ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE MECANIQUE
M.Ing.

PAR
MOHAMED EL BOUZOUIKI

DÉVELOPPEMENT ET ÉTUDE DES ÉLÉMENTS FINIS SPECTRAUX POUR LES ÉCOULEMENTS RAPIDES

MONTRÉAL, LE 16 AVRIL 2002

CE MÉMOIRE A ÉTÉ ÉVALUÉ
PAR UN JURY COMPOSÉ DE :

M. Azzeddine Soulaïmani, directeur de mémoire
Département de génie mécanique à l'École de technologie supérieure

M. Christian Masson, président du jury
Département de génie mécanique à l'École de technologie supérieure

M. Hassan Manouzi, professeur
Département de Mathématique à l'Université Laval

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 16 AVRIL 2002

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

SOMMAIRE

Ce travail consiste en la simulation numérique des écoulements compressibles tridimensionnels et turbulents. On s'intéresse particulièrement à la modélisation de l'écoulement dans la région de la paroi de la couche limite. On vise à développer un élément de paroi qui serait capable de reproduire la séparation de la couche limite.

En effet, l'idée fondamentale de ce travail de recherche consiste à l'utilisation de la méthode des éléments spectraux de haute précision pour mieux résoudre la couche limite des écoulements tridimensionnels pour un fluide visqueux et compressible. Ceci a une importance primordiale pour mieux simuler les interactions entre les couches limites et les chocs qui donnent naissance à la séparation de l'écoulement et par conséquent à l'accroissement de la traînée. La prédiction de la traînée est importante particulièrement en aérodynamique transsonique.

Dans le plan tangent à la paroi solide, la discrétisation est assurée par des éléments finis linéaires sur des triangles. On désire alors enrichir l'interpolation dans la direction normale.

Habituellement on utilise des lois de paroi de type log-linéaire. Cependant, le modèle log-lin seul est incapable de modéliser fidèlement les zones de recirculation ou de décollement. On construit un deuxième modèle qui consiste à rajouter une approximation spectrale au profil log-lin afin d'enrichir l'interpolation dans la direction normale à la paroi. L'élément de paroi est donc un prisme où l'approximation est très riche dans la direction normale et linéaire dans le plan tangent.
Pour la turbulence, on utilise le modèle de Spalart-Allmaras.

La contribution de ce travail consiste à la mise en œuvre d'une bibliothèque de routines pour implémenter les méthodes spectrales p d'ordre élevé en vu d'être appliquées à la modélisation des écoulements rapides compressibles avec couche limite. On étudie le comportement de ces méthodes pour des problèmes modèles unidimensionnels. Ensuite, on montre la généralisation de ces méthodes pour les écoulements compressibles 3D.

AVANT-PROPOS ET REMERCIEMENTS

Ce mémoire a été mené sous la direction de Dr. Azzeddine Soulaïmani, professeur au département du génie mécanique de l'École de technologie supérieure de Montréal.

Je tiens à le remercier pour sa patience, pour le temps qu'il m'a consacré et pour l'aide précieuse et qu'il m'a procurée pour finaliser ce travail.

Par ailleurs, je tiens également à remercier Dr. Christian Masson, directeur du programme de Maîtrise au département du génie mécanique à l'École de technologie supérieure de Montréal, Dr. Hassan Manouzi, professeur au département de mathématique à l'Université Laval et Dr. Éric Laurendeau de Bombardier Aéronautique pour leur engagement en tant que membres du jury d'évaluation de ce travail.

Je remercie aussi tous mes collégues du GRANIT pour leurs encouragements et leurs soutient et surtout Dr. Reda Touihri qui m'a aidé ennormément dans la réalisation de ce travail.

Enfin, je remercie du fond de mon cœur tous les membres de ma famille pour leur amour, leurs encouragements et leur soutient durant toutes mes études.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES ABVERVIATIONS ET SIGLES

c vitesse du son
C_p chaleur spécifique à pression constante
C_v chaleur spécifique à volume constant
$\mathbf{D}$ tenseur de déformations
e énergie totale
E énergie totale par unité de volume
$\mathfrak{I}$ vecteur des forces de volume
f_v forces de volumes
f_1 force selon la direction x
f_2 forces selon la direction y
f_3 forces selon la direction z
F_i^{conv} flux d'Euler
F_i^{diff} flux de diffusion
h dimension caractéristique des éléments
i énergie interne
$\mathbf{I}$ matrice identité
J jacobien de la transformation géométrique
k conductivité thermique
$[\mathbf{K}]$ matrice de rigidité globale
Ma nombre de Mach
$[\mathbf{M}]$ matrice masse globale
$\mathbf{n}$ vecteur unitaire orienté vers l'extérieur
n niveau de pas de temps
N fonctions d'interpolation
P ordre d'interpolation
p pression
Pr nombre de Prandlt
Pe nombre de Peclet global
Pe_h nombre de Peclet local
q source de chaleur
r sources énergétiques
$\mathbf{R}$ ensemble de nombre réels
R résidu
Re nombre de Reynolds
t temps
T température
u vecteur vitesse de composantes u_i
u_1 composante de la vitesse selon x
u_2 composante de la vitesse selon y

u_3 composante de la vitesse selon z
$|u|$ norme de la vitesse = $(u_1 + u_2 + u_3)^{1/2}$
U vecteur quantité de mouvement
U_1 composante de la quantité de mouvement selon x
U_2 composante de la quantité de mouvement selon y
U_3 composante de la quantité de mouvement selon z
V vecteur des fonctions solutions
W formulation variationnelle
x, y, z coordonnées cartésiennes

α angle d'attaque
γ rapport des chaleurs spécifiques
δ symbole de Kronecker
ε seuil de convergence
λ valeurs propres
viscosité dynamique
ν viscosité turbulente
ξ vecteur des coordonnées de l'élément de référence
ρ densité
σ tenseur des contraintes visqueuses de composantes σ_{ij}
Γ_∞, Γ_w, Γ_∞^+, Γ_∞^- : frontières à l'infini, sur le corps, aval, amont
Ω domaine de calcul

Indices:

$*$ valeur de référence
i valeur unitaire selon la direction des x
j valeur unitaire selon la direction des y
k valeur unitaire selon la direction des z
r de référence
∞ à l'infini
$+$ à la paroi
, dérivée par rapport à

Indices supérieure:

i niveau d'itération
e de l'élément

Notation utilisées:

$\{N\}$ vecteur colonne
$\langle N \rangle$ vecteur ligne
[K] matrice de composantes K_{ij}

$\otimes$ produit tensoriel $\left[(u \otimes v)_{ij} = u_i v_j\right]$

Note : Cette liste ne doit pas être considérée exhaustive. Certains symboles sont utilisées de façon particulière dans un contexte précis; les définitions explicites sont alors clairement énoncées dans le texte.

INTRODUCTION

Les développement théoriques de la dynamique de fluides visent à établir et à résoudre les équations gouvernant les différentes catégories d'écoulement. Les équations de la dynamique des fluides Newtoniens stationnaires et instationnaires sont connues depuis plus de 150 années. Cependant, la résolution de ces équations à des échelles très réduites telle que la turbulence reste encore un champs de recherche très actif. La dynamique des fluides expérimentale a joué un rôle très important dans la validation et la mise en évidence des limites des différentes approximations des équations gouvernantes. Le développement de la vitesse d'exécution et de l'espace mémoire des ordinateurs depuis les années 1950 a conduit à l'émergence de la dynamique des fluides numériques (Computational Fluid Dynamics : CFD). Cette branche de la dynamique des fluides vient compléter la théorie et l'expérience en fournissant un outil efficace et relativement peu coûteux de simulations des écoulements réels. La branche CFD présente cinq avantages par rapport à l'expérimental :

- Le temps nécessaire de conception et de développement des modèles numériques très avancés est réduit de manière significative.
- Elle peut simuler des conditions non reproductibles par les modèles expérimentaux (nombre de Mach et de Reynolds très élevés).
- Elle prévoit la compréhension de plusieurs détails et informations.
- Elle est très peu coûteuse que la soufflerie
- Elle consomme très peu d'énergie.

La méthode des éléments finis

Les équations qui gouvernent les écoulements internes ou externes sont, généralement, compliqués à résoudre analytiquement. De plus, la géométrie de la frontière du domaine d'intérêt est, en général, tellement complexe qu'une solution analytique d'une équation

ou d'un système d'équations décrivant un problème physique donné, devienne inaccessible. Ainsi, on a recourt à l'utilisation de méthodes numériques. Ces dernières permettent de transformer l'équation ou le système d'équations aux dérivées partielles en une équation ou un système d'équations algébriques pour la résolution par ordinateurs. Parmi ces méthodes, on trouve la méthode des éléments finis et la méthode des éléments spectraux. A cause de sa puissance et de sa flexibilité, la méthode des éléments finis est maintenant très largement utilisée en ingénierie et dans l'industrie. Cette méthode se caractérise par un ensemble de facteurs qui ont conduit à son succès.

C'est une méthode des résidus pondérés, d'où sa flexibilité d'associer une variété de formulations variationnelles à un modèle différentiel donné.

- Elle peut s'adapter facilement à des maillage non structurés, d'où sa puissance pour modéliser des géométries très complexes.
- Elle permet une bonne précision à un coût raisonnable grâce à l'espace d'approximation des géométries très complexes.
- Elle permet un traitement naturelle des conditions aux limites de type différentiel.

La méthode des éléments spectraux fait aussi partie de la catégorie des méthodes des résidus pondérés et profite à la fois des avantages de la méthode des éléments finis (flexibilité géométrique) et ceux des techniques spectrales (degré de convergence élevé). Les méthodes spectrales peuvent être vues comme étant des méthodes d'éléments finis où les types de polynômes utilisés et/ou les points d'interpolation sont non classiques.

Objectifs

Le but de cette recherche consiste à étudier le comportement des méthodes des éléments finis/éléments spectraux dans la modélisation des écoulement tridimensionnels compressibles et turbulents.

Plan du mémoire

La présentation de ce mémoire est structurée comme suit :

Après ce chapitre d'introduction, la méthode des éléments spectraux est présentée au chapitre 1. Nous introduirons les méthodes dites *h* et les méthodes *p*, et nous focaliserons plus sur les méthodes *p* en mettant en évidence, sur un exemple unidimensionnel, les effets de l'ordre d'interpolation et du choix des points d'interpolation sur la précision de la solution numérique.

Dans le chapitre 2, nous présentons la formulation du modèle mathématique où les équations de Navier-Stokes sous forme conservative et en variables conservatives sont rappelées. Le modèle de turbulence utilisé est également présenté dans ce chapitre.

Le chapitre 3 est consacré à la formulation variationnelle des équations de conservation, nous discutons des conditions aux limites et nous exposons la discrétisation spatio-temporelle de la formulation variationnelle. Les méthodes de stabilisation des problèmes de convection-diffusion, la méthode de Pétrov-Galerkin et la méthode SUPG et GLS font aussi l'objet de ce chapitre. On présentera aussi dans ce chapitre une étude numérique avec les méthodes spectrales *hp* de l'équation de convection dans le cas unidimensionnel.

Enfin, nous clôturons cette recherche par une conclusion générale en formulant quelques recommandations importantes pour les travaux futurs à très court et moyens termes.

CHAPITRE 1

MÉTHODE DES ÉLÉMENTS SPECTRAUX

1.1 Introduction

Ce chapitre est consacré à la présentation de la méthode des éléments spectraux isoparamétriques. Cette méthode fait partie de la catégorie des méthodes aux résidus pondérés et profite à la fois des avantages de la méthode des éléments finis (flexibilité géométrique) et de ceux des techniques spectrales (degré de convergence élevé) [1, 7, 8, 11, 13, 18, 19].

Dans le but d'accroître la précision des calculs par éléments finis, il existe trois approches qui sont classées : *h-méthode*, *p-méthode* et *hp-méthode*.

Dans les méthodes dites de type *h*, l'ordre du polynôme d'interpolation est fixé pour chaque élément et la convergence est atteinte en raffinant le maillage. Les méthodes *p* utilisent un maillage fixe et la convergence est assurée par l'augmentation de l'ordre *p* du polynôme d'interpolation. Toute fois l'augmentation de *p* ne garantit pas une bonne approximation de la solution si le choix des points d'interpolation n'est pas adéquat. Dans ce cas, on doit soit changer la position des points d'interpolation ou diminuer la taille de l'élément. Les méthodes des éléments finis *hp* tirent profit des avantages des deux méthodes *h* et *p* (raffinement + augmentation de *p*). Les méthodes spectrales peuvent être vues comme étant des méthodes d'éléments finis où les types de polynômes utilisés ou la position des points d'interpolation sont non classiques. Dans cette recherche on va s'attarder sur ce genre de méthodes et on va également envisager le cas d'une base d'approximation non polynomiale.

La figure 1 montre l'avantage d'utiliser une méthode *p* avec un choix judicieux des points d'interpolation.

On veut approximer la fonction:

$$f(x)=\begin{cases} 1 & \text{si } -0.5 \le x \le 0.5 \\ 0 & \text{Ailleurs} \end{cases} \tag{1.1}$$

en utilisant un seul élément et une base Lagrangienne d'ordre P=16. On envisage trois types de points d'interpolation : les points équidistants (EQD), les points de collocation de Gauss-Lobatto-Legendre (GLL) et les points de collocation de Gauss-Lobatto-Chebychev (GLC). Les points de GLL et GLC sont montrés sur les figures 3 et 5.

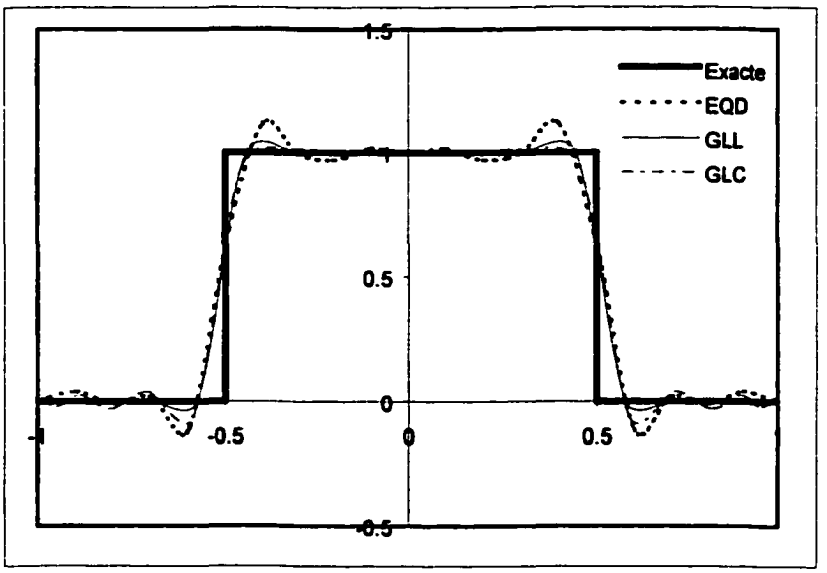


Figure 1- Mise en évidence de l'effet du choix des points d'interpolation

On voit très bien comment un ordre élevé seul est incapable de donner une solution acceptable, (solution oscillante), d'où la nécessité de diminuer la taille *h* de l'élément ou de faire un choix adéquat de la position des points d'interpolation.

Le choix des points d'interpolation sera discuté plus en détail dans la suite de ce chapitre.

1.2 Un problème modèle

Pour mettre en évidence la méthode des éléments spectraux, considérons le problème de Helmholtz suivant :
Trouver une fonction u dans l'espace $L^2[-1, 1]$ vérifiant :

$$\begin{cases} \dfrac{d^2u}{dx^2} - \lambda^2 u(x) = f(x), \quad -1 \le x \le 1 \\ u(1) = u(-1) = 0 \end{cases} \tag{1.2}$$

Pour mettre en oeuvre une approximation par la méthode des éléments spectraux du problème (1.2), on va construire la formulation variationnelle du problème. En multipliant l'équation (1.2) par une fonction test v nulle sur les bords, et en intégrant sur l'intervalle $[-1, 1]$, on obtient :

$$-\int_{-1}^{1} v(x) \frac{d^2u}{dx^2} dx - \lambda^2 \int_{-1}^{1} u(x) v(x) dx = \int_{-1}^{1} f(x) v(x) dx \tag{1.3}$$

En intégrant par parties le premier terme et en utilisant les conditions aux limites, l'équation (1.3) s'écrit :

$$-\int_{-1}^{1} \frac{du}{dx} \frac{dv}{dx} dx - \lambda^2 \int_{-1}^{1} u(x) v(x) dx = \int_{-1}^{1} f(x) v(x) dx \tag{1.4}$$

Soit maintenant $(x_0, x_1, \ldots x_N)$ les points de collocation du domaine $[-1, 1]$ et $(u_0, u_1, \ldots u_N)$ la valeur de u en ces points $u(x_i) = u_i$ pour $0 \leq i \leq N$ (valeurs nodales).

La fonction $u(x)$ peut être approximée à l'aide du polynôme de Lagrange sur les $N+1$ points d'interpolation par:

$$\tilde{u}_N(x) = \sum_{j=0}^{N} h_j(x)\, u_j \tag{1.5}$$

où $h_j(x)$ est le $j^{ème}$ polynôme de Lagrange donnée par :

$$h_j(x) = \prod_{i=0,\ (i \neq j)}^{i=n} \frac{(x - x_i)}{(x_j - x_i)} \tag{1.6}$$

avec

$$h_j(x_i) = \delta_{ij}$$

où δ est le symbole de Kronecker.

En utilisant la méthode de Galerkin, la fonction test v et la fonction f sont interpolées sur $[-1, 1]$ de la même façon que u. L'équation (1.4) devient donc sous sa forme discrétisée :

$$-\int_{-1}^{1} \left\{ \sum_{j=0}^{n} u_j \frac{dh_j}{dx} \sum_{k=0}^{n} v_k \frac{dh_k}{dx} \right\} dx - \lambda^2 \int_{-1}^{1} \left\{ \sum_{j=0}^{n} u_j\, h_j(x) \sum_{k=0}^{n} v_k\, h_k(x) \right\} dx =$$

$$\int_{-1}^{1} \left\{ \sum_{j=0}^{n} f_j\, h_j(x) \sum_{k=0}^{n} v_k\, h_k(x) \right\} dx \tag{1.7}$$

ou encore :

$$-\sum_{j=0}^{n}\sum_{k=0}^{n} u_j v_k \int_{-1}^{1} \frac{dh_j}{dx}\frac{dh_k}{dx}dx - \lambda^2 u_j v_k \int_{-1}^{1} h_j(x) h_k(x) d x =$$

$$\sum_{j=0}^{n}\sum_{k=0}^{n} f_j v_k \int_{-1}^{1} h_j(x) h_k(x) dx \qquad (1.8)$$

Cette équation est valable quelque soit la valeur des v_k . On obtient donc N+1 équations qui peuvent se mettre sous la forme :

$$A^{kj} u_j - \lambda^2 B^{kj} u_j = B^{kj} f_j , \quad 0 \le k \le N \qquad (1.9)$$

avec :

$$A^{kj} = \int_{-1}^{1} \frac{dh_j}{dx}\frac{dh_k}{dx}dx \qquad (1.10)$$

$$B^{kj} = \int_{-1}^{1} h_j(x) h_k(x) dx \qquad (1.11)$$

D'une façon plus générale, pour un domaine $\Omega=[a,b]$ quelconque, on applique le changement de variable suivant :

$$\xi = \frac{2}{l}(x-a) - 1 \qquad (1.12)$$

et on aura :

$$A^{kj} = -\frac{2}{l}\int_{-1}^{1} \frac{dh_j}{dx}\frac{dh_k}{dx}dx \qquad (1.13)$$

$$B^{kj} = \frac{l}{2}\int_{-1}^{1} h_j(x) h_k(x) dx \qquad (1.14)$$

où l est tout simplement la longueur de l'intervalle Ω. Les matrices A et B sont appelées respectivement, matrice de rigidité et matrice masse. Elles sont toutes les deux symétriques pour ce problème modèle. En posant :

$$C^{kj} = A^{kj} - \lambda^2 B^{kj} \tag{1.15}$$

le système final à résoudre pour trouver la solution discrétisée du problème (1.2) est le suivant:

$$C^{kj} u_j = B^{kj} f_j, \quad 0 \le k \le N \tag{1.16}$$

1.3 Points d'interpolation

Dans ce paragraphe, nous présentons le type de discrétisation choisi sur lesquels la solution u du problème (1.2) est représentée par une interpolation de Lagrange. Trois types de points d'interpolation sont présentés et comparés: les points équidistants EQD, les points de Gauss-Lobatto-Chebychev (GLC) et les points de Gauss-Lobatto-Legendre (GLL).

1.3.1 Points d'interpolation équidistants

Les points d'interpolation équidistants sur l'intervalle $[-1,1]$ pour un ordre d'interpolation N, sont donnés par :

$$x_i = -1 + \frac{2i}{N} \quad \text{pour } 0 \le i \le N \tag{1.17}$$

1.3.2 Points de collocation Gauss-Lobatto-Chebychev

Pour tout $x \in [-1, 1]$, nous pouvons définir la famille des polynômes de Chebychev [8] $\{T_k(x), k = 0, 1...\}$ par :

$$T_k = \cos(k\theta), \quad \theta = \arccos(x) \tag{1.18}$$

On peut voir que les polynômes de Chebychev ne sont autre que les fonctions cosinus après un changement de variable.

Cette famille de polynômes peut être aussi définie par la suite récurrente suivante :

$$\left.\begin{aligned} T_0(x) = 1 \quad \text{et} \quad T_1(x) = x \\ \text{et pour } k \geq 2 : \\ T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x) \end{aligned}\right\} \tag{1.19}$$

Sur la figure 2, on montre les polynômes de Chebychev jusqu'à l'ordre P = 5 :

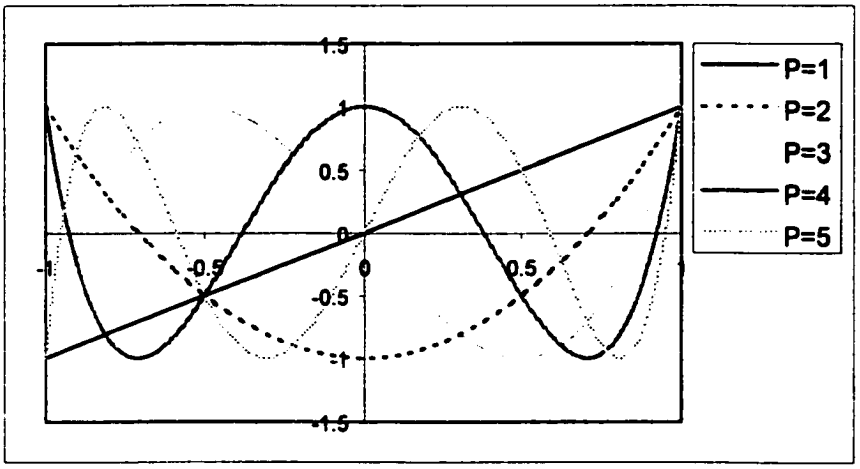


Figure 2- Les polynômes de Chebychev

Les points de collocation de Gauss-Lobatto-Chebychev de l'intervalle $[-1,1]$ sont donnés par la suite des points :

$$x_i = -\cos\left(\frac{i\pi}{N}\right) \quad \text{pour } 0 \le i \le N \tag{1.20}$$

qui sont les zéros du polynôme T_{N+1}.

La figure 3 montre les positions des points de GLC :

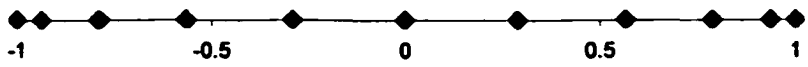


Figure 3- Points de collocation de Gauss-Lobatto-Chebychev (P =10)

En utilisant les points de collocation de GLC, les fonctions d'interpolation de Lagrange s'écrivent [7] :

$$h_j(x) = \frac{2}{N c_j} \sum_{i=0}^{N} \frac{4}{c_i} T_i(x_j) T_i(x) \tag{1.21}$$

avec

$$\left.\begin{aligned} c_k &= 1, \quad 1 \le k \le N-1 \\ c_k &= 2, \quad k = 0, N \end{aligned}\right\} \tag{1.22}$$

Ainsi les matrices A^{ij} et B^{ij} deviennent :

$$A^{ij} = \frac{4}{c_i\, c_j\, N^2} \sum_{k=0}^{n} \sum_{l=0}^{n} \frac{4}{c_k\, c_l} T_k\,(x_i) T_l\,(x_j)\, a_{kl} \tag{1.23}$$

$$B^{ij} = \frac{4}{c_i\, c_j\, N^2} \sum_{k=0}^{n} \sum_{l=0}^{n} \frac{4}{c_k\, c_l} T_k\,(x_i) T_l\,(x_j)\, b_{kl} \tag{1.24}$$

avec

$$a_{kl} = -\int_{-1}^{1} \frac{dT_k}{dx} \frac{dT_l}{dx}\, dx \tag{1.25}$$

$$b_{kl} = \int_{-1}^{1} T_k\, T_l\, dx \tag{1.26}$$

1.3.3 Points de collocation de Gauss-Lobatto-Legendre

Les polynômes de Legendre peuvent être présentés par la suite récurrente suivante [8] :

$$\left.\begin{aligned} & L_0\,(x) = 1 \quad \text{et} \quad L_1\,(x) = x \\ & \text{et pour } k \geq 1 : \\ L_{k+1}\,(x) \;=\; & \frac{2k+1}{k+1}\, x\, L_{k-1}\,(x) - \frac{k}{k+1}\, L_{k-1}\,(x) \end{aligned}\right\} \tag{1.27}$$

Sur la figure 4, on montre les polynômes de Legendre jusqu'à l'ordre P = 5 :

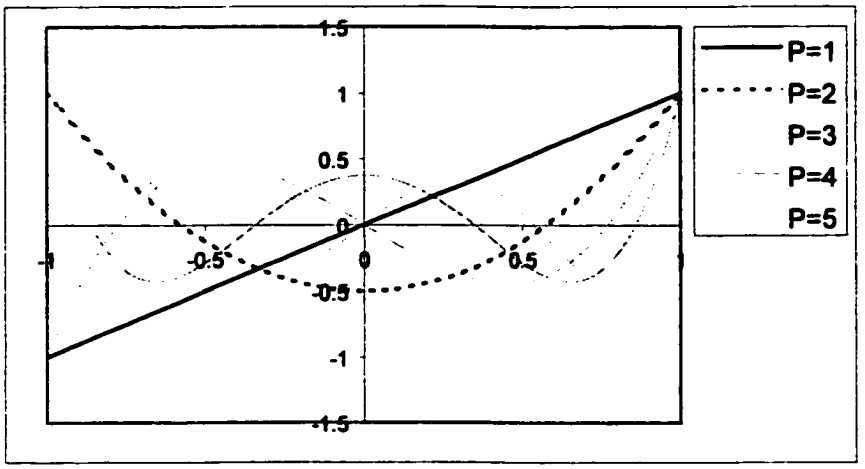


Figure 4- Les polynômes de Legendre

Les points de collocation de Gauss-Lobatto-Legendre de l'intervalle $[-1,1]$, sont définis comme suit : $x_0 = -1$, $x_N = 1$ et les x_j, $1 \le j \le N-1$, sont les zéros du polynôme $L_N^{'}$, classés dans l'ordre croissant.

La figure 5 montre les positions des points de GLL :

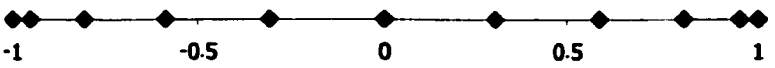


Figure 5- Points de collocation de Gauss-Legendre-Lobatto (P=10)

Pour l'intégration numérique, on utilise la quadrature de Gauss-Lobatto.

La quadrature de Gauss-Lobatto utilise les (N+1) points de GLL associés à des poids d'intégration ω_j définis par :

$$\omega_j = \frac{2}{N(N+1)} \frac{1}{L_N(x_j)^2} \quad \text{pour} \quad j = 0,..., N \tag{1.28}$$

On peut montrer que L'intégration de Gauss-Lobatto est exacte pour les polynômes jusqu'à l'ordre 2N-1 [8].

En utilisant les points de collocations de GLL, les fonctions d'interpolation de Lagrange s'écrivent [8] :

$$h_j(x) = \frac{(1-x^2) L_N'(x)}{N(N+1) L_N(x_j)(x-x_j)} \tag{1.29}$$

Ainsi les matrices A^{ij} et B^{ij} deviennent :

$$A^{ij} = \int_{-1}^{1} \frac{dh_i}{dx} \frac{dh_j}{dx} dx = \sum_{q=0}^{N} \frac{dh_i}{dx}(x_q) \frac{dh_j}{dx}(x_q) \omega_q \tag{1.30}$$

$$B^{ij} = \int_{-1}^{1} h_i(x) h_j(x) dx = \sum_{q=0}^{N} h_i(x_q) h_j(x_q) \omega_q = \omega_i \delta_{ij} \tag{1.31}$$

ce qui veut dire que la matrice masse $\boldsymbol{B}$ est diagonale, ceci est dû au fait que les points d'interpolation sont les mêmes que les points d'intégration. Il aurait été impossible d'obtenir une matrice masse diagonale si par exemple on avait utilisé pour l'intégration numérique, les poids de Gauss car ces derniers ne contient pas les bords (-1 et 1).

Pour le calcul de la matrice A, les dérivées $\frac{dh_i}{dx}$ aux points x_i sont données par :

$$\frac{dh_i}{dx}(x_i) = \begin{cases} \dfrac{L_N(x_i)}{L_N(x_i)(x_i - x_j)} \text{ pour } i \neq j \\ 0 \text{ si } i = j \text{ avec } i \neq j \text{ et } j \neq N \\ -\dfrac{N(N+1)}{4} \text{ si } i = j = 0 \\ \dfrac{N(N+1)}{4} \text{ si } i = j = N \end{cases} \tag{1.32}$$

1.4 Approximation avec les bases de Legendre et de Chebychev

On peut également utiliser les polynômes de Legendre ou de Chebychev comme bases d'approximation:

$$\tilde{u}_N(x) = \sum_{j=0}^{N} \Phi_j(x)\alpha_j \tag{1.33}$$

Dans ce cas, les α_j sont les coordonnées de u dans la base des Φ_j et non plus les degrés de liberté u_j. Ces derniers sont retrouvés par reconstruction à partir des α_j :

$$u_j = \sum_{j=0}^{N} \Phi_j(x_j)\alpha_j \tag{1.34}$$

L'utilisation des polynôme de Legendre comme polynôme d'interpolation, donne une matrice masse diagonale (à cause de l'orthogonalité de cette famille de polynômes).

Sur la figure 6, on montre une approximation avec P=16 de la fonction (1.1) en utilisant les bases d'approximation de Legendre et de Chebychev:

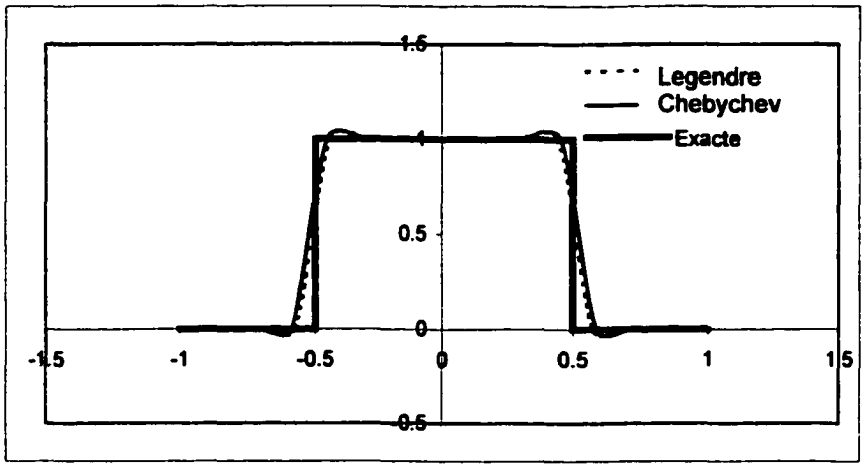


Figure 6- Approximation avec les bases de Legendre et de Chebychev (P=16)

On remarque que ces deux bases d'approximation permettent d'obtenir des bonnes solutions (sans oscillations) avec des précisions comparables. Toutefois on a vu qu'il est possible d'obtenir une solution comparable avec une base lagrangienne utilisant les points de GLL. Sur la figure 7 on voit que le conditionnement de la matrice masse dans le cas d'une base lagrangienne avec les points de GLL est comparable à celui dans le cas d'une base de Legendre.

1.4 Choix du type d'interpolation

Pour faire un choix entre les trois types de discrétisation, on va considérer deux aspects importants, le conditionnement de la matrice masse et l'ordre de convergence de l'interpolation..

1.4.1 Le conditionnement

Le nombre de conditionnement κ d'une matrice **M** est défini comme suit:

$$\kappa(\mathbf{M}) = \frac{\max\limits_{1 \le j \le N+1} |\lambda_j|}{\min\limits_{1 \le j \le N+1} |\lambda_j|} \tag{1.35}$$

où λ_j sont les valeurs propres de la matrice **M**. Quand une matrice est mal conditionnée, le nombre d'itérations requis pour les méthodes itératives devient important [22].

Sur la figure 7, nous donnons le nombre de conditionnement de la matrice masse en fonction de l'ordre du polynôme:

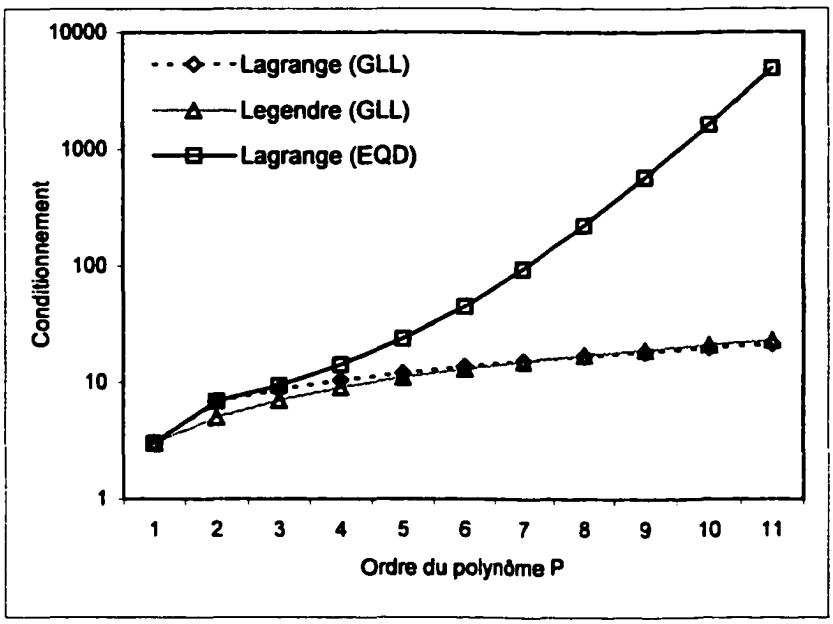


Figure 7- Évolution du nombre de conditionnement masse en fonction de l'ordre du polynôme d'interpolation

En comparant pour les quatre types d'approximation, on remarque qu'avec une base d'approximation Lagragienne des points utilisant des points d'interpolation équidistants, le nombre de conditionnement reste relativement bon jusqu'à l'ordre 3, après il commence à croître. Les points de collocation de Gauss-Lobatto-Legendre donnent un meilleur conditionnement que ce soit avec une base d'approximation de Lagrange ou de Legendre. Ces deux dernières restent comparables et genérent des matrices masse beaucoup mieux conditionnées. Ceci est attribué à la propriété d'orthogonalité des polynômes de Legendre et des polynômes de Lagrange dans le cas où les points de GLL sont à la fois les points d'interpolation et les points d'intégration comme on l'a déjà montré. Cette orthogonalité se traduit par des matrices masses qui sont diagonales et donc mieux conditionnées.

1.4.2 La convergence de l'interpolation

Pour vérifier la convergence, on se propose de comparer les résultats de la résolution problème de Helmoltz suivant :

Trouver une fonction u dans l'espace $L^2[-1, 1]$ vérifiant :

$$\begin{cases} \dfrac{d^2u}{dx^2} - u(x) = f(x), & -1 \le x \le 1 \\ u(1) = u(-1) = 0 \end{cases} \tag{1.36}$$

On considère la solution exacte:

$$u(x) = \sin(\pi x) \tag{1.37}$$

Quand la solution est de la forme (1.37), on sait que $f(x) = -(\pi^2 + 1)\sin(\pi x)$. La solution numérique du problème obtenue avec une base Lagrangienne utilisant les trois types de points d'interpolation EQD, GLL et GLC est montrée à la figure 8:

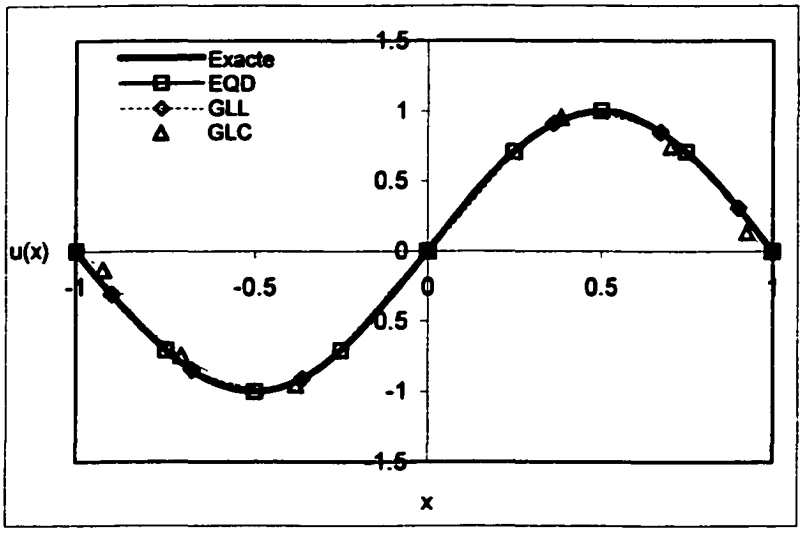


Figure 8- Comparaison des trois types d'interpolation : EQD, GLL et GLC avec P=8

L'erreur est définie comme la différence entre la solution numérique $\tilde{u}(x)$ et la solution analytique $u(x)$;

$$\| \varepsilon \|_{L^2} = \left(\int_{-1}^{1} | u(x) - \tilde{u}(x) |^2 \, dx \right)^{\frac{1}{2}} \tag{1.38}$$

La convergence est assurée si la norme de l'erreur tend vers 0 quand l'ordre de l'interpolation augmente.

Sur la figure 9, nous donnons l'évolution de l'erreur en fonction de l'ordre d'interpolation:

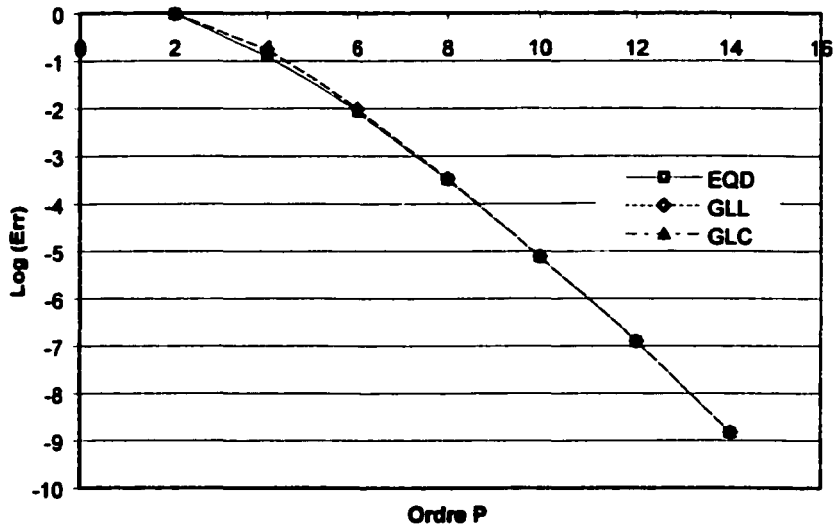


Figure 9- Les précisions obtenues pour les trois types d'interpolation: EQD, GLL et GLC. Évolution de l'erreur en fonction de l'ordre d'interpolation

D'après la figure 9, on voit que, pour ce cas où la solution est très régulière, la convergence est approximativement la même pour les trois types d'interpolation.

Une dénière comparaison consiste à reproduire un profil ressemblant à un profil d'écoulement séparé et voir l'effet du choix des points d'interpolation.

On envisage un profil de la forme :

$$u^+ = f(y^+) - \beta \sin\left(\frac{3}{2}\pi y^+\right)$$

où f étant la fonction de Spalding [12] représantant la loi de paroi et β un facteur caratérisant le gradient de pression adverse.

La figure 10 montre les solutions obtenues pour P = 4, 5 et 6 et avec des points équidistants et des points de GLL. La base utilisée est une base de Lagrange.

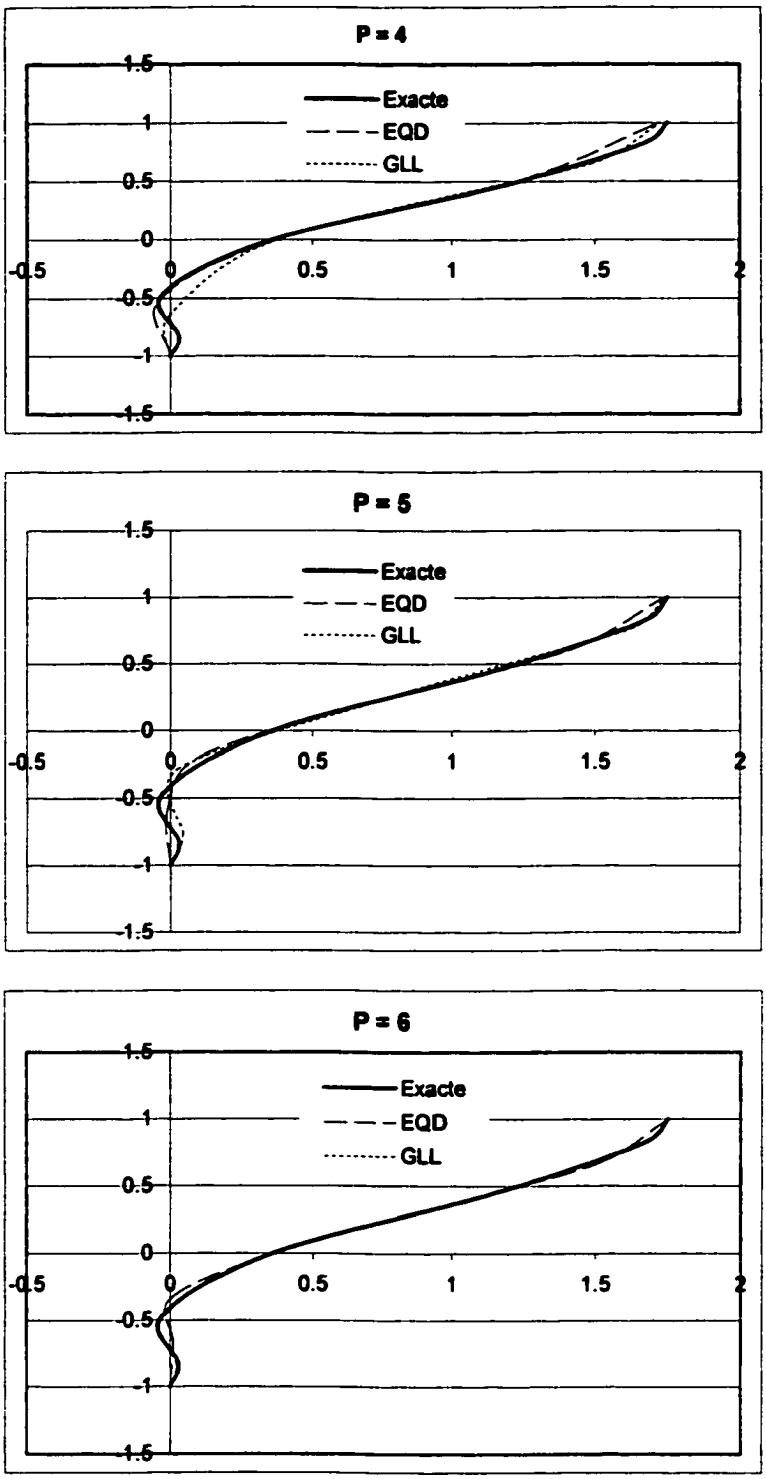


Figure 10- Approximation d'un profil séparé avec des points EQD et des points GLL pour P = 4, 5 et 6

La figure 10 montre un avantage marqué des points GLL par rapport aux points équidistants. On voit très bien qu'avec les points GLL, on arrive à représenter fidélement la séparation de l'écoulement à partir de P = 6 et ceci est dû au fait que les points de GLL sont concentrés aux bords du domaine et donc capables de repésenter les phénomèmes qui se produisent dans ces régions.

1.5 Conclusion

D'après cette brève comparaison et analyse des différents types de bases d'approximation et des points d'interpolation, on peut conclure qu'une base d'approximation d'ordre élevé de Lagrange avec des points de collocation de GLL serait le choix adéquat à faire, vu sa précision comparativement à la base de Legendre et à la base de Chebychev et aussi sa facilité d'implémentation puisqu'on a seulement à introduire les nouvelles positions des points d'interpolation.

N.B: Vous trouverez à l'annexe 1 le programme utilisé pour résoudre le problème de Helmoltz, ainsi que les fonctions et les routines suivantes :

- La fonction PLN qui calcule les polynômes de Legendre.
- La fonction PDN qui calcule les dérivées des polynômes de Legendre.
- La fonction TKP qui calcule les polynômes de Chebychev.
- La fonction TKPP qui calcule les dérivées des polynômes de Chebychev.
- La routine GLOB qui calcule les points de collocation de GLL.
- La routine POIDS_GLOB qui calcule les poids de la quadrature de Gauss-Lobatto.
- La routine GAUSS qui calcule les points de Gauss.
- La routine POIDS_GAUSS qui calcule les poids de la quadrature de Gauss.

CHAPITRE 2

LES ÉQUATIONS DE NAVIER STOKES

2.1 Introduction

L'état d'un écoulement instationnaire compressible d'un fluide newtonien visqueux est décrit par : le champ de vitesse **u**, la masse volumique ρ, la pression p et la température T. Ces variables sont des fonctions des coordonnées et du temps, et sont déterminées à l'aide des équations de base suivantes :

- L'équation de continuité qui traduit la conservation de la masse,
- L'équation de la quantité de mouvement donnée par la seconde loi de Newton,
- L'équation d'énergie qui exprime la conservation de l'énergie,
- L'équation d'état (ou équation constitutive) du fluide qui relie les variables d'état p, ρ et T.

Ces équations de conservation, appelées aussi, équations de Navier-Stokes peuvent être écrites sous diverses formes dépendemment du choix des variables indépendantes. A fin d'établir le choix approprié des variables indépendantes, on va discuter chacune des formes des équations :

2.2 Les diverses formes des équations

2.2.1 Forme conservative

Les équations de conservation de la masse, de la quantité de mouvement et de l'énergie, gouvernant l'état d'un écoulement instationnaire compressible d'un fluide newtonien

visqueux en tout point d'un domaine Ω circonscrit d'une frontière Γ, s'écrivent, sous forme conservative :

- Équation de continuité :

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho \mathbf{u}) = 0 \tag{2.1a}$$

- Équation de quantité de mouvement (Navier-Stokes) :

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla.(\rho \mathbf{u} \otimes \mathbf{u}) + \nabla p = \nabla . \boldsymbol{\sigma} + \rho \mathbf{f} \tag{2.1b}$$

- Équation de l'énergie :

$$\begin{aligned} &\frac{\partial}{\partial t}\left[\rho\left(i + \frac{1}{2}\| \mathbf{u} \|^2 \right) \right] + \nabla.\left[\rho \mathbf{u}\left(i + \frac{1}{2}\| \mathbf{u} \|^2 \right) \right] = \\ &\quad -\nabla.(p\mathbf{u}) + \nabla.(\boldsymbol{\sigma}.\mathbf{u}) - \nabla.\mathbf{q} + r + \rho \mathbf{f}.\mathbf{u} \end{aligned} \tag{2.1c}$$

avec

$$\left.\begin{aligned} e &= \rho\left(T + \frac{1}{2}\| \mathbf{u} \|^2 \right) \\ \rho(\mathbf{u}) &= \lambda\ (\nabla.\mathbf{u})\mathbf{I} + 2\mu \mathbf{D}_{ij} \\ \mathbf{q} &= -\ k \nabla T \\ i &= C_v T \end{aligned}\right\} \tag{2.2}$$

où, $\mathbf{u}$, ρ, p, e, i, σ, $\mathbf{q}$, $\rho\mathbf{f}$ et r sont respectivement le vecteur vitesse, la densité, la pression, l'énergie totale, l'énergie interne, le tenseur de contraintes visqueuses, le flux

de chaleur, les forces volumiques et la source de chaleur. k étant la conductivité thermique, C_v est la chaleur spécifique à volume constant et $\mathbf{D_{ij}}$ est le tenseur taux de déformations qui est donnée par la relation suivante :

$$\mathbf{D_{ij}} = \frac{1}{2}\left[\nabla \mathbf{u} + (\nabla \mathbf{u})^t\right]$$

λ et μ sont les constantes de Lamé que l'on suppose reliées par la loi de Stokes;

$$2\mu + 3\lambda = 0$$

A ces équations, s'ajoute l'équation d'état du fluide, qui, pour un gaz parfait s'écrit :

$$p = \rho R T \tag{2.3}$$

où R est la constante des gaz parfaits ($R = 0.287$ *kJ/(kg.K)* pour l'air).

2.2.2 Forme non-conservative

Afin d'obtenir la forme non-conservative qui utilise comme variables indépendantes la densité ρ, la vitesse $\mathbf{u}$ et la température T, il faut transformer le système d'équations (2-1) de façon à faire apparaître ces variables et leurs dérivées.

Pour ce faire, on va utiliser les identités suivantes :

$$\left.\begin{aligned}
\nabla.(\rho\mathbf{u}) &= \rho\,\nabla.\mathbf{u} + \mathbf{u}.\nabla\rho \\
\nabla.(\rho\mathbf{u}\otimes\mathbf{u}) &= (\mathbf{u}\otimes\mathbf{u}).\nabla\rho + \rho\mathbf{u}\,\nabla.\mathbf{u} + \rho\mathbf{u}.\nabla\mathbf{u} \\
\frac{\partial\rho\mathbf{u}}{\partial t} &= \rho\frac{\partial\mathbf{u}}{\partial t} - \rho\mathbf{u}\nabla.\mathbf{u} - (\mathbf{u}\otimes\mathbf{u}).\nabla\rho \\
\nabla.(p\mathbf{u}) &= p\,\nabla.\mathbf{u} + \mathbf{u}.\nabla p \\
\nabla.(\mathbf{u}.\boldsymbol{\sigma}) &= \nabla\mathbf{u}:\boldsymbol{\sigma} + \mathbf{u}.\nabla.\boldsymbol{\sigma}
\end{aligned}\right\} \qquad (2.4)$$

Après quelques simplifications, on déduit la forme non-conservative des équations (2.1a) et (2.1b) et (2.1c):

$$\frac{\partial\rho}{\partial t} + \mathbf{u}.\nabla\rho + \rho\nabla.\mathbf{u} = 0 \qquad (2.5a)$$

$$\rho\frac{\partial\mathbf{u}}{\partial t} + \rho(\mathbf{u}.\nabla)\mathbf{u} + \nabla p - \nabla.\boldsymbol{\sigma} = \mathbf{f} \qquad (2.5b)$$

$$\rho\frac{\partial i}{\partial t} + \rho\mathbf{u}.\nabla i - p\nabla.\mathbf{u} - \nabla\mathbf{u}.\boldsymbol{\sigma} + \nabla.\mathbf{q} = \rho\,\mathbf{f}.\mathbf{u} + r \qquad (2.5b)$$

Il est à souligner que le système d'équations (2.5) est sous la forme finale de la formulation dite non-conservative : l'équation (2.5a) peut être considérée comme celle de la densité, (2.5b) comme celle de la vitesse et l'équation de (2.5c) comme celle de la température.

2.2.3 Forme conservative en variables conservatives

Cette formulation utilise comme variables indépendantes, la densité ρ, la quantité de mouvement $\mathbf{U} = \rho\,\mathbf{u}$ et l'énergie totale par unité de volume $E = \rho\,e$. En effectuant ces

changements de variables dans le système d'équations (2.1) on aboutit à la forme conservative en variables conservatives qui s'exprime comme suit :

- Équation de continuité :

$$\frac{\partial \rho}{\partial t} + \nabla.\mathbf{U} = 0 \tag{2.6a}$$

- Équation de quantité de mouvement :

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla.(\mathbf{U} \otimes \mathbf{u}) + \nabla p - \nabla.\boldsymbol{\sigma} = \mathbf{f} \tag{2.6b}$$

- Équation d'énergie :

$$\frac{\partial E}{\partial t} + \nabla.[(E+p)\mathbf{u}] - \nabla.(\boldsymbol{\sigma}.\mathbf{u}) - \nabla.\mathbf{q} = r + \mathbf{f}.\mathbf{U} \tag{2.6c}$$

avec

$$\left.\begin{aligned} \mathbf{U} &= \rho \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} \\ E &= \rho\left(i + \frac{u_1^2 + u_2^2 + u_3^2}{2}\right) \\ \sigma &= \mu\left[\nabla\mathbf{u} + (\nabla\mathbf{u})^t - \frac{2}{3}(\nabla.\mathbf{u})\mathbf{I}\right] \\ \mathbf{q} &= -\lambda \nabla T \\ i &= C_v T \end{aligned}\right\} \tag{2.7}$$

Jusqu'à présent, on a fait appel à trois formulations possibles. Dans la littérature, il existe bien sûr d'autres types de formulations plausibles, en particulier la formulation conservative en variables enthalpiques utilisant comme variables indépendantes la pression statique p, la quantité de mouvement **U** et l'enthalpie massique h. Le choix de la forme d'équations qui fera l'objet de ce travail de recherche sera justifié au paragraphe suivant.

2.3 Choix de la forme d'équation

Pour obtenir la forme non-conservative (voir section 2.1.2), on a dû faire certaines manipulations qui ne sont permises que si l'on suppose que la vitesse **u** et la densité ρ sont dérivables; par exemple :

$$\frac{\partial \sigma_{ij} u_i}{\partial x_j} = u_i \frac{\partial \sigma_{ij}}{\partial x_j} + \sigma_{ij} \frac{\partial u_i}{\partial x_j}$$

ce qui n'est vrai que si u_i et σ_{ij} sont dérivables. De même l'identité :

$$\frac{\partial \rho u_i u_j}{\partial x_j} = u_i \frac{\partial \rho u_j}{\partial x_j} + \rho u_j \frac{\partial u_i}{\partial x_j}$$

Ce qui présuppose que ρ et u sont dérivables.

Or, dans les problèmes d'écoulements compressibles présentant des chocs, la vitesse n'est pas continue à travers le choc alors que **U** l'est. Cela suggère intuitivement qu'il faut travailler avec ρ, **U** et E au lieu de ρ, **u** et T, pour obtenir une meilleure résolution

des chocs [5]. Dans cette étude, on va adopter la formulation adimensionnelle de la forme conservative en variables conservatives.

2.4 Formulation adimensionnelle de la forme conservative en variables conservatives

L'adimensionnalisation du système d'équations (2.6) se fait en normalisant par des grandeurs de référence notées avec l'indice r (une vitesse $\mathbf{u}_r$, une densité ρ_r et une longueur L_r sont les grandeurs de base) telles que :

$$\left.\begin{aligned} \mathbf{u}^* = \frac{\mathbf{u}}{\mathbf{u}_r}, \quad p^* = \frac{p}{p_r}, \quad \rho^* = \frac{\rho}{\rho_r} \\ T^* = \frac{T}{T_r}, \quad i^* = \frac{i}{\|\mathbf{u}_r\|^2} \\ x^* = \frac{x}{L_r}, \quad t^* = \frac{t}{t_r}, \quad \mu^* = \frac{\mu}{\mu_r}, \quad \lambda^* = \frac{\lambda}{\lambda_r} \\ p_r = \rho_r \|\mathbf{u}_r\|^2, \quad t_r = L_r / \|\mathbf{u}_r\|^2, \quad T_r = \|\mathbf{u}_r\|^2 / C_v \end{aligned}\right\} \tag{2.8}$$

La forme conservative en variables conservatives des équations de conservation adimensionnelles (en omettant les *) devient alors :

- Équation de continuité :

$$\frac{\partial \rho}{\partial t} + \nabla.\mathbf{U} = 0 \tag{2.9a}$$

- **Équation de quantité de mouvement :**

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla.(\mathbf{U} \otimes \mathbf{u}) + \nabla p - \nabla.\boldsymbol{\sigma} = \mathbf{f} \tag{2.9b}$$

- **Équation d'énergie :**

$$\frac{\partial E}{\partial t} + \nabla.\left[(E + p)\mathbf{u}\right] - \nabla.(\boldsymbol{\sigma}.\mathbf{u}) - \nabla.\mathbf{q} = r + \mathbf{f}.\mathbf{U} \tag{2.9c}$$

avec

$$\left.\begin{aligned} \mathbf{u} &= \frac{\mathbf{U}}{\rho} \\ T &= \frac{E}{\rho} + \frac{\|\mathbf{U}\|^2}{2\rho^2} \\ \boldsymbol{\sigma} &= \frac{\mu}{Re}\left[\nabla\mathbf{U} + (\nabla\mathbf{U})^t - \frac{2}{3}(\nabla.\mathbf{U})\mathbf{I}\right] - \frac{\mu}{\rho^2\, Re}\left[\mathbf{U}.(\nabla\rho)^2 + (\nabla\rho).\mathbf{U}^2 - \frac{2}{3}\left(\mathbf{U}^2.\nabla\rho\right)\mathbf{I}\right] \\ \mathbf{q} &= -\frac{\gamma\,\mu}{Re\ \mathrm{Pr}}\nabla\left[\frac{E}{\rho} + \frac{\|\mathbf{U}\|^2}{2\rho^2}\right] \end{aligned}\right\} \tag{2.10}$$

Re et *Pr* sont respectivement les nombres de Reynolds et de Prandlt :

$$\left.\begin{aligned} Re &= \frac{\rho_r\, L_r\, u_r}{\mu_r} \\ \mathrm{Pr} &= \frac{\mu_r\, C_p}{\lambda_r} \end{aligned}\right\} \tag{2.11}$$

Ces deux nombres sont d'une très grande importance dans l'étude de l'écoulement, le nombre de Reynolds (Re), représente le rapport entre les forces d'inertie aux forces visqueuses. On peut voir à partir de sa définition que plus on a un Re important, plus les termes de convection sont dominants et contrairement, plus Re est faible plus les termes de diffusion sont importants.

Le nombre de Prandlt quant à lui caractérise le transfert de chaleur dans un écoulement visqueux. Il mesure le rapport des coefficients de diffusion dynamiques et thermiques d'un fluide.

Les équations de conservation (2.9) peuvent se mettre sous la forme vectorielle suivante :

$$\mathbf{V}_{,t} + \mathbf{F}_{i,i}^{conv}(\mathbf{V}) = \mathbf{F}_{i,i}^{diff}(\mathbf{V}) + \Im \tag{2.12}$$

avec

$$\mathbf{V} = \begin{Bmatrix} \rho \\ U1 \\ U2 \\ U3 \\ E \end{Bmatrix} \qquad \mathbf{F}_i^{conv} = \begin{Bmatrix} Ui \\ Ui\ u1 + p\delta 1j \\ Ui\ u2 + p\delta 2j \\ Ui\ u3 + p\delta 3j \\ E\ ui + pui \end{Bmatrix} \qquad \mathbf{F}_i^{diff} = \begin{Bmatrix} 0 \\ \sigma 1i \\ \sigma 2i \\ \sigma 3i \\ \sigma\,\mathrm{ij}\ uj - qi \end{Bmatrix} \qquad \Im = \begin{Bmatrix} 0 \\ \rho f1 \\ \rho f1 \\ \rho f1 \\ fiUi \end{Bmatrix}$$

où **V** est le vecteur des variables conservatives. $\mathbf{F}_{i,i}^{conv}$ et $\mathbf{F}_{i,i}^{diff}$ sont les vecteurs flux de convection et de diffusion respectivement, δij est le delta Kronecker (i.e. $\delta ij = 1$ pour $i = j$, et $\delta ij = 0$ pour $i \neq j$), $\Im$ étant le vecteur source.

Une forme quasi-linéaire du système serait :

$$\mathbf{V}_{,t} + \mathbf{A}_i \mathbf{V}_{,i} = \left(\mathbf{K}_{ij} \mathbf{V}_{,j} \right)_{,i} + \Im \tag{2.13}$$

où $\mathbf{A}_i$ sont les matrices jacobiennes de transformation du vecteur flux de convection telles que :

$$\mathbf{A}_i = \mathbf{F}_{i,\mathbf{V}}^{conv} = \frac{\partial \mathbf{F}_i^{conv}}{\partial \mathbf{V}}$$

$\mathbf{K}_{ij}$ sont les matrices de diffusion définies telles que:

$$\mathbf{K}_{ij} \mathbf{V}_{,j} = \mathbf{F}_i^{diff}$$

Pour les formes explicites des matrices $\mathbf{A}_i$ et $\mathbf{K}_{ij}$ voir Réf. [5].
Le choix des conditions aux limites sera discuté au chapitre 4 après l'écriture de la formulation variationnelle du problème.

2.5 Modélisation de la turbulence

2.5.1 Introduction

Le phénomène de turbulence a été analysé dès 1883 par O.Reynolds à partir d'observations réalisées à l'aide d'un procédé, devenu de nos jours très classiques, de visualisation par filtres colorés de l'écoulement dans des conduits rectilignes de section circulaire. L'expérience a été répétée depuis et elle montre que suffisamment loin de l'entrée de la conduite et à faible nombre de Reynolds, les lignes de courant sont parfaitement parallèles à l'axe de la conduite. Dans ce cas, l'écoulement est stationnaire et il est dit laminaire. A plus grand nombre de Reynolds, des instabilités apparaissent sous forme d'ondes, l'écoulement est devenu alors instationnaire tout en restant parfaitement organisé, il n'est pas encore turbulent. A très grand nombre de Reynolds, l'écoulement devient complètement irrégulier, il est dit turbulent. La turbulence n'est donc pas une propriété du fluide, mais un régime d'écoulement.
On ne peut pas dire que l'écoulement n'est plus laminaire dès qu'il est instationnaire, et on ne peut pas dire qu'il est pleinement turbulent dès qu'il n'est plus laminaire. De ce fait, il n'existe pas de définition universelle de la turbulence.

Selon Hinze [16] : *Un écoulement turbulent est un écoulement irrégulier où la vitesse, la pression, la température, etc.... varient de façon aléatoire dans le temps et dans l'espace.*

La turbulence se manifeste lorsque le nombre de Reynolds représentatif de l'écoulement devient grand. Elle est caractérisée par les propriétés suivantes :

- Les variations temporelles et spatiales de la vitesse sont aléatoires,
- Le champ de vitesse est tridimensionnel et rotationnel,
- La turbulence est un phénomène non-linéaire,

- Le fluide en écoulement turbulent peut toujours être considéré comme un milieu continu,
- La turbulence est un phénomène dissipatif,
- La capacité de mélange est accrue,
- Les écoulements turbulents ne sont pas prédictibles.

Le calcul des écoulements turbulents relève principalement de la résolution des équations de la mécanique des fluides, considérés comme milieux continus, qui permet de décrire l'évolution des champs de vitesse et température. Cependant, pour des problèmes complexes, un tel calcul sur ordinateur devient très cher, ainsi, il devient nécessaire de recourir à des techniques permettant de simuler numériquement les écoulements turbulents. La technique la plus courante, dans les applications de types industriels, consiste en une description statistique de l'écoulement. La vitesse, la température, la pression, etc... sont alors décomposées en une composante moyenne et une fluctuation turbulente. Le nouveau système d'équations obtenu régit donc le comportement de l'écoulement moyen. Cependant, la non-linéarité des équations de conservation, gouvernant l'écoulement du fluide, fait apparaître des inconnues supplémentaires sous forme de corrélations entre les composantes fluctuantes. La fermeture du système est alors réalisée au moyen d'hypothèses reposant principalement sur une représentation aussi correcte que possible des propriétés caractéristiques de la turbulence. Ces hypothèses forment un modèle de turbulence.

2.5.2 Moyennes pondérées par la masse

En écoulement turbulent des fluides compressibles, non seulement la vitesse et la pression fluctuent comme c'est le cas dans les écoulements incompressibles, mais aussi la densité, la viscosité, la conductivité, la conductivité thermique et la température T, car dès que le nombre de Mach n'est plus faible, il convient aussi à tenir compte des

variations des propriétés du fluide, telles que la masse volumique, le coefficient de viscosité ou la conductivité thermique.

Les équations instantanées écrites sous forme conservative en terme de variables conservatives :

$$\frac{\partial \rho}{\partial t} + \nabla.\mathbf{U} = 0 \tag{2.14a}$$

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla.(\mathbf{U} \otimes \mathbf{u}) + \nabla p - \nabla.\boldsymbol{\sigma} = \mathbf{f} \tag{2.14b}$$

$$\frac{\partial e}{\partial t} + \nabla.[(e+p)\mathbf{u}] - \nabla.(\boldsymbol{\sigma}.\mathbf{u}) - \nabla.\mathbf{q} = r + \mathbf{f}.\mathbf{U} \tag{2.14c}$$

Les moyennes pondérées par la masse, couramment utilisées dans le cas des fluides compressibles, ont été développées par Favre [9]. Dans ce cas, les moyennes pondérées par la masse de la vitesse, de l'enthalpie totale et de la température sont définit respectivement par:

$$\left.\begin{aligned} u_i &= \tilde{u}_i + u_i' \\ \rho &= \bar{\rho} + \rho' \\ p &= \bar{p} + p' \\ e &= \tilde{e} + e' \\ T &= \tilde{T} + T' \\ h &= \tilde{h} + h' \\ q &= \bar{q} + q' \end{aligned}\right\} \tag{2.15}$$

Il faut noter que p, ρ et q sont décomposés en terme de moyenne de Reynolds et parties fluctuantes, En remplaçant les équations (2.15) dans les équations (2.14), les équations moyennes de conservation de la masse, de la quantité de mouvement et de l'énergie s'écrivent respectivement comme suit:

$$\frac{\partial \overline{\rho}}{\partial t} + \nabla . \widetilde{\mathbf{U}} = 0 \tag{2.16a}$$

$$\frac{\partial \widetilde{\mathbf{U}}}{\partial t} + \nabla . \left(\widetilde{\mathbf{U}} \otimes \widetilde{\mathbf{u}} \right) + \nabla \overline{p} = \nabla . \overline{\boldsymbol{\sigma}} + \nabla . \overline{\boldsymbol{\tau}} + \overline{\rho}\, \mathbf{f} \tag{2.16b}$$

$$\frac{\partial \widetilde{e}}{\partial t} + \nabla . \left[(\widetilde{e} + \overline{p}) \widetilde{\mathbf{u}} \right] = \nabla . (\overline{\boldsymbol{\sigma}} . \widetilde{\mathbf{u}}) + \nabla . (\overline{\boldsymbol{\tau}} . \widetilde{\mathbf{u}}) - \nabla . \overline{\mathbf{q}} - \nabla . \overline{\mathbf{q}_t} + r + \mathbf{f} . \widetilde{\mathbf{U}} \tag{2.16c}$$

avec

$$\widetilde{\mathbf{U}} = \overline{\rho}\, \widetilde{\mathbf{u}} \tag{2.17}$$

$$\overline{\boldsymbol{\sigma}} = \overline{\mu \left[\nabla \mathbf{u} + (\nabla \mathbf{u})^{t} - \frac{2}{3} (\nabla . \mathbf{u}) \mathbf{I} \right]} \tag{2.18}$$

$$\overline{\boldsymbol{\tau}} = - \overline{\rho\, \mathbf{u}' \otimes \mathbf{u}'} \tag{2.19}$$

$$\overline{\mathbf{q}} = - \overline{\lambda\, \nabla T} \tag{2.20}$$

$$\overline{\mathbf{q}_t} = - \overline{\rho\, h' \mathbf{u}'} \tag{2.21}$$

$$\widetilde{e} = \overline{\rho}\, \widetilde{h} - \overline{p} \tag{2.22}$$

$$\overline{p} = (\gamma - 1) \overline{\rho}\, \widetilde{T} \tag{2.23}$$

Avec l'utilisation des moyennes de Reynolds et de Favre, les équations moyennes obtenues ont une forme assez simple, en particulier, l'équation de conservation de la masse qui garde sa forme habituelle. Dans l'équation de conservation de la quantité de mouvement, il apparaît des tentions turbulentes $- \overline{\rho\, \mathbf{u}' \otimes \mathbf{u}'}$ appelées aussi tensions de Reynolds. De même, l'équation de conservation de l'énergie fait intervenir un terme de

diffusion turbulente. Ce dernier terme est interprété comme un flux de chaleur turbulent. L'équation d'état (2.23) garde la même forme.

2.5.3 Fermeture du problème

À cause de la non linéarité des équations de conservation, des inconnus supplémentaires sont apparus à savoir : les tensions turbulentes $-\overline{\rho\,\mathbf{u}'\otimes\mathbf{u}'}$ et le flux de chaleur turbulent. Le système d'équations (2.17) est alors ouvert. La fermeture du système peut être réalisée à l'aide de certains modèles basés sur des hypothèses approximant le plus possible les caractéristiques de la turbulence.

Les tensions de Reynolds : Par analogie avec la loi de Newton reliant la tension visqueuse à la vitesse, Boussinesq propose une relation directe entre la tension turbulente et la vitesse moyenne qui s'écrit, pour une couche limite de fluide incompressible comme :

$$-\overline{\rho\,\mathbf{u}'\otimes\mathbf{u}'}=\mu_t\left[\nabla\mathbf{u}+(\nabla\mathbf{u})^t-\frac{2}{3}(\operatorname{div}\mathbf{u})\delta_{ij}\right]$$

Le flux de chaleur turbulent : L'équation de fermeture la plus utilisée pour le flux de chaleur turbulent $\overline{\mathbf{q_t}}$, découle de l'analogie entre la quantité de mouvement et le transfert de chaleur. $\overline{\mathbf{q_t}}$ est donc supposé proportionnel au gradient de la température :

$$\overline{\mathbf{q}_t}=-\overline{\rho\,h'\mathbf{u}'}=-\frac{\mu_t\,C_p}{Pr_t}\frac{\partial\tilde{T}}{\partial x_j}$$

où Pr_t est le nombre de Prandtl turbulent. Habituellement, les auteurs supposent que Pr_t est constant. En général, $Pr_t = 0.9$

2.5.4 Le modèle de Spalart-Allmaras

Le modèle de Spalart-Allmaras [21] est un modèle à une équation qui se base sur une équation de transport de la viscosité turbulente. Ce modèle contient huit coefficients de fermeture et trois fonctions d'amortissement. Donc, avec ce modèle on est amené à résoudre une équation de turbulence dans tout le domaine de calcul.

Toutefois, pour atteindre une bonne précision, le modèle de Spalart-Allmaras requiert un maillage très fin au voisinage de la paroi où le premier nœud dans la direction verticale doit être situé à une distance telle que $y^+ \leq 10$. Dans de telles conditions, le calcul devient très coûteux en terme de mémoire et du temps *CPU*. Une façon de remédier à ce problème consiste en l'introduction des lois de paroi pour modéliser l'écoulement dans la région de la paroi.

Le modèle contient huit coefficients de fermeture et trois fonctions d'amortissement. Les équations du modèle sont les suivantes :

- La viscosité turbulente cinématique

$$\nu_t = \tilde{\nu} f_{v1} \tag{2.24}$$

- L'équation de la viscosité turbulente

$$\frac{\partial \tilde{\nu}}{\partial t} + U_j \frac{\partial \tilde{\nu}}{\partial x_j} = Cb_1 \left[1 - f_{v2}\right] \tilde{S} \tilde{\nu} - C_{w1} f_w \left(\frac{\tilde{\nu}}{d}\right)^2 + \frac{1}{\sigma} \frac{\partial}{\partial x_k} \left[(\nu + \tilde{\nu})\right] + \frac{Cb_2}{\sigma} \frac{\partial \tilde{\nu}}{\partial x_k} \frac{\partial \tilde{\nu}}{\partial x_k} \tag{2.25}$$

- Les coefficients de fermeture

$$\left.\begin{aligned} &Cb1 = 0.1355, \qquad Cb2 = 0.622, \quad Cv1 = 7.1, \quad \sigma = \frac{2}{3} \\ &Cw1 = \frac{Cb1}{\kappa^2} + \frac{(1 + Cb2)}{\sigma}, \quad Cw2 = 0.3, \quad Cw3 = 2, \quad \kappa = 0.41 \end{aligned}\right\} \qquad (2.26)$$

- Relations auxiliaires

$$f_{v1} = \frac{\chi^3}{\chi^3 + C_{v1}^3}, \quad f_{v2} = \frac{\chi}{\chi f_{v1} + 1}, \quad f_w = g\left[\frac{1 + C_{w3}^6}{g^6 + C_{w3}^6}\right]$$

$$\chi = \frac{\tilde{\nu}}{\nu}, \quad g = r + C_{w2}\left(r^6 - r\right), \quad r = \frac{\tilde{\nu}}{\tilde{S}\,\kappa^2\,d^2}$$

$$\tilde{S} = S + \frac{\tilde{\nu}}{\kappa^2\,d^2} f_{v2}, \quad S = \sqrt{2\,\Omega_{ij}\,\Omega_{ij}}$$

Le tenseur $\Omega_{ij} = \frac{1}{2}\left(\partial U_i / \partial x_j - \partial U_j / \partial x_i\right)$ est le tenseur de rotation et d est la distance au premier point.

2.5.4.1 Le modèle de Spalart-Allmaras à grand nombre de Reynolds modifié

Dans le cas d'un écoulement à grand nombre de Reynolds, le modèle de Spalart-Allmaras s'exprime comme suit :

$$\frac{\partial \nu_t}{\partial t} + \mathbf{u}.\nabla \nu_t - \frac{1}{Re\sigma}\left[\nabla.\left(\nu_t \nabla \nu_t\right) + C_{b2}\nabla.\left(\nabla \nu_t\right)^2\right] - C_{b1}\,\omega\,\nu_t + \frac{C_{w1}}{Re} f_w \left(\frac{\nu_t}{d}\right)^2 = 0 \qquad (2.27)$$

v_t est la viscosité cinématique turbulente, ω est la vorticité et d est la distance normale à partir de la paroi. La fonction de fermeture f_w et les constantes sont données par :

Pour assurer la positivité de la viscosité turbulente v_t, on va résoudre le modèle de turbulence pour $\tilde{v}$ telle que:

$$v_t = e^{\tilde{v}} \quad \Rightarrow \quad \tilde{v} = \log v_t \tag{2.28}$$

Les termes dans le modèle de turbulence sont transformés comme suit :

$$\frac{Dv_t}{Dt} = e^{\tilde{v}} \frac{D\tilde{v}}{Dt} \tag{2.29a}$$

$$Cb_1 S v_t = Cb_1 S e^{\tilde{v}} \tag{2.29b}$$

$$\nabla v_t = e^{\tilde{v}} \nabla \tilde{v} \tag{2.29c}$$

$$v_t \nabla v_t = e^{2\tilde{v}} \nabla \tilde{v} \tag{2.29d}$$

$$\nabla.(v_t \nabla v_t) = \nabla.\left(e^{2\tilde{v}} \nabla \tilde{v}\right) = e^{2\tilde{v}} \nabla.(\nabla \tilde{v}) + 2e^{2\tilde{v}} (\nabla \tilde{v})^2 \tag{2.29e}$$

$$(\nabla v_t)^2 = \left(e^{\tilde{v}} \nabla \tilde{v}\right)^2 = e^{2\tilde{v}} (\nabla \tilde{v})^2 \tag{2.29f}$$

$$\frac{1}{\sigma Re}\left[\nabla.(v_t \nabla v_t) + Cb2\ (\nabla v_t)^2\right] = \frac{e^{2\tilde{v}}}{\sigma Re}\left[\nabla.(\nabla \tilde{v}) + (2 + Cb2)(\nabla \tilde{v})^2\right]$$

$$= \frac{e^{\tilde{v}}}{\sigma Re}\left[\nabla.\left(e^{\tilde{v}} \nabla \tilde{v}\right) + (1 + Cb2) e^{\tilde{v}} (\nabla \tilde{v})^2\right] \tag{2.29g}$$

2.5.4.2 Le Modèle de Spalart-Allmaras sous forme adimensionnelle

$$\frac{\partial v_t}{\partial t} + \mathbf{u}.\nabla v_t - \frac{1}{Re\,\sigma}\left[\nabla.(v_t \nabla v_t) + Cb2\,\nabla.(\nabla v_t)^2\right]$$

$$- Cb1\,\omega\, v_t + \frac{Cw1}{Re} f_w \left(\frac{v_t}{d}\right)^2 = 0 \qquad (2.30)$$

avec

$$f_w = g\left[\frac{1 + Cw3^6}{g^6 + Cw3^6}\right]^{\frac{1}{6}}, \quad g = r + Cw2\left(r^6 - r\right), \quad r = \frac{v_t}{Re\,S\,\kappa^2\,d^2}$$

où $S = \left|\nabla \times \mathbf{u}\right|$ est la vorticité.

2.5.5 Les lois de paroi

Dans la modélisation des écoulements turbulents, on fait souvent appel aux lois dites de la paroi pour modéliser les régions très proches de la paroi où la turbulence n'est plus isotrope et où les modèles de turbulence habituels ne sont plus applicables. Cette région est composée de trois couches distinctes, une couche où la viscosité moléculaire est dominante (la sous-couche laminaire), une couche à l'extrémité externe de la région de paroi où la viscosité turbulente est dominante (la sous-couche inertielle) et une couche comprise entre les deux où les effets des deux types de viscosité sont présents (région tampon).

Le comportement physique de l'écoulement dans la région de la paroi est connu pour le cas d'une plaque plane sans gradient de pression, c'est un comportement qui est linéaire dans la sous couche laminaire, logarithmique dans la deuxième couche et un comportement logarithmique dans la troisième couche. Au chapitre 4, nous présenterons

une nouvelle approche pour implémenter cette loi. Cette dernière consiste en la construction d'un élément spéciale appelé élément Log-Lin.

Ce comportement est aussi connu par la théorie. Il est donné par la loi de Spalding. Nous allons présenter la loi de Spalding et la comparer avec le comportement donné par l'élément Log-Lin.

2.5.6 La loi de Spalding

La fonction de paroi utilisée ici consiste en la loi de paroi développée par Spalding, qui modélise la sous-couche laminaire la région intermédiaire et la région extérieure

$$y^+ = u^+ + e^{-\kappa B}\left[e^{-\kappa u^+} - 1 - \kappa u^+ - \frac{(\kappa u^+)^2}{2} - \frac{(\kappa u^+)^3}{6} \right] \qquad (2.31)$$

avec

$$y^+ = \frac{Re\, \rho\, y\, u_\tau}{\mu}, \quad u^+ = \frac{\| u \|}{u_\tau}, \quad B = 5.5 \qquad (2.32)$$

u_τ est la vitesse de frottement et y est la distance normale à partir de la paroi.

La viscosité cinématique turbulente sur la paroi est calculée de la façon suivante:

$$\nu_t = Re\, u_\tau\, \kappa\, \delta \qquad (2.33)$$

Une comparaison du profil de vitesse donné par l'élément Log-Lin et celui donné avec la loi de paroi de Spalding est présentée à la figure 11:

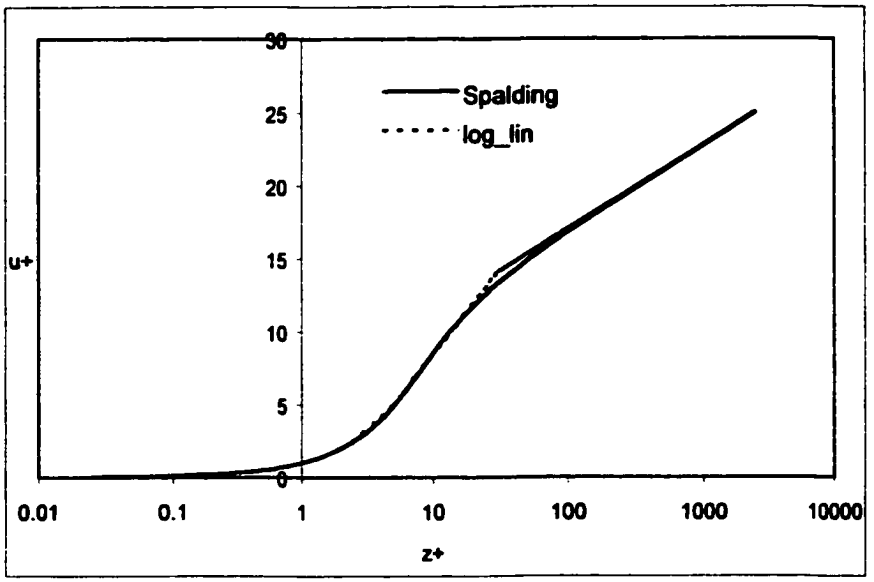


Figure 11- Comparaison du profil Log-Lin avec la loi de paroi de Spalding

2.5.7 Cas d'un gradient de pression

En présence d'un gradient de pression, le comportement logarithmique dans la région de la paroi n'est plus valable et nécessite une correction qui tiendrait compte du gradient de pression. Le champ de vitesse peut même être séparé. Coles [12, 17] a introduit une solution à ce problème qui consiste en la loi de paroi à laquelle est ajouté un terme additionnel traduisant la déviation de profil de vitesse due à un gradient de pression inverse,

$$u^+ = \frac{1}{\kappa} \ln (y^+) + C + \frac{2\Pi}{\kappa} f\left(\frac{y}{\delta}\right) \qquad (2.34)$$

La fonction $f\left(\frac{y}{\delta}\right)$ est appelée fonction de sillage. Elle exprime la déviation du profil de vitesse par rapport au profil logarithmique. Elle est donnée par:

$$f\left(\frac{y}{\delta}\right)=\frac{u^{+}-u^{+}{}_{Log}}{U_{e}^{+}-u^{+}{}_{Log}\Big|_{y=\delta}} \tag{2.35}$$

où U_e^+ est la vitesse à l'infini normalisée par la vitesse de friction.

Π est appelé le paramètre de Coles. Il est directement lié au gradient de pression. Pour un comportement logarithmique pur $\Pi=0$. La figure 12 est extraite du livre de White [12]. Elle montre la corrélation de Π avec le paramètre de Clauser β (paramètre caractérisant le gradient de pression),

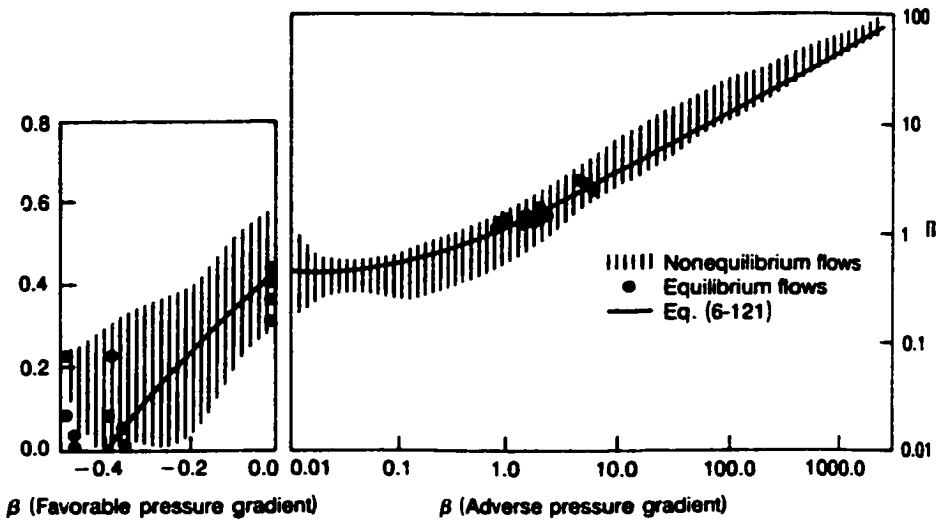


Figure 12- Corrélation de Π avec le paramètre de Clauser β

Plus le gradient de pression devient important, plus le deuxième terme devient dominant et le comportement logarithmique finit par disparaître. Ceci est montré sur la figure 13 extraite de White [12]:

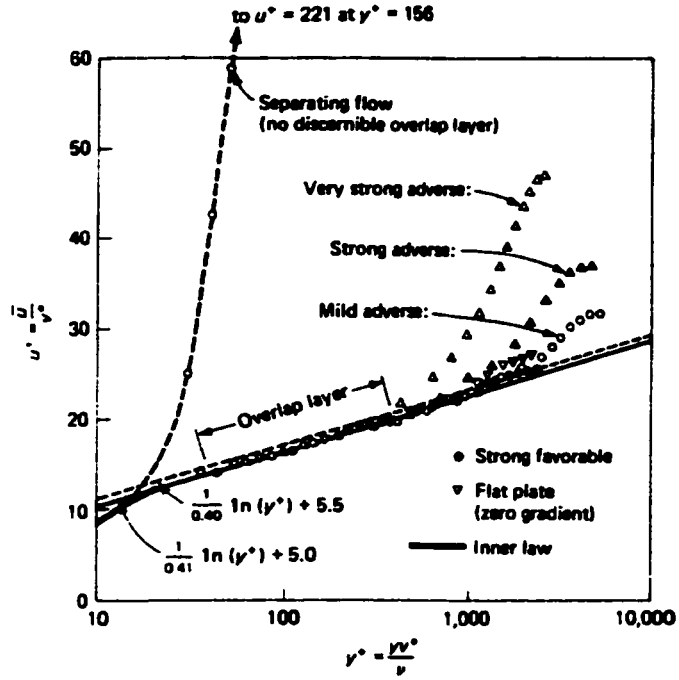


Figure 13- les profils de vitesse avec et sans gradient de pression adverse

CHAPITRE 3

FORMULATION PAR ÉLEMENTS FINIS-SPECTRAUX POUR LA MODÉLISATION DE LA PAROI

3.1 Introduction

Les équations de Navier Stokes, rappelées dans le chapitre précédent forment un système d'équations non linéaires couplées. Ces équations sont très difficiles à résoudre analytiquement sauf dans des cas très particuliers. Ainsi, il devient nécessaire de recourir à l'utilisation des méthodes numériques. Les méthodes numériques permettent de transformer le système d'équations aux dérivées partielles en un système d'équations algébriques. Parmi ces méthodes on trouve : la méthode des différences finies, la méthode des volumes finis, la méthode des éléments finis et la méthode et des éléments spectraux. Dans ce chapitre, on va discuter la mise en œuvre de deux de ces méthodes dans la discrétisation des équations de Navier-Stokes : La méthode des éléments finis et la méthodes des éléments spectraux.

La méthode des éléments finis et la méthode des éléments spectraux utilisent au départ la méthode des résidu pondérés, permettant ainsi d'obtenir pour le système d'équations une formulation intégrale dite forte (ou variationnelle forte). La pondération du résidu est effectuée selon la méthode de Galerkin. Cette méthode consiste à utiliser l'ensemble des variations des variables inconnues comme fonctions de pondération. Certains termes de la formulation intégrale sont intégrés par partie (théorème de Green), ce qui fournit la forme intégrale dite faible. Cette forme présente les avantages suivants :

- L'ordre des dérivées sur les variables inconnues apparaissant dans la forme intégrale diminue, d'où des conditions de dérivabilité moins fortes.

- Les intégrales de contour résultant permettent l'introduction de certaines conditions aux limites naturelles du problème à résoudre.

Le domaine Ω est par la suite décomposé en sous-espaces appelés éléments, l'intégrale sur le domaine complet est égale à la somme des intégrales sur chacun des éléments. Les champs des variables indépendantes sont discrétisés en les approximant par le produit des fonctions d'interpolation, et de coefficients inconnus. Ces coefficients sont les variables recherchées, évaluées aux nœuds, de coordonnées connues et situées sur les éléments. Cette formulation discrétisée fournit un système matriciel pour un nombre n d'inconnues.

3.2 Élément de paroi

Notre domaine de calcul sera décomposé en deux régions différentes : la région de la paroi (éléments prismatiques) et la région externe (éléments tétraédriques) (figure 14). Dans la région externe, les équations de Navier Stokes sont discrétisées par une méthode d'éléments finis stabilisées classique [5]. Dans la région de la paroi, seule la quantité de mouvement en x qui va subir l'approximation Log-Lin-Spectrale dans la direction perpendiculaire, Les autres équations de Navier-Stokes ainsi que l'équation de la viscosité turbulente sont approximées par la méthode des éléments finis classique. Dans un premièr lieu, on présentera la formulation variationnelle du modèle mathématique développé au chapitre 2 ainsi que les conditions aux limites. Ensuite, on va discuter la discrétisation des équations dans la région externe et dans la région de la paroi.

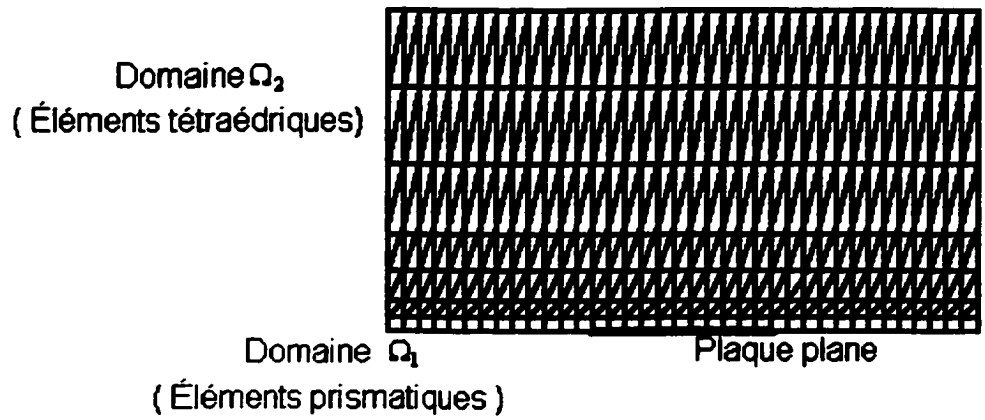


Figure 14- Maillage du domaine de calcul : éléments prismatiques dans la couche visqueuse et éléments tétraédriques dans le reste du domaine

3.3 Formulation variationnelle

A fin de déterminer numériquement une solution pour le système d'équations (2-14), on va pondérer notre système d 'équations (2.14) par des fonctions de pondération suivant l'approximation de Galerkin. Par la suite, on intègre ces équations sur le domaine de calcul Ω, ce qui donne ainsi la formulation variationnelle forte. Les fonctions de pondération des variables ρ, $\mathbf{U}$ et E sont dénotées respectivement par q, ω et θ qui multiplient tour à tour les équations (2-14a) à (2-14c) :

$$W_\rho = \int_\Omega \mathbf{q}.\left[\frac{\partial \rho}{\partial t} + \nabla.(\mathbf{U})\right] d\Omega = 0 \qquad (3.1a)$$

$$W_\rho = \int_\Omega \boldsymbol{\omega}.\left[\frac{\partial \mathbf{U}}{\partial t} + \nabla.(\mathbf{U}\otimes\mathbf{u}) + \nabla p - \underline{\nabla.\boldsymbol{\sigma}} - \mathbf{f}\right] d\Omega = 0 \qquad (3.1b)$$

$$W_E = \int_\Omega \theta.\left[\frac{\partial E}{\partial t} + \nabla.[(E+p)\mathbf{u}] - \underline{\nabla(\boldsymbol{\sigma}.\mathbf{u})} + \underline{\nabla.\mathbf{q}} - \mathbf{f}.\mathbf{U} - r\right] d\Omega = 0 \qquad (3.1c)$$

Afin de réduire l'ordre de dérivation de deux et ainsi satisfaire aux exigences de continuité, les termes soulignés dans les équations (3.1b) et (3.1c) sont sujets à l'intégration par parties. L'utilisation du théorème de Green permet de faire apparaître les intégrales de contour qui peuvent servir à imposer les conditions aux limites naturelles. Nous notons par (f, g) le produit scalaire $L^2(\Omega)$ donné par:

$$(f,g)= \int_{\Omega} fg \, d\Omega$$

Après intégration par parties, le système variationnel (3.1) devient donc :

$$\left(\mathrm{q},\frac{\partial \rho}{\partial t}\right)+b_2\left(\mathbf{U},\mathrm{q}\right)=0 \tag{3.2a}$$

$$\left(\boldsymbol{\omega},\frac{\partial \mathbf{U}}{\partial t}\right)+a\left(\boldsymbol{\omega},\mathbf{U},\rho\right)+c\left(\boldsymbol{\omega},\mathbf{U},\mathbf{U}\right)+b_1\left(\boldsymbol{\omega},p\right)=\left(\mathbf{f},\boldsymbol{\omega}\right)+\left\langle\boldsymbol{\omega},\boldsymbol{\sigma}\right\rangle \tag{3.2b}$$

$$\left(\theta,\frac{\partial e}{\partial t}\right)+\left(\theta,\nabla.\left[\left(\frac{e+p}{\rho}\right)\mathbf{U}\right]\right)-\left(\nabla\theta,\psi\right)+\left(\nabla\theta,\frac{\boldsymbol{\sigma}.\mathbf{U}}{\rho}\right)=$$

$$\left(\theta,r+\mathbf{f}.\mathbf{U}\right)+\left\langle\theta,\psi\right\rangle+\left\langle\theta,\frac{\boldsymbol{\sigma}.\mathbf{U}}{\rho}\right\rangle \tag{3.2c}$$

avec

$$\left.\begin{aligned} a(\boldsymbol{\omega},\mathbf{U},\rho) &= \int_{\Omega} \nabla\boldsymbol{\omega} : \boldsymbol{\sigma}\, d\Omega \\ c(\boldsymbol{\omega},\mathbf{U},\mathbf{U}) &= \int_{\Omega} \boldsymbol{\omega} . \nabla\left(\frac{\mathbf{U}\otimes\mathbf{U}}{\rho}\right) d\Omega \\ b_1(\boldsymbol{\omega},p) &= \int_{\Omega} \boldsymbol{\omega} . \nabla p\, d\Omega \\ b_2(\boldsymbol{\omega},p) &= \int_{\Omega} p\, \nabla . \boldsymbol{\omega}\, d\Omega \\ \langle \boldsymbol{\omega},\boldsymbol{\sigma} \rangle &= \int_{\Gamma} \boldsymbol{\omega} . (\boldsymbol{\sigma}.\mathbf{n})\, d\Gamma \\ \langle \theta,\psi \rangle &= \int_{\Gamma} \theta\psi . \mathbf{n}\, d\Gamma \end{aligned}\right\} \qquad (3.3)$$

L'écriture sous forme vectorielle du système (3.2) est :

$$\int_{\Omega} \left\{ \mathbf{W} . \left[\mathbf{A}_0 \mathbf{V}_{,t} + \mathbf{F}^{conv}_{i,i}(\mathbf{V}) - \mathfrak{I} \right] + \mathbf{W}_{,i} . \left[\mathbf{F}^{diff}_{i,i}(\mathbf{V}) \right] \right\} d\Omega = \oint_{\Gamma} \left\{ \mathbf{W} . \left[\mathbf{F}^{diff}_{i,i}(\mathbf{V}) . \mathbf{n}_i \right] \right\} d\Gamma \qquad (3.4)$$

où $\mathbf{W} = (q, \boldsymbol{\omega}, \theta)^t$ est le vecteur des fonctions de pondération et $\mathbf{V} = (\rho, \mathbf{U}, E)^t$. Sous la forme quasi-linéaire, le système (3.2) s'écrit :

$$\int_{\Omega} \left\{ \mathbf{W} . \left[\mathbf{A}_0 \mathbf{V}_{,t} + \mathbf{A}_i \mathbf{V}_{,i} - \mathfrak{I} \right] + \mathbf{W}_{,i} . \left[\mathbf{K}_{ij} \mathbf{V}_j \right] \right\} d\Omega = \oint_{\Gamma} \left\{ \mathbf{W} . \left[\left(\mathbf{K}_{ij} \mathbf{V}_j \right) . \mathbf{n}_i \right] \right\} d\Gamma \qquad (3.5)$$

3.4 Conditions aux limites

Dans ce travail, on va s'attarder sur le cas d'un écoulement visqueux compressible tridimensionnel turbulent sur une plaque plane. On suppose que l'écoulement est uniforme à l'infini et que les variables sont normalisées par leurs valeurs à l'infini. On pose:

$$\mathbf{U}_\infty = \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix}, \qquad \text{où } \alpha \text{ est l'angle d'attaque.}$$

$$\rho_\infty = 1$$

$$E_\infty = \frac{1}{2} + \frac{1}{\gamma(\gamma-1)M_\infty{}^2}, \qquad \text{où } M_\infty \text{ est le nombre de Mach à l'infini}$$

$$\mathbf{V}_\infty = (\rho_\infty, \mathbf{U}_\infty, E_\infty)^t$$

Quatre types de conditions aux limites sont à imposer, à savoir:

- Conditions à l'entrée
- Conditions à la sortie
- Condition de symétrie
- Conditions à la paroi solide

3.4.1 Conditions à l'entrée

A l'entrée, les conditions aux limites sont de type Dirichlet sur la quantité de mouvement, la densité et l'énergie. Ces variables sont posées égale à leurs valeurs à l'infini $\mathbf{u} = \mathbf{U}_\infty$, $\rho = \rho_\infty$ et $E = E_\infty$. Dans ce cas, les termes de contour dans les formes variationnelles (3.2b) et (3.2c) s'annulent.

3.4.2 Conditions à la sortie

Pour ce qui est des conditions aux limites à la sortie, il faut distinguer deux cas :

- Si l'écoulement est subsonique, une seule condition de Dirichlet est imposée. En général, cette condition correspond à la densité.
- Si l'écoulement est supersonique, aucune condition limite n'est nécessaire.

Si la frontière de sortie est suffisamment loin du corps, l'écoulement est supposé établi. Ainsi, les termes de contour dans les formes variationnelles (3.2b) et (3.2c) sont considérés nuls.

Remarque: Une modification de la formulation variationnelle (3.5) peut englober les conditions limites à l'infini de manière faible, Soulaïmani [3]:

$$\int_{\Omega} \left\{ \mathbf{W} \cdot \left[\mathbf{A}_0 \mathbf{V}_{,t} + \mathbf{A}_i \mathbf{V}_{,i} - \mathfrak{I} \right] + \mathbf{W}_{,i} \cdot \left[\mathbf{K}_{ij} \mathbf{V}_j \right] \right\} d\Omega$$

$$= \oint_{\Gamma} \left\{ \mathbf{W} \cdot \left[\left(\mathbf{K}_{ij} \mathbf{V}_j \right) \cdot \mathbf{n}_i \right] \right\} d\Gamma + \int_{\Gamma} \mathbf{W} \mathbf{A}_n^- \left(\mathbf{V} - \mathbf{V}_\infty \right) d\Gamma \qquad (3.5)$$

avec

$A_n = \sum_i A_i n_i$ où $\vec{n}$ est la normale à la frontière,

$A_n^- = S \Lambda^- S^{-1}$ où Λ étant les valeurs propres de A_n et Λ^- la partie négative.

3.4.3 Conditions sur la paroi solide

Sur la paroi solide, on impose les conditions suivantes :

- Condition d'adhérence: La viscosité du fluide impose que les particules du fluide restent attachées à la paroi solide. Donc on doit imposer une vitesse nulle à la paroi,

$$\mathbf{U}=0 \quad \text{ou} \quad \mathbf{u}=0$$

Dans le cas d'un écoulement non visqueux, on doit imposer une condition d'imperméabilité qui se traduit par une vitesse normale du fluide nulle le long de la paroi,

$$\mathbf{U}.\mathbf{n}=0 \quad \text{ou} \quad \mathbf{u}.\mathbf{n}=0$$

- Aussi, on impose soit une condition de type Dirichlet sous forme d'une répartition de l'énergie par unité de volume (donc une répartition de la température) sur la paroi solide (paroi isotherme), soit une condition de Newman en isolant la paroi de l'extérieur en considérant que le flux de chaleur normal à la paroi soit nul (paroi adiabatique).

  - Condition de Dirichlet :

$$E=\rho T_w=\rho\left(\frac{1}{\gamma(\gamma-1)M_\infty{}^2}\right)\left(1+\frac{\gamma-1}{2}M_\infty{}^2\right)$$

  - Condition de Newman :

$$\mathbf{q}.\mathbf{n}=0$$

Sur une paroi adiabatique ou isotherme, où la condition d'adhérence est imposée, Les termes de contour dans les formes variationnelles (3.2b) et (3.2c) s'annulent.

3.4.4 Condition de symétrie

L'écoulement étudié est symétrique par rapport à la paroi. Seule la moitié du domaine est alors considérée et la condition de symétrie est assurée en imposant la quantité de mouvement transversale à zéro le long du plan de symétrie.

Encore, sur le plan de symétrie, les termes de contour dans les formes variationnelles (3.2b) et (3.2c) sont nuls.

3.5 L'équation de Spalart-Allmaras

La formulation variationnelle de l'équation de Spalart-Allmaras après changement de variable de la viscosité turbulente s'écrit :

$$\int_{\Omega^e} \varphi \frac{\partial \nu_t}{\partial t} d\Omega^e + \int_{\Omega^e} \varphi (u . \nabla \nu_t) d\Omega^e - \frac{1}{Re\sigma} \int_{\Omega^e} \nabla\varphi . (\nu_t \nabla \nu_t) d\Omega^e - Cb1 \int_{\Omega^e} \varphi S\nu_t \, d\,d\Omega^e$$

$$- \frac{Cb2}{\sigma Re} \int_{\Omega^e} \varphi (\nabla \nu_t)^2 d\Omega^e - \frac{Cw1 fw}{Re} \int_{\Omega^e} \nabla\varphi \frac{\nu_t^3}{3 d^2} d\Omega^e - \frac{1}{Re\sigma} \int_{\Gamma} \varphi (\nu_t \nabla \nu_t) . \mathbf{n} \, d\Gamma$$

$$+ \frac{Cw1 fw}{Re} \int_{\Gamma} \nabla\varphi . \left[\left(\frac{\nu_t}{d} \right)^2 \mathbf{n} \right] d\Gamma = 0 \qquad (3.6)$$

où φ sont les fonctions pondération.

Cette forme pourrait être simplifiée en introduisant les simplifications suivantes :

- Au-delà de la distance d suffisamment loin de la paroi, le terme $\frac{Cw1\,fw}{Re} \int_{\Omega^e} \nabla\varphi \, \frac{v_t^3}{3d^2} \, d\Omega^e$ devient négligeable,

- Avec la condition limite $(\nabla v_t \,.\, \mathbf{n}) = 0$, le terme de frontière $\frac{1}{Re\,\sigma} \int_{\Gamma} \varphi \,(v_t \nabla v_t)\,.\,\mathbf{n} \; d\Omega^e$ s'annule.

La forme variationnelle se réduit alors à :

$$\int_{\Omega^e} \varphi \frac{\partial v_t}{\partial t} \, d\Omega^e + \int_{\Omega^e} \varphi (\mathbf{u} \,.\, \nabla v_t) \, d\Omega^e - \frac{1}{Re\sigma} \int_{\Omega^e} \nabla\varphi \,.\, (v_t \nabla v_t) \, d\Omega^e$$

$$- Cb1 \int_{\Omega^e} \varphi S v_t \, d \; d\Omega^e - \frac{Cw1\,fw}{Re} \int_{\Omega^e} \nabla\varphi \, \frac{v_t^3}{3d^2} \, d\Omega^e = 0 \qquad (3.7)$$

3.6 Discrétisation spatiale

Le domaine du fluide $\Omega \subset \mathbf{R}^3$ est subdivisé en sous domaines Ω^e appelés éléments. Ces éléments sont de dimension finie $\Omega^e \subset \Omega$ pour lesquels on peut facilement expliciter une base polynomiale. L'intégrale sur le domaine sera alors une combinaison linéaire des intégrales sur chaque élément du domaine :

$$\int_{\Omega} f(x) \, d\Omega = \sum_{i=1}^{nelt} \int_{\Omega^e} f(x) \, d\Omega^e$$

où *nelt* est le nombre d'éléments total.

Chaque élément Ω^e doit être définit analytiquement, de manière unique, en fonction des nœuds géométriques. Afin de simplifier la définition analytique des éléments analytiques, nous utilisons la notion d'élément de référence. Un élément de référence Ω_0 est un élément de forme très simple, repéré dans un espace de référence, qui peut être transformé en chaque élément réel Ω^e par une transformation géométrique τe :

$$\tau^e : \xi(\xi,\eta,\zeta) \rightarrow \mathbf{X}(x,y,z)$$

ξ peut être considéré comme un système de coordonnées local lié à chaque élément.

Une quantité quelconque u est approximée sur chaque élément réel Ω^e comme :

$$u(\mathbf{X}) = \langle N(\mathbf{X}) \rangle . \{u_n\} = \langle N_1, N_2, ..., N_n \rangle . \begin{Bmatrix} u_1 \\ u_2 \\ : \\ : \\ u_n \end{Bmatrix} \tag{3.8}$$

où

$\langle N(\mathbf{X}) \rangle$: l'ensemble des fonctions d'interpolation sur l'élément de réel,

n : le nombre de nœuds d'un élément sur lequel u est appoximée,

$\{u_n\}$: l'ensemble des valeurs de u aux nœuds de l'élément.

Sur l'élément de référence Ω_0, l'approximation de u s'écrit :

$$u(\xi) = \langle N(\xi) \rangle . \{u_n\} = \langle N_1, N_2, ..., N_n \rangle . \begin{Bmatrix} u_1 \\ u_2 \\ : \\ : \\ u_n \end{Bmatrix} \tag{3.9}$$

avec :

$$\tau^e : \xi(\xi, \eta, \zeta) \rightarrow \mathbf{X}(x,y,z) = \left\langle \overline{N}(\xi) \right\rangle \mathbf{X}_n$$

où $\langle N(\xi) \rangle$ est l'ensemble des fonctions d'interpolation sur l'élément de référence et $\left\langle \overline{N}(\mathbf{X}) \right\rangle$ l'ensemble es fonctions de transformation géométrique de l'élément réel Ω^e en élément de référence Ω_0.

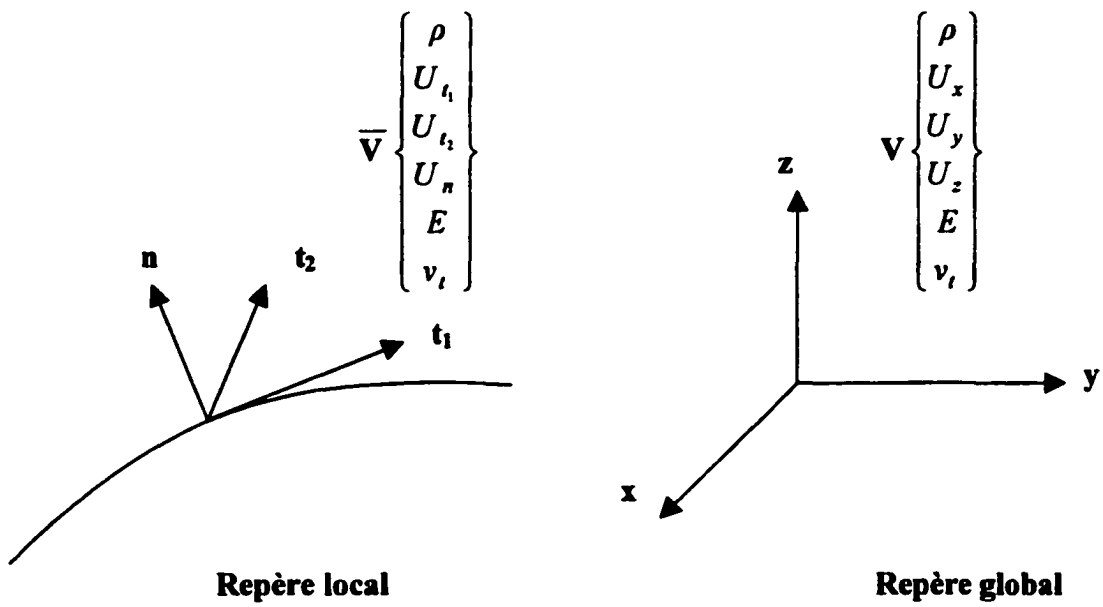
Si les nœuds géométriques coïncident avec les nœuds d'interpolation, les fonctions $\langle N(\xi) \rangle$ et $\left\langle \overline{N}(\xi) \right\rangle$ sont alors identiques et l'élément est dit isoparamétrique.

Dans ce travail, on fait appel à deux types d'éléments : Prismatiques, pour la discrétisation de la région visqueuse et Tétraédrique dans le reste du domaine. Les éléments prismatiques sont utilisés pour une meilleure modélisation de l'écoulement dans la région de la couche limite. Ceci est assuré en utilisant des interpolations spéciales dans la direction normale pour approximer la quantité de mouvement en x. Dans un premier temps, on va envisager une approximation de type Log-Lin pour modéliser les lois de parois. Ensuite, on va rajouter une approximation spectrale au profil Log-Lin pour tenir compte des phénomènes de séparation. Dans le reste du domaine, les variables indépendantes ρ, $\mathbf{U}$ et E sont approximées par des polynômes d'ordre 1 sur le tétraèdre.

3.6.1 Discrétisation dans la région visqueuse proche de la paroi

Pour une meilleure modélisation de la couche limite, le domaine de calcul dans cette région est discrétisé à l'aide d'éléments prismatiques. Le choix de l'interpolation doit être fait de manière à bien modéliser les phénomènes physiques dans cette région (turbulence et séparation) et par conséquent améliorer la solution approchée du problème. En effet, pour une meilleure modélisation de la turbulence, on va utiliser l'interpolation Log-Lin décrite dans le chapitre précédent pour incorporer les lois de paroi. Ensuite, on va mettre en œuvre une interpolation spectrale pour enrichire la quantité de mouvement en x dans la direction verticale. Pour ce faire, on va rajouter des nœuds d'interpolation dans cette direction.

Remarque : Pour une surface quelconque, on doit modifier les équations (3.5) en introduisant la matrice de rotation qui permet le passage d'un repère à l'autre :



Dans la direction **n**, on a l'interpolation Log-Lin-Spectrale pour la première composante de la quantité de mouvement U_{t_1}. Le vecteurs flux dans le repère local s'écrit :

$$\overline{\mathbf{V}}_h = \langle N \rangle_{LL-S} \left\{ \overline{\mathbf{V}} \right\}$$

La formulation variationnelle dans le repère global s'écrit :

$$\int_{\Omega} \left\{ \mathbf{W}_h \cdot \left[\mathbf{A}_0 \mathbf{V}_{h,t} + \mathbf{A}_i \mathbf{V}_{h,i} - \mathfrak{I} \right] + \mathbf{W}_{h,i} \cdot \left[\mathbf{K}_{ij} \mathbf{V}_{h,j} \right] \right\} \, d\Omega$$

On peut écrire les relations suivantes :

$$\mathbf{V}_{h,t} = \frac{\partial \mathbf{V}}{\partial \overline{\mathbf{V}}} \overline{\mathbf{V}}_{h,t} = \mathbf{R} \overline{\mathbf{V}}_{h,t}$$

$$\mathbf{V}_{h,i} = \frac{\partial \mathbf{V}}{\partial \overline{\mathbf{V}}} \overline{\mathbf{V}}_{h,i} = \mathbf{R} \overline{\mathbf{V}}_{h,i}$$

$$\mathbf{V}_{h,j} = \frac{\partial \mathbf{V}}{\partial \overline{\mathbf{V}}} \overline{\mathbf{V}}_{h,j} = \mathbf{R} \overline{\mathbf{V}}_{h,j}$$

où **R** étant la matrice de rotation.

Pour les fonctions de pondérations, on a :

$$\{\mathbf{W}\} = \mathbf{R} \left\{ \overline{\mathbf{W}} \right\}$$

Si on suppose que :

$$\mathbf{W}_h = \mathbf{R}\,\overline{\mathbf{W}}_h$$

où $\mathbf{W}_h$ et $\overline{\mathbf{W}}_h$ étant réspectivement les fonctions de pondérations dans le repère global et dans le repère local,

On a :

$$\overline{\mathbf{W}}_h = \langle N \rangle_{LL-S} \{\overline{\mathbf{W}}\} = \langle N \rangle_{LL-S}\, \mathbf{R}^{\mathbf{T}}\, \{\mathbf{W}\}$$

$$\mathbf{W}_h = \underbrace{\mathbf{R}\langle N \rangle_{LL-S}\, \mathbf{R}^{\mathbf{T}}}_{\langle \mathbf{N} \rangle_g}\, \{\mathbf{W}\} = \mathbf{R}\, \langle N \rangle_{LL-S}\, \{\overline{\mathbf{W}}\}$$

$\langle \mathrm{N} \rangle_g$ étant les fonctions d'interpolation dans le repère global.

La formulation variationnelle (3.5) dans le repère local devient donc :

$$\int_{\Omega} \left\{ \overline{\mathbf{W}}_h \,.\, \mathbf{R}^{\mathbf{T}} \left[\mathbf{A}_0\, \mathbf{R}\, \overline{\mathbf{V}}_{h,t} + \mathbf{A}_i\, \mathbf{R}\, \overline{\mathbf{V}}_{h,i} - \tilde{\mathfrak{I}} \right] + \mathbf{W}_{h,i} \,.\, \mathbf{R}^{\mathbf{T}} \left[\mathbf{K}_{ij}\, \mathbf{R}\, \overline{\mathbf{V}}_{h,j} \right] \right\} \, d\Omega$$

3.6.1.2 Élément Linéaire prismatique

Pour un prisme linéaire (figure 15), les fonctions d'interpolation sont :

$$\begin{aligned}\langle N\rangle_L &= \langle\, N_1 \;\; N_2 \;\; N_3 \;\; N_4 \;\; N_5 \;\; N_6 \,\rangle \\ &= \langle \lambda a \;\; \xi a \;\; \eta a \;\; \lambda b \;\; \xi b \;\; \eta b \,\rangle \end{aligned} \tag{3.10}$$

avec

$$\left.\begin{aligned} \lambda &= 1-\xi-\eta \\ a &= (1-\zeta)/2 \\ b &= (1+\zeta)/2 \end{aligned}\right\} \tag{3.11}$$

où

$$0 \le \xi \le 1, \quad 0 \le \eta \le 1 \quad \text{et} \quad -1 \le \zeta \le 1$$

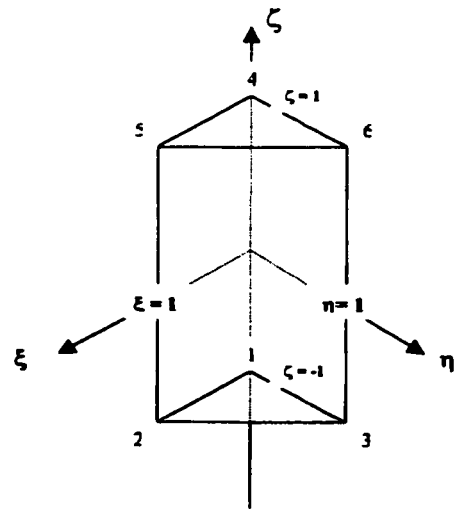


Figure 15- Élément prismatique

3.6.1.2 Élément Log-Lin

Manouzi [14] a introduit l'interpolation Log-Lin. Il a été suivi par plusieurs chercheurs qui ont appliqué cette méthode dans le cas d'écoulement incompressibles et compressibles [21].

A une section donnée selon la direction verticale, la vitesse parallèle au plan xy est interpolée de la façon suivante :

$$u(x,y,z) = R(z)\,U_0(x,y) \tag{3.12}$$

$R(z)$ est une fonction définie par tranche qui tient compte de la forme du profil selon la direction normale. U_0 est la vitesse à l'interface entre l'élément prismatique et l'élément tétraédrique.

La fonction $R(z)$ est définie comme suit [21]:

$$\left.\begin{array}{ll} R(z) = z^+ / C & z^+ \le 5 \\ R(z) = \ln\left(E_1 z^+\right) / \left(k_1 C\right) & 5 \le z^+ \le 30 \\ R(z) = \ln\left(E_2 z^+\right) / \left(k_2 C\right) & 30 \le z^+ \end{array}\right\} \tag{3.13}$$

avec

$$E_1 = 0.543,\quad E_2 = 9.025,\quad k_1 = 0.2 \quad \text{et} \quad k_2 = 0.4 \tag{3.14}$$

où $z^+ = z\,u^* / \nu$ où $u^* = \sqrt{\tau_w / \rho}$ est une échelle de vitesse qui s'appelle la vitesse de frottement, $(0 \le z \le h)$, h étant la hauteur de l'élément.

C est une constante qui d'adimensionnement qui est définie pour chaque couche et donc dépend de h^+, de telle façon que $R(z) = 1$ pour $z = h$.

$$\left.\begin{array}{ll} C = h^+ & h^+ \leq 5 \\ C = \ln\left(E_1 h^+\right)/k_1 & 5 \leq h^+ \leq 30 \\ C = \ln\left(E_2 h^+\right)/k_2 & 30 \leq h^+ \end{array}\right\} \qquad (3.15)$$

Pour un élément de référence, on introduit la coordonnée adimensionnée ζ telle que:

$$z = \frac{h(1+\zeta)}{2}, \quad -1 \leq \zeta \leq 1 \qquad (3.16)$$

La fonction d'interpolation $R(z)$ devient alors :

$$\left.\begin{array}{ll} R(\zeta) = (1+\zeta)\alpha / C & -1 \leq \zeta \leq \zeta_1 \\ R(\zeta) = \ln\left(\beta_1 (1+\zeta)\right)/(k_1 C) & \zeta_1 \leq \zeta \leq \zeta_2 \\ R(\zeta) = \ln(\beta_2 (1+\zeta))/(k_2 C) & \zeta_2 \leq \zeta \leq 1 \end{array}\right\} \qquad (3.17)$$

avec

$$\alpha = u^* h/2\nu, \quad \beta_i = E_i \; u^* h/2\nu, \quad \zeta_1 = 10/h^+ - 1 \text{ et } \zeta_2 = 10/h^+ + 1 \qquad 3.18)$$

La figure 16 montre l'élément prismatique avec l'interpolation Log-Lin et les trois régions de la paroi :

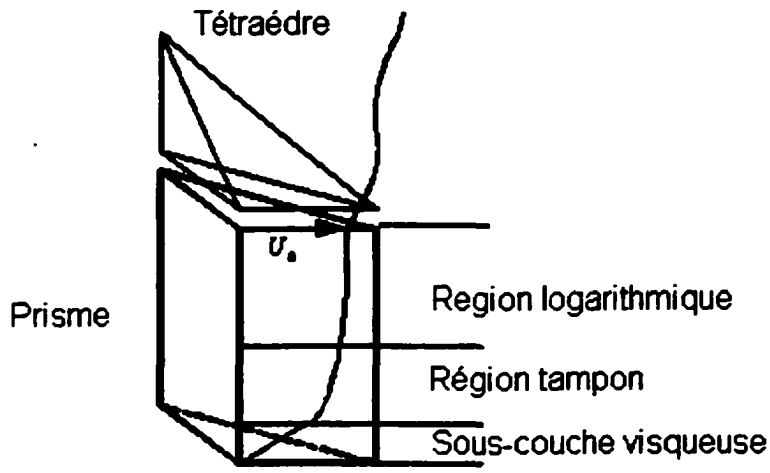


Figure 16- Élément prismatique avec l'interpolation Log-Lin

Maintenant pour l'élément Log-Lin, les nœuds inférieurs vont avoir les mêmes fonctions d'interpolation linéaires données par (3.10). Aux nœuds supérieurs vont être associées des fonctions qui sont le produit d'une interpolation linéaire dans le plan et de la fonction $R(z)$ dans la direction verticale. Les foncions d'interpolation Log-Lin seront donc:

$$\langle N \rangle_{LL} = \langle N_1 \quad N_2 \quad N_3 \quad N_4 \quad N_5 \quad N_6 \rangle$$

$$\langle N \rangle_{LL} = \langle \lambda a \;\; \xi a \;\; \eta a \;\; \lambda R(\zeta) \;\; \xi R(\zeta) \;\; \eta R\zeta) \rangle \qquad (3.19)$$

3.6.1.3 L'interpolation Log-Lin-Spectrale

En présence d'un gradient de pression, le comportement logarithmique dans la région de la paroi n'est plus valable et nécessite une correction qui tiendrait compte du gradient de pression. Le comportement Log-Lin est alors combiné avec un développement spectral de la forme suivante:

$$\tilde{u}(\zeta) = U_0\, R(\zeta) + \sum_{i=0}^{n-1} \hat{u}_i\ \Phi_i(\zeta) \tag{3.20}$$

Avec $R(-1)=0$, $R(1)=1$ et $\tilde{u}(-1)=0$.

Φ_i étant les polynômes de Lagrange calculés avec les points de Gauss-Legendre-Lobatto.

Pour un prisme enrichie selon la direction verticale, les fonctions d'interpolation spectrales sont:

$$\langle N\rangle_S = \langle N_1\ \ N_2\ \ N_3, \ldots,\ N_{3P-2}\ \ N_{3P-1}\ \ N_{3P}\rangle$$
$$\langle N\rangle_S = \langle \lambda h_1(\zeta)\,\xi h_1(\zeta)\,\eta h_1(\zeta), \ldots, \lambda h_j(\zeta)\,\xi h_j(\zeta)\,\eta h_j(\zeta), \ldots, \lambda h_p(\zeta)\,\xi h_p(\zeta)\,\eta h_p(\zeta)\rangle \tag{3.21}$$

où $h_j(\zeta)$ étant le polynôme d'interpolation de Lagrange associé au point d'interpolation j dans la direction verticale (les points d'interpolation sont les points de collocation de Gauss-Legendre-Lobatto).

Dans le cas d'une interpolation Log-Lin combinée avec la correction spectrale, les foncions d'interpolation dans la direction verticale s'écrivent :

$$\langle N\rangle_{LL-S} = \langle N_1\ \ N_2\ \ N_3, \ldots,\ N_{3P-2}\ \ N_{3P-1}\ \ N_{3P},\ N_{3P+1}\ \ N_{3P+2}\ \ N_{3P+3}\rangle$$
$$\langle N\rangle_{LL-S} = \langle \lambda h_1(\zeta)\,\xi h_1(\zeta)\,\eta h_1(\zeta), \ldots, \lambda h_p(\zeta)\,\xi\xi_p(\zeta)\,\eta h_p(\zeta),\ \lambda R(\zeta)\,\xi R(\zeta)\,\eta R(\zeta)\rangle \tag{3.22}$$

Les variables indépendantes ρ, U, E et ν_t et leurs fonctions de pondération correspondantes sont interpolées comme suit:

$$\begin{aligned}
\rho &= \langle N \rangle_{\mathrm{L}} \cdot \{\rho_n\} & q &= \langle N \rangle_{\mathrm{L}} \cdot \{q_n\} \\
U_x &= \langle N \rangle_{\mathrm{LL\text{-}S}} \cdot \{U_{xn}\} & \omega_x &= \langle N \rangle_{\mathrm{S}} \cdot \{\omega_{xn}\} \\
U_y &= \langle N \rangle_{\mathrm{L}} \cdot \{U_{yn}\} & \omega_y &= \langle N \rangle_{\mathrm{L}} \cdot \{\omega_{yn}\} \\
U_z &= \langle N \rangle_{\mathrm{L}} \cdot \{U_{zn}\} & \omega_z &= \langle N \rangle_{\mathrm{L}} \cdot \{\omega_{zn}\} \\
E &= \langle N \rangle_{\mathrm{L}} \cdot \{E_n\} & \theta &= \langle N \rangle_{\mathrm{L}} \cdot \{\theta_n\} \\
\nu_t &= \langle N \rangle_{\mathrm{L}} \cdot \{\nu_{tn}\} & \varphi &= \langle N \rangle_{\mathrm{L}} \cdot \{\varphi_n\}
\end{aligned}$$

Remarque: Il faut noter qu'il s'agit d'une méthode de Petrov-Galerkin pour l'équation de U_x puisque l'espace de ω_x est différent de celui de U_x.

La figure 17 ilustre l'interpolation selon la direction verticale:

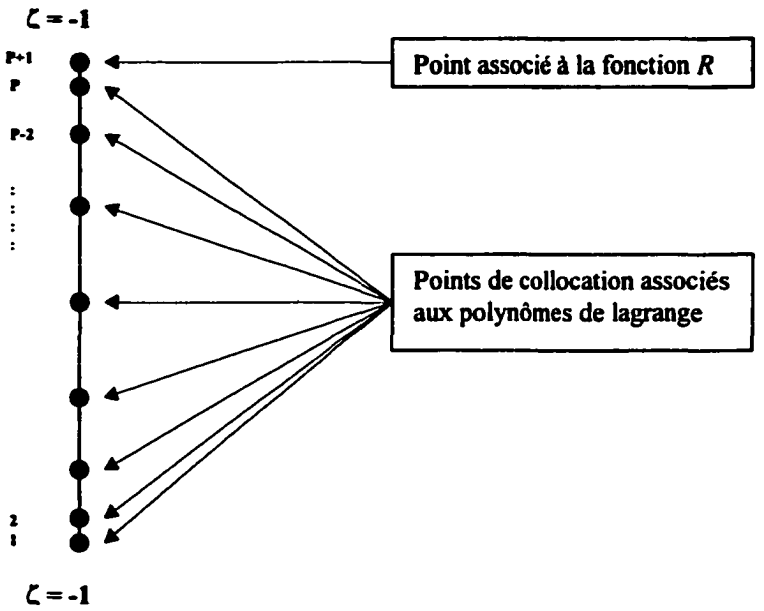


Figure 17- Interpolation dans la direction verticale

Les variables auxiliaires $\mathbf{u}, T$ et p sont exprimées en fonction des variables indépendantes ensuite sont interpolées comme suit :

$$
\begin{aligned}
p &= \langle N \rangle_{\mathrm{L}} \cdot \{p_n\} \\
u_x &= \langle N \rangle_{\mathrm{LL-S}} \cdot \{u_{x_n}\} \\
u_y &= \langle N \rangle_{\mathrm{L}} \cdot \{u_{y_n}\} \\
u_z &= \langle N \rangle_{\mathrm{L}} \cdot \{u_{z_n}\} \\
T &= \langle N \rangle_{\mathrm{L}} \cdot \{T_n\}
\end{aligned}
$$

Toutes les dérivées spatiales dans le système de coordonnées réelles **X** sont transformées en des dérivées dans le système de référence ξ à travers la matrice Jacobienne $[J]$ telle que, pour les dérivées premières:

$$\left\{\frac{\partial V}{\partial \mathbf{X}}\right\} = [J]^{-1} \left\{\frac{\partial V}{\partial \xi}\right\} \tag{3.23}$$

avec

$$[J] = \left\{\frac{\partial \mathbf{X}}{\partial \xi}\right\} \tag{3.24}$$

Finalement, toute intégrale élémentaire sur l'élément réel sera évaluée sur l'élément de référence comme suit:

$$\int_{\Omega^e} f(\mathbf{X})\, d\mathbf{X} = \int_{\Omega^e} f(\xi) \det(J)\, d\xi = \sum_{i=1}^{IPG} w_i f(\xi_i) \tag{3.25}$$

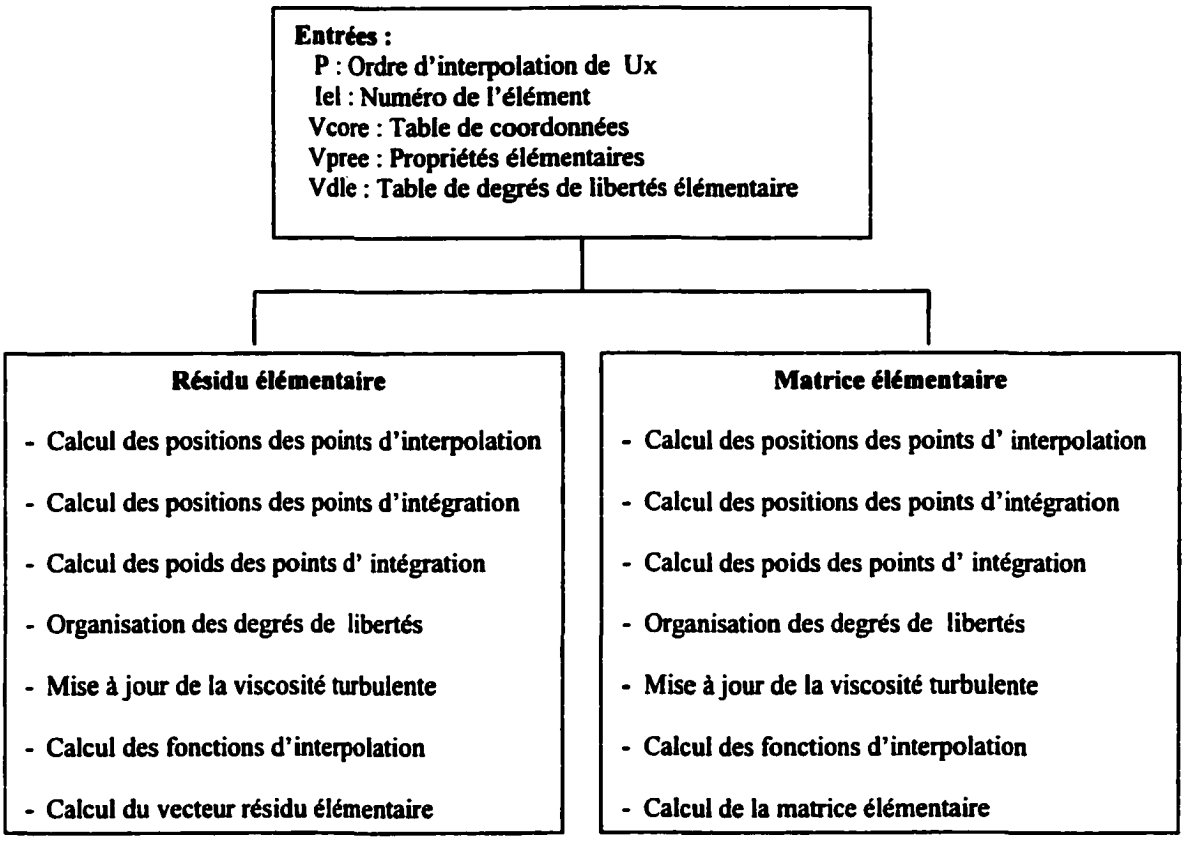
IPG est le nombre de points total d'intégration.
w_i est le poids du point d'intégration

L'intégration numérique est effectuée en utilisant trois points de Gauss dans le plan (ξ, η). Dans la direction verticale ζ, On utilise la quadrature de Gauss-Lobatto avec P+1 points d'intégration telle que définie dans le chapitre 1.

3.6.2 Implémentation

3.6.2.1 Structure globale

La structure globale du programme est présentée sur l'organigramme suivant:



Nous présentons en détail l'organisation de degrés de libertés et le calcul des fonctions d'interpolation.

3.6.2.2 Organisation des degrés de libertés

Nous présentons la structure de données utilisée dans la routine qui calcule le résidu.
Si P est l'ordre de l'interpolation utilisée pour la composante de U selon la direction x.
Les degrés de libertés seront stockés dans le vecteur de degrés de libertés, de la façon suivante:

$$\overbrace{\rho^{(1)} \begin{Bmatrix} U_{1,1}^{(1)} \\ U_{1,2}^{(1)} \\ \vdots \\ U_{1,P}^{(1)} \end{Bmatrix} U_2^{(1)} \; U_3^{(1)} \; E^{(1)} \nu_t{}^{(1)}}^{\text{Noeud 1}} \overbrace{\rho^{(2)} \begin{Bmatrix} U_{1,1}^{(2)} \\ U_{1,2}^{(2)} \\ \vdots \\ U_{1,P}^{(2)} \end{Bmatrix} U_2^{(2)} \; U_3^{(2)} \; E^{(2)} \nu_t{}^{(2)}}^{\text{Noeud 2}} \overbrace{\rho^{(3)} \begin{Bmatrix} U_{1,1}^{(1)} \\ U_{1,2}^{(3)} \\ \vdots \\ U_{1,P}^{(3)} \end{Bmatrix} U_2^{(3)} \; U_3^{(3)} \; E^{(3)} \nu_t{}^{(3}}^{\text{Noeud 3}}$$

$$\overbrace{\rho^{(4)} \; U_2^{(4)} \; U_2^{(4)} \; U_3^{(4)} \; E^{(4)} \nu_t{}^{(4)}}^{\text{Noeud 4}} \quad \overbrace{\rho^{(5)} U_2^{(5)} U_2^{(5)} \; U_3^{(5)} \; E^{(5)} \nu_t{}^{(5)}}^{\text{Noeud 5}} \quad \overbrace{\rho^{(6)} \; U_2^{(6)} U_2^{(6)} \; U_3^{(6)} \; E^{(6)} \nu_t{}^{(6)}}^{\text{Noeud 6}}$$

La partie suivante du code permet de réordonner les degrés de libertés en donnant pour chacun des degrés de liberté, les indices correspondants dans la table précédente.

```
Kro = (/1,1+(5+P),1+2*(5+P),1+3*(5+P),7+3*(5+P),13+3*(5+P)/)
 If (p.eq.1) then
   ku1(1:6)=(/2,2+(5+P),2+2*(5+P),2+3*(5+P),8+3*(5+P),14+3*(5+P)/)
Endif
 If (p.ne.1) then
   ku1(1:3)=(/2,2+(5+P),2+2*(5+P)/)
   pp=0
   Do j = 2, P
      ku1(4+pp) =  1+j
      ku1(5+pp) =  1+j+(5+P)
      ku1(6+pp) =  1+j+2*(5+P)
   pp= pp+3
   Enddo

Ku1(ppp-2:ppp)= (/2+3*(5+P),8+3*(5+P),14+3*(5+P)/)    ! ppp = 3*(P+1)
Endif
Ku2  = (/2+P,2+P+(5+P),2+P+2*(5+P),  8+P+2*(5+P),14+P+2*(5+P),20+P+2*(5+P)/)
Ku3  = (/3+P,3+P+(5+P),3+P+2*(5+P),  9+P+2*(5+P),15+P+2*(5+P),21+P+2*(5+P)/)
Kenr = (/4+P,4+P+(5+P),4+P+2*(5+P),10+P+2*(5+P),16+P+2*(5+P),22+P+2*(5+P)/)
Knut = (/5+P,5+P+(5+P),5+P+2*(5+P),11+P+2*(5+P),17+P+2*(5+P),23+P+2*(5+P)/)
If (P.eq.1) then
    Do i=1,inelc
       Do j=1,inelc
         kpok(j+(i-1)*(P+5))= i+(j-1)*6
       Enddo
    Enddo
 Else


Do i=1, inelc
   If (i.le.3) then
     Do j=1, P+5
        if (j.lt.3) kpok(j+(i-1)*(P+5))= i+(j-1)*6
        if ((j.ge.3).and.(j.lt.P+2)) kpok(j+(i-1)*(P+5))= i+(j-3)*3+9
        if (j.ge.P+2) kpok(j+(i-1)*(P+5))= i+3+(2+P)*3+(j-P-2)*6
     Enddo
   Else
     Do j=1, 6
        if (j.eq.1) kpok(j+(i-4)*6+3*(P+5))= i
        if (j.eq.2) kpok(j+(i-4)*6+3*(P+5))= i+3*P+3
        if (j.eq.3) kpok(j+(i-4)*6+3*(P+5))= i+3*P+9
        if (j.eq.4) kpok(j+(i-4)*6+3*(P+5))= i+3*P+15
        if (j.eq.5) kpok(j+(i-4)*6+3*(P+5))= i+3*P+21
        if (j.eq.6) kpok(j+(i-4)*6+3*(P+5))= i+3*P+27
      Enddo
   Endif
Enddo
Endif
return
Endif
```

La nouvelle structure des degrés de libertés est la suivante :

$$\overbrace{\rho^{(1)}\ \rho^{(2)}\ \rho^{(3)}\ \rho^{(4)}\ \rho^{(5)}\ \rho^{(6)}}^{\text{Densité}}\ \overbrace{\overbrace{U_{1,1}^{(1)}\ U_{1,1}^{(2)}\ U_{1,1}^{(3)}}^{\text{Première couche}}\ \ldots\overbrace{U_{1,j}^{(1)}\ U_{1,j}^{(2)}\ U_{1,j}^{(3)}}^{\text{Couche j}}\ \ldots\overbrace{U_{1,P}^{(1)}\ U_{1,P}^{(2)}\ U_{1,P}^{(3)}}^{\text{Couche P}}}^{\text{Quantité de mvt. en x enrichie associee aux 3 premiers noeuds}}\ \overbrace{U_{1,2}^{(4)}\ U_{1,2}^{(5)}\ U_{1,2}^{(6)}}^{\text{Les ddl associés aux noeuds sup.}}$$

$$\overbrace{U_2^{(1)}\ U_2^{(2)}\ U_2^{(3)}\ U_2^{(4)}\ U_2^{(5)}\ U_2^{(6)}}^{\text{Quantité de mvt. en y}}\ \overbrace{U_2^{(1)}\ U_2^{(2)}\ U_2^{(3)}\ U_2^{(4)}\ U_2^{(5)}\ U_2^{(6)}}^{\text{Quantité de mvt. en y}}$$

$$\overbrace{E^{(1)}\ E^{(2)}\ E^{(3)}E^{(4)}\ E^{(5)}\ E^{(6)}}^{\text{Enérgie}}\ \overbrace{\nu_t^{(1)}\ \nu_t^{(2)}\ \nu_t^{(3)}\ \nu^{(4)}\ \nu_t^{(5)}\ \nu_t^{(6)}}^{\text{Viscosité turbulente}}$$

Maintenant qu'on a les tables des indices, on peut extraire les degrés de libertés. La densité ρ, la composante en y de la quantité de mouvement U_2, la composante en z de la quantité de mouvement U_3, l'Énergie E et la viscosité turbulente ν_t sont extraite de la même façon pour tous les nœuds. La quantité de mouvement en x est enrichie seulement pour les trois premiers nœuds (Nœuds inférieurs). Donc pour chacun de ces trois nœuds, on a P composantes de la quantité de mouvement U_1 , et aussi de la vitesse u_1 et de la température T puisque ces derniers sont calculés à partir de U_1.

On présente ici la partie du code qui fait cette extraction des degrés de libertés:

```
c                    Extraction des degrés de liberté
c*****************************************************************
  Do i=1,inelc     ! Boucle sur les nœuds de l'élement  (inelc = 6)
c*****************************************************************
ii   = kro(i)
jj   = ku2(i)
kk   = ku3(i)
ll   = kenr(i)
mm = knut(i)
c
c------ Density
c
vden(i)  = vdle(ii)
vdent(i) = vdlev(ii)
c
c------ Momentum in x (Enrichie)

      if (i.le.3) then
      nn = ku1(i)
      qm1(i)  = vdle(nn)
      qm1t(i) = vdlev(nn)
      if (p.ne.1) then
      pp=0
      do j=2, P
        qm1(i+3+pp)  = vdle(ku1(i+3+pp))
        qm1t(i+3+pp) = vdlev(ku1(i+3+pp))
      pp=pp+3
      enddo
      endif
      endif
      if (i.gt.3) then
      nn=ku1(i-3+3*P)
      qm1(i-3+3*P)=vdle(nn)
      qm1t(i-3+3*P) = vdlev(nn)
      endif

c------ Momentum in y and z
c
qm2(i)  = vdle(jj)
qm2t(i) = vdlev(jj)
 c
qm3(i)  = vdle(kk)
qm3t(i) = vdlev(kk)
c
c------ Energy
c
venr(i) = vdle(ll)
venrt(i)= vdlev(ll)
```

```
c------ Kinematic turbulent viscosity
c
visco(i) = vdle(mm)
viscot(i)= vdlev(mm)
c
c------ Temperature
c
if (i.le.3) then
  gu = qm1(i)*qm1(i) + qm2(i)*qm2(i) + qm3(i)*qm3(i)
  if (p.ne.1) then
  pp=0
   do j=2, P
    gu = gu + qm1(i+3+pp)*qm1(i+3+pp)
   pp=pp+3
   enddo
 endif
endif
if (i.gt.3) then
gu = qm1(i-3+3*P)*qm1(i-3+3*P)+qm2(i)*qm2(i)+qm3(i)*qm3(i)
endif
vtem(i) = (venr(i) - gu/(2.0*vden(i)))/vden(i)
c
c------ Pressure
c
vpres(i)= gama1*vden(i)*vtem(i)
c
c------ Velocity in x  (Enrichie)
c
if (i.le.3) then
  veloc1(i) = qm1(i)/vden(i)
  if (p.ne.1) then
    pp=0
    do j=2, P
      veloc1(i+3+pp) = qm1(i+3+pp)/vden(i)
     pp=pp+3
    enddo
  endif
endif
 if (i.gt.3) veloc1(i-3+3*P) = qm1(i-3+3*P)/vden(i)
c------ Velocity in y and z direction
c
veloc2(i) = qm2(i)/vden(i)
veloc3(i) = qm3(i)/vden(i)
c
ENDDO      ! Fin de la boucle sur les nœuds de l'élément
```

3.6.2.3 Calcul des fonctions d'interpolation

On présente ici la partie de la routine Shape_prisme qui calcule les fonctions d'interpolation et leurs dérivées aux points d'intégration :

```
c=======================================================================
c     fonctions d interpolation du prisme
c=======================================================================
subroutine shape_prisme(uet,dist,vnu)
use global_data,only : P,npgz,vniL,vniS,vniLL,vzint,vzeta,wpg,
   &                   zero,un,deux,trois,ipg
implicit none

integer i,j,k,iii,jjj,kkk
real*8 uet(3),dist,vnu
real*8 vksi(3),veta(3)
real*8 pln,pdn,hj,hpi,Rz,Rpz
real*8 xsi,eta,zeta,zint
real*8 lamda, a, b
c=======================================================================
c
vksi(1) = 1.0d0/6.0d0
vksi(2) = 2.0d0/3.0d0
vksi(3) = 1.0d0/6.0d0
c------
veta(1) = 1.0d0/6.0d0
veta(2) = 1.0d0/6.0d0
veta(3) = 2.0d0/3.0d0

c=======================================================================
c INTERPOLATION LINEAIRE
c=======================================================================
c
c------ boucle sur les points d'intégration

do i= 1, npgz   ! npgz : le nombre de points d'intégration dans la direction verticale

zeta = vzeta(i)

do j= 1, 3

      xsi = vksi(j)
      eta = veta(j)
      lamda = un - xsi - eta
      a = (un - zeta )/ deux
      b = (un + zeta )/ deux
```

```
c------ fonctions d interpolation
c
      vniL(72*(i-1)+24*(j-1)+1) =  lamda * a
      vniL(72*(i-1)+24*(j-1)+2) =     xsi * a
      vniL(72*(i-1)+24*(j-1)+3) =     eta * a
      vniL(72*(i-1)+24*(j-1)+4) =  lamda * b
      vniL(72*(i-1)+24*(j-1)+5) =     xsi * b
      vniL(72*(i-1)+24*(j-1)+6) =     eta * b
c
c------ derivees par rapport a ksi
c
      vniL(72*(i-1)+24*(j-1)+7)  =   - a
      vniL(72*(i-1)+24*(j-1)+8)  =     a
      vniL(72*(i-1)+24*(j-1)+9)  =  zero
      vniL(72*(i-1)+24*(j-1)+10) =   - b
      vniL(72*(i-1)+24*(j-1)+11) =     b
      vniL(72*(i-1)+24*(j-1)+12) =  zero
c
c------ derivees par rapport a eta
c
      vniL(72*(i-1)+24*(j-1)+13) =   - a
      vniL(72*(i-1)+24*(j-1)+14) = zero
      vniL(72*(i-1)+24*(j-1)+15) =     a
      vniL(72*(i-1)+24*(j-1)+16) =   - b
      vniL(72*(i-1)+24*(j-1)+17) = zero
      vniL(72*(i-1)+24*(j-1)+18) =     b
c
c------ derivees par rapport a zeta
c
      vniL(72*(i-1)+24*(j-1)+19) =  - lamda/deux
      vniL(72*(i-1)+24*(j-1)+20) =     - xsi/deux
      vniL(72*(i-1)+24*(j-1)+21) =     - eta/deux
      vniL(72*(i-1)+24*(j-1)+22) =    lamda/deux
      vniL(72*(i-1)+24*(j-1)+23) =       xsi/deux
      vniL(72*(i-1)+24*(j-1)+24) =       eta/deux

enddo
enddo

c=============================================================
c  INTERPOLATION SPECTRALE
c=============================================================

jjj=0
DO j = 1, npgz
Zeta = vzeta(j)

iii=0
DO i=1,3
xsi = vksi(i)
eta = veta(i)
```

```
c----- Boucle sur k pour balayer les points d'interpolation selon la verticale
c
kkk=0
DO k = 0, P

zint = vzint(k)

c------ Fonctions d interpolation  ( On calcule le produit Ni(x,y)* hj(z) )
c
vniS(1+kkk+iii+jjj) =  (1-xsi-eta)*hj(zeta,zint,P)
vniS(2+kkk+iii+jjj) =           xsi*hj(zeta,zint,P)
vniS(3+kkk+iii+jjj) =           eta*hj(zeta,zint,P)

c------ Dérivées par rapport a ksi
c
vniS(1+kkk+3*(1+P)+iii+jjj)= - hj(zeta,zint,P)
vniS(2+kkk+3*(1+P)+iii+jjj)=   hj(zeta,zint,P)
vniS(3+kkk+3*(1+P)+iii+jjj)=   zero

c------ Dérivées par rapport a eta
c
vniS(1+kkk+6*(1+P)+iii+jjj)= - hj(zeta,zint,P)
vniS(2+kkk+6*(1+P)+iii+jjj)=   zero
vniS(3+kkk+6*(1+P)+iii+jjj)=   hj(zeta,zint,P)

c------ Dérivées par rapport a zeta
c
c------ Ici hp(zeta,zint,P ) represente la derivee de hj(Lagrange) au point zeta
c
vniS(1+kkk+9*(1+P)+iii+jjj)= (1-xsi-eta)*hpi(zeta,k,P,vzint)
vniS(2+kkk+9*(1+P)+iii+jjj)=       xsi*hpi(zeta,k,P,vzint)
vniS(3+kkk+9*(1+P)+iii+jjj)=       eta*hpi(zeta,k,P,vzint)

c------ Fin de la boucle sur k
c
kkk= kkk+ 3
Enddo
c------ Fin de la boucle sur i
c
iii= iii+ 12*(P+1)  ! 24 dans le cas lineair
Enddo
c------ Fin de la boucle sur j
c
jjj= jjj+ 36*(P+1)  ! 72 dans le cas lineair
Enddo
```

```
c=======================================================================
c INTERPOLATION LOG-LIN
c=======================================================================

jjj=0
DO j = 1, npgz

zeta = vzeta(j)

iii=0
DO i=1,3
xsi = vksi(i)
eta = veta(i)
c----- Boucle sur k pour balayer les points d'interpolation selon la verticale
c
kkk=0
DO k = 0, P-1
zint = vzint(k)

c------ Fonctions d interpolation  ( On calcule le produit Ni(x,y)* hj(z) )
c
vniLL(1+kkk+iii+jjj) =  (1-xsi-eta)*hj(zeta,zint,P)
vniLL(2+kkk+iii+jjj) =        xsi*hj(zeta,zint,P)
vniLL(3+kkk+iii+jjj) =        eta*hj(zeta,zint,P)

c------ Dérivées par rapport a ksi
c
vniLL(1+kkk+3*(1+P)+iii+jjj)= - hj(zeta,zint,P)
vniLL(2+kkk+3*(1+P)+iii+jjj)=   hj(zeta,zint,P)
vniLL(3+kkk+3*(1+P)+iii+jjj)=   zero

c------ Dérivées par rapport a eta
c
vniLL(1+kkk+6*(1+P)+iii+jjj)= - hj(zeta,zint,P)
vniLL(2+kkk+6*(1+P)+iii+jjj)=   zero
vniLL(3+kkk+6*(1+P)+iii+jjj)=   hj(zeta,zint,P)

c------ Dérivées par rapport a zeta
c
c------ Ici hpi(zeta,zint,P ) represente la derivee de hj(Lagrange) au point zeta
c
vniLL(1+kkk+9*(1+P)+iii+jjj)= (1-xsi-eta)*hpi(zeta,k,P,vzint)
vniLL(2+kkk+9*(1+P)+iii+jjj)=        xsi*hpi(zeta,k,P,vzint)
vniLL(3+kkk+9*(1+P)+iii+jjj)=        eta*hpi(zeta,k,P,vzint)

c------ Fin de la boucle sur k
c
kkk= kkk+ 3
Enddo
```

```
c------ Fin de la boucle sur i
c
iii= iii+ 12*(P+1)  ! 24 dans le cas lineair
Enddo
c------ Fin de la boucle sur j
c
jjj= jjj+ 36*(P+1)  ! 72 dans le cas lineair
Enddo


c=============================================================
c------ CALCUL DE LA FONCTION D INTERPOLATION POUR LA DERNIERE
c      COUCHE
c=============================================================
c

iii=0
jjj=0
DO j= 1, npgz
c
c------ j : COUCHE PAR COUCHE d INTEGRATION
c
zeta=vzeta(j)
DO i=1,3
c
c------ i : NOEUD PAR NOEUD
c
xsi=vksi(i)
eta=veta(i)
c
c------ Fonctions d interpolation LOGARITHMIQUES
c
vniLL(1+iii+3*P) =  (1-xsi-eta)*Rz(zeta,uet(1),dist,vnu)
vniLL(2+iii+3*P) =           xsi*Rz(zeta,uet(2),dist,vnu)
vniLL(3+iii+3*P) =           eta*Rz(zeta,uet(3),dist,vnu)
c
c------ Dérivées par rapport a ksi LOGARITHMIQUES
c
vniLL(1+3*(1+P)+iii+3*P)= - Rz(zeta,uet(1),dist,vnu)
vniLL(2+3*(1+P)+iii+3*P)=   Rz(zeta,uet(2),dist,vnu)
vniLL(3+3*(1+P)+iii+3*P)=   zero
c
c------ Dérivées par rapport a eta LOGARITHMIQUES
c
vniLL(1+6*(1+P)+iii+3*P)= - Rz(zeta,uet(1),dist,vnu)
vniLL(2+6*(1+P)+iii+3*P)=   zero
vniLL(3+6*(1+P)+iii+3*P)=   Rz(zeta,uet(3),dist,vnu)
```

```
c------ Dérivées par rapport a zeta LOGARITHMIQUES
c
c------ Ici hp(zeta,zint,P ) represente la derivee de hj(Lagrange) au point zeta
c
vniLL(1+9*(1+P)+iii+3*P)= (1-xsi-eta)*Rpz(zeta,uet(1),dist,vnu)
vniLL(2+9*(1+P)+iii+3*P)=          xsi*Rpz(zeta,uet(2),dist,vnu)
vniLL(3+9*(1+P)+iii+3*P)=          eta*Rpz(zeta,uet(3),dist,vnu)
c
c------ Fin de la boucle sur i
c
iii= iii+ 12*(P+1)  ! 24 dans le cas lineair
Enddo
c------ Fin de la boucle sur j
c
jjj= jjj+ 36*(P+1)  ! 72 dans le cas lineair
Enddo
c
c=============================================================
End     ! FIN DU PROGRAMME
c=============================================================
```

3.6.3 Discrétisation dans la région externe

Dans la région externe à la couche limite, on utilise des élément tétraédriques isoparamétriques. Les variables indépendantes sont approximées à l'aide de la méthode des éléments finis P^1. Les foncions d'interpolation linéaires pour un élément tétraédrique s'écrivent:

$$\langle N \rangle = \langle N_1 \quad N_2 \quad N_3 \quad N_4 \rangle$$
$$\langle N \rangle = \langle 1-\xi-\eta-\zeta \quad \xi \quad \eta \quad \zeta \rangle \tag{3.26}$$

Les variables indépendantes ρ, $\mathbf{U}$, E et leurs fonctions de pondération correspondantes sont interpolées comme suit :

$$\rho = \langle N \rangle . \{\rho_n\} \quad q = \langle N \rangle . \{q_n\}$$
$$\mathbf{U} = \langle N \rangle . \{\mathbf{U}_n\} \quad \omega = \langle N \rangle . \{\omega_n\}$$
$$E = \langle N \rangle . \{E_n\} \quad \theta = \langle N \rangle . \{\theta_n\}$$

Lcintégration numérique est effectuée en utilisant quatre points de Gauss.

Une fois l'intégration numérique est effectuée, les matrices et les vecteurs élémentaires seront assemblées pour former le système matriciel global à résoudre:

$$\mathbf{W} = \langle \Phi \rangle \left\{ [\mathbf{M}] \left\{ \frac{\partial \mathbf{V}}{\partial t} \right\} + [\mathbf{k}(\mathbf{V})] \{\mathbf{V}\} - \{\mathfrak{I}\} \right\} = 0 \tag{3.27}$$

Pour résoudre ce système d'équations non-linéaires, on utilise l'algorithme **GMRES**. Vous trouverez à l'annexe 2 une présentation de cet algorithme.

3.7 Discrétisation temporelle

Pour la discrétisation temporelle, on utilise le schéma du second ordre suivant :

$$\frac{\partial \mathbf{V}}{\partial t} = \frac{3\mathbf{V}^n - 4\mathbf{V}^{n-1} + 3\mathbf{V}^{n-2}}{2\Delta t} \tag{3.28}$$

où $\mathbf{V}^n$ est la valeur de V au temps $t = n\,\Delta t$ $(n = 0,1,2,...)$

Au départ, lorsque $n = 0$, on pose $\mathbf{V}^{-1} = \mathbf{V}^0$

3.8 Méthodes de stabilisation

3.8.1 Introduction

Les écoulements à grands nombres de Mach et de Reynolds offrent un caractère fortement non-linéaire et une résolution par la méthode des éléments finis classique devient fort délicate et dépend de plusieurs conditions. Dans les cas de problèmes de convection dominante ou de convection pure, l'approximation par la méthode des éléments finis classique correspond à des approximations type différences finies centrées conduisant à des solutions instables. Pour remédier à ces problèmes, on doit absolument faire appel à certains techniques numériques supplémentaires, qui sont les méthodes de stabilisation. Ces méthodes de stabilisation résultent ainsi en un schéma d'approximation stable, qui permet d'obtenir des résultats numériques favorables.

Les méthodes de stabilisation utilisées sont deux types :

- Viscosité artificielle : celle-ci consiste à ajuster le modèle mathématique en ajoutant une viscosité artificielle à la viscosité du fluide sans pour autant changer de schéma ou de méthode de résolution.
- Streamline Upwinding Petrov-Galerkin (**SUPG**) [2, 14] : cette méthode de décentrage amont est basée sur une formulation de Petrov-Galerkin qui permet d'engendrer une diffusion artificielle dans le sens des lignes de courant.

- Edge Based Stabilized (**EBS**), voir Soulaïmani [6].

Rappelons que le système d'équations global sur lequel on veut appliquer les méthodes de stabilisation ci-dessus, peut s'écrire sous la forme vectorielle suivante :

$$\mathbf{V}_{,t} + \mathbf{F}_{i,i}^{conv}(\mathbf{V}) = \mathbf{F}_{i,i}^{diff}(\mathbf{V}) + \Im \qquad (3.29a)$$

ou sous la forme quasi-linéaire:

$$\mathbf{V}_{,t} + \mathbf{A}_i \mathbf{V}_{,i} = \left(\mathbf{K}_{ij} \mathbf{V}_{,j} \right)_{,i} + \mathfrak{I} \tag{3.29b}$$

où $\mathbf{A}_i$ sont les matrices jacobiennes de transformation du vecteur flux de convection telles que:

$$\mathbf{A}_i = \mathbf{F}_{i,\mathbf{V}}^{conv} = \frac{\partial \mathbf{F}_i^{conv}}{\partial \mathbf{V}}$$

et $\mathbf{K}_{ij}$ sont les matrices de diffusion définies telles que :

$$\mathbf{K}_{ij} \mathbf{V}_{,j} = \mathbf{F}_i^{diff}$$

Le système global sous la forme (3.29b) est connu dans la littérature sous le nom d'équation de convection-diffusion). Dans de paragraphe, on la méthode de Petrov-Galerkin utilisée pour la stabilisation de la solution de (3.29). Deux variantes de la méthode de Petrov-Galerkin seront introduites pour le cas de l'équation de convection-diffusion unidimensionnelle stationnaire: la méthode **SUPG** et la méthode **GLS**. Ensuite, on élargira l'étude de ces méthodes de stabilisation aux équations de Navier-Stokes .

3.8.2 Méthodes de Galerkin et Petrov-Galerkin

Soit le problème simple de convection-diffusion unidimensionnel suivant:

Trouver φ dans l'intervalle $]0, L[$ satisfaisant l'équation de convection-diffusion :

$$u\,\varphi_{,x} - \kappa\,\varphi_{,xx} = 0 \tag{3.30a}$$

et les conditions aux limites de Dirichlet :

$$\left.\begin{aligned} \varphi(0) &= \varphi_0 \\ \varphi(L) &= \varphi_L \end{aligned}\right\} \tag{3.30b}$$

où

u : la vitesse de l'écoulement supposée positive, constante et connue,

κ : la diffusion supposée positive, constante et connue,

φ_0 et φ_L sont des constantes connues.

La formulation variationnelle faible de type Galerkin, du problème de convection-diffusion (3.30), consiste à trouver φ tel que, pour toute fonction de pondération ω, on a :

$$\int_0^L \left(\omega\,u\,\varphi_{,x} + \kappa\,\omega_{,x}\,\varphi_{,x}\right) dx = 0 \tag{3.31}$$

Chaque solution φ doit satisfaire les conditions de Dirichlet (3.30b) et chaque fonction de pondération ω doit satisfaire sa condition homogène :

$$\omega(0) = \omega(L) = 0$$

Sur des éléments finis linéaires P^1 , la forme variationnelle (3.31) peut être approchée par :

$$\int_0^L \left(u\,\varphi_{h,x}\omega_h + \kappa\,\varphi_{h,x}\,\omega_{h,x} \right) dx = 0 \qquad (3.32)$$

En subdivisant l'intervalle $[0, L]$ en des intervalles de longueur égale h, on peut vérifier que l'approximation (3.32) s'écrit:

$$\frac{u}{2h}\left(\varphi_{i+1} - \varphi_{i-1}\right) - \frac{\kappa}{h^2}\left(\varphi_{i+1} - 2\,\varphi_i - \varphi_{i-1}\right) = 0 \qquad (3.33)$$

Ainsi la formulation variationnelle de Galerkin correspond à une approximation de type différences finis centrées, qui peut être instable si le nombre de Peclet est supérieur à 1 (convection dominante) [10] . De la diffusion artificielle est alors nécessaire pour stabiliser la solution. Cette diffusion peut être engendrée via un décentrage, à l'aide de la méthode de Petrov-Galerkin, en prenant des fonctions de pondération de la forme:

$$\tilde{\omega} = \omega + p$$

où ω est la fonction continue et p est la contribution (ou perturbation) discontinue. Dans ce cas, la formulation variationnelle faible de l'équation de convection-diffusion (3.30), selon la méthode de Petrov-Galerkin, s'écrit :

$$\int_0^L \left(u\,\varphi_{h,x}\omega_h + \kappa\,\varphi_{h,x}\omega_{h,x} \right) dx + \sum_{i=1}^{nelt} \int_{x_i}^{x_{i+1}} p\left(u\,\varphi_{,x} + \kappa\,\varphi_{,xx} \right) dx = 0 \qquad (3.34)$$

Le deuxième terme du membre de gauche de l'équation (3.34) constitue la perturbation qu'il faut ajouter à la formulation de Galerkin dans le but de stabiliser la solution.

On peut voir facilement que, si $p = 0$, la formulation (3.34) correspond à la méthode de Galerkin (3.31).

3.8.2.1 Méthode SUPG

La méthode **SUPG** a été proposée par Hughes et ses collaborateurs [2]. C'est une variante de la méthode de Petrov-Galerkin basée sur le concept de la diffusion artificielle optimale [10].

En se basant sur ce concept, le problème (3.30) devient :

Trouver $\varphi \in [0, L] \subset \mathbf{R} \rightarrow \mathbf{R}$ tel que:

$$u\,\varphi_{,x} - (\kappa + \tilde{\kappa})\,\varphi_{,xx} = 0 \tag{3.35a}$$

$$\left.\begin{aligned} \varphi(0) &= \varphi_0 \\ \varphi(L) &= \varphi_L \end{aligned}\right\} \tag{3.35b}$$

où $\tilde{\kappa}$ est le coefficient de diffusion artificielle optimale. Dans ce cas, la formulation variationnelle faible de l'équation (3.35a) s'écrit :

$$\int_0^L \left(\omega\, u\, \varphi_{,x} + (\kappa + \tilde{\kappa})\, \omega_{,x}\, \varphi_{,x}\right) dx = 0 \tag{3.36}$$

Comme $\tilde{\kappa}$ possède un caractère directionnel (u dans le cas unidimensionnel), on parle alors de la méthode **SUPG** (Streamline Upwinding Petov-Galerkin).

La formulation faible, selon la méthode **SUPG**, de l'équation de convection-diffusion unidimensionnelle stationnaire (3.30a) s'écrit alors:

$$\int_0^L \left(\omega\, u\, \varphi_{,x} + \omega_{,x}\, \kappa \kappa_{,x}\right) dx + \sum_{i=1}^{nelt} \int_{x_i}^{x_{i+1}} \tau\ L_{conv}(\omega)\left(u\, \varphi_{,x} + \kappa\ \varphi_{,xx}\right) dx = 0 \qquad (3.37)$$

$$L_{conv}(\omega) = u\, \omega_{,x}$$

avec

$$\tau = \frac{h}{2u}\, \tilde{\xi}(Pe) \qquad (3.38)$$

où $\tilde{\xi}(Pe)$ étant une fonction du nombre de Peclet, particulièrement sensible aux zones où il faut ajouter de la diffusion artificielle [4].

$$\tilde{\xi}(Pe) = \min\left(1, \frac{Pe}{3}\right)$$

Le deuxième terme du membre de gauche de l'équation (3.37) représente l'opérateur **SUPG** qui constitue la perturbation qu'il faut ajouter à la méthode de Galerkin pour stabiliser la solution.

3.8.2.2 Méthode GLS

La méthode **GLS** est aussi une variante de la méthode de Petrov-Galerkin définie par l'addition du terme qui suit à la formulation de Galerkin

$$\ldots + \sum_{i=1}^{nelt} \int_{x_i}^{x_{i+1}} \tau\ L(\omega)\left(u\, \varphi_{,x} + \kappa\ \varphi_{,xx}\right) dx = 0 \qquad (3.39)$$

L'opérateur différentiel L comprend deux parties:

$$L_{conv}(\omega) = u\,\omega_{,x}, \qquad L_{diff}(\omega) = \kappa\,\omega_{,xx}$$

GLS correspond donc la définition:

$$p = \tau\, L(\omega)$$

Remarque : Pour des éléments linéaires, on a $L(\omega) = u\,\omega_{,x}$ sur l'intérieur des éléments; dans ce cas **GLS** est identique à **SUPG**

La généralisation des méthodes de stabilisation **SUPG** et **GLS**, au cas d'un système multidimensionnel instationnaire, fera l'objet du paragraphe suivant qui traitera des équations de Navier-Stokes.

3.8.2.3 Résolution de l'équation de concevtion-diffusion à l'aide d'une méthode *hp*

On va maintenant voir, à travers une étude numérique du problème simple de convection-diffusion unidimensionnel (3.30), si les méthodes spectrales *hp* pourraient présenter un avantage significatif par rapport aux méthode *h* classiques. On a fait les calculs pour trois nombres de Peclet, un Pe modéré, un Pe moyen et un Pe grand et ce avec deux formulation : une formulation de Galerkin et une formulation de Petrov-Galerkin avec la méthode **SUPG**.

On résoud dans le domaine $[-1, 1]$ le problème suivant :

$$u\,\varphi_{,x} - \kappa\,\varphi_{,xx} = 0$$

$$\varphi(-1) = 0 \quad \text{et} \quad \varphi(1) = 1$$

Sur les figures suivantes on montre les différentes solution numériques trouvées accompagnées de l'évolution de l'erreur en fonction de l'ordre d'interpolation pour chacun des cas (Pe_h représente le nombre de Peclet local).

METHODE *h* : Pe = 200.0, Pe_h = 1.25, h = 0.025

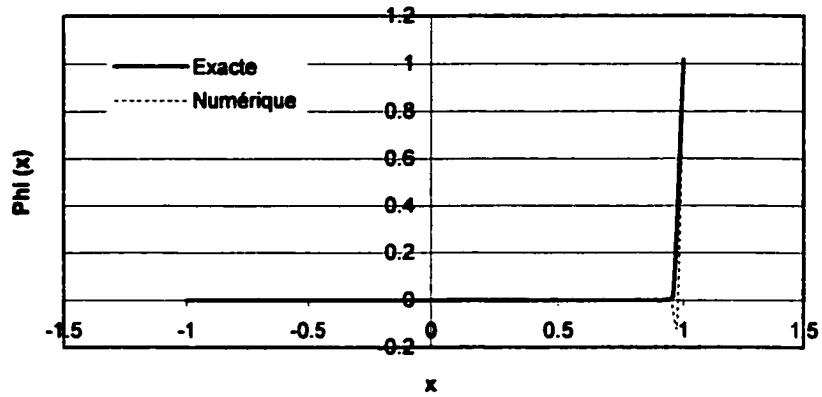


Figure 18- Solution numérique de l'équation de convection-diffusion avec une méthode de Galerkin de type *h*

METHODE *hp* : Pe = 200, Pe_h = 10 et h = 0.2

a- Méthode de Galerkin:

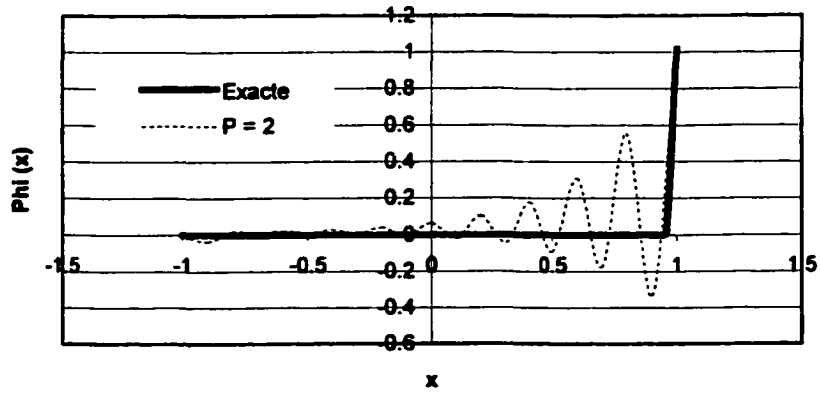


Figure 19a- Solutions numériques de l'équation de convection-diffusion avec la méthode de Galerkin pour P = 2

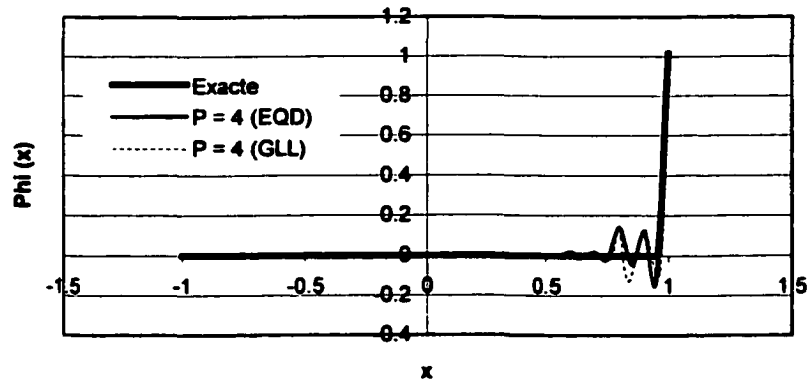


Figure 19b- Solutions numériques de l'équation de convection-diffusion avec la méthode de Galerkin pour P = 4

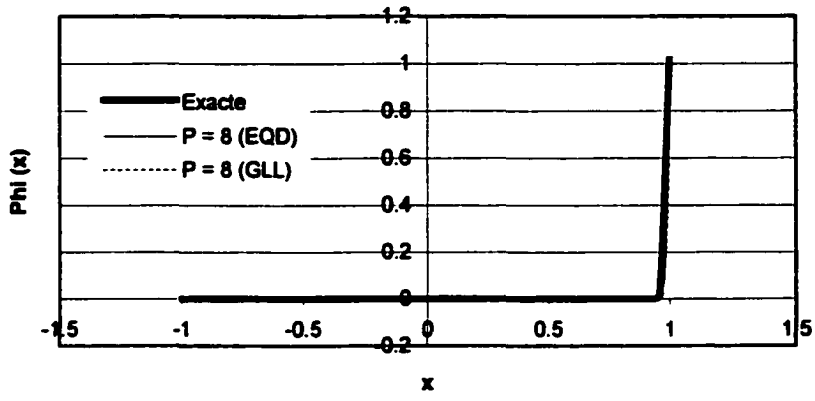


Figure 19c- Solutions numériques de l'équation de convection-diffusion avec la méthode de Galerkin pour P = 8

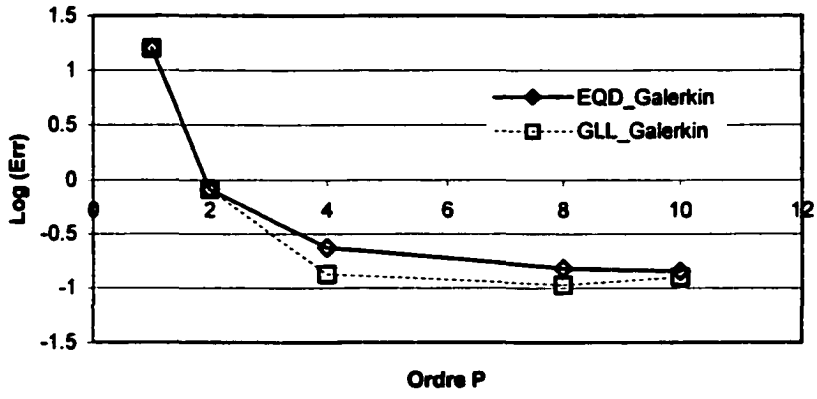


Figure 19d- Évolution de l'erreur pour les points EQD et les points de GLL

b- Méthode de Petrov-Galerkin:

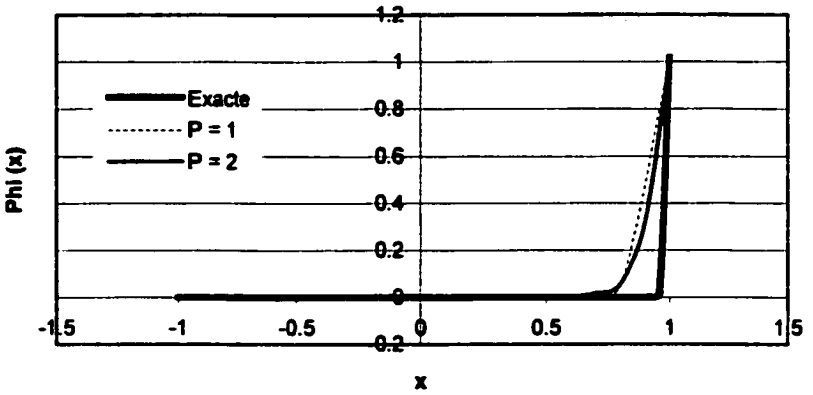


Figure 20a- Solutions numériques de l'équation de convection-diffusion avec la méthode de Petrov-Galerkin pour P = 1 et P = 2

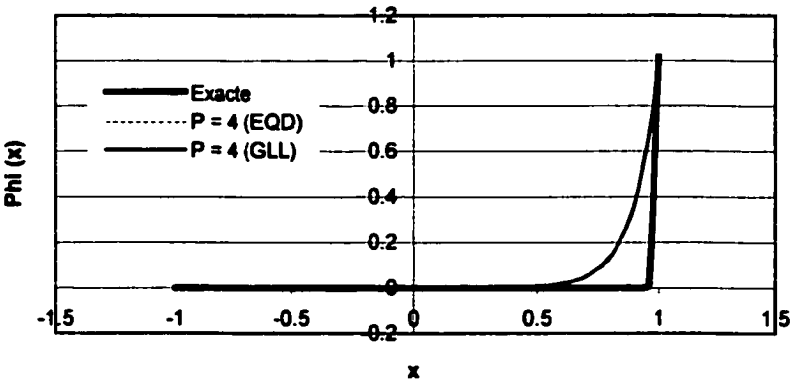


Figure 20b- Solutions numériques de l'équation de convection-diffusion avec la méthode de Petrov-Galerkin pour P = 4

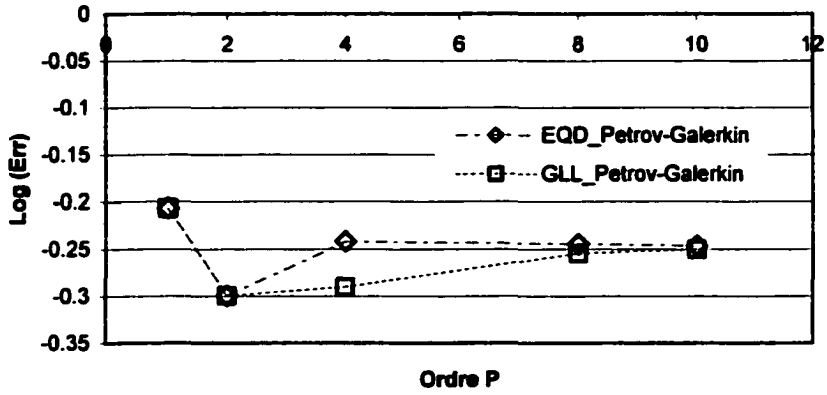


Figure 20c- Évolution de l'erreur pour les points EQD et les points de GLL

METHODE *hp* : Pe = 1000, Pe_h = 50 et h = 0.2

a- Méthode de Galerkin:

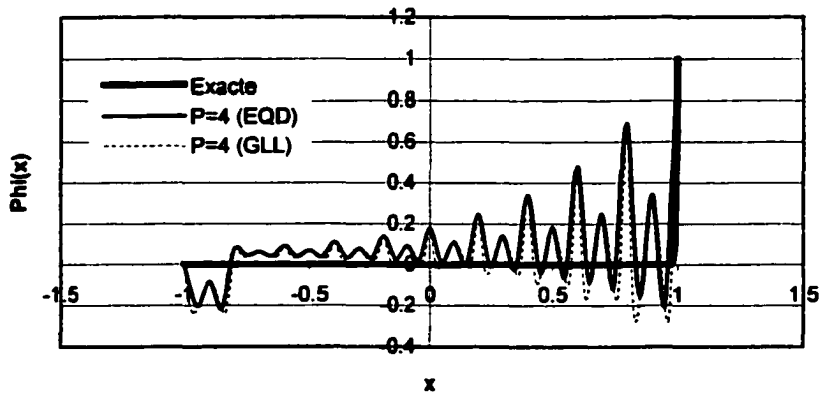


Figure 21a- Solutions numériques de l'équation de convection-diffusion avec la méthode de Galerkin pour P = 4

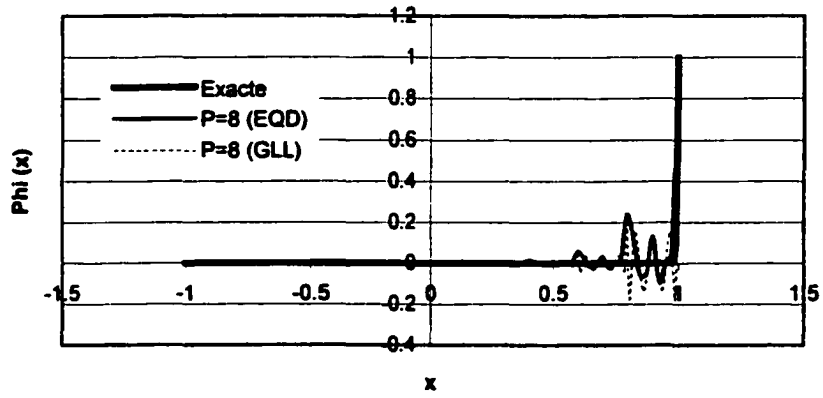


Figure 21b- Solutions numériques de l'équation de convection-diffusion avec la méthode de Galerkin pour P=8

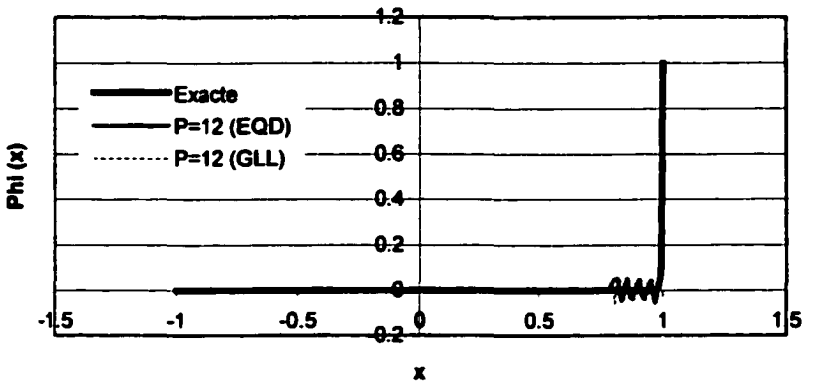


Figure 21c- Solutions numériques de l'équation de convection-diffusion avec la méthode de Galerkin pour P = 12

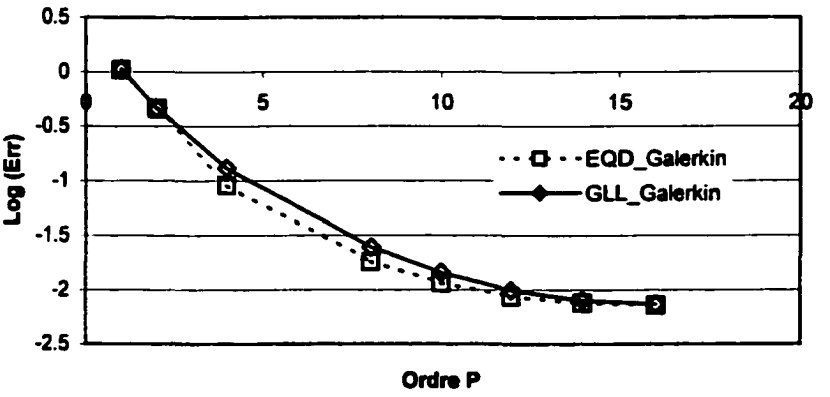


Figure 21d- Évolution de l'erreur pour les points EQD et les points de GLL

b- Méthode de Petrov-Galerkin:

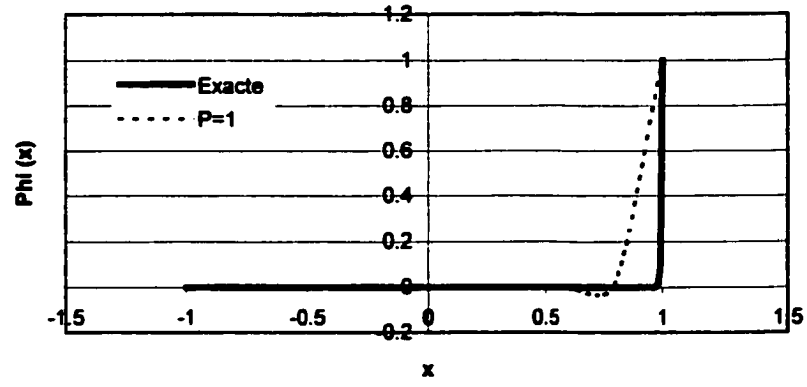


Figure 22a- Solutions numériques de l'équation de convection-diffusion avec la méthode de Petrov-Galerkin pour P = 1

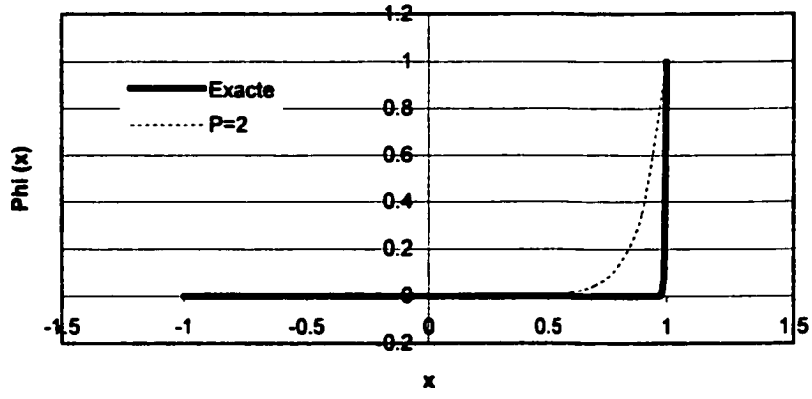


Figure 22b- Solutions numériques de l'équation de convection-diffusion avec la méthode de Petrov-Galerkin pour P = 2

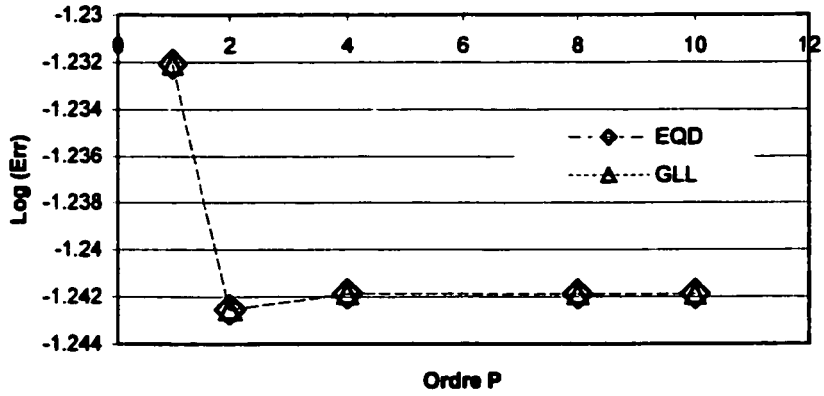


Figure 22c- Évolution de l'erreur pour les points EQD et les points de GLL

En analysant les figures 18, 19 et 21 on peut constater un certain avantage d'utiliser les méthodes *hp* plutôt que les méthodes *h*. On voit très bien que même en utilisant une formulation de Galerkin, en augmentant l'ordre d'interpolation, on parvient quand même à réduire énormément les oscillations de la solution numériques et on voit également que la convergence est très bonne. D'autre part une méthode *h* s'avère peu efficace à réduire les oscillations (figure 18). Toutefois, on constate que le choix des points d'interpolation n'est pas important pour ce genre de problème. En effet, les points GLL ne présentent pas un avantage remarqué par rapport aux points équidistants et seule l'augmentation de l'ordre d'interpolation aide à réduire les oscillations. Pour ce qui est de la formulation de Petrov-Galerkin (figures 20 et 22), ni le choix des points d'interpolation, ni l'augmentation de l'ordre n'ont d'effet sur la qualité de la solution numérique. On voit que la solution numérique est toujours diffusive comparée à la solution exacte. L'augmentation de l'ordre n'aide pas à améliorer la solution et l'erreur reste à peu près la même à partir de l'ordre 2 (figures 20c et 22c), et ce pour les deux types de points d'interpolation.

N.B: Le programme utilisé pour résoudre le problème de convection-diffusion se trouve à l'annexe 1.

3.8.3 Équations de Navier-Stokes

Maintenant que les techniques de stabilisation **SUPG** et **GLS** sont introduites, on peut passer à la généralisation des ces méthodes aux équations de Navier-Stokes.

3.8.3.1 Méthode de stabilisation SUPG

Selon la méthode **SUPG**, les formulations variationnelles faibles de type Galerkin, sous forme vectorielle (3.29a) et sous forme quasilinéaire(3.29b) sont remplacées respectivement par les formulations variationnelles faibles sous forme vectorielle et quasilinéaire suivantes:

$$\int_{\Omega} \left\{ \mathbf{W}.\left[\mathbf{A}_0\mathbf{V}_{,t} + \mathbf{F}^{conv}_{i,i}(\mathbf{V}) - \mathfrak{I}\right] + \mathbf{W}_{,i}.\left[\mathbf{F}^{diff}_{i,i}(\mathbf{V})\right] \right\} d\Omega$$

$$+\sum_{e}\int_{\Omega^e} \left\{ \left(\mathbf{A}^t_i.\mathbf{W}_{,i}\right)\tau \left[\mathbf{A}_0\mathbf{V}_{,t} + \mathbf{F}^{conv}_{i,i}(\mathbf{V}) - \mathbf{F}^{diff}_{i,i}(\mathbf{V}) - \mathfrak{I}\right] \right\} d\Omega^e$$

$$= \oint_{\Gamma} \left\{ \mathbf{W}.\left[\mathbf{F}^{diff}_{i,i}(\mathbf{V}).\mathbf{n}_i\right] \right\} d\Gamma \qquad (3.40)$$

$$\int_{\Omega} \left\{ \mathbf{W}.\left[\mathbf{A}_0\mathbf{V}_{,t} + \mathbf{A}_i\mathbf{V}_{,i} - \mathfrak{I}\right] + \mathbf{W}_{,i}.\left[\mathbf{K}_{ij}\mathbf{V}_{,j}\right] \right\} d\Omega$$

$$+\sum_{e}\int_{\Omega^e}\left\{\left(\mathbf{A}_i^t.\mathbf{W}_{,i}\right)\tau\left[\mathbf{A}_0\mathbf{V}_{,t}+\mathbf{A}_i\mathbf{V}_{,i}-\left(\mathbf{K}_{ij}\mathbf{V}_j\right)\mathbf{V}_j-\mathfrak{I}\right]\right\}d\Omega^e$$

$$=\oint_{\Gamma}\left\{\mathbf{W}.\left[\left(\mathbf{K}_{ij}\mathbf{V}_j\right).\mathbf{n}_i\right]\right\}d\Gamma \qquad (3.41)$$

$$\tilde{\underline{\underline{\tau}}}=\mathbf{A}_0^{-1}\underline{\underline{\tau}} \qquad (3.42)$$

Rappelons que dans le cas des variables conservatives, $\mathbf{V}=(\rho,\mathbf{U},E)^t$ et la matrice $\mathbf{A}_0$ devient l'identité. Par conséquent, la matrice $\tilde{\underline{\underline{\tau}}}$ se réduit à la matrice $\underline{\underline{\tau}}$ correspondant aux variables conservatives **V**.

La formulation variationnelle ci-dessus se distingue par deux propriétés importantes :

- C'est une méthode de résidus pondérés au sens qu'une solution exacte régulière du problème physique original reste encore une solution du problème variationnel. Ceci assure, non seulement une bonne précision de l'approximation mais, aussi une stabilité spatio-temporelle.

- La stabilité est assurée grâce au terme elliptique :

$$\sum_{e}\int_{\Omega^e}d\Omega^e\left\{\left(\mathbf{A}_i^t.\mathbf{W}_{,i}\right)\tilde{\underline{\underline{\tau}}}\;\mathbf{F}_i^{conv}(\mathbf{V})\right\} \qquad (3.43)$$

en écriture vectorielle ou

$$\sum_{e} \int_{\Omega^e} \left\{ \left(\mathbf{A}_i^t . \mathbf{W}_{,i} \right) \tilde{\underline{\tau}} \left[\mathbf{A}_i \mathbf{V}_i \right] \right\} d\Omega^e \tag{3.44}$$

en écriture quasi-linéaire.

3.8.3.2 Méthode de stabilisation GLS

Selon la méthode **GLS**, les formulations variationnelles faibles sous forme vectorielle et quasilinéaire s'écrivent :

$$\int_{\Omega} \left\{ \mathbf{W} . \left[\mathbf{A}_0 \mathbf{V}_{,t} + \mathbf{F}_{i,i}^{conv}(\mathbf{V}) - \mathfrak{I} \right] + \mathbf{W}_{,i} . \left[\mathbf{F}_{i,i}^{diff}(\mathbf{V}) \right] \right\} d\Omega$$

$$+ \sum_{e} \int_{\Omega^e} \left\{ \left[\mathbf{A}_i^t . \mathbf{W}_{,i} + \left(\mathbf{K}_{ij} \mathbf{V}_j \right)_{,j} \right] \tau \left[\mathbf{A}_0 \mathbf{V}_{,t} + \mathbf{F}_{i,i}^{conv}(\mathbf{V}) - \mathbf{F}_{i,i}^{diff}(\mathbf{V}) - \mathfrak{I} \right] \right\} d\Omega^e$$

$$= \oint_{\Gamma} \left\{ \mathbf{W} . \left[\mathbf{F}_{i,i}^{diff}(\mathbf{V}) . \mathbf{n}_i \right] \right\} d\Gamma \tag{3.45}$$

3.8.4 Définition de $\underline{\tau}$

La stabilité dépend énormément de la structure de la matrice de stabilisation $\underline{\tau}$. Le choix de cette matrice est alors fondamental pour le bon comportement des méthodes stabilisation. Ce choix peut être établi selon les critères suivants :

- $\underline{\underline{\tau}}$ est une matrice pleine de même dimension que les matrices $\mathbf{A}_i$.
- Les éléments de $\underline{\underline{\tau}}$ ont la dimension du temps.
- $\underline{\underline{\tau}}$ est une quantité locale dépendant de la géométrie de l'élément.
- $\underline{\underline{\tau}}$ doit se réduire à un scalaire donné par (3.38) dans le cas d'une équation scalaire unidimensionnelle.

A.Soulaïmani et M. Fortin [5] ont proposé et validé numériquement la définition de la matrice $\underline{\underline{\tau}}$ telle que :

$$\underline{\underline{\tau}} = \left[\sum \left| c_{ij} \mathbf{A}_j \right| \right]^{-1} \tilde{\xi}(Pe) \tag{3.46}$$

où $c_{ij} = \dfrac{\partial \xi_i}{\partial x_j}$ sont les coefficients de l'inverse de la matrice jacobienne de transformation géométrique.

Dans le cas d'un système d'équations unidimensionnelles, on a :

$$c_{ij} = c = \frac{\partial \xi}{\partial x} = \frac{2}{h}$$

où h est la taille de l'élément, et la matrice $\underline{\underline{\tau}}$ donnée par (3.46) se réduit à :

$$\underline{\underline{\tau}} = \frac{h}{2} \left| \mathbf{A}_j \right|^{-1} \tilde{\xi}(Pe) \tag{3.47}$$

La matrice $\underline{\underline{\tau}}$ telle que définie par (3.46) vérifie bien les critères énumérés précédemment. En particulier, cette matrice se réduit bien à l'expression (3.47) dans le

cas d'un système d'équations unidimensionnelles et au scalaire donné par (3.38) dans le cas d'une équation scalaire unidimensionnelle.

3.9 Résultats

Pour validér l'élément de paroi Log-Lin, on a commencé par un test simple d'un écoulement sur une plaque plane et sans gradient de pression adverse à un nombre de Reynolds modéré de Re = 200 000.

Les calculs on été dffectués sur un maillage de 104864 éléments :

- Dans la région de la paroi, on a 903 éléments prismatiques de taille $h = 0.005$
- Dans le reste du domaine, on a 103961 éléments tétraédriques.

Le nombre de nœuds total est 27360.

La solution obtenue est donnée dans la figure 23 par le profil de vitesse à une section se trouvant au milieu de la plaque.

À cette section la vitesse de frottement calculée est $u^* = 0.0602$. On constate une très bonne résolution car la vitesse à la hauteur h (qui correspond au premier point à partir de la surface solide) coïncide bien avec la courbe de spalding. Donc le comportement Log-Lin est assuré dans la région de la paroi. Au delà de h on retrouve un comportement de couche limite où la vitesse u se raproche de u_∞.

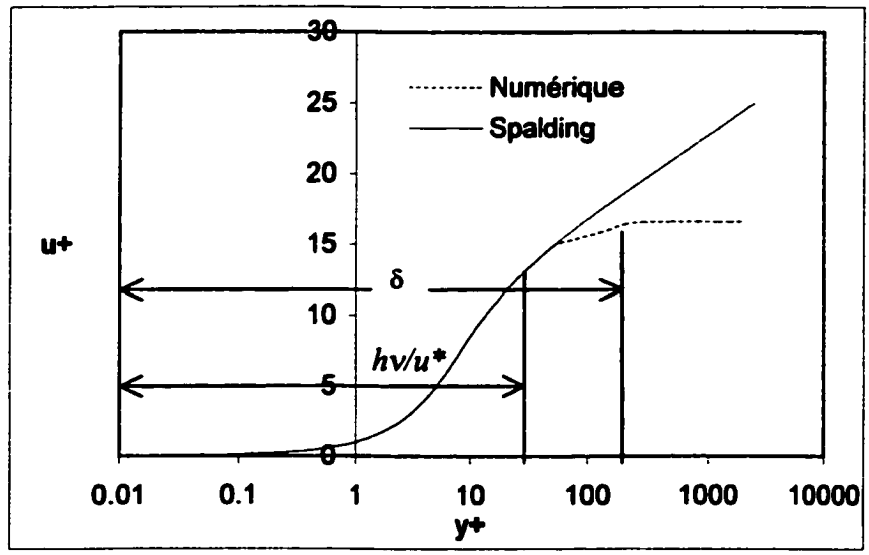


Figure 23- Profil de vitesse sur plaque plane à Re = 200 000

CONCLUSION

En résumé, Les méthodes d'ordre élevé avec une discrétisation spectrale présentent un outil numérique important qui peut être utilisé pour la modélisation des écoulement rapides.

Non seulement elles sont faciles à implémenter, mais aussi elles se sont révélées très précises et d'une grande performance à représenter fidèlement les zones d'oscillations et d'instabilités. Ainsi, couplées avec l'élément de paroi Log-Lin déjà mis au point, on pourrait s'attendre à des résultats importants quant à la modélisation des phénomènes de séparation dans le cas d'un gradient de pression adverse. Aussi, l'étude numérique de l'équation de convection diffusion révèle l'intérêt d'utiliser ces méthodes dans le but d'obtenir de meilleures solutions numériques et qui sont stables dans le cas d'écoulements rapides où la convection devient dominante.

Les objectifs fixés ont été accomplis. On a effectués tous les tests concernant la performance des méthodes spectrales *p*. Un élément de paroi Log-Lin est devellopé. On a mis en œuvre un maillage 3D hybride (prisme / tétraèdre) pour un cas test d'écoulement sur plaque plane ainsi qu'une bibliothèques de routines nécessaires pour implémenter les méthodes spectrales.

RECOMMANDATIONS

Tous les outils nécessaires pour commencer les simulations sans déjà implémentés. Pour commencer, à court terme, on peut déjà étendre les tests sur plaque plane sans gradient de pression à des nombres de Reynolds (Re) élevés afin de bien mettre en évidence l'avantage de l'élément Log-Lin. Ensuite on pourrait coupler l'élément Log-Lin avec une correction spectrale pour voir l'effet de cette dernière dans le cas d'un écoulement sur une plaque plane avec un gradient de pression adverse.

Pour ce qui est des travaux futures à long terme, on recommande une généralisation pour des cas pratiques. Entre autres, les écoulements rapides sur des surfaces courbées comme les ailes d'avions.

BIBLIOGRAPHIE

[1] A. Civindi, A. Quarteroni, E. Zampieri (1999). *Numerical solution of linear elastic problems by spectral collocation methods.* Computer methods in applied mechanics and engineering, vol. 169, pp. 123-134.

[2] A. N. Brooks & T. J. R. Hughes (1980). *Streamline upwind/Petrov-Galerkin methods for advection dominated flows.* Third International Conference on finite elements methods in fluid flows, Banff, Canada.

[3] A. Soulaïmani, A. Forest, Z. Feng, A. BenElhaj & Y. Azami (2001). *A distributed computing-based methodology for computational nonlinear aeroelasticity.* CASI 48th annual Conference, Toronto, Canada, pp. 123-135.

[4] A. Soulaïmani & A. Rebain (2001). *Numerical simulation of tow-dimensional compressible turbulent flows in ejectors.* CSME, vol. 25, n° 2.

[5] A. Soulaïmani & M. Fortin (1994). *Finite element solution of compressible viscous flows using conservative variables.* Computer methods in applied mechanics and engineering, vol. 118, pp. 319-350.

[6] A. Soulaïmani, Y. Saad & A. Rebain (1999). *An edge based stabilized finite element method for solving compressible flows: Formulation and parallel implementation.* Canadian Aeronautics and Space Institute, 46th Annual Conference, pp. 123-132, Montréal.

[7] A. T. Patera (1984). *A spectral element method for fluid dynamics laminar flow in channel expansion.* J. Comp Physics, vol. 54 n° 3, pp. 468-488.

[8] C. Canuto, M, Hussaini, A. Quateroni & T. A. Zang (1987). *Spectral methods in fluid dynamics.* Springer Verlag.

[9] D. C. Wilcox (1997). *Turbulence modeling for CFD.* DCW Industries, 2nd edition.

[10] El Kadri El yamani Nacer-Eddine (1992). *Modelisation des ecoulements compressibles visqueux par la methode des elements finis.* Memoire presente a l'école des gradués de l'université Laval pour l'obtention du grade de Maîtrise es science.

[11] E.M. Ronquist & A. T. Patera (1987). *Spectral Element Multigrid. I. Formulation and Numerical Results.* International journal of numerical methods in engineering, vol. 17, p. 2273

[12] Frank M. White (1991). *Viscous fluid flow.* McGraw-Hill.

[12] G. E. Karniadakis, Spencer & J Sherwin (1999). *Spectral/hp element methods for CFD.* Oxford.

[13] G. J. Le Beau, S. E. Ray, S. K. Aliabadi & T. E. Tezduyar (1993). *SUPG finite element computation of compressible flows with the entropy and conservation variables formulation.* Computer methods in applied mechanics and engineering, vol. 104, pp. 397-422.

[14] H. Manouzi & M. Fortin (1991). *A treatment of wall boundaries for turbulent flows by the use of a transmission finite element method.* International journal for numerical methods in engineering, vol. 31, pp. 113-126.

[15] J. O. Hinze (1989). *Turbulence.* McGraw-Hill, 2nd edition.

[16] J. Cousteix (1989). *Turbulence et couche limite.* CEPADUES-EDITIONS.

[17] Karol, Z. Korczak & A. T. Patera (1986). *An isoparametric spectral element method for solution of the Navier-Stokes equations in complex geometry.* J. Comp Physics, vol. 62, pp. 361-382.

[18] M. R. Shumack, W, Schlutz & J. P. Boyd (1991). *Spectral method solution of the stokes equation on nonstaggered grids.* J. Comp Physics, vol. 94, pp. 30-58.

[19] Neal & T. Frink (1998). *Tetrhedral unstructured Navier-Stokes method for turbulent flows.* AIAA Journal, vol. 36, n° 11.

[20] P. R. Spalart & S. R. Allmaras (1998). *A one-equation turbulence model for aerodynamic flows.* La recherché aerospatiale, 1, pp. 5-21.

[21] T. Utnes & T. S. Meling (1993). *Treatment of turbulent wall boundary conditions using linear-logarithmic elements.* Computer methods in applied mechanics and engineering, vol. 104, pp. 49-76.

[22] Y. Saad (1996). *Iterative methods for sparse linear systems.* PWS.

[23] Y. Saad & M. H. Schultz (1996). GMRES: *a generalized minimal residual algorithm for solving nonsymmetric linear systems.* SIAM Journal on scientific and statistical computing, 7, pp. 856-869.

ANNEXE 1

LISTE DES ROUTINES

La routine qui calcule le résidu élémentaire :

```
c=======================================================================
      subroutine residu_spec(iel,icode,cfl,inel0,idle0,vcore,vpree,vdle,vdlev,vfe)
c=======================================================================
      use global_data,only : P,ndleS,ppp,npgz,ipg,inelc,ndim,
     &                       ichoc,kro,ku1,ku2,ku3,
     &                       kenr,knut,kpok,vniL,vniS,wpg,
     &                       vj,vj1,detj,
     &                       vden,qm1,qm2,qm3,venr,visco,vtem,vpres,
     &                       veloc1,veloc2,veloc3,denx,deny,denz,
     &                       u1x,u1y,u1z,u2x,u2y,u2z,u3x,u3y,u3z,
     &                       prex,prey,prez,temx,temy,temz,v1x,v1y,v1z,
     &                       v2x,v2y,v2z,v3x,v3y,v3z,enrx,enry,enrz,
     &                       vnutx,vnuty,vnutz,dens,u1,u2,u3,enrg,vnut,pres,
     &                       temp,v1,v2,v3,divu,divv,vort1,vort2,vort3,
     &                       vort,vv,uu,vmua,fadv,fstabx,fstaby,fstabz,
     &                       gradx,grady,gradz,vnigL,vnigS,
     &                       vnL,vnS,vnixL,vnixS,vniyL,vniyS,vnizL,vnizS,
     &                       a1,a2,a3,h1,h2,h3,tau,
     &                       gu,vel,hel,dc,coef,gama,
     &                       gama1,ren,prdl,prdt,
     &                       zero,un,deux,trois,quatre,deuti,eps1,
     &                       vzeta,vzint

      implicit none

      real,parameter :: dist= 0.0001d0
      real*8  vites,denss,vmul,vnu(3),uet(3)
      integer i,j,k,ii,jj,kk,ll,mm,nn,pp,ip,ik,i1,iel,iii,
     &        ipp1,ipp2,LL1,LL2,ss1,ss2, icode,inel0,idle0

      real*8 vdent(inelc),qm1t(ppp),qm2t(inelc),qm3t(inelc),
     &       venrt(inelc),viscot(inelc), vcore(18),vdle(ndleS)
     &      , vfe(ndleS),dpas,cfl,cel, z14

      real*8  denst,u1t,u2t,u3t,enrgt,vnutt, vpree(*)

      real*8 s1,s2,vmut,
     &       sig11,sig12,sig13,sig21,sig22,sig23,sig31,sig32,sig33,
     &       heat1,heat2,heat3

      real*8 tausc,vnume,sigma,vdeno,dct,relax,cb1,cb2,vkr
      real*8 vfes(ndleS),fdif1(5),fdif2(5),fdif3(5)
      real*8 vdlev(ndleS), eps

      data cb1/0.1355d0/, cb2/0.622d0/, vkr/0.41d0/
      deuti = deux/trois
```

```
      sigma = deuti
      eps= 1.d-16
      eps1= 1.d-06
      ichoc = 2
      relax= vpree(8)
c*********************************************************************
      if(icode.eq.1)then
c*********************************************************************
      inel0= 6
      idle0= 36
      return
      endif
c*********************************************************************
      if(icode.eq.2)then
c*********************************************************************
c
c------ Les positions des pts d interpolation

      call glob(P,vzint)
c
c------ Les vecteurs poids et les positions des points d integration

      call gauss(npgz,vzeta)
      call poids_gauss(wpg,vzeta,npgz,ipg)
c
c*********************************************************************
      kro = (/1,1+(5+P),1+2*(5+P),1+3*(5+P),7+3*(5+P),13+3*(5+P)/)

      If (p.eq.1) then
      ku1(1:6)=(/2,2+(5+P),2+2*(5+P),2+3*(5+P),8+3*(5+P),14+3*(5+P)/)
      Endif

      if (p.ne.1) then
      ku1(1:3)=(/2,2+(5+P),2+2*(5+P)/)
      pp=0
      do j = 2, P
      ku1(4+pp) =  1+j
      ku1(5+pp) =  1+j+(5+P)
      ku1(6+pp) =  1+j+2*(5+P)
      pp= pp+3
      enddo
      ku1(ppp-2:ppp)= (/2+3*(5+P),8+3*(5+P),14+3*(5+P)/)
      Endif

      ku2 = (/2+P,2+P+(5+P),2+P+2*(5+P),8+P+2*(5+P),
     &       14+P+2*(5+P),20+P+2*(5+P)/)
      ku3 = (/3+P,3+P+(5+P),3+P+2*(5+P),9+P+2*(5+P),
     &       15+P+2*(5+P),21+P+2*(5+P)/)
      kenr= (/4+P,4+P+(5+P),4+P+2*(5+P),10+P+2*(5+P),
     &       16+P+2*(5+P),22+P+2*(5+P)/)
      knut= (/5+P,5+P+(5+P),5+P+2*(5+P),11+P+2*(5+P),
     &       17+P+2*(5+P),23+P+2*(5+P)/)
```

```
      if (P.eq.1) then
       do i=1,inelc
       do j=1,inelc
         kpok(j+(i-1)*(P+5))= i+(j-1)*6
       enddo
       enddo
      else

      do i=1, inelc
       if (i.le.3) then
         do j=1, P+5
         if (j.lt.3) kpok(j+(i-1)*(P+5))= i+(j-1)*6
         if ((j.ge.3).and.(j.lt.P+2)) kpok(j+(i-1)*(P+5))= i+(j-3)*3+9
         if (j.ge.P+2) kpok(j+(i-1)*(P+5))= i+3+(2+P)*3+(j-P-2)*6
         enddo
       else
         do j=1, 6
         if (j.eq.1) kpok(j+(i-4)*6+3*(P+5))= i
         if (j.eq.2) kpok(j+(i-4)*6+3*(P+5))= i+3*P+3
         if (j.eq.3) kpok(j+(i-4)*6+3*(P+5))= i+3*P+9
         if (j.eq.4) kpok(j+(i-4)*6+3*(P+5))= i+3*P+15
         if (j.eq.5) kpok(j+(i-4)*6+3*(P+5))= i+3*P+21
         if (j.eq.6) kpok(j+(i-4)*6+3*(P+5))= i+3*P+27
         enddo
       endif
      enddo
      endif

      return
      endif
c========================= PROPRIETES DU FLUIDE ========================
      gama  = vpree(2)
      gama1 = gama - un
      ren   = vpree(1)
      prdl  = vpree(3)
      prdt  = 0.9d0
c======================= VARIABLES PHYSIQUES ===========================
c
      DO i=1,inelc

c------ Extraction des degrés de libérté

      ii = kro(i)
      jj = ku2(i)
      kk = ku3(i)
      ll = kenr(i)
      mm = knut(i)


c
```

```
c------ Density
      vden(i) = vdle(ii)
      vdent(i)= vdlev(ii)

c------ Momentum in x (Enrichie)

      if (i.le.3) then
      nn = ku1(i)
      qm1(i)  = vdle(nn)
      qm1t(i) = vdlev(nn)
      if (p.ne.1) then
      pp=0
      do j=2, P
        qm1(i+3+pp)  = vdle(ku1(i+3+pp))
        qm1t(i+3+pp) = vdlev(ku1(i+3+pp))
      pp=pp+3
      enddo
      endif
      endif
      if (i.gt.3) then
      nn=ku1(i-3+3*P)
      qm1(i-3+3*P)=vdle(nn)
      qm1t(i-3+3*P) = vdlev(nn)
      endif
c
c------ Momentum in y and z

      qm2(i)  = vdle(jj)
      qm2t(i) = vdlev(jj)

      qm3(i)  = vdle(kk)
      qm3t(i) = vdlev(kk)
c
c------ Energy

      venr(i) = vdle(ll)
      venrt(i)= vdlev(ll)
c
c------ Kinematic turbulent viscosity

      visco(i) = vdle(mm)
      viscot(i)= vdlev(mm)
```

```
c------ Temperature

      if (i.le.3) then
      gu = qm1(i)*qm1(i) + qm2(i)*qm2(i) + qm3(i)*qm3(i)
      if (p.ne.1) then
      pp=0
      do j=2, P
        gu = gu + qm1(i+3+pp)*qm1(i+3+pp)
      pp=pp+3
      enddo
      endif
      endif
      if (i.gt.3) then
      gu = qm1(i-3+3*P)*qm1(i-3+3*P)+qm2(i)*qm2(i)+qm3(i)*qm3(i)
      endif
c------
      vtem(i) = (venr(i) - gu/(2.0*vden(i)))/vden(i)
c------ Pressure
      vpres(i)= gama1*vden(i)*vtem(i)
c
c------ Velocity in x  (Enrichie)

      if (i.le.3) then
      veloc1(i) = qm1(i)/vden(i)
      if (p.ne.1) then
      pp=0
      do j=2, P
       veloc1(i+3+pp) = qm1(i+3+pp)/vden(i)
      pp=pp+3
      enddo
      endif
      endif

      if (i.gt.3) veloc1(i-3+3*P) = qm1(i-3+3*P)/vden(i)
c
c------ Velocity in y and z direction

      veloc2(i) = qm2(i)/vden(i)
      veloc3(i) = qm3(i)/vden(i)
c*****************************************************************
      ENDDO      ! Fin de la boucle sur les noeuds
c*****************************************************************
c------ update friction velocity

      do i=1, inelc
      if (i.gt.3) then
      vites = sqrt(veloc1(i)**2.d0 + veloc2(i)**2.d0 + veloc3(i)**2.d0)
      denss = vden(i)
      vmul = vtem(i)**(0.76d0)
      vnu(i-3) = visco(i)

      if(vites.gt.eps1)then
```

```
      call wall_function(2,vites,denss,vmul,vnu(i-3),ren,
     &                   dist,iel,70,10,uet(i-3))
      endif
      endif
      enddo
c=======================================================================
      call shape_prisme (uet,dist,vnut)
c============================ INITIALISATION ============================
c
      DO i=1,ndleS
        vfes(i)= 0.d0
        vfe(i) = 0.d0
      ENDDO
c
c************* BOUCLE SUR LES POINTS D INTEGRATION ***************
c
      LL1= 0
      LL2= 0
      ss1= 0
      ss2= 0
      ipp1 = 0
      ipp2 = 0
c***********************************************************************
      DO ip = 1,ipg
c***********************************************************************
c
      LL1= 7+ipp1
      LL2= 24+ipp1
      ss1= 1+ppp+ipp2
      ss2= 4*ppp+ipp2
c
c====== Matrice jacobienne, son determinant et son inverse ======
c
      call jacob(vniL(LL1:LL2),vcore,ndim,inelc,vj,vj1,detj)


c------ Taille de l element
c
      if(DETJ.lt.0.d0)then
      write(6,*)' DETJ negatif iel=', iel
      call flush(6)
      stop
      endif

c      write(6,*)' DETJ=', detj
c      call flush(6)
      vel = detj/4
      z14= vcore(3)-vcore(12)

      hel = (vel)**(1.d0/3.d0)
      hel= abs(hel)
```

```
c     write(6,*)' HEL=', hel, vden
c     call flush(6)
c
c==== Dérivees spatiales des fonctions d interpolation Lin ====
c
      call dnidx(vniL(LL1:LL2),vj1,ndim,inelc,vnigL)

      call dnidx(vniS(ss1:ss2),vj1,ndim,ppp  ,vnigS)

      do ik = 1,inelc
      vnixL(ik)= vnigL(ik)
      vniyL(ik)= vnigL(ik+inelc)
      vnizL(ik)= vnigL(ik+2*inelc)
      enddo

      do ik = 1,ppp
      vnixS(ik)= vnigS(ik)
      vniyS(ik)= vnigS(ik+ppp)
      vnizS(ik)= vnigS(ik+2*ppp)
      enddo
c
c=================== SPACE DERIVATIVES OF DOF ==================
c
      denx = 0.0d0
      deny = 0.0d0
      denz = 0.0d0

      u1x  = 0.0d0
      u1y  = 0.0d0
      u1z  = 0.0d0

      u2x  = 0.0d0
      u2y  = 0.0d0
      u2z  = 0.0d0

      u3x  = 0.0d0
      u3y  = 0.0d0
      u3z  = 0.0d0

      enrx = 0.0d0
      enry = 0.0d0
      enrz = 0.0d0

      vnutx= 0.0d0
      vnuty= 0.0d0
      vnutz= 0.0d0

      prex = 0.0d0
      prey = 0.0d0
      prez = 0.0d0

      temx = 0.0d0
```

```
      temy = 0.0d0
      temz = 0.0d0


      v1x  = 0.0d0
      v1y  = 0.0d0
      v1z  = 0.0d0

      v2x  = 0.0d0
      v2y  = 0.0d0
      v2z  = 0.0d0

      v3x  = 0.0d0
      v3y  = 0.0d0
      v3z  = 0.0d0
c
c------ SPECTRAL INTERPOLATION  ( Momentum in x )

      do j=1, ppp

      u1x  = u1x + vnixS(j)*qm1(j)
      u1y  = u1y + vniyS(j)*qm1(j)
      u1z  = u1z + vnizS(j)*qm1(j)

      v1x  = v1x + vnixS(j)*veloc1(j)
      v1y  = v1y + vniyS(j)*veloc1(j)
      v1z  = v1z + vnizS(j)*veloc1(j)

      Enddo
c
c------ LINEAR INTERPOLATIONS
c
c------ Density

      DO k=1, inelc

      denx = denx + vnixL(k)*vden(k)
      deny = deny + vniyL(k)*vden(k)
      denz = denz + vnizL(k)*vden(k)

      Enddo
c
c------ Momentum

      DO k=1, inelc

      u2x  = u2x + vnixL(k)*qm2(k)
      u2y  = u2y + vniyL(k)*qm2(k)
      u2z  = u2z + vnizL(k)*qm2(k)

      u3x  = u3x + vnixL(k)*qm3(k)
      u3y  = u3y + vniyL(k)*qm3(k)
```

```
      u3z  = u3z + vnizL(k)*qm3(k)

      Enddo
c
c------ Energy

      DO k=1, inelc

      enrx = enrx + vnixL(k)*venr(k)
      enry = enry + vniyL(k)*venr(k)
      enrz = enrz + vnizL(k)*venr(k)

      Enddo
c
c------ Kinematic turbulent viscosity

      DO k=1, inelc

      vnutx= vnutx + vnixL(k)*visco(k)
      vnuty= vnuty + vniyL(k)*visco(k)
      vnutz= vnutz + vnizL(k)*visco(k)

      Enddo
c
c------ Pressure

      DO k=1, inelc

      prex = prex + vnixL(k)*vpres(k)
      prey = prey + vniyL(k)*vpres(k)
      prez = prez + vnizL(k)*vpres(k)

      Enddo
c
c------ Temperature

      DO k=1, inelc

      temx = temx + vnixL(k)*vtem(k)
      temy = temy + vniyL(k)*vtem(k)
      temz = temz + vnizL(k)*vtem(k)

      Enddo

c------ Velocity

      DO k=1, inelc

      v2x = v2x + vnixL(k)*veloc2(k)
      v2y = v2y + vniyL(k)*veloc2(k)
      v2z = v2z + vnizL(k)*veloc2(k)
```

```
      v3x = v3x + vnixL(k)*veloc3(k)
      v3y = v3y + vniyL(k)*veloc3(k)
      v3z = v3z + vnizL(k)*veloc3(k)

      Enddo
c*************************************************************************
c
c------ Divergence of velocity

      divv = v1x + v2y + v3z
c
c------ Divergence of momentum

      divu = u1x + u2y + u3z
c
c------ Vorticity

      vort1 = v3y - v2z
      vort2 = v1z - v3x
      vort3 = v2x - v1y

      vort = sqrt(vort1*vort1 + vort2*vort2 + vort3*vort3)

CAS

c
c========================== GRADIENT VECTORS ============================
c
      gradx(1) = denx
      gradx(2) = u1x
      gradx(3) = u2x
      gradx(4) = u3x
      gradx(5) = enrx

      grady(1) = deny
      grady(2) = u1y
      grady(3) = u2y
      grady(4) = u3y
      grady(5) = enry




      gradz(1) = denz
      gradz(2) = u1z
      gradz(3) = u2z
      gradz(4) = u3z
      gradz(5) = enrz
c
c------ Shape functions
```

```
      DO j=1, inelc
      vnL(j) = vniL(ipp1+j)
      ENDDO

      DO j=1, ppp
      vnS(j) = vniS(ipp2+j)
      enddo
c
c------
      dens  = zero
      denst = zero
      u1    = zero
      u2    = zero
      u3    = zero
      u1t   = zero
      u2t   = zero
      u3t   = zero
      enrg  = zero
      enrgt = zero
      vnut  = zero
      vnutt = zero
      pres  = zero
      temp  = zero
      v1    = zero
      v2    = zero
      v3    = zero
c
c------ LINEAR INTERPOLATIONS

      DO k=1,inelc
c
c------ Density

      dens  = vden(k)*vnL(k) +  dens
c
c------ Momentum in y and z

      u2  = qm2(k)*vnL(k) + u2
      u3  = qm3(k)*vnL(k) + u3




c------ Energy

      enrg  = venr(k)*vnL(k) + enrg
c
c------ Kinematic turbulent viscosity

      vnut  = visco(k)*vnL(k) + vnut
c
```

```
c------ Pressure

      pres  = vpres(k)*vnL(k) + pres
c
c------ Temperature

      temp  = vtem(k)*vnL(k) +  temp
c
c------ Velocity

      v2 = veloc2(k)*vnL(k) + v2
      v3 = veloc3(k)*vnL(k) + v3
c------
      ENDDO
c
c------ SPECTRAL INTERPOLATIONS ( Momentum & velocity in x )

      DO k=1,ppp

      u1  =  qm1(k)*vnS(k) +  u1
      v1  =  veloc1(k)*vnS(k) + v1

      ENDDO
c
c------ Velocity norm

      vv = sqrt(v1*v1 + v2*v2 + v3*v3)

c---- local time step
      cel = gama*pres/dens
      if(cel.lt.0.d0)then
       write(6,*)'  cel negative dans iel=',iel,pres,dens
       stop
      endif
      dpas = CFL*hel/(vv+sqrt(cel) +ren/hel)

      denst =  zero
      u1t =  zero
      u2t = zero
      u3t = zero
      enrgt = zero
      vnutt = zero
      DO k=1,inelc
      denst = (vdent(k)*vnL(k))/dpas + denst
      u1t = (qm1t(k)*vnS(k))/dpas + u1t
      u2t = (qm2t(k)*vnL(k))/dpas + u2t
      u3t = (qm3t(k)*vnL(k))/dpas + u3t
      enrgt = (venrt(k)*vnL(k) )/dpas  + enrgt
      vnutt = (viscot(k)*vnL(k))/dpas + vnutt
      enddo
c
c------ Momentum norme square
```

```
      uu = u1*u1 + u2*u2 + u3*u3
c
c============================ INITIALISATION ============================
      DO i=1, 5
        fadv(i)  = 0.d0
      ENDDO
c============================ ADVECTION FLUX ============================
      call aimat_spec()

      do i=1,5
      do j=1,5
        fadv(i)= fadv(i)+ a1(i,j)* gradx(j) + a2(i,j)* grady(j)
     &    + a3(i,j)* gradz(j)
      enddo
      enddo
c
c============================ tau matrix calculation ============================
      if(vpree(10).ne.zero) call tau3d_spec(0.0d0)
c============================ INITIALISATION ============================
      do i=1,5   ! nb d equations de NS
        fstabx(i) = 0.d0
        fstaby(i) = 0.d0
        fstabz(i) = 0.d0
      enddo
c============================ STABILISATION FLUX ============================
      if(vpree(10).ne.zero) then

      h1 = 0.0d0
      h2 = 0.0d0
      h3 = 0.0d0

      do i=1, 5
      do j=1, 5
      do k=1, 5
        h1(i,j) = h1(i,j) + a1(i,k)*tau(k,j)
        h2(i,j) = h2(i,j) + a2(i,k)*tau(k,j)
        h3(i,j) = h3(i,j) + a3(i,k)*tau(k,j)
      enddo
      enddo
      enddo



      do i=1, 5
      do j=1, 5
        fstabx(i) = fstabx(i) + vpree(10)* h1(i,j)*fadv(j)
        fstaby(i) = fstaby(i) + vpree(10)* h2(i,j)*fadv(j)
        fstabz(i) = fstabz(i) + vpree(10)* h3(i,j)*fadv(j)
      enddo
      enddo
c
```

```
      endif
c
c========================= SHOCK CAPTURING OPERATOR ========================
c
      if(vpree(9).ne.0.d0) then

      call shock3d_spec(fadv,ichoc)
      dc = dc*vpree(9)
      if(ichoc.eq.1) dc = dc*hel/2.d0
      else
      dc = 0.d0
      endif
     if(dc.lt.sqrt(eps))dc= uu*hel/2.d0
c============================== INITIALISATION ==============================
c
      DO i=1, 5    !(nombre d equations NS)
        fdif1(i) = 0.d0
        fdif2(i) = 0.d0
        fdif3(i) = 0.d0
      ENDDO
c
c========================== DIFFUSION FLUX ==========================
c
c======= Fluid viscosity augmented or not by shock capt one

      vmua = (1.d0/vpree(1))*( (temp)**0.76d0 )
c
c------ dynamic turbulent viscosity

      vnut = dexp(vnut)
      vmut = dens*vnut

      s1 = vmua + vmut/ren
      s2 = vmua*(gama/vpree(3)) + (gama*vmut)/(ren*prdt)

c     s1 = vmua
c     s2 = s1*(gama/vpree(3))

c------ Stresses

      sig11 = s1*(deux*v1x - deuti*divv)
      sig22 = s1*(deux*v2y - deuti*divv)
      sig33 = s1*(deux*v3z - deuti*divv)

      sig12 = s1*(v1y + v2x)
      sig21 = sig12

      sig13 = s1*(v1z + v3x)
      sig31 = sig13

      sig23 = s1*(v2z + v3y)
      sig32 = sig23
```

```
c
c------ Heat flux

      heat1 = - s2*temx
      heat2 = - s2*temy
      heat3 = - s2*temz
c-------------------------------------------------------
      fdif1(1) = 0.0d0
      fdif1(2) = sig11
      fdif1(3) = sig21
      fdif1(4) = sig31
      fdif1(5) = sig11*v1 + sig12*v2 + sig13*v3 - heat1

      fdif2(1) = 0.0d0
      fdif2(2) = sig12
      fdif2(3) = sig22
      fdif2(4) = sig32
      fdif2(5) = sig21*v1 + sig22*v2 + sig23*v3 - heat2

      fdif3(1) = 0.0d0
      fdif3(2) = sig13
      fdif3(3) = sig23
      fdif3(4) = sig33
      fdif3(5) = sig31*v1 + sig32*v2 + sig33*v3 - heat3

c
c========================== RESIDUAL VECTOR ==========================
coef = wpg(ip)*detj

c------ Corresponding to continuity equation

      do i=1, inelc

      vfes(i) = vfes(i) + coef*(vnL(i)*denst
     &  + vnL(i)*fadv(1)
     &  + vnixL(i)*fdif1(1) + vniyL(i)*fdif2(1) + vnizL(i)*fdif3(1) )
     &  + coef*(
     &  vnixL(i)*fstabx(1) + vniyL(i)*fstaby(1) + vnizL(i)*fstabz(1))
     &  + coef*dc*(
     &  vnixL(i)*gradx(1) + vniyL(i)*grady(1) + vnizL(i)*gradz(1))

      Enddo


c
c------ Corresponding to x direction momentum equation

      do i=1, ppp

      vfes(i+6) = vfes(i+6) + coef*(vnS(i)*ult
     &  + vnS(i)*fadv(2)
     &  + vnixS(i)*fdif1(2) + vniyS(i)*fdif2(2) + vnizS(i)*fdif3(2))
```

```
     & + coef*(
     & vnixS(i)*fstabx(2) + vniyS(i)*fstaby(2) + vnizS(i)*fstabz(2))
     & + coef*dc*(
     & vnixS(i)*gradx(2) + vniyS(i)*grady(2) + vnizS(i)*gradz(2))

       Enddo
c
c------ Corresponding to y direction momentum equation

       do i=1,inelc

       vfes(i+6+ppp) = vfes(i+ 6 +ppp) + coef*(vnL(i)*u2t
     & + vnL(i)*fadv(3)
     & + vnixL(i)*fdif1(3) + vniyL(i)*fdif2(3) + vnizL(i)*fdif3(3) )
     & + coef*(
     & vnixL(i)*fstabx(3) + vniyL(i)*fstaby(3) + vnizL(i)*fstabz(3))
     & + coef*dc*(
     & vnixL(i)*gradx(3) + vniyL(i)*grady(3) + vnizL(i)*gradz(3))

       Enddo
c
c------ Corresponding to z direction momentum equation

       do i=1, inelc

       vfes(i+12+ppp) = vfes(i+12+ppp) + coef*(vnL(i)*u3t
     & + vnL(i)*fadv(4)
     & + vnixL(i)*fdif1(4) + vniyL(i)*fdif2(4) + vnizL(i)*fdif3(4))
     & + coef*(
     & vnixL(i)*fstabx(4) + vniyL(i)*fstaby(4) + vnizL(i)*fstabz(4))
     & + coef*dc*(
     & vnixL(i)*gradx(4) + vniyL(i)*grady(4) + vnizL(i)*gradz(4))

       Enddo









c
c------ Corresponding to energy equation

       do i=1, inelc

       vfes(i+18+ppp) = vfes(i+18+ppp) + coef*(vnL(i)*enrgt
     &  + vnL(i)*fadv(5)
     &  + vnixL(i)*fdif1(5) + vniyL(i)*fdif2(5) + vnizL(i)*fdif3(5))
```

```
     &  + coef*(
     &  vnixL(i)*fstabx(5) + vniyL(i)*fstaby(5) + vnizL(i)*fstabz(5))
     &  + coef*dc*(
     &  vnixL(i)*gradx(5) + vniyL(i)*grady(5) + vnizL(i)*gradz(5))

        Enddo
c*****************************************************************
c------ Corresponding kinematic turbulent viscosity equation
c
c------ tau for stabilization

        tausc= (2.d0*vv/hel)**2 + (4*vnut/(ren*hel*hel))**2
        tausc= 1/ sqrt(tausc)

c------ dct for chock capturing
c
        vnume = v1*vnutx + v2*vnuty + v3*vnutz
     &  - cb1*vort
     &  - ((1.d0+cb2)/(sigma*ren))*vnut*
     &    (vnutx*vnutx + vnuty*vnuty + vnutz*vnutz)

        vnume = (hel/2.d0)*abs(vnume)

        vdeno = sqrt( vnutx*vnutx + vnuty*vnuty + vnutz*vnutz )

        if(vdeno.le.eps1)then
        dct = 0.d0
        else
        dct = vpree(9)*(vnume/vdeno) + relax*(hel*vv/2.d0)
        endif
        if(dct.lt.eps1)dct= hel*vv/2.d0
        dct = 0.d0
c
c---------------------------------------------------------------
c
        do i=1, inelc
        vfes(i+24+ppp) = vfes(i+24+ppp) +
     &   coef*vnL(i)*(vnutt + v1*vnutx + v2*vnuty + v3*vnutz)
     &  + coef*(un/sigma)*(vnut/ren+dct)*
     &   (vnixL(i)*vnutx + vniyL(i)*vnuty + vnizL(i)*vnutz)
     &  - cb1*coef*vnL(i)*vort
     &  - coef*((1.0d0+cb2)/(sigma*ren))*vnL(i)*
     &       (vnutx*vnutx+vnuty*vnuty + vnutz*vnutz)*vnut

c------ Stabilisation

c       if(vpree(10).ne.zero)then
c
c       vfes(i+24+ppp) = vfes(i+24+ppp) +
c
c     &   coef*(vnixL(i)*v1+vniyL(i)*v2+vnizL(i)*v3)*
c     &   tausc*( v1*vnutx + v2*vnuty + v3*vnutz)
```

```
c
c     Endif

c------add GLS stabilization

      if(vpree(10).ne.zero)then
      vfes(i+24+ppp) = vfes(i+24+ppp) +
     &  coef*(vnixL(i)*v1+vniyL(i)*v2+vnizL(i)*v3
     &  -((1.d0+cb2)/(sigma*ren))*(vnutx*vnixL(i)+vnuty*vniyL(i)+
     &                          vnutz*vnizL(i)))*
     &  tausc*( v1*vnutx + v2*vnuty + v3*vnutz - cb1*vort
     &  -((1.d0+cb2)/(sigma*ren))*vnut*
     &      (vnutx*vnutx+vnuty*vnuty+vnutz*vnutz))

      Endif
      Enddo

c**********************************************************************
      ipp1= ipp1+ 24
      ipp2= ipp2+ 4*ppp

      Enddo
c================ Reorder the residual vectors into vector vfe ==========
c

      do i=1,ndleS
      iii = kpok(i)
      vfe(i) = - vfes(iii)
      enddo

      end
```

La routine pour résoudre l'équation de Helmoltz:

```
!=======================================================================
PROGRAM Helmoltz
!=======================================================================

IMPLICIT NONE

REAL*8  ::  Ut, TKP, PLN, hp,hpi, PI
REAL*8  ::  ZETA, ZINT, ERR, du, ll
REAL*8, PARAMETER  :: lambda2 = 1.0d0
INTEGER ::  I, J, K, S, L, NPOL, P, IPG

REAL*8,DIMENSION(:) , POINTER :: VZETA,VZEQ,
   &                             WPG,F,U,C,U1
REAL*8 :: sanal(0:49),VZ(0:49)
REAL*8,DIMENSION(:,:), POINTER :: A, M

PI = 4.0d0*DATAN(1.0d0)
!-------
OPEN(UNIT=11,FILE="exacte.txt",STATUS="UNKNOWN")
OPEN(UNIT=100,FILE="num.txt",STATUS="UNKNOWN")
OPEN(UNIT=200,FILE="err.txt",STATUS="UNKNOWN")
!-----------------------------------------------------------
P = 2
DO WHILE(P.LE.20)
IPG = P
!-----------------------------------------------------------
ALLOCATE(U(0:P),C(0:P),A(0:P,0:P),M(0:P,0:P),F(0:P))
ALLOCATE(VZETA(0:IPG),WPG(1:IPG))
ALLOCATE(VZEQ(0:P))
ALLOCATE(U1(1:IPG))
!-----------------------------------------------------------
!
!------ CALCUL DES POINTS D INTERPOLATION EQUIDISTANTS EN Z :

VZEQ = 0.0d0
DO k=0, P
VZEQ(k)= -1.0d0 + k*(2.0d0/REAL(P))
ENDDO
!-----------------------------------------------------------------------------
!  LES COORDONNES ET LES POIDS DES POINTS D INTEGRATION
!     (POINTS DE GAUSS)
!-----------------------------------------------------------------------------
VZETA=0.0d0
CALL GAUSS(IPG,VZETA)
WPG=0.0d0
CALL POIDS_GAUSS(WPG,VZETA,IPG)
!------------------------------------- LAGRANGE ---------------------------
!------- POINTS DE COLLOCATION DE GAUSS-LOBATTO-LEGENDRE ---------
!-----------------------------------------------------------------------------
```

```
!
!------ LA MATRICE M

M = 0.0d0
F = 0.0d0
C = 0.0d0
U = 0.0d0
!-------------
DO I=1, IPG
!-------------
ZETA= VZETA(I)
DO J=0, P
DO K=0, P
     M(J,K)= M(J,K) + WPG(I)* ( - hpi(ZETA,J,P,VZEQ)*hpi(ZETA,K,P,VZEQ)
    & - lambda2*hp(ZETA,VZEQ(J),P,VZEQ)*hp(ZETA,VZEQ(K),P,VZEQ) )
ENDDO
ENDDO
!


!------ LE VECTEUR F
DO J=0, P
  F(J)= F(J) - WPG(I)*(lambda2+PI**2)*dsin(PI*ZETA)
    &              *hp(ZETA,VZEQ(J),P,VZEQ)
ENDDO
!------
ENDDO     ! FIN DE LA BOUCLE SUR LES POINTS D INTEGRATION
!-----------------------------------------------------------
CALL LU(M(1:P-1,1:P-1),F(1:P-1),P-1,C(1:P-1))  ! Resolution
!-----------------------------------------------------------
!  LA SOLUTION EXACTE CALCULEE AUX POINTS DE COLLOCATION
!      DE GAUSS-LEGENDRE LOBATTO
!-----------------------------------------------------------
C(0)= 0.0d0
C(P)= 0.0d0
DO J=0, P
  ZINT= VZEQ(J)
  WRITE(100,'(1X,1E15.6,1X,1E15.6)')ZINT,C(J)
ENDDO
WRITE(100,'(//)')
!-----------------------------------------------------------
!  CALCUL DE L ERREUR
!-----------------------------------------------------------
U1=0.0d0
DO K= 1, IPG
  DO J= 0, P
  U1(K)= U1(K) + C(J)*hp(VZETA(K),VZEQ(J),P,VZEQ)
  ENDDO
ENDDO

ERR=0.0d0
DO K=1, IPG
```

```
  ERR = ERR + WPG(K)*(dsin(PI*VZETA(K)) - U1(K))**2
ENDDO
ERR = DSQRT(ERR)
WRITE(200,'(1X,I12,1X,1E15.6)')P,ERR
ERR=0.0d0

DEALLOCATE(U,C,A,M,F,VZEQ,VZETA,WPG,U1)
P=P+2
ENDDO
!-------------------------------------------------------------------
!  LA SOLUTION EXACTE AUX POINTS D INTERPOLATION
!-------------------------------------------------------------------
DO I = 0, 49
VZ(I)= -1.0d0 + I*(2.0d0/REAL(49))
ENDDO
DO I = 0, 49
  WRITE(11,'(1X,1E15.6,1X,1E15.6)')VZ(I),(-1.0d0+VZ(I)**2)
ENDDO
WRITE(11,'(//)')
!===================================================================
END     ! fin du programme
```

La routine pour résoudre l'équation de convection-diffusion :

```
PROGRAM stab

INTEGER P,P1
REAL*8,  PARAMETER :: x0 = -1.0d0,xf= 1.0d0
REAL*8   xi,a,h,PI,ZETA,coef,dx,hp,ERR,Ut
REAL*8   nuet,ploc,ksi,tau
INTEGER  nx,nelt,i,j,ii,i1,i2,i3,r,s,k,
   &         ie,ss1,ss2,LL1,LL2,ipp1,ipp2,IPG

REAL*8   vj,vj1,detj
REAL*8   VZ(0:49)
REAL*8,  DIMENSION( : ), POINTER :: XR,CR,vnig1,vnig2
REAL*8,  DIMENSION( : ), POINTER :: VZINT,VNS,VNG1,VNG2
REAL*8,  DIMENSION( : ), POINTER :: VZE,WP,VZETA,WPG,U1
REAL*8,  DIMENSION( : ), POINTER :: VNIL,VNIS, C,CRR,F
REAL*8,  DIMENSION(:,:), POINTER :: COORD,VCORE,vke,vkp
INTEGER, DIMENSION(:,:), POINTER :: CONNEC

c-------------------------------------------------------
REAL*8 ,PARAMETER ::  phi0=0.0d0, phi1=1.0d0
REAL*8 ,PARAMETER ::  nu=10e-2,vit=10.0d0
c-------------------------------------------------------
 PI = 4.0d0*DATAN(1.0d0)
print*," Entrer la taille de l élément "
read*, h
print*," Entrer lordre d interpolation "
read*, P
P1 = P+1
nx= int((xf-x0)/h)+1
nelt= nx-1
c-------------------------------------------------------
c print*,' Le nombre de noeuds total =  ', nx
c print*,' Le nombre d éléments total =  ', nelt
c-------------------------------------------------------
 ALLOCATE(VZINT(0:P),vkp(0:P,0:P),XR(1:P),CR(0:P))
 ALLOCATE(VNS(0:P),VNG1(0:P),VNG2(0:P),vnig1(0:P),vnig2(0:P))
 ALLOCATE(COORD(nx,2),CONNEC(nelt,3),VCORE(NELT,2))
 ALLOCATE(vke(0:nelt*P,0:nelt*P),C(0:nelt*P),CRR(0:nelt*P),F(0:nelt*P))
c-------------------------------------------------------
OPEN(UNIT=10,FILE="num1.txt",STATUS="UNKNOWN")
OPEN(UNIT=11,FILE="ana1.txt",STATUS="UNKNOWN")
```

```
c=======================================================================
c     CRÉATION DU MAILLAGE
c=======================================================================
c
c------ La table de coordonnées
c-----------------------------------

 DO i=1, nx
 COORD(i,1)= i
 COORD(i,2)= -1.0d0+h*(i-1)
 ENDDO

c print*,""
c print*,"      LA TABLE DES COORDONNÉES      "
c print*,""
c print*,(COORD(i,2),i=1,nx)
c print*,""


c------ La table des connéctivités
c-----------------------------------

 DO i=1, nelt

 CONNEC(i,1)= i
 CONNEC(i,2)= i
 CONNEC(i,3)= i + 1
 ENDDO


c print*,""
c print*,"     LA TABLE DES CONNÉCTIVITÉS      "
c print*,""
c print*,(CONNEC(i,2),CONNEC(i,3),i=1,nelt)
c print*,""

c=======================================================================
c     ÉVALUER LES FONCTIONS D INTERPOLATION & LEURS DÉRIVÉES
c     AUX POINTS D INTEGRATION
c=======================================================================
c
c-----------------------------------------------------------
 PRINT(//)'
 PRINT*,"CHOISIR LE TYPE D INTERPOLATION : "
 PRINT*,"    1 POUR EQD     "
 PRINT*,"    2 POUR LGL     "
 PRINT*,"    3 POUR LGC     "
 READ*,NPOL
 PRINT(//)'
c-----------------------------------------------------------
 GO TO (10,20,30),NPOL
c-----------------------------------------------------------
```

```
c------ CALCUL DES POINTS D INTERPOLATION EQUIDISTANTS EN Z :
c     &
c LES COORDONNES ET LES POIDS DES POINTS D INTEGRATION
c     (POINTS DE GAUSS)
c-------------------------------------------------------------------
10 continue
c-------------------------------------------------------------------
 PRINT*,"----- POINTS D INTERPOLATION EQUIDISTANTS ------"
 PRINT*,""
c-------------------------------------------------------------------
 VZINT = 0.0d0
 DO k=0, P
 VZINT(k)= -1.0d0 + k*(2.0d0/REAL(P))
 ENDDO
c-------------------------------------------------------------------
 goto 111
c-------------------------------------------------------------------
c------ CALCUL DES POINTS DE COLLOCATION DE
c     GAUSS-LEGENDRE-LOBATTO EN Z
c-------------------------------------------------------------------
20  continue
c-------------------------------------------------------------------
 PRINT*,"---------- POINTS DE COLLOCATION DE LGL ----------"
 PRINT*,""
c-------------------------------------------------------------------
 VZINT = 0.0d0
 CALL GLOB(P,VZINT)
c-------------------------------------------------------------------
 goto 111
c-------------------------------------------------------------------
c------ CALCUL DES POINTS DE COLLOCATION DE
c     GAUSS-LOBATTO-CHEBYCHEV EN Z
c-------------------------------------------------------------------
30  continue
c-------------------------------------------------------------------
 PRINT*,"--------- POINTS DE COLLOCATION DE LGC ----------"
 PRINT*,""
c-------------------------------------------------------------------
 DO k=0, P
 VZINT(k)= -COS( REAL(K)*PI / REAL(P))
 ENDDO
c
c
c-------------------------------------------------------
c  CALCUL DES COORDONNES ET LES POIDS DES
c  POINTS D INTEGRATION
c  (POINTS DE GAUSS-LEGENDRE-LOBATTO)
c-------------------------------------------------------
 ALLOCATE(VZETA(0:P),WPG(0:P))
 CALL GLOB(P,VZETA)
 CALL POIDS_GLOB(WPG,VZETA,P)
c-------------------------------------------------------
```

```
 vke  = 0.0d0
 F    = 0.0d0
 CRR  = 0.0d0
c-------------------------------------------------------------
c------ BOUCLE SUR LES ÉLÉMENTS
c-------------------------------------------------------------
 DO ie = 1, nelt
c-------------------------------------------------------------
 VCORE= 0.0d0
 VNIS = 0.0d0
 VNS  = 0.0d0
 VNG1 = 0.0d0
 VNG2 = 0.0d0
 vnig1= 0.0d0
 vnig2= 0.0d0
 vkp  = 0.0d0
c-------------------------------------------------------------
 peloc = (vit*h)/ (2.0d0*nu)
 if (peloc/3.0d0.lt.1.0d0) ksi = peloc/3.0d0
 if (peloc/3.0d0.ge.1.0d0) ksi = 1.0d0

c ksi=( (exp(peloc)+exp(-peloc))/(exp(peloc)-exp(-peloc)) )
c    &    -( 1.0d0/peloc )

 tau  = (h*ksi) / (2*vit)
c-------------------------------------------------------------
 VCORE(ie,1)= COORD(CONNEC(ie,2),2)
 VCORE(ie,2)= COORD(CONNEC(ie,3),2)
 vj   = h/2.0d0
 vj1  = 1.0d0/vj
 detj = vj
c-------------------------------------------------------------
c------ CALCUL DES FONCTIONS D' INTERPOLATION
c-------------------------------------------------------------
ALLOCATE(VNIL(0:6*P1-1),VNIS(0:3*P1*P1-1))
call shape (P,VNIL,VNIS,VZINT,VZETA,WPG)
c-------------------------------------------------------------
c------ BOUCLE SUR LES POINTS D INTEGRATION
c-------------------------------------------------------------
      ipp2 = 0
      DO ii = 0, P
c-------------------------------------------------------------
      ZETA= VZETA(ii)

c------ Les fonctions d interpolation et leurs dérivées spatiales

      DO j= 0, P
      VNS (j) = VNIS(    ipp2+j)
      VNG1(j) = VNIS(  P1+ipp2+j)
      VNG2(j) = VNIS(2*P1+ipp2+j)
      Enddo
```

```
c-------------------------------------------------------
DO j= 0, P
 vnig1 (j) = vng1(j)*vj1
 vnig2 (j) = vng2(j)*vj1*vj1
 Enddo
c-------------------------------------------------------
 coef = WPG(ii)*detj
c-------------------------------------------------------
c
c------ LA MATRICE MASSE ELEMENTAIRE

 DO i=0, P
 DO j=0, P

 vkp(i,j) = vkp(i,j)

c---- advection

    &   + coef * vit* VNS(i) * vnig1(j)

c---- diffusion

    &   + coef * nu * vnig1(i) * vnig1(j)
c
c---- stabilisation
c
c    &  + coef * tau*(vit*vnig1(i)) * (vit*vnig1(j) - nu*vnig2(j))


 ENDDO
 ENDDO
c------
 ipp2= ipp2+ 3*P1
c------
c
c-------------------------------------------------------
 ENDDO   ! FIN DE LA BOUCLE SUR LES POINTS D INTEGRATION
c-------------------------------------------------------
c
c-------------------------------------------------------
c   Assemblage
c-------------------------------------------------------
 IF (ie.eq.1) THEN

 do i=0,P
 do j=0,P

 vke(i,j)= vke(i,j) + vkp(i,j)

 enddo
 enddo
 ENDIF
```

```
c------
 IF (ie.gt.1) THEN

 do k = 1,nelt-1
 do i=0,P
 do j=0,P

 vke(P*k+i,P*k+j)=vke(P*k+i,P*k+j)+vkp(i,j)

 enddo
 enddo
 enddo

 ENDIF
c------
c
c-------------------------------------------------------
c TRANSFORMATION DE COORDONNÉES
c-------------------------------------------------------
c PRINT*,"les bords de l elem : ",VCORE(ie,1),VCORE(ie,2)

 DO i=1,P
   XR(i) = (VZINT(i)-VZINT(i-1))*DETJ
 enddo

 CR(0)= VCORE(ie,1)
 CR(P)= VCORE(ie,2)

 dx = 0.0d0

 DO i=0,P
 CR(i) = VCORE(ie,1) - dx
 dx = dx- XR(i+1)
 ENDDO
c------
 do i=0,P
 IF (i.ne.P) then
 CRR(i+(ie-1)*P) =  CRR(i+(ie-1)*P)+CR(i)
 ENDIF
 enddo
c-------------------------------------------------------
 ENDDO     ! FIN DE LA BOUCLE SUR LES ELEMENTS
c-------------------------------------------------------
c
 CRR(0)= x0
 CRR(nelt*P)= xf

c------ LE VECTEUR F

 DO K=0, P*nelt
 F(K) =  - vke(K,0)*PHI0 - vke(K,P*nelt)*PHI1
 ENDDO
```

```
c-------------------------------------------------------------
c      RESOLUTION
c-------------------------------------------------------------
 CALL LU(vke(1:P*nelt-1,1:P*nelt-1),F(1:P*nelt-1),
    &           P*nelt-1,C(1:P*nelt-1))
c-------------------------------------------------------------
c LA SOLUTION EXACTE CALCULEE AUX NOEUDS
c-------------------------------------------------------------
 C(0)    = phi0
 C(P*nelt)= phi1
c------
 DO J=0,P*nelt
   WRITE(10,'(1X,1E15.6,1X,1E15.6)')CRR(J)
   WRITE(20,'(1X,1E15.6,1X,1E15.6)')C(J)
 ENDDO
 WRITE(10,'(//)')
 WRITE(20,'(//)')
c-------------------------------------------------------------
c------ LA SOLUTION EXACTE AUX POINTS D INTERPOLATION
c-------------------------------------------------------------
 DO I = 0, 49
 VZ(I)= -1.0d0 + I*(2.0d0/REAL(49))
 ENDDO
c------
 Pe= vit*2.0d0/nu
 print*,"h =",h
 print*,"Pe = ",Pe
 print*,"peloc = ",peloc
c------
 c2 = (phi0-phi1)/(exp(-Pe/2.0d0)-exp(Pe/2.0d0))
 c1 =  phi0 - c2*exp(-Pe/2.0d0)
 print*,"c1 = ",c1
 print*,"c2 = ",c2
c------
 DO I = 0, 49
   WRITE(11,'(1X,1E15.6,1X,1E15.6)')VZ(I),
   &              c1+c2*exp(Pe*VZ(I)/2.0d0)
 ENDDO
 WRITE(11,'(//)')
```

```
c-------------------------------------------------------
c------ CALCUL DE L ERREUR
c-------------------------------------------------------
c OPEN(UNIT=555,FILE="error.txt",STATUS="UNKNOWN")
c
 IPG = P+1
 ALLOCATE(VZE(IPG),WP(IPG),U1(IPG))

 CALL GAUSS(IPG,VZE)
 CALL POIDS_GAUSS(WP,VZE,IPG)

 U1=0.0d0
 DO K= 1,IPG
   DO I= 1, nelt
   DO J= 0, P
   U1(K)= U1(K) + C(J+P*(I-1))*hp(VZE(K),VZINT(J),P,VZINT)
   ENDDO
   ENDDO
 ENDDO
!------
 ERR=0.0d0
 DO K= 1,IPG
 ERR = ERR + WP(K)*((c1+c2*dexp(Pe*VZE(K)/2.0d0)) - U1(K))**2
 ENDDO
 ERR = DSQRT(ERR)
c WRITE(555,'(1X,1I2,1X,1E15.6)')P,ERR
 PRINT*,"ERR = "
 PRINT*,ERR
 goto 11111
PRINT*,ERR
c=======================================================================
 END      ! FIN DU PROGRAMME
c=======================================================================
c
c
c
c=======================================================================
c     fonctions d interpolation
c=======================================================================
      subroutine shape(P,VNIL,VNIS,VZINT,VZETA,WPG)

 implicit none

 INTEGER P,P1
 REAL*8  VZINT(0:P),VZETA(0:P),WPG(0:P)
 REAL*8  VNIL(0:6*(P+1)-1),VNIS(0:3*(P+1)*(P+1)-1)

 integer i,j,k,jjj
  real*8 hp,hpi,hpii,a,b
 real*8 ZINT,ZETA

 P1= P+1
```

```
c-------------------------------------------------------------
c------ INTERPOLATION DE LAGRANGE P
c-------------------------------------------------------------
c------ boucle sur les points d integration ----------------

 DO j = 0, P
zeta = vzeta(j)

 DO k = 0, P
 ZINT = VZINT(k)

 VNIS(k+    3*P1*j) =  hp (ZETA,ZINT,P,VZINT)
 VNIS(k+  P1+3*P1*j) =  hpi (ZETA,k,P,VZINT)
 VNIS(k+2*P1+3*P1*j) =  hpii(ZETA,k,P,VZINT)

enddo
enddo
c
End   ! FIN DU PROGRAMME
```

La routine qui calcule les points de Gauss-Legendre-Lobatto :

```
c=======================================================================
      subroutine GLOB(N,xjac)
c=======================================================================
c
c  Cette suite de points est calculee par la methode Newton qui sont
c  initialises avec les points de Chebychev
c  Ce sont les zeros de (x*x-1)*la derivee de la niem fonction de legendre
c-----------------------------------------------------------------
c
      IMPLICIT NONE

 integer i,n,nh,it
 real*8 xjac(0:n)
      real*8 eps,pi,b,x,dx,xpdx,delx,fx,fxpdx,pdn,dfdx

      eps=1.0d-12

c------ RECHERCHE DES ZEROS DANS LA PREMIERE MOITIÉ [0,1]

      nh=n/2
      xjac(n) = 1.0d0
      pi=dacos(-1.0d0)
      b=-dcos((n-1)*pi/n)
      it = 0
      do 100 i=n-1,nh+1,-1

      x=b
 40   dx = -1.0d-8
      xpdx = x + dx
      fx = pdn(n,x)
      fxpdx = pdn(n,xpdx)
      dfdx= (fxpdx - fx)/dx
      delx = -fx/dfdx

      if (abs(fx).lt.eps) then
      go to 90
      else
      x = x+delx
      it = it+1
      go to 40
      endif

 90   xjac(i)=x
      b=-dcos((i-1)*pi/n)
100   continue
```

```
c------ UTILISATION DE LA SYMÉTRIE POUR L AUTRE MOITIÉ DES ZEROS

      xjac(0)=-1.0d0
      do 50 i=n-1,nh+1,-1
      xjac(n-i)=-xjac(i)
 50   continue
      if(n.ne.2*(n/2)) return
      xjac(nh)=0.0d0

      return
      end
```

La routine qui calcule les poids de Gauss-Legendre-Lobatto :

```
      SUBROUTINE POIDS_GLOB(WPG,VZETA,NPGZ)
c=======================================================================
IMPLICIT NONE
integer n,npgz,j
REAL*8 WPG(0:NPGZ)
REAL*8 VZETA(0:NPGZ)
REAL*8 x,wz,pln

DO j=0,NPGZ
 N=NPGZ
      X = VZETA(j)
 WPG(J)=1.0d0/(n*(n+1)*pln(n,x)*pln(n,x))
 ENDDO
 return
 END
```

La routine qui calcule les points de Gauss:

```
!=======================================================================
      subroutine gauss(n,vgauss)
!=======================================================================
!
! CE SOUS-PROG CALCULE LES POINTS DE GAUSS AVEC LA METHODE DE DICHOTOMIE
!-----------------------------------------------------------------------
      IMPLICIT NONE

      real*8 xjac(0:n),vgauss(1:n)
real*8 x,x1,x2,e,pi,a,b,dx,pr,pr1,pln
      integer i,n,nh,it
      e = 1.0d-13

if(n.eq.1) then
xjac(1)=0.0d0
return
endif

!
! RECHERCHE DES ZEROS DANS LA PREMIERE MOITIÉ [0,1]
!
nh=n/2
pi = dacos(-1.0d0)

      b=1.0d0
a=-dcos((n-1)*pi/n)

      do 100 i=n,nh+1,-1
      it=1
      dx = 1.0d-15
 30   x1 = (a+b)/2.0d0 - dx
  x2 = (a+b)/2.0d0 + dx

  pr = pln(n,x1)*pln(n,x2)

if (pr.gt.0) then
pr1 = pln(n,b)*pln(n,x2)
if (pr1.gt.0) then
b = x1
it = it +1
      go to 30
      else
a = x2
it = it+1
      go to 30
      endif
      else
x = (x1 + x2)/2.0d0
```

```
if (pln(n,x).lt.e) go to 90
      a = x1
      b = x2
      dx = dx/2.0d0
      it = it +1
     go to 30
      endif

 90   xjac(i)=x
      b=-dcos((i-1)*pi/n)
      a=-dcos((i-2)*pi/n)
 100   continue
!
! UTILISATION DE LA SYMÉTRIE POUR L AUTRE MOITIÉ DES ZEROS
!
do 50 i=n,nh,-1
xjac(n-i+1)=-xjac(i)
  vgauss(1:n)=xjac(1:n)
 50   continue
 201 return
end
```

La routine qui calcule les poids de Gauss:

```
!=======================================================================
      SUBROUTINE POIDS_GAUSS(WPG,VZETA,NPGZ)
!=======================================================================
IMPLICIT NONE

integer npgz,i,j,ij
real*8 wpg(npgZ)
real*8 vzeta(1:npgz)
real*8 x,rnum,rden,wz,pln,ver1

DO j=1,npgz

 x = vzeta(j)
 rnum = 2.0d0*(1.0d0-x*x)
 rden = npgz*npgz*Pln(npgz-1,x)*Pln(npgz-1,x)
 wz = rnum/rden
 wpg(J) =(1.0d0/2.0d0)*wz
 ver1=ver1+wpg(J)

ENDDO
return
END
```

La fonction qui calcule les polynômes de Legendre :

```
 FUNCTION PLN(N,X)
!-------------------------------------------------------
! PLN = NIÈME POLYNÔME DE LEGENDRE
!-------------------------------------------------------

IMPLICIT NONE
 real*8 x,plnm2, plnm1,pln
 integer n,k

      plnm2=1.0d0
      plnm1=x

      DO 10 k=2,n
      pln = (2.0d0*k-1)*x*plnm1/k-(k-1)*plnm2/k
      plnm2=plnm1
      plnm1=pln
 10   continue

      if (n.eq.0) pln = 1.0d0
      if (n.eq.1) pln = x

      return
      END
```

La fonction qui calcule les dérivées des polynômes de Legendre :

```
FUNCTION PDN(N,X)
!---------------------------------------------------------------------
! PDN = DERIVÉE DU NIÈME POLYNÔME DE LEGENDRE
!---------------------------------------------------------------------
IMPLICIT NONE
real*8 pdnm2, pdnm1,pln,x,pdn
integer k,n

      pdnm2=0.0d0
      pdnm1=1.0d0

      DO 10 k=2,n
      pdn = (2.0d0*k-1)*pln(k-1,x)/k
     &     + (2.0d0*k-1)*x*pdnm1/k-(k-1)*pdnm2/k
      pdnm2=pdnm1
      pdnm1=pdn
 10   continue

      if (n.eq.0) pdn = 0.0d0
      if (n.eq.1) pdn = 1.0d0
      return
      END
```

La fonction qui calcule les polynômes de Lagrange :

```
c=======================================================================
      FUNCTION hp(x,xj,n,vxi)
c
c   Calcul du polynôme de lagrange j au point x
c=======================================================================
      IMPLICIT NONE

 real*8 x,xi,xj,h1,h2,hp,p1,p2
integer i,j,n
real*8 vxi(0:n)

      IF (x.eq.xj) then
      hp = 1.0d0
      ELSE

      h1= 1.0d0
      h2= 1.0d0

      do i= 0, n
        if(xj.ne.vxi(i)) then
          h1= h1*(x-vxi(i))
          h2= h2*(xj-vxi(i))
        endif
      enddo
      hp= h1/h2
      ENDIF
      return
      END
```

La fonction qui calcule la dérivée du polynômede Lagrange :

```
c----- CALCUL DES DERIVEES DES POLYNOMES DE LAGRANGE
c     (Calcul de la dérivée de hp au point  x )
c=======================================================================
      FUNCTION hpi(x,i,n,vxi)
c=======================================================================

 IMPLICIT NONE

 real*8 x,h1,h2,hpi
 real*8 vxi(0:n),noni(n),tb1(n),tb2(n)
 Integer i,j,jj,k,n

tb2=vxi(i)
hpi = 0.0d0

c------
 k=0
 do j=0, n
 if(j.ne.i) then
 k=k+1
 noni(k)=vxi(j)
 endif
 enddo
c------
 do j=1, n
 tb1(j)=1
 do jj=1, n
 if(jj.ne.j) tb1(j)=tb1(j)*(x-noni(jj))
 enddo
 enddo
c------
 hpi=sum(tb1)/product(tb2-noni)

      return
      END
```

La fonction qui calcule la dérivée seconde du polynômede Lagrange :

```
c----- CALCUL DES DERIVEES SECONDES DES POLYNOMES DE LAGRANGE
c     (Calcul de la dérivée SECONDE de hp au point  x )
c=======================================================================
      FUNCTION hpii(x,i,n,vxi)
c=======================================================================
 IMPLICIT NONE

 real*8  x,hpii
 real*8  vxi(0:n),noni(n),tb2(n),tb1(n*n)
 Integer n,i,j,k,s,jj,kk

 If (n.eq.1) then
 hpii = 0.0d0
 return
 else

 tb2  = vxi(i)
 hpii = 0.0d0
c------
 k=0
 do j=0, n
 if(j.ne.i) then
 k=k+1
 noni(k)=vxi(j)
 endif
 enddo
c------
 s=0
 do kk=1,k
 do j =1, k

if(j.ne.kk) then
s=s+1

tb1(s)=1.0d0
do jj=1,k
if((jj.ne.j).and.(jj.ne.kk)) tb1(s)= tb1(s)*(x-noni(jj))
enddo

endif
enddo
Enddo
Endif
c
hpii= sum(tb1)/product(tb2-noni)
return
END
```

La fonction qui calcule les polynômes de Chebychev :

```
FUNCTION TKP(K,X)
!-----------------------------------------------------------------------
! CALCUL DU KIÈME POLYNÔME DE CHEBYCHEV EN X
!-----------------------------------------------------------------------

      IMPLICIT NONE

      INTEGER I,K
      REAL*8 x,TCHEB,TCHEB1,TCHEB2,TKP

      IF(K.EQ.0) TCHEB=1.0D0
      IF(K.EQ.1) TCHEB=X
      IF(K.LT.2) GO TO 20
      TCHEB2=1.0D0
      TCHEB1=X
      DO 10 I=2,K
      TCHEB=2.0D0*X*TCHEB1 - TCHEB2
      TCHEB2=TCHEB1
      TCHEB1=TCHEB
 10   CONTINUE
 20   CONTINUE
      TKP=TCHEB

      RETURN
      END
```

La fonction qui calcule les dérivées des polynômes de Chebychev :

```
 FUNCTION TKPP(N,X)
!-----------------------------------------------------------------------
! CALCUL DE LA DERIVÉE DU NIÈME POLYNÔME DE CHEBYCHEV EN X
!-----------------------------------------------------------------------

      IMPLICIT NONE

      INTEGER I,K,N,NPK
      REAL*8 x,SOM,TKP,TKPP

      SOM=0.0D0
      IF(N.EQ.0) GOTO 20
      I=N-(N/2)*2
      IF(I.EQ.1) SOM=SOM+0.50D0*TKP(0,X)
      IF(N.LT.2) GO TO 20
      DO 10 K=1,N-1
      NPK=N+K
      I=NPK-(NPK/2)*2
      SOM=SOM+I*TKP(K,X)
 10   CONTINUE
 20   CONTINUE
      TKPP=2.0D0*N*SOM

      RETURN
      END
```

La fonction Log-Lin et sa dérivée :

```
c=====================================================================
c=====================================================================
c CALCUL DES FONCTIONS D INTERPOLATION "LOG-LIN" SELON Z
c
c INPUT : uet = valeur de la vitesse de froettement
c         rnu = viscosité du fluide.
c         hp1 = la première transition du linéaire au logarithmique.
c         hp2 = le passage de la lere couche log a la seconde.
c         h = hauteur physique de la couche totale.
c         hplus = c la hauteur adimensionnée hplus = h*uet/rnu
c         CONSTANTES DU PROFILS : E1,E2,RK1,RK2
c
c OUTPUT : Rz c est la fonction d interpolation pour une valeur donnée
c          de zeta.
c---------------------------------------------------------------------
        Function Rz(zeta,uet,h,rnu)
c---------------------------------------------------------------------
c
      implicit none
c
      real*8 uet,rnu,hp1,hp2,hplus,zeta,unpzeta,Rz,h
      real*8 alpha,beta1,beta2,rk1,rk2,E1,E2,C,zeta1,zeta2
c
c------ CALCUL DES PARAMETRES
c
        hp1 = 5.0d0
        hp2 = 30.0d0
c
        hplus = h*uet/rnu
c
        zeta1 = 10/hplus - 1.0d0
        zeta2 = 60/hplus - 1.0d0
        rk1 = 0.20d0
        rk2 = 0.40d0
        E1 = 0.543d0
        E2 = 9.025d0
        alpha = uet*h/(2.d0*rnu)
        beta1 = E1*uet*h/(2.d0*rnu)
        beta2 = E2*uet*h/(2.d0*rnu)

c----- CALCUL DE LA CONSTANTE C SE;ON LA VALEUR DE hplus
c
        if(hplus.le.hp1) Then
        c = hplus
        else

        if(hplus.le.hp2) C = dlog(E1*hplus)/rk1
        if(hplus.ge.hp2) C = dlog(E2*hplus)/rk2
        endif
```

```
c------ DEFINITION DES FONCTIONS PAR TRANCHE
c
        unpzeta = 1.0d0 + zeta

        if(zeta.le.zeta1) then
        Rz = unpzeta*alpha/c
        else
        if(zeta.le.zeta2) Then
        Rz = dlog(beta1*unpzeta)/(rk1*c)
        Endif

        if(zeta.ge.zeta2) Then
        Rz = dlog(beta2*unpzeta)/(rk2*c)
        Endif

        Endif

        return
        end
c=====================================================================
c CALCUL DES DERIVÉES D INTERPOLATION "LOG-LIN" SELON Z
c
c INPUT : uet = valeur de la vitesse de froettement
c         rnu = viscosité du fluide.
c         hp1 = la première transition du linéaire au logarithmique.
c         hp2 = le passage de la lere couche log a la seconde.
c         h = hauteur physique de la couche totale.
c       hplus = c la hauteur adimensionnée hplus = h*uet/rnu
c       CONSTANTES DU PROFILS : E1,E2,RK1,RK2
c
c OUTPUT : Rpz c est laderivée d interpolation pour une valeur donnée
de zeta.
c---------------------------------------------------------------------
c
       Function Rpz(zeta,uet,h,rnu)

       implicit none
c
       real*8 uet,rnu,hp1,hp2,hplus,zeta,unpzeta,Rpz,h
       real*8 alpha,beta1,beta2,rk1,rk2,E1,E2,C,zeta1,zeta2
c
c------ DEFINITION DES PARAMÈTRES
c
        hplus = h*uet/rnu
        hp1 = 5.0d0
        hp2 = 30.0d0
        zeta1 = 10/hplus - 1.0d0
        zeta2 = 60/hplus - 1.0d0
        rk1 = 0.20d0
        rk2 = 0.40d0
c
        E1 = 0.543d0
        E2 = 9.025d0
```

```
C
        alpha = uet*h/(2.d0*rnu)
        beta1 = E1*uet*h/(2.d0*rnu)
        beta2 = E2*uet*h/(2.d0*rnu)

c------ CALCUL DE LA CONSTANTE C SELON LA VALEUR DE hplus
c

        if(hplus.le.hp1) Then

        c = hplus
c
        else
c
        if(hplus.le.hp2) then
        C = dlog(E1*hplus)/rk1
        endif
c
        if(hplus.ge.hp2) then
        C = dlog(E2*hplus)/rk2
        endif
c
        endif

c------ CALCUL DE LA DÉRIVÉE PAR TRANCHE
c
        unpzeta = 1.0d0 + zeta
c
        if(zeta.le.zeta1) Then

        Rpz = alpha/c

        Else

        if(zeta.le.zeta2) Then
        Rpz = 1.0d0/(unpzeta*rk1*c)
        Endif
C
        if(zeta.ge.zeta2) Then
        Rpz = 1.0d0/(unpzeta*rk2*c)
        Endif

        Endif
C
        return
        End
```

ANNEXE 2

METHODES DE RESOLUTION

MÉTHODES DE RESOLUTION

1 Introduction

La discrétisation spatio-temporelle et l'intégration numérique sur un élément, de la forme variationnelle présentée au chapitre 3, conduit à un système matriciel élémentaire, L'assemblage des contributions élémentaires donne un système matriciel global qui s'écrit sous la forme:

$$[\mathbf{M}]\left\{\frac{\partial \mathbf{V}}{\partial t}\right\} + [\mathbf{k}(\mathbf{V})]\{\mathbf{V}\} = \{\mathfrak{I}\}$$

ou encore :

$$[\mathbf{K}(\mathbf{V})]\{\mathbf{V}\} = \{\mathfrak{I}\} \qquad (1)$$

où

$[\mathbf{K}(\mathbf{V})]$: La matrice globale incluant la matrice rigidité et la matrice masse globales.

$\{\mathbf{V}\}$: Le vecteur des degrés de liberté global et

$\{\mathfrak{I}\}$: Le vecteur source global.

Ce système est fortement non-linéaire. Certains de ces non-linéarités sont dus essentiellement aux termes de convection et à la compressibilité du fluide. D'autres sont dus aux termes additionnels engendrés par les techniques de stabilisation (voir chapitre 7). La matrice $[\mathbf{K}(\mathbf{V})]$ se trouve donc une matrice creuse, non-linéaire, non symétrique et de grande taille. Il serait donc plus avantageux d'utiliser une méthode itérative plutôt qu'une méthode directe qui nécessite un stockage de la factorisation de $[\mathbf{K}(\mathbf{V})]$. Résoudre le système non-linéaire (4.1) revient donc à chercher itérativement un vecteur $\{\mathbf{V}\}$tel que le résidu :

$$\{\mathbf{R}(\mathbf{V})\}=\{\mathfrak{I}\}-[\mathbf{K}(\mathbf{V})]\{\mathbf{V}\} \tag{2}$$

soit nul au sens numérique.

Soit à l'itération n, le résidu non nul :

$$\{\mathbf{R}(\mathbf{V}n)\}=\{\mathfrak{I}\}-[\mathbf{K}(\mathbf{V}n)]\{\mathbf{V}n\}\neq 0 \tag{3}$$

Le problème revient donc à trouver $\mathbf{V}_{n+1}=\mathbf{V}_n+\delta\mathbf{V}$ tel que le résidu soit nul à l'itération n+1 :

$$\{\mathbf{R}(\mathbf{V}n+1)\}=\{\mathbf{R}(\mathbf{V}n+1+\delta\mathbf{V}n)\}\approx 0 \tag{4}$$

Dans ce qui suit, on exposera deux méthodes itératives : la méthode de gradient conjugué [23] et la méthode **GMRES** « **G**eneralized **M**inimal **RES**idual » développée par Saad et Shultz [24].

A. Soulaïmani et M. Fortin [5], ont employé, avec succès, l'algorithme **GMRES** pour résoudre les problèmes de la mécanique des fluides pour les écoulements compressibles.

2 Méthode de gradient conjugué

La méthode de gradient conjugué est une méthode itérative pour la résolution de systèmes linéaires de la forme :

$$\mathbf{A}\,x=\mathbf{b} \tag{5}$$

en supposant que la matrice $\mathbf{A}=\mathbf{A}^{\mathbf{T}}$ est une matrice de taille N×N définie positive. On remarquera que la résolution de l'équation (3) correspond à la minimisation de la forme quadratique :

$$Q(x) = \frac{1}{2} xT\mathbf{A}\,x - xT\mathbf{b} \qquad (6)$$

On peut facilement vérifier que le gradient au point x vaut précisément $\mathbf{A}\,x - \mathbf{b}$ qui est égale au résidu r en ce point.

L'algorithme de gradient conjugué peut être présenté comme suit :

1- On se donne un vecteur initial x_0 qui peut être quelconque.

2- On calcule le gradient de $Q(x_0)$ en ce point, soit : $r^0 = \mathbf{A}x_0 - \mathbf{b}$

3- On initialise un second vecteur qui est la direction d recherche: $d^0 = -r^0$

4- Mise à jour la solution estimée x : $x^{k+1} = x^k + a_k\, d^k$

a_k étant le pas optimal de recherche à l'itération k calculé par la relation :

$$ak = \frac{rk.dk}{dk.\mathbf{A}\,dk}$$

5- Mise à jour de r : $r^{k+1} = r^k - a_k\, d^k$

6- Mise à jour de d : $d^{k+1} = -r^{k+1} + \beta_k\, d^k$

β est choisi de manière à assurer l'orthogonalité entre deux directions de recherche successives;

$$d^i.\mathbf{A}\,d^j = 0 \qquad \text{si } i \neq j$$

Ainsi, on a

$$d^{k+1}.\mathbf{A}d^{k}=0$$

$$\Leftrightarrow \quad -r^{k+1}.\mathbf{A}d^{k}+\beta_k d^{k}.\mathbf{A}d^{k}=0$$

$$\Leftrightarrow \qquad \beta_k=\frac{r^{k+1}.\mathbf{A}d^{k}}{d^{k}.\mathbf{A}d^{k}}$$

L'algorithme de résolution de gradient conjugué est résumé comme suit :

Algorithme de gradient conjugué

1- Début

Choisir une solution approchée x_0

Choisir la précision ε

2- $d^0 = -r^0$ (résidu initial)

2- Itération: Pour $k=0,\ldots$ jusqu'à convergence

- Calculer $\alpha_k = -\dfrac{r^k.d^k}{d^k.\mathbf{A}d^k}$

- Mise à jour de x : $x^{k+1} = x^k + \alpha_k\, d^k$

- Calculer $\beta_k = \dfrac{r^{k+1}.\mathbf{A}d^k}{d^k.\mathbf{A}d^k}$

- Calculer $r^{k+1} = r^k - \alpha_k\, d^k$

- Calculer $d^{k+1} = -r^{k+1} + \beta_k\, d^k$

- Redémarrer :

Si le critère de convergence est satisfait : $\left\| r^{k+1} \right\| \le \varepsilon \left\| r^0 \right\|$, on s'arrête et

x^{k+1} est la solution approchée du problème pour la précision ε donnée.

Sinon, continuer la boucle au point 2.

Remarques :

- On n'a pas besoin de connaître explicitement la matrice A ou la stocker, mais seulement le produit scalaire de A par un vecteur.

- Revenons au calcul de d^{k+1} :

$$\begin{aligned} d^{k+1} &= -r^{k+1} + \beta_k d^k \\ &= -r^{k+1} + \beta_k \left(-r^k + \beta_{k-1} d^{k-1} \right) \\ &= \sum_{i=0}^{k+1} \gamma_i r^i \end{aligned}$$

Les directions de recherche sont des combinaisons linéaires de $\left\{ r^0, r^1, ..., r^{k+1} \right\}$

- D'autre part :

$$\begin{aligned} r^{k+1} &= r^k + \alpha_k d^k \\ &= r^{k-1} + \alpha_k \mathbf{A} d^k \\ &= r^{k-1} + \alpha_{k-1} \mathbf{A} d^{k-1} + r^{k-1} + \alpha_k \mathbf{A} d^k \\ &= \sum_{i=0}^{k} \delta_i \mathbf{A}^i r^0 \end{aligned}$$

Les résidus successifs sont des combinaisons linéaires de $\left\{ r^0, \mathbf{A}r^0, ..., \mathbf{A}^k r^0 \right\}$

L'espace $\left\{ r^0, \mathbf{A}r^0, ..., \mathbf{A}^k r^0 \right\}$ est appelé sous-espace de Krylov.

Théorèmes :

Du point de vue théorique, la solution exacte est obtenue après un maximum de N itérations si N est la dimension du problème. De plus, on peut montrer que le nombre m d'itérations nécessaire à la résolution de $\mathbf{A}x = \mathbf{b}$ avec une réduction de l'erreur initiale d'un taux ε dépend du nombre spectral de A et est donné par :

$$m \geq \frac{1}{2}\sqrt{\chi(\mathbf{A})}\ln\left(\frac{2}{\varepsilon}\right)$$

où $\chi(\mathbf{A})$ étant le nombre spectral de la matrice A donnée par (1.35).

Pour améliorer la convergence, au lieu de résoudre $\mathbf{A}x = \mathbf{b}$ avec l'algorithme du gradient conjugué, il convient de résoudre $\mathbf{C}^{-1}\mathbf{A}x = \mathbf{C}^{-1}\mathbf{b}$, où C est une matrice symétrique définie positive choisie de sorte que $\chi(\mathbf{C}^{-1}\mathbf{A})$ soit nettement plus petit que $\chi(\mathbf{A})$. La matrice C s'appelle le préconditionneur de A.

2.2 Algorithme GMRES préconditionné

L'objectif est de résoudre par une méthode itérative un système linéaire écrit dans sa forme générale:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{7}$$

où x est le vecteur inconnu, A est une matrice non-symétrique de grande taille, et $\mathbf{b}$ le second membre.

Pour débuter, on décrira le matériel de base employé dans la construction de l'algorithme **GMRES**. Ensuite, on développera les deux stratégies de

préconditionnement : la première qui sera appelée préconditionnement diagonal, où le système est transformé en diagonale, et la deuxième qui elle sera appelée préconditionnement bloc-diagonal. On finira cette section par un organigramme détaillé de la procédure employée dans le programme informatique (**GMRES** préconditionné).

2.2.1 Algorithme GMRES

Considérons une solution approximée de la forme

$$x = x_0 + z \tag{8}$$

où x_0 est une solution initiale, et z est un élément du sous espace de Krylov K_k , de dimension k, associé au résidu $r_0 = \mathbf{b} - \mathbf{A}x_0$ et la matrice **A**:

$$K_k = span\left\{r^0, \mathbf{A}r^0, ..., \mathbf{A}^{k-1}r^0\right\} \tag{9}$$

L'algorithme **GMRES** emploie une base orthonormée de l'espace de Krylov obtenue par la méthode d'orthogonalisation de Gram-Schmidt modifié.

L'algorithme **GMRES** détermine z de façon telle que la fonctionnelle $\|\mathbf{b} - \mathbf{A}(x_0 - z)\|$ soit minimisée.

On construit, via le procédé d'Arnoldi, une base orthonormée $\mathbf{U}_k$, de l'espace de Krylov de dimension k. Le procédé d'Arnoldi est basé sur la méthode d'orthogonalisation de Gram-Schmidt, qui peut être présenté comme suit:

Procédé d'orthogonolisation de Gram-Schmidt

$$u_1 = \frac{r_0}{\| r_0 \|}$$

Pour $i = 1, k$

$$u_{i+1} = \mathbf{A}\, u_i$$

Pour $j = 1, i$

$$\beta_{i+1,j} = \left(u_{i+1}, u_j\right)$$

$$u_{i+1} \leftarrow u_{i+1} - \beta_{i+1,j}\, u_j$$

$$u_{i+1} = \frac{u_{i+1}}{\| u_{i+1} \|}$$

Fin de la boucle sur j

Fin de la boucle sur i

Où u_k est le $k^{\text{ème}}$ vecteur de la base orthogonale de l'espace de Krylov:

$$\mathbf{U_k} = \{u_1, u_2, ..., u_k\} \tag{10}$$

Le fait que $\mathbf{U_k}$ est une base orthonormée de l'espace de Krylov, on peut montrer que la relation suivante est satisfaite:

$$\mathbf{A\,U_k} = \mathbf{U_{k+1}\,H_k} \tag{11}$$

où $\mathbf{H_k}$ est la matrice de Hessenberg supérieure de dimension $(k+1)\times k$

$$\mathbf{H}_k = \begin{bmatrix} \beta_{2,1} & \beta_{3,1} & \cdots & \beta_{k,1} & \beta_{k+1,1} \\ \|u_2\| & \beta_{3,2} & \cdots & \beta_{k,2} & \beta_{k+1,2} \\ 0 & \|u_3\| & \cdots & : & : \\ : & 0 & & : & : \\ : & : & \cdots & \beta_{k,k-1} & : \\ 0 & 0 & \cdots & \|u_k\| & \beta_{k+1,k} \\ 0 & 0 & \cdots & 0 & \|u_{k+1}\| \end{bmatrix} \tag{12}$$

Si on écrit z comme combinaison linéaire des u_j :

$$z = \sum_{j=1}^{k} y_i \, u_j \qquad y \in \mathbf{R}^k \tag{13}$$

et

$$r_0 = U_{k+1} \, e \tag{14}$$

où $e = \{\| r_0 \|, 0, ..., 0\}^T$, possédant k+1 éléments.

Avec ces notations on a :

$$\| \mathbf{b} - \mathbf{A}(x_0 - z) \| = \left\| r_0 - \mathbf{A}\left(\sum_{j=1}^{k} y_j u_j \right) \right\| = \| r_0 - \mathbf{A}\,\mathbf{U}_k y \| = \| \mathbf{U}_{k+1}(e - \mathbf{H}_k y) \|$$

$$= \| e - \mathbf{H}_k y \| \tag{15}$$

Donc, le problème de minimisation dans l'espace de Krylov K_k peut être écrit comme suit:

Trouver $y \in \mathbf{R}^k$ tel que :

$$\min_{z \in k} \| \mathbf{b} - \mathbf{A}(x_0 - z) \| = \min_{y \in \mathbf{R}^k} \| e - \mathbf{H}_k y \| \qquad (16)$$

2.2.2 Préconditionnement

Pour augmenter la performance de la résolution itérative (taux de convergence), on utilise un préconditionneur de type ILUT, Pour l'organigramme de l'algorithme GMRES et le préconditionneur , voir [6].