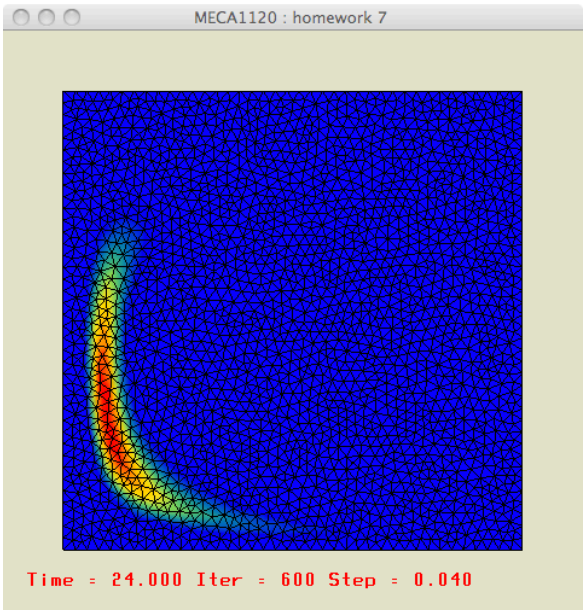
Finite elements for dummies : Méthode de Galerkin discontinue pour une équation de transport



Considérons un océan défini par $\Omega =]0,1[\times]0,1[$.
Il s'agit d'obtenir $c(t,x,y)$ qui satisfait :

$$\frac{\partial c}{\partial t} + \frac{\partial}{\partial x}\Big(uc\Big) + \frac{\partial}{\partial y}\Big(vc\Big) \quad = \quad 0$$

où la vitesse $\mathbf{v}(x,y) = (u(x,y), v(x,y))$ est connue, tandis que $c(t,x,y)$ peut être interprétée comme une concentration d'un traceur passif quelconque. Plus précisément, on utilisera, comme champs de vitesse, la solution analytique de Stommel qui forme un écoulement incompressible. La frontière est parfaitement imperméable et il est inutile de spécifier une valeur de la concentration sur les frontières du domaine.

L'objectif de ce travail est de capturer les premiers instants du transitoire en intégrant avec une méthode d'Euler explicite les équations discrètes obtenues avec la méthode de Galerkin discontinues pour des éléments triangulaires linéaires. Sur chaque élément Ω_e, la solution à un instant t est approchée de manière indépendante par :

$$c(t,x,y) \;\approx\; c_e^h(t,x,y) \;\approx\; \sum_{i=1}^{3} C_i^e(t)\;\phi_i(x,y)$$

Donc, il y a $3n$ degrés de liberté, si n est le nombre de triangles du domaine. On ajoutera -en outre- un degré de liberté associé au monde extérieur du domaine de calcul.

A tout instant, on obtiendra la solution en résolvant pour chaque triangle :

$$\sum_{j=1}^{3} \underbrace{< \phi_i \phi_j >_e}_{A_{ij}^e = \hat{A}_{ij} J_e} \frac{dC_j^e}{dt} \quad = \quad \overbrace{< \left(u\frac{\partial \phi_i}{\partial x} + v\frac{\partial \phi_i}{\partial y} \right) c_e^h >_e + \ll \phi_i \beta c^* \gg_e}^{B_i^e(c^h)} \qquad i = 1,\ldots,3$$

En multipliant par l'inverse de la matrice de masse,

$$\frac{dC_i^e}{dt} \quad = \quad \underbrace{\sum_{j=1}^{3} \hat{A}_{ij}^{-1} B_j^e(c^h)/J_e}_{F_i^e(c^h)} \qquad i = 1,\ldots,3$$

avec la vitesse normale sur un segment définie par $\beta = (un_x + vn_y)$. Le vecteur $\mathbf{n} = (n_x, n_y)$ est le vecteur normal unitaire au segment défini avec une convention d'orientation unique pour tous les segments

orientés. En fonction du signe de β, on prendra la valeur de la concentration c^* sur l'élément de droite ou de gauche par rapport au segment orienté. Ce choix doit évidemment être identique pour les deux triangles partageant un même côté. Sur les segments frontières, nous avons $\beta = 0$ et la contribution des intégrales de ligne devrait théoriquement être nulle [1]. En pratique, nous n'imposerons cela que de manière approchée en fixant une vitesse nulle et une concentration nulle sur le degré de liberté correspondant à l'extérieur du domaine. L'intégration temporelle des équations différentielles ordinaires pour chaque degré de liberté sera effectuée avec une méthode d'Euler explicite : l'implémentation de l'intégration temporelle a été réalisée par nos soins dans le programme principal.

Nous allons aussi tirer profit du fait que la matrice de masse est quasiment identique sur tous les triangles à un facteur constant égal à la superficie du triangle. On calcule donc $\hat{A}_{ij}$ sur l'élément, on l'inverse une unique fois tout au début du programme. Ensuite, il est immédiat d'observer que

$$\left(A^e_{ij}\right)^{-1} = \hat{A}^{-1}_{ij}/J_e$$

où J_e est le jacobien de l'élément. L'inverse de la matrice de masse sur l'élément parent $\hat{A}^{-1}_{ij}$ peut être calculé a priori et il suffit donc de la définir comme suit :

```
double invA[3][3] = {{18.0,-6.0,-6.0},{-6.0,18.0,-6.0},{-6.0,-6.0,18.0}};
```

Une petite structure reprend la description de notre problème :

```
typedef struct {
    femMesh *mesh;
    femEdges *edges;
    femDiscrete *space;
    femIntegration *rule1d;
    femIntegration *rule2d;
    int size;
    double *C;
    double *U;
    double *V;
    double *F;
} femAdvProblem;
```

où le `C` contient les valeurs discrètes de concentration, tandis que `U` et `V` contiennent les deux composantes de la vitesse. On y inclut aussi des pointeurs vers les éléments et les segments du maillage. Le vecteur `F` contiendra le membre de droite qui sera successivement B_i et puis F_i.

Les fonctions permettant de créer un nouveau problème à partir d'un fichier contenant un maillage, ainsi que le destructeur sont fournies par nos bons soins. Il faut encore imposer une condition initiale qui sera ici définie par :

$$c(0,x,y) \;=\; \exp\left(-\frac{(x-0.8)^2}{0.01}\right)\exp\left(-\frac{(y-0.5)^2}{0.01}\right)$$

Pour effectuer cela, une fonction vous est également fournie !

```
void advInitialCondition(femProblem *myProblem);
```

[1] Est-ce vraiment le cas en réalité ?

qui calcule l'interpolation linéaire discontinue du champs de vitesse et de la condition initiale de concentration. On pourrait se demander pourquoi, on calculer une telle interpolation du champs de vitesse dont nous avons une expression analytique. La raison est assez pragmatique : le calcul de la solution analytique de Stommel requiert énormément de ressources de calcul et ne pas répéter ce calcul dans l'intégration temporelle se révèlera donc une façon très efficace d'accélérer la vitesse d'exécution de notre programme.

Plus précisément, on vous demande de concevoir, d'écrire ou de modifier cinq fonctions.

1. Tout d'abord, il s'agit d'écrire deux fonctions pour la gestion des degrés de liberté.

   ```
   void femAdvTriangleMap(femAdvProblem *myProblem, int i, int map[3]);
   void femAdvEdgeMap(femAdvProblem *myProblem, int i, int map[2][2]);
   ```

   Ces deux fonctions doivent permettre d'obtenir les indices des degrés de liberté linéaires discontinus de concentration pour un triangle ou un segment. Pour un segment `map[0]` contiendra les degrés de liberté situé dans le triangle situé à gauche du segment orienté, tandis que `map[1]` contiendra ceux du triangle de droite. Pour les segments frontières, on associera au monde extérieur un degré de liberté supplémentaire $3n$.

2. Finalement, il s'agira d'écrire les trois fonctions permettant d'obtenir tous les terme F_i^e stockés dans le grand vecteur `myProblem->F` de taille $3n + 1$. Le dernier élément de ce vecteur est également associé à un degré de liberté global supplémentaire pour le monde extérieur.

   ```
   void femAdvAddIntegralsTriangles(femAdvProblem *myProblem);
   void femAdvAddIntegralsEdges(femAdvProblem *myProblem);
   void femAdvMultiplyInverseMatrix(femAdvProblem *myProblem);
   ```

   La première fonction accumule toutes les intégrales de surface apparaissant dans les termes B_i^e :

   $$< \left(u\frac{\partial \phi_i}{\partial x} + v\frac{\partial \phi_i}{\partial y} \right) c_e^h >_e$$

   La seconde fonction accumule toutes les intégrales de lignes apparaissant dans les termes B_i^e :

   $$\ll \phi_i \beta c^* \gg_e$$

   La troisième fonction effectue le produit matriciel par l'inverse de la matrice de masse et veille à ce que le résultat final se trouve dans le vecteur `myProblem->F`. Attention, il faut être bien attentifs dans la manière d'effectuer ce calcul pour éviter des petites erreurs malencontreuses :-)

   $$F_i^e = \sum_{j=1}^{3} \hat{A}_{ij}^{-1} B_j^e / J_e$$

   Ces trois fonctions forment les trois étapes du calcul du membre de droite de la méthode d'Euler effectué dans la boucle du programme principal :

   ```
   while (theTime >= theDiscreteTime)
   {
       theIteration += 1;
       theDiscreteTime += theTimeStep;
       for (i=0; i < size; i++)
         F[i] = 0.0;
       femAdvAddIntegralsTriangles(theProblem);
       femAdvAddIntegralsEdges(theProblem);
       femAdvMultiplyInverseMatrix(theProblem);
       for (i=0; i < size; i++)
         C[i] += theTimeStep * F[i];
   }
   ```

3. A propos, que contient le dernier élément du vecteur `myProblem->F` à la fin du calcul ?
   Quelle est la valeur que vous devriez idéalement obtenir ?

4. Toutes vos fonctions seront incluses dans un unique fichier `homework.c`.
   Toujours bien vérifier que la compilation s'exécute correctement sur le serveur.

5. Afin de pouvoir effectuer un test distinct de vos fonctions, des instructions de compilation ont été mis dans le fichier `homework.c`. Il est IMPERATIF de ne pas les retirer, ni de les modifier...