Méthodes Numériques Ecoulements Incompressibles

Résolution de l'équation de Stokes en schéma MAC, par la méthode d'Uzawa et solveur itératif, en Fortran90

JACQUET Pierre pierre.jac.23@gmail.com
RAVEL Guillaume gravel@enseirb-matmeca.fr

Enseignant : Yves Coudière

17 janvier 2017

1 Modélisation mathématique

1.1 Problème de Stokes

L'objectif de ce projet est de se familiariser avec la résolution du problème de Stokes instationnaire (1) par la méthode d'Uzawa.

$$\begin{cases} \partial_t u - \nu \nabla u + \nabla p = f sur \Omega \\ -\nabla . u = 0 \\ u_{|\Gamma_0} = g \\ u_{|\Gamma_1} = h \end{cases} \tag{1}$$

Le domaine de calcul représenté sur la Figure 1 est un pavé dont les frontières horizontales et verticales ont été distinguées.

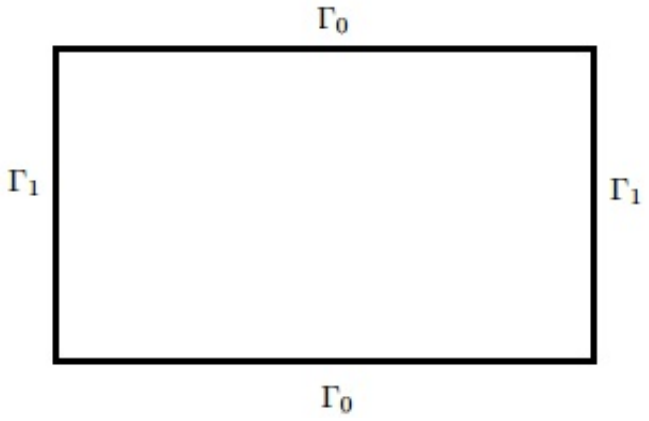

FIGURE 1 – Domaine Ω

Cette équation est issue d'une version simplifiée de l'équation de Navier-Stokes dont les termes inertiels ont été négligés. L'équation de Stokes décrit l'écoulement d'un fluide newtonien incompressible.

1.2 Schéma MAC

En discrétisant par des différences finies l'espace suivant $N_x \times N_y$ mailles, l'équation de Stokes instationnaire (1) devient :

$$\begin{cases} \frac{u_{i,j}^{1,n+1} - u_{i,j}^{1,n}}{dt} - \nu\left(\frac{u_{i+1,j}^{1,n+1} + u_{i-1,j}^{1,n+1} - 2u_{i,j}^{1,n+1}}{dx^2} + \frac{u_{i,j+1}^{1,n+1} + u_{i,j-1}^{1,n+1} - 2u_{i,j}^{1,n+1}}{dy^2}\right) + \frac{p_{i+\frac{1}{2},j+\frac{1}{2}}^{n+1} + p_{i+\frac{1}{2},j-\frac{1}{2}}^{n+1} - p_{i-\frac{1}{2},j+\frac{1}{2}}^{n+1} - p_{i-\frac{1}{2},j-\frac{1}{2}}^{n+1}}{2dx} = f_{i,j}^{1,n} \\ \frac{u_{i,j}^{2,n+1} - u_{i,j}^{2,n}}{dt} - \nu\left(\frac{u_{i+1,j}^{2,n+1} + u_{i-1,j}^{2,n+1} - 2u_{i,j}^{2,n+1}}{dx^2} + \frac{u_{i,j+1}^{2,n+1} + u_{i,j-1}^{2,n+1} - 2u_{i,j}^{2,n+1}}{dy^2}\right) + \frac{p_{i+\frac{1}{2},j+\frac{1}{2}}^{n+1} + p_{i-\frac{1}{2},j+\frac{1}{2}}^{n+1} - p_{i-\frac{1}{2},j-\frac{1}{2}}^{n+1} - p_{i+\frac{1}{2},j-\frac{1}{2}}^{n+1}}{2dy} = f_{i,j}^{2,n} \\ -\frac{u_{i+1,j+1}^{1,n+1} + u_{i+1,j}^{1,n+1} - u_{i,j+1}^{1,n+1} - u_{i,j}^{1,n+1}}{2dx} - \frac{u_{i+1,j+1}^{2,n+1} + u_{i,j+1}^{2,n+1} - u_{i+1,j}^{2,n+1} - u_{i,j}^{2,n+1}}{2dy} = 0 \end{cases} \tag{2}$$

Cette discrétisation est issue du schéma MAC "Marker And Cell". Afin d'obtenir un système avec autant d'inconnues que d'équations, les pressions sont placées au centre des mailles alors que les vitesses sont situées sur les noeuds. Ainsi, le gradient de pression est calculé aux noeuds tandis que la divergence de la vitesse est calculée au centre des mailles comme illustré par la figure 2. De cette manière, nous n'avons pas besoin de la pression aux bords du domaine, qui est inconnue. Quant à la vitesse sur le bord du domaine, on utilise des conditions de Dirichlet non homogènes.

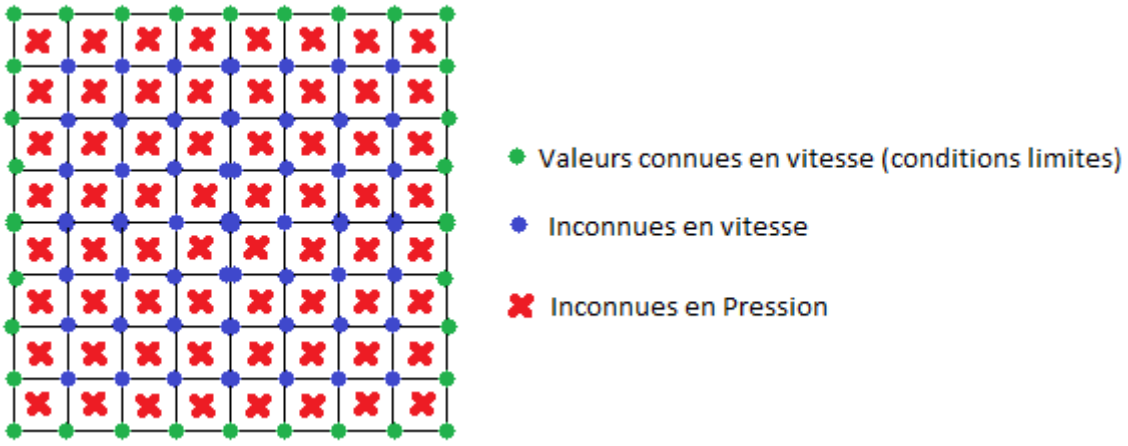

FIGURE 2 – Illustration du maillage - schéma MAC

En plaçant ligne par ligne les $u_{i,j}^{1,n+1}$,$u_{i,j}^{2,n+1}$,$p_{i,j}^{n+1}$,$f_{i,j}^{1,n}$,$f_{i,j}^{2,n}$ dans des vecteurs colonnes U_x^{n+1},U_y^{n+1},P^{n+1},F_x^n,F_y^n respectivement, on obtient :

-U_x^n vecteur des composantes des vitesses selon x de taille $(N_x-1)(N_y-1)$
-U_y^n vecteur des composantes des vitesses selon y de taille $(N_x-1)(N_y-1)$
-P^n vecteur des pressions de taille N_xN_y
-F_x^n termes sources selon x de taille $(N_x-1)(N_y-1)$
-F_y^n termes sources selon y de taille $(N_x-1)(N_y-1)$

D'où le système matriciel suivant :

$$\begin{pmatrix} A & 0 & B_1^T \\ 0 & A & B_2^T \\ B1 & B2 & 0 \end{pmatrix} . \begin{pmatrix} U_x^{n+1} \\ U_y^{n+1} \\ P^{n+1} \end{pmatrix} = \begin{pmatrix} F_x^n \\ F_y^n \\ 0 \end{pmatrix} + \frac{1}{dt} \begin{pmatrix} U_x^n \\ U_y^n \\ 0 \end{pmatrix} + ConditionsLimites \quad (3)$$

Avec :

$$A = \begin{pmatrix} B & C & 0 & \dots & 0 \\ C & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & C \\ 0 & \dots & 0 & C & B \end{pmatrix}$$ où B et C sont les matrices de taille $(N_x - 1) \times (N_y - 1)$ définies comme suit :

$$B = \begin{pmatrix} \alpha & \beta & 0 & \dots & 0 \\ \beta & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta \\ 0 & \dots & 0 & \beta & \alpha \end{pmatrix} \text{ et } C = \begin{pmatrix} \gamma & 0 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 0 & \gamma \end{pmatrix} \text{ avec : } \begin{cases} \alpha = 2\nu(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}) + \frac{1}{\Delta t} \\ \beta = \frac{-\nu}{\Delta x^2} \\ \gamma = \frac{-\nu}{\Delta y^2} \end{cases}$$

$$B1 = \begin{pmatrix} S & 0 & 0 \\ S & \ddots & 0 \\ 0 & \ddots & S \\ 0 & 0 & S \end{pmatrix} \text{ de taille } N_x N_y \times (N_x - 1)(N_y - 1)$$

$$\text{où } S = \frac{1}{2dx} \begin{pmatrix} -1 & 0 & 0 \\ 1 & \ddots & 0 \\ 0 & \ddots & -1 \\ 0 & 0 & 1 \end{pmatrix} \text{ de taille } N_x \times (N_x - 1)$$

$$B2 = \begin{pmatrix} -R & 0 & 0 \\ R & \ddots & 0 \\ 0 & \ddots & -R \\ 0 & 0 & R \end{pmatrix} \text{ de taille } N_x N_y \times (N_x - 1)(N_y - 1)$$

$$\text{où } R = \frac{1}{2dy} \begin{pmatrix} 1 & 0 & 0 \\ 1 & \ddots & 0 \\ 0 & \ddots & 1 \\ 0 & 0 & 1 \end{pmatrix} \text{ de taille } N_x \times (N_x - 1)$$

2 Résolution numérique

2.1 Code

Cette résolution est codée en Fortran90 et s'organise de la manière suivante :
-un programme principal "main.f90" qui appelle successivement les fonctions et les subroutines des autres modules pour résoudre le problème de Stokes

-un module "Tools.f90" réunissant toute les variables du problème discrétisé et les fonctions annexes à la résolution (lecture de fichier/ écriture de sortie / norme / produit scalaire)
-un module "Solver.f90" qui contient tout les produits matriciels essentiels à la résolution sous forme de fonctions ainsi que les algorithmes du gradient conjugué et celui d'Uzawa.
-un module "Fonctions.f90" qui calcule les conditions limites en fonction du choix de l'utilisateur
-un fichier "param.txt" modifiable à la volonté de l'utilisateur afin que la simulation corresponde aux critères voulus.

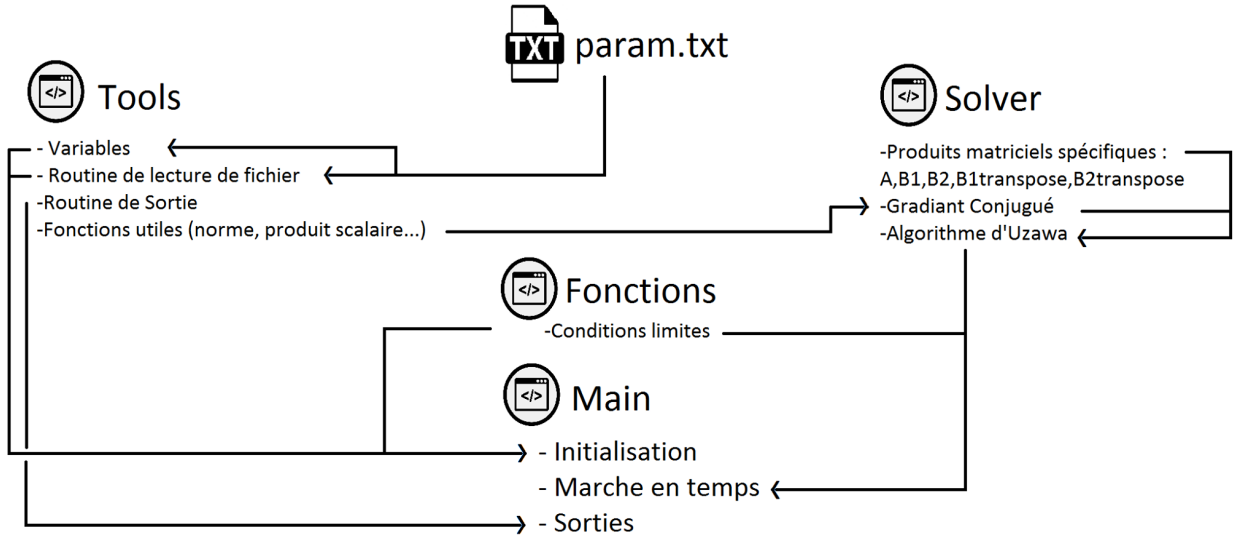

FIGURE 3 – Organisation du code

2.2 Algorithme d'Uzawa

La solution U de (3) réalise aussi le minimum de $J(U) = \frac{1}{2}U^T AU - U^T F$ sous la contrainte $BU = 0$.
Soit P un multiplicateur de Lagrange, qui vérifie : $< BU, P >= 0$, soit $< U, B^T P >= 0$.
U réalisant le minimum de J convexe, on a alors la relation : $\nabla J = -B^T P$.
Par conséquent, le système (3) peut être vu plus simplement comme :

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \cdot \begin{pmatrix} U \\ P \end{pmatrix} = \begin{pmatrix} F \\ 0 \end{pmatrix} \tag{4}$$

Dans ce cadre, la pression P est alors vue comme un multiplicateur de Lagrange.
Lors d'un pas de temps, le système (4) peut alors être résolu avec l'algorithme d'Uzawa :

— Initialisation : on fixe μ et P^0
— Tant que : $\|P^{n+1} - P^n\| = \mu\|BU^n\| > \epsilon$ Faire

— Résolution de $AU^n = F - B^T P^n$ par l'algorithme du gradient conjugué
— Mise à jour de P : $P^{n+1} = P^n + \mu B U^n$

3 Cas tests

3.1 Résolution du Laplacien

Dans un premier temps, la résolution d'un problème de Poisson (5) a été implémentée afin de la valider sur des cas tests triviaux.

$$\begin{cases} -\Delta u = f \text{ sur } \Omega \\ u_{|\Gamma_0} = g \\ u_{|\Gamma_1} = h \end{cases} \tag{5}$$

Ce problème lorsque discrétisé, se rapporte à la résolution du système linéaire $Au = b$ par l'algorithme du gradient conjugué dans lequel le produit matriciel Ax n'est pas calculé directement afin de profiter du caractère fortement creux de la matrice du Laplacien : une fonction le remplacera et minimisera le nombre d'opérations.

Pour valider la résolution, on propose les deux cas tests suivants :

Cas test 1 :

$$\begin{cases} u(x,y) = x(1-x)y(1-y) \\ f(x,y) = 2(y - y^2 + x - x^2) \\ g = h = 0 \end{cases} \tag{6}$$

Cas test 2 :

$$\begin{cases} u(x,y) = sin(x) + cos(y) \\ f(x,y) = sin(x) + cos(y) \\ g = h = f \end{cases} \tag{7}$$

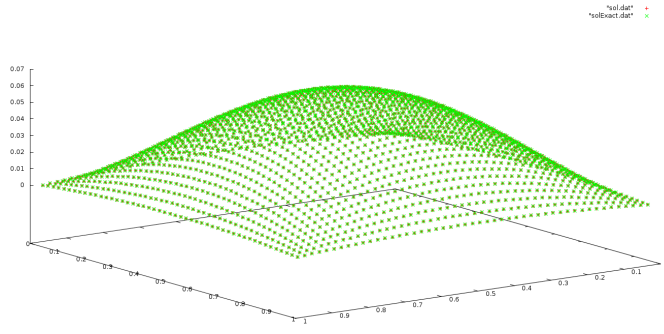

FIGURE 4 – Visualisation des solutions numérique et exacte - Cas test 1

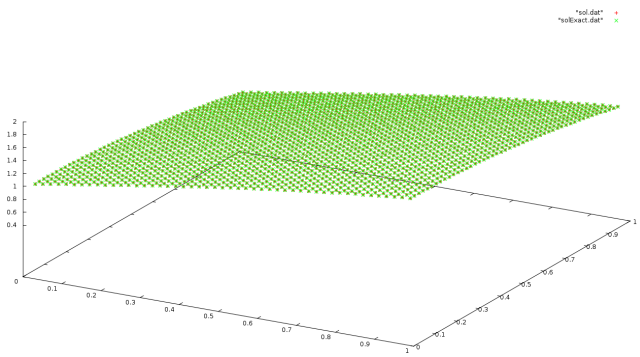

FIGURE 5 – Visualisation des solutions numérique et exacte - Cas test 2

Les Figure 6 et Figure 7 représentant l'évolution de l'erreur suivant l'espace, démontre que le schéma implémenté est bien d'ordre deux comme attendu par le schéma différences finies de la discrétisation du Laplacien.

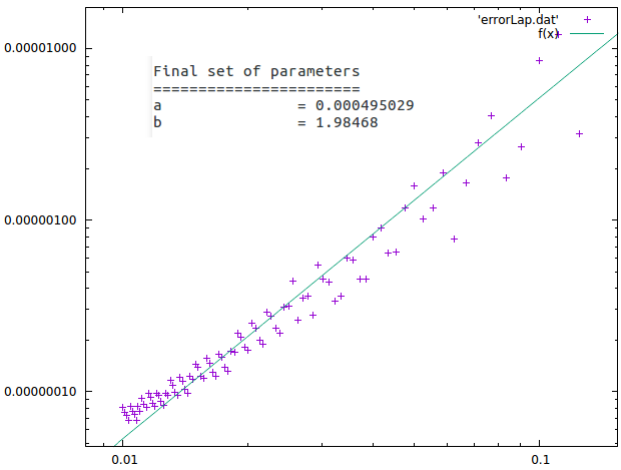

FIGURE 6 – Comparaison de l'erreur relative en norme 2 suivant dx avec $f(x) = ax^b$ - Cas test 1 (échelle log-log)

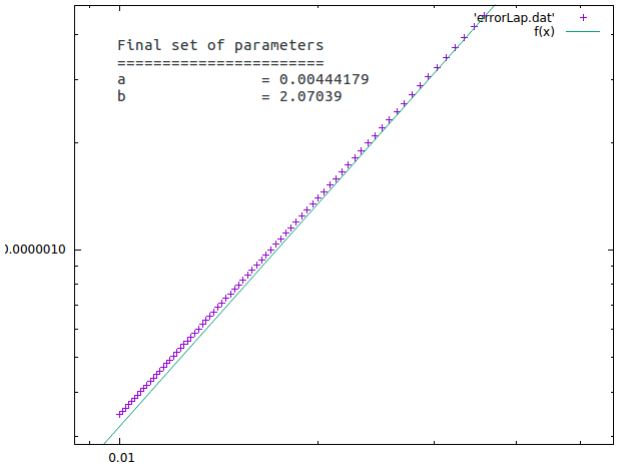

FIGURE 7 – Comparaison de l'erreur relative en norme 2 suivant dx avec $f(x) = ax^b$ - Cas test 2 (échelle log-log)

3.2 Cas Test 0

La résolution du problème de Stokes par l'algorithme d'Uzawa doit maintenant être validée à travers différents cas tests.
Prenons dans un premier temps, les cas tests (0a) (8) et (0b) (9)

Cas test 0a :

$$\begin{cases} U = (u_1, u_2) \text{ avec } u_1, u_2 \text{ constantes} \\ F = (0, 0) \\ G = H = U \\ P = c \end{cases} \tag{8}$$

Cas test 0b :

$$\begin{cases} U = (u_1, u_2) \text{ avec } u_1, u_2 \text{ constantes} \\ F = (0, -g) \\ G = H = U \\ P = -gy + c \end{cases} \tag{9}$$

Pour ces cas tests, la vitesse doit être constante sur tout le domaine.
En revanche, la pression diffère entre les deux cas : constante pour le cas $(0a)$, linéaire en y pour le cas $(0b)$.

Cas test 0a :

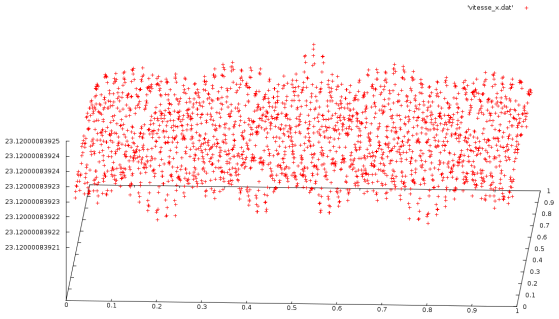

FIGURE 8 – Vitesse selon x

Cas test 0b :

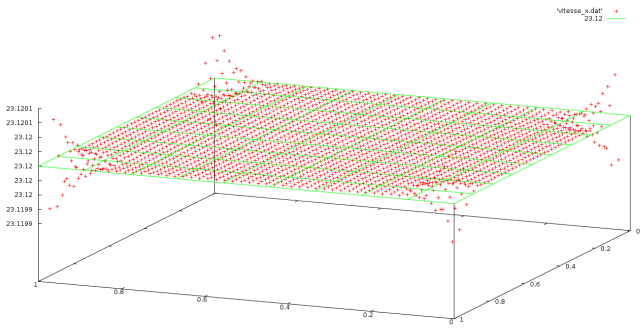

FIGURE 9 – Vitesse selon x

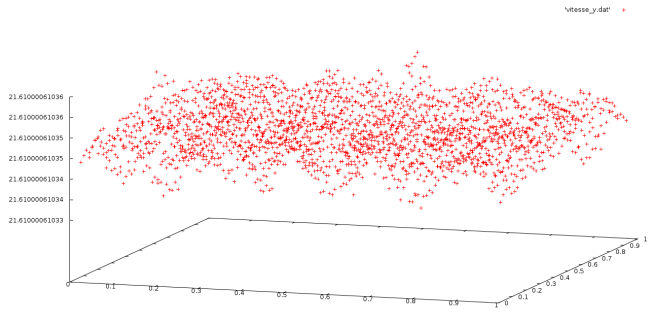

FIGURE 10 – Vitesse selon y

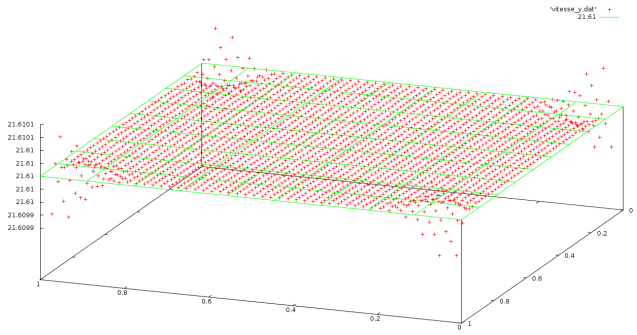

FIGURE 11 – Vitesse selon y

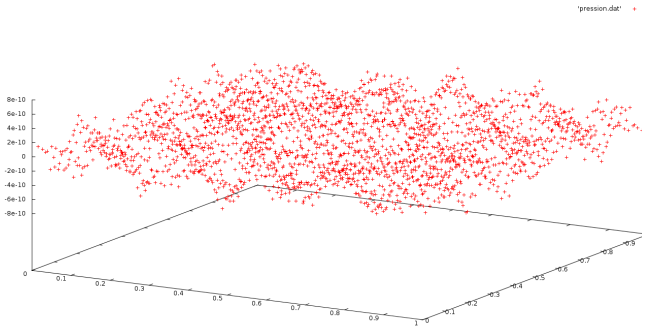

FIGURE 12 – Pression

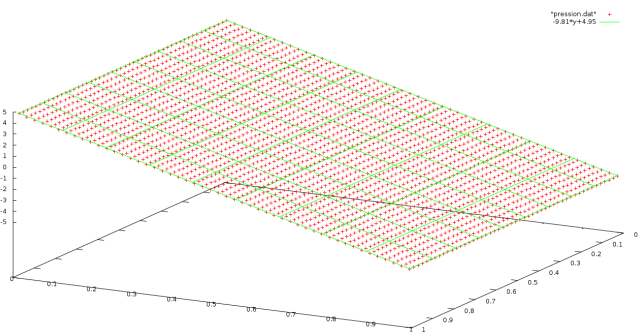

FIGURE 13 – Pression

Pour ces premiers cas tests, nous observons que la solution numérique a bien convergé bien que de prime abord, la solution semble "chaotique" dans le cas test $(0a)$. Si ces courbes sont observées avec une échelle plus large selon l'axe des ordonnées ce sont bien des plans constants qui seront visualisés.

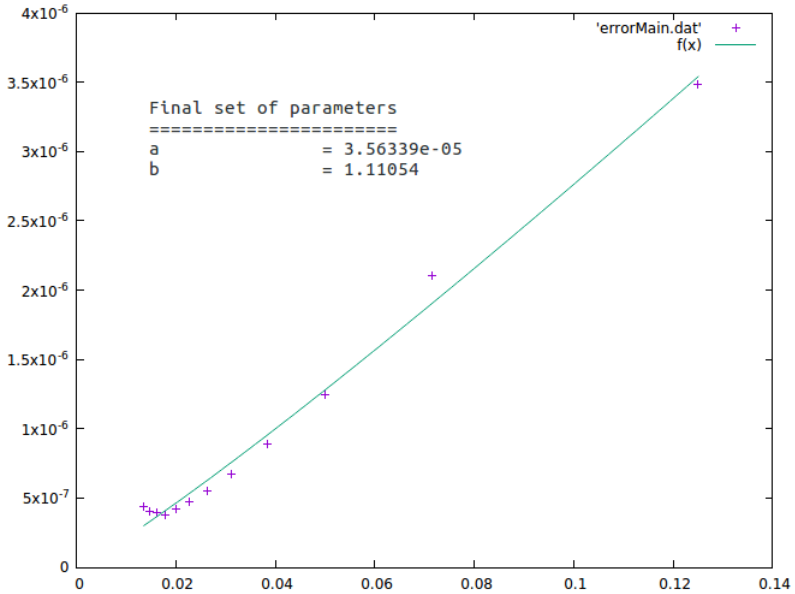

FIGURE 14 – Etude d'ordre (ordre 1 obtenu)

Le Figure 14 représente l'ordre en espace du schéma dans le cadre de problème $(0b)$. Nous obtenons de l'ordre 1 comme attendu, puisque le schéma MAC est d'ordre 1. L'ordre n'a pas pu être observé via le problème $(0a)$ car ce dernier converge en 2 itérations d'Uzawa avec une précision bien supérieure à celle imposée. Ce phénomène s'explique par des conditions limites extrêmement simples qui doivent bien réagir avec des conditions initiales constantes telles que celle que nous imposons.

3.3 Cas tests 1 : TC1a et TC1b

L'algorithme a ensuite été testé sur les cas tests plus compliqués (10) et (11).

Cas test TC1a :

$$\begin{cases} U = (x, -y) \\ F = (1, 1) \\ G = H = U \\ P = x + y + c \end{cases} \tag{10}$$

Cas test TC1b :

$$\begin{cases} U = (x, y) \\ F = (1, 1) \\ G = H = U \\ P = x + y + c \end{cases} \tag{11}$$

Cas test TC1a :

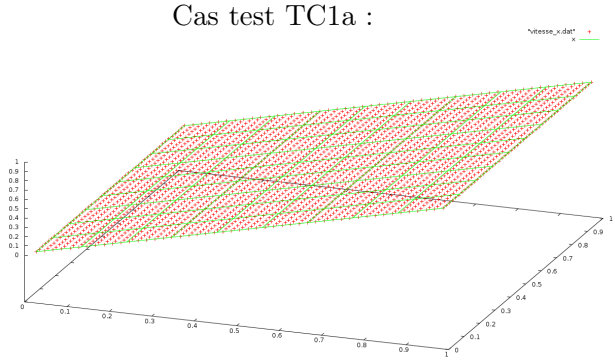

FIGURE 15 – Vitesse selon x

Cas test TC1b :

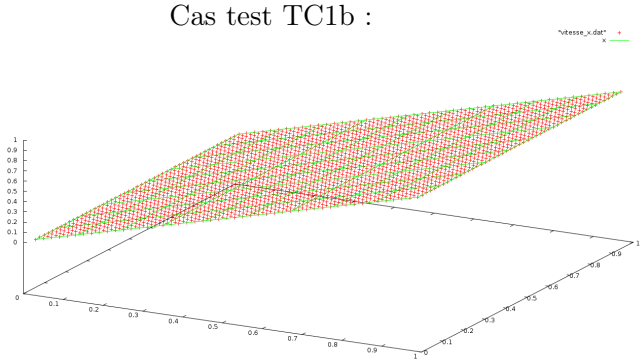

FIGURE 16 – Vitesse selon x

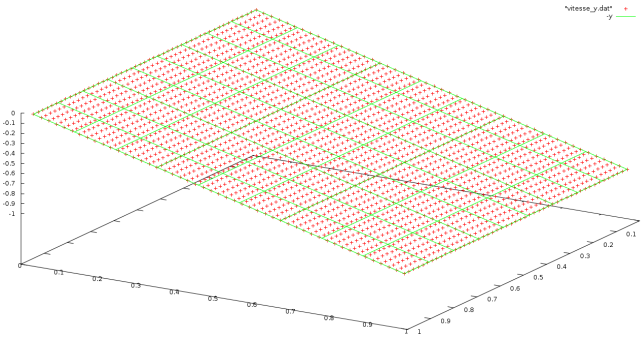

FIGURE 17 – Vitesse selon y

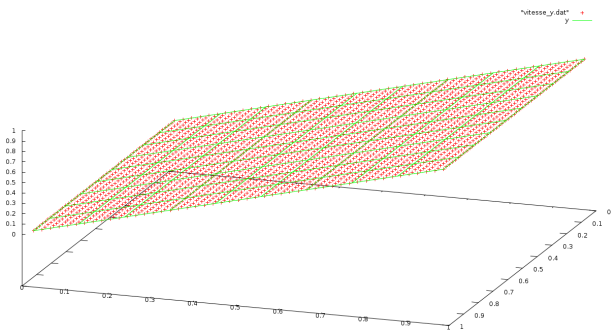

FIGURE 18 – Vitesse selon y

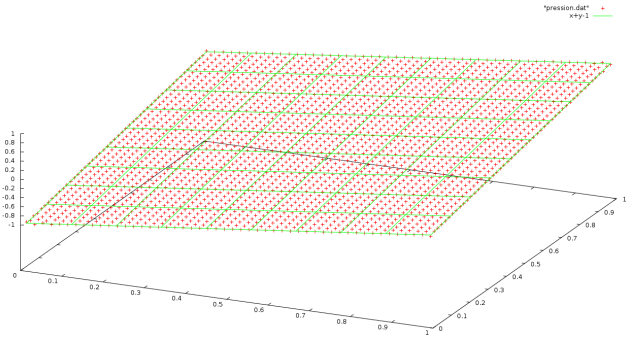

FIGURE 19 – Pression

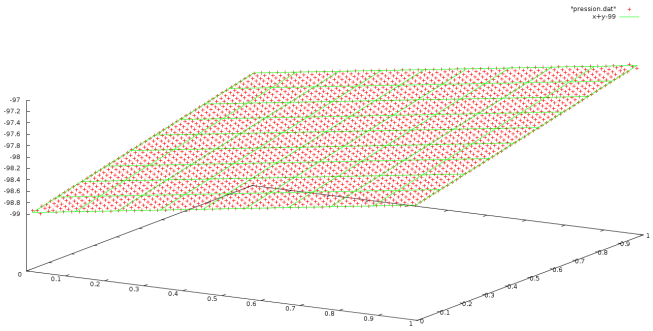

FIGURE 20 – Pression

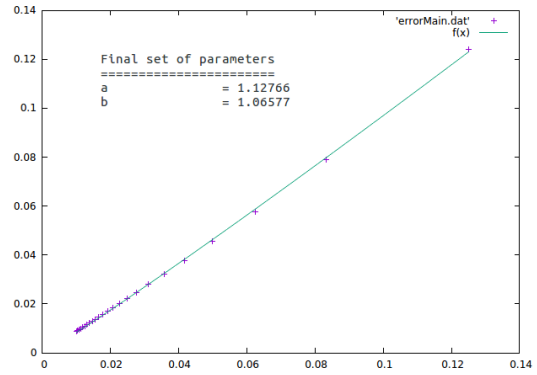

FIGURE 21 – Etude d'ordre (ordre 1 obtenu)

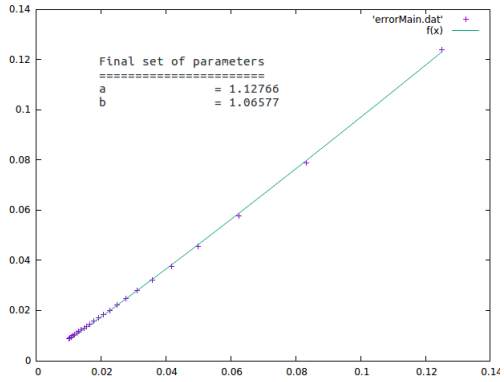

FIGURE 22 – Etude d'ordre (ordre 1 obtenu)

Les Figures 15 à 22 montrent bien les solutions planaires attendues . Les solutions numériques semblent converger correctement vers la solution analytique. Mais un phénomène surprenant est ici observé. Malgré des conditions initiales et au bords similaires entre les cas test 1a et 1b, la pression obtenue par le cas 1a est de moyenne nulle contrairement à celle du cas 1b. Pourtant, chacune de ces pressions s'aligne avec la solution analytique autrement dit le plan $P = x + y + cste$. Afin de répondre au sujet et imposer à la pression une intégrale nulle, une fonction $Moy\_0()$ à été ajoutée à la fin de chaque itération d'Uzawa afin de mettre à jour la valeur de P mais aussi de lui imposer une moyenne nulle.

```
FUNCTION Moy_0(V)
  IMPLICIT NONE
  REAL*8,DIMENSION(:),INTENT(IN)::V
  REAL*8,DIMENSION(size(V))::Moy_0
  REAL*8::moy
  INTEGER::i

  moy=SUM(V)/size(V)

  DO i=1,size(V)
    Moy_0(i)=V(i)-moy
  END DO
END FUNCTION
```

FIGURE 23 – Fonction Moy_0

Cette fonction permet alors d'obtenir les mêmes résultats en x et en y mais de forcer la pression obtenue à être de moyenne nulle, comme illustré par les Figures 1 et 25.

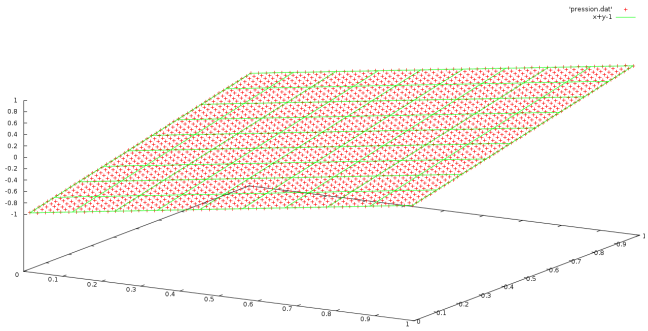

FIGURE 24 – Pression obtenue avant $Moy\_0()$

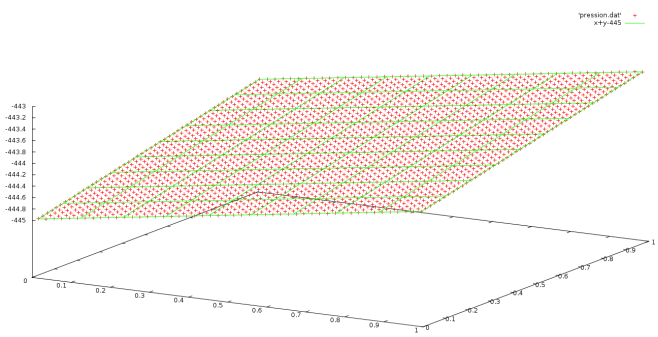

FIGURE 25 – Pression obtenue après $Moy\_0()$

3.4 Cas tests TC2a : Écoulements de Poiseuille

La méthode a ensuite été simulé sur un cas test physique tel qu'un écoulement de Poiseuille avec différentes conditions limites.

Prenons d'abord le cas d'un écoulement de Poiseuille avec une vitesse constante horizontale en entrée et en sortie :

$$\begin{cases} F = (0,0) \\ G = (1,0) \text{ et } H = (0,0) \end{cases} \tag{12}$$

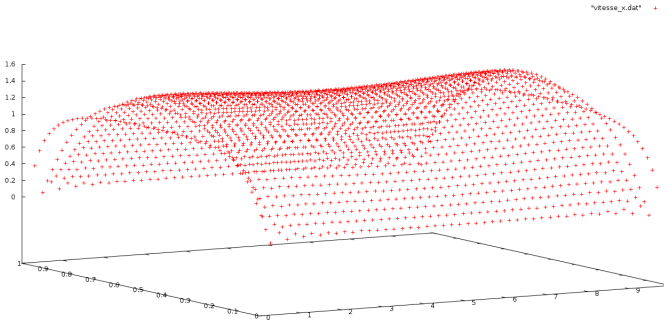

FIGURE 26 – Vitesse selon x

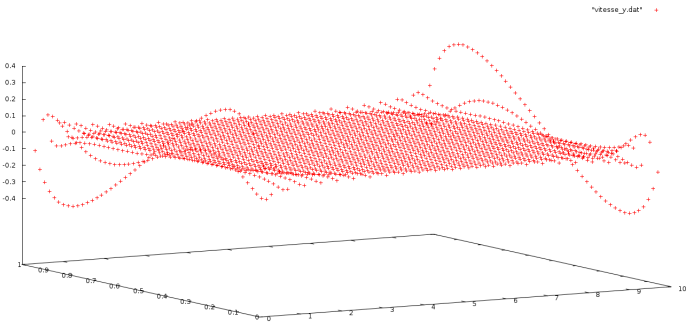

FIGURE 27 – Vitesse selon y

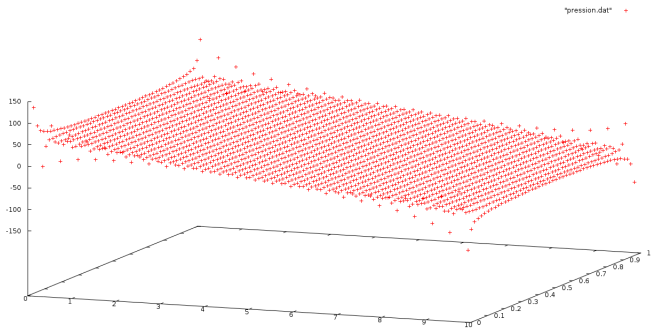

FIGURE 28 – Pression

La pression sur la Figure 28 est bien linéaire en x comme attendu pour un écoulement de Poiseuille. Quant à la vitesse selon x sur la Figure 26, on peut y voir l'établissement de l'écoulement de Poiseuille pleinement développé au milieu du canal. La vitesse en y réagit mal sur les bords 27, cela étant dû aux points singuliers du domaine et aux conditions aux limites.

On teste ensuite l'écoulement où une vitesse de Poiseuille est imposée en entrée et en sortie. On connaît alors la solution analytique.

$$\begin{cases} U = (L_y^2(1 - \frac{y}{L_y})\frac{y}{L_y}, 0) \\ F = (0, 0) \\ G = (L_y^2(1 - \frac{y}{L_y})\frac{y}{L_y}, 0) \text{ et } H = (0, 0) \\ P = -2x + c \end{cases} \tag{13}$$

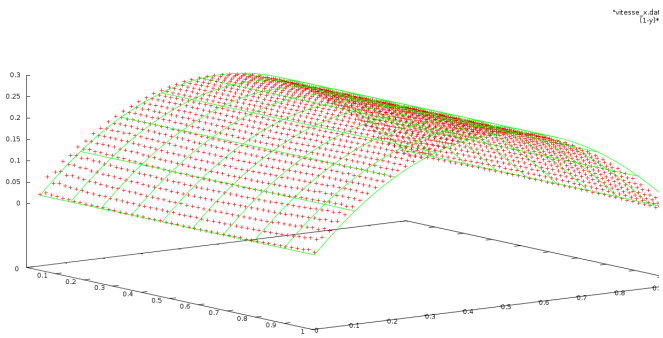

FIGURE 29 – Vitesse selon x

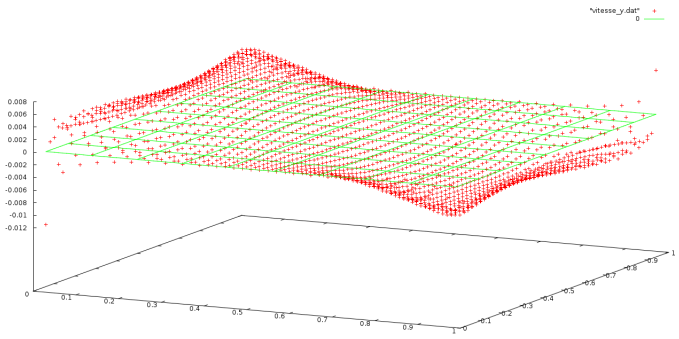

FIGURE 30 – Vitesse selon y

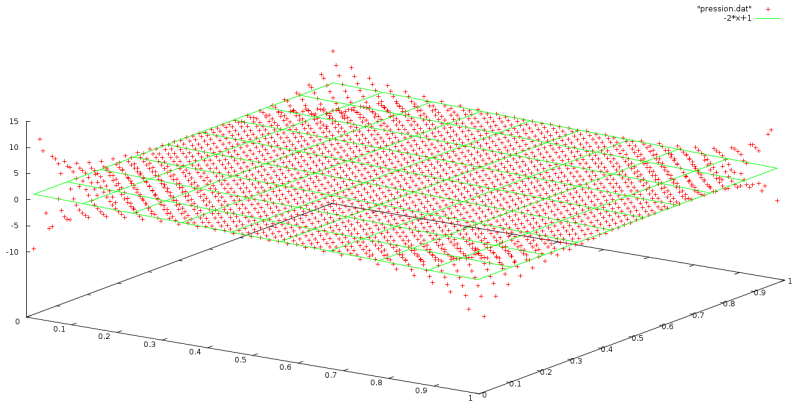

FIGURE 31 – Pression

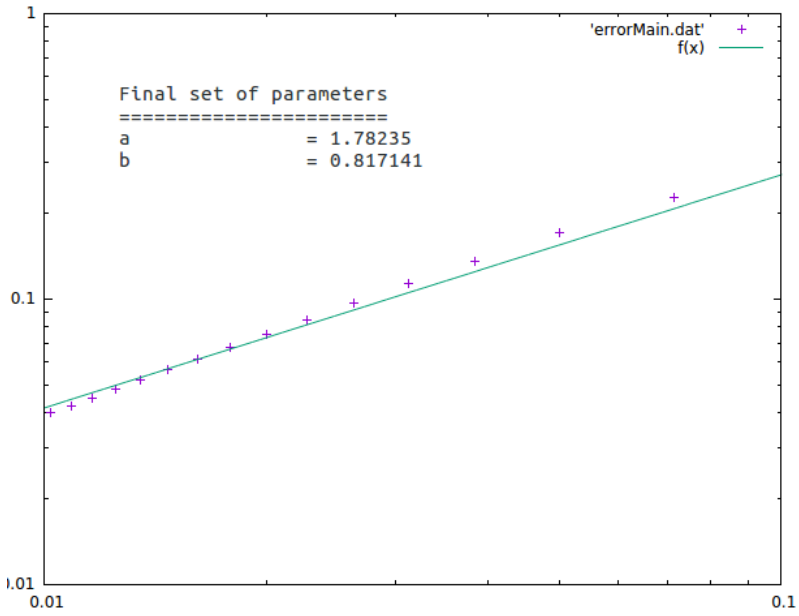

FIGURE 32 – Etude d'ordre (ordre 1 obtenu)

De la même manière que dans le cas test TC1b le nombre d'itération de l'algorithme d'Uzawa influence le résultat en pression. En effet le solution se scinde visuellement en

deux solutions opposés comme illustré par la Figure 34. En revanche, le plan moyen semble correspondre à la solution analytique attendue comme illustrée sur la Figure 33. Cela est bien confirmé lorsque la tolérance de l'algorithme d'Uzawa est augmentée de $1.e-6$ à $1.e-3$. On retrouve alors le plan attendu illustré sur la Figure 31.

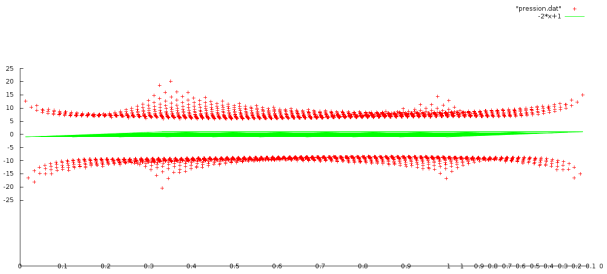

FIGURE 33 – Pression Poiseuille angle de vue 1

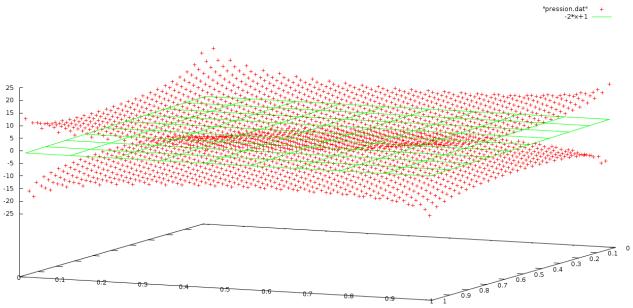

FIGURE 34 – Pression Poiseuille angle de vue 2

A l'image des observations obtenues lors du cas test TC1b, il se trouve que la solution en pression est définie au $Ker(B^T)$ près. En effet le vecteur constant $(1, 1, 1, ...)$ appartient à cet ensemble et c'est pourquoi, la pression est définie à une constante près. Ce problème a été facilement contournée par la fonction $Moy\_0()$ définie précédemment. Dans le cas du Poiseuille, la pression est en plus, définie au vecteur $V = (1, -1, 1, -1, 1, -1, ...)$ près, autrement dit V appartient aussi à $Ker(B^T)$. Cela explique pourquoi la pression obtenue est scindée en deux parties distinctes.
Ainsi, plus l'algorithme d'Uzawa fait d'itérations, plus ces deux surfaces de pression vont s'éloigner l'une de l'autre, tout en préservant un plan moyen exact.
Idéalement, une seconde fonction de filtrage aurait pu être implémentée dans l'algorithme d'Uzawa afin de soustraire du $Ker(B^t)$ la composante induite par le vecteur V.

4 Influence de μ

μ	Cas test 0b		Cas test 1a	
-	Nb Iter	Résidu	Nb Iter	Résidu
1	749	10-4	223	10-4
2	1000	2.10-3	1000	3.10-4
3	-	NaN	-	NaN
0.5	970	10-4	292	10-4
0.1	1000	2.10-3	592	10-4
1.5	641	10-4	191	10-4

FIGURE 35 – Influence de μ sur le nombre d'itération et le résidu - $[(Nx, Ny) = (50, 50)$,$NiterMaxUzawa = 1000$,$EpsilonUzawa = 10^{-4}$,$NiterMaxGradConjugue = 1000$,$EpsilonGradConjugue = 10^{-6}]$

Comme le montre le tableau précédent, μ influence la vitesse de convergence de l'algorithme d'Uzawa. Il existe un μ optimal qui permet de minimiser le nombre d'itération. Ce μ optimal dépend du problème et de ses conditions initiales.

Si l'on corrèle l'augmentation du nombre d'itérations avec les perturbations engendrées par le $Ker(B^T)$, on a alors tout intérêt de choisir un μ adapté pour une question de précision et de vitesse de calcul.

5 Conclusion

Pour conclure, le problème de Stokes a été résolue par l'algorithme d'Uzawa et celui du gradient conjugué. Le gradient conjugué a d'abord été validé sur un problème de Poisson. Ensuite, divers cas tests ont permis d'améliorer la résolution notamment la convergence sur la pression. Enfin, une étude a été portée sur l'influence du coefficient μ d'Uzawa, qui fait diverger la méthode à partir d'un certain seuil. Pour aller plus loin, il faudrait dans un premier temps filtrer les composantes indésirables de la solution appartenant à Ker(B^T). Enfin, la notion instationnaire du schéma implémenté n'a pas été validé. Il aurait été aussi intéressant de poursuivre l'étude sur de tels problèmes.