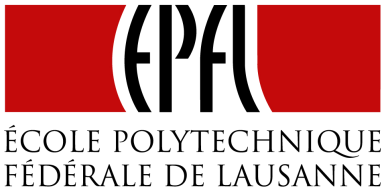


SMA

Résolution numérique du problème de Stokes en $3D$ avec l'élément fini $\mathbb{P}_2 - \mathbb{P}_1$

Jonathan ROCHAT
Sous la direction des professeurs J. Rappaz
et M.Picasso, MATHICSE-ASN

Remerciements

Je tiens à remercier le professeur Jacques Rappaz pour la proposition de ce très beau sujet ainsi que pour m'avoir mis à disposition une place de travail. Je remercie le professeur Marco Picasso pour tout ce que j'ai appris en faisant ce travail et pour toutes les heures passées dans son bureau à discuter du sujet. Je remercie également les doctorants Laurent Michel, Guillaume Jouvet, Stephane Flotron et Samuel Quinodoz pour leurs conseils sur le sujet et les astuces informatiques.

Table des matières

1 Introduction **5**

2 Discrétisation du problème **6**
2.1 Rappel . . . 6
2.2 Forme faible du problème et définition des espaces fonctionnels utilisés 6
2.3 Existence et unicité de la solution . . . 7
2.4 Approximation de Galerkin . . . 9
2.5 Choix des espaces V_h et Q_h . . . 11
2.5.1 Les espaces *inf-sup stables et instables* . . . 11
2.5.2 Les espaces de Taylor-Hood . . . 11
2.6 Algorithme de résolution du problème de Stokes . . . 12
2.6.1 La méthode UMFPACK . . . 12
2.6.2 Méthode directe ou itérative ? . . . 12
2.6.3 La méthode du gradient conjugué . . . 13
2.6.4 La méthode GMRES . . . 13
2.6.5 La méthode d'Uzawa . . . 14

3 Le logiciel FreeFem **15**
3.1 Pression à moyenne nulle . . . 15
3.2 Calcul des erreurs . . . 15
3.3 Quelques mots sur les commandes des méthodes . . . 16

4 Résolution numérique du problème de Stokes **17**
4.1 K-test avec solution exacte . . . 17
4.1.1 2D . . . 17
4.1.2 $3D$ . . . 18
4.2 Application dans un écoulement de Poiseuille . . . 23
4.2.1 Solution exacte d'un écoulement de Poiseuille . . . 23
4.2.2 $2D$ . . . 24
4.2.3 $3D$ . . . 25
4.3 Cavité carrée . . . 29
4.3.1 Résolution avec Uzawa . . . 29
4.3.2 Résolution avec le gradient conjugué . . . 31
4.3.3 Comparaison des deux méthodes . . . 31

5 Conclusion **32**

Bibliographie **33**

6 Annexe : code FreeFem des tests **34**
6.1 K-test sur Solution Exact . . . 34
6.1.1 K-test du triangle . . . 34
6.1.2 K-test du cube . . . 35
6.2 Ecoulement de Poiseuille . . . 36
6.2.1 2D . . . 36
6.2.2 3D . . . 37
6.3 Problème de la cavité carrée . . . 38
6.3.1 Uzawa . . . 38
6.3.2 Gradient Conjugué . . . 39

Résumé

Nous allons étudier le problème de Stokes pour les fluides, d'abord sous une approche théorique et ensuite de manière plus pratique par la résolution de quelques cas classiques en $2D$ et $3D$ avec différentes méthodes sur le logiciel FreeFem de F. Hecht.

1 Introduction

L'équation de Stokes permet de décrire un fluide visqueux coulant lentement dans un lieu étroit ou autour d'un petit objet. La viscosité domine donc sur les effets inertiels. Par exemple, l'écoulement de Poiseuille qui décrit l'évolution d'un fluide dans un tuyau est solution de cette équation. L'écoulement de Couette, qui décrit un fluide visqueux entre deux surfaces dont l'une est en mouvement par rapport à l'autre est aussi régi par ce problème de Stokes.

Dans ce travail nous allons nous intéresser au problème de Stokes suivant : pour Ω un ouvert de $\mathbb{R}^3$, étant donné une fonction $\mathbf{f} : \Omega \to \mathbb{R}^3$, on veut trouver des fonctions $\mathbf{u} : \Omega \to \mathbb{R}^3$ et $p : \Omega \to \mathbb{R}$ telles que

$$\begin{array}{ll} -\nu\Delta\mathbf{u} + \nabla p = \mathbf{f} & \text{dans } \Omega, \\ \text{div } \mathbf{u} = 0 & \text{dans } \Omega, \\ \mathbf{u} = 0 & \text{sur } \partial\Omega. \end{array}$$

Le vecteur $\mathbf{u}$ représente le champ de vitesse du fluide. La fonction scalaire p est la pression qui lui est associée. On considère encore le champ de force $\mathbf{f}$ agissant sur le système et le coeffficient de viscocité cinématique ν. Ici nous nous intéresserons uniquement au cas des fluides incompressibles, régi par l'équation div $\mathbf{u} = 0$.

Le but de ce travail sera la résolution de ce problème avec l'aide des éléments finis $\mathbb{P}2-\mathbb{P}1$ sur le logiciel FreeFem de F. Hecht. Dans les cas pratiques, nous aurons des conditions de bord non nulle $\mathbf{u}$.

Notation 1.1
Soit $d \in \mathbb{N}$, $d \geq 1$. Dans ce travail nous travaillerons avec des champs vectoriels de fonctions $\boldsymbol{v} : \Omega \to \mathbb{R}^d$, que nous noterons

$$\boldsymbol{v} = (v_1, \ldots, v_d).$$

Pour un tel champ, nous écrirons les opérateurs vectoriels associés de la manière suivante :

$$\begin{aligned} \text{div}\boldsymbol{v} &= \sum_{i=1}^{d} \frac{\partial v_i}{\partial x_i}, \\ \nabla \boldsymbol{v} = \begin{pmatrix} \nabla v_1 \\ \ldots \\ \nabla v_d \end{pmatrix} &= \begin{pmatrix} \frac{\partial v_1}{\partial x_1}, \ldots, \frac{\partial v_1}{\partial x_d} \\ \cdots \\ \frac{\partial v_d}{\partial x_1}, \ldots, \frac{\partial v_d}{\partial x_d} \end{pmatrix} \text{ et } \Delta \boldsymbol{v} = \begin{pmatrix} \Delta v_1 \\ \ldots \\ \Delta v_d. \end{pmatrix}. \end{aligned}$$

Nous travaillerons également beaucoup avec des espaces fonctionnels de Hilbert $H^1(\Omega)^d$. La norme d'un champ vectoriel $\boldsymbol{u}$ dans un tel espace se note :

$$\|\boldsymbol{u}\|_{H^1(\Omega)^d} := \left(\sum_{j=1}^{d} \|u_j\|^2_{H^1(\Omega)} \right)^{\frac{1}{2}}.$$

Pour plus de détails sur ces espaces qui sont de type Sobolev, nous vous invitons à consulter [8], [1] ou [9].

Le produit scalaire euclidien entre deux vecteurs $\boldsymbol{a}$ et $\boldsymbol{b}$ sera noté $\boldsymbol{a} \cdot \boldsymbol{b}$. Pour deux matrices A et B, nous désignerons par $A : B$ la trace de la matrice produit de A et B. Cette notation nous sera utile lorsque nous mettrons notre problème sous la formulation faible.

Exemple 1.2
Dans le cas $d = 3$, si $\boldsymbol{u}, \boldsymbol{v} : \Omega \to \mathbb{R}^3$, alors le produit des matrices jabobiennes $\nabla \boldsymbol{u}$ et $\nabla \boldsymbol{v}$ est par définition une matrice de dimension 3. Nous aurons ainsi

$$\int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v} = \int_\Omega tr(\nabla \boldsymbol{u} \nabla \boldsymbol{v}) = \sum_{i=1}^{3} \int_\Omega \nabla u_i \cdot \nabla v_i = \sum_{i,j=1}^{3} \int_\Omega \frac{\partial u_i}{\partial x_j} \frac{\partial v_i}{\partial x_j}.$$

2 Discrétisation du problème

Pratiquement, l'équation de Stokes se résout en dimension 2 ou 3. Néanmoins, dans le souci d'une écriture mathématique plus rigoureuse, nous considérerons dans ce chapitre le problème de Stokes sous une dimension quelconque, c'est-à-dire que nous choisirons un ouvert $\Omega \subset \mathbb{R}^d$, où $d \in \mathbb{N}$ et dont le bord est suffisamment régulier (en général on peut supposer Ω lipschitzien, ce qui nous permet de définir la valeur de la fonction sur le bord grâce à un opérateur de trace, voir [8] ou [1]). Pour étudier la forme variationnelle de notre problème, il est nécessaire de rappeler les résultats d'analyse vectorielle suivants :

2.1 Rappel

Proposition 2.1
Soit Ω un ouvert régulier de $\mathbb{R}^d$ et $\boldsymbol{u}, \boldsymbol{v} : \bar{\Omega} \to \mathbb{R}^d$ des fonctions de classe C^2. Alors

$$-\int_\Omega \Delta \boldsymbol{u} \cdot \boldsymbol{v} = \int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v} - \int_{\partial\Omega} ((\nabla \boldsymbol{u})\boldsymbol{n}) \cdot \boldsymbol{v}$$

où $\boldsymbol{n} = (n_1, \ldots, n_d)$ est le vecteur normal unitaire à $\partial\Omega$.

Démonstration. On développe et on utlise les formules de Green pour les fonctions scalaires :

$$\begin{aligned}
-\int_\Omega \Delta \mathbf{u} \cdot \mathbf{v} &= -\int_\Omega \begin{pmatrix} \Delta u_1 \\ \ldots \\ \Delta u_d \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ \ldots \\ v_d \end{pmatrix} \\
&= -\sum_{k=1}^{d} \int_\Omega (\Delta u_k \cdot v_k) \\
&= \sum_{k=1}^{d} \left(\int_\Omega \nabla u_k \cdot \nabla v_k - \int_{\partial\Omega} (\nabla u_k \cdot \mathbf{n}) \cdot v_k \right) \\
&= \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v} - \int_{\partial\Omega} ((\nabla \mathbf{u})\mathbf{n}) \cdot \mathbf{v}.
\end{aligned}$$

□

Proposition 2.2
Soit Ω un ouvert régulier de $\mathbb{R}^d$ et $p : \bar{\Omega} \to \mathbb{R}$ une fonction de classe C^1 et $\boldsymbol{v} : \bar{\Omega} \to \mathbb{R}^d$ un champ vectoriel de classe C^2 . Alors

$$\int_\Omega \nabla p \cdot \boldsymbol{v} = -\int_\Omega p \cdot \mathrm{div} \boldsymbol{v} + \int_{\partial\Omega} (p\boldsymbol{n}) \cdot \boldsymbol{v},$$

où $\boldsymbol{n} = (n_1, \ldots, n_d)$ est le vecteur normal unitaire à $\partial\Omega$.

Démonstration. Il suffit simplement d'utiliser comme précédemment les formules de Green pour les fonctions scalaires. □

2.2 Forme faible du problème et définition des espaces fonctionnels utilisés

Nous considérons le problème de Stokes suivant : trouver $\mathbf{u} : \Omega \to \mathbb{R}^d$ et $p : \Omega \to \mathbb{R}$ tel que

$$\begin{array}{ll}
-\Delta \mathbf{u} + \nabla p = \mathbf{f} & \text{dans } \Omega \\
\mathrm{div}\ \mathbf{u} = 0 & \text{dans } \Omega, \\
\mathbf{u} = 0 & \text{sur } \partial\Omega.
\end{array} \tag{1}$$

Nous supposons que $\mathbf{f} \in (L^2(\Omega))^d$. Nous multiplions la première équation par un champ vectoriel $\mathbf{v}$ dans l'espace des fonctions tests $(\mathcal{D}(\Omega))^d$ et nous intégrons. Le problème devient

$$-\int_\Omega \Delta \mathbf{u} \cdot \mathbf{v} + \int_\Omega \nabla p \cdot \mathbf{v} = \int_\Omega \mathbf{f} \cdot \mathbf{v} \qquad \forall \mathbf{v} \ \in (\mathcal{D}(\Omega))^d.$$

En utilisant les propositions (2.1) et (2.2), l'expression devient

$$\int_\Omega (\nabla \mathbf{u} : \nabla \mathbf{v} - p\text{div}\mathbf{v}) + \int_{\partial\Omega} ((-(\nabla \mathbf{u})\mathbf{n}) \cdot \mathbf{v} + (p\mathbf{n}) \cdot \mathbf{v}) = \int_\Omega \mathbf{f} \cdot \mathbf{v} \quad \forall \mathbf{v} \in (\mathcal{D}(\Omega))^d.$$

De plus, afin de simplifier le problème variationel, nous cherchons une fonction s'annulant sur $\partial\Omega$. Il est donc naturel de chercher cette fonction dans un espace de Sobolev avec des conditions de bord de Dirichlet. Nous définissons ainsi

$$V := H_0^1(\Omega)^d.$$

Concernant la pression, puisque l'équation de Stokes n'en fait intervenir que les dérivées, nous sommes poussés à imposer une condition sur p qui nous fixe la constante d'intégration afin de garantir l'unicité d'une telle pression. Nous choisissons à cet effet l'espace des fonctions de carré sommable à moyenne nulle :

$$Q = L_0^2(\Omega) := \left\{ q \in L^2(\Omega) \middle| \int_\Omega q = 0 \right\}.$$

Pour l'équation d'incompressibilité, div $\mathbf{u} = 0$, nous la multiplions par une fonction scalaire $q : \Omega \to \mathbb{R}$ dans l'espace Q et nous intégrons. Nous obtenons

$$\int_\Omega q\text{div } \mathbf{u} = 0 \quad \forall q \in Q.$$

Finalement, en définissant les formes bilinéaires $a : V \times V \to \mathbb{R}$ et $b : V \times Q \to \mathbb{R}$:

$$a(\mathbf{u}, \mathbf{v}) := \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v}, \tag{2}$$

$$b(\mathbf{v}, q) := -\int_\Omega q\text{div}\mathbf{v} \tag{3}$$

ainsi que la fonctionnelle linéaire $F : V \to \mathbb{R}$:

$$F(\mathbf{v}) := \int_\Omega \mathbf{f} \cdot \mathbf{v},$$

le problème (1) devient

$$\begin{array}{lll} \text{Trouver } \mathbf{u} \in V, p \in Q \text{ tel que} & & \\ a(\mathbf{u}, \mathbf{v}) + b(\mathbf{v}, p) = F(\mathbf{v}) & \forall \mathbf{v} \in V & \\ b(\mathbf{u}, q) = 0 & \forall q \in Q. & \end{array} \tag{4}$$

Ce problème est appelé la formulation faible de l'équation de Stokes.

2.3 Existence et unicité de la solution

Nous devons maintenant nous assurer que le problème ci-dessus est bien posé. Rappelons à cet effet le fait suivant :

Remarque 2.3
Si u_i est un champ scalaire dans l'espace $H^1(\Omega)$, alors par définition de la norme H^1 :

$$\|\nabla u_i\|_{L^2(\Omega)} := \left(\sum_{j=1}^d \left\| \frac{\partial u_i}{\partial x_j} \right\|_{L^2(\Omega)}^2 \right)^{\frac{1}{2}} \leq \|u_i\|_{H^1(\Omega)}.$$

Nous vérifions ensuite les hypothèses suivantes :

1. La forme bilinéaire $a(\mathbf{u}, \mathbf{v})$ est continue sur V : en effet, grâce à la remarque précédente et aux inégalités de Hölder (dans $L^2(\Omega)$) et de Cauchy-Schwarz

(dans $\mathbb{R}^d$), pour tout $\mathbf{u}, \mathbf{v} \in V$ on a

$$\begin{aligned} |a(\mathbf{u},\mathbf{v})| = \Big| \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v} \Big| &\leq \sum_{i,j=1}^{d} \int_\Omega \Big| \frac{\partial u_i}{\partial x_j} \frac{\partial v_i}{\partial x_j} \Big| \\ &\leq \sum_{i,j=1}^{d} \Big\| \frac{\partial u_i}{\partial x_j} \Big\|_{L^2(\Omega)} \Big\| \frac{\partial v_i}{\partial x_j} \Big\|_{L^2(\Omega)} \\ &\leq \sum_{i=1}^{d} \|u_i\|_{H^1(\Omega)} \|v_i\|_{H^1(\Omega)} \\ &\leq \|\mathbf{u}\|_V \|\mathbf{v}\|_V, \end{aligned}$$

et donc on peut choisir $\gamma = 1$ comme constante de continuité.

2. La forme bilinéaire $a(\mathbf{u}, \mathbf{v})$ est coercive sur V. Remarquons d'abord que pour tout $\mathbf{u} \in V$,

$$a(\mathbf{u},\mathbf{u}) = \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{u} = \sum_{j=1}^{d} \int_\Omega \nabla u_j \cdot \nabla u_j.$$

Vérifions dans un premier temps que la forme bilinéaire

$$a_j(u,v) = \int_\Omega \nabla u_j \cdot \nabla u_j$$

est coercive. En effet, grâce à l'inégalité de Poincaré (voir [8] ou [9]),

$$\|u_j\|_V^2 = \|u_j\|_{L^2(\Omega)}^2 + \|\nabla u_j\|_{L^2(\Omega)}^2 \leq (1 + C_\Omega^2) \|\nabla u_j\|_{L^2(\Omega)}^2$$

où C_Ω est la constante de Poincaré. Par conséquent,

$$a_j(u,u) = \int_\Omega (\nabla u_j)^2 = \|\nabla u_j\|_{L^2(\Omega)}^2 \geq \frac{1}{1 + C_\Omega^2} \|u_j\|_V^2.$$

est donc la coercivité de a_j est satisfaite pour la constante $\alpha = \frac{1}{1+C_\Omega^2} > 0$. Puisque

$$a(\mathbf{u},\mathbf{u}) = \sum_{j=1}^{d} a_j(u,v) \geq \sum_{j=1}^{d} \alpha \|u_j\|_V^2 \geq \alpha \|\mathbf{u}\|_V^2,$$

on a ainsi montré que la forme bilinéaire a est coercive pour la constante de coercivité $\alpha = \frac{1}{1+C_\Omega^2} > 0$.

3. La forme bilinéaire $b(\mathbf{v}, q)$ est continue : en effet par les inégalités de Hölder (dans $L^2(\Omega)$) et de Cauchy-Schwarz (dans $\mathbb{R}^d$),

$$\begin{aligned} |b(\mathbf{u},q)| &= \Big| \int_\Omega q \mathrm{div} \mathbf{u} \Big| \leq \sum_{k=1}^{d} \int_\Omega \Big| \frac{\partial u_k}{\partial x_k} q \Big| \leq \sum_{k=1}^{d} \Big\| \frac{\partial u_k}{\partial x_k} \Big\|_{L^2(\Omega)} \|q\|_{L^2(\Omega)} \\ &\leq \left(\sum_{k=1}^{d} \|u_k\|_V^2 \right)^{\frac{1}{2}} \sqrt{d} \|q\|_{L^2(\Omega)} \leq \delta \|\mathbf{u}\|_V \|q\|_Q \quad \forall \mathbf{u} \in V, q \in Q. \end{aligned}$$

On choisit donc $\delta = \sqrt{d} > 0$ comme constante de continuité.

4. On peut finalement montrer (voir [2],[3]) que les espaces V et Q vérifient la propriété suivante :

$$\exists B^* > 0 \text{ tel que } \quad \forall q \in Q \quad \exists \mathbf{v} \in V, \mathbf{v} \neq 0 : b(\mathbf{v},q) \geq B^* \|\mathbf{v}\|_V \|q\|_Q. \qquad (5)$$

Cette condition est très importante : on l'appelle la **condition de l'inf-sup**.

Finalement, par le théorème 7.4.2 dans [1], (on le trouve aussi dans [2]) notre problème admet une unique solution $(\mathbf{u}, p)$ car il vérifie toutes les conditions précédente. Nous avons l'estimation supplémentaire sur nos solutions :

$$\|\mathbf{u}\|_V \leq \frac{1}{\overline{\alpha}} \|\mathbf{f}\|_{V'}$$

ainsi que

$$\|p\|_Q \leq \frac{1}{B^*} \Big[\left(1 + \frac{1}{\overline{\alpha}}\right) \|\mathbf{f}\|_{V'} \Big],$$

où B^* est la constante dans la condition de l'inf-sup et $\overline{\alpha}$ celle de coercivité. L'espace V' est le dual topologique de V, donc $V' = H^{-1}(\Omega)^d$.

Remarque 2.4
Pour démontrer la condition de l'inf-sup (7)*, par le lemme 4.1 de [2], p.58, il suffit de montrer que pour tout $q \in Q$, il existe un unique $\boldsymbol{v} \in V$ tel que div $\boldsymbol{v} = q$ et*

$$\|\boldsymbol{v}\|_V \leq C \|q\|_Q,$$

où $C = \frac{1}{B^*}$.

2.4 Approximation de Galerkin

Soit $V_h \subset V$ et $Q_h \subset Q$ des sous-espaces de dimension finie tel que $V_h \to V$ et $Q_h \to Q$ lorsque $h \to 0$. Le problème (4) s'approche de la manière suivante :

$$\begin{array}{ll} \text{Trouver } \mathbf{u}_h \in V_h, p_h \in Q_h \text{ tel que} & \\ a(\mathbf{u}_h, \mathbf{v}_h) + b(\mathbf{v}_h, p_h) = F(\mathbf{v}_h) & \forall \mathbf{v}_h \in V_h, \\ b(\mathbf{u}_h, q_h) = 0 & \forall q_h \in Q_h. \end{array} \tag{6}$$

Pour montrer que ce problème est bien défini, on peut vérifier à nouveau les hypothèses du théorème 7.4.2 dans [1]. En particulier, il faut que les espaces V_h et Q_h vérifient à nouveau la condition de l'inf-sup :

$$\exists C^* = C^*(h) > 0 \quad \forall q_h \in Q_h \quad \exists \mathbf{v}_h \in V_h, \mathbf{v}_h \neq 0 : b(\mathbf{v}_h, q_h) \geq C^* \|\mathbf{v}_h\|_V \|q_h\|_Q. \tag{7}$$

Si la constante C^* ne dépend pas de h, alors on dira que la solution $(\mathbf{u}_h, p_h)$ du problème (6) est *stable* . Pour chaque h fixé, nous notons N_h et K_h les dimensions des sous-espaces V_h respectivement Q_h ainsi que leur base associée :

$$\{\mathbf{g}_j | j = 1, \ldots, N_h\} \text{ et } \{\phi_l | l = 1, \ldots, K_h\}.$$

Si on écrit

$$\mathbf{u}_h(\mathbf{x}) = \sum_{j=1}^{N_h} u_j \mathbf{g}_j(\mathbf{x}), \quad p_h(\mathbf{x}) = \sum_{l=1}^{K_h} p_l \phi_l(\mathbf{x})$$

alors on obtient le système linéaire :

$$\begin{array}{ll} A\mathbf{u} + B^T \mathbf{p} & = \begin{pmatrix} \mathbf{F} \\ 0 \end{pmatrix} \\ B\mathbf{u} = 0, & \end{array} \tag{8}$$

où $\mathbf{u} = (u_1, \ldots, u_{N_h})$ et $\mathbf{p} = (p_1, \ldots, p_{K_h})$ sont les vecteurs des inconnues, $A_{ij} := a(\mathbf{g}_j, \mathbf{g}_i)$, $B_{il} := b(\mathbf{g}_l, \phi_i)$ et $\mathbf{F}_i := (\mathbf{f}, \mathbf{g}_i)$. A est une matrice carré $N_h \times N_h$ symétrique et définie positive par définition de la forme bilinéaire $a(\mathbf{u}, \mathbf{v})$. La matrice B est quant à elle rectangulaire de dimension $K_h \times N_h$. On peut aussi écrire le problème (8) sous la forme matricielle

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} \mathbf{F} \\ 0 \end{pmatrix}, \tag{9}$$

Pour que ce problème soit bien défini, il faut que la matrice carrée de dimension $(K_h + N_h)$

$$C := \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}$$

soit inversible. Nous avons pour cela le résultat suivant :

Proposition 2.5
Si $KerB^T = \mathbf{0}$, alors la matrice C est inversible.

Démonstration. Puisque la matrice A est inversible, l'équation (8) peut se réécrire

$$\mathbf{u} = A^{-1}(\mathbf{F} - B^T\mathbf{p}) \tag{10}$$
$$BA^{-1}B^T\mathbf{p} = BA^{-1}\mathbf{F} \tag{11}$$

Notons que (11) ne dépend que de l'inconnue $\mathbf{p}$. De plus on peut observer que la matrice $R = BA^{-1}B^T$ est symétrique et définie positive si $\mathrm{Ker}B^T = \mathbf{0}$. En effet, pour tout $\mathbf{p} \in P_h$ non nul,

$$\mathbf{p}^T BA^{-1}B^T\mathbf{p} > 0$$

car A^{-1} est symétrique et définie positive puisque A l'est. Donc la matrice R est inversible et dans ce cas, le problème (11) est bien posé et il admet une unique solution $\mathbf{p}$. En reconsidérant l'équation (10), puisque A est inversible, $\mathbf{u}$ est donc uniquement déterminé car $\mathbf{p}$ l'est aussi. Par conséquent, le problème ci-dessus est bien posé et la matrice C est inversible.

□

Proposition 2.6
La condition inf-sup (7) *(vérifiée pour une constante $\beta^*(h)$ dépendante de h) est satisfaite si et seulement si $KerB^T = \mathbf{0}$;*

Démonstration. Nous allons démontrer un sens par contraposition. Nous allons montrer que $\mathrm{Ker}B^T \neq \mathbf{0}$ seulement si la condition inf-sup est violée. En effet,

$$\begin{aligned}
\mathrm{Ker}B^T \neq \mathbf{0} \quad &\Rightarrow \quad \exists \mathbf{p} = (p_1, \ldots, p_{K_h}) \neq \mathbf{0} \in \mathbb{R}^{K_h} \text{ tel que } B^T\mathbf{p} = \mathbf{0} \\
&\Rightarrow \quad \exists \mathbf{p} = (p_1, \ldots, p_{K_h}) \in \mathbb{R}^{K_h} \text{tel que } \sum_{j=1}^{K_h} b(\mathbf{g}_i, \phi_j)p_j = \mathbf{0}, \quad \forall i \leq N_h \\
&\Rightarrow \quad \exists \mathbf{p} = (p_1, \ldots, p_{K_h}) \in \mathbb{R}^{K_h} \text{tel que } b(\mathbf{g}_i, \sum_{j=1}^{K_h} p_j\phi_j) = \mathbf{0}, \quad \forall i \leq N_h \\
&\Rightarrow \quad \exists p_h \in Q_h \text{tel que } b(\mathbf{g}_i, p_h) = \mathbf{0}, \quad \forall i \leq N_h \\
&\Rightarrow \quad b(\mathbf{v}_h, p_h) = \mathbf{0} \quad \forall \mathbf{v}_h \in V_h, \mathbf{v}_h \neq 0.
\end{aligned}$$

Cette dernière relation viole clairement la condition de l'inf-sup. Inversement, si on suppose que

$$\exists B^*(h) > 0 \text{ tel que } \quad \forall q_h \in Q_h \quad \exists \mathbf{v}_h \in V_h, \mathbf{v}_h \neq 0 : b(\mathbf{v}_h, q_h) \geq B^* \|\mathbf{v}_h\|_{V_h} \|q_h\|_{Q_h}, \tag{12}$$

alors

$$\begin{aligned}
B^T(p_1, \ldots, p_{K_h}) = \mathbf{0} \quad &\Rightarrow \quad \sum_{j=1}^{K_h} p_j b(\mathbf{g}_i, \phi_j) = 0 \quad \forall i \in \{1, \ldots, N_h\} \\
&\Rightarrow \quad \left\| \sum_{j=1}^{K_h} p_j\phi_j \right\|_{Q_h} = 0 \Rightarrow \sum_{j=1}^{K_h} p_j\phi_j = 0 \\
&\Rightarrow \quad p_j = 0 \quad \forall j \in \{1, \ldots, K_h\}.
\end{aligned}$$

□

Remarque 2.7
Si $KerB^T = \mathbf{0}$, alors on peut facilement se convaincre à l'aide du théorème du rang que $N_h \geq K_h$.

2.5 Choix des espaces V_h et Q_h

2.5.1 Les espaces *inf-sup stables et instables*

Si nous choisissons des espaces V_h et Q_h qui vérifient la condition inf-sup, alors le problème est bien posé et donc il admet une unique solution. Que se passe-t-il si V_h et Q_h ne vérifient pas cette condition ? On peut remarquer qu'alors il existe $p_h^* \in Q_h$ tel que

$$b(\mathbf{v}_h, p_h^*) = \mathbf{0} \quad \forall \mathbf{v}_h \in V_h.$$

Par conséquent, si $(\mathbf{u}_h, p_h)$ est une solution du problème, alors $(\mathbf{u}_h, p_h + p_h^*)$ en est aussi une, puisque

$$a(\mathbf{u}_h, \mathbf{v}_h) + b(\mathbf{v}_h, p_h + p_h^*) = a(\mathbf{u}_h, \mathbf{v}_h) + b(\mathbf{v}_h, p_h) + b(\mathbf{v}_h, p_h^*) = \mathbf{F}(\mathbf{v}_h).$$

Cela signifie que la fonction p_h^* qui ne vérifie pas la condition inf-sup ne peut pas être détectée par l'approximation de Galerkin et donc on perd l'unicité de la solution. Ces fonctions sont appelées *"spurious pressure modes"*. Les espaces V_h et Q_h qui ne satisfont pas la condition inf-sup sont dits *inf-sup instable*. Pour rendre des espaces *inf-sup stable*, on peut recourir à des procédés de stabilisation, que nous n'étudierons pas ici par manque de temps.

2.5.2 Les espaces de Taylor-Hood

Lorsqu'on approxime le problème de Stokes, les fonctions de base ϕ_l, $1 \leq l \leq K_h$ ne sont pas forcément toujours continues, alors que les champs vectoriels de base $\mathbf{g}_i$, $1 \leq i \leq N_h$ le seront toujours. Remarquons de plus que puisque l'équation de Stokes ne fait intervenir que des dérivées d'ordre deux pour le champ $\mathbf{u}$ et d'ordre un pour la pression, il serait ainsi naturel d'utiliser des polynômes de degré $k \geq 1$ dans l'espace V_h et de degré $k-1$ dans l'espace Q_h. Dans ce travail nous nous intéresserons au cas $k = 2$ en résolvant le problème de Stokes sur des espaces d'éléments finis $\mathbb{P}_2$ pour V_h et $\mathbb{P}_1$ pour P_h.

Nous supposerons maintenant que Ω est un ouvert régulier de $\mathbb{R}^2$ ou $\mathbb{R}^3$. Pour chaque h fixé, on considère une triangulation T_h de Ω, c'est-à-dire une décomposition finie et régulière telle que :

$$\overline{\Omega} = \bigcup_{K \in T_h} K$$

vérifiant

1. chaque K est un polyhèdre tel que $\text{int}(K) \neq \emptyset$;
2. $int(K_1) \cap int(K_2) = \emptyset$ pour n'importe quel K_1 disctinct de K_2 ;
3. Si $E = K_1 \cap K_2 \neq \emptyset$, alors soit E est une arête, un plan commun ou un sommet de K_1 et K_2.

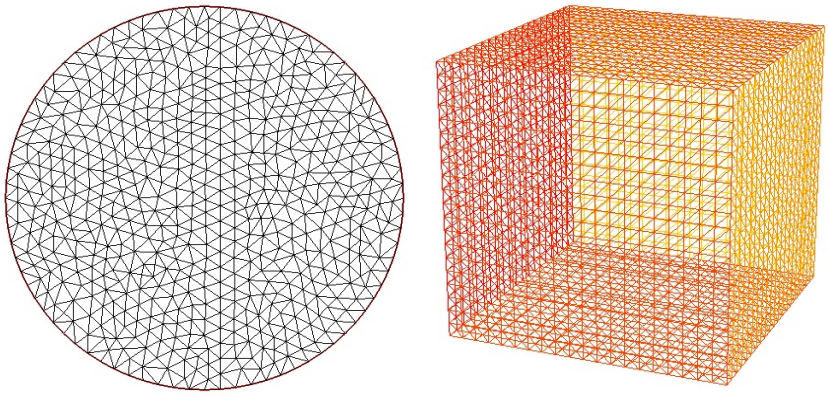


FIGURE 1 – Exemple de triangulation

On définit ensuite les espaces de Taylor-Hood de la manière suivante :

$$Q_h = \{X_h^1 \cap L_0^2(\Omega)\} \quad V_h = \{X_h^2 \cap H_0^1(\Omega)\}^d, d = 2, 3. \tag{13}$$

où, nous le rappelons, $X_h^r = \{\ g \in C(\overline{\Omega}) | g_{|K} \in \mathbb{P}_r, \forall K \in T_h\}$. Ces espaces satisfont la condition de l'inf-sup (voir [2], p.178) et convergent de manière optimale. A cet effet, on a l'estimation de convergence suivante

$$\|\mathbf{u} - \mathbf{u}_h\| + \|p - p_h\| \leq Ch^s(\|\mathbf{u}\|_{s+1} + \|p\|_s), \quad s = 1, 2.$$

(voir [2] p.176). Il existe plusieurs autres espaces d'éléments finis, adaptés soit pour des pressions continues ou discontinues. On peut montrer que les éléments $\mathbb{P}_{1_b} - \mathbb{P}_1$ ($\mathbb{P}_{1_b}$ est un espace $\mathbb{P}_1$ enrichi) sont inf-sup stables alors que les espaces $\mathbb{P}_1 - \mathbb{P}_1$ sont instables. Ici, nous travaillerons exclusivement avec l'élément fini $\mathbb{P}_2 - \mathbb{P}_1$.

2.6 Algorithme de résolution du problème de Stokes

Nous avons montré que le problème de Stokes peut être ramené aux problème suivant :

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} \mathbf{F} \\ 0 \end{pmatrix}, \tag{14}$$

Ce n'est rien d'autre qu'une résolution d'un système linéaire, où la matrice est symétrique et inversible si nos espaces vérifient la condition de l'inf-sup. Nous pouvons donc appliquer les algorithmes connus de résolution de systèmes linéaires pour résoudre ce problème.

2.6.1 La méthode UMFPACK

La méthode UMFPACK est un solveur direct pour des systèmes linéaire de la forme A$\mathbf{x}$=$\mathbf{b}$, où A est une matrice creuse pas forcément symétrique. Cet algorithme a été développé par le professeur Timothy A. Davis.

2.6.2 Méthode directe ou itérative ?

Dans les cas pratiques, nous verrons des réolutions de problème en deux et trois dimensions. Pour obtenir de bons résultats, il est nécessaire de prendre un $N = \frac{1}{h}$ suffisamment grand, car nous prendrons un maillage plus fin. Malheureusement, malgrès un comportement correct en deux dimensions, les méthodes directes comme UMFPACK se prêtent mal à ce genre de calcul pour des cas en 3 dimensions. Ceci provient du fait que la mémoire et le temps de calculs augmentent énormément entre la deuxième et la troisième dimension. En effet, si on veut par exemple résoudre le système Ax=b, où A est la matrice creuse de Galerkin du problème du Laplacien, alors on peut montrer (voir [5] que les ordres de dépendance de la mémoire nécessaire et du temps de calculs (CPU) en fonction de N pour une méthode directe sont donnés par le tableau suivant :

Méthode directe	$2D$	$3D$
Mémoire	$\mathcal{O}(N^3)$	$\mathcal{O}(N^5)$
CPU	$\mathcal{O}(N^4)$	$\mathcal{O}(N^7)$

Cela revient à dire que si pour un problème donné on prend $N = 100$, nous aurions besoin d'une mémoire de $N^5 = 80Gbits$: c'est énorme donc peu efficace. Pour remédier à ce problème, il est nécessaire de passer par des méthodes itératives. En effet, la mémoire et le temps de calculs sont considérablement diminués par rapport aux méthodes directes lors du passage de la deuxième à la troisième dimension :

Méthode itérative	$2D$	$3D$
Mémoire	$\mathcal{O}(N^2)$	$\mathcal{O}(N^3)$
CPU	$\mathcal{O}(N^3)$	$\mathcal{O}(N^4)$

Donc pour $N = 100$ on devra fournir une mémoire de seulement 8 Megabits, ce qui motive à utiliser des méthodes itératives pour résoudre des problèmes de Stokes.

En voici quelques unes :

2.6.3 La méthode du gradient conjugué

En reprenant l'équation (11), on obtient une équation qui ne contient que $\mathbf{p}$ comme inconnue :

$$S\mathbf{p} = \mathbf{g}$$

où $S = BA^{-1}B^T$ est symétrique et définie positive par la proposition 2.5 et $\mathbf{g} = BA^{-1}\mathbf{F}$. On peut donc appliquer l'algorithme du gradient conjugué pour trouver $\mathbf{p}$. L'algorithme s'écrit de la manière suivante : Soit donné les matrices A et B, la fonction $\mathbf{F}$ ainsi qu'un vecteur initiale $\mathbf{p}_0$.

1. Résoudre $A\mathbf{z} = B^T\mathbf{p}_0$.
2. $S\mathbf{x}_0 = B\mathbf{z}$.
3. Résoudre $A\mathbf{e} = \mathbf{F}$.
4. $\mathbf{g} = B\mathbf{e}$.
5. $\mathbf{r}_0 = \mathbf{g} - S\mathbf{p}_0$.
6. $\mathbf{w}_1 = \mathbf{r}_0$.
7. $\alpha_1 = \mathbf{r}_0^T\mathbf{w}_1$.
8. $\mathbf{p}_1 = \mathbf{p}_0 + \alpha_1\mathbf{w}_1$.
9. Résoudre $A\mathbf{z} = B^T\mathbf{w}_1$.
10. $S\mathbf{w}_1 = B\mathbf{z}$.
11. Répéter pour $j = \{1, 2, \ldots\}$, jusqu'à convergence.
12. $\mathbf{r}_j = \mathbf{r}_{j-1} - \alpha_j S\mathbf{w}_j$.
13. $\beta_j = -\frac{\mathbf{r}_j^T S\mathbf{w}_j}{\mathbf{w}_j^T S\mathbf{w}_j}$.
14. $\mathbf{w}_{j+1} = \mathbf{r}_j + \beta_j\mathbf{w}_j$ (Direction de descente).
15. Résoudre $A\mathbf{z} = B^T\mathbf{w}_{j+1}$.
16. $S\mathbf{w}_{j+1} = B\mathbf{z}$.
17. $\alpha_{j+1} = \frac{\mathbf{r}_j^T\mathbf{w}_{j+1}}{\mathbf{w}_{j+1}^T S\mathbf{w}_{j+1}}$.
18. $\mathbf{p}_{j+1} = \mathbf{p}_j + \alpha_{j+1}\mathbf{w}_{j+1}$.

Nous trouvons ainsi la pression $\mathbf{p}$, et en appliquant à nouveau le gradient conjugué à la matrice A, le système linéaire

$$A\mathbf{u} = \mathbf{F} - B^T\mathbf{p}$$

nous donne $\mathbf{u}$.

2.6.4 La méthode GMRES

La méthode itérative de généralisation de Minimisation du Résidu (GMRES) a été developpée en 1986 par Yousef Saad et Martin H. Schultz. Elle permet de résoudre des systèmes linéaires du type $Ax = b$, où A est une matrice régulière, c'est-à-dire Ker A=0. C'est une généralisation de la méthode du gradient conjugué. Pour décrire cet algorithme, nous aurons besoin de la définition suivante :

Définition 2.8 (Espace de Krylov)
Soit $n \in \mathbb{N}$, $\boldsymbol{x}_0 \in \mathbb{R}^N$ et $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$. L'espace de Krylov d'ordre n du vecteur initial $\boldsymbol{r}_0$, noté $K_n(\boldsymbol{r}_0)$, est défini par

$$K_n(\boldsymbol{r}_0) := span\{\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^{n-1}\boldsymbol{r}_0\}.$$

On a $dim\ K_{n-1} \leq dim\ K_n \leq N$ pour $0 \leq n \leq N$. De plus $K_{n-1} \subset K_n$.

Description mathématique de la méthode GMRES On considère K_n l'espace de Krylov de $\mathbf{x}_0$. Alors il existe $\bar{k} \leq N$ tel que $dim K_n = n$ pour tout $n \leq \bar{k}$ et dim $K_n = \bar{k}$ pour tout $n \geq \bar{k}$. L'idée de la méthode GMRES est la suivante : on cherche $\mathbf{x}_n \in K_n$ tel que $\mathbf{x}_n = \mathbf{x}_{n-1} + \xi$, avec $\xi_n \in K_n$, tel que

$$\|\mathbf{b} - A(\mathbf{x}_{n-1} + \xi_n)\| \leq \|\mathbf{b} - A(\mathbf{x}_{n-1} + \xi)\|, \quad \forall \xi \in K_n.$$

C'est un problème de minimisation. Pour plus de détails voir [5]. L'algorithme est le suivant :

1. On pose $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$ et $\mathbf{v}_1 = \mathbf{r}_0$.
2. **for** $n = 1, 2, \ldots$
3. $B^n_{nl} = (\mathbf{v}_{n+1}, \mathbf{v}_{l+1})$, $l = 1, \ldots, n$
4. $d^{(n)}_l = (\mathbf{r}_{n-1}, \mathbf{v}_{l+1}), l = 1, \ldots, n$
5. Résoudre $B^n \mathbf{u}^n = \mathbf{d}^n$
6. $\mathbf{x}_n = \mathbf{x}_{n+1} + \sum_{j=1}^n u^n_j \mathbf{v}_j$
7. $\mathbf{r}_n = \mathbf{r}_{n-1} - \sum_{j=1}^n u^n_j \mathbf{v}_{j+1}$
8. continuer tant que $\|\mathbf{r}_n\| > \epsilon$
9. **end for**

2.6.5 La méthode d'Uzawa

La méthode d"Uzawa", du nom d'un économiste Japonais, utilise la méthode du gradient conjugué. Rappelons-nous, dans la proposition 2.5, nous avions ramené le problème matricielle de Stokes sous la forme d'un système d'équation :

$$\begin{array}{l} \text{Trouver } \mathbf{u} \in V_h, \, \mathbf{p} \in Q_h, \text{ tels que} \\ A\mathbf{u} = \mathbf{F} - B^T\mathbf{p} \\ BA^{-1}B^T\mathbf{p} = BA^{-1}\mathbf{F}. \end{array} \tag{15}$$

Comme la matrice A est symétrique et définie positive, on peut trouver $\mathbf{u}$ connaissant $\mathbf{p}$ par la méthode du gradient conjugué. En multipliant par B on obtient

$$B\mathbf{u} = BA^{-1}(\mathbf{F} - B^T\mathbf{p}).$$

L'idée de la méthode d'Uzawa est de trouver, pour un $\mathbf{p}_0$ donné, $\mathbf{p}$ et $\mathbf{u}$, de manière à ce que le résidu $B\mathbf{u}$ soit minimal pour satisfaire la condition $B\mathbf{u} = 0$. Plus précisément, pour tout itération $n \in \mathbb{N}$, si $\mathbf{p}_n$ est donné et $\alpha \in \mathbb{R}$ est une constante connue, alors on trouve $\mathbf{p}_{n+1}$ de la manière suivante :

1. Calculer $\mathbf{u}_n = A^{-1}(\mathbf{F} - B^T\mathbf{p}_n)$
2. Calculer $\mathbf{p}_{n+1} = \mathbf{p}_n + \alpha B\mathbf{u}_n$

On continue ainsi jusqu'à la convergence de la méthode, c'est-à-dire jusqu'à ce que le résidu

$$\frac{\|\mathbf{p}_{n+1} - \mathbf{p}_n\|}{\|\mathbf{p}_n\|} = \|\alpha B\mathbf{u}_n\|$$

soit suffisamment petit.

3 Le logiciel FreeFem

Freefem++ est un freeware développé par Frédéric Hecht, chercheur au Laboratoire Jacques-Louis Lions de l'Université Pierre et Marie Curie à Paris. Porté sous Windows, Unix et Mac OS, ce logiciel est dédié à la résolution d'équations aux dérivées partielles par des méthodes de type éléments finis. C'est un programme très complet et nous allons en utiliser qu'une toute petite partie pour ce travail. Pour en découvrir toute les facettes, le lecteur peut consulter l'excellent tutorial [**?**].

3.1 Pression à moyenne nulle

Moyenne nulle sur FreeFem Comment maintenant imposer sur FreeFem que nous cherchons une pression à moyenne nulle ? Nous allons pour cela utiliser une astuce numérique. Nous allons légèrement pertuber la condition d'incompressibilité : pour un $\epsilon > 0$ donné assez petit, nous supposerons que

$$\text{div}\mathbf{u} = \epsilon p.$$

Quitte à faire un relèvement, on peut supposer que $\mathbf{u} = 0$ sur $\partial\Omega$. La formulation faible de notre problème devient alors

$$\begin{array}{ll} \text{Trouver } \mathbf{u} \in V, p \in Q \text{ tel que} & \\ a(\mathbf{u},\mathbf{v}) + b(\mathbf{v},p) = F(\mathbf{v}) & \forall \mathbf{v} \in V \\ b(\mathbf{u},q) - \epsilon \int_\Omega pq = 0 & \forall q \in Q. \end{array} \tag{16}$$

ou bien sous forme matricielle :

$$\begin{pmatrix} A & B^T \\ B & -\epsilon I \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} \mathbf{F} \\ 0 \end{pmatrix}, \tag{17}$$

où A, B et $\mathbf{F}$ sont définis dans (14). Enfin, par le théorème de la divergence,

$$\epsilon \int_\Omega p = \int_\Omega \text{div}\mathbf{u} = \int_{\partial\Omega} \mathbf{u}\cdot\mathbf{n} = 0$$

Comme $\epsilon \neq 0$, on a comme désiré $\int_\Omega p = 0$, ce qui nous donne bien une pression à moyenne nulle. En pratique, la valeur de ϵ se situe entre 10^{-6} et 10^{-10}. Cela assure un bon comportement de la matrice de Galerkin car il n'y a pas de zéros sur la diagonale. De plus cette valeur de ϵ est suffisamment petite pour ne pas pertuber la solution du problème de Stokes.

3.2 Calcul des erreurs

Lorsque nous calculerons des problèmes avec solution exacte, nous vérifierons le résultat en calculant les erreurs relatives entre les solutions exactes $\mathbf{u}$ et p et leur approximation de Galerkin sur FreeFem $\mathbf{u}_h$ et p_h.
Pour la pression p, l'erreur absolue se fera pour la norme L^2 :

$$\|p - p_h\|_{L^2(\Omega)} = \left(\int_\Omega (p-p_h)^2\right)^{\frac{1}{2}}.$$

Comme $\mathbf{u}$ se trouve dans un sous-espace de Sobolev de type $(H^1(\Omega))^d$, nous allons estimer l'erreur absolue pour la norme correspondante, c'est-à-dire

$$\|\mathbf{u} - \mathbf{u}_h\|_{(H^1(\Omega))^d} = \left(\sum_{i=1}^d \|u_i - u_{i_h}\|^2_{(H^1(\Omega))}\right)^{\frac{1}{2}}$$

où

$$\|u_i - u_{i_h}\|^d_{H^1(\Omega)} = \|u_i - u_{i_h}\|_{L^2(\Omega)} + \sum_{j=1}^d \left\| \frac{\partial u_i}{\partial x_j} - \frac{\partial u_{i_h}}{\partial x_j} \right\|_{L^2(\Omega)}$$

Finalement, nos erreurs relatives sur $\mathbf{u}$ et p seront calculées de la manière suivante :

$$\mathrm{Er}(\mathbf{u}) = \frac{\|\mathbf{u} - \mathbf{u}_h\|_{(H^1(\Omega))^d}}{\|\mathbf{u}\|_{(H^1(\Omega))^d}}$$

et

$$\mathrm{Er(p)} = \frac{\|p - p_h\|_{L^2(\Omega)}}{\|p\|_{L^2(\Omega)}}.$$

3.3 Quelques mots sur les commandes des méthodes

Après avoir défini les espaces de Taylor-Hood, résoudre le problème de Stokes sur FreeFem revient à minimiser la forme variationnelle suivante :

$$\begin{aligned}&\mathrm{Varf}((\mathbf{u}_h, p_h), (\mathbf{v}_h, q_h)) = \\ &a(\mathbf{u}_h, \mathbf{v}_h) + b(\mathbf{v}_h, p_h) + b(\mathbf{u}_h, q_h) - F(\mathbf{v}_h) \quad \forall \mathbf{v}_h \in V_h, q_h \in Q_h.\end{aligned}$$

On définit les formes bilinéaires a et b et on leur associe leur matrice A et B respectives. Ensuite, on appelle une méthode de Résolution du système linéaire avec la commande suivante :

```
set(C, solver=UMFPACK) ;
```

où C est la matrice de Galerkin. Des paramètres peuvent intervenir dans cette commande, comme par exemple pour le gradient conjugué :

```
set(C, solver=GC, eps=1e-10, nbiter=100) ;
```

Ici "eps" et "nbiter" sont des tests d'arrêts : c'est-à-dire que l'algorithme stoppe dès que

$$\frac{\|\mathbf{p}_{n+1} - \mathbf{p}_n\|}{\|\mathbf{p}_n\|} < \epsilon$$

ou bien que le nombre d'itération atteint "nbiter". Malheureusement, on ne peut pas expliciter facilement des matrices inverses sur FreeFem. Cela veut dire que l'on ne peut pas créer par exemple la matrice $C = BA^{-1}B^T$. Néanmoins, si on veut appliquer la méthode du gradient conjugué pour une telle matrice, il suffit de construire une fonction, couramment appelée "dJ", qui nous renvoie le gradient ∇J de la fonctionnelle du problème. En effet, résoudre le système C$\mathbf{x}$=$\mathbf{b}$ revient à minimiser la fonctionnelle

$$J\mathbf{x} = \mathbf{b} \cdot \mathbf{x} - \frac{1}{2}\mathbf{x}^T C\mathbf{x}$$

dont le gradient vaut $\nabla(\mathbf{x}) = \mathbf{b} - C\mathbf{x}$. On utilise ensuite la commande

```
LinearCG(dJ,x,eps,nbiter);
```

pour résoudre le problème. Enfin, pour la méthode GMRES, on utilise

```
set(A, solver=GMRES, dimKrylov=50, eps=1e-6) ;
```

où A représente la matrice du système linéaire obtenu avec l'approximation de Galerkin. Le paramètre"dimKrylov" fixe la dimension de l'espace de Krylov. Cela signifie qu'après avoir effectué un nombre d'itérations correspondant à la dimension de cet espace, on recommence les itérations à 0 à partir du vecteur $\mathbf{x}_{dimKrylov}$. Cela permet d'alléger la mémoire de la machine : en effet, chaque itération de la méthode GMRES prend quelques mega octets de mémoire. Cette astuce enlève le calcul des premières itérations, ce qui libère de la mémoire pour la machine. Cependant, la convergence n'est pas garantie pour les matrices non-symétriques et indéfinies.

Le paramètre "eps" fixe la tolérance de la méthode. Cela signifie que la méthode s'arrête dès que $\|\mathbf{r}_n\| \leq \epsilon$, où $\mathbf{r}_n$ est le résidu défini dans la section 2.6.4

4 Résolution numérique du problème de Stokes

4.1 K-test avec solution exacte

4.1.1 2D

Soit Ω l'ouvert de $\mathbb{R}^2$ tel que $\partial\Omega$ soit le triangle de sommets $(0,0),(0,1),(1,1)$:

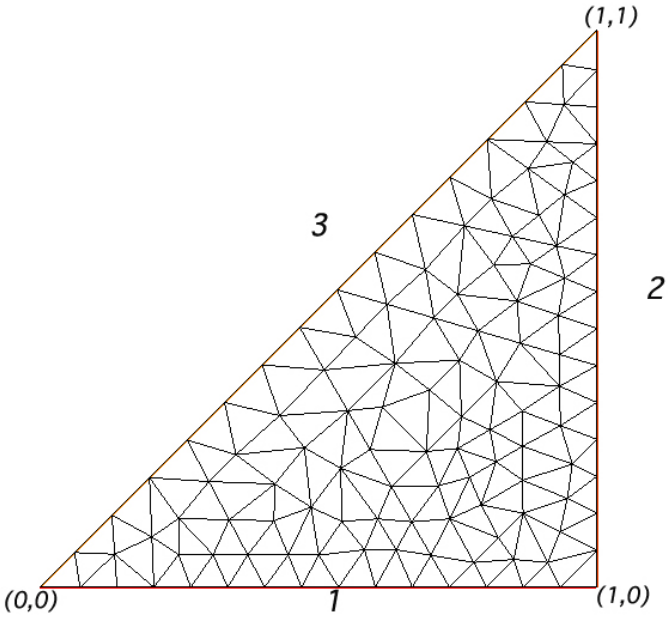


FIGURE 2 – Un maillage de Ω

On considère le problème de Stokes suivant : trouver $\mathbf{u} : \Omega \to \mathbb{R}^2$ et $p_\Omega \to \mathbb{R}$ vérifiant

$$\begin{aligned} -\Delta\mathbf{u} + \nabla p &= \begin{pmatrix} 1 \\ 1 \end{pmatrix} && \text{dans } \Omega, \\ \text{div } \mathbf{u} &= 0 && \text{dans } \Omega, \\ \mathbf{u} &= (x, 0) \text{ sur } 1 \\ \mathbf{u} &= (1, -y) \text{ sur } 2 \\ \mathbf{u} &= (x, -y) \text{ sur } 3 \end{aligned}$$

Ce problème admet la solution évidente $u = (x, -y)$ et $p = x + y$. Cependant, elle n'est pas forcément unique car la pression n'est définie qu'à une constante près puisque le problème de Stokes ne fait intervenir que les dérivées de p. Pour contrer cela, on peut demander que p se trouve dans l'espace des fonctions de carré sommable à moyenne nulle, comme cela a été fait dans la théorie. Mais en pratique, une telle condition sur la pression est difficile à implémenter dans le code (nous le ferons dans les cas en 3 dimensions pour éviter des éventuels problèmes de conditionnement de matrice). Pour ce test, nous procédons plus simplement en fixant p en un point du bord : nous demanderons que $p(0,0) = 0$.

Après avoir créé le maillage en créant $N = 1/h$ noeuds sur chaque côté du triangle et défini notre forme variationnelle (voir code en annexe), le logiciel FreeFem nous donne l'illustration suivante pour $N = 10$:

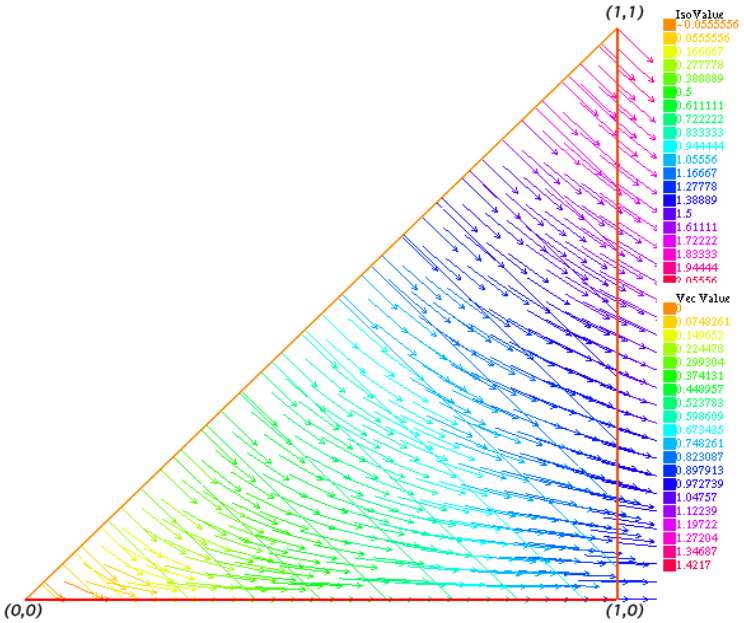


FIGURE 3 – Champ de vitesse et pression pour $N = 10$

L'isovaleur ("Iso Value", à droite en haut) nous donne les valeurs des lignes de niveau de la pression, tandis que le "Vec value" nous donne les valeurs de la norme du vecteur vitesse. Nous trouvons les erreurs suivantes pour $N = \frac{1}{h} = 10$:

h	Er($\mathbf{u}$)	Er(p)
0.1	1.44442e-29	5.49992e-15

Les valeurs de ces erreurs sont satisfaisantes pour un problème en 2D. Cela signifie que l'on a bien $\mathbf{u}_h = \mathbf{u}$, aux erreurs d'arrondi près.

4.1.2 $3D$

Soit Ω le cube $[0,1]^3 \in \mathbb{R}^3$:

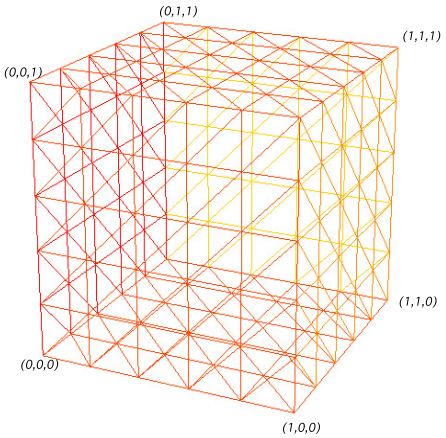


FIGURE 4 – Une triangulation du cube

On considère le problème de Stokes suivant : trouver $\mathbf{u} : \Omega \to \mathbb{R}^3$ et $p : \Omega \to \mathbb{R}$

à moyenne nulle vérifiant

$$\begin{aligned} -\Delta \mathbf{u} + \nabla p &= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} && \text{dans } \Omega, \\ \operatorname{div} \mathbf{u} &= 0 && \text{dans } \Omega, \\ \mathbf{u} &= (x, y, -2z) \text{ sur } \partial\Omega. \end{aligned}$$

Puisque

$$\int_\Omega (x + y + z) = \frac{3}{2},$$

ce problème admet l'unique solution évidente $\mathbf{u} = (x, y, -2z)$ et $p = x + y + z - \frac{3}{2}$

Résolution avec la méthode UMFPACK Nous allons dans un premier temps résoudre ce problème avec la méthode directe UMFPACK. Concernant l'affichage, il est nettement plus difficile de représenter la solution numérique d'un problème en 3 dimensions, car les formes géométriques sont plus complexes à construire, mais surtout, vu le grand nombre de vecteurs vitesses que produit un maillage fin en 2 dimensions, il serait de toute façon difficile d'y voir clair en 3 dimensions car nous aurions une trop forte accumulation de vecteurs. Afin de présenter néanmoins quelques illustrations, nous allons procéder de la façon suivante : puisque dans ce problème l'affichage de la pression consiste seulement en des surfaces de niveau de la fonction $p = x + y + z - \frac{3}{2}$, on peut faire appel à la commande suivante pour représenter p :

```
plot(p, wait=1, value=true, nbiso=10);
```

où le paramètre "nbiso" donne le nombre de surfaces de niveau désirées. Nous obtenons l'illustration suivante :

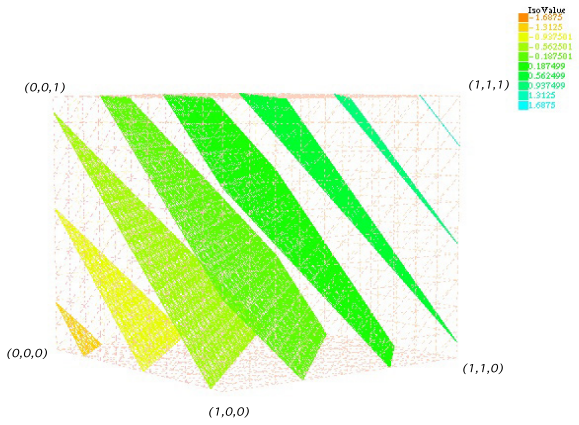


FIGURE 5 – Surface de niveau de p

La position des surfaces de niveau de $p = x + y + z - \frac{3}{2}$ est selon nos attentes, puisque que ce sont des plans parallèles orientés selon le vecteur normal $(1, 1, 1)$. Cette méthode devrait nous donner par conséquent des erreurs négligeables sur la pression.

Pour représenter la vitesse $\mathbf{u}$, nous allons nous ramener à des illustrations en deux dimensions de la manière suivante. L'idée va être de découper notre cube en tranches, et de regarder ce qui se passe à l'intérieur de chacune d'entre elles. Nous aurons ainsi une meilleure idée du comportement général de $\mathbf{u}$ dans le cube. Voici par exemple le champ de vitesse pour le cas où l'on a fixé $y = 0.5$:

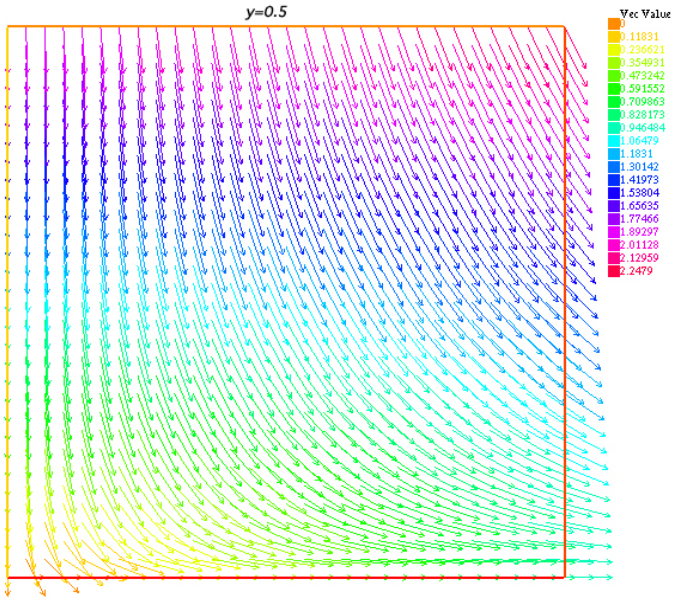


FIGURE 6 – Champ de vitesse et pression sur une coupure du cube, $N = 10$

Ce schéma semble conforme à la théorie, puisque les vecteurs ont un comportement semblable au champ $(x, y, -2z)$. Voyons maintenant si effectivement on tient là la bonne solution : nous allons calculer les erreurs relatives de $\mathbf{u}$ et p. Le tableau suivant résume les résultats pour quelques valeurs de $h = 1/N$:

N	$\mathrm{Er}(\mathbf{u})$	$\mathrm{Er}(p)$
2	$4.44019e^{-11}$	$2.83634e^{-6}$
5	$4.11337e^{-11}$	$1.98239e^{-6}$
10	$4.0791e^{-11}$	$1.6933e^{-6}$
12	$4.0762e^{-11}$	$1.5858e^{-7}$

Ces erreurs sont relativement négligeables pour un cas en 3 dimensions, ce qui nous indique comme précédemment que les erreurs restantes proviennent d'arrondi informatique.

Résolution avec GMRES Nous allons maintenant résoudre cet exemple avec la méthode GMRES, que nous pourrons comparer avec la méthode UMFPACK. Nous commençons avec $h = 0.1$. Tout d'abord nous prendrons une grande valeur pour la dimension de l'espace de Krylov. Cela nous donnera les résultats de convergence de la méthode GMRES de base. Avec $dimKrylov = 10'000$ et $eps = 1e - 6$, il faut 271 itérations pour que l'algorithme s'arrête et nous trouvons que les erreurs sur $\mathbf{u}$ et p sont de l'ordre de $\mathcal{O}(10^{-2})$. Nous remarquons que la convergence est ici nettement moins bonne qu'avec la méthode UMFPACK et qu'en plus cela prend bien plus de temps. Néanmoins, pour améliorer la convergence, on peut faire varier la tolérance :

ϵ	E($\mathbf{u}$)	E($\mathbf{p}$)	Iter	CPU
1e-5	0.0247832	0.317013	132	24.17
1e-6	0.0043736	0.058333	271	33.80
1e-7	0.0007862	0.013457	513	51.39
1e-8	0.0001837	0.004589	899	98.96
1e-9	9.8012e-06	0.0001936	1397	180.53
1e-10	7.8242e-07	1.2873e-05	1654	243.17
1e-11	4.8879e-08	6.7513e-07	1888	301.64
1e-12	4.9027e-09	6.3918e-08	1996	322.21
1e-13	2.9526e-10	5.2090e-09	2302	458.03

TABLE 1 – Méthode GMRES pour N=10

En observant ce tableau, on remarque que si notre tolérance est très petite ($1e-13$), nous obtenons des erreurs du même ordre que celles obtenues avec la méthode UMFPACK. Cependant, pour arriver à un tel résultat, un CPU de 458 secondes et quelques 2302 itérations auront été nécessaires, alors que 43 secondes suffisaient avec UMFPACK. Donc, pour des valeurs de h relativement grande, la méthode GMRES reste tout de même coûteuse en CPU et en nombre d'itérations. Pour les petits cas, on conclut donc que la méthode UMFPACK est plus avantageuse. Nous fixons ensuite la tolérance $\epsilon = 10e^{-10}$, qui est une valeur suffisamment basse pour produire des résultats satisfaisants. Que se passe-t-il lorsque nous faisons varier N sur cette valeur de ϵ ?

N	E($\mathbf{u}$)	E($\mathbf{p}$)	Iter	CPU
1	4.15652e-11	0.316228	28	0.02
2	1.35283e-09	3.00101e-08	50	0.13
3	1.34202e-08	2.53279e-07	210	0.64
4	1.43575e-07 5	3.40425e-06	435	2.56
5	1.29009e-07	2.16605e-06	708	7.94
6	2.16738e-07	3.49953e-06	838	18.61
7	2.20852e-07	2.98913e-06	1077	41.68
8	2.29684e-07	2.89508e-06	1201	78.13
9	7.90118e-07	1.38093e-05	1499	145.95
10	7.8242e-07	1.2873e-05	1654	231.38
11	9.89357e-07	1.69368e-05	1888	361.95

TABLE 2 – Méthode GMRES pour $\epsilon = 10e - 10$

Le nombre d'itérations augmente linéairement tandis que le CPU semble être quadratique. Les erreurs semblent plutôt se stabiliser voire d'augmenter en fonction de N. Cela peut surprendre puisque que l'on s'attend tout de même à ce que nos erreurs diminuent à mesure que N augmente. L'explication est qu'ici nous sommes dans un K-test à solution exacte évidente et l'ordinateur calcule $\mathbf{u}_h = \mathbf{u}$. Les erreurs calculées ici correspondent en fait seulement aux erreurs d'arrondi de l'ordinateur qui ne garde que les 6 premiers chiffres significatifs, ce qui explique une stabilisation de l'erreur à environ $10 \times e^{-6}$. Donc finalement, GMRES ou UMFPACK ? Malgrès le coût de GMRES, je peux au moins tester mon problème sur des valeurs de N allant jusqu'à 20, alors que la mémoire de mon ordinateur m'empêche de dépasser N=13 avec UMFPACK. Ainsi, les méthodes directes sont très efficaces sur des gros maillages, mais pour des problèmes plus importants, il est nécessaire de passer à des méthodes itératives.

Remarque 4.1
Puisque que nous travaillons avec des éléments finis $\mathbb{P}_2 - \mathbb{P}_1$, il aurait été intéressant de choisir un champ vectoriel vitesse qui soit de degré 2. A ce effet, on peut considérer le problème suivant : trouver $\boldsymbol{u} : \Omega \to \mathbb{R}^3$ et $p : \Omega \to \mathbb{R}$ à moyenne nulle

vérifiant :

$$\begin{aligned} -\Delta \boldsymbol{u} + \nabla p &= \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix} && \text{dans } \Omega, \\ \text{div } \boldsymbol{u} &= 0 && \text{dans } \Omega, \\ \boldsymbol{u} &= (y^2, z^2, x^2) \text{ sur } \partial\Omega. \end{aligned}$$

Ce problème admet l'unique solution évidente $u = (y^2, z^2, x^2)$ *et* $p = x+y+z-3/2$. *On obtient dans sa résolution numérique avec UMFPACK ou GMRES des erreurs du même ordre que ce problème, ce qui nous confirme que l'ordinateur calcule bien* $\boldsymbol{u}_h = \boldsymbol{u}$.

4.2 Application dans un écoulement de Poiseuille

Lorsqu'un fluide parcourt un tube infiniment long de section circulaire de rayon constant R ou s'écoule entre deux plaques parallèles distante de h, on parle alors d'écoulement de Poiseuille.

4.2.1 Solution exacte d'un écoulement de Poiseuille

Proposition 4.2
La vitesse du fluide $\boldsymbol{u}$ dans le cas d'un tube d'axe orienté selon z et de section circulaire est donnée par

$$\boldsymbol{u}(r,\theta,z) = \boldsymbol{u}(r) = \left(0, \quad 0, \quad \tfrac{1}{4\nu}\tfrac{dp}{dz}(R^2 - r^2)\right)$$

où R et le rayon de la section du tube, ν est le coefficient de viscocité du fluide et p est la pression agissant à l'intérieur du tube. La vitesse est parallèle à l'axe du tube.

Démonstration. Par des arguments de symétries, la vitesse dans le fluide ne varie ni en θ ni en z. Par conséquent, les seules forces de viscocité sont selon z et transmises radialement. Cette force vaut (voir [5]) :

$$F_{\text{bord}} = -2\pi r \Delta z \nu \frac{du_z}{dr}.$$

Par symétrie également, la variation de la pression est constante le long de l'axe z. Donc

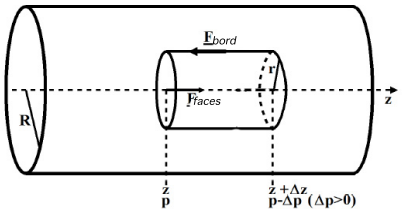
$$\frac{dp}{dz} = C.$$

Les forces dûes à la pression sur les deux faces circulaires du cylindre ont une résultante égale à

$$F_{\text{faces}} = \pi r^2 \Delta z \frac{dp}{dz}.$$

Comme l'écoulement est stationnaire, ces deux forces s'équilibrent et on obtient que le gradient de vitesse est linéaire en r

$$\frac{du_z}{dr} = \Delta p \pi r^2 = \frac{r}{2\nu}\frac{dp}{dz}.$$



Autrement dit, le champ de vitesse est parabolique :

$$u_z(r) = C + \frac{r^2}{4\nu}\frac{dp}{dz}.$$

Puisque que l'on a imposé une vitesse nulle sur les bords du tuyau ($u_z(R) = 0$), on obtient finalement

$$u_z(r) = \frac{R^2}{4\nu}\frac{dp}{dz}\left(1 - \frac{r^2}{R^2}\right).$$

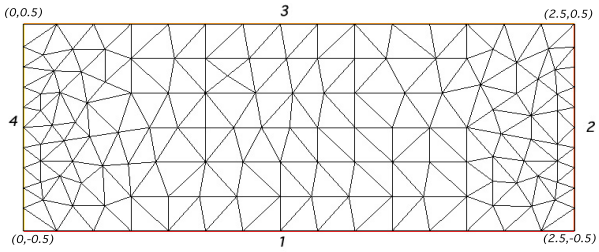
Ceci nous montre que la vitesse est plus importante au centre du conduit que sur les extrémités. □

Remarque 4.3
On peut écrire la formule ci-dessus en fonction du débit D. En effet on a la relation

$$\frac{dp}{dz} = \frac{8\nu}{\pi R^4} D.$$

4.2.2 $2D$

Soit Ω un ouvert de $\mathbb{R}^2$ dont le bord est le rectangle de sommet $(0, -1/2)$, $(2.5, -1/2)$, $(2.5, 1/2)$ et $(0, 1/2)$:



On considère le problème de Stokes : trouver $\mathbf{u} : \Omega \to \mathbb{R}^2$ et $p : \Omega \to \mathbb{R}$

$$\begin{aligned} -\Delta \mathbf{u} + \nabla p &= \begin{pmatrix} 0 \\ 0 \end{pmatrix} && \text{dans } \Omega, \\ \text{div } \mathbf{u} &= 0 && \text{dans } \Omega, \\ \mathbf{u} &= (0,0) \text{ sur } 1,3 \\ \mathbf{u} &= (1/4 - y^2, 0) \text{ sur } 4. \end{aligned}$$

Ce problème admet la solution exacte $u(x, y) = (1/4 - y^2, 0)$ et $p = -2x$ du moment que l'on impose la condition initiale $p(0, 0) = 0$. On peut voir ce problème comme un tube de Poiseuille dans $\mathbb{R}^3$ d'axe selon x et de rayon 2.5 que l'on aurait coupé avec un plan contenant l'axe du cylindre. Cela signifie que l'on a fixé l'angle θ dans la partie théorique précédente, ce qui nous a ramené à un problème en 2 dimensions. Sa résolution sur FreeFem est similaire exemples précédents. Pour le maillage de Ω, nous divisons les côtés 2 et 4 du rectangles en N parties et les côtés 1 et 3 en $\frac{3}{2}N$ parties. On obtient l'illustration suivante :

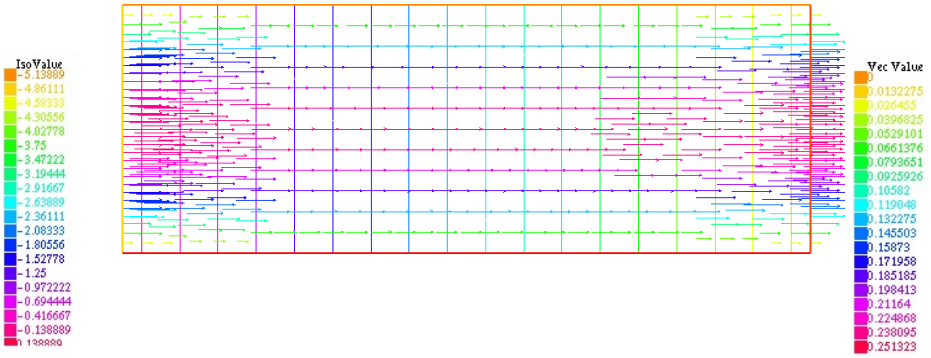


FIGURE 7 – Isovaleurs de p et champs de vitesse $\mathbf{u}$ pour N=10.

Cela nous montre que la vitesse sera maximale lorsque le fluide se trouve au centre du tube et nulle sur les bords, comme la théorie le prévoyait. Les erreurs obtenues sont du même ordre que celles du K-test avec le triangle en 2 dimensions :

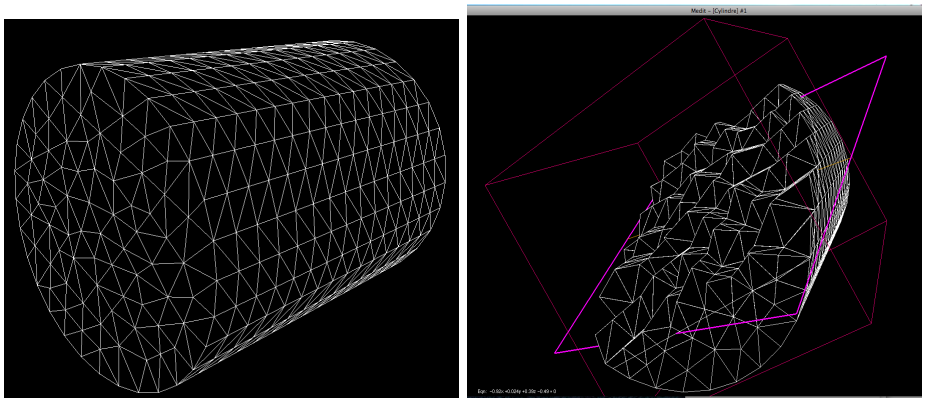
N	Er($\mathbf{u}$)	Er(p)
10	$9.19738e^{-30}$	$5.49243e^{-16}$

4.2.3 $3D$

Nous considérons le problème de Stokes suivant : trouver $\mathbf{u} : \Omega \to \mathbb{R}^3$ et $p : \Omega \to \mathbb{R}$ à moyenne nulle vérifiants :

$$\begin{aligned} -\Delta \mathbf{u} + \nabla p &= \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} && \text{dans } \Omega, \\ \text{div } \mathbf{u} &= 0 && \text{dans } \Omega, \\ \mathbf{u} &= (0,0,0) \text{ sur } 1 \\ \mathbf{u} &= (0,0,1-x^2-y^2) \text{ sur } 0. \end{aligned}$$

Le domaine Ω est un tube de rayon 1 et de longueur 2.5. Si N est un nombre entier, alors notre maillage dépendra de N de la manière suivante : nous diviserons le bord de la base du cylindre en $2N$ sommets (Bord pour z=0, le numéro 0 dans la donnée ci-dessus) et l'axe sera partitionné en N sommets. Les figures ci-dessous nous montrent un cylindre pour $N = 15$ ainsi que les tétraèdres à intérieur.



Ce problème admet la solution exacte $\mathbf{u}(x,y,z) = (0,0,1-x^2-y^2)$ et $p(x,y,z) = -4z + 5$. Pour $N = 10$, la résolution du problème nous donne bien des plans parallèles, correspondant aux surfaces de niveau de p :

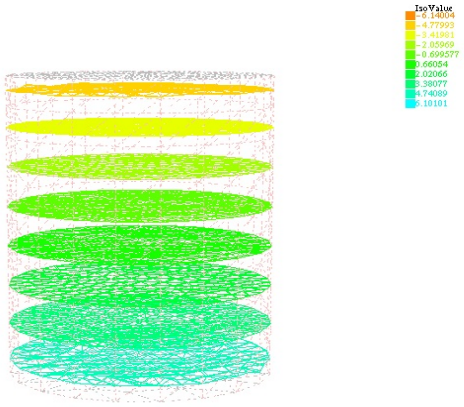


FIGURE 8 – Isovaleurs de p pour $N = 10$

Pour afficher la vitesse, nous procédons de manière similaire à notre problème du cube : nous coupons notre cylindre perpendiculairement à son axe afin d'obtenir des informations sur le fluide à un endroit précis dans le tuyau :

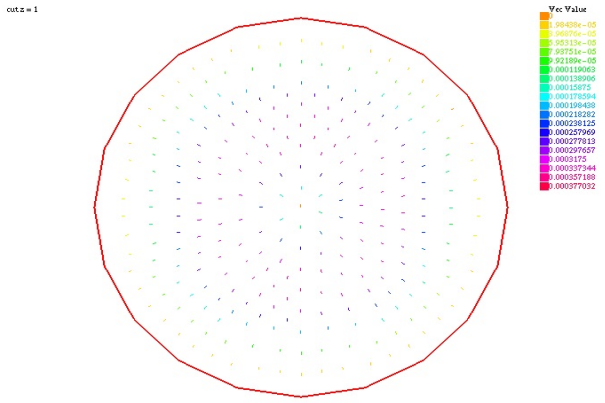


FIGURE 9 – Plan dans le tuyau à l'endroit $z = 1$ pour $N = 10$

Comme la théorie le prévoyait, la vitesse est indépendante de z et donc les informations sur cette coupure sont suffisantes pour la vitesse. Remarquons néanmoins que les vecteurs sont ici représentés par des points, puisque que nous n'en voyons que l'origine. L'ordinateur calcule donc la norme de ces points, et non la norme de tout le vecteur, car dans cette coupure, la variable z est fixée. Cela explique les erreurs des valeurs affichées.

Résolution avec UMFPACK Toutes les illustrations ci-dessus ont été obtenues par la méthode UMPFACK. Similairement au problème du cube, il est difficile d'observer des résultats pour des valeurs de N plus grande, la mémoire disponible étant limitée pour les calculs que demande cette méthode en 3 dimensions, ce qui motive toujours le passage à des méthodes itératives.

Résolution avec GMRES Nous allons refaire une petite analyse de la méthode GMRES sur ce problème. Nous fixons la tolérance à eps= $10 \times e^{-10}$ et nous supposerons toujours que dimKrylov=10'000. Voici les erreurs produites par la méthode, pour N allant de 1 à 19 :

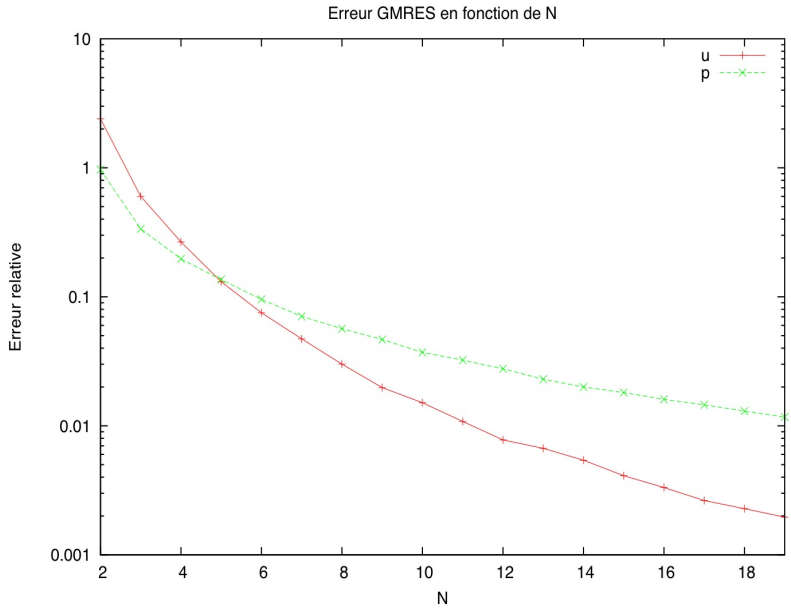
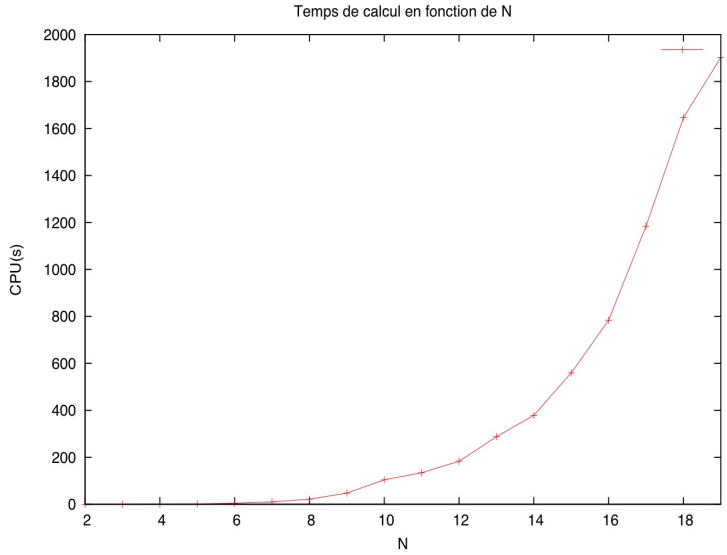


FIGURE 10 – Erreur GMRES

Ce résultat est meilleur que le précédent sur le cube, puisqu'ici les erreurs diminuent clairement en fonction de N. Ceci s'explique par le fait que la solution est parabolique et donc un peu moins "évidente" que celle du cube. L'ordinateur ne calcule donc pas directement la solution et les erreurs restantes ne sont pas seulement celles d'arrondis. Mais ces erreurs sont par conséquent beaucoup plus "grosses" que dans le cas où l'ordinateur calculait directement la solution exacte, comme dans l'exemple précédent. Pourtant, une solution exacte parabolique dans le cube produit des erreurs du même ordre qu'une solution linéaire. Pourquoi a-t-on ici des erreurs si grossières ? Cela vient certainement des conditions de bords et du fait qu'il est possible qu'on ait des erreurs d'interpolation dans le maillage, car il est moins évident d'approcher un cylindre qu'un cube avec des triangles ! Le nombre d'itérations et le temps de calculs augmentent quant à eux fortement en fonction de N. Pour N=19 par exemple, l'ordinateur met pratiquement une demi-heure pour calculer ce petit problème !



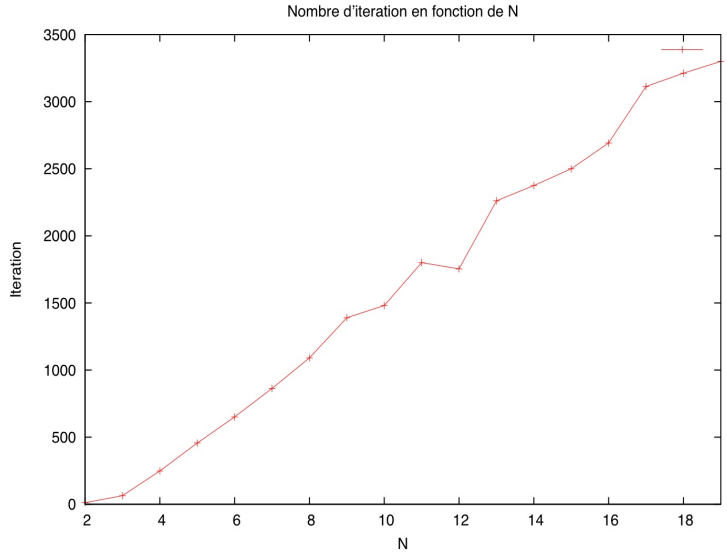


FIGURE 11 – Méthode GMRES : nombre d'itération et CPU en fonction de N

On remarque que le nombre d'itérations croît linéairement en fonction de N, alors que le CPU est un $\mathcal{O}(N^3)$.

Remarque 4.4
En 3 dimension, on pouvait s'attendre à une croissance $\mathcal{O}(N^3)$ pour les nombres d'itération et à un $\mathcal{O}(N^4)$ pour le CPU. Pour obtenir de meilleurs résultats, il faudrait pour cela mesurer ce problème sur des valeurs plus élevées de N. Malheureusement, l'ordinateur avec lequel j'ai fait ces mesures ne possède pas assez de mémoire vive pour cela (2Go).

4.3 Cavité carrée

Soit Ω l'ouvert de $\mathbb{R}^2$ tel que $\partial\Omega$ soit le carré de sommet $(0,0),(1,0),(1,1),(0,1)$. On considère le problème de Stokes suivant : trouver $\mathbf{u} : \Omega \to \mathbb{R}^2$ et $p : \Omega \to \mathbb{R}$ vérifiants

$$\begin{aligned} -\Delta\mathbf{u} + \nabla p &= \begin{pmatrix} 0 \\ 0 \end{pmatrix} && \text{dans } \Omega, \\ \text{div } \mathbf{u} &= 0 && \text{dans } \Omega, \\ \mathbf{u} &= (1,0) \text{ sur } 3 \\ \mathbf{u} &= (0,0) \text{ sur } 1,2,4 \end{aligned}$$

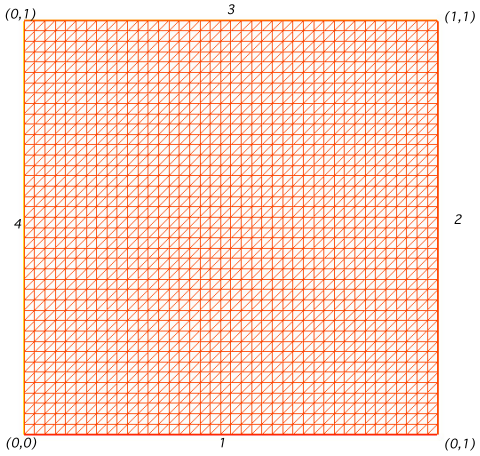


FIGURE 12 – Une triangulation du carré

Cet exemple, dit de la "cavité carrée", est un test standard. On peut le voir comme un cube plein d'eau, où le liquide coule en surface avec une vitesse unitaire selon un axe.

4.3.1 Résolution avec Uzawa

En se donnant $\mathbf{p}_0$ et la constante α, nous construisons sur FreeFem une fonction "Bu" qui nous renvoie, à l'aide du gradient conjugué, le résidu $\alpha B\mathbf{u}_0$. Ensuite, pour calculer

$$\mathbf{p}_{n+1} = \mathbf{p}_n + \alpha B\mathbf{u}_n,$$

nous pouvons utiliser, grâce à notre fonction "Bu", la commande

```
LinearCG(Bu,p[], eps=1.e-10,nbiter=1000);
```

Si l'algorithme converge après un certain nombre d'itération, il nous donne notre couple de solutions $\mathbf{u}$ et $\mathbf{p}$. Avec $N = 140$ et $\alpha = 1$, nous obtenons l'illustration suivante :

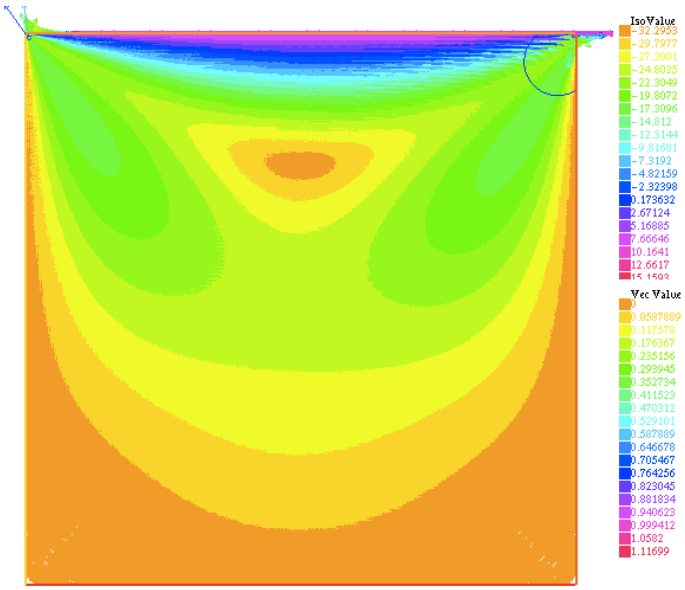


FIGURE 13 – Champ de vitesse $\mathbf{u}$ et pression p pour $N = 140$

Ceci nous montre que le liquide va créer du remous à faible profondeur, alors qu'il reste calme sur les trois autres côtés de la boîte.

Etude du paramètre α Nous allons nous intéresser maintenant à l'importance de la constante α dans l'algorithme d'Uzawa. Pour quelques N fixé, nous allons estimer le α optimal permettant un nombre d'itération et un CPU qui soient le plus petit possible. Pour N valant respectivement 5 et 50 et en prenant le coefficicent de viscocité cinématique $\mu = 0.01$ (c'est la valeur pour l'eau), voici un tableau nous donnant le nombre d'itération de la méthode d'Uzawa ainsi que le CPU, pour quelques valeurs de α :

Alpha	N=5		N=50	
	Iter	CPU	Iter	CPU
10e-4	47	0.11	?	?
10e-1	43	0.1	61	91.56
1	40	0.1	61	91.44
10e3	40	0.1	61	91.79
10e7	40	0.1	61	91.55
10e10	95	0.2	61	91.97
10e12	48	0.11	61	91.41
10e13	57	0.12	82	120.61
10e14	54	0.13	61	90.68
10e15	212	0.4	62	92.36
10e18	154	0.3	141	207.39
10e20	154	0.3	141	207.90

TABLE 3 – Nombre d'itération et CPU en fonction du paramètre α

Nous remarquons que la valeur optimale de α dépend peu de N. En effet, le nombre d'itération et le CPU ne sont pas modifiés pour α valant entre 1 et 10^9. C'est un intervalle énorme. Pourtant, à partir de $\alpha = 10^{10}$, on a un "saut", avant une restabilisation à partir de $\alpha = 10^{18}$. Si on prend un α trop petit, l'algorithme s'arrête au bout d'un certain nombre d'itération, car les arrondis de l'ordinateur provoquent une division par zéro dans les calculs. C'est ce qui se passe par exemple

dans le cas où $\alpha = 10e - 4$ et $N = 50$. Pour conclure, un α valant l'unité est une valeur sûre !

4.3.2 Résolution avec le gradient conjugué

On suit la méthode expliquée dans les sections 2.6.4 et 3.3 en créant une fonction qui nous renvoie, à partir d'un vecteur initiale $\mathbf{p}_0$, la direction de descente $\mathbf{r}_0 = BA^{-1}F - BA^{-1}B^T\mathbf{p}_0$. On trouve ensuite $\mathbf{u}$ en résolvant le système

$$A\mathbf{u} = F - B^T\mathbf{p}.$$

4.3.3 Comparaison des deux méthodes

Pour des N variant entre 5 et 140, j'ai testé la méthode d'Uzawa avec celle du gradient conjugué. Voici ce que j'obtiens :

N	Uzawa		GC	
	Iter	CPU	Iter	CPU
5	40	0.11	27	0.17
10	60	0.8	53	1.40
20	60	6.06	60	11.98
30	60	20.15	60	40.22
40	61	47.51	61	95.34
50	61	91.69	61	183.98
60	61	162.39	61	331.04
70	61	256.83	61	512.32
80	61	380.47	61	762.55
90	61	539.53	61	1085.44
100	61	734.96	61	1474.27
110	61	970.95	61	1963.19
120	61	1256.67	61	2596.82
130	61	1596.66	61	3225.66
140	61	2014.46	61	4053.70

TABLE 4 – Méthodes d'Uzawa et du Gradient Conjugué pour différentes valeurs de N

On remarque qu'à partir d'une valeur de N raisonnable (N=40), le nombre d'itération des deux algorithmes ne dépend plus de N, et est égal pour les deux méthodes. Le CPU augmente plus fortement avec le gradient conjugué que Uzawa, car l'algorithme du gradient conjugué fait beaucoup plus de calculs (On utilise en tout trois fois l'algorithme du gradient conjugué dans une itération, seulement deux avec Uzawa). Dans ce problème, on conclut donc qu'il est inutile de le résoudre avec la méthode du gradient conjugué, à moins de vouloir perdre du temps !

5 Conclusion

Pour résoudre numériquement le problème de Stokes en $\mathbb{P}_2 - \mathbb{P}_1$, il vaut mieux passer par des méthodes itératives que directes, à moins de se trouver dans un gros maillage. Parmis ces méthodes, GMRES est certainement la plus utile avec FreeFem++, car elle permet de résoudre des systèmes où la matrice de Galerkin n'est pas forcément symétrique et définie positive, elle converge relativement bien si on prend un test d'arrêt suffisamment petit et elle n'est pas difficile à implémenter dans FreeFem++. Dans nos cas, le Gradient Conjugué et Uzawa sont un peu plus coûteux pour le problème de Stokes, car l'algorithme correspondant engendre un temps de calculs assez élevé.

Références

[1] A. Quarteroni *Numerical Approximation of Partial Differential Equations.* Springer, New York, 1997.

[2] P. Girault, P-A. Raviart *Finite Element Methods for Navier-Stokes Equations.* Springer, New York, 1986.

[3] M. Discacciati *Numerical Approximation of Partial Differential Equations.* Cour Master MA, EPFL, Lausanne, 2010.

[4] M.Q.Tran *Physique Générale III.* Cour Bachelor MA, EPFL, Lausanne, 2008

[5] M.Picasso *Calcul Scientifique Avancé.* Cour Master MA, EPFL, Lausanne, 2010

[6] J. Rappaz, M.Picasso *Introduction à l'analyse numérique.* COEN, EPFL, Lausanne, 2004

[7] F. Hecht *FreeFem++.* Third Edition, Laboratoire Lion, Paris, 2009

[8] A. Adams *Sobolev Spaces.* Elsevier, London, 2003.

[9] J. Rochat *Les espaces de Sobolev* Travail de semestre Bachelor MA, EPFL, Lausanne, 2009

6 Annexe : code FreeFem des tests

6.1 K-test sur Solution Exact

6.1.1 K-test du triangle

```
// Paramétrisation du bord du domaine
border a(t=0,1){x=t;y=0;label=1;}
border b(t=0,1){x=1;y=t;label=2;}
border c(t=0,1){x=1-t;y=1-t;label=3;}

// Construction du domaine avec N=10
mesh Th =buildmesh(a(10)+b(10)+c(10));

// Affichage du domaine et son maillage
plot(Th);

// Définition des espaces d'éléments finis et des fonctions dans ces espaces
fespace Vh(Th,P2);
Vh u1,u2,v1,v2;
fespace Ph(Th,P1);
Ph p,q;

// Macro pour simplifier l'écriture de la divergence et du gradient
macro div(u,v) (dx(u)+dy(v))//
macro grad(u) [dx(u),dy(u)]//

// Formulation Variationnelle du problème
solve Stokes([u1,u2,p],[v1,v2,q])=
int2d(Th)(grad(u1)'*grad(v1)+grad(u2)'*grad(v2)
-p*(div(v1,v2)) -q*(div(u1,u2))) -int2d(Th)(v1+v2)

//Condition de Bord
//Pression en (0,0)
+on(3,p=x+y)
//U
+on(3,u1=x,u2=-y)
+on(1,u1=x,u2=0)
+on(2,u1=1,u2=-y);

// Résolution numérique du problème

Stokes;

// Définition des erreurs
real eru;
real ere;
real eruf;
real erp;
real epe;
real erpf;

//Erreur absolue de u-uh
eru=(int2d(Th)((u1-x)^2+(dx(u1)-1)^2+(dy(u1)^2)))^2
+(int2d(Th)((u2+y)^2+(dx(u2))^2+(dy(u2)+1)^2))^2;
//Norme de u
ere=int2d(Th)((x^2+1)+(y^2+1));
//Erreur relative de u-uh
```

```
eruf=sqrt(eru)/sqrt(ere);

//Erreur absolue de p-ph
erp=int2d(Th)((p-(x+y))^2);
//Norme de p
epe=int2d(Th)((x+y)^2);
//Erreur relative de p-ph
erpf=sqrt(erp)/sqrt(epe);

// Affichage du calcul des erreurs relatives
cout <<"erreur sur u:"<< eruf << endl;
cout << "erreur sur p:"<< erpf << endl;

//Affichage de u et p
plot([u1,u2],value=true,p,wait=1);
```

6.1.2 K-test du cube

```
load "msh3"
int N;
N=12;
// Construction du cube unitaire
// on fait varier sur l'axe 0 <=z<=1 un carré de côté N
mesh T2d=square(N,N);
mesh3 Th=buildlayers(T2d,N, zbound=[0,1]);

//Affichage du cube
//plot(Th);

//Définition de l'espace d'élément fini de Taylor Hood
fespace VVh(Th,[P2,P2,P2,P1]) ;

//Macro pour le gradient et la divergence
macro Grad(u) [dx(u),dy(u),dz(u)] // EOM
macro div(u1,u2,u3) (dx(u1)+dy(u2)+dz(u3)) // EOM

//Formulation Variationnelle
varf Stokes([u1,u2,u3,p],[v1,v2,v3,q]) =
int3d(Th)(
Grad(u1)'*Grad(v1) + Grad(u2)'*Grad(v2) + Grad(u3)'*Grad(v3)
- div(u1,u2,u3)*q - div(v1,v2,v3)*p-1e-10*q*p)
+int3d(Th)(v1+v2+v3)
//+on(1,p=x+y+z)
+on(1,2,3,4,5,6,u1=x,u2=y,u3=-2*z);

//Création de la matrice de Galerkin à partir de la forme variationnelle
matrix A=Stokes(VVh,VVh) ;

//Appel de la méthode de résolution
set(A,solver=UMFPACK) ;

//Résolution du système
real[int] b= Stokes(0,VVh) ;
VVh [u1,u2,u3,p] ;
u1[]= A^-1 * b ;

//Coupure du cube à un endroit y
fespace V2d(T2d,P2) ;
V2d ux= u1(x,0.5,y) ;
```

```
V2d uz= u3(x,0.5,y) ;
V2d p2= p(x,0.5,y) ;

//Affichage de la coupure
plot([ux,uz],value=true,cmm=" cut y = 1") ;
//Affichage des lignes de niveau de la pression
plot(p,wait=1,value=true,nbiso=10);

//Definition des erreurs
real eru; //erreur sur la viscocité
real eru2;
real eruf;
real erp; //erreur sur la pression
real erp2;
real erpf;

//Erreur absolue de u
eru=int3d(Th)((u1-x)^2+(dx(u1)-1)^2+dy(u1)^2+dz(u1)^2)
   +int3d(Th)((u2-y)^2+dx(u2)^2+(dy(u2)-1)^2+dz(u2)^2)
   +int3d(Th)((u3+2*z)^2+dx(u3)^2+dy(u3)^2+(dz(u3)+2)^2);
// Norme de u
eru2=int3d(Th)(x^2+1+y^2+1+4*z^2+4);
// Erreur relative de u
eruf=sqrt(eru/eru2);

// Erreur absolue de p
erp=int3d(Th)((p-x-y-z+1.5)^2);
// Norme de p
erp2=int3d(Th)((x+y+z-1.5)^2);
//Erreur relative de p
erpf=sqrt(erp/erp2);

//affichage des erreurs relatives
cout <<"erreur sur u:"<< eruf << endl;
cout << "erreur sur p:"<< erpf << endl;
```

Remarque 6.1
Pour résoudre le problème ci-dessus avec la méthode GMRES, il suffit de remplacer

set(A, solver=UMFPACK) ;

par

set(A, solver=GMRES, dimKrylov=50, eps=1e-10) ;

6.2 Ecoulement de Poiseuille

6.2.1 2D

```
// Paramétrisation du bord du domaine
border a(t=0,1){x=t;y=0;label=1;}
border b(t=0,1){x=1;y=t;label=2;}
border c(t=0,1){x=1-t;y=1-t;label=3;}

// Construction du domaine
mesh Th =buildmesh(a(10)+b(10)+c(10));

// Définition des espaces d'éléments finis et des fonctions dans ces espaces
fespace Vh(Th,P2);
Vh u1,u2,v1,v2;
```

```
fespace Ph(Th,P1);
Ph p,q;

// Macro pour simplifier l'écriture de la divergence
macro div(u,v) (dx(u)+dy(v))//

// Problème faible
//Formulation Faible
solve Stokes([u1,u2,p],[v1,v2,q])
= int2d(Th)((dx(u1)*dx(v1)+dx(u2)*dx(v2)+dy(u1)*dy(v1)+dy(u2)*dy(v2)
-p*(div(v1,v2)) -q*(div(u1,u2)))) -int2d(Th)(v1+v2)
//Condition de Bord
+on(3,p=x+y)
+on(3,u1=x,u2=-y)
+on(1,u1=x,u2=0)
+on(2,u1=1,u2=-y);

// Résolution numérique du problème

Stokes;

// Définition des erreurs
int eru;
int erp;
eru= (int2d(Th)((u1+x)^2+(dx(u1)+1)^2+(dy(u1)^2)))^2+
        (int2d(Th)((u2-y)^2+(dx(u2))^2+(dy(u2)-1)^2))^2;
erp=int2d(Th)((p-(x+y))^2);

// Affichage du calcul des erreurs
cout <<"erreur sur u:"<< eru << endl;
cout << "erreur sur p:"<< erp << endl;

//Affichage des isoclines de u et p
plot([u1,u2],p,wait=1);
```

6.2.2 3D

```
load "msh3"
load "medit"
int N;
int Nu;
Nu=1/2;
//On va chercher une pression tel que sa dérivée par rapport à z
// soit constante, disons 1.
N=20;
// Paramétrisation du cylindre
border a(t=0,2*pi){ x=cos(t); y=sin(t);label=1;};
mesh T2d=buildmesh(a(2*N));
//plot(T2d);
mesh3 Th=buildlayers(T2d,N, zbound=[0,2.5]);
medit("Cylindre",Th);
exec("rm Cylindre.bb Cylindre.faces Cylindre.points");
// Element fini de Taylor Hood
fespace VVh(Th,[P2,P2,P2,P1]) ;
//Macri pour le gradient et la divergence
macro Grad(u) [dx(u),dy(u),dz(u)] // EOM
macro div(u1,u2,u3) (dx(u1)+dy(u2)+dz(u3)) // EOM
//Définition de la forme variationnelle
varf Stokes([u1,u2,u3,p],[v1,v2,v3,q]) =
```

```
int3d(Th)(
Grad(u1)'*Grad(v1) + Grad(u2)'*Grad(v2) + Grad(u3)'*Grad(v3)
- div(u1,u2,u3)*q - div(v1,v2,v3)*p +1e-10*q*p)
//+on(0,p=2*z)
+on(0,u1=0,u2=0,u3=1-x^2-y^2)
+on(1,u1=0,u2=0,u3=0);
//Définition de la matrice associée
matrix A=Stokes(VVh,VVh) ;
// Appel de la méthode de résolution
set(A,solver=GMRES,dimKrylov=10000,eps=1e-10) ;
set(A,solver=UMFPACK) ;
//Résolution
real[int] b= Stokes(0,VVh) ;
VVh [u1,u2,u3,p] ;
u1[]= A^-1 * b;

// Affichage d'une coupure du cube à un endroit z
fespace V2d(T2d,P2) ; // 2d finite element space ..
V2d ux= u1(x,y,1) ;
V2d uy= u2(x,y,1) ;
V2d p2= p(x,y,1) ;
//Affichage des solutions
plot([ux,uy],value=true,cmm=" cut z = 1") ;
plot(p,wait=1,value=true,nbiso=10);

//Définition des erreurs
real eru;
real erp;
real ere;
real eruf;
real epe;
real erpf;
//Erreur absolue de u
eru=(int3d(Th)(u2^2+dx(u2)^2+dy(u2)^2+dz(u2)^2))^2
        +(int3d(Th)(u1^2+dx(u1)^2+dy(u1)^2+dz(u1)^2))^2
        +(int3d(Th)((u3-(1-x^2-y^2))^2+(dx(u3)+2*x)^2
        +(dy(u3)+2*y)^2+dz(u3)^2))^2;
//Norme de u
ere=int3d(Th)( (1-x^2-y^2)^2 +4*x^2+4*y^2);
//Erreur relative de u
eruf=sqrt(eru/ere);

//Erreur absolue de p
erp=int3d(Th)((p-(-4*z+5))^2);
//Norme de p
epe=int3d(Th)((-4*z+5)^2);
//Erreur relative de p
erpf=sqrt(erp/epe);

//Affichage des erreurs relatives
cout <<"erreur sur u:"<< eruf << endl;
cout << "erreur sur p:"<< erpf << endl;
```

6.3 Problème de la cavité carrée

6.3.1 Uzawa

```
//Définition du domaine et du maillage
int N;
```

```
N=140;
mesh Th=square(N,N);
real alpha;
alpha=1;
//Espaces de Taylor-Hood
fespace Vh(Th,[P2,P2]);
fespace Qh(Th,P1);
Vh [u1,u2],[v1,v2];
Qh p,q,Bu;
Qh p0=0;

// Macro Gradient et divergence
macro Grad(u) [dx(u),dy(u)] // EOM
macro div(u1,u2) (dx(u1)+dy(u2)) // EOM

//Définition des formes variationnelles
varf b([u1,u2],[q]) = int2d(Th)( -(div(u1,u2)*q));
varf a([u1,u2],[v1,v2])=
int2d(Th)(0.01*( Grad(u1)'*Grad(v1) + Grad(u2)'*Grad(v2) ))
+ on(3,u1=1,u2=0) + on(1,2,4,u1=0,u2=0) ;

//Définition des matrices A, B egt B^T
matrix A=a(Vh,Vh,solver=CG);
matrix B=b(Vh,Qh);
matrix Bt=B';

//Condition de bord sur u => On aura une fonction F.
real [int] F= a(0,Vh);
Vh [b1,b2];

//Construction de la fonction renvoyant alpha*Bu, le résidu.
func real[int] dJ(real[int] & pp)
{
b1[]= Bt*pp; b1[]*=-1; b1[]+= F; u1[]= A^-1*b1[];
real[int] Bu1= B*u1[]; // p p p = Bxu1
Bu1*=alpha;
return Bu1;
};

//Condition initiale
p=0; q=0;
// Calcul de p_{n+1}= p_{n) + alphaBu_n
// On utilise le gradient conjugué en appelant la fonction dJ
LinearCG(dJ,p[],eps=1.e-10,nbiter=1000);

// La convergence du gradient conjugué nous donne p et [u1,u2].

//Affichage des solutions
// coef=0.1 diminue la taille des vecteurs pour éviter
// une trop grande accumulation.
plot([u1,u2],p,wait=1,value=true,coef=0.1);
```

6.3.2 Gradient Conjugué

```
//Définition du maillage
int N;
N=140;
mesh Th=square(N,N);
```

```
// Espaces de Taylor_Hood
fespace Qh(Th,P1);
fespace Vh(Th,[P2,P2]);
Vh [u1,u2],[v1,v2];
Qh p,q;

// macro Divergence et Gradient
macro Grad(u) [dx(u),dy(u)] // EOM
macro div(u1,u2) (dx(u1)+dy(u2)) // EOM

//Définition de la forme variationnelle et condition de bord
varf b([u1,u2],[q]) = int2d(Th)( -div(u1,u2)*q);
varf a([u1,u2],[v1,v2])=
int2d(Th)( 0.01*(Grad(u1)'*Grad(v1) + Grad(u2)'*Grad(v2)))
+ on(3,u1=1,u2=0) + on(1,2,4,u1=0,u2=0) ;

//Définition des matrices A, B et B^T.
matrix A=a(Vh,Vh,solver=CG);
matrix B=b(Vh,Qh);
matrix Bt=B';

// Condition de bord => Fonction F
real [int] F=a(0,Vh);

Vh [b1,b2];
Vh [e1,e2];
Qh g1;
Vh [z1,z2];
Qh r1;
Qh g2;

// On construit une fonction calculant la direction de descente
// r(x0)=BA^-1*F-BA^1B^Tx0 pour un vecteur initiale x0.
func real[int] dJ(real[int] & x0)
{
b1[]=Bt*x0;
e1[]=A^-1*F;
g1[]=B*e1[];
z1[]=A^-1*b1[];
r1[]=B*z1[]; r1[]*=-1; r1[]+=g1[];
//r1[]*=10e10;
return r1[];
};

// Condition initiale
p=0; q=0;
// ON trouve p en appelant la méthode du gradient conjugué par
// la direction de descente dJ et le vecteur initiale p=0.
LinearCG(dJ,p[],eps=1.e-10,nbiter=1000);

//Une fois p trouvé, on peut calculer u de nouveau par
// la méthode du gradient conjugué
Vh [b3,b4];
b4[]=Bt*p[]; b4[]*=-1; b4[]+=F;
u1[]=A^-1*b3[];

// Affichage des solutions
plot([u1,u2],p,value=true,coef=0.1);
```