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

Laboratory 04
Finite Element method for the heat equation

Exercise 1.

Let $\Omega = (0,1)^3$ be the unit cube and $T > 0$. Let us consider the following time-dependent problem:

$$\begin{cases} \dfrac{\partial u}{\partial t} - \boldsymbol{\nabla} \cdot (\mu \boldsymbol{\nabla} u) = f & \text{in } \Omega \times (0,T), & \text{(1a)} \\ \mu \boldsymbol{\nabla} u \cdot \mathbf{n} = 0 & \text{on } \partial\Omega \times (0,T), & \text{(1b)} \\ u = u_0 & \text{in } \Omega \times \{0\}, & \text{(1c)} \end{cases}$$

where $\mathbf{x} = (x,y,z)^T$, $\mu = 0.1$, $f(\mathbf{x},t) = 0$ and

$$u_0(\mathbf{x}) = x(x-1)\,y(y-1)\,z(z-1)\ .$$

1.1. Write the weak formulation to problem (1), and derive the semi-discrete formulation with the finite element method.

1.2. Write the fully discrete formulation of problem (1) using the θ-method to discretize the time derivative.

1.3. Implement in `deal.II` a finite element solver for problem (1) using the theta method to approximate the time derivative.

1.4. Using the implicit Euler method (i.e. setting $\theta = 1$), compute the solution to the problem (1). Set $T = 1$, $\Delta t = 0.05$ and using linear polynomials (degree $r = 1$) on the mesh `mesh/mesh-cube-10.msh`.

Opening the solution in Paraview:

1. plot the solution along the line $y = z = \dfrac{1}{2}$;
2. plot the solution over time at one point on the domain;
3. compute the integral $\displaystyle\int_\Omega u\, d\mathbf{x}$ at times $t = 0$ and $t = T$; what can you observe?

1.5. Compute the solution to the problem using the explicit Euler method (i.e. setting $\theta = 0$), using the same discretization settings as in previous question.

Possibilities for extension

Optimization of the assembly. You can notice that for this problem the system matrix never changes over time. Additionally, the right-hand side vector depends on the matrices A and M, which also do not change over time. A much more efficient code, therefore, would assemble those matrices once and for all, and then reuse them throughout the simulation. Modify the code to exploit this optimization, and compare the performance of the two implementations.

Time adaptivity. We considered a partition of the time domain where all intervals have the same size. However, in practice, it can be useful to choose small time steps during times characterized by fast dynamics (rapid changes), and larger time steps when the dynamics are slow, to save computational time while maintaining accuracy. Even better, the time step can be selected dynamically, depending on the solution itself. This is the idea underlying time-adaptive methods.

Start by modifying the code of Exercise 1 so that it allows a time-varying time step size, but prescribing the evolution of the time step as an input. Then, modify it to automatically choose the time step so that it is smaller when dynamics are fast, and larger when they are slow.