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

Laboratory 08

Finite Element method for the heat equation: convergence analysis and a nonlinear time dependent problem

Exercise 1.

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

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

with $\mu = 1$.

1.1. Knowing that the exact solution to problem (1) is

$$u_{\text{ex}}(\mathbf{x},t) = \sin(5\pi t)\sin(2\pi x)\sin(3\pi y)\sin(4\pi z)\ ,$$

where $\mathbf{x} = (x,y,z)^T$, compute the forcing term f, the boundary datum g and the initial condition u_0.

Solution. There holds:

$$\begin{aligned} \frac{\partial u}{\partial t} &= 5\pi \cos(5\pi t)\sin(2\pi x)\sin(3\pi y)\sin(4\pi z)\ , \\ \frac{\partial^2 u}{\partial x^2} &= -4\pi^2 \sin(5\pi t)\sin(2\pi x)\sin(3\pi y)\sin(4\pi z)\ , \\ \frac{\partial^2 u}{\partial y^2} &= -9\pi^2 \sin(5\pi t)\sin(2\pi x)\sin(3\pi y)\sin(4\pi z)\ , \\ \frac{\partial^2 u}{\partial z^2} &= -16\pi^2 \sin(5\pi t)\sin(2\pi x)\sin(3\pi y)\sin(4\pi z)\ , \end{aligned}$$

so that

$$\begin{aligned} f(\mathbf{x},t) &= \frac{\partial u}{\partial t} - \boldsymbol{\nabla} \cdot (\mu \boldsymbol{\nabla} u) = \frac{\partial u}{\partial t} - \mu \Delta u \\ &= (29\pi^2 \sin(5\pi t) + 5\pi \cos(5\pi t))\exp(-t)\sin(2\pi x)\sin(3\pi y)\sin(4\pi z)\ . \end{aligned}$$

The initial and boundary conditions can be obtained by suitably restricting the exact solution:

$$\begin{aligned} g(\mathbf{x},t) &= u_{\text{ex}}(\mathbf{x},t)|_{\partial\Omega} = 0 \ , \\ u_0(\mathbf{x}) &= u_{\text{ex}}(\mathbf{x},0) = 0 \ . \end{aligned}$$

This approach (computing a forcing term, boundary data and initial conditions so that the equation is satisfied by an exact solution chosen a priori) is known as *method of manufactured solutions*. It is useful when trying to verify the correctness of a finite element solver by performing convergence tests.

1.2. Starting from the code of Laboratory 07, implement a finite element solver for problem (1) with the data computed at Point 1. Then, implement a method `double Heat::compute_error(const VectorTools::NormType & norm_type)` that computes the L^2 and H^1 norms of the error at the final time T between the numerical and the exact solution.

Solution. See `src/lab-08-exercise1.cpp` for the implementation.

1.3. Consider the mesh `mesh/mesh-cube-10.msh` (corresponding to a mesh size $h = 0.1$). Solve the problem (1) with the implicit Euler method (setting $\theta = 1$), with time steps $\Delta t = 0.25, 0.125, 0.0625, 0.03125, 0.015625$ and linear finite elements (degree $r = 1$). Compute the error L^2 and H^1 norms of the error at the final time T and estimate their convergence order with respect to Δt.

Solution. See `src/lab-08-exercise1.cpp` for the implementation.

To estimate the convergence order, we can proceed as follows. Let us consider the L^2 error $e_{L^2} = \|u - u_{\text{ex}}\|_{L^2(\Omega)}$. There holds

$$e_{L^2} \leq C_h h^p + C_{\Delta t} \Delta t^q \ .$$

Since we aim at estimating q, we assume that the term depending on h is very small, i.e. $C_h h^p \approx 0$, and that the above inequality is approximately satisfied as an equality. In other words,

$$e_{L^2} \approx C_{\Delta t} \Delta t^q \ .$$

Starting from the above relation, we want to compute q. Taking two different values of Δt, we have:

$$\begin{aligned} e_{L^2,1} &\approx C_{\Delta t} \Delta t_1^q \ , \\ e_{L^2,2} &\approx C_{\Delta t} \Delta t_2^q \ . \end{aligned}$$

By taking the ratio of the two relations above, we get:

$$\begin{aligned} \frac{e_{L^2,1}}{e_{L^2,2}} &\approx \left(\frac{\Delta t_1}{\Delta t_2}\right)^q , \\ \log\left(\frac{e_{L^2,1}}{e_{L^2,2}}\right) &\approx q \log\left(\frac{\Delta t_1}{\Delta t_2}\right) , \\ q &\approx \frac{\log\left(e_{L^2,1}/e_{L^2,2}\right)}{\log\left(\Delta t_1/\Delta t_2\right)} . \end{aligned}$$

A similar argument holds for the H^1 error $e_{H^1} = \|u - u_{\text{ex}}\|_{H^1(\Omega)}$.

By estimating the convergence order of the L^2 norm with respect to Δt with this approach, we obtain the following values:

Δt	e_{L^2}	q
0.25	1.36×10^{-2}	-
0.125	8.35×10^{-3}	0.7
0.0625	5.5×10^{-3}	0.6
0.03125	4.65×10^{-3}	0.24
0.015625	4.42×10^{-3}	0.07

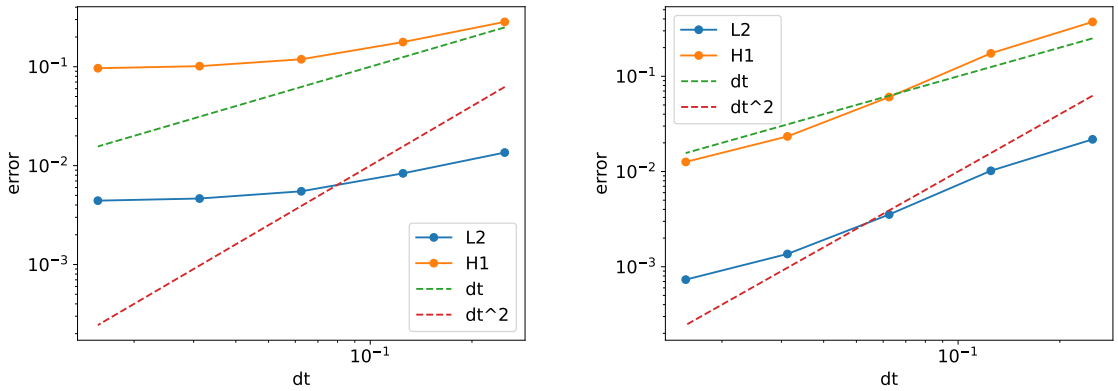
The estimated convergence order is smaller than the expected theoretical value $q = 1$. The reason for this behavior is that we are keeping fixed the mesh size h, so that the component of the error that depends on h does not reduce. Therefore, for a sufficiently small Δt, the error will be dominated by the contribution depending on h, and reducing the time step will not improve the results. We can draw a similar conclusion by looking at the log-log plots of the error against Δt, shown in Figure 1a (the plot is obtained using the script `scripts/plot-convergence.py`).

1.4. Repeat the previous point with quadratic polynomials (degree $r = 2$).

Solution. The error and estimated convergence orders are the following:

Δt	e_{L^2}	q
0.25	2.18×10^{-2}	-
0.125	1.02×10^{-3}	1.1
0.0625	3.54×10^{-3}	1.53
0.03125	1.36×10^{-3}	1.38
0.015625	7.32×10^{-4}	0.90

The error is displayed in Figure 1b. In this case, the convergence rate is in agreement with the theory: having increased the polynomial degree, we have reduced the component of the error associated to the spatial discretization, so that reducing the timestep leads to a significant reduction in the overall error.



(a) (b)

Figure 1: Error in the L^2 and H^1 norms against Δt, using linear (a) and quadratic (b) polynomials on mesh `mesh/mesh-cube-10.msh` using the implicit Euler method ($\theta = 1$).

1.5. Repeat Points 3 and 4 using the mesh `mesh/mesh-cube-20.msh` and using the Crank-Nicolson method (i.e. setting $\theta = \frac{1}{2}$).

Solution. The resulting errors are reported in Figure 2. With linear polynomials, we observe a convergence order smaller than the expected one ($q = 2$) for the Crank-Nicolson method. Conversely, by increasing the polynomial order, we recover the expected quadratic convergence rate.

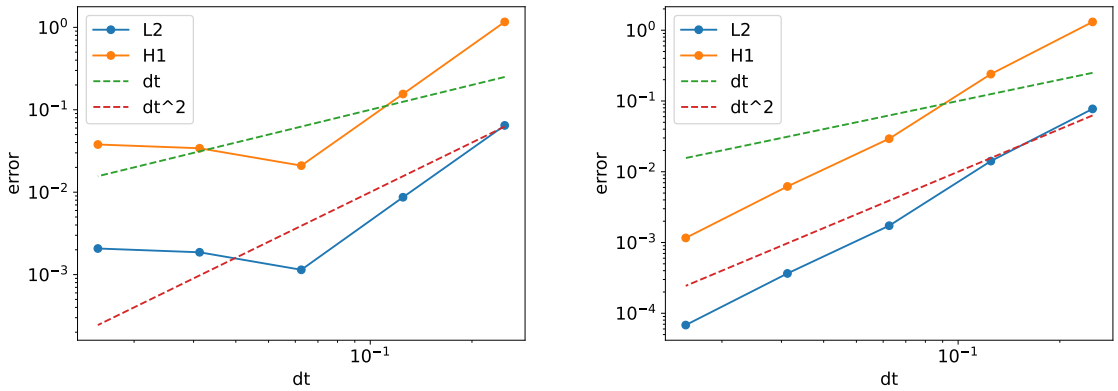
Exercise 2.

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

$$\begin{cases} \dfrac{\partial u}{\partial t} - \boldsymbol{\nabla} \cdot \left(\left(\mu_0 + \mu_1 u^2 \right) \boldsymbol{\nabla} u \right) = f & \text{in } \Omega \times (0,T), \quad (2a) \\ u = g & \text{on } \partial\Omega \times (0,T), \quad (2b) \\ u = u_0 & \text{in } \Omega \times \{0\}, \quad (2c) \end{cases}$$

with $\mu_0 = 0.1$, $\mu_1 = 1$, $g(\mathbf{x}, t) = 0$, $u_0(\mathbf{x}) = 0$ and

$$f(\mathbf{x}, t) = \begin{cases} 2 & \text{if } t < 0.25 \,, \\ 0 & \text{if } t \geq 0.25 \,. \end{cases}$$



(a) (b)

Figure 2: Error in the L^2 and H^1 norms against Δt, using linear (a) and quadratic (b) polynomials on mesh `mesh/mesh-cube-20.msh` using the Crank-Nicolson method ($\theta = 0.5$).

2.1. Implement in `deal.II` a finite element solver for problem (2), using the implicit Euler method for time discretization and Newton's method for linearization. Then, compute the solution using the mesh `mesh/mesh-cube-20.msh`, with linear finite elements (degree $r = 1$) and $\Delta t = 0.05$.

Solution. We begin by deriving the weak formulation to the problem. Let $V = H_0^1(\Omega)$. At all times $t \in (0, T)$, we look for $u(t) \in V$ such that $u(0) = u_0$ and

$$\int_\Omega \frac{\partial u}{\partial t} v \, d\mathbf{x} + \underbrace{\int_\Omega \left(\mu_0 + \mu_1 u^2\right) \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v \, d\mathbf{x}}_{b(u)(v)} = \underbrace{\int_\Omega f \, v \, d\mathbf{x}}_{F(v)} \, .$$

Defining

$$R(u)(v) = \int_\Omega \frac{\partial u}{\partial t} v \, d\mathbf{x} + b(u)(v) - F(v) \, ,$$

the weak formulation reads:

for all $t \in (0, T)$, find $u(t) \in V$ such that $R(u)(v) = 0$ for all $v \in V$ and $u(0) = u_0$.

We now discretize the weak formulation, first in space (using finite elements), then in time (using the implicit Euler method). Let us introduce a mesh over Ω, and let $V_h = V \cap X_h^r(\Omega)$ be the finite element space. The semi-discrete formulation reads:

for all $t \in (0, T)$, find $u_h(t) \in V_h$ such that $R(u_h)(v_h) = 0$ for all $v_h \in V_h$ and $u_h(0) = u_{0,h}$.

Then, we introduce the temporal discretization. Let us partition the interval $(0, T]$ into the sub-intervals $(t_n, t_{n+1}]$, with $n = 0, 1, \ldots, N_T - 1$, $t_0 = 0$, $t_{N_T} = T$ and $t_{n+1} - t_n = \Delta t$.

We denote with a superscript n the approximate solution at time t_n, i.e. $u_h^n \approx u_h(t_n)$. The fully discrete formulation of the problem can be expressed as:

$$\underbrace{\int_\Omega \frac{u_h^{n+1} - u_h^n}{\Delta t} v_h \, d\mathbf{x} + b(u_h^{n+1})(v_h) - F^{n+1}(v_h)}_{R^{n+1}(u_h^{n+1})(v_h)} = 0 \qquad \text{for all } v_h \in V_h, n = 0, 1, \ldots, N_T - 1 \tag{3}$$

This is a nonlinear problem, due to b being nonlinear in u_h^{n+1}. Therefore, we employ Newton's method for its solution. To this end, we compute the derivative of the discrete residual $R^{n+1}(u_h^{n+1})(v_h)$:

$$\begin{aligned} a(u_h^{n+1})(\delta_h, v_h) &= \int_\Omega \frac{\delta_h}{\Delta t} v_h \, d\mathbf{x} + \frac{\mathrm{d}b}{\mathrm{d}u}(\delta_h, v_h) \\ &= \int_\Omega \frac{\delta_h}{\Delta t} v_h \, d\mathbf{x} + \int_\Omega (2\mu_1 u_h^{n+1} \delta_h) \boldsymbol{\nabla} u_h^{n+1} \cdot \boldsymbol{\nabla} v d\mathbf{x} + \int_\Omega (\mu_0 + \mu_1 (u_h^{n+1})^2) \boldsymbol{\nabla} \delta_h \cdot \boldsymbol{\nabla} v_h d\mathbf{x} \, . \end{aligned}$$

Therefore, to solve problem (1), we proceed as follows: given the initial solution u_h^0, we iterate for $n = 0, 1, 2, \ldots, N_T - 1$ and compute u_h^{n+1} by solving (3) using Newton's method. At any fixed time n, Newton's method reads: given the initial guess $u_h^{n+1,(0)} = u_h^n$, iterate for $k = 0, 1, 2, \ldots$ and until convergence:

1. assemble and solve the linear problem: $a(u_h^{n+1,(k)})(\delta_h^{(k)}, v_h) = -R(u_h^{n+1,(k)})(v_h)$ for all $v_h \in V_h$;
2. set $u_h^{n+1,(k+1)} = u_h^{n+1,(k)} + \delta_h^{(k)}$.

The problem at step 1 is solved using finite elements by assembling and solving the associated linear system.

In other words, in order to solve problem (2), we need to perform two nested loops. The outer loop iterates through time (i.e. over the index n). The inner loop applies Newton's method (i.e. iterates over the index k). Within the inner loop, we repeatedly assemble J and $\mathbf{r}$ and solve the associated linear system, until a convergence criterion is satisfied.

See the files `src/lab-08-exercise2.cpp`, `src/HeatNonLinear.hpp` and `src/HeatNonLinear.cpp` for the implementation. Figure 3 shows some snapshots of the solution.

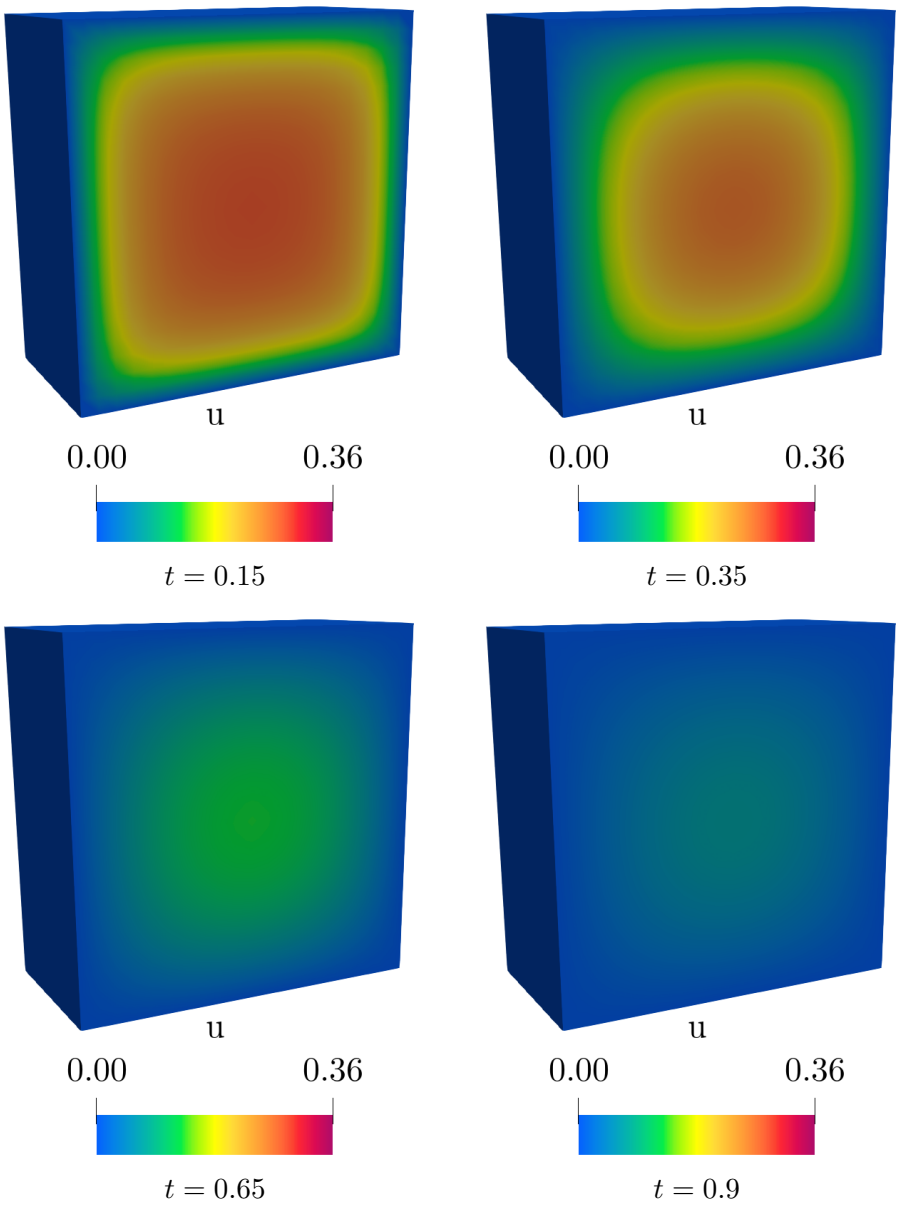


Figure 3: Snapshots of the solution to Exercise 2, clipped along a plane perpendicular to the z axis.