MINESnancy
ARTEM

Troisième année
FICM

Département Énergie & Fluides

Module EFS9AA

Codes numériques pour la résolution de problèmes de l'ingénieur

Initiation à Freefem++ et OpenFoam

version 1.2

Mathieu Jenny

Année universitaire 2018 – 2019

Table des matières

Première partie

OpenFoam

Chapitre 1

Présentation d'OpenFoam v6

1.1 Introduction

OpenFoam est un logiciel de calcul open source qui utilise les volumes finis. Ses capacités en tant que *solver* sont assez similaires à celles de Fluent. En plus d'être à la fois gratuit et performant, il présente l'intérêt d'être modifiable et pérenne (caractère open source) pour s'adapter aux besoins de l'utilisateur.

OpenFoam est installé avec un certains nombres d'outils permettant le *preprocessing* (un mailleur, blockMesh), plusieurs *solvers* (à choisir en fonction du problème à traiter, par exemple icoFoam, nonNewtonianIcoFoam, ...) et un logiciel de *postprocessing* (paraFoam).

Cependant, l'utilisation d'un logiciel de DAO et de maillage externe est nécessaire pour les géométries complexes.

1.2 Organisation du répertoire de calcul

Le répertoire de calcul contient l'ensemble des fichiers qui définissent la configuration de l'écoulement à calculer. Ce répertoire est désigné par $CASE_ROOT. C'est à la racine de ce répertoire que doit être lancé le processeur qui effectue le calcul (par exemple nonNewtonianIcoFoam ou icoFoam).

L'organisation de ce répertoire est la suivante :

$CASE_ROOT : répertoire racine

+ 0 : répertoire contenant les champs initiaux
  - p : défini la pression initial et les conditions limites de pression
  - U : défini la vitesse initial et les conditions limites de pression
+ constant : répertoire contenant le maillage et la définition des constantes
  - transportProperties : définition de la loi de viscosité (nombre de Reynolds).
  + polyMesh
    - blockMeshDict : fichier de définition de la géométrie et des faces pour les conditions limites. Utiliser par blockMesh pour générer le maillage. Inutile en cas d'utilisation d'un mailleur externe.
    - boundary : définition des conditions limites. Ce fichier est généré, soit par blockMesh, soit par le programme d'importation de maillage (fluent3DMeshToFoam, ...).
    - ... : autres fichiers définisant le maillage générés par blockMesh ou par le programme d'importation.

+ system : répertoire contenant les fichiers de définition des schémas de résolution et la durée de la simulation.
  - controlDict : pilotage temporelle (pas de temps, durée de simulation) et définition des entrées/sorties (sauvegardes).
  - fvSchemes : définitions des schémas temporels pour la résolution de l'edp.
  - fvSolution : définitions des algorithmes d'inversion.
  - decomposeParDict : configuration de la parallélisation.

Chapitre 2

Cas d'étude

Le but de ce travail est de réaliser la simulation d'un écoulement de fluide non newtonien dans un tuyau. Nous allons effectuer un calcul de l'écoulement laminaire mais le calcul en régime turbulent peut se faire simplement en utilisant un *solver* adapté incluant un modèle de turbulence ou par simulation directe sur un maillage fin.

2.1 Géométrie et conditions limites

2.1.1 Configuration de calcul

La géométrie et les conditions limites sont représentées sur la figure 2.1 :

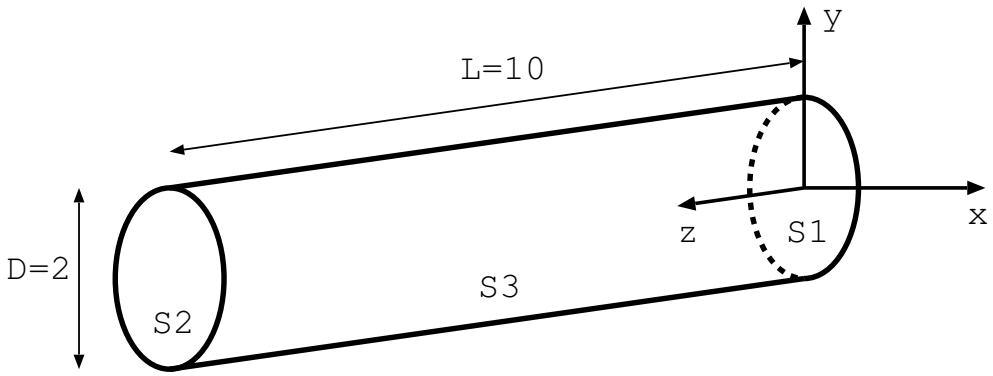

FIGURE 2.1 – Géométrie et conditions limites. Entrée en S1 et sortie en S2. Paroi en S3 (*wall*), vitesse $\vec{u} = \vec{0}$.

La longueur de référence est le rayon R de la section de la canalisation. La longueur du tuyau est fixée à 10 mais on peut envisager de simuler un tuyau plus long, si le maillage ne devient pas trop lourd. Afin de réduire la longueur d'établissement du profil, on va imposer une pression d'entrée (charge) et une pression de sortie libre (pression de sortie nulle).

2.1.2 Réalisation du maillage

La méthode privilégiée consiste à utiliser un mailleur externe. Le maillage obtenu est transféré sous OpenFoam grâce à un programme de conversion. *fluent3DMeshToFoam* convertit un maillage de Ansys Fluent (extension .msh) en maillage OpenFoam. Le maillage peut-être obtenu avec le logiciel de maillage de la suite logiciel Ansys Workbench (méthode actuelle, version $\geq$ 12.1) ou avec le mailleur Gambit (en

fin de vie). Si on utilise le Workbench de Ansys, il faut exporter le maillage. Ce dernier doit être sauvegarder en ASCII, or le fichier de sauvegarde par défaut est en binaire. Pour forcer l'utilisation de l'ASCII, il faut définir la variable d'environnement AWP_WRITE_FLUENT_MESH_ASCII en tapant dans un terminal :

```
> export AWP_WRITE_FLUENT_MESH_ASCII=1
```

Cette variable doit être imposée avant de lancer le Workbench dans le même terminal en tapant et en remplaçant xy par le numéro de la version de Ansys Fluent installée ($1xy \geq 121$) :

```
> /usr/ansys_inc/v1xy/Framework/bin/Linux64/runwb2 &
```

Dans le Workbench, créez un projet Fluent. Définissez la géométrie du tuyau (figure 2.1) et nommez les surfaces englobant le volume fluide. Editez le maillage en mettant 80 divisions sur les arrêtes de S1 et S2. Nommez les surfaces S1, S2 et S3 en in, out et slat. Une fois le maillage généré, exportez-le au format fichier d'entrée Fluent (.msh).

2.1.3 Mise en place du répertoire de calcul

Ouvrez un terminal. Testez l'installation d'OpenFoam en tapant :

```
> icoFoam -help
```

Un message d'aide doit apparaître. **Si et seulement si ce n'est pas le cas**, mettez en place les variables d'environnement pour pouvoir utiliser OpenFoam version x (n'oubliez pas de remplacer x par le bon numéro de version, par exemple 6) :

```
> source /opt/openfoam6/etc/bashrc
```

On peut ajouter cette commande à la fin du fichier .bashrc pour ne plus avoir besoin de la retaper à chaque fois que l'on ouvre un terminal. Re-testez votre installation avec la première commande.

Si tout va bien, créez un répertoire de travail et de calcul :

```
> mkdir -p $FOAM_RUN
```

On va partir de l'exemple qui utilise le solver *nonNewtonianIcoFoam*. On le copie donc dans notre répertoire de travail que l'on nomme comme on veut :

```
> cp -r $FOAM_TUTORIALS/incompressible/nonNewtonianIcoFoam/offsetCylinder
$FOAM_RUN/montp3a
```

On copie le maillage Fluent dans notre répertoire de travail :

```
> cp /chemin_vers_mon_maillage/monmaillage.msh $FOAM_RUN/montp3a/.
```

On exporte le maillage Fluent vers OpenFoam :

```
> fluent3DMeshToFoam $FOAM_RUN/montp3a/monmaillage.msh
```

Il reste maintenant à définir la loi de viscosité et le nombre de Reynolds, les conditions limites, la durée de simulation, *etc* ... Afin de configurer notre calcul, il nous faut quelques éléments de théorie et poser les équations qui gouvernent le problème.

2.2 Équations et théorie

2.2.1 Loi de viscosité

Pour la suite, nous allons nous limiter à deux cas.

— Cas newtonien : Le nombre de Reynolds est donné par $\mu = 1/Re$. Dans le fichier *transportProperties*, il faut indiquer :
`transportModel Newtonian;`

— Cas non newtonien : Nous allons utiliser un fluide non Newtonien visqueux. La viscosité μ est donnée par la loi de Bird-Carreau :

$$\mu(\dot{\gamma}) = \mu_\infty + (\mu_0 - \mu_\infty)\left(1 + (k\dot{\gamma})^2\right)^{\frac{n-1}{2}} \tag{2.1}$$

qui est une fonction connue du deuxième invariant du tenseur des taux de déformation, $\dot{\gamma}$, donné par :

$$\dot{\gamma} = \left(\frac{1}{2}\dot{\gamma}_{ij}\dot{\gamma}_{ij}\right)^{1/2} \tag{2.2}$$

On peut rappeler que le tenseur des taux de déformation est définie à partir du gradient de la vitesse $\bar{\nabla}\vec{u}$ comme suit :

$$\bar{\dot{\gamma}} = \bar{\nabla}\vec{u} + \bar{\nabla}\vec{u}^T \tag{2.3}$$

où l'exposant ^{T} indique la transposée. En coordonnées cartésiennes, on a :

$$\dot{\gamma}_{ij} = \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \tag{2.4}$$

μ_∞, μ_0, k et n sont des paramètres de la loi de viscosité.
On décide de choisir μ_0 comme paramètre pour fixer le nombre de Reynolds de paroi $Re_w = 1/\mu_w$ avec μ_w la viscosité à la paroi. Les autres paramètres de la loi de Carreau sont alors donnés en prenant : $\mu_\infty = 2.10^{-3}\mu_0$, $k = 30$ et $n = 0.5$ (c'est un choix !). Pour utiliser la loi de Bird-Carreau, il faut indiquer dans le fichier *transportProperties* :
`transportModel BirdCarreau;`

2.2.2 Équations de conservation de la masse et de la quantité de mouvement

On considère un fluide incompressible, donc la divergence de la vitesse est nulle :

$$\nabla.\vec{u} = 0 \tag{2.5}$$

La conservation de quantité de mouvement s'écrit :

$$\rho\left(\frac{\partial\vec{u}}{\partial t} + \bar{\nabla}\vec{u}.\vec{u}\right) = -\nabla p + \nabla.\left[\mu(\bar{\nabla}\vec{u} + \bar{\nabla}\vec{u}^T)\right] \tag{2.6}$$

où p est la pression du fluide et $\rho = 1$ sa masse volumique (grandeur de référence).

Les équations résolues par le solveur *nonNewtonianIcoFoam* sont lisibles dans le fichier source de l'application *nonNewtonianIcoFoam.C* qui se trouve dans le répertoire (OpenFoam version 2xy ...) : *$FOAM_INST_DIR/openfoam2xy/applications/solvers/incompressible/nonNewtonianIcoFoam*.

1. Avec un éditeur de texte, ouvrez *nonNewtonianIcoFoam.C* et repérez la définition des équations résolues. Vérifiez la correspondance avec l'équation 2.6.

2.2.3 Solution 1D de base

Dans un premier temps, on définit la solution de base de l'écoulement comme suit :

— $\vec{u} = W_b(r)\vec{e}_z$ avec r la coordonnée cylindrique suivant le rayon.

— On choisit de prendre la vitesse au centre de la conduite comme vitesse de référence, d'où $W_b(r = 0) = 1$. Bien entendu, la condition limite S3 impose $W_b(r = 1) = 0$.

— On définit le Reynolds de paroi Re_w avec la viscosité de l'écoulement de base à la paroi $\mu_b(r = 1)$, soit, avec nos notations :

$$Re_w = \frac{\rho W_b(r = 0)R}{\mu_b(r = 1)} = \frac{1}{\mu_b(r = 1)} \tag{2.7}$$

Pour calculer W_b, nous allons écrire les équations que vérifie l'écoulement de base ainsi que les conditions que ce dernier vérifie d'après sa définition ci-dessus pour un Reynolds de paroi Re_w donné.

D'après la forme de solution cherchée, les équations de Navier-Stokes 2.6 se réduisent à :

$$0 = \tilde{G}_p + \frac{1}{r}\frac{d}{dr}\left(r\tilde{\mu}_b\frac{dW_b}{dr}\right) \tag{2.8}$$

avec $dp/dz = -G_p = -\mu_0\tilde{G}_p$ le gradient de pression constant suivant z constant et $\tilde{\mu}_b = \mu_b/\mu_0$. La conditions limite est $W_b(r = 1) = 0$. Il faut déterminer le gradient de pression réduit $\tilde{G}_p$ tel que :

$$W_b(r = 0) = 1 \tag{2.9}$$

par résolution numérique de 2.8 avec la condition limite $W_b(r = 1) = 0$ (condition *wall*).

Dans notre cas, on peut écrire d'après 2.1 et le choix de loi de viscosité que :

$$\tilde{\mu}_b(r = 1) = \tilde{\mu}(\dot{\gamma}_b(r = 1)) = \tilde{\mu}_\infty + (1 - \tilde{\mu}_\infty)\left(1 + \left(k\frac{dW_b}{dr}(r = 1)\right)^2\right)^{\frac{n-1}{2}} \tag{2.10}$$

avec $\tilde{\mu}_\infty = 2 \times 10^{-3}$, $k = 30$ et $n = 0.5$. Cela permet d'obtenir d'après la définition du Reynolds de paroi :

$$\mu_0 = \frac{1}{Re_w\tilde{\mu}_b(r = 1)} \tag{2.11}$$

Dans le cas non Newtonien, ces calculs doivent être fait numériquement (avec Mathematica, Matlab ou encore OpenFoam) pour obtenir le gradient de pression et μ_0 à imposer pour avoir le Reynolds de paroi Re_w visé. Cela a été fait pour vous. Cependant, dans le cas Newtonien, le calcul peut être fait à la main.

1. Montrer que dans le cas Newtonien, l'équation 2.8 et les conditions limites d'adhérence à la paroi conduisent à :
$$W_b(r) = \frac{G_p Re}{4}(1 - r^2) \tag{2.12}$$
2. Montrer qu'on a alors $G_p = 4/Re$.

2.3 Conditions limites

Afin d'atteindre un profile de vitesse établit sur une faible longueur de tuyau, les conditions limites en S1 et S2 sont données par des conditions de pression imposée :

— Sur S1 (entrée) : La pression $p = p_e$ est calculée à partir du gradient de pression G_p. Pour un tuyau de longueur L, on a donc $p_e = LG_p$. Les conditions limites de pression sont imposées dans le fichier p par les lignes suivantes :

```
type fixedValue;
value uniform 0.0;
```

Pour la vitesse, on met dans le fichier U les lignes suivantes :

```
type pressureInletVelocity;
value uniform (0 0 0);
```

— Sur S2 (sortie) : on impose une pression de sortie nulle : $p_s = 0$. Pour la vitesse, on utilise :

```
type inletOutlet;
inletValue uniform (0 0 0);
value uniform (0 0 0);
```

— Sur S3, on impose une vitesse nulle (condition d'adhérence). Pour cela, il faut mettre les lignes :

```
type fixedValue;
value uniform (0 0 0);
```

dans U et

```
type zeroGradient;
```

dans p.

Chapitre 3

Calcul et post-traitement

Nous allons d'abord traiter le cas Newtonien pour $Re = 400$.

1. Modifier les fichiers *transportProperties*, p et U en conséquence.

3.1 Parallélisation

1. Dans le fichier *controlDict*, mettre un pas de temps de 0.1 et une durée de calcul de 10. Lancer la simulation avec la commande time pour avoir une indication du temps de calcul :
   `> time nonNewtonianIcoFoam`
2. Nous allons maintenant optimiser la largeur de bande de l'opérateur en renumérotant les points du maillage :
   `> renumberMesh -overwrite`
   Relancer le calcul avec la commande :
   `> time nonNewtonianIcoFoam`
   Y a-t-il un gain de temps ?
3. Pour paralléliser le calcul, copier le fichier *decomposeParDict* de référence :
   `> cp $FOAM_ETC/caseDicts/annotated/decomposeParDict $FOAM_RUN/montp3a/system/.`
   Dans le fichier *decomposeParDict*, le nombre de sous-domaines (*i.e.* le nombre de processus) doit être égal au nombre de CPU physiques disponibles, c'est-à-dire 4 :
   `numberOfSubdomains 4;`
   On lance la décomposition du domaine avec la commande :
   `> decomposePar`
   Il faut optimiser la numérotation de chaque sous-maillage avec la commande :
   `> mpirun -np 4 renumberMesh -overwrite -parallel`
   On peut maintenant lancer le calcul en parallèle sur 4 CPU :
   `> time mpirun -np 4 nonNewtonianIcoFoam -parallel`
   Quelle est le gain par rapport à l'utilisation d'un seul CPU ? On pourra essayer avec 2 et 8 CPU en fonction de l'avance du travail.

Maintenant que nous savons paralléliser notre calcul, il est temps de lancer le calcul sur une durée plus longue. Nous allons choisir :
`endTime 500;`
Attention à ne pas faire trop de sauvegarde en modifiant certaines lignes du fichier *controlDict*. Pour lancer le calcul, la commande `time` n'est plus nécessaire maintenant.

3.2 ParaFoam

Pour le post-traitement, nous utilisons le logiciel paraFoam. Avant cela, il faut reconstituer le résultat du calcul sur l'ensemble du maillage. Pour cela, on tape la commande :

```
> reconstructPar
```

On lance le logiciel de post-traitement avec la commande :

```
> paraFoam
```

1. Sélectionner le dernier champ calculé (Time = 500). Appliquer le changement.
2. Dans le menu *Filters / Data Analysis*, sélectionner *Plot Over Line*. Tracer le profil de vitesse en $z = 5$.
3. Sauvegarder les données (*File / Save Data*) au format csv et importer-les sous Mathematica ou Matlab. Comparer le profil au profil théorique ($W_b = 1 - r^2$).
4. Lancer le calcul pour le cas non Newtonien. On prend $p_e = 1.4824 \times 10^{-1}$ et $\mu_0 = 2.3187 \times 10^{-2}$. On utilise l'écoulement de fluide Newtonien à $t = 500$ comme condition initiale. On choisit une durée de simulation de 500 entre l'instant initial et le dernier pas de temps calculé.
5. Tracer le profil de vitesse obtenu en fin de calcul et comparer-le au profil Newtonien (en $z = 5$). Pour cela, on pourra utiliser un logiciel externe comme Matlab ou Mathematica.
6. Tracer le profil de viscosité en $z = 5$. Quel est le Reynolds de paroi Re_w ?

Deuxième partie

Freefem++

Chapitre 4

Présentation de Freefem++

4.1 Introduction

Freefem++ [1] est un logiciel de prototypage de code de calcul open source qui utilise les éléments finis. Il permet de réaliser un programme qui résout des équations aux dérivées partielles définies par l'utilisateur. Il inclue plusieurs types de solvers qui peuvent être choisi dans le programme que l'on écrit. Il est possible de réaliser le maillage directement avec des instructions sous FreeFem++ lorsque la géométrie est simple. La définition géométrique du maillage repose sur des courbes paramétriques dont on doit donner l'équation en coordonnées cartésiennes. Bien qu'à l'origine, FreeFem++ ne pouvait gérer que des maillages 2D, il est maintenant possible de générer des maillages 3D et de réaliser des calculs 3D. Lorsque la géométrie du domaine de calcul est complexe, il est possible d'importer des maillages sous FreeFem++.

Bien que FreeFem++ puisse être utilisé pour faire du calcul parallèle sur des systèmes comprenant de nombreux degré de liberté, FreeFem++ se distingue des codes près à l'emploi comme Fluent ou OpenFoam par une très grande capacité d'adaptation à un système d'équations spécifiques à résoudre, ou, en d'autres termes, à la possibilité de tester relativement rapidement différents modèles. Évidemment, cette souplesse d'adaptation à un prix, FreeFem++ ne sait rien faire par lui-même, il faut écrire un programme (même simple) et ses performances sont légèrement en deçà de celles d'un code optimisé pour un problème spécifique. Ainsi, FreeFem++ peut être vu comme une boite à outils au service du numéricien.

Pour finir, il faut noter que les auteurs de FreeFem++ diffusent librement son code source, ce qui en fait un outil pérenne et multi-plateformes.

4.2 Tutoriel

Nous allons reprendre l'exemple de la documentation de Freefem++ (téléchargeable à l'adresse `http://www.freefem.org/ff++/ftp/freefem++doc.pdf`) qui consiste à résoudre :

$$\Delta u = f \tag{4.1}$$

dans une cavité carrée de coté $a = 1$. Les points du plan sont repérés dans le système d'axes $(O, \mathbf{e}_x, \mathbf{e}_y)$. Le champ scalaire $u(x, y)$ est à calculer et $f(x, y)$ est une fonction connue que l'on prendra ici nulle, *i.e.* $f = 0$. Afin de ne pas avoir une solution triviale nulle, nous allons ajouter les conditions limites suivantes :

$$u(x, y = 0) = 1,\ u(x, y = 1) = 0,\ \partial u/\partial x(x = 0, y) = 0 \text{ et } \partial u/\partial x(x = 1, y) = 0 \tag{4.2}$$

Sur une surface Ω, la formulation faible de l'équation (4.1) conduit à

$$-\int_\Omega \left(\frac{\partial u}{\partial x}\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\frac{\partial v}{\partial y}\right) d\Omega = \int_\Omega f v\, d\Omega - \int_{\delta\Omega}\frac{\partial u}{\partial n} v\, dl \tag{4.3}$$

$\delta\Omega$ est le bord externe du domaine considéré, $\mathbf{n}$ le vecteur normal au bord du domaine et dl est un déplacement élémentaire le long du bord. Le terme de gauche de l'équation (4.3) définit l'opérateur numérique qu'il faut inverser pour résoudre le problème. Dans le terme de droite, on voit des termes connus et les conditions limites. On peut constater que les conditions de Neumann sont directement écrites la formulation (4.3). Les conditions limites de Dirichlet sont incluses par la méthode de pénalisation (voir mon cours de 2A [2] ou la documentation de Freefem++).

Le code permettant de résoudre ce problème est d'afficher les isovaleurs de u dans la cavité est donné ci-après :

```
verbosity=4;

mesh Th=square(10,10,[10*x,5*y]);

fespace Vh(Th,P1);
Vh u,v;

func f=0;

problem laplacien(u,v,solver=LU,eps=-1.0e-6) = int2d(Th)(dx(u)*dx(v) + dy(u)*dy(v))
                                             + int2d(Th) ( v*f )
                                             + on(1,u=1.0)
                                             + on(3,u=0);

real cpu=clock();
laplacien; // SOLVE THE PROBLEM
plot(u);
cout << " CPU = " << clock()-cpu << endl;
```

Copiez le programme ci-dessus dans un fichier texte, nommez-le `laplacien.edp` et exécutez-le avec la commande :

```
> FreeFem++ laplacien.edp
```

Bibliographie

[1] F. Hecht. New development in freefem++. *J. Numer. Math.*, 20(3-4) :251–265, 2012.

[2] M. Jenny. Méthodes numériques pour la mécanique-énergétique. École des Mines de Nancy - Département EPT - 2A, 2016.

Chapitre 5

Cas d'étude

Le but de ce travail est de réaliser la simulation de la convection thermique naturelle qui apparaît lorsque l'on soumet un fluide à un gradient de température vertical. La cellule convective de Rayleigh-Bénard est une configuration de référence pour les échangeurs thermiques. L'instabilité qui est à l'origine de l'apparition des rouleaux convectifs a été l'objet de très nombreuses études scientifiques. En dessous d'un certain seuil de gradient de température, la convection ne peut pas se maintenir et les échanges thermiques se font seulement par conduction. Dans la suite, nous allons étudier l'évolution d'une perturbation de cet état de base conductif pour déterminer le seuil d'apparition de la convection et l'évolution des échanges thermiques à travers le fluide contenue dans la cellule.

5.1 Géométrie, conditions limites et régime conductif

5.1.1 Cellule de Rayleigh-Bénard

La géométrie et les conditions limites sont représentées sur la figure 5.1 :

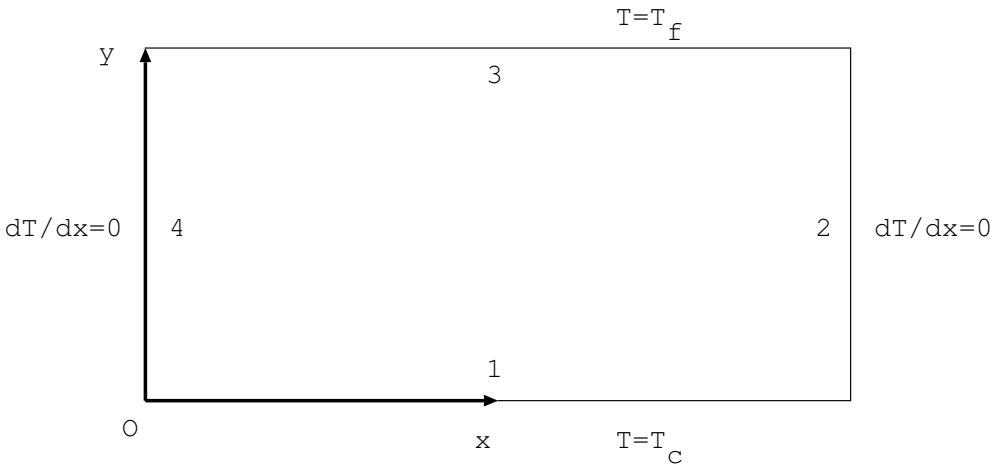

FIGURE 5.1 – Géométrie et conditions limites en température ($T_f < T_c$). Les parois sont repérées par des labels numérotés dans le sens trigonométrique. Les conditions en 2 et 4 peuvent aussi être prises comme périodiques. C'est ce que nous ferons par la suite.

La longueur de référence est la hauteur de la cellule d. La largeur de la cellule est fixée en fonction de la longueur d'onde des cellules de convection. Si on impose des conditions de périodicité entre les parois 2 et 4, il faut que la largeur de la cavité soit un multiple entier de la longueur d'onde.

5.1.2 Solution conductive

Lorsque le gradient de température entre la paroi chaude et la paroi froide n'est pas trop grand, le fluide reste statique et seule la conduction est à l'œuvre. Quand le régime stationnaire est atteint, le profile de température est linéaire :

$$T_{cond}(y) = T_c - \frac{\delta T y}{d} \tag{5.1}$$

avec $\delta T = T_c - T_f$.

Nous allons étudier l'évolution temporelle d'une perturbation de cet état de base conductif.

5.2 Équations et théorie

5.2.1 Équations aux perturbations

Pour écrire les équations aux perturbations, nous notons v la vitesse du fluide et θ l'écart relatif de température par rapport au profil conductif T_{cond}. On obtient donc

$$\theta = \frac{T - T_{cond}}{\delta T} \tag{5.2}$$

Dans le cadre de l'approximation de Boussinesq, l'adimensionnalisation des équations, en considérant la vitesse de diffusion thermique $v_{th} = \kappa/d$ comme référence, fait apparaître deux paramètres sans dimension. Le nombre de Prandtl s'écrit :

$$Pr = \eta/(\rho_0 \kappa) \tag{5.3}$$

avec η la viscosité dynamique du fluide, κ sa diffusivité thermique et ρ_0 sa masse volumique de référence. Enfin, le nombre de Rayleigh est donné par :

$$Ra = \rho_0 g \beta \delta T d^3/(\eta\kappa) \tag{5.4}$$

avec g l'accélération de pesanteur et β le coefficient de dilatation thermique.

Choisir la vitesse de diffusion thermique v_{th} comme référence conduit à de très grandes valeurs de vitesse lorsque le seuil convectif est franchit. En effet, les équations de Navier-Stokes montrent que le terme convectif est alors de l'ordre de :

$$\|\overline{\overline{\boldsymbol{\nabla}}}\mathbf{v}.\mathbf{v}\| \sim RaPr\theta \tag{5.5}$$

avec θ de l'ordre de 1 par construction. Ainsi, nous choisissons la vitesse de référence :

$$\frac{v_{ref}}{v_{th}} = \sqrt{RaPr} \tag{5.6}$$

Avec cette vitesse de référence (5.6), les équations de Navier-Stokes et de la chaleur, sans dimensions, s'écrivent :

$$\frac{\partial \mathbf{v}}{\partial t} + \overline{\overline{\boldsymbol{\nabla}}}\mathbf{v}.\mathbf{v} = -\boldsymbol{\nabla}\Pi + \sqrt{Pr/Ra}\boldsymbol{\nabla}\overline{\overline{\dot{\gamma}}} + \theta \mathbf{e}_y \tag{5.7}$$

$$\frac{\partial \theta}{\partial t} + \mathbf{v}.\boldsymbol{\nabla}\theta = \frac{1}{\sqrt{RaPr}}\Delta\theta + \mathbf{v}.\mathbf{e}_y \tag{5.8}$$

Le tenseur du taux de déformation sans dimension est donné par :

$$\overline{\overline{\dot{\gamma}}} = \overline{\overline{\nabla}}\mathbf{v} + \overline{\overline{\nabla}}\mathbf{v}^T \tag{5.9}$$

et Π est la perturbation de pression sans dimension.

Au système d'équations (5.7) et (5.8), il faut ajouter la conservation de la masse pour un fluide incompressible :

$$\nabla\mathbf{v} = 0 \tag{5.10}$$

Pour la perturbation, on considérera des conditions limites d'adhérence pour la vitesse et rigide pour la température, *i. e.* en $y = 0$ et $y = d$, $\mathbf{v} = 0$ et $\theta = 0$. La périodicité dans la direction x s'applique bien entendu à la perturbation.

5.2.2 Nombre de Nusselt

Le nombre de Nusselt Nu est le rapport entre le transfert thermique total (conduction + convection) et le transfert thermique conductif. On montre que l'on a :

$$Nu - 1 = -\int_3 \frac{\partial\theta}{\partial y} dx \tag{5.11}$$

Il est clair que lorsque la convection se développe $Nu > 1$ et que lorsque seule la conduction opère $Nu = 1$. Ainsi, en-dessous du seuil d'instabilité, la valeur du nombre de Nusselt revient vers 1 suite à une perturbation alors qu'au-delà du seuil, la valeur du nombre de Nusselt s'écarte de 1 pour atteindre une valeur supérieure.

On peut noter que plus le nombre de Nusselt est élevé, plus le transfert thermique est important. La convection accroît les échanges thermiques.

5.3 Réalisation du programme

5.3.1 Formulation faible

Afin de pouvoir résoudre notre problème avec des éléments finis, il faut réécrire les équations (5.7), (5.8) et (5.10) en formulation faible. Pour cela, nous les projetons sur des fonctions tests en les intégrant sur un domaine Ω de bord $\delta\Omega$. Les champs $\mathbf{w}$, φ et q sont, respectivement, les fonctions tests de vitesse, de température et de pression.

$$\begin{aligned}
\int_\Omega \left[\frac{\partial\mathbf{v}}{\partial t} + \overline{\overline{\nabla}}\mathbf{v}.\mathbf{v}\right].\mathbf{w}\ d\Omega &= \int_\Omega \left[\left(\Pi\overline{\overline{I}} - \sqrt{Pr/Ra}\,\overline{\overline{\dot{\gamma}}}\right) : \overline{\overline{\nabla}}\mathbf{w} + \theta\mathbf{e}_y.\mathbf{w}\right] d\Omega \\
&\quad + \int_{\delta\Omega} \mathbf{w}.\left(-\Pi\overline{\overline{I}} + \overline{\overline{\tau}}\right).\mathbf{n}\ dl, \qquad (5.12) \\
\int_\Omega \left[\frac{\partial\theta}{\partial t} + \mathbf{v}.\nabla\theta\right]\varphi\ d\Omega &= \int_\Omega \left[-\frac{1}{\sqrt{RaPr}}\nabla\theta.\nabla\varphi + (\mathbf{v}.\mathbf{e}_y)\,\varphi\right] d\Omega \\
&\quad + \int_{\delta\Omega} \varphi\nabla\theta.\mathbf{n}\ dl, \qquad (5.13) \\
\int_\Omega q\,\mathrm{div}(\mathbf{v})\ d\Omega &= 0, \qquad (5.14)
\end{aligned}$$

On peut remarquer que la formulation ci-dessus fait explicitement apparaître la contrainte dans l'équation de Navier-Stokes (5.12). Vous pouvez vous reporter au tome 0 du cours de MMCSF [1] pour retrouver les expressions (5.12 – 5.14) à partir des équations (5.7), (5.8) et (5.10).

5.3.2 Schéma temporel

Nous allons utiliser un schéma implicite d'ordre 1 en temps. L'indice $n+1$ désigne le pas de temps courant et l'indice n, le pas de temps précédent.

$$\begin{aligned}
&\int_\Omega \left[\frac{\mathbf{v}_{n+1}}{\delta t}.\mathbf{w} + \left(-\Pi_{n+1}\overline{\overline{\boldsymbol{I}}} + \sqrt{Pr/Ra}\,\overline{\overline{\dot{\gamma}}}_{n+1}\right) : \overline{\overline{\boldsymbol{\nabla}}}\mathbf{w}\right] d\Omega = \\
&\int_\Omega \left[\left(\frac{\mathbf{v}_n}{\delta t} - \left(\overline{\overline{\boldsymbol{\nabla}}}\mathbf{v}.\mathbf{v}\right)_n\right) + \theta_n \mathbf{e}_y\right].\mathbf{w}\; d\Omega + \int_{\delta\Omega} \mathbf{w}.\left(-\Pi\overline{\overline{\boldsymbol{I}}} + \overline{\overline{\boldsymbol{\tau}}}\right)_{n+1}.\mathbf{n}\; dl,
\end{aligned} \tag{5.15}$$

$$\begin{aligned}
\int_\Omega \left[\frac{\theta_{n+1}}{\delta t}\varphi + \frac{1}{\sqrt{RaPr}}\boldsymbol{\nabla}\theta_{n+1}.\boldsymbol{\nabla}\varphi\right] d\Omega \;=\; &\int_\Omega \left[\frac{\theta_n}{\delta t} - (\mathbf{v}.\boldsymbol{\nabla}\theta)_n + \mathbf{v}_n.\mathbf{e}_y\right]\varphi\; d\Omega \\
&+ \int_{\delta\Omega} \varphi\boldsymbol{\nabla}\theta_{n+1}.\mathbf{n}\; dl,
\end{aligned} \tag{5.16}$$

$$\int_\Omega q\,\mathrm{div}(\mathbf{v}_{n+1})\; d\Omega = 0\;. \tag{5.17}$$

On peut réécrire le système d'équations (5.15 – 5.17) sous forme matricielle :

$$AX + GP = B, \tag{5.18}$$

$$DX = 0\;. \tag{5.19}$$

avec X l'ensemble des inconnues de vitesse et de température et P l'ensemble des inconnues de pression.

Identifiez les opérateurs représentés par les matrices A, G et D. Que représente B ?

Donnez la vitesse et la température en fonction de la pression. À partir de la relation (5.19), donnez X en éliminant la pression.

Proposez un algorithme de résolution du système (5.18 – 5.19).

5.3.3 Programme

Le programme est un fichier texte que l'on appellera `thermoconv.edp`. Les symboles `//` permettent de délimiter des commentaires dans le programme. Ils ne sont donc pas exécuter et ne servent qu'à faciliter la lecture et la compréhension du programme. Cela dit, mettre des commentaires dans un programme est indispensable pour s'assurer qu'il sera compréhensible par d'autres personnes que son auteur.

```
// Version 2D sequentielle periodique !

include "getARGV.idp" // Permet de passer des parametres par la ligne de commande

real ttgv=1e30;
real ttpv=1e-6;

// Parametre numerique

real dt=0.01; // pas de temps
```

```
int np=30; // nombre de mailles selon x
int mp=30; // nombre de mailles selon y
int N=getARGV("-N",1000); // nombre d'iterations
int frequenceSauvegarde=50; // Frequence de la sauvegarde des champs

// Parametres physiques

real kx=3.116; // nombre d'onde avec adherence
real x0 = 0.0; // origine x
real x1 = 1.0*2*pi/kx; // largeur selon x
real y0=0; // origine selon y
real y1=1; // hauteur selon y

real Ra=getARGV("-Ra",1800); // nombre de Rayleigh
real Pr=1.0; // nombre de Prandt

// Vitesse et de temps de reference
real Vref=sqrt(Ra*Pr); // Mettre 1 pour revenir a la vitesse de diffusion thermique
real tref=1./Vref;

//Variables

real alpha=1./dt; // inverse du pas de temps
real nn=1./mp; // pas d'espace

real Nu; // nombre de Nusselt
real temps; // variable temporel
```

Ici, il faut générer le maillage `Th` et on pourra l'afficher avec la commande `plot`.

```
fespace VVh(Th,[P2,P2,P2],periodic=[[2,y],[4,y]]);
VVh [v1,v2,t],[f1,f2,f3];
// [v1,v2,t] vitesses et temperature
// [f1,f2,f3] fonction test

fespace Vh(Th,P2,periodic=[[2,y],[4,y]]);
Vh v1o,v2o,to;
// [v1o,v2o,to] vitesses et temperatures a l'instant precedent

fespace VMh(Th,[P1],periodic=[[2,y],[4,y]]);

fespace Mh(Th,P1,periodic=[[2,y],[4,y]]);
Mh p,pw;
```

```
string lecture="Ra"+Ra;
string ecriture="Ra"+Ra;

int ifrst=getARGV("-load",0);

if(ifrst)
{
    Th=readmesh(lecture+"ThRef.msh"); // maillage 2d
    {
    ifstream file(lecture+"vt.txt"); // champ de vitesse et temperature
    file>>v1[];
    }
    {
    ifstream file(lecture+"p.txt"); // champ de pression
    file>>p[];
    }
    {
    ifstream file(lecture+"temps.txt"); // variable temporelle
    file>>temps;
    }
    {
    ifstream file(lecture+"dt.txt"); // pas de temps
    file>>dt;
    }
    ttpv=-abs(ttpv);
    alpha=1./dt;
}
else
{ //on part de zero
    temps=0.0;
    p=0.0;
    [v1,v2,t]=[-0.1*32*y/kx*(y-y1)*(2*y-y1)*sin(kx*x),0.1*16*y^2*(y-y1)^2*cos(kx*x),0];
    plot([v1,v2],wait=1,cmm="Vitesse initiale");
}
```

Montrez que le champs de vitesse initial respecte les conditions limites et qu'il est à divergence nulle.

```
//Sauvegarde du maillage
savemesh(Th,ecriture+"ThRef.msh");

// Définition des macros

macro div(v1,v2) (dx(v1)+dy(v2)) //
```

```
varf vDiv([v1,v2,t],[q],qforder=4)=int2d(Th)(div(v1,v2)*q);
matrix MDiv=vDiv(VVh,VMh);

macro grad(u) [dx(u),dy(u)] //

// Probleme formulation faible

problem vRBnn([v1,v2,t],[f1,f2,f3],solver=CG,eps=ttpv)=
             int2d(Th)(alpha*(v1*f1 + v2*f2 + t*f3)
             + (Pr*tref)*(2*dx(v1)*dx(f1)+2*dy(v2)*dy(f2)
                          +(dy(v1)+dx(v2))*(dy(f1)+dx(f2)))
             + tref*(dx(t)*dx(f3)+dy(t)*dy(f3)))
             - int2d(Th)(f1*(alpha*v1o-[v1o,v2o]'*grad(v1o))
                         +f2*(alpha*v2o-[v1o,v2o]'*grad(v2o))
                         +f3*(alpha*to-[v1o,v2o]'*grad(to))
             + div(f1,f2)*pw
             + (Ra*Pr*tref^2)*to*f2
             + v2o*f3)
             + on(1,3,v1=0,v2=0,t=0);
```

Montrez que l'on peut rendre implicite les termes de couplage entre la vitesse et la température sans modifier la symétrie de l'opérateur linéaire définit par `vRBnn`.

Si on conserve le couplage implicite entre vitesse et température, que se passerait-il si l'on prenait une autre vitesse de référence ? Quelle serait les conséquences pour le choix du solveur ?

```
// Fonction divup
func real[int] divup(real[int] & pp)
{
   pw[]=pp;
   vRBnn;
   real[int] divu=MDiv*v1[];
   return divu;
}
```

Que calcule la fonction divup ?

```
//Preconditionneur
varf vA(p,q) = int2d(Th)((grad(p)'*grad(q)));
varf vM(p,q) = int2d(Th,qft=qf2pT)(p*q);
matrix pAM=vM(Mh,Mh,solver=UMFPACK);
matrix pAA=vA(Mh,Mh,solver=UMFPACK);
```

```
real vism=tref*Pr;

func real[int] Precon(real[int] & p)
 {
   real[int] pa= pAA^-1*p;
   real[int] pm= pAM^-1*p;
   real[int] pp= alpha*pa+vism*pm;
   return pp;
 }

//Boucle temporelle

int res;
real dtc;
v1o=v1;
v2o=v2;
to=t;

int i=1;

while (i<=N)
{
    i++;
    temps+=dt;
    cout << "temps=" << temps << " dt=" << dt << endl;
    res=LinearCG(divup,p[],precon=Precon,veps=ttpv,nbiter=50,verbosity=10);
    assert(res==1);
    ttpv=-abs(ttpv);
    divup(p[]);
    v1o=v1;
    v2o=v2;
    to=t;

    cout << " v1 max " << v1[].linfty
         << " v2 max " << v2[].linfty
         << " t max " << t[].linfty << endl;

     // Verification de dt
     pw=hTriangle/sqrt(v1^2+v2^2);
     dtc=pw[].min;
     if(dt>dtc)
     {
        dt=0.5*dtc;
        alpha=1./dt;
        cout << "dt change to " << dt << " CFL limit " << dtc << endl;
```

```
      }
      if(dt<0.1*dtc)
      {
         dt=min(0.5*dtc,0.1);
         alpha=1./dt;
         cout << "dt change to " << dt << " CFL limit " << dtc << endl;}
      // Calcul des variables temporelles

      Nu=int1d(Th,3,qfe=qf2pE)(-dy(t));

      //Sauvegarde des variables temporelles
      {ofstream ff(ecriture+"Nu_t.txt",append); ff << temps << " " << Nu << endl;};
      plot([v1,v2],value=true,wait=0,cmm="[v1,v2] au temps = "+temps);

    // sauvegarde du champ complet tout les n pas

    if(i%frequenceSauvegarde==0)
    {
        {ofstream f(ecriture+"vt.txt");
         ofstream g(ecriture+"p.txt");
         ofstream h(ecriture+"temps.txt");
         ofstream l(ecriture+"dt.txt");
         f << v1[]; // sauvegarde de [v1,v2,t]
         g << p[];   // sauvegarde de la pression
         h << temps;
         l << dt;}
     }
} // fin de la boucle temporelle

//Sauvegarde du calcul final
plot([v1,v2],value=true,wait=0,cmm="[v1,v2] au temps = "+temps,ps=ecriture+"vitesse.ps");
{
ofstream f(ecriture+"vt.txt");
ofstream g(ecriture+"p.txt");
ofstream h(ecriture+"temps.txt");
ofstream l(ecriture+"dt.txt");
f << v1[]; // sauvegarde de [v1,v2,v3,t]
g << p[];   // sauvegarde de la pression
h << temps;
l << dt;
}
```

Une fois votre programme écrit et sauvegardé, vous pouvez le lancer pour le tester avec la commande :

```
> FreeFem++ thermoconv.edp -Ra 1800 -N 10
```

Bibliographie

[1] E. Plaut. Mécanique des milieux continus solides et fluides - Tome 0 - Le calcul tensoriel : outil mathématique pour la physique des milieux continus. École des Mines de Nancy - 1A, 2016.

Chapitre 6

Calcul et post-traitement

Le post-traitement des calculs sous Freefem++ peut se faire avec n'importe quel logiciel de traitement de donnée comme Matlab, Scilab, Octave ou encore Mathematica (non, je ne mettrai pas Excel dans la liste...).

Dans notre cas, on sauvegarde le nombre de Nusselt à chaque pas de temps. Cela nous permet de déterminer le seuil d'apparition de la convection, *i. e.* la valeur du nombre de Rayleigh à partir de laquelle le nombre de Nusselt est plus grand que 1.

En lançant deux calculs, l'un à $Ra = 1650$ et l'autre à $Ra = 1750$, donner une estimation de la valeur critique du Rayleigh Ra_c. On prendra un nombre de Prandtl $Pr = 1$.

On peut aussi tracer le nombre de Nusselt Nu en fonction du nombre de Rayleigh Ra, ce qui permet de déterminer le coefficient de convection thermique h dans chaque cas.