Exercises for Tutorial00

Poisson Equation

Exercise 1 Poisson Equation with Piecewise Linear Elements

In this exercise you will:

- Work on the problem presented in the tutorial and see how the code works.
- Modify the implementation to solve a slightly different problem.
- Determine the convergence rate of the finite element method.

1. Warming up:
   The code presented in this tutorial solves the following elliptic equation with Dirichlet boundary conditions

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \partial\Omega, \end{aligned}$$

   where

$$f(x) = -2d \quad \text{and} \quad g(x) = \sum_{i=1}^{d} (x)_i^2.$$

   First of all check that $u(x) = \sum_{i=1}^{d} (x)_i^2$ is a solution for any dimension d.

   This program solves this equation for $d = 2$. The executable can be found in the build directory of this exercise:

```
cd dune-course/release-build/dune-pdelab-tutorials/
  ↪ tutorial00/exercise/task
```

   You can run the program with `./exercise00`. Some parameters of the program are controlled by the ini file `src_dir/tutorial00.ini`:

```
[grid]
refinement=4

[grid.twod]
filename=unitsquare.msh

[output]
filename=solution_4
```

Here you may set the mesh refinement and the names of the input and output files. Run the program with different refinement levels. Visualize the solution in ParaView, use the Calculator filter to visualize the difference $|u - u_h|$ and determine the maximum error. Note that u and u_h are called `exact` and `fesol` in the ParaView output.

2. Solving a new problem:
   Now consider

$$f(x) = -\sum_{i=1}^{d} 6(x)_i \quad \text{and} \quad g(x) = \sum_{i=1}^{d} (x)_i^3 \tag{1}$$

   and check that $u(x) = \sum_{i=1}^{d} (x)_i^3$ solves the PDE. Implement the new f and g, rebuild and rerun your program.

3. Analysis of finite element error:
   Produce a sequence of output files for different levels of mesh refinements $0, 1, 2, \dots$ with suitable output filenames. Visualize the error $|u - u_h|$ in ParaView and determine the maximum error on each level. If you want you can also try to calculate the L2-norm of the error in ParaView and determine the convergence rate by hand. In a later exercise you will see how to calculate the L2-norm directly in the c++ source.

4. Use a different solver:
   You may also try to exchange the iterative linear solver by replacing it with the following lines in the driver file `driver.hh`:

```
typedef Dune::PDELab::ISTLBackend_SEQ_BCGS_SSOR LS;
LS ls(5000,true);
```

   Compare the number of iterations which is given by the following lines in the output (here we have 12 iterations):

```
=== CGSolver
12       1.9016e-09
=== rate=0.144679, T=0.02362, TIT=0.00196833, IT=12
```

Exercise 2 Extending the Local Operator

Now consider the extended equation of the form

$$\begin{aligned} -\nabla \cdot (k(x)\nabla u) + a(x)u &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \partial\Omega, \end{aligned} \tag{2}$$

with scalar functions $k(x)$, $a(x)$.

1. In the first step show that the weak formulation of the problem involves the new bilinear form

$$a(u, v) = \int_\Omega k(x)\nabla u(x) \cdot \nabla v(x) + a(x)u(x)v(x)\, dx. \tag{3}$$

2. The main work is to extend the local operator given by the class `PoissonP1`. This can be done in several steps:
   - Copy the class `PoissonP1` to a new file and rename it.
   - Provide analytic functions for $k(x)$ and $a(x)$ and pass grid functions to the local operator like it is done for $f(x)$.
   - Extend the local operator to first handle $k(x)$ by assuming the function $k(x)$ to be *constant* on mesh elements.
   - Now extend the local operator to handle $a(x)u(x)$. Also assume the function $a(x)$ to be *constant* on mesh elements.
3. Plug in your new local operator in the driver code and test it. In the solution the code is tested for $k(x) = 2(x)_1$, $a(x) = \sum_{i=1}^{d}(x)_i$ and $g = \sum_{i=1}^{d}(x)_i^3$ by choosing the right hand side $f(x)$ in such a way that $u(x) = \sum_{i=1}^{d}(x)_i^3$ solves the PDE.