Numerische Methoden in der Strömungsmechanik
OvGU WiSe 2023/24

Practical Exercises 3

In this exercise we are going to numerically investigate the convergence behavior of the discrete solution u_h to the exact solution u in different norms.

Recalling Finite Elements course, theoretical error bounds read:

$$\begin{aligned} \|u - u_h\|_{L_2(\Omega)} &\leq Ch^{r+1}\|f\|, \\ \|u - u_h\|_{H^1(\Omega)} &\leq Ch^{r}\|f\|. \end{aligned}$$

Problem 3.1

At first we consider an inhomogeneous 2D Laplace problem on the unit square with zero Dirichlet boundary conditions:

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega = (0,1)^2, \\ u &= g \quad \text{on } \partial\Omega. \end{aligned} \tag{1}$$

In order to compute the discretization error we want to set the right-hand side f and the Dirichlet data g in such a way that the exact solution of system (1) is given by

$$u(x,y) := \sin(\pi x)\cos(\pi y) \qquad (x,y) \in \Omega.$$

Compute the corresponding right-hand side $f = -\Delta u$ analytically and implement this problem in GASCOIGNE by modifying the right-hand side and boundary data (use `M_PI` for an approximation of π). Solve and visualize the problem.

Write down derivatives of the exact solution u, namely ∇u. Soon you will need it for error evaluation.

Problem 3.2

Now, we are interested in different error norms.

The next part is to take control over the refinement loop, in the `main.cc` file you can find definition of `LocalLoop` class

```
class LocalLoop : public StdLoop {}
```

It inherits from `StdLoop` that implements diverse outer loops, e.g. mesh adaptation or time stepping. Please switch from `StdLoop` to `LocalLoop`, solve the problem again and compare solutions.

Problem 3.3

To compute functionals in Gascoigne we need to implement `DomainFunctional`. Please find in the `main.cc`

```
class H1norm : public virtual DomainFunctional {
```

Implement functionals to compute the errors in the L^2-norm $\|\cdot\|$, and H^1-norm $\|\nabla\cdot\|$ for every component of the solution u_h (in our case there is only one component).

In the exercise you can find pre-prepared code that is currently commented out.

In order to compute global errors, you need to sum the evaluation of error functional J on every element:

$$\|u-u_h\| = \left(\sum_{T\in\Omega_h} J(u-u_h) \right)^{1/2}$$

Please implement J method in your functional class, where you have access to your finite element solution U and spatial vertex v.

```
double J(const FemFunction& U, const Vertex2d& v) const
  { ... }
```

The parameters `FemFunction& U` and `Vertex2d& v` are the solution u and the point v. The solution `U` itself is a vector so that it holds $u(x,y) \sim$ `U[0].m()`. You can access the derivatives of the by $\partial_x u(x,y) \sim$ `U[0].x()` and $\partial_y u(x,y) \sim$ `U[0].y()`. Spatial coordinates of a point v are given by `v.x()` and `v.y()`.

The integration is done under the hood. At the end you need to apply square root to your result.

Compute these errors for different levels of mesh refinement and for both Q^1- and Q^2-elements. You can switch to Q^2-elements by changing `CGQ1` to `CGQ2` in the `gascoigne.param` file.

Write the computed errors to the files `convergence_q1.dat` and `covergence_q2.dat`. Which rate of convergence do you observe? Compare these results with the theoretical predictions.

You you can write this files directly in your code or copy and past it from the program output.

```
std::string filename = "convergence_q1.dat";
std::fstream s(filename, s.binary | s.trunc | s.in | s.out);
s << "#␣" << "h␣␣" << "h^1␣" << "l^2" << endl;
```

Problem 3.4

In this exercise, we visualize the results of problem 3.2 with gnuplot. Therefore, open gnuplot by typing `gnuplot` into the terminal. You will find a prompt `gnuplot>`. First, we create a logarithmic plot of the L^2-error over the mesh size h:

```
gnuplot> set logscale xy
gnuplot> plot 'convergence_q1.dat' using 1:2 title 'L2' with lp
```

It is possible to visualize the three computed norms concurrently in one plot:

```
gnuplot> plot 'convergence_q1.dat' using 1:2 title 'L2' with lp,
'convergence_q1.dat' using 1:3 title 'H1' with lp,
'convergence_q1.dat' using 1:4 title 'Linf' with lp
```

gnuplot can also be used to determine an estimate for the order of convergence via a least squares fit: Therefore, we define a function `f(x)` that describes the convergence with variable parameteres `A` and `B` and fit it to the data:

```
gnuplot> f(x) = A * (x ** B)
gnuplot> fit f(x) 'convergence_q1.dat' using 1:2 via A,B
```

The important output of the `fit.log` is

```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
A               = 0.478391         +/- 0.0004705    (0.09836%)
B               = 1.99301          +/- 0.0006817    (0.0342%)
```

that tells you that the estimated order of convergence is `B = 1.9930` ± `0.0007`.

Finally, check visually that the fit is good by plotting `f(x)` together with the data:

```
gnuplot> plot 'convergence_q1.dat' using 1:2 title 'L2', f(x)
```

Problem 3.5

Now, we want to consider the exact solution

$$u(x,y) := \sin(k\pi x)\cos(k\pi y) \qquad (x,y) \in \Omega$$

of system (1) where $k \in \mathbb{Z}$ will be read from the parameter file. Therefore, add a new block `Equation` to the parameter file:

```
//Block Equation
k  10
```

Note that this block has to appear before the dummy block `//Block End`.

This new data entry must be read within Gascoigne. We will combine all data that is required to define the problem in the class `MyProblemData` given in `problem.h`. Here, you can access the parameter file within the function `BasicInit(const ParamFile &pf)`. To read from the file, insert the following lines to `MyProblemData::BasicInit(...)`

```
DataFormatHandler DFH;
DFH.insert("k",&k, 1);
FileScanner FS(DFH);
FS.readfile(pf, "Equation");
```

and, naturally, you'll have to define a variable `double k` and add it to the class `MyProblemData`. Later on, all classes that know the problem data class, such as `MyRightHandSide` or `H1Norm`, have access to this variable via `data.k`.

Since we change the exact solution, calculate the corresponding right-hand side $f = -\Delta u$ again analytically and make the necessary changes in the `operator()` functions of `MyExactSolution` and `MyRightHandSide`. Here, you can access the value of `k` by `data.k`.

Solve and visualize the problem for $k \in \{10, 50\}$. Try to interprete the convergence behavior. You will have to use fine meshes!

Problem 3.6 (Extra)

In most situations the analytical solution is unknown. In this case, one can only compare coarse solutions to a reference solution $\tilde{u}$ computed on a very fine mesh.

In this exercise we are interested in the error measured in the H^1-seminorm $\|\nabla \cdot\|_{L^2(\Omega)}$. Even if we approximate $\|\nabla(u - u_h)\|_{L^2(\Omega)} \approx \|\nabla(\tilde{u} - u_h)\|_{L^2(\Omega)}$ a direct calculation of this error needs interpolation between solutions on different meshes. For this reason, use the Galerkin orthogonality to show the following identity:

$$\begin{aligned}\|\nabla(u - u_h)\|_{L^2(\Omega)} &= \left(\left|\|\nabla u\|^2_{L^2(\Omega)} - \|\nabla u_h\|^2_{L^2(\Omega)}\right|\right)^{1/2} \\ &\approx \left(\left|\|\nabla \tilde{u}\|^2_{L^2(\Omega)} - \|\nabla u_h\|^2_{L^2(\Omega)}\right|\right)^{1/2}.\end{aligned}$$

a) Use the L-shaped domain from exercise sheet 2 and set the right-hand side equal to one, i.e. $f = 1$. This problem has a solution which is unknown to us. Compute a reference solution on a very fine mesh. Calculate its H^1-seminorm using the error computations from the previous exercises and setting the "exact solution" to 0.

b) Use the formula above to compute the error.

   For this, finish the implementation Use `cout` to print the error to the terminal.

c) Run the program on different mesh levels and for Q^1- and Q^2-elements. Which rate of convergence do you observe? Try to explain your observations.