Exercises for Tutorial01

Poisson with Nonlinearity

The code of tutorial 01 solves the problem

$$\begin{aligned} -\Delta u + q(u) &= f && \text{in } \Omega, \\ u &= g && \text{on } \Gamma_D \subseteq \partial\Omega, \\ -\nabla u \cdot \nu &= j && \text{on } \Gamma_N = \partial\Omega \setminus \Gamma_D \end{aligned}$$

with the nonlinear function $q(u) = \eta u^2$. Compared to the tutorial code this exercise covers less variability (e.g. only 2D) for increased compilation speed. Still you can easily change the degree of the discretization, the grid, or the nonlinearity.

In this exercise you will get to know the code, play around with different settings and implement a different way to handle the Dirichlet boundary conditions.

Exercise 1 Warming Up

First we want to get familiar with the code.

1. Try out different grids, polynomial degrees and values for η. All these settings can be modified through the ini file `tutorial01.ini` and can be tested without rebuilding your program. Here are some suggestions that could be interesting:

   - Use higher values for η.
   - Run the code with the following setting:

```
[grid]
manager=yasp # set to ug | yasp
refinement=1 # be careful

[grid.structured]
LX=1.0
LY=1.0
NX=2
NY=2

[grid.twod]
filename=unitsquare.msh

[fem]
degree=1 # set to 1 | 2

[problem]
eta=2.0
```

```
[output]
filename=degree1_subsampling0
subsampling=1
```

Try all combinations of `degree=1|2` and `subsampling=1|2` with appropriate `filename`. Look at your solutions using ParaView and the `warp by scalar` filter. You can see the underlying grid by choosing `surface with edges` instead of `surface` in the ParaView drop down menu. How does subsampling change the output?

2. It is easy to implement different nonlinearities. Use $q(u) = \exp(\eta u)$ by adjusting the file `problem.hh`.

3. Go back to $q(u) = \eta u^2$. Now we want to see how good our approximation is. Change the function $f(x)$ in the file `problem.hh` to $f(x) = -2d + \eta(\sum_{i=1}^{d}(x)_i^2)^2$ where d is the dimension (and therefore size of x). Then $u(x) = \sum_{i=1}^{d}(x)_i^2 = g(x)$ is the exact solution. Visualize the exact solution like it is done in tutorial00. We start with the ini file:

```
[grid]
manager=yasp # set to ug | yasp
refinement=1 # be careful

...

[fem]
degree=1 # set to 1 | 2

[problem]
eta=100.0

[output]
filename=yasp_ref1
subsampling=1
```

Use ParaView to see how the maximal error $\max |u - u_h|$ behaves for different refinement levels `refinement=1|...|5`. Then try again for `degree=2`. What happens here? Does the behaviour change when you use `ug` instead of `yasp`?

Exercise 2 Nitsche's method for weak Dirichlet boundary conditions

In this exercise we want to implement Dirichlet boundary conditions in a weak sense by using Nitsche's method. Instead of incorporating the Dirichlet boundary condition into the Ansatz space we modify the residual:

$$r^{\text{Nitsche}}(u,v) = \int_\Omega \nabla u \cdot \nabla v + (q(u) - f)v \, dx + \int_{\Gamma_N} jv \, ds$$
$$- \int_{\Gamma_D} \nabla u \cdot \nu v \, ds - \int_{\Gamma_D} (u - g)\nabla v \cdot \nu \, ds + \eta_{stab} \int_{\Gamma_D} (u - g)v \, ds.$$

Here η_{stab} denotes a stabilization parameter that should be equal to $\eta_{stab} = c/h$ for a constant $c > 0$ large enough. This stabilization term is necessary to ensure coercivity of the bilinear form.

In order to implement this method you have to do the following:

- Go to `exercise01.cc` and include the new file `nitschenonlinearpoissonfem` instead of `nonlinearpoissonfem`.
- In the file `driver.hh` you have to turn off the constraints. The code is already there and you just have to comment/uncomment the parts marked with

```
//== Exercise 2 {
...
// ...
//== }
```

  to

```
//== Exercise 2 {
// ...
...
//== }
```

  By changing these lines you use no constraints, an empty constraints container and construct the grid operator without constraints. Besides that you use the new `NitscheNonlinearPoissonFEM` local operator that expects the stabilization parameter η_{stab}.

- The key part is adding the `alpha_boundary` method to the new local operator in the file `nitschenonlinearpoissonfem.hh`. Take a close look at the `lambda_volume`, `lambda_boundary` and `alpha_volume` methods and you should be on your way.

  Hint: The code for generating the transformation is already there:

```
// transform gradients of shape functions to real element
const auto S = geo_inside.jacobianInverseTransposed(local);
```

When you have done all that, test your implementation. Use the test case from exercise 1 with $f(x) = -2d + \eta(\sum_{i=1}^{d}(x)_i^2)^2$ and exact solution $u(x) = \sum_{i=1}^{d}(x)_i^2 = g(x)$ and compare it to your approximation.

Introduce the parameter

```
[fem]
stab = 100
```

in the ini file and look at the maximal error $\max|u - u_h|$ for `stab=10|100|1000`.