Finite Elements (WS 2021)   **Exercise 7**

Prof. Dr. Peter Bastian, Michal Tóth   Submission date: 13 Dec 2021

IWR, Universität Heidelberg

Exercise 1   *Homogeneous Dirichlet problem with $\mathbb{P}^1$ elements*

Let $\Omega = [a, b] \subset \mathbb{R}$ be a real 1D domain and $\mathcal{T}_N$ be a equidistant grid on Ω with grid size $h = (b-a)/N$ for $N \in \mathbb{N}$. Let

$$V = \{v \in H^1(\Omega) \mid v(a) = v(b) = 0\}$$

be a vector space and

$$V_h = \{v_h \in C^0(\Omega) \mid \forall s \in \mathcal{T} : v_h\big|_s \in \mathbb{P}^1(s) \quad \wedge \quad v_h(a) = v_h(b) = 0\}$$

be a finite-dimensional subspace. In addition, let l be a continuous linear form $l : V \to \mathbb{R}$ and define a bilinear form

$$a(u, v) = \int_\Omega \nabla u \cdot \nabla v dx.$$

The vectors $u \in V$ and $u_h \in V_h$ fulfill

$$a(u, v) = l(v), \qquad \forall v \in V$$

and

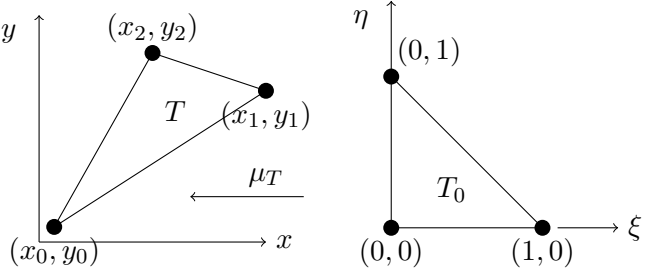$$a(u_h, v_h) = l(v_h), \qquad \forall v_h \in V_h.$$

1. Show, that $(\cdot, \cdot)_V = a(\cdot, \cdot)$ induces a scalar product on V.

2. Show, that $u(a + ih) = u_h(a + ih)$ for $i \in 0, \ldots, N$.

   Hints: Choose a simple basis for the test space; You will derive a system of equations with a unique solution.

(7 Points)

Exercise 2   *Affine mapping of $\mathbb{P}^2$ basis*

In 2D, we consider a unit triangle T_0 with nodes $n_0 = (0, 0)$, $n_1 = (1, 0)$, $n_2 = (0, 1)$ and an arbitrary triangle $T \subset \mathbb{R}^2$ with nodes $a_0 = (x_0, y_0)$, $a_1 = (x_1, y_1)$, $a_2 = (x_2, y_2)$, see picture.

1. The linear function $p \in \mathbb{P}_1(T_0)$ on T_0 can be defined using values in points n_i and the definition is unique. Find a node-basis $\tilde{\varphi}_i$, $i = 0, 1, 2$ of $\mathbb{P}_1(T_0)$ fulfilling $\tilde{\varphi}_i(n_j) = \delta_{ij}$.

2. Find a reference affine mapping $\mu_T : T_0 \to T$. Is this mapping unique and invertible?

3. The functions φ_i, $i = 0, 1, 2$ are given by

$$\varphi_i(x, y) := \tilde{\varphi}_i(\mu_T^{-1}(x, y)).$$

   Prove that $\varphi \in \mathbb{P}_1(T)$ and $\varphi_i(a_j) = \delta_{i,j}$.

4. If you want to integrate a function $v \in \mathbb{P}_1(T)$ on the T, you can first integrate it on the reference element T_0 (no change in quadrature points) and the result should be modified (regarding original element). Which factor should stay in front of the second integral?

$$\int_T v(x,y)dxdy = \cdots \int_{T_0} v\left(\mu_T\left(\xi,\eta\right)\right)d\xi d\eta.$$

(5 Points)

Exercise 3 *Elliptic operator in PDELab*

In this exercise, you will solve a PDE numerically for the first time. The program in the directory *uebungen/uebung08* in *dune-npde* solves a generic convection-diffusion problem

$$
\begin{aligned}
\nabla \cdot (-A(x)\nabla u + b(x)u) + c(x)u &= f \text{ in } \Omega, \\
u &= g \text{ on } \partial\Omega_D \, (Dirichlet) \\
(b(x,u) - A(x)\nabla u) \cdot n &= j \text{ on } \partial\Omega_N \, (Neumann) \\
-(A(x)\nabla u) \cdot n &= o \text{ on } \partial\Omega_O \, (Outflow)
\end{aligned}
$$

using P^k finite elements on a square domain. The specific parameters are defined in problem.hh. (Directory *uebungen/uebung07* was skipped.)

We do not need to implement the actual bilinear form ourselves, as that is already provided by PDELab via *ConvectionDiffusionFEM*.

1. Try to understand what the code does. What do the solver parameters mentioned in `uebung08.cc` (see *TODO* tag) do?

2. Write down explicitly the PDE being solved as well as the analytical solution. (It is implemented by the function `g`, which serves as a Dirichlet condition and the reference solution in the error computation.)

3. Look up *SubsamplingVTKWriter* in the DUNE documentation and use it instead of the *VTK-Writer*. Note that it requires an additional integer argument for its constructor. Compare VTK outputs of the subsampling and non-subsampling versions. What does SubsamplingVTK-Writer do?

4. You can set different polynomial degrees in `uebung08.ini`. What L_2 convergence rates do you expect? Do the numerical results match your expectation?

5. As you can see, the error computation in H_1 seminorm is not implemented yet. First, implement the gradient of the solution in `exact_gradient.hh`, we need it as our analytical reference. Then implement the H_1 seminorm error computation in analogy to the L_2 version. What convergence rates do you get? What convergence rate would you expect for the full H_1 norm?

6. Play around with `problem.hh`. For example, set Neumann conditions on parts of the boundary. Why do convergence rate computations break?

(10 Points)