# Scientific Computing II

Summer Semester 2014
Lecturer: Prof. Dr. Beuchler
Assistent: Bastian Bohn

universitätbonn

## Excercise sheet 3.

Closing date **29.04.2014**.

**Theoretical exercise 1.** (Triangle classes [5 points])

For the triangle $\Delta$ with corners $A, B, C$ let $D$ denote the midpoint of the edge $AB$ and $F$ the midpoint of $BC$. Define $t(\Delta) := \frac{\text{area}(\Delta)}{\text{diam}(\Delta)^2}$. Let $t(\Delta) \geq \frac{\sqrt{7}}{8}$ and $\overline{AC} \leq \overline{BC} \leq \overline{AB}$. Prove that $\overline{AC} \geq \max(\overline{CD}, \overline{AD})$ and $\overline{CD} \geq \overline{CF}$.

**Theoretical exercise 2.** (Triangle bisection [5 points])

Let $\Delta_{0,1}$ be a triangle. A bisection along its longest edge results in the triangles $\Delta_{1,1}, \Delta_{1,2}$ on level 1. Continuing this process of bisecting along the longest edge for the two new triangles and iterating further we get triangles $\Delta_{k,j}$ on level $k$ for $j = 1, \ldots, 2^k$. Let $l_k$ denote the length of the longest edge of any triangle on level $k$.

a) Prove that the maximum edge length $l_2$ of any edge in any of the resulting triangles on level 2 fulfills

$$l_2 \leq \frac{\sqrt{3}}{2} l_0.$$

b) We call a triangle $\Delta_{0,1}$ "suitable" if there exists an $N > 0$ such that

$$l_{2k} \leq \left(\frac{\sqrt{3}}{2}\right)^{\min(k,N)} \cdot \left(\frac{1}{2}\right)^{\max(k-N,0)} \cdot l_0 \quad \text{for } k \geq 0.$$

Prove that $\Delta_{0,1}$ is suitable if for each $i = 1, \ldots, 4$ either

- $\Delta_{2,i}$ is suitable or
- $\Delta_{2,i}$ is similar to $\Delta_{0,1}$ and $\text{diam}(\Delta_{2,i}) = \frac{\text{diam}(\Delta_{0,1})}{2}$.

**Programming exercise 1.** (Adaptive FEM [25 points])

The closing date for submission of the programming exercise is **May, 25th**. Please mail your commented and compilable code to `bohn@ins.uni-bonn.de`. Points are given for correctness, readability and runtime complexity.

Implement the Rivara algorithm for adaption in a triangle-based FE method (with linear and quadratic Lagrangian basis) to solve 2nd order elliptic PDEs

$$\begin{aligned} -\text{div}(A(x)\,\text{grad}\,u(x)) + c(x)u(x) &= f(x) \quad \text{on } \Omega \\ u &= 0 \quad \text{on } \Gamma_1 \subset \partial\Omega \\ \frac{\partial u}{\partial \vec{n}} &= g \text{ on } \Gamma_2 = \partial\Omega \setminus \Gamma_1 \end{aligned}$$

with elementwise constant $A, c, f$ where $A(x)$ is a diagonal matrix. As a basis you can use your own non-adaptive FE code you created in Scientific Computing I or the code

from the website (this corresponds to last semesters' reference code after the last exercise sheet) supplied in `FEMCode.tar`. The Linux Makefile compiles the code and creates a doxygen documentation. Feel free to implement any helper routine you might need.
You will need to enhance the code in the following way

a) Create an elementwise residual error estimator $\eta_K^2 = h_K^2 \|r\|_{L_2(K)}^2 + h_K \|R\|_{L_2(\partial K)}^2$ where $r = f + \operatorname{div}(A\nabla u_h) - cu_h$ is the element residual on $K$ and

$$R = \begin{cases} g - (A\nabla u_h) \cdot \vec{n}_K & \text{on } \Gamma_2 \\ -[(A\nabla u_h) \cdot \vec{n}] & \text{on } \partial K \subset \Omega \setminus (\Gamma_1 \cup \Gamma_2) \end{cases}$$

is the boundary residual where $n_K$ is the outer normal vector of the element $K$ and $[(A\nabla u_h) \cdot \vec{n}]$ is the jump discontinuity at the interface.

Taking the reference code as basis, the most important things you need to implement will be:

- The determination of the element diameter $h_K$: This can be done by using the coordinate information of the nodes of an instacne of the `Element` class.
- The evaluation of second order derivatives for quadratic Lagrange basis functions has to be implemented in `Basis.cc`,
- The determination of the normal vector of an edge and the evaluation of the normal derivatives of the basis functions (standard first order derivatives are already implemented in `Basis.cc`),
- A numerical integration routine to evaluate the $L_2$-norms. (The 7-point rule implemented in `IntegrationRule.cc` for triangles can be used for integrals over the reference element $\hat{K}$),
- The information to which element(s) a certain edge belongs can be useful. If you implement it as a new variable in the class `Edge` you have to be careful to update/set the information at the correct places in the code.

b) Implement the element bisection along the longest edge for an instance of `Element`. The following steps are the most important to be done:

- Determine the longest edge of the element.
- Bisect the longest edge, i.e. create a new node, two new edges and mark the old edge as refined (you can use the already implemented `RefineEdge` routine in `Mesh.cc`). Then create a new edge to bisect the element. You additionally have to determine if an element shares the old edge and mark this element (and/or the edge) as (possibly) nonconformal. Remark: The `isRefined` information of an edge might be helpful to determine if an possibly nonconforming element still uses an old edge and therefore needs to be refined later on.
- Delete the old element and create the two new ones to store them in the `elements` vector.

c) Implement the Rivara algorithm (Algorithm 7.3 from the lecture). To do this you need to be able to determine all nonconforming elements, bisect an element along its longest edge and also bisect by connecting the "midpoints" of two edges. For the last part, you can again use most of the code you implemented for bisecting along the longest edge.

d) Test your code: Use the grid and PDE information from `SampleGrid.txt` (readable by `createTriMeshAndPDEFromFile`) and run the Rivara algorithm five times. In every iteration mark the (roughly) 33% of all elements which employ the largest local error according to the residual error estimator. Visualize the solution and the resulting grid using a VTK-file.

With the input file you will be able to solve the PDE

$$\begin{aligned} -\Delta_{x,y} u(r,\theta) &= (\frac{175}{2} r + 100 r^{-\frac{1}{2}} - \frac{375}{2}) \cdot \sin(\frac{1}{2}\theta) \\ &\qquad \text{on } \Omega = (-1,1) \times (-1,1) \setminus \{(x,0) \mid 0 \le x \le 1\} \\ u &= 0 \quad \text{on } \partial\Omega \end{aligned}$$

where $(r, \theta)$ are the polar coordinates corresponding to the cartesian coordinates $(x, y)$. By $\Delta_{x,y}$ we mean the Laplace operator with respect to the cartesian coordinates.

The solution can be written as

$$u(r,\theta) = \begin{cases} (10 r^{\frac{1}{2}} - 50 r^{\frac{3}{2}} + 50 r^2 - 10 r^3) \cdot \sin(\frac{1}{2}\theta) & \text{if } r \le 1 \\ 0 & \text{else} \end{cases}$$

Inspect the convergence rates of the $\ell_2$ and the $\ell_\infty$ error on the FE-nodes for this problem.

Everything is set in the `SampleGrid.txt` file except for the right hand side. To reset the right hand side (elementwise constant) call `resetRHS(tornRHS)` before solving the PDE. The right hand side as well as the corresponding analytic solution are implemented in `Functions.cc`.