Numerical Methods for Partial Differential Equations
A.Y. 2022/2023

Laboratory 10
Finite Element method and domain decomposition algorithms

Exercise 1.

Let $\Omega \subset \mathbb{R}^2$ be the domain shown in Figure 1. Let us consider the following Poisson problem:

$$\begin{cases} -\Delta u = 0 & \text{in } \Omega \,, \\ u = 0 & \text{on } \Gamma_0 \,, \\ u = 1 & \text{on } \Gamma_1 \,, \\ \boldsymbol{\nabla} u \cdot \mathbf{n} = 0 & \text{on } \Gamma_2 \cup \Gamma_3 \,. \end{cases} \tag{1}$$

We partition the domain into two subdomains Ω_0 and Ω_1, as shown in Figure 2.

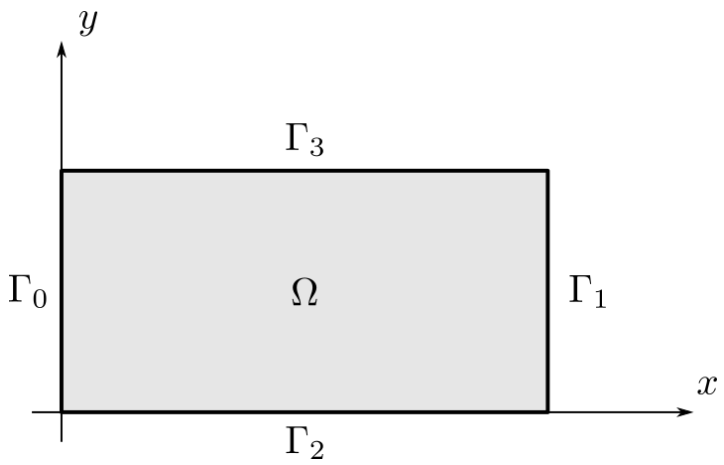


Figure 1: Domain Ω and its boundary portions.

1.1. Write an iterative algorithm for the solution of problem (1) based on the decomposition of Ω into Ω_0 and Ω_1 shown in Figure 2, using Dirichlet interface conditions for the first subdomain and Neumann interface conditions for the second subdomain.

Solution. Let us define $u_0 = u|_{\Omega_0}$ and $u_1 = u|_{\Omega_1}$, and let $\Sigma = \partial\Omega_0 \cap \partial\Omega_1$ be their interface. The Dirichlet-Neumann (DN) algorithm is the following: given an initial guess $u_1^{(0)}$, iterate for $k = 0, 1, 2, \ldots$ and until convergence:

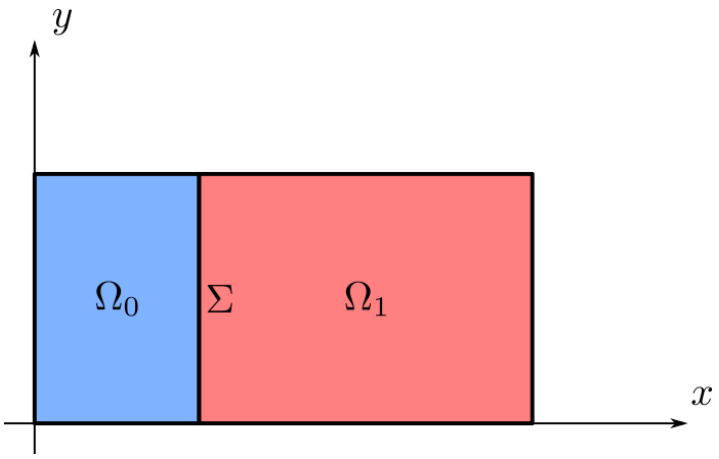


Figure 2: Decomposition of the domain Ω into Ω_0 and Ω_1 along the interface Σ.

1. solve the problem restricted to Ω_0, providing Dirichlet conditions on the interface:

$$\begin{cases} -\Delta u_0^{(k+1)} = 0 & \text{in } \Omega \ , \\ u_0^{(k+1)} = 0 & \text{on } \Gamma_0 \ , \\ u_0^{(k+1)} = u_1^{(k)} & \text{on } \Sigma \ , \\ \boldsymbol{\nabla} u_0^{(k+1)} \cdot \mathbf{n} = 0 & \text{on } \partial\Omega_0 \backslash (\Gamma_0 \cup \Sigma) \ ; \end{cases}$$

2. solve the problem restricted to Ω_1, providing Neumann conditions on the interface:

$$\begin{cases} -\Delta u_1^{(k+1)} = 0 & \text{in } \Omega \ , \\ u_1^{(k+1)} = 1 & \text{on } \Gamma_1 \ , \\ \boldsymbol{\nabla} u_1^{(k+1)} \cdot \mathbf{n} = \boldsymbol{\nabla} u_0^{(k+1)} \cdot \mathbf{n} & \text{on } \Sigma \ , \\ \boldsymbol{\nabla} u_1^{(k+1)} \cdot \mathbf{n} = 0 & \text{on } \partial\Omega_1 \backslash (\Gamma_1 \cup \Sigma) \ . \end{cases}$$

We use the increment between subsequent iterations of u_1 to stop the algorithm, that is we stop the iterations when $\|u_1^{(k+1)} - u_1^{(k)}\|$ falls below a prescribed tolerance. Other possible convergence criteria are based on checking the norm of the residual, either over the whole domain Ω or on the interface Σ.

1.2. Implement a solver for problem (1) based on the algorithm derived at previous point.

Solution. See `src/lab-10.cpp` for the implementation.

The class `Poisson2D` implements a basic solver for the 2D Poisson problem (as seen in Laboratories 3 and 4). In addition, it exposes two methods, `apply_interface_dirichlet` and `apply_interface_neumann`, that can be used to apply conditions on Σ taking the boundary data from the solution of another Poisson problem.

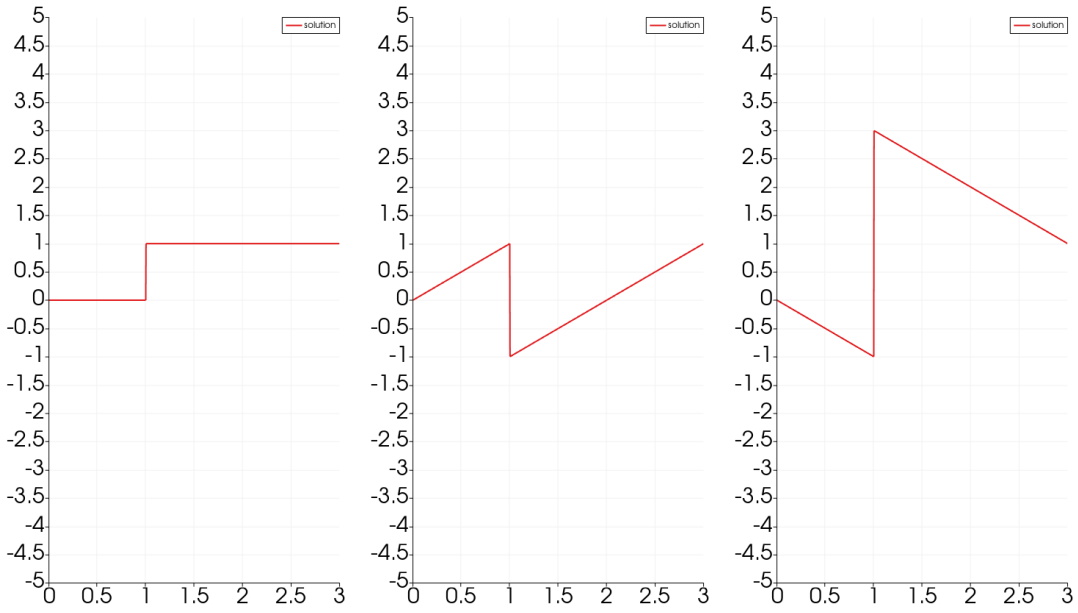


Figure 3: First three iteration of the solution obtained using the Dirichlet-Neumann algorithm. The solution is plotted along the line $y = 1$.

Then, within the `main` function, we initialize two instances of `Poisson2D`, and we solve them iteratively, applying Dirichlet conditions on the first and Neumann conditions on the second. The first few iterations of the result are shown in Figure 3. The algorithm does not converge, and the solution diverges as the number of iterations k increases.

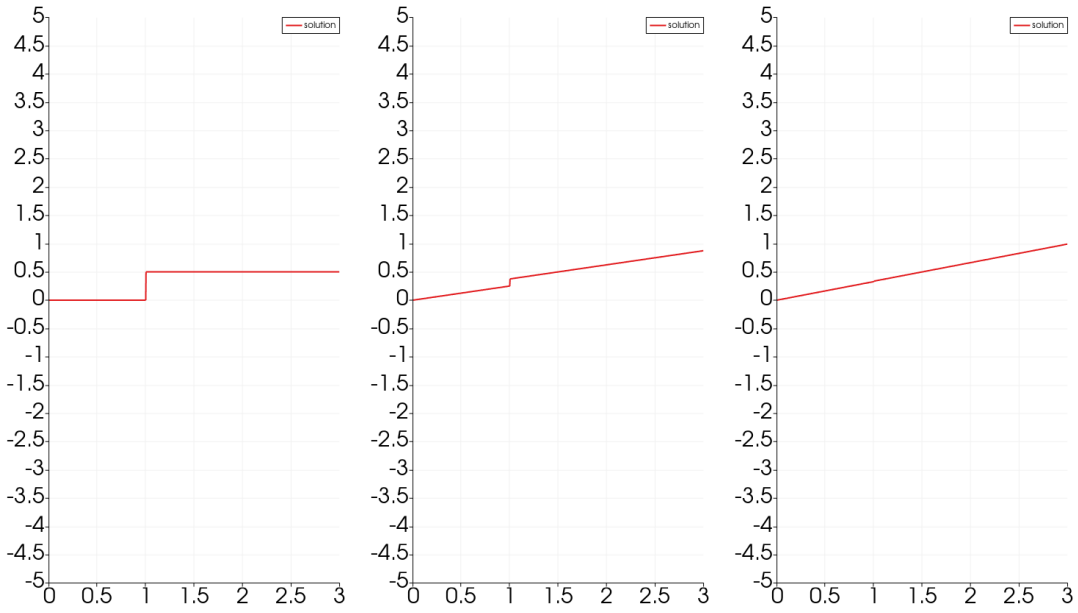
One possible way around this issue is to implement a relaxation. Given a relaxation coefficient $\lambda \in (0, 1]$, the DN algorithm with relaxation reads: given an initial guess $u_1^{(0)}$, iterate for $k = 0, 1, 2, \ldots$ and until convergence:

1. solve the problem restricted to Ω_0, providing Dirichlet conditions on the interface:

$$\begin{cases} -\Delta u_0^{(k+1)} = 0 & \text{in } \Omega \ , \\ u_0^{(k+1)} = 0 & \text{on } \Gamma_0 \ , \\ u_0^{(k+1)} = u_1^{(k)} & \text{on } \Sigma \ , \\ \boldsymbol{\nabla} u_0^{(k+1)} \cdot \mathbf{n} = 0 & \text{on } \partial\Omega_0 \backslash (\Gamma_0 \cup \Sigma) \ ; \end{cases}$$

2. solve the problem restricted to Ω_1, providing Neumann conditions on the interface:

$$\begin{cases} -\Delta \widetilde{u_1} = 0 & \text{in } \Omega \ , \\ \widetilde{u_1} = 1 & \text{on } \Gamma_1 \ , \\ \boldsymbol{\nabla} \widetilde{u_1} \cdot \mathbf{n} = \boldsymbol{\nabla} u_0^{(k+1)} \cdot \mathbf{n} & \text{on } \Sigma \ , \\ \boldsymbol{\nabla} \widetilde{u_1} \cdot \mathbf{n} = 0 & \text{on } \partial\Omega_1 \backslash (\Gamma_1 \cup \Sigma) \ ; \end{cases}$$



$k = 0$ $k = 2$ $k = 6$

Figure 4: First three iteration of the solution obtained using the Dirichlet-Neumann algorithm, with relaxation coefficient $\lambda = 0.5$. The solution is plotted along the line $y = 1$.

3. set $u_1^{(k+1)} = \lambda \widetilde{u_1} + (1 - \lambda) u_1^{(k)}$.

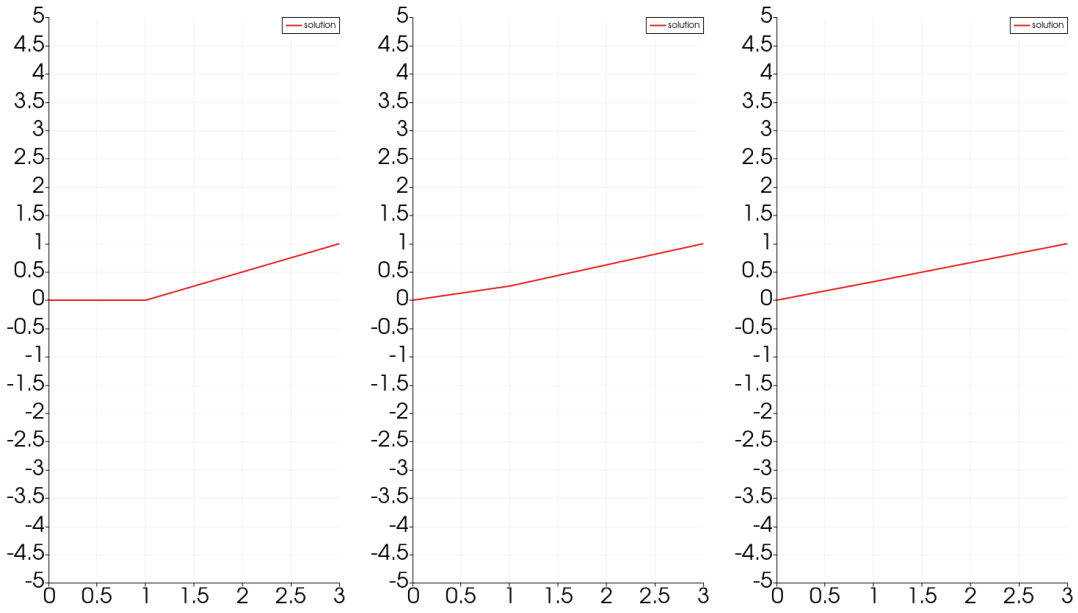
It can be shown that, for λ small enough, convergence is guaranteed (although it is not trivial to determine the maximum λ for which this happens, in the general case).

The relaxation is implemented by the `Poisson2D::apply_relaxation` method, that takes in input the old solution $u_1^{(k)}$ and the relaxation coefficient λ. For example, by setting $\lambda = 0.5$, we obtain the results shown in Figure 4. In this case, the algorithm converges to the solution rather quickly, and in 17 iterations the increment between subsequent iterations falls below the tolerance 10^{-4}.

An alternative solution to the divergence of the DN algorithm is to apply instead a Neumann-Dirichlet algorithm, based on applying Neumann interface conditions to u_0 and Dirichlet interface conditions to u_1. The algorithm, including a relaxation step, reads: given an initial guess $u_1^{(0)}$, iterate for $k = 0, 1, 2, \ldots$ and until convergence:

1. solve the problem restricted to Ω_0, providing Dirichlet conditions on the interface:

$$\begin{cases} -\Delta u_0^{(k+1)} = 0 & \text{in } \Omega \,, \\ u_0^{(k+1)} = 0 & \text{on } \Gamma_0 \,, \\ \boldsymbol{\nabla} u_0^{(k+1)} \cdot \mathbf{n} = \boldsymbol{\nabla} u_1^{(k)} \cdot \mathbf{n} & \text{on } \Sigma \,, \\ \boldsymbol{\nabla} u_0^{(k+1)} \cdot \mathbf{n} = 0 & \text{on } \partial\Omega_0 \backslash (\Gamma_0 \cup \Sigma) \,; \end{cases}$$



$k = 0$ $k = 2$ $k = 6$

Figure 5: First three iteration of the solution obtained using the Neumann-Dirichlet algorithm, with relaxation coefficient $\lambda = 1.0$. The solution is plotted along the line $y = 1$.

2. solve the problem restricted to Ω_1, providing Neumann conditions on the interface:

$$\begin{cases} -\Delta \widetilde{u_1} = 0 & \text{in } \Omega \,, \\ \widetilde{u_1} = 1 & \text{on } \Gamma_1 \,, \\ \widetilde{u_1} = u_0^{(k+1)} & \text{on } \Sigma \,, \\ \boldsymbol{\nabla} \widetilde{u_1} \cdot \mathbf{n} = 0 & \text{on } \partial\Omega_1 \backslash (\Gamma_1 \cup \Sigma) \,; \end{cases}$$

3. set $u_1^{(k+1)} = \lambda \widetilde{u_1} + (1 - \lambda) u_1^{(k)}$.

In the code, this can be obtained by swapping the Dirichlet and Neumann interface conditions in the iterative loop. The result, computed setting $\lambda = 1$ (i.e. with no relaxation) is shown in Figure 5. In this case, convergence occurs in 18 iterations.