Numerical Methods for Saddle Point Problems

Prof. Annalisa Buffa

Project 2

Pollution of a drinking water reservoir

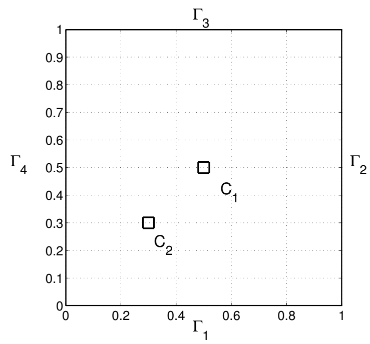

Figure 1: Vertical view of the computational domain.

Consider a pump extracting water from an underground reservoir, denoted Ω. Figure 1 shows a vertical view of the reservoir considered. Water flows from left to right of the reservoir because of a difference of pressure ("pressure head"), while the upper and lower boundaries Γ_1 and Γ_3 are considered impervious (i.e. no water flows through these boundaries). The pump is placed in the middle of the domain (C_1 area in Figure 1). Next, consider a pollution accident in the circular area C_2 located in the bottom-left quadrant of the domain (e.g. leaking sewer). Our goal is to see whether (and possibly how much) the pollutant agent will reach the water pump.

The mathematical model we consider consists of the Darcy equation for the water flow in the reservoir and a diffusion-transport equation for the pollutant agent, i.e. we assume that the pollutant will not affect the water flow ("passive transport"). The boundary conditions for the diffusion equation we prescribe are no pollutant concentration on the upstream boundary Γ_4 and on Γ_1 and Γ_3 (i.e. we assume that these boundaries are far enough from the pollution zone). Finally, we prescribe a homogeneous Neumann condition on Γ_2, i.e. the pollutant concentration does not change in the direction of the flow.
The strong formulation of the system of equations is thus

$$
\begin{aligned}
-k\nabla p &= \mathbf{u} && \text{in } \Omega,\\
\nabla \cdot \mathbf{u} &= f && \text{in } \Omega,\\
p &= p_{in} && \text{on } \Gamma_4,\\
p &= p_{out} && \text{on } \Gamma_2,\\
\mathbf{u}\cdot\mathbf{n} &= 0 && \text{on } \Gamma_1 \cup \Gamma_3,
\end{aligned}
$$

$$
\begin{aligned}
-\nu\Delta c + \mathbf{u}\cdot\nabla c &= g && \text{in } \Omega\\
c &= 0 && \text{on } \Gamma_1 \cup \Gamma_3 \cup \Gamma_4\\
\nu\nabla c\cdot\mathbf{n} &= 0 && \text{on } \Gamma_2.
\end{aligned}
$$

Let $k = 1, p_{in} = 2, p_{out} = 0, \nu = 0.05$. The pump is modeled as a forcing term localized in the square C_1 located in the middle of the domain,

$$f(\mathbf{x}) = \begin{cases} -1000 & \text{if } \mathbf{x} \in C_1 \\ 0 & \text{if } \mathbf{x} \notin C_1. \end{cases}$$

C_1 is centered in $(0.5, 0.5)$ and has sides of length 0.04. Similarly, the polluting accident is a forcing term localized in C_2,

$$g(\mathbf{x}) = \begin{cases} 1000 & \text{if } \mathbf{x} \in C_2 \\ 0 & \text{if } \mathbf{x} \notin C_2. \end{cases}$$

C_2 is centered in $(0.3, 0.3)$ and has sides of length 0.04.

Answer the following points

1. Analyze the well-posedness of the system of equations. Consider a mixed-form weak formulation for the Darcy equation.

2. Propose a suitable numerical discretization for the weak formulation of both the mixed-form of Darcy problem and the diffusion-transport equation for the pollutant agent. Then state their expected rate of convergence.

3. Solve numerically the full system and compute the total amount of pollutant being extracted, $\varphi(c) = \int_{C_1} c$.

   Some `FreeFem` hints:

   - `FreeFem` can generate meshes with subregions. To this end, parametrize both the larger computational domain and the smaller areas in anti-clockwise (or clockwise) sense, then build the mesh with in usual way `mesh Th=buildmesh(a(10)+b(10)+...)`.
     `FreeFem` will automatically denote each subregion by an integer value ("flag"). Given a point `(x,y)`, you can recover the flag of its subregion as `Th(x,y).region`.
   - Regionwise functions can be defined as

     ```
     fespace Qh(Th,P0);
     Qh f=1.*(region==1)+0.*(region==2);
     ```

   - It is possible to compute an integral over a subregion with the command `int2d(Th,label)(...)`

   Refer to the `FreeFem` manual for more details.

4. Implement a numerical procedure that iteratively refines the grid (e.g. with the command `Th=splitmesh(Th,2)`) and solves the problem until the relative difference of two consecutive values of $\varphi(c)$ is lower than 0.01.

5. Consider now a non-uniform mesh refinement, `FreeFem++` command `adaptmesh(Th,...,err=tol,hmax=... ,nbvx=100000,iso=1)`, (`tol` and `hmax` should be divided at each iteration of the adaptive algorithm by some factor 2: `tol=tol/2`, `hmax=hmax/2`) and monitor the convergence of the quantity $\varphi(c)$ as the grids get finer.