TMA026, Partial Differential Equations - second course
Computer Exercise 2
FEM, Time-dependent and Non-linear Problems

2.5 Bonus Points

April 2, 2018

Hand-in format The hand-in format is an informal report. Each task should be presented and then followed by its solution. Make sure that everything that is asked for is included such as tables, figures, conclusions, etc. The code should be presented in an appendix at the end of the report and it should be well-structured, well-commented, and easy to read and understand. Use of LaTeX and `anslistings.sty` is encouraged.

Introduction The purpose of this computer exercise is to solve time-dependent and nonlinear problems using the FEniCS software and to perform convergence studies. For a domain $\Omega \subset \mathbb{R}^2$ with boundary Γ, and a time $T > 0$, consider the following initial-boundary value problem

$$\begin{cases} \dot{u} - \epsilon \Delta u = f(u) & \text{in } \Omega \times (0, T], \\ u = 0 & \text{on } \Gamma \times [0, T], \\ u = u_0 & \text{in } \Omega \times \{0\}, \end{cases} \tag{1}$$

for some functions ϵ, f and u_0.

Task 1 (Time-dependency) Consider problem (1) with $\Omega = [0, 1]^2$ and let $\epsilon = 1$, $f(u) = 0$ and $u_0 = 1$. Derive a variational formulation and implement code for solving the discrete version in FEniCS. Use the generalized theta-method for applying finite difference in time. Compute the solution using both the Backward Euler and Crank-Nicolson time stepping scheme. Save both solutions to file and plot them using, e.g., ParaView.

Implementation hint: Have a look at some suitable FEniCS demos to find out how functions, boundary conditions, forms, etc. are implemented.

Task 2 (Error Convergence) For the problem in Task 1, let the final time $T = 0.3$. Compute the order of convergence of the L^2-error at the final time with respect to both the mesh-size h and time-step k, one at a time. Instead of an exact solution, compute a reference solution using a fine mesh in both space and time. For the h-convergence study, use Crank-Nicholson and a fixed sufficiently small k. For the k-convergence study, compute the order of convergence for both Backward Euler and Crank-Nicholson, one at a time, using a fixed sufficiently small h. Comment on the results.

Implementation hint: Create a callable Python-function and put the relevant code from the previous task in its function body. The Python-function should take as input discretization parameters for defining a spatial and temporal mesh and return the finite element solution or relevant norm of the error and mesh-size. Useful commands: `mesh.hmax()`, `errornorm`, `numpy.polyfit`.

Task 3 (Error Estimate) Here we are going to recover the result of Thm 10.3 by studying the error in the previous task for short final times T. For $T = 10^{-p}$, where $p = 1, 2, 3, \dots, 9$, compute the order of h-convergence of the L^2-error at the final time. For each T, let $k = 0.01T$. What happens? Now let the initial condition fulfill the boundary conditions, e.g., $u_0 = x(1-x)y(1-y)$ and redo the convergence study. Do you get the same behaviour?

Implementation hint: Put the convergence code from Task 2 in a callable Python-function and modify inputs in order to enable looping over final times and initial conditions.

Task 4 (Non-linearity) This task concerns the non-linear Allen-Cahn equation. Consider problem (1). Take $0 < \epsilon \ll 1$ and $f(u) = u - u^3$. For a subset $D \subset \Omega$, let $u_0(D) = 1$ and $u_0(\Omega \setminus D) = 0$. The subset D may be anything from a connected polygon to unconnected pseudorandom spots. Implement the following Implicit-Explicit Euler type time-stepping method in FEniCS: For $n = 1, \dots, N$, find $u_h^n \in V_h$ such that:

$$\frac{u_h^n - u_h^{n-1}}{k} + \epsilon(\nabla u_h^n, \nabla v) = (u_h^{n-1} - (u_h^{n-1})^3, v), \quad \forall v \in V_h.$$

Save the solution to file and plot it using, e.g., ParaView.

Implementation hint: Import the FEniCS module `mshr` to create more interesting domains and to define subdomains. For defining the initial value, you may want to define a subclass of `Expression` and overload the `eval_cell` function. Maybe let it use a `MeshFunction`. Useful commands: `Rectangle`, `domain.set_subdomain`, `generate_mesh`, `domain.inside`, `project`.

Task 5 (Error Convergence) Consider again the Allan-Cahn problem in Task 4, but without possible random and mesh-dependent initial conditions. Let, e.g., $\Omega = [0,1]^2$ and $u_0 = \sin(2\pi x)\sin(2\pi y)$. Compute the order of convergence of the L^2-error at the final time with respect to both the mesh-size h and the time-step k, one at a time. Compute a reference solution using a fine spatial and temporal mesh. Comment on the results.

Implementation hint: See the implementation hint for Task 2.

Notice that the FEniCS book is freely available online[1]. The study of the first chapter, "A FEniCS tutorial", pages $1 - 73$, is strongly recommended.

[1] https://fenicsproject.org/book/