Course Project – Part I

NUMN33/FMNN45(F): Numerical Methods for PDEs

Eskil Hansen, Robert Klöfkorn

This is Part I of the overall course project which is based on DUNE and DUNE-FEM in your own Python environment.
In this part, we consider elliptic partial differential equations and investigate various properties. The project consists of both theoretical and practical exercises (tasks).

Please describe you solutions and observation in report form. For the practical exercises it should include

- a description of the problem considered,
- a description of your test cases in such a way so that they are reproducible, and
- a description of your tests, i.e. what is tested and why ...,
- your own interpretation of the results.
- Include important parts of your code as code snippets or append the entire code at the of the report.

Displaying code in Latex documents can, for example, be done like in this template: https://gitlab.maths.lu.se/robertk/thesislatextemplate. Many other examples can be found online.

Grading:

- All "non-living" and "non-generative AI" help is allowed, as long as proper references are given and you are able to explain the results during the oral exam. The use of generative AI is discouraged.
- **NUMN33:**
  - Minimum of 4 points in each section (theoretical/practical) and minimum 12 points in total for grade **G**.
  - Grade **G** and a minimum of 18 points total for a grade **VG**.
- **FMNN45:**
  - 4 points in each section (theoretical/practical) and minimum 12 points in total for a grade **3**.

  - Grade **G** and a minimum 16 points total for a grade **4**.
  - Grade **G** and a minimum 20 points total for a grade **5**.
- **FMNN45F:**
  - 6 points in each section (theoretical/practical) and minimum 16 points in total for grade **G**.

This assignment has **14 tasks** and **24 points** in total.

Theoretical Exercises

Let Ω be a bounded domain in $\mathbb{R}^d$ with a smooth boundary $\partial\Omega$.

Task 1 (2 pts)

Give the definitions of the Sobolev spaces $H^1(\Omega)$ and $H_0^1(\Omega)$.

Task 2 (2 pts)

Let $u \in C^1([a,b])$. Check that the classical derivative u' is also a weak derivative.

Task 3 (4 pts)

Prove that the weak derivative is unique on $L^2(\Omega)$.
Hint: $C_0^\infty(\Omega)$ is dense in $L^2(\Omega)$.

Task 4 (4 pts)

Prove that $u \mapsto ||\nabla u||_{L^2(\Omega)}$ and $u \mapsto ||u||_{H^1(\Omega)}$ are equivalent norms on $H_0^1(\Omega)$.

Practical Exercises

Grid Construction

Task 5 (1 pt)

Given $\Omega := [-\frac{1}{2}, 2] \times [-\frac{1}{2}, 1] \subset \mathbb{R}^2$. Setup a Cartesian grid that tessellates Ω with 10 cells in the x direction and 20 in the y direction.

Task 6 (1 pt)

Tessellate Ω with a triangular grid with twice as many cells as the Cartesian grid.

Task 7 (1 pt)

Plot both grids.

Task 8 (1 pt)

Create a triangular grid of the above domain Ω with a hole inside. You can either define the triangles yourself or use `pygmsh` for that purpose. More explanation can be found here: https://dune-project.org/sphinx/content/sphinx/dune-fem/othergrids_nb.html

Task 9 (2 pts)

Compute the grid width h of the created grid with

$$h := \max_{E \in \mathcal{T}_h} h_E.$$

h_E can be chosen in different ways. Choose one and justify your choice. Compare your results with

```
from dune.fem.utility import gridWidth
h = gridWidth( gridView )
```

Grid Functions

Task 10 (1 pt)

Write a piecewise constant grid function that on each element E returns h_E.

Task 11 (1 pt)

Create a degrees of freedom (dof) mapper with **vertex layout**. Print the number of dofs and compare with the grid's number of vertices.

Linear Interpolation

We compute the linear interpolation of a grid function u over a triangular grid.
We need to attach one degree of freedom to each vertex p which corresponds to the value of the function at p which we store in a `numpy array`.
On each triangle with vertices p_0, p_1, p_2 the discrete function is now given by the unique linear function taking the values $u(p_0), u(p_1), u(p_2)$ at the three vertices. This function is

$$u_h(e, \hat{x}) = u(p_0)(1 - \hat{x}_0 - \hat{x}_1) + u(p_1)\hat{x}_0 + u(p_2)\hat{x}_1$$

where $\hat{x}_0, \hat{x}_1$ are the coordinates in the reference element $\hat{E}$.

Task 12 (1 pt)

Write a function `linearInterpolation` that takes a grid function `u` and a `gridView` and returns the `mapper` and vector of degrees of freedom (`dofs`), i.e., the `numpy` `array` storing the values of u at the vertices.

Note: note that a grid function has an attribute `gridView`.

Then write a grid function u_h that returns the linear interpolation for the given dof vector and mapper.

Use the unstructured triangular grid from above and refine it 3 times globally.

Task 13 (2 pts)

And finally, let's compute the maximum interpolation error by comparing the linear interpolation u_h at the center of each element with the value of u. Write a grid function that returns the error for each element E and print the maximal value. Compute

$$e = \max_{E \in \mathcal{T}_h} |u(\omega_E) - u_h(\omega_E)|,$$

where ω_E is the center of an element E.

Task 14 (1 pt)

Compare the error on two different refinement levels. Is there a relation to h?

Lycka till!