Universität Stuttgart

mathematik ians fakultät8

Prof. Dr. B. Haasdonk — Exercise for the Lecture — 21.12.2022

Introduction to the Numerics of PDEs

WS 2022/2023– Programming Sheet 2

Submission: until 18.1.2023 at 11:30 in ILIAS.

Discussion: at 20.1.2023 in the exercise session.

Solve the following programming tasks by implementing corresponding programs. Insert your name and student-identification number in the head of your programs. Generate a PDF file with your solution (name, student-id, diagrams, outputs, explanations, insights) Upload the PDF file and your executable programs to ILIAS before the submission deadline.

Programming Task (Linear FEM) ....

For the following task we make use of the solution of NUMDGL21/22 Programming Sheet 3, where code for computing 2D triangular meshes for some domains was generated, you find this sheet and the solution in our ILIAS course.

We want to address the Poisson-problem

$$\begin{aligned} -\Delta u &= 1 \quad \text{on } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned}$$

for the two domains Ω from NUMDGL 21/22 Programming Sheet 3. We want to apply Lagrange FEM with piecewise linear and globally continuous ansatz functions for the numerical approximation.

a) Let I denote the set of indices of the inner points of a triangulation `[p,e,t]` and set $n := |I|$. Write a program `[A,b]=poisson_init(p,e,t)`, which computes the stiffness matrix $A = (a_{ij})_{i,j=1}^n \in \mathbb{R}^{n\times n}$ as `sparse`-matrix and the right hand side $b = (b_i)_{i=1}^n \in \mathbb{R}^n$ with

$$\begin{aligned} a_{ij} &= \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j \, dx \quad i,j = 1,\dots,n \\ b_i &= \int_\Omega \varphi_i \, dx, \quad i = 1,\dots,n. \end{aligned}$$

Here φ_i should denote the i-th element of the nodal basis (“hat function”).

b) Write a program `u=poisson(p,e,t)`, which calls `poisson_init`, solves the equation $Ad = b$ and visualizes the FEM solution

$$u_h(x) = \sum_{i\in I} d_i \varphi_i(x).$$

For the graphical output extend the program `mypdeplot` as provided on the ILIAS page in order to enable calls of the form `mypdeplot(p,e,t,'zdata',u)` (cmp. `doc pdeplot`). Test your program with the initial triangulations from the NUMDGL21/22 Programming Sheet 3, and store the generated visualizations.

c) Implement an adaptive Poisson solver `poisson_adapt(g,tol)`, which first generates an initial triangulation with `initmesh` of the domain `g` and calls `poisson` with this triangulation. Then the error should be estimated for each triangle t_k via

$$\sum_{\substack{l \\ t_l \text{ is neighbour of } t_k}} |S_{kl}|^2 \left|\nabla u_{h,k}(s_{kl}) - \nabla u_{h,l}(s_{kl})\right|^2 ,$$

where $\nabla u_{h,j}$ denotes the the restriction of $\nabla u_h(x)$ to $\overline{t_j}$, S_{kl} is the joint edge of t_k and t_l and $s_{kl} \in S_{kl}$ is the edge-midpoint. All triangles, which have an error (estimator) larger than `tol`/(Number of triangles) should be refined via `bisect` as provided in ILIAS and the complete scheme should be iterated until all triangles have a sufficiently small error (estimator). Test your program with the domains `g` from NUMDGL 21/22 Programming Sheet 3 and `tol=0.1`. Determine for all iterations the estimated error and a graphical illustration of the to-be-refined triangles.