Numerical Methods for Partial Differential Equations
A.Y. 2025/2026

Laboratory 07
Finite Element method for the Stokes problem

Exercise 1.

Let $\Omega \subset \mathbb{R}^3$ be the domain shown in Figure 1. Let us consider the stationary Stokes problem:

$$\begin{cases} -\nu\Delta\mathbf{u} + \boldsymbol{\nabla} p = \mathbf{f} & \text{in } \Omega, \quad (1a) \\ \boldsymbol{\nabla}\cdot\mathbf{u} = 0 & \text{in } \Omega, \quad (1b) \\ \mathbf{u} = \mathbf{u}_{\text{in}} & \text{on } \Gamma_{\text{in}}, \quad (1c) \\ \nu(\boldsymbol{\nabla}\mathbf{u})\mathbf{n} - p\mathbf{n} = -p_{\text{out}}\mathbf{n} & \text{on } \Gamma_{\text{out}}, \quad (1d) \\ \mathbf{u} = \mathbf{0} & \text{on } \Gamma_{\text{wall}}, \quad (1e) \end{cases}$$

where $\mathbf{u} : \Omega \to \mathbb{R}^3$ and $p : \Omega \to \mathbb{R}$ are the velocity and pressure fields of a viscous, incompressible fluid, $\nu = 1\,\text{m}^2/\text{s}$, $\mathbf{f} = \mathbf{0}$ m/s^2, $\mathbf{u}_{\text{in}} = (-\alpha\, y\,(2-y)\,(1-z)(2-z)$ m/s$, 0$ m/s$, 0$ m/s$)^T$, $\alpha = 1/(\text{m}^3\cdot\text{s})$, $p_{\text{out}} = 10\,\text{Pa}$.

1.1. Derive the weak formulation of the problem.

Solution. Let us introduce the function spaces

$$\begin{aligned} V &= \{\mathbf{v} \in \left[H^1(\Omega)\right]^3 \ : \ \mathbf{v} = \mathbf{u}_{\text{in}} \text{ on } \Gamma_{\text{in}},\ \mathbf{v} = \mathbf{0} \text{ on } \Gamma_{\text{wall}}\}\ , \\ V_0 &= \{\mathbf{v} \in \left[H^1(\Omega)\right]^3 \ : \ \mathbf{v} = \mathbf{0} \text{ on } \Gamma_{\text{in}},\ \mathbf{v} = \mathbf{0} \text{ on } \Gamma_{\text{wall}}\}\ , \\ Q &= L^2(\Omega)\ . \end{aligned}$$

Then, we take $\mathbf{v} \in V_0$, multiply it to (1a) in (1) and integrate over Ω and by parts: we obtain

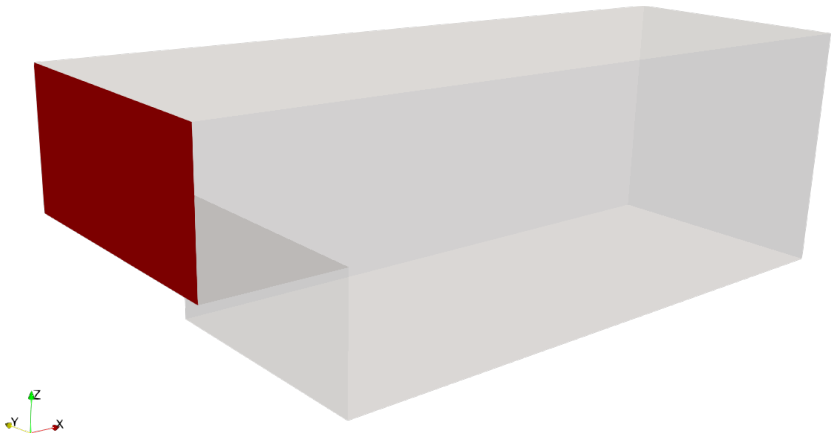
$$\underbrace{\int_\Omega \nu\boldsymbol{\nabla}\mathbf{u} : \boldsymbol{\nabla}\mathbf{v}\, d\mathbf{x}}_{a(\mathbf{u},\mathbf{v})} \underbrace{- \int_\Omega p\,\boldsymbol{\nabla}\cdot\mathbf{v}\, d\mathbf{x}}_{b(v,p)} = \underbrace{\int_\Omega \mathbf{f}\cdot\mathbf{v}\, d\mathbf{x} + \int_{\Gamma_{\text{out}}} -p\mathbf{n}\cdot\mathbf{v}\, d\sigma}_{F(\mathbf{v})}\ . \quad (2)$$

We can write the velocity in terms of a lifting function for the boundary datum:

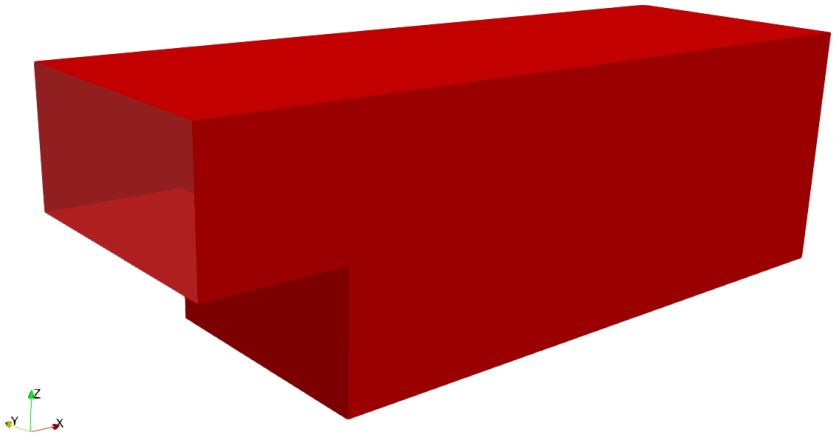
$$\mathbf{u} = \mathbf{u}_0 + \mathbf{R}(\mathbf{u}_{\text{in}})\ , \quad (3)$$

where $\mathbf{u}_0 \in V_0$ and $\mathbf{R}(\mathbf{u}_{\text{in}}) \in V$ is such that $\mathbf{R}(\mathbf{u}_{\text{in}}) = \mathbf{u}_{\text{in}}$ on Γ_{in} and $\boldsymbol{\nabla}\cdot\mathbf{R}(\mathbf{u}_{\text{in}}) = 0$. Inserting (3) into (2), we obtain

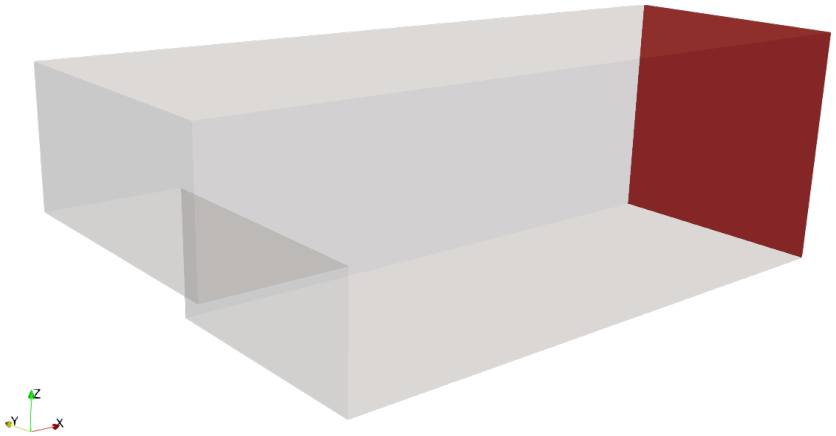
$$a(\mathbf{u}_0, \mathbf{v}) + b(\mathbf{v}, p) = F(v) - a(\mathbf{R}(\mathbf{u}_{\text{in}}, \mathbf{v}))\ .$$



Γ_{in} (mesh tag 0).



Γ_{wall} (mesh tag 1).



Γ_{out} (mesh tag 2).

Figure 1: Domain and partition of its boundary.

We proceed similarly for (1b): we take $q \in Q$, multiply it to (1b) and integrate over Ω:

$$\begin{aligned}\int_\Omega q\, \boldsymbol{\nabla} \cdot \mathbf{u}\, d\mathbf{x} &= 0\ ,\\ \int_\Omega q\, \boldsymbol{\nabla} \cdot \mathbf{u}_0\, d\mathbf{x} &= 0\ ,\\ b(\mathbf{u}_0, q) &= 0\ .\end{aligned}$$

Therefore, the weak formulation reads: find $\mathbf{u}_0 \in V_0$ and $p \in Q$ such that, for all $\mathbf{v} \in V_0$ and $q \in Q$, there holds

$$\begin{aligned}&a(\mathbf{u}_0, \mathbf{v}) + b(\mathbf{v}, p) = F(\mathbf{v}) - a(\mathbf{R}(\mathbf{u}_{\text{in}}, \mathbf{v})\ ,\\ &b(\mathbf{u}, q) = 0\ .\end{aligned}$$

1.2. Derive the finite element formulation to problem (1).

Solution. Let us introduce a mesh over Ω, and let $V_{0,h} = V_0 \cap [X_h^{r_u}(\Omega)]^3$ and $Q_h = Q \cap X_h^{r_p}(\Omega)$. Then, the discrete weak formulation reads: find $\mathbf{u}_{0,h} \in V_{0,h}$ and $p_h \in Q_h$ such that, for all $\mathbf{v}_h \in V_{0,h}$ and $q_h \in Q_h$, there holds

$$\begin{aligned}&a(\mathbf{u}_{0,h}, \mathbf{v}_h) + b(\mathbf{v}_h, p_h) = F(\mathbf{v}_h) - a(\mathbf{R}_h(\mathbf{u}_{\text{in}}), \mathbf{v}_h)\ ,\\ &b(\mathbf{u}_h, q_h) = 0\ .\end{aligned}$$

Upon discretization, this leads to an algebraic system of the form

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ \mathbf{P} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{0} \end{bmatrix}\ , \tag{4}$$

where, denoting by $\{\boldsymbol{\varphi}_i\}_{i=1}^{N_h^{\mathbf{u}}}$ the basis functions for the velocity space and by $\{\psi_i\}_{i=1}^{N_h^p}$ those for the pressure space,

$$\begin{aligned}A_{ij} &= \int_\Omega \nu \boldsymbol{\nabla}\boldsymbol{\varphi}_i : \boldsymbol{\nabla}\boldsymbol{\varphi}_j\, d\mathbf{x}\ ,\\ B_{ij} &= -\int_\Omega \psi_i\, \boldsymbol{\nabla} \cdot \boldsymbol{\varphi}_j\, d\mathbf{x}\ ,\\ F_i &= \int_\Omega \mathbf{f} \cdot \boldsymbol{\varphi}_i\, d\mathbf{x} + \int_{\Gamma_{\text{out}}} -p_{\text{out}} \mathbf{n} \cdot \boldsymbol{\varphi}_i\, d\sigma\ .\end{aligned}$$

1.3. Implement a finite element solver for (1) and compute its numerical solution using the mesh `mesh/mesh-step-5.msh` (whose boundary tags are described in Figure 1).

Solution. See `src/exercise-01.cpp` for the implementation. Refer to the comments inside the source files for a detailed explanation of what the code does. Below, we provide an overview of the major changes with respect to previous laboratories.

One significant difference is that we use block matrices and vectors to reflect the block structure of the linear system (4). This means that we use different classes to represent linear algebra objects, and their initialization is slightly different from the one of the non-block counterparts. Notice that this is not strictly necessary, and we could use non-block linear algebra just as well. However, block linear algebra data structures facilitate the implementation of a block-based preconditioner, which is quite important for the Stokes problem, as we discuss below.

Within the assembly, we rely on the `dealii::FEValuesExtractors` classes to easily access basis functions for velocity and pressure. This allows to write the assembly code in a way that closely resembles the discrete weak formulation.

The solution of the linear system requires special care in this case. Indeed, the matrix of (4) has a particular form, associated to the *saddle point* nature of the problem, that causes the zero block on the bottom right. Saddle point problems of this type are challenging for black box preconditioners (such as the ones we have used so far), that are mostly blind to the structure of the matrix. Moreover, the classes of `deal.II` that implement those preconditioners do not support block matrices and vectors.

Therefore, we implement our own preconditioner classes. We consider two different preconditioners for the problem:

$$P_1 = \begin{bmatrix} A & 0 \\ 0 & \frac{1}{\nu} M_p \end{bmatrix} \qquad P_2 = \begin{bmatrix} A & 0 \\ B & \frac{1}{\nu} M_p \end{bmatrix} ,$$

where M_p is the mass matrix associated to the pressure space, whose entries are

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

To implement a preconditioner, we need to specify the action of its inverse on a vector: our classes must implement a method `vmult` that computes, given a vector $\mathbf{x}$, the result of $P_1^{-1}\mathbf{x}$ (or $P_2^{-1}\mathbf{x}$), with $\mathbf{x} = (\mathbf{x_u}, \mathbf{x}_p)^T$.

Let us consider the preconditioner P_1. There holds:

$$P_1^{-1}\mathbf{x} = \begin{bmatrix} A^{-1} & 0 \\ 0 & \nu M_p^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{x_u} \\ \mathbf{x}_p \end{bmatrix} = \begin{bmatrix} A^{-1}\mathbf{x_u} \\ \nu M_p^{-1}\mathbf{x}_p \end{bmatrix} .$$

To compute the products $A^{-1}\mathbf{x_u}$ and $M_p^{-1}\mathbf{x}_p$, we solve the associated linear system with an iterative method, using a black box preconditioner as done previously. Since the preconditioner serves as an approximation of a system matrix, we can solve these systems with a low accuracy (i.e. with a large tolerance). As a matter of fact, we could just replace the operation $A^{-1}\mathbf{x_u}$ with $P_A^{-1}\mathbf{x_u}$, where P_A is a preconditioner for A, and we could do the same with M_p.

Solving the system using the preconditioner P_1 requires 250 GMRES iterations, whereas using the preconditioner P_2 requires 152 GMRES iterations. In qualitative terms, that is because the preconditioner P_2 is "more similar" to the system matrix than the preconditioner P_1.

However, the application of the preconditioner P_2 is potentially more costly than that of P_1, and the two should be compared in terms of computational times. This can be done either using the terminal command `time` or, in a more refined way, using the `deal.II` class `TimerOutput`.

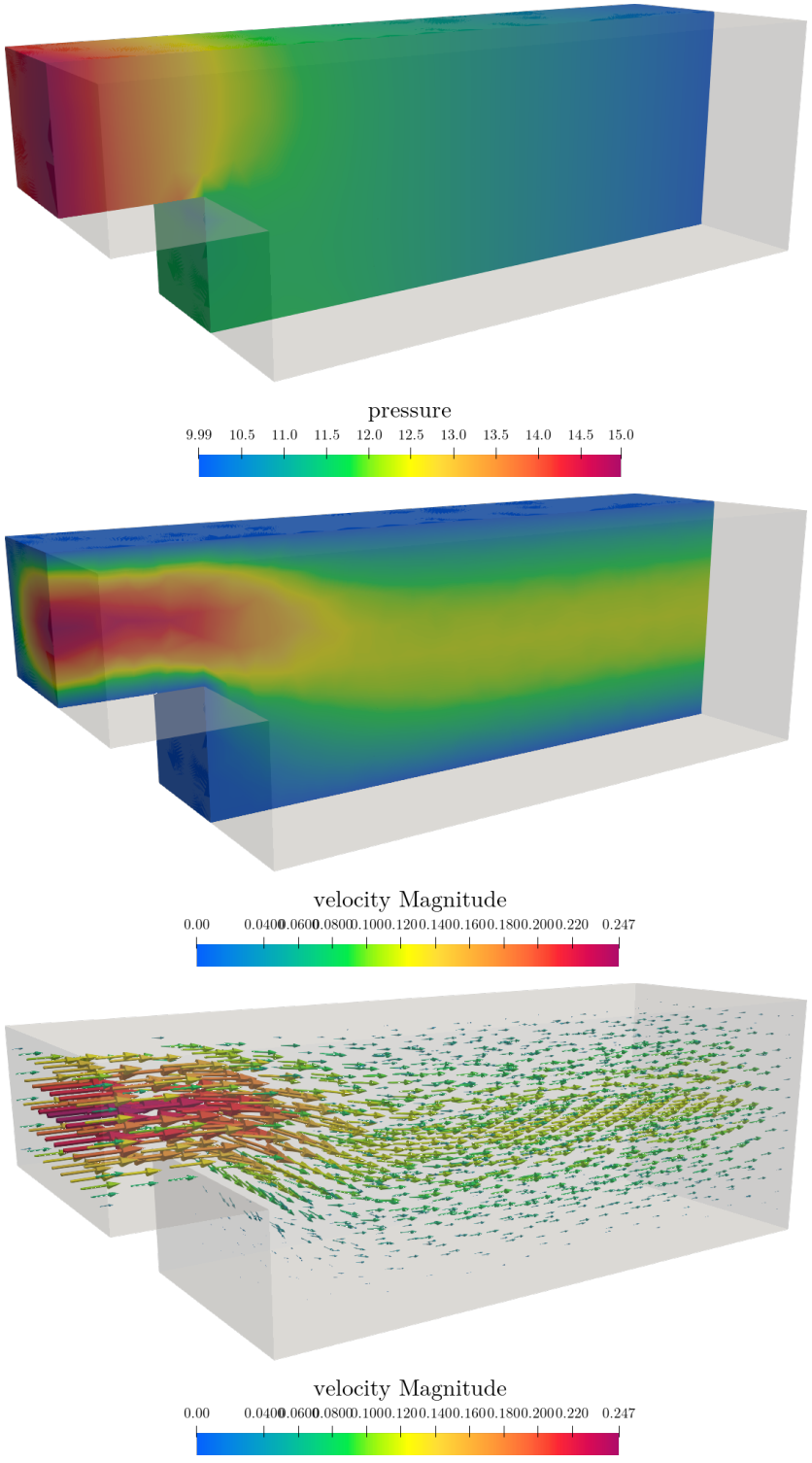


Figure 2: From top to bottom: pressure solution, velocity magnitude solution and vectors representing the velocity field (obtained using the Paraview filter "Glyph").