UNIVERSITÉ DE MONTRÉAL

MÉTHODES DE RUNGE-KUTTA IMPLICITES POUR LA RÉSOLUTION D'ÉQUATIONS DE CONVECTION-DIFFUSION EN RÉGIME INSTATIONNAIRE

ÉRIC ST-AMAND
DÉPARTEMENT DE GÉNIE MÉCANIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE MÉCANIQUE)
AVRIL 2009

Your file Votre référence
ISBN: 978-0-494-53926-2
Our file Notre référence
ISBN: 978-0-494-53926-2

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

MÉTHODES DE RUNGE-KUTTA IMPLICITES POUR LA RÉSOLUTION D'ÉQUATIONS DE CONVECTION-DIFFUSION EN RÉGIME INSTATIONNAIRE

présenté par : ST-AMAND Éric

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. PELLETIER Dominique, Ph.D., président

M. GARON André, Ph.D., membre et directeur de recherche

Mme. FARINAS Marie-Isabelle, Ph.D., membre et codirectrice de recherche

M. URQUIZA José, Doctorat, membre et codirecteur de recherche

M. REGGIO Marcelo, Ph.D., membre

REMERCIEMENTS

Mes remerciements vont d'abord à mon directeur de recherche, M. André Garon, pour m'avoir offert la possibilité de faire cette recherche en m'apportant un soutient adéquat à tous les niveaux, que ce soit financier, académique ou émotif.

Je remercie également José Urquiza et Marie-Isabelle Farinas qui, principalement avant de quitter l'École Polytechnique, m'ont offert un soutien académique. Je remercie par la même occasion Jérôme Vétel et Eddy Petro pour toutes les discussions intéressantes et l'aide académique qu'ils ont apportés durant ma recherche.

Je remercie ensuite Dominique Pelletier et Marcelo Reggio pour avoir accepté d'être respectivement le président du jury et membre du jury pour la soutenance de mon mémoire.

Finalement, j'envoie un dernier remerciement, mais non le moindre, à ma copine, Geneviève, qui m'a offert des encouragements et un support d'une constance irréprochable.

Merci à tous.

RÉSUMÉ

Ce présent mémoire porte sur l'implémentation des méthodes de Runge-Kutta implicites (RKI) dans un code d'éléments finis pour résoudre des phénomènes en régime transitoire. Étant donné l'augmentation constante de la puissance de calcul des ordinateurs modernes, il devient réaliste de chercher une meilleure précision que celle obtenue avec les méthodes d'ordre un - typiquement le schéma d'Euler. L'objectif visé est donc d'étudier ces méthodes en termes de leur précision et de leur stabilité numérique.

Ce mémoire se décompose en deux parties, la première partie étant consacrée à l'étude des méthodes RKI et la seconde portant sur le passage d'un programme d'éléments finis existant, séquentiel, à une version parallèle. Ces deux parties sont connexes dans le sens où les RKI causeront une augmentation du temps de calcul et le traitement parallèle, une réduction. En appliquant les deux, le gain net sera un gain en précision de calcul.

Dans la première partie, les RKI sont décrites en détails et des propriétés de stabilité sont énoncées pour certaines d'entres elles (A-stabilité, L-stabilité, etc.). Comme il y a une infinité de RKI, un choix judicieux doit être fait lors de l'implémentation de ces méthodes afin d'optimiser la précision et la stabilité du schéma par rapport au temps de calcul requis. Les méthodes possédant la propriété L-stable constituent un choix intéressant puisqu'elles ne sont pas sensibles au fait qu'une équation soit dite raide. Typiquement, un écoulement dont le nombre de Reynolds local varie beaucoup d'un élément à l'autre sera raide. Certaines méthodes RKI ont la propriété que leur dernière évaluation intermédiaire équivaut à la solution au pas de temps suivant, ce qui évite d'avoir à calculer explicitement cette solution. Ceci constitue un avantage algorithmique et une réduction de la charge de calculs à effectuer.

On montre par une expérience numérique, faite sur le logiciel MATLAB, que les méthodes de RKI présentent des comportements plus stables que ceux d'autres méthodes d'ordres élevées, notamment les formules aux différences finies arrières (BDF), un groupe de méthodes largement utilisées dans la littérature. Il est vrai que l'utilisation des RKI nécessite plus de calculs, mais la stabilité de ces méthodes reste un avantage indéniable.

Dans la seconde partie, on parallélise un programme d'éléments finis, codé en C++ et développé à l'École Polytechnique de Montréal. Entre autres, il faut assurer une gestion efficace et sans conflit des structures de données. Le maillage original doit être partitionné en zones, chacune d'elles étant traitées par un processeur différent. Le partitionnement est réalisé à l'aide de la librairie METIS. Puis, le calcul proprement dit est parallélisé en suivant le standard OpenMP pour le traitement parallèle sur les multiprocesseurs à mémoire partagée. Les calculs sont effectués sur un IBM P690 REGATTA possédant 16 processeurs et 64 Go de mémoire partagée. On démontre les avantages du traitement parallèle sont démontrés en mesurant le temps de calcul pour obtenir la solution aux équations de Navier-Stokes en régime stationnaire en utilisant 2, 4, 8 et 16 processeurs. Le gain de vitesse enregistré est de presque 8 lorsqu'on exécute le programme sur 16 processeurs. Dans le cas étudié ici, on démontre de façon empirique que ce gain de vitesse pourrait être amélioré avec un accès à plus de processeurs.

Ce mémoire se veut une base théorique. Il permet de souligner les principales notions concernant les méthodes de Runge-Kutta implicites. On commence tout juste à s'intéresser de façon pratique à ces méthodes étant donné la complexité des calculs numériques impliqués versus la capacité des ressources informatiques. À partir des notions étudiées ici, il devient possible de réaliser l'implémentation des RKI dans un code parallèle, tâche qui n'a pas été fait lors de cette recherche.

ABSTRACT

This master thesis deals with the implementation of implicit Runge-Kutta methods (IRK) into a finite element code for the computation of unsteady phenomena. Since modern computers have a constant increase in computation power, it becomes realistic to look for more accurate methods than those of first order - typically the Euler scheme. The targeted objective is to study these methods in term of their accuracy and numerical stability.

This master thesis is split in two parts, the first one being for the study of IRK methods and the second, for the transformation of an existing, sequential, finite element program into a parallel one. These two parts are connected if one consider that the implementation of IRK methods will raise the computation time required to solve a problem, whereas the use of parallel computation will lower that time. By applying both the IRK methods and the parallel computation, the net gain will be an accuracy gain.

In the first part, the IRK are described in details and their stability properties are stated for some of them (A-stability, L-stability, etc.). Since there are many ways to design an IRK method, a wise choice must be made when one wants to implement these methods so that the resulting scheme accuracy and numerical stability, relative to computation time, can be optimized. Methods whose stability properties include L-stability constitute an interesting choice since they are not sensitive to stiff equations. Typically, a fluid flow in which the cell Reynolds number varies widely amongst elements will be stiff. Some IRK methods also have their last stage equal to the next time-step solution, avoiding the need to compute that solution explicitly. This represents an algorithmic advantage and a reduction in computation load.

It is shown with a numerical experiment, done on MATLAB, that the IRK methods have better stability behaviours than other high order methods, one group of them being the well known backward difference formulae (BDF), which are widely

used in the literature. It is true that using IRK requires more computation, but the inherent stability of these methods is an incomparable advantage.

In the second part, a finite element program, coded in C++ and developed at École Polytechnique de Montréal, is transformed into a parallel version. Among other things, there is a need to ensure an efficient and conflict-free management of data structures. The original mesh must be partitioned into zones, each of these zones being treated by a different processor. The partitioning is done with the help of the METIS library. Then, the actual computation is parallelized by following the OpenMP standard for parallel computation over shared memory multiprocessors. The computations are made on an IBM P690 REGATTA possessing 16 processors and 64 Gb of shared memory. Parallel programming advantages are made clear by measuring the computation time required to find the solution to the steady-state Navier-Stokes equations with 2, 4, 8 and 16 processors. The recorded speedup is close to 8 on 16 processors. In the case studied here, it is also shown empirically that this speedup could be enhanced by using even more processors.

This master thesis represents a theoretical basis that underline notions which are already known, but which, since the complexity of involved computation relative to computers hardware, are just beginning to feel interesting in a practical way. From these notions, it becomes possible to implement RKI methods into a parallel code, a work that has not been done explicitly in this research.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

- 3D : tridimensionnel
- ARK : méthode de Runge-Kutta additive
- BDF : Formules aux différences arrières (« Backward differentiation formula »)
- CONT : Stabilisation de l'équation de continuité
- CPU : Central Processing Unit, équivalent à processeur.
- ÉDA : Équation différentielle algébrique
- ÉDO : Équation différentielle ordinaire
- ÉDP : Équation différentielle partielle
- ESDIRK : Explicit Singly Diagonal Implicit Runge-Kutta
- IBM : International Business Machines Corporation
- INRIA : Institut national de recherche en informatique et en automatique
- IRM : Imagerie par résonnance magnétique
- METIS : Librairie de partitionnement de maillage
- MPI : Message Passing Interface
- N-S : Navier-Stokes
- OpenMP : Standard de traitement parallèle en mémoire partagée
- P1/P1 : Éléments triangulaires, linéaires en vitesse et linéaires en pression, pour discrétiser l'équation de Navier-Stokes
- P2/P1 : Éléments triangulaires, quadratiques en vitesse et linéaires en pression
- PARDISO : Solveur direct
- PIV : Vélocimétrie par image de particules (« Particle Image Velocimetry »)
- PSPG : Pressure-Stabilization Petrov-Galerkin
- Q2/Q1 : Éléments quadrangulaires, quadratiques en vitesse et linéaire en pression

- RadauIIA1 : méthode de classe Radau de type IIA et d'ordre 1. Les autres classes suivent cette appellation.
- RK : méthode de Runge-Kutta
- RKDI : méthode de Runge-Kutta diagonalement implicite
- RKE : méthode de Runge-Kutta explicite
- RKEDIU : méthode de Runge-Kutta explicite, diagonalement implicite et à valeur uniforme sur la diagonale
- RKI : méthode de Runge-Kutta implicite
- RK-IMEX : méthode de Runge-Kutta implicite-explicite
- SUPG : Streamline-Upwind Petrov-Galerkin

LISTE DES SYMBOLES

- a_{ij} : Coefficient de la ligne i, colonne j, de la matrice a
- a_{ij}, b_i, c_i : Coefficients d'une méthode numérique multi-étages
- B(p), C(η), D(ζ) : Hypothèses simplificatrices
- C , α : Constante
- E_n : Erreur absolue au temps n
- Fr : Nombre de Froude
- g_i : Solution (interne) de l'étage i d'une méthode multi-étages
- G(z) : Fonction stabilité d'une méthode numérique
- h , h_e : Taille des éléments
- $H^1(\Omega)$: Le premier espace de Hilbert sur Ω
- i : Nombre imaginaire i = $\sqrt{-1}$
- Im(z) : Partie imaginaire du nombre complexe z
- k : Nombre de pas d'une méthode multi-pas
- $L^2(\Omega)$: L'espace des fonctions de carrés sommables sur Ω
- p : Nombre de processeurs
- p , q : Ordre de convergence
- Pe : Nombre de Péclet
- R(T) : Résidu de l'équation dépendante de T
- Re : Nombre de Reynolds
- Re(z) : Partie réelle du nombre complexe z
- s : Nombre d'étages d'une méthode numérique multi-étages
- S(p) : Gain de vitesse
- ST : Terme de stabilisation

- t : Temps
- t_p : Temps de calcul d'un code parallèle
- t_s : Temps de calcul d'un code séquentiel
- T : Maillage
- T^{ex} : Solution exacte de T
- T^N : Solution discrétisée de T (approximative)
- V : Espace des fonctions tests
- w : Fonction dans V
- y'(t) : Dérivée de y par rapport à t
- y^n : Solution y au temps n
- $*y^n$: Solution exacte au temps n
- z : Paramètre pour le calcul de τ_{CONT}
- z : Paramètre pouvant être complexe
- Γ : Bord de Ω
- Δt , h : Pas de temps
- ε , δ : Coefficient de perturbation
- ζ^j : Valeur à la puissance j de la solution y_j
- λ : Paramètre, valeur propre
- ξ : Gain d'amplification de la solution
- $\rho(\zeta)$, $\sigma(\zeta)$: Polynômes générateurs d'une méthode multi-pas
- τ_{SUPG} : Facteur d'échelle de temps pour la méthode SUPG
- τ_{PSPG} : Facteur d'échelle de temps pour la méthode PSPG
- τ_{CONT} : Facteur d'échelle de temps pour la méthode CONT
- Ψ : Fonction d'interpolation

- Ω , Ω_{adim} : Sous-espace de $\mathbb{R}^3$ (adim : adimensionnel)
- Ω_{1D} , $\Omega_{1D,adim}$: Sous-espace de $\mathbb{R}$ (adim : adimensionnel)
- $\varnothing$: Espace vide ou nul
- $\mathbb{R}^3$: Espace des réels en trois dimensions
- $\mathbb{C}^-$: Partie négative de l'espace des nombres complexes
- $\|\cdot\|_{0,\Omega}$: Norme sur $L^2(\Omega)$
- $\|\cdot\|_{1,\Omega}$: Norme sur $H^1(\Omega)$
- ∇ : Opérateur Nabla
- ∞ : L'infini

INTRODUCTION

Les méthodes d'intégration numérique ont vu le jour bien avant l'ordinateur avec lequel nous les mettons en œuvre maintenant. C'est au début du siècle dernier que des mathématiciens comme C. Runge et M.W. Kutta mettent au point la famille de méthodes maintenant bien connue que sont les méthodes de Runge-Kutta. Des mathématiciens les précèdent, comme L. Euler, et d'autres les succèdent, comme J. Crank et P. Nicholson. Ils nous ont légué une boîte formidable d'outils qui demeurent très pertinents dans le domaine des simulations numériques.

Ce mémoire traite de deux sujets complémentaires. La première partie concerne l'application des méthodes de Runge-Kutta implicites (RKI) pour la résolution d'équations différentielles partielles représentant des phénomènes transitoires. La deuxième partie concerne la création d'une version parallèle d'un programme d'éléments finis créé à l'École Polytechnique de Montréal.

Mes travaux de recherche tenteront donc de répondre à la question suivante : Comment améliorer la précision du calcul par élément finis tout en réduisant au minimum le temps de calcul ?

Les hypothèses qui sont avancées pour répondre à cette question sont :

- les méthodes de Runge-Kutta implicites seront à la fois plus précises que les méthodes actuellement implémentées dans le logiciel (méthode d'Euler Implicite, méthode de Gear et méthode du trapèze) et plus stables numériquement que les méthodes de Runge-Kutta explicites ou que les méthodes de différentiation arrière (BDF);
- le traitement parallèle du code offrira un gain de vitesse substantiel.

De fait, si les méthodes de Runge-Kutta implicites sont complexes, il ne va sans dire que leur précision et leur stabilité sont très intéressantes. En les implémentant, on allonge sans doute le temps de calcul, mais le traitement parallèle viendra réduire ce

dernier. On peut avancer comme idée que le gain net sera un gain en précision. Autrement dit, le logiciel sera plus efficace qu'avant ces modifications. Cette idée s'appuie évidemment sur un accès fiable à une bonne quantité de ressources informatiques, ressources qui sont de plus en plus accessibles à l'École Polytechnique de Montréal.

Le présent mémoire consigne donc les diverses expériences réalisées au cours de ma recherche. Les chapitres seront divisés comme suit. Le premier chapitre présente une revue bibliographique de certaines études portant sur les RKI. Ceci constitue une base sur laquelle la présente étude est fondée. Ensuite, quelques notions théoriques seront révisées afin de bien situer le lecteur. Ces notions se retrouvent au chapitre 2 et concernent principalement la base de l'analyse d'erreur et la méthode des éléments finis. Le chapitre 3 enchaîne avec une brève discussion sur l'équation de transport de chaleur. Le chapitre 4 s'attaque au cœur du sujet de la première partie, soit la théorie décrivant les méthodes de RKI. Il sera question ici d'étudier la théorie concernant la stabilité et la précision de ces méthodes. Finalement, le chapitre 5 présente les résultats obtenus lors de simulations tests sur le logiciel MATLAB.

Le chapitre 6 marque le début de la deuxième partie avec une seconde revue bibliographique, cette fois-ci détaillant les recherches en traitement parallèle. Le chapitre 7 aborde les équations de Navier-Stokes, équations qui serviront de banc d'essai pour le code parallèle. Le chapitre suivant inclut à la fois la théorie pour évaluer les performances d'un programme ainsi que les modifications qui ont été apportées au programme séquentiel original. Le chapitre termine avec la présentation des résultats obtenus sur le gain de vitesse dû au traitement parallèle du programme d'éléments finis. Pour chronométrer le programme, les équations de Navier-Stokes (3D) en régime permanent seront résolues dans une géométrie complexe représentant un vrai cas d'ingénierie.

PARTIE I - LES SCHÉMAS DE RUNGE-KUTTA IMPLICITES

CHAPITRE 1 - REVUE BIBLIOGRAPHIQUE

La première partie de ce mémoire présente quelques bases théoriques suivies d'une étude de performance et de stabilité des méthodes de Runge-Kutta implicites. Bien que ces méthodes existent depuis plusieurs décennies, leur utilisation est peu fréquente en raison de l'importance des ressources informatiques qu'elles nécessitent.

Ce premier chapitre procède à une revue bibliographique des connaissances sur les schémas numériques d'intégration en temps des équations différentielles ordinaires et subséquemment des équations aux dérivées partielles.

Il existe de nombreuses méthodes d'intégration en temps pour résoudre les équations de Navier-Stokes en régime transitoire (et d'autres équations d'évolution) : le schéma d'Euler (explicite et implicite, ou de façon générale, les θ-schémas), les formules aux différences arrières (« backward differential formula », BDF), les méthodes de Runge-Kutta (explicites et implicites) et d'autres encore. Pour définir ces schémas de façon simple, nous utiliserons un modèle scalaire, aussi appelé équation test de Dahlquist (Hairer, Nørsett, & Wanner, 1993). Elle est fréquemment utilisée pour l'analyse des schémas d'intégration en temps (Dettmer & Peric, 2003; Kanevsky, Carpenter, Gottlieb & Hesthaven, 2007).

Soit l'équation différentielle ordinaire avec une solution initiale à t = 0 :

$$y'(t) = f(t,y) = \lambda y(t) \quad t > 0, \lambda < 0 \quad y(0) = y_0 \tag{1.1}$$

La solution analytique de l'équation (1.1) est :

$$y(t) = y_0 e^{\lambda t} \tag{1.2}$$

Il est à noter que λ peut prendre une valeur complexe.

Les θ-schémas

Les θ-schémas (Euler généralisé) sont sans doute la famille de schémas la plus utilisée (Bochev, Gunzburger & Lehoucq, 2007; Bochev, Gunzburger & Shadid, 2004; Dettmer & Peric, 2003; Lacasse, Garon & Pelletier, 2007; Rang, 2008). Il s'agit de remplacer la dérivée de y(t) par une différence finie d'ordre 1 :

$$y'(t) = \frac{y^{n+1} - y^n}{\Delta t} = f(t,y) = \lambda\left[\theta y^{n+1} + (1-\theta)y^n\right] \tag{1.3}$$

où l'indice n définit une étape de temps et Δt, le pas de temps. Dans cette équation, si θ = 0, on obtient le schéma d'Euler explicite, si θ = 1, le schéma d'Euler implicite et si θ = ½, le schéma de Crank-Nicholson. Le schéma de Crank-Nicholson est aussi bien connu sous le nom de règle du trapèze (Gresho, Sani & Engelman, 1998). Le schéma d'intégration se trouve en isolant y^{n+1} :

$$y^{n+1} = y^n\left(\frac{1+\Delta t\lambda(1-\theta)}{1-\Delta t\lambda\theta}\right) \tag{1.4}$$

Les formules aux différences arrières

Le schéma de Gear, qui est en fait la BDF d'ordre deux, est différent du θ-schéma. Les BDF d'ordre deux et plus entrent dans une catégorie que l'on appelle les méthodes linéaires à pas multiples, car il faut conserver en mémoire plusieurs étapes de temps. Le schéma de Gear utilise une différence finie d'ordre 2 pour la dérivée de y(t) :

$$y'(t) = \frac{3y^{n+1} - 4y^n + y^{n-1}}{2\Delta t} = f(t,y) = \lambda y^{n+1} \tag{1.5}$$

Le schéma de Gear est équivalent à ce que d'autres auteurs appellent le « Three-level fully implicit » (Fletcher, 1991), que l'on pourrait traduire par « Schéma implicite à trois niveaux ». On retrouve les BDF dans certains ouvrages (Hairer,

Nørsett & Wanner 1993; Gresho, Sani, & Engelman, 1998). De façon générale, les BDF s'expriment ainsi :

$$\frac{1}{\Delta t}\sum_{j=1}^{k}\frac{1}{j}\Delta^j y_{n+1} = f_{n+1} \tag{1.6}$$

avec

$$\Delta^j y_{n+1} = \Delta^{j-1} y_{n+1} - \Delta^{j-1} y_n \quad , \quad \Delta^0 y_n = y_n$$

Pour les schémas BDF implicites, le paramètre k indique le nombre de solutions antérieures qui sont utilisées (et donc conservées en mémoire) pour le calcul de la dérivée. Les BDF sont appelées « méthode à k-pas » ou encore « méthode multi-pas », mais ces dénominations se rapportent à la même définition. Si k = 1, on retrouve la méthode à un pas, qui est Euler implicite (Hairer, Nørsett & Wanner 1993).

Méthodes de Runge-Kutta

Les méthodes de Runge-Kutta font partie des méthodes à pas unique puisque seule la solution au temps précédent est nécessaire pour calculer la solution au pas de temps suivant. Cependant, pour chaque pas de temps, plusieurs évaluations intermédiaires de l'équation à intégrer sont nécessaires. Ces évaluations intermédiaires sont représentées par des étages. Si une méthode de Runge-Kutta nécessite trois évaluations intermédiaires, alors cette méthode possède trois étages.

Une méthode de Runge-Kutta possédant s étages est définie de la façon suivante :

$$y'(t) = \frac{y^{n+1} - y^n}{\Delta t} = \sum_{i=1}^{s} b_i f_i(t, y) \tag{1.7a}$$

avec

$$f_i(t,y) = f(t^n + c_i\Delta t, y^n + \Delta t\sum_{j=1}^{s} a_{ij}f_j(t,y)) \quad , \; i = 1, 2, \ldots, s \tag{1.7b}$$

Depuis les travaux de Butcher durant les années soixante (Butcher, 1964, 1975), il est coutume d'utiliser la notation (le tableau) de Butcher pour décrire les méthodes de Runge-Kutta :

$$\begin{array}{c|cccccc} c_1 & a_{11} & a_{12} & a_{13} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & & & \\ c_3 & a_{31} & & a_{33} & & \\ \vdots & \vdots & & & \ddots & \\ c_s & a_{s1} & & & & a_{ss} \\ \hline & b_1 & b_2 & b_3 & \cdots & b_s \end{array} = \begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ & \mathbf{b} \end{array} \tag{1.8}$$

Selon les valeurs des coefficients a_{ij}, nous distinguons trois cas différents :

- Si $a_{ij} = 0 \;\; \forall \;\; i \leq j$, alors la méthode est explicite (RKE);
- Si $a_{ij} = 0 \;\; \forall \;\; i < j$ et au moins un $a_{ii} \neq 0$, alors la méthode est diagonalement implicite (RKDI);
- Sinon, la méthode est implicite (RKI).

Le défi le plus apparent dans le cas où la méthode est implicite est qu'un système d'équations linéaires devra être résolu à chaque pas de temps. Pour les méthodes explicites, il s'agit simplement d'une série d'évaluations de la fonction à intégrer.

Autres méthodes

Ce mémoire s'arrête aux méthodes de Runge-Kutta implicites et tente d'en illustrer les propriétés et de démontrer l'intérêt de les utiliser en pratique pour la résolution d'équations d'évolution. Cependant, d'autres familles de méthodes de Runge-Kutta, généralement plus récentes, peuvent présenter un certain avantage lors de situations particulières. Cette section présente quelques unes de ces familles.

Une de ces familles se nomme Runge-Kutta Implicite-Explicite (RK-IMEX). À l'origine, la combinaison de méthodes implicites et explicites a été utilisée pour les méthodes multi-pas (les BDF), puis aux méthodes de Runge-Kutta (Ascher, Ruuth, & Spiteri, 1997). Cette approche est justifiée pour une équation aux dérivées partielles dont les échelles de temps de convection et de diffusion sont différentes d'une région du domaine de calcul à l'autre. Ainsi, pour l'équation de convection-diffusion (transitoire), Ascher, Ruuth & Spiteri (1997) utilisent une méthode de Runge-Kutta explicite pour intégrer le terme de convection et une méthode implicite pour le terme de diffusion.

Les méthodes de Runge-Kutta additives (RKA) sont une autre famille de Runge-Kutta où plusieurs schémas existants sont additionnés afin de profiter de leurs avantages individuels (Kennedy & Carpenter, 2003). On identifie le nombre de schémas combinés en ajoutant un indice à l'abréviation RKA. Par exemple, RKA_2 désigne une méthode de Runge-Kutta additive combinant deux schémas. Les RK-IMEX sont des RKA_2.

Généralement (Kennedy & Carpenter, 2003), les RKA_2 combinent une méthode de Runge-Kutta explicite (RKE) traditionnelle avec une méthode de Runge-Kutta dite explicite, diagonalement implicite et à valeur uniforme sur la diagonale (RKEDIU). Une méthode RKEDIU (connu sous le sigle ESDIRK en anglais) est un cas particulier des RKDI où le premier étage se calcule explicitement et où tous les autres étages ont la même valeur pour les coefficients de la diagonale du tableau de Butcher.

On peut également utiliser les RKA_2 pour intégrer différemment des régions distinctes d'un domaine de calcul. Il n'est pas rare que la taille des éléments d'un maillage présente de grandes variations, notamment pour bien représenter les couches limites. Si le rapport entre la taille maximale et la taille minimale des éléments est grand, il peut être avantageux (i.e. temps de calcul réduit) de partitionner le maillage

en deux régions et d'utiliser des méthodes d'intégration en temps différente sur chacune d'elles (Kanevsky & al., 2007).

Mentionnons également les méthodes de Runge-Kutta imbriquées. Ces méthodes permettent deux évaluations successives à un pas de temps donné, l'un d'ordre p et l'autre d'ordre p + 1. La différence entre les deux évaluations permet d'établir un contrôle sur le pas de temps afin d'atteindre une précision voulue (Hairer, Nørsett & Wanner, 1993; Kanevsky & al., 2007). Ces méthodes ne découlent pas de l'addition de deux méthodes existantes (ce ne sont pas des RKA). Gresho nomme ces méthodes « intégrateurs intelligents » (Gresho, Sani & Engelman, 1998), car la méthode gère elle-même le pas de temps.

Quelques applications

Les méthodes d'intégration en temps mentionnées ci-haut peuvent être appliquées à divers modèles mathématiques autant en mécanique des fluides qu'en mécanique du solide. Par exemple, Büttner et Simeon (2003) démontrent la viabilité des méthodes de Runge-Kutta pour la résolution de problèmes plastiques (déformation irréversible d'un matériau). Huerta et Donea (2002) analysent l'impact des techniques de stabilisation numérique sur quelques méthodes de Runge-Kutta.

La littérature présente peu d'applications concrètes et directes des méthodes d'intégration en temps. Ces méthodes sont généralement intégrées à des logiciels de calculs (par éléments finis ou autre) et ces logiciels permettent l'étude de cas réels d'ingénierie. La compréhension des outils mathématiques en arrière-plan du logiciel est une étape fondamentale pour le bon développement de ces logiciels.

CHAPITRE 2 - QUELQUES NOTIONS THÉORIQUES UTILES

Ce deuxième chapitre couvre toutes les notions théoriques qui seront utiles pour la compréhension et l'étude des méthodes de Runge-Kutta implicites. La méthode des éléments finis est également abordée.

Ordre de convergence et critères de stabilité

Quel que soit le schéma d'intégration en temps utilisé, il faut s'assurer de certaines propriétés. Essentiellement, le schéma doit converger. La convergence est difficile à montrer directement. Indirectement, il y aura convergence si les deux critères suivants (Fletcher, 1991) sont respectés :

- → Le schéma est consistant;
- → Le schéma est stable.

Le premier critère signifie que la discrétisation (spatiale et temporelle) représente fidèlement l'équation aux dérivées partielles approximée, jumelée à une suite de conditions initiales et limites appropriées. Lorsque la taille de discrétisation tend vers zéro, on doit retrouver l'équation originale. Cela est également vérifiable par des développements en séries de Taylor. On suppose ici que l'équation aux dérivées partielles possède une solution unique.

En pratique, on ne peut faire tendre la taille de discrétisation vers zéro. Il faut alors introduire le concept de précision de la solution numérique. Soit $*y^n$ la solution exacte, au temps n, d'une équation différentielle ordinaire, i.e. $*y^n = y(t_n)$ et soit y^n la solution approximée par un schéma d'intégration en temps. Alors l'erreur sur la solution est :

$$E_n = \left| y^n - *y^n \right| \tag{2.1}$$

De plus, si le pas Δt est petit, on aura :

$$E_n \leq C\Delta t^p \tag{2.2}$$

où C est une constante. Si le schéma est consistant, alors E_n tend vers zéro lorsque Δt tend vers zéro. Cependant on peut difficilement prendre $\Delta t \approx 0$, alors on évalue l'erreur sur deux discrétisations successives, avec $(\Delta t)_2 = (\Delta t)_1 / 2$ pour un $(\Delta t)_1$ assez petit. On en déduit alors que :

$$p = \frac{\log\left((E_n)_2 / (E_n)_1 \right)}{\log\left(1/2 \right)} \tag{2.3}$$

et p est l'ordre de convergence du schéma. On dit que l'erreur est d'ordre p, $O(\Delta t^p)$, c'est-à-dire que le premier terme composant l'erreur est fonction de Δt^p.

Le second critère (schéma stable) représente la réaction du schéma à une perturbation (qui peut être aussi petite que les erreurs d'arrondi). Un schéma est instable si les erreurs introduites augmentent à chaque pas de temps. Inversement, si ces erreurs diminuent, le schéma est stable. Fletcher (1991) explique dans son ouvrage deux procédures pour déterminer les critères de stabilité d'un schéma : l'analyse matricielle et l'analyse de Von Neumann.

À titre d'illustration, considérons l'équation (1.4). La solution est stable si :

$$\left| \frac{1 + \Delta t \lambda (1-\theta)}{1 - \Delta t \lambda \theta} \right| < 1 \tag{2.4}$$

puisqu'à ce moment, $y^{n+1} < y^n$ et la solution numérique décroît, ce qui est conforme au comportement de la solution analytique (1.2) qui décroît de façon monotone. La relation (2.4) est donc un critère pour assurer la stabilité. Il est à noter que pour le schéma d'Euler explicite ($\theta = 0$), le critère de stabilité devient :

$$\Delta t < -2/\lambda \quad , \quad \lambda < 0 \tag{2.5}$$

Pour Euler implicite, ($\theta = 1$), il n'y a aucun critère de stabilité, ce qui induit un certain intérêt pour les méthodes implicites.

L'équation considérée est rarement une équation scalaire et on se retrouve généralement avec un système d'équations ayant chacune une solution en différents points (i.e. éléments finis, différences finis, etc.). L'analyse de stabilité devient alors plus complexe. On retrouvera des détails sur la stabilité des méthodes de Runge-Kutta implicites au quatrième chapitre.

Méthode des éléments finis

La méthode des éléments finis a vu ses débuts avec les ouvrages de pionniers comme Richard Courant (1888 - 1972), dans les années quarante. Courant se basait sur les théories de la formulation variationnelle développée par Walter Ritz (1878 - 1909) et Boris Grigoryevich Galerkin (1871 - 1945) au début du siècle ainsi que par des travaux qui remontent jusqu'à Léonard Euler (1707 –1783). À ce moment, des ingénieurs civils commencent à s'intéresser à la méthode, mais le manque de puissance de calcul est un handicap majeur. Avec l'apparition de l'ordinateur dans les années soixante, John H. Argyris et Olgierd C. Zienkiewiecz ont ravivé l'intérêt de la méthode et lui ont donné la forme et l'utilité actuelle[*].

La méthode des éléments finis consiste d'une part à discrétiser un domaine Ω en éléments, formant ensemble le maillage, et d'autre part à donner une formulation variationnelle élémentaire (discrétisée) d'un système d'équations aux dérivées partielles. Le nouveau système d'équations résultant est un système d'équations algébriques ou encore un système d'équations différentielles ordinaires.

Soit Ω un domaine quelconque dans $\mathbb{R}^3$. Une partition en éléments (une triangulation dans le cas d'éléments triangulaires) de Ω s'écrit T et est définie ainsi :

[*] Tiré du site web de CAD-FEM AG en Allemagne. Gebald, C. (date n/d), *Histoire de la Méthode des Eléments Finis*, CADFEM Computer Aided Engineering, consulté le 12 mars 2009, http://www.cadfem.ch/index.php?id=4824

$$T = \left\{ \Omega_e \mid \bigcup_{e=1}^{Ne} \Omega_e = \Omega \, , \bigcap_{e=1}^{Ne} \Omega_e = \varnothing \right\} \tag{2.6}$$

où les Ω_e sont les éléments et Ne est le nombre d'éléments. Cette partition est aussi définie par une table des coordonnées donnant les positions géométriques de chaque nœud ainsi que d'une table des connectivités donnant les positions topologiques des nœuds. Cette dernière table indique pour chaque élément les nœuds faisant partie de cet élément. D'autres formulations existent pour ces tables.

Espaces fonctionnels et formulation variationnelle

La méthode des éléments finis repose sur certains espaces fonctionnels. L'espace $L^2(\Omega)$ est l'ensemble de toutes les fonctions de carré intégrable :

$$L^2(\Omega) = \left\{ u \mid \int_\Omega u^2 d\Omega < \infty \right\} \tag{2.7}$$

et $H^1(\Omega)$ (espace de Hilbert) est le sous-ensemble de $L^2(\Omega)$:

$$H^1(\Omega) = \left\{ u \in L^2(\Omega) \mid \frac{\partial u}{\partial x_i} \in L^2(\Omega) \right\} \tag{2.8}$$

où les x_i sont les composantes de $\mathbf{x}$. Les normes de ces espaces sont :

$$\|u\|_{0,\Omega} = \left(\int_\Omega u^2 d\Omega \right)^{1/2} \text{ pour } L^2(\Omega) \tag{2.9}$$

$$\|u\|_{1,\Omega} = \left(\int_\Omega u^2 d\Omega + \int_\Omega \left(\frac{\partial u}{\partial x_i} \right)^2 d\Omega \right)^{1/2} \text{ pour } H^1(\Omega) \tag{2.10}$$

La formulation variationnelle d'une équation implique que celle-ci soit affaiblie en multipliant son résidu fort par une fonction test $w(\mathbf{x})$ puis en intégrant sur tout le domaine. Pour présenter la démarche, on utilise l'équation de diffusion stationnaire unidimensionnelle :

$$-\frac{d^2T}{dx^2} = f(x) \text{ dans } \Omega =]a; b[\tag{2.11}$$

Cette démarche s'étend directement à plusieurs dimensions. La formulation variationnelle sur Ω ($d\Omega = dx$) est :

$$-\int_\Omega w(x)\frac{d^2T}{dx^2}dx = \int_\Omega w(x)f(x)dx \quad \forall w \in V \tag{2.12}$$

où w(x) est appelée fonction test et l'espace V est $L^2(\Omega)$. Le membre de gauche doit être intégré par parties. Cela aura aussi l'effet de faire ressortir une expression introduisant les flux de chaleur aux frontières du problème :

$$-\int_\Omega w(x)\frac{d^2T}{dx^2}dx = \int_\Omega \frac{dw(x)}{dx}\frac{dT}{dx}dx - [w(b)F(b) + w(a)F(a)] \tag{2.13}$$

où

$$F(b) = \left.\frac{dT}{dx}\right|_{x=b} \quad \text{et} \quad F(a) = -\left.\frac{dT}{dx}\right|_{x=a}$$

représentent les flux aux frontières. On fait le choix pour l'instant que la fonction w(x) est telle que w(a) = w(b) = 0. Il est à noter que ce choix est adéquat seulement si les conditions frontières sont des conditions de Dirichlet. On dit alors qu'il s'agit d'une condition essentielle ou encore d'une imposition forte. En contrepartie, une condition de Neumann impose un flux de chaleur. Cette condition est dite naturelle, d'imposition faible. Ceci dit, l'équation (2.12) devient :

$$\int_\Omega \frac{dw(x)}{dx}\frac{dT}{dx}dx = \int_\Omega w(x)f(x)dx \quad \forall w \in V \tag{2.14}$$

Plus succinctement, l'équation (2.14) se réécrit :

$$\alpha(T,w) = \ell(w) \quad \forall w \in V \tag{2.15}$$

où $\ell(w)$ est une forme linéaire continue sur V et $\alpha(T,w)$ une forme bilinéaire continue sur $V \times V$ et coercive (elliptique). Le lien entre (2.14) et (2.15) est direct. Par le théorème de Lax-Milgram (Raviart & Thomas, 1983), il existe une solution unique au problème variationnel.

Formulation variationnelle discrète

La discrétisation de la solution T(x) s'écrit :

$$T(x) \approx T^N(x) + T_r(x) = \sum_{j=1}^{N} \Psi_j(x) T_j + T_r(x) \tag{2.16}$$

où $\Psi_j(x)$ constitue une suite de fonctions d'interpolation globale linéairement indépendantes et $T_r(x)$ une fonction de relèvement qui dépend des conditions limites (pour des conditions limites essentielles homogènes, $T_r = 0$). Le problème variationnel discret devient :

Trouver la fonction $T^N \in V^N \subset V$ telle que :

$$\alpha(T^N, w^N) = \ell(w^N) \quad \forall w^N \in V^N \tag{2.17}$$

ou encore, en introduisant (2.16) :

$$\sum_{j=1}^{N} \alpha(\Psi_j, w_i) \cdot T_j = \ell(w_i) \quad i = 1,2,\ldots,N \tag{2.18}$$

La formulation par éléments finis standard (Ritz-Galerkin) prend les fonctions tests du même espace que les fonctions d'interpolation. On transpose les interpolations au niveau local (élémentaire) et on choisit une base d'interpolation de Lagrange spécifique à un élément de référence. Ce passage à l'élément de référence s'accompagne d'un changement de variable(s). On peut ensuite utiliser l'intégration numérique pour calculer les intégrales. Finalement, le système matriciel résultant peut

être résolu par diverses méthodes directes ou itératives. Ces méthodes de résolution ne sont pas décrites dans le présent mémoire.

Il est à noter que la discrétisation par éléments finis d'équations en régime stationnaire conduit généralement à un système d'équations algébriques alors que pour les équations en régime transitoire, la discrétisation conduit à un système d'équations différentielles ordinaires. On résout alors ce système par des schémas d'intégration en temps (voir le chapitre 1 pour certaines méthodes et le chapitre 4 pour les RKI en particulier). Cette façon de procéder entre dans une formulation semi-discrète puisque seul l'espace est discrétisé par éléments finis. On appelle aussi cette formulation la méthode des lignes (Fletcher, 1991; Hairer, Nørsett, & Wanner, 1993).

CHAPITRE 3 - ÉQUATION DE TRANSPORT, FORME ADIMENSIONNELLE ET DISCRÉTISATION

L'équation de transport est une version particulière de l'équation de la conservation de l'énergie dans laquelle on ne considère que les transferts de chaleur par convection et par diffusion (conduction) en régime transitoire. Sous forme différentielle, cette équation s'écrit ainsi :

$$\rho C\left(\frac{\partial T}{\partial t}+\mathbf{u}\cdot\nabla T\right)=\nabla\cdot(k\nabla T)+f(t,\mathbf{x}) \text{ dans } \Omega \tag{3.1}$$

où

T	→	champ scalaire de température;
t	→	temps;
u	→	champ de vitesse, **u** = (u,v) en 2D et **u** = (u,v,w) en 3D;
ρ	→	densité du fluide;
C	→	chaleur spécifique du fluide;
k	→	coefficient de diffusion;
x	→	vecteur position, **x** = (x,y) en 2D et **y** = (x,y,z) en 3D;
f(t,**x**)	→	champ source de chaleur (scalaire).

De façon générale, les conditions initiale et frontières pour l'équation (3.1) sont :

$$T = T_0 \text{ à } t = 0 \tag{3.2}$$

$$T = T_d \text{ sur } \Gamma_d \tag{3.3}$$

$$k\nabla T\cdot\mathbf{n} = F_n \text{ sur } \Gamma_n \tag{3.4}$$

Dans les conditions (3.2) à (3.4), T_0 est une solution initiale connue sur tout le domaine, F_n exprime un flux de chaleur à la frontière, l'indice « d » fait référence à une condition de type Dirichlet et l'indice « n » fait référence à une condition de type Neumann.

L'équivalent unidimensionnel de l'équation (3.1) est :

$$\rho C\left(\frac{\partial T}{\partial t}+u\frac{\partial T}{\partial x}\right)=\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+f(t,x) \text{ dans } \Omega_{1D} \in]0,1[\qquad (3.5)$$

avec les conditions initiale (3.2) et frontière (3.3) identiques au cas général, et la condition de Neumann est :

$$k\frac{\partial T}{\partial x}=F_n \text{ sur } \Gamma_n \text{ et le flux sortant en } x = 1. \qquad (3.6a)$$

ou

$$-k\frac{\partial T}{\partial x}=F_n \text{ sur } \Gamma_n \text{ et le flux sortant en } x = 0. \qquad (3.6b)$$

Forme adimensionnelle

Une bonne pratique en simulation numérique consiste à écrire les équations sous forme adimensionnelle. On peut alors transposer une solution T d'un problème modèle de grandeurs réduites vers un problème de grandeurs réelles. Pour ce faire, il est nécessaire de choisir quelques grandeurs de référence et de respecter les lois de similitude. Ces lois indiquent simplement que la géométrie du modèle réduit doit être une reproduction identique de la géométrie originale, à un facteur d'échelle près. De plus, il faut s'assurer de la similitude dynamique, c'est-à-dire que les rapports de forces entre le modèle dimensionnel et le modèle adimensionnel soient conservés. En posant que ρ, C, k et u sont constants, les grandeurs de référence sont :

T_∞ → température de référence;
L → longueur caractéristique;
f_∞ → terme source de référence.
τ → temps caractéristique;

On définit ainsi les variables adimensionnelles à partir des variables dimensionnelles. Ces dernières sont notées par un « ' » ci-dessous.

$$x = x'/L \quad t = t'/\tau \quad T = T'/T_\infty \quad f = f'/f_\infty \tag{3.7}$$

Pour mettre sous forme adimensionnelle, il suffit de remplacer dans l'équation (3.5) les variables dimensionnelles par les variables adimensionnelles (3.7) :

$$\frac{\rho C T_\infty}{\tau}\frac{\partial T}{\partial t} + \frac{\rho C u T_\infty}{L}\frac{\partial T}{\partial x} = \frac{k T_\infty}{L^2}\frac{\partial^2 T}{\partial x^2} + f_\infty f(t', x') \tag{3.8}$$

En multipliant chaque membre de l'équation par $L/\rho C u T_\infty$:

$$\frac{L}{\tau u}\frac{\partial T}{\partial t} + \frac{\partial T}{\partial x} = \frac{1}{Pe}\frac{\partial^2 T}{\partial x^2} + \frac{L f_\infty}{\rho C u T_\infty} f(t', x') \tag{3.9}$$

Dans l'équation précédente, Pe est le nombre de Peclet. Le nombre de Peclet exprime l'importance relative entre les mécanismes de convection et ceux de diffusion. Il est défini selon la formule suivante :

$$Pe = \frac{\rho C u L}{k} \tag{3.10}$$

Il faut choisir les valeurs des grandeurs de référence. Puisque le domaine de calcul va de x' = 0 à x = 1, la longueur caractéristique est L = 1. Pour les autres grandeurs de référence, on propose :

$$\tau = \frac{L}{u} \quad \text{et} \quad f_\infty = \frac{\rho C u T_\infty}{L} \tag{3.11}$$

Avec ce choix, l'équation transport 1D devient :

$$\frac{\partial T}{\partial t} + \frac{\partial T}{\partial x} = \frac{1}{Pe}\frac{\partial^2 T}{\partial x^2} + f(t', x') \text{ dans } \Omega_{1D,adim} \in [0,1] \tag{3.12}$$

Le terme source est toujours fonction des variables dimensionnelles t' et x'. Il est possible de le modifier afin de le définir en fonction de x. Il ne s'agit que d'un changement de variable indépendante :

$$f(t',x') = f(\tau t, Lx) = F(t,x) \tag{3.13}$$

Dans le cas présent, L = 1 et τ = 1 (ce qui implique u = 1), donc F(t,x) = f(t,x). Finalement, les conditions initiale et frontières deviennent :

$$T = T_0/T_\infty \quad \text{à } t = 0 \tag{3.14}$$

$$T = T_d/T_\infty \quad \text{sur } \Gamma_{d,adim} \tag{3.15}$$

$$\pm k\frac{\partial T}{\partial x} = \frac{LF_n}{k_\infty T_\infty} = H_n \quad \text{sur } \Gamma_{n,adim} \tag{3.16}$$

L'équation (3.12) sera utilisée pour effectuer quelques tests pour valider les ordres de convergence des méthodes de Runge-Kutta implicites (voir chapitre 5).

Discrétisation par la méthode des éléments finis

Pour l'équation de chaleur (3.12), la discrétisation par éléments finis conduit au problème suivant :

Trouver $T^N \in V^N$ telle que $\forall w^N \in V^N$

$$\begin{aligned}&\int_\Omega w^N(x)\frac{\partial T^N}{\partial t}dx + \int_\Omega w^N(x)\frac{\partial T^N}{\partial x}dx + \frac{1}{Pe}\int_\Omega \frac{\partial w^N}{\partial x}\frac{\partial T^N}{\partial x}dx \\ &\quad = \int_\Omega w^N(x)f(t,x)dx + w^N(\Gamma_n)H_n\end{aligned} \tag{3.17}$$

Le dernier terme de (3.17) représente les conditions frontières de Neumann, avec H_n la valeur adimensionnelle du flux de chaleur et Γ_n la frontière concernée. Puisque le domaine est unidimensionnel, ce terme ne sera en fait qu'une valeur scalaire qui s'ajoute au degré de liberté correspondant dans la matrice globale. À noter que la solution et les fonctions tests sont dans le même espace ($L^2(\Omega)$).

Stabilisation numérique (SUPG)

Dans les cas où les phénomènes de convection sont prépondérants sur ceux de diffusion, des oscillations surviendront si on utilise la formulation standard Ritz-Galerkin et un maillage dont la taille des éléments est trop grande. Cela est dû au fait que l'information se propage dans un sens en particulier (i.e. équation à tendance hyperbolique). Afin de remédier à ces oscillations, un mécanisme de stabilisation doit être introduit. Un choix adéquat pour l'équation (3.17) est la stabilisation SUPG (Streamline-Upwind Petrov-Galerkin) développée par Brooks et Hugues (Brooks & Hughes, 1981). La méthode SUPG ajoute un terme de stabilisation, souvent noté ST (De Mulder, 1997), à la formulation variationnelle discrétisée :

$$ST = \sum_e \left(\int_{\Omega_e} \tau_{SUPG} \mathbf{u} \cdot \nabla w^N \left[R(T^N) \right] d\Omega \right) \tag{3.18}$$

Dans l'équation (3.18), les variables représentent :

$\mathbf{u}$	$\rightarrow$	vecteur champ de vitesse, $\mathbf{u}$ = (u,v) en 2D et $\mathbf{u}$ = (u,v,w) en 3D;
w^N	$\rightarrow$	fonction test sur l'élément;
τ_{SUPG}	$\rightarrow$	facteur d'échelle de temps basé sur l'élément;
$R(T^N)$	$\rightarrow$	résidu fort de l'équation différentielle.

La version unidimensionnelle de (3.18) donne :

$$ST = \sum_e \left(\int_{\Omega_e} \tau_{SUPG} u \frac{\partial w^N}{\partial x} \left[R(T^N) \right] dx \right) \tag{3.19}$$

de telle sorte que l'équation (3.17) devient :

$$\begin{aligned} &\int_\Omega w^N(x) \frac{\partial T^N}{\partial t} dx + \int_\Omega w^N(x) \frac{\partial T^N}{\partial x} dx + \frac{1}{Pe} \int_\Omega \frac{\partial w^N}{\partial x} \frac{\partial T^N}{\partial x} dx \\ &+ \sum_e \left(\int_{\Omega_e} \tau_{SUPG} u \frac{\partial w^N}{\partial x} \left[R(T^N) \right] dx \right) = \int_\Omega w^N(x) f(t,x) dx + w^N(\Gamma_n) H_n \end{aligned} \tag{3.20}$$

et le résidu est simplement :

$$R(T^N) = \frac{\partial T}{\partial t} + \frac{\partial T}{\partial x} - \frac{1}{Pe}\frac{\partial^2 T}{\partial x^2} - f(t,x) \tag{3.20}$$

Il est à noter qu'avec le résidu fort, la consistance de l'équation variationnelle discrétisée est conservée, ce qui est préférable pour atteindre des ordres de convergence supérieur à un (De Mulder, 1997). Le facteur d'échelle τ_{SUPG} est évalué comme suit :

$$\tau_{SUPG} = \frac{h_e}{2u}\zeta(Pe^N) \tag{3.21}$$

et

$$\zeta(Pe^N) = \coth(Pe^N) - 1/Pe^N \tag{3.22}$$

où Pe^N est le nombre de Peclet local. Cette démarche est optimale pour les éléments finis unidimensionnels avec interpolation linéaire sur des équations en régime stationnaire. Étant donné les cas multidimensionnels présentés dans ce mémoire, on utilisera la définition de τ_{SUPG} proposée par Tezduyar & Osawa (1999). Il en sera de même pour les autres paramètres de stabilisation que l'on retrouve dans la deuxième partie de ce mémoire. La stabilisation SUPG est traitée par plusieurs chercheurs (Bochev, Gunzburger & Shadid, 2004; Franca & Oliveira, 2003; Tezduyar & Senga, 2007).

La stabilisation sera nécessaire lorsque le nombre de Peclet local est plus grand que l'unité :

$$Pe^N = \frac{\rho C u h}{2k} \geq 1 \tag{3.23}$$

Lorsque la vitesse est variable, on se servira plutôt du nombre de Peclet local suivant :

$$Pe^N = \frac{\rho C \mathbf{u}(\mathbf{x}) h}{2k} \geq 1 \qquad (3.24)$$

Il est donc possible que le transfert de chaleur par convection ne soit prépondérant sur celui de diffusion que sur une partie du domaine et non sur la totalité du domaine. De même, la stabilisation SUPG peut être appliquée seulement là où les besoins se font sentir, au prix d'une programmation légèrement plus complexe.

CHAPITRE 4 - ÉTUDE DES MÉTHODES IMPLICITES DE RUNGE-KUTTA

Pour atteindre une meilleure précision sur l'évolution de l'écoulement, il est intéressant d'introduire des méthodes d'intégration en temps d'ordre de convergence élevé. Jusqu'à maintenant, nous utilisions la formule d'Euler implicite, qui est d'ordre un, pour faire évoluer la solution d'un pas de temps au suivant. Bien que cette méthode présente des propriétés idéales de stabilité, il se trouve que pour atteindre un bon niveau de précision, le pas de temps doit être petit. On doit donc faire beaucoup d'étape de temps.

Ce chapitre est dédié à l'étude des méthodes implicites de Runge-Kutta (RKI) qui peuvent atteindre des ordres élevés, soit trois, quatre, cinq ou même plus. Pour un pas de temps donné, plus l'ordre de la méthode est élevé et plus l'erreur entre la solution numérique et la solution analytique est faible. Nous prendrons pour acquis que les méthodes de Runge-Kutta, de façon générale, possèdent une solution unique et qu'elles produisent des schémas consistants, fidèles aux équations aux dérivées partielles. La preuve de ces acquis se trouve dans Hairer, Nørsett et Wanner (1993) ainsi que dans les ouvrages cités par ces auteurs. Cependant, l'étude qui suit s'oriente vers la stabilité de ces méthodes.

La fonction de stabilité

Les équations (1.7a-b) donnent les méthodes de Runge-Kutta, implicites ou explicites. En utilisant cette définition, une méthode de Runge-Kutta composée de *s* étages peut s'écrire également (pour y' = f(t,y) et h = Δt) :

$$y_{n+1} = y_n + h\sum_{j=1}^{s} b_j f(t_0 + c_j h, g_j) \tag{4.1a}$$

avec les évaluations internes

$$g_i = y_n + h\sum_{j=1}^{s} a_{ij} f(t_0 + c_j h, g_j) \quad i = 1,\ldots,s \tag{4.1b}$$

Selon les critères donnés sous le tableau de Butcher (1.8), la méthode de Runge-Kutta est dite implicite si au moins un des $a_{ij} \neq 0 \; \forall \; i < j$.

La notion de stabilité numérique des schémas d'intégration en temps a été introduite au chapitre 2. Il s'agit d'une condition essentielle pour atteindre une solution numérique qui soit représentative des équations étudiées. Avec des schémas multi-étages tels les RKI, on voudra :

$$|G(z)| < 1 \tag{4.2}$$

où G(z) est appelée la fonction de stabilité. Le critère (4.2) assure la stabilité et impose une restriction sur z. De façon générale, z peut être complexe et est déterminé par l'équation discrétisée (il dépend de h). L'étude de stabilité se fait dans le plan complexe.

Il est indispensable de déterminer la fonction de stabilité pour toutes les méthodes RKI que l'on veut utiliser. À nouveau pour l'équation modèle (1.1), la fonction de stabilité de (4.1a-b) est :

$$G(z) = \left[1 + z \cdot \mathbf{b}^T \cdot (\mathbf{I} - z\mathbf{A})^{-1} \boldsymbol{\iota}\right] \tag{4.3}$$

où

$$\mathbf{A} = (a_{ij})_{i,j=1}^{s} \quad \mathbf{b}^T = (b_1, b_2, ..., b_s) \quad \boldsymbol{\iota} = (1,1,...,1)^T \quad \mathbf{I} = (\delta_{ij})_{i,j=1}^{s} \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

Les variables données dans (4.3) sont :

- A → la matrice des coefficients a_{ij} donnée dans le tableau de Butcher;
- $\mathbf{b}^T$ → le vecteur b transposé donné dans le tableau de Butcher;
- $\boldsymbol{\iota}$ → le vecteur « identité »;
- $\mathbf{I}$ → la matrice identité.

La preuve de l'équation (4.3) est donnée dans (Hairer & Wanner, 1996). Il suffit de remplacer $f(t,y) = \lambda y(t)$, avec $z = \lambda h$, dans (4.1b). Ceci donne un système d'équations linéaires dont la solution est insérée dans (4.1a) pour obtenir (4.3).

Méthodes « A-stable »

La fonction de stabilité, pour $z = \lambda h$ quelconque (complexe) et $\lambda < 0$, définit sur le plan complexe un domaine de stabilité :

$$S = \{ z \in \mathbb{C} \mid |G(z)| \leq 1 \} \tag{4.4}$$

Si le domaine S englobe la partie négative du plan complexe, alors la méthode est dite A-stable :

$$S \supset \mathbb{C}^- = \{ z \mid \mathrm{Re}(z) < 0 \} \tag{4.5}$$

On remarque que pour satisfaire (4.5), $\lambda < 0$, ce qui est donné justement par l'équation modèle. La solution numérique possède donc le même caractère de stabilité que la solution exacte quelque soit λ, c'est-à-dire que la solution numérique est stable si l'ÉDO est stable et elle est instable si l'ÉDO est instable.

Pour une méthode de Runge-Kutta ayant l'équation (4.3) comme fonction de stabilité, la propriété de A-stabilité s'applique à la méthode si et seulement si :

$$|G(ix)| \leq 1 \quad \forall x \in \mathbb{R} \tag{4.6}$$

et G(z) existe pour $\mathrm{Re}(z) < 0$. Le coefficient i est le nombre imaginaire $i = \sqrt{-1}$. Ainsi, si la fonction de stabilité est inférieure à l'unité sur tout l'axe imaginaire du plan complexe, alors elle l'est pour toute la partie négative du plan complexe.

À titre de comparaison, la fonction de stabilité des méthodes de Runge-Kutta explicites (RKE) d'ordre p est :

$$G(z) = 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \ldots + \frac{z^p}{p!} + O(z^{p+1}) \tag{4.7}$$

On peut visualiser dans le plan complexe le domaine de stabilité des RKE et des RKI. Pour les méthodes explicites, la figure 4.1 affiche ces domaines pour un ordre p variant de 1 à 4. La courbe « RKE ordre 4 » correspond au schéma RK4, bien connue et largement utilisée. Sur cette figure, le critère de stabilité d'une méthode sur le pas de temps est respecté pour les z à l'intérieur de la courbe fermée correspondant à cette méthode. L'intérieur des courbes fermées délimitent les domaines de stabilité des schémas RKE. On note que, pour un λ fixe, si h devient trop grand, alors z se retrouve à gauche des zones de stabilité et le schéma devient instable.

La stabilité des RKI est tout autre, comme le montre la figure 4.2. Les différentes courbes de cette figure représentent un nombre d'étages s variant de 1 à 4. L'ordre est relié au nombre d'étages selon la famille de RKI utilisée (Gauss, Radau ou Lobatto). Avec s = 1, il s'agit du schéma d'Euler implicite (RadauIIA1). Les autres méthodes sont RadauIIA3, RadauIIA5 et LobattoIIIC6. Cette fois-ci, le domaine de stabilité se retrouve à l'extérieur des courbes fermées. On remarque que la totalité de la partie négative du plan fait partie de la zone stable, ce qui indique que h peut prendre n'importe quelle valeur sans causer d'instabilité. Les schémas RKI affichés sur la figure 4.2 sont tous A-stables.

La figure 4.3 est un agrandissement autour de l'axe imaginaire de la figure 4.2 qui permet de bien voir que les domaines de stabilité incluent toute la portion négative du plan.

Finalement, Chipman (1971) fait référence à la propriété *A-stabilité forte*, mais par comparaison avec les travaux d'auteurs subséquents (Gresho, Sani, & Engelman, 1998; Hairer & Wanner, 1996; Rang, 2008), on peut davantage relier la définition qu'il donne à la propriété de L-stabilité décrite à la prochaine sous-section.

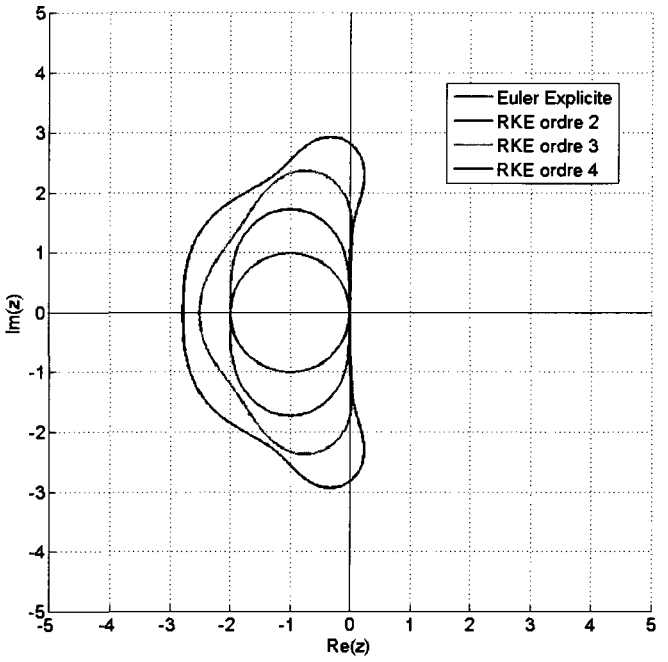

Figure 4.1 : Domaine de stabilité pour les RKE d'ordre 1 à ordre 4; le domaine de stabilité est à l'intérieur des courbes

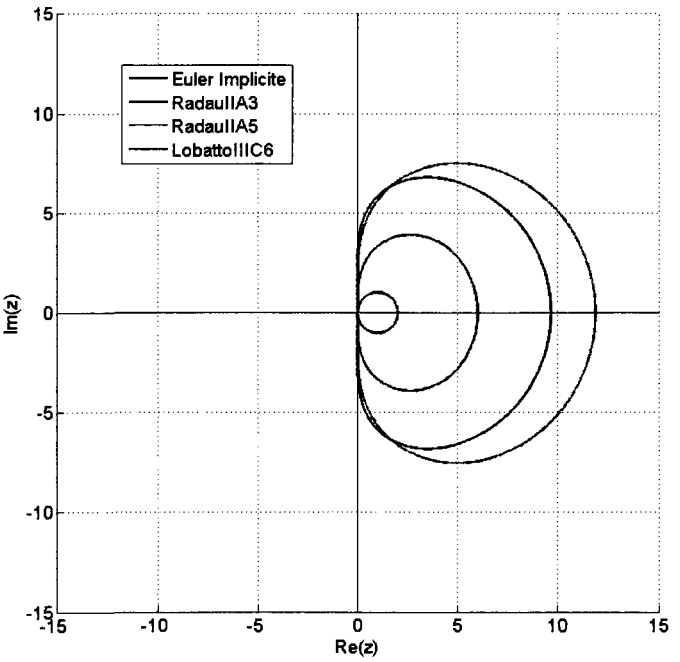

Figure 4.2 : Domaine de stabilité pour les RKI d'ordre 1, 3,5 et 6; le domaine de stabilité est à l'extérieur des courbes

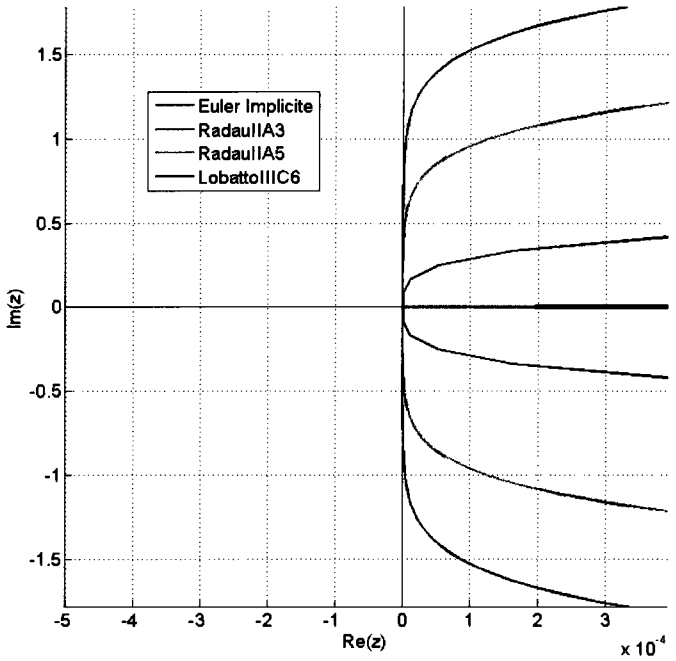

Figure 4.3 : Agrandissement autour de Re(z) = 0 de la figure précédente

Méthodes « L-stable »

Il a été observé que parfois, même si une méthode est A-stable, la solution numérique ne s'approche que très lentement de la solution recherchée. Cela se produit lorsque G(z) s'approche de l'unité lorsque $z \rightarrow -\infty$. La méthode possédant une telle fonction de stabilité donnera bien sûr une bonne solution, mais cette solution présentera une forte oscillation qui s'estompe très lentement.

Une méthode est dite L-stable si elle est A-stable et si :

$$\lim_{|z|\to\infty}(G(z)) = 0 \tag{4.8}$$

Par comparaison avec ce qui a été rapporté ci-haut dans les travaux de Chipman (1971), Rang (2008) parle de la propriété de *A-stabilité forte* lorsque :

$$\lim_{|z|\to\infty}(G(z)) < 1$$

La propriété de *A-stabilité forte* peut donc être vue comme intermédiaire entre *A-stabilité* et *L-stabilité.*

Le schéma d'Euler implicite est L-stable, mais le schéma du trapèze implicite est seulement A-stable. Voici un exemple tiré de Hairer et Wanner (1996) pour illustrer la situation.

Soit l'équation différentielle suivante :

$$\begin{aligned} y' &= -2000(y - \cos(t)) \\ y(t_0) &= 0 \qquad t_0 = 0 \le t \le 1{,}5 \end{aligned} \tag{4.9}$$

Les deux schémas mentionnés ci-haut avec 40 étapes de temps sont :

- Euler implicite (h = Δt = 1,5 / 40)

$$y^{n+1} = y^n \left[\frac{1}{1+2000h} \right] + \frac{2000h\cos(t^n)}{1+2000h} \tag{4.10}$$

- Trapèze implicite (h = Δt = 1,5 / 40)

$$y^{n+1} = y^n \left[\frac{1-1000h}{1+1000h} \right] + \frac{1000h\left(\cos(t^n)+\cos(t^{n+1})\right)}{1+1000h} \tag{4.11}$$

On voit sur la figure 4.4 les oscillations du schéma du trapèze implicite alors que celui d'Euler implicite ne présente pas d'oscillation. En fait, le schéma du trapèze a des difficultés à réduire l'influence de la phase de transition entre la condition initiale et la solution recherchée.

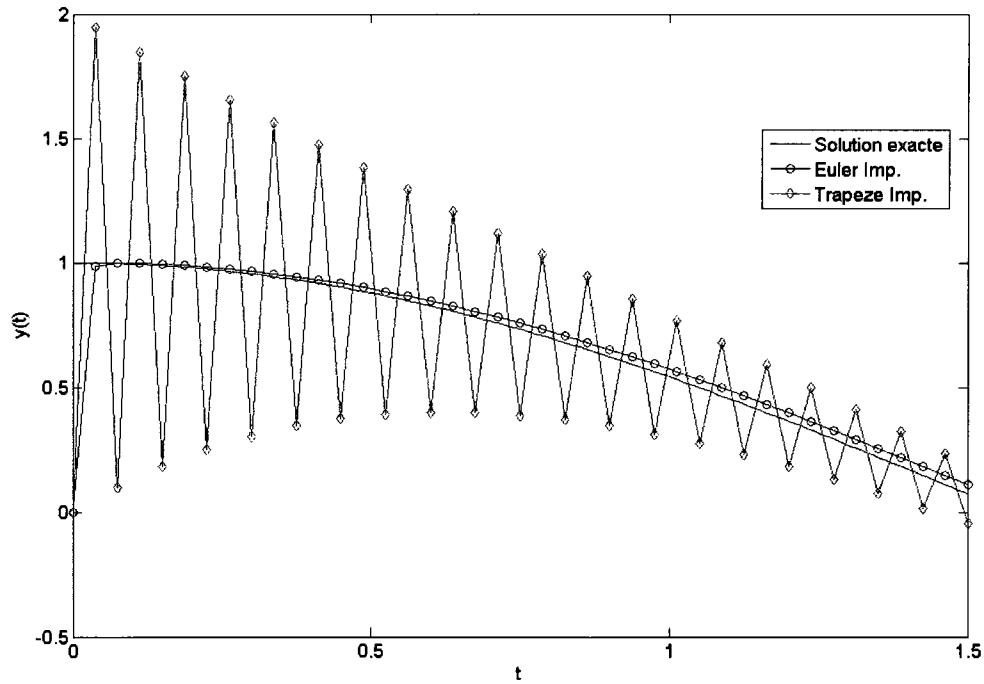

Figure 4.4 : Graphique de la solution à l'équation (4.9); présence d'oscillations décroissantes lorsque la méthode n'est pas L-stable

Les fonctions de stabilité des deux schémas présentés ci-haut sont respectivement :

$$G(z)=\frac{1}{1-z} \quad \text{et} \quad G(z)=\frac{1+z/2}{1-z/2} \tag{4.12}$$

Donc, l'équation (4.8) est satisfaite pour Euler implicite (L-stable), mais pas pour le trapèze implicite. Cette dernière est seulement A-stable. Il faut remarquer que les propriétés de A-stabilité et de L-stabilité ont plus d'importance lorsque l'équation différentielle est raide. L'équation (4.9) peut être qualifiée de raide étant donné le facteur d'amplification de 2000 devant la variable « y ». Fletcher (1991) donne une définition algébrique d'un comportement raide. Sachant que λ_k représente l'ensemble des valeurs propres de la matrice A résultante de la discrétisation spatiale (équation semi-discrétisée), alors le rapport de raideur de l'équation est donné par :

$$R = \left|\mathrm{Re}(\lambda_k)_{max}\right| \Big/ \left|\mathrm{Re}(\lambda_k)_{min}\right|$$

où Re(λ) représente la partie réelle de λ. Si R est de l'ordre de 10^4 et plus, alors l'équation est dite raide. Typiquement, pour un schéma explicite, la valeur propre maximale engendrera une contrainte sur le pas de temps (voir, par exemple, l'équation (2.5) pour le critère sur le schéma d'Euler explicite). Pour les schémas implicites, ces critères n'apparaissent pas et rendent ces schémas désirables pour résoudre les équations raides.

Gresho et Sani & Engelman (1998) défendent le schéma du trapèze implicite pour la résolution de l'équation de transport de chaleur, même si ce schéma n'est pas L-stable. Ce schéma possède d'autres bonnes propriétés, notamment celle de ne pas introduire de dissipation numérique. De plus, les oscillations peuvent être fortement réduites en introduisant un contrôle sur la grandeur du pas de temps. Ce schéma est d'ailleurs d'ordre supérieur à celui d'Euler implicite.

Différentes familles de RKI

Les méthodes de Runge-Kutta implicites se regroupent en différentes familles. Cette section présente quelques unes d'entre elles qui possèdent de bonnes propriétés de stabilité.

Les RKI peuvent, de façon générale, être définies par trois hypothèses simplificatrices :

$$\begin{aligned} B(p):&\quad \sum_{i=1}^{s} b_i c_i^{q-1} = \frac{1}{q} && q = 1,\ldots,p; \\ C(\eta):&\quad \sum_{j=1}^{s} a_{ij} c_j^{q-1} = \frac{c_i^q}{q} && i = 1,\ldots,s;\quad q = 1,\ldots,\eta; \\ D(\zeta):&\quad \sum_{i=1}^{s} b_i c_i^{q-1} a_{ij} = \frac{b_j}{q}(1 - c_j^q) && j = 1,\ldots,s;\quad q = 1,\ldots,\zeta. \end{aligned} \tag{4.13}$$

Le théorème IV.5.1 (p. 71) de Hairer et Wanner (1996), originalement énoncé par Butcher, est primordial quant à la détermination de l'ordre des méthodes :

Théorème :

Si les coefficients b_i, c_i, a_{ij} d'une méthode de Runge-Kutta satisfont B(p), C(η), D(ζ) avec $p \leq \eta + \zeta + 1$ et $p \leq 2\eta + 2$, alors la méthode est d'ordre p.

Les méthodes de Gauss (Kuntzmann-Butcher) et Lobatto-Gauss

Ces méthodes, élaborées par Butcher, sont inspirées de la quadrature de Gauss. Il faut d'abord choisir les points (les c_i, i = 1 à N) qui sont les zéros du polynôme de Legendre décalé (i.e. l'intervalle est [0;1] au lieu de [-1;1]) de degré s :

$$\frac{d^s}{dx^s}\left(x^s(x-1)^s\right) = 0 \qquad 0 \leq x \leq 1 \tag{4.14}$$

Butcher et Ehle (Butcher, 1964; Ehle, 1968) mentionnent que les méthodes de type Gauss à s étages sont d'ordre p = 2s. De plus, les fonctions stabilité de ces

méthodes sont données par l'approximation (s,s)-Padé (Hairer & Wanner, 1996) et ces méthodes sont A-stables. Les coefficients b_i et a_{ij} s'obtiennent des hypothèses simplificatrices (4.13). Les tableaux 4.1 et 4.2 donnent les coefficients pour les méthodes de Gauss d'ordre deux et d'ordre quatre. La méthode d'ordre 2 est équivalente à la méthode du point milieu implicite, une variante de la méthode du trapèze.

$$\begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array}$$

Tableau 4.1 : Tableau de Butcher pour la méthode de Gauss d'ordre 2

$$\begin{array}{c|cc} \frac{1}{2}-\frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4}-\frac{\sqrt{3}}{6} \\ \frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{4}+\frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Tableau 4.2 : Tableau de Butcher pour la méthode de Gauss d'ordre 4

Au niveau numérique, les méthodes de type Gauss sont avantageuses de par leur précision et leur stabilité. Cependant, au niveau algorithmique, il est nécessaire de calculer l'équation (4.1a) à la fin de l'étape de temps pour déterminer y_{n+1}. Cette étape additionnelle disparaît lorsque le dernier point d'intégration, c_s, vaut un, qu'on retrouve dans les méthodes des deux types suivants. Un autre avantage y sera décrit.

Les méthodes de Radau de type IIA

Les méthodes de Radau de type IIA (Radau à droite) à s étages ont leurs points de collocation donnés par les zéros du polynôme suivant :

$$\frac{d^{s-1}}{dx^{s-1}}\left(x^{s-1}(x-1)^s\right) \qquad 0 \le x \le 1 \tag{4.15}$$

Encore une fois, on déduit les b_i et les a_{ij}, donnant ainsi les tableaux 4.3, 4.4 et 4.5. La méthode d'ordre 1 est équivalente à la méthode d'Euler implicite.

1	1
	1

Tableau 4.3 : Tableau de Butcher pour la méthode de RadauIIA d'ordre 1

$\frac{1}{3}$	$\frac{5}{12}$	$-\frac{1}{12}$
1	$\frac{3}{4}$	$\frac{1}{4}$
	$\frac{3}{4}$	$\frac{1}{4}$

Tableau 4.4 : Tableau de Butcher pour la méthode de RadauIIA d'ordre 3

$\frac{4-\sqrt{6}}{10}$	$\frac{88-7\sqrt{6}}{360}$	$\frac{296-169\sqrt{6}}{1800}$	$\frac{-2+3\sqrt{6}}{225}$
$\frac{4+\sqrt{6}}{10}$	$\frac{296+169\sqrt{6}}{1800}$	$\frac{88+7\sqrt{6}}{360}$	$\frac{-2-3\sqrt{6}}{225}$
1	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$
	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$

Tableau 4.5 : Tableau de Butcher pour la méthode de RadauIIA d'ordre 5

Les méthodes de RadauIIA sont A-stable et elles sont d'ordre p = 2s - 1. Leur fonction de stabilité est l'approximation (s-1,s)-Padé.

Il a été mentionné que le point $c_s = 1$ présente un avantage : le dernier étage, à une étape de temps donnée, est équivalent à la solution au pas de temps suivant. Un autre avantage réside dans le fait que si $c_s = 1$, alors $a_{sj} = b_j$, $j = 1,\ldots,s$.

L'égalité (4.16) définit la propriété selon laquelle la méthode est *précise pour les équations raides* (Hairer & Wanner, 1996; Rang, 2008). Une proposition intéressante est alors énoncée par Hairer et Wanner :

Proposition :

Si une méthode de Runge-Kutta implicite est A-stable et satisfait au moins l'une des deux conditions suivantes :

$$a_{sj} = b_j \qquad j = 1,\ldots,s \tag{4.16}$$

$$a_{i1} = b_1 \qquad i = 1,\ldots,s \tag{4.17}$$

alors la fonction de stabilité est nulle lorsque z tend vers l'infini.

Cela indique, par l'équation (4.8), que la méthode est également L-stable. Les conditions (4.16) et (4.17) surviennent respectivement dans les familles des RadauIIA et des LobattoIIIC.

Les méthodes de Lobatto de type IIIC

Les méthodes de LobattoIIIC à s étages sont définies à l'aide des zéros du polynôme suivant :

$$\frac{d^{s-2}}{dx^{s-2}}\left(x^{s-1}(x-1)^{s-1}\right) \qquad 0 \le x \le 1 \tag{4.18}$$

Pour déterminer les coefficients a_{ij} et b_i, on impose que $a_{i1} = b_1$ pour i allant de 1 à s, puis on trouve les autres coefficients avec les hypothèses simplificatrices. Comme on impose l'égalité (4.17), on sait immédiatement que ces méthodes sont L-stables. La

propriété L-stable des méthodes de RadauIIA et LobattoIIIC est un avantage certain par rapport aux méthodes de Gauss lorsque les équations à intégrer sont raides.

Les méthodes de LobattoIIIC sont d'ordre p = 2s - 2. La fonction de stabilité est donnée par l'approximation (s-2,s)-Padé. Elles sont A-stables et L-stables. Voici donc les tableaux 4.6, 4.7 et 4.8 pour ces méthodes :

$$\begin{array}{c|cc} 0 & \frac{1}{2} & -\frac{1}{2} \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Tableau 4.6 : Tableau de Butcher pour la méthode de LobattoIIIC d'ordre 2

$$\begin{array}{c|ccc} 0 & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{6} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

Tableau 4.7 : Tableau de Butcher pour la méthode de LobattoIIIC d'ordre 4

$$\begin{array}{c|cccc} 0 & \frac{1}{12} & -\frac{\sqrt{5}}{12} & \frac{\sqrt{5}}{12} & -\frac{1}{12} \\ \frac{5-\sqrt{5}}{10} & \frac{1}{12} & \frac{1}{4} & \frac{10-7\sqrt{5}}{60} & \frac{\sqrt{5}}{60} \\ \frac{5+\sqrt{5}}{10} & \frac{1}{12} & \frac{10+7\sqrt{5}}{60} & \frac{1}{4} & -\frac{\sqrt{5}}{60} \\ 1 & \frac{1}{12} & \frac{5}{12} & \frac{5}{12} & \frac{1}{12} \\ \hline & \frac{1}{12} & \frac{5}{12} & \frac{5}{12} & \frac{1}{12} \end{array}$$

Tableau 4.8 : Tableau de Butcher pour la méthode de LobattoIIIC d'ordre 6

RadauIA, LobattoIIIA et LobattoIIIB

Ces trois méthodes sont mentionnées ici simplement pour indiquer qu'elles existent. Elles ont le même taux de convergence que les méthodes de RadauIIA et de

LobattoIIIC, mais présentent des aspects moins intéressants d'un point de vue numérique ou algorithmique. Les méthodes LobattoIIIA et LobattoIIIB ne sont pas L-stables, ce qui suggère un risque d'oscillations comme vu à la figure 4.4. La méthode RadauIA est L-stable, mais elle ne possède pas la propriété de précision pour les équations raides (équation (4.16)). Cela veut dire que le dernier étage de cette méthode n'est pas identique à la solution au pas de temps suivant. Cette solution doit donc être calculée par la suite, entraînant un coût de calcul supplémentaire.

Réduction d'ordre

Certaines méthodes de Runge-Kutta souffrent parfois du phénomène de réduction d'ordre. Cela se produit principalement lorsque l'équation à intégrer est raide. En fait, chaque étage d'une méthode RK possède un ordre de convergence propre à cet étage qui peut être différent de l'ordre globale de la méthode. La condition B(p) des hypothèses simplificatrices (4.13) indique l'ordre p de la méthode et la condition C(q) l'ordre maximal q de chaque étage. Si $q < p$ et que la méthode considérée est appliquée à une équation raide, alors l'ordre de la méthode peut diminuer de p à q.

Voici un exemple pour illustrer la situation avec les méthodes de RadauIIA. Soit l'équation différentielle :

$$\begin{aligned} y' &= -2\times 10^5 (y - \cos(t)) \\ y(t_0) &= 0 \qquad t_0 = 0 \le t \le 1{,}5 \end{aligned} \tag{4.28}$$

qui est similaire à l'équation (4.9), mais avec un comportement encore plus raide. L'équation est intégrée par la méthode de RadauIIA3 (2 étages, ordre global 3) avec successivement 20, 40, 80 pas de temps, etc. À chaque intégration, le pas de temps diminue d'un facteur 2 et selon l'équation (2.2) l'erreur absolue doit diminuer d'un facteur 8 (2^3). Or on constate que l'erreur diminue d'un facteur 4, ce qui représente un

taux de convergence de 2. La méthode souffre donc de la perte d'un ordre de convergence. Les résultats sont présentés au tableau 4.9.

Erreur absolue		Ordre de convergence
E1	3,1072E-9	
E2	7,7769E-10	1,9983
E3	1,9440E-10	2,0002
E4	4,8537E-11	2,0019
E5	1,2097E-11	2,0044
E6	3,0054E-12	2,0090
E7	7,4202E-13	2,0180
E8	1,8099E-13	2,0355
E9	4,3132E-14	2,0691
E10	9,8810E-15	2,1260
E11	2,1094E-15	2,2278
E12	3,1919E-16	2,7244
E13	9,7145E-17	1,7162

Tableau 4.9 : Valeurs de l'erreur absolue et du taux de convergence de la solution au problème (4.28) pour différentes discrétisations

Selon les hypothèses (4.13), on déduit le même phénomène. Pour B(p), on a :

$$B(p=3): \quad \frac{3}{4}\left(\frac{1}{3}\right)^2 + \frac{1}{4}(1)^2 = \frac{1}{12} + \frac{1}{4} = \frac{1}{3}$$

$$B(p=4): \quad \frac{3}{4}\left(\frac{1}{3}\right)^3 + \frac{1}{4}(1)^3 = \frac{1}{36} + \frac{1}{4} \neq \frac{1}{4}$$

et p = 3 puisque la condition B(4) n'est pas respectée. De même, pour C(η), l'hypothèse devient :

$$C(q=2):\quad \begin{cases} s=1: & \frac{5}{12}\left(\frac{1}{3}\right)^1 - \frac{1}{12}(1)^1 = \frac{5}{36} - \frac{3}{36} = \frac{1}{18} \\ s=2: & \text{équivalent à B(2)} \end{cases}$$

$$C(q=3):\quad \begin{cases} s=1: & \frac{5}{12}\left(\frac{1}{3}\right)^2 - \frac{1}{12}(1)^2 = \frac{5}{108} - \frac{9}{108} \neq \frac{1}{54} \\ s=2: & \text{équivalent à B(3)} \end{cases}$$

On obtient q = 2 puisque la condition C(3) n'est pas respectée. Avec ces hypothèses simplificatrices, on en déduit que la méthode RadauIIA3 perd un ordre de convergence lorsqu'elle est appliquée à une équation raide. En refaisant le test avec RadauIIA5, on obtient une perte de deux ordres. On retrouve dans Hairer et Wanner (1996) un tableau présentant les ordres pour les erreurs locales et globales de différentes méthodes discutées dans ce mémoire. Ces ordres de convergence, déterminés pour des équations raides uniquement, sont tirés d'une analyse dans laquelle le pas de temps h tend vers 0 alors que le facteur complexe $z = \lambda h$ tend vers l'infini, λ étant très grand (2×10^5 dans cet exemple-ci). L'idée sous-jacente est qu'on souhaite avoir un pas de temps beaucoup plus grand que λ^{-1}.

Stabilité des BDF

Les méthodes BDF ont également été abordées au premier chapitre. Maintenant, il s'agit de déterminer si elles sont plus adéquates que les RKI mentionnées ci-haut.

Soit la méthode (1.6) appliquée à l'équation test (1.1) :

$$\sum_{j=1}^{k} \frac{1}{j} \Delta^j y_{n+k} = z y_{n+k} \tag{4.19}$$

avec la formule récursive

$$\Delta^j y_{n+k} = \Delta^{j-1} y_{n+k} - \Delta^{j-1} y_{n+k-1} \quad , \quad \Delta^0 y_n = y_n$$

et où $z = \lambda h$. On développe la sommation, ce qui donne :

$$\alpha_k y_{n+k} + \alpha_{k-1} y_{n+k-1} + \cdots + \alpha_0 y_n = z y_{n+k} \tag{4.20}$$

Si on pose $y_j = \zeta^j$, alors l'équation (4.20), divisée par ζ^n, peut être réécrite de la façon suivante :

$$\alpha_k \zeta^k + \alpha_{k-1} \zeta^{k-1} + \cdots + \alpha_0 - z\zeta^k = \rho(\zeta) - z\sigma(\zeta) = 0 \tag{4.21}$$

L'équation (4.21) est l'équation caractéristique de la méthode (4.19) et cette méthode est stable si et seulement si toutes les racines de (4.21) sont inférieures ou égales à 1. Dans le cas d'une racine multiple, il faudra que celle-ci soit strictement inférieures à 1 (Hairer & Wanner, 1996). Les polynômes $\rho(\zeta)$ et $\sigma(\zeta)$ sont appelés les polynômes générateurs de la méthode (4.19). Le domaine de stabilité est :

$$S = \left\{ z \in \mathbb{C} \,\middle|\, |\zeta_j(z)| \le 1 \quad j = 1 \text{ à } k \right\} \tag{4.22}$$

L'équation caractéristique peut être apparentée à la fonction de stabilité pour les méthodes de Runge-Kutta. Pour un k élevé, il est difficile de calculer les racines de ce polynôme en fonction de z. Il est possible par contre d'évaluer les valeurs de z qui satisfont l'équation (4.21) :

$$z = \frac{\rho(\zeta)}{\sigma(\zeta)} = \frac{\alpha_k \zeta^k + \alpha_{k-1} \zeta^{k-1} + \cdots + \alpha_0}{\zeta^k} \tag{4.23}$$

On peut alors définir le critère de stabilité sur z en prenant $|\zeta| = 1$:

$$z = \sum_{j=1}^{k} \frac{1}{j} \left(1 - \frac{1}{\zeta} \right)^j \tag{4.24}$$

et

$$\frac{1}{\zeta} = e^{-i\theta} = \cos(\theta) - i\sin(\theta) \quad 0 \le \theta \le 2\pi$$

L'équation (4.24) définit alors une courbe orientée, paramétrée par θ, dont la zone à gauche de cette courbe est le domaine de stabilité (Hairer & Wanner, 1996). À la figure 4.5, on voit les courbes pour les méthodes BDF 1 à 6. Les courbes sont paramétrés dans le sens horaire des aiguilles d'une montre, donc le domaine de stabilité est à l'extérieur de la zone délimitée par la courbe. Tel que mentionné au premier chapitre, les méthodes BDF avec $k \geq 7$ sont instables. La courbe orientée correspondante n'engloberait aucun domaine de stabilité.

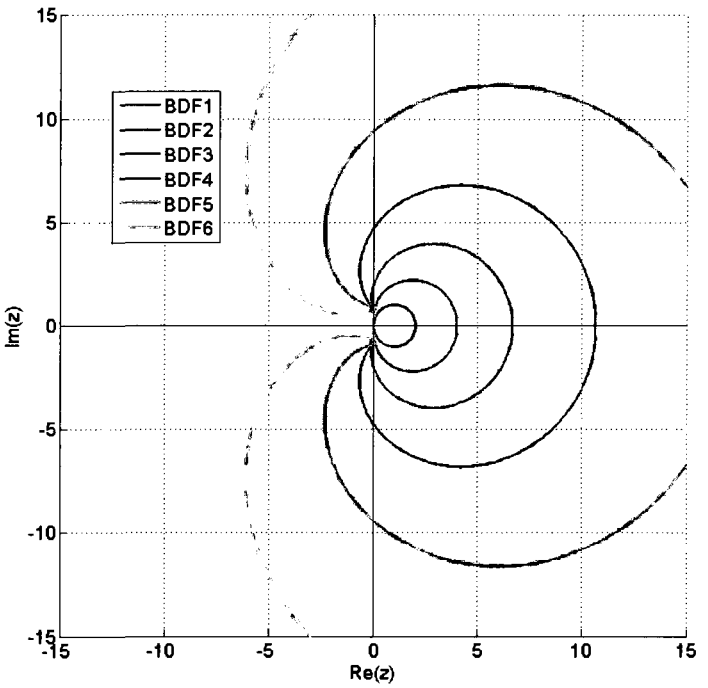

Figure 4.5 : Domaine de stabilité pour les BDF; le domaine de stabilité est à l'extérieur des courbes

La figure 4.6 présente un agrandissement autour de l'axe imaginaire afin de bien voir que les schémas BDF 1 (Euler implicite) et BDF 2 (schéma de Gear) sont A-stables. En effet, sur la figure 4.6, on voit que seules les courbes fermées représentant les schémas BDF 1 et BDF 2 ne traversent pas l'axe imaginaire vers la partie négative du plan complexe. Par contre, les schémas avec $3 \leq k \leq 6$ ne sont que conditionnellement stables, car leurs courbes fermées traversent l'axe imaginaire. De

plus à la figure 4.5, on voit que les schémas BDF 5 et 6 présentent de grandes zones à l'intérieur de la partie négative du plan complexe. Ces zones indiquent un critère à respecter sur la grandeur du pas de temps utilisé. Ceci est d'autant plus critique dans le cas où la partie imaginaire de z est beaucoup plus grande que la partie réelle. Ce cas peut être associé, pour l'équation de transport de chaleur (3.12), à un phénomène de convection dominant (Gresho, Sani, & Engelman, 1998).

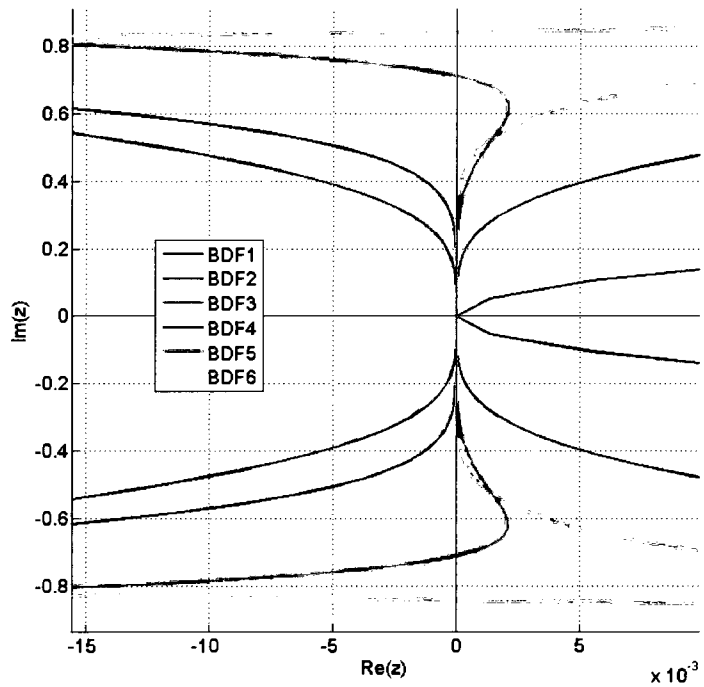

Figure 4.6 : Agrandissement autour de Re(z) = 0 pour le domaine de stabilité des BDF

On a vu que le schéma d'Euler implicite est L-stable. Voyons maintenant si le schéma de Gear est également L-stable. Il est le seul candidat des BDF, puisque les autres ne sont pas A-stables.

Le schéma de Gear (1.5) appliqué à l'équation test (1.1) devient :

$$\frac{3y_{n+1} - 4y_n + y_{n-1}}{2h} = \lambda y_{n+1} \tag{4.25}$$

On isole facilement y_{n+1} :

$$y_{n+1} = \frac{4y_n - y_{n-1}}{3 - 2z} \qquad z = \lambda h \tag{4.26}$$

La solution à l'étape n est $y_{n+1} = \xi y_n$ où ξ est l'amplification de la solution. En insérant cette solution dans (4.26), on obtient :

$$\xi^{n+1} y_0 = \frac{4\xi^n y_0 - \xi^{n-1} y_0}{3 - 2z} \tag{4.27}$$

que l'on divise par $\xi^{n-1} y_0$. On trouve le gain ξ, qui doit être égal ou inférieur à 1, en résolvant l'équation quadratique :

$$\xi = \frac{2 \pm \sqrt{1 + 2z}}{3 - 2z}$$

L'équation (4.8) indique que la méthode est L-stable si la limite de ξ tend vers 0 lorsque z tend vers l'infini, ce qui est observable ici. Le schéma de Gear est donc aussi L-stable. C'est le seul schéma BDF d'ordre plus grand que un à détenir la propriété de L-stabilité.

Les systèmes d'équations différentielles-algébriques

Les méthodes de Runge-Kutta ont été à l'origine développées pour résoudre des équations différentielles ordinaires (ÉDO), de la forme y' = f(y,t) (Hairer, Lubich, & Roche, 1989). Cependant, les ÉDO ne sont qu'un cas particulier d'une classe plus large, celle des équations différentielles-algébriques (ÉDA). Il est donc justifié de se demander si les méthodes de Runge-Kutta sont efficaces pour résoudre les ÉDA étant donné que plusieurs phénomènes physiques sont décrits par des ÉDA.

L'index d'une ÉDA est une mesure de la sensibilité des solutions aux perturbations ce cette équation (Hairer, Lubich, & Roche, 1989). De façon générale,

les ÉDA sont un cas limite des problèmes à perturbation singulière. On peut énoncer un problème à perturbation singulière ainsi :

$$\begin{aligned} y' &= f(y,z) \\ \varepsilon z' &= g(y,z) \end{aligned} \tag{4.28}$$

Si ε tend vers 0, le problème devient de plus en plus raide et à la limite on obtient le système réduit :

$$\begin{aligned} y' &= f(y,z) \\ 0 &= g(y,z) \end{aligned} \tag{4.29}$$

Ce système réduit est d'index 1 si la dérivée de g par rapport à z est non-nulle. Pour un système d'équations vectorielles, c'est le déterminant de la matrice jacobienne qui doit être non-nul. Dans le cas contraire, l'index est au minimum 2. La discrétisation par éléments finis de l'équation (3.12) nous conduit à un système d'ÉDO. Il n'en est pas de même pour les équations de Navier-Stokes que l'on retrouve à la deuxième partie de ce mémoire. Les équations de Navier-Stokes en régime incompressible sont un système d'ÉDA d'index 2 (Rang, 2008), ce qui représente un système de la forme :

$$\begin{aligned} y' &= f(y,z) \\ 0 &= g(y) \end{aligned} \tag{4.30}$$

Pour les équations de Navier-Stokes, la variable y sera le vecteur de vitesse du fluide et z, le multiplicateur de la contrainte g(y) = 0, la pression. Les détails sur la théorie des ÉDA dépassent le cadre de ce mémoire. La présentation sur ce sujet s'arrêtera donc ici.

Choix des méthodes à implémenter

Suite aux lectures des auteurs cités dans ce chapitre ainsi qu'à l'analyse de stabilité des méthodes mentionnées, il est clair que les méthodes de Runge-Kutta implicites comportent des avantages indéniables comparativement aux méthodes BDF.

Les RKI sont des méthodes auto-démarrantes (tout comme les RKE). Les BDF, quant à elles, nécessitent la solution à plusieurs pas de temps avant de démarrer. Au temps initial, le manque d'informations sur la solution aux étapes de temps précédentes conduit celui qui les utilise à démarrer l'intégration en temps avec une autre méthode d'ordre équivalente ou à prendre un schéma BDF qui nécessite moins d'étapes de temps, au prix d'un ordre de convergence plus bas. Ce démarrage à un ordre plus faible peut avoir des conséquences néfastes sur l'ordre du schéma utilisé pour produire l'intégration en temps voulue. Dans ce cas, on doit utiliser une BDF d'ordre plus bas avec un pas de temps Δt beaucoup plus faible que pour le reste de l'intégration en temps. Cela nécessite un programme qui permette de faire varier le pas de temps en cours de simulation.

Les RKI possèdent de meilleures propriétés de stabilité que les BDF. Comme mentionné auparavant, tous les schémas RKI d'ordre élevé (3 et plus) sont A-stable alors que les méthodes BDF ne le sont pas, à partir de l'ordre 3. De plus, seules les LobattoIIIA, LobattoIIIB et les méthodes de Gauss ne sont pas L-stables alors que les LobattoIIIC, RadauIA et RadauIIA le sont, tout comme la BDF d'ordre 2 (schéma de Gear). La propriété de L-stabilité est un avantage énorme lorsque les équations à intégrer sont raides, ce qui est le cas dans les phénomènes de couches limites où les nombres de Reynolds et de Péclet locaux varient beaucoup sur une courte distance. On doit donc assurer une bonne stabilité de l'intégration en temps pour ne pas prendre des pas de temps si petit au point de rendre une simulation irréaliste en termes de temps de calcul. Les méthodes L-stables sont également beaucoup plus fiables lorsque la simulation est réalisée sur une longue période de temps.

Pour ce qui est du choix des méthodes RKI, car il y en a plusieurs, les méthodes de LobattoIIIC et de RadauIIA sont à privilégier étant donné leur propriété de précision pour des équations raides (équation (4.16)), en plus de leur caractère L-stable. Cette propriété nous évite de recalculer la solution au pas de temps suivant puisqu'elle est identique à la solution du dernier étage (coefficient $c_s = 1$).

Schéma	Ordre s = nombre d'étages	A-stable	L-stable	Précis pour les équations raides
Gauss	2s	oui	non	non
RadauIA	2s – 1	oui	oui	non
RadauIIA	2s – 1	oui	**oui**	**oui**
LobattoIIIA	2s – 2	oui	non	non
LobattoIIIB	2s – 2	oui	non	non
LobattoIIIC	2s – 2	oui	**oui**	**oui**

Tableau 4.10 : Tableau résumé des différentes propriétés de stabilité des schémas RKI

En contrepartie, les méthodes RKI sont relativement complexe à implémenter, légèrement plus que les BDF. De plus, le nombre d'équations et d'inconnus sont multipliés par le nombre d'étages de la méthode, ce qui augmente rapidement le temps de calcul et le coût en mémoire. Une gestion efficace des structures de données est nécessaire. Ces inconvénients ont été pendant longtemps un frein à leur implémentation, mais il semblerait que la situation évolue avec la puissance informatique aujourd'hui accessible.

CHAPITRE 5 - ESSAIS DES MÉTHODES RKI SUR UN CODE D'ÉLÉMENTS FINIS UNIDIMENSIONNEL

Le cas test

Soit un domaine Ω = x =]0;1[définit dans $\mathbb{R}$. Soit T(t,x) une distribution de chaleur le long du domaine Ω. Cette distribution de chaleur est gouvernée par l'équation de la chaleur 1D adimensionnelle (3.12) ainsi que par les conditions frontières (de Dirichlet) suivantes :

$$T(t,0)=0 \quad \text{et} \quad T(t,1)=1 \tag{5.1}$$

Afin d'évaluer les taux de convergence des méthodes de Runge-Kutta implicites, la technique des solutions manufacturées est employée. Il s'agit de prescrire une distribution de température sur le domaine et de choisir le terme source f(t,x) de l'équation (3.12) en fonction de la distribution prescrite. Cette distribution est :

$$T(t,x)=(1-t^7)x+t^7x^7 \tag{5.2}$$

La distribution (5.2) indique qu'à t = 0, la température varie linéairement avec x, puis lorsque t augmente, la distribution de température ressemble à celle d'une couche limite. Considérant l'équation (3.12), le terme source f(t,x) est obtenu en calculant la dérivée de la température par rapport au temps et les dérivées première et deuxième de la température par rapport à l'espace :

$$\begin{aligned}
\frac{\partial T}{\partial t} &= -7t^6x+7t^6x^7 \\
\frac{\partial T}{\partial x} &= (1-t^7)+7t^7x^6 \\
\frac{\partial^2 T}{\partial x^2} &= 42t^7x^5
\end{aligned} \tag{5.3}$$

De (3.12) et (5.3), on tire f(t,x) :

$$f(t,x) = -7t^6x + 7t^6x^7 + (1-t^7) + 7t^7x^6 - \frac{42t^7x^5}{Pe} \tag{5.4}$$

Pour le nombre de Peclet, deux cas tests sont d'intérêt :

Pe = 1;

Pe = 10^6.

Dans le premier cas, le transfert de chaleur se produit autant par le phénomène de convection que par celui de diffusion. Dans le second cas, la convection est nettement prépondérante. Afin d'assurer la convergence du calcul dans ce second cas, il faut employer une méthode de stabilisation (voir chapitre 3).

La simulation

Un programme MATLAB a été mis au point pour cette expérience numérique. Ce programme résout la formulation variationnelle discrétisée et stabilisée (3.20). Le tableau 5.1 indique les pas de temps considérés ainsi que la taille des éléments considérés.

Discrétisation en temps								
Nombre de pas de temps	10	20	40	80	160	320	640	1280*
Δt	0,1	0,05	0,025	0,0125	0,00625	0,003125	0,0015625	0,00078125
Discrétisation en espace								
Nombre d'éléments	10		20		40		80	
h	0,1		0,05		0,025		0,0125	

* Uniquement pour BDF5 et BDF6.

Tableau 5.1 : Taille de discrétisation en temps et en espace pour les cas tests

Résolution du problème avec la forme résidu-correction

Le système d'équations est résolu sous la forme résidu-correction. Cette forme est tirée de la méthode itérative de Newton pour résoudre les systèmes d'équations non-linéaires. Soit le système non-linéaire suivant :

$$\mathbf{A(x)x = b} \tag{5.5}$$

La méthode de Newton peut être énoncée ainsi :

Partant d'une solution initiale $\mathbf{x}^0$ satisfaisant :

$$\mathbf{F}(\mathbf{x}^0) = \mathbf{b} - \mathbf{A}(\mathbf{x}^0)\mathbf{x}^0 \tag{5.6}$$

obtenir $\Delta\mathbf{x}$, la correction, telle que :

$$\mathbf{F}(\mathbf{x}^0 + \Delta\mathbf{x}) = \mathbf{0} \tag{5.7}$$

Ici, $\mathbf{F}(\mathbf{x})$ est le vecteur résidu du système d'équations. Une approximation de l'équation (5.7) s'obtient en considérant un développement de Taylor de degré 1 :

$$\mathbf{F}(\mathbf{x}^0 + \Delta\mathbf{x}) = \mathbf{F}(\mathbf{x}^0) + \left\{\frac{\partial \mathbf{F}(\mathbf{x}^0)}{\partial \mathbf{x}^0}\right\}\Delta\mathbf{x} + \cdots = \mathbf{0} \tag{5.8}$$

Le terme entre accolade est nommé la matrice tangente ou encore la matrice jacobienne. L'équation (5.8) représente donc un nouveau système, cette fois-ci linéaire, à résoudre à chaque itération :

$$\frac{\partial \mathbf{F}(\mathbf{x}^0)}{\partial \mathbf{x}^0}\Delta\mathbf{x} = -\mathbf{F}(\mathbf{x}^0) \tag{5.9}$$

Lorsqu'on obtient $\Delta\mathbf{x}$, on forme un nouveau vecteur solution, $\mathbf{x}^1$: $\mathbf{x}^1 = \mathbf{x}^0 + \Delta\mathbf{x}$. Le processus se répète jusqu'à ce que :

$$|\Delta\mathbf{x}| < \varepsilon \quad \text{et} \quad |\mathbf{F}(\mathbf{x})| < \varepsilon \tag{5.10}$$

On imposera que la correction ainsi que le résidu soient plus petits qu'une certaine tolérance pour assurer la convergence de la méthode de Newton vers la bonne solution. Si seulement l'une des deux conditions est respectée, on peut se retrouver dans un minimum local, condition à éviter.

La matrice tangente, au niveau élémentaire, est reconstruite numériquement, c'est-à-dire qu'on détermine chaque colonne de la matrice en perturbant tour à tour chaque ligne du vecteur solution **x** avec un terme δ. Ceci donne un résidu perturbé auquel on soustrait le résidu de la solution normale, le tout divisé par la perturbation δ. Le résultat donne une colonne de la matrice tangente. Autrement dit, la colonne A_j est donnée par :

$$A_j = \frac{\mathbf{F}(\hat{\mathbf{x}}) - \mathbf{F}(\mathbf{x})}{\delta} \quad \text{où } \hat{x}_i = \begin{cases} x_i + \delta & \text{si } i = j \\ x_i & \text{sinon} \end{cases}$$

et i représente une ligne du vecteur **x**. À noter que pour le cas test mentionné dans ce chapitre (équation de convection-diffusion), le système d'équations est linéaire, ce qui implique que la matrice tangente n'est calculée qu'une seule fois, car elle ne varie pas d'une itération à l'autre. Les itérations suivantes ne font que corriger en rafale la solution recherchée jusqu'à ce que les conditions (5.10) soient satisfaites.

Départager l'erreur en temps de l'erreur en espace

La solution numérique de l'équation de notre cas test contiendra un terme d'erreur dû à la méthode numérique choisie. Le calcul de l'erreur, E_T, en norme $L^2(\Omega)$ se fait selon l'équation (2.9) :

$$\|E_T\|_{0,\Omega} = \left(\int_\Omega \left(T^{ex} - T^N \right)^2 d\Omega \right)^{1/2} \tag{5.11}$$

T^{ex} est la solution exacte (aux nœuds du maillage) prescrite au départ par (5.2) et T^N est la solution numérique approximée. Cependant, T^N est entachée de deux sources d'erreurs : l'erreur spatiale due au choix de l'élément de référence (la base de Lagrange), et l'erreur temporelle due au choix de la méthode d'intégration en temps. Il est utile d'introduire T^h, une solution numérique très précise en temps. Par inégalité triangulaire, on peut affirmer :

$$\left\|E_T\right\|_{0,\Omega} = \left\|T^{ex} - T^N\right\|_{0,\Omega} \leq \underbrace{\left\|T^{ex} - T^h\right\|_{0,\Omega}}_{O(h^p)} + \underbrace{\left\|T^h - T^N\right\|_{0,\Omega}}_{O(\Delta t^q)} \tag{5.12}$$

Pour évaluer T^h, on calcule plusieurs solutions successives avec $\Delta t_{i+1} = \Delta t_i / 2$, puis on utilise l'extrapolation de Richardson. Ce procédé permet d'estimer une solution très près d'être exacte en temps lorsque Δt est petit. L'extrapolation (récursive) de Richardson va comme ceci :

Soit une suite de solutions $\Psi = \{T^N_1, T^N_2, T^N_3, \ldots\}$ pour laquelle $\Delta t_{i+1} = \Delta t_i / 2$. Alors on peut utiliser l'extrapolation de Richardson de façon récursive pour déterminer $T_{12} = aT_1 + bT_2$, tout comme $T_{23} = aT_2 + bT_3$. Ensuite, $T_{123} = a'T_{12} + b'T_{23}$, etc.

Les valeurs a et b sont :

$$a = \frac{-1}{\left(\Delta t_1 / \Delta t_2\right)^k - 1} \qquad b = \frac{\left(\Delta t_1 / \Delta t_2\right)^k}{\left(\Delta t_1 / \Delta t_2\right)^k - 1} \tag{5.13}$$

En remplaçant Δt_2 par $\Delta t_1 / 2$, on trouve :

$$a = \frac{-1}{2^k - 1} \qquad b = \frac{2^k}{2^k - 1}$$

et k prend tour à tour une valeur égale aux puissances de Δt dans l'expression de l'erreur. Cette suite de valeurs de k peut être vérifiée avec un développement de Taylor du schéma. Par exemple, pour Euler implicite :

$$T^{n+1} - T^n = \Delta t G(T^{n+1}) \tag{5.14}$$

où G(T) représente la discrétisation spatiale reliée à la méthode d'éléments finis. Si on suppose qu'il n'y a pas d'erreur en espace (pour alléger la démonstration), alors le développement en série de Taylor pour T^{n+1} est :

$$T^{n+1} = T^n + \frac{\partial T}{\partial t}\Delta t + \frac{1}{2}\frac{\partial^2 T}{\partial t^2}\Delta t^2 + \cdots \tag{5.15}$$

Avec (5.15) insérée dans (5.14) :

$$\frac{1}{2}\frac{\partial^2 T}{\partial t^2}\Delta t + \cdots = \frac{\partial T}{\partial t} + G(T^{n+1}) \tag{5.16}$$

L'équation (5.16) doit être consistante avec l'équation originale (3.12) et elle l'est puisque, lorsque $\Delta t \rightarrow 0$ (et $T^{n+1} \rightarrow T$), on se retrouve avec le membre de droite de (5.16), qui est l'équation de départ. La différence entre les deux donne l'erreur et est simplement :

$$E_T = \frac{1}{2}\frac{\partial^2 T}{\partial t^2}\Delta t + \cdots \tag{5.17}$$

La puissance sur Δt du terme d'erreur (5.17) est $k = 1$ et les termes suivants ont des puissances $k = 2$, $k = 3$, etc., composant ainsi la suite des k à utiliser dans l'extrapolation de Richardson. Ainsi, T_1, T_2 et T_0 sont d'ordre 1, T_{12} et T_{23} sont d'ordre 2, T_{123} d'ordre 3, etc. Comme on utilise sept pas de temps différents et successivement divisés par deux, alors la solution T^h presque exacte en temps sera d'ordre sept, ce qui est bien plus élevé que l'ordre 2 ou 3 pour la discrétisation spatiale.

Cette méthode permet donc d'évaluer en une seule étape l'ordre de convergence en temps (q) et l'ordre de convergence en espace (p).

Résultats

Les pages qui suivent contiennent plusieurs tableaux recueillant des mesures de l'erreur en norme L^2. Dans chaque tableau, le taux est calculé selon l'équation (2.3) en impliquant les deux solutions les plus fines en temps pour le taux en temps et les deux solutions les plus fines en espace pour le taux en espace. Chaque colonne des tableaux est à une certaine discrétisation temporelle et chaque ligne est à une certaine discrétisation spatiale. Quelques graphiques sont également montrés pour illustrer les

cas instables. Sur ces graphiques, le trait (bleu) est la solution exacte et les ronds (rouges) la solution numérique.

À noter que les simulations présentées ici sont celles qui ont été réalisées avec les éléments quadratiques. Donc, pour chaque méthode, on s'attend à un taux de convergence de 3 en espace. Les résultats avec les éléments linéaires ont été similaires. Quelques cas avec des éléments linéaires sont également présentés.

Chaque cas est d'abord identifié par son nombre de Péclet, soit 1 ou 1 million. Ensuite, pour chacun de ces cas, on donne la norme de l'erreur en temps et la norme de l'erreur en espace. Ces combinaisons, sont regroupées en quatre sous-tableaux. Chaque sous-tableau présente les valeurs de l'erreur sur la solution en fonction des discrétisations spatiales (h) et temporelles (Δt). La colonne identifiée « Taux q » contient le taux de convergence de la norme en temps, q, et ce taux est calculé avec les deux dernières valeurs de l'erreur en temps selon l'équation (2.3). La ligne identifiée « Taux p » contient le taux de convergence de la norme en espace, p, calculé avec les dernières valeurs de l'erreur en espace selon l'équation (2.3). Ce tableau permet donc de voir quelles combinaisons de discrétisations temps / espace posent problème.

Il est à noter que si une discrétisation en temps particulière (une valeur de Δt précise) cause l'instabilité d'un schéma, l'extrapolation de Richardson ne sera plus valide et on ne pourra pas départager l'erreur en espace de l'erreur en temps.

Pe = 1								
Norme en temps								
h \ Δt	0,1	0,05	0,025	0,0125	0,00625	0,003125	0,0015625	**Taux q**
0,1	4,8492E-02	2,5461E-02	1,3042E-02	6,5999E-03	3,3198E-03	1,6648E-03	8,3367E-04	**0,9978**
0,05	4,8493E-02	2,5462E-02	1,3042E-02	6,6000E-03	3,3198E-03	1,6649E-03	8,3368E-04	**0,9978**
0,025	4,8493E-02	2,5462E-02	1,3042E-02	6,6000E-03	3,3198E-03	1,6649E-03	8,3368E-04	**0,9978**
0,0125	4,8493E-02	2,5462E-02	1,3042E-02	6,6000E-03	3,3198E-03	1,6649E-03	8,3368E-04	**0,9978**
Norme en espace								
h \ Δt	0,1000	0,0500	0,0250	0,0125	0,00625	0,003125	0,0015625	
0,1	3,9949E-04	3,9949E-04	3,9949E-04	3,9949E-04	3,9949E-04	3,9949E-04	3,9949E-04	
0,05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222E-05	
0,025	6,2867E-06	6,2867E-06	6,2867E-06	6,2867E-06	6,2867E-06	6,2867E-06	6,2867E-06	
0,0125	7,8611E-07	7,8611E-07	7,8611E-07	7,8611E-07	7,8611E-07	7,8611E-07	7,8611E-07	
Taux p	**2,9995**	**2,9995**	**2,9995**	**2,9995**	**2,9995**	**2,9995**	**2,9995**	
Pe = 10^6								
Norme en temps								
h \ Δt	0,1	0,05	0,025	0,0125	0,00625	0,003125	0,0015625	**Taux q**
0,1	1,2603E-01	6,3080E-02	3,1490E-02	1,5724E-02	7,8554E-03	3,9259E-03	1,9625E-03	**1,0003**
0,05	1,3070E-01	6,5220E-02	3,2507E-02	1,6218E-02	8,0992E-03	4,0469E-03	2,0228E-03	**1,0005**
0,025	1,3282E-01	6,6182E-02	3,2961E-02	1,6439E-02	8,2075E-03	4,1006E-03	2,0495E-03	**1,0006**
0,0125	1,3380E-01	6,6624E-02	3,3169E-02	1,6539E-02	8,2567E-03	4,1250E-03	2,0617E-03	**1,0006**
Norme en espace								
h \ Δt	0,1000	0,0500	0,0250	0,0125	0,00625	0,003125	0,0015625	
0,1	3,9413E-04	3,9413E-04	3,9413E-04	3,9413E-04	3,9413E-04	3,9413E-04	3,9413E-04	
0,05	5,0405E-05	5,0405E-05	5,0405E-05	5,0405E-05	5,0405E-05	5,0405E-05	5,0405E-05	
0,025	6,3685E-06	6,3685E-06	6,3685E-06	6,3685E-06	6,3685E-06	6,3685E-06	6,3685E-06	
0,0125	8,0017E-07	8,0017E-07	8,0017E-07	8,0017E-07	8,0017E-07	8,0017E-07	8,0017E-07	
Taux p	**2,9926**	**2,9926**	**2,9926**	**2,9926**	**2,9926**	**2,9926**	**2,9926**	

Tableau 5.2 : Calcul d'erreur et taux de convergence en temps et en espace pour la méthode RadauIIA1

Pour RadauIIA1, nous avons dans les deux cas tests un taux de 1 en temps, ainsi que le taux en espace de 3, relié aux éléments quadratiques. Le schéma que l'on connaît mieux sous le nom d'Euler implicite est donc stable (L-stable).

Pe = 1								
Norme en temps								
h \ Δt	0,1	0,05	0,025	0,0125	0,00625	0,003125	0,0015625	**Taux q**
0,1	1,0570E-03	1,4626E-04	1,9265E-05	2,4755E-06	3,1397E-07	3,9544E-08	4,9623E-09	**2,9944**
0,05	1,0570E-03	1,4626E-04	1,9265E-05	2,4755E-06	3,1397E-07	3,9544E-08	4,9623E-09	**2,9944**
0,025	1,0570E-03	1,4626E-04	1,9265E-05	2,4755E-06	3,1397E-07	3,9544E-08	4,9623E-09	**2,9944**
0,0125	1,0570E-03	1,4626E-04	1,9265E-05	2,4755E-06	3,1397E-07	3,9544E-08	4,9623E-09	**2,9944**
Norme en espace								
h \ Δt	0,1000	0,0500	0,0250	0,0125	0,00625	0,003125	0,0015625	
0,1	3,9949E-04	3,9949E-04	3,9949E-04	3,9949E-04	3,9949E-04	3,9949E-04	3,9949E-04	
0,05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222E-05	
0,025	6,2867E-06	6,2867E-06	6,2867E-06	6,2867E-06	6,2867E-06	6,2867E-06	6,2867E-06	
0,0125	7,8611E-07	7,8611E-07	7,8611E-07	7,8611E-07	7,8611E-07	7,8611E-07	7,8611E-07	
Taux p	**2,9995**	**2,9995**	**2,9995**	**2,9995**	**2,9995**	**2,9995**	**2,9995**	
Pe = 10^6								
Norme en temps								
h \ Δt	0,1	0,05	0,025	0,0125	0,00625	0,003125	0,0015625	**Taux q**
0,1	7,7908E-04	9,6343E-05	1,1963E-05	1,4899E-06	1,8589E-07	2,3214E-08	2,9004E-09	**3,0007**
0,05	7,7442E-04	9,5964E-05	1,1931E-05	1,4870E-06	1,8559E-07	2,3181E-08	2,8965E-09	**3,0006**
0,025	7,7779E-04	9,6655E-05	1,2036E-05	1,5013E-06	1,8745E-07	2,3418E-08	2,9264E-09	**3,0004**
0,0125	7,8133E-04	9,7284E-05	1,2127E-05	1,5134E-06	1,8901E-07	2,3616E-08	2,9513E-09	**3,0003**
Norme en espace								
h \ Δt	0,1000	0,0500	0,0250	0,0125	0,00625	0,003125	0,0015625	
0,1	3,9413E-04	3,9413E-04	3,9413E-04	3,9413E-04	3,9413E-04	3,9413E-04	3,9413E-04	
0,05	5,0405E-05	5,0405E-05	5,0405E-05	5,0405E-05	5,0405E-05	5,0405E-05	5,0405E-05	
0,025	6,3685E-06	6,3685E-06	6,3685E-06	6,3685E-06	6,3685E-06	6,3685E-06	6,3685E-06	
0,0125	8,0017E-07	8,0017E-07	8,0017E-07	8,0017E-07	8,0017E-07	8,0017E-07	8,0017E-07	
Taux p	**2,9926**	**2,9926**	**2,9926**	**2,9926**	**2,9926**	**2,9926**	**2,9926**	

Tableau 5.3 : Calcul d'erreur et taux de convergence en temps et en espace pour la méthode RadauIIA3

Pe = 1								
Norme en temps								
h \ Δt	0,1	0,05	0,025	0,0125	0,00625	0,003125	0,0015625	**Taux q**
0,1	7,4197E-06	3,0574E-07	1,1672E-08	4,2020E-10	1,4466E-11	4,7508E-13	1,5247E-14	**4,9616**
0,05	7,4198E-06	3,0572E-07	1,1669E-08	4,1987E-10	1,4461E-11	4,7681E-13	1,5413E-14	**4,9512**
0,025	7,4198E-06	3,0572E-07	1,1669E-08	4,1985E-10	1,4459E-11	4,7659E-13	1,5377E-14	**4,9539**
0,0125	7,4198E-06	3,0572E-07	1,1669E-08	4,1985E-10	1,4459E-11	4,7654E-13	1,5373E-14	**4,9541**
Norme en espace								
h \ Δt	0,1000	0,0500	0,0250	0,0125	0,00625	0,003125	0,0015625	
0,1	3,9949E-04	3,9949E-04	3,9949E-04	3,9949E-04	3,9949E-04	3,9949E-04	3,9949E-04	
0,05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222E-05	
0,025	6,2867E-06	6,2867E-06	6,2867E-06	6,2867E-06	6,2867E-06	6,2867E-06	6,2867E-06	
0,0125	7,8611E-07	7,8611E-07	7,8611E-07	7,8611E-07	7,8611E-07	7,8611E-07	7,8611E-07	
Taux p	**2,9995**	**2,9995**	**2,9995**	**2,9995**	**2,9995**	**2,9995**	**2,9995**	
Pe = 10^6								
Norme en temps								
h \ Δt	0,1	0,05	0,025	0,0125	0,00625	0,003125	0,0015625	**Taux q**
0,1	1,4303E-06	4,5833E-08	1,4480E-09	4,5557E-11	1,4312E-12	4,8409E-14	1,6886E-15	**4,8414**
0,05	1,8395E-06	5,9449E-08	1,8782E-09	5,8974E-11	1,8527E-12	6,1987E-14	2,1207E-15	**4,8693**
0,025	2,0789E-06	6,7467E-08	2,1364E-09	6,7085E-11	2,1055E-12	6,9555E-14	2,3455E-15	**4,8902**
0,0125	2,2068E-06	7,1697E-08	2,2740E-09	7,1467E-11	2,2424E-12	7,3659E-14	2,4635E-15	**4,9021**
Norme en espace								
h \ Δt	0,1000	0,0500	0,0250	0,0125	0,00625	0,003125	0,0015625	
0,1	3,9413E-04	3,9413E-04	3,9413E-04	3,9413E-04	3,9413E-04	3,9413E-04	3,9413E-04	
0,05	5,0405E-05	5,0405E-05	5,0405E-05	5,0405E-05	5,0405E-05	5,0405E-05	5,0405E-05	
0,025	6,3685E-06	6,3685E-06	6,3685E-06	6,3685E-06	6,3685E-06	6,3685E-06	6,3685E-06	
0,0125	8,0017E-07	8,0017E-07	8,0017E-07	8,0017E-07	8,0017E-07	8,0017E-07	8,0017E-07	
Taux p	**2,9926**	**2,9926**	**2,9926**	**2,9926**	**2,9926**	**2,9926**	**2,9926**	

Tableau 5.4 : Calcul d'erreur et taux de convergence en temps et en espace pour la méthode RadauIIA5

Pour RadauIIA3, encore une fois le schéma est très stable et présente un taux optimal de 3 pour les deux cas tests.

Pour RadauIIA5, les taux en temps n'affichent pas tout à fait le taux attendu de 5 car nous atteignons des erreurs de l'ordre de la précision machine ($\approx 2{,}2 \times 10^{-16}$). On voit bien que le taux est de 5 si on calcule le taux avec les discrétisations temporelles moins fines. Cette méthode est en fait stable et très précise.

Pour les autres méthodes RKI ainsi que pour les schémas BDF2 à BDF4, seules les erreurs en temps et les taux de convergence appropriés sont donnés, et ce uniquement pour le maillage le plus fin (h = 1 / 80). On présente par la suite les tableaux complets pour les schémas BDF5 et BDF6, car ces deux méthodes présentent des instabilités.

Norme en temps								
h \ Δt	0,1	0,05	0,025	0,0125	0,00625	0,003125	0,0015625	**Taux q**
Pe = 1								
0,0125	2,1505E-02	6,6598E-03	1,8790E-03	5,0307E-04	1,3069E-04	3,3368E-05	8,4358E-06	**1,9839**
Pe = 10^6								
0,0125	2,2116E-02	5,4627E-03	1,3568E-03	3,3815E-04	8,4417E-05	2,1090E-05	5,2706E-06	**2,0005**

Tableau 5.5 : Calcul d'erreur et taux de convergence en temps pour la méthode LobattoIIIC2

Pour LobattoIIIC2, on obtient le taux de 2 en temps (et 3 en espace même si cela n'est pas affiché ici).

Norme en temps								
h \ Δt	0,1	0,05	0,025	0,0125	0,00625	0,003125	0,0015625	**Taux q**
Pe = 1								
0,0125	2,3726E-04	2,0297E-05	1,6185E-06	1,2158E-07	8,6957E-09	5,9878E-10	3,9816E-11	**3,9106**
Pe = 10^6								
0,0125	8,0869E-05	5,4577E-06	3,5277E-07	2,2383E-08	1,4082E-09	8,8266E-11	5,5246E-12	**3,9979**

Tableau 5.6 : Calcul d'erreur et taux de convergence en temps pour la méthode LobattoIIIC4

Pour LobattoIIIC4, le taux est bien de 4 en temps.

Norme en temps								
h \ Δt	0,1	0,05	0,025	0,0125	0,00625	0,003125	0,0015625	**Taux q**
Pe = 1								
0,0125	1,8519E-06	6,6304E-08	2,1069E-09	6,1880E-11	1,7297E-12	4,7531E-14	1,0507E-15	**5,4994**
Pe = 10^6								
0,0125	2,2802E-07	4,3296E-09	8,1994E-11	1,5417E-12	4,8219E-14	1,2756E-14	3,8334E-16	**5,0565**

Tableau 5.7 : Calcul d'erreur et taux de convergence en temps pour la méthode LobattoIIIC6

Pour LobattoIIIC6, le taux théorique est de 6, mais le taux observé dans les deux cas tests est plus faible. On pourrait croire au même phénomène que pour RadauIIA5 à propos de la précision machine, mais en fait, si on calcule le taux avec les discrétisation temporelle $\Delta t = 0,05$ et $\Delta t = 0,025$, on obtient :

$$\frac{\log\left(\frac{6,6304 \cdot 10^{-8}}{2,1069 \cdot 10^{-9}}\right)}{\log(2)} = 4,9759$$

Le taux est seulement de 5 pour ces discrétisations également. Pour Pe = 10^6 par contre, le taux est mieux à ces discrétisations, mais pas optimal :

$$\frac{\log\left(\frac{4,3296 \cdot 10^{-9}}{8,1994 \cdot 10^{-11}}\right)}{\log(2)} = 5,7226$$

Une unité d'ordre a donc été perdue. Pour les essais détaillés ici, le taux est meilleur pour le cas test hautement convectif. Cependant, entre $\Delta t = 0,00625$ et $\Delta t = 0,003125$, l'erreur n'a presque pas diminué. Il est possible que l'on capture presque la solution (5.2), qui est de degré 7 en temps, avec un schéma d'ordre 6. Il serait intéressant de tester une solution similaire à (5.2), mais de degré 8 ou même 9 en temps. La possibilité évoquée ci-haut ne se présenterait peut-être pas avec une telle solution prescrite.

Norme en temps								
h \ Δt	0,1	0,05	0,025	0,0125	0,00625	0,003125	0,0015625	**Taux q**
Pe = 1								
0,0125	1,3918E-02	4,0654E-03	1,1005E-03	2,8638E-04	7,3052E-05	1,8448E-05	4,6354E-06	**1,9927**
Pe = 10^6								
0,0125	3,6996E-02	1,0779E-02	2,9281E-03	7,6429E-04	1,9531E-04	4,9372E-05	1,2412E-05	**1,9920**

Tableau 5.7 : Calcul d'erreur et taux de convergence en temps pour la méthode BDF2

Pour la méthode BDF2 (Schéma de Gear), on retrouve bien l'ordre optimal de 2. Les taux en espace sont toujours 3 (ce taux n'est pas affiché au tableau ci-haut).

Norme en temps								
h \ Δt	0,1	0,05	0,025	0,0125	0,00625	0,003125	0,0015625	**Taux q**
Pe = 1								
0,0125	4,1046E-03	6,3164E-04	8,7333E-05	1,1472E-05	1,4698E-06	1,8599E-07	2,3391E-08	**2,9912**
Pe = 10^6								
0,0125	1,1535E-02	1,8223E-03	2,5536E-04	3,3769E-05	4,3408E-06	5,5020E-07	6,9254E-08	**2,9900**

Tableau 5.8 : Calcul d'erreur et taux de convergence en temps pour la méthode BDF3

Pour BDF3, les taux sont de 3 dans les deux cas tests.

Norme en temps								
h \ Δt	0,1	0,05	0,025	0,0125	0,00625	0,003125	0,0015625	**Taux q**
Pe = 1								
0,0125	1,0807E-03	8,3051E-05	5,7194E-06	3,7467E-07	2,3966E-08	1,5152E-09	9,5334E-11	**3,9904**
Pe = 10^6								
0,0125	3,3798E-03	2,6998E-04	1,8930E-05	1,2509E-06	1,5924E-07	5,1089E-09	3,2083E-10	**3,9931**

Tableau 5.9 : Calcul d'erreur et taux de convergence en temps pour la méthode BDF4

Pour BDF4, les taux sont de 4. Pour BDF 5, on commence à expérimenter des instabilités. Il a donc été jugé qu'il serait mieux d'insérer tout le tableau des erreurs et des taux de convergence pour voir précisément quelles situations posent problème. Il en sera de même pour BDF6.

Pe = 1									
Norme en temps									
Δt / h	0,1	0,05	0,025	0,0125	0,00625	0,0031	0,00156	0,00078125	**Taux q**
0,1	2,2460E-04	8,0780E-06	2,6943E-07	8,6853E-09	2,7563 E-10	8,7599 E-12	3,6667 E-13	1,4303E-14	**4,6801**
0,05	2,2461E-04	8,0782E-06	2,6943E-07	8,6854E-09	2,7563 E-10	8,7598 E-12	3,6638 E-13	1,4277E-14	**4,6816**
0,025	2,2461E-04	8,0782E-06	2,6943E-07	8,6855E-09	2,7563 E-10	8,7603 E-12	3,6690 E-13	1,4316E-14	**4,6797**
0,0125	2,2461E-04	8,0782E-06	2,6943E-07	8,6855E-09	2,7563 E-10	8,7600 E-12	3,6670 E-13	1,4298E-14	**4,6807**
Norme en espace									
Δt / h	0,1000	0,0500	0,0250	0,0125	0,00625	0,003125	0,0015625	0,00078	
0,1	3,9949E-04	3,9949E-04	3,9949E-04	3,9949E-04	3,9949E-04	3,9949E-04	3,9949E-04	3,9949 E-04	
0,05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222 E-05	
0,025	6,2867E-06	6,2867E-06	6,2867E-06	6,2867E-06	6,2867E-06	6,2867E-06	6,2867E-06	6,2867 E-06	
0,0125	7,8611E-07	7,8611E-07	7,8611E-07	7,8611E-07	7,8611E-07	7,8611E-07	7,8611E-07	7,8611 E-07	
Taux p	**2,9995**	**2,9995**	**2,9995**	**2,9995**	**2,9995**	**2,9995**	**2,9995**	**2,9995**	

Tableau 5.10a : Calcul d'erreur et taux de convergence en temps et en espace pour la méthode BDF5 -- Pe = 1

Examinons le cas Pe = 1. Pour BDF5 (tableau 5.10a) et BDF6 (tableau 5.11a), on retrouve le taux de 3 en espace, ce qui est attendu. Pour le taux en temps, on s'éloigne quelque peu des valeurs théoriques de 5 et 6, respectivement. Cependant, si on calcule le taux de convergence en temps avec les discrétisations temporelles plus grossières, on retrouve les taux attendus. Il se produit donc sensiblement le même phénomène qu'avec RadauIIA5, où le fait que la valeur de l'erreur soit près de la précision machine résulte en une mauvaise évaluation du taux de convergence. De plus, une erreur de 10^{-14} est extrêmement faible, donc la méthode est adéquate pour Pe = 1.

Pe = 10^6									
Norme en temps									
Δt / h	0,1	0,05	0,025	0,0125	0,00625	0,003125	0,00156	0,00078	**Taux q**
0,1	7,5474E-04	2,8386 E-05	9,6185 E-07	3,1246E-08	9,9490 E-10	3,1470 E-11	1,0807 E-12	3,6693E-14	**4,8804**
0,05	7,9274E-04	2,9955 E-05	1,0188 E-06	3,3764E-08	1,0529 E-09	3,3306 E-11	1,1384 E-12	3,8481E-14	**4,8867**
0,025	8,1064E-04	3,0703 E-05	1,0578 E-06	6,8932E-06	6,7583 E-06	8,7487 E-11	7,9203 E-11	7,9150E-11	**0,0010**
0,013 5	2,6177E+02	2,6177 E+02	2,6177 E+02	2,6177E+02	2,8720 E+02	1,9307 E+05	2,6177 E+02	2,6177E+02	**0,0000**
Norme en espace									
Δt / h	0,1000	0,0500	0,0250	0,0125	0,00625	0,003125	0,0015625	0,00078	
0,1	3,9413 E-04	3,9413E-04	3,9413E-04	3,9413E-04	3,9413 E-04	3,9413E-04	3,9413E-04	3,9413 E-04	
0,05	5,0405 E-05	5,0405E-05	5,0405E-05	5,0405E-05	5,0405 E-05	5,0405E-05	5,0405E-05	5,0405 E-05	
0,025	6,3685 E-06	6,3685E-06	6,3685E-06	6,3685E-06	6,3685 E-06	6,3685E-06	6,3685E-06	6,3685 E-06	
0,0125	2,6177 E+02	2,6177E+02	2,6177E+02	2,6177E+02	2,6177 E+02	2,6177E+02	2,6177E+02	2,6177 E+02	
Taux p	**-25,293**	**-25,293**	**-25,293**	**-25,293**	**-25,293**	**-25,293**	**-25,293**	**-25,293**	

Tableau 5.10b : Calcul d'erreur et taux de convergence en temps et en espace pour la méthode BDF5 -- Pe = 10^6

Pour Pe = 10^6 (tableaux 5.10b et 5.11b), on obtient des comportements aberrants. Pour BDF5 avec h = 0,025, l'erreur ne tend pas vers 0 lorsque $\Delta t \rightarrow 0$ (taux nul). En fait, à chaque division de Δt, l'erreur semble ne pas diminuer d'un facteur constant. Ceci est un indice de la présence d'une faible instabilité, non perceptible graphiquement. Avec h = 0,0125, l'instabilité est claire. On peut voir cette instabilité à figures 5.1. En effet, cette figure montre la distribution de température donnée par la solution (5.2) (trait bleu) et la distribution de température numérique (cercles rouges) obtenue de la méthode des éléments finis (éléments quadratiques) couplée au schéma d'intégration en temps BDF5. Le domaine decalcul va de x = 0 à x = 1 et la solution affichée est pour le temps final, soit t = 1.

Pe = 1									
Norme en temps									
Δt / h	0,1	0,05	0,025	0,0125	0,00625	0,003125	0,0015625	0,00078125	**Taux q**
0,1	2,7840 E-05	4,4119 E-07	6,8006E-09	1,0633 E-10	1,7156E-12	1,0336E-13	9,0827E-14	2,7819E-15	**5,0290**
0,05	2,7840 E-05	4,4120 E-07	6,8007E-09	1,0633 E-10	1,7164E-12	1,0418E-13	9,1434E-14	2,8053E-15	**5,0265**
0,025	2,7840 E-05	4,4120 E-07	6,8007E-09	1,0633 E-10	1,7156E-12	1,0378E-13	9,0645E-14	2,7752E-15	**5,0296**
0,0125	2,7840 E-05	4,4120 E-07	6,8007E-09	1,0633 E-10	1,7163E-12	1,0429E-13	9,1288E-14	2,7992E-15	**5,0273**
Norme en espace									
Δt / h	0,1000	0,0500	0,0250	0,0125	0,00625	0,003125	0,0015625	0,00078125	
0,1	3,9949 E-04	3,9949E-04	3,9949E-04	3,9949E-04	3,9949E-04	3,9949E-04	3,9949E-04	3,9949E-04	
0,05	5,0222 E-05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222E-05	5,0222E-05	
0,025	6,2867 E-06	6,2867E-06	6,2867E-06	6,2867E-06	6,2867E-06	6,2867E-06	6,2867E-06	6,2867E-06	
0,0125	7,8611 E-07	7,8611E-07	7,8611E-07	7,8611E-07	7,8611E-07	7,8611E-07	7,8611E-07	7,8611E-07	
Taux p	**2,9995**	**2,9995**	**2,9995**	**2,9995**	**2,9995**	**2,9995**	**2,9995**	**2,9995**	

Tableau 5.11a : Calcul d'erreur et taux de convergence en temps et en espace pour la méthode BDF6 -- Pe = 1

La figure 5.2 présente des instabilités encore plus grandes lorsque h = 0,00625 (les valeurs de l'erreur en temps et en espace pour la discrétisation h = 0,00625 et Δt = 0,00625 ne sont pas présentes au tableau 5.10).

Pour la méthode BDF6 avec h = 0,025, les instabilités sont déjà très présentes comme on peut le voir aux figures 5.3 et 5.4. Fait étonnant, c'est uniquement pour Δt = 0,0125 et Δt = 0,00625 que les instabilités sont mesurables. Une discrétisation plus fine en temps sera stable. La figure 5.5 illustre le phénomène. Sur cette figure, on y voit une partie du domaine de stabilité, dans le plan complexe, pour les schémas BDF5 (courbe verte) et BDF6 (courbe bleu) (Voir la figure 4.5 pour les domaines de stabilité complets).

$Pe = 10^6$									
Norme en temps									
Δt / h	0,1	0,05	0,025	0,0125	0,00625	0,003125	0,001563	0,000781	**Taux q**
0,1	1,2135 E-04	1,9751 E-06	7,1262E-08	4,6777E-10	7,4126E-12	1,7630E-13	8,1813 E-14	2,4807 E-15	**5,0435**
0,05	1,3131 E-04	2,2197 E-06	2,7566E-06	2,9942E-05	1,2223E-10	2,1330E-13	1,1578 E-13	8,2930 E-14	**0,4814**
0,025	4,1328 E-03	4,1470 E-03	4,1472E-03	6,0579E-02	2,9236E+03	3,9939E-03	4,1472 E-03	4,1472 E-03	**0,0000**
0,0125	8,4657 E+19	8,4657 E+19	8,4657E+19	8,4657E+19	8,4657E+19	2,5802E+23	8,4657 E+19	8,4657 E+19	**0,0000**
Norme en espace									
Δt / h	0,1000	0,0500	0,0250	0,0125	0,00625	0,003125	0,0015625	0,00078125	
0,1	3,9413 E-04	3,9413 E-04	3,9413E-04	3,9413E-04	3,9413E-04	3,9413E-04	3,9413E-04	3,9413E-04	
0,05	5,0405 E-05	5,0405 E-05	5,0405E-05	5,0405E-05	5,0405E-05	5,0405E-05	5,0405E-05	5,0405E-05	
0,025	4,1478 E-03	4,1478 E-03	4,1478E-03	4,1478E-03	4,1478E-03	4,1478E-03	4,1478E-03	4,1478E-03	
0,0125	8,4657 E+19	8,4657 E+19	8,4657E+19	8,4657E+19	8,4657E+19	8,4657E+19	8,4657E+19	8,4657E+19	
Taux p	**-74,1117**	**-74,1117**	**-74,1117**	**-74,1117**	**-74,1117**	**-74,1117**	**-74,1117**	**-74,1117**	

Tableau 5.11b : Calcul d'erreur et taux de convergence en temps et en espace pour la méthode BDF6 -- $Pe = 10^6$

On montre que l'équation de transport s'apparente à l'équation test (1.1) avec un paramètre λ complexe et une condition initiale (1.2) (Gresho, Sani, & Engelman, 1998). La partie réelle de λ représente la diffusion et la partie imaginaire, la convection. De plus, le paramètre z du plan complexe de la figure 5.5 est :

$$z = h \cdot \mathrm{Re}(\lambda) + i \cdot \Delta t \cdot \mathrm{Im}(\lambda)$$

Plus la discrétisation spatiale est fine (petit h), plus on s'approche de l'axe imaginaire du plan complexe et plus la discrétisation en temps est fine (petit Δt), plus on s'approche de l'axe réel du plan complexe. De plus, à un haut nombre de Péclet, le paramètre z se situe bien plus près de l'axe imaginaire que de l'axe réel puisque la diffusion est négligeable par rapport à la convection.

Sur la figure 5.5, quatre croix ont été tracés sur le graphique où il est plausible que les discrétisations particulières du tableau 5.11 soient distribuées (h = 0,025 dans tous les cas). La première croix, en magenta, localise de façon illustrative une valeur de z pour la discrétisation spatiale / temporelle avec Δt = 0,025. Suivent les discrétisations Δt = 0,0125 en bleu, Δt = 0,00625 en noir et Δt = 0,003125 en rouge. La croix bleue et la rouge correspondent aux figures 5.3 et 5.4 respectivement. Ils sont à l'intérieur de la courbe orientée de la méthode BDF6, donc dans la région instable.

Les deux autres croix correspondent à des discrétisations ne souffrant pas d'instabilité, ce qui illustre ce qui a été écrit plus tôt : pour BDF6 avec des éléments quadratiques et h = 0,0125, certaines discrétisations en temps sont stables et d'autres, instables. Il est difficile à priori de savoir de quelle côté de la courbe orientée nous nous trouverons.

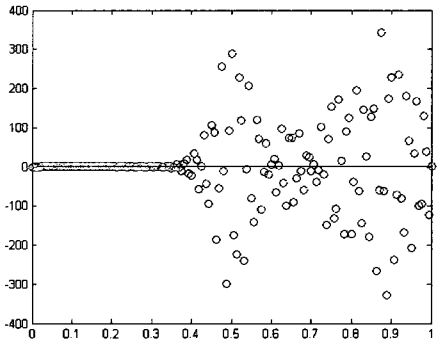

Figure 5.1 : Graphique de la solution exacte (bleu) et de la solution numérique (rouge); éléments quadratiques, schéma BDF5, h = 0,0125, Δt = 0,00625, Pe = 10^6

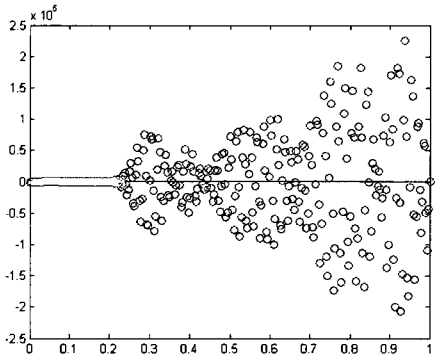

Figure 5.2 : Graphique de la solution exacte (bleu) et de la solution numérique (rouge); éléments quadratiques, schéma BDF5, h = 0,00625, Δt = 0,00625, Pe = 10^6

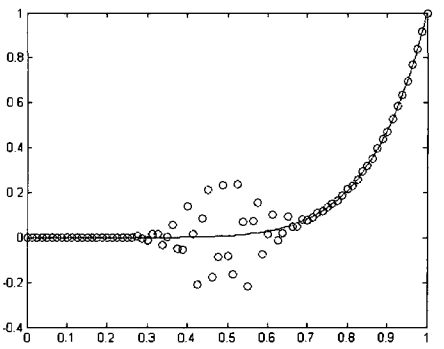

Figure 5.3 : Graphique de la solution exacte (bleu) et de la solution numérique (rouge); éléments quadratiques, schéma BDF6, h = 0,025, Δt = 0,0125, Pe = 10^6

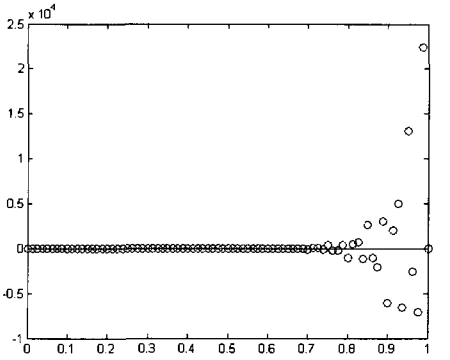

Figure 5.4 : Graphique de la solution exacte (bleu) et de la solution numérique (rouge); éléments quadratiques, schéma BDF6, h = 0,025, Δt = 0,00625, Pe = 10^6

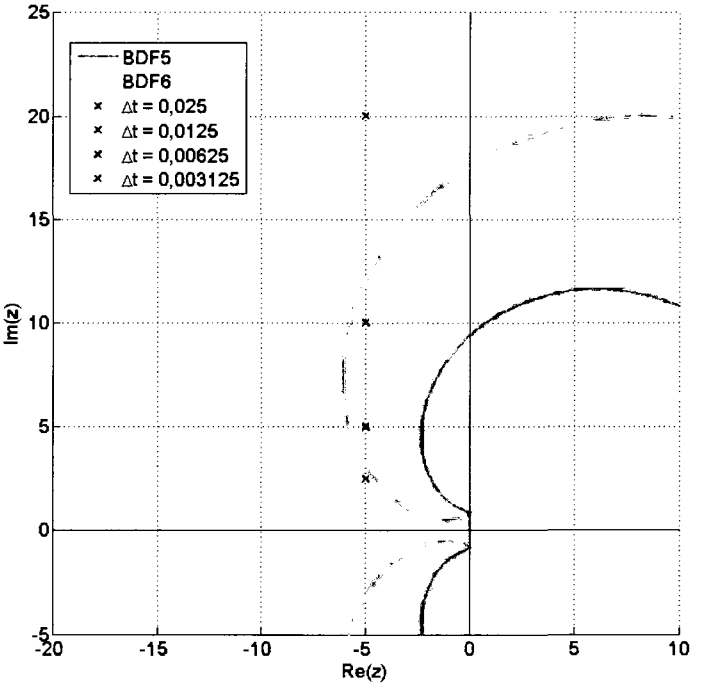

Figure 5.5 : Domaine de stabilité des schémas BDF5 et BDF6; quelques situations plausibles pour le paramètre z avec la méthode BDF6

Toujours pour BDF6, mais cette fois avec h = 0,0125, on voit que la solution est instable au point de donner des valeurs aberrantes. Les graphiques correspondants ne seront pas illustrés pour éviter la redondance.

Les figures 5.6 à 5.9 présentent différentes intensités d'instabilités pour les éléments linéaires. Sur la figure 5.6, on voit que ces instabilités sont assez faibles pour les paramètres indiqués. Avec une discrétisation plus fine en temps, les instabilités ne sont plus visibles (figure 5.7). Ce sont des cas très ponctuels. Par contre, pour BDF6 (figure 5.8 et 5.9), la situation est aussi mauvaise qu'avec les éléments quadratiques dans le même ordre de grandeur de h et Δt. D'autres cas similaires auraient pu être illustrés.

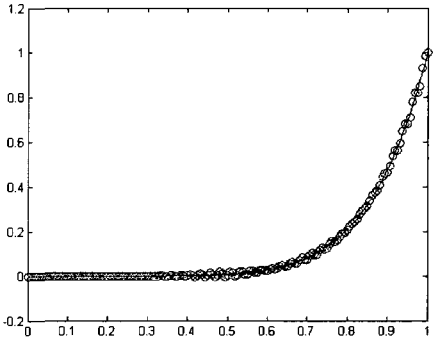

Figure 5.6 : Graphique de la solution exacte (bleu) et de la solution numérique (rouge); éléments linéaires, schéma BDF5, h = 0,00625, Δt = 0,00625, Pe = 10^6

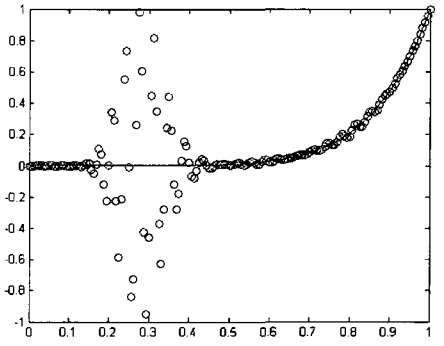

Figure 5.8 : Graphique de la solution exacte (bleu) et de la solution numérique (rouge); éléments linéaires, schéma BDF6, h = 0,00625, Δt = 0,0125, Pe = 10^6

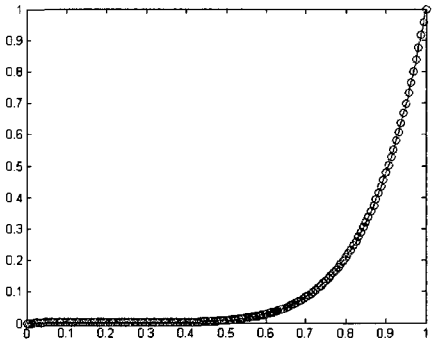

Figure 5.7 : Graphique de la solution exacte (bleu) et de la solution numérique (rouge); éléments linéaires, schéma BDF5, h = 0,00625, Δt = 0,003125, Pe = 10^6

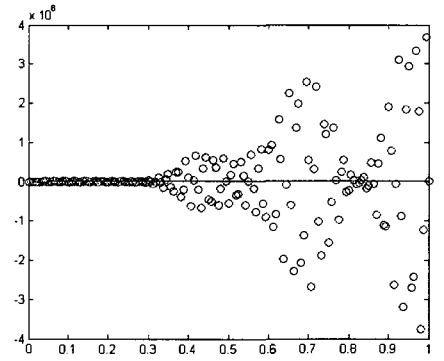

Figure 5.9 : Graphique de la solution exacte (bleu) et de la solution numérique (rouge); éléments linéaires, schéma BDF6, h = 0,00625, Δt = 0,00625, Pe = 10^6

Discussion

Les deux cas tests effectués (Pe = 1 et Pe = 10^6) nous permettent d'apprécier les propriétés idéales de stabilités des méthodes de Runge-Kutta Implicites. Par contre, leur utilisation reste délicate à implémenter. Il faut également être vigilant en ce qui a trait au phénomène de réduction d'ordre lorsque l'équation à intégrer est raide.

Les méthodes RKI augmentent le nombre d'inconnues d'un système d'équations d'un facteur égal au nombre d'étages de la méthode. Donc, pour LobattoIIIC6 (4 étages) cela devient vite très coûteux en mémoire et en temps de calcul. Une méthode avec un bon rapport mémoire versus précision semble être RadauIIA5 (3 étages). C'est d'ailleurs cette méthode qu'utilisent Hairer et Wanner (1996).

Toutes les BDF nécessitent un système d'équations d'un seul étage, ce qui requiert moins de mémoire. Cependant, on doit conserver les solutions antérieures nécessaires selon chaque BDF. De plus, ces méthodes ne sont pas auto-démarrantes lorsque nous ne connaissons pas la solution précédant le temps initial de la simulation. On doit alors démarrer l'intégration avec une méthode d'ordre inférieur. Les RKI ne souffrent pas de cet inconvénient.

Idéalement, un code d'éléments finis devrait être conçu de façon à pouvoir accepter des BDF et des RKI sans distinction. Ce code serait également muni d'une routine pour détecter la raideur des équations physiques et appliquer la méthode d'intégration en temps adéquate.

PARTIE II - LA PROGRAMMATION PARALLÈLE

CHAPITRE 6 - SECONDE REVUE BIBLIOGRAPHIQUE

Il sera maintenant question du traitement parallèle. Selon le Dictionnaire encyclopédique bilingue de la micro-informatique[2], le traitement parallèle se définit ainsi : « Méthode de traitement qui ne peut s'exécuter que sur un type d'ordinateur équipé de deux processeurs ou plus, tournant simultanément. » Comme le disent de façon imagée les auteurs Gropp, Lusk et Skjellum (1999), il est plus simple d'utiliser plusieurs bœufs pour tirer un plus gros char, plutôt que d'élever un seul bœuf gigantesque.

Les types d'ordinateurs multiprocesseurs sont souvent catégorisés selon la configuration de la mémoire. On dit d'un multiprocesseur qu'il est à mémoire partagée si tous les processeurs ont un accès commun à une unique zone mémoire. Le multiprocesseur à mémoire distribuée, quant à lui, présente autant de zones de mémoire que de processeurs et chacun d'eux ne peuvent accéder qu'à leur zone respective (Wilkinson & Allen, 2005). L'ordinateur multiprocesseur utilisé lors de ce mémoire est une machine à la disposition du réseau de recherche de l'École Polytechnique de Montréal. Il s'agit d'un IBM P690, connu sous le nom de Regatta. Le Regatta possède 16 processeurs POWER4 (1,3 Ghz) avec une mémoire partagée de 64 Gigaoctets (IBM, 2007. Il a été commercialisé en 2001, puis discontinué en 2005. Plus d'informations sont disponibles sur le site web d'IBM.

Les standards de programmation

Le paysage de la programmation en parallèle est dominé par deux standards : OpenMP et MPI. Le standard OpenMP (OpenMP Architecture Review Board, 2005) inclut des directives de compilation (#pragma en langage C++) qui permettent à un

[2] Microsoft Press (1999), *Dictionnaire encyclopédique bilingue de la micro-informatique*, Les Ulis, France, 535 p.

compilateur de cibler les régions du code qui devront être exécutées en parallèle. En plus des directives, le standard définit certaines règles à suivre afin de définir une librairie de fonctions utiles. L'interprétation des directives et l'implémentation des fonctions sont laissées libres au concepteur du compilateur tant que celui-ci respecte le standard.

Le standard MPI (Message Passing Interface Forum, 2008), quant à lui, définit une vaste librairie de fonctions dont le but principal est d'assurer des communications et des échanges de données entre différents processeurs. Le programme est donc codé avec des fonctions de communication qui sont contenues dans la librairie. Quelques implémentations libres de ces librairies existent, dont MPICH et LAM/MPI, en plus des implémentations commerciales telles celle d'IBM.

Pour ce mémoire, le standard OpenMP sera utilisé puisque le Regatta est un multiprocesseur à mémoire partagée. Cependant, les multiprocesseurs à mémoire partagée peuvent simuler un multiprocesseur à mémoire distribuée et donc le standard MPI aurait aussi pu être utilisé. L'inverse n'est pas possible. Par contre, le standard MPI 2.0 (Message Passing Interface Forum, 2008) définit les fonctions MPI I/O (Input / Output) qui permettent à un programme d'écrire des fichiers segmentés sur plusieurs disques durs en utilisant un système de fichiers parallèles.

Historique du traitement parallèle

L'ordinateur tel qu'on le connaît émerge durant les années quarante. Il s'agit d'une machine monoprocesseur, mais l'idée d'une machine multiprocesseur existe déjà. Cependant, les coûts de développement de ces multiprocesseurs empêchent à toute fin pratique leur élaboration. Il en est ainsi jusqu'aux années soixante-dix, moment où on commercialise les premiers multiprocesseurs destinés aux calculs à grande échelle. IBM, Cray Inc. et d'autres fabricants sont de la partie (Wilson, 1995).

Le bémol à ce nouvel engouement est que les programmes n'arrivent pas à s'adapter aux rapides évolutions des architectures des ordinateurs. Les programmeurs sont dans l'impossibilité de produire des logiciels qui tiennent compte des perpétuels changements apportés par les ingénieurs en informatique. Les langages de programmation connus servent à écrire des programmes séquentiels et on doit donc repenser tout le langage.

Durant les années quatre-vingt-dix, l'intérêt pour les multiprocesseurs décroît de façon significative étant donné le gain important des monoprocesseurs. En effet, plusieurs compagnies offrent des monoprocesseurs de plus en plus performants qui exécutent des programmes séquentiels maints fois testés. Il n'y a alors aucun intérêt pour une compagnie à investir temps et argent dans une machine multiprocesseur et un langage de programmation parallèle alors qu'il n'est même pas certain que l'on puisse en extraire une meilleure performance qu'un monoprocesseur exécutant le programme séquentiel, souvent déjà acquis de cette compagnie (Wilson, 1995).

Ces dix dernières années, les multiprocesseurs sont revenus en force étant donné que la puissance des monoprocesseurs semble plafonner (Petersen & Arbenz, 2004). Les méthodes de calculs modernes dépassent de loin la puissance d'un seul processeur et on doit en utiliser plusieurs à la fois pour atteindre un calcul précis en un temps relativement court. À l'écriture de ce mémoire, les plus gros ordinateurs, tels qu'annoncé sur le site web Top500.org[3], dépassent les 100 000 processeurs ! Afin d'aider les programmeurs, les standards OpenMP et MPI ont fait leur apparition.

Des références et des applications

Le traitement parallèle est une branche importante de l'informatique et plusieurs ouvrages lui sont dédiés. Mentionnons en guise de référence les ouvrages suivants : Baase & Gelder (1999), Chapman, Jost, & Pas (2008), Gropp, Lusk & Skjellum (1999), Petersen & Arbenz, (2004), Wilkinson & Allen, (2005), Wilson, (1995).

[3] http://www.top500.org/

En mécanique des fluides, le traitement parallèle est tout indiqué. La complexité des équations de Navier-Stokes en régime transitoire tridimensionnelle conduit à un nombre astronomique de calculs. Les auteurs da Cunha et Bortoli (2001) offrent une courte revue de plusieurs auteurs ayant utilisé le traitement parallèle pour le calcul des équations de Navier-Stokes. Cependant, comme ces auteurs ont orienté leurs méthodes vers le standard MPI, la revue faite par da Cunha et Bortoli concerne d'autres ouvrages impliquant MPI. Leur article explique le développement d'un solveur parallèle pour les équations de Navier-Stokes. Ils appliquent le solveur au calcul d'écoulements rotatifs. En mémoire distribuée, il y aura généralement une décomposition du domaine de calcul en sous-domaines. À chaque processeur est assigné le calcul d'un sous-domaine. À noter que leur méthode d'intégration en temps en est une de Runge-Kutta explicite, ce qui est hors du contexte de la première partie de ce mémoire.

Dere et Sotelino (2008) utilisent également la décomposition de domaine (donc MPI) pour l'analyse de structure. D'autres auteurs (Kalro & Tezduyar, 1998) ont effectué des travaux similaires sur l'écoulement incompressible autour d'une sphère. Ils utilisent la règle du trapèze implicite pour l'intégration en temps. On comprend qu'ils emploient eux aussi le paradigme de passage de messages pour ce qui est du traitement parallèle. Ils mentionnent que le traitement parallèle par passage de messages est plus efficace et plus explicite que la programmation en mémoire partagée, alors que cette dernière est généralement plus simple à implémenter.

L'utilisation la plus évidente du standard OpenMP, dans le cadre de ce mémoire, est l'implémentation du solveur PARDISO (Schenk & Gartner, 2004, 2006). Ce solveur utilise le standard OpenMP pour la résolution de systèmes linéaires $A\mathbf{x} = \mathbf{b}$ où la matrice A est creuse et possiblement non-symétrique. Tai, Zhao et Liew (2005) réalisent le calcul des équations pseudo-compressible de Navier-Stokes en utilisant un hybride OpenMP / MPI. En effet, il est possible de connecter entres eux plusieurs multiprocesseurs à mémoire partagée, ce qui constitue une grappe de calcul

intéressante et performante. Ces auteurs utilisent MPI pour décomposer le domaine de calcul en plusieurs sous-domaines, puis ensuite les calculs sur chacun des sous-domaines sont implémentés avec OpenMP. Tout comme pour ce mémoire, ces auteurs utilisent le programme METIS (Karypis & Kumar, 1998) pour la décomposition de domaine. En ce qui concerne cette dernière application, il existe aussi une version parallèle de METIS, soit ParMETIS (Karypis, Schloegel, & Kumar, 2003), qui utilise le standard MPI pour effectuer en parallèle les tâches de METIS.

Comme mentionné ci-haut, le standard OpenMP est utilisé lors de ce mémoire même si la littérature semble contenir plus d'utilisations du standard MPI. La raison à cela est que le programme original (séquentiel) est déjà existant et à ce moment, il est généralement beaucoup plus simple de rajouter des directives de compilation plutôt que de rajouter des fonctions de communication entre processeurs. Utiliser le standard MPI impliquerait des modifications plus importantes du programme original.

CHAPITRE 7 - ÉQUATIONS DE NAVIER-STOKES, FORME ADIMENSIONNELLE ET DISCRÉTISATION

Les équations de Navier-Stokes

En mécanique des fluides, les équations de Navier-Stokes définissent l'écoulement d'un fluide visqueux. Ce sont des équations aux dérivées partielles, vectorielles, non-linéaires et d'une complexité intéressante. Pour les étudier, il est indispensable de les traiter numériquement (hormis les cas les plus simples), ce qui entraîne tout un lot de défis à surmonter. Des auteurs (da Cunha & de Bortoli, 2001; Grant, Webster, & Zhang, 1998) soulignent que la mécanique des fluides offre des problèmes où les efforts de calcul sont vastes.

La forme incompressible des équations de Navier-Stokes reste adéquate pour la majorité des applications. Ceci dit, cette présente recherche traitera seulement des équations incompressibles de Navier-Stokes et n'entrera pas dans les détails de la compressibilité (i.e. à un nombre de Mach élevé).

Soit un domaine quelconque tridimensionnel Ω dans $\mathbb{R}^3$. La frontière de ce domaine est $\partial\Omega = \Gamma_d \cup \Gamma_n$ et $\Gamma_d \cap \Gamma_n = \emptyset$. Les équations de Navier-Stokes découlent de deux lois de conservation : la conservation de la masse (aussi appelée continuité) et la conservation de la quantité de mouvement. Sous forme différentielle, ces équations sont :

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) = \nabla\cdot\boldsymbol{\sigma} + \rho\mathbf{f} \text{ dans } \Omega \tag{7.1}$$

$$\nabla\cdot\mathbf{u} = 0 \text{ dans } \Omega \tag{7.2}$$

Le tenseur des contraintes σ est composé de deux contributions distinctes :

$$\boldsymbol{\sigma} = -p\mathbf{I} + 2\mu\dot{\gamma}(\mathbf{u}) \tag{7.3}$$

Dans les trois équations précédentes, les variables indiquent :

$\mathbf{u}$ → vecteur champ de vitesse, $\mathbf{u}$ = (u,v) en 2D et $\mathbf{u}$ = (u,v,w) en 3D;

ρ → densité du fluide;

t → temps;

$\boldsymbol{\sigma}$ → tenseur des contraintes surfaciques;

$\mathbf{f}$ → forces extérieures agissant sur le fluide;

p → pression du fluide;

$\mathbf{I}$ → tenseur métrique;

μ → viscosité dynamique;

$\dot{\gamma}(\mathbf{u})$ → tenseur des taux de déformation.

Par ailleurs, le tenseur de déformation est le tenseur symétrique du gradient du champ vectoriel de vitesse $\mathbf{u}$:

$$\dot{\gamma}(\mathbf{u}) = \frac{1}{2}\left[\nabla\mathbf{u} + \nabla^{T}\mathbf{u}\right] \tag{7.4}$$

L'équation (7.4) fait le lien entre le champ de vitesse et les contraintes (visqueuses). Il est déterminé à partir de l'étude du mouvement général d'une particule de fluide (Ryhming, 1984). Avec les hypothèses d'incompressibilité et de fluide newtoniens (μ = constante), l'équation (7.1) s'écrit ainsi :

$$\rho\left(\frac{\partial\mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) = -\nabla p + \mu\nabla^{2}\mathbf{u} + \rho\mathbf{f} \text{ dans } \Omega \tag{7.5}$$

Les équations sont complétées par les conditions initiales et frontières générales :

$$\mathbf{u} = \mathbf{u}_{o} \text{ à } t = 0; \tag{7.6}$$

$$\mathbf{u} = \mathbf{u}_{d} \text{ sur } \Gamma_{d}; \tag{7.7}$$

$$\boldsymbol{\sigma}\cdot\mathbf{n} = \mathbf{t}_{n} \text{ sur } \Gamma_{n}. \tag{7.8}$$

Dans les équations (7.7) et (7.8), l'indice « d » fait référence à une condition de type Dirichlet et l'indice « n » fait référence à une condition de type Neumann. L'indice « n » ne doit pas être confondu avec « **n** » le vecteur normal à une surface.

Forme adimensionnelle

Tout comme pour l'équation de transport de chaleur, il est adéquat d'utiliser la forme adimensionnelle des équations de Navier-Stokes. Certaines grandeurs de référence ont été définies au chapitre 3 et il faudra également définir les grandeurs de référence suivantes :

μ_∞ → viscosité de référence;
g → accélération gravitationnelle;

On définit ainsi les variables adimensionnelles à partir des variables dimensionnelles (ci-dessous notées par un « ' »).

$$\begin{aligned} x &= x'/L \quad y = y'/L \quad z = z'/L \quad \nabla = L\nabla' \\ \rho &= \rho'/\rho_\infty \quad \mu = \mu'/\mu_\infty \quad \mathbf{f} = \mathbf{f}'/g \\ \mathbf{u} &= \mathbf{u}'/U_\infty \quad p = p'/\rho_\infty U_\infty^2 \quad t = t'/\tau \end{aligned} \tag{7.9}$$

Pour l'équation de continuité (7.2), le changement de variable donne l'équation de continuité sous forme adimensionnelle. Le transfert est direct :

$$\nabla' \cdot \mathbf{u}' = \frac{U_\infty}{L} \nabla \cdot \mathbf{u} = 0 \quad \rightarrow \quad \nabla \cdot \mathbf{u} = 0 \text{ dans } \Omega_{adim} \tag{7.10}$$

Pour l'équation de la quantité de mouvement (7.5), il faudra d'abord effectuer le changement de variable :

$$\rho'\left(\frac{\partial \mathbf{u}'}{\partial t'}+(\mathbf{u}'\cdot\nabla')\mathbf{u}'\right)=-\nabla' p'+\mu'\nabla'^{2}\mathbf{u}'+\rho'\mathbf{f}'$$

$$\frac{\rho_\infty U_\infty}{\tau}\rho\frac{\partial \mathbf{u}}{\partial t}+\frac{\rho_\infty U_\infty^2}{L}\rho(\mathbf{u}\cdot\nabla)\mathbf{u}=-\frac{\rho_\infty U_\infty^2}{L}\nabla p+\frac{\mu_\infty U_\infty}{L^2}\mu\nabla^2\mathbf{u}+\rho_\infty g\rho\mathbf{f} \tag{7.11}$$

et ensuite multiplier les deux côtés de l'équation par $L / \rho_\infty U_\infty^2$:

$$\frac{L}{U_\infty\tau}\rho\frac{\partial \mathbf{u}}{\partial t}+\rho(\mathbf{u}\cdot\nabla)\mathbf{u}=-\nabla p+\frac{1}{Re}\mu\nabla^2\mathbf{u}+\frac{1}{Fr^2}\mathbf{f} \tag{7.12}$$

Dans l'équation précédente, Re est le nombre de Reynolds et Fr le nombre de Froude. Ils sont définis selon les deux formules suivantes :

$$Re=\frac{\rho_\infty U_\infty L}{\mu_\infty}\qquad Fr=\left(\frac{U_\infty^2}{gL}\right)^{1/2} \tag{7.13}$$

L'équation (7.12) nécessite maintenant de faire le choix des valeurs des dimensions de référence. Ce choix est a priori arbitraire, mais il en va de soi qu'un choix judicieux est tout à notre avantage. Voici le choix proposé, en plus des choix effectués au chapitre 3 :

$$\mu_\infty=\mu'\quad \tau=L/U_\infty \tag{7.14}$$

Ce choix conduit à $\rho = 1$ et $\mu = 1$ et implique une viscosité encore constante (modèle newtonien). De plus, puisque toutes les simulations de ce mémoire seront des écoulements internes (par exemple, la carotide), le terme des forces extérieures f peut être incorporé au gradient de pression (la pression est dite dynamique). Ceci dit, il disparaît de l'équation (7.12), qui devient finalement :

$$\frac{\partial \mathbf{u}}{\partial t}+(\mathbf{u}\cdot\nabla)\mathbf{u}=-\nabla p+\frac{1}{Re}\nabla^2\mathbf{u}\ \text{dans}\ \Omega_{adim} \tag{7.15}$$

L'avantage immédiat ici est que nous n'avons pas eu à choisir explicitement L et U_∞. Ainsi, la géométrie de la carotide, qui a d'abord été obtenue expérimentalement, est mise à l'échelle afin que l'aire de la section d'entrée soit unitaire. De façon similaire, le débit adimensionnel sera unitaire et le profil de vitesse à l'entrée sera choisi en tenant compte de ce débit. Typiquement, le profil en entrée sera un profil de Poiseuille ou similaire.

Discrétisation des équations de Navier-Stokes

Pour la présentation suivante, $(\cdot\,,\cdot)_\Omega$ désigne le produit scalaire usuel en norme $L_2(\Omega)$ dont la norme (2.9) n'est qu'un cas particulier de ce produit scalaire. De plus, la formulation mixte, où la continuité est incorporée à l'équation du mouvement, est utilisée. Ainsi, on ne se retrouve qu'avec une équation au lieu de deux. Soit les espaces fonctionnels suivants pour la solution et les fonctions tests (De Mulder, 1997) :

$$\begin{aligned} S_u^N &= \left\{ \mathbf{u}^N \middle| \mathbf{u}^N \in \left(H^{1N}(\Omega)\right)^D, \mathbf{u}^N = \mathbf{u}_d \text{ sur } \Gamma_d \right\} \\ V_w^N &= \left\{ \mathbf{w}^N \middle| \mathbf{w}^N \in \left(H^{1N}(\Omega)\right)^D, \mathbf{w}^N = \mathbf{0} \text{ sur } \Gamma_d \right\} \\ S_p^N &= V_q^N = \left\{ q^N \middle| q^N \in H^{1N}(\Omega) \right\} \\ H^{1N}(\Omega) &\subset H^1(\Omega) \end{aligned} \tag{7.16}$$

L'exposant « D » est le nombre de dimensions du domaine et l'indice « d » fait référence à une condition limite de Dirichlet. De même, $H^{1N}(\Omega)$ est le sous-espace de $H^1(\Omega)$ qui incorpore uniquement les fonctions impliquées dans la discrétisation. La discrétisation par éléments finis des équations (7.10) et (7.15) est donc :

Trouver $(\mathbf{u}^N, p^N) \in S_u^N \times S_p^N$ telle que $\forall (\mathbf{w}^N, q^N) \in V_w^N \times V_q^N$

$$\left(\mathbf{w}^N,\left(\frac{\partial \mathbf{u}^N}{\partial t}+\mathbf{u}^N\cdot\nabla\mathbf{u}^N\right)\right)_\Omega-\left(\nabla\cdot\mathbf{w}^N,p^N\right)_\Omega+\left(\nabla\mathbf{w}^N,\frac{1}{Re}\nabla\mathbf{u}^N\right)_\Omega \\ +\left(q^N,\nabla\cdot\mathbf{u}^N\right)_\Omega=\left(\mathbf{w}^N,\boldsymbol{\theta}\right)_{\Gamma_\theta} \tag{7.17}$$

Le membre de droite contient la traction imposée à la frontière :

$$\boldsymbol{\theta}=\left[-p^N\mathbf{I}+\frac{1}{Re}\nabla\mathbf{u}^N\right]\cdot\mathbf{n}\quad\text{sur }\Gamma_n \tag{7.18}$$

où **n** est le vecteur normal à la frontière. Le quatrième terme du membre de gauche est la contrainte d'incompressibilité (où la fonction test sur la pression joue le rôle d'un multiplicateur de Lagrange).

Stabilisation numérique (SUPG, PSPG, CONT)

Tout comme pour l'équation de transport, la discrétisation des équations de Navier-Stokes risquent de souffrir d'instabilités numériques, par exemple, lorsque le phénomène est hautement convectif. Ainsi, pour pallier à ce type d'instabilités, un terme de stabilisation SUPG similaire à l'équation (3.18) sera utilisé.

Cependant, d'autres sources d'instabilités sont possibles. Ainsi, certains types d'éléments, notamment les éléments P1/P1 (interpolation linéaire en vitesse [$\mathbf{u}^N$] et en pression [p^N]), présentent un champ de pression en damier. Ces fortes oscillations sont dues au non respect de la condition « inf-sup » (De Mulder, 1997; Thomasset, 1981), que l'on nomme aussi la condition LBB ou encore la condition Ladysenskaja-Babuška-Brezzi :

Il existe une constante positive C, indépendante de h_e, la taille des éléments, telle que :

$$\inf_{q^N\neq 0\in V_q^N}\sup_{\mathbf{w}^N\neq 0\in V_v^N}\left\{\frac{\int_\Omega q^N\nabla\cdot\mathbf{w}^N d\Omega}{\left(\int_\Omega\left\|\nabla\mathbf{w}^N\right\|^2 d\Omega\right)^{1/2}\left(\int_\Omega (q^N)^2 d\Omega\right)^{1/2}}\right\}\geq C \tag{7.19}$$

Ici, les variables sont :

$\mathbf{w}^N \rightarrow$ fonction d'interpolation (vectorielle) sur le vecteur vitesse;

$q^N \rightarrow$ fonction d'interpolation (scalaire) sur la pression.

et les fonctions d'interpolations sont égales aux fonctions tests (Galerkin standard). Cette condition stipule que l'interpolation en pression ne peut pas être choisie indépendamment de l'interpolation en vitesse. Elles sont liées par la condition LBB. Généralement, lorsque l'interpolation en pression est plus basse que l'interpolation en vitesse, la condition est respectée. Ce n'est pas le cas pour toutes les fonctions d'interpolations disponibles. L'élément P2/P1, par exemple, est adéquat. Cependant, il est plus coûteux en termes de calculs que l'élément P1/P1 stabilisé puisque chaque matrice élémentaire sera de dimension plus grande.

La stabilisation PSPG est inspirée de la stabilisation SUPG en ce sens qu'elle est aussi une méthode résiduelle. Il s'agit encore d'ajouter un terme « ST » à la formulation variationnelle :

$$ST = \sum_e \left(\int_{\Omega_e} \tau_{PSPG} \nabla q^N \left[R(p^N, \mathbf{u}^N) \right] d\Omega \right) \tag{7.20}$$

Cette fois-ci, le facteur d'échelle de temps est :

$$\tau_{PSPG} = \alpha \frac{h_e^2}{2\mu} \tag{7.21}$$

avec $\alpha \geq 0{,}1$ (Hughes, Franca, & Balestra, 1986) dans le cas des éléments Q1/Q1 (quadrangles) et $\alpha = 1/6$ pour les éléments P1/P1 (De Mulder, 1997). Plus d'informations se trouvent dans les travaux de Hughes et Franca (Hughes, Franca, & Balestra, 1986). L'équation (7.21) est valide peu importe le nombre de dimensions du problème.

Un troisième facteur de stabilité est applicable aux équations de Stokes et de Navier-Stokes. En fait, ce critère stabilise l'équation de continuité. Son facteur d'échelle de temps est désigné dans ce mémoire par τ_{CONT}. Tezduyar et Osawa (1999) propose aussi un choix pour ce paramètre. Ces auteurs le nomment τ_{LSIC}, pour « Least-Squares on Incompressibility Constant ». Le terme de stabilité qui en découle est :

$$ST = \sum_e \left(\int_{\Omega_e} \tau_{CONT} \nabla \cdot \mathbf{w}^N \rho \nabla \cdot \mathbf{u}^N d\Omega \right) \tag{7.22}$$

avec, pour les problèmes 1D, le paramètre

$$\tau_{CONT} = \frac{h_e \left| \mathbf{u}^N \right|}{2} \tag{7.23}$$

Pour ce mémoire, le paramètre τ_{CONT} est une interprétation multidimensionnelle de (7.23) avec en plus un paramètre discontinu, fonction du nombre de Reynolds, qui assure le comportement asymptotique du facteur d'échelle de temps :

$$\tau_{CONT} = \frac{h_e \left\| \mathbf{u}^N \right\|}{2} z(Re) \qquad z(Re) = \begin{cases} Re/3 & si \quad Re \le 3 \\ 1 & si \quad Re \ge 3 \end{cases} \tag{7.24}$$

On prendra $h_e = h_{UGN}$, car cette valeur de la taille élémentaire est orientée dans le sens de l'écoulement. Pour plus de détails à ce sujet, notamment sur le choix particulier proposé par Tezduyar et Osawa concernant la taille élémentaire h_{UGN} ainsi que sur l'équation (7.24), voir le mémoire de Verdier (Verdier, 2007).

Pour pallier aux instabilités qui dégradent la solution des équations de Navier-Stokes sans pour autant générer un maillage immensément dense, deux termes de stabilisation sont introduits dans l'équation (7.17), qui devient alors :

$$\left(\mathbf{w}^N,\left(\frac{\partial \mathbf{u}^N}{\partial t}+\mathbf{u}^N\cdot\nabla\mathbf{u}^N\right)\right)_\Omega-\left(\nabla\cdot\mathbf{w}^N,p^N\right)_\Omega+\left(\nabla\mathbf{w}^N,\frac{1}{Re}\nabla\mathbf{u}^N\right)_\Omega$$
$$+\left(q^N,\nabla\cdot\mathbf{u}^N\right)_\Omega+\sum_e\left(\int_{\Omega_e}\left[\tau_{SUPG}\mathbf{u}^N\cdot\nabla\mathbf{w}^N+\frac{1}{\rho}\tau_{PSPG}\nabla q^N\right]\cdot\left[R(\mathbf{u}^N,p^N)\right]d\Omega\right)_{\Omega_e} \qquad (7.25)$$
$$+\sum_e\left(\int_{\Omega_e}\left[\tau_{CONT}\nabla\cdot\mathbf{u}^N\rho\nabla\cdot\mathbf{w}^N\right]d\Omega\right)_{\Omega_e}=\left(\mathbf{w}^N,\boldsymbol{\theta}\right)_{\Gamma_\theta}$$

La première sommation regroupe la stabilisation SUPG / PSPG et la deuxième la stabilisation de la contrainte d'incompressibilité. Le facteur d'échelle de temps est suggéré par Tezduyar et Osawa (1999) :

$$\tau_{SUPG}=\left[\left(\frac{2u}{h}\right)^2+\left(\frac{2}{\Delta t}\right)^2+\left(\frac{4k}{h^2}\right)^2\right]^{-1/2} \qquad (7.26)$$

De plus, ces auteurs suggèrent d'utiliser $\tau_{PSPG} = \tau_{SUPG}$, ce que nous faisons. Le paramètre τ_{CONT} est donné par (7.24). À noter que le deuxième terme du paramètre (7.26) n'est introduit que pour un calcul en régime transitoire. Il est également important de mentionner que nous ne sommes pour l'instant pas totalement certains de l'utilité de ce terme en régime transitoire, mais nous le gardons pour faire suite aux études précédentes

Choix des éléments

Évidemment, le choix des éléments aura une influence directe sur la solution des équations de Navier-Stokes. Étant donné la formulation stabilisée (7.25), les éléments linéaires P1/P1 seront utilisés dans les tests numériques de ce mémoire. Pour plus de détails sur le choix des éléments, voir encore une fois le mémoire de Verdier (2007).

CHAPITRE 8 - UTILISATION DU TRAITEMENT PARALLÈLE POUR RÉSOUDRE LES ÉQUATIONS DE NAVIER-STOKES

La possibilité d'utiliser le traitement parallèle pour résoudre l'équation (7.25) par éléments finis nous permet d'utiliser un maillage bien plus important qu'auparavant tout en complétant le calcul en un temps raisonnable. Cependant, la complexité de programmation est accrue par le fait qu'il ne faille pas causer de conflits de données entre les différents processeurs. Cela reste possible, d'une part, si le maillage initial est judicieusement découpé en zones distinctes, et d'autre part, si les emplacements de calculs élémentaires sont distincts pour chacun des processeurs. À noter que les termes processeurs et processus sont utilisés de façon interchangeable puisqu'il est supposé qu'un seul processus ne s'exécute par processeur.

Le partitionnement de maillage

En analyse par éléments finis, une étape importante de calcul en parallèle consiste à répartir le maillage en partitions disjointes. L'exemple suivant servira d'illustration. Supposons qu'un maillage T doit être partitionné en 4 partitions P_i, i allant de 1 à 4. Alors les relations suivantes sont nécessaires :

$$\begin{aligned} P_i \bigcap P_j = \varnothing \quad & \begin{cases} i, j = 1...4 \\ i \neq j \end{cases} \\ \bigcup_{i=1}^{4} P_i = T & \end{aligned} \tag{8.1}$$

Ces conditions sont nécessaires puisque si deux processeurs référencent le même élément, donc les mêmes degrés de libertés, il y aura fort probablement une dépendance de données qui ne sera pas respectée. Les critères de Bernstein (Petersen & Arbenz, 2004) seront invalidés. Il y a toutes les chances que le résultat final soit erroné. Les critères de Bernstein peuvent s'énoncer comme suit : pour deux processus, i et j, s'exécutant en parallèle, il est nécessaire qu'à tout moment les trois conditions suivantes soient respectées :

$$\begin{aligned} I_i \bigcap O_j &= \varnothing \\ O_i \bigcap I_j &= \varnothing \\ O_i \bigcap O_j &= \varnothing \end{aligned} \tag{8.2}$$

où I et O sont respectivement des ensembles de données en entrée (Input) ou en sortie (Output). Ces conditions signifient qu'aucun espace mémoire ne peut être lu par le processus i si le processus j écrit à cet espace mémoire et ceci est vrai si on inverse i et j. De plus, les processus i et j ne peuvent pas écrire simultanément à un même espace mémoire. Ces conditions découlent du fait qu'on ne peut pas savoir à priori quel processus exécute son instruction en premier. Par exemple, si x = 0 initialement et que le processus i écrit la variable x = 2, on ne peut pas prédire ce que lira le processus j entre 0 ou 2 si on lui indique d'aller lire x.

En éléments finis, deux éléments adjacents référencent les mêmes degrés de liberté reliés à leurs nœuds communs. Ceci nous indique que non seulement les relations (8.1) sont nécessaires, mais qu'en plus, il faut s'assurer d'avoir une couche d'éléments séparant les deux partitions. Cette couche d'éléments nous assurera de respecter les conditions de Bernstein (8.2) en ayant des sous-domaines complètement disjoints. Nous nommerons cette couche d'éléments la partition frontière P_f. La relation (8.1) à laquelle on ajoute la frontière s'écrit ainsi :

$$\begin{aligned} P_i \bigcap P_j = \varnothing \quad & \begin{cases} i,j = 1...4 \\ i \neq j \end{cases} \\ P_i \bigcap P_f = \varnothing \quad & i = 1...4 \\ \bigcup_{i=1}^{4} P_i \bigcup P_f &= T \end{aligned} \tag{8.3}$$

Figure 8.1 : Exemple de partitionnement de maillage avec une couche d'éléments frontières

La figure 8.1 illustre un partitionnement de maillage. On voit les quatre partitions auxquelles s'ajoute la partition frontière. L'idée pour traiter le domaine en exécution parallèle sera de créer d'abord ce partitionnement en exécution séquentielle, de résoudre en parallèle les quatre partitions principales, puis finalement de résoudre en séquentiel la partition frontière. Évidemment, cette méthode ne permettra pas un gain de vitesse proportionnel au nombre de processeurs, mais on peut estimer que le gain sera appréciable tout de même pour les maillages de grandes tailles. En effet, plus le maillage est dense et plus la partition frontière devient négligeable par rapport aux partitions résolues durant l'exécution parallèle.

Pour accomplir le partitionnement du maillage, la librairie de fonctions METIS a été testée. Cette librairie a été développée par George Karypis, professeur associé à l'université du Minnesota. La librairie vient avec un guide d'utilisation (Karypis & Kumar, 1998). D'autres librairies implémentent à peu de choses près les mêmes techniques de partitionnement de maillage, comme par exemple la librairie Chaco, créée par Bruce Hendrikson et Robert Leland, du Sandia National Laboratories. La librairie Chaco a été légèrement testée, mais des problèmes d'exécution ont été rencontrés alors que METIS fonctionne adéquatement. De plus, ce mémoire n'a pas

comme objectif de comparer entre eux des outils de partitionnement de maillage. Pour ce qui est de l'exécution séquentielle du partitionnement de maillage, il est vrai que cette étape nuit également au fait de vouloir exécuter le programme d'éléments finis en parallèle. Cependant, les fonctions de la librairie METIS s'exécute très rapidement, même sur un gros maillage tridimensionnel. Le temps de partitionnement est négligeable par rapport au reste du programme d'éléments finis. On peut donc en conclure que la majeure partie du programme sera exécutée en parallèle.

Le problème de partitionnement de maillage est NP-complet, ce qui signifie qu'aucun algorithme optimal n'existe pour trouver la solution. Ce problème se résume à créer un découpage du maillage en parts égales tel que le nombre d'éléments en contact appartenant à deux partitions différentes est minimal. Les techniques les plus connues sont : la méthode spectrale, les méthodes géométriques et les schémas de partitionnement multi-étages (dont METIS). Ces derniers ont suscité beaucoup d'intérêt étant donné leur efficacité. On les appelle schémas multi-étages (« multilevel »), car ils commencent d'abord par réduire la taille du maillage en joignant des groupes de nœuds en multi-noeuds. Cette étape, qui est répétée successivement, vise à rendre plus grossier le maillage (« Coarsening phase »). Ensuite, le partitionnement est réalisé sur un maillage beaucoup plus grossier (typiquement 100 fois plus grossier que le maillage original). Finalement, on retourne à la finesse initiale en raffinant successivement le maillage partitionné. Pour mieux comprendre le fonctionnement de cette technique, il est utile de lire le guide de METIS (Karypis & Kumar, 1998) ainsi que les références citées dans ce guide.

Boucle sur les éléments

Un programme d'éléments finis typique comporte une boucle sur les éléments dans laquelle on calcule une matrice élémentaire ainsi qu'un membre de droite élémentaire, pour chaque élément. C'est cette boucle qui doit être parallélisée étant donné que le maillage a été subdivisé en partitions. Avec le standard OpenMP, il

s'agit d'ajouter avant la boucle, dans le programme C++, une directive #pragma omp (voir l'annexe). Mais cet ajout ne suffit pas, il faudra également s'assurer de respecter les conditions (8.2). Spécifiquement, la boucle sur les éléments requiert de l'espace mémoire pour stocker les coordonnées et la solution élémentaires, ainsi que l'information nécessaire à l'assemblage vers les structures globales. Si cette espace mémoire est la même pour tous les processeurs (en mémoire partagée) alors il est évident que les conditions (8.2) ne seront pas respectées puisque un processeur voudra écrire, par exemple, une coordonnée élémentaire, alors que l'autre processeur voudra la lire. Le résultat est indéterminé.

Le remède à ce problème est fort simple : Il faut autant d'espaces mémoire que de processeurs. Ainsi, les conflits sont évités car chaque processeur travaille avec son propre espace mémoire. Chacun calcule des matrices élémentaires et les assemblent aux bons endroits dans la matrice globale. Aucun processeur ne fait référence au même degré de liberté en même temps lors de cet assemblage. La matrice globale est donc assemblée correctement et prête à être résolue. L'exécution parallèle telle qu'instaurée lors de ce mémoire se termine ici. La matrice globale est ensuite factorisée et résolue par le solveur PARDISO, qui a déjà été introduit et qui est lui aussi exécutée en parallèle. Nous n'entrons pas dans les détails de cette exécution particulière.

Essais sur la version parallèle du programme d'éléments finis

La géométrie qui sera utilisée pour tester la version parallèle du programme d'éléments finis est celle de la carotide. Ce modèle numérique 3D provient de mesures IRM sur un patient qui ont été réalisées par Marc Thiriet de l'INRIA. La figure 8.2 présente la géométrie de la carotide. La sortie interne envoie le sang au cerveau alors que les sorties externes irriguent le visage.

De ce modèle sont tirés deux maillages présentés au tableau suivant :

maillage	nombre d'éléments	nombre de nœuds	nombre de degrés de liberté	nombre d'inconnus
M1	587 472	103 397	413 588	346 865
M2	1 478 983	253 783	1 015 132	882 493

Tableau 8.1 : Informations sur les deux maillages de la carotide

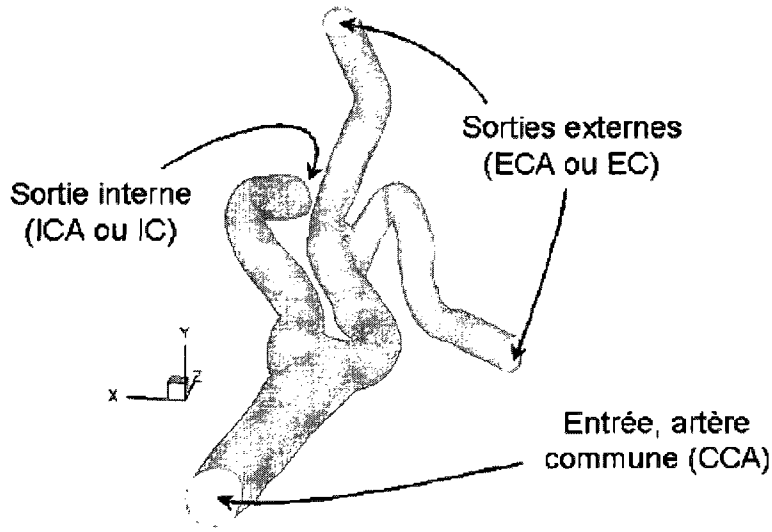

Figure 8.2 : Image de la géométrie de la carotide telle qu'obtenue par IRM

Avec les équations de Navier-Stokes incompressibles en 3D, nous avons quatre variables, soit u, v et w pour le vecteur vitesse et p pour la pression. Le nombre de degrés de liberté est quatre fois le nombre de nœuds. Le nombre d'inconnus indique les degrés de liberté qui sont à déterminer, i.e. qui ne sont pas donnés par une condition de Dirichlet. À noter que le maillage M2 conduit à une matrice considérable et cette matrice ne peut s'exprimer qu'en format compressé, c'est-à-dire qu'on ne conserve en mémoire que les coefficients de la matrice qui sont non-nuls.

Les équations modèles introduites dans le programme sont celles de Navier-Stokes adimensionnelles, équations (7.10) et (7.15). La discrétisation par éléments finis (P1/P1 stabilisé) conduit à l'équation mixte (7.25). On impose une condition d'adhérence à la paroi, $\mathbf{u} = 0$. À l'entrée de la carotide, on impose un profil de vitesse $\mathbf{u} = U_{PIV}(x,y,z)$ tiré de mesures expérimentales, mesures qui ont été prises à l'École Polytechnique de Montréal au laboratoire de PIV (Vélocimétrie par Images de Particules). Pour plus de détails sur les mesures PIV, voir le mémoire de Verdier (2007).

Résultats des essais

La variable d'intérêt lors de cette simulation, hormis que cette dernière doit donner une représentation juste de la mécanique des fluides impliquée dans la géométrie, est le temps d'exécution du programme, de son lancement à son arrêt. Les essais sont réalisés sur le Regatta mentionné au chapitre 6 de ce mémoire. Les résultats sont donnés au tableau 8.2.

Le programme séquentiel initial ne l'est pas totalement, puisqu'avant même de débuter ce mémoire, le solveur PARDISO était déjà implémenté dans le programme et donc une partie s'exécutait déjà en parallèle. Afin de comparer les anciennes performances, les temps d'exécution pour les configurations mentionnées à la deuxième colonne du tableau 8.2 ont été mesurés. La ligne indiquée par « Séquentiel Solveur 1 CPU » représente le programme exécuté totalement en séquentiel, solveur compris. Les temps sont indiqués en secondes et ce sont les temps réels d'exécution (ce qu'on appelle le « wall clock time »), et non le temps où un CPU était utilisé (« CPU time »).

Maillage	Nombre de processeurs	Temps d'exécution (s)	Gain de vitesse*	Efficacité (%)	Gain de vitesse**	Efficacité (%)
M1	Séquentiel Solveur 1 CPU	3044	1,000	1,000	-	-
	Séquentiel Solveur 2 CPU	2799	1,088	0,544	-	-
	Séquentiel Solveur 4 CPU	2755	1,105	0,276	-	-
	Séquentiel Solveur 8 CPU	2554	1,192	0,149	-	-
	Séquentiel Solveur 16 CPU	2605	1,169	0,073	1,000	-
	Parallèle 2 CPU	2189	1,391	0,695	1,190	59,5
	Parallèle 4 CPU	1101	2,765	0,691	2,366	59,2
	Parallèle 8 CPU	745	4,086	0,511	3,497	43,7
	Parallèle 16 CPU	370	8,227	0,514	7,041	44,0
M4	Séquentiel Solveur 1 CPU	20207	1,000	1,000	-	-
	Séquentiel Solveur 2 CPU	18847	1,072	0,536	-	-
	Séquentiel Solveur 4 CPU	18596	1,087	0,272	-	-
	Séquentiel Solveur 8 CPU	17022	1,187	0,148	-	-
	Séquentiel Solveur 16 CPU	17085	1,183	0,074	1,000	-
	Parallèle 2 CPU	14577	1,386	0,693	1,172	58,6
	Parallèle 4 CPU	7444	2,715	0,679	2,295	57,4
	Parallèle 8 CPU	3691	5,475	0,684	4,629	57,9
	Parallèle 16 CPU	2289	8,828	0,552	7,464	46,6

* Ce gain de vitesse est par rapport au temps séquentiel avec le solveur à 1 CPU.
** Ce gain de vitesse est par rapport au temps séquentiel avec le solveur à 16 CPU.

Tableau 8.2 : Résultats des essais réalisés sur le Regatta

Le gain de vitesse (Wilkinson & Allen, 2005) à la quatrième colonne se calcule aisément par :

$$S(p) = \frac{t_s}{t_p} \tag{8.4}$$

où p est le nombre de processeur, t_s est le temps d'exécution séquentiel avec le solveur séquentiel également et t_p est le temps d'exécution parallèle, qui est fonction de p. Le tableau 8.2 donne également le gain de vitesse par rapport au temps d'exécution séquentiel, mais avec le solveur parallèle (16 CPU). On retrouve alors ces gains de vitesse à la sixième colonne. Un tiret indique une information vide de sens.

L'efficacité est simplement le ratio du gain de vitesse au nombre de processeurs, p, et est donné en pourcentage. Clairement, pour un programme qualifié de parallèlement embarrassant, c'est-à-dire que la totalité du programme roule en parallèle et qu'aucune instruction donnée à un processeur ne dépend de l'exécution d'instructions de d'autres processeurs, alors le gain de vitesse est linéaire et égal à p. Autrement dit, $t_p = t_s / p$, donc $S = p$. On obtient ainsi une efficacité de 100 %. Ce cas reste une exception.

Pour le maillage M1, on voit selon les résultats du tableau 8.2 que la seule implication du parallélisme au niveau du solveur PARDISO ne représente pas un gain important par rapport à l'exécution entière du programme. En effet, le programme séquentiel, mais utilisant PARDISO à 16 CPU possède une efficacité de 7 % seulement. Il ne serait pas très efficace d'en rester là en termes de traitement parallèle. Lorsque le maillage est partitionné en sous-domaine (ce qui introduit des instructions supplémentaires dans le programme pour utiliser la librairie METIS) et que la boucle sur les éléments est traitée en parallèle, alors l'efficacité sur 16 CPU passe à 51 %, une augmentation appréciable. Le gain de vitesse est légèrement supérieur à 8 à ce nombre de processeurs. Le passage du programme séquentiel, solveur à 16 CPU, à parallèle 16 CPU présente un gain de vitesse de 7.

Pour le maillage M4, l'efficacité du programme séquentiel avec 16 processeurs pour le solveur n'a pas été plus appréciable qu'au maillage précédent. Elle se situe encore à 7 %. Cela laisse croire que le solveur PARDISO ne gagne pas à résoudre des problèmes plus gros. Par contre, lorsque le traitement parallèle est appliqué sur la boucle élémentaire, on atteint jusqu'à 55 % d'efficacité sur 16 processeurs. C'est 4 % de plus qu'au maillage M1. Cette amélioration provient du fait que la partition jouant le rôle de frontière entre les autres partitions devient de plus en plus négligeable en termes de nombre d'éléments par rapport au nombre total d'éléments. Cette amélioration tendrait à plafonner si on prenait des maillages encore plus gros. Ici, le

passage du programme séquentiel, solveur à 16 CPU, à parallèle 16 CPU présente un gain de vitesse de presque 7,5.

Calcul théorique du gain de vitesse

Comme mentionné plus tôt, rares sont les programme parallèlement embarrassant. Dans presque tous les programmes, il y aura une fraction « f » d'instructions qui ne peuvent pas être parallélisées. À ce moment, le gain de vitesse est déterminé par :

$$S(p) = \frac{t_s}{f t_s + (1-f)\,t_s/p} = \frac{p}{1+(p-1)f} \tag{8.5}$$

La relation (8.5) est connue sous le nom de loi d'Amdhal. Le cas parallèlement embarrassant est donné par f = 0. On a alors un gain linéaire (voir figure 8.3).

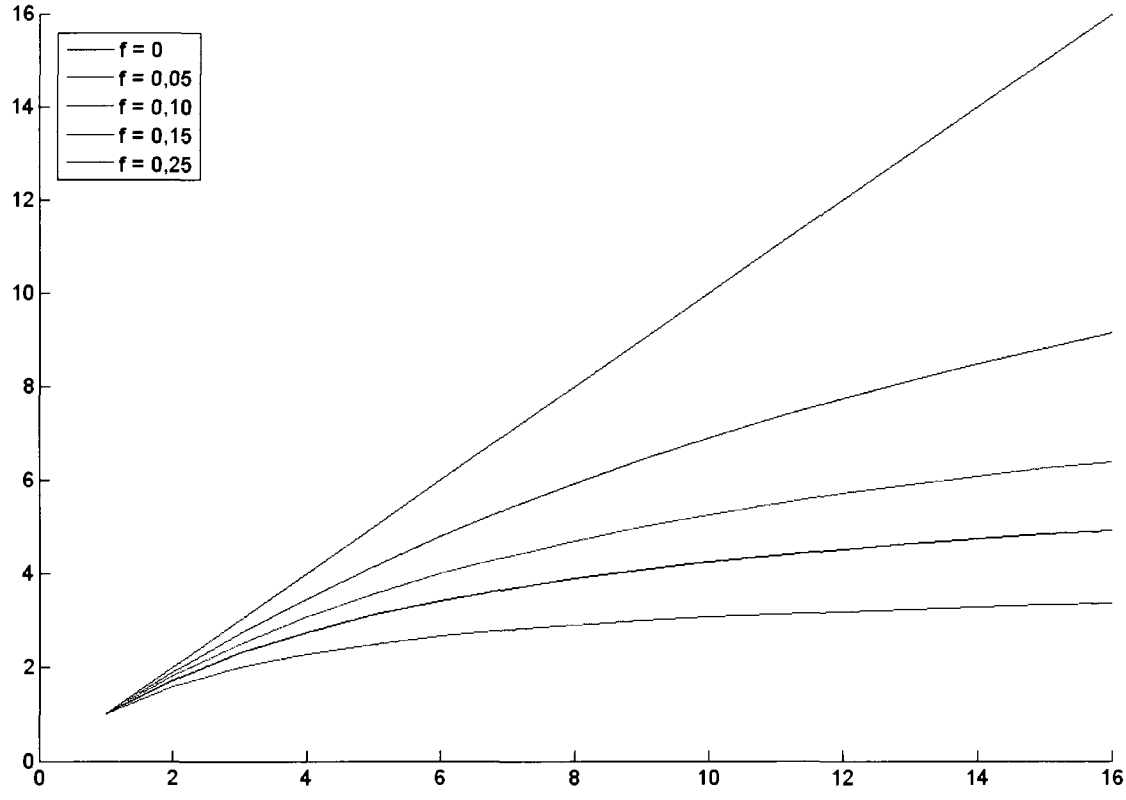

Figure 8.3 : Gain de vitesse en fonction de f, la fraction non parallélisée d'un programme

À la limite, lorsque le nombre de processeurs disponibles devient très grand (Wilkinson & Allen, 2005), on trouve :

$$\lim_{p\to\infty} S(p) = \frac{1}{f} \tag{8.6}$$

Cela veut dire qu'avec une fraction relativement faible d'instructions non parallélisée, par exemple 5 %, le code parallèle est restreint à un gain de vitesse maximal de 20, peu importe le nombre de processeurs utilisés. La figure 8.3 indique le gain de vitesse en fonction du nombre de processeurs pour certains f.

Sans tenter de le déterminer explicitement et en se basant sur les résultats du tableau 8.2, on peut affirmer que la fraction non parallélisée du présent code est d'environ 5 à 7 % (entre la courbe rouge et la verte). Cela peut sembler impressionnant si on remarque qu'en terme de code source, la boucle sur les éléments ne prend pas 95 % des fichiers sources. Cependant, la boucle est exécutée de nombreuses fois car le nombre d'éléments est grand. De plus, il y a beaucoup d'opérations effectuées à chaque tour de boucle. La factorisation et le calcul de la solution, tâches effectuées par PARDISO, semblent être peu coûteuses en comparaison de l'assemblage.

En supposant que le coefficient f pour le présent programme soit entre 5 et 7 %, le gain de vitesse maximal atteignable est entre 14 et 20. Un gain de 8 a été atteint par rapport au programme séquentiel. Il y a donc encore place a amélioration en utilisant un multiprocesseur plus puissant (plus grand p) que le Regatta actuellement disponible. Il faut cependant être vigilant, car augmenter le nombre de processeurs augmente le nombre de partitions du maillage et donc le nombre d'éléments qui composent la frontière entre les partitions augmente également. Ces éléments frontières sont calculés en mode séquentiel, ce qui augmente le facteur f et ainsi réduit le gain de vitesse maximal.

La figure 8.4 illustre la carotide avec des plans où la vitesse en direction z est affichée. La direction z est la principale direction dans l'artère commune. La gradation de couleur varie du bleu pour la vitesse la plus basse au rouge pour la vitesse la plus haute.

À noter que les conditions limites du problème sont différentes de celles utilisées par Verdier (2007) qui voulait imposer un débit à la sortie de l'artère interne. Dans le cas présent, aucune imposition n'est faite aux trois sorties de la carotide. L'intérêt de la présente simulation n'était que de mesurer le temps d'exécution et non une certaine précision de calcul. La figure 8.4 fait preuve d'un calcul que l'on pourra juger satisfaisant qualitativement.

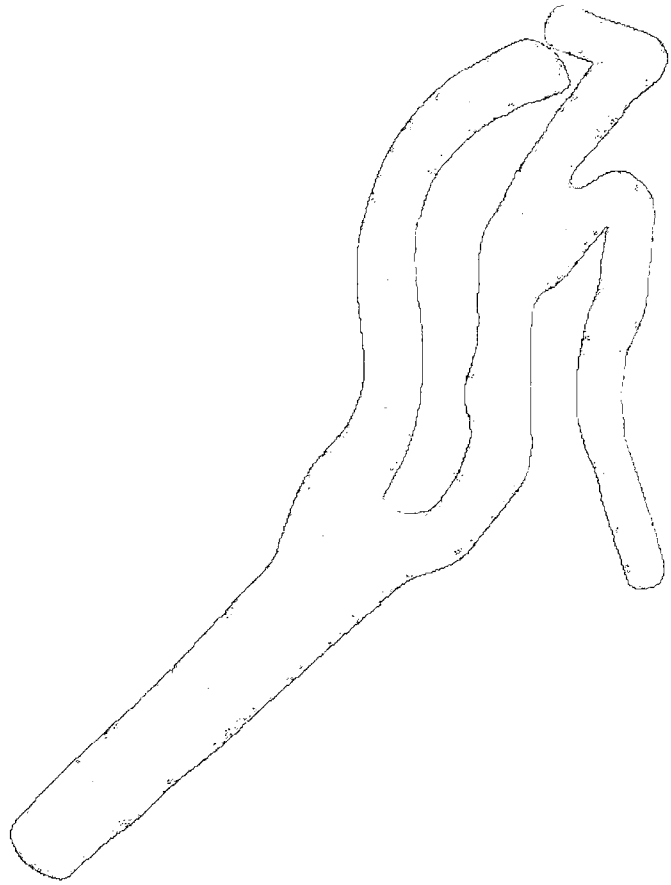

Figure 8.4 : Aperçu de l'écoulement dans la carotide

CONCLUSION ET RECOMMANDATIONS

Les différentes études présentées dans ce mémoire viennent valider les hypothèses faites lors de l'introduction. D'une part, les méthodes de Runge-Kutta implicites présentent une meilleure précision que les schémas d'intégration en temps les plus utilisés en pratique. D'autre part, leurs bonnes propriétés de stabilité sont très avantageuses lorsque les équations à intégrer sont raides. Cependant, les RKI sont très coûteuses en temps de calcul et en mémoire.

D'autre part, le traitement parallèle permet généralement de faire un gain de vitesse considérable par rapport à un code séquentiel effectuant les mêmes tâches. Les techniques de traitement parallèle, une fois bien maîtrisées, permettent de tirer toute la puissance des multiprocesseurs, puissance largement supérieur aux plus performants monoprocesseurs.

Les hypothèses étant validées, voici quelques recommandations pour guider les travaux futurs dans les domaines concernés par ce mémoire :

- L'implémentation des RKI devra être faite dans le programme d'éléments finis pour tirer profit de la précision accrue, tout en réduisant le temps de calcul grâce au traitement parallèle. Les méthodes RKI sont très utiles lorsque la précision et la stabilité sont primordiales.
- Je recommande l'utilisation « intelligente » de ces méthodes, c'est-à-dire qu'il faut les utiliser seulement lorsque l'ordre de convergence en temps doit être élevé afin de ne pas alourdir le temps de calcul à chaque simulation. Lorsque la précision en temps n'est pas d'une nécessité justifiable, il est tout de même préférable d'utiliser le schéma d'Euler implicite, le schéma de Gear ou encore le schéma du trapèze implicite. Le choix d'une méthode d'intégration en temps dépend aussi de d'autres facteurs, comme la propension à la dispersion et à la dissipation numérique.

- Il serait utile d'étudier plus en profondeur les équations différentielles algébriques, car les équations de Navier-Stokes en régime incompressible font partie de cette catégorie. Ce mémoire n'a pas su élucider tous les mystères entourant l'efficacité des méthodes RKI et autres méthodes d'intégration en temps lorsqu'il s'agit d'intégrer des ÉDA.
- Pour ce qui est du traitement parallèle, le noyau du programme, soit la boucle sur les éléments, a été parallélisée. Cependant, d'autres tâches pourront être parallélisées de la même façon, notamment les opérations de post-traitement. Éventuellement, les éléments frontières à deux zones traitées par deux processeurs différents pourront eux aussi être séparés en zones. Ceci implique d'utiliser récursivement le programme METIS. Le gain à atteindre est relativement faible et il faut d'abord voir s'il en vaut l'effort. D'autres procédures peuvent être élaborées pour traiter les frontières en parallèle.
- Je suggère également de faire une étude extensive sur la précision du programme parallèle afin de s'assurer du bon fonctionnement. Ce mémoire ne contient pas d'analyse détaillée de la précision lorsque le traitement parallèle est utilisé.

LISTE DE RÉFÉRENCES

Ascher, U. M., Ruuth, S. J., & Spiteri, R. J. (1997). Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics, 25*(2-3), 151-167.

Baase, S., & Gelder, A. V. (1999). *Computer Algorithms: Introduction to Design and Analysis* (3e éd.). Reading, MA: Addison-Wesley Longman Publishing Co., Inc.

Bochev, P. B., Gunzburger, M. D., & Lehoucq, R. B. (2007). On stabilized finite element methods for the Stokes problem in the small time step limit. *International Journal for Numerical Methods in Fluids, 53*(4), 573-597.

Bochev, P. B., Gunzburger, M. D., & Shadid, J. N. (2004). Stability of the SUPG finite element method for transient advection-diffusion problems. *Computer Methods in Applied Mechanics and Engineering, 193*(23-26), 2301-2323.

Brooks, A. N., & Hughes, T. J. R. (1981). Streamline Upwind / Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering, 32*(1-3), 199-259.

Butcher, J. C. (1964). Implicit Runge-Kutta processes. *Math. Comput., 18*(IV), 50-64.

Butcher, J. C. (1975). A stability property of implicit Runge-Kutta methods. *BIT (Nordisk Tidskrift for Informationsbehandling), 15*(4), 358-361.

Büttner, J., & Simeon, B. (2003). Time integration of the dual problem of elastoplasticity by Runge-Kutta methods. *SIAM Journal on Numerical Analysis, 41*(4), 1564-1584.

Chapman, B., Jost, G., & Pas, R. v. d. (2008). *Using OpenMP : portable shared memory parallel programming*. Cambridge, Mass.: MIT Press.

Chipman, F. H. (1971). A-stable Runge-Kutta processes. *BIT (Nordisk Tidskrift for Informationsbehandling), 11*(4), 384-388.

da Cunha, R. D., & de Bortoli, A. L. (2001). A parallel Navier-Stokes solver for the rotating flow problem. *Concurrency and Computation Practice & Experience, 13*(3), 163-180.

De Mulder, T. (1997). Stabilized finite element methods (SUPG, GLS, ...) for incompressible flows. *Lecture series - von Karman Institute for fluid dynamics, 2*, H1-H53.

Dere, Y., & Sotelino, E. D. (2008). Domain-by-domain algorithm for nonlinear finite-element analysis of structures. *Journal of Computing in Civil Engineering, 22*(1), 58-67.

Dettmer, W., & Peric, D. (2003). An Analysis of the Time Integration Algorithms for the Finite Element Solutions of Incompressible Navier-Stokes Equations Based on a Stabilized Formulation. *Computer Methods in Applied Mechanics and Engineering, 192*, 1177-1226.

Ehle, B. L. (1968). High order A-Stable methods for the numerical solution of systems of DEs. *BIT, 8*(4), 276-278.

Fletcher, C. A. J. (1991). Computational techniques for fluid dynamics 1 : Fundamental and General Techniques (2^{e} éd.). Berlin: Springer-Verlag.

Franca, L. P., & Oliveira, S. P. (2003). Pressure Bubbles Stabilization Features in the Stokes Problem. *Computer Methods in Applied Mechanics and Engineering, 192*(16-18), 1929-1937.

Grant, P. W., Webster, M. F., & Zhang, X. (1998). Coarse grain parallel finite element simulations for incompressible flows. *International Journal for Numerical Methods in Engineering, 41*(7), 1321-1337.

Gresho, P. M., Sani, R. L., & Engelman, M. S. (1998). *Incompressible flow and the finite element method. Volume 1, Advection-diffusion.* Chichester, England: Wiley.

Gropp, W., Lusk, E., & Skjellum, A. (1999). *Using MPI : portable parallel programming with the message-passing interface* (2e éd.). Cambridge, Mass.: MIT Press.

Hairer, E., Lubich, C., & Roche, M. (1989). The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods. Berlin: Springer-Verlag.

Hairer, E., Nørsett, S. P., & Wanner, G. (1993). *Solving ordinary differential equations I - Nonstiff Problems* (2e éd.). Berlin ; New York: Springer.

Hairer, E., & Wanner, G. (1996). Solving ordinary differential equations II - Stiff and Differential-Algebraic Problems (2e éd.). Berlin ; New York: Springer.

Huerta, A., & Donea, J. (2002). Time-accurate solution of stabilized convection-diffusion-reaction equations: I - Time and space discretization. *Communications in Numerical Methods in Engineering, 18*(8), 565-573.

Hughes, T. J. R., Franca, L. P., & Balestra, M. (1986). New FEM for CFD: V. Circumventing the Babuska-Brezzi condition: A stable Petrov-Galerkin formulation of the Stokes problem accomodating equal-order interpolations. *Computer Methods in Applied Mechanics and Engineering, 59*(1), 85-99.

International Business Machines Corp. (2007). 7040-681 IBM pSeries 690. Consulté le 19 septembre 2008, http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype =dd&subtype=sm&appname=pseries&htmlfid=897/ENUS7040-681

Kalro, V., & Tezduyar, T. (1998). 3D computation of unsteady flow past a sphere with a parallel finite element method. *Computer Methods in Applied Mechanics and Engineering, 151*(1-2), 267-276.

Kanevsky, A., Carpenter, M. H., Gottlieb, D., & Hesthaven, J. S. (2007). Application of implicit-explicit high order Runge-Kutta methods to discontinuous-Galerkin schemes. *Journal of Computational Physics, 225*(2), 1753-1781.

Karypis, G., & Kumar, V. (1998). METIS, A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. Minneapolis: Department of Computer Science & Engineering at the University of Minnesota Consulté le 6 février 2008, tiré: http://glaros.dtc.umn.edu/gkhome/metis/metis/download

Karypis, G., Schloegel, K., & Kumar, V. (2003). *PARMETIS, Parallel Graph Partitionning and Sparse Matrix Ordering Library*. Minneapolis: Department of Computer Science & Engineering at the University of Minnesota Consulté le 6 février 2008, tiré: http://glaros.dtc.umn.edu/gkhome/metis/metis/download

Kennedy, C. A., & Carpenter, M. H. (2003). Additive Runge-Kutta schemes for convection-diffusion-reaction equations. *Applied Numerical Mathematics, 44*(1-2), 139-181.

Lacasse, D., Garon, A., & Pelletier, D. (2007). Development of an adaptive Discontinuous-Galerkin finite element method for advection-reaction equations. *Computer Methods in Applied Mechanics and Engineering, 196*(17-20), 2071-2083.

Message Passing Interface Forum (2008). *MPI: A Message-Passing Interface Standard.*

OpenMP Architecture Review Board (2005). *OpenMP Application Program Interface.*

Petersen, W. P., & Arbenz, P. (2004). *Introduction to Parallel Computing - A Practical Guide with Exemples in C*. New York: Oxford University Press Inc.

Rang, J. (2008). Pressure corrected implicit θ-schemes for the incompressible Navier-Stokes equations. *Applied Mathematics and Computation, 201*(1-2), 747-761.

Raviart, P.-A., & Thomas, J. M. (1983). Introduction à l'analyse numérique des équations aux dérivées partielles. Paris: Masson.

Ryhming, I. L. (1984). *Dynamique des fluides : un cours de base du deuxième cycle universitaire* (1er éd.). Lausanne, Suisse: Presses Polytechniques Romandes.

Schenk, O., & Gartner, K. (2004). Solving unsymmetric sparse systems of linear equations with PARDISO. *Future Generation Computer Systems, 20*(3), 475-487.

Schenk, O., & Gartner, K. (2006). On fast factorization pivoting methods for sparse symmetric indefinite systems. *Electronic Transactions on Numerical Analysis, 23*, 158-179.

Tai, C. H., Zhao, Y., & Liew, K. M. (2005). Parallel-multigrid computation of unsteady incompressible viscous flows using a matrix-free implicit method and high-resolution characteristics-based scheme. *Computer Methods in Applied Mechanics and Engineering, 194*(36-38), 3949-3983.

Tezduyar, T. E., & Osawa, Y. (1999). Finite Element Stabilization Parameters Computed from Element Matrices and Vectors. *Computer Methods in Applied Mechanics and Engineering, 190*, 411-430.

Tezduyar, T. E., & Senga, M. (2007). SUPG finite element computation of inviscid supersonic flows with shock-capturing. *Computers & Fluids, 36*(1), 147-159.

Thomasset, F. (1981). Implementation of finite element methods for Navier-Stokes equations. New York: Springer-Verlag.

Verdier, M. (2007). Méthodes numériques pour le calcul de l'écoulement dans l'artère carotide. École polytechnique de Montréal, Montréal.

Wilkinson, A. B., & Allen, C. M. (2005). Parallel programming : techniques and applications using networked workstations and parallel computers (2e éd.). Upper Saddle River, N.J.: Pearson Prentice Hall.

Wilson, G. V. (1995). *Practical parallel programming*. Cambridge, Mass.: MIT Press.

ANNEXE

MODIFICATIONS SUR EF POUR CRÉER UNE VERSION PARALLÈLE

Macro à la compilation

Une macro a été définie : « _OPENMP_ ». Cette macro est utilisée à plusieurs endroits où il faut faire la distinction entre une compilation du code séquentiel et la compilation du code parallèle. Parfois, on utilisera aussi certaines variables, lorsque disponibles, pour faire cette distinction.

Nouvelles librairies

Les librairies suivantes ont été ajoutées.

libMETIS : contient le programme METIS dont toutes les fonctions ont été renommées en ajoutant « EF5_ » au début de chacune d'elles.

libEF_P : Une version parallélisée de libEF.

libAXB_PARDISO_2EF_P : Une version parallélisée de libAXB_PARDISO_2EF.

Les fichiers qui suivent ont été modifiés dans ces nouvelles librairies. Ainsi, tout l'ancien programme a été conservé. On peut également compiler l'ancien programme (séquentiel) ou le nouveau (parallèle) selon le choix du « remake » indiqué par « _P ».

Modifications au code source

EF5.C

Si la macro _OPENMP_ a été définie, alors on doit effectuer le coloriage du maillage. L'objet de type DOMAINEGEOMETRIQUE, nommé « geometrique », exécute la méthode COLORIAGE.

DOMAINEGEOMETRIQUE.C

Une méthode ajoutée : COLORIAGE. Cette méthode passe l'exécution du coloriage sur chacune des partitions du domaine, c'est-à-dire sur chaque sous-domaine. Le découpage selon METIS se fait si approprié. Les autres méthodes ne sont pas modifiées.

PARTITIONGEOMETRIQUE.C

Quelques méthodes sont ajoutées, principalement USEMETIS et COLORIAGE pour effectuer le découpage. La méthode USEMETIS prépare les variables pour METIS et appelle ensuite ce programme. Les données de sorties de METIS sont deux vecteurs : l'un donnant le numéro de processeur relié à chaque élément et l'autre donnant le numéro de processeur relié à chaque nœud. La méthode CONNECTIVITEEFFECTIVE, comme son nom l'indique, détermine une connectivité effective d'une partition. Elle est nécessaire pour le bon fonctionnement de METIS.

Voici un exemple de connectivité effective. Sur la partition G0 (qui est généralement le sous-domaine de plus haute dimension), les numéros de nœuds dans la table de connectivité sont bien numérotés. Cependant, sur G1 et suivants, soit les sous-domaines frontières, les numéros de nœuds dans la connectivité ne sont pas adéquats (Tableau A.1). La méthode CONNECTIVITEEFFECTIVE permet de transformer cette table pour que METIS produise une exécution conforme (Tableau A.2). Dans les deux tables, il y a treize (13) nœuds effectifs, mais leurs numéros sont différents.

Élément	Nœuds
1	1-4-12
2	23-2-7
3	2-5-7
4	16-8-9
5	13-18-1
6	18-16-5
7	7-10-13

Tableau A.1 : Connectivité sur une frontière du domaine

Élément	Nœuds
1	1-3-9
2	13-2-5
3	2-4-5
4	11-6-7
5	10-12-1
6	12-11-4
7	5-8-10

Tableau A.2 : Connectivité effective pour METIS sur une frontière du domaine

La méthode COLORIAGE utilise les données à la sortie de METIS pour séparer les éléments sur les processeurs. La figure A.1 ci-dessous est une bonne indication de l'exécution de cette méthode. À la fin de cette méthode, la partition contient deux attributs de type DECOUPAGEGEOMETRIQUE : un vecteur d'objets nommé « decoupage » et un objet seul nommé « frontiere ».

DECOUPAGEGEOMETRIQUE.C

Les deux objets mentionnés ci-haut sont d'un type qui n'était pas implémenté avant dans EF. Cette classe est calquée sur la classe PARTITIONGEOMETRIQUE. Elle contient des attributs définissant quels éléments est dédié à quel processeur. La table de connectivité complète reste dans la partition, mais chaque objet du vecteur « decoupage » possède sa propre table de connectivité définissant les éléments reliés à sa zone. Cette table indique quel élément de la partition est référencé par quel élément du découpage. Sur la figure A.1, on voit que les éléments en bleus appartiennent au processeur 1 et ceux en jaunes, au processeur 2. Les éléments en roses décrivent la frontière entre les deux zones.

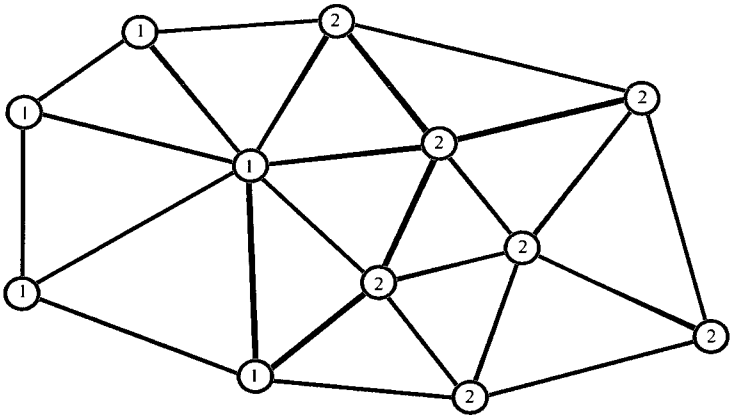

Figure A.1 : Découpage typique des éléments après le découpage en zones

TOPOLOGIEELEMENT.C

Le fichier contenant la topologie des éléments a été légèrement modifié. La structure de données contenant les paramètres des différents éléments disponibles dans EF a reçus un paramètre de plus. Ce paramètre, appelé « indiceMetis » (de type EFint) permet d'indiquer le type d'éléments à METIS parmi les choix suivants : triangles (1), tétraèdres (2), prismes (3) et carrés (4) (Karypis & Kumar, 1998). Si un élément dans EF n'est pas de ce type, alors indiceMetis vaut 0 pour ce type inconnu de METIS. Une nouvelle méthode, INDICEMETIS, permet de retourner « indiceMetis » pour l'élément concerné.

PROBLEME.C

C'est le fichier central ou le calcul est effectué en parallèle. Si la macro _OPENMP_ est définie, alors on doit lire plusieurs fois le fichier PDE afin de construire un système d'équations pour chaque processeur. Si on exécute en séquentiel, le système comporte M équations où M est spécifié dans le fichier PDE. Si on exécute en parallèle avec N processeurs, alors le système comporte M*N équations. À chaque lecture du fichier PDE, M équations et donc M espaces de travail sont créés pour les différents processeurs. Ces espaces de travail sont nécessaires pour avoir un

vecteur d'adressage par processeur, entre autres. Ceci entraîne une duplication de certaines informations, mais le coût mémoire relié est faible. Faire autrement aurait nécessité plus de travail de programmation pour une très petite économie de mémoire.

La méthode SOMMERESIDU calcule le résidu élémentaire. On y retrouve la directive « pragma » suivante :

```
#pragma omp parallel for firstprivate(decoupage, Nelm) if(nb_zones >
1)
```

Afin de comprendre le standard OpenMP (OpenMP Architecture Review Board, 2005), voici la signification des différents éléments :

#pragma

En C/C++, on utilise le pragma pour signaler une directive à la compilation, tout comme un « define ».

omp

Le pragma omp indique au compilateur qu'il s'agit d'une directive relié à la programmation en mémoire partagée (dont le standard OpenMP).

parallel for

La directive est de faire une boucle for en parallèle.

firstprivate(decoupage, Nelm)

Tout ce qui suit le « for » est une clause. La clause firstprivate désigne la liste de variables, dans ce cas « decoupage » et « Nelm », à être dupliquées pour chacun des threads avec une valeur initiale donnée avant la directive pragma.

if(nb_zones > 1)

Si la condition n'est pas respectée, la directive au complet n'a pas lieue. Ici, « nb_zones » est le nombre de zones de découpage du sous-domaine (de la partition).

Ainsi, la boucle est faite en parallèle si plus d'un processeur est présent. Chaque processeur possède sa propre valeur de « decoupage » et de « Nelm », chacune des

deux initialisée à une certaine valeur commune (NULL et 0 respectivement). Chaque processeur exécute son itération de la boucle. Ainsi, chacun d'eux récupèrent leur nombre d'éléments (Nelm) et leur connectivité respective (découpage). Le compteur de la boucle est implicitement privé à chacun des processeurs. Puisque chaque processeur a une structure de données qui pointe sur une partie unique du maillage, aucun conflit ne survient.

Pour faciliter la lecture du code, deux méthodes privées ont été ajoutées : BOUCLEELEM_LOCAL et BOUCLEELEM_GLOBAL. Ces méthodes, comme leurs noms l'indiquent, exécute la boucle sur les éléments, avec un adressage et un calcul de résidu. Le cas LOCAL est le cas standard et le cas GLOBAL est utilisé lorsque le calcul du résidu requiert une intégrale sur toute la partition pour chacun des éléments

MATRICE_PARDISO.C

Ce fichier est complémentaire au fichier PROBLEME.C, car il contient les informations spécifiques à la matrice du solveur PARDISO (Schenk & Gartner, 2004, 2006). Les modifications à la méthode SOMME sont très semblables aux modifications faites sur SOMMERESIDU dans PROBLEME.C.