Polytech'Nice Sophia-Antipolis
Méthodes Numériques pour les EDP - MAM4 - Année 2024-2025

TP : EF pour le calcul de l'équation de la chaleur

Le but de ce TP est l'implémentation d'un petit code éléments finis pour le calcul de l'équation de la chaleur.

Le TP sera réalisé par groupe de deux maximum. La date limite pour rendre le devoir est le 17 mars. Il est attendu d'envoyer un code complet ainsi qu'un compte-rendu.

Il est **très fortement conseillé** d'utiliser le langage Julia : une documentation complète est disponible sur la documentation en ligne. La syntaxe de base est très similaire à Matlab ou Python ; les packages utiles sont rappelés au cours des questions. Le chargement des packages s'effectue avec la commande `using nomDuPackage` en tête du fichier.

Chaque question doit être traitée en séparant l'écriture d'une fonction et sa validation. Toute fonction n'ayant pas été **correctement validée (ou pire, ne s'exécutant pas)** ne pourra pas rapporter le maximum des points. Les fonctions doivent être rassemblées dans un fichier `TP_Nom1_Nom2.jl`, et la validation être décrite dans le compte-rendu au format PDF (rédigé en LaTeX), avec l'explication de ce qui est validé et le comportement observé. Les fonctions s'écrivent :

```
function maFonction(argument1, argument2, ...)
# Remplir les instructions
return resultat
end
```

et sont appelées dans le REPL Julia :

```
> include(''TP_Nom1_Nom2.jl'') #Pour charger les fonctions écrites dans le fichier
> resultat = maFonction(argument1, argument2, ...)
```

Le sujet est constitué de deux parties : une première partie **obligatoire**, dont la validation suffit à dépasser la moyenne, qui porte sur la formulation éléments finis P1 en 1D ; une seconde partie facultative, où plusieurs possibilités d'extension sont proposées. Il n'est pas nécessaire de faire toute la seconde partie pour avoir la note maximale. **Il est attendu une certaine originalité dans les productions :** les codes trop similaires ne seront pas bien évalués.

Première partie : code EF P1 1D

On considère un domaine $[0, L]$ et une fonction $f : [0, 1] \to \mathbb{R}$. On cherche à résoudre l'équation de la chaleur avec conditions aux limites de Dirichlet homogènes :

$$\begin{cases} -u''(x) = f(x) & 0 < x < L \\ u(0) = 0;\ u(L) = 0 \end{cases} \tag{1}$$

Etape 1 (1 point)**.**
Ecrire une fonction f qui a en entrée x un nombre réel, et en sortie $f(x)$ (pré-remplir avec des valeurs arbitraires).

Etape 2 (1 points)**.**
Ecrire une fonction de maillage du domaine $[0, L]$. Cette fonction prendra en entrée la taille du domaine L, le nombre d'éléments NE et le nom du fichier de maillage en sortie, et devra produire en sortie un fichier selon le format suivant :

1. ligne 1 : Nombre de noeuds NN, nombre d'éléments NE
2. NN lignes suivantes : numéro de noeud (tabulation) coordonnées du noeud
3. NE lignes suivantes : numéro d'élément (tabulation) liste des noeuds de l'élément
4. Ligne suivante : nombre de noeuds ND avec des conditions limites de Dirichlet
5. ND lignes suivantes : numéros des noeuds avec conditions de Dirichlet

On pourra pour cela utiliser les fonctions `out=open(fileName,''w'')` et `println(out,''...'')` de Julia. **Expliquer comment vous avez validé le comportement correct de la fonction.**

Etape 3 (1 points)**.**
Ecrire également une fonction de lecture du fichier de maillage selon ce format. La fonction prendra en entrée le nom du fichier de maillage (et éventuellement la dimension du problème, cf la partie 2) et renverra les positions des noeuds, les éléments et les noeuds sur conditions de Dirichlet. On pourra utiliser la fonction `readdlm(fileName)` du package `DelimitedFiles`. **Expliquer comment vous avez validé le comportement correct de la fonction.**

Etape 4 (4 points)**.**
Ecrire une fonction ayant en entrée un numéro d'élément, la liste des noeuds et des éléments et la fonction f, et en sortie les tables élémentaires (matrice de rigidité et second membre) correspondants. **Expliquer comment vous vérifiez que le résultat obtenu est correct et correspond bien, d'une part à $\int_{e_i} \nabla u \cdot \nabla v$, et d'autre part à $\int_{e_i} fv$.**

Etape 5 (4 points)**.**
Ecrire la fonction d'assemblage qui a en entrée les noeuds, les éléments et la fonction f et qui donne en sortie la matrice de rigidité A et le second membre b. **Expliquer comment vous vérifiez que le résultat obtenu est correct et correspond bien, d'une part à $\int_0^L \nabla u \cdot \nabla v$, et d'autre part à $\int_0^L fv$.**

Etape 6 (2 points)**.**
Écrire la fonction d'imposition des conditions aux limites, qui a en entrée la matrice de rigidité A, le vecteur second membre b et la liste des noeuds avec conditions de Dirichlet, et en sortie la même matrice mais avec conditions aux limites traitées par pénalisation. **Expliquer comment on vérifie que le résultat obtenu est correct.**

Etape 7 (1 point)**.**
Résoudre le système linéaire pour trouver le vecteur des degrés de liberté de la fonction déplacement $u_h(x)$.

Etape 8 (2 points)**.**
Valider toute la chaîne de calcul, pour deux fonctions f différentes pour lesquelles on connaîtra une solution exacte $u(x)$. Tracer les graphiques comparant la solution calculée $u_h(x)$ et la solution exacte $u(x)$. On pourra utiliser la fonction `plot(x, [uh u], labels=["...","..."])` du package `Plots`.

Partie 2 : extensions possibles

Les étapes suivantes sont indépendantes.

Etape 9 (4 points)**.**
Ecrire une fonction de calcul de l'erreur pour un certaine norme à préciser. On pourra tracer l'évolution de l'erreur en fonction de la taille du maillage h en échelle log-log. Que peut-on déduire de cette courbe ?

Etape 10 (4 points)**.**
Ecrire une fonction d'imposition des conditions aux limites alternative, qui a en entrée la matrice de rigidité A, le vecteur second membre b et la liste des noeuds avec conditions de Dirichlet, et en sortie la même matrice mais sans les lignes et colonnes correspondant aux conditions de Dirichlet. Ecrire également la fonction qui réintègre la valeur correcte de uh sur les conditions aux limites.

Élements finis 2D

Etape 11 (2 points)**.**
Modifier la fonction d'écriture du maillage de l'étape 2 pour écrire un fichier de maillage 2D d'un carré $[0, L] \times [0, L]$. On pourra prendre un maillage inspiré du TD 8.

Etape 12 (2 points)**.**
Modifier la fonction de lecture du maillage de l'étape 3 pour lire un fichier de maillage 2D d'un carré $[0, L] \times [0, L]$. On pourra prendre en entrée le fichier exemple `mesh2D.msh` sur la page du cours.

Etape 13 (6 points)**.**
Écrire la fonction qui donne les tables élémentaires en 2D comme dans l'étape 4. **Valider le résultat en visualisant la solution obtenue avec la fonction** `plot(x, y, u, st=:surface)`**.**

Élements finis P2 en 1D

Etape 14 (6 points)**.**
Reprendre les étapes 4 (attention, il faut changer les noeuds et les éléments) et 8 pour des éléments finis P2 en 1D (cf le TD 6). On pourra reprendre également l'étape 9 pour voir s'il y a une amélioration de la vitesse de convergence.