Projet : Cours DEA C++ et éléments finis.

Présenté à

L'Université Pierre et Marie Curie (Paris VI)

par

Nadia MHIOUAH et Eric DALISSIER

Navier-Stokes 2D

Le 10 février 2008

M. HECHT

Table des matières

Remerciments

Nous remercions M. Frederic HECHT pour sa patience, pour sa gentillesse. Nous avons beaucoup appris durant la réalisation de ce projet.

1 Calcul de la solution du problème de Navier-Stokes 2d à l'aide de FreeFem++

1.1 Position du problème

Soit un fluide incompressible de vitesse $u \in H^1(\Omega)^2$ et pression $p \in L^2(\Omega)$, solution des équations de Navier-Stokes incompressible dans un domaine Ω de $\mathbb{R}^2$.

$$\frac{\partial u}{\partial t} + u.\nabla u - \nu \Delta u + \nabla p = 0 \tag{1}$$

$$\nabla . u = 0 \tag{2}$$

plus des conditions aux limites de type Dirichlet $u|_\Gamma = u_\Gamma$.
Le domaine de calcul pourra être une marche de $\frac{1}{2}$ dans un canal de hauteur 1. En utilisant le logiciel **FreeFem++** capable de générer automatiquement des maillages triangulaires à partir des frontières du domaine de calcul, on obtient :

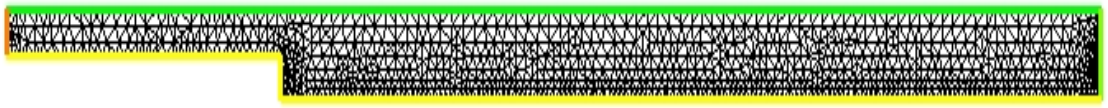

FIG. 1 – Maillage du domaine

La figure 1 présente le domaine de calcul. L'entrée du domaine s'effectue par le bord gauche, et la sortie par le bord droit. La frontière supérieure est considérée comme étant "à l'infini", et on y applique une condition d'imperméabilité afin que le fluide ne s'échappe pas par la frontière supérieure.
On définit les bords du domaine en utilisant la figure 1 :

- Γ_1 le bord inférieur (de couleur jaune)
- Γ_2 le bord de sortie (de couleur verte claire)
- Γ_3 le bord supérieur (de couleur vert)
- Γ_4 le bord d'entrée (de couleur orange)

1.2 Conditions aux bords

Reste enfin à déterminer la condition à imposer sur le bord de droite Γ_3. Pour cela on se sert de la condition de fluide incompressible :

$$div(u) = 0 \tag{3}$$

On applique la formule de Stokes à cette condition :

$$\int_{\Omega} div(u)dx = \int_{\partial\Omega} u\vec{n} \tag{4}$$

Avec l'hypothèse que

$$\partial\Omega = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4 \tag{5}$$

En utilisant les expressions (1), (2) et (3), on obtient :

$$\int_{\Gamma_1} u\vec{n} + \int_{\Gamma_2} u\vec{n} + \int_{\Gamma_3} u\vec{n} + \int_{\Gamma_4} u\vec{n} = 0 \tag{6}$$

D'après les hypothèses faites ci-dessus $u = 0$ sur Γ_2 et Γ_4, on déduit de (6) :

$$\int_{\Gamma_2} u\vec{n} = -\int_{\Gamma_4} u\vec{n} \tag{7}$$

On pose n_{Γ_1} (resp. n_{Γ_2}) la normale extérieure de Γ_1 (resp. Γ_2). On a donc $n_{\Gamma_1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ et $n_{\Gamma_1} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$
En remplaçant dans (5) :
$\int_{\Gamma_4} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = -\int_{\Gamma_2} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \begin{pmatrix} -1 \\ 0 \end{pmatrix}$
On a donc :

$$|\ \Gamma_4\ |\ u_1|_{\Gamma_4} = |\ \Gamma_2\ |\ u_1|_{\Gamma_2} \tag{8}$$

et :

$$u_1|_{\Gamma_2} = \frac{|\ \Gamma_4\ |}{|\ \Gamma_2\ |} \tag{9}$$

Dans notre problème de Stokes, nous n'imposons des conditions aux bords que pour la vitesse. Cela nous laisse donc une marge assez importante pour la pression. En effet, dans le problème initial , on a ∇p. Donc la pression est calculée à une constante près. Car $p + c_1$ et $p + c_2$ auront le même gradient.
Cette constante n'est pas prise de la même façon sous FreeFem++ et sous C++. Etudions d'abord le cas de FreeFem++. Pour cela partons de la formulation variationnelle du problème :

$$\int_{\Omega} \nabla u.\nabla v - \int_{\Omega} \nabla .vp - \int_{\Omega} \nabla .uq - \int_{\Gamma} (\nabla u.n)v - \int_{\Gamma} u.nq - \int_{\Omega} fv = 0 \tag{10}$$

Seulement dans ce cas là, la matrice, que l'on inverse, peut ne pas être inversible. Ce qui serait un gros problème.

$$\begin{pmatrix} A & B \\ B^t & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} F \\ 0 \end{pmatrix}$$

Pour faciliter la convergence de méthodes de résolution des systèmes linéaires on place sur la diagonale de la matrice nulle $-\epsilon$, et on assure l'inversibilité de la matrice. Dans la formulation variationnelle, cela revient a rajouté $\int_\Omega \epsilon qp$ à la formulation variationnelle.
On aboutit alors à cette formulation variationnelle : $\forall v \in H^1(\Omega)\ \forall q \in H^1(\Omega)/\mathbb{R}\oplus\mathbb{R}$

$$\int_\Omega \nabla u.\nabla v - \int_\Omega \nabla .vp - \int_\Omega \nabla .uq - \int_\Gamma (\nabla u.n)v - \int_\Gamma u.nq - \int_\Omega fv - \int_\Omega \epsilon qp = 0 \quad (11)$$

FreeFem++ va prendre $v = 0$ et $q = 1$. On aura alors :

$$\int_\Omega \epsilon p.1 - \int_\Gamma (u.n) * 1 = 0 \quad (12)$$

Donc on aura que :

$$\int_\Omega p = \frac{1}{\epsilon} \int_\Gamma u.n \quad (13)$$

Grace à cela nous avons pu vérifer que nos conditions aux limites étaient bonnes. Quelles vérifiaient à la fois la condition $div(u) = 0$ et $\int_\Omega p = \frac{1}{\epsilon} \int_\Gamma u.n$. En effet, cela nous donne l'ordre de grandeur de p, dans notre cas comme on a pris $\epsilon = 10^{-6}$. On arrive donc à p$\approx 10^{+6}$*u. Comme les valeurs de u sont très petite devant cette valeur on ne le verra pas sur les isovaleurs.

Pour le résultat sous C++, nous n avons pas cette condition sur p. Tout simplement car nous n avons pas rajouté ce terme de la même façon. Car nous n'avons pas mis que c'était $\frac{1}{\epsilon} \int_\Gamma u.n$, nous avons seulement mis dans la matrice à inverser $-\epsilon$ sur la diagonale des pressions. Mais nous n'avons rien mis dans le second membre. Donc on n'a aucune condition au bord sur p.
Voila la principale différence entre freefem et c++, pour l un, nous avons une condition juste sur u et p, et pour l'autre juste sur u.

1.3 Programme FreeFem++

On écrit le fichier de commandes suivant, pour calculer la solution "exacte" en utilisant **FreeFem++** :

```
border a(t=1,0){x=0;y=0.5+0.5*t;label=4;}; //      mesh
border b(t=0,1){x=5*t;y=0.5;label=1;};
```

```
border c(t=1,0){x=5 ;y=0.5*t;label=1;};
border d(t=0,1){x=5+15*t;y=0;label=1;};
border e(t=0,1){x=20 ;y=t;label=2;};
border h(t=1,0){x=20*t;y=1;label=3;};
mesh Th = buildmesh( a(7) + b(40) + c(10) + d(150) + e(50) + h(100));
plot(Th,wait=1,ps="solfm.ps");
savemesh(Th,"maillage.msh");

int n=1;

fespace Xh(Th,P2);                        //      definition of the velocity component space
fespace Mh(Th,P1);                       //      definition of the pressure space
Xh u2,v2;
Xh u1,v1;
Mh p,q;

int i=0;
real nu=1.;
real alpha=0.;
Xh up1,up2;
problem NS (u1,u2,p,v1,v2,q,solver=Crout,init=i) =
    int2d(Th)( alpha*( u1*v1 + u2*v2)
              + nu * ( dx(u1)*dx(v1) + dy(u1)*dy(v1) + dx(u2)*dx(v2) + dy(u2)*dy(v2) )
               - p*q*(0.000001)
               - p*dx(v1) - p*dy(v2)
               - dx(u1)*q - dy(u2)*q)
+ on(4*(-4*y*y+6*y-2),u2=0) //boundary conditions
   + on(1,3,u1=0,u2=0)
   + on(2,u1=-2*y*y+2*y+20,u2=0);

    NS;
     plot(coef=0.2,cmm=" [u1,u2] et p ",value=1,p,[u1,u2], ps="solfinal.ps");
```

1.4 Représentation de la solution

On represente le vecteur u dans la figure ci-dessous :

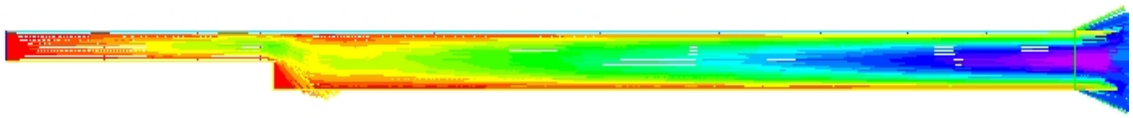

FIG. 2 – Représentation de la solution du problème de Stokes obtenue par la programmation FreeFem++

2 Discrétisation du problème de Navier-Stokes 2d

2.1 Formulation variationnelle

Rappelons le problème de Navier-Stokes :
Soit un fluide incompressible de vitesse $u \in H^1(\Omega)^2$ et de pression $p \in L^2(\Omega)$ solution des équation de Navier-Stokes incompressible dans un domaine Ω de $\mathbb{R}^2$.

$$\frac{\partial u}{\partial t} + u.\nabla u - \nu\Delta u + \nabla p = 0 \tag{14}$$

$$\nabla.u = 0 \tag{15}$$

plus des conditions de Dirichlet $u_|\Gamma = u_\Gamma$ avec $\Gamma = \partial\Omega$. En utilisant la dérivée particulaire pour approcher le terme $\frac{\partial u}{\partial t} + u.\nabla u$, on obtient le shéma suivant :

$$\frac{u^{n+1} - u^n \circ \chi^n}{\Delta t} \approx \frac{\partial u}{\partial t} + u.\nabla u$$

où $\chi^n(x)$ est la fonction de transport qui donne la position de la particule x au temps t^n et qui était en $\chi^n(x)$ au temps $t^{n-1} = t^n - \Delta t$. Donc, on a $\chi^n(x) = \zeta_x(t^{n-1})$ où $\zeta_x(t)$ est solution de l'équation différentielle rétrograde suivante $\mathrm{d}\zeta/\mathrm{d}t = u(\zeta_x(t))$ avec condition finale $\zeta_x(t^n) = x$.
Le domaine de calcul pourra être une marche de $\frac{1}{2}$ dans un canal de hauteur 1.
Nous obtenons le schéma temporelle, suivant d'ordre 1 :

$$u^{n+1} - \nu\Delta t\Delta u^{n+1} + \Delta t\nabla p^{n+1} = u^n \circ \chi^n \tag{16}$$

$$\nabla.u = 0 \tag{17}$$

Ecrivons la formulation variationnelle du problème de Navier-Stokes. On obtient donc en multipliant par une fonction test w et en intégrant su Ω :
$\forall w \in H^1(\Omega)^2$

$$\int_\Omega u^{n+1}w + \nu\Delta t\int_\Omega \nabla u^{n+1}\nabla w - \nu\Delta t\int_{\partial\Omega}\frac{\partial u^{n+1}}{\partial n}w -$$
$$\Delta t\int_\Omega p^{n+1}\nabla w + \Delta t\int_{\partial\Omega}(p^{n+1}.n)w = \int_\Omega (u^n \circ \chi^n)w \tag{18}$$

On pose :

$$a(u^{n+1}, w) = \int_\Omega u^{n+1}w + \nu\Delta t\int_\Omega \nabla u^{n+1}\nabla w - \nu\Delta t\int_{\partial\Omega}\frac{\partial u^{n+1}}{\partial n}w \tag{19}$$

$$b(w, p^{n+1}) = -\Delta t\int_\Omega p^{n+1}\nabla w + \Delta t\int_{\partial\Omega}(p^{n+1}.n)w \tag{20}$$

$$l(w) = \int_\Omega (u^n \circ \chi^n)w \tag{21}$$

Nous réecrivons le problème :
Chercher un couple $(u^{n+1}, p^{n+1}) \in V_h \times M_h$ tel que :

$$\forall w \in V_h, \;\; a(u^{n+1}, w) + b(w, p^{n+1}) = l(w), \tag{22}$$

$$\forall q \in M_h, \;\; b(u^{n+1}, q) = 0 \tag{23}$$

Dans la partie suivante nous négligerons les termes de bords, il seront étudier dans le paragraphe (?).

2.2 Problème discret

Dans la suite de ce rapport, afin de simplifier notre problème nous poserons $\alpha = \frac{1}{\Delta t} = 0$, nous réecrivons donc (19)-(21) :

$$a(u^{n+1}, w) = \nu \int_\Omega \nabla u^{n+1} \nabla w - \nu \int_{\partial\Omega} \frac{\partial u^{n+1}}{\partial n} w \tag{24}$$

$$b(w, p^{n+1}) = -\int_\Omega p^{n+1} \nabla w + \int_{\partial\Omega} (p^{n+1}.n) w \tag{25}$$

$$l(w) = \int_\Omega (u^n \circ \chi^n) w \tag{26}$$

et nous poserons par la suite $\forall w \in V_h, l(w) = 0$.

2.2.1 Définition des espaces V_h et M_h

Donnons nous une triangulisation régulière de Ω et définissons $\Omega_h = \bigcup_{k=1}^{nbt} T_k$, et notons v^i, $i \in \{1, \ldots, nbv\}$ les sommets de cette triangulation. On peut alors écrire :

$$\int_\Omega f(x)dx \approx \int_{\Omega_h} f(x)dx = \sum_{k=1}^{nbt} \int_{T_k} f(x)dx, \quad \forall \text{ f intégrables} \tag{27}$$

Définissons alors les espaces discrets V_h et M_h par :

$$V_h = \{v_h \in \mathcal{C}^0(\Omega_h), \forall k \in \{1, \ldots, nbt\}, v_h|_{T_k} \in P^2\} \tag{28}$$

$$M_h = \{v_h \in \mathcal{C}^0(\Omega_h), \forall k \in \{1, \ldots, nbt\}, v_h|_{T_k} \in P^1\} \tag{29}$$

où P^k désigne l'espace vectoriel des polinômes de deux variables de degré global inférieur ou égal à k.
Les fonctions de V_h sont entièrement déterminées par leurs valeurs en chacun des sommets v^i, $i \in \{0, \ldots, nbv - 1\}$ ainsi qu'aux points milieux de toutes les arêtes du maillage. La dimension de l'espace V_h est égale au nombre total $nbv + nbe - 1$

(où nbe désigne le nombre d'arêtes) des noeuds du maillage. Les fonctions de M_h, elles, sont entièrement déterminées par leurs valeurs en chacun des sommets v^i, $i \in \{0, \ldots, nbv - 1\}$. La dimension de M_h est égale au nombre total nbv des sommets du maillage.

2.2.2 Fonctions de base

Une base de M_h est formée des fonctions de M_h qui valent 1 en un noeud et 0 pour tous les autres.
Dans le cas des polynômes de degré 1 (P^1), nous pouvons utiliser les coordonnées barycentriques des sommets des triangles comme fonctions de base. Nous noterons $\lambda^i, i \in \{0, \ldots, 2\}$ la fonction de base associée aux sommets v^i du maillage vérifiant :

$$\forall (i,j) \in \{0, \ldots, 2\}^2, \quad \lambda_i(v^j) = \delta_{i,j} \tag{30}$$

On obtient donc par le calcul :

$$\begin{array}{ll} \lambda_0 & = 1 - x_1 - x_2 \\ \lambda_1 & = x_1 \\ \lambda_2 & = x_2 \end{array} \tag{31}$$

avec $X = (x_1, x_2)$ un point du maillage.

Remarque : Les coordonnées barycentriques sont conservées par la transformation affine $\mathcal{F}_T$ (voir le paragraphe (?)).

Une base de V_h est formée des fonctions de V_h qui valent 1 en un noeud et 0 pour tous les autres.
Nous noterons $\phi^i, i \in \{0, \ldots, 2\}$ la fonction de base associée aux sommets v^i du maillage, et $\phi^{3+i}, i \in \{0, \ldots, 2\}$ la fonction de base associée aux milieux des arêtes d'indice i du maillage. Par analogie, le milieu de la i-ème arête sera aussi noté v^i, avec $i \in \{3, \ldots, 5\}$. On déduit donc :

$$\forall (i,j) \in \{0, \ldots, 5\}^2, \quad \phi^i(v^j) = \delta_{i,j} \tag{32}$$

$$\begin{array}{ll} \phi_0 & = \lambda_0(2\lambda_0 - 1) \\ \phi_1 & = \lambda_1(2\lambda_1 - 1) \\ \phi_2 & = \lambda_2(2\lambda_2 - 1) \\ \phi_3 & = 4\lambda_1\lambda_2 \\ \phi_4 & = 4\lambda_0\lambda_2 \\ \phi_5 & = 4\lambda_0\lambda_1 \end{array} \tag{33}$$

2.2.3 Discrétisation en espace

On note u_h la solution approchée du problème de Navier-Stokes appartenant à V_h et on déduit de (17) son expression :

$$u_h(x) = \sum_{i=1}^{nbv+nbe} u_i \phi^i(x), \quad \forall x \in \Omega \tag{34}$$

Revenons à l'équation (14), et à l'aide de (18) on écrit $a_h(u^{n+1}, w)$ $\forall w \in H^1(\Omega)^2$:

$$a_h(u^{n+1}, w) = \sum_{k=1}^{nbt} \int_{T_k} \left(\nu \nabla u^{n+1} \nabla w\right) - \nu \sum_{k=1}^{nbt} \int_{\partial T_k} \frac{\partial u^{n+1}}{\partial n} w \tag{35}$$

Prenons $w = \phi$ avec ϕ la fonction de base de l'espace V_h définit dans le paragraphe précédent. Nous obtenons la formulation discrète suivante :

$$\int_{T_k} \left(\nu \nabla u^{n+1}(x) \nabla w(x)\right) \mathrm{d}x = \sum_{i;j} u_i^{n+1} \int_{T_k} \left(\nu \nabla \phi_i(x) \nabla \phi_j(x)\right) \mathrm{d}x \tag{36}$$

Par analogie on obtient pour le premier membre de b

$$\int_{T_k} \left(p^{n+1} \nabla w(x)\right) \mathrm{d}x = \sum_{i;j} p_i^{n+1} \int_{T_k} \left(\lambda_i(x) \nabla \phi_j(x)\right) \mathrm{d}x \tag{37}$$

L'idée à présent est d'effectuer un changement de variable afin de simplifier les calculs. Nous utiliserons une application affine de $\mathbb{R}^2$ dans $\mathbb{R}^2$ qui permet de passer du triangle courant au triangle de référence. Cette application est décrite dans le paragraphe suivant :

2.2.4 La transformation affine

Soit $\hat{T}$ le triangle de référence qui a pour sommets les points :
$\hat{a}^1 = (1, 0),\ \hat{a}^2 = (0, 1),\ \hat{a}^3 = (0, 0)$
Soit T un triangle non-dégénéré de sommets a^1, a^2, a^3. Il existe une et une seule matrice inversible B_T de $\mathbb{R}^{2,2}$ et un seul vecteur b_T de $\mathbb{R}^2$ tels que

$$\forall 1 \leq i \leq 3, a^i = B_T \hat{a}^i + b_T$$

En effet, $i = 3$ donne $b_T = a^3$ et $i = 1$, $i = 2$ donnent la colonne i de B_T : $B_T^i = a^i - a^3$. Donc B_T est inversible ssi T n'est pas dégénéré. On note $\mathcal{F}_T$ l'application affine de $\mathbb{R}^2$ dans $\mathbb{R}^2$

$$\forall \hat{x} \in \mathbb{R}^2,\ x = \mathcal{F}_T(\hat{x}) = B_T \hat{x} + b_T$$

. $\mathcal{F}_T$ est inversible, car B_T l'est et

$$\forall x \in \mathbb{R}^2,\ \hat{x} = \mathcal{F}_T^{-1}(x) = B_T^{-1}(x - b_T)$$

. De plus,

$$T = \mathcal{F}_T(\hat{T})$$

On en déduit,

$$\int_{T_k} f(x)\mathrm{d}x = \mid T_k \mid \int_{\hat{T}} f(\hat{x})\mathrm{d}\hat{x} \tag{38}$$

Reste alors à choisir une formule d'intégration numérique afin de calculer les intégrales.

2.2.5 Formule d'intégration numérique

Nous avons choisit d'utiliser une formule d'intégration à 3 points (exacte pour les polinômes de degrés 2). Réécrivons (25) en utilisant la formule d'intégration choisit :

$$\mid T_k \mid \int_{\hat{T}} f(\hat{x})\mathrm{d}\hat{x} \approx \mid T_k \mid \sum_{k=0}^{2} w_k f(x_k) \tag{39}$$

avec w_k les poids d'intégrations et x_k les points d'intégrations, et $\mid T_k \mid$ l'aire du triangle T_k.

2.3 Formulation variationnelle discrète

2.3.1 Ecriture matricielle

Dans ce paragraphe nous allons négliger les termes de bords que l'on décrit dans le paragraphe suivant.
Le but de ce paragraphe et d'écrire notre problême sous forme matricielle en utilisant tout ce qui a été expliqué précédemment. Nous cherchons donc à écrire un système de la forme $A.X = b$ avec A une matrice inversible.
A l'aide de (43), (45), et (48), on écrit le premier terme de a_h :

$$\sum_{k=0}^{nbt-1} \mid T_k \mid \sum_{l=0}^{2} \sum_{i,j} u_i^{n+1} w_l \nu \nabla\phi_i(x_l)\nabla\phi_j(x_l) \tag{40}$$

De la même façon écrivons le premier terme de b_h :

$$\sum_{k=0}^{nbt-1} \mid T_k \mid \sum_{l=0}^{2} \sum_{i,j} p_i^{n+1} w_l \lambda_i(x_l)\nabla\phi_j(x_l) \tag{41}$$

L'écriture de la matrice globale A s'effectue en deux étapes, la première étant la construction de la matrice élémentaire de chacun des triangles du maillage et la deuxième étape étant l'assemblage de toutes les matrices élémentaires dans la matrice globale.
Soit un triangle K quelconque du maillage, la matrice élémentaire (notée `MatElement`) s'écrit :

$$\begin{pmatrix} C & 0 & B_x \\ 0 & C & B_y \\ B_x^t & B_y^t & -I\epsilon \end{pmatrix}$$

C est une matrice $(nbv + nbe) \times (nbv + nbe)$ avec,

$$C_{ij} = |T_k| \sum_{l=0}^{2} w_l \nabla\phi_i(x_l)\nabla\phi_j(x_l)$$

B_x et B_y sont des matrice $nbv \times nbv$ avec,

$$B_{x_{ij}} = |T_k| \sum_{l=0}^{2} w_l \lambda_i(x_l) div_x(\phi_j(x_l))$$

$$B_{y_{ij}} = |T_k| \sum_{l=0}^{2} w_l \lambda_i(x_l) div_y(\phi_j(x_l))$$

Avec $w_l = 1/3$ pour $l = \{0, 1, 2\}$ et les points d'intégrations suivants :

l	0	1	2
x	0.5	0	0.5
y	0.5	0.5	0.

Remarque : Dans le cas où $\alpha \neq 0$, nous devrions utiliser une autre formule d'interpolation, en effet nous aurons besoin d'une formule exacte au minimum à l'ordre 4. Il faudra alors, en général, une formule à 7 points d'intégration.
Reste alors à effectuer l'assemblage de la matrice globale, ici, notée A (dans le programme elle est appelée `MatGlob`). Les coefficients de la matrice globale sont les coefficients des différentes matrices élémentaires, la difficulté, ici, est de bien placer les coefficients dans la matrice globale. Pour cela, on utilise les numéros globaux des noeuds du maillage, il faudra alors pouvoir trouver une correspondance entre les numéros locaux des noeuds du maillage et les numéros globaux (cette correspondance est faite par les fonctions `Th` et `Ik` dans le programme).

2.3.2 Conditions aux bords

Ayant vu les conditions que l'on doit imposer à u, c'est à dire $div\ u = 0$. On va se servir de la méthode de la "très grande valeur" pour rentrer cette condition dans le système $Ax = b$. Cette méthode consiste à identifier les les coordonnées des

points du bord du domaine dans notre matrice A, de remplacer cette valeur par le TGV (la très grande valeur). Ensuite on identifie la coordonnée de ce même point dans le second membre b. À cette dernière place, on met la valeur que l'on veut imposer au bord , multiplier par le TGV.
Cela aura pour conséquence, que lors du calcule de la solution $x = b.A^{-1}$. Avec cette méthode TGV, on gardera les valeurs imposées aux points du bord. En effet, seul la valeur de b sera pris en compte car très grande devant toutes les autres valeurs qu on pourrait avoir avec le produit matrice vecteur.
Les conditions aux limites, que l'on a imposé, sont des paraboles. Ainsi on n a pas de problème numérique au raccord de 2 labels du maillage. Car si un triangle posséde un coté sur un bord et un autre sur un autre, on ne saura pas quelles conditions on devra lui imposer. Donc le plus simple était de faire une condition qui se raccorde continuement au intersection de deux frontières différentes.
Á l'entrée de notre canal, nous avons voulu une parabole de maximum 1 en X. On a donc pris :

$$X = 4 * (-4Y^2 + 6Y - 2) \tag{42}$$

À la sortie de notre canal, il fallait vérifiée la condition :

$$\int_{\Gamma_e} u.n + \int_{\Gamma_s} u.n = 0$$

Qui est la traduction de la condition $div\ u = 0$. Comme Γ_e est de norme $\frac{1}{2}$, et Γ_s est de norme 1, il faut adapter la parabole en sortie. Pour cela, on va lui donner un maximum deux fois plus petit, donc un max de $\frac{1}{2}$ en X. On a donc :

$$X = -2Y^2 + 2Y + 20 \tag{43}$$

Les normals étant opposées, nous devons juste vérifier que :
$\int_{\Gamma_e} 4 * (-4Y^2 + 6Y - 2)dY = \int_{\Gamma_s} -2Y^2 + 2Y + 20dY$
On le vérifie :

En X=0 :
$$\begin{aligned} \int_{\frac{1}{2}}^{1} 4 * (-4Y^2 + 6Y - 2) - XdY &= [-\tfrac{16}{3}Y^3 + 12Y^2 - 4Y]_{\frac{1}{2}}^{1} \\ &= \tfrac{1}{3} \end{aligned}$$
En X=20 : (car notre domaine est de taille 20)
$$\begin{aligned} \int_{0}^{1} (-2Y^2 + 2Y + 20) - XdY &= [-\tfrac{2}{3}Y^3 + Y^2 + 20Y]_{0}^{1} \\ &= \tfrac{1}{3} + 20 - X \end{aligned}$$

3 Programmation

3.1 Les fonctions de base

Les fonctions de base sont programmées dans la classe `BuildBase`. Cette classe est faite d'un contruscteur qui prend comme paramêtre le numéro du triangle du maillage, et les coordonnées du point d'interpolation. Il va calculer toutes les fonctions de base. Cette méthode de calcule a été déclarée en private pour ne pas qu'un autre programme intervienne dans ce calcul simple. Par contre nous sommes obligés de déclarer les variables calculées en public, pour qu'elles soient accessibles de n'importe quels programmes.

```
class BuildBase{
public:
R Lambda[3];
R2 DLambda[3];
R PhiU[6];
R2 NablaPhiU[6];

public:
BuildBase(int triangle, R2 point);

private:
void initLambda( R2 point);
void initDLambda(int triangle);
void initPhiU();
void initNablaPhiU();
};
```

3.2 La matrice élémentaire

On déclare une matrice de taille $((nbv + nbe)2 + nbv) \times ((nbv + nbe)2 + nbv)$, puis on incrémente. On va d'abord calculer la matrice élémentaire en bouclant sur les indices de la matrices, puis sur les points d'interpolations.
Afin d'éviter des calculs inutiles, nous allons enregistrer les résultats trouvés dans une variable que l'on réutilisera ensuite. Afin qu' il n y ait aucun problème dans la gestion de la variable globale `MatElement`, qui contient la matrice élémentaire avant son incorporation dans `MatGlob`, nous allons l'initialiser à 0 avant que tout calcul soit fait sur un triangle.

3.3 La matrice globale

On construit la matrice globale à l'aide d'une map. (`map< <int,int>double>`). Cela a plusieurs avantages :

- vitesse d'accès aux informations, en effet on se déplace dans une map à l'aide de pointeurs.
- pas de tableaux dynamique à gérér et à créer
- avec une matrice creuse comme la notre, on ne charge que les valeurs non nulles, économie de place
- cela peut aussi éviter les membres non initialisés d'une matrice, et donc les valeurs aléatoires

Construction de la matrice globale :

```
for (int il=0;il<15;++il)
for (int jl=0;jl<15;++jl)
MatGlob[make_pair(numglob2[il],numglob2[jl])] += MatElement[il][jl];
}
```

Cela nous garantie que les données sont toutes initialisées , car `MatElement` l'est.

3.4 Résolution du système

On utilise UMFPACK pour la résolution du systéme. Mais juste avant la résolution, on doit passer la matrice globale sous la forme d une matrice de MORSE. Ceci est traité par le programme `SparseMatrixRC<double>`.
Ensuite on lance UMFPACK, qui nous rendra la solution dans un vecteur qu on aura initialisé.
On utilise ce solveur, car il est rapide et fonctionne pour presque tous les types de matrices, en particulier pour les matrices symétriques.

3.5 Représentation de la solution

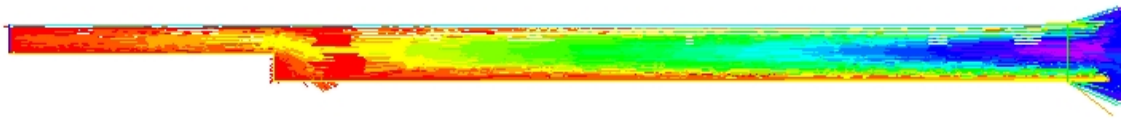

FIG. 3 – Représentation de la solution du problème de Stokes obtenue par la programmation C++

Comme FreeFem++ ne numérote pas de la même façon que nous les milieux des arêtes, nous avons dû faire une correspondance entre les 2 numérotations, afin d'afficher le bon vecteur au bon endroit du maillage.
Pour réaliser cela, nous avons écris un programme qui pour chaque triangle nous donne les 3 numéros globaux des sommets , puis les 3 numéros, numérotés par C++, des milieux. Ensuite il ne reste plus qu'a faire la correspondance sous FreeFem.
Juste un ennui rencontré, quand on essaie de lancer dans le programme C++ un script FreeFem++ qui justement modifie cette numérotation, cela ne fonctionne pas. Les variables n'arrivant pas à être initialisées. Mais le même script exécuté dans un terminal, fonctionnera très bien.

4 C++ vs. FreeFem++

Entre C++ et FreeFem++, nous avons la même allure de courbe, mais pour un maillage assez raffiné comme celui utilisé, les différences sont importantes. Pour commencer la normalisation de la pression, sous FreeFem++, on a la pression qui est $\frac{1}{\epsilon} * \int_\Gamma u.n$.
Ensuite pour la vitesse,nous n'avons pas exactement les mêmes isovaleurs. Mais l'ordre de grandeur reste le même. Une hypothèse restant envisageable, les arrondis suivant la programmation choisie peuvent être différents.
La vitesse de calcule entre FreeFem++ et C++ est semblable ($\simeq$0.7s). Seulement sous FreeFem++ la programmation est plus rapide. On peut d'ailleurs plus facilement trouver les erreurs mathématiques.
C++ a l'avantage de mieux modéliser la Pression (sans la normaliser à l'ordre de $\frac{1}{\epsilon}$).
Dans les deux cas, nous voyons une perturbation se former juste après la marche. La vitesse de l'écoulement étant suffisament important pour atténuer cet effet.
Enfin pour terminer, on peut remarquer que le modèle mathématique a ses limites, car les vecteurs vitesses du fluide traversent les parois supposées imperméables. Cela peut s'expliquer par le mauvais choix de la taille des vecteurs par FreeFem++ lors de la visualisation.

5 Conclusion

La réalisation de ce projet nous a permis d'apprendre des méthodes et des raisonnements, aussi bien en mathématique qu'en programmation.
En mathématiques, la mise en oeuvre des différents cours suivis pendant l'année, en effet nous avons dû appliquer des méthodes d'approximations pour résoudre le problème.
En informatique, la principale difficulté que nous avons rencontré à été le débuguage, qui nous a permis d'apprendre à ne plus faire de la programmation "plat de nouilles".