Introduction

L'objectif de ce TP consiste à résoudre numériquement, par la méthode des éléments finis, divers problèmes variationnels en dimension 2 d'espace. A cet effet, on va utiliser le logiciel libre FreeFem++ développé au Laboratoire Jacques-Louis Lions de Paris 6. Ce dernier permet de résoudre très simplement de nombreux problèmes variationnels.

Une documentation de FreeFem++ est accessible sur www.freefem.org, à l'adresse suivante

http://www.freefem.org/ff++/ftp/freefem++doc.pdf

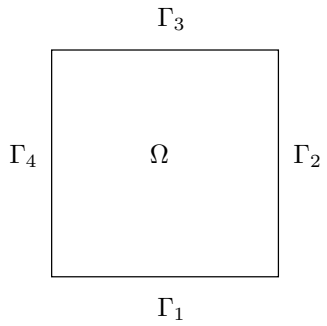
1 Le Laplacien

1.1 Prise en main

On se propose de résoudre numériquement le problème consistant à déterminer u tel que

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

dans le carré $\Omega =]0,1[^2$:



Le script FreeFem++ ci-dessous résout précisément ce problème en dix lignes seulement (sans comptabiliser les commentaires) ! On utilise le rouge pour les commandes FreeFem++.

```
//Nombre de mailles suivant x et y
int Nbnoeuds=10;
```

Le texte suivant // constitue des commentaires ignorés par FreeFem++. Chaque nouvelle variable introduite doit être précédée de son type (ici int, c'est-à-dire un entier).

```
//Definition du maillage
mesh Th=square(Nbnoeuds,Nbnoeuds,[x,y]);
```

La fonction `square` retourne un maillage structuré. Les deux premiers arguments fixent le nombre de noeuds suivant x et y respectivement. Le troisième argument est une paramétrisation de Ω pour x et y variant entre 0 et 1 (dans notre cas, c'est l'identité). Les côtés du carré sont numérotés de 1 à 4, dans le sens trigonométrique, le côté inférieur portant le label 1 (voir la figure).

```
//Fonction de x et de y
func f=x*y;

//Definition de l'espace des elements finis P1 associe
//au maillage Th
fespace Vh(Th,P1);

//uh et vh sont des elements de Vh
Vh uh,vh;
```

Les fonctions u_h et v_h appartiennent à l'espace V_h des fonctions P_1. Notons que si l'on souhaite utiliser des éléments finis P_2 et non P_1, il suffit de remplacer `P1` par `P2` dans la définition de `Vh` .

```
//Definition du probleme variationnel
problem chaleur(uh,vh,solver=LU)=
        int2d(Th)(dx(uh)*dx(vh)+dy(uh)*dy(vh))
        -int2d(Th)(f*vh)
        +on(1,2,3,4,uh=0)
;
```

La fonction `problem` permet de définir un problème variationnel, que nous dénommons ici `chaleur` . Ici, on définit le problème consistant à déterminer

$$u_h \in V_h^0 = \{w_h \in V_h \ : \ w_h = 0, \text{ sur } \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4\}$$

tel que

$$\int_\Omega \nabla u_h . \nabla v_h dx - \int_\Omega f v_h dx = 0,$$

pour tout $v_h \in V_h^0$. Notons que cette commande définit le problème mais ne le résout pas. Le problème (1) est résolu numériquement par la commande

```
//Resolution du probleme
chaleur;

//On affiche le resultat
plot(uh,wait=1);
```

Remarque - La méthode de résolution utilisée pour résoudre le système est la factorisation LU.

Recopier ce script (c'est-à-dire les parties rouges) à l'aide de votre éditeur de texte favori et sauvez-le dans un fichier (sous le nom `chaleur.edp` par exemple), ou bien faites un copier-coller du fichier chaleur.edp.

Remarque - `acroread` est munis d'une fonction permettant de faire des copier-coller. Pour exécuter le script sous `FreeFem++`, il suffit de taper la commande shell

```
FreeFem++ chaleur.edp
```

Le résultat du calcul s'affiche dans une fenêtre graphique. Pour reprendre la main dans le shell, cliquer dans cette dernière.

1.2 Est-ce que ça converge ?

On souhaite vérifier que la solution obtenue à l'aide de `FreeFem++` converge bien vers la solution exacte lorsque le pas du maillage tend vers zéro.

A cet effet, on choisit un second membre f de sorte que la solution du Laplacien u soit connue (en fait, on choisit plutôt u et on en déduit f), puis on calcule la solution discrète u_h pour des maillages de plus en plus fins. Une fois le choix de f et de u effectués, vérifier la décroissance de $\|u - u_h\|_{L^2}$ et de $\|\nabla u - \nabla u_h\|_{L^2}$.

Quelques indications pour répondre à la question:

- Le type réel est `real`. Attention, lors de la manipulation de quantités réelles, les entiers doivent être suivis d'un point. Dans le cas contraire, `FreeFem++` les interprète comme des variables de type `int` , ce qui peut être source d'erreurs.
- Les intégrales sur le maillage `Th` se calculent à l'aide de
  `int2d(Th)(` expression à intégrer `)`
- On peut faire des boucles sous `FreeFem++`. La syntaxe est identique à celle du `c++`
  ```
  int Nbiter=10;
  for(int i=0;i<Nbiter;i++){
  On fait ceci-cela ...
  };
  ```
- On peut effectuer des sorties textes soit par la sortie texte standard, soit sur la fenêtre graphique `FreeFem++`
  ```
  real erreur=0.;
  cout<<''erreur ='' <<erreur<<endl;
  ```
  et pour une sortie sur la fenêtre graphique en même temps que u_h,
  ```
  real erreur=0.;
  string legende=''erreur ='' +erreur;
  plot(uh,cmm=legende,wait=1);
  ```

1.3 Quelques variantes

On peut aisément modifier le script initial pour résoudre des problèmes variationnels du même type, avec conditions de Neumann, ou de Fourier par exemple.

Après avoir déterminé leur formulation variationnelle, résoudre numériquement à l'aide de `FreeFem++` les deux problèmes suivants

$$\begin{cases} -\Delta u + u = f & \text{dans } \Omega \\ \frac{\partial u}{\partial n} = 0 & \text{sur } \partial\Omega \end{cases} \tag{2}$$

$$\begin{cases} -\Delta u = f & \text{dans } \Omega \\ \alpha u + \frac{\partial u}{\partial n} = g & \text{sur } \partial\Omega, \end{cases} \tag{3}$$

où f et g sont des fonctions quelconques que vous choisirez (non toutes deux nulles tout de même), et α est un réel strictement positif.

Indication: Dans la formulation variationnelle du problème (3) apparaît une intégrale sur le bord de Ω. Une intégrale sur bord $\Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4$ se définit sous `FreeFem++` par la commande

```
int1d(Th,1,2,3,4)(expression à intégrer );
```

2 Résolution de l'élasticité linéarisée

Le logiciel `FreeFem++` permet également de résoudre des problèmes vectoriels. On va chercher à résoudre le problème de l'élasticité linéaire dans une poutre de longueur L et de largeur unité, fixée à son extrémité gauche, soumise à des forces volumiques f.

2.1 Position du problème

Soit $\Omega =]0, L[\times]0, 1[$, on note Γ_i, $i = 1, \ldots, 4$ les cotés du rectangle Ω en utilisant la même numérotation que précédemment. On introduit un maillage régulier $\mathcal{T}_h$ et W_h l'espace des fonctions P_1 à valeurs vectorielles (dans $\mathbb{R}^2$) associé. L'approximation de Galerkin du problème de l'élasticité linéaire consiste à déterminer

$$u_h = [u_h^1, u_h^2] \in X_h = \left\{ [w_h^1, w_h^2] \in W_h \ : \ w_h^1 = 0 \text{ et } w_h^2 = 0 \text{ sur } \Gamma_4 \right\}$$

tel que

$$\int_\Omega 2\mu e(u_h) : e(v_h) + \lambda(\operatorname{div} u_h)(\operatorname{div} v_h) dx - \int_\Omega f.v_h dx = 0, \tag{4}$$

pour tout $v_h \in X_h$, où $e(v_h)$ est le tenseur métrique linéarisé,

$$e(v_h) = (\nabla v_h + (\nabla v_h)^T)/2,$$

et μ, λ sont les coefficients de Lamé du matériau.

Résoudre ce problème variationnel à l'aide de `FreeFem++`. On choisira λ, μ et f à sa convenance.

Indications:

La définition de l'espace d'éléments finis P_1 à valeurs dans $\mathbb{R}^2$ s'effectue sous `FreeFem++` par la commande

```
fespace Wh(Th,[P1,P1]);
```

Un élément de W_h est désigné sous `FreeFem++` par ses deux composantes. L'initialisation d'un élément $u_h \in W_h$ s'effectue donc par la commande

```
Wh [uh1,uh2];
```

L'expression développée de la formulation variationnelle (4) est

$$\int_\Omega 2\mu\left(\partial_x u_h^1 \partial_x v_h^1 + \partial_y u_h^2 \partial_y v_h^2 + (\partial_x u_h^2 + \partial_y u_h^1)(\partial_x v_h^2 + \partial_y v_h^1)/2\right) + \lambda(\partial_x u_h^1 + \partial_y u_h^2)(\partial_x v_h^1 + \partial_y v_h^2)dx - \int_\Omega f^1 v_h^1 + f^2 v_h^2 dx = 0$$

2.2 Affichage

L'affichage de données vectorielles à l'aide de la fonction `plot` n'est pas idéal. Par contre, `FreeFem++` offre la possibilité de visualiser la déformation d'un maillage $\mathcal{T}_h$. Dans un premier temps, on définit le maillage déformé

```
real exa=0.1;   //coefficient d'exageration
mesh Sh=movemesh(Th,[x+exa*uh1,y+exa*uh2]);
```

Il suffit alors d'afficher le maillage déformé $\mathcal{S}_h$

```
plot(Sh);
```

Enfin, une donnée intéressante consiste à visualiser la norme du tenseur des contraintes. Observez-vous l'apparition de singularités ?

2.3 Raffinement de maillage

On peut améliorer le calcul précédent en travaillant sur un maillage de plus en plus fin. Cependant, utiliser un maillage uniforme n'est pas optimal. Il s'avère plus intéressant de raffiner dans les régions où il se passe réellement quelque chose, c'est à dire où le gradient de la solution u_h varie rapidement. A nouveau, `FreeFem++` propose une solution clé en main. La fonction `adaptmesh` permet de raffiner le maillage en l'adaptant à une fonction spécifiée:

```
real erreur=0.001;
Th=adaptmesh(Th,uh1,uh2,err=erreur);
```

permet d'adapter le maillage avec une précision inversement proportionnelle à `erreur` en fonction de $u_h = [u_h^1, u_h^2]$.