Projet C++
Résolution d'une équation aux dérivées partielles de type Poisson par une méthode des éléments finis en 2d

Les origines de la méthode des éléments finis remontent aux années 1950 lorsque des ingénieurs l'utilisèrent afin de simuler des problèmes de mécanique des milieux continus déformables. Depuis, le champ d'applications s'est considérablement étendu et les fondements théoriques de la méthode se sont amplement consolidés. Il existe de nos jours un nombre important de logiciels commerciaux et académiques qui utilisent la méthode des éléments finis comme un outil de simulation robuste pour des problèmes de mécanique des milieux continus, de mécanique des fluides, de thermique, d'électromagnétisme ou de finance, pour ne citer que quelques exemples.

Dans ce projet, on s'interesse à la résolution d'une équation aux dérivées partielles (EDP) de type équation de Poisson dans un domaine borné $\Omega \subset \mathbb{R}^2$. De nombreux modeles physiques peuvent se mettre sous cette forme (transfert conductif de chaleur, électrostatique, ...). De plus, nous ajoutons des conditions aux limites de type Dirichlet (valeur imposée de l'inconnue) sur le bord du domaine que l'on note Γ. Le problème consiste alors à trouver une fonction u à valeur réelle, $u : \Omega \longrightarrow \mathbb{R}^2$, qui vérifie le système suivant :

$$\begin{cases} -\Delta u = f & \text{dans } \Omega \\ u = g & \text{sur } \Gamma \end{cases} \tag{1}$$

avec :
- f une fonction définie sur Ω à valeur réelle
- g une fonction définie sur Γ à valeur réelle

La solution explicite d'un tel problème n'existe pas en général. Cependant, nous pouvons trouver une solution approchée à ce problème en utilisant une méthode numérique comme la méthode des éléments finis.

Notations :
- Laplacien de u : $\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$
- Gradient de u : $\nabla u = (\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y})$

1 Lecture d'un maillage

Un maillage est la discrétisation spatiale d'un milieu continu en un ensemble d'éléments géométriques simples appelés mailles. Dans ce projet, nous choisissons d'utiliser un partitionement du domaine Ω en N_e triangles disjoints, noté $\{K_e\}_{e=1}^{N_e}$ tel que

$$\overline{\Omega}_h = \bigcup_{e=1}^{N_e} K_e$$

forme un recouvrement exact du domaine Ω si celui-ci est polygonale, sinon le domaine discrétisé sera approché par un polygone.

Les maillages que nous allons utiliser ont été générés à l'aide du logiciel Gmsh. La génération d'un maillage est réaliser par la description d'une géométrie (via un fichier .geo) et d'une taille caractéristique des mailles. Le maillage est ensuite sauvegardé dans un fichier texte. La première étape de ce projet consistera à charger un maillage en mémoire à partir d'un fichier dans des structures de donnéee C++ que vous devrez définir.

Plusieurs fichiers de maillages (.msh) sont fournis :

- square2d_4elt.msh : un maillage du carré $[0,1] \times [0,1]$ très grossier composé seuleument de 4 triangles. Il servira principalement à développer/debugger votre programme.
- square2d_M0.msh : un maillage du carré $[0,1] \times [0,1]$ composé de 1474 triangles (figure 1).
- square2d_M1.msh : un maillage du carré $[0,1] \times [0,1]$ composé de 5824 triangles (figure 2).
- square2d_M2.msh : un maillage du carré $[0,1] \times [0,1]$ composé de 23252 triangles (figure 3).
- square2d_perforated.msh : un maillage du carré $[0,1] \times [0,1]$ avec 30 perforations circulaire. Il est composé de 42184 triangles (figure 4).

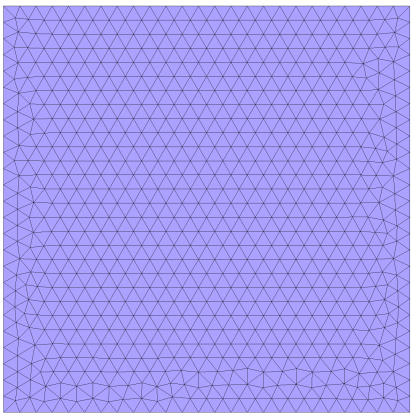


Figure 1 – square2d_M0.msh

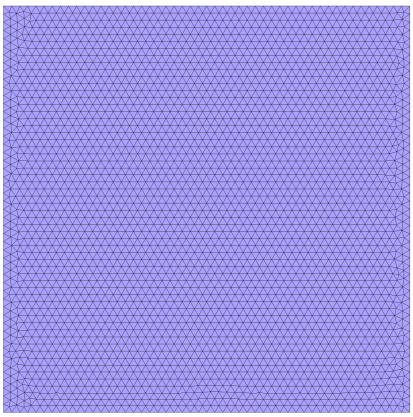


Figure 2 – square2d_M1.msh

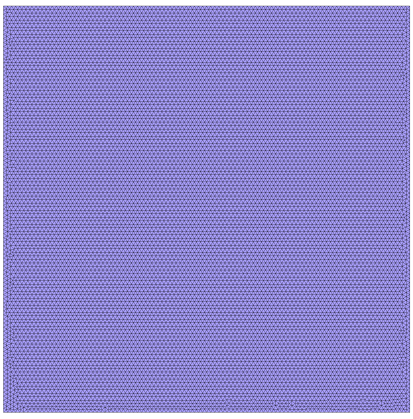


Figure 3 – square2d_M2.msh

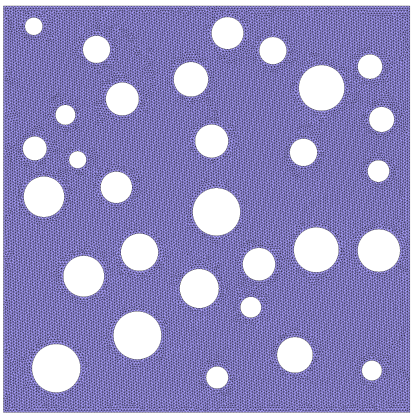


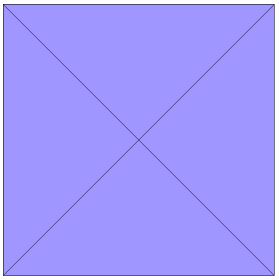
Figure 4 – square2d_perforated.msh

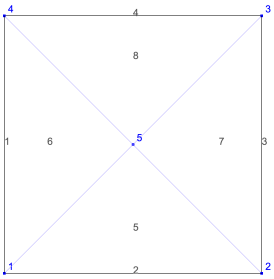
Remarques : Nous allons utiliser la version 2 du format msh qui est assez ancien mais qui est plus simple à charger que les dernières versions. Les maillages fournis ont été générés avec les fichiers square2d.geo et square_perforated.geo. La taille caractéristique des mailles est définie par le paramètre h se trouvant au début du fichier .geo. Pour générer un maillage à partir d'un .geo, vous devez soit ouvrir le logiciel et faire un export au format msh Version 2 ASCII, soit en ligne de commande : gmsh -2 square2d.geo -format msh2 -o square2d.msh

Pour décrire ce format de fichier, nous allons considérer le ficher square2d_4elt.msh dont le contenu est affiché ci-dessous. Il représente la discrétisation d'un carré composé seulement de 4 triangles. De plus, les arêtes (i.e. segments) se trouvant sur le bord du domaine sont également définies de manière à pouvoir imposer les conditions aux limites de l'EDP. Les 2 figures ci-dessous illustre le maillage et la numérotation des points, segments aux bord et triangles.

square2d_4elt.msh

```
$MeshFormat
2.2 0 8
$EndMeshFormat
$PhysicalNames
2
1 1 "Gamma"
2 2 "Omega"
$EndPhysicalNames
$Nodes
5
1 0 0 0
2 1 0 0
3 1 1 0
4 0 1 0
5 0.5 0.5 0
$EndNodes
$Elements
8
1 1 2 1 1 4 1
2 1 2 1 2 1 2
3 1 2 1 3 2 3
4 1 2 1 4 3 4
5 2 2 2 6 1 2 5
6 2 2 2 6 4 1 5
7 2 2 2 6 2 3 5
8 2 2 2 6 3 4 5
$EndElements
```





On peut voir que ce fichier est composé de 4 sections :

- **MeshFormat** : description du format msh utilisé. Ici, il faut bien vérifier que le premier argument 2.2. Les 2 autres valeurs non pas d'importance pour le projet.
- **PhysicalNames** : description des marqueurs (dit physiques) des éléments du maillage. Un marqueur associera une chaine de caractères à un entier que l'on retrouvera dans la description des éléments. La première valeur est le nombre de marqueur. Ensuite chaque ligne est composée de 3 valeurs :
  1. un entier pour la dimension de l'élément (1 pour les segments, 2 pour les triangles)
  2. un entier représentant le marqueur
  3. une chaine de caractères associée au marqueur
- **Nodes** : description des points présents dans le maillage. La première valeur représente le nombre de point. Ensuite chaque ligne représente un point dans le maillage avec :
  - le premier nombre est un identifiant du point, ici un entier.
  - les 3 autres valeurs sont les coordonnées x, y et z du point.
- **Elements** : une liste des éléments décrivant le maillage. Pour ce projet, nous utiliserons uniquement des maillages bidimensionnels composés de segment et de triangle. La première valeur est le nombre d'élément qui va être décrit. Ensuite chaque ligne représente la description d'un élément (segment ou triangle) de la manière suivante :
  - le premier nombre est un identifiant de l'élément, ici un entier.
  - la seconde valeur (un entier) est le type d'élément. Cette valeur vaudra 1 pour les segments et 2 pour les triangles.
  - la troisième valeur (un entier) correspond aux nombres de propriétés (dit tag) associé à cet élément. Ici ce sera toujours 2.
  - comme la précédente valeur vaudra toujours 2, nous aurons ensuite 2 nombres entiers. Le premier étant le marqueur physique (voir section PhysicalNames). Le deuxième un tag lié à la géométrie que nous ignorerons.
  - pour finir, nous auront les identifiants des points formant l'élément. Ainsi, il y a aura 2 nombres dans le cas d'un segment et 3 nombres pour les triangles.

Remarque : Remarquer que les identifiants des points ou éléments commencent à 1 (et non 0).

Objectifs : Ecrire un programme C++ permettant de charger ces maillages en mémoire. Vous devrez ensuite calculer l'aire du maillage (en sommant l'aire de chaque triangle) et la longueur du bord du domaine. Les résultats attendus sont :

— pour le carré sans perforation : aire=1 et longueur du bord=4
— pour le carré avec perforation : aire ≈ 0.837725 et longueur du bord ≈ 11.5086

Indications : Si l'on considère un triangle T composé des 3 sommets A_1, A_2 et A_3 qui sont représentés respectivement par les points (x_1, y_1), (x_2, y_2) et (x_3, y_3). L'aire du triangle T est alors donnée par la relation suivante :

$$2Aire(T) = \det \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{pmatrix}$$

2 Post-traitements

Dans cette deuxième partie, nous souhaitons effectuer les deux opérations de post-traitements suivantes :

1. Visualiser une fonction/expression définie sur un maillage grâce au logiciel ParaView par la connaissance de ces valeurs aux noeuds (i.e. points) du maillage. Cette fonction pourrait être une solution de notre EDP que nous traiterons dans la section 3.
2. Évaluer l'intégrale d'une fonction/expression sur le maillage. Plusieurs méthodes dites formules de quadrature seront proposées et le choix de la méthode dépendra de l'expression qui sera intégrée.

2.1 Visualisation sous ParaView

Le logiciel ParaView est très utilisé en calcul scientifique pour ces fonctionnalités de visualisation et de post-traitement. Pour cela, nous allons construire un fichier au format VTK (compatible avec ParaView) pour exporter un maillage ainsi qu'une (ou plusieurs) fonction à l'aide de ces valeurs en chaque point du maillage.

Pour décrire le format VTK, nous allons utiliser le maillage composé de 4 triangles. La fonction exportée sera $f(x, y) = x^2 + y^2$. Le fichier VTK qui devra être généré pour ce maillage est fourni sous le nom square2d_4elt.vtk. Essayer déjà d'ouvrir ce fichier sous ParaView. Maintenant, nous allons décrire le format VTK en se basant sur ce fichier. La première partie du fichier commence par une description du format et du type de donnée que nous allons exporter :

```
# vtk DataFile Version 2.0
my example
ASCII
DATASET UNSTRUCTURED_GRID
```

La première ligne donne la version du format du fichier VTK. La deuxième ligne est un nom que l'on donne à l'export (ce nom n'a pas vraiment d'importance pour notre utilisation). Les 2 dernières spécifient le type de donnée. Cette partie sera toujours identique pour tous les exports que nous ferons dans ce projet.

La partie suivante liste l'ensemble des points du maillage :

```
POINTS 5 float
0 0 0
1 0 0
1 1 0
0 1 0
0.5 0.5 0
```

Le nombre 5 correspond au nombre de point et le mot clé float représente la précision utilisée par le logiciel ParaView. Ensuite, nous avons 5 lignes qui représentent les coordonnées x, y et z de chaque point. Comme nous utilisons un maillage 2d, la coordonnée z sera toujours 0. Il est important de comprendre que chaque point est associé à un identifiant de point VTK (numérotation au sens de VTK). Le premier point a un identifiant égal à 0, le deuxième à 1, etc...

Les 2 parties suivantes correspondent à la description des éléments (triangles) du maillage. La première donne pour chaque élément la liste de point décrivant l'élément (donc ici il y aura toujours 3 identifiants de points pour un triangle) :

```
CELLS 4 16
3 0 1 4
3 3 0 4
3 1 2 4
3 2 3 4
```

Après le mot clé CELLS, nous avons le nombre d'élément (i.e. le nombre de triangle du maillage). Le nombre 16 représente le nombre de valeur à lire dans la suite de cette section. Pour chaque ligne, nous avons la description d'un triangle, donc composé de 3 points, c'est la première valeur qui apparait à chaque fois. Ensuite nous avons les identifiants des points composant le triangle dans la numérotation VTK (voir commentaire de la section précédente). Ainsi, nous avons 4 triangles dans cet exemple qui sont représentés chacun par 3 points, donc le nombre 16 est égale au nombre de triangle multiplié par 4.

La partie suivante donne le type d'élément par rapport à la description précédente. Comme nous ne traitons que des triangles, le type dit triangle VTK sera toujours égale à 5.

```
CELL_TYPES 4
5
5
5
5
```

Le nombre 4 représente le nombre d'élément (triangle). Il suffit ensuite de rajouter autant de 5 qu'il y a de triangle dans le maillage.

Indications : A partir de là, vous pouvez déjà charger ce fichier VTK dans ParaView, et vous devrez ainsi pouvoir visualiser le maillage.

Enfin, nous allons rajouter les valeurs d'une fonction en chaque point du maillage. Ici on considère la fonction $f(x, y) = x^2 + y^2$.

```
POINT_DATA 5
SCALARS toto float 1
LOOKUP_TABLE default
0
1
2
1
0.5
```

Dans la première ligne, le nombre 5 représente le nombre de point. La deuxième ligne donne les caractéristiques de la fonction. Dans ce projet, nous utiliserons uniquement des fonctions scalaires (i.e. à valeur dans $\mathbb{R}$) et dont la précision float sera suffisant pour la visualisation. Par conséquent le seul paramètre que vous pourrez choisir est ici défini par le mot toto qui représentera la fonction visualisable sous ParaView avec ce nom. Vous pouvez ainsi définir plusieurs champs à visualiser dans ce format en rajoutant des sections similaires mais en changeant ce nom. La troisième ligne doit être ajoutée mais ne vous préoccupez pas de son rôle. Les lignes suivantes représentent la valeur de la fonction en chaque point du maillage (en respectant la numérotation VTK).

Objectifs : Permettre à votre programme C++ de générer des fichiers VTK à partir d'un maillage et de la définition d'une expression pour la fonction f. Cette expression sera ainsi évaluée en chacun des points du maillage. Un outil d'export de fichier VTK devra être développé avec en

paramètre un maillage, une fonction évaluée aux noeuds du maillage et un nom de fonction qui apparaitra sous ParaView. Le fichier sauvegardé pourra être nommé par ce nom suivi du suffixe .vtk (par exemple toto.vtk). Les figures 5 et 6 illustre une visualisation avec ParaView sur le maillage square2d_M2.msh et square2d_perforated.msh.

FIGURE 5 – $f(x,y) = x^2 + y^2$

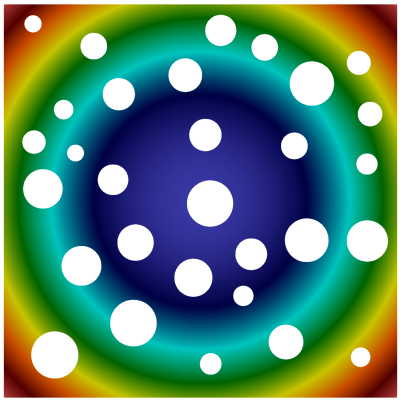


FIGURE 6 – $f(x,y) = (x-0.5)^2 + (y-0.5)^2$

2.2 Intégration numérique

L'intégration numérique est une technique permettant d'effectuer des calculs d'intégrale exact ou approché selon le type d'expression à intégrer. La plupart des formules d'intégration numérique proviennent de méthodes d'interpolations polynomiales. Le calcul de l'intégrale d'une fonction est approché par le calcul exact de l'intégrale d'un polynôme interpolant cette fonction en certains points $\{x_q\}_{q=1}^{p}$ que l'on appelle points d'intégration. On obtient ainsi une forme générale pour les quadratures numériques sur un triangle K :

$$\int_K f(x,y)dxdy \approx \sum_{q=1}^{p} w_q f(x_q)$$

avec les coefficients w_q dit poids de la formule de quadrature.

Le coût d'une formule de quadrature est mesuré par le nombre d'évaluation de la fonction f nécessaire, donc par le nombre de points d'intégration. La précision d'une formule d'intégration numérique est définie par l'ordre polynomial maximal pour lequel l'intégration d'un polynôme est exacte. On dit qu'une formule de quadrature est d'ordre k si l'ordre polynomial maximal est k.

L'intégration d'une expression sur notre domaine de calcul discrétisé noté Ω_h se ramène à l'intégration de l'expression sur chaque élément du maillage. Nous avons ainsi :

$$\int_{\Omega_h} f = \sum_{e=1}^{N_e} \int_{K_e} f$$

Dans le cas de triangles, les points d'intégration sont exprimés à l'aide des coordonnées barycentriques λ_1, λ_2 et λ_3. Si l'on considère un triangle défini par 3 sommets A_1, A_2 et A_3 qui sont représentés respectivement par les points (x_1, y_1), (x_2, y_2) et (x_3, y_3), alors un point (x, y) peut-être exprimé à l'aide de ses coordonnées barycentriques de la manière suivante :

$$\begin{aligned} x &= \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 \\ y &= \lambda_1 y_1 + \lambda_2 y_2 + \lambda_3 y_3 \end{aligned}$$

On rappelle que les coordonnées barycentriques doivent vérifier :

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

Nous allons maintenant présenter quelques formule de quadrature en définissant à chaque fois les coordonnées barycentriques ainsi que les poids associés. On notera par S l'aire du triangle sur lequel l'intégration s'effectue.

Quadrature d'ordre 1 : 1 point d'intégration
- coord. barycentriques : $\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$
- poids : S

Quadrature d'ordre 2 : 3 points d'intégration
- coord. barycentriques : $\left(\frac{2}{3}, \frac{1}{6}, \frac{1}{6}\right)$, $\left(\frac{1}{6}, \frac{2}{3}, \frac{1}{6}\right)$, $\left(\frac{1}{6}, \frac{1}{6}, \frac{2}{3}\right)$
- poids : les 3 poids sont égale à $\frac{1}{3}S$

Quadrature d'ordre 3 : 4 points d'intégration
- coord. barycentriques : $\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$, $\left(\frac{3}{5}, \frac{1}{5}, \frac{1}{5}\right)$, $\left(\frac{1}{5}, \frac{3}{5}, \frac{1}{5}\right)$, $\left(\frac{1}{5}, \frac{1}{5}, \frac{3}{5}\right)$
- poids : le premier avec $-\frac{9}{16}S$ et les 3 autres avec $\frac{25}{48}S$

Objectifs : Implémenter une fonction/classe permettant d'effectuer le calcul d'une intégrale avec un paramètre permettant de choisir une des formules de quadratures présentées ci-dessus. Vous devrez réaliser des tests en vérifiant l'exactitude des formules par intégration de polynôme comme expression sur le domaine sans perforation. Le résultat sera ainsi connu. Analyser également le comportement des quadratures sur des expressions non polynomiales en fonction du raffinement (i.e. en comparant sur les maillages M0, M1 et M2).

3 Une méthode de résolution par éléments finis

Dans cette troisième partie, nous allons utiliser une méthode par éléments finis pour résoudre le problème présenté en introduction, et trouver ainsi une solution approchée de l'EDP. La description de la méthode sera brève et un certain nombre de détails mathématiques ne seront pas expliqué. Cependant vous aurez l'essentiel pour mettre en œuvre cette méthode dans votre programme.

Dans la suite, nous supposons avoir un maillage du domaine Ω formé par N_e triangles que l'on note Ω_h. La définition de ces triangles est faite à l'aide de N_p points (tous différents) que l'on note par l'ensemble de coordonnées $\{(x_i, y_i)\}_{i=1}^{N_p}$.

3.1 Approximation par élément fini de Lagrange $\mathbb{P}_1$

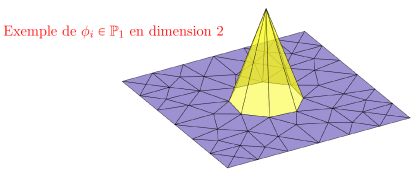
Pour commencer, nous avons besoin de définir un espace d'approximation pour la solution de l'EDP que l'on notera V_h. Ce sera l'espace des fonctions qui sont continues sur le maillage et affine sur chaque triangle du maillage. Celui-ci s'exprime par :

$$V_h = \left\{ v \in C^0(\Omega) \text{ tel que } v_{|K_e} \in \mathbb{P}_1,\ \forall e = 1...N_e \right\}$$

La dimension de V_h est égale au nombre de point du maillage N_p. La construction d'une base de cet espace s'effectue en choissant les N_p fonctions ϕ_i de V_h qui vérifie

$$\phi_i(x_j, y_j) = \delta_{ij}, \quad \forall i = 1...N_p \text{ et } \forall j = 1...N_p$$

La figure ci-contre illustre un exemple d'une des fonctions de base de V_h. Toutes ces fonctions sont de la même



forme, une fonction ϕ_i vaux 1 sur le noeud x_i, y_i et zéro sur tous les autres noeuds. Ainsi, une fonction $u_h \in V_h$ sera déterminée par ses valeurs aux nœuds du maillage par la relation suivante :

$$u_h(x,y) = \sum_{i=0}^{N_p} u_i \Phi_i(x,y)$$

avec $\{u_i \equiv u(x_i, y_i)\}_{i=1}^{N_p}$ l'ensemble des valeurs que l'on devra déterminer. C'est valeurs sont appelées les degrés de liberté.

Soit K un triangle composé des 3 sommets A_1, A_2 et A_3 qui sont représenté respectivement par les points (x_1, y_1), (x_2, y_2) et (x_3, y_3). On appellera fonction de forme de l'élément K, les restrictions des fonctions de base dans cet élément. Dans chaque triangle, il n'y a que 3 fonctions de bases non nulles. Les restrictions de ces 3 fonctions de base dans le triangle K sont notées par ϕ_1^K, ϕ_2^K, ϕ_3^K. Ce sont trois fonctions polynomiales de degrés 1 prenant la valeur 1 en un des sommets et nulles aux 2 autres. L'indice 1,2 ou 3 des fonctions de forme représente le sommet pour lequel la valeur vaut 1. Par exemple, ϕ_1^K est la fonction affine qui vaut 1 au sommet A_1 et zéro aux sommets A_2 et A_3. Ainsi, les fonctions de formes dans un triangle K sont définies par les expressions suivantes :

$$\phi_1^K(x,y) = \frac{x_2y_3 - x_3y_2 + x(y_2 - y_3) + y(x_3 - x_2)}{2Aire(T)}$$
$$\phi_2^K(x,y) = \frac{x_3y_1 - x_1y_3 + x(y_3 - y_1) + y(x_1 - x_3)}{2Aire(T)}$$
$$\phi_3^K(x,y) = \frac{x_1y_2 - x_2y_1 + x(y_1 - y_2) + y(x_2 - x_1)}{2Aire(T)}$$

3.2 Formulation variationnelle discrète

Pour résoudre l'EDP présentée en introduction par la méthode des éléments finis, il faut également transformer ce problème en une formulation dite variationnelle. Nous avons pour cela besoin d'introduire les espaces d'approximation suivants :

$$V_{0h} = \{v \in V_h \mid v = 0 \text{ sur } \Gamma\} \text{ et } V_{gh} = \{v \in V_h \mid v = g \text{ sur } \Gamma\}$$

Après une manipulation de l'EDP, la formulation variationnelle discrète est alors donnée par le problème suivant :

$$\begin{cases} \text{Trouver } u_h \in V_{gh} \text{ telle que} \\ \displaystyle\int_{\Omega_h} \nabla u_h \cdot \nabla v = \int_{\Omega} f\ v, \quad \forall v \in V_{0h} \end{cases} \tag{2}$$

On note par I_i l'ensemble des indices des nœuds du maillage correspondant à une valeur inconnue, c'est-à-dire ne touchant pas le bord. On note par I_b l'ensemble des indices des nœuds du maillage se trouvant sur Γ, c'est-à-dire touchant le bord. En utilisant l'écriture de u_h avec les fonctions de base de l'espace V_h, cette formulation peut s'exprimer comme la combinaison des 2 énoncés suivants :

- $\begin{cases} \text{Trouver les valeurs } u_j \text{ pour } j \in I_i \text{ telles que} \\ \displaystyle\sum_{j=1}^{N_p} \left(\int_{\Omega_h} \nabla\phi_j \cdot \nabla\phi_i \right) u_j = \int_{\Omega_h} f\ \phi_i, \quad \forall i \in I_i \end{cases}$
- $\begin{cases} \text{Imposer les valeurs } u_j \text{ pour } j \in I_b \text{ telles que} \\ u_j = g(x_j, y_j) \end{cases}$

On obtient un système de N_p équations à N_p inconnues, où on rappelle que N_p désigne le nombre de points du maillage. Ce système s'écrit sous la forme matricielle

$$AU = F$$

où les entrées de la matrice A et du vecteur second-membre F qui sont données par :

$$
\begin{aligned}
A_{ij} &= \int_{\Omega_h} \nabla\phi_j \cdot \nabla\phi_i = \sum_{e=1}^{N_e} \int_{K_e} \nabla\phi_j \cdot \nabla\phi_i, \quad \forall i \in I_i, \forall j = 1...N_p \\
A_{ii} &= 1, \quad \forall i \in I_b \\
F_i &= \int_{\Omega_h} f\ \phi_i = \sum_{e=1}^{N_e} \int_{K_e} f\ \phi_i, \quad \forall i \in I_i \\
F_i &= g(x_i, y_i), \quad \forall i \in I_b
\end{aligned}
$$

3.3 Algorithme d'assemblage

On peut observer que la matrice A est très creuse car un grand nombre de ces coefficients sont nuls. Cela est dû au choix des fonction de base ϕ_i qui ont un support limité. L'assemblage de la matrice A et du vecteur F avec la méthode des éléments finis se ramène à l'assemblage de matrices et vecteurs dit élémentaires, car ils sont calculés pour chacun des éléments du maillage. Ainsi pour chaque triangle K du maillage, la matrice élémentaire A_k et vecteur élémentaire F_k sont évaluée de la manière suivante :

$$
A_K = \begin{pmatrix}
\int_K \nabla\phi_1^K \cdot \nabla\phi_1^K & \int_K \nabla\phi_1^K \cdot \nabla\phi_2^K & \int_K \nabla\phi_1^K \cdot \nabla\phi_3^K \\
\int_K \nabla\phi_2^K \cdot \nabla\phi_1^K & \int_K \nabla\phi_2^K \cdot \nabla\phi_2^K & \int_K \nabla\phi_2^K \cdot \nabla\phi_3^K \\
\int_K \nabla\phi_3^K \cdot \nabla\phi_1^K & \int_K \nabla\phi_3^K \cdot \nabla\phi_2^K & \int_K \nabla\phi_3^K \cdot \nabla\phi_3^K
\end{pmatrix} \text{ et } F_K = \begin{pmatrix}
\int_K f\ \phi_1^K \\
\int_K f\ \phi_2^K \\
\int_K f\ \phi_3^K
\end{pmatrix}
$$

Require: A une matrice et F un vecteur, chacun mis à zéro.

1: **for** $e = 1$ **to** N_e **do**
2: | Calcul matrice élémentaire A_{K_e}
3: | Calcul vecteur élémentaire F_{K_e}
4: | **for** $s = 1$ **to** 3 **do**
5: | | $i = \text{GlobalDof}(K_e, s)$
6: | | **for** $t = 1$ **to** 3 **do**
7: | | | $j = \text{GlobalDof}(K_e, t)$
8: | | | $A(i,j) = A(i,j) + A_{K_e}(s,t)$
9: | | **end for**
10: | | $F(i) = F(i) + F_{K_e}(s)$
11: | **end for**
12: **end for**
13: **for** $i \in I_b$ **do**
14: | **for** $j = 1$ **to** N_p **do**
15: | | $A(i,j) = 0$
16: | **end for**
17: | $A(i,i) = 1$
18: | $F(i) = g(x_i, y_i)$
19: **end for**

Dans cet algorithme, vous trouverez une fonction nommée GlobalDof. Elle prend en argument 2 paramètres : un triangle du maillage et l'indice d'un sommet dans ce triangle. La fonction renverra alors l'identifiant du degré de liberté globale sous la forme d'un entier. Dans notre cas, il correspondra à l'identifiant du point dans le maillage, il sera ainsi compris entre 1 et N_p inclue. Cependant, il faudra faire attention aux structures de données C++ (comme les tableaux, les vecteurs et matrices Eigen). Celles-ci ont des indices qui commencent à zéro et non pas à 1.

De plus, la deuxième partie de l'algorithme (ligne 13 à 19) correspond à la prise en compte des conditions aux limites de type Dirichlet. En notant I_b, l'ensemble des degrés de liberté au bord, cette méthode consiste à modifier les lignes I_b de la matrice A ainsi que les valeurs du vecteur F aux indices I_b. Pour cela, il faut mettre à zéro toutes les entrées de ces lignes, sauf pour le terme sur la diagonale qui vaudra 1. Le vecteur F sera modifié en mettant la valeur définie par la condition aux limites. Cette technique est appelée méthode par élimination.

3.4 Solution du problème

Pour résoudre le système $AU = F$, nous devons inverser la matrice A qui est potentiellement de grande taille mais creuse. De nombreuses techniques existent, soit en calculant explicitement l'inverse, soit en utilisant une méthode itérative. Cependant, nous n'allons pas implémenter une telle procédure mais plutôt utiliser la librairie EIGEN http://eigen.tuxfamily.org/dox/. Elle fournit des fonctionnalités pour décrire les matrices denses, les matrices creuses et les vecteurs. Vous devrez ainsi utiliser :

— Les matrices denses pour les matrices élémentaires A_k.
— Les matrices creuses pour la matrice globale A.
— Les vecteurs pour les vecteurs élémentaires F_k et le vecteur global F

Cette librairie dispose également de plusieurs solvers permettant de résoudre le système linéaire. Une fois résolus par EIGEN, vous obtiendrez le vecteur U qui représentera l'approximation éléments finis du problème. Chaque valeur u_i correspond à la valeur de la solution au point (x_i, y_i).

3.5 Applications

3.5.1 Terme source constant

Dans un premier on s'intéressera à résoudre l'EDP en utilisant un terme source f constant, égale à 1. De plus, nous imposerons des conditions de Dirichlet homogène, c'est-à-dire $u = 0$ sur le bord. Ceci se traduit par la recherche d'une fonction u vérifiant le système suivant :

$$\begin{cases} -\Delta u = 1 & \text{dans } \Omega \\ u = 0 & \text{sur } \Gamma \end{cases}$$

Les figures 7 et 8 illustrent les solutions numériques obtenues par cette méthode des éléments finis. Ces visualisations ont été réalisées avec PARAVIEW en utilisant le coloris *Rainbow Desaturated*.

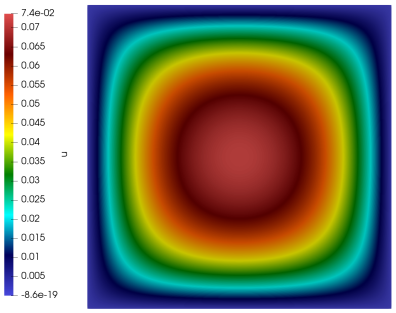


FIGURE 7 – square2d_M2.msh

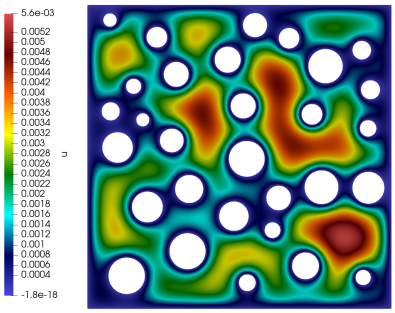


FIGURE 8 – square2d_perforated.msh

3.5.2 Convergence

Convergence de la solution approchée TODO

4 Objectifs du projet

Les objectifs du projet consistent à implémenter un ou plusieurs programme C++ permettant de traiter les 3 parties décrites précédemment. La qualité du code C++, les choix des structures de données, l'efficacité du programme ainsi que les annotations facilitant la compréhension du code seront évalués. Le ou les programmes devront être paramétrable, dans le sens où l'on pourra obtenir des résultats en fonction de paramètres donnés par l'utilisateur. Ce sera soit en ligne de commande

ou soit via un fichier de configuration donné à l'exécutable (la deuxième méthode est préférable). Les objectifs de chacune des parties devront pouvoir être testé grâce à la documentation fournies.

Un rapport de projet est demandé. Il contribuera à peu près à la moitié de la note. Celui-ci doit fournir une description sommaire du projet, définir les fonctionalités du programme C++, le choix des structure de données, la structure du code, des détails mathématiques ou autre non fournis, les tests qui ont été effectué ainsi que les commandes permettant de reproduire les résulats.

Au niveau programmation C++, vous êtes libre d'utiliser n'importe quelle fonctionnalité de la STL ou de la librairie EIGEN. Une autre librairie externe ne sera pas autorisée.

Annexes

Évaluation d'une expression

Dans le fichier example_evaluator.cpp fourni, vous trouverez un petit programme qui permet de stocker et d'évaluer une expression (dépendant de x et y) dans une classe. De plus, grâce à l'héritage, on peut utiliser uniquement le type de base pour le passage des expressions en paramètre. Ce principe pourra être étendu pour l'évaluation de fonction de base par exemple. Ça pourra être utile pour la partie intégration numérique.

Assemblage et résolution d'un système linéaire creux

Dans le fichier example_assembly.cpp fourni, vous trouverez un programme qui initialise une matrice creuse, effectue l'élimination de quelques lignes de la matrice et résout le système linéaire. La construction de la matrice creuse s'effectue à l'aide de la notion de triplet, qui va accumuler toutes les valeurs à ajouter. Ce triplet est représenté par (i,j,val) avec i et j les indices d'une entrée dans la matrice et val la valeur que l'on ajoute. Un point important pour le bon fonctionnement de la méthode d'élimination est le choix du type de matrice creuse qui est fait en spécifiant l'option template **Eigen::RowMajor**.