Écoulements aérothermiques :Benchmarks et Applications avec Feel++ et FreeFem++

O Mokhtari, A. Popescu, U. Tan, J. Traoré, X. Yau, Jean-Baptiste Wahl

Écoulements aérothermiques : Benchmarks et Applications avec Feel++ et FreeFem++

PROJET COLLABORATIF

Université Pierre et Marie Curie
Master 2 : Ingénierie Mathématique
O. MOKHTARI, A. POPESCU, U. TAN, J. TRAORÉ, X. YAU

Université de Strasbourg
Master 2 : Calcul Scientifique et Mathématiques de l'Information
J.B. WAHL

ENCADRANTS :
UPMC : E. GODLEWSKI, F. HECHT, M. POSTEL,
UDS : C. PRUD'HOMME, Y. HOARAU
Airbus Group : M. FOUQUEMBERGH

Contents

Introduction

Origine du projet Les travaux présentés dans ce document font suite à un précédent travail commencé en juin 2013 lors d'un stage de Master 1 effectué par J.B. Wahl à l'Université de Strasbourg. Le but de ce stage était de fournir un code éléments finis de simulation d'écoulements aérothermiques.

Ce projet a été proposé par l'entreprise Airbus Group qui aimerait appliquer des méthodes de réduction d'ordre sur de tels modèles afin de les utiliser davantage dans leur chaine de design et de production.

A l'issue de ce stage, un premier code a été écrit en langage C++ utilisant la bibliothèque de calcul scientifique `Feel++`. Si les résultats semblaient intuitivement cohérents, il était nécessaire de procéder à une validation minutieuse du code.

C'est ainsi que ce travail de collaboration avec des étudiants du Master 2 Ingénierie Mathématique de l'Université Pierre et Marie Curie (UPMC) a été initié.

L'idée de cette collaboration est de valider le modèle et le code utilisé en comparant les résultats obtenus à l'aide de `Feel++` avec ceux de simulations équivalentes réalisées avec le logiciel `FreeFem++`, par des étudiants de l'UPMC (Paris).

Déroulement de la collaboration Il s'agit d'une collaboration de type benchmark où chacune des parties travaille de son côté à la résolution d'un même problème. Il ne s'agit donc pas d'une collaboration à proprement parler où les deux parties travailleraient sur un même code et participeraient ensemble à son développement.

Néanmoins, on peut bel et bien parler de collaboration dans le sens où de nombreux échanges ont eu lieu, à propos de la compréhension du problème et du choix des paramètres, des méthodes utilisées ou encore de leur implémentation. Cette collaboration consiste également à comparer les résultats obtenus, à observer les différences, à comprendre ensemble leurs origines et à proposer certaines pistes.

Afin de faciliter les échanges, différents outils ont été utilisés:

- la mise en place d'une liste de diffusion mettant en jeu l'ensemble des parties permettant d'échanger sur les différents résultats obtenus,
- le service GitHub (`https://github.com/`) pour l'échange de données et le partage des codes sources,
- plusieurs séances de visioconférences, constituant des échéances importantes et permettant de déterminer les différentes marches à suivre et les prochaines étapes du projet.

Plan du document Dans le présent document, nous commençons par présenter le modèle considéré, en rappelant les différentes approximations physiques envisagées, qui nous permettent d'écrire les équations de Navier-Stokes pour un fluide incompressible, dans l'approximation de Boussinesq.

Nous détaillons ensuite les différentes méthodes numériques utilisées, leur implémentation sous `Feel++` d'abord, puis sous `FreeFem++`, les choix d'initialisation ainsi que le traitement du problème de l'unicité de la solution.

La suite de ce document présente les résultats obtenus, sur les deux cas-tests étudiés. Ainsi, la troisième partie concerne le benchmark bien connu de la cavité chauffée pour lequel la littérature fournit de nombreux résultats auxquels nous avons pu comparer les nôtres. Enfin, la dernière partie traite du cas-test plus compliqué qu'est celui d'une salle machine, sur lequel nous comparons nos résultats dans le cadre d'une simulation cette fois-ci non stationnaire.

CHAPTER 1

Modélisation d'un problème d'aérothermie

1.1 Les équations de Navier-Stokes dans l'approximation de Boussinesq

On rappelera dans cette section les formes différentielles et intégrales des lois de conservation aboutissant aux équations de Navier-Stokes, qui régissent la dynamique associée à notre problème. Par la suite, on détaillera l'approximation de Boussinesq, qui constitue un modèle particulièrement adapté à la description des problèmes aérothermiques, en exhibant par une approche simple, le couplage dynamique/thermique du fluide.

1.1.1 Dérivée matérielle temporelle

Afin d'établir les différentes lois de conservation, nous aurons besoin de considérer la variation de l'intégrale d'une fonction (scalaire ou vectorielle) sur un volume occupé par le fluide. Nous utiliserons pour cela, la notion de volume matériel qui réfère à un volume de fluide dont le nombre de particules (prises dans l'approximation du continuum) ne varie pas au cours du temps. Considérons à présent, un de ces volumes matériel $V(t)$, ayant pour frontière $\partial V(t)$, $\partial V(t)$ étant une surface régulière fermée dont les points se déplacent au temps t à la vitesse

$$\mathbf{u} = \mathbf{u}(\mathbf{x}, t) \quad \forall \mathbf{x} \in \partial V(t).$$

Alors, la dérivée temporelle de l'intégrale d'une fonction vectorielle $\mathbf{f} = \mathbf{f}(\mathbf{x}, t)$ sur un de ces volumes matériels $V(t)$ est donnée par le théorème de transport de Reynolds :

$$\frac{d}{dt}\int_{V(t)} \mathbf{f}\, dV = \int_{V(t)} \frac{\partial \mathbf{f}}{\partial t}\, dV + \int_{\partial V(t)} (\mathbf{u}\cdot\mathbf{n})\mathbf{f}\, dS.$$

Note : ce théorème est valable pour toute fonction $\mathbf{f}$ suffisamment régulière.

1.1.2 Equation de continuité

La mécanique newtonienne impose la conservation de la masse M (en kg) contenue dans un volume matériel $V(t)$. Ainsi,

$$\frac{dM}{dt} = \frac{d}{dt}\int_{V(t)} \rho\, dV,$$

où ρ représente la densité du fluide en $kg.m^{-3}$.

En appliquant à cette équation le théorème de transport de Reynolds et le théorème de la divergence, on obtient :

$$0 = \frac{dM}{dt} = \int_{V(t)} \frac{\partial \rho}{\partial t}\, dV + \int_{\partial V(t)} \rho \mathbf{u} \cdot \mathbf{n}\, dS = \int_{V(t)} \left(\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) \right) dV.$$

Dès lors, puisque la relation précédente est valide pour tout choix de $V(t)$, nous avons

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0,$$

en tout point du fluide. Cette équation correspond à la forme locale conservative de l'équation de continuité. On rappelle que l'incompressibilité est une approximation valable lorsque la vitesse d'écoulement du fluide est très nettement inférieure à la vitesse du son dans le milieu (faible nombre de Mach) : on considère alors, avec cette approximation, que la densité ρ est constante en espace et en temps.

Ainsi, pour ce type de fluide, on a

$$\frac{\partial \rho}{\partial t} = 0,$$

d'où nécessairement,

$$\nabla \cdot \mathbf{u} = 0.$$

Pour un fluide incompressible, la vitesse $\mathbf{u}$ correspond donc à un champ rotationnel.

1.1.3 Equation bilan de la quantité de mouvement

L'équation de la quantité de mouvement met en relation la variation de la quantité de mouvement d'une portion de fluide avec la somme des forces s'exerçant sur elle. La quantité de mouvement d'un volume matériel est définie par:

$$\int_{V(t)} \rho \mathbf{u}\, dV$$

En utilisant le théorème de transport de Reynolds et le théorème de la divergence, – l'opération $\otimes$ désignant le produit tensoriel – on obtient :

$$\frac{d}{dt} \int_{V(t)} \rho \mathbf{u}\, dV = \int_{V(t)} \frac{\partial (\rho \mathbf{u})}{\partial t}\, dV + \int_{\partial V(t)} (\rho \mathbf{u} \otimes \mathbf{u}) \cdot \mathbf{n}\, dS = \int_{V(t)} \left(\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) \right) dV,$$

En appliquant l'équation de continuité, on obtient:

$$\frac{d}{dt} \int_{V(t)} \rho \mathbf{u}\, dV = \int_{V(t)} \rho \frac{d\mathbf{u}}{dt}\, dV$$

On remarquera ici que le membre de gauche équivaut à la somme du produit de la masse par l'accélération.

Considérons à présent, les forces appliquées sur notre portion de fluide. On distinguera les forces – issues d'un champ de force – par unité de volume, des forces de contact par unité de surface.

Note : ici, on ne tiendra pas compte des forces intérieures (d'origine moléculaire) car on supposera que, dans un volume matériel $V(t)$, elles s'opposent deux à deux.

On notera par f la distribution de forces par unité de volume. Ainsi, la force volumique totale appliquée à un volume matériel $V(t)$ est définie par

$$\int_{V(t)} = \rho \mathbf{f}\, dV.$$

L'exemple le plus courant est le cas où la distribution de forces f est égale à

$$\mathbf{f} = -g\,\mathbf{e_3}$$

où g est la gravité et $\mathbf{e_3}$ le vecteur unitaire pointant vers le haut.

De la même manière, on peut écrire la force surfacique totale appliquée à un élément de surface dS comme,

$$\int_{\partial V(t)} \sigma \cdot \mathbf{n}\, dS,$$

où σ désigne le tenseur des contraintes de Cauchy. D'après le théorème de la divergence, on a:

$$\int_{V(t)} \nabla \cdot \sigma\, dV.$$

Ainsi, en utilisant la seconde loi de Newton, on obtient pour tout volume matériel $V(t)$,

$$\int_{V(t)} \rho \frac{d\mathbf{u}}{dt}\, dV = \int_{V(t)} = \rho\mathbf{f}\, dV + \int_{V(t)} \nabla \cdot \sigma\, dV,$$

ce qui impose en tout point du fluide

$$\rho \frac{d\mathbf{u}}{dt} = \rho\mathbf{f} + \nabla \cdot \sigma.$$

En appliquant le théorème de Reynolds sur la vitesse, on a :

$$\frac{du}{dt} = \frac{\partial u}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u},$$

ce qui nous donne

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u} \cdot \nabla)\mathbf{u} = \rho\mathbf{f} + \nabla \cdot \sigma.$$

Enfin, en appliquant cette équation au cas des fluides newtoniens à propriétés constantes, on a, comme tenseur des contraintes de Cauchy (appelé aussi tenseur de contraintes visqueuses),

$$\sigma = -p\mathbb{I} + \mu\left((\nabla\mathbf{u}) + (\nabla\mathbf{u})^T\right),$$

où p correspond à la pression, μ (constante) à la viscocité dynamique et $\mathbb{I}$ à la matrice identité.

On obtient alors,

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u} \cdot \nabla)\mathbf{u} = \rho\mathbf{f} + \nabla \cdot \sigma = \rho\mathbf{f} - \nabla p + \mu\mathbf{\Delta} u.$$

Cette dernière expression correspond à la forme locale conservative de la quantité de mouvement pour un fluide newtonien. Cette équation est non-linéaire du fait du terme de convection (ou advection) $\rho(\mathbf{u} \cdot \nabla)\mathbf{u}$. C'est un aspect non négligeable qui rend sa résolution et son étude particulièrement complexe.

1.1.4 Approximation de Boussinesq

L'approximation de Boussinesq repose sur le fait que, dans certains fluides où la température $T(\mathbf{x}, t)$ varie faiblement, la densité y varie elle aussi faiblement et de manière proportionnelle. Cette variation de la densité en fonction de la température engendre l'existence d'un mouvement de convection (naturelle) porté par les forces de flottaison. Dans l'approximation de Boussinesq, la variation de la densité en fonction de la température est négligée partout,

excepté pour le terme de flottaison. Ainsi, comme cette variation n'affecte pas l'équation de continuité, on a :

$$\nabla \cdot u = 0$$

Notons à présent par ρ_0 la masse volumique de référence à la température de référence T_0. Pour de faibles différences de température, on peut alors écrire,

$$\rho = \rho_0[1 - \alpha(T(\mathbf{x}, t) - T_0)]$$

où α désigne le coefficient d'expansion volumique du fluide considéré et $T(\mathbf{x}, t)$ désigne la température. Dès lors, pour le terme de flottaison, on peut écrire,

$$\rho\mathbf{f} = -\rho g\mathbf{e_3} = -g\rho_0[1 - \alpha(T - T_0)]\mathbf{e_3}.$$

Cela nous donne alors comme équation-bilan de la quantité de mouvement:

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u} \cdot \nabla)\mathbf{u} = -g\rho_0[1 - \alpha(T - T_0)]\mathbf{e_3} - \nabla p + \mu\boldsymbol{\Delta} u.$$

De la même manière, on peut démontrer que dans le cadre de l'approximation de Boussinesq et à partir de l'équation de conservation de l'énergie (Navier-Stokes), on obtient :

$$\frac{\partial T}{\partial t} + (\mathbf{u} \cdot \nabla)T = \kappa\Delta T.$$

Cette approximation permet donc d'expliquer les phénomènes de convection naturelle initiés par les gradients de température et d'obtenir le couplage entre la température et le champ vitesse/pression.

1.2 Formulation forte du problème

On se place dans l'approximation de Boussinesq pour l'écoulement bidimensionnel d'un fluide newtonien, incompressible et instationnaire.

Note : le cas stationnaire se décrit par des dérivées temporelles égales à 0.

Soit Ω un ouvert borné dans $\mathbb{R}^2$ de frontière lipschitzienne $\partial\Omega = \Gamma$. On notera par $\mathbf{x}$ la coordonnée cartésienne courante (en m), et par $t \in [0, T_f]$ le temps (en s). De la même manière, $\mathbf{u}, p$ et T désignent respectivement les champs de vitesse (vectoriels, en $m.s^{-1}$), de pression (en Pa) et de température (en K). Les équations gouvernant notre problème sont alors:

$$\begin{cases} \dfrac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla\mathbf{u} + \dfrac{1}{\rho}\nabla p - \nu\Delta\mathbf{u} = g\beta(T - T_0)\mathbf{e_2} & \text{dans } \Omega \times [0, T_f], \\ \nabla \cdot \mathbf{u} = 0 & \text{dans } \Omega \times [0, T_f], \\ \dfrac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T - \kappa\Delta T = 0 & \text{dans } \Omega \times [0, T_f]. \end{cases} \tag{1.1}$$

ρ désignant la masse volumique ($kg.m^{-3}$), κ le coefficient de diffusivité thermique ($m^2.s^{-1}$), $\nu = \frac{\mu}{\rho}$ la viscosité cinématique ($m^2.s^{-1}$) et β le coefficient d'expansion volumique (K^{-1}).

Le problème précédent admet une infinité de solutions. Afin d'en restreindre le nombre, nous allons ajouter les conditions aux limites associées à notre système.

1.3 Conditions limites

1.3.1 Conditions initiales

Commençons tout d'abord avec la condition initiale suivante,

$$\begin{cases} \mathbf{u}(\mathbf{x},0) = \mathbf{u_0} \quad \forall\, \mathbf{x} \in \Omega, \\ \nabla \cdot \mathbf{u_0} = 0, \\ T(\mathbf{x},0) = T_{ini} \quad \forall\, \mathbf{x} \in \Omega \end{cases} \tag{1.2}$$

Celle-ci traduit simplement l'existence d'un champ de vitesse (dans la pratique nulle) animant notre fluide à $t = 0$. De la même manière, on considère qu'à $t = 0$, il règne une température initiale T_{ini} constante en tout point du domaine.

1.3.2 Conditions aux bords

Les tableaux ci-dessous reprennent les différents types de conditions aux bords employés dans cette étude.

Partie fluide :

Type	Expression	Description
Dirichlet homogène	$\mathbf{u} = 0$	"No-slip condition" : les particules de fluide adhèrent aux parois. La vitesse des particules est nulle sur les bords considérés.
Dirichlet	$\mathbf{u} = F_{D_U}$	Impose un profil de vitesse défini sur le bord considéré. Utilisée sur les entrées du fluide
Neumann	$(\frac{1}{\rho} p\mathrm{I} - \nu\nabla\mathbf{u}) \cdot \mathbf{n} = 0$	"Do-Nothing condition". Utilisée sur les sorties du fluide.

Partie thermique :

Type	Expression	Description
Dirichlet	$T = F_{D_T}$	Impose un profil de température sur le bord considéré.
Neumann	$\kappa\frac{\partial T}{\partial \mathbf{n}} = F_{N_T}$	Impose un flux constant sur le bord considéré.
Robin	$\kappa\frac{\partial T}{\partial \mathbf{n}} = \frac{\lambda}{\rho C_p}(T_R - T)$	Impose un flux proportionel au différentiel de température entre le bord considéré et le milieu. Ici, C_p désigne la chaleur spécifique en K^{-1} et λ la conductivité thermique du bord considéré en $W.m^{-1}.K^{-1}$.

CHAPTER 2

Méthodes numériques

2.1 Feel++

Feel++ (http://www.feelpp.org/) est une librairie C++ pour la résolution d'équations aux dérivées partielles, utilisant les méthodes de Galerkin, comme la méthode des éléments finis ou les méthodes spectrales [8]. Dans cette partie, nous donnerons un aperçu de l'utilisation de cette bibliothèque, et de quelques exemples d'implémentation issus de notre code.

2.1.1 Formes variationnelles

Soit Ω le domaine de résolution et soient $\mathbf{u}$, T , p les champs de vitesse, de température et de pression du fluide considéré. Il s'agit ici de résoudre le système suivant:

$$\begin{cases} \dfrac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} + \dfrac{1}{\rho}\nabla p - \nu\Delta\mathbf{u} = g\beta(T-T_0)\mathbf{e}_2 & \text{dans } \Omega \\ \nabla\cdot\mathbf{u} = 0 & \text{dans } \Omega \\ \dfrac{\partial T}{\partial t} + \mathbf{u}\cdot\nabla T - k\Delta T = 0 \end{cases} \tag{2.1}$$

Nous définissons pour cela un ensemble de fonctions "tests" $\{(\mathbf{v}, q, s) \in [H^1(\Omega)]^n \times L^2(\Omega) \times H^1(\Omega)\}$.

Partie fluide

En multipliant la première équation (équation des moments) par $\mathbf{v}$ et en intégrant sur Ω, nous obtenons

$$\int_\Omega \frac{\partial \mathbf{u}}{\partial t}\cdot\mathbf{v}\,dx + \int_\Omega (\mathbf{u}\cdot\nabla\mathbf{u})\cdot\mathbf{v}\,dx + \int_\Omega \frac{1}{\rho}\nabla p\cdot\mathbf{v}\,dx - \int_\Omega \nu(\Delta\mathbf{u})\cdot\mathbf{v}\,dx = \int_\Omega g\beta T\mathbf{e}_2\cdot\mathbf{v}\,dx \tag{2.2}$$

La formule d'intégration par parties nous donne

$$-\nu\int_\Omega (\Delta\mathbf{u})\cdot\mathbf{v}\,dx = -\nu\int_\Gamma (\nabla\mathbf{u})\mathbf{n}\cdot\mathbf{v}\,d\tau + \nu\int_\Omega \nabla\mathbf{u}:\nabla\mathbf{v}\,dx \tag{2.3}$$

et

$$\frac{1}{\rho}\int_\Omega \nabla p\cdot\mathbf{v}\,dx = \frac{1}{\rho}\int_\Gamma p\mathbf{n}\cdot\mathbf{v}\,d\tau - \frac{1}{\rho}\int_\Omega p\nabla\cdot\mathbf{v}\,dx\,. \tag{2.4}$$

L'équation de continuité nous donne de plus

$$\int_\Omega \frac{1}{\rho} q(\nabla \cdot \mathbf{u})\, dx = 0\,. \tag{2.5}$$

D'où

$$\int_\Omega \frac{\partial \mathbf{u}}{\partial t} \cdot \mathbf{v}\, dx + \int_\Omega (\mathbf{u} \cdot \nabla \mathbf{u}) \cdot \mathbf{v}\, dx + \nu \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v}\, dx + \frac{1}{\rho} \int_\Omega q(\nabla \cdot \mathbf{u}) - p(\nabla \cdot \mathbf{v})\, dx + \int_\Gamma (\frac{1}{\rho} pI - \nu \nabla \mathbf{u})\mathbf{n} \cdot \mathbf{v}\, d\tau = \int_\Omega g\beta T \mathbf{e}_2 \cdot \mathbf{v}\, dx \tag{2.6}$$

Notons alors

$$\sigma(p, \mathbf{u}) = (\frac{1}{\rho} pI - \nu \nabla \mathbf{u})\mathbf{n}\,,$$

le terme $\int_\Gamma \sigma(p, \mathbf{u}) \cdot \mathbf{v}\, d\tau$ permettant d'imposer les conditions aux bords.

Partie thermique

Dans la même idée, nous mutliplions la dernière équation 3.1 par la fonction test s et nous intégrons sur Ω:

$$\int_\Omega \frac{\partial T}{\partial t} s\, dx + \int_\Omega \mathbf{u} \cdot (\nabla T) s\, dx - \int_\Omega k(\Delta T) s\, dx = 0\,. \tag{2.7}$$

Comme précédemment, la formule d'intégration par parties nous donne

$$\int_\Omega k(\Delta T) s\, dx = \int_\Gamma k \frac{\partial T}{\partial \mathbf{n}} s\, d\tau - \int_\Omega k \nabla T \cdot \nabla s\, dx \tag{2.8}$$

D'où

$$\int_\Omega \frac{\partial T}{\partial t} s\, dx + \int_\Omega \mathbf{u} \cdot (\nabla T) s\, dx + k \int_\Omega \nabla T \cdot \nabla s\, dx - k \int_\Gamma \frac{\partial T}{\partial \mathbf{n}} s\, d\tau = 0\,. \tag{2.9}$$

Le terme $k \int_\Gamma \frac{\partial T}{\partial \mathbf{n}} s\, d\tau$ nous permettra d'imposer les conditions aux limites.

Implémentation

L'implémentation sous `Feel++` colle aux formes variationnelles définies ci-dessus.

Comme premier exemple, voici la déclaration du résidu de la fonction $F(U)$ pour l'algorithme de Newton (*cf.* 2.1.5).

```
      // -- Fluid -- //
    form1( Xh_F, _vector=R ) =
        integrate( _range = elements( mesh ),
                   _expr = trans( gradv(u)*idv(u) )*id(v) );
    form1( Xh_F, _vector=R ) +=
        integrate( _range = elements( mesh ),
                   _expr = b*trace( gradv(u)*trans( grad(v) ) ) );
    form1( Xh_F, _vector=R ) +=
        integrate( _range = elements( mesh ),
                   _expr = a*divv(u)*id(q) - a*idv(p)*div(v)  );

    // -- RHS -- //
#if CONVECTION_DIM == 2
    form1( Xh_F, _vector=R ) +=
        integrate( _range = elements( mesh ),
                   _expr = -expansion*trans( vec( cst(0.), idv(t) ) )*id(v) );
#else
    form1( Xh_F, _vector=R ) +=
```

```
        integrate ( _range = elements( mesh ),
                    _expr = -expansion*trans( vec( cst(0.), cst(0.),idv(t) ) )
                    *id(v) );
#endif

    // -- Time Component -- //
    form1( Xh_F, _vector=R ) +=
        integrate( _range = elements(mesh),
                   _expr = -trans( idv(bdfu_poly) )*id(v) );
    form1( Xh_F, _vector=R ) +=
        integrate( _range = elements(mesh),
                   _expr = trans( idv(u)
                   )*id(v)*bdf_F->polyDerivCoefficient(0) );
```

2.1.2 Traitement des conditions limites

Nous reprenons ici les différentes conditions aux bords envisagées. Les conditions de Dirichlet sont imposées faiblement; il faut pour cela rajouter quelques termes à la forme variationnelle comme décrite ci-dessous.

Fluide

Condition de Dirichlet : Soit une fonction $g \in L^2$, on pose

$$\mathbf{u} = \mathbf{g} \text{ sur } \mathbf{\Gamma_D}$$

Revenons alors sur le terme

$$\int_\Gamma \sigma(p, \mathbf{u}) \cdot \mathbf{v} \, d\tau$$

qui ne disparait pas de l'équation puisque $\mathbf{v}$ est supposée quelconque sur Γ_D. Nous rajoutons par contre un terme de symétrisation

$$\int_{\Gamma_D} \sigma(q, \mathbf{v}) \cdot (\mathbf{u} - \mathbf{g}) \, \mathbf{d}\tau \, ,$$

ainsi qu'un terme de pénalisation

$$\int_{\Gamma_D} \gamma(\mathbf{u} - \mathbf{g}) \cdot \mathbf{v} \, \mathbf{d}\tau \, ,$$

tous les deux identiquement nuls sur l'ensemble Γ_D.

Condition de Neumann : on impose $\sigma(p, \mathbf{u}) = 0$ sur l'ensemble Γ_N. Le terme $\int_{\Gamma_N} \sigma(p, \mathbf{u}) \cdot \mathbf{v} \, d\tau$ disparait alors de la formulation variationnelle.

Cela revient à laisser le fluide se comporter comme il le souhaite; cette condition sera utilisée sur les sorties d'air (ce qui revient à dire qu'il n'y a pas d'aspiration).

La forme variationnelle de l'équation des moments devient alors

$$\int_\Omega \frac{\partial \mathbf{u}}{\partial t} \cdot \mathbf{v} \, dx + \int_\Omega (\mathbf{u} \cdot \nabla \mathbf{u}) \cdot \mathbf{v} \, dx + \nu \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v} \, dx + \frac{1}{\rho} \int_\Omega q(\nabla \cdot \mathbf{u}) - p(\nabla \cdot \mathbf{v}) \, dx +$$
$$\int_{\Gamma_D} \sigma(p, \mathbf{u}) \cdot \mathbf{v} \; + \sigma(q, \mathbf{v}) \cdot \mathbf{u} + \gamma \mathbf{u} \cdot \mathbf{v} \, d\tau$$
$$= \int_\Omega g\beta T \mathbf{e}_2 \cdot \mathbf{v} \, dx + \int_{\Gamma_D} \sigma(p, \mathbf{v}) \cdot \mathbf{g} + \gamma \mathbf{g} \cdot \mathbf{v} \, \mathbf{d}\tau \quad (2.10)$$

Thermique

Condition de Dirichlet : correspondant à une surface (Γ_D) maintenue selon un profil de température $T_D \in L^2$:

$$T = T_D \text{ sur } \Gamma_D \ .$$

Nous rajoutons au terme

$$k \int_{\Gamma_D} \frac{\partial T}{\partial \mathbf{n}} s \ d\tau$$

le terme de symétrisation

$$k \int_{\Gamma_D} \frac{\partial s}{\partial \mathbf{n}} (T - T_D) \ d\tau$$

et celui de pénalisation

$$k \int_{\Gamma_D} \gamma s (T - T_D) \ d\tau$$

Condition de Neumann sur la température : correspondant à un flux de chaleur constant sur le bord Γ_N :

$$k \frac{\partial T}{\partial \mathbf{n}} = F \text{ sur } \Gamma_N \ .$$

Le terme

$$k \int_{\Gamma_N} \frac{\partial T}{\partial \mathbf{n}} s \ d\tau$$

devient

$$\int_{\Gamma_N} F s \ d\tau \ .$$

Condition de Robin sur la température : correspondant à un flux de chaleur dépendant du différentiel de température sur le bord Γ_R :

$$k \frac{\partial T}{\partial \mathbf{n}} = \lambda (T_R - T) \text{ sur } \Gamma_R \ .$$

Le terme

$$k \int_{\Gamma_R} \frac{\partial T}{\partial \mathbf{n}} s \ d\tau$$

devient

$$\int_{\Gamma_R} \lambda (T_R - T) s \ d\tau \ ,$$

λ étant un coefficient de conductivité thermique.

La forme variationnelle de l'équation d'énergie devient alors

$$\begin{aligned} \int_\Omega \frac{\partial T}{\partial t} s \ dx + \int_\Omega \mathbf{u} \cdot (\nabla T) s \ dx + k \int_\Omega \nabla T \cdot \nabla s \ dx \\ - k \int_{\Gamma_D} \frac{\partial T}{\partial \mathbf{n}} s + \frac{\partial s}{\partial \mathbf{n}} T + \gamma s T \ d\tau - \int_{\Gamma_N} F s \ d\tau \ + \int_{\Gamma_R} \lambda (T - T_R) s \ d\tau \\ = -k \int_{\Gamma_D} \frac{\partial s}{\partial \mathbf{n}} T_D + \gamma s T_D \ d\tau \ . \end{aligned} \quad (2.11)$$

Implémentation des conditions aux bords

Les conditions limites sont données à l'aide d'un fichier de configuration, ce qui évite de devoir recompiler le code à chaque changement. Elles sont ensuite lues et les matrices sont remplies de façon adéquate grâce au code suivant

```
    // -- SET BC ON TEMPERATURE -- //
    for (int i=0 ; i<BCT.size() ; i++ ){
      auto T0 = BCT[i].expr[0] ;
      double lambda = BCT[i].lambda;
      // Dirichlet conditions T=T0
      if (BCT[i].type == 1){
        bl += integrate ( _range = markedfaces( mesh, BCT[i].marker  ),
                          _expr = -c*gradt(t)*N()*id(s)
                                  - c*grad(s)*N()*idt(t)
                                  + c*gamma*idt(t)*id(s)/hFace() );

        l += integrate ( _range = markedfaces( mesh, BCT[i].marker  ),
                         _expr = - c*grad(s)*N()*T0
                                 + c*gamma*T0*id(s)/hFace() );
      }
      // Neumann conditions
      else if (BCT[i].type == 2){
        l += integrate ( _range = markedfaces( mesh, BCT[i].marker ),
                         _expr = id(s)*lambda/rho/Cp*T0 );
      }
      // Robin conditions
      else if (BCT[i].type == 3){
        bl += integrate ( _range = markedfaces( mesh, BCT[i].marker ),
                          _expr = id( s )*lambda /rho/Cp*idt(t) );
        l += integrate ( _range = markedfaces( mesh, BCT[i].marker ),
                         _expr = id(s)*lambda/rho/Cp*T0  );
      }
    }
```

La partie fluide s'implémente comme suit:

```
    // -- Boundary Conditions -- //
    for (int i=0 ; i<BCU.size() ; i++ ){
        // Dirichlet conditions : u = U0
        if (BCU[i].type == 1){
            auto ux = BCU[i].expr[0];
            auto uy = BCU[i].expr[1];
#if CONVECTION_DIM==2
            auto U0 = vec(ux,uy);
#else
            auto uz = BCU[i].expr[2];
            auto U0 = vec(ux,uy,uz);
#endif
            form1( Xh_F, R ) +=
                integrate( _range = markedfaces(mesh, BCU[i].marker),
                           _expr = -trans( SigmaNv )*id(v)
                           - trans( SigmaN )*( idv(u) - U0 )
                           + b*gamma*trans( idv(u) - U0 )*id(v)/hFace() );
        }
        // Neumann conditions
        else if (BCU[i].type == 2){
            auto ux = BCU[i].expr[0];
            auto uy = BCU[i].expr[1];
#if CONVECTION_DIM==2
            auto U0 = vec(ux,uy);
#else
            auto uz = BCU[i].expr[2];
            auto U0 = vec(ux,uy,uz);
#endif
            form1( Xh_F, _vector=R ) +=
                integrate( _range = markedfaces( mesh, BCU[i].marker ),
                           _expr = -trans( U0 )*id(v) );
        }
```

```
        // No-slip : u = 0
        else if ( BCU[i].type == 4 ){
            form1( Xh_F, R ) +=
                integrate( _range = markedfaces(mesh, BCU[i].marker),
                           _expr = -trans( SigmaNv )*id(v)
                           - trans( SigmaN )*idv(u)
                           + b*gamma*trans( idv(u) )*id(v)/hFace() );
        }
    }
```

2.1.3 Espaces de fonctions

Soit $\mathcal{T}_h = \cup L_k$ le maillage choisi sur le domaine de résolution Ω. Nous choisissons de rechercher notre solution dans l'espace de fonctions $[\mathbb{P}_h^2]^2 \times \mathbb{P}_h^1 \times \mathbb{P}_h^2$ avec la définition suivante

$$\mathbb{P}_h^i = \{v \in \mathcal{C}^0(\Omega), v|_{L_k} \in \mathbb{P}_i\} \tag{2.12}$$

où $\mathbb{P}_i$ est l'ensemble des polynomes de degré au plus i.

Implémentation des espaces de fonctions

Ces espaces de fonctions sont construits au moyen de classes templates généralement utilisées avec des `typedef` comme ici:

```
    typedef Lagrange<Order_s, Vectorial,Continuous,PointSetFekete> basis_u_type;
    typedef Lagrange<Order_p, Scalar,Continuous,PointSetFekete> basis_p_type;
    typedef Lagrange<Order_t, Scalar,Continuous,PointSetFekete> basis_t_type;

    typedef bases< basis_u_type , basis_p_type , basis_l_type > basis_type_F;
    typedef bases< basis_t_type > basis_type_T;

    typedef FunctionSpace<mesh_type, basis_type_F> space_type_F;
    typedef FunctionSpace<mesh_type, basis_type_T> space_type_T;

    typedef typename element_type_F:: sub_element<0>::type element_u_type;
    typedef typename element_type_F:: sub_element<1>::type element_p_type;
    typedef typename element_type_T:: sub_element<0>::type element_t_type;
```

2.1.4 Schéma temporel

Voici les choix que nous avons fait en ce qui concerne la discrétisation temporelle :

- Approximation de la dérivée d'ordre 1 (BDF)
- Schéma implicite
- Découplé Fluide/Température
- Boucle de correction à chaque itération

Note : Si *a priori* notre code a été écrit pour tourner avec un ordre de BDF quelconque jusqu'à 4, nous avons rencontré des instabilités pour les ordres supérieurs à 1 dûes au fait que les conditions limites sont imposées un peu brutalement au démarrage. Ceci nous a donc poussés à introduire une initialisation progressive des conditions limites décrites par les équations 2.17.

Soit δt le pas de temps, on note $\mathbf{u}_n, p_n$, et T_n les champs de vitesse, pression et température calculés au temps $t_0 + n\delta t$. Une fois la discrétisation temporelle effectuée, il s'agit à chaque pas de temps de trouver $\mathbf{u}_n, p_n$, et T_n tels que

$$\begin{cases} \displaystyle\int_\Omega \frac{\mathbf{u}_n}{\delta t} \cdot \mathbf{v}\, dx + \int_\Omega (\mathbf{u}_n \cdot \nabla \mathbf{u}_n) \cdot \mathbf{v}\, dx + \nu \int_\Omega \nabla \mathbf{u}_n : \nabla \mathbf{v}\, dx + \\ \qquad \dfrac{1}{\rho} \displaystyle\int_\Omega q(\nabla \cdot \mathbf{u}_n) - p_n(\nabla \cdot \mathbf{v})\, dx + \int_\Gamma \sigma(p_n, \mathbf{u}_n) \cdot \mathbf{v}\, d\tau \\ \qquad\qquad - \displaystyle\int_\Omega g\beta T_n \mathbf{e}_2 \cdot \mathbf{v}\, dx = \int_\Omega \frac{\mathbf{u}_{n-1}}{\delta t} \cdot \mathbf{v}\, dx, \\ \displaystyle\int_\Omega \frac{T_n}{\delta t} \cdot s\, dx + \int_\Omega \mathbf{u}_n \cdot (\nabla T_n) s\, dx + \\ \qquad k \displaystyle\int_\Omega \nabla T_n \cdot \nabla s\, dx - k \int_\Gamma \frac{\partial T_n}{\partial \mathbf{n}} s\, d\tau = \int_\Omega \frac{T_{n-1}}{\delta t} \cdot s\, dx. \end{cases} \tag{2.13}$$

En pratique, la résolution étant découplée, nous résolvons d'abord la partie thermique. Ainsi à chaque itération, nous calculons T_{n+1} en utilisant $\mathbf{u}_n$ comme paramètre de l'équation d'énergie puis nous calculons $(\mathbf{u}_{n+1}, p_{n+1})$ en utilisant T_{n+1} comme paramètre dans l'équation des moments.

Une correction est ensuite effectuée en recalculant T_{n+1} à partir de $\mathbf{u}_{n+1}$. Cette correction est effectuée en boucle tant que la différence entre deux valeurs de T_{n+1} n'est pas suffisamment petite. La figure 2.1 résume le schéma utilisé.

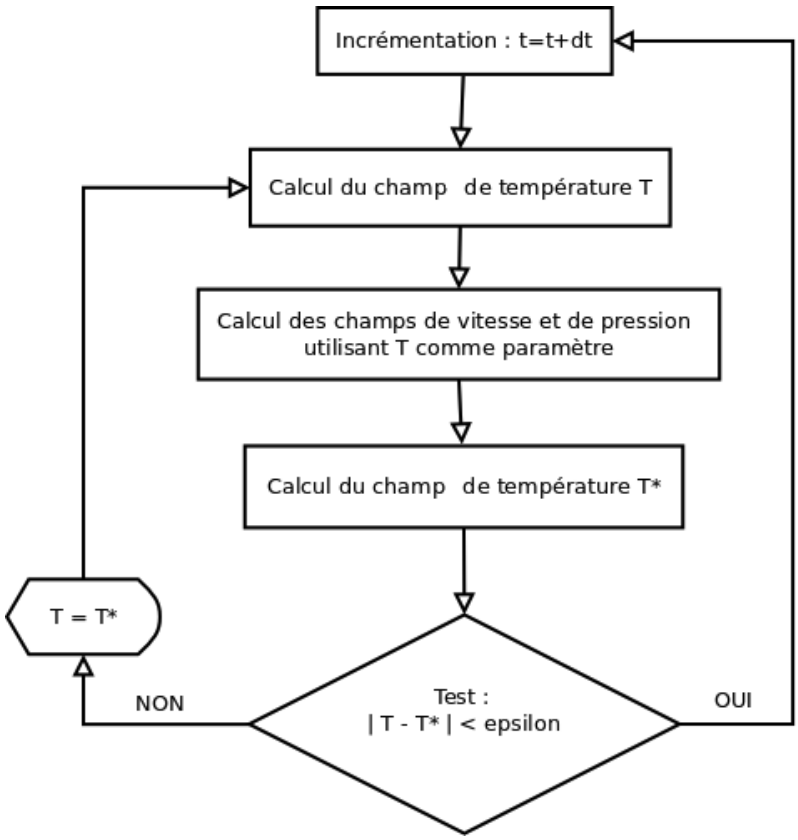

Figure 2.1: Schéma de résolution

Implémentation du schéma temporel

Feel++ contient une classe BDF qui fournit tous les outils pour construire une boucle temporelle ainsi que les approximations des dérivées à l'ordre désiré. Ci-dessous la boucle principale de l'algorithme de résolution.

```
    bdf_T->start();
for( bdf_F->start() ;  bdf_F->isFinished()==false ; bdf_F->next() ){
       // -- HEAT SUB-LOOP -- //
       while ( (bdf_T->time() < bdf_F->time()) && bdf_T->isFinished()==false )
       {
           this->solveT();
           bdf_T->shiftRight(t);
           bdf_T->next();
       }

       // -- SOLVE T -- //
       this->solveT();

       // -- SOLVE U -- //
       M_backend_F->nlSolve( _solution = U );

       // -- CORRECTION -- //

       Tbuff = t;
       this->solveT();
       Tdiff = normLinf( _range=elements( mesh ),
                         _expr= idv(Tbuff)-idv(t),
                         _pset=_Q<5>() ).template get<0>() ;
       while( Tdiff>onionTol) {
         M_backend_F->nlSolve( _solution = U );
         Tbuff = t;
         this->solveT();
         Tdiff = normLinf( _range=elements( mesh ),
                           _expr= idv(Tbuff)-idv(t),
                           _pset=_Q<5>() ).template get<0>() ;
       }

       // -- INCREMENTATION -- //
       bdf_F->shiftRight(U);
       bdf_T->shiftRight(t);
       bdf_T->next();
   }
```

2.1.5 Outils numériques

Méthode de Newton

La résolution de la partie fluide étant clairement non linéaire, nous faisons le choix d'une résolution à l'aide d'une méthode de Newton dont nous rappelons brièvement le principe.

Le but est de résoudre une équation de type $F(U) = 0$ avec F une fonction différentiable de $\mathbb{R}^N$ dans $\mathbb{R}^N$.

Algorithme :

1. Initialisation avec un vecteur U_0 bien choisi.
2. A chaque itération, on cherche U_{k+1} tel que

$$F'(U_k)(U_{k+1} - U_k) + F(U_k) = 0$$

   $F'(U_k)$ étant la matrice Jacobienne de la fonction F au point U_k
3. L'algorithme s'arrête lorsque la quantité $|F(U_k)|$ est inférieure à un seuil choisi.

En pratique, les itérations ainsi que l'éxécution de l'algorithme sont gérées par `Feel++` via la classe `Backend`. Il suffit donc d'écrire les fonctions d'actualisation du vecteur $F(U_k)$ (`updateResidual`) et de la matrice $F'(U_k)$ (`updateJacobian`). Ces deux fonctions sont alors liées au solveur en quelques lignes:

```
M_backend_F->nlSolver()->residual = boost::bind( &self_type::updateResidual,
                                                 boost::ref( *this ), _1, _2 );
M_backend_F->nlSolver()->jacobian = boost::bind( &self_type::updateJacobian,
                                                 boost::ref( *this ), _1, _2 );
```

puis `Feel++` gère le reste à chaque appel de `M_backend_F->nlSolve( _solution = U )`

2.1.6 Multiplicateurs de Lagrange

Dans les cas où aucune condition aux limites ne fait intervenir la pression, on remarque que celle-ci est définie à une constante additive près. En pratique, cela correspond à une géométrie fermée, sans entrée ni sortie d'air, comme celle de la cavité entrainée.

Il existe alors plusieurs solutions pour se ramener à un problème bien posé. On peut par exemple imposer la valeur de la pression sur un unique noeud du maillage. Dans le cadre de notre programme, nous avons fait le choix de chercher une pression à moyenne nulle, `i.e.` $\int_\Omega p \, dx = 0$. L'introduction de cette contrainte supplémentaire se passe très bien avec l'utilisation d'un multiplicateur de Lagrange. On cherche alors $U = (\mathbf{u}, p, T, \xi) \in [H^1(\Omega)]^n \times L^2(\Omega) \times H^1(\Omega) \times \mathbb{R}$ tel que

$$\int_\Omega \frac{\partial \mathbf{u}}{\partial t} \cdot \mathbf{v} \, dx + \int_\Omega (\mathbf{u} \cdot \nabla \mathbf{u}) \cdot \mathbf{v} \, dx + \nu \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v} \, dx + \frac{1}{\rho} \int_\Omega q(\nabla \cdot \mathbf{u}) - p(\nabla \cdot \mathbf{v}) \, dx + \int_\Gamma \sigma(p, \mathbf{u}) \cdot \mathbf{v} \, d\tau + \int_\Omega p\eta + q\xi \, dx = \int_\Omega g\beta T \mathbf{e}_2 \cdot \mathbf{v} \, dx \quad (2.14)$$

$$\int_\Omega \frac{\partial T}{\partial t} s \, dx + \int_\Omega \mathbf{u} \cdot (\nabla T) s \, dx + k \int_\Omega \nabla T \cdot \nabla s \, dx - k \int_\Gamma \frac{\partial T}{\partial \mathbf{n}} s \, d\tau = 0 \, . \quad (2.15)$$

pour tout $(\mathbf{v}, q, s, \eta) \in [H^1(\Omega)]^n \times L^2(\Omega) \times H^1(\Omega) \times \mathbb{R}$.
Pour des raisons pratiques d'implémentation, l'espace du multiplicateur est toujours utilisé. Une option du programme permet de modifier la forme variationnelle afin d'utiliser ou non le multiplicateur de Lagrange. Quand ce dernier n'est pas nécessaire, le terme $\int_\Omega p\eta + q\xi \, dx$ est remplacé par $\int_\Omega \xi\eta \, dx$, ce qui permet de garder une matrice inversible en remplissant le dernier élément avec un 1 sans intervenir sur le reste du système.

Implémentation des multiplicateurs de Lagrange L'espace de fonctions "fluide" a donc été légèrement agrandi afin d'y faire tenir le multiplicateur de Lagrange. Les quelques lignes sont également introduites dans la construction du résidu.

```
    // -- Lagrange Multiplier -- //
    if (LM){
        form1( Xh_F, _vector=R ) +=
            integrate( _range= elements( mesh ),
                       _expr = id(q)*idv(xi) + idv(p)*id(eta) );
    }
    else {
        form1( Xh_F, _vector=R ) +=
            integrate( _range = elements( mesh ),
                       _expr = idv(xi)*id(eta) );
    }
```

2.2 FreeFem++

FreeFem++ (http://www.freefem.org/ff++/, [4] et [3]) est un solveur numérique qui permet de résoudre, par la méthode des éléments finis, des équations aux dérivées partielles en 2D comme en 3D. Ecrit en C++, ce logiciel possède un générateur automatique de maillages et emploie des algorithmes itératifs pour traiter les cas paraboliques et hyperboliques. De plus, la parallélisation – non utilisée pour ce projet – permet d'apporter encore plus de puissance à ce solveur [2].
De nombreux exemples sont déjà implémentés ce qui facilite la prise en main; ils nécessitent néanmoins une maîtrise des formes variationnelles des équations aux dérivées partielles associées.

2.2.1 Présentation

Dans FreeFem++, les équations de Navier-Stokes dans l'approximation de Boussinesq [7] sont déjà implémentées. Par conséquent, il nous suffit de construire la géométrie de la salle machine (cf. chapitre 4) pour se ramener à notre cas i.e. un inlet, un outlet ainsi que deux "rectangles" pour représenter les core 1 1 et core 2. Malgré la présence d'un générateur puissant de maillages dans FreeFem++, nous devons importer le maillage imposé (sous la forme .msh) pour satisfaire les conditions du benchmark.
Ceci se fait dans FreeFem++ comme suit :

```
// IMPORTATION MESH

load "gmsh";
mesh Th = gmshload("bensh.msh");
plot(Th,wait=1);


// DONNEES DU PROBLEME

real nu             = 0.5e-2;
real rho            = 1.;
real kT             = 0.5e-2;
real beta           = -3.4112e-3; // car on a un - dans l'equation.
real C              = 1000.;
real g              = 9.81;
real lambda2        = 5.;
real Uin            = 0.75;
real T0init         = 20.;
real Tc1            = 40.;
real Tc2            = 60.;
real Ti             = 15.;
real dt             = 1.;
int  Nitert         = 30;


//DONNEES COMPLEMENTAIRES

int  NiterNewtonMax = 1; // nombre d'iterations pour Newton.
int  i              = 0;
real t              = 0;
real eps            = 1e-6;
real br2            = lambda2/(C*rho);


// LABELISATION DES BORDS

int lin = 1, ldir = 2, lcore1 = 3, lcore2 = 4, lout = 5;


// MACROS POUR LA FORMULATION VARIATIONNELLE
```

```
macro grad(u)        [dx(u),dy(u)] //
macro Grad(u)        [grad(u#1),grad(u#2)] //
macro div(u)         ( dx(u#1) + dy(u#2) ) //
macro ugrad(u,v)     ( [u#1,u#2]' * grad(v) ) //
macro UgradV(u,v,T)  [ugrad(u,v#1),ugrad(u,v#2),ugrad(u,T)] //
```

Dans FreeFem++, il n'y a pas besoin de découpler les équations car l'implémentation est telle qu'elle permet de résoudre la partie fluide et thermique dans le même code. Nous obtenons une formulation variationnelle qui fait apparaître le champ de vitesse, la pression et la température comme variables du système d'équations. Nous implémentons aussi des macros nécessaires à la formulation variationnelle du problème.
Mais tout d'abord, nous allons amener quelques précisions sur l'implémentation des espaces de fonctions.

2.2.2 Espaces de fonctions

Nous prenons les mêmes espaces de fonctions que définis précédemment, i.e. $[P_h^2]^2$ pour le champ de vitesse, P_h^1 pour la pression ainsi que P_h^2 pour la température.
On génère les variables, fonction-tests; on initialise le champ de vitesse, pression et température aux conditions initiales puis on implémente un profil de vitesse parabolique au niveau de l'inlet :

```
// DISCRETISATIONS DES ESPACES

fespace Mh(Th,[P2,P2,P1,P2]); // vectorial FE space for velocity and pressure
fespace Vh(Th,P2);
fespace Wh(Th,P2); // pour calculer la T init


// CREATION DES VARIABLES

Mh [u1,u2,p,T];
Mh [up1,up2,pp,Tp]; // valeur au pas de temps precedent
Mh [upp1,upp2,ppp,Tpp]; // valeur au pas encore d'avant
Mh [uppp1,uppp2,pppp,Tppp]; // valeur au pas encore d'avant
Mh [uw1,uw2,pw,Tw];
Mh [v1,v2,q,TT];
Vh U;


// INITIALISATION DES VARIABLES

Wh Tinit, TTinit, Tc;
Wh u1init = (y>0.9) & (x<0.001) ? 400*(y-0.9)*(1-y) : 0;

[u1,u2,p,T] = [0,0,0,T0init];
```

Dans cet exemple, on initialise la température à une valeur constante ($T0init$) sur tout le domaine. Cependant, comme on le verra plus loin en 2.2.5, cette initialisation n'est pas forcément la meilleure.

2.2.3 Formulation faible et conditions aux limites

Présentons maintenant l'implémentation sous FreeFem++ de la formulation variationnelle des équations de Navier-Stokes dans l'approximation de Boussinesq ainsi que des conditions aux limites associées.

```
// COEFFICIENTS POUR LE SCHEMA EN TEMPS

real[int] cdt1 = [1./dt,-1./dt,0.,0.];                 // ordre 1
real[int] cdt2 = [1.5/dt,-2./dt,0.5/dt,0.];            // ordre 2
real[int] cdt3 = [11./6./dt,-3./dt,1.5/dt,-1./3./dt]; // ordre 3


// FORMULATION VARIATIONNELLE ET CONDITIONS LIMITES

for(int itert=1;itert<=Nitert;itert++)
{
    uppp1[] = upp1[];
    upp1[]  = up1[];
    up1[]   = u1[];
    t = itert*dt;
    // Initialisation progressive de la temperature
    Tc2 = T0init + 60.*(1-exp(-0.5*t));

    for (i=0;i<NiterNewtonMax;i++) {
        solve BoussinesqNL([uw1,uw2,pw,Tw],[v1,v2,q,TT])

        // DF(Un) * Un+1
        = int2d(Th) (
        [uw1,uw2,Tw]' * [v1,v2,TT] * cdt1[0]
        + UgradV(u,uw,Tw)' * [v1,v2,TT]
        + UgradV(uw,u,T)'  * [v1,v2,TT]
        + (Grad(uw):Grad(v)) * nu
        + g * beta * Tw * v2
        + grad(Tw)' * grad(TT) * kT
        - div(uw) * q - div(v) * pw
        )
        + int1d(Th,4)(TT * Tw * br2)

        // -DF(Un) + F(Un)
        + int2d(Th)(
        [up1,up2,Tp]' * [v1,v2,TT] * cdt1[1]
        + [upp1,upp2,Tpp]' * [v1,v2,TT] * cdt1[2]
        + [uppp1,uppp2,Tppp]' * [v1,v2,TT] * cdt1[3]
        - UgradV(u,u,T)' * [v1,v2,TT]
        + g * beta * (-T0init) * v2
        )
        - int1d(Th,4)(TT * Tc2 * br2)

        // Initialisation progressive de la vitesse
        + on(lin, uw1 = Uin*u1init*(1 - exp(-0.5*t)), uw2 = 0.)

        + on(ldir, lcore1, lcore2, uw1=0., uw2=0.)
        + on(lin, Tw = T0init - 5*(1 - exp(-0.5*t)))
        + on(3, Tw = T0init + 20*(1 - exp(-0.5*t)));

        u1[] = uw1[];

        if (u1[].l2 < 1e-4) break;

        // Etablissement progressif de l'ordre
        cdt1 = cdt2;
        cdt2 = cdt3;

        plot(T,wait=0,cmm="temperature en t = "+t,fill=1);

    }
}
```

Les conditions aux limites de type Dirichlet sont implémentées grâce à la fonction `on(`*`bord`*`,`*`condition`*`)`. Les conditions de type Neumann sont quant à elles imposées grâce aux intégrales de type `int1d`.

2.2.4 Schéma temporel

Comme schéma en temps, nous avons choisi le schéma BDF (Backward Differenciation Formula) qui est une méthode multi-pas implicite où l'on écrit l'EDO au temps t_{n+1} et remplace la dérivée par un taux d'accroissement d'ordre élevé.

$$y' = f(t, y), \quad y(t_0) = y_0$$

La méthode nécessite la résolution de l'EDP à chaque pas de temps. On fera pour cela une seule itération de la méthode de Newton en temps. Présentons le schéma à différents ordres que nous avons implémenté dans FreeFem++ :

- BDF1: $y_{n+1} - y_n = hf(t_{n+1}, y_{n+1})$
- BDF2: $y_{n+2} - \frac{4}{3}y_{n+1} + \frac{1}{3}y_n = \frac{2}{3}hf(t_{n+2}, y_{n+2})$
- BDF3: $y_{n+3} - \frac{18}{11}y_{n+2} + \frac{9}{11}y_{n+1} - \frac{2}{11}y_n = \frac{6}{11}hf(t_{n+3}, y_{n+3})$

Sachant que le schéma BDF est A-stable jusqu'à l'ordre 2, il n'est plus utile de monter plus haut en ordre pour chercher plus de stabilité. Mais par conséquent, cela nous permettra d'augmenter le pas de temps quand nous utilisons l'ordre 3 pour obtenir les mêmes résultats que pour un pas de temps plus petit sur un ordre en dessous, ce qui finalement, nous permet d'accélérer considérablement le calcul (cf. section 4.3)

Pour l'implémentation de ce schéma dans FreeFem++, tout d'abord, nous avons besoin d'initialiser les coefficients du schéma à différents ordres dans des tableaux que nous nommerons `cdt1`, `cdt2` et `cdt3` qui représentent respectivement l'ordre du schéma de manière croissante. On initialise les valeurs de u_2, u_1 et u_0 à u_1 puis on l'implémente comme décrit ci-dessous.

A la première itération de temps, nous commençons à l'ordre 1. Puis à la deuxième, on monte à l'ordre 2. Et à la troisième, on passe à l'ordre 3 et nous resterons à cette ordre jusqu'à la fin des itérations en temps. De cette manière, nous pouvons éviter les instabilités des premières itérations dues aux changements trop brutales des conditions limites. Pour remédier à ce problème, nous avons aussi implémenté un nouveau `Tc2` qui augmente de manière progressive avec le temps. Comme cela, nous pouvons même utiliser l'ordre 3 dès la première itération temporelle sans qu'il y ait d'instabilité.

2.2.5 Méthode de Newton

La formulation variationnelle établie au chapitre 1 peut s'écrire sous une forme simplifiée: Trouver $u_h \in V_h$ vérifiant des conditions de Dirichlet sur le bord et tel que:

$$F(u_h, v_h) = 0 \quad \forall v_h \in V_{0h}$$

où V_h est l'espace des éléments finis $[\mathbb{P}^2, \mathbb{P}^1, \mathbb{P}^2]$ sur le maillage.

Pour résoudre cette équation, nous utiliserons la méthode de Newton qui s'écrit comme suit:

- Étant donnée u_h^0 vérifiant les conditions aux limites,
- Trouver $w_h^n \in V_{0h}$ solution de: $\mathrm{dF}(u_h^n, v_h)(w_h^n) = F(u_h^n, v_h) \quad \forall v_h \in V_{0h}$
  où $\mathrm{dF}(u, v)$ est la différentielle de F.

- $u_h^{n+1} = u_h^n - w_h^n$
- arrêt: $||w_h^n||_{L_2} < \varepsilon$.

Remarque: Étant donnée $w_h^n \in V_{0h}$, les conditions de type Dirichlet vérifiées par u_h^n ne varient pas au cours de l'algorithme; elles doivent être donc initialisées une bonne fois pour toute dans u_h^0.

Ainsi, une étape importante de l'algorithme consiste à initialiser u_h^0 afin d'y inclure les conditions de Dirichlet.

Initialisation de la cavité

- Initialisation de la vitesse : la vitesse est initialisée à 0 dans tout le domaine.
- Initialisation de la température : la température étant fixée à une valeur de 1 sur le bord gauche et une valeur de 0 dans le cas adimensionné, on initialise celle-ci à l'aide de la fonction suivante :

  $$T(x) = 1 - x$$

  On a donc bien :

  $$T_{(x=0)} = 1;$$
  $$T_{(x=1)} = 0;$$

Pour la cavité, on utilise donc l'initialisation suivante:

$$u_h^0 = [0, 0, T(x, y)] = [0, 0, 1 - x]$$

Remarque: Cette initialisation linéaire de la température n'est pas forcément la meilleure, en particulier pour un Rayleigh grand.

Initialisation de la salle machine

- Initialisation de la vitesse : Pour ce qui est de la salle machine, le fluide est entraîné par un profil de Poiseuille en entrée (Inlet) défini par:

  $$v(y) = 0.75(1 - y)(y - 0.9)400 \quad \forall (x, y) \in \{0\} \times [0.9, 1]$$

  On peut voir ce profil comme une condition de Dirichlet sur l'Inlet. Ailleurs, on initialise la vitesse à 0.
- Initialisation de la température: Trois possibilités sont retenues.

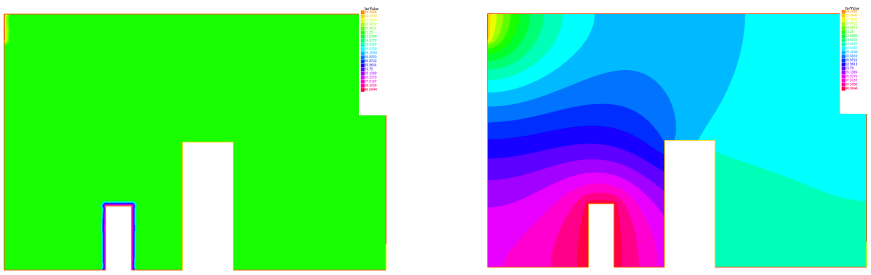

Figure 2.2: Deux premières possibilités pour l'initialisation de la température

1. La première possibilité consiste à fixer la température à une valeur de référence ($T_{\text{ref}} = 20$ degrés), et on impose les conditions de Dirichlet sur les bords. Cette initialisation est effectuée comme suit sous `FreeFem++`:

```
/*  Premier choix d'initialisation */

real Uin = 0.75; // Vitesse en entree

real Tin=15.;   // Temperature en entree
real Tc1 = 40;  // Temperature du core 1
real T0init=20.;// Temperature de reference

// On impose les conditions de Dirichlet:
varf vTinit(Tinit, TTinit)
= on(lcorps1,Tinit=Tc1) + on(lin,Tinit=Tin);
Tinit[] =vTinit(0,Wh,tgv=1);
Tinit[] = Tinit[]? Tinit[] : T0init;

// A l'entree: profil de Poiseuille
Wh u1init = (y>0.9) & (x<0.001) ? (y-0.9)*(1-y)*400 : 0;

// Initialisation
[u1,u2,p,T]=[u1init*Uin,0,0,Tinit];
```

2. Une autre initialisation possible de la température est de résoudre l'équation de la chaleur:

$$\begin{cases} \Delta T = 0 \\ \text{Conditions de Robin et de Dirichlet sur les bords} \end{cases} \tag{2.16}$$

Physiquement, cette initialisation correspondrait à faire chauffer la salle machine, puis, une fois la température établie, on fait entrer le fluide. On procède ainsi pour initialiser:

```
/* 2eme possibilite d'initialisation: */

real Uin = 0.75;
real Tc1 = 40;
real Tc2 = 60;
real Tini = 20;
real Ti=15;
real T0init=20;

Wh Tinit, TTinit;

// On resout l'equation de la chaleur
solve PTinit(Tinit, TTinit,solver = CG) =
int2d(Th)(kT*grad(Tinit)'*grad(TTinit)) + on(lcore1,Tinit=Tc1)
- int1d(Th,lcore2)(br2*TTinit*Tc2)
+ int1d(Th,lcore2)(br2*TTinit*Tinit)+on(lin,Tinit=Ti);

// A l'entree: profil de Poiseuille:
Wh u1init = (y>0.9) & (x<0.001) ? (y-0.9)*(1-y)*400 : 0;

// Initialisation:
[u1,u2,p,T]=[u1init*Uin,0,0,Tinit];
```

3. Enfin, on peut, et c'est ce que le benchmark impose, utiliser des conditions progressives aux limites qui dépendent du temps. Les conditions de Dirichlet deviennent:

$$\begin{cases} v(t) = v_{\text{Poiseuille}}(1 - e^{-t/2}) \text{ sur l'inlet} \\ T(t) = T_{\text{final}} - (T_{\text{final}} - T_{\text{ref}})e^{-t/2} \text{ sur tous les bords.} \end{cases} \tag{2.17}$$

Physiquement, cette initialisation est la plus réaliste: on fait entrer le fluide dans la salle machine et on chauffe en même temps, le tout de manière progressive. Ainsi, la température est à $T_{\text{ref}} = 20$ degrés partout, puis on commence à chauffer (notamment grâce au `core 1`).

Conditions aux limites progressives

Comme on a pu le voir, la méthode de Newton ne nous permet pas directement d'implémenter des conditions aux limites progressives. En effet, dans l'algorithme décrit précédemment, les conditions aux limites sont fixées une fois pour toutes dans u_h^0, et seules les composantes du vecteur correspondant à l'intérieur du domaine sont modifiées, w_h^0 étant nulle au bord. Nous allons donc adapter le schéma à notre problème en le réécrivant:

$$\begin{aligned} & dF(u_h^n, v_h)(w_h^n) = F(u_h^n, v_h) \\ \Rightarrow \quad & dF(u_h^n, v_h)(u_h^{n+1}) = dF(u_h^n, v_h) - F(u_h^n, v_h) \end{aligned}$$

On peut alors définir les conditions limites directement sur u_h^n. L'algorithme devient:

- Prendre $u_h^0 = [0, 0, T_{ref}]$
- Trouver u_h^{n+1} tel que:
  - $v = v(t)$ sur l'inlet
  - $T = T(t)$ sur les bords Dirichlet
  - $\forall v_h \in V_{0h}$: $dF(u_h^n, v_h)(u_h^{n+1}) = dF(u_h^n, v_h) - F(u_h^n, v_h)$
  - Arrêt: $\|u_h^{n+1} - u_h^n\| < \varepsilon$

On peut ainsi définir directement, à chaque pas de temps, les conditions aux limites vérifiées par u_h^{n+1}.
L'implémentation sous `FreeFem++` se fait en quelques lignes. Pour résoudre l'équation matricielle obtenue à chaque itération, différents solveurs linéaires sont disponibles (UMFPACK, SUPERLU, Crout, gradient conjugué...). Ici, nous utiliserons le solveur utilisé par défaut par la fonction `solve`: UMFPACK.

```
/* Schema modifie pour l'initialisation progressive. */

for(int itert=1;itert<=Nitert;itert++)  // Boucle en temps
{
    up1[]=u1[];
    t=itert*dt;
    // Temperature de reference du CPU2 (fonction du temps)
    Tc2=T0init + 60.*(1-exp(-0.5*t));

    // Algorithme de Newton:
    for (i=0;i<NiterNewtonMax;i++)
    {
        // On resout DF(Un)*Un+1-DF(Un)+F(Un) = 0
        solve BoussinesqNL([uw1,uw2,pw,Tw],[v1,v2,q,TT]) =

        // DF(Un)*Un+1
        int2d(Th) (
        [uw1,uw2,Tw]'*[v1,v2,TT]* cdt1[0]
        + UgradV(u,uw,Tw)' * [v1,v2,TT]
        + UgradV(uw,u,T)' * [v1,v2,TT]
        + ( Grad(uw):Grad(v)) * nu
        + g*beta*Tw*v2
        + grad(Tw)'*grad(TT)*kT
        - div(uw)*q -div(v)*pw
        )
        + int1d(Th,4)(TT*Tw*br2)

        // -DF(Un)+F(Un)
        + int2d(Th)(
        [up1,up2,Tp]'*[v1,v2,TT]* cdt1[1]
        + [upp1,upp2,Tpp]'*[v1,v2,TT]* cdt1[2]
        - UgradV(u,u,T)' * [v1,v2,TT]
        + g*beta*(-Tini)*v2
```

```
        )
        - int1d(Th,4)(TT*Tc2*br2)

        // Conditions aux limites progressives
        + on(lin,uw1=Uin*u1init*(1-exp(-0.5*t)),uw2=0.)
        + on(ldir,lcore1,lcore2,uw1=0.,uw2=0.)
        + on(lin,Tw=T0init- 5*(1-exp(-0.5*t)))
        + on(3,Tw=T0init + 20*(1-exp(-0.5*t)));

        // Test d'arret
        err = u1[]-uw1[];
        if(err.linfty<1e-9) break;
        u1[] =uw1[];
    }
}
```

2.2.6 Pression à moyenne nulle

Dans le cas de la cavité, la pression étant définie à une constante additive près, on impose que celle-ci soit à moyenne nulle pour garantir l'unicité de la solution. Pour ce faire, on perturbe légèrement la condition d'incompressibilité $div(u) = 0$, en écrivant:

$$div(u) = \varepsilon p \text{ avec, par exemple } \varepsilon = 10^{-6}$$

La formulation variationnelle est alors de la forme:

$$\begin{aligned} a(u,v) + b(v,p) &= f(v) \\ b(u,q) - \varepsilon \int_\tau pq &= 0 \end{aligned}$$

2.2.7 Vers des grands Rayleigh

Le nombre adimensionné de Rayleigh représente le rapport entre les termes convectifs et conductifs. Lorsque celui-ci augmente, des couches limites apparaissent, et le schéma stationnaire décrit plus haut ne permet pas d'obtenir des solutions. En effet, celui-ci explose dès lors que $\mathrm{Ray} \sim 10^6$, même en prenant un pas de temps $\mathrm{dt} = 0.001$.

Une manière de résoudre ce problème est de revenir au cas stationnaire et d'effectuer une continuation sur le nombre de Rayleigh. Typiquement, cela consiste à effectuer l'algorithme de Newton à partir d'un Rayleigh assez petit, obtenu en le multipliant par un cœfficient inférieur à un. Une fois la convergence du Newton établie, on augmente le nombre de Rayleigh, et on réitère le procédé, jusqu'à atteindre la valeur souhaitée.

Remarque 1: A propos du passage du cas stationnaire au cas instationnaire Afin de pouvoir passer du code stationnaire au code instationnaire, on souhaite connaître le temps nécessaire à la convergence du code instationnaire vers un état stationnaire. Pour cela, on étudie la variation de la température en un point fixé (ici, le centre de la cavité $(0.5, 0.5)$) en fonction du temps.

On remarque, dans la figure 2.3, que la convergence vers l'état stationnaire est assurée au bout d'un temps assez court. On pourra ainsi se limiter, par exemple, à $t = 5$.

Remarque: La continuation sur le nombre de Rayleigh n'est pas forcément linéaire. En fait, on remarque que l'algorithme est assez stable une fois la convergence établie pour les premiers Rayleigh: l'augmentation peut ainsi se faire assez "brutalement".

Voici quelques lignes de l'implémentation sous FreeFem ++ : ce code permet d'obtenir des résultats pour des Rayleigh allant jusqu'à 10^8.

```
/*  Continuation sur le nombre de Rayleigh... */
int nstep = 20;
for(int step=1; step <= nstep; ++step)
{
```

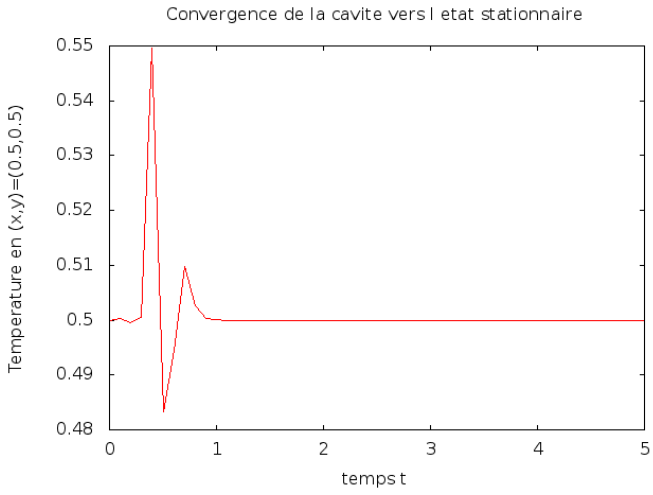

Figure 2.3: Variation au cours du temps de la température au centre de la cavité

```
    // Le coefficient multiplicatif du nombre de Rayleigh
    real coef = (step/real(nstep))^3;

    // Algorithme de Newton
    for (int k=0;k<=100;k++)
    {
      solve BoussinesqNL([uw1,uw2,pw,Tw],[v1,v2,q,TT])
      = int2d(Th) (
      UgradV(u,uw,Tw)' * [v1,v2,TT]
      + UgradV(uw,u,T)'  * [v1,v2,TT]
      - ( Grad(uw):Grad(v) ) * Pr
      + coef*Ray *Tw*Pr*v2  // CONTINUATION
      + grad(Tw)'*grad(TT)
      - div(uw)*q -div(v)*pw - eps*pw*q
      )
      - int2d(Th)(
      UgradV(u,u,T)' * [v1,v2,TT]
      - ( Grad(u):Grad(v) ) * Pr
      + coef*Ray *T*Pr*v2 // CONTINUATION
      +  grad(T)'*grad(TT)
      - div(u)*q -div(v)*p
      )
      + on(1,2,3,4, uw1=.0,uw2=.0) + on(4,Tw=0) + on(2,Tw=0);

      u1[] -= uw1[];
      real err=uw1[].linfty ;
      if(err < 1e-6) break;
      plot(T,wait=0,fill=1);
    };
}
```

L'apparition de couches limites sur les bords du domaine impose également de raffiner le maillage sur les bords...

2.2.8 Raffinement du maillage sur les bords

- Le mailleur de `FreeFem++` permet facilement de créer un maillage sur le carré unité de taille $N \times N$
- Pour des maillages non réguliers (par exemple, raffinés sur le bord), il est possible d'utiliser sous `FreeFem++` une fonction décrivant la répartition des mailles. Pour des Rayleigh grands, nous utiliserons la fonction suivante:

$$f(x) = \frac{1 + \tanh(c(2x-1))}{2\tanh(c)} \text{ avec } c \text{ une constante positive.}$$

Remarquons que, lorsque c est très petit (proche de 0), nous avons $f(x) \approx x$: on retrouve ainsi un maillage régulier. L'implémentation sous `FreeFem++` se fait de la manière suivante:

```
/* Maillage raffine sur les bords sous FreeFem++ */
load "MUMPS_seq"
real  c=4;
int N=80;
macro f(x) ((1+tanh(c*(x*2-1))/tanh(c))/2)//
mesh Th=square(N,N,[f(x),f(y)],flags=0);
plot(Th,wait=1);
```

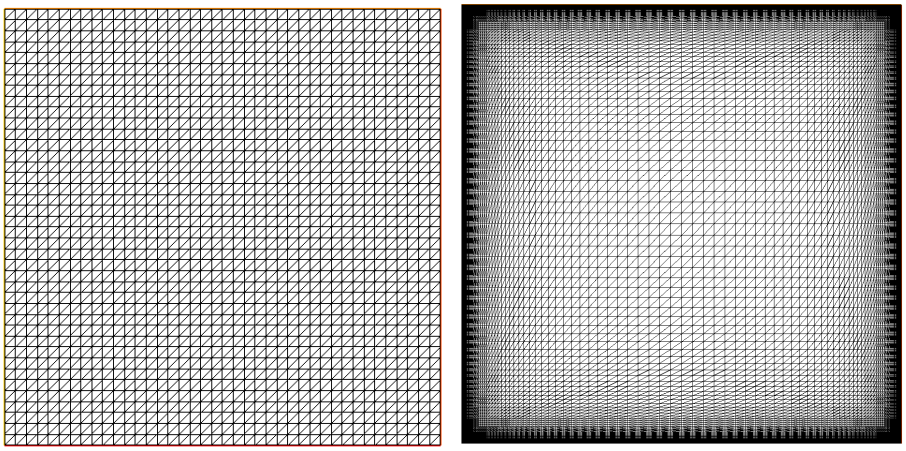

Figure 2.4: Maillage initial, et maillage raffiné

CHAPTER 3

Benchmark 1: cavité entrainée par convection naturelle

Cette section décrit le premier cas-test sur lequel nous nous sommes penchés, qui est celui de la cavité entrainée par convection naturelle. Ce cas-test, qui constitue une simplification du problème de la marche, présente l'avantage d'être assez référencé dans la littérature.

Dans cette section, on définira les équations ainsi que les conditions limites propres à ce cas. Nous comparons les résultats générés par `Feel++` et `FreeFem++` à ceux d'une étude de référence [9].

3.1 Description du modèle

On considère une cavité fermée à deux dimensions, modélisée par un carré unité (voir figure 3.1). Les parois verticales (gauche et droite) sont maintenues respectivement aux températures $T = T_C$ et $T = T_F$. Les parois horizontales sont considérées comme adiabatiques (aucun transfert de chaleur). Le fluide est supposé incompressible, Newtonien et pris dans l'approximation de Boussinesq. Les équations stationnaires adimensionées gourvernant notre problème – telles qu'elles sont données dans [9] – sont les suivantes :

$$\begin{cases} \mathbf{u}^* \cdot \nabla \mathbf{u}^* + \nabla p^* - Pr \Delta \mathbf{u}^* = RaPrT^* \mathbf{e_2} & , \text{ dans } \Omega, \\ \nabla \cdot \mathbf{u}^* = 0 & , \text{ dans } \Omega, \\ \mathbf{u}^* \cdot \nabla T^* - \Delta T^* = 0 & , \text{ dans } \Omega, \\ \mathbf{u}^* = 0 & , \text{ sur } \partial\Omega, \\ T^* = T_C^* = 1 & , \text{ sur } \Gamma_1, \\ T^* = T_F^* = 0 & , \text{ sur } \Gamma_3, \\ \dfrac{\partial T^*}{\partial \mathbf{n}} = 0 & , \text{ sur } \Gamma_2 \cup \Gamma_4, \end{cases} \tag{3.1}$$

On notera respectivement $\mathbf{u}^*$, p^*, T^* les champs de vitesse, de pression et de température adimensionnés. Les nombres de Rayleigh et de Prandtl seront notés quant à eux Ra et Pr. Plus de détails sont donnés dans [9].

Les conditions aux bords sur la vitesse sont de type No-slip. Pour la température, on utilisera des conditions de Dirichlet sur les parois latérales gauche et droite. Les parois horizontales seront modélisées par une condition de Neumann homogène (pas de transfert thermique).

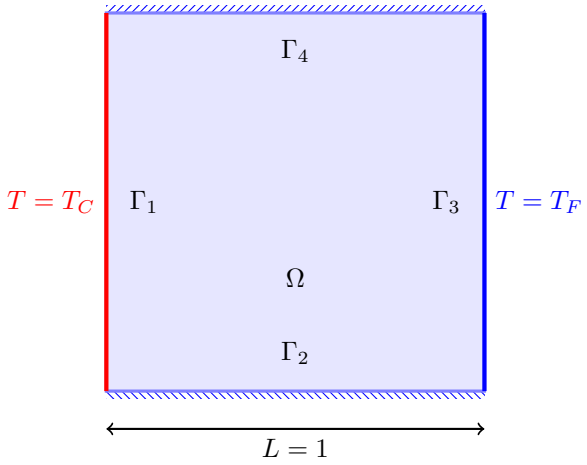

Figure 3.1: Géométrie du modèle

3.1.1 Paramètres dimensionnés

De par le caractère dimensionné des équations employées dans le code `Feel++`, nous avons jugé utile de détailler les paramètres physiques employés. Afin de simuler la même physique, ceux-ci ont été calculés à partir des nombres de Rayleigh et de Prandtl fournis dans le papier de référence.

Le tableau suivant reprend les paramètres utilisés, en fonction des valeurs adimensionnées. La longueur caractéristique a été fixée à $L_0 = 1m$, et l'écart de température caractéristique à $\Theta = 20K$.

Description	Paramètre dimensionné	Valeur	Unité
Longueur caractéristique	L_0	1.00	$m.s^{-1}$
Ecart de température caractéristique	Θ	20.00	K
Cœfficient d'expansion thermique	β	3.4112e-3	K^{-1}
Gravité	g	9.81	$m.s^{-2}$
Température de référence	T_0	273.15	K
Densité	ρ	1.000	$kg.m^{-3}$
Champ de vitesse	$\mathbf{u}$	$\frac{k}{L_0}\mathbf{u}^*$	$m.s^{-1}$
Champ de température	T	$T_0 + \Theta T^*$	K
Champ de pression	p	$\frac{k^2\rho}{L_0^2}p^*$	Pa
Viscosité cinématique	ν	$\sqrt{\frac{g\beta\Theta L_0^3 Pr}{Ra}}$	$m^2.s^{-1}$
Diffusivité thermique	κ	$\frac{\nu}{Pr}$	$m^2.s^{-1}$

Figure 3.2: Tableau récapitulatif des valeurs pour les grandeurs physiques associées au cas test

3.2 Présentation des résultats

Le nombre de Prandtl est fixé à $Pr = 0.71$ pour toutes les simulations du benchmark.

Les résultats numériques obtenus avec `Feel++` ou `FreeFem++` sont comparés à différents résultats de la littérature lorsque cela est possible.

	$\mathbf{u}_y$			T		
x	Feel++	FreeFem++	Ref[9]	Feel++	FreeFem++	Ref[9]
0.00	0.00442	0.00000	0.00000	1.00000	1.00000	1.00000
0.10	59.36030	59.31090	67.66134	0.59405	0.59387	0.61325
0.20	13.77730	13.75720	18.84612	0.47141	0.47148	0.47001
0.30	−0.31234	−0.30107	−0.25020	0.48124	0.48125	0.48141
0.40	−1.21166	−1.20562	−1.87750	0.49480	0.49477	0.49680
0.50	−0.00129	0.00000	−0.00526	0.50003	0.50000	0.49974
0.60	1.20633	1.20562	1.86782	0.50526	0.50523	0.50273
0.70	0.29863	0.30107	0.23980	0.51877	0.51875	0.51827
0.80	−13.77460	−13.75720	−18.86443	0.52856	0.52853	0.52992
0.90	−59.28730	−59.31090	−67.67705	0.40602	0.40613	0.38659
1.00	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Table 3.1: Sensibilité des données à la grille spatiale : convergence de la composante verticale de la vitesse ($\mathbf{u}_y$) et de la température (T) à mi-hauteur ($y = 0.5$). Grille 21×21.

	$\mathbf{u}_y$			T		
x	Feel++	FreeFem++	Ref[9]	Feel++	FreeFem++	Ref[9]
0.00	0.00056	0.00000	0.00000	1.00000	1.00000	1.00000
0.10	59.36740	59.34520	64.88415	0.59405	0.59404	0.62232
0.20	13.77820	13.77410	17.11070	0.47146	0.47144	0.49798
0.30	−0.30519	−0.30314	−0.52181	0.48124	0.48124	0.50583
0.40	−1.20856	−1.20812	−1.84403	0.49477	0.49477	0.51923
0.50	0.00006	0.00000	−0.00005	0.50000	0.50000	0.52500
0.60	1.20825	1.20812	1.84395	0.50523	0.50523	0.53078
0.70	0.30512	0.30314	0.52105	0.51877	0.51876	0.54416
0.80	−13.77670	−13.77410	−17.11324	0.52856	0.52856	0.52020
0.90	−59.33500	−59.34520	−64.88387	0.40596	0.40596	0.42784
1.00	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Table 3.2: Sensibilité des données à la grille spatiale : convergence de la composante verticale de la vitesse ($\mathbf{u}_y$) et de la température (T) à mi-hauteur ($y = 0.5$). Grille 41×41.

	$\mathbf{u}_y$			T		
x	Feel++	FreeFem++	Ref[9]	Feel++	FreeFem++	Ref[9]
0.00	0.00072	0.00000	0.00000	1.00000	1.00000	1.00000
0.10	59.34720	59.34800	63.74017	0.59405	0.59406	0.59636
0.20	13.77610	13.77540	16.49249	0.47144	0.47144	0.47007
0.30	−0.30378	−0.30324	−0.61296	0.48124	0.48124	0.48355
0.40	−1.20862	−1.20829	−1.83100	0.49477	0.49477	0.49787
0.50	−0.00001	0.00080	0.50000	0.50013	0.50000	0.50000
0.60	1.20862	1.20829	1.82940	0.50523	0.50523	0.50197
0.70	0.30346	0.30324	0.61041	0.51876	0.51876	0.51629
0.80	−13.77630	−13.77540	−16.49663	0.52856	0.52856	0.52977
0.90	−59.35360	−59.34800	−63.73526	0.40595	0.40595	0.40351
1.00	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Table 3.3: Sensibilité des données à la grille spatiale : convergence de la composante verticale de la vitesse ($\mathbf{u}_y$) et de la température (T) à mi-hauteur ($y = 0.5$). Grille 81×81.

3.2.1 Convergence selon le maillage

L'objectif des tableaux 3.1, 3.2 et 3.3 est de prouver la non-dépendance de la solution au regard de la grille employée. Les résultats obtenus sont comparés à ceux de [9].

Pour ce faire, nous utiliserons un nombre de Rayleigh fixé à $Ra = 10^5$ et nous mesurerons à mi-hauteur ($y = 0.5$) la composante verticale de la vitesse ainsi que la température pour plusieurs valeurs de x. Différentes grilles seront employées: 21×21, 41×41, 81×81. Par ailleurs, certains paramètres dimensionnés seront fixés: $\nu = 2,18.10^{-3}$ et $\kappa = 3,07.10{-}3$.

3.2.2 Distributions de vitesse

Composante verticale

Nous proposons ici d'étudier la distribution de la vitesse verticale à mi-hauteur ($y = 0.5$).

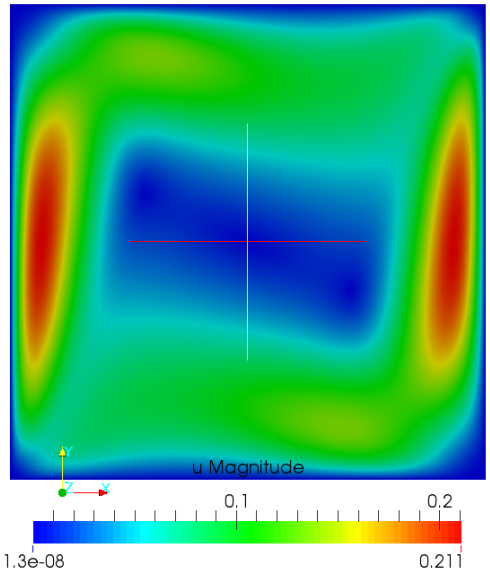

Figure 3.3: Norme de la composante verticale de la vitesse, obtenue à l'aide de `Feel++`.

Les mesures seront faites pour $10^3 \leq Ra \leq 10^6$ sur une grille homogène 101×101. Les résultats de `Feel++` et `FreeFem++` sont comparés dans la figure 3.4.

Nous mesurons ensuite le maximum de $\mathbf{u}_y$ sur le domaine ainsi que sa localisation. Nous pouvons ainsi le comparer avec deux résultats de [9], tableau 3.4.

	Feel++			FreeFem++	Ref[9]			Ref'[9]		
Ra	MAX	x	y	MAX	MAX	x	y	MAX	x	y
$1 \cdot 10^3$	3.701	0.18	0.49	3.701	3.692	0.19	0.49	3.690	0.18	0.48
$1 \cdot 10^4$	19.691	0.12	0.48	19.661	19.910	0.12	0.47	20.100	0.12	0.47
$1 \cdot 10^5$	68.637	0.07	0.51	68.521	70.810	0.07	0.49	70.830	0.07	0.49
$1 \cdot 10^6$	221.317	0.96	0.53	221.079	228.050	0.04	0.44	227.880	0.04	0.47
$1 \cdot 10^7$	0.000	0.00	0.00	704.557	0.000	0.00	0.00	0.000	0.00	0.00
$1 \cdot 10^8$	0.000	0.00	0.00	2,303.020	0.000	0.00	0.00	0.000	0.00	0.00

Table 3.4: Maximum de $\mathbf{u}_y$ et sa localisation sur le domaine en fonction de Ra

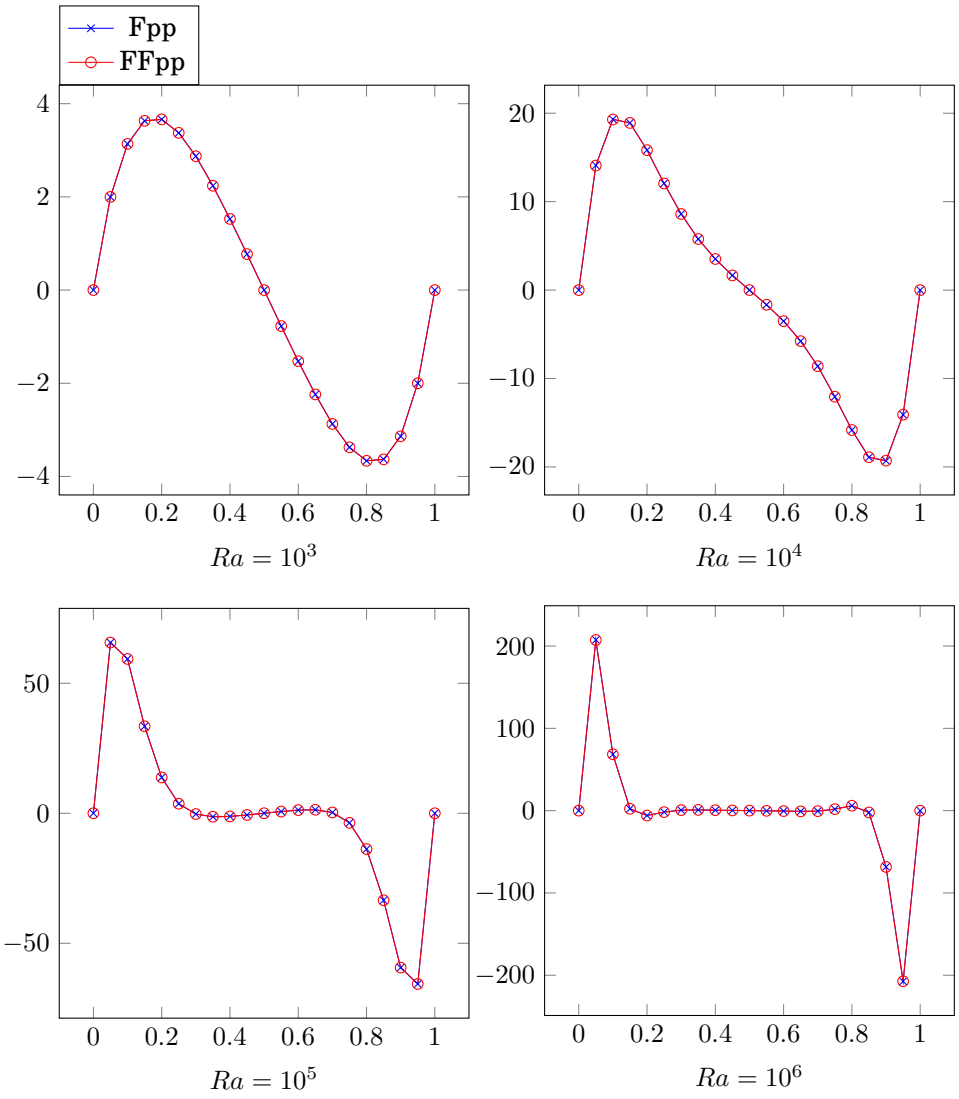

Figure 3.4: Représentation de la composante verticale $\mathbf{u}_y$ de la vitesse en fonction de x, pour $y = 0.5$ fixé. En abscisse: x, en ordonnée $\mathbf{u}_y$.

La localisation du maximum n'a pu être effectuée sous `Freefem++`. On remarque que les vitesses obtenues pour des Rayleigh élevés (10^7 et 10^8) sont surprenantes. Malheureusement, ces résultats n'ont pu être comparés, en raison du fait que `Feel++` ne gère pas les maillages non réguliers, tel celui proposé par `Freefem++` (figure 2.4). Un maillage raffiné sur tout le carré demanderait un calcul bien trop long.

Composante horizontale

On réitère la même démarche que précédemment mais avec, cette fois-ci, la vitesse horizontale à mi-largeur ($x = 0.5$). La figure 3.5 montre l'allure de la distribution entre les résultats obtenus par `Feel++` et `FreeFem++`, toujours sur une grille homogène 101×101. Les maxima de la composante horizontale de la vitesse sont donnés en tableau 3.5.

	Feel++			FreeFem++	Ref[9]			Ref'[9]		
Ra	MAX	x	y	MAX	MAX	x	y	MAX	x	y
$1 \cdot 10^3$	3.653	0.51	0.81	3.653	3.657	0.51	0.81	3.648	0.52	0.82
$1 \cdot 10^4$	16.219	0.52	0.18	16.206	16.140	0.49	0.81	15.968	0.50	0.82
$1 \cdot 10^5$	43.903	0.30	0.89	43.896	41.880	0.28	0.88	41.820	0.29	0.88
$1 \cdot 10^6$	128.747	0.81	0.06	128.829	114.300	0.16	0.93	114.530	0.17	0.93

Table 3.5: Maximum de $\mathbf{u}_x$ et sa localisation sur le domaine en fonction de Ra

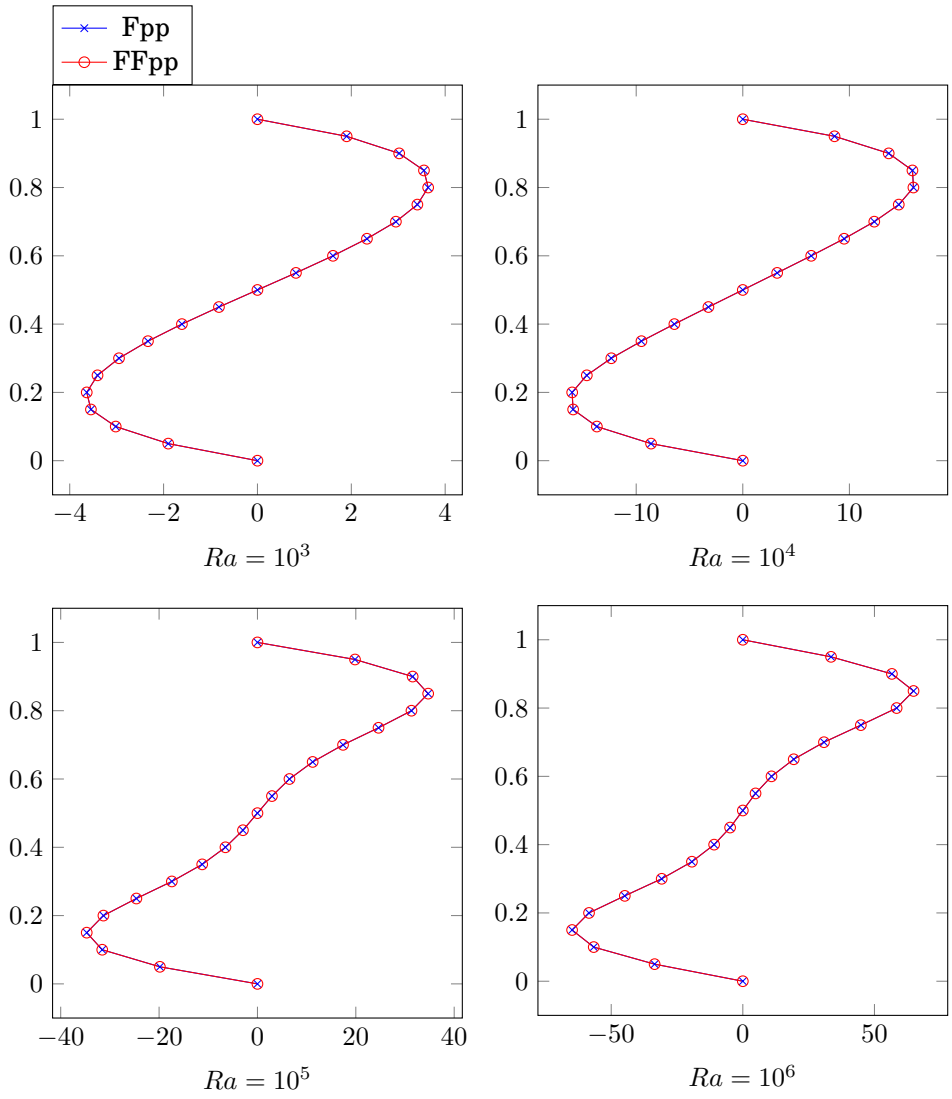

Figure 3.5: Représentation de la composante horizontale $\mathbf{u}_x$ de la vitesse pour $x = 0.5$ fixé. En abscisse: $\mathbf{u}_x$, en ordonnée: y.

Ra	Feel++	FreeFem++	Ref[9]	Ref'[9]	Ref[1]	Ref[6]	Ref[5]
$1 \cdot 10^3$	1.118	1.117	1.117	1.073	1.12	1.117	1.074
$1 \cdot 10^4$	2.246	2.233	2.254	2.155	2.24	2.243	2.084
$1 \cdot 10^5$	4.522	4.364	4.598	4.352	4.52	4.521	4.300
$1 \cdot 10^6$	8.839	8.823	8.976	8.632	8.80	8.806	8.743
$1 \cdot 10^7$	0.000	16.700	0.000	0.000	0.00	0.000	0.000
$1 \cdot 10^8$	0.000	31.760	0.000	0.000	0.00	0.000	0.000

Table 3.6: Nombre de Nusselt moyenné sur le bord chaud $x = 0$

3.2.3 Mesures du nombre de Nusselt

Dans cette section, nous allons étudier la distribution de la température le long des parois verticales gauche et droite. Cela peut être caractérisé par le nombre de Nusselt Nu donné par la relation suivante :

$$\mathrm{Nu}_{\mathrm{local}} = \pm \frac{\partial \Theta}{\partial x}|_{\mathrm{bords}}$$

Le nombre de Nusselt local a été calculé sur le bord froid, ainsi que sur le bord chaud, figure 3.7. Ces résultats proviennent de la même simulation que pour l'étude de vitesse, sur une grille de 101×101, avec un nombre de Rayleigh de 10^3 et 10^5.

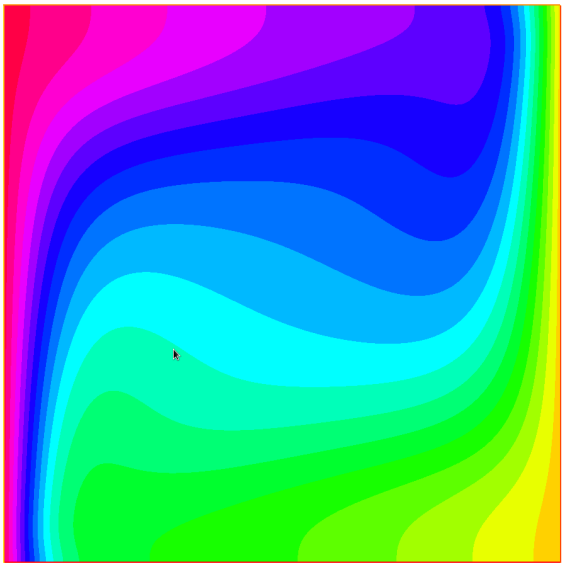

Figure 3.6: Profil de température, obtenu à l'aide de Freefem++, pour $Ra = 10^5$. On constate que l'air froid (en jaune) plus lourd, est entraîné vers le bas de la cavité par les forces de gravité tandis que l'air chaud (en bleu) monte. Pour un tel nombre de Rayleigh, il n'y a pas encore de couches limites.

Le tableau 3.6 compare le nombre de Nusselt moyenné sur le bord chaud, avec les différentes valeurs de référence.

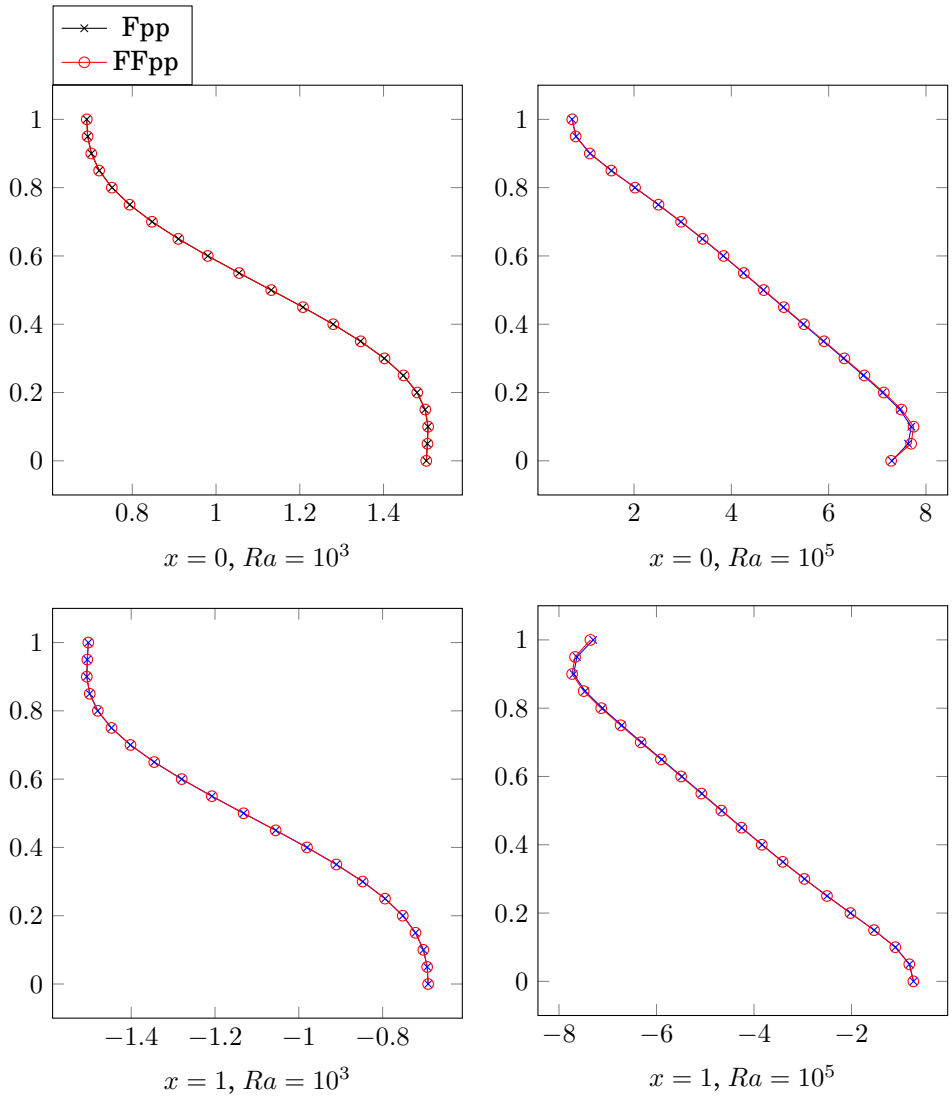

Figure 3.7: Nombre de Nusselt local, calculé pour des différents paramètres.
En abscisse: Nu, en ordonnée: y.

3.3 Bilan

Les courbes et tableaux présentés dans cette partie montrent des résultats extrêmement semblables entre `Freefem++` et `Feel++`, le tout étant plus ou moins corroboré par les données fournies par la littérature. La grande ressemblance entre les résultats des deux codes provient du fait que nous avons implémenté les équations en utilisant pratiquement les mêmes méthodes numériques sur un problème assez simple. Les petites différences que l'on peut observer avec les résultats des références peuvent peut-etre s'expliquer par une différence dans le choix des méthodes.

Ce premier cas-test est certes assez simple, mais les résultats obtenus sont très encourageants. Nous pouvons désormais nous attaquer à un problème plus compliqué: le problème de la salle machine.

CHAPTER 4

Benchmark 2: salle machine

On propose ici un cas test d'aérothermie qui a pour objectif d'établir des données comparatives entre deux codes éléments finis indépendants, permettant ainsi la vérification de leur capacité de résolution. Cette étude mettra en parallèle un code développé avec la librairie `Feel++` et un code développé avec la librairie `FreeFem++`. On rappelle que dans ce benchmark, on se place dans une plage de paramètres physiques excluant tout régime d'écoulement turbulent. Cet impératif provient du fait qu'à l'heure actuelle, nos codes n'incorporent aucun modèle de turbulence.

4.1 Description du modèle

4.1.1 Géométrie et maillage

Notre géométrie correspond à un rectangle $[0, 1.5] \times [0, 1]$ aux parois adiabatiques auxquelles nous avons ajouté une entrée de fluide `inlet`, une sortie de fluide `outlet` ainsi que deux sources internes de température (`core 1`,`core 2`). Le fluide rentre (via l'`inlet`) suivant un profil de Poiseuille et à une température donnée. La sortie du fluide s'effectuera à pression constante et sera modélisée par une "do-nothing" condition. Le `core 1` représente une source de chaleur à température constante, tandis que le `core 2` représente une source qui génère de la chaleur de manière proportionnelle au gradient de température à sa surface. Une vue isométrique de la géométrie du problème est donnée figure 4.1.

Note : Afin d'effectuer des comparaisons dans le cadre de ce benchmark, un maillage identique sera utilisé dans les deux codes, celui-ci étant généré à l'aide de `Gmsh` au format `.msh`.

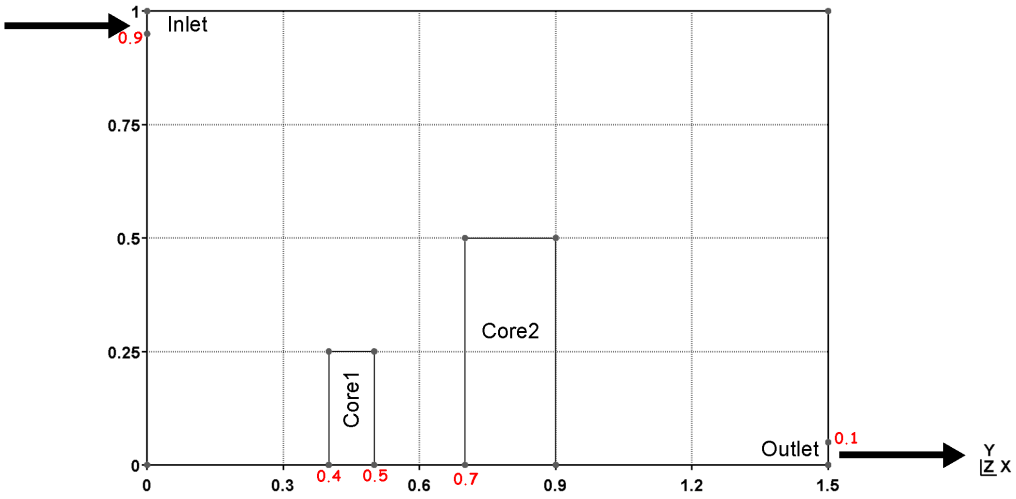

Figure 4.1: Géométrie du cas test, vue isométrique

4.1.2 Conditions aux limites

Commençons tout d'abord avec la condition initiale suivante:

$$\begin{cases} \mathbf{u} = 0 \text{ dans } \Omega, \text{ à } t = 0, \\ \nabla \cdot \mathbf{u_0} = 0, \\ T(\mathbf{x}, 0) = T_{ini} \quad \forall \mathbf{x} \end{cases} \tag{4.1}$$

Celle-ci traduit l'existence d'un champ de vitesse (dans la pratique nulle) animant notre fluide à $t = 0$. Différents choix d'initi alisation sont donnés dans la section 2.2.5.

Détaillons à présent les conditions aux bords de notre problème. Nous avons avec $\mathbf{n}$ le vecteur pointant vers l'extérieur et normal au bord considéré, (voir tableau ci-dessous)

Mots clés	Bords	Variable	Type	Expression
Wall	Γ_W	T	Adiabatique	$\mathbf{u} = 0$
Wall	Γ_W	$\mathbf{u}$	Dirichlet homogène	$\frac{\partial T}{\partial \mathbf{n}} = 0$
Outlet	Γ_O	$\mathbf{u}$ et p	Neumann	$(\frac{1}{\rho} p\mathbb{I} - \nu \nabla \mathbf{u}) \cdot \mathbf{n} = 0$
Outlet	Γ_0	T	Neumann	$\nabla T \cdot \mathbf{n} = 0$
Inlet	Γ_I	$\mathbf{u}$	Dirichlet	$\mathbf{u} = \mathbf{u}^{\mathbf{I}}_{\mathbf{pois}}$
Inlet	Γ_I	T	Dirichlet	$T = T_I$
core1	Γ_{c1}	T	Dirichlet	$T = T_{c1}$
core2	Γ_{c2}	T	Robin	$-\kappa(\nabla T \cdot \mathbf{n}) = \frac{\lambda_{c2}}{\rho C_p}(T - T_{c2})$

Figure 4.2: Tableau récapitulatif des conditions aux bords appliquées à notre cas test

On vérifiera sans problème que l'union disjointe de l'adhérence des différents sous-ensembles précédents est bien égale à l'adhérence de $\overline{\Gamma}$.

4.1.3 Paramètres

Le tableau ci-dessous exprime quant à lui, les grandeurs physiques (valeur et unité) associées à nos variables.

Nom	Paramètre	Valeur	Unité
Viscosité cinématique	ν	$0.500e^{-2}$	$m^2.s^{-1}$
Densité	ρ	1.000	$kg.m^{-3}$
Diffusivité thermique	κ	$0.500e^{-2}$	$m^2.s^{-1}$
Cœfficient d'expansion thermique	β	$3.4112e^{-3}$	K^{-1}
Chaleur spécifique	C_p	1000	$J.kg^{-1}.K^{-1}$
Température de référence	T_0	273.15	K
Gravité	g	9.81	$m.s^{-2}$
Température initiale	T_{ini}	$293,15$	K
Norme vitesse inlet	$\|\|\mathbf{u}^{\mathbf{I}}_{\mathbf{pois}}(t)\|\|$	$0.75\gamma(t)$	$m.s^{-1}$
Température inlet	$T_I(t)$	$293,15 - 5\gamma(t)$	K
Température `core 1`	$T_{c1}(t)$	$293,15 + 20\gamma(t)$	K
Température `core 2`	$T_{c2}(t)$	$293,15 + 40\gamma(t)$	K
Conductivité thermique `core 2`	λ_{c2}	5	$W.m^{-1}.K^{-1}$
Pas de temps	Δt	$0.50/0.10$	s
Temps final	T_f	60	s
Fonction de continuation	$\gamma(t)$	$1 - \exp(-0.5t)$	

Figure 4.3: Tableau récapitulatif des valeurs pour les grandeurs physiques associées à notre cas test

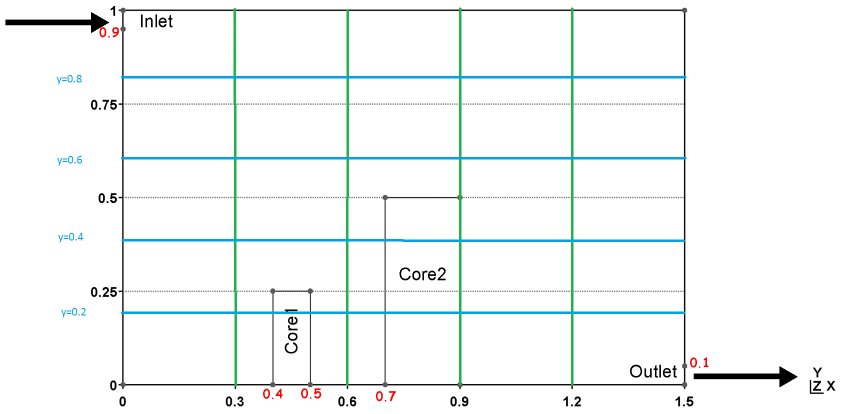

Figure 4.4: Localisation des lignes de mesures de températures

4.2 Présentation des premiers résultats

Nous nous proposons ici de comparer les résultats entre les deux codes avec un schéma temporel d'ordre 2 et un pas de temps de $0.5s$.

Une première comparaison grossière peut se faire sur l'allure des isovaleurs (figure 4.5). Les profils semblent alors assez proches. Nous souhaitons donc analyser la différence entre les deux codes de façon plus fine. Nous allons pour cela effectuer plusieurs séries de mesures sur différentes coupes horizontales et verticales. Les résultats de ces mesures sont présentés ci-contre.

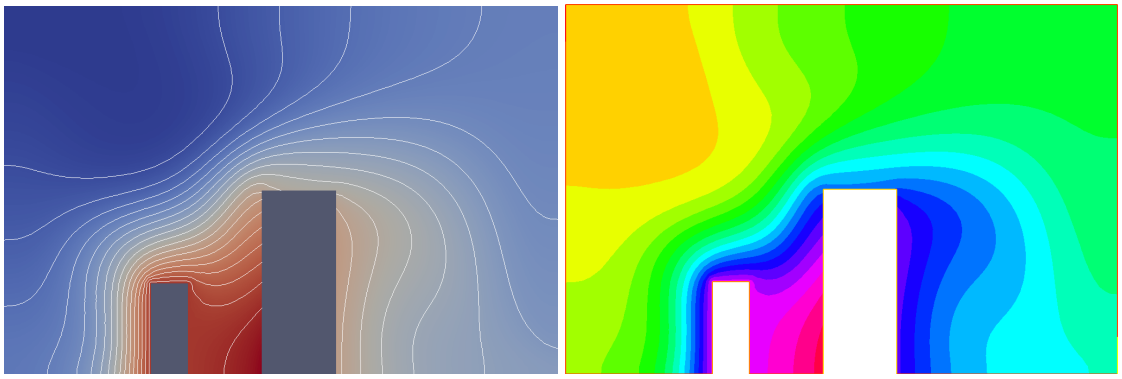

Figure 4.5: Comparaison des profils de température

4.2.1 Profils de températures

Profils verticaux

Les graphes de la figure 4.6 présentent différents profils de température verticaux pour des abscisses différentes. Nous avons choisi arbitrairement d'évaluer ces données à $t = 30s$.

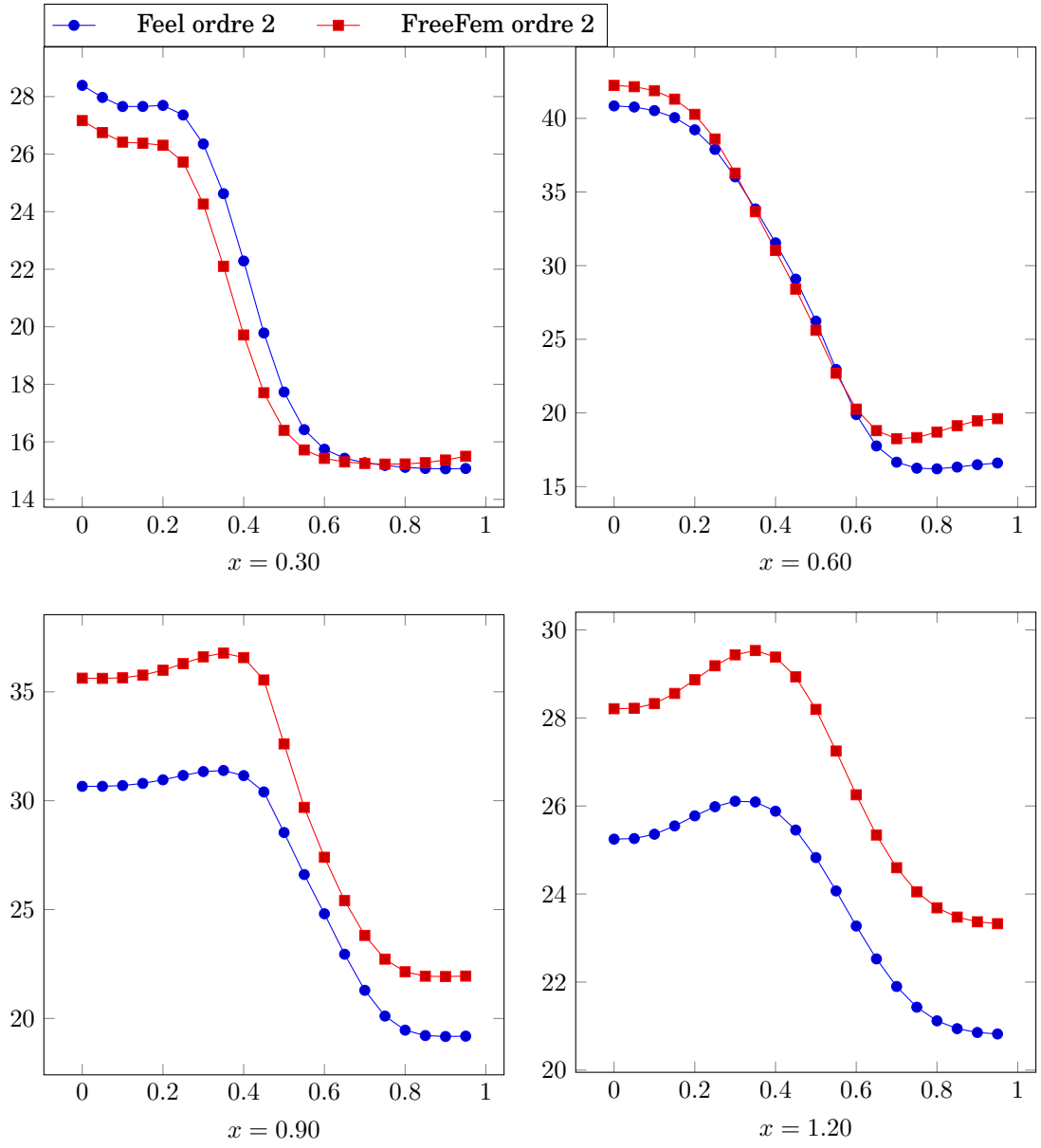

Figure 4.6: Profils de température verticaux à $t = 30s$ en fonction de y en différentes valeurs de x. En abscisse: y, en ordonnée la température T.

Profils horizontaux

De la même façon, les graphes de la figure 4.7 présentent les profiles de température horizontaux à différentes hauteurs.

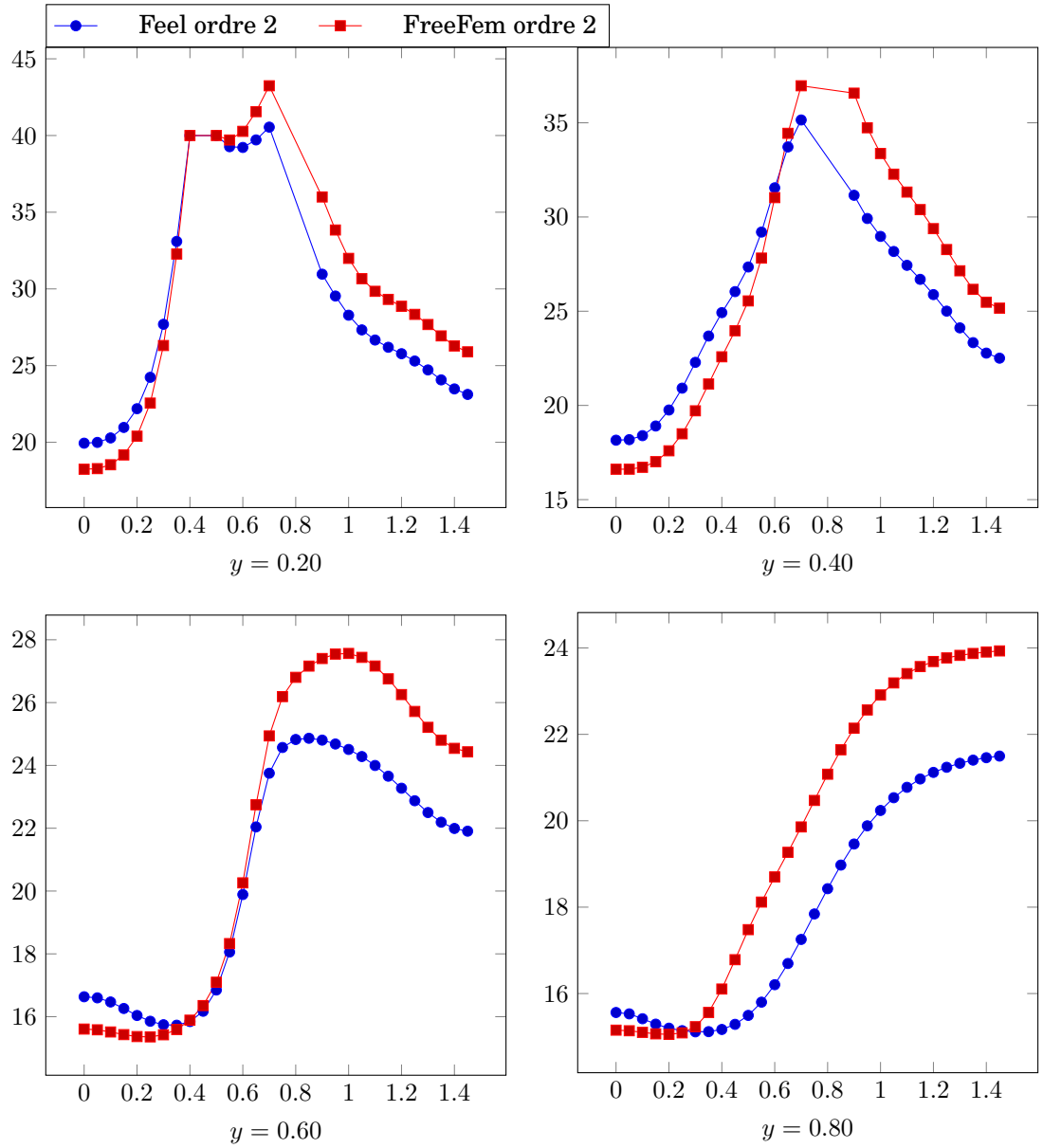

Figure 4.7: Profils de température horizontaux à $t = 30s$ en fonction de y en différentes valeurs de x.
En abscisse: x, en ordonnée: la température T

Conformément à la figure, les profils de températures sont assez similaires. Mais ceux-ci présentent un décalage de quelques degrés. La recherche de l'origine de cette différence nous amène à l'étude de la convergence en espace et en temps de nos codes.

4.3 Convergence temporelle

Nous allons tenter d'évaluer dans cette section l'impact du choix du schéma temporel sur les résultats obtenus. Pour cela, nous allons observer les variations de température en un point pour différents pas de temps et différents ordres d'approximation de la dérivée temporelle.

Cette étude a été réalisée dans les deux cas avec le même ensemble de paramètres et sur les mêmes points de mesure mais les résultats ne seront pas comparés entre les deux codes.

4.3.1 Sous FreeFem++

Afin d'étudier les convergences temporelle et spatiale, nous choisissons quatre points caractéristiques du domaine. Ces points sont:

- X1 = (0.6 ; 0,2)
- X2 = (0.6 ; 0.4)
- X3 = (0.9 ; 0.4)
- X4 = (1.2 ; 0.2)

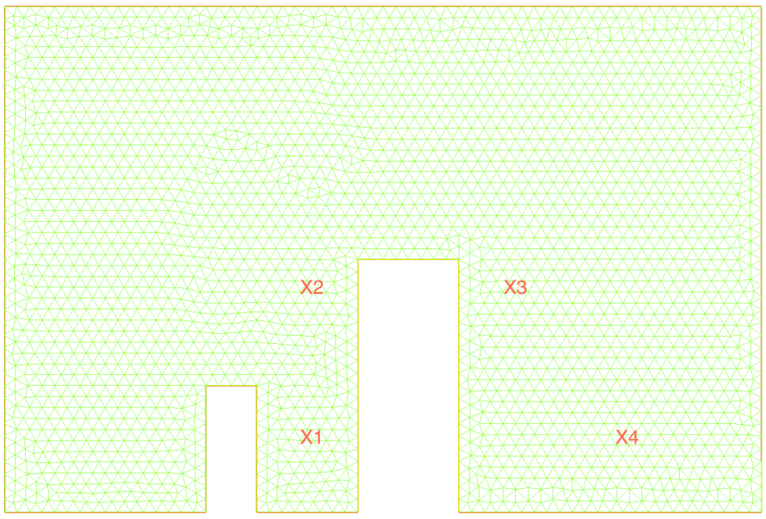

Figure 4.8: Quatre points sensibles du domaine.

Pour ces quatre points, nous avons tracé leurs évolutions thermiques en fonction du temps avec un schéma en temps d'ordre 1 avec un pas de temps de 0.10 et une seule itération de Newton. Nous obtenons :

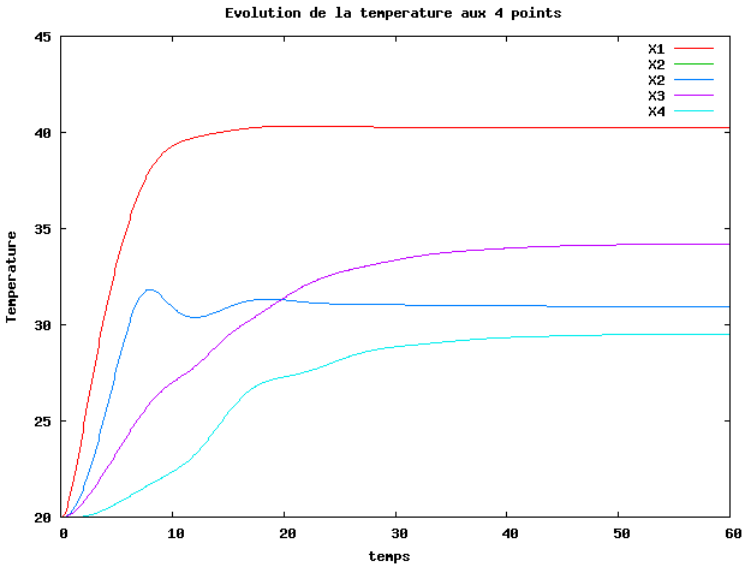

Figure 4.9: FreeFem++ : Evolution thermique des 4 points au cours du temps

Nous étudierons l'évolution thermique du point X2 car c'est la courbe présentant le plus de fluctuations. Par conséquent, il sera intéressant de le garder en point de base pour d'autres études par exemple pour la convergence en ordre pour les schémas temporels ou en maillage. En ce point, nous remarquons qu'avec un pas de temps plus faible, nous obtenons de meilleurs résultats à l'ordre 1.

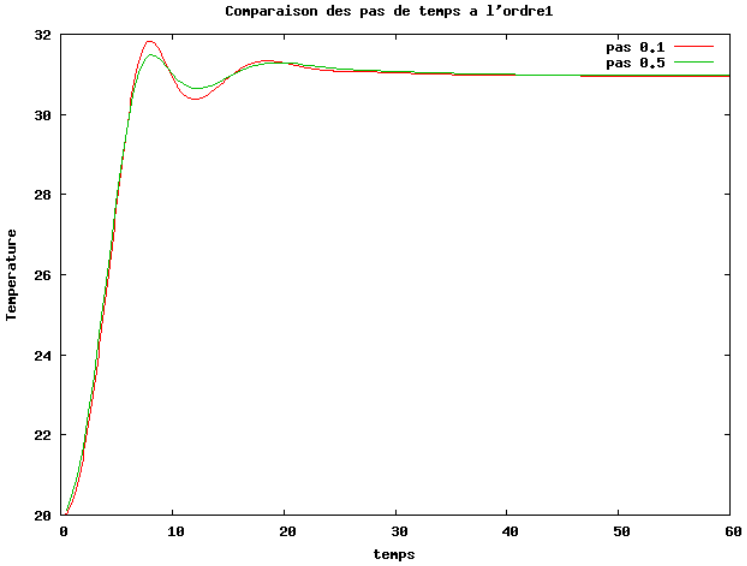

Figure 4.10: FreeFem++ : Evolution thermique de X2 pour des pas de temps 0.1 et 0.5 à l'ordre 1 au cours du temps

Alors qu'avec un schéma temporel à l'ordre 2, nous remarquons qu'avec différents pas de temps (0.50 et 0.10), nous n'obtenons pas de différences. Ainsi il pourrait être judicieux de garder le pas de temps à 0.50 pour la suite en ordre 2. De plus, nous remarquons que l'implémentation des conditions aux limites progressives (cf. 2.2.5) fonctionne correctement car il n'y a pas d'instabilité visible aux premières itérations.

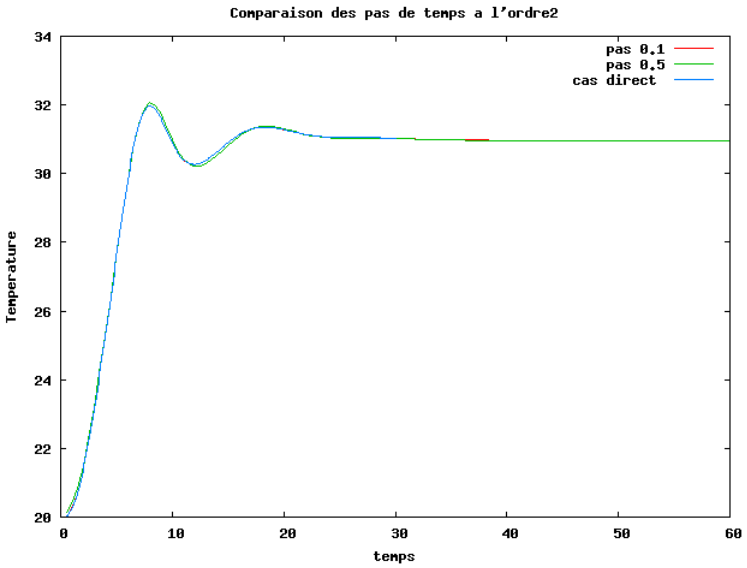

Figure 4.11: FreeFem++ : Evolution thermique de X2 à différents pas de temps.
Le cas direct correspond au cas où l'on passe directement à l'ordre 2 (cf. 2.2.4).

Traçons aussi, avec un pas de temps à 0.10, l'évolution thermique du point X2 par rapport aux trois ordres du schéma temporel.

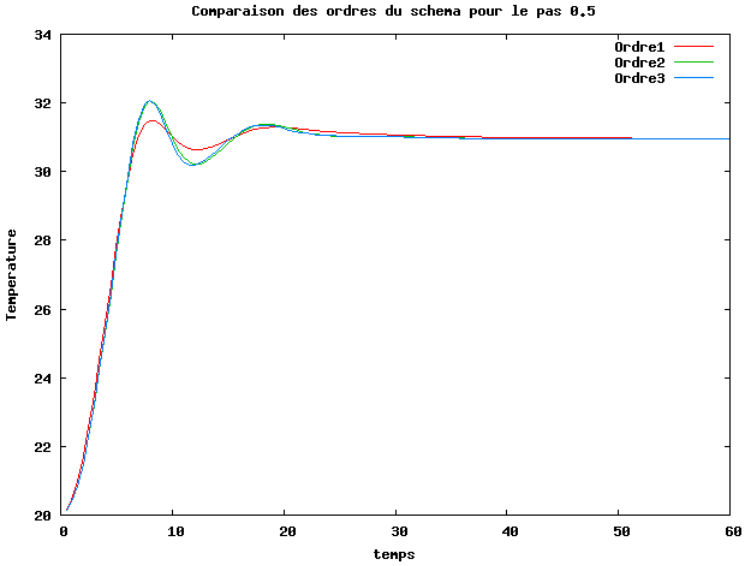

Figure 4.12: FreeFem++ : Evolution thermique de X2 par rapport aux 3 ordres du schéma temporel

Nous observons bien qu'avec un même pas de temps à 0.50, l'ordre 2 et 3 ne présentent presque pas de différence. Ceci est dû à la A-stabilité du schéma comme expliqué précédemment. Par conséquent, nous pouvons utiliser l'ordre 3 avec un pas de temps plus grand sans crainte d'instabilité ce qui permet d'accélérer considérablement les calculs.

4.3.2 Sous Feel++

Nous choisissons le même point $X2$ pour faire nos mesures. La figure 4.13 reprend les différentes mesures pour des ordres de schéma temporel et des pas de temps différents. Si les résultats pour l'ordre 1, avec un pas de temps $\delta t = 0.5$ semblent ne pas capter correctement toutes les variations, la différence entre les autres schémas reste négligeable.

Nous arrivons donc à la même conclusion que pour l'étude de convergence temporelle avec FreeFem++ : le choix du schéma temporel ne peut pas être la cause de telle divergence dans les résultats. Nous allons donc nous diriger vers une étude de convergence selon maillage dans la section suivante.

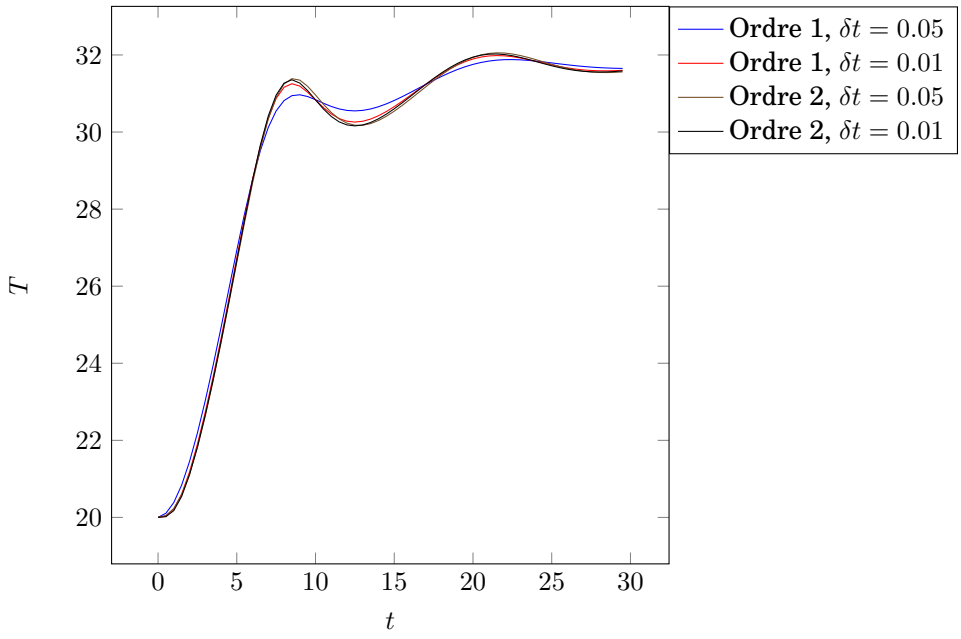

Figure 4.13: Évolution de la température en fonction du temps, pour différents schémas temporels.

4.4 Convergence spatiale

Les résultats ne semblant pas tellement dépendre du schéma temporel, nous décidons d'étudier la convergence des résultats selon le maillage choisi. Comme pour la convergence temporelle, les résultats sont présentés de façon indépendante pour les deux codes.

4.4.1 Sous Feel++

Le but de ces mesures est de vérifier la dépendance de nos résultats par rapport à la finesse du maillage choisi.

Nous effectuons pour cela des mesures de température en 4 points du domaine, en conservant exactement le même jeu de paramètres mais pour des finesses différentes.

Nous choisissons les paramètres suivants :

- Conditions limites telles que dans les tableaux 4.2 et 4.3
- Pas de temps : $0.1s$
- Ordre d'approximation de la dérivé temporelle : 2
- Espace élément finis : $\mathbb{P}_h^2\mathbb{P}_h^1\mathbb{P}_h^2$

Nous remarquons figure 4.14 des différences en fonction de la finesse du maillage et notamment en ce qui concerne l'état asymptotique qu'atteint le système. Il nous semble donc naturel d'approfondir cette étude de convergence à l'état stationnaire.

Nous choisissons pour cela d'effectuer deux coupes de température (se coupant en $(1.20, 0.6)$ qui semble être le point avec le plus de divergence) à l'état stationnaire, pour différentes finesses de maillage.

Nous constatons alors que les résultats semblent converger pour un maillage de longueur caractéristique $h = 0.010$, figure 4.14.

Une dernière série de mesure a été réalisé sur un maillage hétérogène, en gardant une finesse de $h = 0.010$ sur l'entrée d'air. La convergence en maillage ne semble pas plus rapide, figure 4.16.

Nous pouvons conclure que s'il existe bien des variations en fonction du maillage, elles restent tout à fait négligeables (en ce qui concerne Feel++) et ne peuvent pas expliquer les différences observées. Il ne parait donc pas nécessaire, avec Feel++, d'utiliser un maillage beaucoup plus fin que celui proposé initialement.

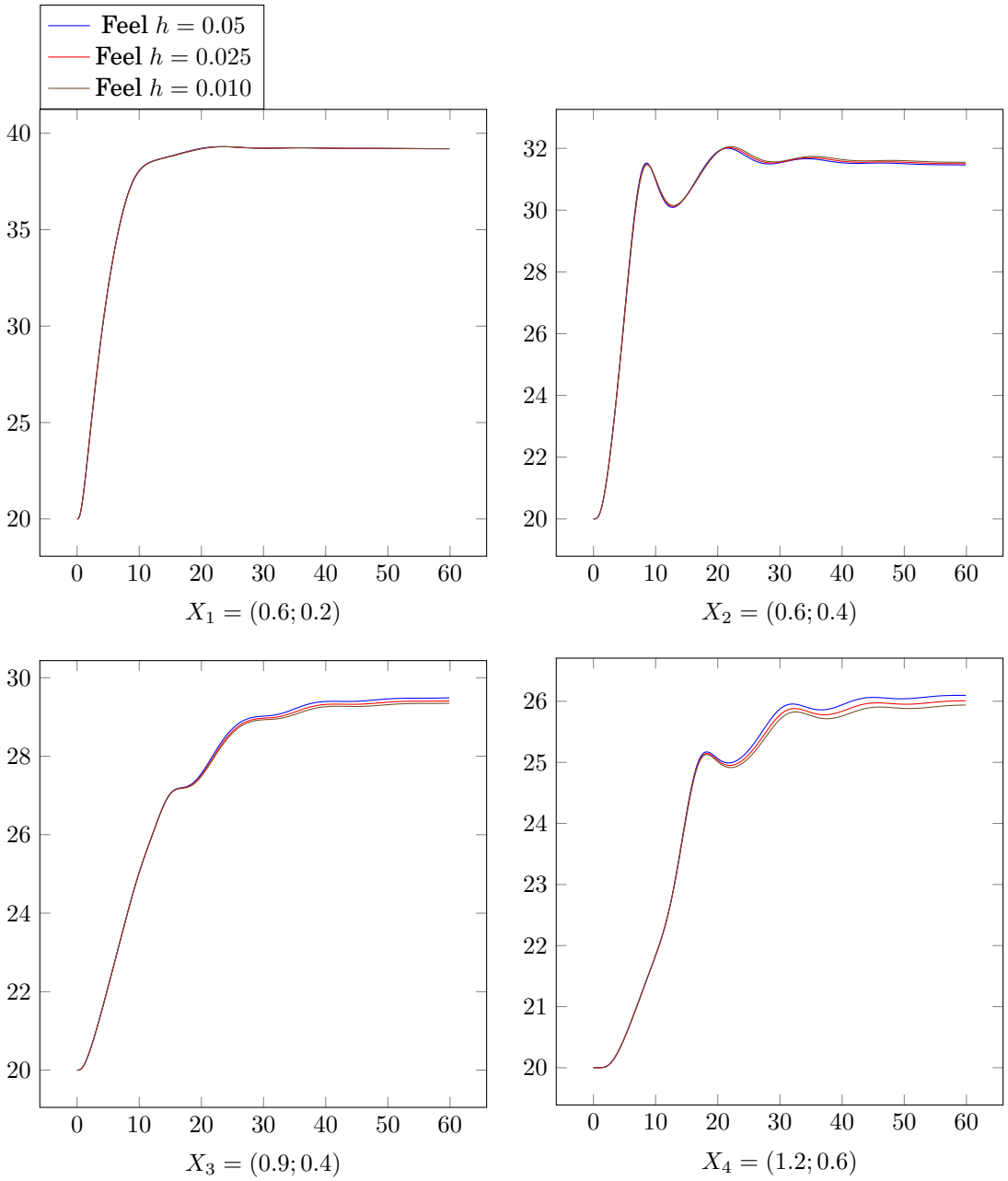

Figure 4.14: Convergence de la température selon le maillage aux quatre points choisis du domaine. En abscisse: le temps t, en ordonnée la température T.

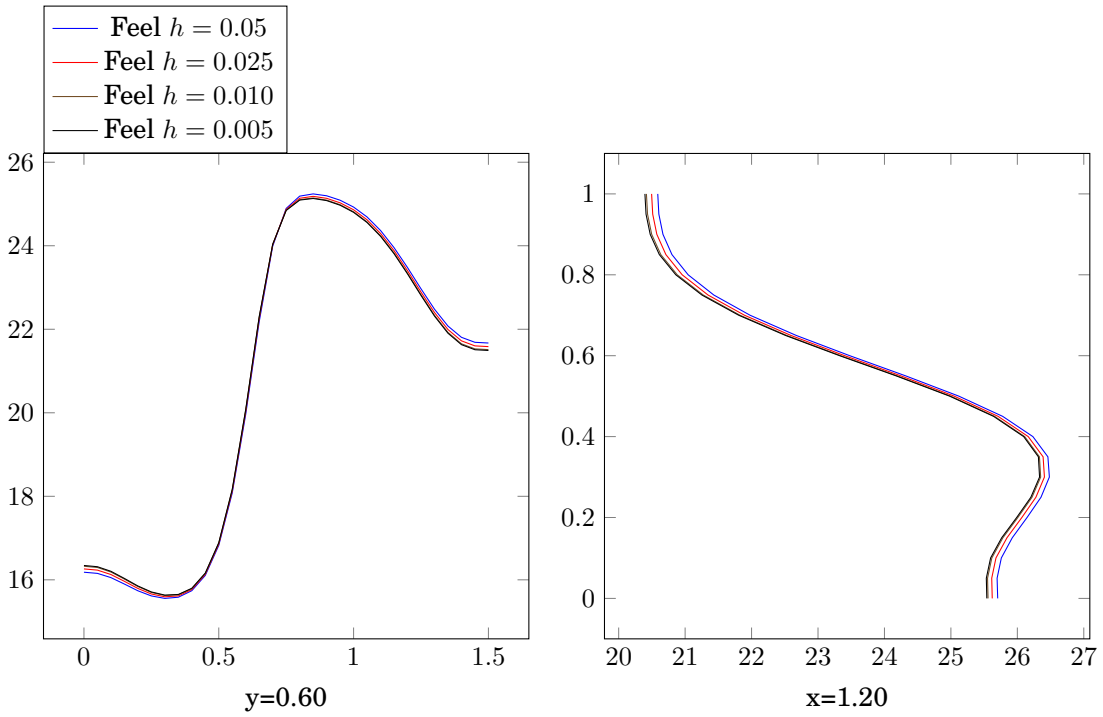

Figure 4.15: Coupe de température à l'état stationnaire, selon maillage, le long de lignes fixées.

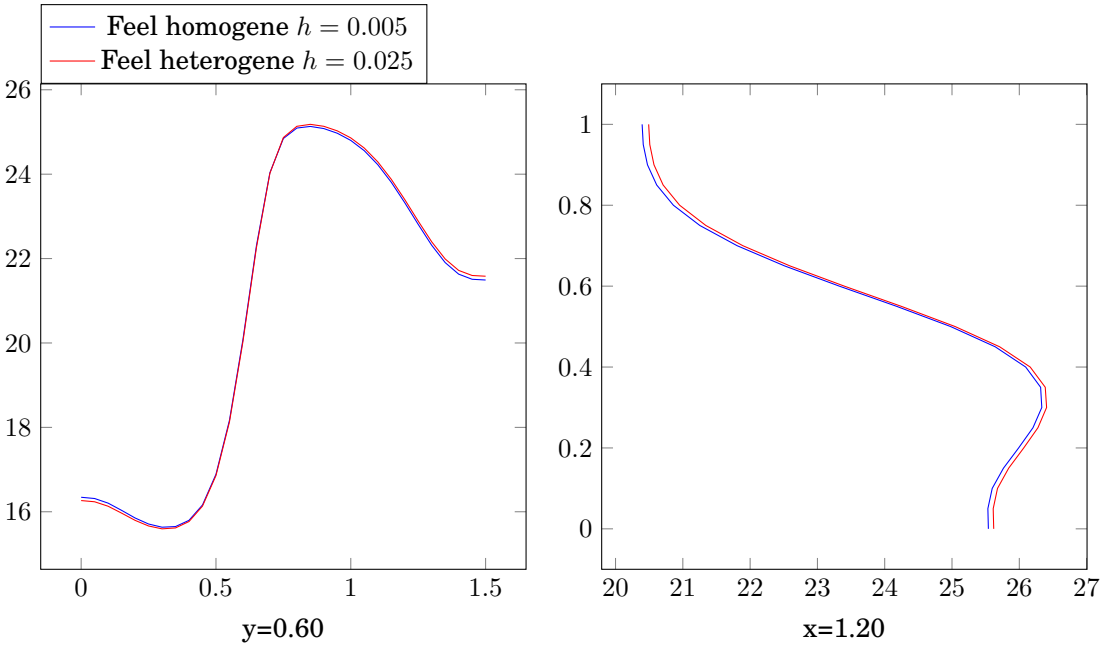

Figure 4.16: Coupe de température à l'état stationnaire, maillage hétérogène ou maillage de référence ($h = 0.005$), le long de lignes fixées

4.4.2 Sous FreeFem++

Maillages sous `FreeFem++`

Afin d'effectuer des mesures en convergence de maillage, nous devons générer des maillages de plus en plus raffinés. Ceux-ci se fait à l'aide du mailleur de FreeFem++ comme suit:

```
/* Creation du maillage de la salle machine sous FreeFem++*/
bool adaptation=0;
mesh Th;
int lin = 1, lout = 5, lcorps1 = 3, lcorps2 = 4, ldir = 2;

// Definition des "bords" permettant d'imposer les conditions de bord.
border Gamma1(t=1,0.9) {x=0; y=t; label=lin;}
border GammaWl(t=0.9,0) {x=0; y=t;label=ldir;}
border GammaO1(t=0,0.4) {x=t; y=0;label=ldir;}
border GammaO2(t=0,0.25) {x=0.4; y=t;label=lcorps1;}
border GammaO3(t=0.4,0.5) {x=t; y=0.25;label=lcorps1;}
border GammaO4(t=0.25,0) {x=0.5; y=t;label=lcorps1;}
border GammaO5(t=0.5,0.7) {x=t; y=0;label=ldir;}
border GammaO6(t=0,0.5) {x=0.7; y=t;label=lcorps2;}
border GammaO7(t=0.7,0.9) {x=t; y=0.5;label=lcorps2;}
border GammaO8(t=0.5,0) {x=0.9; y=t;label=lcorps2;}
border GammaO9(t=0.9,1.5) {x=t; y=0;label=ldir;}

border GammaO(t=0,0.1) {x=1.5; y=t;label=lout;}
border GammaWr(t=0.1,1) {x=1.5; y=t;label=ldir;}
border GammaPla(t=1.5,0) {x=t; y=1;label=ldir;}

// Nombre de points de discretisations sur un cote
int N=200;
real meshsize=1./N;

Th=buildmesh(Gamma1(N)+GammaWl(N)+GammaO1(N)
+GammaO2(N)+GammaO3(N)+GammaO4(N)+GammaO5(N)
+GammaO6(N)+GammaO7(N)+GammaO8(N)+GammaO9(N)
+GammaO(N)+GammaWr(N)+GammaPla(N));

Th=adaptmesh(Th,IsMetric=1,meshsize,nbvx=100000);

// plot(Th,wait=1);
savemesh(Th,"maillageraffine"+N+".msh");
```

Résultats

Nous traçons l'évolution temporelle de la température au point X2 défini précédemment en fonction de la finesse du maillage, figure 4.17. Le nombre N correspond au nombres de points de discrétisation sur la largeur de la salle machine.

Remarque: Le maillage initialement proposé pour le benchmark correspond à $N = 50$.

On constate que, pour ce qui est de `FreeFem++` tout du moins, la convergence en maillage n'est pas assurée pour $N = 50$. L'algorithme semble bien plus dépendre du maillage que ne peut en dépendre `Feel++`. Cette étude nous permet donc de proposer un nouveau maillage pour le benchmark, celui avec $h = 0.01$ (correspondant au cas $N = 100$) qu'on pourra utiliser pour les prochaines mesures.

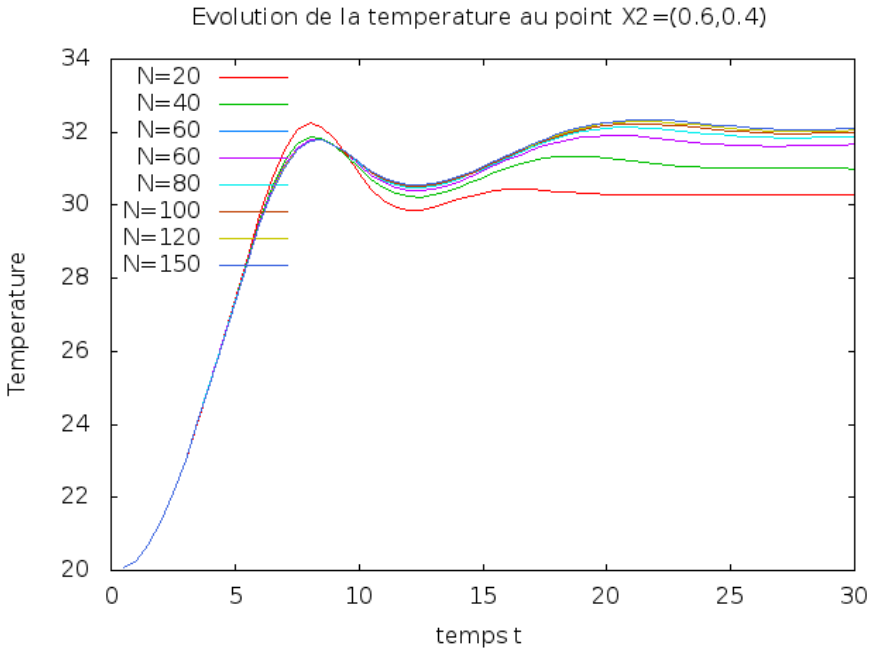

Figure 4.17: Convergence en maillage sous `FreeFem ++`

4.5 Présentation des nouvelles mesures

L'étude de convergence précédente nous a montré que le maillage initialement proposé par le benchmark n'était pas assez raffiné. Ainsi, on se propose d'effectuer de nouvelles mesures avec un nouveau maillage, comme défini dans la section précédente.

On constate que le raffinement du maillage sous `Freefem++` permet d'obtenir des résultats se rapprochant de ceux obtenus par `Feel++`. Néanmoins, des différences subsistent encore entre les résultats des deux codes, notamment au niveau du `core 2`, où des conditions de type Robin ont été implémentées, semblant être à l'origine de cette discorde qui nécessiterait d'être étudiée plus en détail...

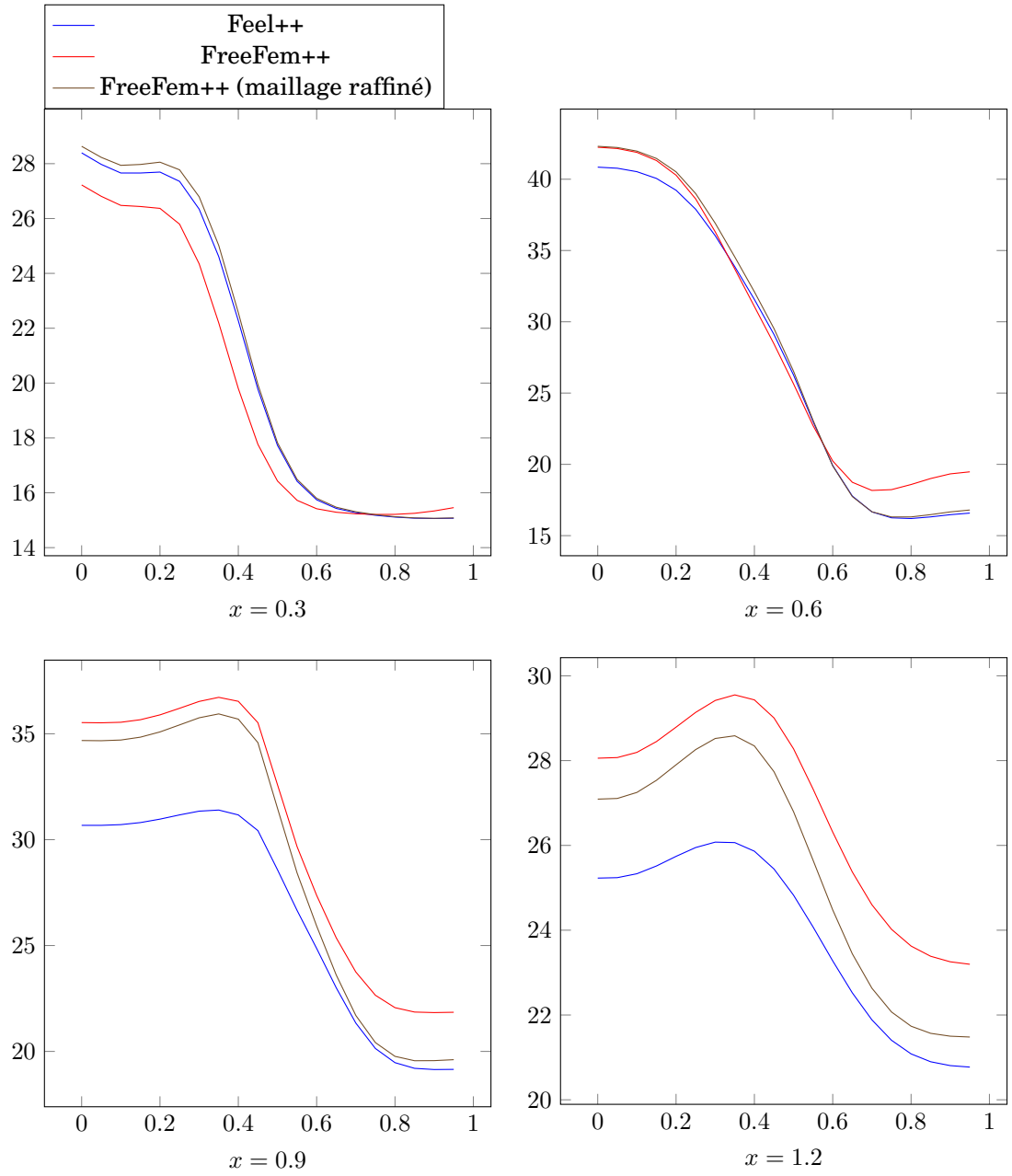

Figure 4.18: Profils de température verticaux à $t = 30s$ en fonction de y en différentes valeurs de x.
En abscisse: y, en ordonnée: la température T.

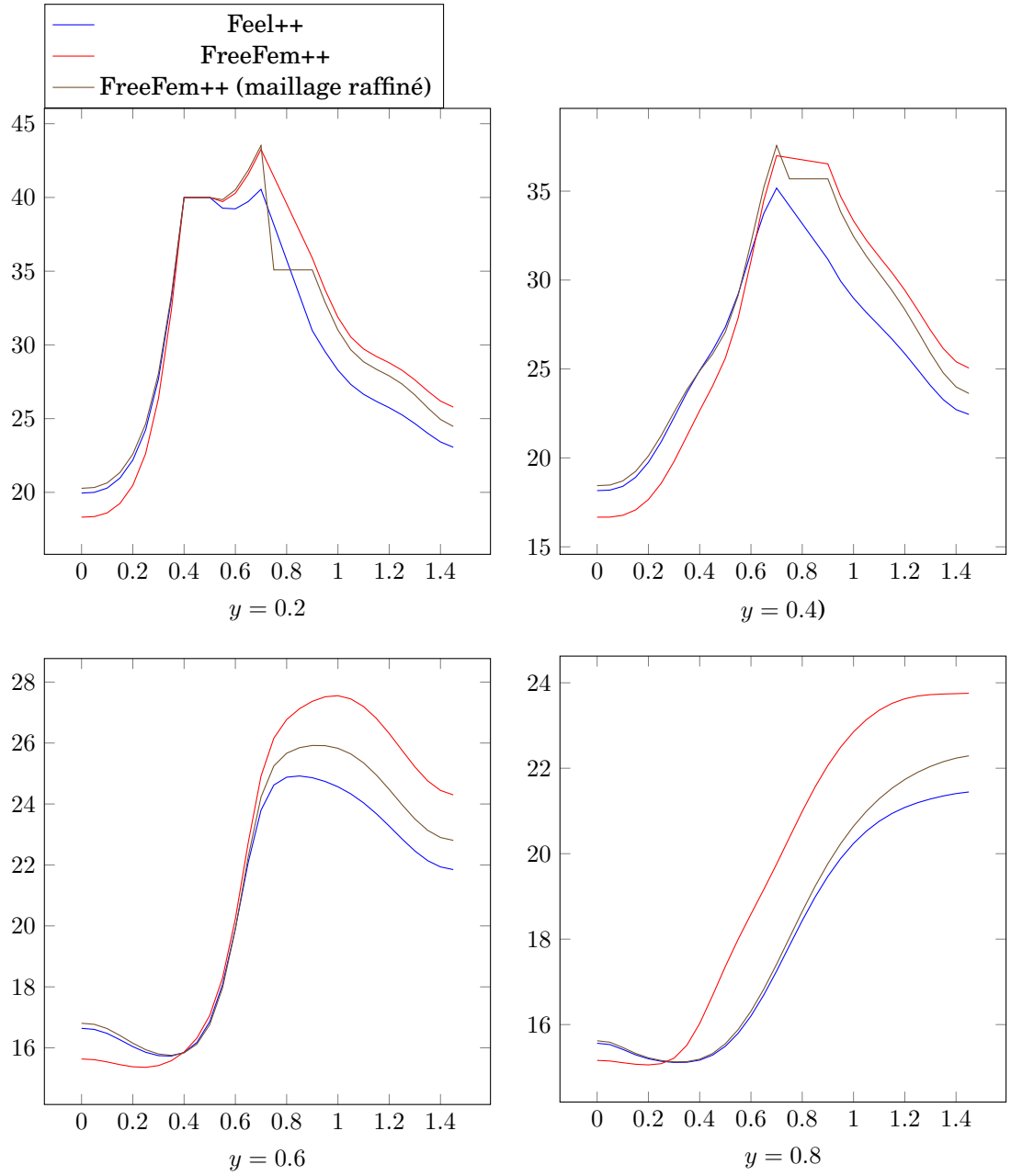

Figure 4.19: Profils de température horizontaux à $t = 30s$ en fonction de y en différentes valeurs de x.
En abscisse: x, en ordonnée: la température T

Conclusion

Bilan de l'étude L'étude sur la cavité chauffée par convection naturelle a montré, entre les codes `Feel++` et `FreeFem++`, des résultats très similaires (une différence de l'ordre de 10^{-5}). Ces résultats concordent plutôt bien avec ceux fournis par la littérature [9]. Ces résultats encourageants nous ont permis de passer à un problème plus complexe.

Dans le cas de la salle machine, l'analyse qualitative des isovaleurs de température nous montre des résultats convaincants de part et d'autre avec notamment la chaleur se confinant autour des deux "cores". Mais, avec une étude plus précise, les profils de température - bien que ressemblants - présentent un décalage de quelques degrés.

Le principal désaccord se situe autour du `core 2`. Cela pourrait s'expliquer par une implémentation différente des conditions aux limites (de type Robin) sur ce `core`. Une comparaison de la quantité de chaleur dégagée pourrait donc s'avérer intéressante. En effet, l'introduction de ces conditions constitue la principale différence entre le problème de la salle machine et celui de la cavité, pour lequel on a obtenu de bons résultats.

Les différences obtenues au niveau des résultats nous ont amenés à effectuer une étude plus approfondie. Les études de convergence en temps et en maillage ont révélé un comportement différent selon les deux codes : `FreeFem++` semble davantage dépendre de la qualité du maillage, tandis que `Feel++` dépend surtout de la discrétisation temporelle. A l'avenir, il nous faudrait comprendre avec plus de profondeur, les raisons de ces comportements différents.

Suite à cette étude, un maillage plus raffiné a été utilisé du coté de l'UPMC. Sous cette configuration, les résultats obtenus semblent mieux concorder avec ceux de Strasbourg, sauf encore autour du `core 2`. Une idée serait de repasser à des conditions plus simples sur ce `core`, pour confirmer que ce sont bien des conditions de type Robin qui posent problème.

Retour sur expérience Cette collaboration a été une expérience très enrichissante. D'un point de vue scientifique, ce projet nous a permis d'étudier en profondeur un problème – issu de l'industrie – et de prendre en main des logiciels dédiés, tels que `FreeFem++` et `Feel++`. Nous avons pu tester ces logiciels et comparer les différences qu'il peut y avoir dans l'implémentation d'un même problème. Ce projet nous a également permis de nous rendre compte de la difficulté de la simulation numérique, tant au niveau de la compréhension des équations issues de la mécanique qu'au niveau de l'implémentation de celles-ci. Mais

surtout, nous avons constaté l'importance de l'étape cruciale qu'est la validation des résultats et donc de l'intérêt d'un benchmark qui constitue, avec l'étude expérimentale, l'une des rares manières de valider un code. Sans ce benchmark, nous n'aurions en effet très certainement jamais remis en question nos premiers résultats, qui s'avéraient être erronés; celui-ci nous a donc permis de trouver l'origine de nos erreurs et de les corriger. Une bonne communication est également très importante: on a pu ainsi mieux cerner la cause des différences qui subsistent encore entre nos résultats, bien que de nombreuses pistes restent encore à explorer.

Il nous a donc été donné la chance de mieux ressentir pendant quelques mois la vie d'un chercheur et par la même occasion, de comprendre que c'est une communication permanente qui permet de progresser.

Nous remercions l'ensemble des équipes encadrantes, qui ont permis de mettre en œuvre cette collaboration et nous ont donné l'opportunité de participer à cette belle expérience.

APPENDIX A

Temps de calcul

Nous calculons ici le temps de calcul en fonction de la taille du maillage. Pour ce faire, nous utiliserons les maillages utilisés lors de l'étude de la convergence en maillage. La courbe ci-dessous représente le temps de calcul en fonction du nombre de sommets du maillage.

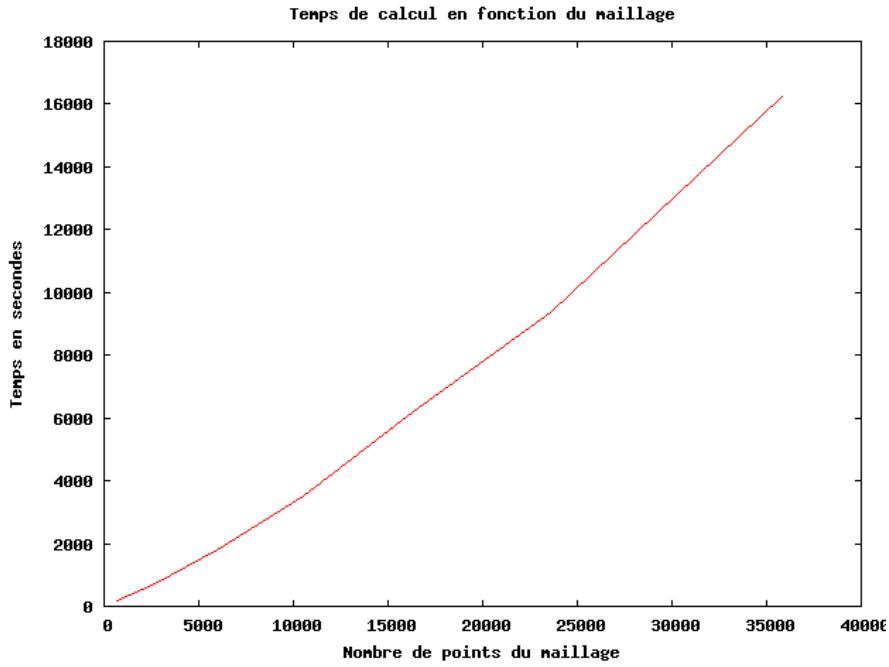

Figure A.1: Temps de calcul en fonction du nombre de sommets du maillage sous `FreeFem++`

On remarque que, avec l'algorithme sous `FreeFem++`, la dépendance du temps de calcul par rapport au nombre de sommets du maillage est quasi-linéaire.

Bibliography

[1] DE VAHL DAVIS, D. Natural convection of air in a square cavity: A bench mark solution. *Int. J. Numer. Meth. Fluids 3* (1983), 249–264.

[2] DOBRZYNSKI, C., PIRONNEAU, O., AND FREY, P. Numerical coupling for air flow computations in complex architectures. *European Congress on Computational Methods in Applied Sciences and Engineering, ECCOMAS 2004, Jyvaskyla, Finland* (2004).

[3] HECHT, F. New developments in freefem++. *J. Numer. Math. 20*, 3-4 (2012).

[4] HECHT, F., AULIAC, S., PIRONNEAU, O., MORICE, J., LE HYARIC, A., AND OHTSUKA, K. *FreeFem++*. 2012. http://www.freefem.org/ff++/.

[5] MANZARI, M. T. An explicit finite element algorithm for convective heat transfer problems. *Int. J. Numer. Meth. Fluids 9* (1999), 860–877.

[6] MASSAROTTI, N., NITHIARASU, P., AND ZIENKIEWICZ, O. C. Characteristic-based-split (cbs), algorithm for incompressible flow problems with heat transfer. *Int. J. Numer. Meth. Fluids 8* (1998), 969–990.

[7] PIRONNEAU, O. Cfd on unstructured meshes. http://www.ann.jussieu.fr/hecht/, 2012. (supports pédagogiques au module NSF03 – *Numerical Methods for Fluid Mechanics*).

[8] PRUD'HOMME, C. Feel manual: A library for finite and spectral element methods in 1d, 2d and 3d. http://www.feelpp.org/.

[9] WAN, D. C., PATNAIK, B. S. V., AND WEI, G. W. A new benchmark quality solution for the buoyancy-driven cavity by discrete singular convolution. *Numerical Heat Transfer 40* (2001), 199–228.