Résolution du problème de Neumann par éléments finis en C++

Objectifs

Réaliser un programme C++ permettant de résoudre par éléments finis d'ordre 1 le problème de Neumann dans un domaine borné Ω de dimension 1 ou 2 :

$$\begin{cases} k^2 u - \Delta u = f & \text{dans } \Omega \quad (f \in L^2(\Omega)), \\ \frac{\partial u}{\partial n} = 0 & \text{sur } \partial\Omega. \end{cases}$$

Un peu comme cela été fait dans le cadre d'un projet Matlab plus tôt dans l'année, mais ici en utilisant le langage C++ *(en orienté objet)* afin de permettre une résolution plus rapide à l'occasion de l'introduction de la programmation scientifique.

Formulation du problème

La discrétisation de la formulation de ce problème dans l'espace des éléments finies V_h, donné par :

$$V_h = \{v \in C^0(\overline{\Omega}) : v|_{E_\ell} \in \mathbb{P}_1(E_\ell), \forall \ell = 1, L\}$$

où $\bigcup_{\ell=1}^{L} E_\ell = \overline{\Omega}$ (E_ℓ est un segment ou un triangle), conduit au système linéaire :

$$(\mathbb{K} + k^2\mathbb{M})\vec{U} = \mathbb{M}\vec{F}$$

avec :

$$K_{ij} = \int_\Omega \nabla w_i \cdot \nabla w_j \, dx, \quad M_{ij} = \int_\Omega w_i w_j \, dx, \quad F_i = f(M_i), \quad \forall i, j = 1, N,$$

et $w_i \in V_h$ tel que $w_i(M_j) = \delta_{ij} \quad \forall i, j = 1, N$
Il faudra donc fabriquer les matrices de masse et de rigidité $\mathbb{M}$ et $\mathbb{K}$ et résoudre le système linéaire pour obtenir la solution approchée.

1 Classes implémentées

1 **Classe Point :** gestion de points 1D ou 2D

a. Implémentation des constructeurs
b. Surcharge de l'opérateur `«` pour afficher un point sous la forme (x) ou (x, y)
c. Surcharge des opérateurs des fonctions membres et externes pour les points
d. Fonction `mesure` pour calculer la surface d'un triangle donné par trois sommets A, B, C, avec :

$$\text{mesure}(A, B, C) = \frac{1}{2}|\overrightarrow{AB} \times \overrightarrow{AC}|.$$

2 **Classe Maillage :** gestion des maillages 1D et 2D

a. Implémentation des constructeurs qui exporte les données du maillage dans un fichier, ou chargent des données de maillage provenant d'un fichier.
b. Fonctions d'affichage avec un méthode en `print` et une surcharge de l'opérateur `«`
c. Classe `Maillage1D` pour le maillage d'un segment $[a, b]$ en m intervalles.

d. Classe `Maillage2D` pour le maillage du carré unité $[0,1] \times [0,1]$ en $m \times n$ rectangles, chaque rectangle étant subdivisé en deux triangles aléatoirement, qui peut être dilaté selon l'espace que l'on veut mailler.

e. Le carré unité $[0,1] \times [0,1]$ est maillé en se basant sur un découpage en $m > 0$ segments suivant l'axe x et en $n > 0$ segments suivant l'axe y. Ce découpage conduit à $m \times n$ rectangles de taille $1/m$ par $1/n$. En parcourant les rectangles de gauche à droite et de bas en haut, les coordonnées des rectangles (nœuds du maillage) sont données par les formules suivantes :

$$\forall i = 1, \ldots, m,\ j = 1, \ldots, n$$

$$P_{ij} = \left(\frac{i-1}{m}, \frac{j-1}{n}\right), \quad P_{i+1,j} = \left(\frac{i}{m}, \frac{j-1}{n}\right),$$

$$P_{i,j+1} = \left(\frac{i-1}{m}, \frac{j}{n}\right), \quad P_{i+1,j+1} = \left(\frac{i}{m}, \frac{j}{n}\right).$$

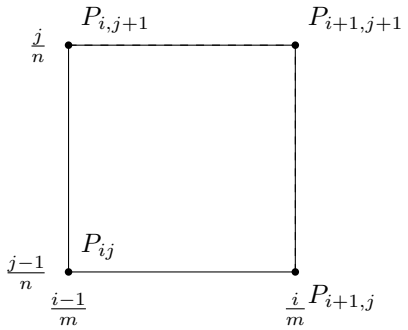

FIGURE 1 – Coordonnées des nœuds d'un rectangle dans un maillage 2D.

Chacun des rectangles est ensuite découpé en deux triangles de façon aléatoire :

$$(P_{i+1,j}, P_{i+1,j+1}, P_{ij}), \quad (P_{i,j+1}, P_{ij}, P_{i+1,j+1}),$$

ou bien :

$$(P_{ij}, P_{i+1,j}, P_{i,j+1}), \quad (P_{i+1,j+1}, P_{i,j+1}, P_{i+1,j}).$$

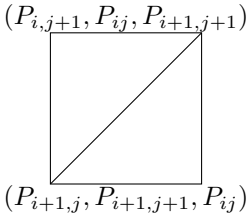

FIGURE 2 – Découpage du rectangle en triangles.

3 **Classe vecteur :** gestion des vecteurs

4 **Classe Sparse :** gestion des matrices creuses en utilisant des `map`

5 **Classe éléments finis :** calcul des matrices de masse et de rigidité élémentaires

a. Implémenter les classes `EF1D` et `EF2D` héritant de la classe `EF`, pour calculer les matrices de masse et de rigidité M^ℓ et K^ℓ

b. Les expressions sont données en fonction des sommets S_1^ℓ, S_2^ℓ d'un segment et $S_1^\ell, S_2^\ell, S_3^\ell$ d'un triangle. Nous définissons les termes suivants :

$$s_{ij} = S_j^\ell - S_i^\ell, \quad m = \text{mes}(S_1^\ell, S_2^\ell, S_3^\ell) = \frac{1}{2}|s_{12} \times s_{13}|$$

c. Pour un segment :

$$M_\ell = \frac{|s_{12}|}{6}\begin{pmatrix}2 & 1\\ 1 & 2\end{pmatrix} \quad \text{et} \quad K_\ell = \frac{1}{|s_{12}|}\begin{pmatrix}1 & -1\\ -1 & 1\end{pmatrix}$$

d. Pour un triangle :

$$M_\ell = \frac{m}{12}\begin{pmatrix}2 & 1 & 1\\ 1 & 2 & 1\\ 1 & 1 & 2\end{pmatrix} \quad \text{et} \quad K_\ell = \frac{1}{4m}\begin{pmatrix}s_{23}|s_{23} & -s_{13}|s_{23} & s_{12}|s_{23}\\ -s_{13}|s_{23} & s_{13}|s_{13} & -s_{12}|s_{13}\\ s_{12}|s_{23} & -s_{12}|s_{13} & s_{12}|s_{12}\end{pmatrix}$$

6 **Classe Neumann :** résolution de l'équation

a. Implémenter le constructeur pour initialiser le maillage, le nombre d'onde k, et les fonctions f et u_{exacte} (si connue).

b. Fonction `assembleMatrices` pour assembler les matrices globales $\mathbb{M}$ et $\mathbb{K}$.

c. Fonction `resoudre` pour construire et résoudre le système $(\mathbb{K} + k^2\mathbb{M})\vec{U} = \mathbb{M}\vec{F}$.

d. Fonction `calculErreur_L2` pour calculer l'erreur L^2 entre la solution exacte et la solution approchée qui s'écrit :

$$\text{erreur} = \sqrt{\int_\Omega |u - u_{\text{exacte}}|^2\, dx} = \sqrt{(\vec{U} - \vec{U}_{\text{exacte}})^T \mathbb{M} (\vec{U} - \vec{U}_{\text{exacte}})}$$

e. Fonction `exporte` pour sauvegarder la solution et les coordonnées dans un fichier