. Xu F, Johnson EL, Wang C, Jafari A, Yang C-H, Sacks MS, Krishnamurthy A, Hsu M-C (2021) Computational investigation of left ventricular hemodynamics following bioprosthetic aortic and mitral valve replacement. Mech Res Commun 112:103604
93. Bazilevs Y, Hsu M-C, Zhang Y, Wang W, Kvamsdal T, Hentschel S, Isaksen J (2010) Computational fluid-structure interaction: methods and application to cerebral aneurysms. Biomech Model Mechanobiol 9:481–498
94. Arzani A (2018) Accounting for residence-time in blood rheology models: do we really need non-Newtonian blood flow modelling in large arteries? J R Soc Interface 15(146):20180486
95. Hughes TJR, Wells GN (2005) Conservation properties for the Galerkin and stabilised forms of the advection-diffusion and incompressible Navier-Stokes equations. Comput Methods Appl Mech Eng 194(9):1141–1159
96. Bazilevs Y, Gohean JR, Hughes TJR, Moser RD, Zhang Y (2009) Patient-specific isogeometric fluid-structure interaction analysis of thoracic aortic blood flow due to implantation of the Jarvik 2000 left ventricular assist device. Comput Methods Appl Mech Eng 198:3534–3550
97. https://www.rhino3d.com/: Rhinoceros3D software
98. Sacks MS, Zhang W, Wognum S (2016) A novel fibre-ensemble level constitutive model for exogenous cross-linked collagenous tissues. Interface Focus 6:20150090
99. Zhang W, Zakerzadeh R, Zhang W, Sacks MS (2019) A material modeling approach for the effective response of planar soft tissues for efficient computational simulations. J Mech Behav Biomed Mater 89:168–198
100. Zhang W, Motiwale S, Hsu M-C, Sacks MS (2021) Simulating the time evolving geometry, mechanical properties, and fibrous structure of bioprosthetic heart valve leaflets under cyclic loading. J Mech Behav Biomed Mater 123:104745
101. Saad Y, Schultz M (1986) GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J Sci Stat Comput 7:856–869
102. Davis TA (2004) Algorithm 832: UMFPACK v4.3—an unsymmetric-pattern multifrontal method. ACM Trans Math Softw 30(2):196–199
103. Borazjani I (2013) Fluid-structure interaction, immersed boundary-finite element method simulations of bio-prosthetic heart valves. Comput Methods Appl Mech Eng 257:103–116
104. Flamini V, DeAnda A, Griffith BE (2016) Immersed boundary-finite element model of fluid-structure interaction in the aortic root. Theoret Comput Fluid Dyn 30(1):139–164
105. Becsek B, Pietrasanta L, Obrist D (2020) Turbulent systolic flow downstream of a bioprosthetic aortic valve: velocity spectra, wall shear stresses, and turbulent dissipation rates. Front Physiol 11
106. Nitti A, De Cillis G, de Tullio MD (2022) Numerical investigation of turbulent features past different mechanical aortic valves. J Fluid Mech 940:43
107. Thubrikar MJ, Deck JD, Aouad J, Nolan SP (1983) Role of mechanical stress in calcification of aortic bioprosthetic valves. J Thorac Cardiovasc Surg 86(1):115–125
108. https://github.com/david-kamensky/tIGAr: tIGAr source code
109. Johnson EL, Rajanna MR, Yang C-H, Hsu M-C (2022) Effects of membrane and flexural stiffnesses on aortic valve dynamics: identifying the mechanics of leaflet flutter in thinner biological tissues. Forces Mech 6:100053