Einführung in die Numerik Partieller Differentialgleichungen I (WS 16/17)
Homework 10

(V. Aizinger, B. Reuter)

Deadline for submission:	Exercises: 20.1. **before** the lecture
	Prog. exercises: 27.1. **before** the lecture
Return of corrected assignments:	in the practical sections

Exercise 22: 2+2+2=6P

Prove by applying the theorems of section 3.4 of the lecture that

$$\text{a) } |\hat{a}|_{k-j,\infty,\hat{K}} \leq C h_K^{k-j} |a|_{k-j,\infty,K}\,,$$
$$\text{b) } |\hat{p}|_{j,\hat{K}} \leq C h_K^{j} |\det(B)|^{-1/2} |p|_{j,K}\,,$$
$$\text{c) } \|\hat{q}\|_{0,\hat{K}} \leq C |\det(B)|^{-1/2} \|q\|_{0,K}$$

for $a \in W_\infty^k(K)$, $p, q \in \mathcal{P}_{k-1}(K)$ and $0 \leq j \leq k-1$. B is the matrix of the transformation $F(\hat{x}) = B\hat{x} + d$ which maps the reference element $\hat{K}$ onto an arbitrary element K and the functions $\hat{v}$ are defined by $\hat{v}(\hat{x}) = v(F(\hat{x}))$.

Exercise 23: 8P

Derive analogously to Theorem 3.28 of the lecture an interpolation result for the Hermite element (53) and the boundary value problem (30) with Dirichlet boundary conditions. The basis functions in barycentric coordinates for this Hermite elements are:

$$\begin{aligned}
-2\lambda_i^3 + 3\lambda_i^2 - 7\lambda_i \sum_{j<k,\; j\neq i,\; k\neq i} \lambda_j \lambda_k \qquad & i = 1,\dots,d+1 \\
\lambda_i\lambda_j(2\lambda_i + \lambda_j - 1) \qquad & i,j = 1,\dots,d+1;\; i \neq j \\
27\lambda_i\lambda_j\lambda_k \qquad & i,j,k = 1,\dots,d+1;\; i<j<k
\end{aligned}$$

Hint: First check that the number of the basis functions is equal to $\dim \mathcal{P}_3(K)$. Then show that $p = I_K(p)$ for all $p \in \mathcal{P}_3(K)$. After that apply the Bramble-Hilbert-Lemma in the same way as in the lecture to prove that

$$|\hat{v} - I_{\hat{K}}(\hat{v})|_{m,\hat{K}} \leq C|\hat{v}|_{k+1,\hat{K}}.$$

Last, apply Theorems 3.25 and 3.26 of the lecture to obtain the estimate.

Programming exercise 9: 8P

In this exercise, you'll extend your Finite Element solver from programming exercise 8. Instead of the structured triangular mesh use the provided triangular mesh in the ADCIRC format (ADCIRC is an ocean circulation simulator).

Your task is to read in the mesh and add new boundary conditions to solve the two-dimensional Poisson problem with Dirichlet and Neumann boundary conditions on the given domain Ω

$$\begin{aligned}
-\Delta u &= f && \text{in } \Omega, \\
\nabla u \cdot \nu &= v && \text{on } \Gamma_N \subset \partial\Omega\,, \qquad u = g \qquad \text{on } \Gamma_D = \partial\Omega \setminus \Gamma_N\,,
\end{aligned}$$

with ν being the unit normal pointing outwards of Ω.

In **StudOn** you will find a ZIP-archive with all necessary files (changed/new files compared to programming exercise 8 are: `mesh.h`, `exercise9.{h,c}`, `fem.{h,c}`, and `Makefile`; `norm.{h,c}` are no longer needed).

In `fem.c`, a new function for VTK-output of the mesh, your computed solution, the exact solution and the difference of both was added. Use, e. g., free packages *Paraview* (`http://paraview.org`) or *VisIt* (`https://wci.llnl.gov/simulation/computer-codes/visit/`) for plotting.

A description of the ADCIRC file format for the mesh is given as a PDF-file in the ZIP-archive. **Read it carefully before starting to implement!**
You have to implement two new functions: (1) `import_mesh_adcirc()` that reads in the given mesh file and fills the mesh data structure, and (2) `apply_nbc()` that modifies the linear system to account for the Neumann boundary conditions.

Start out by implementing the mesh import function and test it with Dirichlet boundary conditions only. Two suitable testcases are given in `exercise9.c`. Verify that you have reasonably low errors and have a look at the visualization of your solution. You should be able to re-use your functions from programming exercise 8. If you did not implement them in a generally applicable way for unstructured triangular grids you might have to modify them.
In the next step add the Neumann boundary condition. Assume all *open sea* vertices to form the Neumann boundary and all *land* vertices to form the Dirichlet boundary (see the PDF-file for details). Use the trapezoidal rule $\left(\int_a^b g(x)\mathrm{d}x \approx \frac{b-a}{2}(g(b)+g(a))\right)$ to evaluate the boundary integrals. Note that the error might slightly increase compared to using only Dirichlet boundaries.

You can implement your methods in `C` or `C++`, simply use `implement_me.c` or `implement_me.cpp`, respectively, and delete the other (or, if you want to mix both, delete only the respective routines from the other file). Submit your `implement_me.{c,cpp}` (and any additional files if you needed them) via StudOn.