TPs : Volumes finis pour les problèmes elliptiques

F. Boyer

Institut de Mathématiques de Marseille
Aix-Marseille Université

1. Prise en main du format de maillage et des jeux de données utilisés (fourni).
2. Programmation de VF4 :
   - Pour le Laplacien puis pour une diffusion à coefficients variables réguliers
   - Avec conditions aux limites de Dirichlet homogène puis éventuellement non-homogènes
   - Enfin : prise en compte des discontinuités de coefficients
3. Tests de convergence sur diverses familles de maillages et des solutions analytiques.
4. Utilité de la condition d'orthogonalité

- 3 librairies de fonctions :
  - `donnees2D.sci`
    Différents jeux de données
  - `maillages2D.sci`
    Définition des maillages disponibles, chargement et manipulation. Calculs de normes discrètes. Routines de tracé des fonctions discrètes.
  - `schemas2D.sci`
    Fonctions d'assemblage des schémas
- 2 programmes principaux :
  - `VF2D.sce`
    Calcul et tracé des solutions approchées/exactes
  - `courbes_erreur2D.sce`
    Tracé de courbes d'erreurs

Les maillages 2D :

- Stockés dans 3 fichiers (fournis) :
  - `xxxx_sommets`
  - `xxxx_centres`
  - `xxxx_aretes`
- Sont chargés dans Scilab par la commande

```
[maillage]=lecture_maillage(xxxx);
```

- La variable `maillage` ainsi obtenue est une structure qui contient
  - `maillage.nom` : un nom qui décrit le maillage
  - `maillage.nb_vol` : nombre de volumes de contrôle
  - `maillage.nb_som` : nombre de sommets
  - `maillage.nb_are` : nombre d'arêtes
  - `maillage.sommets` : une matrice de taille `nb_som x 2`
  - `maillage.centres` : une matrice de taille `nb_vol x 2`
  - `maillage.aretes` : une matrice de taille `nb_are x 19`

- Sommets : coordonnées du sommet numéro `i`

  `maillage.sommets(i,_X)`, `maillage.sommets(i,_Y)`

- Centres : coordonnées du centre du volume `j`

  `maillage.centres(j,_X)`, `maillage.centres(j,_Y)`

- Arêtes : Pour l'arête numéro `k`
  - Numéros des deux sommets

    `maillage.aretes(k,_DEB)`, `maillage.aretes(k,_FIN)`

  - Numéros des deux volumes $\mathcal{K}$ et $\mathcal{L}$

    `maillage.aretes(k,_K)`, `maillage.aretes(k,_L)`

  - Mesure de l'arête $|\sigma|$ et distance $d_{\mathcal{K}\mathcal{L}}$

    `maillage.aretes(k,_MES)`, `maillage.aretes(k,_DKL)`

  - Mesures des quarts de diamant

    `maillage.aretes(k,_MES_K_DEB)`, ...

  - Label (`>=0` à l'interieur, `=-1` bord Dirichlet, `<-1` bord Neumann)

    `maillage.aretes(k,_LABEL)`.

Autres fonctions utiles :

- Pour évaluer une fonction sur les sommets ou sur les centres

```
evaluation(m.centres,f)

evaluation(m.sommets,f)
```

- Pour dessiner une fonction constante par mailles $u^{\mathcal{T}}$

```
trace_fonction(m,u,nwin,ep_maillage)
```

  - `m` est le maillage
  - `u` un vecteur colonne de taille `m.nb_vol` contenant les valeurs
  - `nwin` est le numéro de la fenêtre graphique ou l'on veut tracer
  - `ep_maillage` est l'épaisseur du trait pour dessiner le maillage (0 si on ne veut pas afficher le maillage).

- Pour calculer la norme L^2 de $u^{\mathcal{T}}$

```
norme_L2(m,u)
```

- Pour calculer une norme H^1_0 discrète de $u^{\mathcal{T}}$

```
norme_H1(m,u)
```

Les jeux de données :

Structure générale similaire au cas 1D

- Un jeu de données `donnees` est une **structure** qui contient
  - `donnees.nom` : le nom du cas test.
  - `donnees.source` : le terme source (une fonction $\mathbb{R} \to \mathbb{R}$).
  - `donnees.bordD` : les données au bord (une fonction $\mathbb{R} \to \mathbb{R}$).
  - `donnees.uexacte` : la solution exacte (une fonction $\mathbb{R} \to \mathbb{R}$).
  - `donnees.coeff_k` : le coeff de diffusion (une fonction $\mathbb{R} \to \mathbb{R}$).
  - `donnees.maillages` : donne (éventuellement) une liste de noms de maillages spécifiquement utilisables pour ce cas test.
  - + d'autres paramètres éventuellement utiles pour le calcul.

LES JEUX DE DONNÉES :

Structure générale similaire au cas 1D

- Un jeu de données **donnees** est une **structure** qui contient
  - **donnees.nom** : le nom du cas test.
  - **donnees.source** : le terme source (une fonction $\mathbb{R} \to \mathbb{R}$).
  - **donnees.bordD** : les données au bord (une fonction $\mathbb{R} \to \mathbb{R}$).
  - **donnees.uexacte** : la solution exacte (une fonction $\mathbb{R} \to \mathbb{R}$).
  - **donnees.coeff_k** : le coeff de diffusion (une fonction $\mathbb{R} \to \mathbb{R}$).
  - **donnees.maillages** : donne (éventuellement) une liste de noms de maillages spécifiquement utilisables pour ce cas test.
  - + d'autres paramètres éventuellement utiles pour le calcul.
- Liste de jeux de données : `cas_test=list(...,...,...)`
  Pour tout entier `i` convenable `cas_test(i)` est un jeu de données.
- Pour rajouter un jeu de données :

```
cas_test($+1)=struct("nom",xxxx,...
                     "uexacte",xxxx,...
                     "bordD",xxxx,...
                     "source",xxxx,...
                     "coeff_k",xxxx);
```

Fonctions d'assemblage des schémas

- Syntaxe générale : `[A,b]=const_schema_xxxx(m,donnees)`
  - Entrée :
    - `m` est un maillage
    - `donnees` un jeu de données
  - Sortie :
    - `A` est la matrice du système
    - `b` le second membre
- **Trois schémas prévus mais ne sont pas codés :**
  - `xxxx=VF4` : le schéma VF4
  - `xxxx=DDFV_iso` : le schéma DDFV diffusion isotrope
  - `xxxx=DDFV_aniso` : le schéma DDFV diffusion anisotrope

Structure du programme `VF2D.sce`

- Chargement des libraires de fonctions.
- Interrogation utilisateur :
  - Choix du cas test
  - Choix du maillage (la liste des maillages proposée dépend du cas test)
  - S'il s'agit d'une famille de maillages : choix du niveau de raffinement.
  - Choix du schéma à utiliser (`VF4`, `DDFV_iso` ou `DDFV_aniso`).
- Lecture du maillage
- **Assemblage du schéma**
- Résolution du système linéaire (par UMFPACK si disponible)
- Tracé de la solution approchée (et exacte si disponible) et évaluation de l'erreur.

Structure du programme `courbes_erreur2D.sce`

- Chargement des libraires de fonctions.
- Interrogation utilisateur :
  - Choix du cas test
  - Choix d'une **famille** de maillages
  - Choix des niveaux de raffinement min et max souhaités
  - Choix du schéma à utiliser (`VF4`, `DDFV_iso`, `DDFV_aniso`).
- Pour chaque maillage choisi :
  - Lecture du maillage
  - **Assemblage du schéma**
  - Résolution du système linéaire (par UMFPACK si disponible)
  - Evaluation et stockage de l'erreur.
- Tracé des courbes d'erreur en fonction du pas du maillage en échelle logarithmique.

1. Programmation de VF4 pour la diffusion isotrope homogène + Dirichlet homogène

   fonction `const_schema_VF4`

2. Prendre en compte les conditions de Dirichlet non-homogène
3. Prendre en compte un coefficient de diffusion (scalaire !) variable régulier

   fonction `calcul_coeff_k_interf`
   cas où `donnees.methode=="exacte"`

4. Prendre en compte un coefficient de diffusion (scalaire !) discontinu à travers les arêtes du maillage

   cas où `donnees.methode=="arithmetique"`

   cas où `donnees.methode=="harmonique"`

1. Programmation de DDFV isotrope (coefficients scalaires !) :
   - Avec conditions aux limites de Dirichlet homogène
   - Avec conditions aux limites de Dirichlet non-homogène
2. Tests de convergence sur diverses familles de maillages et des solutions analytiques.
3. Vérification de la robustesse de la méthode.

Pour les courageux :

1. Programmation de DDFV anisotrope : tenseur de diffusion plein.

Inconnues et equations

- Il y a maintenant `m.nb_vol + m.nb_som` inconnues et équations.

Inconnues primales : `u(1:m.nb_vol)`,

Inconnues duales : `u(1+m.nb_vol:m.nb_som+m.nb_vol)`.

Informations géométriques à utiliser

- Normale primale de `K` vers `L`

`m.aretes(i,_N_KL_X)`, `m.aretes(i,_N_KL_Y)`

- Normale duale de `DEB` vers `FIN`

`m.aretes(i,_NS_DEBFIN_X)`, `m.aretes(i,_NS_DEBFIN_Y)`

Inconnues et equations

- Il y a maintenant `m.nb_vol + m.nb_som` inconnues et équations.

Inconnues primales : `u(1:m.nb_vol)`,

Inconnues duales : `u(1+m.nb_vol:m.nb_som+m.nb_vol)`.

Informations géométriques à utiliser

- Normale primale de `K` vers `L`

`m.aretes(i,_N_KL_X)`, `m.aretes(i,_N_KL_Y)`

- Normale duale de `DEB` vers `FIN`

`m.aretes(i,_NS_DEBFIN_X)`, `m.aretes(i,_NS_DEBFIN_Y)`

Pour les courageux

- Dans les jeux de données

$$A = \begin{pmatrix} \texttt{donnees.coeff\_mat.Axx} & \texttt{donnees.coeff\_mat.Axy} \\ \texttt{donnees.coeff\_mat.Axy} & \texttt{donnees.coeff\_mat.Ayy} \end{pmatrix}$$

- **Fourni :** Calcul des composantes du tenseur $A_{\mathcal{D}}$ dans la base $(\boldsymbol{\nu}_{\mathcal{K}}, \boldsymbol{\nu}_{\mathcal{K}^*})$

`[ADNN,ADNNS,ADNSNS]=calcul_coeff_mat(m,donnees)`

Inconnues et equations

- Il y a maintenant `m.nb_vol + m.nb_som` inconnues et équations.

Inconnues primales : `u(1:m.nb_vol)`,

Inconnues duales : `u(1+m.nb_vol:m.nb_som+m.nb_vol)`.

Informations géométriques à utiliser

- Normale primale de `K` vers `L`

`m.aretes(i,_N_KL_X)`, `m.aretes(i,_N_KL_Y)`

- Normale duale de `DEB` vers `FIN`

`m.aretes(i,_NS_DEBFIN_X)`, `m.aretes(i,_NS_DEBFIN_Y)`

Pour les courageux

- Dans les jeux de données

$$A = \begin{pmatrix} \texttt{donnees.coeff\_mat.Axx} & \texttt{donnees.coeff\_mat.Axy} \\ \texttt{donnees.coeff\_mat.Axy} & \texttt{donnees.coeff\_mat.Ayy} \end{pmatrix}$$

- **Fourni :** Calcul des composantes du tenseur $A_{\mathcal{D}}$ dans la base $(\boldsymbol{\nu}_{\mathcal{K}}, \boldsymbol{\nu}_{\mathcal{K}^*})$

`[ADNN,ADNNS,ADNSNS]=calcul_coeff_mat(m,donnees)`

Au travail Thierry !