TP5 : Résolution de l'équation d'advection 2D en volumes finis

Introduction du TP

Il s'agit du dernier TP noté. L'évaluation sera effectuée sur la base d'un compte-rendu présentant les différentes améliorations que vous avez apportées au code et les résultats de son exploitation sur les cas proposés et tous ceux que vous trouverez pertinent. Il vous sera également demandé de déposer une archive (.zip ou .tar.gz) de votre code source afin qu'il puisse être évalué.

Modèle d'advection-diffusion

Dans ce TP, nous nous intéressons au problème physique du transport (advection + diffusion) d'une certaine quantité (qui pourrait être une quantité de matière, de chaleur, et pourquoi pas de mouvement) dans un volume donné. Ce problème est très courant en mécanique, notamment des fluides, bien évidemment. Comme vous l'avez vu en cours, ce problème, même si il paraît simple, n'est pas si aisé à résoudre en pratique. Nous nous baserons donc sur l'équation d'advection-diffusion sous forme conservative avec terme source :

$$\left\{ \begin{array}{rcl} \frac{\partial \phi}{\partial t} + \nabla \cdot (\mathbf{V}\phi) - \nabla \cdot (\mu \nabla \phi) & = & S, \\ \phi(\vec{x}, t=0) & = & \phi_0(\vec{x}), \end{array} \right. \tag{1}$$

où $\phi(\mathbf{x}, t)$ représente la quantité transportée, $\mu(\mathbf{x}, t)$ le coefficient de diffusion et $\mathbf{V}(\mathbf{x}, t)$ est un champ de vitesse (par exemple, la vitesse de l'écoulement fluide sous-jacent). S est ici un **terme source** qui permet aisément de faire varier ϕ *de l'extérieur* (source de chaleur, injection ou absorption de matière, etc.).

Pour résoudre cette équation, nous allons utiliser la méthode des Volumes Finis (notée VF par la suite). Nous considérerons une triangulation $\mathcal{T}_h$ d'un domaine 2D et nous chercherons la solution approchée dans l'espace des fonctions constantes sur chaque triangle T_i :

$$\mathcal{V}_h = \Big\{ \phi \text{ telle que } \phi_{|T_i} = \bar{\phi}_i \in \mathbb{R},\ \forall T_i \in \mathcal{T}_h \Big\},$$

où $\bar{\phi}_i = \frac{1}{|T_i|} \int_{T_i} \phi(\mathbf{x}) d\mathbf{x}$ est la valeur moyenne de ϕ dans le triangle T_i. Par la suite, nous omettrons la barre sur $\bar{\phi}_i$ pour faciliter la lecture.

La résolution du problème passe, dans un premier temps, par l'intégration de l'équation dans chaque volume de contrôle T_i :

$$\int_{T_i} \partial_t \phi + \nabla \cdot (\mathbf{V}\phi) - \nabla \cdot (\mu \nabla \phi)\, d\mathbf{x} = \int_{T_i} S\ d\mathbf{x}, \tag{2}$$

$$\partial_t \bar{\phi}_i + \int_{T_i} \nabla \cdot (\mathbf{V}\phi) - \nabla \cdot (\mu \nabla \phi)\, d\mathbf{x} = \bar{S}_i. \tag{3}$$

qui, par intégration par parties, peut être réécrite :

$$\partial_t \bar{\phi}_i + \int_{\partial T_i} ((\mathbf{V}\phi) \cdot \boldsymbol{n}) - ((\mu \nabla \phi) \cdot \boldsymbol{n})\, d\mathbf{x} = \bar{S}_i.$$

1 Annabelle Collin et Mathieu Coquerelle - pour nous contacter au sujet de ce TP : annabelle.collin@inria.fr et mathieu.coquerelle@bordeaux-inp.fr

où $\boldsymbol{n}$ représente la normale sortante au triangle T_i sur son pourtour. **Ce calcul devra être détaillé dans le rapport**. Cette stratégie est également utilisée par le biais du théorême de Green-Ostrogradski.

Les stratégies d'intégration temporelle, sont quant-à elles, considérées comme acquises.

Résolution à l'aide des volumes finis

Les trois principales difficultés d'un code de VF résident en :

- la discrétisation correcte de chaque terme – et notamment des flux – de l'équation ;
- la gestion correcte des conditions de bord ;
- l'assemblage des différents termes sous la forme d'un système linéaire puis sa résolution.

Documentation de la classe DataFile

Comme pour le TP précédent, vous utiliserez la classe *DataFile* pour lire le fichier de données. Voici une rapide documentation de la classe *DataFile* (rappels).

1. Construction d'un objet de la classe *DataFile* et lecture du fichier :

```
DataFile* data_file = new DataFile("data.txt");
```

2. Récupération des paramètres :

Nom de la Fonction	Type renvoyé	Variable
Get_t0()	double	t_0
Get_tfinal()	double	T_{final}
Get_dt()	double	Δt
Get_scheme()	string	Schémas : ExplicitEuler ; ImplicitEuler
Get_mesh_name()	string	Nom du maillage
Get_numerical_flux_choice()	string	Choix du flux numérique : Upwind ; Centered
Get_results()	string	Nom du dossier résultats
Get_BC_type()	vector<string>	Choix des conditions aux bords : Neumann ; Dirichlet

Documentation de la classe Function

D'autres informations ont besoin d'être données pour pouvoir résoudre le problème :

- la condition initiale $(x, y) \mapsto \phi_0(x, y)$,
- le terme source $(t, x, y) \mapsto S(t, x, y)$,
- la vitesse $(t, x, y) \mapsto V(t, x, y)$,
- la solution exacte $(t, x, y) \mapsto \phi_{exacte}(t, x, y)$ si elle existe,
- les conditions aux bords.

Toutes ces informations sont données dans la classe *Function*.

1. Construction d'un objet de la classe *Function* :

```
Function* fct = new Function(data_file);
```

2. Exemple d'utilisation sur la fonction *Source_term* :

```
double x(2), y(4), t(0.1);
// S(x=2,y=4,t=0.1)
double S = fct->Source_term(x,y,t);
```

Documentation de la classe Mesh2D

Nous cherchons à résoudre l'équation d'advection-diffusion dans un domaine quelconque, pas nécessairement rectangulaire, ce qui ce rend l'utilisation d'une méthode de Différences Finies (DF) impossible.

Tout code de VF est basé sur un maillage sous-jacent. Générer, écrire, lire et manipuler un maillage n'est pas si simple qu'il n'y parait, surtout en 3D. C'est pourquoi nous nous focaliserons sur un **maillage 2D non structuré** [1] basé sur des triangles (aussi nommés *cellules* ou *volumes de contrôle*). Ce maillage pourra être généré, par exemple, par le logiciel `Gmsh` ; il sera au format « .mesh » qui est un format « brut » aisément lisible *à l'oeil nu*. En contre-partie, ce format présente le strict nécessaire pour décrire un maillage, et pas plus.

0. Dans un premier temps, pour vous familiariser avec ce format et la gestion du maillage, parcourez le fichier `Meshes/square_mini.mesh`. Imaginez et dessinez comment les liens entre les différents éléments géométriques, les indices associés, et ce qu'il faudrait connaître/calculer, dans le cadre des VF, pour permettre les calculs des flux.

Voici une documentation de la classe *Mesh2D* :

1. Construction d'un objet de la classe *Mesh2D* :

```
Mesh2D* mesh = new Mesh2D();
```

2. Lecture du maillage (et construction des informations nécessaires à la méthode des volumes finis) :

```
mesh->Read_mesh(data_file->Get_mesh_name());
```

3. Récupération des paramètres :

Nom de la Fonction	Type renvoyé	Variable
Get_vertices()	vector<Vertex>	Vec. : les points
Get_triangles()	vector<Triangle>	Vec. : les triangles
Get_triangles_center()	Matrix<double, Dyn., 2>	Mat. (N_tri x 2) : les centres des triangles
Get_triangles_area()	VectorXd	Vec. (N_tri) : les aires des triangles
Get_edges()	vector<Edge>	Vec. : toutes les edges
Get_edges_length()	VectorXd	Vec. (N_edges) : les longueurs des arêtes
Get_edges_normal()	Matrix<double, Dyn., 2>	Mat. (N_edges x 2) : les normales des arêtes
Get_edges_center()	Matrix<double, Dyn., 2>	Mat. (N_edges x 2) : les centres des arêtes

4. Deux exemples d'utilisation de la classe :
Si on souhaite récupérer la valeur du terme source au centre du triangle i au temps t :

1. Un maillage non structuré est un maillage composé de cellules qui n'ont pas d'ordre particulier et dont les connexions (*quel est le numéro de la cellule voisine à celle étudiée ?*) sont quelconques. À l'opposé, un maillage cartésien est un maillage structuré : il est *direct* de retrouver l'indice d'une cellule voisine, comme vous l'avez constaté dans le TP sur les DF 2D.

```
double S = fct->Source_term(msh->Get_triangles_center()(i,0),
                            msh->Get_triangles_center()(i,1), t);
```

Si on souhaite récupérer la référence du triangle 1 de l'arête i et le type de conditions aux bords (si c'est bien une arête de bord !) :

```
int t1 = _msh->Get_edges()[i].Get_T1();
string BC = _msh->Get_edges()[i].Get_BC(); //Renvoie "Dirichlet" ou "Neumann"
```

Algorithme général *basé sur les volumes de contrôle*

Algorithme

- Lire le fichier de données ;
- Lire le maillage ;
- Initialiser les valeurs ϕ_i^0 en chaque cellule i ;
- Pour chaque pas de temps $t^n \to t^{n+1}$:
  - Pour chaque volume de contrôle Ω_i : **exprimer les flux avec le schéma numérique spatial choisi** et **appliquer les conditions de bord** ;
  - Remplir le système linéaire sous forme matricielle : $M\phi^{n+1} = RHS$ avec le schéma numérique temporel choisi, où ϕ^{n+1} est le vecteur formé des inconnues ϕ_i^{n+1} en chaque triangle T_i ;
  - Résoudre le système matriciel pour obtenir ϕ^{n+1}.

Calcul de la solution

Le problème d'advection diffusion (1) peut être écrit sous la forme générale :

$$\frac{d\phi}{dt} = -F(t,\phi) + S(t).$$

Avec une méthode explicite, l'intégration de l'équation précédente nous donne :

$$\phi^{n+1} = \phi^n - \delta t\, F\left(t^n, \phi^n\right) + \delta t\, S^n. \tag{4}$$

Avec une méthode implicite, l'intégration de l'équation nous donne :

$$\phi^{n+1} = \phi^n - \delta t\, F\left(t^{n+1}, \phi^{n+1}\right) + \delta t\, S^{n+1}. \tag{5}$$

Dans le cas général (i.e. F potentiellement non-linéaire), ce problème est relativement compliqué et il peut s'avérer nécessaire d'utiliser une méthode de type Newton (par exemple). Dans notre cas, le terme F dépend linéairement de ϕ ; on peut alors écrire :

Flux sous forme matricielle

$$F\left(t, \phi^{n+¤}\right) = A\phi^{n+¤} + b \tag{6}$$

où A est une matrice carrée et b un vecteur. Ici, le symbole ¤ est utilisé pour généraliser : si ¤ = 0 alors c'est un schéma explicite en temps et si ¤ = 1 alors c'est un schéma implicite.

On peut alors réécrire l'équation explicite (4) sous la forme :

$$\phi^{n+1} = \phi^n - \delta t \left(A\phi^n + b\right) + \delta t\, S^n$$

qu'on peut calculer directement (i.e. *explicitement*) à partir des valeurs connues de ϕ^n, b et S.

On peut alors aussi récrire l'équation implicite (5), en réarrangeant les termes, sous la forme :

$$\left(Id + \delta t A\right) \phi^{n+1} = \phi^n - \delta t\, b + \delta t\, S^{n+1}$$

qui est plus compliqué à résoudre puisqu'il faut inverser la matrice $(Id + \delta t A)$.

Cette méthodologie générale a été utilisée dans le TP 4 pour pouvoir résoudre l'équation de la chaleur sous forme DF, et nous utiliserons ce même principe pour les VF. La difficulté réside donc en l'identification des éléments (non nuls) de A et de b. Ceux-ci sont les mêmes, que la méthode temporelle soit explicite ou implicite.

Composition de A et b

- En DF, pour la résolution de l'équation de la chaleur, la matrice A était composée de l'opérateur Laplacien : on retrouvait ainsi des lignes composées de termes $-\frac{1}{h_y^2}; \cdots; -\frac{1}{h_x^2}; \frac{2}{h_x^2} + \frac{2}{h_y^2}; -\frac{1}{h_x^2}; \cdots; -\frac{1}{h_y^2}$. Ceci à quelques exceptions près : l'introduction des conditions de bord venait modifier certaines lignes de la matrice et remplir le vecteur b en conséquence.
- En VF, pour la résolution de l'équation d'advection-diffusion, les termes $A_{i,k}$ de la matrice A vont dépendre des flux entre les triangles T_i et T_k du maillage. **Le calcul de ces flux et la construction de la matrice associée est la partie compliquée du problème que nous allons implémenter**. De la même manière qu'en DF, le vecteur b sera utile pour gérer les conditions de bord.

Calcul des flux

Dans les étapes 1 et 2, nous décrivons comment traiter dans le cas général le calcul des flux pour les triangles internes du maillages ; dans ce cas, le vecteur $b = 0$. Les conditions de bord seront décrites dans l'étape 3 (b sera alors différent du vecteur nul).

Étape 1 : comprendre le principe (basé sur les volumes de contrôle)

Dans cette première étape, nous cherchons à remplir la matrice A. Pour ce faire, nous allons calculer l'ensemble des flux associés à chaque triangle T_i.

Comme vu dans le polycopié de préparation, on peut exprimer la somme des flux sur les bords du triangle T_i de la manière suivante :

$$\left(F\left(t, \phi\right)\right)_i = F_i = \sum_{k \in \text{voisins}(i)} \widetilde{F_{e_{i,k}}} = \frac{1}{|T_i|} \sum_{k \in \text{voisins}(i)} |e_{i,k}|\, \overline{F}_{e_{i,k}} \tag{7}$$

où $|T_i|$ est l'aire du triangle i, $e_{i,k}$ l'arête entre T_i et T_k, $|e_{i,k}|$ la longueur de celle-ci. Ici, $\widetilde{F_{e_{i,k}}}$ dénote la contribution (positive ou négative) du voisin k à la variation de ϕ_i par le flux associé (donc entre T_i et T_k) ; son unité est en ϕ/s.
En décomposant, $\widetilde{F_{e_{i,k}}}$, nous pouvons définir $\overline{F}_{e_{i,k}}$ avec $\widetilde{F_{e_{i,k}}} = \frac{1}{|T_i|}|e_{i,k}|\,\overline{F}_{e_{i,k}}$: c'est la valeur moyenne du flux traversant $e_{i,k}$; son unité est donc en $\phi/s/m$. Par convention, on considère que le flux va de l'intérieur du triangle T_i vers l'extérieur (en direction de T_k, quel que soit T_k voisin).

Ce flux *numérique* moyen $\overline{F}_{e_{i,k}}$ va être calculé en fonction des valeurs (potentiellement inconnues) de ϕ associées aux triangles T_i et T_k, **et uniquement de celles-ci**. On peut donc l'écrire sous la forme :

Flux moyen entre T_i et T_k :

$$\overline{F}_{e_{i,k}} = \alpha_{e_{i,k}}\phi_i + \beta_{e_{i,k}}\phi_k \tag{8}$$

où $\alpha_{e_{i,k}}$ et $\beta_{e_{i,k}}$ vont dépendre :

— des types des flux : flux diffusif ou advectif,

— de leur discrétisation : schéma centré ou amont.

Ainsi, en assemblant les équations (7) et (8) et en identifiant avec la définition (6), nous sommes capables d'écrire les lignes de la matrice A. Par exemple, pour un triangle T_i (i.e. $i^{ième}$ équation, i.e. ligne i de la matrice) et ses trois voisins T_{k_1}, T_{k_2} et T_{k_3}, on peut généraliser la forme de la matrice A (rappel : $F = A\phi$ lorsque $b = 0$) telle que :

$$\begin{pmatrix} \vdots \\ F_i \\ \vdots \end{pmatrix} = \begin{pmatrix} & & & \cdots & & & \\ \frac{|e_{i,k_1}|\beta_{e_{i,k_1}}}{|T_i|} & \cdots & \frac{\sum_{l\in\{1,2,3\}}|e_{i,k_l}|\alpha_{e_{i,k_l}}}{|T_i|} & \cdots & \frac{|e_{i,k_2}|\beta_{e_{i,k_2}}}{|T_i|} & \cdots & \frac{|e_{i,k_3}|\beta_{e_{i,k_3}}}{|T_i|} \\ & & & \cdots & & & \end{pmatrix} \begin{pmatrix} \phi_{k_1} \\ \vdots \\ \phi_i \\ \vdots \\ \phi_{k_2} \\ \vdots \\ \phi_{k_3} \end{pmatrix}$$

où tous les autres termes de la ligne i sont nuls.

Les valeurs des coefficients α et β dépendent du type et de la discrétisation du flux désiré. Pour séparer le traitement des flux diffusifs (D) et advectifs (A), nous décomposons $\alpha = \alpha^D + \alpha^A$ et $\beta = \beta^D + \beta^A$. Ainsi, pour :

- le flux diffusif avec un schéma centré : $\overline{F}_{e_{i,k}} \simeq -\mu\frac{\phi_k - \phi_i}{\delta_{i,k}} \Rightarrow \alpha^D = \frac{\mu}{\delta_{i,k}}$ et $\beta^D = -\frac{\mu}{\delta_{i,k}}$ où μ est le coefficient de diffusion au centre de l'arête et $\delta_{i,k}$ la distance entre les centres de T_i et T_k.
- le flux advectif avec un schéma centré : $\overline{F}_{e_{i,k}} \simeq \frac{\phi_i + \phi_k}{2} \cdot v_n \Rightarrow \alpha^A = \beta^A = \frac{v_n}{2}$ où v_n est la vitesse normale tq $v_n = \mathrm{u} \cdot \mathrm{n}$ au centre de l'arête ;
- le flux advectif avec un schéma décentré (upwind) : soit $v_n \geq 0$ (flux sortant de T_i, ne dépend que de ϕ_i) $\Rightarrow$ $\overline{F}_{e_{i,k}} \simeq \phi_i \cdot v_n \Rightarrow \alpha^A = v_n$ et $\beta^A = 0$, soit $v_n < 0$ (flux entrant depuis T_k, ne dépend que de ϕ_k) $\Rightarrow$ $F_{e_{i,k}} \simeq \phi_k \cdot v_n \Rightarrow \alpha^A = 0$ et $\beta^A = v_n$;

Étape 2 : implémenter la méthode (basée sur les flux !)

Une méthodologie *basée sur les flux* pour les VF nous permettrait d'assurer la conservativité de la méthode. La stratégie est basée sur un principe simple : **toute quantité qui est retranchée d'un côté et ajoutée de l'autre** (et inversement). Exemple : si il y a un flux de chaleur de $+5J$ de T_i vers T_k, cela signifie que T_i *perd* $5J$ et que T_k *gagne* $5J$.

Concrètement : avec la méthodologie précédente basée sur les *volumes de contrôle*, on se rend rapidement compte qu'on calcule deux fois les mêmes flux. En effet, on remarque que le flux *numérique* $\overline{F}_{e_{i,k}}$ (de T_i vers T_k) est égal à l'opposé du flux *numérique* $\overline{F}_{e_{k,i}}$ (de T_k vers T_i), soit : $\overline{F}_{e_{i,k}} = -\overline{F}_{e_{k,i}}$. Or, *en réalité*, il n'y a qu'un **unique** flux, noté $\mathbf{j}$, qui correspond au flux *physique* de ϕ ; celui-ci est projeté au travers d'une surface (ici une arête) entre deux volumes de contrôle. En d'autres termes : la quantité de ϕ retranchée (resp. ajoutée) à l'un des volumes de contrôle est ajoutée (resp. retranchée) à l'autre. En reprenant le lien entre flux physique et flux numérique, nous pouvons écrire :

$$\overline{F}_{e_{i,k}} = \mathbf{j} \cdot \mathbf{n}_{i,k} = -\mathbf{j} \cdot \mathbf{n}_{k,i} = -\overline{F}_{e_{k,i}}$$

où, géométriquement, l'arête $e_{i,k} = e_{k,i}$ possède une **unique** normale $\mathbf{n}_{i,k} = -\mathbf{n}_{k,i}$ orientée de T_i vers T_k, au signe près : en effe,t cela ne change rien à l'unicité du flux physique $\mathbf{j}$.

En conséquence, il paraît alors plus efficace de faire une boucle sur les arêtes plutôt que sur les triangles. Si on considère une arête $e_{i,k}$ entre T_i et T_k, le calcul du flux $\overline{F}_{e_{i,k}}$ va donc impacter deux lignes de la matrice A : la ligne i et la ligne k.

Ainsi, en reprennant le formalisme de l'étape précédente, nous pouvons écrire :

Répartition des flux par les arêtes :

$$\begin{pmatrix} \vdots \\ F_i \\ \vdots \\ F_k \\ \vdots \end{pmatrix} = \begin{pmatrix} & & \cdots & & \\ \cdots & \frac{|e_{i,k}|\alpha_{e_{i,k}}}{|T_i|} & \cdots & \frac{|e_{i,k}|\beta_{e_{i,k}}}{|T_i|} & \cdots \\ & & \cdots & & \\ \cdots & -\frac{|e_{i,k}|\alpha_{e_{i,k}}}{|T_k|} & \cdots & -\frac{|e_{i,k}|\beta_{e_{i,k}}}{|T_k|} & \cdots \\ & & \cdots & & \end{pmatrix} \begin{pmatrix} \vdots \\ \phi_i \\ \vdots \\ \phi_k \\ \vdots \end{pmatrix} \tag{9}$$

Notez bien la variation sur le dénominateur, spécifique à chaque triangle, donc différent sur chaque ligne.

Attention

Pour remplir la matrice, il ne faut pas oublier de faire la somme des contributions de chaque arête car, au total, 3 arêtes doivent venir remplir chaque ligne. Au final, les méthodes des étapes 1 et 2 doivent donner **rigoureusement la même matrice** !

Étape 3 : gérer les conditions de bord

Nous venons de voir comment fabriquer la matrice A dans le cas de flux internes via les arêtes. Toutefois, lorsque l'arête se situe sur le bord (*boundary*) du domaine, il est nécessaire de faire un traitement particulier, comme cela a été fait dans le TP6. En effet, dans ces cas, T_k, et donc ϕ_k, n'existent pas ! Les formules générales précédentes n'y ont donc pas de sens.

Nous allons utiliser le principe des « noeuds fantômes » (*ghost* en anglais) : imaginez qu'il existe un triangle *fantôme* $\widetilde{T_k}$ de l'autre côté d'une arête de bord. Si nous sommes capables de trouver une valeur approximative de la valeur $\widetilde{\phi_k}$ associée à $\widetilde{T_k}$ à partir du champ ϕ à l'intérieur, alors nous pourrons calculer de la même manière le flux à travers cette arête :

$$\widetilde{F_{e_{i,k}}} = \frac{|e_{i,k}|\,\alpha_{e_{i,k}}}{|T_i|}\phi_i + \frac{|e_{i,k}|\,\beta_{e_{i,k}}}{|T_i|}\widetilde{\phi_k} \tag{10}$$

Condition de Neumann :

— Vérifier que pour le terme de diffusion la condition de Neumann non homogène sur le triangle i : $\partial\phi/\partial n = g$ ne modifie pas la matrice mais seulement le vecteur b :

$$\widetilde{F_{e_{i,k}}} = -\frac{|e_{i,k}|\,\mu}{|T_i|}g.$$

— Ensuite, pour le terme d'advection, la condition de Neumann non homogène peut être traduite par $\widetilde{\phi_k} = \phi_i + \delta_{ik}g$: en effet, $\frac{\partial\phi}{\partial n} \simeq \frac{\widetilde{\phi_k}-\phi_i}{\delta_{i,k}} = g$. L'équation (10) peut alors être récrite :

$$\widetilde{F_{e_{i,k}}} = \frac{|e_{i,k}|\,\alpha^A_{e_{i,k}}}{|T_i|}\phi_i + \frac{|e_{i,k}|\,\beta^A_{e_{i,k}}}{|T_i|}\widetilde{\phi_k} = \frac{|e_{i,k}|}{|T_i|}\left(\alpha^A_{e_{i,k}} + \beta^A_{e_{i,k}}\right)\phi_i + \beta^A_{e_{i,k}}\frac{|e_{i,k}|}{|T_i|}\delta_{ik}g,$$

avec $\delta_{ik} = 2\delta_{ie}$ où δ_{ie} est la distance entre le centre de l'arête et le milieu du triangle.

Condition de Dirichlet : La condition $\phi_e = h$, où ϕ_e est la valeur de ϕ sur l'arête, est un peu plus délicate.

— Pour le terme de diffusion, l'équation (10) peut être récrite :

$$\widetilde{F_{e_{i,k}}} = \frac{|e_{i,k}|\,\alpha^D_{e_{i,e}}}{|T_i|}\phi_i + \frac{|e_{i,k}|\,\beta^D_{e_{i,e}}}{|T_i|}h$$

avec $\delta_{i,e}$ ($=\delta_{i,k}/2$) la distance entre le centre du triangle i et le milieu de l'arête e et

$$\alpha^D_{i,e} = \frac{\mu}{\delta_{i,e}} \text{ and } \beta^D_{i,e} = -\frac{\mu}{\delta_{i,e}}$$

— Pour le terme d'advection, comme nous connaissons ϕ_i dans le triangle à l'intérieur et ϕ_e au bord, nous pouvons extrapoler cette valeur de l'autre côté de l'arête, au centre du triangle fantôme. Ainsi, on peut écrire un DL à l'ordre 1 : $\widetilde{\phi_k} \simeq \phi_i + \delta_{i,k}\phi_i' \simeq \phi_i + \delta_{i,k}\frac{\phi_e-\phi_i}{\delta_{i,e}}$ soit, avec $\delta_{i,e} = \delta_{i,k}/2$: $\widetilde{\phi_k} \simeq 2\phi_e - \phi_i = 2h - \phi_i$. Ainsi, dans ce cas, l'équation (10) peut être réécrite pour la partie d'advection :

$$\widetilde{F_{e_{i,k}}} = \frac{|e_{i,k}|\,\alpha^A_{e_{i,k}}}{|T_i|}\phi_i + \frac{|e_{i,k}|\,\beta^A_{e_{i,k}}}{|T_i|}(2h-\phi_i) = \frac{|e_{i,k}|}{|T_i|}\left(\alpha^A_{e_{i,k}} - \beta^A_{e_{i,k}}\right)\phi_i + 2\beta^A_{e_{i,k}}\frac{|e_{i,k}|}{|T_i|}h$$

Implémentation

La version initiale du code est disponible ici.

1. Il y a 3 fonctions à implémenter dans le code (tout le reste est déjà implémenté). La fonction

```
1 void FiniteVolume::Build_flux_mat_and_rhs(const double& t);
```

construit la matrice des flux $\_mat\_flux$ (A) et le vecteur $\_BC\_RHS$ (b). Ces variables privées de la classe *FiniteVolume* peuvent se récupérer dans la classe *TimeScheme* avec des fonctions $Get_*()$ comme dans les précédents TP.
Les 2 fonctions

```
void EulerScheme::Advance();
void ImplicitEulerScheme::Advance();
```

correspondent à UNE avancée en temps des schémas d'Euler explicite et implicite.
Il est tout à fait possible d'ajouter des fonctions intermédiaires pour construire ces 3 fonctions.
Pour valider votre code, des cas tests correspondants à différents scénarios ont été créés :

- diffusion_hom_neumann : solution obtenue avec $V = 0$ (pas d'advection) et des conditions de Neumann homogènes ! Cela permet de construire la matrice correspondant seulement à l'équation de diffusion (avec $b = 0$).
- diffusion_all_BC : solution obtenue avec $V = 0$ (pas d'advection) et toutes conditions aux bords entre Neumann et Dirichlet (à changer dans le fichier de données).
- advection_hom_neumann : solution obtenue avec $\mu = 0$ (pas de diffusion) et des conditions de Neumann homogènes ! Cela permet de construire dans un premier temps la matrice correspondant seulement à l'équation d'advection (et d'avoir toujours $b = 0$).
- advection_all_BC : solution obtenue avec $\mu = 0$ (pas de diffusion) et toutes conditions aux bords possibles entre Neumann et Dirichlet !
- diffusion_advection_all_BC : solution générale ($\mu \neq 0$ et $V \neq 0$) avec toutes conditions aux bords possibles.

Nous vous conseillons d'implémenter la matrice des flux et le vecteur b au fur est à mesure pour valider progressivement les différents cas dans l'ordre proposé au dessus ! Le fichier *TP_5_validation.pdf* accessible ici détaille les résultats à obtenir.

2. Le fichier *data.txt* qui se trouve dans le dossier *Data* correspond à une simulation avec *advection* et *diffusion* sur un maillage relativement raffiné. Lancer la simulation et comparer la solution exacte et la solution approchée sous *Paraview*. Maintenant place à l'exploitation du code !

Exploitation du code

Les différents cas `data*.txt` qui vous sont fournis peuvent être utilisés ou vous servir d'exemple pour changer les conditions de bord, les méthodes numériques, etc. Les fonctions de la condition initiale, de la vitesse, du terme source et des conditions aux bords doivent être données dans le fichier *Function.cpp* (suivre les commentaires "// TODO for real case"). Vous serez amenés à les modifier pour répondre aux questions.

Consigne

Dans tous les exercices, c'est à vous d'identifier et de justifier les bons paramètres numériques qui vous permettront de répondre aux questions.

Attention : précision et performance

Votre objectif est d'apporter une réponse aux questions ; pour y arriver, vous allez devoir faire des choix de maillages, de méthode numérique et de pas de temps, notamment. Lesquels choisir n'est pas une chose toujours évidente et requiert d'y réfléchir. Fixez-vous pour l'objectif d'obtenir le résultat numérique le plus précis possible pour un temps de calcul le plus court possible. Ces deux critères sont souvent antinomiques... mais on peut chercher une approche tendant vers l'optimalité ! Si, pour obtenir votre solution, votre calcul doit tourner pendant 3h pour un résultat simple, vous pouvez être sûr qu'il y a un choix qui n'a peut-être pas été bien fait :

- quelle est la précision que j'attends sur mes résultats ?
- peut-on réduire la taille du maillage et obtenir un résultat tout à fait satisfaisant ?
- peut-on augmenter le pas de temps tout en conservant une bonne qualité de résultat ?

Remarque : Vous n'êtes pas obligés de sauvegarder la solution à chaque pas de temps.

Si vous souhaitez ajouter aux fichiers `Paraview` le champ de vitesse, il suffit d'ajouter ces quelques lignes à la fin de la fonction *Save_Sol* juste avant *solution.close()* :

```
solution << "VECTORS vel float" << endl;
for (int i = 0 ; i < _msh->Get_triangles().size() ; ++i)
{
  solution << _fct->Velocity_x(_msh->Get_triangles_center()(i,0),
   _msh->Get_triangles_center()(i,1),n*_df->Get_dt())
   << " " << _fct->Velocity_y(_msh->Get_triangles_center()(i,0),
   _msh->Get_triangles_center()(i,1),n*_df->Get_dt()) << " 0" << endl;
}
solution << endl;
```

Cas de référence

Pour le cas de référence, nous allons travailler sur le maillage de référence `square` avec le fichier *data.toml*. En sortie au format VTK les champs suivants sont aussi accessibles :

- Nombre de CFL local : $\sigma = \frac{\delta t}{\delta x} |\mathbf{u}|$

- Nombre de Péclet local : $Pe = \frac{\delta x |\mathbf{u}|}{\mu}$

Exercice

1. Sélectionner un exemple numérique montrant que le schéma d'advection centré avec une méthode explicite est instable.
2. Étudiez *numériquement* les conditions de stabilité des différents schémas.
3. Faites la « démonstration » (pas au sens mathématique formel !) de la convergence des calculs en espace et en temps sur un ou deux cas illustratifs.

Réchauffement d'un fluide par des résistances

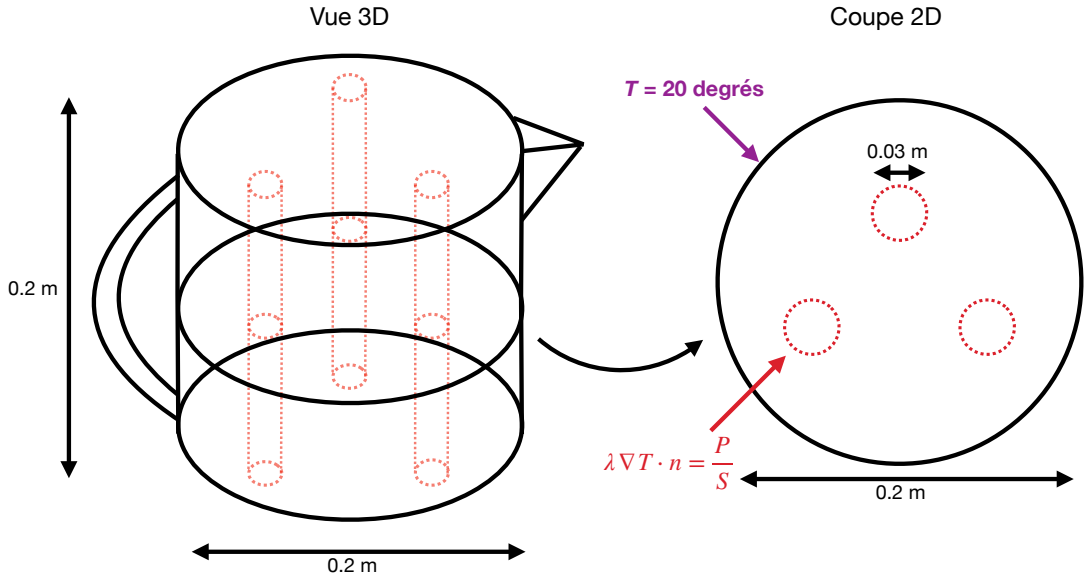

FIGURE 1 – Bouilloire

Nous proposons d'étudier le problème du réchauffement de l'eau dans une bouilloire dont les informations sont données dans la figure 1. Le liquide est chauffé par des résistances cylindriques radiants de la chaleur, en absence de convection. L'équation que nous cherchons à résoudre est donc l'équation de la chaleur :

$$\rho c_p \frac{\partial T}{\partial t} - \lambda \Delta T = 0,$$

où ρ est la masse volumique de l'eau, c_p la chaleur spécifique et λ la conductivité thermique. Nous nous placerons donc dans une coupe 2D où les résistances (des tubes) sont des « trous circulaires » dans le domaine de calcul « fluide ». Dans ce cadre, on considère alors que le domaine volumique 3D est le domaine surfacique 2D extrudé sur 0.2m de profondeur. Toutes les grandeurs volumiques sont donc en réalité exprimées « par mètre ».

Exercice

Initialement, la température de l'eau est à 15°C. La température est chauffée via les résistances :

$$\lambda \nabla T \cdot n = \lambda \partial T / \partial n = \psi = \frac{P}{S_T},$$

où P est la puissance électrique totale transformée en chaleur dans les trois résistances et où S_T est la surface totale de contact des résistances :

$$S_T = 3S = 3 * 2\pi RH,$$

avec R le rayon de chaque résistance et H la hauteur de la bouilloire.
La condition de bord de type Neumann qu'il faut appliquer aux bords des résistances est bien évidemment relative à un gradient de température : $\partial T / \partial n$, soit :

$$\partial T / \partial n = \psi / \lambda = \frac{P}{3S\lambda}$$

Plus précisément nous avons :

```
Pkettle = 2200; // W
P = Pkettle/H; // W . m-1
S = 2*M_PI*R*H/H; // m
psi = (P/3.)/S; // Flux issu des résistances : (P/3)/S
```

Ne pas oublier de diviser par λ !

1. Commentez rapidement la géométrie donnée dans `kettle2D.geo` (réalisme de la géométrie ; référence des arêtes pour les conditions aux bords ; génération de plusieurs maillages plus ou moins raffinés). Remarque : l'unité du maillage est en mètre.
2. À quel temps la température moyenne $T_{moy} = \left(= \frac{\sum_{k=1}^{N_T} |T_k| \phi_k}{\sum_{k=1}^{N_T} |T_k|} \right)$ dans la bouilloire atteint (puis dépasse) 70°C ?
3. À quel temps la température d'ébullition dans la bouilloire est-elle atteinte ? En quoi cela pose t-il problème : d'un point de vue thermodynamique et simulation numériquement ?
4. Quelles seraient les modifications à faire au niveau du code de calcul pour prendre en compte ce phénomène (modélisation ; schéma) ?
5. Quel est l'impact de la taille du maillage sur les résultats et le temps de calcul ? Que conseillez-vous ?
6. Quelle est la puissance de la bouilloire à appliquer pour obtenir une solution **stationnaire** à T_{moy} =70°C ? Pour la déduire, vous adopterez une approche expérimentale adaptée. Faire une comparaison énergétique (consommation totale) avec la solution obtenue à la question 2.
   Remarque : pour ne pas avoir à recompiler, ajoutez dans votre fichier de données :

   ```
   [physics]
   P_kettle = 2200.0
   ```

   et récupérer cette valeur dans le fichier *fonction.cpp* : *_df->Get_P_kettle*().

Remarque : Au bout de 60 secondes, la température moyenne dans la bouilloire est d'environ 20.7 degrés.

Réchauffement d'un fluide par une résistance et écoulement potentiel

Dans la continuité de l'exercice précédent, nous considérons l'équation d'advection-diffusion suivante :

$$\rho c_p \left(\frac{\partial T}{\partial t} + (\mathbf{u} \cdot \nabla) T \right) - \lambda \Delta T = 0,$$

autour, cette fois-ci, **d'une unique résistance** cylindrique. Dans le cas d'un nombre de Reynolds petit (ie. écoulement de Stokes, $Re \ll 1$, purement visqueux), le champ de vitesse du fluide contournant un cylindre est connu de manière analytique par la résolution des équations de Navier-Stokes. Il découle d'un champ potentiel et le champ de vitesse associé, en coordonnées cylindriques, est :

$$\begin{cases} V_r = & \left(1 - \frac{R^2}{r^2}\right) U \, cos\theta \\ V_\theta = & -\left(1 + \frac{R^2}{r^2}\right) U \, sin\theta \end{cases}$$

où R est le rayon du cylindre, U la vitesse d'entrée à l'infini et r la distance au centre du cylindre centré en 0. Nous considérons les mêmes propriétés des matériaux que dans l'exercice précédent. Toutefois, ici, nous considérerons une **puissance moins importante** : $P = 150W$.

Exercice

1. Étudiez d'abord le fichier `circle_with_hole.geo` et générez vos maillages.
2. Commencez par simuler le problème diffusif seul (i.e. à vitesse nulle). Vous devriez trouver une température moyenne stationnaire d'environ $T_{moy}(\infty) \simeq 100$°C.
3. Dans la suite de l'exercice, vous allez étudier l'écoulement en faisant croître la vitesse de $U = 10^{-7}$ m.s^{-1} jusqu'à $U = 10^{-3}$ m.s^{-1}. Quels sont les nombres de Reynolds associés ?
4. Quels sont les schémas les plus adaptés à vos calculs ? Vous préciserez également les pas de temps et d'espace utilisés par la suite.
5. Tracez l'évolution de la température moyenne dans le domaine étudié pour différents cas. Comment expliquez-vous son évolution ?
6. À quel moment atteignez-vous la température d'ébullition (100°C) de l'eau (quelque part dans le domaine) ? Quelle est la température moyenne associée ?
7. Quelle valeur « optimale » de la vitesse trouvez-vous permettant d'éviter l'ébullition ? Quelle est la température moyenne associée ? Quel est le Reynolds associé ?

Remarque : Au bout de 60 secondes, la température moyenne dans la bouilloire est d'environ 15.6 degrés.