Flow in a driven cavity and non conforming mesh coupling

Numerics for Fluids, Structures and Electromagnetics, 2022-2023

Student Name	Student Number
Mossinelli Giacomo	359961
Timpano Daniele	352898

1 Introduction

1.1 Abstract

In the context of the course Numerics for fluids, structures and electromagnetics (MATH-468), the students Giacomo Mossinelli and Daniele Timpano tackled a benchmark problem for the solution of Navier-Stokes equations: the lid-cavity problem. The present report focuses on the main results of the students' work, as a starting point: the proof of the **compatibility condition**, the derivation of a **weak formulation** for the driven cavity problem and the choice of appropriate **functional spaces** and **finite element spaces** for velocity and pressure. Following this preliminary discussion, the problem was solved numerically using `FenicsX` implementing a locally refined mesh and investigating the use of Mortar method with two subdomains.

1.2 Context

Navier-Stokes partial differential equations describe the movement of a viscous fluid. The equations are derived applying continuity of mass, momentum and energy on a so called *control volume*, whose position can remain fixed or move with fluid, depending on the choice of an Eulerian or Lagrangian approach. Specifically, supposing a generic domain $\Omega \subset \mathbb{R}^d$:

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} - \operatorname{div}\left[\nu\left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right)\right] + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{f}, & \mathbf{x} \in \Omega, t > 0, \\ \operatorname{div} \mathbf{u} = 0, & \mathbf{x} \in \Omega, t > 0 \end{cases}$$

The first equation represents **the conservation of momentum**, while the second represents the conservation of mass, which is called **the continuity equation**. The term $\operatorname{div}\left[\nu\left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right)\right]$ (which is equal to $\nu\Delta\mathbf{u}$ using the continuity equation) describes molecular **diffusion**, while $(\mathbf{u}\cdot\nabla)\mathbf{u}$ represents convective **transport**. These equations are generally called incompressible Navier Stokes Equations, and fluids satisfying incompressibility condition $\operatorname{div}\mathbf{u} = 0$ are called **incompressible** [12].

Considering **steady state flow** of incompressible fluids, neglecting the convective transport term in the hypothesis Re $<<$ 1 and focusing on a two-dimensional domain, we are left with the problem object of this study. Let Ω be $\subset \mathbb{R}^2$, let's consider velocity $u : \Omega \to \mathbb{R}^2$ and pressure $p : \Omega \to \mathbb{R}$:

$$\begin{aligned} -\Delta\mathbf{u} + \nabla p &= \mathbf{f} \quad \text{in } \Omega \\ \operatorname{div}\mathbf{u} &= 0 \quad \text{in } \Omega, \\ \mathbf{u} &= \mathbf{g} \quad \text{on } \partial\Omega \end{aligned}$$

where $f : \Omega \to \mathbb{R}^2$ and $g : \Omega \to \mathbb{R}^2$ are two given functions. In particular, we assumed that $\mathbf{f} \in [H^{-1}(\Omega)]^2 = \mathbf{H}^{-1}(\Omega)$ and $\mathbf{g} \in \left[H^{1/2}(\partial\Omega)\right]^2 = \mathbf{H}^{1/2}(\partial\Omega)$.

1.3 Goals of numerical solution

In order to implement the problem for the geometry shown in Figure 1, we will set $f = 0$ and consider the following boundary conditions, typical of the lid driven cavity problem: square domain with **Dirichlet boundary conditions on all sides**, with **three stationary sides** and one **moving side** (with velocity tangent to the side). Please note that existing literature refers to similar problems where the moving lid is replaced by another fluid moving (the problem is in this case called a shear cavity problem) [8].

Difficulties may arise in solving this problem with Finite Element Method (FEM) due to the motion of the upper walls sliding on the fixed vertical walls, which makes the **vorticity** and the **pressure** singular in two corners of the cavity [8]. Following the previous discussion g will be set as:

$$\text{g}= \begin{cases} (1,0)^T & \text{on } \Gamma_3, \\ \mathbf{0} & \text{on } \Gamma_1 \cup \Gamma_2 \cup \Gamma_4 \end{cases}$$

The solution of the problem was carried out using a **finer discretization** when close to the boundary and attempting to the implementation of a Morthar method on two submeshes coupled through Lagrange multipliers.

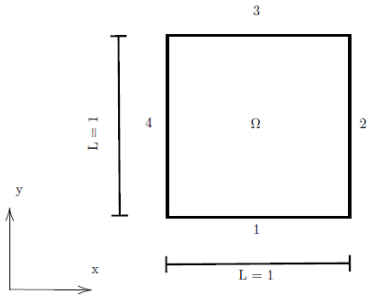

Figure 1: Geometry setup for the problem

2 Analytical description

2.1 Compatibility condition of the problem

Given the Stokes problem defined in 1.2, it is possible to prove that this system of equations admits a solution only under the compatibility condition $\int_{\partial\Omega} \boldsymbol{g} \cdot \boldsymbol{n} = 0$.

Proof. The statement follows directly from the divergence theorem. In fact, since Ω is a bounded domain, we can consider the second equation of system 1.2 to state that:

$$\int_\Omega \operatorname{div} \boldsymbol{u} dx = \int_{\partial\Omega} \boldsymbol{u} \cdot \boldsymbol{n} ds = \int_{\partial\Omega} \boldsymbol{g} \cdot \boldsymbol{n} ds \tag{1}$$

Since we are imposing that $div(\boldsymbol{u}) = 0$, then it is immediate to verify that:

$$\int_{\partial\Omega} \boldsymbol{g} \cdot \boldsymbol{n} ds = 0 \tag{2}$$

□

2.2 Weak formulation of the problem

To derive the weak formulation of the Stokes system in 1.2 we perform the following integration by parts:

$$\begin{aligned} \int_\Omega \nabla p \cdot \boldsymbol{v} dx &= -\int_\Omega p \operatorname{div}(\boldsymbol{v}) dx + \int_{\partial\Omega} p\boldsymbol{n} \cdot \boldsymbol{v} ds \\ \int_\Omega \boldsymbol{v} \cdot \Delta \boldsymbol{u} dx &= -\int_\Omega \nabla \boldsymbol{v} : \nabla \boldsymbol{u} dx + \int_{\partial\Omega} \boldsymbol{v} \cdot \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} ds \end{aligned} \tag{3}$$

Since in section 2.1 we have shown that the problem admits a solution only under the compatibility condition $\int_{\partial\Omega} \boldsymbol{g} \cdot \boldsymbol{n} = 0$, it follows that the weak formulation is:
Find $\boldsymbol{u} \in V$ *and* $p \in Q$ *such that*

$$\begin{aligned} a(\boldsymbol{u}, \boldsymbol{v}) + b(p, \boldsymbol{v}) &= F(\boldsymbol{v}) \qquad \forall \boldsymbol{v} \in V \\ b(q, \boldsymbol{u}) &= 0 \qquad \forall q \in Q \end{aligned} \tag{4}$$

where:

$$\begin{aligned} a(\boldsymbol{u}, \boldsymbol{v}) &= \int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v} dx \\ b(p, \boldsymbol{v}) &= -\int_\Omega p \operatorname{div}(\boldsymbol{v}) dx \\ F(\boldsymbol{v}) &= \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} dx \end{aligned}$$

Due to the Dirichlet conditions on the boundary, we define $V := \{\boldsymbol{v} \in [H^1(\Omega)]^2 : \boldsymbol{v} = \boldsymbol{g} \text{ on } \partial\Omega\}$, the linear space of the velocity test functions.

Since $\Gamma_D = \partial\Omega$, the pressure appears only in terms of its gradient. As a consequence, for any possible constant c, the couple $(\boldsymbol{u}, p + c)$ can be a solution. To avoid this ambiguity, the pressure space is $Q := \{q \in L^2(\Omega) : \int_\Omega q dx = 0\}$. Another possible choice could be a more generic space $\hat{Q} := \{q \in L^2(\Omega) : \exists \boldsymbol{x_0} \in \Omega \ s.t. \ q(\boldsymbol{x_0}) = q_0\}$, but this form requires a pointwise definition of the elements in $\hat{Q}$, which is not coherent with the choice of a L^2 space for q [12].

The (mixed) finite element discrete formulation of the problem can then be written as:
Find $u_h \in V_h$ *and* $p_h \in Q_h$ *such that*

$$\begin{aligned} a(\boldsymbol{u_h}, \boldsymbol{v_h}) + b(p_h, \boldsymbol{v_h}) &= F(\boldsymbol{v_h}) \qquad \forall \boldsymbol{v_h} \in V_h \\ b(q_h, \boldsymbol{u_h}) &= 0 \qquad \forall q_h \in Q_h \end{aligned} \tag{5}$$

where $V_h \subset V$ and $Q_h \subset Q$ are finite-dimensional spaces.

2.3 Finite Elements spaces

According to Saddle Point Problem theory developped throughout the course and extensively discussed in [12], a generic saddle point problem with the above-mentioned bilinear forms will satisfy the existence, uniqueness and stability of the solution only if:

- $a(\cdot, \cdot)$ is continuous and coercive;
- $b(\cdot, \cdot)$ satisfies the inf-sup condition;

$$\forall \mu_h \in Q_h \ \exists v_h \in V_h, v_h \neq 0 : b(\mu_h, v_h) \geq \beta_h \|v_h\|_V \|\mu_h\|_Q . \tag{6}$$

For the sake of conciseness, extensive proof of the previous conditions is not summarised in this report but is instead included in the cited literature. With the current theoretical background, further discussion can be developed concerning the **choice of a suitable couple of spaces**. Generally speaking, the larger the velocity space V_h, the higher the probability that the *inf-sup condition* is satisfied. **Choices considering discontinuous pressures and triangular mesh elements:**

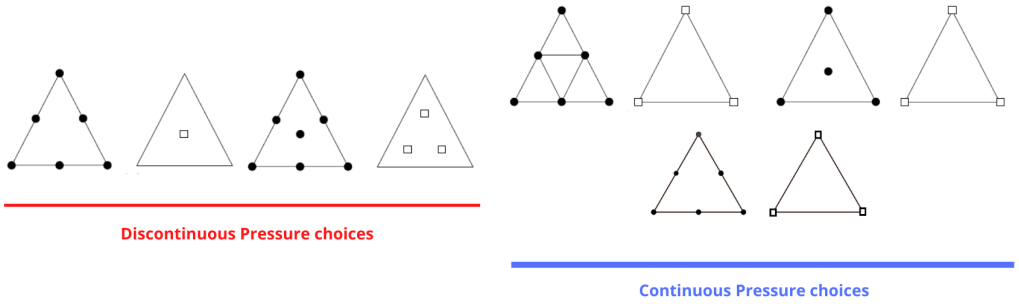

Figure 2: Possible stable elements' choices (drawn accordingly to the order in the bullet list)

- $\mathbb{P}_2 - \mathbb{P}_0$
- piecewise linear discontinuous elements for the **pressure**, while the **velocity** components are made by piecewise quadratic continuous elements enriched by a cubic bubble function on each triangle. These are the so-called **Crouzeix-Raviart** elements.

On the other hand, choices considering **continuous pressures** include:

- pressure as piecewise linear continuous function, whereas velocity is defined with piecewise linear polynomials on everyone of the **four sub-simplices**;
- pressure as piecewise linear continuous function, whereas velocity is defined with piecewise linear polynomials enriched by a cubic bubble function **(mini-element)**;
- the pair $\mathbb{P}_2 - \mathbb{P}_1$, the lowest possible degrees of the **Taylor Hood elements** family which are definitely inf-sup stable;

For the first part of this report, the latter choice, being the reference choice for many benchmark problems available within the literature, has been considered.

2.4 A-priori estimate for the error on the velocity

System 5 can be seen as a more general saddle-point problem in the form:

$$\begin{aligned} a(\boldsymbol{u_h}, \boldsymbol{v_h}) + b(p_h, \boldsymbol{v_h}) &=< \boldsymbol{f}, \boldsymbol{v_h} >_{V' \times V} \qquad \forall \boldsymbol{v_h} \in V_h \\ b(q_h, \boldsymbol{u_h}) &=< \sigma, q_h >_{Q' \times Q} \qquad \forall q_h \in Q_h \end{aligned} \tag{7}$$

Consequently, we can set $Z_h(\sigma) = \{v_h \in V_h |\ b(v_h, q_h) =< \sigma, q_h >, \forall q_h \in Q_h\}$. A necessary condition in order to have existence of a solution is that $Z_h(\sigma) \neq \emptyset$. It is immediate to see the relation with the Stokes problem defining $Z_h(0) = \{v_h \in V_h |\ b(v_h, q_h) = 0, \quad \forall q_h \in Q_h\} = KerB_h$.
We then assume that:

$$\inf_{\boldsymbol{u_h} \in KerB_h} \sup_{\boldsymbol{v_h} \in KerB_h} \frac{a(\boldsymbol{u_h}, \boldsymbol{v_h})}{||\boldsymbol{u_h}||_V ||\boldsymbol{v_h}||_V} \geq \alpha_h \tag{8}$$

We can then introduce the following proposition:

Proposition 1. *Let $(\boldsymbol{u}, p)$ be solution of the continuous form of system 7. Assume that $Z_h(\sigma) \neq \emptyset$ and condition 8 holds and let $(\boldsymbol{u_h}, p_h)$ be solution of system 7. We can state that*

$$||\boldsymbol{u} - \boldsymbol{u_h}||_V \leq (1 + \frac{\gamma}{\alpha_h}) \inf_{\boldsymbol{w_h} \in Z_h(\sigma)} ||\boldsymbol{u} - \boldsymbol{w_h}||_V + \frac{\delta}{\alpha_h} \inf_{q_h \in Q_h} ||p - q_h||_Q \tag{9}$$

Moreover, if $KerB_h \subset KerB$, this can become

$$||\boldsymbol{u} - \boldsymbol{u_h}||_V \leq (1 + \frac{\gamma}{\alpha_h}) \inf_{\boldsymbol{w_h} \in Z_h(\sigma)} ||\boldsymbol{u} - \boldsymbol{w_h}||_V \tag{10}$$

where $\gamma = ||a||$ and $\delta = ||b||$.

Proof. Given $\boldsymbol{w_h} \in Z_h(\sigma)$, since $\boldsymbol{w_h} - \boldsymbol{u_h} \in KerB_h$, we have:

$$\alpha_h ||\boldsymbol{w_h} - \boldsymbol{u_h}||_V \leq \sup_{\boldsymbol{v_h} \in KerB_h} \frac{a(\boldsymbol{w_h} - \boldsymbol{u_h}, \boldsymbol{v_h})}{||\boldsymbol{v_h}||} = \sup_{\boldsymbol{v_h} \in KerB_h} \frac{a(\boldsymbol{w_h} - \boldsymbol{u}, \boldsymbol{v_h}) + a(\boldsymbol{u} - \boldsymbol{u_h}, \boldsymbol{v_h})}{||\boldsymbol{v_h}||_V} =$$

$$\sup_{\boldsymbol{v_h} \in KerB_h} \frac{a(\boldsymbol{w_h} - \boldsymbol{u}, \boldsymbol{v_h}) - b(\boldsymbol{v_h}, p - p_h)}{||\boldsymbol{v_h}||}$$

If $KerB_h \subset KerB$, then $\boldsymbol{v_h} \in KerB$ and $\sup_{\boldsymbol{v_h} \in KerB_h} \frac{a(\boldsymbol{w_h} - \boldsymbol{u}, \boldsymbol{v_h}) + a(\boldsymbol{u} - \boldsymbol{u_h}, \boldsymbol{v_h})}{||\boldsymbol{v_h}||_V} \leq \gamma ||\boldsymbol{w_h} - \boldsymbol{u}||_V$. Using the triangle inequality 10 follows. In general case, where this inclusion is not true, we can exploit the continuity of $b(\boldsymbol{v_h}, p - p_h)$ and obtain that:

$$\alpha_h ||\boldsymbol{u_h} - \boldsymbol{w_h}||_V \leq \sup_{\boldsymbol{v_h} \in KerB_h} \frac{a(\boldsymbol{w_h} - \boldsymbol{u}, \boldsymbol{v_h}) - b(\boldsymbol{v_h}, p - q_h)}{||\boldsymbol{v_h}||} \leq \gamma ||\boldsymbol{u} - \boldsymbol{w_h}||_V + \delta ||p - q_h||_Q \tag{11}$$

9 follows from triangular inequality. □

Since we still have to find a bound for both $||p - q_h||_Q$ and $||\boldsymbol{u} - \boldsymbol{w_h}||_V$, we can show that:

Proposition 2. *Given the assumptions of Prop.1, let $\beta_h > 0$ be the LBB-condition's constant s.t. $sup_{\boldsymbol{v_h} \in V_h} \frac{b(\boldsymbol{v_h}, q_h)}{||\boldsymbol{v_h}||_V} \geq \beta_h ||q_h||_Q$, then we have:*

$$\inf_{\boldsymbol{w_h} \in Z_h(\sigma)} ||\boldsymbol{w_h} - \boldsymbol{u}||_V \leq (1 + \frac{\delta}{\beta_h}) \inf_{\boldsymbol{v_h} \in V_h} ||\boldsymbol{v_h} - \boldsymbol{u}||_V \tag{12}$$

Proof. Given $\boldsymbol{v_h} \in V_h$, we look for $\boldsymbol{r_h} \in V$ s.t. $b(\boldsymbol{r_h}, q_h) = b(\boldsymbol{u} - \boldsymbol{v_h}, q_h) \quad \forall q_h \in Q_h$. The existence of at least one solution is ensured by the fact that $Z_h(\sigma) \neq \emptyset$. Since we can say that

$$||\boldsymbol{r_h}||_V \leq \sup_{q_h \in Q_h} \frac{1}{\beta_h} \frac{b(\boldsymbol{u} - \boldsymbol{v_h}, q_h)}{||q_h||_Q} \leq \frac{\delta}{\beta_h} ||\boldsymbol{u} - \boldsymbol{v_h}||_V \tag{13}$$

Since we know that $\boldsymbol{w_h} = \boldsymbol{r_h} + \boldsymbol{v_h} \in Z_h(\sigma)$, thus $||\boldsymbol{u} - \boldsymbol{w_h}||_V \leq ||\boldsymbol{u} - \boldsymbol{v_h}||_V + ||\boldsymbol{r_h}||_V \leq (1 + \frac{\delta}{\beta_h}) ||\boldsymbol{u} - \boldsymbol{v_h}||_V$ □

For what concerns the pressure estimate, we can state that:

Proposition 3. *Given coercivity on $Z_h(0)$, continuity and LBB condition defined as in Prop.2, it is possible to prove that*

$$||p - p_h||_Q \leq \frac{\gamma}{\beta_h} (1 + \frac{\gamma}{\alpha_h}) \inf_{\boldsymbol{w_h} \in Q_h} ||\boldsymbol{u} - \boldsymbol{v_h}||_V + (1 + \frac{\delta}{\beta_h} + \frac{\gamma \delta}{\alpha_h \beta_h}) \inf_{q_h \in Q_h} ||q - q_h||_Q \tag{14}$$

Proof. Owing the LBB condition, it is possible to state that, for every $q_h \in Q_h$ we can write:

$$||p_h - q_h||_Q \leq \frac{1}{\beta_h} \sup_{\boldsymbol{v_h} \in V_h, \boldsymbol{v_h} \neq 0} \frac{b(\boldsymbol{v_h}, p_h - q_h)}{||\boldsymbol{v_h}||_V} \tag{15}$$

Then, by subtracting 7 with is continuous counter-part and adding and subtracting $b(\boldsymbol{u_h}, q_h)$, we obtain $b(\boldsymbol{v_h}, p_h - q_h) = a(\boldsymbol{u} - \boldsymbol{u_h}, \boldsymbol{v_h}) + b(\boldsymbol{v_h}, p - q_h)$. Then, using 15 and the continuity inequalities we achieve:

$$||p_h - q_h||_Q \leq \frac{1}{\beta_h} (\gamma ||\boldsymbol{u} - \boldsymbol{u_h}||_V + \delta ||p - q_h||_Q) \tag{16}$$

Finally, 9 brings to the thesis. □

From the previous propositions, it is finally possible to deduce the *estimate for the velocity field*:

$$||\boldsymbol{u} - \boldsymbol{u_h}||_V \leq (1 + \frac{\gamma}{\alpha_h})(1 + \frac{\delta}{\beta_h}) \inf_{\boldsymbol{v_h} \in V_h} ||\boldsymbol{u} - \boldsymbol{v_h}||_V + \frac{\delta}{\alpha_h} \inf_{q_h \in Q_h} ||p - q_h||_Q \tag{17}$$

where δ and γ are the continuity constants, while α_h and β_h are, respectively, the coercivity and inf-sup constants. This procedure has been described following [9], but a similar process is described in [12] and [2].

Assuming, that $\boldsymbol{u}$ and p are regular enough and assuming that β_h and α_h are bounded from below (which is the case when we are considering a proper couple of spaces, such as suggested in paragraph 2.3), then the previous formulation can be used to deduce an error estimate for velocity of the form:

$$||\boldsymbol{u} - \boldsymbol{u_h}||_V \leq C_1 h^r ||\boldsymbol{u}||_{H^{r+1}} + C_2 h^{m+1} ||p||_{H^{m+1}} \tag{18}$$

where r and m are the polynomial degrees of the chosen spaces for velocity and pressure respectively.

3 Mortar method

Defined the domain Ω, we can create a **partition** of this region into pairwise-disjoint non-empty open subregions $\Omega_i \subset \Omega$, $i = 1, ..., N$ s.t. $\bar{\Omega} = \cup_{i=1}^{N} \bar{\Omega}_i$. We define $\Gamma_{ij} = \Gamma_{ji} = \partial\Omega_j \cap \partial\Omega_i$ the interfaces between the portions i and j of the domain, with $i \neq j$, s.t. $\Gamma = \cup_{ij} \Gamma_{ij}$. In general, finding a solution of a system by a mortar method means finding a polynomial and continuous discrete solution inside every subregion Ω_i that satisfies a weak continuity condition on Γ [12].

The choice of mesh and polynomial degree is independent in each subdomain and there is no reason two meshes of neighbouring subdomains coincide at the interface. We must then define the space of the local velocities $V_h(\Omega_k)$ and of the local pressures $Q_h(\Omega_k)$ In our specific case, we separate the domain in two distinguished parts, divided by Γ. In particular, the mortar method with a suitable variational operator, allows to reach an optimal transmission of informations between adjacent subdomains [3].

The mortar method was evidenced to be extremely useful in fluid dynamics simulations where the viscosity of the fluid changes consistently and boundary layers formation has to be handled [4]. Due to technical issues, the method could not be implemented in Fenicsx. While a **suitable code** for problem solution was produced, the choice of finite element spaces and the properties of the mortar method are hereby discussed following a literature survey. Bernardi, Maday, and Patera [6] proved the mortar finite element method is **as accurate as** the usual finite element method, while convergence results were extensively studied in [7]. Stability for the **classical Taylor-Hood choice**, implemented both in the standard solution script handed and in the mortar one, was proved in [4] and the results can be extended to the choices of Finite Element spaces mentioned in Section 2.3, suggesting **discontinuous pressure fields could be used** with Crouzeix-Raviart element choice or with $\mathbb{P}_2 - \mathbb{P}_0$. Additionally [10] suggests a cheaper peculiar choice of non-conforming spaces on a rectangular mesh (thus not applying the mortar method). In a continuous boundary condition scenario, it employs a $\mathbb{P}_1$-nonconforming quadrilateral element for the velocity component-wise, while the pressure is approximated by a subspace of the piecewise constant functions whose dimension is two less than the number of squares in the mesh.

The weak formulation becomes:

Find $(\boldsymbol{u_{1h}}, \boldsymbol{u_{2h}}, p_{1h}, p_{2h}, \boldsymbol{\lambda_h}) \in V_h(\Omega_1) \times V_h(\Omega_2) \times Q_h(\Omega_1) \times Q_h(\Omega_2) \times M_h$ such that

$$\begin{aligned}
\int_{\Omega_1} \nabla \boldsymbol{u_{1h}} : \nabla \boldsymbol{v_{1h}} - \int_{\Omega_1} p_{1h} div(\boldsymbol{v_{1h}}) + \int_{\Gamma} (\nabla \boldsymbol{\lambda_h} \cdot \boldsymbol{n}) \cdot \boldsymbol{v_{1h}} &= \int_{\Omega_1} \boldsymbol{f} \cdot \boldsymbol{v_{1h}} \qquad \forall \boldsymbol{v_{1h}} \in V_h(\Omega_1) \\
\int_{\Omega_1} q_{1h} div(\boldsymbol{u_{1h}}) &= 0 \qquad \forall q_{1h} \in Q_h(\Omega_1) \\
\int_{\Gamma} -\frac{\partial \boldsymbol{u_{1h}}}{\partial \boldsymbol{n}} \cdot \boldsymbol{\mu_h} + \int_{\Gamma} (p_{1h} \boldsymbol{n}) \cdot \boldsymbol{\mu_h} + \int_{\Gamma} \frac{\partial \boldsymbol{u_{2h}}}{\partial \boldsymbol{n}} \cdot \boldsymbol{\mu_h} - \int_{\Gamma} (p_{2h} \boldsymbol{n}) \cdot \boldsymbol{\mu_h} &= 0 \qquad \forall \boldsymbol{\mu_h} \in M_h \\
\int_{\Omega_2} \nabla \boldsymbol{u_{2h}} : \nabla \boldsymbol{v_{2h}} - \int_{\Omega_2} p_{2h} div(\boldsymbol{v_{2h}}) + \int_{\Gamma} (\nabla \boldsymbol{\lambda_h} \cdot \boldsymbol{n}) \cdot \boldsymbol{v_{2h}} &= \int_{\Omega_2} \boldsymbol{f} \cdot \boldsymbol{v_{2h}} \qquad \forall \boldsymbol{v_{2h}} \in V_h(\Omega_2) \\
\int_{\Omega_2} q_{2h} div(\boldsymbol{u_{2h}}) &= 0 \qquad \forall q_{2h} \in Q_h(\Omega_2)
\end{aligned} \tag{19}$$

where $M_h = \{\boldsymbol{\mu_h} \in [C(\bar{\Gamma})]^2 : \boldsymbol{\mu_h} \in [P_r(e)]^2 \quad \forall e \in \zeta_s, \boldsymbol{\mu_h} \in [P_{r-1}(e)]^2$ if first or last element of $\Gamma\}$ as reported in [4], where ζ_s is the set of edges of Ω_s, the slave part of the domain, e denotes a edge of the discretization and r is the polynomial degree of the discrete velocity space. Since the flux of the fluid goes from left to right, it seems reasonable to consider the element Ω_1 as the master element and Ω_2 as slave.

4 Results and Discussion

The results for the lid-driven cavity problem as solved in Fenicsx are included in this section. In the following, pressure and velocity are plotted for different mesh refinements, considering both local and uniform refinements. The reason for choosing local mesh refinements in the first place was the necessity to better describe what happens close to the lid boundary where the **gradients** in the flow and in pressures are higher.

Results for locally refined mesh reveal the peculiarities of the lid cavity problem with regards to the pressure field. The boundary condition imposed is not continuous on the boundary, causing singular pressures around **point (0,1) and (1,1)**. As we refine the mesh closer to these points we obtain higher and higher pressure values, as expected we get closer to the **infinite value of pressure** that we would obtain near the lid boundary. This phenomenon is confirmed by the gross mesh refinements discussed hereafter and the convergence order with respect to a finely refined mesh solution. Please note that a more realistic behaviour for the pressures nearby the boundary could be obtained by imposing a parabolic behaviour to the `x` component of the velocity field on the lid boundary.

Velocity field for locally refined mesh shows the expected behaviour. The velocity approaches value 1 close to the upper boundary where the lid boundary condition dominates. The flow is directed towards the right wall of the cavity and falls back to the bottom, hence a **vortical flow** is created whose shape and velocity values are dependant on the Reynolds number. Let us recall that the Stokes equations are referring to a creeping flow, meaning that the Reynolds number is $\ll 1$. Generally, for higher Reynolds numbers we would need to take into account the convective transport term neglected in section 1.2; we would hence expect the velocities in the middle of the cavity to be higher because of less impactful energy loss due to the viscous term. This comparison is discussed as future improvement in the Appendix.

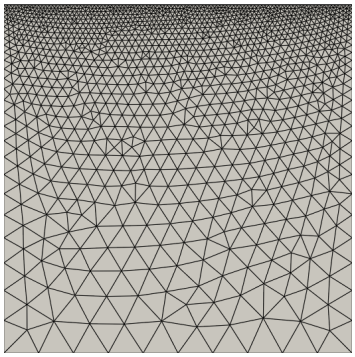

Figure 3: Locally refined mesh

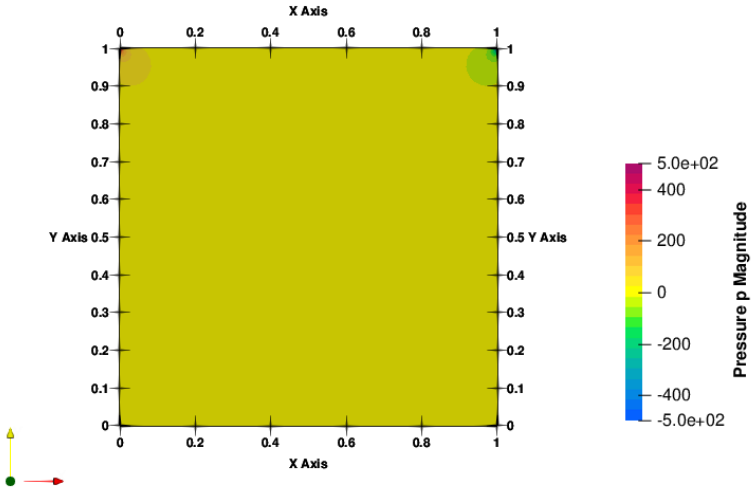

Figure 4: Pressure for locally refined mesh

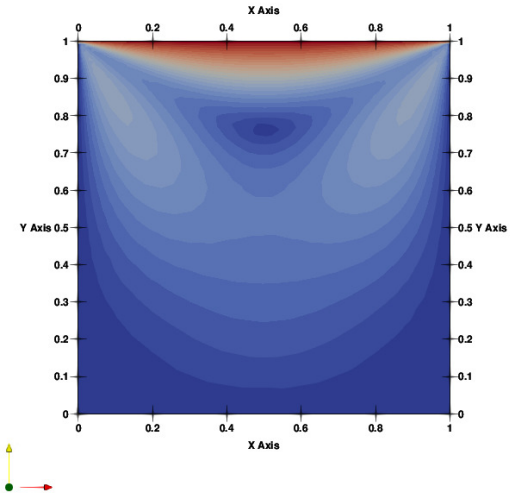

Figure 5: Velocity field | Color gradient plot for locally refined mesh

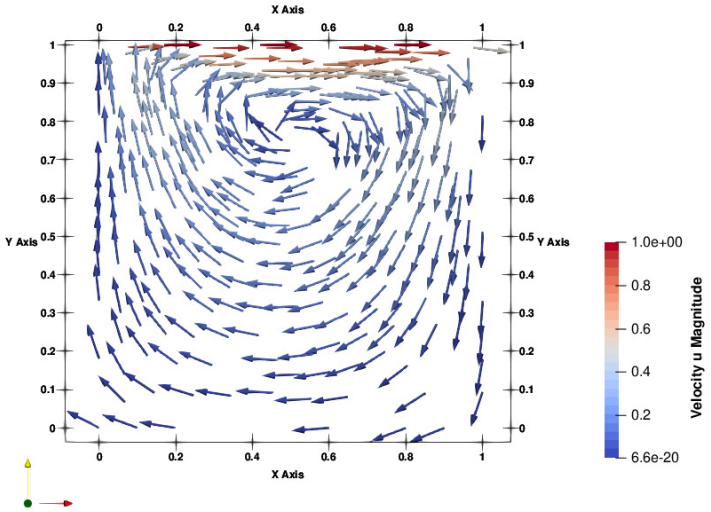

Figure 6: Velocity field | Vector plot for locally refined mesh

Results for uniform mesh confirm what was previously introduced with regards to infinite pressures at the lid boundary. A coarse uniform refinement (i.e. for example mesh-size= 0.2) brings about numerical pressures one order of magnitude lower with respect to the locally refined case. On the other hand, trivially the velocity field approximation is not altered and the same vortical behaviour is shown, however a coarser uniform mesh results into an **inaccurate description of the velocity gradient** close to the slipping lid.

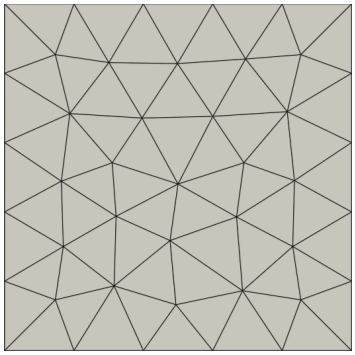

Figure 7: Uniform coarse mesh

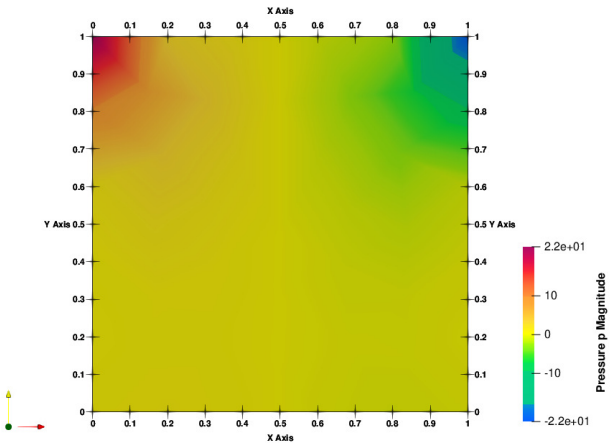

Figure 8: Pressure for uniform coarse mesh

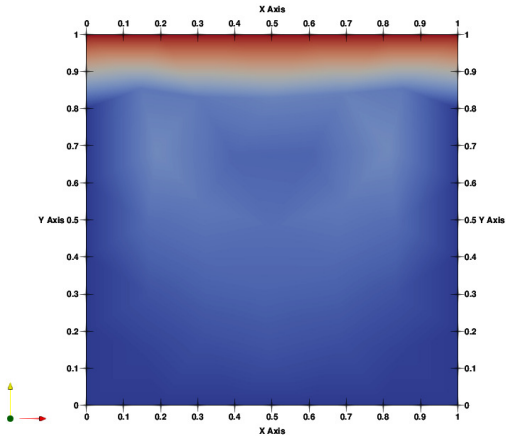

Figure 9: Velocity field | Color gradient plot for uniform coarse mesh

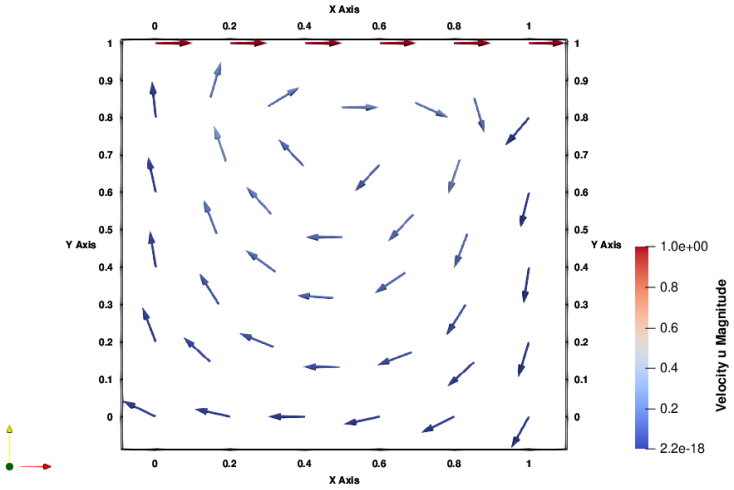

Figure 10: Velocity field | Vector plot for uniform coarse mesh

Convergence order considering L^2 norm on Fenicsx for the pressure field confirms the error estimates expected for Taylor-Hood elements. Please note that no reference solution was provided for the pressure given its peculiar behaviour near the boundary, so the students approached converge rates estimates with a technique analogous to the one used in the class to solve Poisson problem convergence [1]. Namely, a **fine uniform** mesh solution was computed and used as reference, the coarse mesh solutions obtained for various refinements were hence interpolated in the finer mesh to be confronted with the above-mentioned "self-made" benchmark. Figure 11 and 12 show convergence rates for both uniformly refined meshes and locally refined meshes, hence to show the latter tend to well approximate the correct solution too. For small enough mesh sizes, **convergence of order 2** for the pressure L^2 norm is confirmed ($\approx h^2$), as expected by the general implication: $\mathcal{P}_k \mapsto h^{k+1}\text{in}\| \cdot \|_{L^2}$.

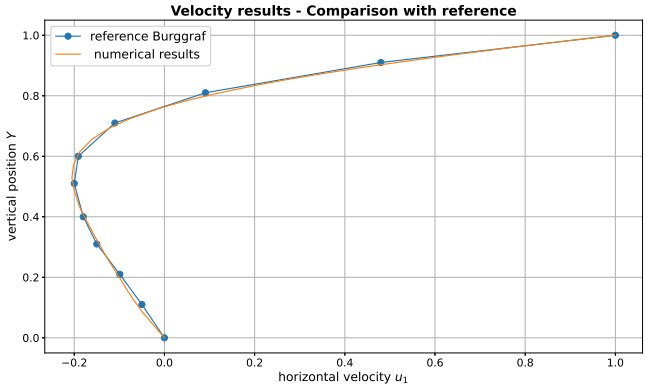

Figure 13: Comparison for horizontal velocity on mesh central vertical line - numerical results and reference found in [5] and [11]

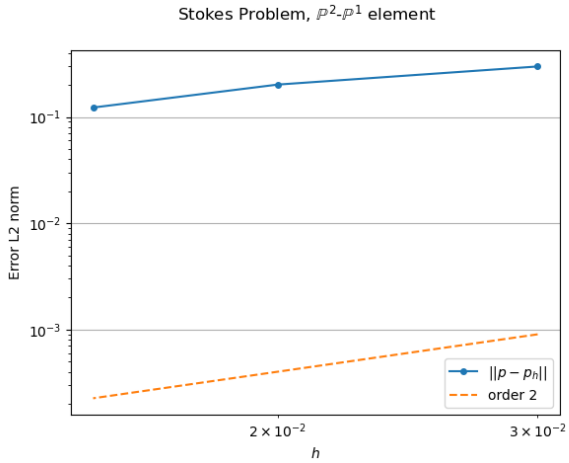

Figure 11: Convergence - L2 error for uniformly refined meshes

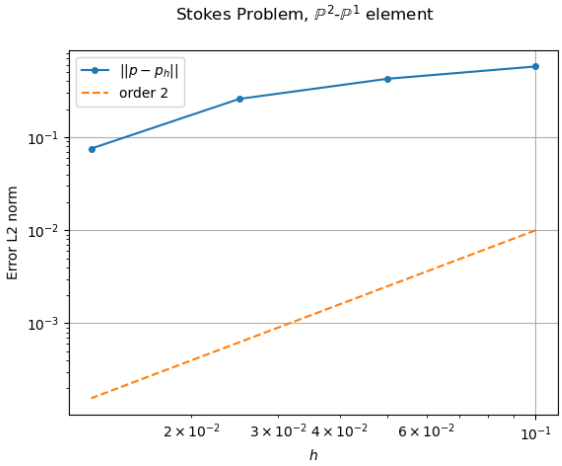

Figure 12: Convergence - L2 error for locally refined meshes

As for **velocity analysis**, the approach preferred was the comparison with benchmark solutions from literature. Since the Stokes equations refer to creeping flow scenario where the Reynolds number is assumed to be smaller than one, Burggraf results for Reynolds=0 were used as available in [11] and [5]. Please note that the cited paper does not contain complete information on velocity in all the mesh, therefore the **horizontal velocity on mesh central vertical line** was considered. Numerical results were extracted along mesh central nodes using `Paraview` and show perfect agreement with the reference solution as shown in Figure 13. H^1 norm convergence rate was not included for the sake of conciseness, but given the pressure convergence rate being confirmed and the good agreement with the benchmark solution for velocity, it is expected to follow order 2.

5 Conclusion

In the context of the Numerics for Fluids Structures and Electromagnetics course during EPFL Winter Semester 2022-2023, the working group studied the benchmark **Stokes lid-cavity problem** accordingly to the requested results of the course final assignment. Due to limitations of the Fenicsx platform and VirtualBox Ubuntu, the mortar method was only investigated on a theoretical basis and a suitable coding script for a conforming mesh has been developped. The group strived for the discussion of all the requested points despite the technical issues.

Appendix

A Additional discussion on high Reynolds number results

For the sake of comprehensiveness and following the short discussion had during the last lecture of the course concerning Navier-Stokes equations, this very short note in the Appendix is just meant to state what would have been needed in order to simulate this problem for higher Reynolds number in Fenicsx. The convective transport term $(\mathbf{u} \cdot \nabla)\mathbf{u}$ cannot be neglected in this scenario and hence leads to a trilinear form $c(u, u, v)$ which would require a different approach. For instance `TrialFunctions` in Fenicsx cannot be used anymore when a non-linear term is added, hence Newton iterative solver would be needed. This problem was not addressed in the report as it went beyond the scope of the assignment.

B Additional numerical solutions using alternative Finite Element Spaces

For the sake of complete discussion of various finite element spaces choice, the working group also changed the finite element spaces and plotted convergence rates for pressure in the **mini-element case** and in the $\mathbb{P}_2 - \mathbb{P}_0$, qualitative quick explanation of the results obtained and convergence plots are included hereafter. The velocity field description was not significantly altered by the different finite element spaces choice, while the maximum pressure reached by means of different discretizations near the two singular points varied significantly in the three simulated cases.

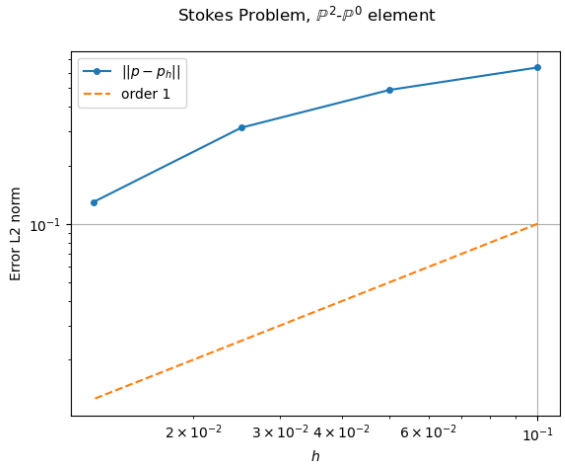

Figure 14: Convergence - L2 error P2-P0 for uniformily refined meshes

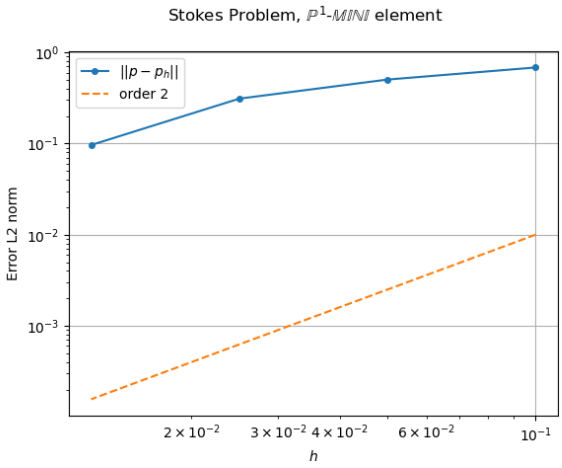

Figure 15: Convergence - L2 error P1-Mini for uniformily refined meshes

C Code

The core of the code for the main results included in the report is reported hereafter. Following a code snippet for **locally refined mesh generation**.

```
# MESH DEFINITION - REFINED
gmsh.initialize()
gmsh.model.add("refined")
fine_dim = 0.01
coarse_dim= 0.1
gmsh.model.geo.addPoint(0, 0, 0, coarse_dim, 1)
gmsh.model.geo.addPoint(1, 0, 0, coarse_dim, 2)
gmsh.model.geo.addPoint(0, 1, 0, fine_dim, 3)
gmsh.model.geo.addPoint(1, 1, 0, fine_dim, 4)

gmsh.model.geo.addLine(1, 2, 1)
gmsh.model.geo.addLine(2, 4, 2)
gmsh.model.geo.addLine(4, 3, 3)
gmsh.model.geo.addLine(3, 1, 4)

gmsh.model.geo.addCurveLoop([1, 2, 3, 4], 1)
gmsh.model.geo.addPlaneSurface([1], 1)
gmsh.model.geo.synchronize()

gmsh.model.addPhysicalGroup(1, [1, 2, 4], 5)
gmsh.model.addPhysicalGroup(2, [1], name = "My surface")
gmsh.model.mesh.generate(2)

msh, cell_markers, facet_markers = gmshio.model_to_mesh(gmsh.model,
↪  MPI.COMM_SELF, 0, gdim=2)
x = SpatialCoordinate(msh)
```

Definition of finite element spaces for Taylor-Hood is included hereafter:

```
#### DEFINITION OF SPACES  AND BOUNDARY CONDITIONS####

P2 = ufl.VectorElement("Lagrange", msh.ufl_cell(), 2)
P1 = ufl.FiniteElement("Lagrange", msh.ufl_cell(), 1)
V, Q = FunctionSpace(msh, P2), FunctionSpace(msh, P1)

# We define boundary conditions:
# No-slip boundary condition for velocity field
noslip = np.zeros(msh.geometry.dim, dtype=PETSc.ScalarType)
facets = locate_entities_boundary(msh, 1, noslip_boundary)
bc0 = dirichletbc(noslip, locate_dofs_topological(V, 1, facets), V)

# Driving velocity condition u = (1, 0) on top boundary (y = 1)
lid_velocity = Function(V)
lid_velocity.interpolate(lid_velocity_expression)
facets = locate_entities_boundary(msh, 1, lid)
bc1 = dirichletbc(lid_velocity, locate_dofs_topological(V, 1, facets))

# Collect Dirichlet boundary conditions
bcs = [bc0, bc1]
```

Definition of the **bilinear form and block preconditioner** for nested iterative matrix solver:

```
#### DEFINITION OF THE BILINEAR FORM ####

# We now define the bilinear and linear forms corresponding to the weak
# mixed formulation of the Stokes equations in a blocked structure:

# Define variational problem
(u, p) = ufl.TrialFunction(V), ufl.TrialFunction(Q)
(v, q) = ufl.TestFunction(V), ufl.TestFunction(Q)
f = Constant(msh, (PETSc.ScalarType(0), PETSc.ScalarType(0)))

a = form([[inner(grad(u), grad(v)) * dx, inner(p, div(v)) * dx],
          [inner(div(u), q) * dx, None]])
L = form([inner(f, v) * dx, inner(Constant(msh, PETSc.ScalarType(0)), q) * dx])

# We will use a block-diagonal preconditioner to solve this problem:
a_p11 = form(inner(p, q) * dx)
a_p = [[a[0][0], None],
       [None, a_p11]]
```

Additional details are included in the project folder. A snippet of the **convergence module** is included hereafter; convergence requires the coarser mesh solution to be interpolated on a finely refined mesh in order to confront the self-made benchmark with the solution for a given mesh size :

```
#INTERPOLATE COARSE SOLUTION TO LOCALLY REFINED MESH

    ################ INTERPOLATING COARSE TO FINE
    ↪  ##############################################
    ph_fine = fem.Function(Q_ref, dtype = ScalarType)
    ndofs_ref = len(ph_fine.x.array)

    # Create bounding box for function evaluation #CERCA DI CAPIRE MEGLIO COME
    ↪  FUNZIONA ESATTAMENTE QUESTO
    bb_tree = geometry.BoundingBoxTree(msh, 2)

    # Check against standard table value
    dof_vertex_coordinates = Q_ref.tabulate_dof_coordinates()
    cell_candidates = geometry.compute_collisions(bb_tree,
    ↪  dof_vertex_coordinates)
    cells = geometry.compute_colliding_cells(msh, cell_candidates,
    ↪  dof_vertex_coordinates)

    #QUI STAI ESSENZIALMENTE INTERPOLANDO LA SOLUZIONE COARSE IN UNA MESH FINE
    ↪  COME QUELLA UTILIZZATA
    #PER IL RIFERIMENTO, COSI' DA POTER POI VALUTARE L'ERRORE COME LA DIFFERENZA
    ↪  TRA COARSE E FINE MESH
    cell_of_dof = np.ndarray(ndofs_ref)
    for i in range(0,len(cells)) :
        cell_of_dof[i] = cells.links(i)[0]

    ph_fine.x.array[:] =
    ↪  (ph_coarse.eval(dof_vertex_coordinates,cell_of_dof)).flatten()
```

```
22
23      ################ COMPUTING THE ERROR ############################
24      maxh[mesh_iterator] = meshsize
25
26      error_p_h = fem.assemble_scalar(fem.form(inner(p_ref-ph_fine, p_ref-ph_fine)*
        ↪  dx))
27      norm_p = fem.assemble_scalar(fem.form( p_ref**2 * dx))
28
29      # # error over all parallel processes
30      L2_error_p[mesh_iterator] = np.sqrt(msh.comm.allreduce(error_p_h/norm_p,
        ↪  op=MPI.SUM))
31
32      meshsize = 0.5*meshsize
33      # gmsh.model.mesh.refine() # Applies uniform refinement to the mesh
34
35      if mesh_iterator > 0:gmsh.initialize()
36  gmsh.model.add("mortar")
37  meshsize= 0.05
38
39  #creation of the down rectangle
40
41  gmsh.model.geo.addPoint(0, 0, 0, meshsize, 1)
42  gmsh.model.geo.addPoint(1, 0, 0, meshsize, 2)
43  gmsh.model.geo.addPoint(0, 0.5, 0, meshsize, 3)
44  gmsh.model.geo.addPoint(1, 0.5, 0, meshsize, 4)
45  gmsh.model.geo.addPoint(0, 1, 0, meshsize, 5)
46  gmsh.model.geo.addPoint(1, 1, 0, meshsize, 6)
47
48  #linking the lines of the rectangles
49
50  gmsh.model.geo.addLine(1, 2, 1)
51  gmsh.model.geo.addLine(2, 4, 2)
52  gmsh.model.geo.addLine(4, 3, 3) #interface
53  gmsh.model.geo.addLine(3, 1, 4)
54  gmsh.model.geo.addLine(3, 5, 5)
55  gmsh.model.geo.addLine(5, 6, 6) #lid boundary
56  gmsh.model.geo.addLine(6, 4, 7)
57
58  #line between the two rectangles
59  # line= gmsh.model.geo.addLine(3, 4)
60
61  #creation of the loops
62
63  loop_down= gmsh.model.geo.addCurveLoop([1, 2, 3, 4])
64  loop_up= gmsh.model.geo.addCurveLoop([3, 5, 6, 7])
65
66  #creation of the surface
67
68  rect_down= gmsh.model.geo.addPlaneSurface([loop_down])
69  rect_up= gmsh.model.geo.addPlaneSurface([loop_up])
70
71  gmsh.model.geo.synchronize()
72
73  gmsh.model.addPhysicalGroup(1, [4, 1, 2], 1) #down boundary physical group
```

```
gmsh.model.addPhysicalGroup(1, [3], 2) #interface physical group
gmsh.model.addPhysicalGroup(1, [6], 3) #lid boundary physical group
gmsh.model.addPhysicalGroup(1, [5], 4) #upper left boundary segment
gmsh.model.addPhysicalGroup(1, [7], 5) #upper right boundary segment
gmsh.model.addPhysicalGroup(2, [rect_down], 1)
gmsh.model.addPhysicalGroup(2, [rect_up], 2)
gmsh.model.mesh.generate(2)

#mesh, subdomains, boundaries
mesh, subdomains, boundaries = dolfinx.io.gmshio.model_to_mesh(gmsh.model,
↪  comm=mpi4py.MPI.COMM_WORLD, rank=0, gdim=2)
gmsh.finalize()

#locate cells in the two subdomains
cells_Omega1 = subdomains.indices[subdomains.values == 1]
cells_Omega2 = subdomains.indices[subdomains.values == 2]

#locate boundary facets

facets_down_boundary = boundaries.indices[boundaries.values == 1]
facets_interface = boundaries.indices[boundaries.values == 2]
facets_lid_boundary = boundaries.indices[boundaries.values == 3]
facets_up_left = boundaries.indices[boundaries.values == 4]
facets_up_right = boundaries.indices[boundaries.values == 5]
        print('Convergence rate in L2 norm for pressureat refinement step
        ↪  ',mesh_iterator,
            ' for the Taylor-Hood element is ',
            (np.log(L2_error_p[mesh_iterator]) -
            ↪  np.log(L2_error_p[mesh_iterator-1]) )/  \
            (np.log(maxh[mesh_iterator]) - np.log(maxh[mesh_iterator-1]) ) )
    ############################################################

plt.loglog(maxh[:], L2_error_p[:], label='$||p - p_h||$', marker='.',
↪  markersize='8.0', linestyle='-')
plt.loglog(maxh[:], maxh[:]**2, '--', label= 'order 2')
gmsh.finalize()
plt.xlabel('$h$')
plt.ylabel('Error L2 norm')
plt.grid(True)
plt.suptitle('Stokes Problem, $\mathbb{P}^2$-$\mathbb{P}^1$ element')
plt.legend()
plt.show()
plt.savefig('stokes_TH_convergence_rates.png')
```

Mesh generation for **mortar method implementation** is included hereafter:

```
gmsh.initialize()
gmsh.model.add("mortar")
meshsize= 0.05

#creation of the down rectangle
gmsh.model.geo.addPoint(0, 0, 0, meshsize, 1)
gmsh.model.geo.addPoint(1, 0, 0, meshsize, 2)
gmsh.model.geo.addPoint(0, 0.5, 0, meshsize, 3)
```

```
gmsh.model.geo.addPoint(1, 0.5, 0, meshsize, 4)
gmsh.model.geo.addPoint(0, 1, 0, meshsize, 5)
gmsh.model.geo.addPoint(1, 1, 0, meshsize, 6)

#linking the lines of the rectangles

gmsh.model.geo.addLine(1, 2, 1)
gmsh.model.geo.addLine(2, 4, 2)
gmsh.model.geo.addLine(4, 3, 3) #interface
gmsh.model.geo.addLine(3, 1, 4)
gmsh.model.geo.addLine(3, 5, 5)
gmsh.model.geo.addLine(5, 6, 6) #lid boundary
gmsh.model.geo.addLine(6, 4, 7)

#line between the two rectangles
# line= gmsh.model.geo.addLine(3, 4)

#creation of the loops

loop_down= gmsh.model.geo.addCurveLoop([1, 2, 3, 4])
loop_up= gmsh.model.geo.addCurveLoop([3, 5, 6, 7])

#creation of the surface

rect_down= gmsh.model.geo.addPlaneSurface([loop_down])
rect_up= gmsh.model.geo.addPlaneSurface([loop_up])

gmsh.model.geo.synchronize()
gmsh.model.addPhysicalGroup(1, [4, 1, 2], 1) #down boundary physical group
gmsh.model.addPhysicalGroup(1, [3], 2) #interface physical group
gmsh.model.addPhysicalGroup(1, [6], 3) #lid boundary physical group
gmsh.model.addPhysicalGroup(1, [5], 4) #upper left boundary segment
gmsh.model.addPhysicalGroup(1, [7], 5) #upper right boundary segment
gmsh.model.addPhysicalGroup(2, [rect_down], 1)
gmsh.model.addPhysicalGroup(2, [rect_up], 2)
gmsh.model.mesh.generate(2)

#mesh, subdomains, boundaries
mesh, subdomains, boundaries = dolfinx.io.gmshio.model_to_mesh(gmsh.model,
↪  comm=mpi4py.MPI.COMM_WORLD, rank=0, gdim=2)
gmsh.finalize()

#locate cells in the two subdomains
cells_Omega1 = subdomains.indices[subdomains.values == 1]
cells_Omega2 = subdomains.indices[subdomains.values == 2]

#locate boundary facets
facets_down_boundary = boundaries.indices[boundaries.values == 1]
facets_interface = boundaries.indices[boundaries.values == 2]
facets_lid_boundary = boundaries.indices[boundaries.values == 3]
facets_up_left = boundaries.indices[boundaries.values == 4]
facets_up_right = boundaries.indices[boundaries.values == 5]
```

Finally, **blocked form definition** for mortar implementation is reported:

```
#DEFINITION OF THE MEASURES
dx = ufl.Measure("dx")(subdomain_data=subdomains)
dS = ufl.Measure("dS")(subdomain_data=boundaries)
dS = dS(2)  # restrict to the interface, which has facet ID equal to 2
n= FacetNormal(mesh)
#DEFINITION OF THE SPACES
#We firstly define vector and finite element for upper and lower half of the domain
V1_element = ufl.VectorElement("Lagrange", mesh.ufl_cell(), 2)
Q1_element = ufl.FiniteElement("Lagrange", mesh.ufl_cell(), 1)
W1_element = ufl.MixedElement(V1_element, Q1_element)
W1 = dolfinx.fem.FunctionSpace(mesh, W1_element)
V1, _ = W1.sub(0).collapse()
Q1, _ = W1.sub(1).collapse()
V2_element = ufl.VectorElement("Lagrange", mesh.ufl_cell(), 2)
Q2_element = ufl.FiniteElement("Lagrange", mesh.ufl_cell(), 1)
W2_element = ufl.MixedElement(V2_element, Q2_element)
W2 = dolfinx.fem.FunctionSpace(mesh, W2_element)
V2, _ = W2.sub(0).collapse()
Q2, _ = W2.sub(1).collapse()
#space of the lagrange multiplier
M_element = ufl.VectorElement("Lagrange", mesh.ufl_cell(), 1)
M = dolfinx.fem.FunctionSpace(mesh, M_element)

# Assemble the block linear system
(u1, u2, p1, p2, l) = (ufl.TrialFunction(V1), ufl.TrialFunction(V2), ufl.TrialFunction(Q1), ufl.TrialFunction(Q2), ufl.TrialFunction(M))
(v1, v2, q1, q2, m) = (ufl.TestFunction(V1), ufl.TestFunction(V2), ufl.TestFunction(Q1), ufl.TestFunction(Q2), ufl.TestFunction(M))
a = [[ufl.inner(ufl.grad(u1), ufl.grad(v1)) * dx(1), None, -inner(p1, div(v1)) * dx(1), None,  ufl.inner(ufl.dot(grad(l), n),v1)("-")*dS],
     [None, ufl.inner(ufl.grad(u2), ufl.grad(v2)) * dx(2), None, -inner(p2, div(v2)) * dx(2), -ufl.inner(ufl.dot(grad(l), n),v2)("+")*dS],
     [-ufl.inner(q1, ufl.div(u1))*dx(1), None, None, None, -ufl.inner(l, n*q1)("-")*dS],
     [None, -ufl.inner(q2, ufl.div(u2))*dx(2), None, None,  ufl.inner(l, n*q2)("+")*dS],
     [ufl.inner(ufl.dot(grad(u1), n),m)("-")*dS, -ufl.inner(ufl.dot(grad(u2), n),m)("+")*dS, -ufl.inner(m, n*p1)("-")*dS, ufl.inner(m, n*p2)("+")*dS, None]]

L = [ufl.inner(zero_V1, v1) * dx(1), ufl.inner(zero_V2, v2) * dx(2), ufl.inner(zero_Q1, q1) * dx(1), ufl.inner(zero_Q2, q2) * dx(2), ufl.inner(zero_M, l)("-")*dS]
a_cpp = dolfinx.fem.form(a)
L_cpp = dolfinx.fem.form(L)
A = multiphenicsx.fem.petsc.assemble_matrix_block(a_cpp, bcs=bcs, restriction=(restriction, restriction))
A.assemble()
F = multiphenicsx.fem.petsc.assemble_vector_block(L_cpp, a_cpp, bcs=bcs, restriction=restriction)
```

References

[1] B.Kapidani A.Buffa. Exercises for numerics for fluids, structures and electromagnets. Week 4, 2022-2023.

[2] Jean-Luc Guermond Alexandre Ern. *Theory and Practice of Finite Elements*. Springer, 2004.

[3] Ben Belgacem. The mortar finite element method with lagrange multipliers. *Numerische Mathematik*, pages 173–197, 1999.

[4] Faker Ben Belgacem. The mixed mortar finite element method for the incompressible stokes problem: Convergence analysis. 2000.

[5] O.R. Burggraf. Analytical and numerical studies of the structure of steady separated flows. July 1964.

[6] Bernardi et al. Domain decomposition by the mortar element method. *Asymptotic and Numerical Methods for Partial Differential Equations with Critical Parameters*, 1992.

[7] Suri et al. Uniform hp convergence results for the mortar finite element method. 2000.

[8] L.Quartapelle F.Auteri, N.Parolini. Numerical investigation on the stability of singular driven cavity flow. *Journal of Computational Physics*.

[9] Michel Fortin Franco Brezzi. *Mixed and Hybrid Finite Element Methods*. Springer.

[10] R. Lim. Nonconforming finite element method applied to the driven cavity problem. 2015.

[11] M.Aydin and R.T.Fenner. Boundary element analysis of driven cavity flow for low and moderate reynolds numbers. *INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS*, 2001.

[12] Alfio Quarteroni. *Numerical models for differential problems*. Springer, 2009.