Institut für Numerische Simulation
Rheinische Friedrich-Wilhelms-Universität Bonn

universitätbonn

Exercises to Wissenschaftliches Rechnen I/Scientific Computing I (V3E1/F4E1)

Winter 2016/17

Prof. Dr. Martin Rumpf
Alexander Effland — Stefanie Heyden — Stefan Simon — Sascha Tölkes

Problem sheet 9

Please hand in the solutions on Tuesday January 10!

Exercise 27 **4 Points**

Let V be a Hilbert space and a any V-elliptic bilinear form. Show that the conditions of Theorem 4.1 with $U = V$ are satisfied.

Exercise 28 **4 Points**

Let $\Omega \subset \mathbb{R}^n$ and $A \in \mathbb{R}^{n,n}$ be a symmetric and positive definite matrix. Prove that

$$\frac{1}{2}\int_\Omega A\nabla u \cdot \nabla u \,\mathrm{d}x = \sup_{v \in H_0^1(\Omega)} \int_\Omega A\nabla u \cdot \nabla v - \frac{1}{2} A\nabla v \cdot \nabla v \,\mathrm{d}x$$

holds true for all $u \in H_0^1(\Omega)$.

Exercise 29 **4 Points**

Let $\Omega \subset \mathbb{R}^n$. Show that for any $f \in H_0^2(\Omega)$ the inequality

$$\|f\|_{1,2,\Omega}^2 \le \sqrt{2}\|f\|_{0,2,\Omega}\|f\|_{2,2,\Omega}$$

is valid.

Exercise 30 **4 Points**

Let $f, g \in \mathbb{R}^n$ and $F, G : \mathbb{R}^n \to \mathbb{R}$ defined by $F(x) = \frac{1}{2}|x|^2 - f \cdot x$ and $G(x) = g \cdot x$. For the constrained minimization problem

$$\min_{x \in \mathbb{R}^n \,:\, G(x)=0} F(x)$$

consider the Lagrangian $L(x, \lambda) = F(x) + \lambda G(x)$.
(i.) Compute the first and second derivative of L.
(ii.) Describe a Newton method to compute a saddle point of L.

Programming task 3

In this programming task, the code of the last two tasks will be combined to implement an adaptive grid refinement algorithm using an a-posteriori error estimator. First, consider Poisson's problem

$$\begin{aligned} - \triangle u &= f \text{ on } \Omega\,, \\ u &= u^\partial \text{ on } \partial\Omega\,. \end{aligned} \tag{1}$$

as already known from programming task 1. Here, we will assume $f \equiv 0$ and $u^\partial \neq 0$. The method to handle non-zero boundary conditions that was discussed in the lecture can be translated into the following algorithm:

1. Fill a vector $\bar{u}^\partial$ with boundary values
2. Apply the (unmasked) stiffness matrix to $\bar{u}^\partial$
3. Set right hand side vector b to $\bar{f} - L\bar{u}^\partial$, where $\bar{f}$ is the right hand side contribution resulting from f, as discussed in programming task 1 ($\bar{f} = 0$ here)
4. In b overwrite the entries corresponding to boundary nodes with the respective values of $\bar{u}^\partial$
5. Mask L as discussed in programming task 1
6. Solve

To efficiently implement this method, the class
`UnitTriangleFELinWeightedStiffIntegrator`
(from which `StiffnessMatrixIntegrator` is derived) provides methods
`assemble()` and `assembleDirichlet()`.
Now, consider the a-posteriori error estimator for Poisson's problem

$$\eta_T := \left[\|h_T \left(\operatorname{div}(a \bigtriangledown u_h) + f_h\right)\|^2_{0,2,T} + \sum_{E \in \mathcal{E}^0(T)} \|h_E^{\frac{1}{2}} \left[a \bigtriangledown u_h \cdot n_E\right]_E \|^2_{0,2,E} \right]^{\frac{1}{2}}. \tag{2}$$

For adaptive grid refinement, we will refine the elements contributing the top α% of the error. A good choice is $\alpha = 30$.

Task: Solve problem (1) for $f \equiv 0$ and

$$u^\partial(x) = \begin{cases} \left(x_1^2 + x_2^2\right)^{\frac{1}{3}} \sin\left(\frac{2}{3}\operatorname{atan2}(x_2, x_1)\right) & x \in \partial\Omega\,, \\ 0 & x \in \Omega\,. \end{cases} \tag{3}$$

Compare the numerical solution u_h to the exact solution u (the right hand side from equation (3) extended to the whole domain Ω) in the L^2- and the H^1-norm using adaptive and uniform grid refinement. Your program output should be two tables containing the number of elements and $\|u - u_h\|$ in the two norms for adaptive and uniform refinement, respectively.
The code that needs to be filled in is located in the following files:

exercise_3/ex3.cpp
: The handling of L and $\bar{u}^\partial$ needed for the solution of (1), the grid refinement and program output.

exercise_3/errorEstimator.h
: The evaluation of η_T and the code for element marking and refinement.

exercise_3/errorMeasurements.h
: The evaluation of the L^2- and the H^1-norm using center-of-mass quadrature.

The updated code framework and a new computational domain will be made available on the lecture website.

Note on programming task 1: In `rhs.h`, line 87 (`evalRHS()`) $\mathbf{2} \cdot$ rhs has to be returned despite the function name implying that the factor is not needed. A more convenient implementation can be achieved by changing line 79 of `rhs.h` to

```
RealType nl = 2.0 * el.getAreaOfFlattenedTriangle() * evalRHS(cartCoord);
```

thus moving the factor 2 out of `evalRHS()`.