Numerical Approximation of PDEs

Spring Semester 2025

Lecturer: Prof. Annalisa Buffa Assistant: Mohamed Ben Abdelouahab

Session 4: March 20, 2025

Exercise 1. Suppose that $\mathcal{T}$ is a triangulation of a domain $\Omega \subseteq \mathbb{R}^2$ and that $V_h \subset H^1(\Omega)$ is the first-order finite element space with respect to that triangulation. Show that there exists a constant $C > 0$ such that for every vertex x of the triangulation and every $v \in V_h$:

$$|v(x)| \leq Ch^{-1}\|v\|_{L^2(\Omega)}.$$

Here, h is the maximum diameter of the cells.

Exercise 2. The so-called **barycentric coordinates** are a popular tool in finite element methods. Consider the reference triangle $\hat{K}$ with vertices $\hat{v}_0 = (0,0)^T$, $\hat{v}_1 = (1,0)^T$ and $\hat{v}_3 = (0,1)^T$, as sketched in the lecture / lecture notes. The barycentric coordinates are linear functions $\hat{\lambda}_0, \hat{\lambda}_1, \hat{\lambda}_2$ that satisfy

$$\hat{\lambda}_i(v_j) = \delta_{ij}.$$

(In other words, they coincide with the hat functions.)

1. Check that $\hat{\lambda}_0 = 1 - x - y$, $\hat{\lambda}_1 = x$, and $\hat{\lambda}_2 = y$ on the reference triangle. Show that for any point $\hat{v} \in \hat{K}$ we have

   $$\hat{v} = \hat{\lambda}_0(v)\hat{v}_0 + \hat{\lambda}_1(v)\hat{v}_1 + \hat{\lambda}_2(v)\hat{v}_2.$$

   (This is why they are called "coordinates".)

2. Show that a basis of $P_2(\hat{K})$ is given by

   $$\hat{\lambda}_0\hat{\lambda}_0,\ \hat{\lambda}_0\hat{\lambda}_1,\ \hat{\lambda}_0\hat{\lambda}_2,\ \hat{\lambda}_1\hat{\lambda}_1,\ \hat{\lambda}_1\hat{\lambda}_2,\ \hat{\lambda}_2\hat{\lambda}_2.$$

Exercise 3. A general definition of the p-th order polynomial space $P_p(\hat{K})$ over $\hat{K}$ is the space of p-th order polynomials with canonical basis $\{\hat{x}^i\hat{y}^j \,|\, i + j \leq p\}$, where we define $N_p := \dim P_p(\hat{K})$. Let $\hat{X} = (0, \frac{1}{p}, \frac{2}{p}, \ldots, \frac{p-1}{p}, 1)$. A particularly useful FEM basis for this space is the basis $\{\hat{\phi}_1, \ldots, \hat{\phi}_{N_p}\}$ that satisfies $\hat{\phi}_i(\hat{p}_j) = \delta_{ij}$, where $\hat{p}_j \in \{(\hat{x}, \hat{y}) \in \hat{X} \times \hat{X} \,|\, \hat{x} + \hat{y} \leq 1\}$.

1. Show that $N_p := \dim P_p(\hat{K}) = \frac{p^2+3p+2}{2}$.

2. Derive a linear system of equations to find the weights $\{a_{jk}\}$ of $\hat{\phi}_i = \sum_{j,k} a_{jk}\,\hat{x}^j\hat{y}^k$ given that $\hat{\phi}_i(\hat{p}_j) = \delta_{ij}$.

3. On moodle you will fine the python template `code_04_03_template.py` which you can use to implement an algorithm capable of finding the a_{jk} of each $\hat{\phi}_i$.

Exercise 4. [Building local stiffness and mass matrices] We use reference transformations to compute the local matrices over triangles.

1. Compute the local stiffness matrix for an arbitrary triangle K

$$\left(A^{loc,K}\right)_{i,j} = \int_K \nabla\varphi_i \cdot \nabla\varphi_j \, dx\, dy, \qquad i,j = 1,2,3$$

where φ_i, φ_j are the $\mathbb{P}_1$ Lagrange basis functions in 2D.
Hint: derive expressions for the linear map

$$\begin{array}{rccc} F_K: & \hat{K} & \to & K \\ & \hat{x} & \mapsto & B_K\hat{x} + b_K \end{array}$$

where $\hat{K}$ is the reference element and $B_K \in \mathbb{R}^{2\times 2}$ and $b_K \in \mathbb{R}^{2\times 1}$. Then compute the local matrix $A^{loc,K}$ by recasting the integral over the reference element, as discussed in the lecture. Note that the ϕ_i and their local counterparts have a constant gradient !

2. Compute the local mass matrix for an arbitrary triangle K

$$\left(M^{loc,K}\right)_{i,j} = \int_K \varphi_i\varphi_j \, dx\, dy, \qquad i,j = 1,2,3$$

where φ_i, φ_j are the Lagrange basis functions, by recasting the integral over the reference element.

3. Implement this as a Python code to assemble the full matrices. On Moodle, you find Python codes `mesh.py` and `code_04_04_template.py`. The file `mesh.py` provides you with a ready-to-go mesh class that you can use. [1] The file `code_04_04_template.py` provides you two functions: `stiffness_matrix` and `mass_matrix`. You can complete these functions. These take a mesh as input and produce the respective matrices.

4. In the file `code_04_04_template.py` you also find the method `load_vector`. Implement this function to compute the load vector in the case of a piecewise constant right hand side. The function takes as input: the mesh, and the constant value F of the right-hand side.

[1] This library relies on pygmsh, which should be easy to install with the command `pip install pygmsh`. To use the code **you do not need to understand the mesh.py, util.py and solve.py files in detail. Everything is explained in the template script.** Once you have finalised your implementation, you may run the script and it will plot a mesh and the solution of reaction-diffusion benchmark problem for you making use of the implemented matrices.