Advanced Discretization Methods (WS 19/20)

Homework 2

(P. Knabner, L. Wester)

Deadline for submission (theory):	November 5th, 2019, 12:15
Deadline for submission (programming):	November 5th, 2019, 12:15

Recall: The triple $\{K, P, \Sigma\}$ is called a finite element, if

1. $K \subset \mathbb{R}^d$ is compact, connected, has Lipschitz boundary, $\overset{\circ}{K} \neq \emptyset$
2. P is a vector space of functions $p : K \to \mathbb{R}^m$, $m \in \mathbb{N}$
3. $\Sigma = \{\sigma_1, ..., \sigma_{n_{\text{loc}}}\} \subset P^*$, such that the mapping

$$P \ni p \mapsto (\sigma_1(p), ..., \sigma_{n_{\text{loc}}}(p))^T \in \mathbb{R}^{n_{\text{loc}}}$$

is bijective, i.e. Σ is a basis of $P*$.

Exercise 5: A different kind of element (4+4+2)

Let $T \subset \mathbb{R}^d$ be a simplex with vertices $x_0, \dots, x_d \in \mathbb{R}^d$. We consider the Crouzeix-Raviart element $(T, \mathbb{P}_1, \Sigma)$, $\Sigma := \{\sigma_0, \dots, \sigma_d\}$ with

$$\sigma_i(p) := \frac{1}{|e_i|} \int_{e_i} p(x) \, \mathrm{d}x \quad \text{for } p \in \mathbb{P}_1$$

where e_i is the edge (face) opposite to x_i.

a) Let $v \colon T \to \mathbb{R}$ be a linear polynomial. Prove that

$$\int_T v(x) \, \mathrm{d}x = |T| \, v\left(\frac{1}{d+1} \sum_{i=0}^{d} x_i\right).$$

b) Prove that the Crouzeix-Raviart element is indeed a finite element.
(**Hint:** Consider $\theta_i(x) := 1 - d\lambda_i(x)$ as possible shape function.)

c) Let $v_h \in X_h^{CR}$, where X_h^{CR} is the Crouzeix-Raviart space for a triangulation $\mathcal{T}_h$, i.e.:

$$X_h^{CR} := \{v_h \in L^1(\Omega) \mid \ v_h|_T \in \mathbb{P}_1 \ \forall T \in \mathcal{T}_h, \int_e [\![v_h]\!] \, \mathrm{d}x = 0 \ \forall \text{ interior edges } e\}$$

If e is the common edge of two elements T_1, T_2, then $[\![v_h]\!]$ refers to the jump across e, i.e. $[\![v_h]\!](x_0) = \lim_{T_1 \ni x_1 \to x_0} v_h(x_1) - \lim_{T_2 \ni x_2 \to x_0} v_h(x_2)$.
Prove that v_h is continuous at the centroids of all interior faces e.

Exercise 6: A different kind of elliptic equation (3+2+2+3)

Let $\Omega \subset \mathbb{R}^2$ be a smooth, bounded domain. A simple model to describe the bending of a plate is the biharmonic equation:

$$\Delta^2 u := \Delta(\Delta u) = f$$

with boundary values $u = 0$, $Du = 0$ on $\partial\Omega$.

a) Derive a weak formulation in the space $X := H_0^{2,2}(\Omega)$ and briefly comment on why this system is not overconstrained.

b) For $\phi \in C_0^\infty(\Omega)$, prove that

$$\int_\Omega (\Delta\phi(x))^2 \, \mathrm{d}x = |\phi|_{2,2}^2.$$

Conclude that this identity also holds for $v \in H_0^{2,2}(\Omega)$.

c) Derive a version of Poincaré's inequality in X.

d) Prove the existence of a unique solution $u \in X$ for $f \in X^*$.

Programming exercise 1: Stationary diffusion-reaction equation (20)

The goal of this exercise is to write a program that solves the stationary diffusion-reaction equation

$$\begin{aligned} -\nabla \cdot (a(x)\nabla u) + r(x)u &= f(x) \quad \text{in } \Omega \\ u &= u_D(x) \quad \text{on } \partial\Omega \end{aligned}$$

on a two-dimensional domain Ω with scalar coefficients a, r using linear Lagrange elements.

Go on StudOn and familiarize yourself with the templates that have been deposited there. As part of this exercise you should fill in the missing parts in `AssembleMatrices.m`, `AssembleRHS.m` and `TestProg1.m`.
In the first file, you need to assemble the system matrices `A` and `M` given by

$$\texttt{A(i,j)} \approx \int_\Omega a(x)\nabla\varphi_j \cdot \nabla\varphi_i \, \mathrm{d}x, \qquad \texttt{M(i,j)} \approx \int_\Omega r(x)\varphi_j\varphi_i \, \mathrm{d}x.$$

Evaluate the integrals on each triangle using the trapezoidal quadrature rule introduced in the first exercise session.
In the second file, you need to further construct the right-hand side `rhs` according to

$$\texttt{rhs(i,1)} \approx \int_\Omega f(x)\varphi_i \,\mathrm{d}x$$

again using the above-mentioned quadrature rule.
Finally you can test your routines using the script `TestProg1.m`. Here, you still need to fill in the missing code that introduces the Dirichlet boundary condition and then finally solves the linear system. Test your program using the example provided in the template on the square $[0,1]\times[0,2]$ for a handful of meshes, which can be generated through `gen_mesh_rectangle.m`.