Institut für Numerische Simulation
Rheinische Friedrich-Wilhelms-Universität Bonn

universitätbonn

Practical Lab
Variational Methods and Inverse Problems in Imaging

Summer term 2014

Prof. Dr. M. Rumpf – A. Effland, B. Geihe, S. Simon, S. Tölkes

Problem sheet 3 **June 3rd, 2014**

Problem 3 (Primal–dual methods) For the rest of the practical course we will study *primal–dual* methods, which will be used in all of the three seperate projects.

These methods consist of two steps:

1 Solve a variation of Poisson's problem,

2 Project point-wise on a convex set with C^1-boundary.

The Poisson problem and the projection will vary from problem to problem, but the general structure remains the same. Therefore it is advisable to make the problem specific parts easily exchangable.

We will solve problems in 2D and 3D using *affine* (i. e. simplicial) Finite Elements to be compliant with the mathematical theory which is or will be given in the lecture.

Tasks: Implement a general framework for primal–dual methods, especially

0 Use affine Finite Elements in 2D and 3D. Since the theory is based on simplicial Finite Elements, we have to use the adaptive affine simplicial configurator `qc::adaptiveSimplex::AdaptiveSimplexConfiguratorLin`.

1 Implement a framework to solve Poisson's problem on an adaptively discretized domain. Solve an example problem using non-zero Neumann boundary conditions:

$$-\Delta u = u_0, \quad \partial_n u = -1 \text{ on } \{x = 0\} \cup \{x = 1\}$$
$$\text{and } \partial_n u = 1 \text{ on } \{y = 0\} \cup \{y = 1\} \setminus (\{x = 0\} \cup \{x = 1\})$$

2 Implement a general projection framework which allows you to easily exchange the sets and methods used for pointwise projection. To test your projection code, use an ellipsis $E = \{(x, y) \in \mathbb{R}^2 : \frac{x^2}{a^2} + \frac{y^2}{b^2} \leq 1\}$ as set. For a given point $q \notin E$ make the ansatz $q = p + \lambda n$, where $p \in \partial E$, $\lambda \in \mathbb{R}$ and n is a normal vector to p w. r. t. ∂E. Then derive a polynomial equation for λ and compute it using Newton's method. For this, there is the class `aol::NewtonIterationBase` available in the QuocMeshes, alternatively you can write a simple Newton's method yourself.