Implementation of a fluid-solid interaction solver

P. Potì, G. Cavoletti, M. Parimbelli, F. Pizzolato

February 22, 2026

Contents

1 Strong Formulation

The problem models a stationary fluid-structure interaction on the domain $\Omega = \Omega_f \cup \Omega_s$. The fluid flow in Ω_f is governed by the Stokes equations, while the solid deformation in Ω_s is described by linear elasticity. The strong formulation of the coupled problem reads:

$$\begin{cases} -\nu\Delta\mathbf{u} + \nabla p = \mathbf{0} & \text{in } \Omega_f, \\ \nabla \cdot \mathbf{u} = 0 & \text{in } \Omega_f, \\ -\nabla \cdot \sigma(\mathbf{d}) = \mathbf{0} & \text{in } \Omega_s, \end{cases} \tag{1}$$

where the solid stress tensor is given by $\sigma(\mathbf{d}) = \mu(\nabla\mathbf{d} + (\nabla\mathbf{d})^T) + \lambda(\nabla \cdot \mathbf{d})\mathbf{I}$. The system is closed by the following boundary and interface conditions:

$$\begin{cases} \mathbf{u} = \boldsymbol{\varphi} & \text{on } \partial\Omega_{f_1}, \\ \nu\frac{\partial\mathbf{u}}{\partial\mathbf{n}} - p\mathbf{n} = \mathbf{0} & \text{on } \partial\Omega_{f_2}, \\ \mathbf{u} = \mathbf{0} & \text{on } \Sigma, \\ \mathbf{d} = \mathbf{0} & \text{on } \partial\Omega_s \setminus \Sigma, \\ \sigma(\mathbf{d})\mathbf{n} = \nu\frac{\partial\mathbf{u}}{\partial\mathbf{n}} - p\mathbf{n} & \text{on } \Sigma. \end{cases} \tag{2}$$

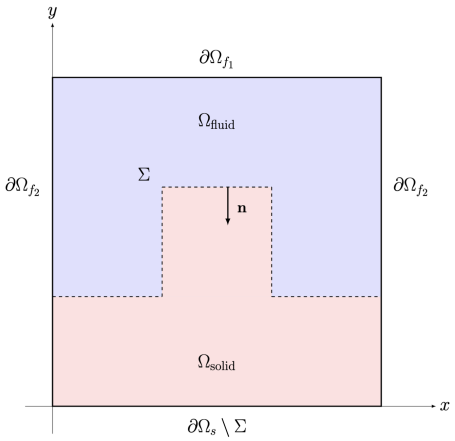


Figure 1: Domain of the problem and partition of its boundary

Here, $\Sigma = \partial\Omega_f \cap \partial\Omega_s$ denotes the fluid-structure interface, $\partial\Omega_{f_1}$ is the Dirichlet inlet boundary, and $\partial\Omega_{f_2}$ represents the free-flow outlet. The normal vector $\mathbf{n}$ is defined on the respective boundaries. Physically, this formulation assumes a steady flow where inertial effects are negligible due to low Reynolds number ($\text{Re} \ll 1$), justifying the omission of the non-linear advection term, and models the solid structure as a linear elastic material undergoing small deformations.

2 Weak formulation

To derive the weak formulation, the governing equations are multiplied by test functions $\mathbf{v} \in \mathbb{V}$, $q \in \mathbb{Q}$, and $\mathbf{b} \in \mathbb{B}$ (defined below) and integrated over their respective domains.

Considering the fluid domain Ω_f first, the momentum equation is integrated as:

$$\int_{\Omega_f} (-\nu \Delta \mathbf{u} + \nabla p) \cdot \mathbf{v} \, d\mathbf{x} = 0.$$

Applying Green's first identity and the divergence theorem yields:

$$\int_{\Omega_f} \nu \nabla \mathbf{u} : \nabla \mathbf{v} \, d\mathbf{x} - \int_{\partial\Omega_f} \nu \frac{\partial \mathbf{u}}{\partial \mathbf{n}} \cdot \mathbf{v} \, d\Gamma - \int_{\Omega_f} p \nabla \cdot \mathbf{v} \, d\mathbf{x} + \int_{\partial\Omega_f} p\mathbf{n} \cdot \mathbf{v} \, d\Gamma = 0.$$

The fluid boundary $\partial\Omega_f$ consists of the interface Σ, the upper boundary $\partial\Omega_{f_1}$, and the outlet $\partial\Omega_{f_2}$. Splitting the boundary integral over these regions, the term on $\partial\Omega_{f_2}$ vanishes due to the free-flow condition. The contributions on $\partial\Omega_{f_1}$ and Σ are zero because the test functions $\mathbf{v}$ satisfy homogeneous Dirichlet conditions ($\mathbf{v} = \mathbf{0}$) on these boundaries. Thus, the velocity test space is defined as:

$$V_0 = \{\mathbf{v} \in [H^1(\Omega_f)]^2 : \mathbf{v}|_\Sigma = \mathbf{0}, \, \mathbf{v}|_{\partial\Omega_{f_1}} = \mathbf{0}\}.$$

For the mass conservation equation, integration against the test function q gives:

$$\int_{\Omega_f} (\nabla \cdot \mathbf{u}) \, q \, d\mathbf{x} = 0,$$

with q belonging to the pressure space $\mathbb{Q} = L^2(\Omega_f)$.

Consequently, the weak formulation for the Stokes problem reads: find $\mathbf{u} \in V = \{\mathbf{v} \in [H^1(\Omega_f)]^2 : \mathbf{v}|_\Sigma = \mathbf{0}, \mathbf{v}|_{\partial\Omega_{f_1}} = \boldsymbol{\varphi}\}$ and $p \in \mathbb{Q}$ such that

$$\begin{cases} a(\mathbf{u}, \mathbf{v}) + b(\mathbf{v}, p) = 0 & \forall \mathbf{v} \in V_0, \\ b(\mathbf{u}, q) = 0 & \forall q \in \mathbb{Q}, \end{cases}$$

where the bilinear forms are defined as:

$$\begin{aligned} a(\mathbf{u}, \mathbf{v}) &= \int_{\Omega_f} \nu \nabla \mathbf{u} : \nabla \mathbf{v} \, d\mathbf{x}, \\ b(\mathbf{v}, p) &= -\int_{\Omega_f} p(\nabla \cdot \mathbf{v}) \, d\mathbf{x}. \end{aligned} \tag{3}$$

Since the trial space V (imposing $\mathbf{u} = \boldsymbol{\varphi}$ on $\partial\Omega_{f_1}$) and the test space V_0 (imposing $\mathbf{v} = \mathbf{0}$) do not coincide, a lifting of the boundary data is applied. Decomposing the solution as $\mathbf{u} = \mathbf{u}_0 + R_\varphi$, where $\mathbf{u}_0 \in V_0$ and $R_\varphi \in [H^1(\Omega_f)]^2$ is an arbitrary lifting function satisfying $R_\varphi|_{\partial\Omega_{f_1}} = \boldsymbol{\varphi}$ and $R_\varphi|_\Sigma = \mathbf{0}$, the problem is transformed. The weak formulation becomes: find $\mathbf{u}_0 \in V_0$ and $p \in \mathbb{Q}$ such that

$$\begin{cases} a(\mathbf{u}_0, \mathbf{v}) + b(\mathbf{v}, p) = F(\mathbf{v}) & \forall \mathbf{v} \in V_0, \\ b(\mathbf{u}_0, q) = G(q) & \forall q \in \mathbb{Q}, \end{cases}$$

where the source functionals are given by:

$$F(\mathbf{v}) = -a(R_\varphi, \mathbf{v}), \quad G(q) = -b(R_\varphi, q).$$

A similar procedure is applied to the elasticity problem. The momentum equation is tested against $\mathbf{b} \in \mathbb{B}$:

$$\int_{\Omega_s} (\nabla \cdot \sigma(\mathbf{d})) \cdot \mathbf{b}\, d\mathbf{x} = 0.$$

Applying Green's first identity for tensors yields:

$$\int_{\Omega_s} \sigma(\mathbf{d}) : \nabla \mathbf{b}\, d\mathbf{x} - \int_{\partial\Omega_s} (\sigma(\mathbf{d})\mathbf{n}) \cdot \mathbf{b}\, d\Gamma = 0.$$

Substituting the interface condition $\sigma(\mathbf{d})\mathbf{n} = \nu \frac{\partial \mathbf{u}}{\partial \mathbf{n}} - p\mathbf{n}$ on Σ, and observing that contributions on the remaining boundaries vanish due to Dirichlet constraints on $\mathbf{b}$, the equation becomes:

$$\int_{\Omega_s} \sigma(\mathbf{d}) : \nabla \mathbf{b}\, d\mathbf{x} - \int_{\Sigma} (\nu \frac{\partial \mathbf{u}}{\partial \mathbf{n}} - p\mathbf{n}) \cdot \mathbf{b}\, d\Gamma = 0.$$

Using the constitutive law $\sigma(\mathbf{d}) = \mu(\nabla \mathbf{d} + (\nabla \mathbf{d})^T) + \lambda(\nabla \cdot \mathbf{d})\mathbf{I}$, the weak formulation for the solid is: find $\mathbf{d} \in \mathbb{B} = [H^1_{\Gamma_s}(\Omega_s)]^2$ such that

$$c(\mathbf{d}, \mathbf{b}) + m(p, \mathbf{b}) + n(\mathbf{u}, \mathbf{b}) = 0 \quad \forall \mathbf{b} \in \mathbb{B},$$

The forms are defined as:

$$\begin{aligned} c(\mathbf{d}, \mathbf{b}) &= \int_{\Omega_s} \left[\mu(\nabla \mathbf{d} + (\nabla \mathbf{d})^T) : \nabla \mathbf{b} + \lambda(\nabla \cdot \mathbf{d})(\nabla \cdot \mathbf{b}) \right] d\mathbf{x}, \\ m(p, \mathbf{b}) &= -\int_{\Sigma} (p\mathbf{n}) \cdot \mathbf{b}\, d\Gamma, \\ n(\mathbf{u}, \mathbf{b}) &= \int_{\Sigma} \nu \frac{\partial \mathbf{u}}{\partial \mathbf{n}} \cdot \mathbf{b}\, d\Gamma. \end{aligned} \tag{4}$$

In this formulation, the variables $\mathbf{u}_0, p$ are defined on Ω_f and $\mathbf{d}$ on Ω_s. To construct a unified system, these fields are extended by zero to the entire domain $\Omega = \Omega_f \cup \Omega_s$. The global function spaces are defined as:

$$\begin{aligned} \mathbf{u}_0 \in V &= \{\mathbf{v} \in [L^2(\Omega)]^2 : \mathbf{v}|_{\Omega_f} \in V_0, \quad \mathbf{v}|_{\Omega_s} = \mathbf{0}\}, \\ p \in Q &= \{q \in L^2(\Omega) : \quad q|_{\Omega_f} \in \mathbb{Q}, \quad q|_{\Omega_s} = 0\}, \\ \mathbf{d} \in B &= \{\mathbf{w} \in [L^2(\Omega)]^2 : \mathbf{w}|_{\Omega_s} \in \mathbb{B}, \quad \mathbf{w}|_{\Omega_f} = \mathbf{0}\}. \end{aligned}$$

Let $Z = \{\psi : \psi(x) = 0\}$ represent the zero space. Summing the weak forms yields: find $y = \{\mathbf{u}_0, p, \mathbf{d}\}$ in the product space

$$Y = \{y = \{\mathbf{u}_0, p, \mathbf{d}\} : y|_{\Omega_f} \in V \times Q \times Z^2, \quad y|_{\Omega_s} \in Z^2 \times Z \times B\}$$

such that

$$a(\mathbf{u}_0, \mathbf{v}) + b(\mathbf{v}, p) + b(\mathbf{u}_0, q) + c(\mathbf{d}, \mathbf{b}) + m(p, \mathbf{b}) + n(\mathbf{u}_0, \mathbf{b}) = F(\mathbf{v}) + G(q) \tag{5}$$

holds for all test functions $z = \{\mathbf{v}, q, \mathbf{b}\} \in Y$.

3 Discretization

The problem is discretized using finite dimensional subspaces V_h, B_h, Q_h approximating V_0, B, Q. Considering a triangulation $\mathcal{T}_h$ of the domain Ω, we introduce the finite element space:

$$Q_r = \{v_h \in C^0(\Omega) : v_h|_K \in \mathbb{P}_r \, \forall K \in \mathcal{T}_h\}, \quad r = 1, 2, \dots \tag{6}$$

which consists of globally continuous functions that are polynomials of degree r on each element $K \in \mathcal{T}_h$. The specific subspaces are defined as:

$$V_h = \left\{ \mathbf{v}_h \in V : \forall K \in \mathcal{T}_h, \mathbf{v}_h|_K \in \begin{cases} Q^d_{p+1} & \text{if } K \subset \Omega_f \\ 0 & \text{if } K \subset \Omega_s \end{cases} \text{ and } \mathbf{v}_h|_{\Omega_f} \in C^0 \right\}$$

$$Q_h = \left\{ q_h \in Q : \forall K \in \mathcal{T}_h, q_h|_K \in \begin{cases} Q_p & \text{if } K \subset \Omega_f \\ 0 & \text{if } K \subset \Omega_s \end{cases} \text{ and } q_h|_{\Omega_f} \in C^0 \right\}$$

$$B_h = \left\{ \mathbf{b}_h \in B : \forall K \in \mathcal{T}_h, \mathbf{b}_h|_K \in \begin{cases} Q^d_r & \text{if } K \subset \Omega_s \\ Z^d & \text{if } K \subset \Omega_f \end{cases} \text{ and } \mathbf{b}_h|_{\Omega_s} \in C^0 \right\}$$

To ensure the well-posedness of the discrete Stokes problem and satisfy the LBB (inf-sup) condition, we employ the Taylor-Hood element pair[5]. This choice approximates the velocity field using polynomials of degree $p+1$ (typically quadratic, $p = 1$) and the pressure using polynomials of degree p (linear), preventing spurious pressure modes. We can now choose a set of basis functions for each of the subspaces. The discrete solution is expanded using basis functions constructed from the standard scalar Lagrangian basis $\{\varphi_j\}$. For the vector-valued velocity and displacement fields, it is convenient to define vector basis functions component-wise. The velocity expansion is given by:

$$\mathbf{u}_h(\mathbf{x}) = \sum_{i=1}^{N_h^u} u_i \boldsymbol{\varphi}_i(\mathbf{x}) \,,$$

where N_h^u denotes the dimension of the velocity space V_h, the coefficients u_i are set to 0 in Ω_s (the interface Σ is included because of the boundary conditions), $\boldsymbol{\varphi}_i$ are the vector-valued basis functions, starting from the Lagrangian scalar basis functions φ_j (with $j = 1, 2, \dots, N_h^u/2$), which are defined as:

$$\boldsymbol{\varphi}_1 = \begin{bmatrix} \varphi_1 \\ 0 \end{bmatrix}, \quad \boldsymbol{\varphi}_2 = \begin{bmatrix} 0 \\ \varphi_1 \end{bmatrix}, \quad \boldsymbol{\varphi}_3 = \begin{bmatrix} \varphi_2 \\ 0 \end{bmatrix}, \quad \boldsymbol{\varphi}_4 = \begin{bmatrix} 0 \\ \varphi_2 \end{bmatrix}, \quad \dots$$

The same can be said for $\mathbf{d_h}$ which can be written as

$$\mathbf{d}_h(\mathbf{x}) = \sum_{i=1}^{N_h^d} d_i \boldsymbol{\psi}_i(\mathbf{x}) \,,$$

where N_h^d denotes the dimension of the discretized space, the coefficients d_i are set to 0 in $\Omega_f \setminus \Sigma$ (this means that the displacement DoFs on the interface are

free), and $\boldsymbol{\psi}_i$ are another set of vector-valued Lagrangian basis functions. The pressure is a scalar function, which means it is sufficient to write it as:

$$p_h(\mathbf{x}) = \sum_{i=1}^{N_h^p} p_i \varepsilon_i(\mathbf{x}),$$

with p_i set to 0 in $\Omega_s \setminus \Sigma$, N_h^p symbolizing the dimension of the subspace and ε_i denoting the standard Lagrangian basis functions. This leads to the discrete weak formulation, which reads: find $y_h = \{\mathbf{u}_h, p_h, \mathbf{d_h}\}$ in the space

$$Y_h = \{y_h = \{\mathbf{u}_{0h}, p_h, \mathbf{d}_h\} : y_h|_{\Omega_f} \in V_h \times Q_h \times Z_h^2, \quad y_h|_{\Omega_s} \in Z_h^2 \times Z_h \times B_h\}$$

such that

$$a(\mathbf{u}_h, \mathbf{v_h}) + b(\mathbf{v_h}, p_h) + b(\mathbf{u}_h, q_h) + c(\mathbf{d_h}, \mathbf{b_h}) + m(p_h, \mathbf{b_h}) + n(\mathbf{u}_h, \mathbf{b}_h) = F(\mathbf{v_h}) + G(q_h)$$

holds for all test functions $z_h = \{\mathbf{v_h}, q_h, \mathbf{b_h}\} \in Y_h$. This can be rewritten as an algebraic system of the form

$$\begin{bmatrix} A & B^T & 0 \\ B & 0 & 0 \\ N & M & C \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ \mathbf{P} \\ \mathbf{D} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{G} \\ \mathbf{0} \end{bmatrix}, \tag{7}$$

where $\mathbf{U}, \mathbf{P}, \mathbf{D}$ are the vectors of unknowns. Denoting by $\{\boldsymbol{\varphi}_j\}$ the basis for velocity, $\{\varepsilon_j\}$ for pressure, and $\{\boldsymbol{\psi}_j\}$ for displacement, the matrix entries are defined as:

$$\begin{aligned} A_{ij} &= a(\boldsymbol{\varphi}_j, \boldsymbol{\varphi}_i), & B_{ij} &= b(\boldsymbol{\varphi}_j, \varepsilon_i), & & \\ C_{ij} &= c(\boldsymbol{\psi}_j, \boldsymbol{\psi}_i), & M_{ij} &= m(\varepsilon_j, \boldsymbol{\psi}_i), & N_{ij} &= n(\boldsymbol{\varphi}_j, \boldsymbol{\psi}_i), \\ F_i &= F(\boldsymbol{\varphi}_i), & G_i &= G(\varepsilon_i). & & \end{aligned}$$

4 Computational implementation

4.1 Code Structure

Once obtained our discrete representation, we proceeded as usual to create a class called FSI, which will allow us to solve the problem and get a visual representation of the solution. The simulation proceeds through a standard pipeline executed in the `run()` method:

1. **Setup:** Initializes the `DoFHandler`, and allocates memory for block-sparse matrices and vectors.

2. **Assembly:** Uses `FEValues` to compute local cell contributions. A specialized method, `assemble_interface_term`, enforces the coupling conditions at the fluid-solid interface.

3. **Solving:** The system is solved using `SolverGMRES`, and a specialized preconditioner which will be detailed later on.

4. **Refinement:** Adaptive mesh refinement is performed based on error estimators to optimize computational resources.

This process is repeated multiple times, to achieve a more accurate solution. The simulation uses the `hp` finite element framework in `deal.II`[1] to manage the multi-physics coupling. Although the method does not apply p-adaptivity (varying polynomial degrees across cells), these classes are necessary to assign distinct finite element systems to the fluid and solid subdomains within a single `DoFHandler`.

4.2 Preconditioning

Since we have reordered degrees of freedom such that we first have all velocity degrees of freedom, then pressure, and then all displacements, the system matrix can be written as:

$$A_{\text{global}} = \begin{pmatrix} A_{\text{fluid}} & 0 \\ R & A_{\text{solid}} \end{pmatrix}$$

where A_{fluid} is the Stokes matrix for velocity and pressure, A_{solid} results from the elasticity equations for the displacements, and R is the matrix that comes from the interface conditions. The inverse of such a block-triangular matrix is given by:

$$A_{\text{global}}^{-1} = \begin{pmatrix} A_{\text{fluid}}^{-1} & 0 \\ -A_{\text{solid}}^{-1} R A_{\text{fluid}}^{-1} & A_{\text{solid}}^{-1} \end{pmatrix}$$

Applying this matrix requires one solve with A_{fluid} and A_{solid} each, since:

$$\begin{pmatrix} p_x \\ p_y \end{pmatrix} = \begin{pmatrix} A_{\text{fluid}}^{-1} & 0 \\ -A_{\text{solid}}^{-1} R A_{\text{fluid}}^{-1} & A_{\text{solid}}^{-1} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

can be computed by solving the following linear systems sequentially:

$$A_{\text{fluid}} p_x = x \qquad A_{\text{solid}} p_y = y - R p_x.$$

We can therefore expect that

$$\widetilde{A_{\text{global}}^{-1}} = \begin{pmatrix} \widetilde{A_{\text{fluid}}^{-1}} & 0 \\ \widetilde{-A_{\text{solid}}^{-1} R A_{\text{fluid}}^{-1}} & \widetilde{A_{\text{solid}}^{-1}} \end{pmatrix}$$

would be a good preconditioner if $\widetilde{A_{\text{fluid}}^{-1}} \approx A_{\text{fluid}}^{-1}$ and $\widetilde{A_{\text{solid}}^{-1}} \approx A_{\text{solid}}^{-1}$.

To implement the solver, it is necessary to split the fluid block A_{fluid} into its velocity stiffness block A_{uu} and pressure coupling blocks A_{up} and A_{pu}. Because the Stokes equations lack a direct pressure-pressure term, the global matrix is represented in a 3×3 block form:

$$A_{\text{global}} = \begin{pmatrix} A_{uu} & A_{up} & 0 \\ A_{pu} & 0 & 0 \\ R_{du} & R_{dp} & A_{dd} \end{pmatrix}$$

where the interface matrix R has been decomposed into R_{du} and R_{dp}, representing the coupling of fluid velocity and fluid pressure to the solid displacements, respectively.

The fluid solve is approximated through the preconditioner:

$$P = \begin{bmatrix} A_{uu} & 0 \\ A_{pu} & \frac{1}{\nu} M_p \end{bmatrix}$$

where M_p is the pressure mass matrix with entries

$$(M_p)_{ij} = \int_\Omega \psi_i \psi_j \, d\mathbf{x}.$$

The action of P^{-1} on a vector $\mathbf{x} = [\mathbf{x}_u, \mathbf{x}_p]^T$ is given by:

$$P^{-1}\mathbf{x} = \begin{bmatrix} A_{uu}^{-1} & 0 \\ -\nu M_p^{-1} A_{pu} A_{uu}^{-1} & \nu M_p^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{x}_u \\ \mathbf{x}_p \end{bmatrix} = \begin{bmatrix} A_{uu}^{-1}\mathbf{x}_u \\ -\nu M_p^{-1} A_{pu} A_{uu}^{-1}\mathbf{x}_u + \nu M_p^{-1}\mathbf{x}_p \end{bmatrix}.$$

In practice, applying our preconditioner proceeds in three steps. First, we solve

$$A_{uu}\, \tilde{u} = x_u$$

using an algebraic multigrid (AMG) preconditioner. Second, we form the pressure residual

$$r_p = x_p - A_{pu}\tilde{u}$$

and compute the pressure update

$$\tilde{p} = \nu M_p^{-1} r_p.$$

Finally, the solid displacement update is computed by solving

$$A_{dd}\, \tilde{d} = \hat{r}_d, \qquad \hat{r}_d = x_d - \left(R_{du}\tilde{u} + R_{dp}\tilde{p}\right),$$

where $\hat{r}_d$ incorporates the interface coupling from the previously computed fluid updates $\tilde{u}$ and $\tilde{p}$. This last solve, just like the previous ones, is approximated by a single V-cycle of an algebraic multigrid method applied to A_{dd}.

4.3 Adaptive Mesh Refinement

After computing the solution, our code proceeds to call the mesh refinement method. Mesh refinement serves as an adaptive strategy to optimize computational resources by increasing resolution specifically in regions requiring higher precision.

In this context, the process uses the class `KellyErrorEstimator`, an implementation of the error indicator by Kelly, De S. R. Gago, Zienkiewicz and Babuska [3]. Let

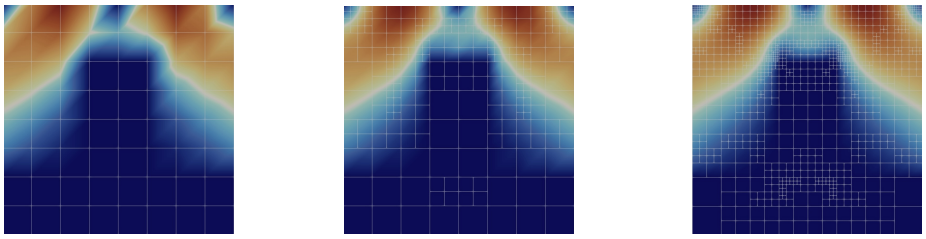
$$\eta_K^2 = \sum_{F \in \partial K} c_F \int_{\partial K_F} \left[\left[a \frac{\partial u_h}{\partial n} \right] \right]^2 \tag{8}$$

be the error estimator for cell K. $[[\cdot]]$ is the notation used to denote jump of the function in square brackets at the face, and c_F is a factor related to the geometry of the face. The overall error estimate is then computed as

$$\eta^2 = \sum_K \eta_K^2. \tag{9}$$

In our implementation, the workflow begins with the calculation of independent error indicators for the Stokes and elasticity components, facilitated by `ComponentMask` objects that isolate the relevant degrees of freedom for each physical model. Once calculated, these indicators are merged using a heuristic weight to ensure that the refinement remains balanced and does not become disproportionately concentrated within the elastic region[2]. A critical aspect of this implementation involves managing the fluid-structure interface, where the `KellyErrorEstimator` objects often identify the inherent physical discontinuity as a massive numerical error. To prevent irrationally large refinement spikes that would otherwise occur at this boundary, the error indicators for cells adjacent to the interface are manually reset to zero, shifting the focus of refinement toward the internal physics of each respective subdomain. The entire solution and adaptation process is repeated over multiple iterations, progressively improving the solution accuracy with each refinement pass.

4.4 First tests

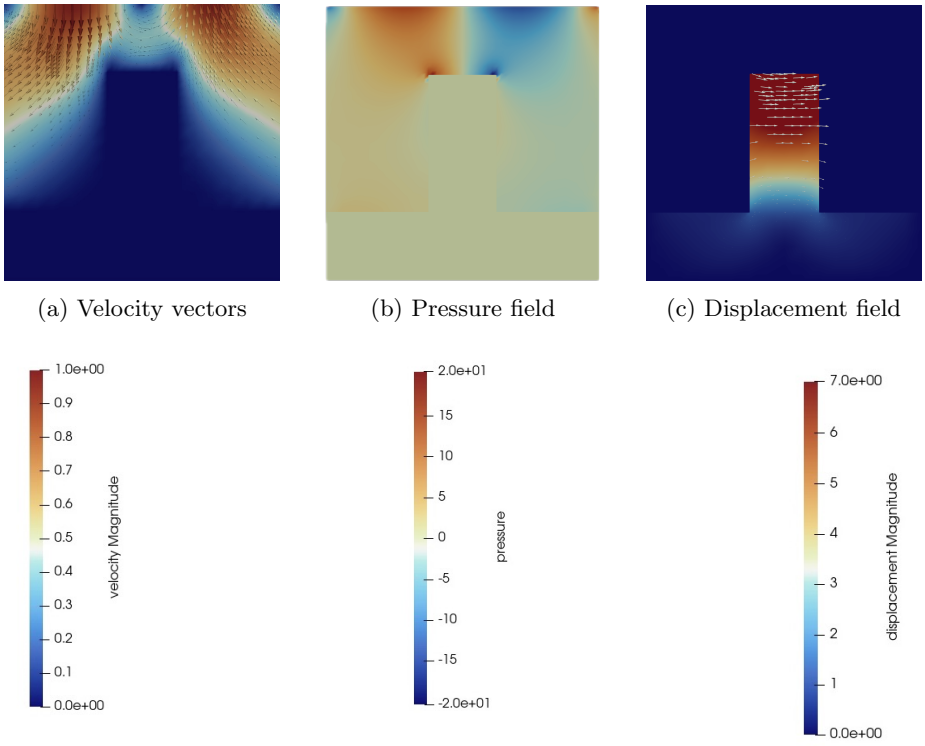


(a) Initial Mesh (b) Intermediate Step (c) Final Mesh

Figure 2: Evolution of the adaptive mesh refinement. The initial uniform grid (left) is refined based on velocity error indicators. By the final cycle (right), the mesh density is concentrated near the fluid-solid interface Σ and high-gradient regions.

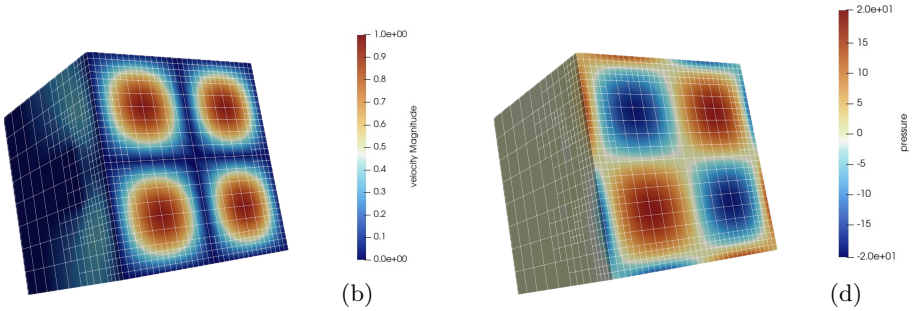
We chose to run a first test of our code in a two-dimensional setting. The fluid sub-domain is characterized by a kinematic viscosity $\nu = 2.0$, while the solid sub-domain uses Lamé parameters $\mu = 1.0Pa$ and $\lambda = 10.0Pa$. Boundary conditions include a sinusoidal inlet velocity defined as $v_{in} = \sin(\pi x)$ on Ω_{f_1} and zero-displacement constraints on $\partial\Omega_s \setminus \Sigma$. Numerical integration is performed using Gauss quadrature rules of order $p+2$, where p corresponds to the respective polynomial degrees of the variables. The simulation follows an iterative

refinement strategy consisting of $10 - 2 \times dim$ cycles. In each cycle, the mesh is refined based on a weighted Kelly error estimator (weights of 4.0 for Stokes and 1.0 for elasticity) targeting the top 30% of cells with the highest error. The resulting linear system is solved as detailed above and the final mesh is shown in Fig 2, while Fig 3 shows in detail the velocity, pressure and displacement fields defined on the two different subdomains. In the second test we carried out, we kept the parameters unchanged but moved to a 3D setting where we used as inlet velocity $v_{in} = \sin(\pi x)\sin(\pi y)$. Our codes allowed us to switch easily and the results are shown in Fig. 4.



(a) Velocity vectors (b) Pressure field (c) Displacement field

Figure 3: Simulation results. Top row: velocity, pressure, and displacement fields. Bottom row: corresponding value scales.



(a) 3D velocity Cube at cycle 2

(b) Velocity scale

(c) 3D pressure Cube at cycle 2

(d) Pressure scale

Figure 4: Pressure and velocity fields in the 3D domain

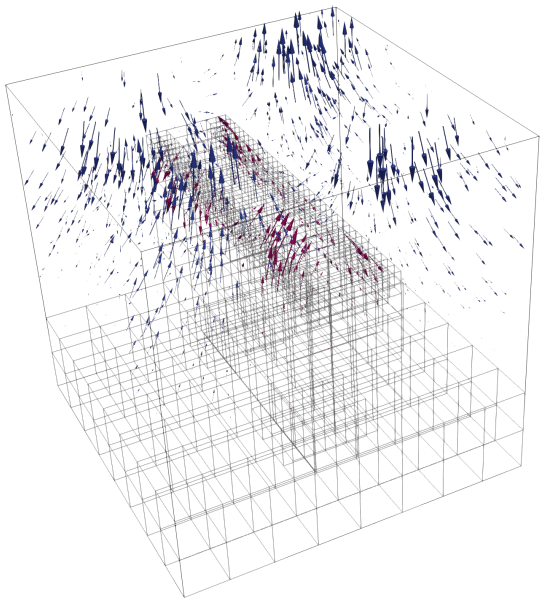
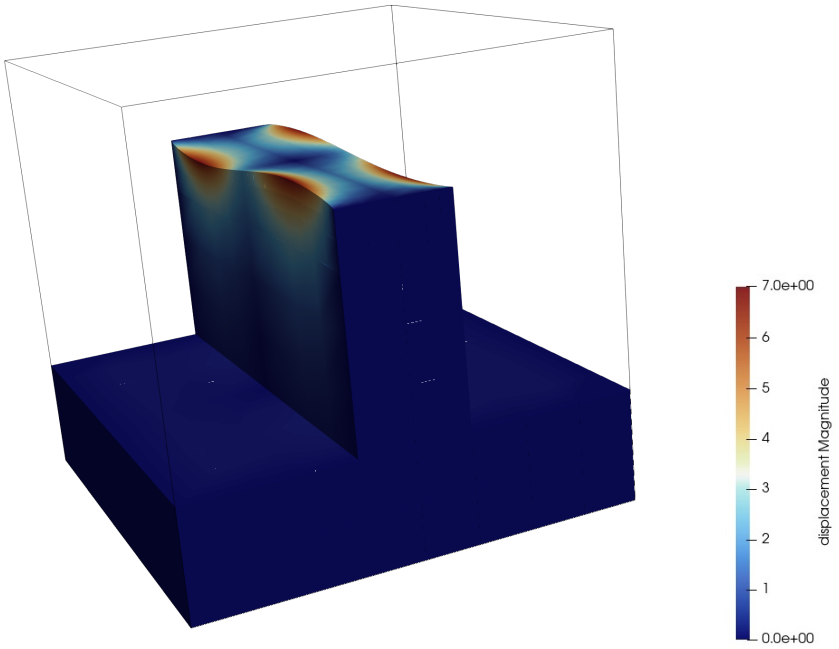


Figure 5: Vector plot for velocity (in blue) and for solid displacement (in red).



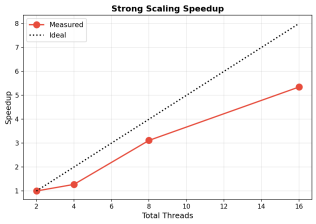
(a) The solid domain was warped by the displacement solution

5 Parallelism and scalability of the code

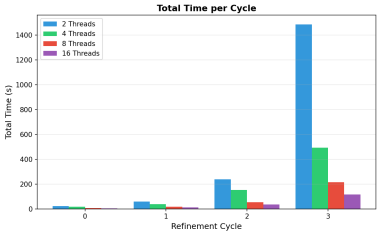
Following the initial sequential validation of the solver, a parallel implementation was developed to significantly reduce computational times and tackle higher-resolution meshes. The parallelization strategy relies on a hybrid distributed-shared memory paradigm, combining MPI for inter-process communication and OpenMP for intra-node multi-threading. To investigate the code's scalability properties, a series of performance tests were conducted on the MOX-HPC cluster [4], keeping all of the conditions described in Sec.4.4 fixed. Specifically, the simulations were executed on computing nodes equipped with dual Intel Xeon Gold 6238R processors (2.20 GHz), featuring a total of 56 physical cores (112 hardware threads via Hyper-Threading) and 512 GB of RAM per node. Due to the queue scheduling policies of the cluster, each job was allocated a maximum of 28 hardware threads. To optimize the workload distribution within these limits, the hybrid execution was balanced by pinning 2 OpenMP threads to each MPI process.

5.1 Strong Scalability

To evaluate the strong scalability of the implemented solver, we tested the computational performance by keeping the global problem size fixed while gradually increasing the number of physical cores used.



(a) The reference problem consisted of an initial $8 \times 8 \times 8$ cubic mesh with no refinement cycles.



(b) The reference problem consisted of an initial $8 \times 8 \times 8$ cubic mesh undergoing 3 adaptive refinement cycles.

Figure 7: Strong scaling results

Figure 7b highlights two main trends: the expected growth in execution time with each adaptive refinement cycle and, crucially, the strong scalability of the solver. Within the same cycle, doubling the computing resources results in a proportional halving of the execution time

Figure 7a demonstrates a nearly linear speedup trend, confirming that execution time decreases inversely with the number of threads for a fixed workload. It is worth noting that this plot specifically isolates the execution times on the unrefined baseline mesh. This choice was made to evaluate the pure parallel efficiency of the solver, avoiding the superlinear speedup artifacts observed during adaptive refinement cycles.

5.2 Weak Scalability

To test the weak scalability of our implementation, we evaluated the solver's performance by increasing the global problem size proportionally to the number of allocated threads, aiming to maintain a constant computational load per process. The baseline simulation, executed on the lowest thread count, uses an initial $8 \times 8 \times 8$ cubic mesh. To effectively double the total number of hexahedral elements alongside a doubling of the computational resources, the number of subdivisions n along each spatial dimension was scaled by a factor of $\sqrt[3]{2}$. It is important to note, however, that in finite element discretizations, the global number of Degrees of Freedom (DoFs) does not grow perfectly linearly with the number of grid elements due to the sharing of nodes, edges, and faces between adjacent cells. Consequently, the actual computational workload per thread is not strictly constant, and the resulting performance data must be interpreted accounting for this intrinsic, slight non-linear increase in the algebraic system size.

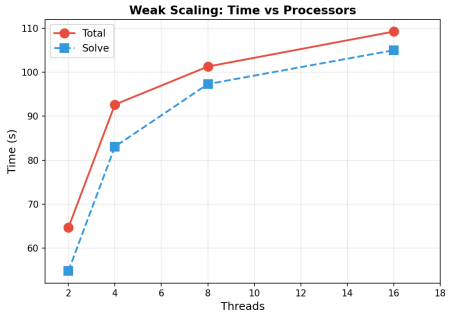


Figure 8: Weak Scaling results

Figure 8 plots the execution time against the number of threads for the increasing-size problem. While an ideal weak scaling test assumes a perfectly constant execution time, our results show a significantly lower time for the smallest problem sizes before the curve stabilizes. This initial disparity is natural: the baseline configurations benefit from a lower number of DoFs per element and near-zero MPI communication overhead. As the scale increases, this moderate, sub-linear growth eventually flattens, confirming the solid weak scalability of our implementation.

5.3 Effect of the preconditioner

We compared the performance of the GMRES solver equipped with our preconditioner against a direct solver using `SparseDirectUMFPACK` across four cycles of 3D adaptive mesh refinement, reaching approximately 5.4×10^5 degrees of freedom on the finest mesh.

Cycle	GMRES iters	Solve (prec) [s]	Solve (UMFPACK) [s]
0	48	20.9	4.1
1	51	46.7	19.7
2	51	196.3	96.7
3	73	853.1	1198.2

Table 1: Per-cycle comparison of the preconditioned GMRES solver against the UMFPACK direct solver.

Table 1 shows the scaling difference between the two algorithms. While the direct solver is faster for coarser meshes (Cycles 0 to 2), its execution time grows non-linearly as the system expands, a known limitation of direct solvers in 3D problems due to matrix fill-in.

In contrast, our preconditioned GMRES demonstrates solid algorithmic scalability. The preconditioner effectively controls the condition number of the system, keeping the iteration count stable across the first three cycles, with only a moderate increase at the finest level, expected behavior as the system becomes

increasingly harder to precondition at higher resolutions. Ultimately, this test proves that while direct solvers are suitable for small-scale testing, the preconditioned iterative method is strictly necessary for high-resolution 3D simulations, driving an overall 13% reduction in total execution time and a 1.18× speedup in the core solve phase.

References

[1] D. Arndt, W. Bangerth, M. Feder, M. Fehling, R. Gassmöller, T. Heister, L. Heltai, M. Kronbichler, M. Maier, P. Munch, J.-P. Pelteret, S. Sticko, B. Turcksin, and D. Wells. The `deal.II` library, version 9.4. *Journal of Numerical Mathematics*, 30(3):231–246, 2022.

[2] W. Bangerth, T. Heister, and G. Kanschat. `deal.II` step-46: Fluid-structure interaction. `deal.II` Tutorial Programs, `https://www.dealii.org/current/doxygen/deal.II/step_46.html`.

[3] D. W. Kelly, J. P. De S. R. Gago, O. C. Zienkiewicz, and I. Babuška. A posteriori error analysis and adaptive processes in the finite element method: Part I – error analysis. *International Journal for Numerical Methods in Engineering*, 19(11):1593–1619, 1983.

[4] MOX – Dipartimento di Matematica, Politecnico di Milano. HPC@MOX – high performance computing facilities. `https://mox.polimi.it/research-areas/hpcmox/`, 2024.

[5] C. Taylor and P. Hood. A numerical solution of the Navier-Stokes equations using the finite element technique. *Computers & Fluids*, 1(1):73–100, 1973.