TP 2 : Assembly Algorithm

BUI Minh Thi - PHAM Hieu
INSA Toulouse
Finite Element
2023-2024
4GMM

Let Ω be an open bounded subset of $\mathbb{R}^2$ with polygonal boundaries. Consider the following problem with Neumann boundary conditions: Find $u \in H^1(\Omega)$ such that

$$\left\{ \begin{array}{rcll} -\Delta u + u &=& f & \text{in } \Omega, \\ \partial_n u &=& 0 & \text{on } \partial\Omega, \end{array} \right. \tag{1}$$

Continuous Problem

1. Write the continuous variational formulation of this problem.

Let $v \in H^1(\Omega)$ a test function:

$$\int_\Omega (-\Delta u + u)v = \int_\Omega fv,$$

Applying Green's formula, we have:

$$\int_\Omega \nabla u \cdot \nabla v - \int_{\partial\Omega} (\nabla u \cdot n)v + \int_\Omega uv = \int_\Omega fv,$$

Where the boundary term vanishes due to the Neumann condition:

$$\int_\Omega \nabla u \cdot \nabla v + \int_\Omega uv = \int_\Omega fv,$$

Hence, the continuous variation formulation of this problem is

$$u \in H^1(\Omega) \quad \text{such that} \int_\Omega \nabla u \cdot \nabla v + \int_\Omega uv = \int_\Omega fv \quad \forall v \in H^1(\Omega)$$

Discrete problem

We want to implement the discrete scheme using P_1 Lagrange Finite Element in Python. Let $\mathcal{T}_h$ be an admissible mesh of Ω and V_h the approximation space of $H^1(\Omega)$. We denote by $(T_\ell)_{\ell=1..N_T}$ the triangles of this mesh $\mathcal{T}_h$, $(M_i)_{i=1..N}$ the nodes of this mesh and $(\varphi_i)_{i=1..N}$ the basis elements of V_h.

2. Write the discrete variational formulation solved by the discrete solution $u_h \in V_h$.

$$V_h \subset C^0(\overline{\Omega}) \cap H^1(\Omega)$$

$$V_h = \left\{ v \in C^0(\overline{\Omega}) \mid v_{h|T} \in P_1(T), \forall T \in \mathcal{T}_h \right\}$$

So $\dim(V_h) = N$ (number of mesh nodes). The basis of V_h is $\{\varphi_i\}_{i=1..N}$ where:

$$\varphi_i(x) = \begin{cases} \lambda_i^T(x) & \text{if } x \in T \text{ with } M_i \in T \\ 0 & \text{otherwise} \end{cases}$$

Note u_h the approximate solution of u, the discrete variation formulation of this problem is:

$$u_h \in V_h \quad \text{such that} \int_\Omega \nabla u_h \cdot \nabla v_h + \int_\Omega u_h v_h = \int_\Omega f v_h \quad \forall v_h \in V_h$$

3. By construction, this solution is expressed as

$$u_h(x,y) = \sum_{i=1..N} u_h(M_i)\varphi_i(x,y) \quad \forall (x,y) \in \Omega.$$

Express the discrete problem as an equivalent linear system

$$(A + \mathcal{M})U = F,$$

with $U = (u_h(M_i))_{i=1..N} \in \mathbb{R}^N$ the unknown vector, $\mathcal{M}$ the mass matrix and A the stiffness matrix. The discrete solution u_h can be expressed as:

$$u_h = \sum_{i=1}^{N} \alpha_i \varphi_i$$

where a_j are the coordinates of u_h in V_h. At the mesh nodes M_j, we have:

$$u_h(M_j) = \sum_{i=1}^{N} \alpha_i \varphi_i(M_j) = \alpha_j$$

Thus, we can write:

$$u_h = \sum_{i=1}^{N} u_h(M_i)\varphi_i = \sum_{i=1}^{N} U_i \varphi_i$$

The discrete variational formulation is then:

$$\sum_{i=1}^{N} u_h(M_i) \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j + \sum_{i=1}^{N} u_h(M_i) \int_\Omega \varphi_i \varphi_j = \int_\Omega f \varphi_j \quad \forall j = 1, \ldots, N$$

This can be written in matrix form as:

$$(A + \mathcal{M})U = F$$

where:

$$U = \begin{bmatrix} u_h(M_1) \\ \vdots \\ u_h(M_N) \end{bmatrix}$$

and

$$F_i = \int_\Omega f \varphi_i$$

The stiffness matrix A and the mass matrix $\mathcal{M}$ are also defined by:

$$(A_{ij})_{1 \le i,j \le N}, \quad A_{ij} = \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j$$

$$(\mathcal{M}_{ij})_{1 \le i,j \le N}, \quad \mathcal{M}_{ij} = \int_\Omega \varphi_i \varphi_j$$

Assembly

We want to solve this problem on a square. In the given code, we have already implemented a structured mesh on this square. It only remains to implement the two matrices $\mathcal{M}$ and A. To do so, we recall the assembly algorithm.

```
$\mathcal{M} = 0,\ A = 0$
**For all** $\ell = 1, N_T$
    Get nodes coordinates of $T_\ell$
    Compute local matrices $A^{T_\ell}$
    **For** $i = 1, 3$
        $I = \text{local} \rightarrow \text{global}(\ell, i)$
        **For** $j = 1, 3$
            $J = \text{local} \rightarrow \text{global}(\ell, j)$
            $\mathcal{M}_{IJ} += \mathcal{M}^{T_\ell}_{ij},\ A_{IJ} += A^{T_\ell}_{ij}$
    **End** $\ell$
```

4. What is the name of the connectivity class in the code? How do you obtain the coordinate of a node numbered i of your mesh?

   - The code connectivity class is $TabTri$, where we can access the global indices of the vertices (nodes) of the triangle through the triangle index. Thus, we can connect the local matrix to the global matrix, i.e. obtain the global vertices' indices through $TabTri$ if we already know the triangle's index and the local vertices' indices.
   - To obtain the coordinates of nodes numbered i in the mesh, we use the $TabSom$ table which contains the coordinates of all vertices.

   Here is an example for triangle l:

```
# Get the indices of triangle’s vertices
nodes = TabTri[l]
# Get the coordinates of triangle’s vertices
x1, y1 = TabSom[nodes[0]]
x2, y2 = TabSom[nodes[1]]
x3, y3 = TabSom[nodes[2]]

# Calculate the local matrices for this triangle
Me_local, Ae_local = local_matrices_p1(x1, y1, x2, y2, x3, y3)

# Assemble global matrices
for i in range(3):  # Loop over local nodes
    I = nodes[i]  # Global index
    for j in range(3):
        J = nodes[j]  # Global index
        M[I, J] += Me_local[i, j]  # Assemble global mass matrix
        A[I, J] += Ae_local[i, j]  # Assemble global stiffness matrix
```

5. Complete the assembly algorithm in the code.

   The detailed code can be found in the link to a GitHub repository in Code Availability.

   Now it remains to compute the local matrices. Let consider a triangle T with nodes $M_1(x_1, y_1)$, $M_2(x_2, y_2)$, and $M_3(x_3, y_3)$.

The local mass matrix $\mathcal{M}^T$ and the local stiffness matrix A^T are defined as:

$$\mathcal{M}_{ij}^T = \int_T \varphi_i \varphi_j = \int_T \lambda_i \lambda_j$$

$$A_{ij}^T = \int_T \nabla \varphi_i \cdot \nabla \varphi_j = \int_T \nabla \lambda_i \cdot \nabla \lambda_j$$

Each of these coefficients are then computed using the general formulas of barycentric coordinates :

$$\lambda_1(x,y) = \frac{1}{2|T|}\left[(y_2 - y_3)(x - x_3) - (x_2 - x_3)(y - y_3)\right]$$

$$\lambda_2(x,y) = \frac{1}{2|T|}\left[(y_3 - y_1)(x - x_1) - (x_3 - x_1)(y - y_1)\right]$$

$$\lambda_3(x,y) = \frac{1}{2|T|}\left[(y_1 - y_2)(x - x_2) - (x_1 - x_2)(y - y_2)\right]$$

where $|T|$ is the triangle area and $\lambda_1, \lambda_2, \lambda_3$ are the three barycentric coordinates of T.

- For the local mass matrix $\mathcal{M}^T$, the calculi are simple by using the general formula of the integration of barycentric coordinate on a triangle T:

  $$\int_T \lambda_1^k \lambda_2^l \lambda_3^n = \frac{2|T|\, k!\, l!\, m!}{(k+l+m+2)!}$$

  For example:

  $$\int_T \lambda_1^2 = \frac{|T|}{6}, \quad \int_T \lambda_1 \lambda_2 = \frac{|T|}{24}$$

  So, the local mass matrix $\mathcal{M}^T$ is:

  $$\mathcal{M}^T = \frac{|T|}{12}\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}$$

- For the local stiffness matrix A^T, we need to compute the gradients of these functions.
  For example, the gradient of the function $\nabla \lambda_1(x)$ is:

  $$\nabla \lambda_1(x) = \frac{1}{2|T|}\begin{bmatrix} y_2 - y_3 \\ -(x_2 - x_3) \end{bmatrix}$$

  Similarly, the gradient of the function $\nabla \lambda_2(x)$ is:

  $$\nabla \lambda_2(x) = \frac{1}{2|T|}\begin{bmatrix} y_3 - y_1 \\ -(x_3 - x_1) \end{bmatrix}$$

  The integral of the product of the gradients is:

  $$\int_T \nabla \lambda_1 \cdot \nabla \lambda_2 = \frac{1}{4|T|^2}|T|\left[(y_2 - y_3)(y_3 - y_1) + (x_2 - x_3)(x_3 - x_1)\right]$$

  So, the local stiffness matrix A^T can be calculated as:

  $$A^T = \frac{|T|}{4}\begin{pmatrix} y_2 - y_3 & x_3 - x_2 \\ y_3 - y_1 & x_1 - x_3 \\ y_1 - y_2 & x_2 - x_1 \end{pmatrix}\begin{pmatrix} y_2 - y_3 & y_3 - y_1 & y_1 - y_2 \\ x_3 - x_2 & x_1 - x_3 & x_2 - x_1 \end{pmatrix}$$

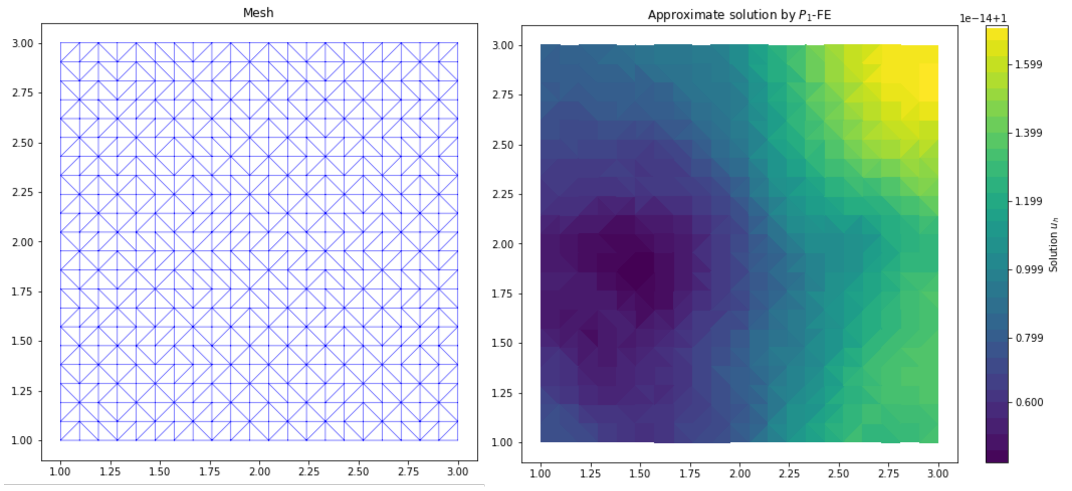
Right hand side

It now remains to implement the right hand side F.

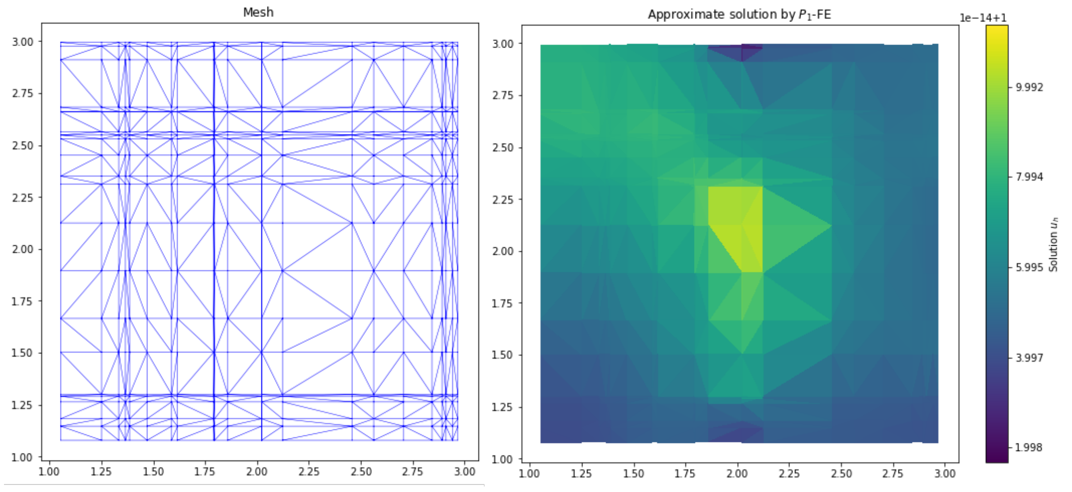
6. First, we assume that $f = 1$. Observing that $f \in V_h$, write the right hand side F using the mass matrix $\mathcal{M}$.

   If $f = 1$, remind that $\varphi_i(M_j) = \delta_{ij}$, the integral F_i can be calculated through the mass matrix $\mathcal{M}$ as follows:

$$F_i = \int_\Omega f\varphi_i = \int_\Omega \varphi_i = \sum_{j=1}^{N} \int_\Omega \varphi_j\varphi_i = \sum_{j=1}^{N} \mathcal{M}_{ij}$$



(a) Uniform mesh



(b) Unstructured mesh

Figure 1: Approximation solution u_h by $P_1 - FE$ in $\Omega = [1,3] \times [1,3]$ for $f = 1$

   In both two cases, we can conclude that the solution is $u = 1$. By substituting this into (1), we verify $f = 1$ and $\partial_n u = 0$.

7. Recall the definition of F for a general data $f \in C^0(\Omega)$. Approximating f by $I_h^{(1)} f$ (its interpolation using P_1 Lagrange FE), give an approximation of the right hand side using the mass matrix. We will assume that replacing f by $I_h^{(1)} f$ with the P_1-FE does not change the result.

   If $f \in C^0(\overline{\Omega})$, then $I_h^{(1)}(f) = \sum_{j=1}^{N} f(M_j)\varphi_j$ is the interpolation of f using P_1 Lagrange finite elements, where $\mathcal{M}$ is the mass matrix.

The integral F_i can be approximated as follows:

$$F_i = \int_\Omega f\varphi_i \approx \int_\Omega I_h^{(1)}(f)\varphi_i = \sum_{j=1}^{N} f(M_j)\int_\Omega \varphi_j\varphi_i = \mathcal{M}_i \cdot F_{values}$$

where:

$$F_{values} = \begin{bmatrix} f(M_1) \\ \vdots \\ f(M_N) \end{bmatrix}$$

Hence, an approximation of right hand side is $F \approx \mathcal{M} \cdot F_{values}$

The error bound for the approximation can be expressed as:

$$\text{Error}_{\text{approx}} = \left| \int_\Omega f\varphi_i - \int_\Omega I_h^{(1)}(f)\varphi_i \right| \leq \left\| f - I_h^{(1)}(f) \right\|_{L^2(\Omega)} \|\varphi_i\|_{L^2(\Omega)}$$

Using the approximation properties, this can be further bounded by:

$$\left\| f - I_h^{(1)}(f) \right\|_{L^2(\Omega)} \leq ch^2 |f|_{H^2(\Omega)}$$

where c is a constant, h is the mesh size.

Hence, it is reasonable to assume that replacing f by $I_h^{(1)} f$ with the P_1-FE does not change the result.

Validation

We want to verify that our code gives a 'good' solution u_h. To do so, we solve our continuous problem with the solution $u(x,y) = \cos(\pi x)\cos(\pi y)$ for all $(x,y) \in \Omega$ on the unit square.

8. Compute the corresponding right hand side f.

The exact continuous solution is given by:

$$u(x,y) = \cos(\pi x)\cos(\pi y)$$

Substitute this into the equation $-\Delta u + u = f$ to find f:

$$-\Delta(\cos(\pi x)\cos(\pi y)) + \cos(\pi x)\cos(\pi y) = f(x,y)$$

The gradient of u is:

$$\nabla u = \begin{pmatrix} -\pi \sin(\pi x)\cos(\pi y) \\ -\pi \cos(\pi x)\sin(\pi y) \end{pmatrix}$$

The Laplacian of u, Δu is:

$$\Delta u = -\pi^2 \cos(\pi x)\cos(\pi y) - \pi^2 \cos(\pi x)\cos(\pi y) = -2\pi^2 \cos(\pi x)\cos(\pi y)$$

Thus,

$$f(x,y) = (2\pi^2 + 1)\cos(\pi x)\cos(\pi y)$$

For the boundary conditions, consider the domain $\Omega = [1,3] \times [1,3]$:

$$\Gamma_1 : \{(x,y) \in \mathbb{R}^2 \mid y = 1, x \in [1,3]\}$$

$$\Gamma_2 : \{(x,y) \in \mathbb{R}^2 \mid x = 3, y \in [1,3]\}$$

At Γ_1:

$$\nabla u|_{\Gamma_1} = \begin{pmatrix} \pi \sin(\pi x) \\ 0 \end{pmatrix}, \quad \vec{n}_{\Gamma_1} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$$

Therefore,

$$\nabla u|_{\Gamma_1} \cdot \vec{n}_{\Gamma_1} = 0$$

At Γ_2:

$$\nabla u|_{\Gamma_2} = \begin{pmatrix} 0 \\ \pi \sin(\pi y) \end{pmatrix}, \quad \vec{n}_{\Gamma_2} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Hence,

$$\nabla u|_{\Gamma_2} \cdot \vec{n}_{\Gamma_2} = 0$$

The given solution is satisfied with boundary condition:

$$\partial_n u = 0$$

9. Compute the error term $\|I_h^{(1)}(u) - u_h\|_{L^2(\Omega)}$ using the mass matrix $\mathcal{M}$. Plot the quantity $\log(\|I_h^{(1)}(u) - u_h\|_{L^2(\Omega)}/\|u\|_{L^2(\Omega)})$ with respect to $\log 1/h$ with h the mesh size.

   The L^2 norm of the error for the interpolation solution $I_h^{(1)}(u)$ and the finite element solution u_h is computed by:

$$\|I_h^{(1)}(u) - u_h\|^2_{L^2(\Omega)} = \int_\Omega (I_h^{(1)}(u) - u_h)^2 \, d\Omega$$

   Given:

$$u_h = \sum_{i=1}^N U_i \varphi_i, \quad I_h^{(1)}(u) = \sum_{i=1}^N u(M_i)\varphi_i$$

   Thus, the error norm is:

$$\left\| I_h^{(1)}(u) - u_h \right\|^2_{L^2} = \int_\Omega \left(\sum_{i=1}^N \left(u(M_i) - U_i\right) \varphi_i \right) \cdot \left(\sum_{j=1}^N \left(u(M_j) - U_j\right) \varphi_j \right)$$

   This simplifies to:

$$\left\| I_h^{(1)}(u) - u_h \right\|^2_{L^2} = \sum_{i,j} \left(u(M_i) - U_i\right)\left(u(M_j) - U_j\right) \int_\Omega \varphi_i \cdot \varphi_j$$

   This can be approximated using the mass matrix $\mathcal{M}$:

$$\|I_h^{(1)}(u) - u_h\|^2_{L^2} = (I_h^{(1)}(u) - u_h)^T \mathcal{M} (I_h^{(1)}(u) - u_h)$$

10. Compute the error term $\|\nabla I_h^{(1)}(u) - \nabla u_h\|_{L^2(\Omega)}$ using the stiffness matrix A. Plot the quantity $\log(\|\nabla I_h^{(1)}(u) - \nabla u_h\|_{L^2(\Omega)}/\|\nabla u\|_{L^2(\Omega)})$ with respect to $\log 1/h$.

    The H^1 semi-norm error is computed for the gradient of the interpolation $I_h^{(1)}(u)$ and u_h:

$$\left| I_h^{(1)}(u) - u_h \right|^2_{H^1} = \|\nabla I_h^{(1)}(u) - \nabla u_h\|^2_{L^2(\Omega)^2} = \int_\Omega (\nabla I_h^{(1)}(u) - \nabla u_h)^2 \, d\Omega$$

    The error term involving the gradient can be expressed as:

$$\left| I_h^{(1)}(u) - u_h \right|^2_{H^1} = \int_\Omega \left(\sum_{i=1}^N \left(u(M_i) - U_i\right) \nabla\varphi_i \right) \cdot \left(\sum_{j=1}^N \left(u(M_j) - U_j\right) \nabla\varphi_j \right)$$

This simplifies to:

$$\left|I_h^{(1)}(u) - u_h\right|_{H^1}^2 = \sum_{i,j} (u(M_i) - U_i)\,(u(M_j) - U_j) \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j$$

In matrix form, it can be written as:

$$\left|I_h^{(1)}(u) - u_h\right|_{H^1}^2 = (I_h^{(1)}(u) - u_h)^T A (I_h^{(1)}(u) - u_h)$$

where A is the stiffness matrix with entries $A_{ij} = \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j$.

11. Applying numerical value

Below, we plot the quantity $\log\left(\frac{\|I_h^{(1)}(u)-u_h\|_{L^2(\Omega)}}{\|u\|_{L^2(\Omega)}}\right)$ and $\log\left(\frac{\|\nabla I_h^{(1)}(u)-\nabla u_h\|_{L^2(\Omega)}}{\|\nabla u\|_{L^2(\Omega)}}\right)$ with respect to $\log\left(\frac{1}{h}\right)$. In this case, we use the uniform mesh.

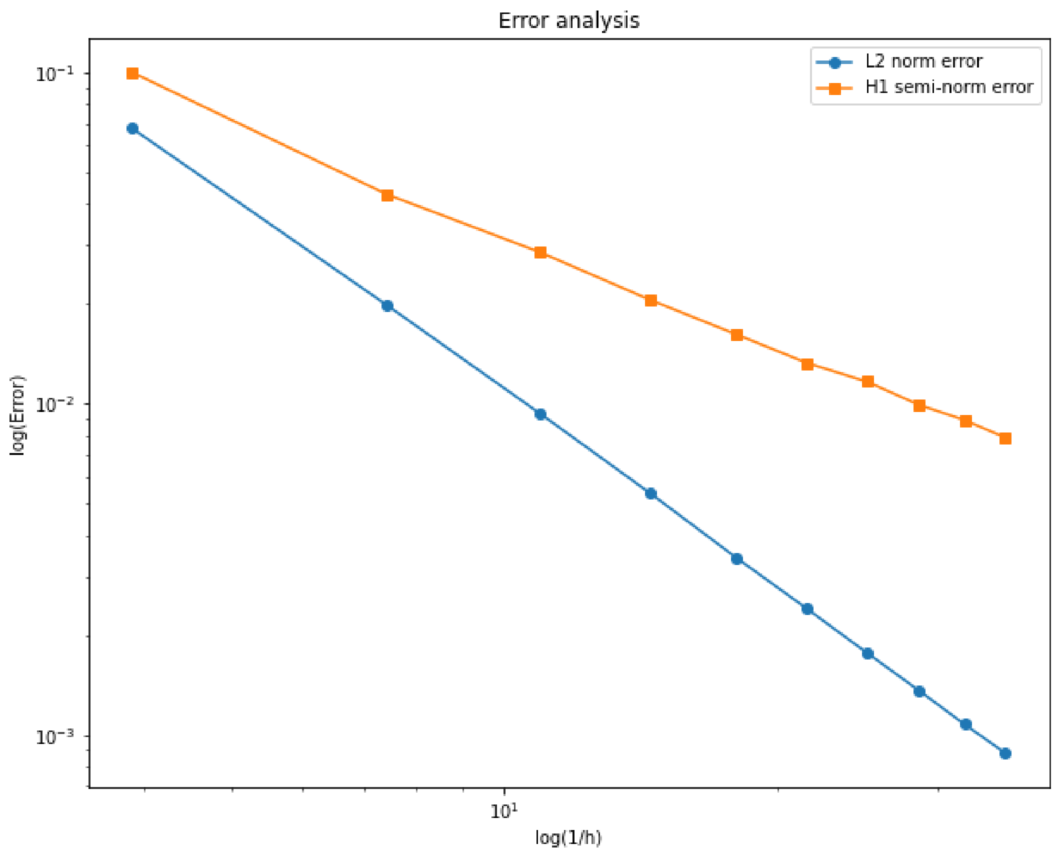


Figure 2: Error analysis

From the figure, we observe that the quantity of $\log(Error)$ with respect to $\log\left(\frac{1}{h}\right)$ is nearly a straight line with a different slope depending on the norm.

Moreover, we have the slope of L^2 norm error line is -1.9675565192571387, which means that the order of the convergence of the error in the L2 norm is nearly 2.

Similarly, the slope of H^1 semi-norm error line is -1.1262150280794765, which means that the order of the convergence of the error in the H^1 semi-norm is nearly 1.

This result is compatible with the theory of the interpolation error :

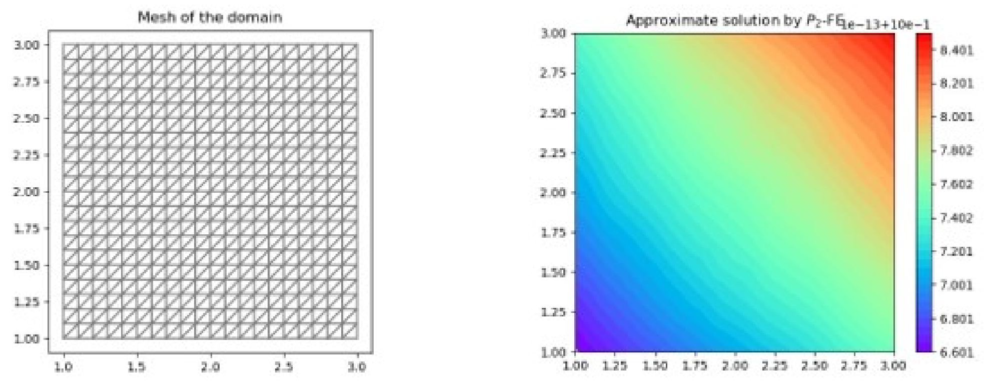
Theorem (Interpolation Error): Let $\mathcal{T}_h$ be a regular mesh of Ω and $v \in H_0^1(\Omega) \cap H^2(\Omega)$. Then, there exists $c > 0$ independent of the mesh size h and of v such that

$$\left\|I_h^{(1)}(v) - v\right\|_{L^2(\Omega)} + h\left|I_h^{(1)}(v) - v\right|_{H^1(\Omega)} \leq ch^2 \|v\|_{H^2(\Omega)}$$

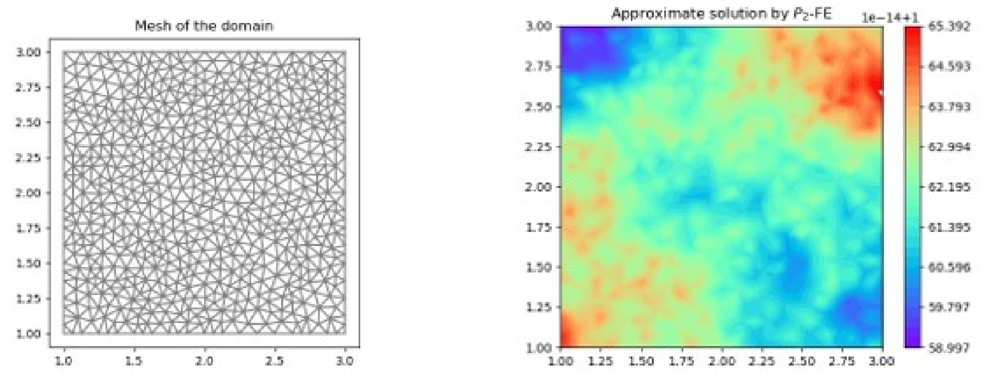
Extensions

12. P_2 Lagrange Finite Element

    By using the FEM library in Python, we can solve this problem by P_2 Lagrange Finite element method. In this report, we use two different meshes : uniform mesh and the mesh created by the function ***generate mesh*** of the FEM library.



(a) Uniform mesh



(b) mesh by *"generate mesh"*

Figure 3: Approximation solution u_h by $P_2 - FE$ in $\Omega = [1, 3] \times [1, 3]$ for $f = 1$

Code Availability

https://github.com/BMThi/Finite-Element-Methods-Model-Reductions/blob/main/TP2