Stabilité Inf-Sup de schémas numériques pour le problèmes de Stokes

Flore Nabet

Stage de master 2 réalisé sous la tutelle de Franck Boyer

Table des matières

Introduction

Le but de ce travail est l'étude numérique de la constante Inf-Sup sur des schémas éléments finis et volumes finis pour le problème de Stokes 2D :

$$\text{Trouver } u \in (H^1(\Omega))^2 \text{ et } p \in L^2(\Omega) \text{ tels que : } \begin{cases} -\Delta u + \nabla p = f \text{ dans } \Omega \\ \operatorname{div} u = 0 \text{ dans } \Omega, \end{cases} \tag{1}$$

où l'on fait les hypothèses suivantes :
– Ω est un ouvert borné de $\mathbb{R}^n$ polygonal ou régulier,
– $f \in L^2(\Omega)$,
– $u = 0$ sur $\partial\Omega$,
– $\int_\Omega p = 0$.

On va pour cela discrétiser le problème soit en éléments finis soit en volumes finis et essayer de calculer la constante Inf-Sup discrète pour différents maillages.

Dans un premier chapitre, on étudie donc le problème discret pour se ramener à un problème matriciel. Ainsi, trouver la constante Inf-Sup discrète du problème de Stokes revient à trouver certaines valeurs propres d'une matrice qui dépend de la discrétisation choisie.
L'un des problèmes importants que l'on va rencontrer tout le long de ce travail est la modélisation de l'espace $L^2(\Omega)$ tel que $\int_\Omega p = 0$ qui est l'espace des pressions.
On expliquera ensuite certaines des méthodes que l'on peut utiliser ici tout en les comparant pour choisir la plus efficace qui sera celle que l'on utilisera principalement.
Enfin, on décrira les cas tests choisis ainsi que les différents tests que l'on va effectuer qui seront principalement des tests de convergence du schéma puis des tests sur les valeurs propres de la matrice en question.

Dans un second chapitre, on va étudier différents schémas éléments finis. On commence par réaliser une première partie de tests dont les maillages et les matrices utilisées seront calculées par le logiciel FreeFem++ dont on expliquera les fonctionnalités utilisées. Ces tests nous permettront de vérifier certains résultats connus sur la minimisation uniforme de la constante Inf-Sup puis d'en extrapoler d'autres.
Ensuite, ce logiciel ne réalisant que des maillages triangles on va construire à la main le schéma Q_1/P_0 ce qui nous permettra de vérifier que dans ce cas la constante Inf-Sup n'est pas uniformément minorée.
Enfin, on réalisera des tests à l'aide du logiciel PELICANS qui nous permet d'avoir des raffinements locaux adaptatifs. Dans ce cas, aucun résultat théorique n'est connu. On va donc tester différents raffinements pour voir quels résultats on pourrait obtenir.

Le troisième chapitre est, quand à lui, consacré à l'étude des schémas volumes finis et plus particulièrement au schéma DDFV dont on ne savait pas, prouver ou réfuter l'existence d'une minoration uniforme pour la constante Inf-Sup. Ce chapitre nous permet notemment de montrer qu'il existe des cas pour lesquels la constante Inf-Sup n'est pas uniformément minorée bien que le schéma converge.

Chapitre 1

Etude du problème posé

I Le problème de la constante Inf-Sup

I.1 Formulation faible

1.1.1 Rappels

Les résultats donnés ici sont démontrés dans [EG04, Chap. 3]

Définition 1.1 (Problèmes de type point-selle)
Soient deux espaces de Hilbert X et M, a une forme bilinéaire continue sur $X \times X$, b une forme bilinéaire continue sur $X \times M$. Pour toutes formes linéaires $L \in X'$ et $G \in M'$, on cherche $u \in X$ et $p \in M$ tels que :

$$\begin{cases} a(u,v) + b(v,p) = L(v), \forall v \in X \\ b(u,q) = G(q), \forall q \in M. \end{cases} \tag{I.1}$$

Théorème 1.1 (Condition Inf-Sup)
Si a est coercive sur X tout entier ou sur $Z = \{v \in X, b(v,q) = 0, \forall q \in M\}$, alors le problème (I.1) *est bien posé si et seulement si la condition suivante est vérifiée :*

$$\exists \beta > 0, \ \inf_{p \in M} \left(\sup_{v \in X} \frac{b(v,p)}{||v||_X ||p||_M} \right) \geq \beta. \tag{I.2}$$

La méthode de Galerkin consiste à choisir X_h et M_h deux sous-espaces de dimensions finies respectivement de X et M puis à considérer le problème approché suivant : trouver $u_h \in X_h$ et $p_h \in M_h$ tels que :

$$\begin{cases} a(u_h, v_h) + b(v_h, p_h) = L(v_h), \forall v_h \in X_h \\ b(u_h, q_h) = G(q_h), \forall q_h \in M_h. \end{cases} \tag{I.3}$$

Proposition 1.1
Le problème approché (I.3) *est bien posé si et seulement s'il vérifie une condition Inf-Sup discrète sur b qui s'écrit :*

$$\inf_{p_h \in M_h} \left(\sup_{v_h \in X_h} \frac{b(v_h, p_h)}{||v_h||_X ||p_h||_M} \right) \geq \beta_h > 0. \tag{I.4}$$

De plus, les estimations d'erreurs sont en $\frac{1}{\beta_h}$:

Proposition 1.2
Sous l'hypothèse que (I.4) *soit vérifiée, la solution (u_h, p_h) du problème* (I.3) *satisfait les estimations :*

$$\|u - u_h\|_X \leq \left(1 + \frac{\|a\|}{\alpha}\right) \left(1 + \frac{\|b\|}{\beta_h}\right) d(u, X_h) + \frac{\|b\|}{\alpha} d(p, M_h),$$
$$\|p - p_h\|_M \leq \left(1 + \frac{\|a\|}{\alpha}\right) \left(1 + \frac{\|b\|}{\beta_h}\right) \frac{\|a\|}{\beta_h} d(u, X_h) + \left(1 + \frac{\|b\|}{\beta_h} + \frac{\|b\|}{\alpha} \frac{\|a\|}{\beta_h}\right) d(p, M_h),$$

où α est la constante de coercivité de a.

Ces estimations d'erreurs explosent si $\beta_h \to 0$ et donc montrent bien l'importance du fait que la constante Inf-Sup vérifie $\inf_h \beta_h > 0$, ce qui explique le but de ce travail.

1.1.2 Cas du problème de Stokes

Pour le problème de Stokes (1), la condition $u = 0$ sur $\partial\Omega$ nous incite à prendre $X = (H_0^1(\Omega))^2$, alors que la condition $\int_\Omega p = 0$ nous fait choisir $M = L_0^2(\Omega) = \{p \in L^2(\Omega) : \int_\Omega p = 0\}$.
Pour obtenir les formes bilinéaires a et b,

- on multiplie la première équation de (1) par $v \in X$ puis on intègre par partie,
- on multiplie par $q \in M$ la seconde équation puis on intègre.

On obtient ainsi un problème du type (I.1) avec :

$$\begin{cases} a(u,v) = \int_\Omega \nabla u(x) : \nabla v(x) dx \\ b(v,p) = -\int_\Omega p(x) \operatorname{div} v(x) dx \\ L(v) = \int_\Omega f(x) v(x) dx \\ G(q) = 0. \end{cases}$$

On choisit alors X_h un sous-espace de dimension finie N_u de X et M_h un sous-espace de dimension finie N_p de M. Soient $(\psi_i)_{1\leq i\leq Np}$ une base de M_h et $(\varphi_i)_{1\leq i\leq Nu}$ une base de X_h telle que pour $i \in \{1, ..., \frac{N_u}{2}\}$,

$$\varphi_i = \begin{pmatrix} \alpha_i \\ 0 \end{pmatrix} \text{ et } \varphi_{i+\frac{N_u}{2}} = \begin{pmatrix} 0 \\ \alpha_i \end{pmatrix}.$$

Remarque 1.1
Tous les schémas éléments finis n'ont pas une base de cette forme mais c'est le cas de ceux avec lesquels on va travailler.

Ce qui nous intéresse ici est l'évolution de la constante inf-sup quand h tend vers 0 lorsque l'on raffine notre maillage. Cette constante β_h est donnée par :

$$\beta_h = \inf_{p_h \neq 0 \in M_h} \left(\sup_{u_h \neq 0 \in X_h} \frac{b(u_h, p_h)}{||p_h||_M ||u_h||_X} \right) > 0 \text{ d'après (I.4)}, \tag{I.5}$$

lorsque les couples X_h et M_h sont bien choisis.

I.2 Formulation matricielle

Soient $p_h \in M_h$ et $u_h \in X_h$, alors $p_h = \sum_{i=1}^{N_p} p_i \psi_i$ et $u_h = \sum_{i=1}^{N_u} u_i \varphi_i$.
On pose ainsi : $U = (u_1, ..., u_{N_u})$ et $P = (p_1, ..., p_{N_p})$ et ce problème peut s'écrire sous la forme matricielle suivante :
Trouver $(U, P) \in \mathbb{R}^{N_u} \times \mathbb{R}^{N_p}$ tels que :

$$\begin{cases} RU + B^T P = F \\ BU = 0, \end{cases} \tag{I.6}$$

où :

- la matrice de rigidité $R = (\int_\Omega \nabla\varphi_i : \nabla\varphi_j)_{1\leq i,j\leq N_u}$ du système est telle que $(RU, V) = a(u_h, v_h)$.
- la matrice de la divergence $B = (-\int_\Omega \psi_i \operatorname{div} \varphi_j)_{1\leq i\leq N_p, 1\leq j\leq N_u}$ est telle que $(BU, P) = b(u_h, p_h)$.
- le second membre F est le vecteur $F = (\int_\Omega f\varphi_i)_{1\leq i\leq N_u}$.

On utilisera également la matrice de masse relative à la pression $M = (\int_\Omega \psi_i\psi_j)_{1\leq i,j\leq N_p}$ de ce système qui est telle que $(MP, Q) = (p_h, q_h)_M$.

Avec ces notations, la constante Inf-Sup (I.5) est donnée par :

$$\beta_h = \inf_{P\neq 0} \sup_{U\neq 0} \frac{(B^T P, U)}{(MP, P)^{1/2} (RU, U)^{1/2}}. \tag{I.7}$$

Posons $V = R^{1/2}U$, alors R étant symétrique, on a :

$$\begin{cases} (RU, U) & = (R^{1/2}U, R^{1/2}U) = (V, V) \\ (B^T P, U) & = (B^T P, R^{-1/2}V) = (R^{-1/2}B^T P, V). \end{cases}$$

La condition inf-sup (I.7) devient alors, en notant $\|\cdot\|_2$ la norme euclidienne :

$$\beta_h = \inf_{P\neq 0} \frac{1}{(MP,P)^{1/2}} \sup_{V\neq 0} \frac{(R^{-1/2}B^TP,V)}{\|V\|_{\mathbb{R}^{N_u}}} = \inf_{P\neq 0} \frac{\left\|R^{-1/2}B^TP\right\|_2}{(MP,P)^{1/2}}$$

et $\left\|R^{-1/2}B^TP\right\|_2^2 = (BR^{-1}B^TP,P)$, donc si on pose $Q = M^{1/2}P$, M étant symétrique on a $(MP,P) = (Q,Q)$, ce qui nous donne :

$$\beta_h^2 = \inf_{Q\neq 0} \frac{(M^{-1/2}BR^{-1}B^TM^{-1/2}Q,Q)}{\|Q\|_2^2} = \text{la plus petite valeur propre de } M^{-1/2}BR^{-1}B^TM^{-1/2},$$

et ainsi,

$$\beta_h = \sqrt{\text{la plus petite valeur propre de } M^{-1/2}BR^{-1}B^TM^{-1/2}}. \tag{I.8}$$

Remarquons que $M^{-1/2}BR^{-1}B^TM^{-1/2} = (R^{-1/2}B^TM^{-1/2})^T(R^{-1/2}B^TM^{-1/2})$, donc cette matrice est symétrique positive et ses valeurs propres sont bien toutes positives.

II Les méthodes utilisées

Notre but est donc de calculer la plus petite valeur propre de la matrice $M^{-1/2}BR^{-1}B^TM^{-1/2}$ pour voir, selon les cas, si la constante Inf-Sup est uniformément minorée ou pas. Mais, on peut également se demander ce qu'il en est des deuxième et troisième valeurs propres de cette matrice. En effet, elle peuvent tendre vers 0 en même temps que la constante Inf-Sup (voir le schéma Q_1/P_0 Chap. 2, Partie II) ou bien être minorée alors que la constante Inf-Sup tend vers 0 (voir le schéma DDFV pour un maillage localement raffiné ou bidomaine Chap. 3, Partie II).
Commençons déjà par remarquer que le spectre de la matrice $M^{-1/2}BR^{-1}B^TM^{-1/2}$ est identique à celui de la matrice $M^{-1}BR^{-1}B^T$:

$$\begin{aligned}\lambda \text{ valeur propre de } M^{-1/2}BR^{-1}B^TM^{-1/2} &\iff \lambda \text{ valeur propre de } M^{-1/2}M^{-1/2}BR^{-1}B^T\\ &\iff \lambda \text{ valeur propre de } M^{-1}BR^{-1}B^T.\end{aligned}$$

Le calcul de M^{-1} étant moins lourd que celui de $M^{-1/2}$ on préfère travailler sur la matrice $M^{-1}BR^{-1}B^T$ mais, contrairement à la matrice $M^{-1/2}BR^{-1}B^TM^{-1/2}$ elle n'est pas symétrique. Cependant, elle est symétrique pour le produit scalaire de la matrice M et c'est comme cela que l'on va raisonner.
A partir de ceci, nous allons utiliser différentes méthodes. Cependant, quelque soit la méthode utilisée, un problème va survenir. En effet, l'espace choisi pour la pression est l'espace $L_0^2(\Omega) = \{p \in L^2(\Omega) : \int_\Omega p = 0\}$ qu'il est difficile (quelque soit le logiciel de calcul utilisé) de modéliser. Souvent, la contrainte $\int_\Omega p = 0$ est prise en compte par des méthodes de pénalisations. Or, ce n'est pas ce que l'on veut ici car on cherche à discrétiser l'espace $L_0^2(\Omega)$. On va donc se placer sur l'espace $L^2(\Omega)$ et essayer de se ramener à $L_0^2(\Omega)$.
Le supplémentaire orthogonal pour le produit scalaire L^2 des fonctions à moyenne nulle sont les fonctions constantes, il nous faut donc enlever les pressions constantes. Sinon,la plus petite valeur propre que l'on obtiendra sera toujours 0 car le vecteur $1 = \begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix}$ sera dans le noyau de B^T. En effet, :

$$\forall u \in H_0^1(\Omega), (U, B^T1) = (BU,1) = -\int_\Omega \operatorname{div} u = -\int_{\partial\Omega} u.n = 0 \Rightarrow B^T\begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix} = 0.$$

II.1 La méthode Scilab

La commande $Spec$ de Scilab nous donne toutes les valeurs propres de la matrice. Il suffit alors d'enlever la plus petite qui est nulle et de récupérer les trois suivantes. Le problème de cette méthode est qu'elle est très coûteuse et que lorsque la matrice devient trop grosse elle devient inutilisable et ceci pour plusieurs raisons :

1. Le calcul de $M^{-1/2}$ (ou M^{-1}) et de R^{-1} est long et les matrices sont pleines.
2. Elle nous donne toutes les valeurs propres alors que l'on ne veut que les plus petites, il y a donc beaucoup de calculs inutiles.

II.2 La méthode de la puissance inverse

2.2.1 Principe général

Cette méthode permet de trouver la plus petite valeur propre d'une matrice.
Principe général de la méthode pour une matrice A ([LT94, p504-509]) :

- On choisi un vecteur initial Q_0 de norme 1.
- On cherche P_0 tel que $AP_0 = Q_0$.
- On pose $Q_1 = \frac{P_0}{\|P_0\|_2}$ pour l'itération suivante.
- On continue jusqu'à ce que le critère d'arrêt $\frac{\left|\frac{1}{\|P_i\|_2} - \frac{1}{\|P_{i-1}\|_2}\right|}{\frac{1}{\|P_i\|_2}} < \varepsilon$ soit vérifié (avec ε choisi suffisamment petit).
- La valeur finale de $\frac{1}{\|P_i\|_2}$ est la valeur de la plus petite valeur propre de la matrice A et le vecteur propre correspondant est alors $\frac{P_i}{\|P_i\|_2}$.

Principe de la méthode pour la matrice $M^{-1}BR^{-1}B^T$:

- On choisi un vecteur initial Q_0 de norme euclidienne égale à 1.
- On cherche $P_0 \in \mathbb{R}^{N_p}$ tel que $(M^{-1}BR^{-1}B^T)P_0 = Q_0$, pour cela :
  - on pose $U_0 = -R^{-1}B^TP_0$,
  - on résout alors le problème de Stokes suivant : $(*)\begin{cases} RU_0 + B^TP_0 = 0 \\ BU_0 = -MQ_0 \end{cases}$.
- On pose $Q_1 = \frac{P_0}{\|P_0\|_2}$ pour l'itération suivante.
- On continue jusqu'à ce que le critère d'arrêt $\frac{\left|\frac{1}{\|P_i\|_2} - \frac{1}{\|P_{i-1}\|_2}\right|}{\frac{1}{\|P_i\|_2}} < \varepsilon$ soit vérifié (avec ε choisi suffisemment petit).
- La valeur finale de $\frac{1}{\|P_i\|_2}$ est alors la valeur de la plus petite valeur propre de la matrice $M^{-1}BR^{-1}B^T$.

On utilise alors deux méthodes pour supprimer les pressions constantes pour que $(*)$ ait une solution.

2.2.2 Une première méthode

1. On modifie M en $\tilde{M}$ de sorte que si p est à moyenne nulle, c'est à dire si $(MP, 1) = 0$, alors $(MP, P) = (\tilde{M}\tilde{P}, \tilde{P})$ où $\tilde{P} \in \mathbb{R}^{N_p-1}$ est le vecteur P auquel on a enlevé sa dernière coordonnéee.
   Appelons $m_{i,j}$ les coefficients de M pour $1 \leq i, j \leq N_p$, $\tilde{m}_{i,j}$ ceux de $\tilde{M}$ et p_i ceux de P (pour $1 \leq i \leq N_p$) ainsi que ceux de $\tilde{P}$ (pour $1 \leq i \leq N_p - 1$).
   On a alors $p_{N_p} = -\frac{\sum_{i=1}^{N_p}\sum_{j=1}^{N_p-1} m_{i,j}p_j}{\sum_{i=1}^{N_p} m_{i,N_p}}$, car :

$$0 = (MP, 1) = \sum_{i=1}^{N_p}\sum_{j=1}^{N_p} m_{i,j}p_j = \sum_{i=1}^{N_p}(\sum_{j=1}^{N_p-1} m_{i,j}p_j + m_{i,N_p}p_{N_p}) = \sum_{i=1}^{N_p}\sum_{j=1}^{N_p-1} m_{i,j}p_j + p_{N_p}(\sum_{i=1}^{N_p} m_{i,N_p}),$$

   et donc :

$$\begin{aligned}
(MP, P) &= \sum_{i=1}^{N_p}\left(\sum_{j=1}^{N_p-1} m_{i,j}p_ip_j + m_{i,N_p}p_ip_{N_p}\right) = \sum_{i=1}^{N_p}\sum_{j=1}^{N_p-1} m_{i,j}p_ip_j + p_{N_p}\sum_{i=1}^{N_p} m_{i,N_p}p_i \\
&= \sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1} m_{i,j}p_ip_j + p_{N_p}\sum_{j=1}^{N_p-1} m_{N_p,j}p_j + p_{N_p}\sum_{i=1}^{N_p} m_{i,N_p}p_i \\
&= \sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1} m_{i,j}p_ip_j + 2p_{N_p}\sum_{i=1}^{N_p-1} m_{i,N_p}p_i + m_{N_p,N_p}p_{N_p}^2 \text{ car } M \text{ est symétrique} \\
&= \sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1} m_{i,j}p_ip_j - \frac{2}{\sum_{i=1}^{N_p} m_{i,N_p}}\left(\sum_{i=1}^{N_p}\sum_{j=1}^{N_p-1} m_{i,j}p_j\right)\left(\sum_{i=1}^{N_p-1} m_{i,N_p}p_i\right) \\
&+ \frac{m_{N_p,N_p}}{\left(\sum_{i=1}^{N_p} m_{i,N_p}\right)^2}\left(\sum_{i=1}^{N_p}\sum_{j=1}^{N_p-1} m_{i,j}p_j\right)^2
\end{aligned}$$

$$
\begin{aligned}
(MP,P) = &\sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1} m_{i,j}p_ip_j - \frac{2}{\sum_{i=1}^{N_p} m_{i,N_p}}\left(\sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1} m_{i,j}p_j\right)\left(\sum_{i=1}^{N_p-1} m_{i,N_p}p_i\right)\\
&- \frac{2}{\sum_{i=1}^{N_p} m_{i,N_p}}\left(\sum_{i=1}^{N_p-1} m_{i,N_p}p_i\right)^2 + \frac{m_{N_p,N_p}}{\left(\sum_{i=1}^{N_p} m_{i,N_p}\right)^2}\left(\sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1} m_{i,j}p_j\right)^2\\
&+ \frac{m_{N_p,N_p}}{\left(\sum_{i=1}^{N_p} m_{i,N_p}\right)^2}\left(\sum_{i=1}^{N_p-1} m_{i,N_p}p_i\right)^2 + \frac{2m_{N_p,N_p}}{\sum_{i=1}^{N_p} m_{i,N_p}}\left(\sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1} m_{i,j}p_j\right)\left(\sum_{i=1}^{N_p-1} m_{i,N_p}p_i\right)\\
= &\sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1} m_{i,j}p_ip_j + \frac{2}{\sum_{i=1}^{N_p} m_{i,N_p}}\left(\frac{m_{N_p,N_p}}{\sum_{i=1}^{N_p} m_{i,N_p}} - 1\right)\left(\sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1} m_{i,j}p_j\right)\left(\sum_{i=1}^{N_p-1} m_{i,N_p}p_i\right)\\
&+ \frac{1}{\sum_{i=1}^{N_p} m_{i,N_p}}\left(\frac{m_{N_p,N_p}}{\sum_{i=1}^{N_p} m_{i,N_p}} - 2\right)\left(\sum_{i=1}^{N_p-1} m_{i,N_p}p_i\right)^2 + \frac{m_{N_p,N_p}}{\left(\sum_{i=1}^{N_p} m_{i,N_p}\right)^2}\left(\sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1} m_{i,j}p_j\right)^2,
\end{aligned}
$$

car $\displaystyle\sum_{i=1}^{N_p}\sum_{j=1}^{N_p-1} m_{i,j}p_j = \sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1} m_{i,j}p_j + \sum_{j=1}^{N_p-1} m_{N_p,j}p_j.$

Rappelons que l'on cherche la plus petite valeur propre de la matrice $M^{-1/2}BR^{-1}B^TM^{-1/2}$, cette matrice est clairement symétrique (car M et R le sont), ce qui nous assure que les valeurs propres sont toutes réelles. Lorsque l'on change M en $\tilde{M}$, on aimerait donc garder cette symétrie.
Réécrivons donc les différents termes obtenus de manière à ce que $\tilde{M}$ soit symétrique. On a alors :

$$
\begin{aligned}
2\left(\sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1} m_{i,j}p_j\right)\left(\sum_{i=1}^{N_p-1} m_{i,N_p}p_i\right) &= \left(\sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1} m_{i,j}p_j\right)\left(\sum_{i=1}^{N_p-1} m_{i,N_p}p_i\right)\\
&+ \left(\sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1} m_{i,j}p_i\right)\left(\sum_{j=1}^{N_p-1} m_{j,N_p}p_j\right)\\
&= \sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1}\left(\sum_{k=1}^{N_p-1} m_{k,j}\right)m_{i,N_p}p_ip_j + \sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1}\left(\sum_{k=1}^{N_p-1} m_{i,k}\right)m_{j,N_p}p_ip_j\\
&= \sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1}\left[\left(\sum_{k=1}^{N_p-1} m_{k,j}\right)m_{i,N_p} + \left(\sum_{k=1}^{N_p-1} m_{i,k}\right)m_{j,N_p}\right]p_ip_j,
\end{aligned}
$$

$$
\left(\sum_{i=1}^{N_p-1} m_{i,N_p}p_i\right)^2 = \left(\sum_{i=1}^{N_p-1} m_{i,N_p}p_i\right)\left(\sum_{j=1}^{N_p-1} m_{j,N_p}p_j\right) = \sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1} m_{i,N_p}m_{j,N_p}p_ip_j,
$$

$$
\left(\sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1} m_{i,j}p_j\right)^2 = \left(\sum_{k=1}^{N_p-1}\sum_{j=1}^{N_p-1} m_{k,j}p_j\right)\left(\sum_{l=1}^{N_p-1}\sum_{i=1}^{N_p-1} m_{l,i}p_i\right) = \sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1}\left(\sum_{k=1}^{N_p-1} m_{k,j}\right)\left(\sum_{l=1}^{N_p-1} m_{l,i}\right)p_ip_j.
$$

Ainsi, si l'on veut que $(MP,P)=(\tilde{M}\tilde{P},\tilde{P})=\sum_{i=1}^{N_p-1}\sum_{j=1}^{N_p-1}\tilde{m}_{i,j}p_ip_j$, on doit poser :

$$\begin{aligned}\tilde{m}_{i,j} = m_{i,j} &+ \frac{1}{\sum_{i=1}^{N_p} m_{i,N_p}}\left(\frac{m_{N_p,N_p}}{\sum_{i=1}^{N_p} m_{i,N_p}} - 2\right) m_{i,N_p}m_{j,N_p} \\ &+ \frac{1}{\sum_{i=1}^{N_p} m_{i,N_p}}\left(\frac{m_{N_p,N_p}}{\sum_{i=1}^{N_p} m_{i,N_p}} - 1\right)\left[\left(\sum_{k=1}^{N_p-1} m_{k,j}\right) m_{i,N_p} + \left(\sum_{k=1}^{N_p-1} m_{i,k}\right) m_{j,N_p}\right] \\ &+ \frac{m_{N_p,N_p}}{\left(\sum_{i=1}^{N_p} m_{i,N_p}\right)^2}\left(\sum_{k=1}^{N_p-1} m_{k,j}\right)\left(\sum_{l=1}^{N_p-1} m_{l,i}\right).\end{aligned}$$

2. On modifie $B=(b_{i,j})_{\substack{1\le i\le N_p\\1\le j\le N_u}}$ en $\tilde{B}=(\tilde{b}_{i,j})_{\substack{1\le i\le N_p-1\\1\le j\le N_u}}$ tel que :

$$B^TP = B^T\begin{pmatrix} p_1 \\ \vdots \\ p_{N_p-1} \\ -\dfrac{\sum_{i=1}^{N_p}\sum_{j=1}^{N_p-1} m_{i,j}p_j}{\sum_{i=1}^{N_p} m_{i,N_p}} \end{pmatrix} = \tilde{B}^T\begin{pmatrix} p_1 \\ \vdots \\ p_{N_p-1}\end{pmatrix} = \tilde{B}^T\tilde{P}.$$

Or pour $1\le i\le N_u$, on sait que :

$$\begin{cases} (B^TP)_i &= \displaystyle\sum_{j=1}^{N_p-1} b_{j,i}p_j - \frac{b_{N_p,i}}{\sum_{i=1}^{N_p} m_{i,N_p}}\sum_{k=1}^{N_p}\sum_{j=1}^{N_p-1} m_{k,j}p_j = \sum_{j=1}^{N_p-1}\left(b_{j,i} - \frac{b_{N_p,i}}{\sum_{i=1}^{N_p} m_{i,N_p}}\left(\sum_{k=1}^{N_p} m_{k,j}\right)\right)p_j, \\ (\tilde{B}^T\tilde{P})_i &= \displaystyle\sum_{j=1}^{N_p-1}\tilde{b}_{j,i}p_j. \end{cases}$$

Il nous faut donc poser :

$$\tilde{b}_{i,j} = b_{i,j} - \frac{b_{N_p,j}}{\sum_{k=1}^{N_p} m_{k,N_p}}\left(\sum_{k=1}^{N_p} m_{k,i}\right).$$

La construction des ces deux matrices nous permet de nous ramener à un nouveau problème de Stokes :

$$\begin{cases} RU_0 + \tilde{B}^T\tilde{P}_0 = 0 \\ \tilde{B}U_0 = -\tilde{M}Q_0 \end{cases}$$

qui est équivalent au problème $(*)$.
Cependant, la construction de ces deux nouvelles matrices est long et remplit le stencil de la matrice. On va donc essayer de trouver un moyen de ne pas modifier les matrices.

2.2.3 Une seconde méthode

A chaque itération, on cherche à résoudre un problème de Stokes de la forme :

$$\begin{cases} RU + B^T P = 0 \\ BU = -MQ \text{ où } Q \text{ est pris tel que } (MQ, 1) = 0. \end{cases} \tag{II.1}$$

1. On pose $\tilde{B}$ la matrice B privée de sa dernière ligne et $\widetilde{MQ}$ le vecteur MQ privé de sa dernière ligne.
2. On cherche $(U, \tilde{P})$ solution du problème de Stokes :

$$\begin{cases} RU + \tilde{B}^T \tilde{P} = 0 \\ \tilde{B}U = -\widetilde{MQ} \end{cases} \tag{II.2}$$

3. on pose $\tilde{\tilde{P}} = \begin{pmatrix} \tilde{P} \\ 0 \end{pmatrix}$, puis $P = \tilde{\tilde{P}} - \frac{1}{|\Omega|}(M\tilde{\tilde{P}}, 1) \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$, de sorte que $(MP, 1) = 0$.

On vérifie alors que si (II.2) est vérifié pour $\tilde{P}$, (II.1) l'est aussi avec le P que l'on a choisi. En effet, on a tout d'abord :

$$RU + B^T P = RU + B^T \tilde{\tilde{P}} - \frac{1}{|\Omega|}(M\tilde{\tilde{P}}, 1) B^T \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = RU + B^T \tilde{\tilde{P}} = RU + \tilde{B}^T \tilde{P} = 0$$

car $B^T \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = 0$ et $B^T \tilde{\tilde{P}} = \tilde{B}^T \tilde{P}$.

Il nous reste donc à montrer que $BU = -MQ$ sachant que $\tilde{B}U = -\widetilde{MQ}$, c'est à dire qu'il nous suffit de montrer que la dernière ligne de BU est égale à la dernière ligne de $-MQ$.
Or, on a choisi Q tel que $(MQ, 1) = 0$ et on a également $(BU, 1) = (U, B^T 1) = 0$, donc :

$$(BU + MQ, 1) = 0 \Rightarrow (BU + MQ)_{N_p} = 0 \text{ car } \forall i \in \{1, \ldots, N_p - 1\}, (BU)_i = -(MQ)_i.$$

2.2.4 Les autres valeurs propres : méthode de déflation

Pour trouver la $k^{\text{ième}}$ plus petite valeur propre de la matrice $M^{-1/2} B R^{-1} B^T M^{-1/2}$, il faut faire une méthode de la puissance inverse en se plaçant dans l'orthogonal du sous-espace propre engendré par les $(k-1)^{\text{ième}}$ plus petites valeurs propres de cette matrice.

Méthode pour trouver la deuxième plus petite valeur propre de cette matrice :

1. On récupère le dernier vecteur P (qui est de norme 1) obtenu avec la méthode de la puissance inverse faite pour trouver la plus petite valeur propre. C'est le vecteur propre associé à cette valeur propre, on le nomme $v_{propre1}$.
2. On prend un vecteur quelconque Q_0 à moyenne nulle.
3. On place Q_0 dans l'orthogonal du sous-espace engendré par $v_{propre1}$: $Q_0 = Q_0 - (MQ_0, v_{propre1}) \frac{v_{propre1}}{(Mv_{propre1}, v_{propre1})}$.
4. On norme ce nouveau vecteur Q_0.
5. On effectue une nouvelle méthode de la puissance inverse avec Q_0 comme vecteur initial.
6. A l'itération suivante, on utilise le vecteur $Q_1 = P_0 - (MP_0, v_{propre1}) \frac{v_{propre1}}{(Mv_{propre1}, v_{propre1})}$ que l'on normalise où P_0 est le vecteur solution du problème de Stokes.
7. On s'arrête alors lorsque le critère d'arrêt est vérifié.

Méthode pour trouver la troisième plus petite valeur propre de cette matrice :

1. On récupère le dernier vecteur P de cette deuxième méthode de la puissance. On le normalise et on le nomme $v_{propre2}$.
2. On prend un vecteur quelconque Q_0 à moyenne nulle.

3. On place Q_0 dans l'orthogonal du sous-espace engendré par $v_{propre1}$ et $v_{propre2}$:

$$Q_0 = Q_0 - (MQ_0, v_{propre1})\frac{v_{propre1}}{(Mv_{propre1}, v_{propre1})} - (MQ, v_{propre2})\frac{v_{propre2}}{(Mv_{propre2}, v_{propre2})}$$

puis on le normalise.

4. On effectue une nouvelle méthode de la puissance avec ce vecteur Q_0 comme vecteur initial puis on pose :

$$Q_1 = P_0 - (MP_0, v_{propre1})\frac{v_{propre1}}{(Mv_{propre1}, v_{propre1})} - (MP_0, v_{propre2})\frac{v_{propre2}}{(Mv_{propre2}, v_{propre2})}$$

que l'on normalise.

II.3 La méthode de Rayleigh-Ritz : Itérations de sous-espaces

2.3.1 Principe général

La méthode d'itérations de sous-espaces (voir [Saa92, p115-121]) consiste à trouver les k plus petites valeurs propres d'une matrice $A \in \mathcal{M}_{N\times N}$ symétrique dont les valeurs propres sont telles que $|\lambda_1| \leq \ldots \leq |\lambda_k| < |\lambda_{k+1}| \leq \ldots \leq |\lambda_N|$.
La première idée que l'on pourrait avoir est de prendre k vecteurs $Q_0 = [Q_0^1, \ldots, Q_0^k]$ auxquels on applique la méthode de la puissance inverse, c'est à dire qu'on les normalise à chaque itération :

- à l'itération l on cherche $\tilde{Q}_{l+1}$ tel que $A\tilde{Q}_{l+1} = Q_l$,
- puis pour tout $i \in \{1, \ldots, k\}$, $Q_{l+1}^i = \frac{\tilde{Q}_{l+1}^i}{\|\tilde{Q}_{l+1}^i\|}$.

Posons $E_l = Vect(Q_l) = Vect((A^{-1})^l Q_0)$. Alors, on va bien obtenir une base de l'espace E_l mais le problème de cette méthode est que tous les vecteurs Q^i vont, en général, converger vers le même vecteur qui est le vecteur propre associé à la plus petite valeur propre.
Bauer a alors l'idée de rétablir l'indépendance de ces vecteurs et d'obtenir une base orthonormée de l'espace E_l par un processus tel que la factorisation QR :

- à l'itération l on cherche $\tilde{Q}_{l+1}$ tel que $A\tilde{Q}_{l+1} = Q_l$,
- puis par factorisation QR, on a : $\tilde{Q}_{l+1} = Q_{l+1}R_{l+1}$.

Ceci force alors les vecteurs à rester orthogonaux et pour tout $i \in \{1, \ldots, k\}$, le vecteur Q^i converge vers le vecteur propre associé à la i^e plus petite valeur propre de A noté λ_i. L'inconvénient de cette méthode est que sa vitesse de convergence est la même que celle de la méthode de la puissance inverse, c'est à dire qu'elle est de l'ordre de $\frac{\lambda_i}{\lambda_{i+1}}$ où $|\lambda_1| \leq \ldots \leq |\lambda_k|$.
Pour remédier à cela, la méthode d'itérations de sous-espaces de Rayleigh-Ritz permet de "bien" choisir la base orthonormée de E_l et pour cela on choisit celle qui diagonalise la forme bilinéaire $a_{E_l\times E_l}$ où a est la forme bilinéaire associée à la matrice A. On commence par partir d'une base orthogonale $\tilde{Q}_l$ de E_l (c'est en fait celle obtenue par la méthode de Bauer) et on regarde la matrice de $a_{E_l\times E_l}$ dans cette base :

$$\left(mat(a_{E_l\times E_l}, \tilde{Q}_l)\right)_{i,j} = a(\tilde{Q}_l^i, \tilde{Q}_l^j) = (\tilde{Q}_l^i)^T A\tilde{Q}_l^j = e_i^T\left((\tilde{Q}_l)^T A\tilde{Q}_l\right)e_j \text{ où } (e_i)_{i=1,\ldots,k} \text{ est la base canonique } \mathbb{R}^k.$$

On diagonalise alors cette matrice $\tilde{Q}_l^T A\tilde{Q}_l$ dans une base orthonormée Y_l puis on prend $Q_l = \tilde{Q}_l Y_l$ comme nouvelle base de E_l.
Algorithme :

- On choisit une matrice initiale $Q_0 = [Q_0^1, \ldots, Q_0^k] \in \mathcal{M}_{N\times k}$.
- On pose $P_1 = A^{-1}Q_0$ c'est à dire que l'on cherche P_1 telle que $AP_1 = Q_0$.
- Par une factorisation QR, on trouve $\tilde{Q}_1$ et $\tilde{R}_1$ tels que $P_1 = \tilde{Q}_1\tilde{R}_1$ de sorte que $\tilde{Q}_1^T\tilde{Q}_1 = Id$.
- On pose $B_1 = \tilde{Q}_1^T A\tilde{Q}_1$.
- On cherche les vecteurs de propres Y_1 de B_1, c'est à dire que $[Y_1, D_1] = \texttt{spec}(B_1)$ où les coefficients diagonaux de D_1 sont les valeurs propres de B_1, on a donc : $Y_1^T Y_1 = Id$ et $Y_1^T B_1 Y_1 = D_1$.
- On pose $Q_1 = \tilde{Q}_1 Y_1$ de sorte que :
  - $Q_1^T Q_1 = Y_1^T\tilde{Q}_1^T\tilde{Q}_1 Y_1 = Id$,
  - $Q_1^T AQ_1 = Y_1^T B_1 Y_1 = D_1$.
- On continue jusqu'à ce que le critère d'arrêt soit vérifié :

$$\forall l \in \{1, ..., k\},\ \frac{|vp_i^l - vp_{i-1}^l|}{|vp_i^l|} < \varepsilon \text{ où } vp_i^l \text{ est la } l^e \text{ plus petite valeur propre de } D_i.$$

- Les k premières valeurs propres sont alors les coefficients diagonaux de la dernière matrice D calculée et les vecteurs propres sont les colonnes de Q correspondantes.

Notations :
- $\mathcal{S}_l$ sous-espace engendré par Q_l,
- $\mathcal{P}_l$ projection orthogonale sur $\mathcal{S}_l$,
- Soient $|\lambda_1| \leq |\lambda_2| \leq \ldots \leq |\lambda_k| < |\lambda_{k+1}| \leq \ldots \leq |\lambda_N|$ les valeurs propres de A et u_i les vecteurs propres associés,
- P le projecteur spectral associé à $|\lambda_1|, \ldots, |\lambda_k|$.

Théorème 2.1 ([Saa92, p119-120])
Soit $\mathcal{S}_0 = Vect\{q_1, \ldots, q_k\}$ et on suppose que les vecteurs $\{Pq_i\}_{i=1,\ldots,k}$ sont linéairement indépendants.
Alors, pour tout vecteur propre u_i de A, pour tout $i \in \{1, \ldots, k\}$, il existe un unique vecteur $s_i \in \mathcal{S}_0$ tel que $Ps_i = u_i$.
De plus, l'inégalité suivante est vérifiée :

$$\|(I - \mathcal{P}_l)u_i\|_2 \leq \|u_i - s_i\|_2 \left|\frac{\lambda_i}{\lambda_{k+1}}\right|^l.$$

Preuve :
Les vecteurs $\{Pq_i\}_{i=1,\ldots,k}$ étant supposés indépendants, ils forment une base du sous-espace $P\mathbb{C}^n$ auquel appartient u_i, donc :

$$u_i = \sum_{j=1}^{k} \eta_j Pq_j = P\sum_{j=1}^{k} \eta_j q_j \equiv Ps_i.$$

Posons $s_i = u_i + w$, alors $w = s_i - u_i = s_i - Ps_i = (I - P)s_i$.
Posons $\tilde{A} = A^{-1}$ et $\beta_i = \frac{1}{\lambda_i}$ les valeurs propres de $\tilde{A}$ telles que $|\beta_1| \geq |\beta_2| \geq \ldots \geq |\beta_k| > |\beta_{k+1}| \geq \ldots \geq |\beta_N|$.
Soit $y \in \mathcal{S}_l = Vect(Q_l) = Vect(\tilde{A}^l Q_0)$ défini par $y = \left(\frac{1}{\beta_i}\right)^l \tilde{A}^l s_i$. Ensuite, par définition de s_i, on a :

$$y = \left(\frac{1}{\beta_i}\right)^l \tilde{A}^l(u_i + w) \Rightarrow y - \left(\frac{1}{\beta_i}\right)^l \tilde{A}^l u_i = \left(\frac{1}{\beta_i}\right)^l \tilde{A}^l w.$$

Or, u_i est un vecteur propre de A donc de $\tilde{A}$ d'où : $\tilde{A}u_i = \beta_i u_i \Rightarrow \tilde{A}^l u_i = \beta_i^l u_i$ et donc :

$$y - u_i = \left(\frac{1}{\beta_i}\right)^l \tilde{A}^l w$$

Notons $\mathcal{W}$ le sous-espace propre correspondant aux valeurs propres $\beta_{k+1}, \ldots, \beta_N$.
On remarque alors que $w \in \mathcal{W}$ par définition de w et :

$$y - u_i = \left(\frac{1}{\beta_i}\right)^l [\tilde{A}_{|\mathcal{W}}]^l w,$$

puis, par l'inégalité de Cauchy-Schwarz :

$$\|y - u_i\|_2 \leq \left\| \left[\frac{1}{\beta_i}\tilde{A}_{|\mathcal{W}}\right]^l \right\|_2 \|w\|_2.$$

Les valeurs propres de $\tilde{A}_{|\mathcal{W}}$ étant $\beta_{k+1}, \ldots, \beta_N$, le rayon spectral de $\left[\frac{1}{\beta_i}\tilde{A}_{|\mathcal{W}}\right]$ est $\frac{\beta_{k+1}}{\beta_i}$ et on a :

$$\left\| \left[\frac{1}{\beta_i}\tilde{A}_{|\mathcal{W}}\right]^l \right\|_2 = \left(\frac{\beta_{k+1}}{\beta_i}\right)^l.$$

en utilisant que $||A||_2 = \rho(A)$.
Enfin, on sait que $\|(I - \mathcal{P}_l)u_i\|_2 = \min_{y \in \mathcal{S}_l} ||y - u_i||_2$ et donc :

$$\|(I - \mathcal{P}_l)u_i\|_2 \leq ||y - u_i||_2 \leq \left(\frac{\beta_{k+1}}{\beta_i}\right)^l \|w\|_2 = \left(\frac{\beta_{k+1}}{\beta_i}\right)^l \|s_i - u_i\|_2.$$

Sachant que les valeurs propres de A^{-1} sont égales à l'inverse de celles de A, on obtient :

$$\|(I - \mathcal{P}_l)u_i\|_2 \leq \left(\frac{\lambda_i}{\lambda_{k+1}}\right)^l \|s_i - u_i\|_2$$

■

Théorème 2.2 ([LT94, p509-543])
Supposons que les vecteurs $\{Pq_i\}_{i=1,\dots,k}$ *soient linéairement indépendants.*
Alors, pour tout $i \in \{1,\dots,k\}$, *il existe* $(\tilde{\lambda}_i^{(l)}, \tilde{u}_i^l)$ *élément propre de la matrice* $B_l = \tilde{Q}_l^T A \tilde{Q}_l$ *obtenue après* l *itérations de la méthode du sous-espace tel que :*

$$\left|\lambda_i - \tilde{\lambda}_i^{(l)}\right| = 0\left(\left|\frac{\lambda_i}{\lambda_{k+1}}\right|^l\right).$$

Preuve :
Soit $i \in \{1,\dots,k\}$, alors on à l'itération l on a $B_l = \tilde{Q}_l^T A \tilde{Q}_l$ avec $\tilde{Q}_l^T \tilde{Q}_l = Id$ et $\tilde{Q}_l \tilde{Q}_l^T = \mathcal{P}_l$ et telle que $B_l = Y_l D_l Y_l^{-1}$.

- Soit y_i tel que $\tilde{Q}_l y_i = \frac{\mathcal{P}_l u_i}{||\mathcal{P}_l u_i||}$, alors :

$$\begin{cases} ||\tilde{Q}_l y_i||_2^2 = (\tilde{Q}_l y_i, \tilde{Q}_l y_i) = (y_i, y_i) = ||y_i||_2^2 \\ ||\tilde{Q}_l y_i||_2 = 1 \end{cases} \Rightarrow ||y_i||_2 = 1.$$

- Posons $\mu = (B_l y_i, y_i)$.

  Lemme 2.1
  Il existe $\tilde{\lambda}_i^{(l)} \in Spec(B_l)$ *telle que :*

$$|\tilde{\lambda}_i^{(l)} - \mu| \leq \|Y_l\|_2 \left\|Y_l^{-1}\right\|_2 \|r\|_2 \text{ avec } r = B_l y_i - \mu y_i. \tag{II.3}$$

  Preuve :

  - Si $\min_i |\tilde{\lambda}_i^{(l)} - \mu| = 0$, on a le résultat.
  - Si $\min_i |\tilde{\lambda}_i^{(l)} - \mu| > 0$, alors $\mu \notin Spec(B_l)$.

    Donc, $B_l y_i - \mu y_i \neq 0$ et $B_l - \tilde{\lambda}_i^{(l)} I$ n'est pas inversible.
    Comme $r = B_l y_i - \mu y_i = (Y_l D_l Y_l^{-1} y_i - \mu y_i) = Y_l (D_l - \mu I) Y_l^{-1} y_i$, alors $y_i = Y_l (D_l - \mu I)^{-1} Y_l^{-1} r$.
    On a donc :

$$1 = \|y_i\|_2 \leq \|Y_l\|_2 \left\|(D_l - \mu I)^{-1}\right\|_2 \left\|Y_l^{-1}\right\|_2 \|r\|_2$$

    et $\left\|(D_l - \mu I)^{-1}\right\|_2 = \frac{1}{\min_i |\tilde{\lambda}_i^{(l)} - \mu|}$, d'où :

$$\min_i |\tilde{\lambda}_i^{(l)} - \mu| \leq \|Y_l\|_2 \left\|Y_l^{-1}\right\|_2 \|r\|_2 .$$

■

- Soit $\nu \in \mathbb{C}$, alors :

$$\begin{aligned} B_l y_i - \nu y_i &= B_l y_i - \mu y_i + \mu y_i - \nu y_i = (B_l y_i - \mu y_i) + (\mu - \nu) y_i = r + (\mu - \nu) y_i \\ \text{et } (r, y_i) &= (B_l y_i - \mu y_i, y_i) = (B_l y_i, y_i) - \mu ||y_i||_2^2 = 0 \\ \Rightarrow \|B_l y_i - \nu y_i\|_2^2 &= ||r||_2^2 + |\mu - \nu|^2 \geq ||r||_2^2 \text{ par le théorème de Pythagore.} \end{aligned}$$

  On a ainsi :

$$\|r\|_2 \leq \|B_l y_i - \nu y_i\|_2 \text{, pour tout } \nu \in \mathbb{C}.$$

  Prenons $\nu = \lambda_i \in Spec(A)$, alors :

$$\|r\|_2 \leq \|B_l y_i - \lambda_i y_i\|_2 . \tag{II.4}$$

- On montre alors le résultat suivant :

  Lemme 2.2

$$\|B_l y_i - \lambda_i y_i\|_2 \leq \alpha \frac{\|(I - \mathcal{P}_l) u_i\|_2}{\|\mathcal{P}_l u_i\|_2} \text{ avec } \alpha = \|\mathcal{P}_l A (I - \mathcal{P}_l)\|_2 . \tag{II.5}$$

  Preuve :
  Posons $A_l = \mathcal{P}_l A \mathcal{P}_l = (\tilde{Q}_l \tilde{Q}_l^T) A (\tilde{Q}_l \tilde{Q}_l^T) = \tilde{Q}_l B_l \tilde{Q}_l^T$, alors $B_l = \tilde{Q}_l^T A_l \tilde{Q}_l$; donc :

$$(B_l - \lambda_i I)y_i = (\tilde{Q}_l^T A_l \tilde{Q}_l - \lambda_i I)y_i = \tilde{Q}_l^T (A_l - \lambda_i I)\tilde{Q}_l y_i = \tilde{Q}_l^T (A_l - \lambda_i I)\frac{\mathcal{P}_l u_i}{\|\mathcal{P}_l u_i\|_2}$$

$$\Rightarrow \|(B_l - \lambda_i I)y_i\|_2 = \frac{\left\|\tilde{Q}_l^T (A_l - \lambda_i I)\mathcal{P}_l u_i\right\|_2}{\|\mathcal{P}_l u_i\|_2} \leq \frac{\|(A_l - \lambda_i I)\mathcal{P}_l u_i\|_2}{\|\mathcal{P}_l u_i\|_2} \text{ car } ||\tilde{Q}_l^T|| = ||\tilde{Q}_l|| = 1.$$

Or, $(A_l - \lambda_i I)\mathcal{P}_l u_i = (\mathcal{P}_l A \mathcal{P}_l - \lambda_i I)\mathcal{P}_l u_i = \mathcal{P}_l (A - \lambda_i I)\mathcal{P}_l u_i$ car $\mathcal{P}_l^2 = \mathcal{P}_l$, et :

$$(A - \lambda_i I)u_i = 0 \Rightarrow \mathcal{P}_l (A - \lambda_i I)u_i = 0$$

d'où :

$$\begin{aligned}(A_l - \lambda_i I)\mathcal{P}_l u_i &= \mathcal{P}_l (A - \lambda_i I)(\mathcal{P}_l u_i - u_i) = -\mathcal{P}_l (A - \lambda_i I)(I - \mathcal{P}_l)u_i \\ &= -\mathcal{P}_l (A - \lambda_i I)(I - \mathcal{P}_l)(I - \mathcal{P}_l)u_i \text{ car } (I - \mathcal{P}_l)^2 = I - \mathcal{P}_l \\ &= -\mathcal{P}_l A (I - \mathcal{P}_l)(I - \mathcal{P}_l)u_i \text{ car } \mathcal{P}_l (A - \lambda_i I)(I - \mathcal{P}_l) = \mathcal{P}_l A (I - \mathcal{P}_l) - \lambda_i \mathcal{P}_l (I - \mathcal{P}_l) = \mathcal{P}_l A (I - \mathcal{P}_l).\end{aligned}$$

On obtient alors :

$$\|(A_l - \lambda_i I)\mathcal{P}_l u_i\|_2 \leq \|\mathcal{P}_l A (I - \mathcal{P}_l)\|_2 \, \|(I - \mathcal{P}_l)u_i\|_2 .$$

■

– En regroupant ces résultats, on a alors :

$$\begin{aligned}|\mu - \tilde{\lambda}_i^{(l)}| &\leq \|Y_l\|_2 \left\|Y_l^{-1}\right\|_2 \|r\|_2 \text{ d'après (II.3)} \\ &\leq \|Y_l\|_2 \left\|Y_l^{-1}\right\|_2 \|B_l y_i - \lambda_i y_i\|_2 \text{ d'après (II.4)} \\ &\leq \alpha \|Y_l\|_2 \left\|Y_l^{-1}\right\|_2 \frac{\|(I - \mathcal{P}_l)u_i\|_2}{\|\mathcal{P}_l u_i\|_2} \text{ d'après (II.5).}\end{aligned}$$

– De plus, on sait que $\mu = (B_l y_i, y_i)$, donc :

$$\begin{aligned}\mu &= (\tilde{Q}_l^T \mathcal{P}_l A \mathcal{P}_l \tilde{Q}_l y_i, y_i) = (\mathcal{P}_l A \mathcal{P}_l \tilde{Q}_l y_i, \tilde{Q}_l y_i) = \left(\mathcal{P}_l A \mathcal{P}_l \frac{\mathcal{P}_l u_i}{\|\mathcal{P}_l u_i\|_2}, \frac{\mathcal{P}_l u_i}{\|\mathcal{P}_l u_i\|_2}\right) \\ &= \left(\mathcal{P}_l \frac{A\mathcal{P}_l u_i}{\|\mathcal{P}_l u_i\|_2}, \frac{\mathcal{P}_l u_i}{\|\mathcal{P}_l u_i\|_2}\right) = \left(\frac{A\mathcal{P}_l u_i}{\|\mathcal{P}_l u_i\|_2}, \frac{\mathcal{P}_l^2 u_i}{\|\mathcal{P}_l u_i\|_2}\right) \\ &= \frac{(A\mathcal{P}_l u_i, \mathcal{P}_l u_i)}{\|\mathcal{P}_l u_i\|_2^2}.\end{aligned}$$

– On obtient ainsi :

$$\lambda_i - \mu = \lambda_i \frac{\|\mathcal{P}_l u_i\|_2^2}{\|\mathcal{P}_l u_i\|_2^2} - \frac{(A\mathcal{P}_l u_i, \mathcal{P}_l u_i)}{\|\mathcal{P}_l u_i\|_2^2} = \frac{(\lambda_i \mathcal{P}_l u_i - A\mathcal{P}_l u_i, \mathcal{P}_l u_i)}{\|\mathcal{P}_l u_i\|_2^2} = \frac{((\lambda_i I - A)\mathcal{P}_l u_i, \mathcal{P}_l u_i)}{\|\mathcal{P}_l u_i\|_2^2}.$$

Or,

$$(A - \lambda_i I)\mathcal{P}_l u_i = (A - \lambda_i I)(u_i - u_i + \mathcal{P}_l u_i) = -(A - \lambda_i I)(I - \mathcal{P}_l)u_i + \underbrace{(A - \lambda_i)u_i}_{=0}$$

et donc :

$$\lambda_i - \mu = \frac{((A - \lambda_i I)(I - \mathcal{P}_l)u_i, \mathcal{P}_l u_i)}{\|\mathcal{P}_l u_i\|_2^2}.$$

par l'inégalité de Cauchy-Schwartz, on obtient alors :

$$|\lambda_i - \mu| \leq \frac{\|(A - \lambda_i I)(I - \mathcal{P}_l)u_i\|_2 \, \|\mathcal{P}_l u_i\|_2}{\|\mathcal{P}_l u_i\|_2^2} \leq \frac{\|(A - \lambda_i I)\|_2 \, \|(I - \mathcal{P}_l)u_i\|_2}{\|\mathcal{P}_l u_i\|_2}.$$

– On a enfin :

$$\begin{aligned}|\lambda_i - \lambda_i^{(l)}| &\leq |\lambda_i - \mu| + |\mu - \lambda_i^{(l)}| \\ &\leq \underbrace{\left(\|(A-\lambda_i I)\|_2 + \alpha \|Y_l\|_2 \left\|Y_l^{-1}\right\|_2\right)}_{=\beta} \frac{\|(I-\mathcal{P}_l)u_i\|_2}{\|\mathcal{P}_l u_i\|_2}\end{aligned}$$

– En appliquant le théorème 2.1 on a alors que :

$$\|(I-\mathcal{P}_l)u_i\|_2 = 0\left(\left|\frac{\lambda_i}{\lambda_{k+1}}\right|^l\right),$$

d'où le résultat.

■

Conclusion :
Ainsi, le facteur de convergence de la plus petite valeur propre est en $\left|\frac{\lambda_1}{\lambda_{k+1}}\right|$ qui est bien sûr plus petit que $\left|\frac{\lambda_i}{\lambda_{k+1}}\right|$ pour $i = 1, \ldots, k$.
Pour l fixé, on obtient donc les valeurs propres avec une précision plus grande sur λ_1, puis λ_2, etc...
Lorsque l'on cherche les p plus petits éléments propres, il vaut donc mieux prendre $k > p$ de façon à améliorer le facteur de convergence.

Regardons maintenant l'algorithme pour la matrice non symétrique $M^{-1}BR^{-1}B^T$. Il faut appliquer ce que l'on faisait pour une matrice A symétrique mais cette fois avec le produit scalaire associé à la matrice M :

– On choisit $Q_0 = [Q_0^1, \ldots, Q_0^k]$ aléatoirement M-orthogonal.
– On cherche P_1 telle que $(M^{-1}BR^{-1}B^T)P_1 = Q_0$, c'est à dire que l'on résout le problème de Stokes :

$$(*)\begin{cases} RU_1 + B^T P_1 = 0 \\ BU_1 = -MQ_0 \end{cases}.$$

– On cherche $\tilde{Q}_1 \in \mathcal{M}_{N_p,k}(\mathbb{R})$ et $\tilde{R}_1 \in \mathcal{M}_{k,k}(\mathbb{R})$ telles que $\tilde{Q}_1^T M \tilde{Q}_1 = Id$ et $P_1 = \tilde{Q}_1 \tilde{R}_1$. Ceci revient à effectuer un procédé de Gram-Schmidt sur P_1 pour le produit scalaire de M.
– On pose alors $B_1 = \tilde{Q}_1^T M(M^{-1}BR^{-1}B^T)\tilde{Q}_1 = \tilde{Q}_1^T BR^{-1}B^T\tilde{Q}_1$ qui est bien symétrique.
On peut par contre remarquer que le calcul de $BR^{-1}B^T$ va être lourd (car il faut inverser R qui, lorsque l'on raffine, est de plus en plus grosse). Il faut donc essayer de se ramener à un autre calcul :

$$\begin{aligned}B_1 &= \tilde{Q}_1^T BR^{-1}B^T\tilde{Q}_1 = \tilde{Q}_1^T BR^{-1}(B^T P_1)\tilde{R}_1^{-1} \text{ d'après l'étape précédente} \\ &= \tilde{Q}_1^T BR^{-1}(-RU_1)\tilde{R}_1^{-1} \text{ d'après la première équation de } (*) \\ &= -\tilde{Q}_1^T(BU_1)\tilde{R}_1^{-1} = -\tilde{Q}_1^T(-MQ_0)\tilde{R}_1^{-1} \text{ d'après la seconde équation de } (*) \\ &= \tilde{Q}_1^T MQ_0\tilde{R}_1^{-1}.\end{aligned}$$

On remarque que ce calcul est moins lourd que le précédent car la matrice $\tilde{R}_1$ est petite (on utilisera principalement $k = 3$) et donc facile à inverser.
– On cherche les vecteurs propres de B_1 : $[Y_1, D_1] = \texttt{spec}(B_1)$.
$B_1 \in \mathcal{M}_{k,k}(\mathbb{R})$ donc l'uilisation de la commande `spec` n'augmente pas le temps de calcul du programme.
– On pose $Q_1 = \tilde{Q}_1 Y_1$ de sorte que :
• $Q_1^T MQ_1 = Y_1^T\tilde{Q}_1^T M\tilde{Q}_1 Y_1 = Id$,
• $Q_1^T M(M^{-1}BR^{-1}B^T)Q_1 = Y_1^T\tilde{Q}_1^T(BR^{-1}B^T)\tilde{Q}_1 Y_1 = Y_1^T B_1 Y_1 = D_1$.
– On continue jusqu'à ce que le critère d'arrêt soit vérifié.

Regardons maintenant comment remédier au problème des pressions constantes pour pouvoir résoudre le problème $(*)$.

2.3.2 Le problème des pressions constantes

Ce problème va intervenir lorsque l'on veut, à chaque itération, trouver P tel que $M^{-1}BR^{-1}B^T P = Q$. En effet, le vecteur constant unité étant dans le noyau de B^T, la matrice $M^{-1}BR^{-1}B^T$ n'est pas inversible.
Or, cette étape revient à résoudre le problème de Stokes $\begin{cases} RU + B^T P = 0 \\ BU = -MQ \end{cases}$, on va donc raisonner de la même manière que pour la méthode de la puissance inverse.

1. On choisit $Q_0 = [Q_0^1, \dots, Q_0^k]$ où $\forall i \in \{1, \dots, k\}$, Q^i est choisi aléatoirement à moyenne nulle.
2. On résout le problème : Trouver U_1 et $\tilde{P}_1$ tels que $\begin{cases} RU_1 + \tilde{B}^T \tilde{P}_1 = 0 \\ \tilde{B}U_1 = -\widetilde{MQ}_0 \end{cases}$.
3. On pose $\tilde{\tilde{P}}_1 = \begin{pmatrix} \tilde{P}_1 \\ \underbrace{0 \dots 0}_{kfois} \end{pmatrix}$ puis $P_1 = \tilde{\tilde{P}}_1 - \frac{1}{|\Omega|} \left[(M\tilde{\tilde{P}}_1^1, 1) \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, \dots, (M\tilde{\tilde{P}}_0^k, 1) \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \right]$ où $\tilde{\tilde{P}}_1 = [\tilde{\tilde{P}}_1^1, \dots, \tilde{\tilde{P}}_1^k]$.

   On remarque que l'on a construit P_1 de manière à ce que pour tout $i \in \{1, \dots, k\}$, on ait $(MP_1^i, 1) = 0$.
   Montrons qu'ainsi les vecteurs colonnes de la matrice $Q_1 = \tilde{Q}_1 Y_1$ seront également tels que $(MQ_1^i, 1) = 0$.
   Preuve :

   – Calculons d'abord $\tilde{Q}_1 \tilde{R}_1$ sachant que $\tilde{Q}_1 = [\tilde{Q}_1^1 \dots \tilde{Q}_1^k]$ et $\tilde{R}_1 = \begin{pmatrix} r_{1,1} & r_{1,2} & \dots & r_{1,k} \\ 0 & r_{2,2} & \dots & r_{2,k} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & r_{k,k} \end{pmatrix}$.

   On obtient, $\tilde{Q}_1 \tilde{R}_1 = [\tilde{Q}_1^1 r_{1,1}, \tilde{Q}_1^1 r_{1,2} + \tilde{Q}_1^2 r_{2,2}, \dots, \sum_{j=1}^{k} \tilde{Q}_1^j r_{j,k}]$.

   – On sait que $P_1 = \tilde{Q}_1 \tilde{R}_1$, donc si $P_1 = [P_1^1, \dots, P_1^k]$, on a :

$$\forall i \in \{1, \dots, k\}, P_1^i = \sum_{j=1}^{i} \tilde{Q}_1^j r_{j,i} \Longrightarrow \forall i \in \{1, \dots, k\}, (MP_1^i, 1) = \sum_{j=1}^{i} (M\tilde{Q}_1^j, 1) r_{j,i}.$$

   – Rappelons que les valeurs propres de la matrice $M^{-1}BR^{-1}B^T$ sont strictement positives, alors :
   - $(MP_1^1, 1) = 0 \Longrightarrow (M\tilde{Q}_1^1, 1) = \frac{1}{r_{1,1}}(MP_1^1, 1) = 0$,
   - $(M\tilde{Q}_1^2, 1) = \frac{1}{r_{2,2}}((MP_1^2, 1) - r_{1,1}(M\tilde{Q}_1^1, 1)) = 0$ car $(P_1^2, 1) = 0$,
   - sachant que $\forall i \in \{1, \dots, k\}$, $(MP_1^i, 1) = 0$, une récurrence immédiate nous donne que :

$$\forall i \in \{1, \dots, k\}, (M\tilde{Q}_1^i, 1) = 0,$$

   – On vient de montrer que $(M\tilde{Q}_1^i, 1) = 0$ pour tout $i \in \{1, \dots, k\}$, il reste à montrer que c'est aussi le cas pour Q_1^i.

   Rappelons que $\tilde{Q}_1 = [\tilde{Q}_1^1, \dots, \tilde{Q}_1^k]$ et on note $Y_1 = [Y_1^1, \dots, Y_1^k]$ avec pour tout $i \in \{1, \dots, k\}$, $Y_1^i = \begin{pmatrix} Y_1^{i,1} \\ \vdots \\ Y_1^{i,k} \end{pmatrix}$.

   Alors, sachant que $Q_1 = \tilde{Q}_1 Y_1$, on a pour tout $i \in \{1, \dots, k\}$,

$$Q_1^i = \sum_{j=1}^{k} \tilde{Q}_1^j Y_1^{i,j} \Rightarrow (MQ_1^i, 1) = \sum_{j=1}^{k} (M\tilde{Q}_1^j, 1) Y_1^{i,j} = 0.$$

■

Il suffit maintenant de vérifier que le couple $\begin{pmatrix} U_1 \\ P_1 \end{pmatrix} = \begin{pmatrix} U_1^1 & \dots & U_1^k \\ P_1^1 & \dots & P_1^k \end{pmatrix}$ trouvé vérifie bien le problème précédent.

– Pour la première ligne de l'équation :

$$RU_1 + B^T P_1 = RU_1 + B^T \tilde{\tilde{P}}_1 - \frac{1}{|\Omega|} \left[(M\tilde{\tilde{P}}_1^1, 1) B^T \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, \dots, (M\tilde{\tilde{P}}_1^k, 1) B^T \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \right]$$

$$= RU_1 + B^T \tilde{\tilde{P}}_1 = RU_1 + \tilde{B}^T \tilde{P}_1 = 0 \text{ car } B^T \tilde{\tilde{P}}_1 = \tilde{B}^T \tilde{P}_1.$$

– Sachant que $\tilde{B}U_1 = -\widetilde{MQ_0}$, il reste à montrer que pour la dernière ligne, on a :

$$[(BU_1)^1, \dots, (BU_1)^k] = BU_1 = -MQ_0 = [-(MQ_0)^1, \dots, -(MQ_0)^k],$$

c'est à dire que : $\forall i \in \{1, \dots, k\}$, $(BU_1)^i_{N_p} = -(MQ_0)^i_{N_p}$.
Or, $(BU_1)^i_{N_p} = (BU_1^i)_{N_p}$ et $(MQ_0)^i_{N_p} = (MQ_0^i)_{N_p}$, il faut donc montrer que :

$$\forall i \in \{1, \dots, k\}, (BU_1^i)_{N_p} = -(MQ_0^i)_{N_p},$$

en sachant que $\forall i \in \{1,\dots,k\}, \forall l \in \{1,\dots,N_p-1\}, (BU_1^i)_l = -(MQ_0^i)_l(1)$.
Soit $i \in \{1,\dots,k\}$, on sait que $(BU_1^i, 1) = 0 = (MQ_0^i, 1)$ donc :

$$(BU_1^i + MQ_0^i, 1) = 0 \Rightarrow (BU_1^i + MQ_0^i)_{N_p} = 0 \text{ d'après } (1).$$

4. On peut maintenant continuer normalement la méthode décrite ci-dessus jusqu'à obtenir la vérification du critère d'arrêt.

II.4 Comparaison des différentes méthodes

2.4.1 Sur une matrice donnée explicitement

On construit une matrice $A \in \mathcal{M}_{N\times N}$ pour N assez grand (1000 par exemple) dont les valeurs propres sont :

$$\lambda_1 = 1,\ \lambda_2 = 1.1 \text{ et } \lambda_3 = \dots = \lambda_N = 4.$$

On commence alors par comparer les temps de calcul des trois méthodes :
- Pour la méthode Scilab, le temps moyen de calcul est de 3s.
- Pour la méthode de la puissance inverse, il est de 0.35s.
- Enfin, la méthode d'itérations de sous-espaces calculant les deux premières valeurs propres donne un temps moyen de calcul de 0.08s.

On voit donc bien que la méthode d'itérations de sous-espaces de Rayleigh-Ritz est la plus rapide, mais comparons maintenant les deux méthodes itératives par leur vitesse de convergence.
Pour le critère d'arrêt on choisit $\varepsilon = 10^{-8}$ pour les deux méthodes.
- La méthode d'itérations de sous-espaces de Rayleigh-Ritz converge en 15 itérations en moyenne.
  Or $(\frac{\lambda_1}{\lambda_3})^{15} = (\frac{1}{4})^{15} \approx 10^{-9}$ ce qui confirme bien le fait que sa vitesse de convergence est en $\frac{\lambda_1}{\lambda_3}$.
- La méthode de la puissance inverse converge en 80 à 100 itérations.
  On a alors $(\frac{\lambda_1}{\lambda_2})^{100} = (\frac{1}{1.1})^{100} \approx 10^{-5}$ et on voit bien que la vitesse de convergence, qui est en $\frac{\lambda_1}{\lambda_2}$ est plus lente car les deux valeurs propres sont proches.

2.4.2 Sur un des cas des tests Inf-Sup que l'on effectue

On va effectuer ces tests sur un élément fini triangle donné par le logiciel FreeFem++.
On choisit l'élément fini $\mathbb{P}_1-$bulle$/\mathbb{P}_1$ (décrit dans le chapitre 2) sur un domaine carré avec un maillage triangle raffiné non uniformément (voir figure 1.2 Chap. 2) dont le pas est $h = 0.128847$ et tel que $N_p = 297$ et $N_u = 1618$ avec $\varepsilon = 10^{-7}$ pour le critère d'arrêt.
- Lorsque l'on utilise la méthode Scilab, en enlevant à la main les pressions constantes dans les matrices de masse et de divergence on obtient un temps de calcul de 27.2s dont 0.625s pour enlever les pressions constantes et 26s pour calculer la matrice $M^{-1}BR^{-1}B^T$ (et même 29s pour la matrice $M^{-1/2}BR^{-1}B^TM^{-1/2}$).
  En remarquant que la matrice utilisée est relativement petite on voit que ceci est assez long. De plus, la commande `Spec` devient inutilisable lorsque la matrice devient trop grosse.
- Lorsque l'on utilise la méthode de la puissance inverse en enlevant à la main les pressions constantes, le temps de calcul devient de 0.8s pour le calcul d'une valeur propre en 28 itérations. On remarque déjà que ceci est beaucoup plus rapide que la méthode précédente.
- Lorsque l'on effectue la méthode de la puissance inverse avec la deuxième méthode en calculant les trois plus petites valeurs propres (et non simplement la première comme pour la méthode précédente) le temps de calcul est de 0.3s ce qui est déjà moins que précédemment alors que l'on fait plus de calcul.
  Le nombre d'itérations effectuées est de 20 pour la première valeur propre, 45 pour la seconde et 200 pour la dernière.
- Pour la méthode d'itérations de sous-espaces elle nous donne directement les trois plus petites valeurs propres en un temps moyen de 0.4s avec 180 itérations.

On constate que les deux dernières méthodes sont les plus efficaces mais pour vérifier laquelle l'est le plus testons les sur une matrice plus grosse.
On choisi l'élément fini $\mathbb{P}_2/\mathbb{P}_1$ avec le cercle unité pour domaine et un maillage de triangle (voir figure (1.3) Chap.2), de pas $h = 0.0616211$ avec $N_p = 2348$ et $N_u = 17818$.
Dans ce cas, le temps de calcul est à peu près le même (environ 50s) mais :
- la méthode d'itérations de sous-espaces de Rayleigh-Ritz converge en 560 itérations,
- la méthode de la puissance inverse converge en 1000 itérations pour la plus petite valeur prore, 1000 pour la seconde et 400 pour la troisième.

Encore une fois la méthode d'itérations de sous-espaces de Rayleigh-Ritz est la plus efficace, c'est donc celle que l'on utilisera majoritairement par la suite.

III Les différents tests effectués

III.1 Convergence du schéma

On va commencer par effectuer des tests pour vérifier que le schéma converge bien (ou pas) vers la bonne solution. On effectue deux tests différents pour cela :

1. On trace, pour un pas donné, soit l'erreur entre solution exacte et solution approchée (c'est à dire solution exacte $-$ solution approchée) , soit la solution exacte et la solution approchée.
2. On trace pour plusieurs valeurs du pas l'erreur entre solution exacte et solution approchée en fonction du pas.

Dans tous les cas on a besoin de connaître la solution exacte. Pour cela, on se donne un u et un p manufacturés puis on calcule le f correspondant pour que (u, p) vérifie :

$$\begin{cases} -\Delta u + \nabla p = f \text{ dans } \Omega \\ \operatorname{div} u = 0 \text{ dans } \Omega. \end{cases}$$

On va utiliser deux cas tests différents sur le carré unité :

1. Un produit de sinus et cosinus :

$$u(x,y) = \begin{pmatrix} -2\pi \sin^2 \pi x \cos \pi y \sin \pi y \\ 2\pi \sin^2 \pi y \cos \pi x \sin \pi x \end{pmatrix} \text{ et } p(x,y) = x + y - 1.$$

2. Une solution polynomiale :

$$u(x,y) = \begin{pmatrix} 1000x^2(1-x)^2 2y(1-y)(1-2y) \\ -1000y^2(1-y)^2 2x(1-x)(1-2x) \end{pmatrix} \text{ et } p(x,y) = x^2 + y^2 - \frac{2}{3}.$$

III.2 Les problèmes de valeurs propres

On effectue principalement trois tests sur les valeurs propres :

1. Lors de la méthode de la puissance pour trouver la valeur de la constante Inf-Sup on récupère différentes données :
   - à chaque itération la valeur de $\lambda_i = \frac{1}{||P_i||}$ (où $||P_i||$ est le vecteur P utilisé à l'itération i) qui convergera vers la bonne valeur propre à la dernière itération,
   - le terme $\frac{|\frac{1}{||P_i||_2} - \frac{1}{||P_{i-1}||_2}|}{\frac{1}{||P_i||_2}}$ que l'on appelle le résidu,
   - le nombre d'itération que l'on effectue pour converger vers la bonne valeur propre.

   On peut alors tracer deux courbes :
   - la valeur du résidu en fonction des itérations,
   - la valeur du terme $\frac{|\lambda - \lambda_i|}{\lambda}$ en fonction des itérations.
2. On trace le mode propre : On récupère le dernier vecteur P normalisé de la méthode de la puissance ou de la méthode d'itérations de sous-espaces de Rayleigh-Ritz. On remarque alors que si Q est un vecteur propre associé à la valeur propre λ de la matrice $M^{-1/2}BR^{-1}B^TM^{-1/2}$, alors :

$$M^{-1/2}BR^{-1}B^TM^{-1/2}Q = \lambda Q \Leftrightarrow M^{-1}BR^{-1}B^TM^{-1/2}Q = \lambda M^{-1/2}Q,$$

   et donc le vecteur $P = M^{-1/2}Q$ est un vecteur propre de la matrice $M^{-1}BR^{-1}B^T$ associé à la valeur propre λ.
   Ainsi, le vecteur P trouvé numériquement est exactement le vecteur correspondant aux pressions du problème initial.
3. On calcule les 3 premières valeurs propres de notre problème puis :
   - on affiche pour chaque valeur propre : le pas du maillage, la valeur de la valeur propre obtenue et le nombre d'itérations effectuées pour l'obtenir,
   - on trace chacune des valeur propre en fonction du pas.

On remarque que dans ce cas on n'a pas besoin de connaître la solution exacte du problème mais seulement les matrices de rigidité, de divegence et de masse en pression du système.

Chapitre 2

Les schémas éléments finis

I Divers éléments finis utilisant le logiciel FreeFem

I.1 Utilisation du logiciel FreeFem++

FreeFem++ est un logiciel écrit en C++ qui permet de résoudre des équations aux dérivées partielles par la méthode des éléments finis.
En ce qui nous concerne, on s'en servira pour :

- construire toutes les matrices dont on a besoin : matrice de masse en pression, matrice de rigidité, matrice de divergence et lorsqu'on effectue des tests avec la solution exacte également l'interpolé du terme source, la matrice de masse en vitesse et la solution exacte. On les enregistre ensuite dans Scilab pour faire les différents tests.
- utiliser des maillages différents, plus ou moins raffinés, et connaître le pas de chacun de ces maillages ;
- tracer l'erreur entre solution exacte et solution approchée pour un pas de maillage choisi ainsi que le mode propre.

Vu que l'on effectue deux types de tests différents : un utilisant la solution exacte et l'autre non, il y a deux programmes principaux :

- le programme `sortie_matrices_M_R_B.edp` qui nous donne la matrice de masse en pression M, la matrice de rigidité R et la matrice de divergence B. Quelque soit le type de test éffectué, on lancera toujours ce programme,
- le programme `sortie_Mvit_secmembre_solexacte.edp` qui donne la matrice de masse en vitesse Mvit, le second membre et la solution exacte du problème. On ne lance ce programme que lorsqu'on utilise la solution exacte.

Ces programmes utilisent quelques commandes spécifiques :

- La commande `include` qui permet d'inclure dans le fichier un autre fichier FreeFem et ainsi ne pas modifier le programme principal lorque l'on change des donnnées telles que l'élément fini utilisé, le maillage, le raffinement, ...
- La construction du maillage T_h se fait en utilisant la commande `mesh Th=...` où après avoir indiqué la forme générale du domaine, on indique le nombre de triangles (donc le degré de raffinement) que l'on souhaite à l'intérieur.
- La commande `fespace` permet de construire les espaces éléments finis que l'on a choisis mais peut aussi être utilisée pour connaître le pas du maillage.
  L'espace des vitesses X_h étant vectoriel, il faut taper `fespace Xh=(Th,[EFv,EFv]);` pour l'espace des pressions M_h étant scalaire, il suffit de faire `fespace Mh=(Th,EFp)` où `EFv` et `EFp` sont les éléments finis choisis pour la vitesse et la pression respectivement.
  Pour obtenir le pas du maillage, il faut construire un nouveau sous-espace P_h avec l'élément fini $\mathbb{P}_0$ qui est constant par maille : `fespace Ph=(Th,P0)`, puis la commande `hTriangle` définie dans ce nouvel espace `P0` nous donne le pas du maillage.
- Pour définir des variables il suffit de donner le nom de l'espace dans lequel on veut définir la variable puis le nom que l'on souhaite donner à celle-ci. Par exemple, pour définir un vecteur p_h dans M_h, il suffit d'écrire `Mh ph`.
- La commande `varf` permet de définir un forme bilinéaire. Elle va nous permettre de construire les matrices et vecteurs dont on a besoin. Pour cela, on construit d'abord une forme bilinéaire `mat` avec `varf mat(`α,β`)=...` . Ensuite,
  - pour une matrice, on utilise `matrix<real> M=mat(EF1,EF2)` où `EF1` et `EF2` sont les éléments finis auquels appartiennent α et β respectivement ;
  - pour le second membre, on utilise la commande `real[int] vect=mat(0,EF).`
- Pour mettre des conditions aux bords de type Dirichlet dans la formulation variationnelle, on utilise la commande `on`. Dans notre problème, on a mis des conditions de Dirichlet homogène aux bords en vitesse. Or FreeFem++ prend en compte ces degrés de libertés dans les matrices et il résout le problème de Stokes par une méthode de pénalisation. Il nous faut donc eliminer des matrices ces degrés de libertés du bord :

1. On commence par mettre des conditions de Dirichlet non homogène dans la formulation variationnelle correspondant à la forme bilinéaire a (par exemple $u = 1$ sur $\partial\Omega$) pour repérer ces degrés de libertés.
2. On construit ensuite le vecteur ddl en utilisant la commande `real[int]` sur la matrice de la forme bilinéaire a. Ce vecteur est alors constitué :
   - de 0 au niveau des degrés de liberés qui ne sont pas au bord,
   - de 10^{30} au niveau des degrès de libertés du bord.
3. On enregistre alors ce vecteur et on le lit dans Scilab.
4. On repère ensuite les coordonnées des éléments non nuls du vecteur ddl grâce à la commande `find` et on les place dans un nouveau vecteur.
5. On élimine alors :
   - les lignes et les colonnes correspondant à ces coordonnées dans la matrice de rigidité et la matrice de masse en vitesse,
   - les colonnes correspondantes dans la matrices de divergence,
   - les coordonnées correspondantes pour l'interpolée du terme source ainsi que l'interpolée de la solution exacte.

– La commande `func` permet de construire une fonction. On construit ainsi la solution exacte `solex_fonction` de notre problème. Puis, pour avoir l'interpolé de la solution exacte dans le maillage `solex_vecteur`, il suffit d'écrire `solex_vecteur=solex_fonction`.
– `cout` permet d'afficher des données à l'écran alors que `ofstream` permet d'enregistrer des fichiers.

I.2 Utilisation des données

1.2.1 Les différents domaines

On va utiliser 3 domaines différents que l'on va plus ou moins raffiner :
– Le carré unité que l'on raffine en donnant le nombre de triangles que l'on souhaite avoir en abscisse nb_x et en ordonnée nb_y : `mesh T_h=square(nb_x,nb_y).`
  On va raffiner ce domaine de deux manières différentes :
  – uniformément : on commence par `nb_x=nb_y=2`, puis à chaque raffinement on les multiplie par 2,
  – non uniformément : on prend le même maillage initial puis on multiplie par 2 `nb_x` mais par 4 `nb_y`.

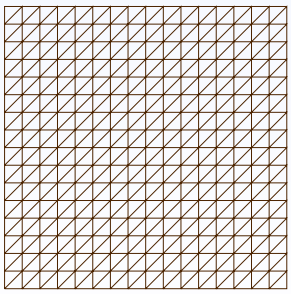


FIG. 1.1: Domaine carré uniformément raffiné

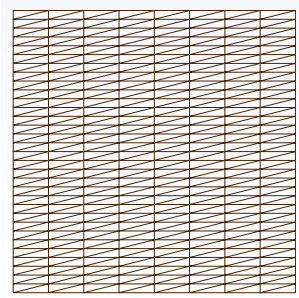


FIG. 1.2: Domaine carré raffiné non uniformément

– Le cercle unité raffiné en donnant le nombre `nb_triangle` de triangle que l'on veut sur le bord du cercle :

```
border  c1(t=0,2*pi){x=cos(t);y=sin(t);};
mesh Th=buildmesh(nb_triangle)
```

  On commence avec un maillage initial comportant 10 triangles puis on raffine au fur et à mesure en multipliant par 2 le nombre de triangles à chaque niveau de raffinement.
– Une couronne que l'on raffine en donnant le nombre `n1` de triangles sur le bord du cercle de plus grand diamètre et le nombre `n2` de triangles sur le bord interne (si `n2>0`) ou externe (si `n2<0`) du deuxième cercle :

```
border c1(t=0,2*pi){x=cos(t);y=sin(t);};
border c2(t=0,2*pi){x=0.3+0.2*cos(t);y=0.2*sin(t);};
mesh Th=buildmesh(c1(n1)+c2(n2));
```

  On raffine ce maillage de manière à ce que les triangles ne se trouvent qu'à l'intérieur de la couronne (donc `n2<0`) et que le raffinement ne soit pas le même aux extrémités des cercles.
  On commence avec un maillage initial ayant le même nombre de triangles des 2 extrémités : `n1=n2=10` puis à chaque niveau de raffinement, on multiplie `n1` par 2 et `n2` par 4.

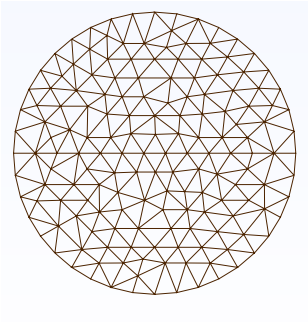


FIG. 1.3: Domaine de type cercle

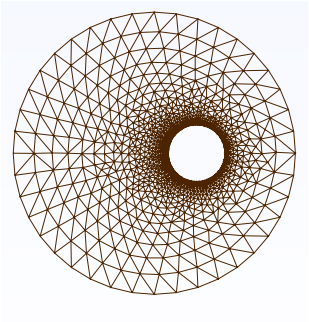


FIG. 1.4: Domaine de type couronne

1.2.2 Les différents éléments finis

Pour la pression , on utilise toujours le même élément fini $\mathbb{P}_1$. Vu que l'on se trouve en 2D, les degrés de libertés de cet élément sont les sommets de chaque triangle du maillage.

Pour la vitesse, on choisit d'associer le même élément fini à chaque composante. Pour chacune on va utiliser 3 types d'éléments finis qui diffèrent par leur nombre de degré de libertés ou leur stencil. On peut déjà noter que quelque soit le type d'élément fini utilisé pour la vitesse, le nombre d'inconnues sera multiplié par 2.

- L'élément fini $\mathbb{P}_2$: pour chacun des triangles du maillage les degrés de liberté se trouvent : à chaque sommet, au milieu de chaque arête et au centre de gravité du triangle. Les matrices que l'on cherche deviennent alors assez vite, dès que l'on raffine un peu, trop grosses et pas assez creuses (car le stencil est trop gros) ce qui ralentit l'algorithme dans un premier temps pour finir par le stopper.
- L'élément fini $\mathbb{P}_1$-iso-$\mathbb{P}_2$ remédie un peu à se problème. Dans ce cas, on remplace les degrés de libertés $\mathbb{P}_2$ par des degrés de libertés $\mathbb{P}_1$. Pour cela, on subdivise chaque triangle du maillage en quatre nouveaux triangles en reliant les milieux des côtés. On applique alors l'élément fini $\mathbb{P}_1$ à ces nouveaux triangles. Ainsi, les matrices sont un peu plus petites (car on a 6 degrés de libertés pour chaque triangle au lieu de 7) mais surtout le stencil des matrices est plus petit (c'est à dire qu'elles sont plus creuses) car on utilise l'élément fini $\mathbb{P}_1$ au lieu du $\mathbb{P}_2$.
- L'élément fini $\mathbb{P}_1$bulle donne quand à lui des matrices beaucoup plus petites car on a que 4 degrés de libertés sur chaque triangle. En effet, ce sont les mêmes degrés de libertés qu'en $\mathbb{P}_1$ (donc à chaque sommet du triangle) mais on rajoute un degré de liberté au centre de gravité du triangle.

Quelque soit celui des trois que l'on utilise, on sait que la condition Inf-Sup est vérifiée uniformément par rapport à h.

Théorème 1.1 (Inf-Sup $\mathbb{P}_2/\mathbb{P}_1$[EG04, p171])
Soit $(\mathcal{T}_h)_h$ une famille régulière de maillages affines par des simplexes.
Si pour la famille de maillages $(\mathcal{T}_h)_h$ tout triangle a au plus une arête sur $\partial\Omega$, les espaces X_h et M_h vérifient la condition Inf-Sup uniformément par rapport à h.

Théorème 1.2 (Inf-Sup $\mathbb{P}_1$iso$\mathbb{P}_2/\mathbb{P}_1$[EG04, p169])
Soit $(\mathcal{T}_h)_h$ une famille régulière de triangulations affines.
Les espaces X_h et M_h satisfont la condition Inf-Sup uniformément en h.

Théorème 1.3 (Inf-Sup $\mathbb{P}_1$bulle$/\mathbb{P}_1$[EG04, p174])
Soit $\mathcal{T}_h$ un maillage affine de Ω composé de simplexes.
Si la famille de maillages affines $(\mathcal{T}_h)_h$ est régulière, les espaces X_h et M_h sont compatibles uniformément par rapport à h, c'est à dire que la constante Inf-Sup est vérifiée indépendamment de h.

On va donc tout d'abord le vérifier avec le raffinement uniforme du carré unité puis voir ce qu'il se passe pour les autres maillages.

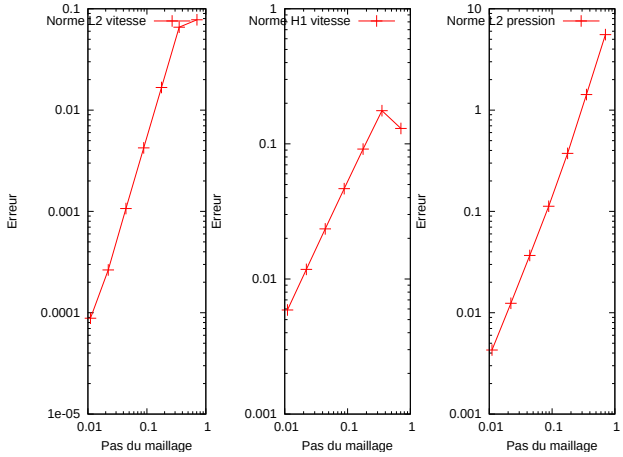
I.3 Résultats numériques

1.3.1 Convergence du schéma

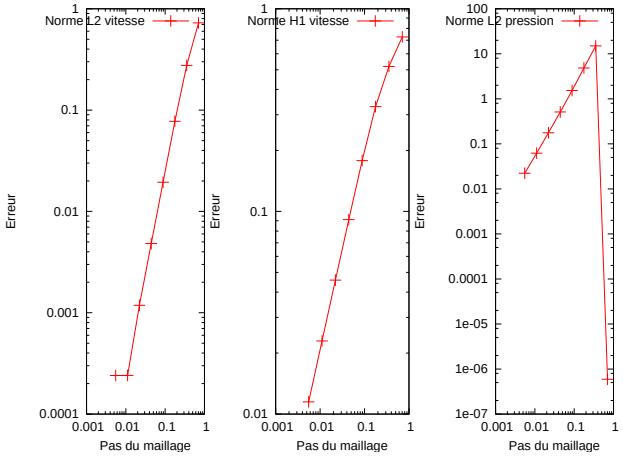
Regardons tout d'abord pour les éléments finis $\mathbb{P}_1$iso$\mathbb{P}_2/\mathbb{P}_1$ et $\mathbb{P}_1$_bulle$/\mathbb{P}_1$ si le schéma est convergent ou pas pour deux types de maillages :

– un domaine carré que l'on raffine uniformément (voir figure 1.1), on sait donc que la constante Inf-Sup est minorée dans les deux cas et que le schéma converge. On va donc vérifier ce résultat ;
– un domaine carré raffiné non uniformément (voir figure 1.2) où, dans ce cas, on ne sait pas comment va évoluer la constante Inf-Sup et donc si le schéma converge ou pas.

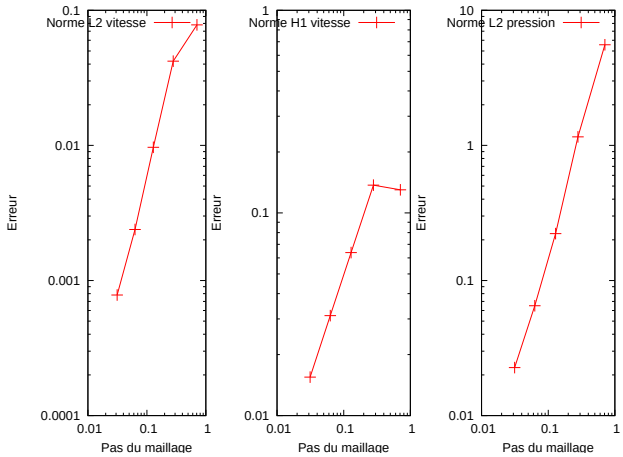
Commençons d'abord par regarder la courbe de l'erreur en fonction du pas pour ces 4 maillages en échelle log. Sur chaque graphe, on a (de gauche à droite) :
– l'erreur en norme L^2 de la vitesse,
– l'erreur en norme H^1 de la vitesse,
– l'erreur en norme L^2 de la pression.



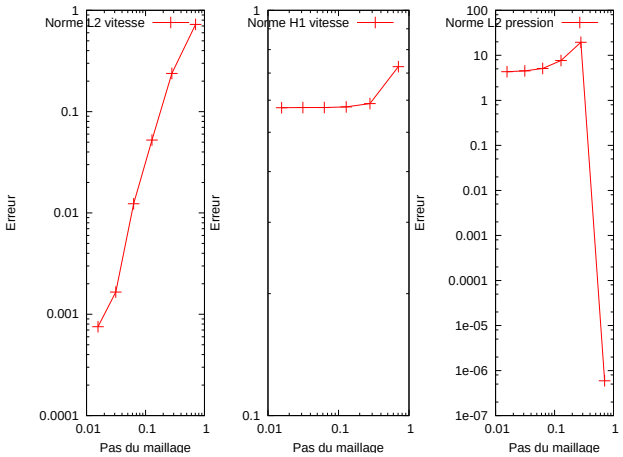
Raffinement uniforme en $\mathbb{P}_1$iso$\mathbb{P}_2/\mathbb{P}_1$



Raffinement uniforme en $\mathbb{P}_1$_bulle$/\mathbb{P}_1$



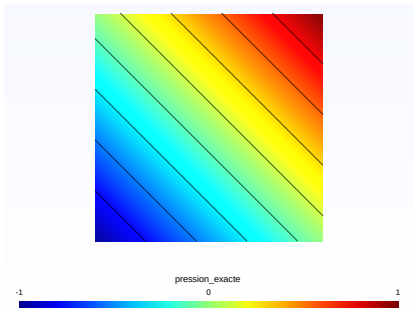
Raffinement non uniforme en $\mathbb{P}_1$iso$\mathbb{P}_2/\mathbb{P}_1$



Raffinement non uniforme en $\mathbb{P}_1$_bulle$/\mathbb{P}_1$

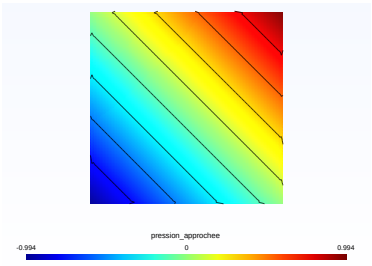
Dans le cas non uniforme, ces résultats semblent montrer que le schéma ne converge pas dans le cas de l'élément fini $\mathbb{P}_1$_bulle$/\mathbb{P}_1$ contrairement au cas $\mathbb{P}_1$iso$\mathbb{P}_2/\mathbb{P}_1$. Cependant on remarque un phénomène étrange qui est que le schéma semble converger pour la norme L^2 en vitesse et pas du tout pour la norme H^1 ni pour la norme L^2 en pression.

Illustrons d'abord la pression exacte :

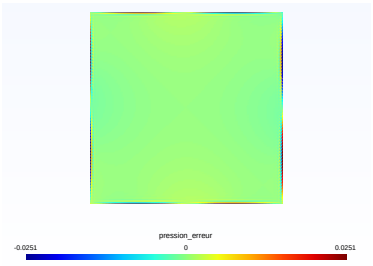


Solution exacte en pression

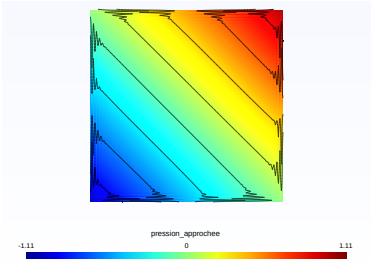
Ensuite, observons le graphe de la pression approchée et de l'erreur en pression entre les deux solutions.
– Lorsque l'on raffine uniformément, on choisit comme pas du maillage $h = 0.031311$,
– Lorsque l'on raffine non uniformément, on choisit comme pas du maillage $h = 0.0110485$.



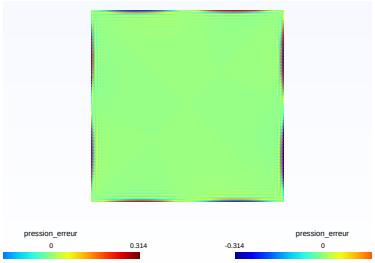
Pression approchée en $\mathbb{P}_1\text{iso}\mathbb{P}_2/\mathbb{P}_1$ uniforme



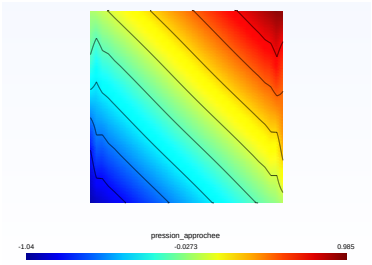
Erreur en $\mathbb{P}_1\text{iso}\mathbb{P}_2/\mathbb{P}_1$ uniforme



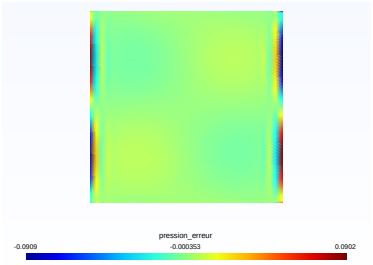
Pression approchée en $\mathbb{P}_1$_bulle$/\mathbb{P}_1$ uniforme



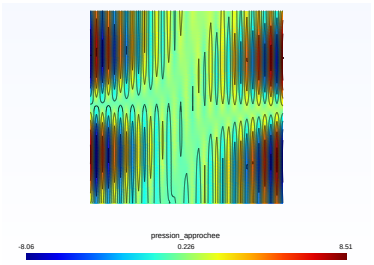
Erreur en $\mathbb{P}_1$_bulle$/\mathbb{P}_1$ uniforme



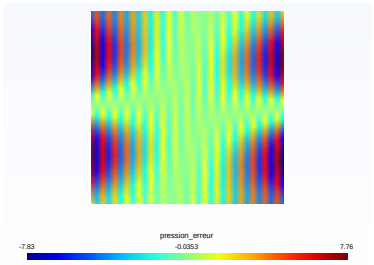
Pression approchée en $\mathbb{P}_1\text{iso}\mathbb{P}_2/\mathbb{P}_1$ non uniforme



Erreur en $\mathbb{P}_1\text{iso}\mathbb{P}_2/\mathbb{P}_1$ non uniforme

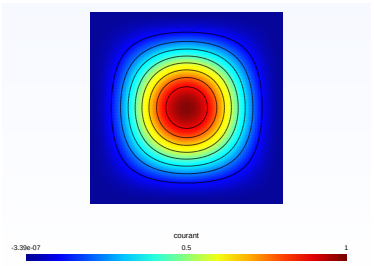


Pression approchée en $\mathbb{P}_1$_bulle$/\mathbb{P}_1$ non uniforme

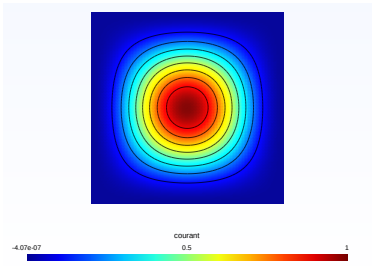


Erreur en $\mathbb{P}_1$_bulle$/\mathbb{P}_1$ non uniforme

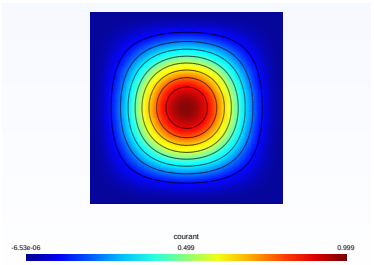
Enfin, observons pour ces mêmes maillages, la fonction de courant :



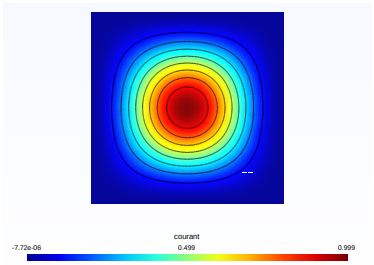
Fonction de courant $\mathbb{P}_1$iso$\mathbb{P}_2/\mathbb{P}_1$ uniforme



Fonction de courant $\mathbb{P}_1$iso$\mathbb{P}_2/\mathbb{P}_1$ uniforme



Fonction de courant $\mathbb{P}_1$_bulle$/\mathbb{P}_1$ non uniforme

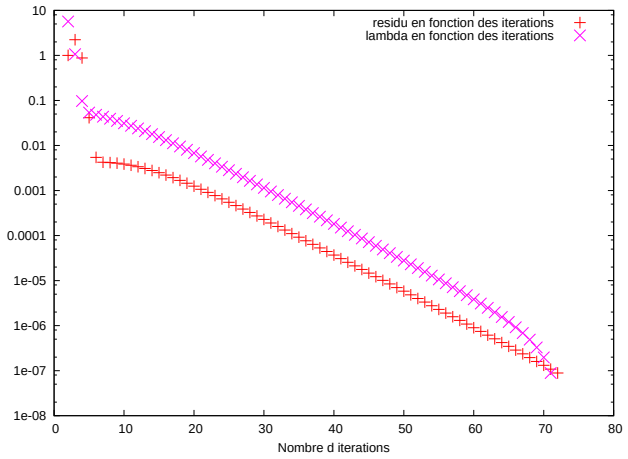


Fonction de courant $\mathbb{P}_1$_bulle$/\mathbb{P}_1$ non uniforme

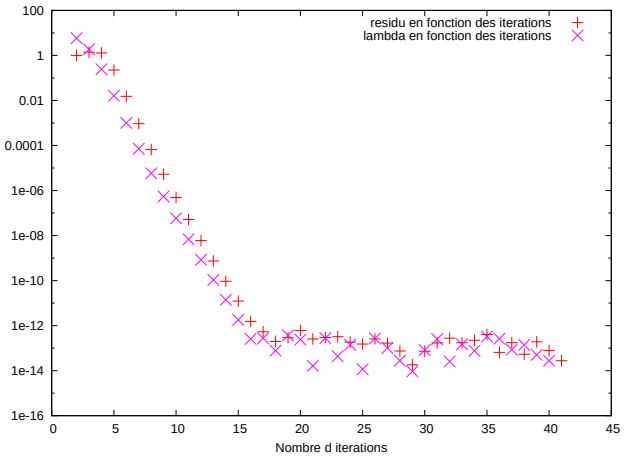
Là encore, pour les maillages uniformément raffinés on observe la convergence de la solution approchée vers la solution exacte alors que pour le maillage raffiné non uniformément seul le problème défini avec l'élément fini $\mathbb{P}_1$iso$\mathbb{P}_2/\mathbb{P}_1$ semble converger.
En effet, on remarque qu'en ce qui concerne la pression le schéma ne converge pas pour l'élément fini $\mathbb{P}_1$_bulle$/\mathbb{P}_1$ dans le cas du domaine carré raffiné non uniformément mais cependant la fonction de courant semble correcte.
On peut alors se demander ce qu'il en est de la constante Inf-Sup dans ces différents cas.

1.3.2 Comparaison des méthodes de calcul des valeurs propres

Regardons dans ce cas ce que donne les courbes traçant le résidu ainsi que le terme $\frac{|\lambda-\lambda_i|}{\lambda}$ en fonction du nombre d'itérations pour la méthode de la puissance inverse ainsi que pour la méthode d'itérations de sous-espaces de Rayleigh-Ritz.
On se place ici dans le cas de l'élément fini $\mathbb{P}_2/\mathbb{P}_1$ sur le maillage couronne avec un pas $h = 0.107046$ avec $N_p = 8953$ et $N_u = 67303$.
Cet exemple montre là encore que la méthod'itérations de sous-espaces de Rayleigh-Ritz est beaucoup plus efficace en nombre d'itérations effectuées que la méthode de la puissance inverse pour le calcul des valeurs propres.



Méthode de la puissance inverse



Méthode d'itérations de sous-espaces de Rayleigh-Ritz

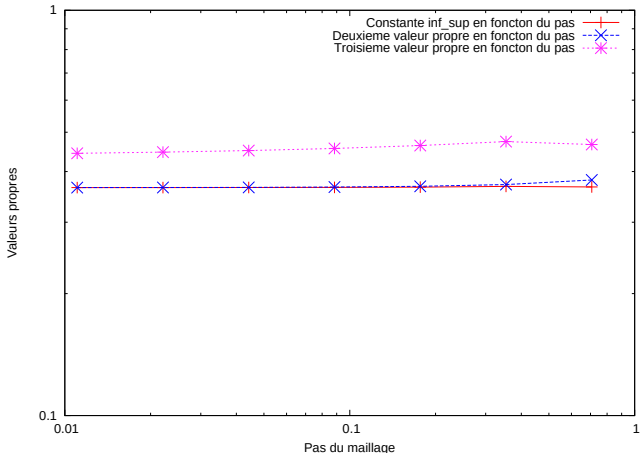
1.3.3 Les problèmes de valeur propre

Valeurs propres en fonction du pas du maillage :

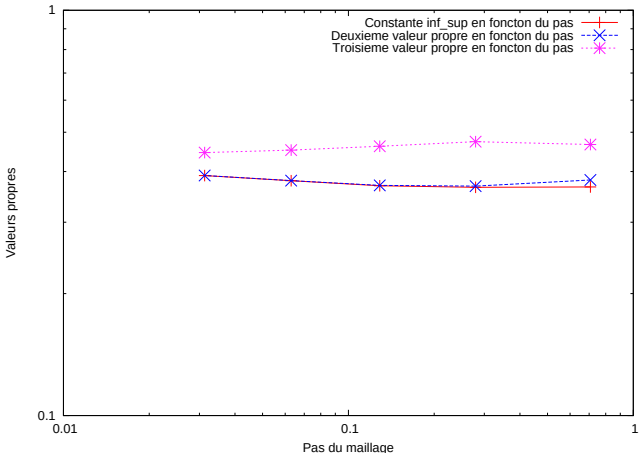
On va tout d'abord étudier pour chaque type de maillage et chaque type d'élément fini l'influence du raffinement (uniforme ou non).
Pour cela, on affiche ici le graphe des valeurs propres obtenues en fonction du pas du maillage ainsi que le tableau contenant :

- le pas du maillage h,
- les trois plus petites valeurs propres vp_1, vp_2, vp_2,
- le nombre d'itération effectuées it.

Dans le cas de l'élément fini $\mathbb{P}_2/\mathbb{P}_1$, on remarque que la constante Inf-Sup est bien uniformément minorée mais elle est également minorée dans les exemples où le raffinement est non uniforme.



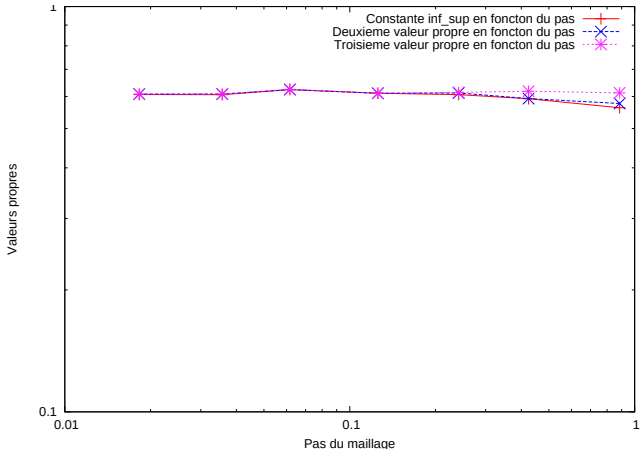
Carré uniforme $\mathbb{P}_2/\mathbb{P}_1$



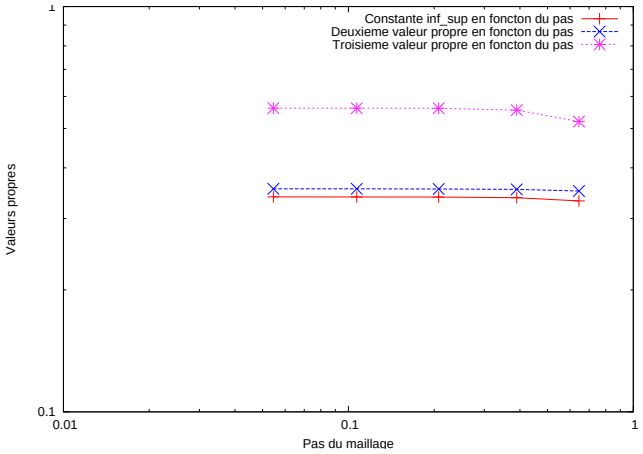
Carré non uniforme $\mathbb{P}_2/\mathbb{P}_1$

Dans le cas de l'élément fini $\mathbb{P}_1$iso$\mathbb{P}_2/\mathbb{P}_1$, on remarque là encore que la constante Inf-Sup est bien uniformément minorée mais elle est également minorée dans les exemples où le raffinement est non uniforme.

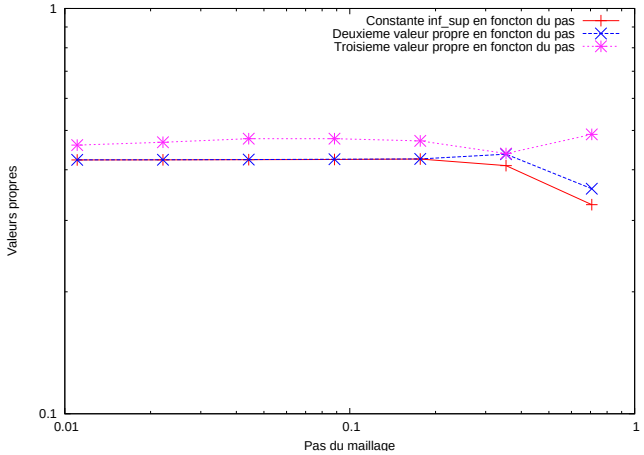
Dans le cas de l'élément fini $\mathbb{P}_1$_bulle$/\mathbb{P}_1$, la constante Inf-Sup est bien uniformément minorée mais lorsque l'on effectue un raffinement non uniforme sur le carré unité elle tend vers 0 tout comme les valeurs propres suivantes. Dans ce cas, la constante Inf-Sup n'est pas minorée uniformément ce dont nous pouvions nous douter vu que le schéma ne semblait pas converger.



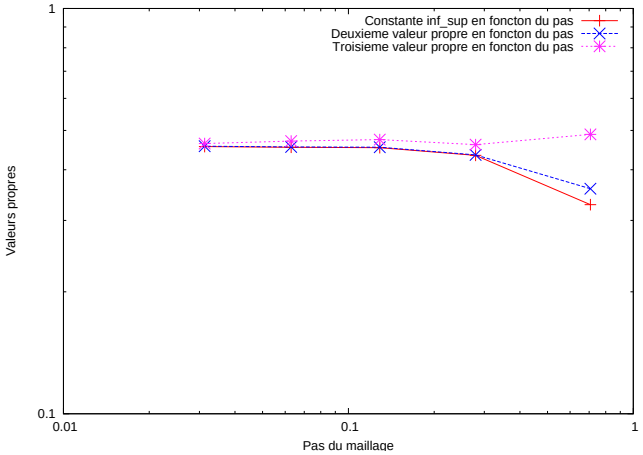
Cercle $\mathbb{P}_2/\mathbb{P}_1$



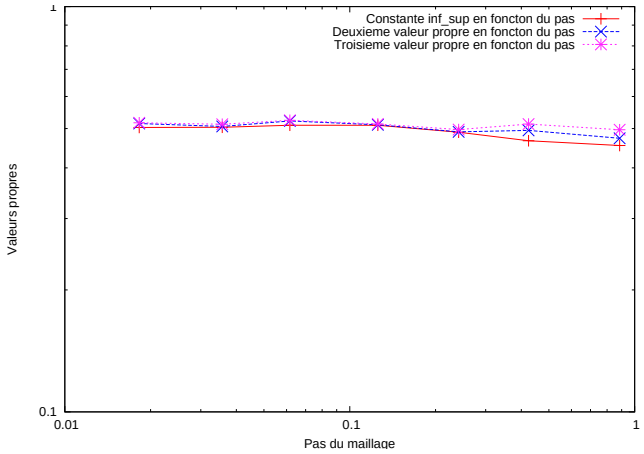
Couronne $\mathbb{P}_2/\mathbb{P}_1$



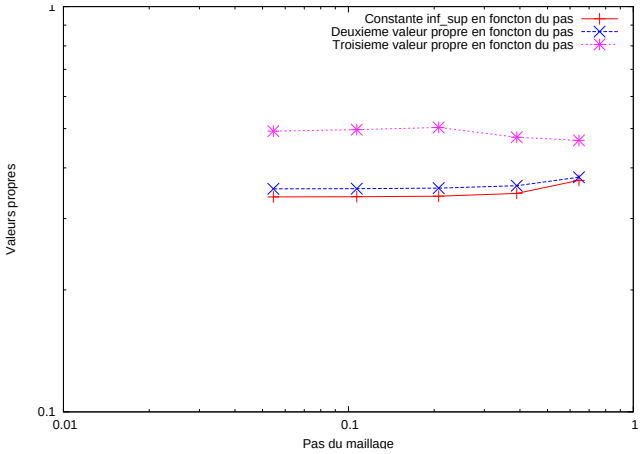
Carré uniforme $\mathbb{P}_1\text{iso}\mathbb{P}_2/\mathbb{P}_1$



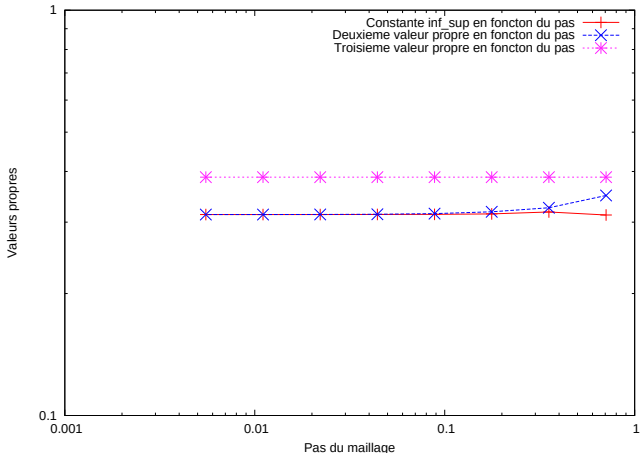
Carré non uniforme $\mathbb{P}_1\text{iso}\mathbb{P}_2/\mathbb{P}_1$



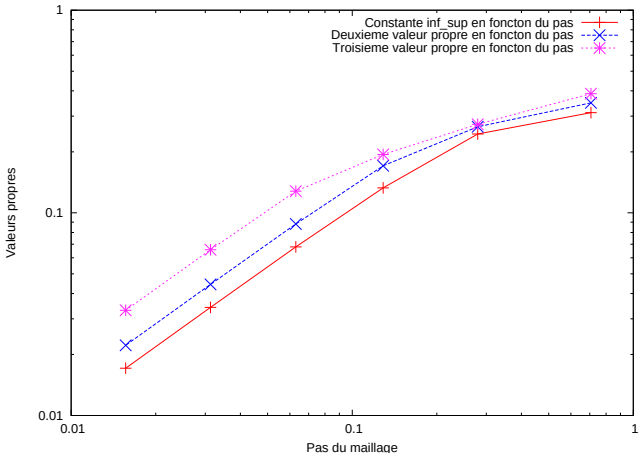
Cercle $\mathbb{P}_1\text{iso}\mathbb{P}_2/\mathbb{P}_1$



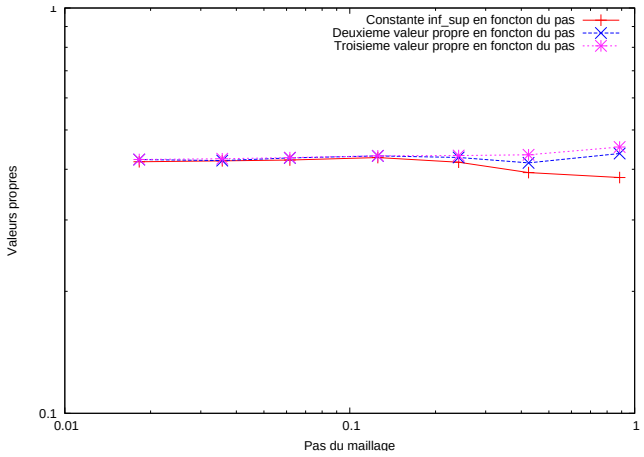
Couronne $\mathbb{P}_1\text{iso}\mathbb{P}_2/\mathbb{P}_1$



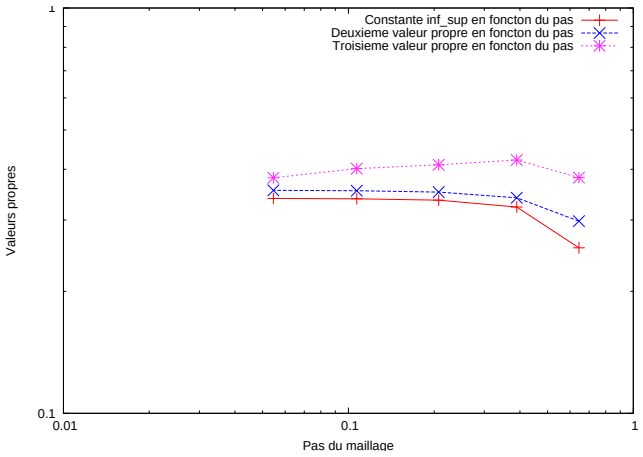
Carré uniforme $\mathbb{P}_1$_bulle/$\mathbb{P}_1$



Carré non uniforme $\mathbb{P}_1$_bulle/$\mathbb{P}_1$



Cercle $\mathbb{P}_1$_bulle/$\mathbb{P}_1$



Couronne $\mathbb{P}_1$_bulle/$\mathbb{P}_1$

	$\mathbb{P}_1$-bulle/$\mathbb{P}_1$				$\mathbb{P}_1$-iso$\mathbb{P}_2$/$\mathbb{P}_1$				$\mathbb{P}_2$/$\mathbb{P}_1$			
h	vp_1	vp_2	vp_3	it	vp_1	vp_2	vp_3	it	vp_1	vp_2	vp_3	it
0.707107	0.312379	0.349069	0.387298	29	0.328109	0.359182	0.48903	21	0.36657	0.381346	0.466441	29
0.353553	0.31776	0.325555	0.387298	60	0.409414	0.436938	0.438714	25	0.367675	0.371444	0.474287	82
0.176777	0.314316	0.318101	0.387298	84	0.425147	0.425378	0.471333	131	0.366191	0.367783	0.463791	112
0.0883883	0.313571	0.314846	0.387299	297	0.424038	0.424659	0.477189	134	0.365568	0.366252	0.456143	273
0.0441942	0.313289	0.31374	0.3873	952	0.423498	0.423902	0.477174	381	0.365295	0.365595	0.45072	130
0.0220971	0.313187	0.313349	0.387305	768	0.423125	0.423401	0.467562	494	0.365175	0.365308	0.446709	546
0.0110485	0.313151	0.313209	0.387328	1991	0.42286	0.423055	0.460083	523	0.365121	0.365181	0.443644	329
0.00552427	0.313138	0.313158	0.387333	1445								

Raffinement du carré uniformément

	$\mathbb{P}_1$-bulle/$\mathbb{P}_1$				$\mathbb{P}_1$-iso$\mathbb{P}_2$/$\mathbb{P}_1$				$\mathbb{P}_2$/$\mathbb{P}_1$			
h	vp_1	vp_2	vp_3	it	vp_1	vp_2	vp_3	it	vp_1	vp_2	vp_3	it
0.707107	0.312379	0.349069	0.387298	29	0.328109	0.359182	0.48903	24	0.36657	0.381346	0.466441	29
0.279508	0.244949	0.26601	0.273471	29	0.434009	0.435258	0.461341	196	0.365678	0.368259	0.474107	157
0.128847	0.132842	0.170668	0.194285	253	0.453384	0.454614	0.474823	226	0.368956	0.369813	0.461795	220
0.0629864	0.0679366	0.0881372	0.127944	38	0.454341	0.455319	0.470732	164	0.379638	0.380004	0.451832	219
0.031311	0.034166	0.0443525	0.0657502	20	0.4564	0.457345	0.464121	192	0.390964	0.391149	0.445422	499
0.0156326	0.017108	0.0222099	0.033052	17								

Raffinement du carré non uniformémément

	$\mathbb{P}_1$-bulle/$\mathbb{P}_1$				$\mathbb{P}_1$-iso$\mathbb{P}_2$/$\mathbb{P}_1$				$\mathbb{P}_2$/$\mathbb{P}_1$			
h	vp_1	vp_2	vp_3	it	vp_1	vp_2	vp_3	it	vp_1	vp_2	vp_3	it
0.885421	0.381634	0.43758	0.453906	125	0.454197	0.47321	0.496796	21	0.562892	0.57661	0.612472	40
0.424264	0.392526	0.414915	0.434352	163	0.466593	0.494882	0.513081	90	0.59223	0.593063	0.618291	314
0.24109	0.416378	0.427806	0.43257	67	0.489824	0.490896	0.49725	258	0.606264	0.612005	0.613932	175
0.125459	0.427625	0.431583	0.431892	170	0.509588	0.510849	0.513397	273	0.611076	0.611114	0.611239	3856
0.0616211	0.421703	0.426673	0.427457	225	0.509583	0.522229	0.524006	597	0.624068	0.624076	0.624788	548
0.0356803	0.419582	0.420273	0.424732	1003	0.503654	0.506197	0.512594	1399	0.606064	0.608382	0.608478	1122
0.0182395	0.417361	0.422682	0.422959	1340	0.503173	0.514527	0.516905	1118	0.60674	0.608239	0.608264	260

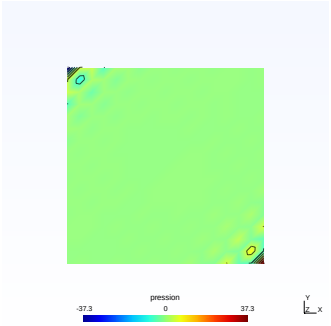
Raffinement du cercle

	$\mathbb{P}_1$-bulle/$\mathbb{P}_1$				$\mathbb{P}_1$-iso$\mathbb{P}_2$/$\mathbb{P}_1$				$\mathbb{P}_2$/$\mathbb{P}_1$			
h	vp_1	vp_2	vp_3	it	vp_1	vp_2	vp_3	it	vp_1	vp_2	vp_3	it
0.644508	0.256045	0.29795	0.381524	66	0.37274	0.379394	0.467562	84	0.331466	0.350604	0.520359	112
0.389973	0.322859	0.340209	0.421531	208	0.346014	0.361373	0.476095	88	0.337606	0.353912	0.555269	379
0.207403	0.335734	0.351306	0.410447	106	0.34063	0.356488	0.50334	127	0.338748	0.354766	0.560976	40
0.107046	0.338335	0.354176	0.401579	264	0.339476	0.355386	0.497056	80	0.339022	0.354973	0.561386	37
0.0545437	0.338927	0.354832	0.381117	1106	0.339198	0.355122	0.492946	95	0.339088	0.355023	0.561413	36

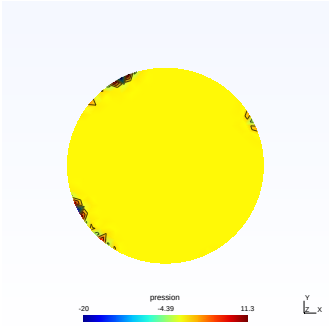
Raffinement de la couronne

Les modes propres :

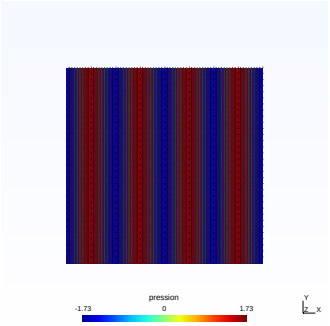
Regardons alors, dans certains cas, les modes propres correspondant pour un pas donné :
- le cercle en $\mathbb{P}_2/\mathbb{P}_1$ pour un pas $h = 0.24109$,
- le carré uniformément raffiné en $\mathbb{P}_1$iso$\mathbb{P}_2/\mathbb{P}_1$ avec un pas $h = 0.176777$,
- le carré non uniformément raffiné en $\mathbb{P}_1$_bulle/$\mathbb{P}_1$ avec un pas $h = 0.128847$ pour voir comment est le mode propre si la constante Inf-Sup tend vers 0.



Mode propre $\mathbb{P}_1$iso$\mathbb{P}_2/\mathbb{P}_1$



Mode propre $\mathbb{P}_2/\mathbb{P}_1$



Mode propre $\mathbb{P}_1$_bulle/$\mathbb{P}_1$

Les modes propres semblent montrer des oscillations près du bord.
Pour le mode propre associé à l'élément fini $\mathbb{P}_1$bulle/$\mathbb{P}_1$ dont la constante Inf-Sup tend vers 0, il semble y avoir une sinusoïde suivant la direction du raffinement le plus fort.

Conclusion :

- En ce qui concerne le raffinement uniforme des différents éléments finis, on voit bien que les résultats numériques confirment les résultats théoriques connus.
- Pour le raffinement non uniforme, on observe deux comportements différents :
  - La constante Inf-Sup des éléments finis $\mathbb{P}_2/\mathbb{P}_1$ et $\mathbb{P}_1$iso$\mathbb{P}_2/\mathbb{P}_1$ semble être tout de même minorée.
  - Celle de l'élément fini $\mathbb{P}_1$-bulle/$\mathbb{P}_1$ tend clairement vers 0.

II L'élément fini $\mathbb{Q}_1/\mathbb{P}_0$

On a déjà des résultats nous disant que cet élément fini est Inf-Sup instable ([EG04, p163-164]).
En effet, dans le cas $\mathbb{Q}_1/\mathbb{P}_0$ la condition Inf-Sup n'est pas satisfaite car il existe un champ de pression $\Psi_h \in M_h$ tel que :

$$\int_\Omega \Psi_h \operatorname{div} u_h = 0 \text{ pour tout } u_h \in X_h.$$

Pour remedier à ceci on peut essayer de diminuer la taille de M_h en imposant à la pression d'être orthogonale à cette fonction appelée "mode parasite". On a alors :

Théorème 2.1
Il existe deux constantes c_1 et c_2 telles que :

$$c_1 h \leq \beta_h \leq c_2 h.$$

On va donc vérifier numériquement que c'est bien ce que l'on retrouve.

II.1 Implémentation du problème

Le logiciel FreeFem++ ne réalisant que des maillages triangles, on va construire à la main les matrices dont on a besoin (matrice de masse en pression M, matrice de rigidité R et matrice de divergence B) pour résoudre le problème de Stokes avec les éléments fini $\mathbb{Q}_1$ en vitesse et $\mathbb{P}_0$ en pression sur un maillage carré.

2.1.1 Construction des matrices

Plaçons nous sur le carré unité maillé par des rectangles uniformes. Soit M le nombre de colonnes et N le nombre de lignes, le pas en x est donc $h_1 = 1/M$ et celui en y est $h_2 = 1/N$.

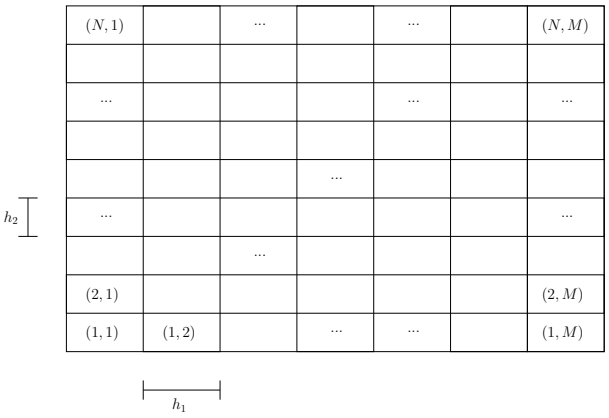


FIG. 2.1: Maillage

On a donc MN mailles et ainsi $N_p = MN$ inconnues en pression, celle-ci étant constante par maille.
En Q_1, les degrés de libertés sont les sommets de chaque maille. On a donc $(N+1)(M+1)$ inconnues pour chaque composante de la vitesse. Or, on sait que la vitesse est nulle au bord, il nous faut donc enlever les degrés de libertés du bord. Il y en a $2(N+M)$, on a donc en tout $\frac{N_u}{2} = (N+1)(M+1) - 2(N+M)$ inconnues pour chaque composante en vitesse.
Pour la vitesse, à chaque sommet i, la fonction de base associée φ_i est telle que pour $i \in \{1, \ldots, \frac{N_u}{2}\}$:

$$\begin{cases} \varphi_i = \begin{pmatrix} \alpha_i \\ 0 \end{pmatrix} \text{ pour la première composante de la vitesse} \\ \varphi_{i+\frac{N_u}{2}} = \begin{pmatrix} 0 \\ \alpha_i \end{pmatrix} \text{ pour la deuxième composante de la vitesse} \end{cases}$$

Soit $(\psi_i)_{i=1,\dots,N_p}$ les fonctions de bases de la pression.
Notons $u = (u_1, u_2)$ le vecteur vitesse et p le vecteur pression, alors :

$$u_1 = \sum_{i=1}^{N_u/2} u_{1,i}\alpha_i,\ u_2 = \sum_{i=1}^{N_u/2} u_{2,i}\alpha_i \text{ et } p = \sum_{i=1}^{N_p} p_i\psi_i.$$

La matrice de rigidité : La matrice de rigidité du problème de Stokes s'écrit :

$$R = \left(\int_\Omega \nabla\varphi_i : \nabla\varphi_j\right)_{1\le i,j\le N_u/2} = \begin{pmatrix} \left(\int_\Omega \nabla\alpha_i.\nabla\alpha_j\right)_{1\le i,j\le N_u/2} & 0 \\ 0 & \left(\int_\Omega \nabla\alpha_i.\nabla\alpha_j\right)_{1\le i,j\le N_u/2} \end{pmatrix}.$$

Pour construire cette matrice, il nous suffit donc de construire la matrice de rigidité associée au Laplacien :

$$R_{Lap} = \left(\int_\Omega \nabla\alpha_i.\nabla\alpha_j\right)_{1\le i,j\le N_u/2}.$$

De plus, $\int_\Omega \nabla\alpha_i.\nabla\alpha_j = \sum_{k=1}^{MN} \int_{C_k} \nabla\alpha_i.\nabla\alpha_j$ où C_k est la maille numéro k et sachant que $\int_{C_k} \nabla\alpha_i.\nabla\alpha_j = 0$ si i ou j ne sont pas associés à des sommets de la maille C_k.
La construction de R se fait alors en plusieurs étapes :

1. On calcule la matrice $R_{Lap(C_k)} = (\int_{C_k} \nabla\alpha_i.\nabla\alpha_j)_{1\le i,j\le 4}$ qui représente la matrice de rigidité du Laplacien sur une maille k dont les sommets sont toujours numérotés comme ceci :

FIG. 2.2: Numérotation d'une maille

Pour ceci, pour $i = 1, \dots, 4$, calculons α_i et $\nabla\alpha_i$ sur une maille C_k quelconque. Nous savons déjà que les fonctions de bases locales associées à l'élément de référence R, sont :

$$\begin{aligned}
\theta_1(x_1, x_2) &= (1 - x_1)(1 - x_2) \\
\theta_2(x_1, x_2) &= x_1(1 - x_2) \\
\theta_3(x_1, x_2) &= x_1x_2 \\
\theta_4(x_1, x_2) &= (1 - x_1)x_2.
\end{aligned}$$

Soit F_k la transformation affine qui permet de passer de R à C_k, c'est à dire : $\forall(x_1, x_2) \in R$,

$$y = F_k(x_1, x_2) = A^1(C_k) + x_1\overrightarrow{A^1(C_k)A^2(C_k)} + x_2\overrightarrow{A^1(C_k)A^4(C_k)}$$

où A^i sont les 4 sommets du rectangle de référence.
On a donc : $\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_1^1(C_k) + h_1x_1 \\ x_2^1(C_k) + h_2x_2 \end{pmatrix}$, c'est à dire :

$$F_k^{-1}(y) = (x_1, x_2) = (\frac{y_1 - x_1^1(C_k)}{h_1}, \frac{y_2 - x_2^1(C_k)}{h_2}) \text{ et ainsi :}$$

$$\begin{aligned}
&\alpha_{1_{|C_k}}(y) = (1 - x_1)(1 - x_2) && \text{et} && \nabla\alpha_1 = (-\frac{1 - x_2}{h_1}, -\frac{1 - x_1}{h_2}) \\
&\alpha_{2_{|C_k}}(y) = x_1(1 - x_2) && \text{et} && \nabla\alpha_2 = (\frac{1 - x_2}{h_1}, -\frac{x_1}{h_2}) \\
&\alpha_{3_{|C_k}}(y) = x_1x_2 && \text{et} && \nabla\alpha_3 = (\frac{x_2}{h_1}, \frac{x_1}{h_2}) \\
&\alpha_{4_{|C_k}}(y) = (1 - x_1)x_2 && \text{et} && \nabla\alpha_4 = (-\frac{x_2}{h_1}, \frac{1 - x_1}{h_2})
\end{aligned}$$

car $\nabla\alpha_i = (\frac{\partial\alpha_i}{\partial y_1}, \frac{\partial\alpha_i}{\partial y_2}) = (\frac{\partial\alpha_i}{\partial x_1}\frac{\partial x_1}{\partial y_1}, \frac{\partial\alpha_i}{\partial y_2}\frac{\partial x_2}{\partial y_2}) = (\frac{\partial\alpha_i}{\partial x_1}\frac{1}{h_1}, \frac{\partial\alpha_i}{\partial y_2}\frac{1}{h_2})$.

On peut maintenant calculer les différents termes de $R_{Lap(C_k)}$, on obtient :

$$R_{Lap(C_k)} = \frac{h_1h_2}{3}\begin{pmatrix} \frac{1}{h_1^2}+\frac{1}{h_2^2} & -\frac{1}{h_1^2}+\frac{1}{2h_2^2} & -\frac{1}{2}(\frac{1}{h_1^2}+\frac{1}{h_2^2}) & \frac{1}{2h_1^2}-\frac{1}{h_2^2} \\ \frac{-1}{h_1^2}+\frac{1}{2h_2^2} & \frac{1}{h_1^2}+\frac{1}{h_2^2} & \frac{1}{2h_1^2}-\frac{1}{h_2^2} & -\frac{1}{2}(\frac{1}{h_1^2}+\frac{1}{h_2^2}) \\ -\frac{1}{2}(\frac{1}{h_1^2}+\frac{1}{h_2^2}) & \frac{1}{2h_1^2}-\frac{1}{h_2^2} & \frac{1}{h_1^2}+\frac{1}{h_2^2} & -\frac{1}{h_1^2}+\frac{1}{2h_2^2} \\ \frac{1}{2h_1^2}-\frac{1}{h_2^2} & -\frac{1}{2}(\frac{1}{h_1^2}+\frac{1}{h_2^2}) & -\frac{1}{h_1^2}+\frac{1}{2h_2^2} & \frac{1}{h_1^2}+\frac{1}{h_2^2} \end{pmatrix}.$$

2. On numérote correctement notre vecteur inconnu :
   - On décide tout d'abord de le ranger dans un certain ordre : $\begin{pmatrix} u_1 \\ u_2 \\ p \end{pmatrix}$.
   - Puis, on construit une application a_1 qui pour chaque inconnue de la première composante de la vitesse renvoie sa place dans le vecteur. Les inconnues de la première composante en vitesse sont notées $u_{1,(\alpha+1/2,\beta+1/2)}$ avec $1 \leq \alpha \leq M-1$ et $1 \leq \beta \leq M-1$.
     On a alors $a_1 : (\alpha, \beta) \mapsto \beta + (\alpha-1)(M-1)$.
   - Les inconnues de la deuxième composante sont alors rangées dans le même ordre et placées juste après celles de la première composante, c'est à dire que l'inconnue $u_{2,(\alpha+1/2,\beta+1/2)}$ est à la place $\frac{N_u}{2} + \beta + (\alpha-1)(M-1)$.
   - On place enfin les inconnues pour la pression, celle-ci étant constante par maille il faut construire une application a_2 qui à chaque maille $K_{i,j}$ associe sa place dans le vecteur inconnu où $1 \leq i \leq N$ et $1 \leq j \leq M$.
     On obtient alors $a_2 : (i,j) \mapsto N_u + (j + (i-1)M)$.
3. On construit maintenant l'application a_3, qui à une cellule $K_{i,j}$ associe un vecteur contenant le numéro des 4 sommets correspondants rangés dans le bon ordre.

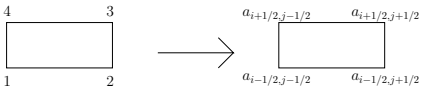


FIG. 2.3: Correspondance entre une maille et la numérotation de ses sommets dans le maillage

On a alors, $a_3 : K_{i,j} \mapsto place_{i,j} = \begin{pmatrix} (j-1)+(i-2)(M-1) \\ j+(i-2)(M-1) \\ j+(i-1)(M-1) \\ (j-1)+(i-1)(M-1) \end{pmatrix}$

4. Pour finir, on construit la matrice R_{Lap}.
   Pour $i \in \{1,\dots,N\}$ et $j \in \{1,\dots,M\}$, on parcours les cellules $K_{i,j}$, pour chaque cellule, pour $k,l \in \{1,2,3,4\}$:
   - si $place_{i,j}(k) \neq -1$ et $place_{i,j}(l) \neq -1$, alors on prend la coordonnée (k,l) de la matrice $R_{Lap(C_k)}$, et on la met à la place $(place_{i,j}(k), place_{i,j}(l))$
   - sinon, ça veut dire que c'est un sommet du bord et donc on ne fait rien.

La matrice de masse : La matrice de rigidité étant construite intéressons nous à la matrice de masse $M = (\int_\Omega \psi_\lambda\psi_\mu)_{1\leq\lambda,\mu\leq N_p}$ où $\psi_\lambda = \psi_{K_{i,j}} = \chi_{K_{i,j}}$.
On a alors $\int_\Omega \psi_\lambda\psi_\mu = \sum_{k=1}^{N_p}\int_{C_k}\psi_\lambda\psi_\mu$ et $\int_{C_k}\psi_\lambda\psi_\mu = \begin{cases} 0 \text{ si } \lambda \neq \mu \\ |C_k| = h_1h_2 \text{ si } \lambda = \mu \end{cases}$.
Ainsi, on a $M = h_1h_2I_d$.

La matrice de divergence : Il ne nous reste maintenant plus qu'à construire la matrice de divergence B. On sait que :

$$(BU,P) = -\int_\Omega p \operatorname{div} u = -\sum_{k=1}^{N_p}\int_{C_k} p \operatorname{div} u = -\sum_{k=1}^{N_p} p_k \int_{C_k} \operatorname{div} u,$$

par définition des fonctions de base en pression.
De plus, si $u = (u^1, u^2)$:

$$\begin{aligned}\int_{C_k} \operatorname{div} u = \int_{K_{i,j}} \operatorname{div} u = \int_{\partial K_{i,j}} u.\vec{n} \\ = \frac{h_2}{2}(u^1_{i+1/2,j+1/2} + u^1_{i-1/2,j+1/2}) + \frac{h_1}{2}(u^2_{i+1/2,j-1/2} + u^2_{i+1/2,j+1/2}) \\ - \frac{h_2}{2}(u^1_{i+1/2,j-1/2} + u^1_{i-1/2,j-1/2}) - \frac{h_1}{2}(u^2_{i-1/2,j-1/2} + u^2_{i-1/2,j+1/2})\end{aligned}$$

On construit maintenant les vecteurs $C_{h_1} = \begin{pmatrix} -\frac{h_1}{2} \\ -\frac{h_1}{2} \\ \frac{h_1}{2} \\ \frac{h_1}{2} \end{pmatrix}$ et $C_{h_2} = \begin{pmatrix} -\frac{h_2}{2} \\ \frac{h_2}{2} \\ \frac{h_2}{2} \\ -\frac{h_2}{2} \end{pmatrix}$.

Pour construire $B \in \mathcal{M}_{N_p \times N_u}$, on parcours alors les cellules et dans la cellule $K_{i,j} = C_k$, pour $l = 1, \ldots, 4$:
- on prend la composante $-C_{h_2}(l)$ que l'on met en position $(j + (i-1)M, place_{i,j}(l))$ dans la matrice B,
- on prend la composante $-C_{h_1}(l)$ que l'on met en position $(j + (i-1)M, \frac{N_u}{2} + place_{i,j}(l)$ dans la matrice B.

Pour tester si la matrice de divergence construite est correcte, on prend une matrice aléatoire $A \in \mathcal{M}_{2\times 2}$ et on pose $u(x, y) = A \begin{pmatrix} x \\ y \end{pmatrix}$, alors on doit avoir $\operatorname{div} u = \operatorname{Tr} A$.
Or, on a :

$$\begin{cases} (BU, P) & = -\sum_{k=1}^{N_p} p_k \int_{C_k} \operatorname{div} u = -\sum_{k=1}^{N_p} |C_k| p_k \operatorname{Tr} A \text{ sauf sur le bord,} \\ \text{et } (BU, P) & = \sum_{k=1}^{N_p} (BU)_k p_k \end{cases}$$

Ainsi, on doit retrouver (mis à part sur le bord) que :

$$(BU)_k = |C_k| \operatorname{Tr} A = h_1 h_2 \operatorname{Tr} A,$$

ce qui est bien le cas.

2.1.2 Le mode en damier

Maintenant que les matrices dont on a besoin sont construites, on peut trouver la constante inf-sup grâce à la méthode de la puissance ou la méthode QR.
On se retrouve cette fois confronté à deux problèmes :
- on a toujours le vecteur des pressions constantes qu'il faut enlever
- on a un autre vecteur qui se trouve dans le noyau de B^T. On l'appelle le mode en damier, il est très oscillant et s'écrit :

$$\Psi_h = \sum_{i=1}^{N}\sum_{j=1}^{M} \chi_{K_{i,j}} (-1)^{i+j} = \sum_{i=1}^{N}\sum_{j=1}^{M} \psi_{K_{i,j}} (-1)^{i+j}$$

On doit donc enlever ces deux vecteurs $p_1 = (1, \ldots, 1)$ et $p_2 = \Psi_h$ de M_h.
Notons η l'écriture vectorielle de Ψ_h, on a alors $\eta = (\eta_i)_{1 \le i \le N_p}$ avec $\eta_i = \pm 1$ pour tout $i \in \{1, \ldots, N_p\}$ et on peut remarquer que l'on a forcément $\eta_{N_p - 1} \neq \eta_{N_p}$.

La méthode de la puissance inverse :

Rappelons que l'on cherche à résoudre le problème de Stokes suivant :

$$\begin{cases} RU + B^T P = 0 \\ BU = -MQ \end{cases}$$

où :

1. on choisit aléatoirement $\tilde{Q}$ tel que $(M\tilde{Q}, 1) = 0$,

2. on prend $Q = \tilde{Q} - \frac{1}{||\Psi_h||^2}(M\tilde{Q}, \Psi_h)M^{-1}\Psi_h$ de sorte que $(MQ, \Psi_h) = 0$. Or $M = h_1 h_2 Id$, donc :

$$Q = \tilde{Q} - \frac{1}{||\Psi_h||^2}(\tilde{Q}, \Psi_h)\Psi_h.$$

On remarque alors que l'on a bien $(MQ, 1) = 0$ car on a choisi de prendre le nombre de lignes N et le nombre de colonnes M comme des multiples de 2. On a donc un nombre pair de mailles et ainsi $(\Psi, 1) = 0$.

3. On pose ensuite :
   - $\tilde{B}$ la matrice B privée de ces deux dernières lignes,
   - $\widetilde{MQ}$ est le vecteur MQ privé de ces deux dernières coordonnées.
4. On résout alors le problème de Stokes :

$$\begin{cases} RU + \tilde{B}^T\tilde{P} = 0 \\ \tilde{B}U = -\widetilde{MQ} \end{cases}.$$

Puis, on pose :

- $\tilde{\tilde{P}} = \begin{pmatrix} \tilde{P} \\ 0 \\ 0 \end{pmatrix}$,
- puis $\bar{P} = \tilde{\tilde{P}} - \frac{1}{|\Omega|}(M\tilde{\tilde{P}}, 1)\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$,
- et enfin : $P = \bar{P} - \frac{1}{||\Psi_h||^2}(M\bar{P}, \Psi_h)M^{-1}\Psi_h = \bar{P} - \frac{1}{||\Psi_h||^2}(\bar{P}, \Psi_h)\Psi_h$.

On a donc bien $(MP, 1) = 0$ et $(MP, \Psi_h) = 0$.

Vérifions alors que trouver $\tilde{P}$ nous permet de trouver P.
Tout d'abord,

$$\begin{aligned} RU + B^T P &= RU + B^T\left(\bar{P} - \frac{1}{||\Psi_h||^2}(\bar{P}, \Psi_h)\Psi_h\right) = RU + B^T\bar{P} \text{ car } B^T\Psi_h = 0 \\ &= RU + B^T\left(\tilde{\tilde{P}} - \frac{1}{|\Omega|}(M\tilde{\tilde{P}}, 1)\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}\right) = RU + B^T\tilde{\tilde{P}} \text{ car } B^T\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \\ &= RU + \tilde{B}^T\tilde{P} = 0. \end{aligned}$$

Puis, il suffit de vérifier que $BU = -MQ$ pour les deux dernières coordonnées.
On sait que :

$$\begin{cases} \sum_{i=1}^{N_p}\sum_{j=1}^{N_u} b_{i,j}u_j = (BU, 1) = (U, B^T 1) = 0, \\ \sum_{i=1}^{N_p}\sum_{j=1}^{N_u} \eta_i b_{i,j}u_j = (BU, \Psi_h) = (U, B^T\Psi_h) = 0, \\ \sum_{i=1}^{N_p}\sum_{j=1}^{N_p} m_{i,j}q_j = (MQ, 1) = 0, \\ \sum_{i=1}^{N_p}\sum_{j=1}^{N_p} \eta_i m_{i,j}q_j = (MQ, \Psi_h) = 0, \end{cases}$$

c'est à dire que :

$$\begin{cases} \sum_{i=1}^{N_p-2}\sum_{j=1}^{N_u} b_{i,j}u_j + \sum_{j=1}^{N_u} b_{N_p-1,j}u_j + \sum_{j=1}^{N_u} b_{N_p,j}u_j = 0 & (1), \\ \sum_{i=1}^{N_p-2}\sum_{j=1}^{N_u} \eta_i b_{i,j}u_j + \sum_{j=1}^{N_u} \eta_{N_p-1} b_{N_p-1,j}u_j + \sum_{j=1}^{N_u} \eta_{N_p} b_{N_p,j}u_j = 0 & (2), \\ \sum_{i=1}^{N_p-2}\sum_{j=1}^{N_p} m_{i,j}q_j + \sum_{j=1}^{N_p} m_{N_p-1,j}q_j + \sum_{j=1}^{N_p} m_{N_p,j}q_j = 0 & (3), \\ \sum_{i=1}^{N_p-2}\sum_{j=1}^{N_p} \eta_i m_{i,j}q_j + \sum_{j=1}^{N_p} \eta N_p - 1 m_{N_p-1,j}q_j + \sum_{j=1}^{N_p} \eta N_p m_{N_p,j}q_j = 0 & (4). \end{cases}$$

On peut alors effectuer les opérations suivantes sur les lignes :
- (1) + η_{N_p-1}(2),
- (1) + η_{N_p}(2),
- (3) + η_{N_p-1}(4),
- (3) + η_{N_p}(4).

On obtient alors le système d'équations (si N_p est pair) :

$$\begin{cases} (BU)_{N_p-1} = \sum_{j=1}^{N_u} b_{N_p-1,j}u_j = -\frac{1}{2}\sum_{i=1}^{N_p-2}\sum_{j=1}^{N_u}(1+\eta_{N_p-1}\eta_i)b_{i,j}u_j \\ (BU)_{N_p} = \sum_{j=1}^{N_u} b_{N_p,j}u_j = -\frac{1}{2}\sum_{i=1}^{N_p-2}\sum_{j=1}^{N_u}(1+\eta_{N_p}\eta_i)b_{i,j}u_j \\ (MQ)_{N_p-1} = \sum_{j=1}^{N_p} m_{N_p-1,j}q_j = -\frac{1}{2}\sum_{i=1}^{N_p-2}\sum_{j=1}^{N_p}(1+\eta_{N_p-1}\eta_i)m_{i,j}q_j \\ (MQ)_{N_p} = \sum_{j=1}^{N_p} m_{N_p,j}q_j = -\frac{1}{2}\sum_{i=1}^{N_p-2}\sum_{j=1}^{N_p}(1+\eta_{N_p-1}\eta_i)m_{i,j}q_j. \end{cases}$$

Or, $\tilde{B}U = -\widetilde{MQ}$, donc on sait que $\forall i \in 1, \ldots, N_p - 2$,

$$\sum_{j=1}^{N_u} b_{i,j}u_j = (\tilde{B}U)_i = -(\widetilde{MQ})_i = -\sum_{j=1}^{N_p} m_{i,j}q_j.$$

Ainsi, en combinant :
- la première et la troisième équation on obtient que $(BU)_{N_p-1} = -(MQ)_{N_p-1}$,
- la deuxième et la dernière équation on obtient que $(BU)_{N_p} = -(MQ)_{N_p}$.

Ce qui est bien le résultat attendu.

La méthode d'itérations de sous-espaces de Rayleigh-Ritz :

Rappelons que pour cette méthode, s'il on veut connaître les k plus petites valeurs propres, à chaque itération il faut résoudre le problème : Trouver $\begin{pmatrix} U \\ P \end{pmatrix} = \begin{pmatrix} U^1 & \ldots & U^k \\ P^1 & \ldots & P^M \end{pmatrix}$ tels que $\begin{cases} RU + B^T P = 0 \\ BU = -MQ \end{cases}$.

Pour cela :

1. On choisit aléatoirement $\bar{Q}_0 = [\bar{Q}_0^1, \ldots, \bar{Q}_0^k]$ dont chaque colonne $\bar{Q}_0^i$ est prise à moyenne nulle.
2. On les modifie pour qu'il soient M-orthogonaux, c'est à dire que $\bar{Q}_0^T M \bar{Q}_0 = Id$.
3. On prends $Q_0 = \tilde{Q}_0 - \frac{1}{||\Psi_h||^2}[(M\tilde{Q}_0^1, \Psi_h)M^{-1}\Psi_h, \ldots, (M\tilde{Q}_0^k, \Psi_h)M^{-1}\Psi_h] = \tilde{Q}_0 - \frac{1}{||\Psi_h||^2}[(\tilde{Q}_0^1, \Psi_h)\Psi_h, \ldots, (\tilde{Q}_0^k, \Psi_h)\Psi_h]$, de sorte que pour chaque colonne Q_0^i de Q_0 on ait $(MQ_0^i, \Psi_h) = 0$.
4. On pose :
   - $\tilde{B}$ la matrice B privée de ses deux dernières lignes,
   - $\widetilde{MQ_0}$ la matrice MQ_0 privée de ses deux dernières lignes ;

puis on résout le problème : trouver $\begin{pmatrix} U_1 \\ \tilde{P}_1 \end{pmatrix} = \begin{pmatrix} U_1^1 & \dots & U_1^k \\ \tilde{P}_1^1 & \dots & \tilde{P}_1^k \end{pmatrix}$ tels que $\begin{cases} RU_1 + \tilde{B}^T \tilde{P}_1 = 0 \\ \tilde{B}U_1 = -\widetilde{MQ_1} \end{cases}$.

5. On pose ensuite :
   - $\tilde{\tilde{P}}_1 = \begin{pmatrix} \tilde{P}_1 \\ 0 \dots 0 \\ \underbrace{0 \dots 0}_{kfois} \end{pmatrix}$,
   - puis $\bar{P}_1 = \tilde{\tilde{P}}_1 - \frac{1}{|\Omega|} \left[(M\tilde{\tilde{P}}_1^1, 1) \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, \dots, (M\tilde{\tilde{P}}_0^k, 1) \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \right]$ où $\tilde{\tilde{P}}_1 = [\tilde{\tilde{P}}_1^1, \dots, \tilde{\tilde{P}}_1^k]$,
   - et enfin $P_1 = \bar{P}_1 - \frac{1}{||\Psi_h||^2}[(\bar{P}_1^1, \Psi_h)\Psi_h, \dots, (\bar{P}_1^k, \Psi_h)\Psi_h]$ où $\bar{P}_1 = [\bar{P}_1^1, \dots, \bar{P}_1^k]$.
6. Par une factorisation QR, on trouve $\tilde{Q}_1$ et R_1 tels que $P_1 = \tilde{Q}_1 R_1$ de sorte que $\tilde{Q}_1^T M \tilde{Q}_1 = Id$.
7. On pose $B_1 = \tilde{Q}_1^T M (M^{-1} B R^{-1} B^T) \tilde{Q}_1$.
8. On cherche les vecteurs de propres Y_1 de B_1, c'est à dire que $[Y_1, D_1] = spec(B_1)$.
9. On pose $Q_1 = \tilde{Q}_1 Y_1$.
   On remarque alors qu'on a construit P_1 de manière à ce que ses vecteurs colonnes P_1^i soient tels que $(MP_1^i, 1) = 0$ et $(MP_1^i, \Psi_h) = 0$ et donc les vecteurs colonnes de Q_1 seront également tels que pour tout $i \in \{1, \dots, k\}$, $(MQ_1^i, 1) = 0$ et $(MQ_1^i, \Psi_h) = 0$.
   En effet, on a déjà montré (Chap. 1 p 12-13) que si pour tout $i \in \{1, \dots, k\}$, $(MP_1^i, 1) = 0$ alors $(MQ_1^i, 1) = 0$ et on montre exactement de la même manière que si $(MP_1^i, \Psi_h) = 0$ alors $(MQ_1^i, \Psi_h) = 0$.

Il reste maintenant à vérifier que le couple $\begin{pmatrix} U_1 \\ P_1 \end{pmatrix}$ ainsi construit est bien solution de notre problème.

– Pour la première équation, pour les mêmes raisons que précédemment on a :

$$\begin{aligned} RU_1 + B^T P_1 &= RU_1 + B^T \bar{P}_1 - \frac{1}{||\Psi_h||^2}[(\bar{P}_1^1, \Psi_h) B^T \Psi_h, \dots, (\bar{P}_1^k, \Psi_h) B^T \Psi_h] = RU_1 + B^T \bar{P}_1 \\ &= RU_1 + B^T \tilde{\tilde{P}}_1 - \frac{1}{|\Omega|} \left[(M\tilde{\tilde{P}}_1^1, 1) B^T \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, \dots, (M\tilde{\tilde{P}}_0^k, 1) B^T \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \right] = RU_1 + B^T \tilde{\tilde{P}}_1 \\ &= RU_1 + \tilde{B}^T \tilde{P}_1 = 0. \end{aligned}$$

– Sachant que $\tilde{B}U_1 = -\widetilde{MQ_0}$, il reste à montrer que $BU_1 = -MQ_0$ seulement pour les deux dernières lignes, c'est à dire :

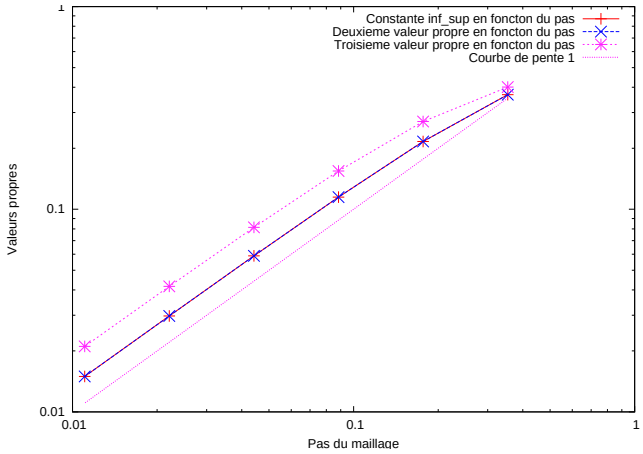
$$\begin{aligned} &\begin{cases} (BU_1)_{N_p - 1} = -(MQ_0)_{N_p - 1} \\ (BU_1)_{N_p} = -(MQ_0)_{N_p} \end{cases} \\ \iff &\begin{cases} \forall i \in \{1, \dots, k\}, (BU_1^i)_{N_p - 1} = (BU_1)^i_{N_p - 1} = -(MQ_0)^i_{N_p - 1} = -(MQ_0^i)_{N_p - 1} \\ \forall i \in \{1, \dots, k\}, (BU_1^i)_{N_p} = (BU_1)^i_{N_p} = -(MQ_0)^i_{N_p} = -(MQ_0^i)_{N_p} \end{cases} . \end{aligned}$$

Or, Q_0 étant pris tel que $\forall i \in \{1, \dots, k\}, (MQ_0^i, 1) = (MQ_0^i, \Psi_h) = 0$, on peut appliquer ce que l'on a fait pour la méthode de la puissance inverse pour tout $i \in \{1, \dots, k\}$ et on obtient bien le résultat.

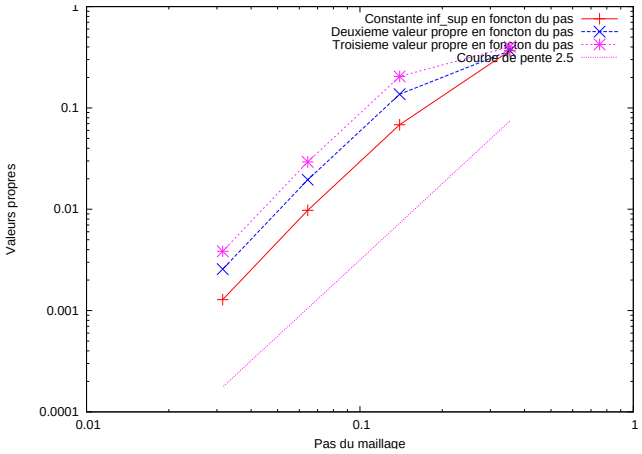
II.2 Résultats numériques

Sachant que ce schéma ne converge pas, on va s'intéresser ici au comportement des différentes valeurs propres ainsi qu'aux modes propres correspondants. On étudie deux cas tests sur le carré unité :

– on raffine uniformément en divisant par 2 le pas en x h_1 et par 2 celui en y h_2,
– on raffine "non uniformément" en divisant par 2 le pas en x et par 4 celui en y.



Raffinement uniforme



Raffinement non uniforme

h	vp_1	vp_2	vp_3	it
0.353553	0.367598	0.367598	0.400736	28
0.176777	0.2159	0.2159	0.270943	14
0.0883883	0.114818	0.114818	0.154456	15
0.0441942	0.058864	0.058864	0.0814242	15
0.0220971	0.0297589	0.0297589	0.0416532	14
0.0110485	0.0149563	0.0149563	0.0210465	17

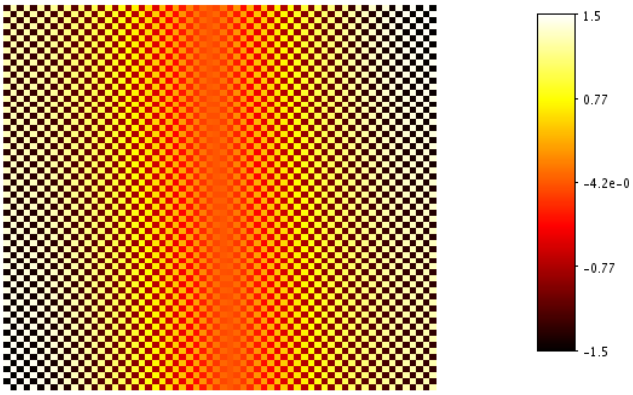
Raffinement uniforme

h	vp_1	vp_2	vp_3	it
0.353553	0.367598	0.367598	0.400736	30
0.139754	0.0682838	0.136528	0.204582	24
0.0644235	0.00975532	0.0195186	0.0292979	14
0.0314932	0.00128233	0.00256476	0.00384736	18

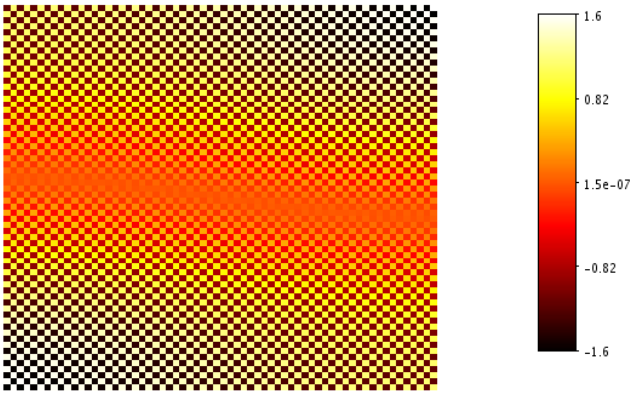
Raffinement non uniforme

Ces résultats confirment bien le fait que la constante Inf-Sup n'est pas n'est pas uniformément minorée et montre que c'est également le cas des deux valeurs propres suivantes.

Les deux premières valeurs propres étant égales dans le cas uniforme, regardons les modes propres correspondants.

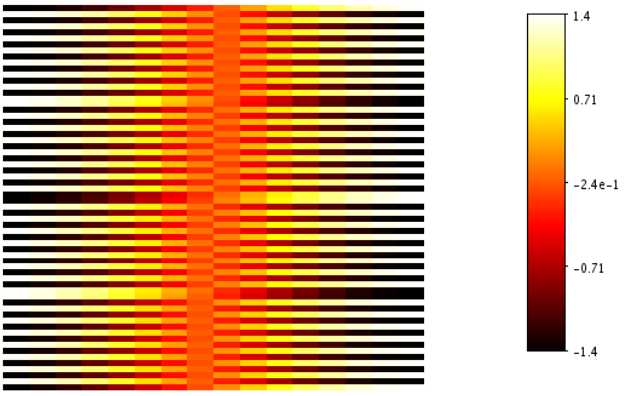


Mode propre $h_1 = h_2 = 1/64$ première valeur propre

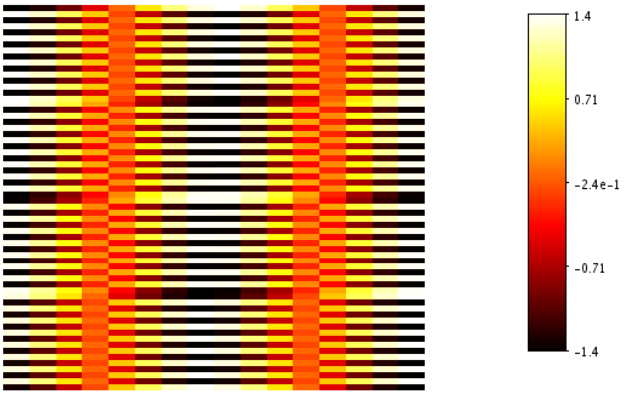


Mode propre $h_1 = h_2 = 1/64$ deuxième valeur propre

Pour les deux premières valeurs propres non nulles de cet élément fini les différents modes propres nous montrent que l'on a là encore un mode en damier avec une géométrie particulière.



Mode propre $h_1 = 1/32 = \frac{h_2}{2}$ première valeur propre



Mode propre $h_1 = 1/32 = \frac{h_2}{2}$ deuxième valeur propre

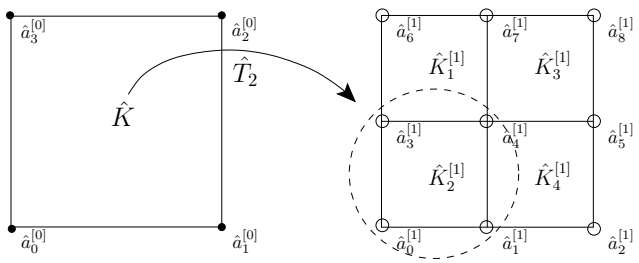
III Utilisation du logiciel PELICANS : Raffinement local

III.1 La méthode de raffinement local

3.1.1 Principe général

Nous présentons ici rapidement la méthode de raffinement utilisée par le logiciel PELICANS. Pour les démonstrations de ces résultats ainsi que des informations plus précises on pourra se référer à la thèse de Sebastian Minjeaud [Min10].

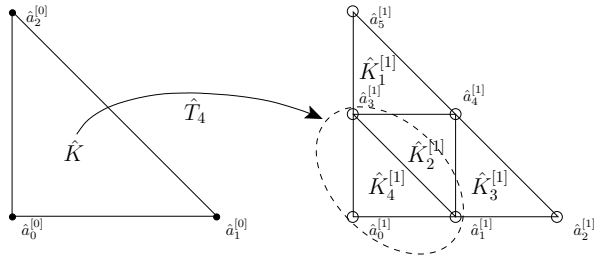
Dans la méthode des éléments finis, les inconnues sont exprimées comme une combinaison linéaire de fonctions de base dont la résolution spatiale est directement déterminée par la taille des mailles qui leur sont associées. Ainsi, pour augmenter la précision d'une zone choisie, il suffit de diviser chaque maille de cette zone en quelques mailles de plus petit diamètre. Le logiciel PELICANS utilise la méthode CHARMS (Conforming Hierarchical Adaptive Refinement MethodS) qui adopte le point de vue de "raffinement des fonctions de base" plutôt que celui des cellules ce qui évite l'apparition de mailles non conformes.
Cette approche repose sur la donnée d'ensembles B_j, $j \in [\![0, J]\!]$de fonctions de base de résolution spatiale d'autant plus fine que j est grand qui engendrent des espaces d'approximations emboités $X_0 \subset \ldots \subset X_J$, $J \geq 1$. Le raffinement local est alors réalisé en utilisant des espaces d'approximation multiniveaux, c'est-à-dire des espaces engendrés par des fonctions de base sélectionnées dans chacun des ensembles B_j, $j \in [\![0, J]\!]$selon la résolution souhaitée dans chacune des parties du domaine. Des procédures appelées CHARMS permettent de raffiner les espaces d'approximation multiniveaux, c'est-à-dire d'ajouter des fonctions de base des familles engendrant ces espaces tout en préservant leur indépendance linéaire.
La méthode repose sur la propriété fondamentale suivante : puisque $X_j \subset X_{j+1}$, toute fonction de base de B_j s'exprime comme une combinaison linéaire de certaines fonctions de base de B_{j+1}. Ces combinaisons linéaires établissent des relations parents/enfants entre les fonctions de base de deux niveaux consécutifs. Raffiner une fonction de base consiste à retirer la fonction de base et à ajouter ses enfants.



$$\hat{\varphi}_0 = \hat{\varphi}_0^{[1]} + \frac{1}{2}\hat{\varphi}_1^{[1]} + \frac{1}{2}\hat{\varphi}_3^{[1]} + \frac{1}{4}\hat{\varphi}_4^{[1]}$$
$$\hat{\varphi}_1 = \hat{\varphi}_2^{[1]} + \frac{1}{2}\hat{\varphi}_1^{[1]} + \frac{1}{2}\hat{\varphi}_5^{[1]} + \frac{1}{4}\hat{\varphi}_4^{[1]}$$
$$\hat{\varphi}_2 = \hat{\varphi}_8^{[1]} + \frac{1}{2}\hat{\varphi}_5^{[1]} + \frac{1}{2}\hat{\varphi}_7^{[1]} + \frac{1}{4}\hat{\varphi}_4^{[1]}$$
$$\hat{\varphi}_3 = \hat{\varphi}_6^{[1]} + \frac{1}{2}\hat{\varphi}_3^{[1]} + \frac{1}{2}\hat{\varphi}_7^{[1]} + \frac{1}{4}\hat{\varphi}_4^{[1]}$$

FIG. 3.1: Motif et équations de raffinement associés à l'élément carré-$\mathbb{Q}_1$

Ainsi, on commence par se donner un motif de raffinement (cf Fig. 3.1 et Fig. 3.2) et un maillage $\mathcal{T}_0$ géométriquement conforme de Ω généré à partir de l'élément de référence. En appliquant uniformément le motif de raffinement, on peut construire :
– une hiérarchie de maillages emboîtés $\mathcal{T}_0$, $\mathcal{T}_1$, …, $\mathcal{T}_J$ (cf Fig. 3.3 et Fig. 3.4),



$$\hat{\varphi}_0 = \hat{\varphi}_0^{[1]} + \frac{1}{2}\hat{\varphi}_1^{[1]} + \frac{1}{2}\hat{\varphi}_3^{[1]}$$
$$\hat{\varphi}_1 = \hat{\varphi}_2^{[1]} + \frac{1}{2}\hat{\varphi}_1^{[1]} + \frac{1}{2}\hat{\varphi}_4^{[1]}$$
$$\hat{\varphi}_2 = \hat{\varphi}_5^{[1]} + \frac{1}{2}\hat{\varphi}_3^{[1]} + \frac{1}{2}\hat{\varphi}_4^{[1]}$$

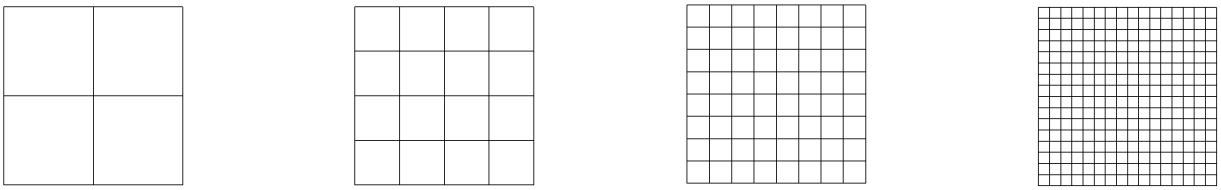
FIG. 3.2: Motif et équations de raffinement associés à l'élément triangle-$\mathbb{P}_1$

– une hiérarchie d'espaces d'approximation éléments finis $X_0 \subset X_1 \subset \ldots \subset X_J$,
– des ensembles de fonctions de base B_0, B_1, …, B_J, engendrant les espaces d'approximation précités,deux ensembles consécutifs étant reliés par les équations de raffinement.

La table 2.1 donne un résumé des notations utilisées dans la suite.

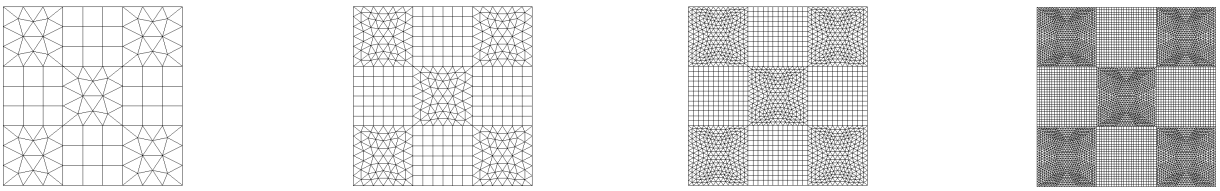
	Maillages	Ensembles de fonctions de bases	Espaces d'approximation
Niveau 0	$\mathcal{T}_0$	$B_0 = \{\varphi_k^{[0]}; k = 1, \ldots, N_{ddl}^{[0]}\}$	$X_0 = Vect(B_0)$
Niveau 1	$\mathcal{T}_1$	$B_1 = \{\varphi_k^{[1]}; k = 1, \ldots, N_{ddl}^{[1]}\}$	$X_1 = Vect(B_1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Niveau J	$\mathcal{T}_J$	$B_J = \{\varphi_k^{[J]}; k = 1, \ldots, N_{ddl}^{[J]}\}$	$X_J = Vect(B_J)$

TAB. 2.1: Hiérarchie conceptuelle d'espaces éléments finis emboîtés.



niveau 0 niveau 1 niveau 2 niveau 3

FIG. 3.3: Carré-$\mathbb{Q}_1$. Maillages emboîtés $\mathcal{T}_j$.



niveau 0 niveau 1 niveau 2 niveau 3

FIG. 3.4: Tri/Quadr-angle-$\mathbb{P}_1/\mathbb{Q}_1$. Maillages emboîtés $\mathcal{T}_j$.

On peut remarquer que la hiérarchie de maillages emboîtés n'est jamais construite explicitement. Cette structure conceptuelle est introduite pour expliquer la méthode de raffinement mais, en pratique, le motif de raffinement peut être appliqué localement aux endroits où les fonctions de base doivent être effectivement raffinées.

Ceci étant construit, on cherche à sélectionner une famille libre $\mathcal{B}^\star$ de fonctions de base dans l'ensemble multiveaux $\cup_{j=0}^{J} B_j$.

Pour cela, on choisit $\mathcal{B}^\star$ telle que deux noeuds associés à deux fonctions de bases distinctes n'aient jamais la même position géométrique.

Définition 3.1 (Base multiniveau et espace d'approximation multiniveau)
$\mathcal{B}^\star$ est alors appelée base multiniveaux et tout espace engendré par une base multiniveaux est appelé espace d'approximations multiniveau.

Remarque 3.1
Soit $\mathcal{V} = vect(\mathcal{B}^\star)$ un espace d'approximation multiniveau et $u = \sum_{\varphi \in \mathcal{B}^\star} u_\varphi \varphi \in \mathcal{V}$. La coordonnée u_φ de u dans la base multiniveau $\mathcal{B}^\star$ n'est pas nécessairement la valeur de u au noeud associé à la fonction de base φ puisque une fonction de base (d'un niveau plus "grossier") intervenant dans la décomposition de u peut avoir une contribution non nulle en ce noeud.

La base multiniveau ainsi construite $\mathcal{B}^\star$ est donc bien une famille libre mais l'on voudrait également qu'aucune information ne soit perdue au cours du processus de raffinement, ceci signifie que lorsque $\mathcal{B}$ est obtenue à partir de $\mathcal{B}^\star$ par le raffinement d'une fonction de base alors toute fonction de base représentable dans la base $\mathcal{B}^\star$ l'est (encore) dans la base $\mathcal{B}$.

Définition 3.2 (Fonction de base $\mathcal{B}$-raffinée)
Soit $\mathcal{B}$ une base multiniveau. Soit $j \in [\![0, J]\!]$ et $k \in [\![1, N_{ddl}^{[j]}]\!]$. La fonction de base $\varphi_k^{[j]}$ est dite $\mathcal{B}$-raffinée ssi et seulement s'il existe une fonction de base $\varphi_k^{[j']}$, $j \in [\![j+1, J]\!]$, $k' \in [\![1, N_{ddl}^{[j']}]\!]$ de niveau plus grand dont le noeud géométrique se trouve au même endroit que celui de $\varphi_k^{[j]}$.

A partir de ceci, on peut décrire la procédure de raffinement :
Soit $\mathcal{B}^\star$ une base multiniveau et soit $\varphi_k^{[j]}$ une fonction de base appartenant à $\mathcal{B}^\star$. Raffiner la fonction de base donnée $\varphi_k^{[j]} \in \mathcal{B}^\star$ consiste à produire une nouvelle base multiniveau $\mathcal{B}$ à partir de $\mathcal{B}^\star$ en

- enlevant cette fonction de base $\varphi_k^{[j]}$,
- ajoutant tous ces enfants $\varphi_l^{[j+1]}$ qui ne sont pas $\mathcal{B}^\star$-raffinés.

C'est à dire que l'on a :

$$\mathcal{B} = \mathcal{B}^\star \backslash \{\varphi_k^{[j]}\} \cup \{ \text{ enfants de } \varphi_k^{[j]} \text{ non } \mathcal{B}^\star\text{-raffinés}\}.$$

Or, une propriété souhaitable de la procédure de raffinement est qu'elle permette de conserver l'information. Cela signifie que, si $\mathcal{B}$ est obtenue à partir de $\mathcal{B}^\star$ par le raffinement d'une fonction de base alors $vect(\mathcal{B}^\star) \subset vect(\mathcal{B})$, i.e. la base raffinée $\mathcal{B}$ autorise la représentation exacte de chaque fonction de la base originale $\mathcal{B}^\star$ ce qui n'est pas le cas ici.
Pour remédier à ceci, on a la propriété suivante :

Proposition 3.1
Soit $\mathcal{B}$ une base multiniveau satisfaisant la propriété $(\mathcal{P}_{LO})$ suivante :

Tout enfant d'une fonction de base $\mathcal{B}$-raffinée est soit lui même $\mathcal{B}$-raffiné, soit dans $\mathcal{B}$.

Alors, toutes les fonctions de base $\mathcal{B}$-raffinées appartiennent à $vect(\mathcal{B})$.

On a alors :

Théorème 3.1
Soit $\mathcal{B}$ une base multiniveau satisfaisant la propriété $(\mathcal{P}_{LO})$, et obtenue à partir d'une base multiniveau $\mathcal{B}^\star$ par la procédure de raffinement précédente. Alors, nous avons :

$$vect(\mathcal{B}^\star) \subset vect(\mathcal{B}).$$

Ainsi, on peut garantir :

- que l'algorithme d'adaptation que nous proposons préserve l'indépendance linéaire des fonctions de base sélectionnées sur les différents niveaux,
- qu'aucune information n'est perdue lorsque nous raffinons une fonction de base.

3.1.2 La règle "au-plus-un-niveau-de-différence"

Lorsque les espaces multiniveaux sont utilisés comme espaces d'approximation dans une méthode de Galerkin, il peut être intéressant d'imposer une condition sur le nombre de niveaux de raffinement séparant deux fonctions de base dont les supports s'intersectent. En effet, ceci permet de contrôler la largeur de bande de la matrice de rigidité (i.e. nombre d'éléments non nuls par ligne) et ainsi garantir la structure creuse de cette matrice.
La figure 3.5 montre une situation simple où l'algorithme précédent mène à la construction d'espaces d'approximation de fonctions de base dont les supports s'intersectent deux à deux. Le domaine est choisi carré, le maillage initial est formé d'une seule

maille et la base multiniveau $\mathcal{B}_0$ constituée des quatre fonctions de base de niveau 0. La base multiniveau $\mathcal{B}_{k+1}$ est ensuite obtenue par raffinement de la fonction base (de niveau k) appartenant à $\mathcal{B}_k$ associée au noeud (indiqué par une flèche sur la figure) placé dans le coin en bas à gauche du domaine. Les matrices de rigidité associées à ces espaces d'approximation sont pleines (elles ne possèdent aucun élément nul a priori). Nous ajoutons alors des règles pratiques à l'algorithme précédent pour éviter d'aboutir à ce type de configuration. Ces règles ont pour effet d'augmenter le nombre de fonctions de base effectivement raffinées.

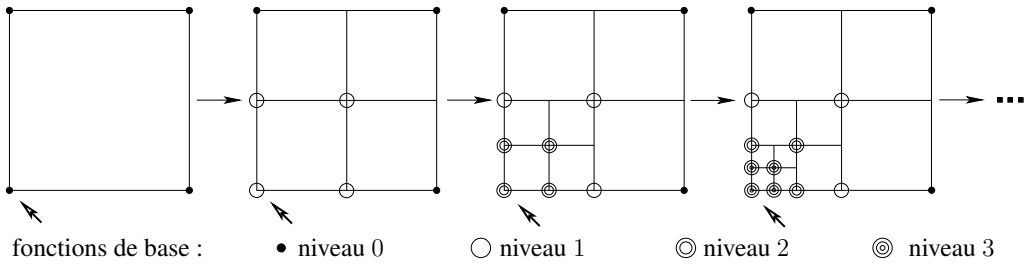


FIG. 3.5: Exemple d'espaces multiniveaux conduisant à assembler des matrices de rigidités pleines.

Pour cela, nous introduisons la notion d'ascendants d'une fonction de base $\varphi_k^{[j]}$.

Définition 3.3 (Ascendants)
Soit $\varphi_k^{[j]}$ une fonction de base de niveau j. Nous appelons ascendant de la fonction de base $\varphi_k^{[j]}$ toute fonction de base de niveau $j-1$ dont le support intersecte celui de $\varphi_k^{[j]}$.

Ainsi, on ajoute à l'algorithme précédent la règle "au-plus-un-niveau-de-différence" qui dit que :

Lorsque l'on raffine une fonction de base, on raffine également tous ses ascendants.

Cette règle pratique permet de garantir que l'application de l'algorithme précédent ne conduira pas à une augmentation de l'écart entre les niveaux des fonctions de base des espaces d'approximation produits. A titre d'exemple, reprenons la séquence de raffinement présentée sur la figure 3.5 mais appliquons maintenant la règle pratique énoncée. Nous obtenons la séquence de raffinement présentée sur la figure 3.6. Cette règle force le raffinement de fonctions de base supplémentaires (marquées par des flèches en pointillés). Les bases multiniveaux contiennent plus de fonctions de base que celles présentées sur la figure A METTRE mais les matrices assemblées contiennent moins de coefficients non nuls.

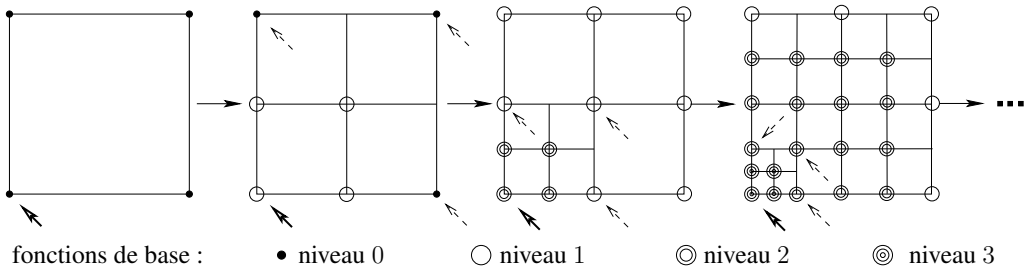


FIG. 3.6: Application de la règle "au-plus-un-niveau-de-différence"

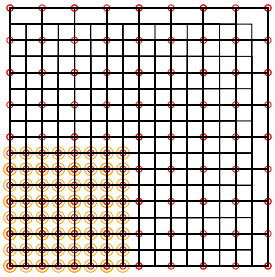
III.2 Les différents cas tests

Le jeu de données de base pour les tests avec PELICANS a été fait par Sebastian Minjeaud qui nous a permis de construire la matrice de résolution complète du système de Stokes, la matrice de masse en vitesse, le vecteur qui se trouve dans le noyau de la transposée de la matrice de divergence (voir la section suivante) ainsi que les résultats du calcul.
Nous en avons ensuite modifié les données pour pouvoir effectuer différents tests.

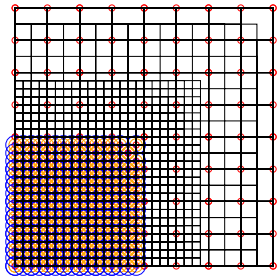
Voilà les différents cas tests dont on schématisera à chaque fois les 3 premières étapes de raffinement.
Pour les différents cas test on repère sur un tableau le nombre n du raffinements effectués, la taille N_u de la matrice de rigidité, le rapport $R_z = \frac{N_n}{N_u}$ où N_n est le nombre d'éléments non nuls de la matrice de rigidité et le stencil N_s de la matrice de rigidité c'est à dire le nombre maximal d'éléments non nuls par ligne.

– 1^e cas test : on raffine progressivement le noeud en bas à gauche du maillage sans utiliser la règle de "au-plus-un-niveau-de-différence".
On effectue alors 11 étapes :

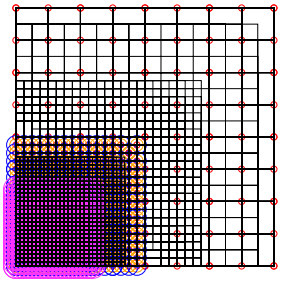
1. on raffine d'abord la zone $x < 0.5$ et $y < 0.5$,
2. puis la zone $x < 0.4$ et $y < 0.4$,
3. puis la zone $x < 0.3$ et $y < 0.3$,
4. puis la zone $x < 0.2$ et $y < 0.2$,
5. puis la zone $x < 0.1$ et $y < 0.1$,
6. puis la zone $x < 0.05$ et $y < 0.05$,
7. puis la zone $x < 0.04$ et $y < 0.04$,
8. puis la zone $x < 0.03$ et $y < 0.03$,
9. puis la zone $x < 0.02$ et $y < 0.02$,
10. puis la zone $x < 0.01$ et $y < 0.01$,
11. et enfin la zone $x < 0.005$ et $y < 0.005$.



Etape 1



Etape 2

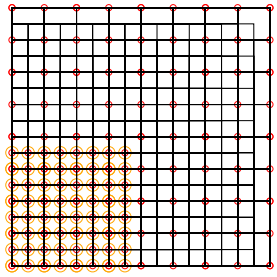


Etape 3

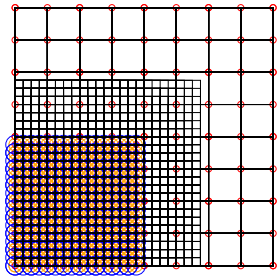
n	N_u	N_z	N_s
1	178	13.6742	42
2	498	17.0643	122
3	1058	17.983	180
4	2122	17.3657	180
5	3186	17.1607	180
6	4250	17.0584	180
7	6978	16.851	180
8	12866	16.6356	180
9	23114	16.4853	180
10	33362	16.4274	180
11	43610	16.3967	180

1^e cas test

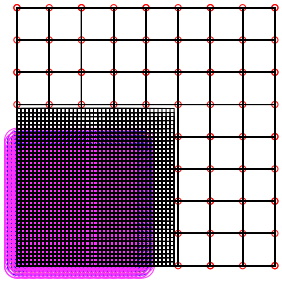
– 2^e cas test : on raffine la zone $x < 0.5$ et $y < 0.5$ six fois sans utiliser la règle "au-plus-un-niveau-de-différence".



Etape 1

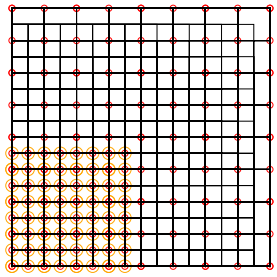


Etape 2

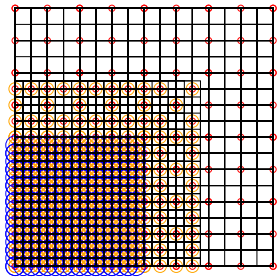


Etape 3

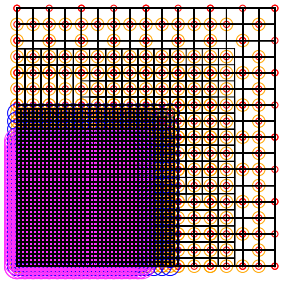
– 3^e cas test : on raffine la zone $x < 0.5$ et $y < 0.5$ six fois mais cette fois en utilisant la règle "au-plus-un-niveau-de-différence".



Etape 1



Etape 2

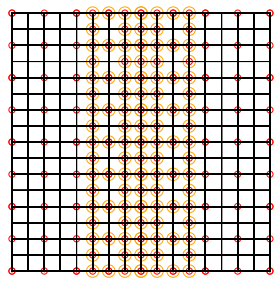


Etape 3

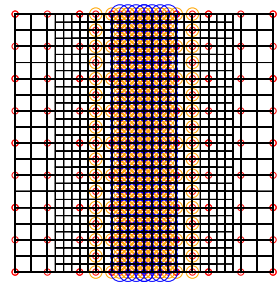
	Sans la règle			Avec la règle		
n	N_u	N_z	N_s	N_u	N_u	N_s
1	178	13.6742	42	178	13.6742	42
2	530	17.2113	134	618	16.1068	54
3	2002	19.1019	510	2410	16.2066	56
4	8018	19.8875	2030	9002	16.3397	56
5	32338	20.2175	8142	34410	16.2377	56
6	130130	20.3649	32654	134314	16.1371	56

Cas tests 2 et 3

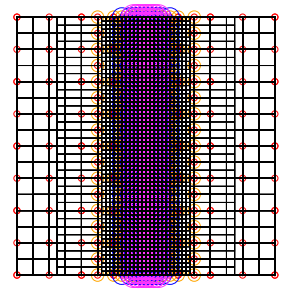
– 4^e cas test : on raffine la zone $x < 0.45$ et $x > 0.55$ qui encadre la droite d'équation $x = 0.5$ avec un pas $h = 1/4$ six sans utiliser la règle "au-plus-un-niveau-de-différence".



Etape 1

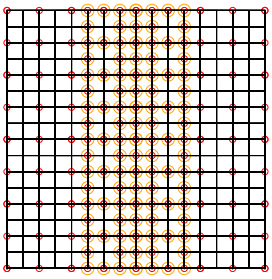


Etape 2

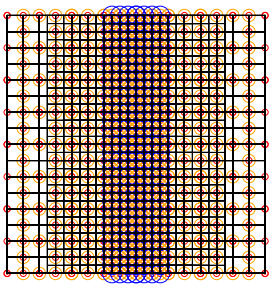


Etape 3

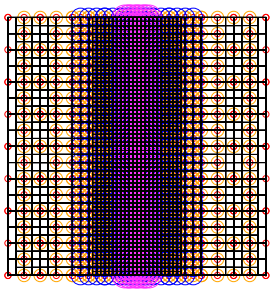
– 5^e cas test : on raffine la zone $x < 0.45$ et $x > 0.55$ qui encadre la droite d'équation $x = 0.5$ avec un pas $h = 1/4$ six fois mais cette fois en utilisant la règle "au-plus-un-niveau-de-différence".



Etape 1



Etape 2

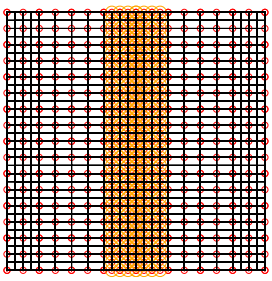


Etape 3

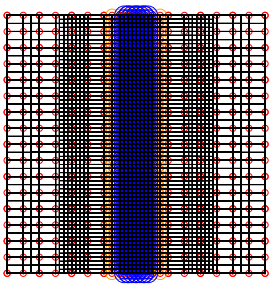
	Sans la règle			Avec la règle		
n	N_u	N_z	N_s	N_u	N_u	N_s
1	234	15.8034	42	234	15.8034	42
2	514	21.3268	84	698	16.5301	47
3	1210	27.1355	182	1770	18.0441	51
4	4138	30.1986	570	5330	17.9666	56
5	15626	31.3325	2050	19258	17.258	52
6	57042	32.1162	7312	65266	16.8147	52

Cas tests 4 et 5

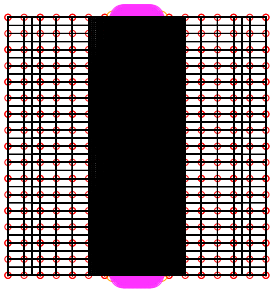
– 6^e cas test : on raffine la zone $x < 0.45$ et $x > 0.55$ qui encadre la droite d'équation $x = 0.5$ avec un pas $h = 1/8$ cinq fois sans utiliser la règle "au-plus-un-niveau-de-différence".



Etape 1

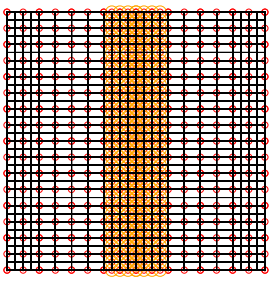


Etape 2

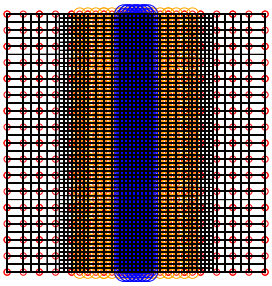


Etape 3

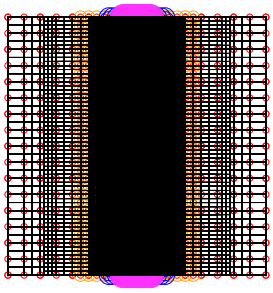
– 7^e cas test : on raffine la zone $x < 0.45$ et $x > 0.55$ qui encadre la droite d'équation $x = 0.5$ avec un pas $h = 1/8$ six fois mais cette fois en utilisant la règle "au-plus-un-niveau-de-différence".



Etape 1



Etape 2

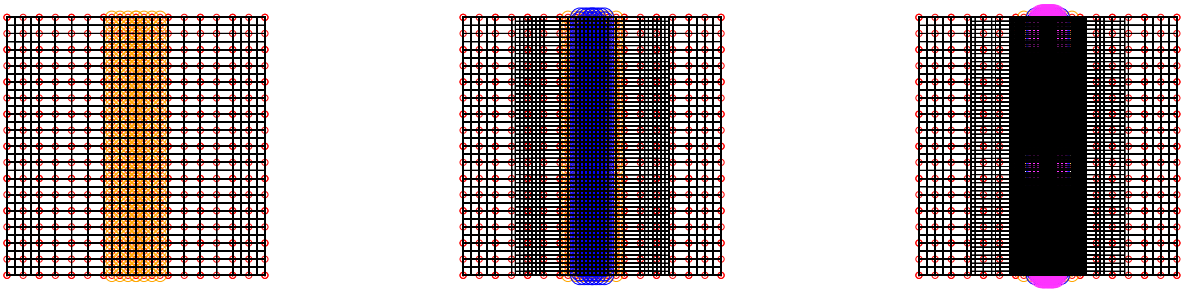


Etape 3

	Sans la règle			Avec la règle		
n	N_u	N_z	N_s	N_u	N_u	N_s
1	730	16.3753	47	730	16.3753	47
2	1426	21.3534	97	1802	17.9545	51
3	4354	25.0147	293	5362	17.937	56
4	15842	26.5675	1037	19258	17.258	52
5	57258	27.505	3675	65266	16.8147	52
6	215874	27.8136	13685	230018	16.4873	52

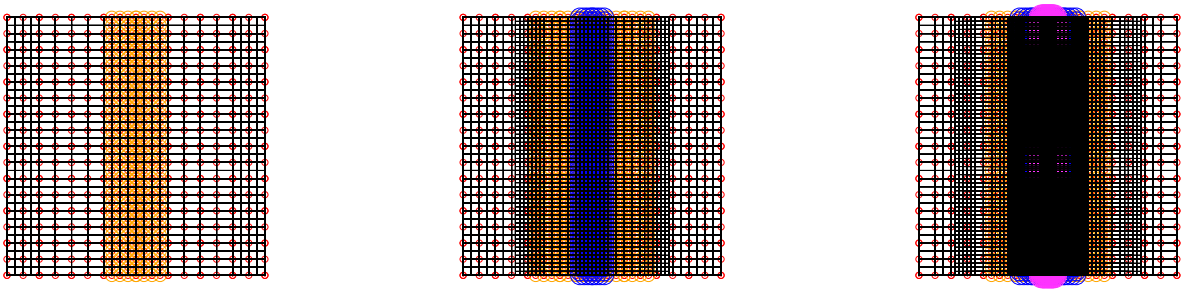
Cas tests 6 et 7

– 8^e cas test : on raffine la zone $x < 0.475$ et $x > 0.525$ qui encadre la droite d'équation $x = 0.5$ avec un pas $h = 1/8$ six fois sans utiliser la règle "au-plus-un-niveau-de-différence".



Etape 1 Etape 2 Etape 3

– 9^e cas test : on raffine la zone $x < 0.475$ et $x > 0.525$ qui encadre la droite d'équation $x = 0.5$ avec un pas $h = 1/8$ six fois mais cette fois en utilisant la règle "au-plus-un-niveau-de-différence".

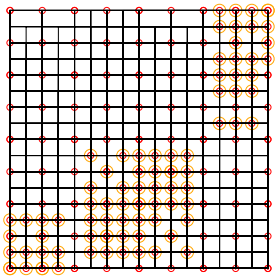


Etape 1 Etape 2 Etape 3

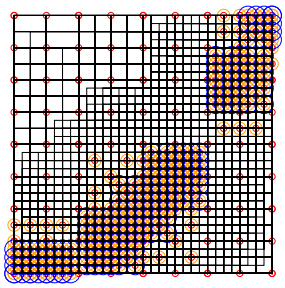
	Sans la règle			Avec la règle		
n	N_u	N_z	N_s	N_u	N_u	N_s
1	730	16.3753	47	730	16.3753	47
2	1298	21.3143	89	1674	18.0657	47
3	2698	27.3751	187	3834	18.7507	52
4	8570	31.1377	575	10978	18.2855	56
5	31578	32.5473	2055	39202	17.3808	52
6	114466	33.3869	7317	131290	16.8667	52

Cas tests 8 et 9

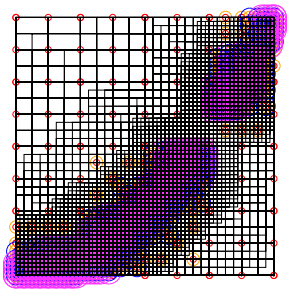
– 10^e cas test : on raffine la courbe $y = x^2$ de part et d'autre d'un paramètre $\varepsilon = 0.05$ avec un pas $h = 1/4$ six fois sans utiliser la règle "au-plus-un-niveau-de-différence".



Etape 1

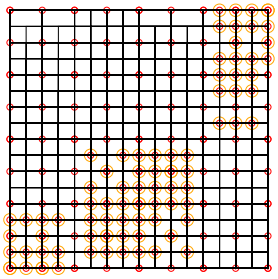


Etape 2

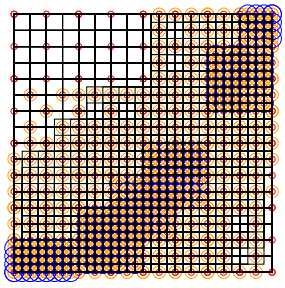


Etape 3

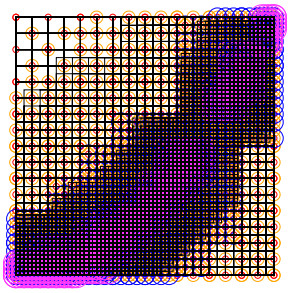
– 11^e cas test : on raffine la courbe $y = x^2$ de part et d'autre d'un paramètre $\varepsilon = 0.05$ avec un pas $h = 1/4$ six fois mais cette fois en utilisant la règle "au-plus-un-niveau-de-différence".



Etape 1



Etape 2

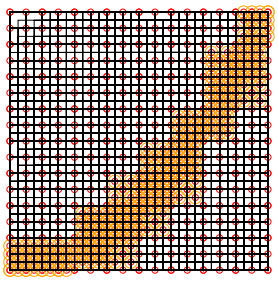


Etape 3

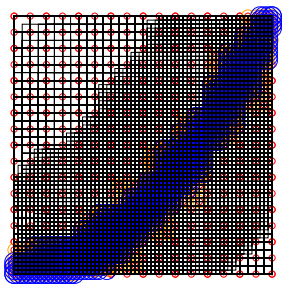
	Sans la règle			Avec la règle		
n	N_u	N_z	N_s	N_u	N_u	N_s
1	202	14.3069	43	202	14.3069	43
2	482	19.7967	104	662	17.1088	57
3	1310	25.5008	302	2042	18.2968	66
4	4034	29.8597	966	6178	18.3286	77
5	13462	32.3832	3330	18758	17.9722	77
6	48454	33.4897	12234	61002	17.3686	77

Cas tests 10 et 11

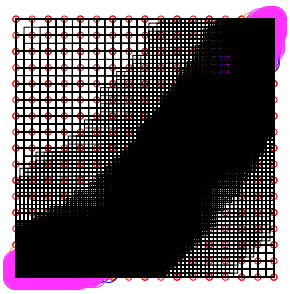
– 12^e cas test : on raffine la courbe $y = x^2$ de part et d'autre d'un paramètre $\varepsilon = 0.05$ avec un pas $h = 1/8$ six fois sans utiliser la règle "au-plus-un-niveau-de-différence".



Etape 1

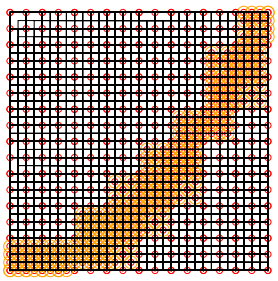


Etape 2

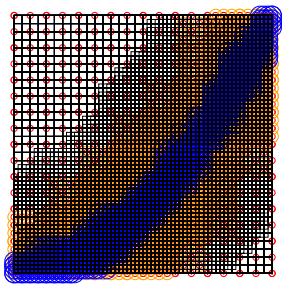


Etape 3

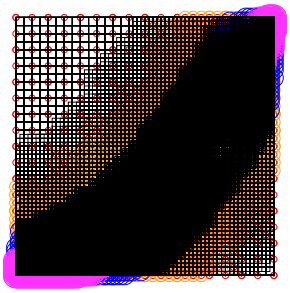
– 13^e cas test : on raffine la courbe $y = x^2$ de part et d'autre d'un paramètre $\varepsilon = 0.05$ avec un pas $h = 1/8$ six fois mais cette fois en utilisant la règle "au-plus-un-niveau-de-différence".



Etape 1



Etape 2

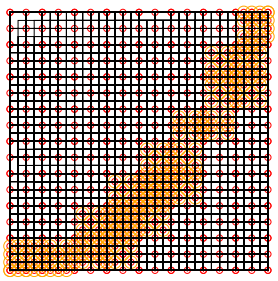


Etape 3

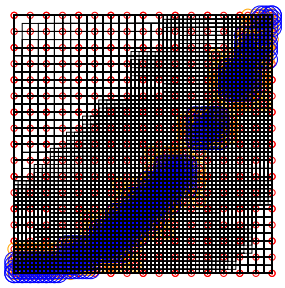
	Sans la règle			Avec la règle		
n	N_u	N_z	N_s	N_u	N_u	N_s
1	790	16.7975	59	790	16.7975	59
2	1786	22.0168	168	2266	18.2798	66
3	4962	26.522	540	6802	18.2185	66
4	15782	29.1797	1834	20606	17.8384	77
5	55434	30.3048	6686	66770	17.2788	77
6	207182	30.6073	25548	232186	16.7925	77

Cas tests 12 et 13

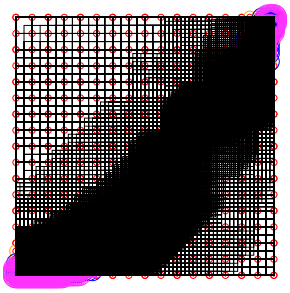
– 14^e cas test : on raffine la courbe $y = x^2$ de part et d'autre d'un paramètre $\varepsilon = 0.025$ avec un pas $h = 1/8$ six fois sans utiliser la règle "au-plus-un-niveau-de-différence".



Etape 1

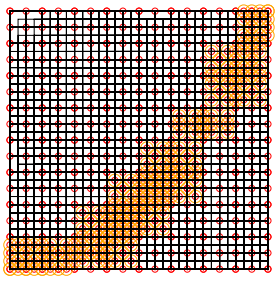


Etape 2

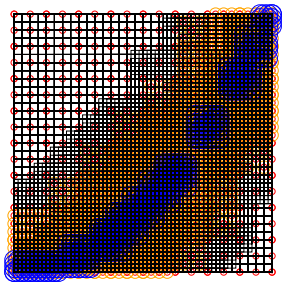


Etape 3

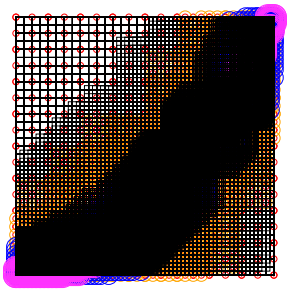
– 15^e cas test : on raffine la courbe $y = x^2$ de part et d'autre d'un paramètre $\varepsilon = 0.025$ avec un pas $h = 1/8$ six fois mais cette fois en utilisant la règle "au-plus-un-niveau-de-différence".



Etape 1



Etape 2



Etape 3

	Sans la règle			Avec la règle		
n	N_u	N_z	N_s	N_u	N_u	N_s
1	726	16.1405	53	726	16.1405	53
2	1318	20.9848	114	1862	18.0913	59
3	3038	27.0566	342	4926	18.8514	71
4	8658	31.9942	1102	13878	18.7095	75
5	28306	34.7838	3834	40310	18.1722	77
6	100782	35.9535	14096	128622	17.4745	77

Cas tests 14 et 15

Ces tableaux nous montrent bien l'intérêt de la règle "au-plus-un-niveau-de-différence". En effet, le stencil de la matrice de rigidité reste quasi-constant lorsque cette règle est activée alors qu'il augmente considérablement dans le cas contraire. Les calculs sont donc beaucoup plus long lorsque la règle "au-plus-un-niveau-de-différence" est désactivée.

III.3 Les méthodes de valeurs propres

Avec ce logiciel on va encore rencontrer un problème avec la modélisation de l'espace des pressions $L^2_0(\Omega)$ mais cette fois le vecteur présent dans le noyau de la transposée de la matrice de divergence B ne sera pas le vecteur constant unité. En effet, on se retrouve ici avec du raffinement local et donc une pression constante ne correspond pas à un vecteur de degrés de liberté constant. Les fonctions de bases en pression $(\Psi_i)_{i=1\ldots,N_p}$ utilisées ne sont pas du même niveau et ont donc des supports qui se recouvrent. Il faut ainsi ajuster les valeurs des degrés de liberté de la pression en fonction du support des différentes fonctions de bases.
Appelons v_{noyau} le vecteur présent dans le noyau de B^T.

Cette différence engendre quelques modifications dans les programmes calculant les valeurs propres.

3.3.1 La méthode de la puissance inverse :

Rappelons que pour cette méthode on veut, à chaque itération, résoudre le problème de Stokes suivant :

$$\begin{cases} RU + B^T P = 0 \\ BU = -MQ \end{cases}$$

avec Q pris tel que q soit à moyenne nulle afin que ce premier problème de Stokes soit résolu à l'aide d'un second :

$$\begin{cases} RU + \tilde{B}^T \tilde{P} = 0 \\ \tilde{B}U = -\widetilde{MQ} \end{cases} .$$

Ici, on n'a pas $B^T \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = 0$ mais $B^T v_{noyau} = 0$ on va donc choisir Q tel que $(MQ, v_{noyau}) = 0$.
La méthode devient alors :

1. On choisit $\tilde{Q}$ aléatoirement.
2. On pose $Q = \tilde{Q} - \frac{1}{(Mv_{noyau}, v_{noyau})}(M\tilde{Q}, v_{noyau})v_{noyau}$ de sorte que $(MQ, v_{noyau}) = 0$.
3. On pose $\tilde{\tilde{P}} = \begin{pmatrix} \tilde{P} \\ 0 \end{pmatrix}$ puis $P = \tilde{\tilde{P}} - \frac{1}{(Mv_{noyau}, v_{noyau})}(M\tilde{\tilde{P}}, v_{noyau})v_{noyau}$ pour que l'on ait bien $(MP, v_{noyau}) = 0$ ce qui sera utile à l'itération suivante vu que l'on pose $Q = P$.

Vérifions que là encore le premier problème de Stokes est vérifié lorsque le second l'est.

– Tout d'abord, on a bien :

$$RU + B^T P = RU + B^T \tilde{\tilde{P}} - \frac{1}{(Mv_{noyau}, v_{noyau})}(M\tilde{\tilde{P}}, v_{noyau})B^T v_{noyau} = RU + B^T \tilde{\tilde{P}} = RU + \tilde{B}^T \tilde{P}.$$

– Ensuite, on a $\tilde{B}U = -\widetilde{MQ}$, il suffit donc de vérifier que $(BU)_{N_p} = -(MQ)_{N_p}$ sachant que :

$$\begin{cases} (BU, v_{noyau}) = (U, B^T v_{noyau}) = 0 \text{ et } (MQ, v_{noyau}) = 0 \Rightarrow (BU + MQ, v_{noyau}) = 0, \\ \forall i \in \{1, \ldots, N_p - 1\}, (BU)_i = -(MQ)_i. \end{cases}$$

On a donc bien $(BU)_{N_p} = -(MQ)_{N_p}$.

3.3.2 La méthode d'itérations de sous-espaces de Rayleigh-Ritz :

Rappelons que s'il on veut calculer k valeurs propres, on souhaite, à chaque itération, résoudre le problème de Stokes :

$$\text{Trouver } \begin{pmatrix} U \\ P \end{pmatrix} = \begin{pmatrix} U^1 & \dots & U^k \\ P^1 & \dots & P^k \end{pmatrix} \text{ tels que } \begin{cases} RU + B^T P = 0 \\ BU = -MQ \end{cases},$$

et commence là encore par résoudre le problème :

$$\text{Trouver } \begin{pmatrix} U \\ \tilde{P} \end{pmatrix} = \begin{pmatrix} U^1 & \dots & U^k \\ \tilde{P}^1 & \dots & \tilde{P}^k \end{pmatrix} \text{ tels que } \begin{cases} RU + \tilde{B}^T \tilde{P} = 0 \\ \tilde{B}U = -\widetilde{MQ} \end{cases}.$$

Le raisonnement est alors le suivant :

1. On choisit k vecteurs $\bar{Q}_0^1, \dots, \bar{Q}_0^k$ aléatoires.
2. On pose $Q_0^i = \bar{Q}_0^i - \frac{1}{(Mv_{noyau}, v_{noyau})}(M\bar{Q}_0^i, v_{noyau})v_{noyau}$ pour tout $i \in \{1, \dots, k\}$ de sorte que $(MQ_0^i, v_{noyau}) = 0$, puis on pose $Q_0 = [Q_0^1, \dots, Q_0^M]$.
3. On modifie Q_0^i pour tout $i \in \{2, \dots, k\}$ en procédant par ordre croissant de manière à ce que $Q_0^T M Q_0 = Id$:
$$Q_0^i = Q_0^i - \sum_{j=2}^{i} (MQ_0^i, Q_0^j) \frac{Q_0^j}{(MQ_0^j, Q_0^j)} \text{ (puis on les norme).}$$
4. On trouve U_1 et $\tilde{P}_1$ tels que $\begin{cases} RU_1 + \tilde{B}^T \tilde{P}_1 = 0 \\ \tilde{B}U_1 = -\widetilde{MQ_0} \end{cases}$.
5. On pose $\tilde{\tilde{P}}_1 = \begin{pmatrix} \tilde{P}_1 \\ \underbrace{0 \dots 0}_{k fois} \end{pmatrix}$ puis $P_1 = \tilde{\tilde{P}}_1 - \frac{1}{|\Omega|}[(M\tilde{\tilde{P}}_1^1, v_{noyau})v_{noyau}, \dots, (M\tilde{\tilde{P}}_1^k, v_{noyau})v_{noyau}]$ où $\tilde{\tilde{P}}_1 = [\tilde{\tilde{P}}_1^1, \dots, \tilde{\tilde{P}}_1^k]$.
6. Par une factorisation QR, on trouve $\tilde{Q}_1$ et R_1 tels que $P_1 = \tilde{Q}_1 R_1$ de sorte que $\tilde{Q}_1^T M \tilde{Q}_1 = Id$.
7. On pose $B_1 = \tilde{Q}_1^T M (M^{-1} B R^{-1} B^T) \tilde{Q}_1$.
8. On cherche les vecteurs de propres Y_1 de B_1, c'est à dire que $[Y_1, D_1] = spec(B_1)$.
9. On pose $Q_1 = \tilde{Q}_1 Y_1$.
   On remarque alors que par un raisonnement analogue à celui du premier chapitre on montre que, P_1 étant choisi tel que $(MP_1^i, v_{noyau}) = 0$, $Q_1 = \tilde{Q}_1 Y_1$ sera lui aussi tel que $(MQ_1^i, v_{noyau}) = 0$ pour tout $i \in \{1, \dots, k\}$.

Ainsi, on a tout d'abord :

$$\begin{aligned} RU_1 + B^T P_1 &= RU_1 + B^T \tilde{\tilde{P}}_1 - \frac{1}{|\Omega|}[(M\tilde{\tilde{P}}_1^1, v_{noyau})B^T v_{noyau}, \dots, (M\tilde{\tilde{P}}_1^k, v_{noyau})B^T v_{noyau}] \\ &= RU_1 + B^T \tilde{\tilde{P}}_1 = RU_1 + \tilde{B}^T \tilde{P}_1 = 0. \end{aligned}$$

Enfin, $\tilde{B}U_1 = -\widetilde{MQ_0}$, donc $\forall i \in \{1, \dots, k\}$, $\forall l \in \{1, \dots, N_p - 1\}$, $(BU_1^i)_l = -(MQ_0^i)_l$, il reste donc à montrer que :

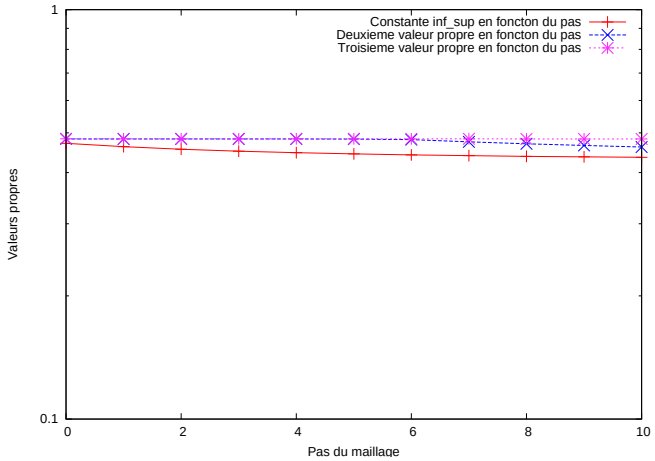
$$\forall i \in \{1, \dots, k\}, (BU_1^i)_{N_p} = -(MQ_0^i)_{N_p}$$

Ceci se montre alors en appliquant le même raisonnement que pour la méthode de la puissance inverse pour tout $i \in \{1, \dots, k\}$ et ainsi la résolution du second problème entraîne bien la résolution du premier.

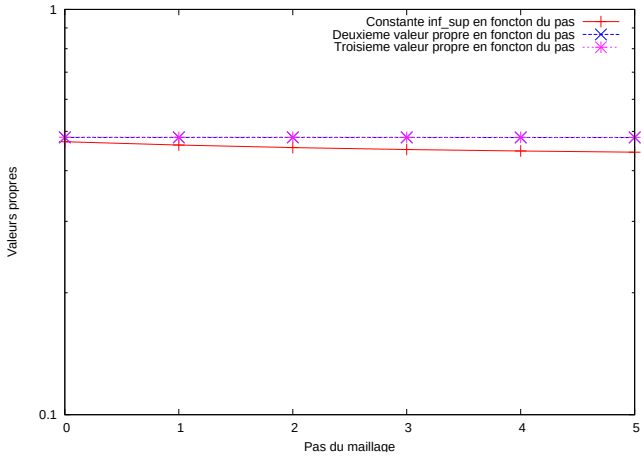
III.4 Résultats numériques

Avec le logiciel PELICANS, on n'effectuera pas de test entre les solutions exacte et approchée pour étudier la convergence mais seulement les tests sur les valeurs propres.

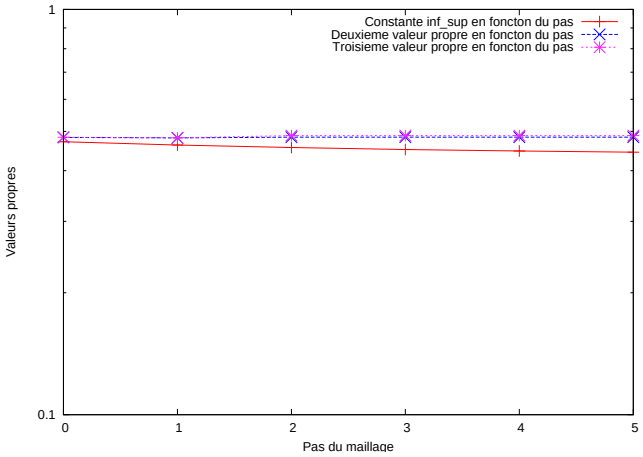
– 1^e cas test :



– 2^e et 3^e cas tests :

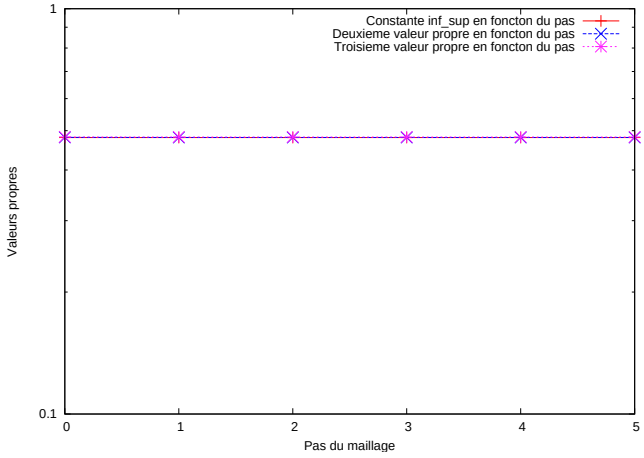


Raffinement zone sans la règle
"au-plus-un-niveau-de-différence"

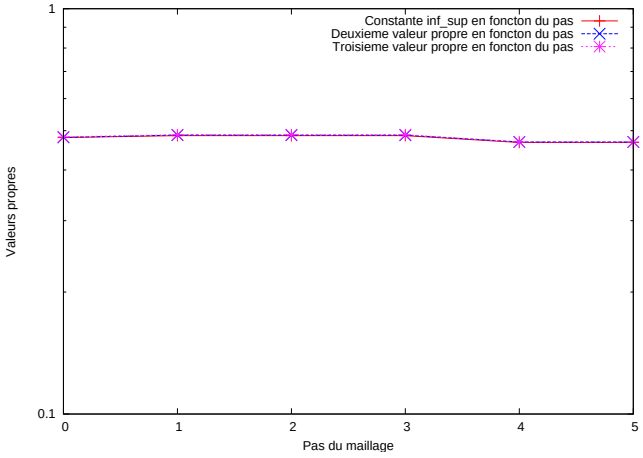


Raffinement zone avec la règle
"au-plus-un-niveau-de-différence"

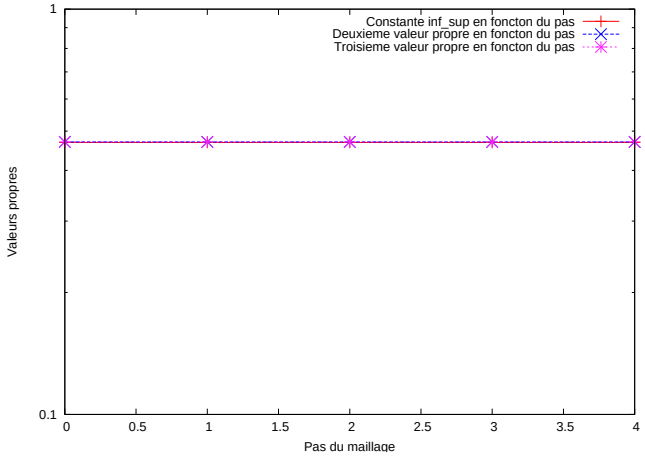
– Cas tests 4 à 9 :



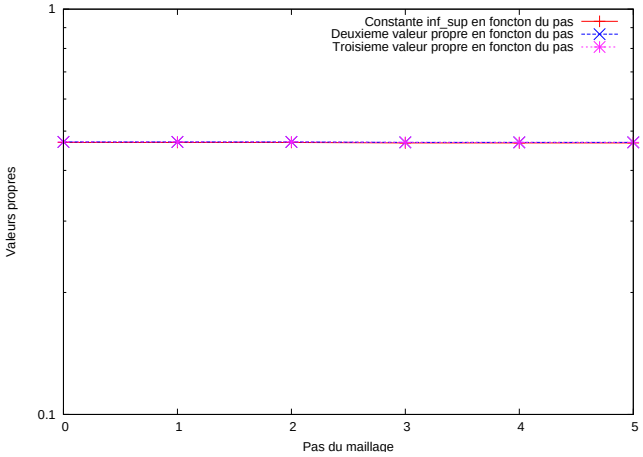
Raffinement droite, $h = 1/4$, sans la règle
"au-plus-un-niveau-de-différence"



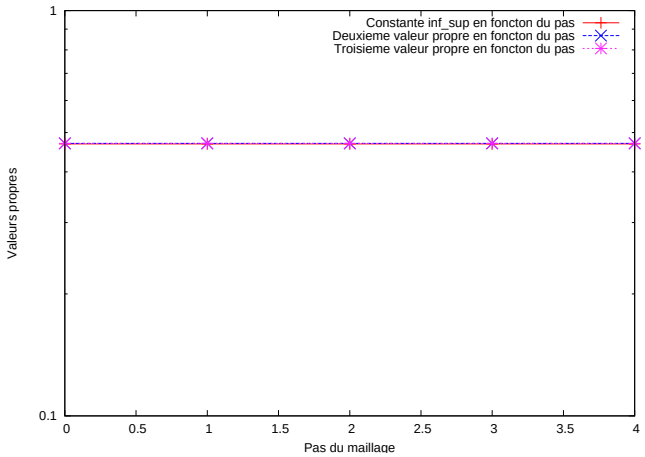
Raffinement droite, $h = 1/4$, avec la règle
"au-plus-un-niveau-de-différence"



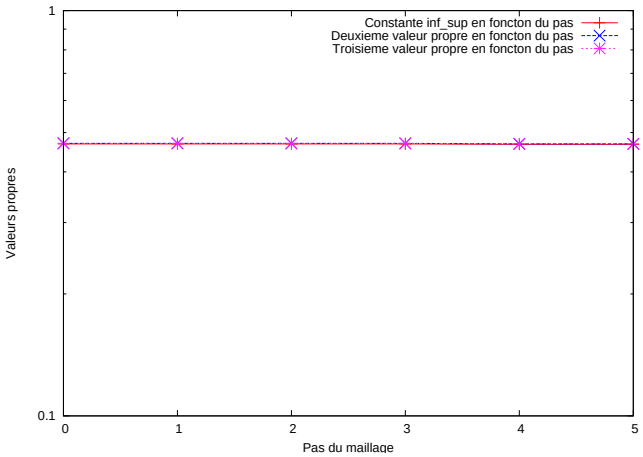
Raffinement droite, $h = 1/8$, $\varepsilon = 0.05$, sans la règle "au-plus-un-niveau-de-différence"



Raffinement droite, $h = 1/8$, $\varepsilon = 0.05$, avec la règle "au-plus-un-niveau-de-différence"



Raffinement droite, $h = 1/8$, $\varepsilon = 0.025$, sans la règle "au-plus-un-niveau-de-différence"

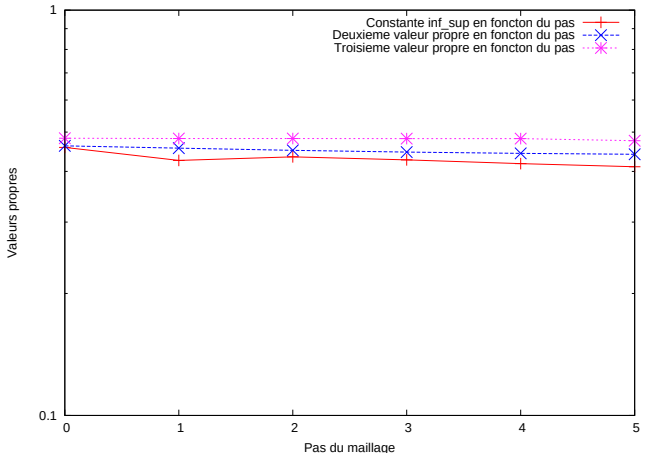


Raffinement droite, $h = 1/8$, $\varepsilon = 0.025$, avec la règle "au-plus-un-niveau-de-différence"

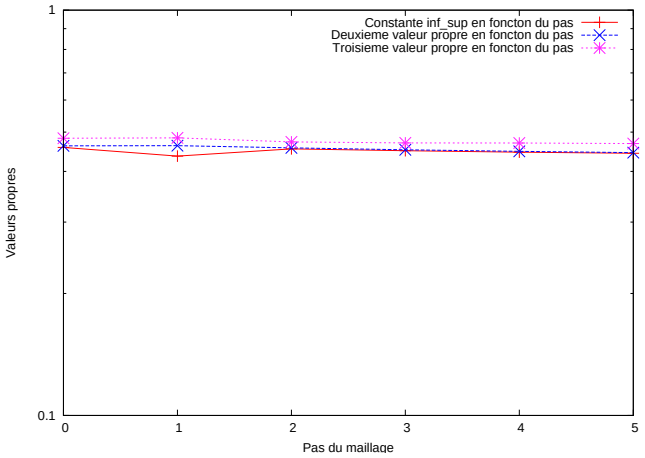
	Pas du maillage $h = 1/4$			Pas du maillage $h = 1/8$					
				$\varepsilon = 0.05$			$\varepsilon = 0.025$		
n	vp_1	vp_2	vp_3	vp_1	vp_2	vp_3	vp_1	vp_2	vp_3
1	0.481061	0.481414	0.482601	0.468904	0.470352	0.470353	0.468904	0.470352	0.470353
2	0.486377	0.487657	0.488910	0.468796	0.47024	0.470243	0.468796	0.470240	0.470243
3	0.48634	0.487618	0.488868	0.468795	0.470239	0.470241	0.468795	0.470239	0.470242
4	0.486339	0.487617	0.488866	0.467728	0.469127	0.469128	0.468795	0.470239	0.470241
5	0.467728	0.469127	0.469128	0.467726	0.469123	0.469124	0.467727	0.469126	0.469127
6	0.467726	0.469123	0.469124	0.46772	0.4691	0.469111	0.467724	0.469116	0.46912

Raffinement de la droite avec la règle "au-plus-un-niveau-de-differnce"

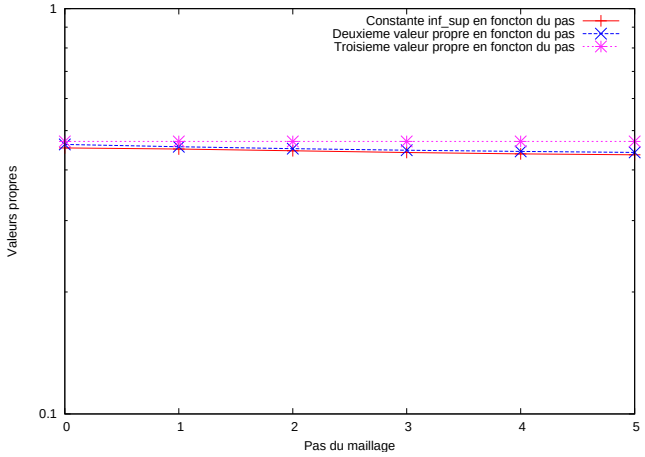
– Cas tests 10 à 15 :



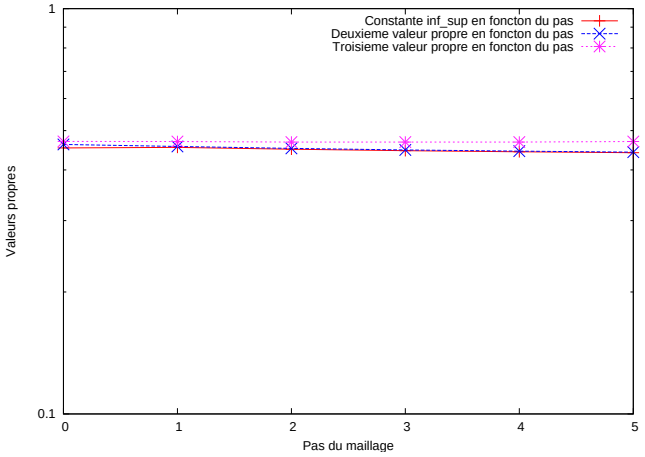
Raffinement courbe, $h = 1/4$, sans la règle "au-plus-un-niveau-de-différence"



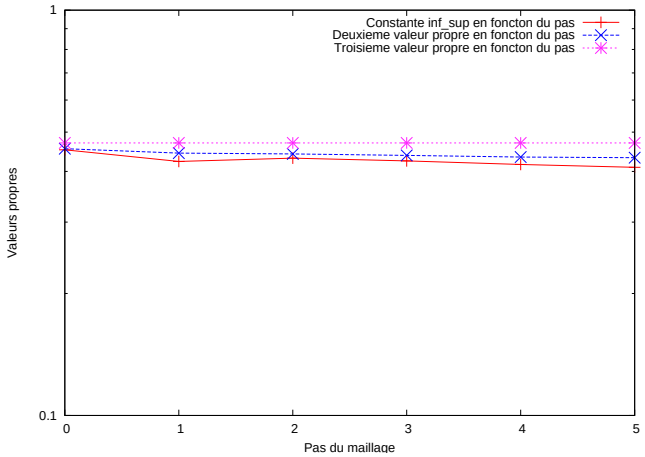
Raffinement courbe, $h = 1/4$, avec la règle "au-plus-un-niveau-de-différence"



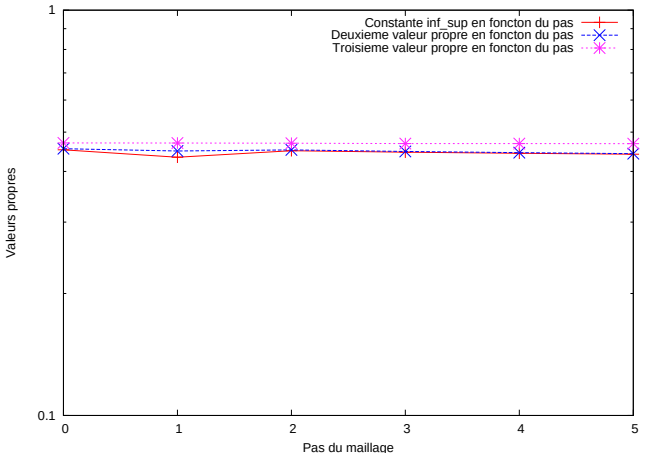
Raffinement courbe, $h = 1/8$, $\varepsilon = 0.05$, sans la règle "au-plus-un-niveau-de-différence"



Raffinement courbe, $h = 1/8$, $\varepsilon = 0.05$, avec la règle "au-plus-un-niveau-de-différence"



Raffinement courbe, $h = 1/8$, $\varepsilon = 0.025$, sans la règle "au-plus-un-niveau-de-différence"



Raffinement courbe, $h = 1/8$, $\varepsilon = 0.025$, avec la règle "au-plus-un-niveau-de-différence"

	Pas du maillage $h = 1/4$			Pas du maillage $h = 1/8$					
				$\varepsilon = 0.05$			$\varepsilon = 0.025$		
n	vp_1	vp_2	vp_3	vp_1	vp_2	vp_3	vp_1	vp_2	vp_3
1	0.458587	0.462485	0.482939	0.453283	0.462273	0.470363	0.452402	0.455136	0.470363
2	0.425736	0.456570	0.482277	0.45046	0.456295	0.470363	0.423367	0.443878	0.470362
3	0.434442	0.450784	0.48214	0.445984	0.451429	0.470362	0.43134	0.441875	0.470362
4	0.427107	0.446409	0.482022	0.441615	0.447478	0.470362	0.424715	0.4381	0.470362
5	0.418022	0.443254	0.481940	0.438249	0.444401	0.47036	0.41627	0.434157	0.470361
6	0.410868	0.440858	0.476321	0.435797	0.441948	0.47035	0.409461	0.432411	0.470361

Raffinement de la courbe avec la règle "au-plus-un-niveau-de-differnce"

Conclusion Les résultats numériques semblent montrer que quelque soit :

- la zone que l'on raffine,
- le pas du maillage,
- la taille de la zone de raffinement,
- l'application ou non de la règle "au-plus-un-niveau-de-différence ;

la constante Inf-Sup est minorée.
On peut donc conjecturer que la stratégie de raffinement CHARMS préserve la stabilité Inf-Sup.

Chapitre 3

Les schémas volumes finis

On retrouvera dans ce chapitre différentes notations et différents résultats empruntés à la thèse de Stella Krell [Kre10].
Les programmes de ce chapitre ont été codés en Scilab en partant d'un programme existant qui résolvait le Laplacien.
Ce que l'on sait sur la constante Inf-Sup a été vérifié pour des schémas de type éléments finis, le but de ce chapitre est de voir ce qu'il en est pour des schémas de types volumes finis.
Nous choisissons ici de prendre $\Omega =]0,1[^2$.

I Le schéma VF4

On sait que ce schéma n'est pas consistant, mais on va tout de même regarder l'évolution de la constante Inf-Sup pour des maillages admissibles.

I.1 Présentation du schéma

1.1.1 Le maillage VF4

Définition 1.1 (Maillage admissible VF4)
Soit un ouvert polygonal borné de $\mathbb{R}^2$.
Un maillage $\mathcal{T}$ admissible de Ω au sens des volumes finis est donné par :

1. *un ensemble $\mathfrak{M}$ d'ouverts polygonaux convexes disjoints 2 à 2, appelés volumes de contrôle $\mathcal{K}$, tels que $\overline{\cup_{\mathcal{K}}\mathcal{K}} = \bar{\Omega}$. On note $\partial\mathfrak{M}$ l'ensemble des bords de volume de contrôle de $\mathfrak{M}$ inclus dans $\partial\Omega$ qui sont considérés comme des volumes de contrôle dégénérés.*
2. *Pour tous les volumes de contrôle voisins $\mathcal{K}$ et $\mathcal{L}$, on suppose que $\partial\mathcal{K} \cap \partial\mathcal{L}$ est un côté de chaque volume de contrôle, et il est appelé arête σ du maillage $\mathcal{T}$, notée $\sigma = \mathcal{K}|\mathcal{L}$. On note $\mathcal{E}$ l'ensemble de ces arêtes.*
3. *A chaque volume de contrôle $\mathcal{K} \in \mathfrak{M}$, on associe un point $x_{\mathcal{K}} \in \mathcal{K}$, appelé centre du volume de contrôle, et à chaque volume de contrôle du bord $\mathcal{K} \in \partial\mathfrak{M}$, on associe un point $x_{\mathcal{K}}$ tel que $x_{\mathcal{K}}$ soit un point de la maille du bord $\mathcal{K}$. On impose pour $\sigma = \mathcal{K}|\mathcal{L}$ que la ligne joignant $x_{\mathcal{K}}$ à $x_{\mathcal{L}}$ est orthogonale à l'arête $\sigma = \mathcal{K}|\mathcal{L}$ et pour $\sigma \in \partial\Omega \cap \bar{\mathcal{K}}$ que la ligne joignant $x_{\mathcal{K}}$ à $x_{\mathcal{L}}$ est orthogonale à l'arête σ.*

Notations :
- $size(\mathcal{T})$ est le pas du maillage,
- $|\mathcal{K}|$ est le volume de $\mathcal{K}$
- $\mathcal{E}_{int}$ est l'ensemble de toutes les arêtes intérieure de $\mathcal{T}$,
- $\mathcal{E}_{ext}$ est l'ensemble de toutes les arêtes extérieures de $\mathcal{T}$,
- $\mathcal{E}_{\mathcal{K}}$ est l'ensemble des arêtes du volume de contrôle $\mathcal{K}$,
- $d_{\mathcal{K}\mathcal{L}}$ est la distance de $x_{\mathcal{K}}$ à $x_{\mathcal{L}}$,
- $\vec{n}_{\mathcal{K}\sigma} = \vec{n}_{\mathcal{K}\mathcal{L}}$ est la normale sortante à $\mathcal{K}$.

Pour le problème de Stokes, on travaille avec un maillage admissible et on choisit comme inconnues :
- pour la vitesse, une inconnue (vectorielle) par volume de contrôle notée $u_{\mathcal{K}}$ et qui est l'approximation de la solution $u(x_{\mathcal{K}})$ au centre du volume de contrôle $\mathcal{K}$, donc $u^{\mathcal{T}} = (u_{\mathcal{K}})_{\mathcal{K}\in\mathcal{T}}$;
- pour la pression, une inconnue par arête σ notée p_σ donc : $p^{\mathcal{E}} = (p_\sigma)_{\sigma\in\mathcal{E}}$.

1.1.2 Les opérateurs discrets

Intégrons sur un volume de contrôle $\mathcal{K}$ l'équation $\mathrm{div}(-\nabla u + pId) = f$, on obtient :

$$|\mathcal{K}| f_{\mathcal{K}} := \int_{\mathcal{K}} f = \int_{\mathcal{K}} \mathrm{div}(-\nabla u + pId) = \sum_{\sigma \in \mathcal{E}_{\mathcal{K}}} \int_{\sigma} (-\nabla u + pId)\vec{n}_{\mathcal{K}\sigma} = \sum_{\sigma \in \mathcal{E}_{\mathcal{K}}} \left(\int_{\sigma} -\nabla u \vec{n}_{\mathcal{K}\sigma} + \int_{\sigma} p \cdot \vec{n}_{\mathcal{K}\sigma} \right).$$

On pose alors $\bar{F}_{\mathcal{K},\sigma} = \int_{\sigma} -\nabla u \vec{n}_{\mathcal{K}\sigma}$, et deux cas se présentent :

- Si $\sigma \in \mathcal{E}_{int}$, $\sigma = \mathcal{K}|\mathcal{L}$, alors $x_{\mathcal{L}} - x_{\mathcal{K}} = d_{\mathcal{K}\mathcal{L}} \vec{n}_{\mathcal{K}\mathcal{L}}$ et donc si $x \in \sigma$, $(\nabla u(x)) \cdot \vec{n}_{\mathcal{K}\mathcal{L}} = \frac{u(x_{\mathcal{L}}) - u(x_{\mathcal{K}})}{d_{\mathcal{K}\mathcal{L}}} + O(size(\mathcal{T}))$.
  D'où, $\bar{F}_{\mathcal{K},\sigma} = -|\sigma| \frac{u(x_{\mathcal{L}}) - u(x_{\mathcal{K}})}{d_{\mathcal{K}\mathcal{L}}} + O(size(\mathcal{T})^2)$ et on pose alors comme flux approché :

  $$F_{\mathcal{K},\sigma}(u^{\mathcal{T}}) = -|\sigma| \frac{u_{\mathcal{L}} - u_{\mathcal{K}}}{d_{\mathcal{K}\mathcal{L}}}.$$

  On remarque alors que l'on a la conservativité du schéma : $F_{\mathcal{K},\sigma}(u^{\mathcal{T}}) = -F_{\mathcal{L},\sigma}(u^{\mathcal{T}})$.
- Si $\sigma \in \mathcal{E}_{ext}$, alors $x_{\sigma} - x_{\mathcal{K}} = d_{\mathcal{K}\sigma} \vec{n}_{\mathcal{K}\sigma}$ et donc $(\nabla u(x)) \cdot \vec{n}_{\mathcal{K}\sigma} \sim \frac{u(x_{\sigma}) - u(x_{\mathcal{K}})}{d_{\mathcal{K}\sigma}} = \frac{0 - u(x_{\mathcal{K}})}{d_{\mathcal{K}\sigma}}$.
  D'où, $\bar{F}_{\mathcal{K},\sigma} = -|\sigma| \frac{-u(x_{\mathcal{K}})}{d_{\mathcal{K}\sigma}} + O(size(\mathcal{T})^2)$ et on pose alors comme flux approché :

  $$F_{\mathcal{K},\sigma}(u^{\mathcal{T}}) = -|\sigma| \frac{-u_{\mathcal{K}}}{d_{\mathcal{K}\sigma}}.$$

L'inconnue en pression étant localisée aux arêtes, le terme $\sum_{\sigma \in \mathcal{E}_{\mathcal{K}}} \int_{\sigma} p \cdot \vec{n}_{\mathcal{K}\sigma}$ nous donne comme approximation du gradient discret :

$$|\mathcal{K}|(\text{"}\nabla p\text{"})_{\mathcal{K}} = \sum_{\sigma \in \mathcal{E}_{\mathcal{K}}} |\sigma| p_{\sigma} \vec{n}_{\mathcal{K}\sigma}.$$

Pour obtenir la formule de la divergence discrète, on se sert du fait que dans le cas continu, on a : $\int_{\Omega} u \cdot \nabla p = -\int_{\Omega} p \, \mathrm{div}\, u$, alors dans le cas discret on doit avoir :

$$\sum_{\mathcal{K} \in \mathcal{T}} |\mathcal{K}| u_{\mathcal{K}} (\text{"}\nabla p\text{"})_{\mathcal{K}} = -\sum_{\sigma \in \mathcal{E}} |\sigma| p_{\sigma} (\text{" div } u\text{"})_{\sigma}. \tag{I.1}$$

Ainsi,

$$\sum_{\mathcal{K} \in \mathcal{T}} |\mathcal{K}| u_{\mathcal{K}} (\text{"}\nabla p\text{"})_{\mathcal{K}} = \sum_{\mathcal{K} \in \mathcal{T}} \sum_{\sigma \in \mathcal{E}_{\mathcal{K}}} |\sigma| p_{\sigma} u_{\mathcal{K}} \cdot \vec{n}_{\mathcal{K}\sigma} = \sum_{\sigma \in \mathcal{E}} |\sigma| p_{\sigma} (u_{\mathcal{K}} \cdot \vec{n}_{\mathcal{K}\mathcal{L}} + u_{\mathcal{L}} \cdot \vec{n}_{\mathcal{L}\mathcal{K}}) = -\sum_{\sigma \in \mathcal{E}} |\sigma| p_{\sigma} (u_{\mathcal{L}} - u_{\mathcal{K}}) \cdot \vec{n}_{\mathcal{K}\mathcal{L}}, \tag{I.2}$$

et donc $(\text{" div } u\text{"})_{\sigma} = \frac{2}{d_{\mathcal{K}\mathcal{L}}} (u_{\mathcal{L}} - v_{\mathcal{K}}) \cdot \vec{n}_{\mathcal{K}\mathcal{L}}$ où $u_{\mathcal{L}} = 0$ si $\sigma_{\mathcal{K}\mathcal{L}} \in \partial\Omega$.

On peut alors remarquer avec les définitions que nous avons données du gradient discret $(\text{"}\nabla p\text{"})_{\mathcal{K}}$ et de la divergence discrète $(\text{" div } u\text{"})_{\sigma}$ que bien que le gradient soit consistant la divergence discrète ne l'est pas car elle ne tient compte du champ de vitesse que dans une seule direction.

Le problème discret consiste donc à chercher $u^{\mathcal{T}} = (u_{\mathcal{K}})_{\mathcal{K} \in \mathcal{T}}$ et $p^{\mathcal{E}} = (p_{\sigma})_{\sigma \in \mathcal{E}}$ tels que :

$$\begin{cases} \forall \mathcal{K} \in \mathcal{T}, \ \displaystyle\sum_{\sigma \in \mathcal{E}_{\mathcal{K}}} F_{\mathcal{K},\sigma}(u^{\mathcal{T}}) + |\mathcal{K}| (\text{"}\nabla p\text{"})_{\mathcal{K}} = |\mathcal{K}| f_{\mathcal{K}}, \\ (\text{" div } v\text{"}) = 0. \end{cases}$$

On peut également s'intéresser au schéma présenté dans [EGH] où l'on choisit les inconnues en vitesses de la même manière mais la pression est, quand à elle, discrétisée avec l'élément fini $\mathbb{P}_1$ (cf section suivante).

Définition 1.2 (Produits scalaires discrets)
Pour l'espace des pressions : $(p^{\mathcal{E}}, q^{\mathcal{E}}) = \displaystyle\sum_{\sigma \in \varepsilon_{\mathcal{K}}} \frac{|\sigma| d_{\mathcal{K}\mathcal{L}}}{2} p_{\sigma} q_{\sigma}$.
Pour la vitesse : $(u^{\mathcal{T}}, v^{\mathcal{T}}) = \displaystyle\sum_{\mathcal{K} \in \mathcal{T}} |\mathcal{K}| u_{\mathcal{K}} \cdot v_{\mathcal{K}}$.

I.2 Construction des matrices

Pour construire les différentes matrices, on ne va pas parcourir les volumes de contrôle mais les arêtes du maillage.
Rappelons que sous forme matricielle le problème de Stokes s'écrit :

$$\begin{cases} RU + B^T P = F \\ BU = 0 \end{cases}$$

L'espace de la vitesse étant de dimension 2, on choisit de ranger le vecteur inconnu U comme tel : $U = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$. Il faudra également considérer les deux composantes de la vitesse lorsque l'on définit les matrices.

La matrice de rigidité :
D'après le problème discret que nous venons de voir, la matrice de rigidité R contient le flux discret $\bar{F}_{\mathcal{K},\sigma}$. R est alors $R \in \mathcal{M}_{2n_c}$ (pour prendre en compte les deux composantes de la vitesse) où n_c est le nombre de volumes de contrôle du maillage.
On a alors $R = \begin{pmatrix} \bar{R} & 0 \\ 0 & \bar{R} \end{pmatrix}$, où la matrice $\bar{R}$ est construite comme suit :
On parcourt les arêtes et lorsque l'on est sur l'arête $\sigma = \sigma_{\mathcal{KL}}$:

- sur la ligne correspondant au volume de contrôle $\mathcal{K}$, on rajoute le terme :
  - $+\frac{|\sigma|}{d_{\mathcal{KL}}}$ à la colonne correspondant au volume de contrôle $\mathcal{K}$,
  - $-\frac{|\sigma|}{d_{\mathcal{KL}}}$ à la colonne correspondant au volume de contrôle $\mathcal{L}$ si $\mathcal{K}$ n'est pas sur le bord du maillage ;
- si $\mathcal{K}$ est un volume de contrôle ne se trouvant pas sur le bord du maillage, sur la ligne correspondant au volume de contrôle $\mathcal{L}$, on rajoute le terme :
  - $-\frac{|\sigma|}{d_{\mathcal{KL}}}$ à la colonne correspondant au volume de contrôle $\mathcal{K}$,
  - $+\frac{|\sigma|}{d_{\mathcal{KL}}}$ à la colonne correspondant au volume de contrôle $\mathcal{L}$.

La matrice de masse :
La matrice de masse M est définie par $(MP, Q) = (p^{\mathcal{E}}, q^{\mathcal{E}})$, donc d'après la définition du produit scalaire discret, $M \in \mathcal{M}_{n_a}$ où n_a est le nombre d'arêtes du maillage et est telle que :

$$M = diag\left(\frac{|\sigma| d_{\mathcal{KL}}}{2}\right).$$

La matrice de divergence :
La matrice de divergence B est telle que $(BV, P) = b(v^{\mathcal{T}}, p^{\mathcal{E}})$, donc d'après (I.2) :

$$(BV, P) = \sum_{\sigma \in \mathcal{E}} |\sigma| p_\sigma \left[(v^1_{\mathcal{K}} - v^1_{\mathcal{L}}) \cdot \vec{n}^1_{\mathcal{KL}} + (v^2_{\mathcal{K}} - v^2_{\mathcal{L}}) \cdot \vec{n}^2_{\mathcal{KL}} \right] \text{ où } v_{\mathcal{L}} = 0 \text{ si } \sigma \in \partial\Omega.$$

Ainsi, $B \in \mathcal{M}_{n_a, 2n_c}$ et on la construit toujours en parcourant les arêtes.
Lorsque l'on est sur l'arête $\sigma = \sigma_{\mathcal{KL}}$:

- on rajoute le terme $+|\sigma|\vec{n}^1_{\mathcal{KL}}$ sur la colonne correspondant au volume de contrôle $\mathcal{K}$,
- on rajoute le terme $+|\sigma|\vec{n}^2_{\mathcal{KL}}$ sur la colonne correspondant au volume de contrôle $\mathcal{K}$ + nombre de volumes de contrôle,
- Si $\sigma \notin \partial\Omega$, on ajoute le terme :
  - $-|\sigma|\vec{n}^1_{\mathcal{KL}}$ sur la colonne correspondant au volume de contrôle $\mathcal{L}$,
  - $-|\sigma|\vec{n}^2_{\mathcal{KL}}$ sur la colonne correspondant au volume de contrôle $\mathcal{L}$ + nombre de volumes de contrôle.

On peut alors vérifier que B^T est bien la matrice du gradient discret et que $B^T \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = 0$.

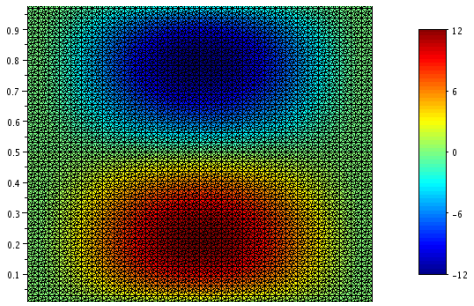
I.3 Résultats numériques

Vu que ce schéma n'est pas consistant on va seulement regarder les résultats dans le cas d'un maillage triangle admissible.

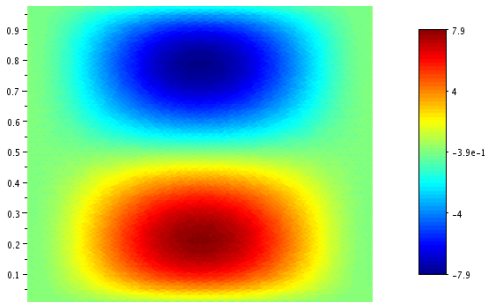
1.3.1 Non convergence du schéma

On vérifie tout d'abord que le schéma ne converge pas vers la bonne solution, sur l'exemple donnant une solution polynomiale.
On va alors tracer :

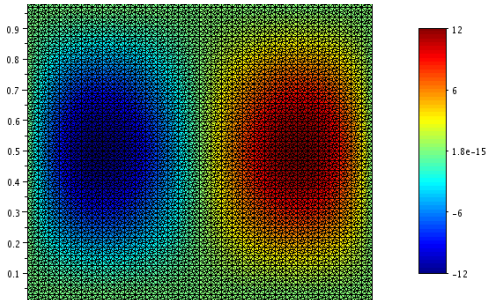
- la solution exacte et la solution approchée au pas $h = 1.5625.10^{-2}$ pour la vitesse et pour la pression,
- la courbe d'erreur en norme infinie, L^2 et H^1 pour la vitesse et en norme infinie, L^2 pour la pression.



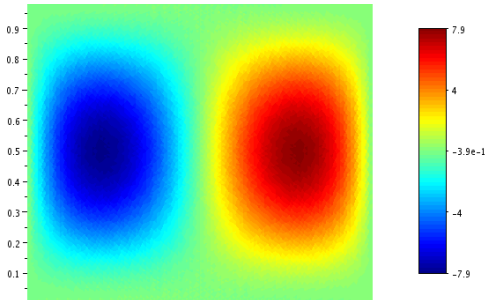
Première composante de la vitesse exacte



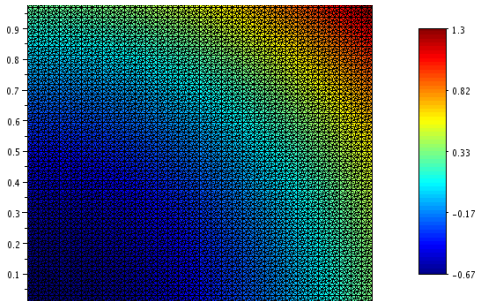
Première composante de la vitesse approchée



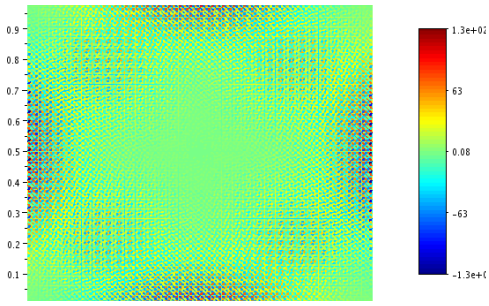
Deuxième composante de la vitesse exacte



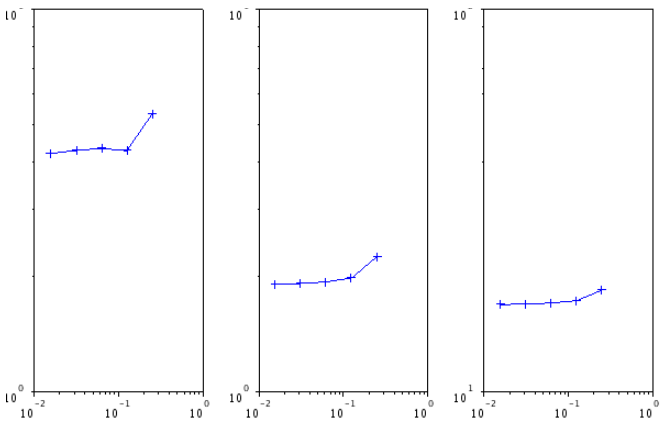
Deuxième composante de la vitesse approchée



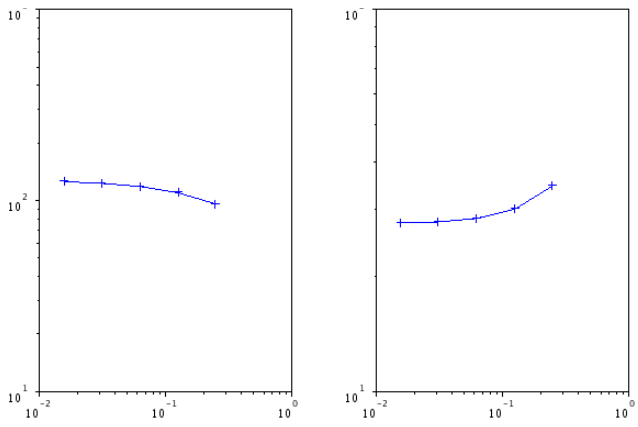
Pression exacte



Pression approchée



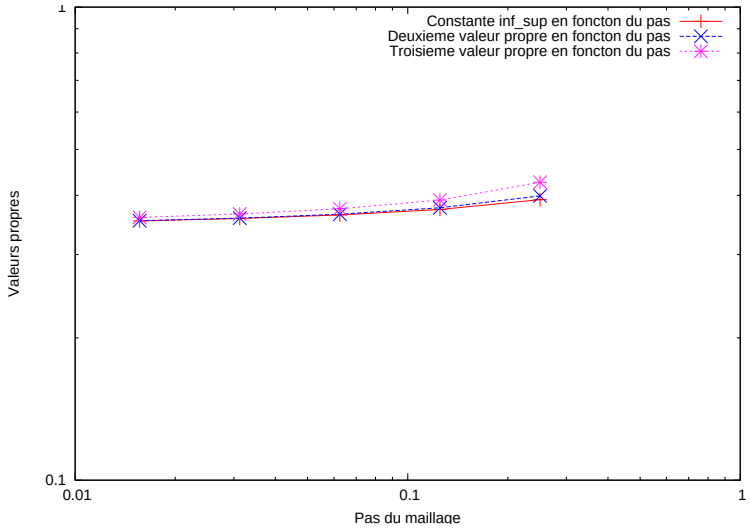
Erreur en vitesse ($L^\infty/L^2/H^1$)



Erreur en pression (L^∞/L^2)

1.3.2 Les problèmes de valeurs propres

On regarde alors les différentes valeurs de la constante Inf-Sup qui semblent malgré tout être uniformément minorée.



II Le schéma VF4 version EGH

II.1 Présentation du schéma

Comme dans la section précédente (dont on gardera les notations), on se place dans le cas d'un maillage admissible. La différence essentielle avec le schéma VF4 se trouve dans le choix des inconnues en pressions. En effet, les inconnues en vitesses sont les mêmes que pour le schéma VF4, c'est à dire une inconnue par maille et qui se trouve au centre de celle-ci. Par contre, au lieu de choisir la pression constante par arête on choisit de la discrétiser avec l'élément fini $\mathbb{P}_1$. Les inconnues se trouvent donc cette fois aux sommets du maillage.

Ainsi, on définit de nouveaux opérateurs discrets $(\text{"}\nabla p\text{"})_\mathcal{K}$ et $(\text{" div } u\text{"})_\sigma$.
On a en effet,

$$|\mathcal{K}|(\text{"}\nabla p\text{"})_\mathcal{K} \approx \int_\mathcal{K} \nabla p = \int_\mathcal{K} \text{div}(pId) = \sum_{\sigma\in\mathcal{E}_k} \int_\sigma p\vec{n} = \sum_{\sigma\in\mathcal{E}_k} |\sigma| \frac{1}{2}(p_{s_1} + p_{s_2})\vec{n},$$

où p_{s_1} et p_{s_2} sont les deux sommets de σ.
Puis,

$$\begin{aligned}\sum_{\mathcal{K}\in\mathcal{T}} |\mathcal{K}| u_\mathcal{K} (\text{"}\nabla p\text{"})_\mathcal{K} &= \sum_{\mathcal{K}\in\mathcal{T}} \sum_{\sigma\in\mathcal{E}_\mathcal{K}} |\sigma| \frac{1}{2}(p_{s_1} + p_{s_2}) u_\mathcal{K} \cdot \vec{n}_{\mathcal{K}\sigma} = \sum_{\sigma\in\mathcal{E}} |\sigma| \frac{1}{2}(p_{s_1} + p_{s_2})(u_\mathcal{K} \cdot \vec{n}_{\mathcal{K}\mathcal{L}} + u_\mathcal{L} \cdot \vec{n}_{\mathcal{L}\mathcal{K}}) \\ &= -\sum_{\sigma\in\mathcal{E}} |\sigma| \frac{1}{2}(p_{s_1} + p_{s_2})(u_\mathcal{L} - u_\mathcal{K}) \cdot \vec{n}_{\mathcal{K}\mathcal{L}}.\end{aligned}$$

On a alors le résultat suivant :

Théorème 2.1 ([BEH04])
Soit $\mathcal{T}$ un maillage admissible de Ω formé de triangle équilatéraux.
Alors, si on suppose qu'il existe une solution $(u, p) \in (H^2(\Omega))^2 \times H^1(\Omega)$ solution du problème de stokes :

$$\begin{cases} -\Delta u + \nabla p = f \\ \text{div } u = 0 \end{cases} \quad \text{avec } u = 0 \text{ sur } \partial\Omega.$$

On a l'estimation d'erreur suivante :

$$||e|| \leq Ch \left(||u||_{(H^2(\Omega))^2} + ||p||_{H^1(\Omega)} \right).$$

Cependant, ce résultat n'est vrai que sur des maillages formés de triangles équilatéraux sinon on a problème de consistance de l'opérateur divergence discrète et le schéma n'est pas consistant.

II.2 Construction des matrices

On commence par remarquer que la matrice de rigidité est identique puisque l'on a gardé les mêmes inconnues en vitesse.

La matrice de divergence :
Cette fois la matrice de divergence est telle que $B \in \mathcal{M}_{2n_c,n_s}$ (avec n_s le nombre de sommets) et :

$$(BV,P) = \sum_{\sigma\in\mathcal{E}} \frac{|\sigma|}{2}(p_{s_1}+p_{s_2})\left[(v^1_{\mathcal{K}}-v^1_{\mathcal{L}})\cdot\vec{n}^1_{\mathcal{K}\mathcal{L}} + (v^2_{\mathcal{K}}-v^2_{\mathcal{L}})\cdot\vec{n}^2_{\mathcal{K}\mathcal{L}}\right] \text{ où } v_{\mathcal{L}}=0 \text{ si } \sigma\in\partial\Omega.$$

Pour la construire, on parcourt là encore les arêtes et lorsque l'on est sur l'arête $\sigma = \sigma_{\mathcal{K}\mathcal{L}}$:
- on ajoute le terme $+\frac{|\sigma|}{2}\vec{n}^1_{\mathcal{K}\mathcal{L}}$ à la coordonnée correspondant au sommet s_1 et au volume de contrôle $\mathcal{K}$,
- on ajoute le terme $+\frac{|\sigma|}{2}\vec{n}^2_{\mathcal{K}\mathcal{L}}$ à la coordonnée correspondant au sommet s_1 et au volume de contrôle $\mathcal{K}$ + nombre de volumes de contrôle,
- on ajoute le terme $+\frac{|\sigma|}{2}\vec{n}^1_{\mathcal{K}\mathcal{L}}$ à la coordonnée correspondant au sommet s_2 et au volume de contrôle $\mathcal{K}$,
- on ajoute le terme $+\frac{|\sigma|}{2}\vec{n}^2_{\mathcal{K}\mathcal{L}}$ à la coordonnée correspondant au sommet s_2 et au volume de contrôle $\mathcal{K}$ + nombre de volumes de contrôle,
- Si $\sigma \notin \partial\Omega$, on ajoute le terme :
  - $-\frac{|\sigma|}{2}\vec{n}^1_{\mathcal{K}\mathcal{L}}$ à la coordonnée correspondant au sommet s_1 et volume de contrôle $\mathcal{L}$,
  - $-\frac{|\sigma|}{2}\vec{n}^2_{\mathcal{K}\mathcal{L}}$ à la coordonnée correspondant au sommet s_1 et au volume de contrôle $\mathcal{L}$ + nombre de volumes de contrôle,
  - $-\frac{|\sigma|}{2}\vec{n}^1_{\mathcal{K}\mathcal{L}}$ à la coordonnée correspondant au sommet s_2 et au volume de contrôle $\mathcal{L}$,
  - $-\frac{|\sigma|}{2}\vec{n}^2_{\mathcal{K}\mathcal{L}}$ à la coordonnée correspondant au sommet s_2 et au volume de contrôle $\mathcal{L}$ + nombre de volumes de contrôle.

La matrice de masse :
Rappelons que $M = \left(\int_\Omega \psi_i\psi_j\right)_{1\le i,j\le n_s} = \left(\sum_{\mathcal{K}\in\mathcal{T}}\int_{\mathcal{K}}\psi_i\psi_j\right)_{1\le i,j\le n_s}$ et on a :

$$\int_{\mathcal{K}}\psi_i(x)\psi_j(x)dx = \int_{\hat{\mathcal{K}}}\psi_i(T_{\mathcal{K}}\hat{x})\psi_j(T_{\mathcal{K}}\hat{x})|J_{\mathcal{K}}|d\hat{x} = \int_{\hat{\mathcal{K}}}\theta_i(\hat{x})\theta_j(\hat{x})|J_{\mathcal{K}}|d\hat{x} = 2|\mathcal{K}|\int_{\hat{\mathcal{K}}}\theta_i(\hat{x})\theta_j(\hat{x})d\hat{x},$$

où :
- $\hat{\mathcal{K}}$ représente l'élément de référence c'est à dire le simplexe unité de sommets $a_1=(0,0)$, $a_2=(0,1)$ et $a_3=(1,0)$;
- $\forall i=1,2,3$, θ_i sont les fonctions de forme locale de l'espace d'approximation $\mathbb{P}_1$.
  Sachant que $\forall i,j=1,2,3$, $\theta_i(a_j)=\delta_{i,j}$, on a :
  - $\theta_1(x_1,x_2) = 1-x_1-x_2$,
  - $\theta_2(x_1,x_2) = x_1$,
  - $\theta_3(x_1,x_2) = x_2$;
- $T_{\mathcal{K}}$ est la transformation affine qui transforme $\hat{\mathcal{K}}$ en $\mathcal{K}$;
- $|J_{\mathcal{K}}|$ est le jacobien de la transformation.

On calcule alors :

$$\int_{\hat{\mathcal{K}}}\theta_i(\hat{x})\theta_j(\hat{x}) = \frac{1}{24} \text{ si } i\neq j$$
$$\int_{\hat{\mathcal{K}}}\theta_i(\hat{x})\theta_i(\hat{x}) = \frac{1}{12}$$

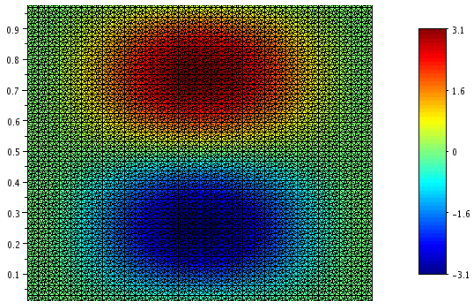
Ainsi, pour construire la matrice de masse, on parcourt cette fois le maillage par maille et lorsque l'on est sur la maille $\mathcal{K}$ de sommets s_1, s_2 et s_3, on ajoute (pour $i,j=1,2,3$) :
- $2|\mathcal{K}|\int_{\hat{\mathcal{K}}}\theta_i(\hat{x})\theta_j(\hat{x}) = \frac{1}{12}|\mathcal{K}|$ à la coordonnée correspondant aux sommets s_i et s_j si $i\neq j$,
- $2|\mathcal{K}|\int_{\hat{\mathcal{K}}}\theta_i(\hat{x})\theta_i(\hat{x}) = \frac{1}{6}|\mathcal{K}|$ à la coordonnée correspondant aux sommets s_i et s_i.

II.3 Résultats numériques

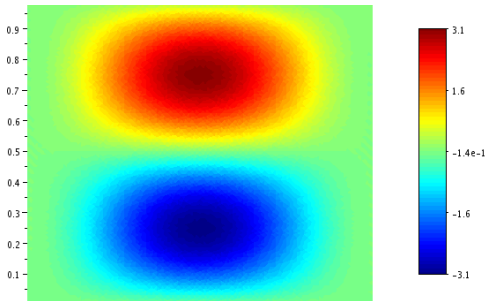
On regarde les résultats de ce schéma sur le maillage triangles admissibles.

2.3.1 Convergence du schéma

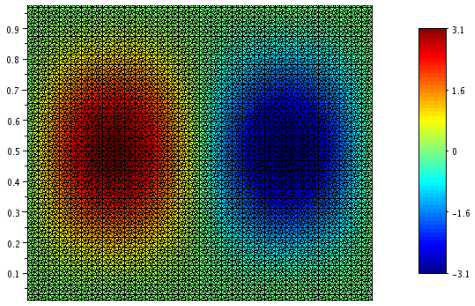
On va alors tracer :
- la solution exacte et la solution approchée au pas $h = 1.5625.10^{-2}$ pour la vitesse et pour la pression,
- la courbe d'erreur en norme infinie, L^2 et H^1 pour la vitesse et en norme infinie, L^2 pour la pression pour le produit de sinus et cosinus.



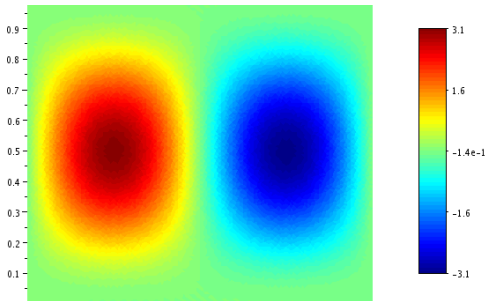
Première composante de la vitesse exacte



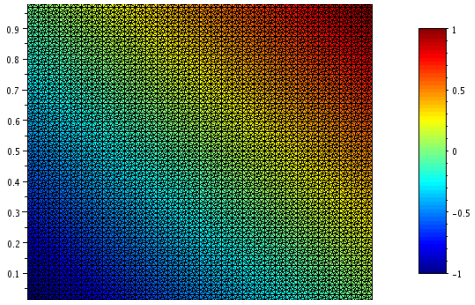
Première composante de la vitesse approchée



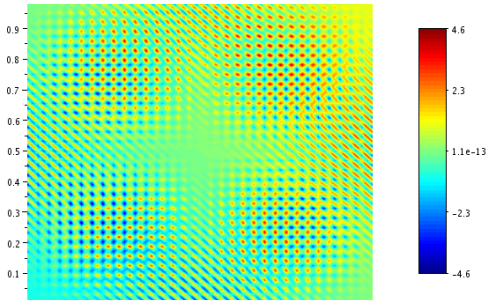
Deuxième composante de la vitesse exacte



Deuxième composante de la vitesse approchée

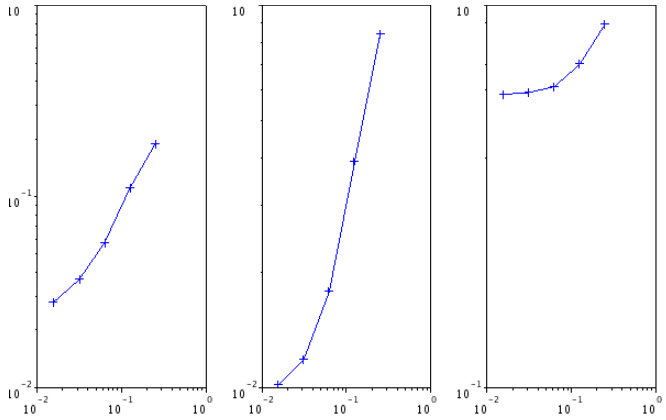


Pression exacte

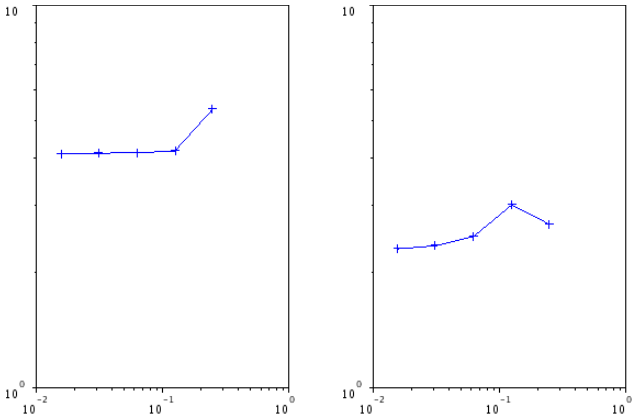


Pression approchée

Le maillage considéré ici n'étant pas formé de triangles équilatéraux, il n'est pas étonnant de voir que le schéma ne converge pas.



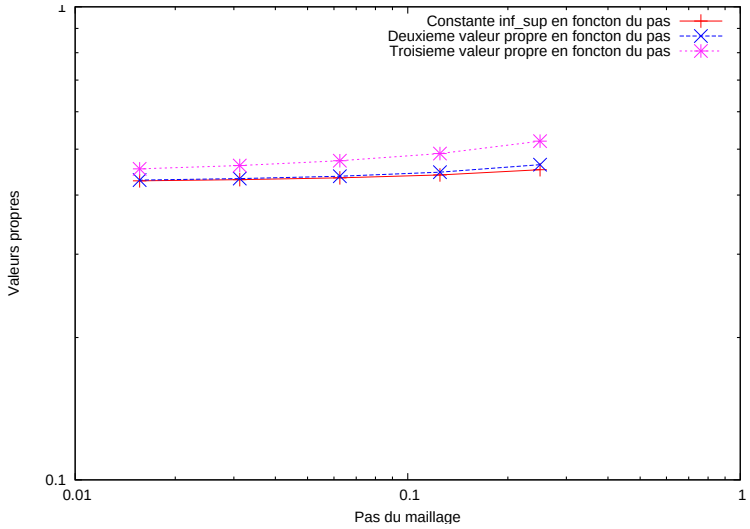
Erreur en vitesse (L^∞, L^2, H^1)



Erreur en pression (L^∞, L^2)

2.3.2 Les problèmes de valeurs propres

On regarde alors les différentes valeurs de la constante Inf-Sup qui semblent malgré tout être uniformément minorée. En conclusion, le schéma est Inf-Sup stable mais pas consistant sur un maillage quelconque.



Conclusion :
Le schéma semble être Inf-Sup stable bien qu'il ne soit pas consistant sur un maillage quelconque.

III Le schéma DDFV

On introduit ce schéma car on cherche à regarder l'évolution de la constante Inf-Sup sur un schéma consistant et stable.

III.1 Présentation du schéma

3.1.1 Le maillage DDFV

Définition 3.1
On appelle maillage DDFV, $\mathcal{T}$ un maillage constitué d'un maillage primal $\mathfrak{M} \cup \partial\mathfrak{M}$ et d'un maillage dual $\mathfrak{M}^* \cup \partial\mathfrak{M}^*$.

- *Le maillage $\mathfrak{M}$ est un ensemble de polygones disjoints appelés mailles primales $\mathcal{K} \in \Omega$ tels que $\cup\bar{\mathcal{K}} = \bar{\Omega}$.*
  On note $\partial\mathfrak{M}$ l'ensemble des bords des mailles primales de $\mathfrak{M}$ inclus dans $\partial\Omega$ qui sont considérés comme des mailles primales dégénérées. A chaque maille primale $\mathcal{K} \in \mathfrak{M}$, on associe un point $x_\mathcal{K}$ appelé centre du volume de contrôle $\mathcal{K}$.
  On a ainsi une famille de points $X = \{x_\mathcal{K}, \mathcal{K} \in \mathfrak{M}\}$, appelés noeuds du maillage primal.

On remarque que l'on a pas mis d'inconnue sur le bord $\partial\mathfrak{M}$ car on choisit d'imposer une condition de Dirichlet forte et on les impose donc nulle.

- *Les noeuds du maillage dual sont les sommets du maillage primal. L'ensemble de ces noeuds X^* se décompose en $X^* = X^*_{int} \cup X^*_{ext}$ où $X^*_{int} \cap \partial\Omega = \emptyset$ et $X^*_{ext} \subset \partial\Omega$. Les ensembles $\mathfrak{M}^* \cup \partial\mathfrak{M}^*$ sont deux familles de mailles duales. A chaque point $x_{\mathcal{K}^*} \in X^*_{int}$ (respectivement $x_{\mathcal{K}^*} \in X^*_{ext}$), on associe un polygone $\mathcal{K}^*$ où ses sommets sont $\{x_\mathcal{K} \in X$, tel que $x_{\mathcal{K}^*} \in \bar{\mathcal{K}}, \mathcal{K} \in \mathfrak{M}\}$ (respectivement $\{x_{\mathcal{K}^*}\} \cup \{x_\mathcal{K} \in X$, tel que $x_{\mathcal{K}^*} \in \bar{\mathcal{K}}, \mathcal{K} \in (\mathfrak{M} \cup \partial\mathfrak{M})\}$). On suppose que l'intérieur des mailles duales sont tous disjointes.*
- *Pour toutes les mailles primales voisines $\mathcal{K}$ et $\mathcal{L}$, on suppose que $\partial\mathcal{K} \cap \partial\mathcal{L}$ est un segment que l'on appelle une arête σ du maillage primal $\mathfrak{M}$, notée $\sigma = \mathcal{K}|\mathcal{L}$. On note ε l'ensemble de ces arêtes.*
- *On note de même $\sigma^* = \mathcal{K}^*|\mathcal{L}^*$ et $\mathcal{E}^*$ pour le maillage dual $\mathfrak{M}^* \cup \partial\mathfrak{M}^*$.*

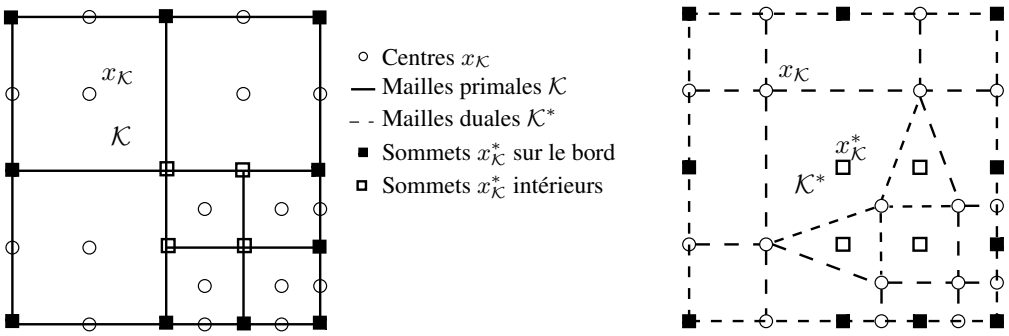


FIG. 3.1: Un maillage primal $\mathfrak{M} \cup \partial\mathfrak{M}$(à gauche) et un maillage dual $\mathfrak{M}^* \cup \partial\mathfrak{M}^*$(à droite)

Définition 3.2 (Les mailles diamants)
Grâce aux mailles primales et duales, on définit les diamants $\mathcal{D}$ du maillage, les quadrilatères tels que leurs diagonales principales soient une arête primale $\sigma = \mathcal{K}|\mathcal{L} = (x_{\mathcal{K}^*}, x_{\mathcal{L}^*})$ et une arête duale $\sigma^* = \mathcal{K}^*|\mathcal{L}^* = (x_\mathcal{K}, x_\mathcal{L})$, d'où la notation $\mathcal{D} = \mathcal{D}_{\sigma,\sigma^*}$.
On remarque que les diamants sont une réunion de deux triangles disjoints et ne sont pas forcément convexes. En outre, si $\sigma \in \mathcal{E} \cap \partial\bar{\Omega}$, le quadrilatère $\mathcal{D}_{\sigma,\sigma^*}$ est dégénéré, c'est un triangle. L'ensemble des diamants est noté $\mathfrak{D}$ et on a $\bar{\Omega} = \cup_{\mathcal{D}\in\mathfrak{D}}\bar{\mathcal{D}}$. On a également que les intérieurs des diamants sont disjoints.

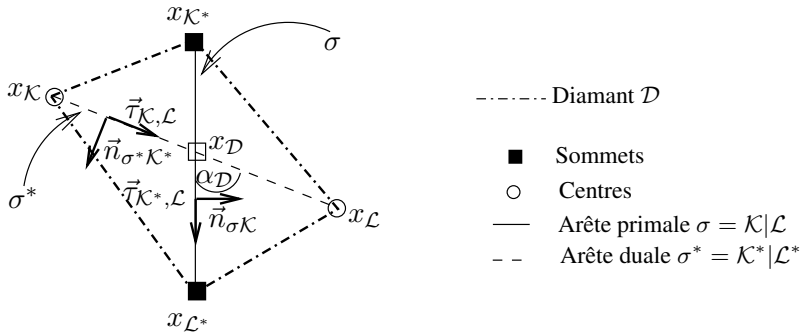


FIG. 3.2: Un diamant $\mathcal{D} = \mathcal{D}_{\sigma,\sigma^*}$

Notations :
- $|\mathcal{K}|$ (respectievement $|\mathcal{K}^*|$) est la mesure du volume de contrôle $\mathcal{K}$ (respectievement $\mathcal{K}^*$),
- $|\sigma|$ (respectievement $|\sigma^*|$) est la mesure de l'arête σ (respectievement de σ^*),
- $|\mathcal{D}|$ est la mesure du diamant $\mathcal{D}$,
- $\alpha_\mathcal{D}$ est l'angle entre σ et σ^*,
- $\vec{n}_{\sigma,\mathcal{K}}$ (respectievement $\vec{n}_{\sigma^*,\mathcal{K}^*}$) est la normale unitaire à σ (respectivement σ^*) sortante de $\mathcal{K}$ (respectievement $\mathcal{K}^*$),
- $\mathfrak{D}_\mathcal{K}$ (respectievement $\mathfrak{D}_{\mathcal{K}^*}$) est l'ensemble des diamants intersectant la maille $\mathcal{K}$ (respectievement $\mathcal{K}^*$),
- $d_{\mathcal{K}^*,\mathcal{L}}$ est la distance entre $x_{\mathcal{K}^*}$ et $x_\mathcal{L}$,
- $d_\mathcal{D}$ est le diamètre du diamant $\mathcal{D}_{\sigma,\sigma^*}$,
- $\mathcal{E}_\mathcal{D}$ est l'ensemble des côtés intérieurs de $\mathcal{D}$.

Dans un schéma DDFV, on a de l'ordre de deux fois plus d'inconnues que dans la méthode classique VF4 et aussi de l'ordre de deux fois plus d'équations, mais cela permet de définir une approximation complète du gradient et pas uniquement dans la direction normale. On peut alors ainsi lever la condition d'orthogonalité du maillage.

– pour la vitesse, on a une inconnue (vectorielle) primale par volume de contrôle notée $u_{\mathcal{K}}$ qui est l'approximation de la solution $u(x_{\mathcal{K}})$ au centre du volume de contrôle $\mathcal{K}$ et une inconnue (vectorielle) duale par maille duale notée $u_{\mathcal{K}^*}$ qui est l'approximation de la solution $u(x_{\mathcal{K}^*})$ au sommet $x_{\mathcal{K}^*}$, on a donc :

$$u^{\mathcal{T}} = ((u_{\mathcal{K}})_{\mathcal{K}\in\mathfrak{M}}, (u_{\mathcal{K}^*})_{\mathcal{K}^*\in(\mathfrak{M}^*\cup\partial\mathfrak{M}^*)}) \in (\mathbb{R}^2)^{\mathcal{T}};$$

– pour la pression, une inconnue par arête σ donc par diamant $\mathcal{D}$ (car on a une bijection entre arêtes et diamants) notée $p_\sigma = p_{\mathcal{D}}$ en supposant celle-ci constante par arête, donc :

$$p^{\mathfrak{D}} = (p_{\mathcal{D}})_{\mathcal{D}\in\mathfrak{D}} \in (\mathbb{R})^{\mathfrak{D}}.$$

3.1.2 Les opérateurs discrets

Définition 3.3 (Gradient discret)
Le gradient discret d'un champ de vecteurs de $(\mathbb{R}^2)^{\mathcal{T}}$ est défini de la manière suivante : $\nabla^{\mathfrak{D}} : u^{\mathcal{T}} \in (\mathbb{R}^2)\mathcal{T} \mapsto (\nabla^{\mathcal{D}} u^{\mathcal{T}})_{\mathcal{D}\in\mathfrak{D}} \in (\mathcal{M}_2(\mathbb{R}))^{\mathfrak{D}}$, avec pour $\mathcal{D} \in \mathfrak{D}$:

$$\begin{aligned}\nabla^{\mathcal{D}} u^{\mathcal{T}} &= \frac{1}{\sin(\alpha_{\mathcal{D}})}\left[\frac{u_{\mathcal{L}} - u_{\mathcal{K}}}{|\sigma^*|}\otimes\vec{n}_{\sigma\mathcal{K}} + \frac{u_{\mathcal{L}^*} - u_{\mathcal{K}^*}}{|\sigma|}\otimes\vec{n}_{\sigma^*\mathcal{K}^*}\right]\\ &= \frac{1}{2|\mathcal{D}|}\left[|\sigma|(u_{\mathcal{L}} - u_{\mathcal{K}})\otimes\vec{n}_{\sigma\mathcal{K}} + |\sigma^*|(u_{\mathcal{L}^*} - u_{\mathcal{K}^*})\otimes\vec{n}_{\sigma^*\mathcal{K}^*}\right],\end{aligned}$$

où $\otimes$ représente le produit tensoriel.

Une définition équivalente est alors que :

$$\begin{cases}\nabla^{\mathcal{D}} u^{\mathcal{T}}.(x_{\mathcal{L}} - x_{\mathcal{K}}) = u_{\mathcal{L}} - u_{\mathcal{K}},\\ \nabla^{\mathcal{D}} u^{\mathcal{T}}.(x_{\mathcal{L}^*} - x_{\mathcal{K}^*}) = u_{\mathcal{L}^*} - u_{\mathcal{K}^*}.\end{cases}$$

Définition 3.4 (Divergence discrète)
L'opérateur de divergence discrète d'un champ de tenseurs discrets de $(\mathcal{M}_2(\mathbb{R}))^{\mathfrak{D}}$ est défini de la manière suivante :

$$\mathrm{div}^{\mathcal{T}} : \xi^{\mathfrak{D}} \in (\mathcal{M}_2(\mathbb{R}))^{\mathfrak{D}} \mapsto \mathrm{div}^{\mathcal{T}}\,\xi^{\mathfrak{D}} \in (\mathbb{R}^2)^{\mathcal{T}}.$$

Soit $\xi^{\mathfrak{D}} = (\xi^{\mathcal{D}})_{\mathcal{D}\in\mathfrak{D}} \in (\mathcal{M}_2(\mathbb{R}))^{\mathfrak{D}}$, on pose :

$$\mathrm{div}^{\mathcal{T}}\,\xi^{\mathfrak{D}} = (\mathrm{div}^{\mathfrak{M}}\,\xi^{\mathfrak{D}}, \mathrm{div}^{\partial\mathfrak{M}}\,\xi^{\mathfrak{D}}, \mathrm{div}^{\mathfrak{M}^*}\,\xi^{\mathfrak{D}}, \mathrm{div}^{\partial\mathfrak{M}^*}\,\xi^{\mathfrak{D}}),$$

où $\mathrm{div}^{\mathfrak{M}}\,\xi^{\mathfrak{D}} = (\mathrm{div}^{\mathcal{K}}\,\xi^{\mathfrak{D}})_{\mathcal{K}\in\mathfrak{M}}$, $\mathrm{div}^{\partial\mathfrak{M}}\,\xi^{\mathfrak{D}} = 0$, $\mathrm{div}^{\mathfrak{M}^*}\,\xi^{\mathfrak{D}} = (\mathrm{div}^{\mathcal{K}^*}\,\xi^{\mathfrak{D}})_{\mathcal{K}^*\in\mathfrak{M}^*}$ *et* $\mathrm{div}^{\partial\mathfrak{M}^*}\,\xi^{\mathfrak{D}} = (Div^{\mathcal{K}^*}\xi^{\mathfrak{D}})_{\mathcal{K}^*\in\partial\mathfrak{M}^*}$ *avec :*

$$\begin{aligned}\mathrm{div}^{\mathcal{K}}\,\xi^{\mathfrak{D}} &= \frac{1}{|\mathcal{K}|}\sum_{\mathcal{D}_{\sigma,\sigma^*}\in\mathfrak{D}_{\mathcal{K}}} |\sigma|\xi^{\mathcal{D}}.\vec{n}_{\sigma\mathcal{K}}, \forall\mathcal{K}\in\mathfrak{M},\\ \mathrm{div}^{\mathcal{K}^*}\,\xi^{\mathfrak{D}} &= \frac{1}{|\mathcal{K}^*|}\sum_{\mathcal{D}_{\sigma,\sigma^*}\in\mathfrak{D}^*_{\mathcal{K}}} |\sigma^*|\xi^{\mathcal{D}}.\vec{n}_{\sigma^*\mathcal{K}^*}, \forall\mathcal{K}^*\in\mathfrak{M}^*,\\ \mathrm{div}^{\mathcal{K}^*}\,\xi^{\mathfrak{D}} &= \frac{1}{|\mathcal{K}^*|}\left(\sum_{\mathcal{D}_{\sigma,\sigma^*}\in\mathfrak{D}^*_{\mathcal{K}}} |\sigma^*|\xi^{\mathcal{D}}.\vec{n}_{\sigma^*\mathcal{K}^*} + \sum_{\mathcal{D}_{\sigma,\sigma^*}\in\mathfrak{D}^*_{\mathcal{K}}\cap\mathfrak{D}_{ext}} d_{\mathcal{K}^*,\mathcal{L}}\xi^{\mathcal{D}}.\vec{n}_{\sigma\mathcal{K}}\right), \forall\mathcal{K}^*\in\partial\mathfrak{M}^*.\end{aligned}$$

Définition 3.5
La divergence discrète d'un champ de vecteurs de $(\mathbb{R}^2)^{\mathcal{T}}$ est définie de la manière suivante :
$\mathrm{div}^{\mathfrak{D}} : u^{\mathcal{T}} \in (\mathbb{R}^2)\mathcal{T} \mapsto (\mathrm{div}^{\mathcal{D}}\,u^{\mathcal{T}})_{\mathcal{D}\in\mathfrak{D}} \in (\mathcal{M}_2(\mathbb{R}))^{\mathfrak{D}}$, *avec :*

$$\mathrm{div}^{\mathcal{D}}\,u^{\mathcal{T}} = \mathrm{Tr}(\nabla^{\mathcal{D}} u^{\mathcal{T}}) = \frac{1}{2|\mathcal{D}|}\left[|\sigma|(u_{\mathcal{L}} - u_{\mathcal{K}}).\vec{n}_{\sigma\mathcal{K}} + |\sigma^*|(u_{\mathcal{L}^*} - u_{\mathcal{K}^*}).\vec{n}_{\sigma^*\mathcal{K}^*}\right], \forall\mathcal{D}\in\mathfrak{D}.$$

On définit également les produits scalaires discrets suivants :

$$[\![u^{\mathcal{T}}, v^{\mathcal{T}}]\!]_{\mathcal{T}} = \frac{1}{2}\left(\sum_{\mathcal{K}\in\mathfrak{M}} |\mathcal{K}| u_{\mathcal{K}}.\, v_{\mathcal{K}} + \sum_{\mathcal{K}^*\in\mathfrak{M}^*\cup\partial\mathfrak{M}^*} |\mathcal{K}^*| u_{\mathcal{K}^*}.\, v_{\mathcal{K}^*}\right), \forall u^{\mathcal{T}}, v^{\mathcal{T}} \in (\mathbb{R}^2)^{\mathcal{T}}$$
$$(p^{\mathfrak{D}}, q^{\mathfrak{D}})_{\mathfrak{D}} = \sum_{\mathcal{D}\in\mathfrak{D}} |\mathcal{D}| p^{\mathcal{D}} q^{\mathcal{D}}, \forall p^{\mathfrak{D}}, q^{\mathfrak{D}} \in \mathbb{R}^{\mathfrak{D}},$$

ainsi que les normes correspondantes :

$$||u^{\mathcal{T}}||_2 = [\![u^{\mathcal{T}}, v^{\mathcal{T}}]\!]_{\mathcal{T}}^{\frac{1}{2}}, \forall u^{\mathcal{T}} \in (\mathbb{R}^2)^{\mathcal{T}},$$
$$||p^{\mathfrak{D}}||_2 = (p^{\mathfrak{D}}, q^{\mathfrak{D}})_{\mathfrak{D}}^{\frac{1}{2}}, \forall p^{\mathfrak{D}} \in \mathbb{R}^{\mathfrak{D}}.$$

3.1.3 Problème discret

Etudions les flux numériques pour obtenir le problème discret.
Intégrons sur un volume de contrôle $\mathcal{K}$ l'équation $\operatorname{div}(-\nabla u + pId) = f$, on obtient :

$$|\mathcal{K}| f_{\mathcal{K}} := \int_{\mathcal{K}} f = \int_{\mathcal{K}} \operatorname{div}(-\nabla u + pId) = \sum_{\sigma\in\mathcal{E}_{\mathcal{K}}} \int_{\sigma} (-\nabla u + pId)\vec{n}_{\mathcal{K}\sigma}$$
$$\approx |\mathcal{K}| \operatorname{div}^{\mathcal{K}}(-\nabla^{\mathfrak{D}} u^{\mathcal{T}} + p^{\mathfrak{D}} Id) = \sum_{\mathcal{D}_{\sigma,\sigma^*}\in\mathfrak{D}_{\mathcal{K}}} |\sigma|(-\nabla^{\mathfrak{D}} u^{\mathcal{T}} + p^{\mathfrak{D}} Id).\, \vec{n}_{\sigma\mathcal{K}}.$$

En faisant de même sur $\mathcal{K}^*$, on obtient :

$$|\mathcal{K}^*| f_{\mathcal{K}^*} := \int_{\mathcal{K}^*} f = \int_{\mathcal{K}^*} \operatorname{div}(-\nabla u + pId) = \sum_{\sigma^*\in\mathcal{E}_{\mathcal{K}^*}} \int_{\sigma^*} (-\nabla u + pId)\vec{n}_{\mathcal{K}^*\sigma^*}$$
$$\approx |\mathcal{K}^*| \operatorname{div}^{\mathcal{K}^*}(-\nabla^{\mathfrak{D}} u^{\mathcal{T}} + p^{\mathfrak{D}} Id) = \sum_{\mathcal{D}_{\sigma,\sigma^*}\in\mathfrak{D}_{\mathcal{K}^*}} |\sigma^*|(-\nabla^{\mathfrak{D}} u^{\mathcal{T}} + p^{\mathfrak{D}} Id).\, \vec{n}_{\sigma^*\mathcal{K}^*}.$$

Posons alors :

$$F_{\mathcal{K},\mathcal{D}_{\sigma,\sigma^*}}(u^{\mathcal{T}}, p^{\mathfrak{D}}) = |\sigma|(-\nabla^{\mathfrak{D}} u^{\mathcal{T}} + p^{\mathfrak{D}} Id).\, \vec{n}_{\sigma\mathcal{K}}, \text{ si } \mathcal{D}_{\sigma,\sigma^*} \in \mathfrak{D}_{\mathcal{K}},$$
$$F_{\mathcal{K}^*,\mathcal{D}_{\sigma,\sigma^*}}(u^{\mathcal{T}}, p^{\mathfrak{D}}) = |\sigma^*|(-\nabla^{\mathfrak{D}} u^{\mathcal{T}} + p^{\mathfrak{D}} Id).\, \vec{n}_{\sigma^*\mathcal{K}^*}, \text{ si } \mathcal{D}_{\sigma,\sigma^*} \in \mathfrak{D}_{\mathcal{K}^*},$$

en utilisant la définition du gradient discret, on obtient :

$$F_{\mathcal{K},\mathcal{D}_{\sigma,\sigma^*}}(u^{\mathcal{T}}, p^{\mathfrak{D}}) = \frac{|\sigma|^2}{2|\mathcal{D}|}(u_{\mathcal{K}} - u_{\mathcal{L}}) + \frac{|\sigma||\sigma^*|}{2|\mathcal{D}|}(u_{\mathcal{K}^*} - u_{\mathcal{L}*}) + |\sigma|\vec{n}_{\sigma\mathcal{K}}.\, p^{\mathfrak{D}}$$
$$F_{\mathcal{K}^*,\mathcal{D}_{\sigma,\sigma^*}}(u^{\mathcal{T}}, p^{\mathfrak{D}}) = \frac{|\sigma||\sigma^*|}{2|\mathcal{D}|}(u_{\mathcal{K}} - u_{\mathcal{L}}) + \frac{|\sigma^*|^2}{2|\mathcal{D}|}(u_{\mathcal{K}^*} - u_{\mathcal{L}*}) + |\sigma^*|\vec{n}_{\sigma^*\mathcal{K}^*}.\, p^{\mathfrak{D}}.$$

Le problème discret consiste donc à trouver $u^{\mathcal{T}} \in (\mathbb{R}^2)^{\mathcal{T}}$ et $p^{\mathfrak{D}} \in \mathbb{R}^{\mathfrak{D}}$ tels que :

$$\begin{cases} \forall \mathcal{K} \in \mathfrak{M}, \displaystyle\sum_{\mathcal{D}_{\sigma,\sigma^*}\in\mathfrak{D}_{\mathcal{K}}} F_{\mathcal{K},\mathcal{D}_{\sigma,\sigma^*}}(u^{\mathcal{T}}, p^{\mathfrak{D}}) = |\mathcal{K}| f_{\mathcal{K}}, \\ \forall \mathcal{K}^* \in \mathfrak{M}^* \cup \partial\mathfrak{M}^*, \displaystyle\sum_{\mathcal{D}_{\sigma,\sigma^*}\in\mathfrak{D}_{\mathcal{K}^*}} F_{\mathcal{K}^*,\mathcal{D}_{\sigma,\sigma^*}}(u^{\mathcal{T}}, p^{\mathfrak{D}}) = |\mathcal{K}^*| f_{\mathcal{K}^*}, \\ \forall \mathcal{D} \in \mathfrak{D},\ \operatorname{div}^{\mathcal{D}} u^{\mathcal{T}} = 0, \\ \displaystyle\sum_{\mathcal{D}\in\mathfrak{D}} |\mathcal{D}| p^{\mathcal{D}} = 0. \end{cases}$$

III.2 Construction des matrices

Là encore pour construire les différentes matrices, on va pas parcourir les arêtes du maillage.
Rappelons que sous forme matricielle le problème de Stokes s'écrit :

$$\begin{cases} RU + B^T P = F \\ BU = 0 \end{cases}$$

L'espace de la vitesse étant vectoriel, on choisit de ranger le vecteur inconnue U comme tel : $U = \begin{pmatrix} u^1 \\ u^2 \end{pmatrix}$ tel que chaque composante u^i est telle que $u^i = \begin{pmatrix} u^i_{\mathcal{K}} \\ u^i_{\mathcal{K}^*} \end{pmatrix}$.

La matrice de rigidité :
Utilisons le problème discret pour obtenir la matrice de rigidité R.
Commençons par remarquer que $R \in \mathcal{M}_{2(n_s+n_c)}$ où n_s est le nombre de sommets du maillage et n_c le nombre de volumes de contrôle. D'après le problème sous forme discrète, on a :$R = \begin{pmatrix} \bar{R} & 0 \\ 0 & \bar{R} \end{pmatrix}$, où la matrice $\bar{R}$ est construite comme suit :
On parcours les arêtes et lorsque l'on est sur l'arête $\sigma = \sigma_{\mathcal{KL}}$:

- sur la ligne correspondant au volume de contrôle $\mathcal{K}$ notée $l_{\mathcal{K}}$, on rajoute le terme :
  - $+\frac{|\sigma|^2}{2|\mathcal{D}|}$ à la colonne correspondant au volume de contrôle $\mathcal{K}$ notée $c_{\mathcal{K}}$,
  - $-\frac{|\sigma|^2}{2|\mathcal{D}|}$ à la colonne correspondant au volume de contrôle $\mathcal{L}$ notée $c_{\mathcal{L}}$ si $\mathcal{K} \notin \partial\mathfrak{M}$,
  - $+\frac{|\sigma||\sigma^*|}{2|\mathcal{D}|}\vec{n}_{\sigma\mathcal{K}}.\vec{n}_{\sigma^*\mathcal{K}^*}$ à la colonne correspondant au sommet $\mathcal{K}^*$ c'est à dire à la colonne $n_c + c_{\mathcal{K}}$, si $\mathcal{K} \notin \partial\mathfrak{M}$,
  - $-\frac{|\sigma||\sigma^*|}{2|\mathcal{D}|}\vec{n}_{\sigma\mathcal{K}}.\vec{n}_{\sigma^*\mathcal{K}^*}$ à la colonne correspondant au sommet $\mathcal{L}^*$ c'est à dire à la colonne $n_c + c_{\mathcal{L}}$, si $\mathcal{K} \notin \partial\mathfrak{M}$.
- si $\mathcal{K} \notin \partial\mathfrak{M}$, sur la ligne correspondant au volume de contrôle $\mathcal{L}$ notée $c_{\mathcal{L}}$, on rajoute le terme :
  - $-\frac{|\sigma|^2}{2|\mathcal{D}|}$ à la colonne $c_{\mathcal{K}}$,
  - $-\frac{|\sigma|^2}{2|\mathcal{D}|}$ à la colonne $c_{\mathcal{L}}$,
  - $-\frac{|\sigma||\sigma^*|}{2|\mathcal{D}|}\vec{n}_{\sigma\mathcal{K}}.\vec{n}_{\sigma^*\mathcal{K}^*}$ à la colonne $n_c + c_{\mathcal{K}}$,
  - $+\frac{|\sigma||\sigma^*|}{2|\mathcal{D}|}\vec{n}_{\sigma\mathcal{K}}.\vec{n}_{\sigma^*\mathcal{K}^*}$ à la colonne $n_c + c_{\mathcal{L}}$.
- si $\mathcal{K} \notin \partial\mathfrak{M}$, sur la ligne correspondant au sommet $\mathcal{K}^*$, c'est à dire la ligne $n_c + l_{\mathcal{K}}$, on rajoute le terme :
  - $+\frac{|\sigma||\sigma^*|}{2|\mathcal{D}|}\vec{n}_{\sigma\mathcal{K}}.\vec{n}_{\sigma^*\mathcal{K}^*}$ à la colonne $c_{\mathcal{K}}$,
  - $-\frac{|\sigma||\sigma^*|}{2|\mathcal{D}|}\vec{n}_{\sigma\mathcal{K}}.\vec{n}_{\sigma^*\mathcal{K}^*}$ à la colonne $c_{\mathcal{L}}$,
  - $+\frac{|\sigma^*|^2}{2|\mathcal{D}|}$ à la colonne $n_c + c_{\mathcal{K}}$,
  - $-\frac{|\sigma^*|^2}{2|\mathcal{D}|}$à la colonne $n_c + c_{\mathcal{L}}$.
- si $\mathcal{K} \notin \partial\mathfrak{M}$, sur la ligne correspondant au sommet $\mathcal{L}^*$, c'est à dire la ligne $n_c + l_{\mathcal{L}}$, on rajoute le terme :
  - $-\frac{|\sigma||\sigma^*|}{2|\mathcal{D}|}\vec{n}_{\sigma\mathcal{K}}.\vec{n}_{\sigma^*\mathcal{K}^*}$ à la colonne $c_{\mathcal{K}}$,
  - $+\frac{|\sigma||\sigma^*|}{2|\mathcal{D}|}\vec{n}_{\sigma\mathcal{K}}.\vec{n}_{\sigma^*\mathcal{K}^*}$ à la colonne $c_{\mathcal{L}}$,
  - $-\frac{|\sigma^*|^2}{2|\mathcal{D}|}$ à la colonne $n_c + c_{\mathcal{K}}$,
  - $+\frac{|\sigma^*|^2}{2|\mathcal{D}|}$à la colonne $n_c + c_{\mathcal{L}}$.

La matrice de masse :
La matrice de masse M est définie par $(MP, Q) = (p^{\mathfrak{D}}, q^{\mathfrak{D}})_{\mathfrak{D}}$, donc d'après la définition du produit scalaire discret, $M \in \mathcal{M}_{n_a}$ où n_a est le nombre d'arêtes du maillage :

$$M = diag(|\mathcal{D}|).$$

La matrice de divergence :
La matrice de divergence B est telle que $(BV, P) = b(v^{\mathcal{T}}, p^{\mathfrak{D}})$, donc :

$$(BV, P) = -\sum_{\mathcal{D}\in\mathfrak{D}} |\mathcal{D}| p^{\mathcal{D}} \operatorname{div}^{\mathcal{D}} v^{\mathcal{T}} = \sum_{\mathcal{D}\in\mathfrak{D}} \frac{1}{2} \left[|\sigma|\vec{n}_{\sigma\mathcal{K}}.(u_{\mathcal{K}} - u_{\mathcal{L}}) + |\sigma^*|\vec{n}_{\sigma^*\mathcal{K}^*}.(u_{\mathcal{K}^*} - u_{\mathcal{L}^*})\right] p^{\mathcal{D}}.$$

Ainsi, $B \in \mathcal{M}_{n_a, 2(n_c+n_s)}$ et lorsque l'on est sur l'arête $\sigma = \sigma_{\mathcal{KL}}$, on ajoute :

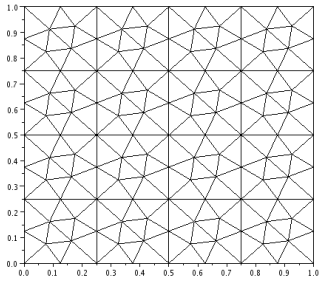
- si $\mathcal{K} \in \partial\mathfrak{M}$:
  - $+\frac{1}{2}|\sigma|\vec{n}^1_{\sigma\mathcal{K}}$ à la colonne $c_{\mathcal{K}}$ pour la première composante de la vitesse,
  - $+\frac{1}{2}|\sigma|\vec{n}^2_{\sigma\mathcal{K}}$ à la colonne $c_{\mathcal{K}} + n_c + n_s$ pour la deuxième composante de la vitesse.
- si $\mathcal{K} \notin \partial\mathfrak{M}$:
  - $+\frac{1}{2}|\sigma|\vec{n}^1_{\sigma\mathcal{K}}$ à la colonne $c_{\mathcal{K}}$ pour la première composante de la vitesse,
  - $+\frac{1}{2}|\sigma|\vec{n}^2_{\sigma\mathcal{K}}$ à la colonne $c_{\mathcal{K}} + n_c + n_s$ pour la deuxième composante de la vitesse,
  - $-\frac{1}{2}|\sigma|\vec{n}^1_{\sigma\mathcal{K}}$ à la colonne $c_{\mathcal{L}}$,
  - $-\frac{1}{2}|\sigma|\vec{n}^2_{\sigma\mathcal{K}}$ à la colonne $c_{\mathcal{L}} + n_c + n_s$,

- $+\frac{1}{2}|\sigma^*|\vec{n}^1_{\sigma^*\mathcal{K}^*}$ à la colonne $c_\mathcal{K} + n_c$,
- $+\frac{1}{2}|\sigma^*|\vec{n}^2_{\sigma^*\mathcal{K}^*}$ à la colonne $c_\mathcal{K} + 2n_c + n_s$,
- $-\frac{1}{2}|\sigma^*|\vec{n}^1_{\sigma^*\mathcal{K}^*}$ à la colonne $c_\mathcal{L} + n_c$,
- $-\frac{1}{2}|\sigma^*|\vec{n}^2_{\sigma^*\mathcal{K}^*}$ à la colonne $c_\mathcal{L} + 2n_c + n_s$.

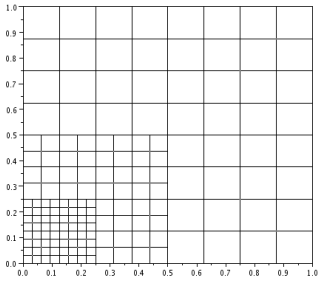
On peut alors vérifier que $B^T \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = 0$ et que les colonnes de B correspondants aux $u_{\mathcal{K}^*}$ sont bien nulles.

III.3 Résultats numériques

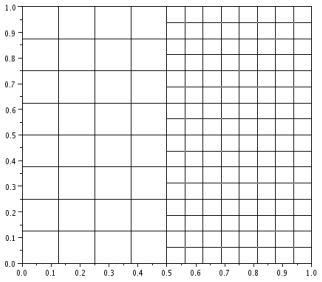
On utilise ici 4 maillages différents :
– un maillage triangle (celui utilisé pour $VF4$) où les centres des mailles sont pris au centre du cercle circonscrit du triangle,
– un maillage triangle centre de gravité où le centre d'une maille est cette fois le centre de gravité du triangle correspondant,
– un maillage localement raffiné,
– un maillage rectangle bidomaine.



Les maillages triangles



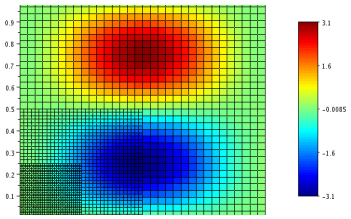
Le maillage localement raffiné



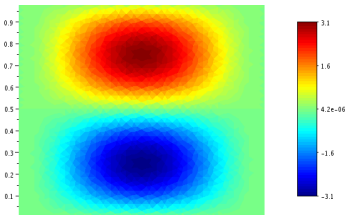
Le maillage rectangle bidomaine

3.3.1 Convergence du schéma

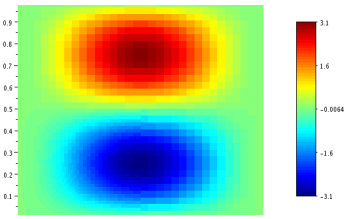
On va commencer par vérifier que le schéma converge bien vers la bonne solution pour ces 4 maillages en affichant les mêmes tracés que pour le schéma $VF4$ pour le maillage localement raffiné comme solution exacte le produit de sinus et cosinus. Dans le cas du tracé des solutions exacte et approchée le pas du maillage est $h = 3.125.10^{-2}$.



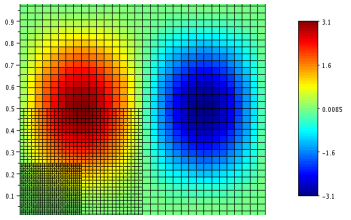
Première composante de la vitesse exacte



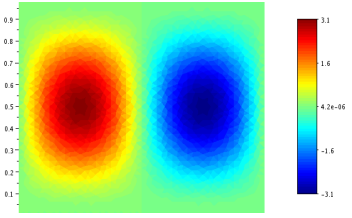
Vitesse approchée triangle



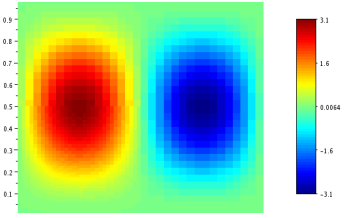
Vitesse approchée localement raffiné



Deuxième composante de la vitesse exacte

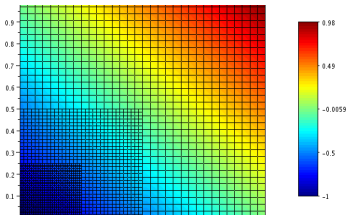


Vitesse approchée triangle

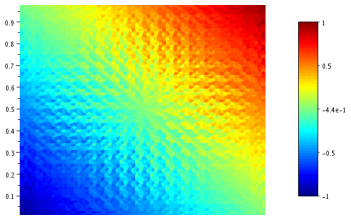


Vitesse approchée localement raffiné

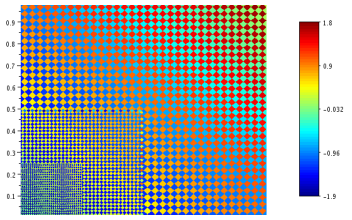
On obtient dans chaque cas, un ordre 1 en pression et en vitesse en un ordre 1 pour la norme H^1 et un ordre 2 pour la norme L^2.



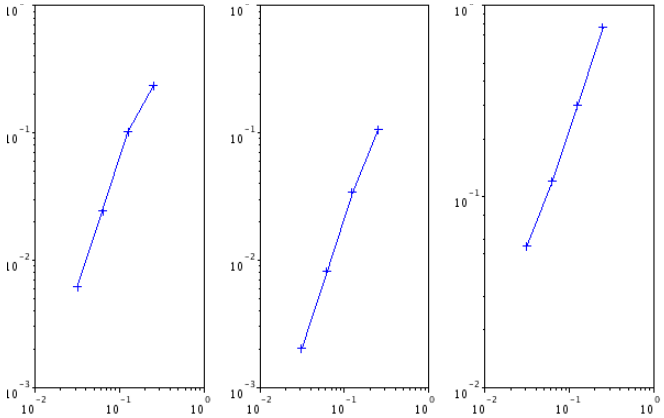
Pression exacte



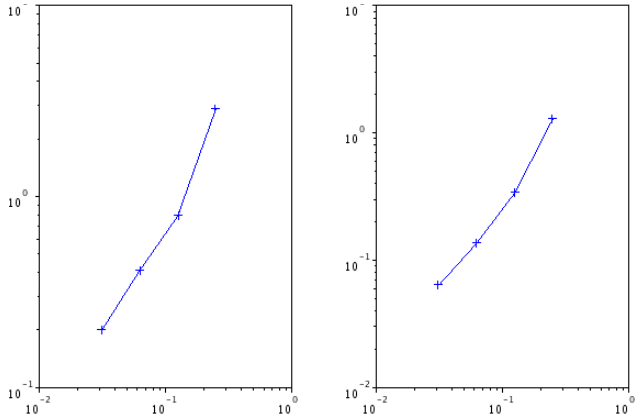
Pression approchée triangle



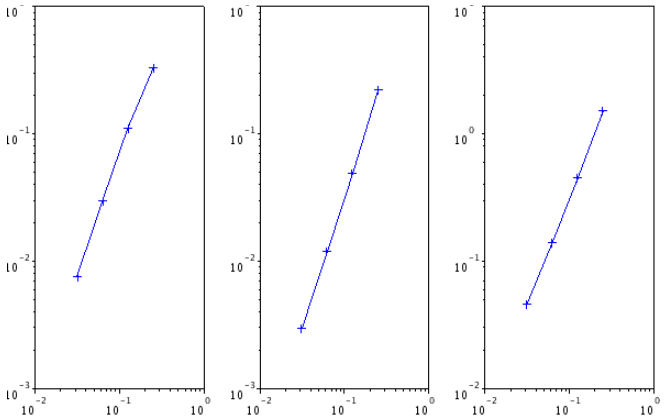
Pression approchée localement raffiné



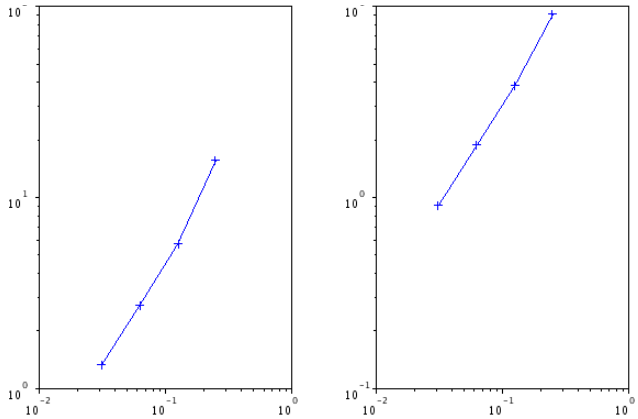
Erreur en vitesse maillage triangle ($L^\infty/L^2/H^1$)



Erreur en pression maillage triangle (L^∞/L^2)



Erreur en vitesse localement raffiné ($L^\infty/L^2/H^1$)

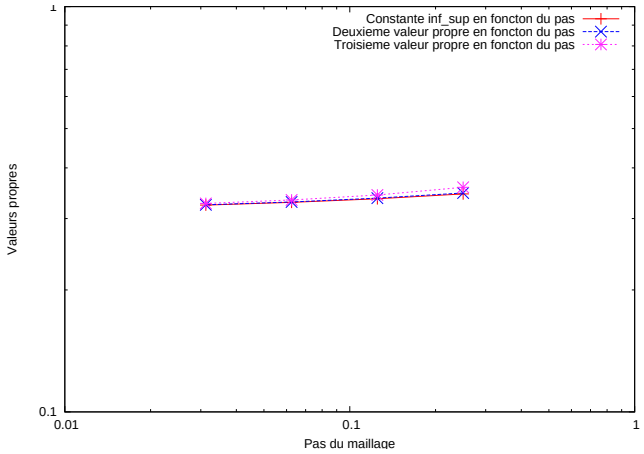


Erreur en pression localement raffiné (L^∞/L^2)

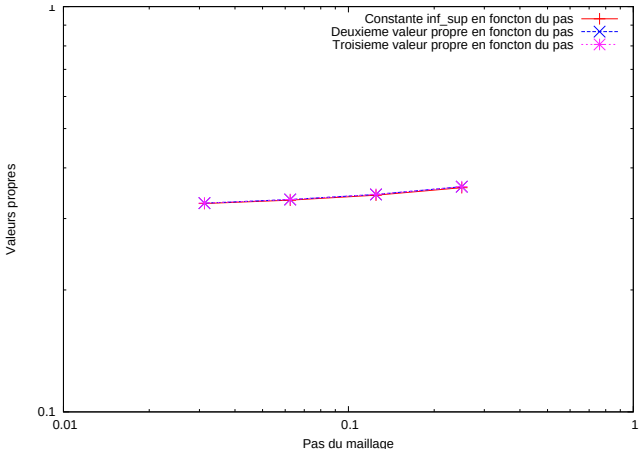
3.3.2 Les problèmes de valeurs propres

Regardons tout d'abord les valeurs des différentes valeurs propres en fonction du pas du maillage. Dans certains cas, la constante Inf-Sup semble être minorée uniformément.

	Triangles				Triangles centre de gravité			
h	vp_1	vp_2	vp_3	it	vp_1	vp_2	vp_3	it
0.25	0.345112	0.346884	0.358196	60	0.357457	0.359448	0.359524	49
0.125	0.33558	0.33682	0.342908	53	0.342523	0.34404	0.344065	55
0.0625	0.328832	0.329574	0.333328	110	0.333126	0.334182	0.334216	87
0.03125	0.323851	0.324307	0.326801	111	0.326688	0.327449	0.327483	97

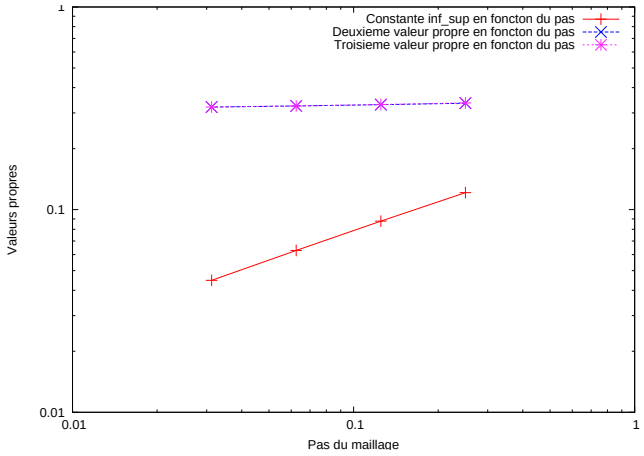


Maillage triangle

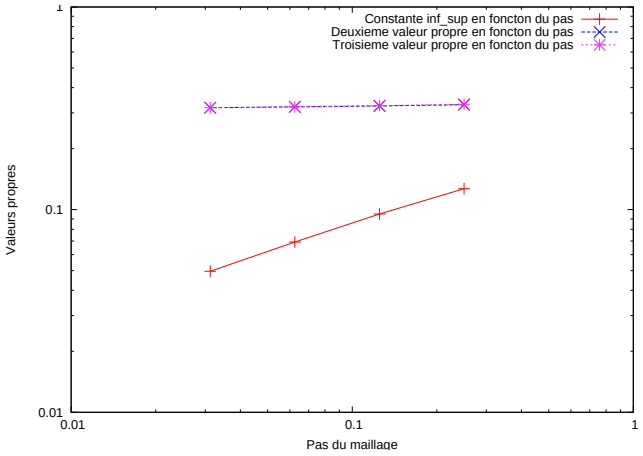


Maillage triangle centre de gravité

Dans d'autres cas, la constante Inf-Sup tend vers 0 alors que les autres valeurs propres sont bornées :



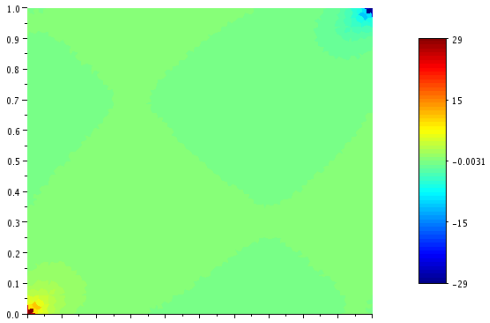
Maillage rectangle bidomaine



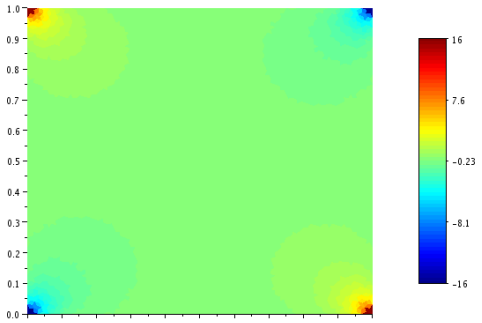
Maillage localement raffiné

	Bidomaine				Localement raffiné			
h	vp_1	vp_2	vp_3	it	vp_1	vp_2	vp_3	it
0.25	0.121347	0.335225	0.336293	567	0.126894	0.329736	0.330911	114
0.125	0.0877552	0.329639	0.330207	448	0.0951351	0.324774	0.326067	131
0.0625	0.0628534	0.32475	0.324869	485	0.0691824	0.321364	0.321827	166
0.03125	0.0447933	0.320825	0.320858	709	0.0496684	0.318359	0.318523	200

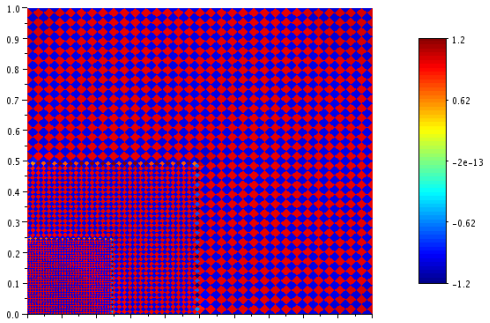
Regardons alors les modes propres correspondant pour un pas $h = 3.125.10^{-2}$.



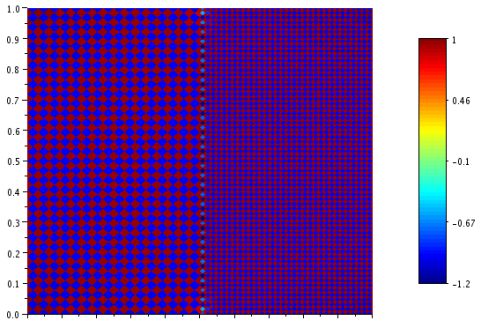
Mode propre triangle



Mode propre triangle centre de gravité



Mode propre localement raffiné



Mode propre rectangle bidomaine

Conclusion :
On a donc trouvé des maillages pour lesquels le schéma converge mais dont la première valeur propre tend vers 0.
On voit que les deux modes propres dont la constante Inf-Sup tend vers 0 sont des modes en damier. L'influence de cette constante Inf-Sup est faible car elle se retrouve seulement au niveau des interfaces et sur un unique mode. En effet, contrairement au cas $\mathbb{P}1\text{bulle}/\mathbb{P}_1$ ou au cas $\mathbb{Q}_1/\mathbb{P}_0$ seule la première valeur propre tend vers 0 et les suivantes semblent minorées. Ceci peut expliquer le bon comportement du schéma malgré tout.

IV Le schéma DDFV stabilisé

Le schéma précédent étant bien posé seulement pour des maillages particuliers, il faut ajouter un terme de stabilisation. On pense alors à deux types de stabilisation :

- une première dont le terme de stabilisation est proportionnel à la pression et à laquelle on a commencé à s'intéresser,
- puis une seconde correspondant à une approximation volume fini non consistante de l'opérateur de Laplace inspirée de la méthode de Brezzi-Pitkäranta que l'on a pas eu le temps d'étudier numériquement.

Dans tous les cas, la constante Inf-Sup n'est plus la même et il va donc falloir se ramener à un nouveau problème de valeur propre.

IV.1 Une nouvelle constante Inf-Sup

4.1.1 Rappels dans le cadre d'un problème mixte

Dans le cas d'un problème mixte tel que le problème de Stokes,

- On sait que :

Théorème 4.1 (Banach-Necas-Babuska)
Soient V et W deux espaces de Hilbert, a une forme bilinéaire continue sur $V \times W$. Alors les propositions suivantes sont équivalentes :

1. *Pour toute forme linéaire continue L sur W, il existe un unique $u \in V$ tel que :*

$$a(u, w) = L(w), \forall x \in W. \tag{IV.1}$$

2. *Les deux conditions suivantes sont vérifiées :*

$$\exists \alpha > 0,\ \inf_{v \in V} \left(\sup_{w \in W} \frac{a(v, w)}{||v||_V ||w||_W} \right) \geq \alpha, \tag{BNB1}$$

$$(\forall v \in V, a(u, w) = 0) \Rightarrow w = 0. \tag{BNB2}$$

– Puis, on remarque que le problème consistant à trouver $u \in X$ et $p \in M$ tels que :

$$\begin{cases} a(u, v) + b(v, p) = L(v), \forall v \in X, \\ b(u, q) = G(q), \forall q \in M. \end{cases} \tag{IV.2}$$

peut se mettre sous la forme générale d'un problème variationnel en constatant qu'il est équivalent à trouver $(u, p) \in X \times M$ tel que :

$$a(u, v) + b(v, p) - b(u, q) = L(v) - G(q), \forall v \in X, \forall q \in M. \tag{IV.3}$$

– On pose alors $V = W = X \times M$ et $c((u, p), (v, q)) = a(u, v) + b(v, p) - b(u, q)$.

– En appliquant le théorème BNB on sait alors que le problème est bien posé si et seulement si les conditions (BNB1) et (BNB2) sont vérifiées pour la forme c.

– A partir de là on a bien le Théorème 1.1 (cf p.5) sur la Condition Inf-Sup.

4.1.2 Le cas de la stabilisation

Le problème que l'on a lorsque l'on stabilise le schéma DDFV c'est que l'on ne résoud plus le problème (IV.2) mais le problème consistant à trouver $u \in X$ et $p \in M$ tels que :

$$\begin{cases} a(u, v) + b(v, p) = L(v), \forall v \in X, \\ b(u, q) - s(p, q) = G(q), \forall q \in M. \end{cases} \tag{IV.4}$$

où s représente le terme de stabilisation.
Il est donc équivalent de trouver $(u, p) \in X \times M$ tel que :

$$a(u, v) + b(v, p) - b(u, q) + s(p, q) = L(v) - G(q), \forall v \in X, \forall q \in M. \tag{IV.5}$$

On pose donc encore $V = W = X \times M$ mais $\tilde{c}((u, p), (v, q)) = a(u, v) + b(v, p) - b(u, q) + s(p, q)$.

On ne sait donc pas se ramener cette fois à la condition Inf-Sup du théorème 1.1.

D'après le théorème BNB, on va donc étudier la constante :

$$\alpha_h = \inf_{(u_h, p_h) \in V_h} \left(\sup_{(v_h, q_h) \in W_h} \frac{\tilde{c}((u_h, p_h), (v_h, q_h))}{||(u_h, p_h)||_V ||(v_h, q_h)||_W} \right), \tag{IV.6}$$

ce qui s'écrit sous la forme matricielle suivante :

$$\alpha_h = \inf_{(U,P)} \left(\sup_{(V,Q)} \frac{\begin{pmatrix} R & B^T \\ B & -S \end{pmatrix} \begin{pmatrix} U \\ P \end{pmatrix} \cdot \begin{pmatrix} V \\ Q \end{pmatrix}}{\left\| \begin{pmatrix} U \\ P \end{pmatrix} \right\|_{X \times M} \left\| \begin{pmatrix} V \\ Q \end{pmatrix} \right\|_{X \times M}} \right).$$

Essayons donc là encore de se ramener à un problème de valeurs propres.
On a :

$$\begin{cases} \left\| \begin{pmatrix} U \\ P \end{pmatrix} \right\|^2_{X\times M} = \|U\|^2_X + \|P\|^2_M = (RU,U) + (MP,P) \\ \begin{pmatrix} R & B^T \\ B & -S \end{pmatrix} \begin{pmatrix} U \\ P \end{pmatrix} \cdot \begin{pmatrix} V \\ Q \end{pmatrix} = (RU,V) + (B^TP,V) + (BU,Q) - (SP,Q) \end{cases}$$

Posons $\tilde{V} = R^{1/2}V$ et $\tilde{Q} = M^{1/2}Q$, alors :

$$\alpha_h = \inf_{(U,P)} \frac{1}{\left\| \begin{pmatrix} U \\ P \end{pmatrix} \right\|_{X\times M}} \left(\sup_{(\tilde{V},\tilde{Q})} \frac{\begin{pmatrix} R^{1/2}U + R^{-1/2}B^TP \\ M^{-1/2}BU - M^{-1/2}SP \end{pmatrix} \cdot \begin{pmatrix} \tilde{V} \\ \tilde{Q} \end{pmatrix}}{\left\| \begin{pmatrix} \tilde{V} \\ \tilde{Q} \end{pmatrix} \right\|_{\mathbb{R}^{N_U}\times N_P}} \right) = \inf_{(U,P)} \frac{\left\| \begin{pmatrix} R^{1/2}U + R^{-1/2}B^TP \\ M^{-1/2}BU - M^{-1/2}SP \end{pmatrix} \right\|}{\left\| \begin{pmatrix} U \\ P \end{pmatrix} \right\|_{X\times M}}$$

en utilisant que, R et M étant symétrique, on a :

$$\begin{cases} (RU,V) = (U,RV) = (U,R^{1/2}\tilde{V}) = (R^{1/2}U,\tilde{V}) \\ (B^TP,V) = (B^TP,R^{-1/2}\tilde{V}) = (R^{-1/2}B^TP,\tilde{V}) \\ (BU,Q) = (BU,M^{-1/2}\tilde{Q}) = (M^{-1/2}BU,\tilde{Q}) \\ (SP,Q) = (SP,M^{-1/2}\tilde{Q}) = (M^{-1/2}SP,\tilde{Q}) \\ \left\| \begin{pmatrix} V \\ Q \end{pmatrix} \right\|^2_{X\times M} = (RV,V) + (MQ,Q) = (R^{1/2}V,R^{1/2}V) + (M^{1/2}Q,M^{1/2}Q) = (\tilde{V},\tilde{V}) + (\tilde{Q},\tilde{Q}) \\ = ||\tilde{V}||^2_{\mathbb{R}^{N_U}} + ||\tilde{Q}||^2_{\mathbb{R}^{N_P}} = \left\| \begin{pmatrix} \tilde{V} \\ \tilde{Q} \end{pmatrix} \right\|^2_{\mathbb{R}^{N_U}\times N_P} \end{cases}$$

Posons maintenant $\tilde{U} = R^{1/2}U$ et $\tilde{P} = M^{1/2}P$, alors $\left\| \begin{pmatrix} U \\ P \end{pmatrix} \right\|_{X\times M} = \left\| \begin{pmatrix} \tilde{U} \\ \tilde{P} \end{pmatrix} \right\|_{\mathbb{R}^{N_U}\times N_P}$ et :

$$\left\| \begin{pmatrix} R^{1/2}U + R^{-1/2}B^TP \\ M^{-1/2}BU - M^{-1/2}SP \end{pmatrix} \right\|^2 = ||R^{1/2}U + R^{-1/2}B^TP||^2 + ||M^{-1/2}BU - M^{-1/2}SP||^2$$

On a alors :

$$\begin{aligned} ||R^{1/2}U + R^{-1/2}B^TP||^2 &= (R^{1/2}U + R^{-1/2}B^TP, R^{1/2}U + R^{-1/2}B^TP) \\ &= (R^{1/2}U, R^{1/2}U) + (R^{1/2}U, R^{-1/2}B^TP) + (R^{-1/2}B^TP, R^{1/2}U) + (R^{-1/2}B^TP, R^{-1/2}B^TP) \\ &= (\tilde{U},\tilde{U}) + (\tilde{U}, R^{-1/2}B^TM^{-1/2}\tilde{P}) + (R^{-1/2}B^TM^{-1/2}\tilde{P},\tilde{U}) + (R^{-1/2}B^TM^{-1/2}\tilde{P}, R^{-1/2}B^TM^{-1/2}\tilde{P}) \\ &= (\tilde{U},\tilde{U}) + (R^{-1/2}B^TM^{-1/2}\tilde{P},\tilde{U}) + (M^{-1/2}BR^{-1/2}\tilde{U},\tilde{P}) + (M^{-1/2}BR^{-1}B^TM^{-1/2}\tilde{P},\tilde{P}) \\ &= \begin{pmatrix} Id & R^{-1/2}B^TM^{-1/2} \\ M^{-1/2}BR^{-1/2} & M^{-1/2}BR^{-1}B^TM^{-1/2} \end{pmatrix} \begin{pmatrix} \tilde{U} \\ \tilde{P} \end{pmatrix} \cdot \begin{pmatrix} \tilde{U} \\ \tilde{P} \end{pmatrix}, \end{aligned}$$

et :

$$\begin{aligned} ||M^{-1/2}BU - M^{-1/2}SP||^2 &= (M^{-1/2}BU - M^{-1/2}SP, M^{-1/2}BU - M^{-1/2}SP) = (M^{-1}(BU-SP), BU-SP) \\ &= (M^{-1}BU, BU) - (M^{-1}BU, SP) - (M^{-1}SP, BU) + (M^{-1}SP, SP) \\ &= (M^{-1}BR^{-1/2}\tilde{U}, BR^{-1/2}\tilde{U}) - (M^{-1}BR^{-1/2}\tilde{U}, SM^{-1/2}\tilde{P}) \\ &- (M^{-1}SM^{-1/2}\tilde{P}, BR^{-1/2}\tilde{U}) + (M^{-1}SM^{-1/2}\tilde{P}, SM^{-1/2}\tilde{P}) \\ &= (R^{-1/2}B^TM^{-1}BR^{-1/2}\tilde{U},\tilde{U}) - (M^{-1/2}S^TM^{-1}BR^{-1/2}\tilde{U},\tilde{P}) \\ &- (R^{-1/2}B^TM^{-1}SM^{-1/2}\tilde{P},\tilde{U}) + (M^{-1/2}S^TM^{-1}SM^{-1/2}\tilde{P},\tilde{P}) \\ &= \begin{pmatrix} R^{-1/2}B^TM^{-1}BR^{-1/2} & -R^{-1/2}B^TM^{-1}SM^{-1/2} \\ -M^{-1/2}S^TM^{-1}BR^{-1/2} & M^{-1/2}S^TM^{-1}SM^{-1/2} \end{pmatrix} \begin{pmatrix} \tilde{U} \\ \tilde{P} \end{pmatrix} \cdot \begin{pmatrix} \tilde{U} \\ \tilde{P} \end{pmatrix}. \end{aligned}$$

On obtient alors :

$$\alpha_h^2 = \inf_{(\tilde{U},\tilde{P})} \frac{\begin{pmatrix} Id + R^{-1/2}B^TM^{-1}BR^{-1/2} & R^{-1/2}B^TM^{-1/2} - R^{-1/2}B^TM^{-1}SM^{-1/2} \\ M^{-1/2}BR^{-1/2} - M^{-1/2}S^TM^{-1}BR^{-1/2} & M^{-1/2}BR^{-1}B^TM^{-1/2} + M^{-1/2}S^TM^{-1}SM^{-1/2} \end{pmatrix} \begin{pmatrix} \tilde{U} \\ \tilde{P} \end{pmatrix} \cdot \begin{pmatrix} \tilde{U} \\ \tilde{P} \end{pmatrix}}{\left\| \begin{pmatrix} \tilde{U} \\ \tilde{P} \end{pmatrix} \right\|^2}$$

$=$ plus petite valeur propre de la matrice Mat_totale,

avec :

$$Mat\_totale = \begin{pmatrix} Id + R^{-1/2}B^TM^{-1}BR^{-1/2} & R^{-1/2}B^TM^{-1/2} - R^{-1/2}B^TM^{-1}SM^{-1/2} \\ M^{-1/2}BR^{-1/2} - M^{-1/2}S^TM^{-1}BR^{-1/2} & M^{-1/2}BR^{-1}B^TM^{-1/2} + M^{-1/2}S^TM^{-1}SM^{-1/2} \end{pmatrix}$$

De plus,

$$\begin{aligned}
\lambda \text{ valeur propre de } Mat_totale &\iff \exists \begin{pmatrix} U \\ P \end{pmatrix} \in (\mathbb{R}^{N_U} \times \mathbb{R}^{N_p})^* \text{ tel que } Mat_totale \begin{pmatrix} U \\ P \end{pmatrix} = \lambda \begin{pmatrix} U \\ P \end{pmatrix} \\
&\iff \exists \begin{pmatrix} U \\ P \end{pmatrix} \in (\mathbb{R}^{N_U} \times \mathbb{R}^{N_p})^* \text{ tel que :} \\
&\begin{pmatrix} U + R^{-1/2}B^TM^{-1}BR^{-1/2}U + R^{-1/2}B^T(Id - M^{-1}S)M^{-1/2}P \\ M^{-1/2}(Id - S^TM^{-1})BR^{-1/2}U + M^{-1/2}(BR^{-1}B^T + S^TM^{-1}S)M^{-1/2}P \end{pmatrix} = \lambda \begin{pmatrix} U \\ P \end{pmatrix} \\
&\iff \exists \begin{pmatrix} \bar{U} \\ \bar{P} \end{pmatrix} = \begin{pmatrix} R^{-1/2}U \\ M^{-1/2}V \end{pmatrix} \in (\mathbb{R}^{N_U} \times \mathbb{R}^{N_p})^* \text{ tel que :} \\
&\begin{cases} \bar{U} + R^{-1}B^TM^{-1}B\bar{U} + R^{-1}B^T(Id - M^{-1}S)\bar{P} = \lambda\bar{U} \\ M^{-1}(Id - S^TM^{-1})B\bar{U} + M^{-1}(BR^{-1}B^T + S^TM^{-1}S)\bar{P} = \lambda\bar{P} \end{cases} \\
&\iff \lambda \text{ valeur propre de } \widetilde{Mat_totale} = \begin{pmatrix} Id + R^{-1}B^TM^{-1}B & R^{-1}B^T(Id - M^{-1}S) \\ M^{-1}(Id - S^TM^{-1})B & M^{-1}(BR^{-1}B^T + S^TM^{-1}S) \end{pmatrix}.
\end{aligned}$$

IV.2 Une stabilisation proportionnelle à la pression

4.2.1 Etude théorique

Le problème discret

Cette stabilisation consiste à ajouter un terme proportionnel à la préssion, c'est à dire que l'on a le problème discret suivant :

$$\begin{cases} \text{Trouver } u^{\mathcal{T}} \in (\mathbb{R}^2)^{\mathcal{T}} \text{ et } p^{\mathfrak{D}} \in \mathbb{R}^{\mathfrak{D}} \text{ tels que,} \\ \forall \mathcal{K} \in \mathfrak{M}, \displaystyle\sum_{\mathcal{D}_{\sigma,\sigma^*} \in \mathfrak{D}_{\mathcal{K}}} F_{\mathcal{K},\mathcal{D}_{\sigma,\sigma^*}}(u^{\mathcal{T}}, p^{\mathfrak{D}}) = |\mathcal{K}| f_{\mathcal{K}}, \\ \forall \mathcal{K}^* \in \mathfrak{M}^* \cup \partial\mathfrak{M}^*, \displaystyle\sum_{\mathcal{D}_{\sigma,\sigma^*} \in \mathfrak{D}_{\mathcal{K}^*}} F_{\mathcal{K}^*,\mathcal{D}_{\sigma,\sigma^*}}(u^{\mathcal{T}}, p^{\mathfrak{D}}) = |\mathcal{K}^*| f_{\mathcal{K}^*}, \\ \forall \mathcal{D} \in \mathfrak{D}, \ \mathrm{div}^{\mathcal{D}}\, u^{\mathcal{T}} + \lambda (size(\mathcal{T}))^{\gamma} p^{\mathfrak{D}} = 0, \\ \displaystyle\sum_{\mathcal{D} \in \mathfrak{D}} |\mathcal{D}| p^{\mathcal{D}} = 0. \end{cases} \tag{IV.7}$$

Dans [Kre10, Th. V.4 p113] on a alors l'estimation d'erreur suivante :

Théorème 4.2
On suppose que la solution (u, p) du problème de Stokes appartient à $(H^2(\Omega))^2 \times H^1(\Omega)$. On suppose que $\lambda > 0$.
Soit $(u^{\mathcal{T}}, p^{\mathfrak{D}}) \in (\mathbb{R}^2)^{\mathcal{T}} \times \mathbb{R}^{\mathfrak{D}}$ la solution du schéma DDFV stabilisé (IV.7)*. Alors, il existe une constante $C > 0$ dépendant uniquement de $\mathcal{T}$, λ, $||u||_{H^2}$ et $||p||_H^1$, telle que :*

$$||\nabla u - \nabla^{\mathfrak{D}} u^{\mathcal{T}}||_2 \leq C size(\mathcal{T})^{\frac{1}{2}}. \tag{IV.8}$$

Problème sous forme matricielle

Ainsi, sous forme matricielle, cela revient simplement à prendre $S = \lambda h^\gamma M$ où h est le pas du maillage. On obtient alors :

$$\widetilde{Mat\_totale} = \begin{pmatrix} Id + R^{-1}B^T M^{-1}B & (1-\lambda h^\gamma)R^{-1}B^T \\ (1-\lambda h^\gamma)M^{-1}B & M^{-1}BR^{-1}B^T + \lambda^2 h^{2\gamma} Id \end{pmatrix}.$$

Prenons alors $U = 0$ et $P = 1$, sachant que $B^T P = 0$, on a :

$$\widetilde{Mat\_totale} \begin{pmatrix} U \\ P \end{pmatrix} = \begin{pmatrix} U \\ \lambda^2 h^{2\gamma} P \end{pmatrix} = \lambda^2 h^{2\gamma} \begin{pmatrix} U \\ P \end{pmatrix}.$$

Ainsi, le terme $\lambda^2 h^{2\gamma}$ est une valeur propre de $\widetilde{Mat\_totale}$ associée à une pression constante, c'est à dire à $p \in L^2(\Omega) \backslash L^2_0(\Omega)$. On sait alors qu'elle tend vers 0 avec le pas du maillage h et on ne la prendra pas en compte lorsqu'elle intervient.

4.2.2 Résultats numériques

Dans cette partie, on a calculé les différentes valeurs propres de la matrice $\widetilde{Mat\_totale}$ avec la méthode `spec` de Scilab. Ceci n'est donc pas du tout optimisé et il faudrait utilisé l'algorithme d'itérations de sous-espaces de Rayleigh-Ritz à la matrice $\widetilde{Mat\_totale}$ en pensant là encore à enlever les pressions constantes.

On présente ici pour les 4 maillages précédents :

- Les tableaux donnant les 4 plus petites valeurs propres de la matrice $\widetilde{Mat\_totale}$ pour différentes valeurs de γ et λ. Ces tableaux permettent de remarquer que la plus petite valeur propre est bien λh^γ associée à un vecteur propre de pression constante.
- On oublie alors cette valeur propre et on trace les 3 valeurs propres suivantes en fonction du pas du maillage.

On note vp_h la plus petite valeur propre qui est celle correspondant aux pressions constantes et vp_i, $i = 1, 2, 3$ les suivantes.

	$\lambda = 0.01$				$\lambda = 0.001$			
h	vp_1	vp_2	vp_3	vp_h	vp_1	vp_2	vp_3	vp_h
0.25	0.0637583	0.0649393	0.0744931	0.0025	0.061631	0.0628141	0.0723846	0.00025
0.125	0.0591404	0.0600556	0.0649532	0.00125	0.0580738	0.0589897	0.0638918	0.000125
0.0625	0.0557492	0.0563405	0.0591034	0.000625	0.0552146	0.0558062	0.0585704	6.25e-05

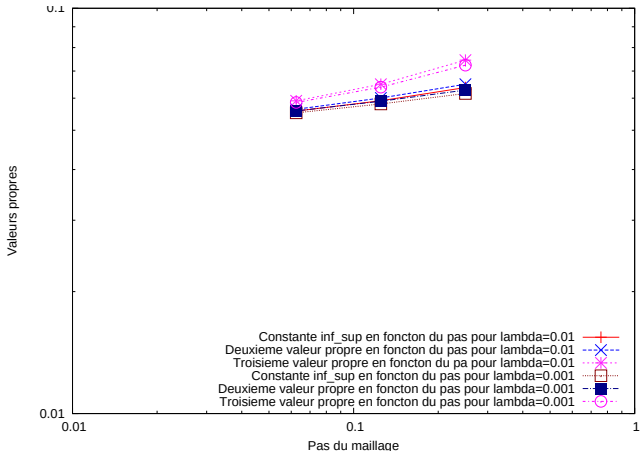
Maillage triangles $\gamma = 1$

	$\lambda = 0.01$				$\lambda = 0.001$			
h	vp_1	vp_2	vp_3	vp_h	vp_1	vp_2	vp_3	vp_h
0.25	0.0742964	0.0743324	0.0764459	0.0025	0.0721876	0.0722237	0.0743407	0.00025
0.125	0.0649713	0.0651659	0.0663787	0.00125	0.0639099	0.0641047	0.0653186	0.000125
0.0625	0.0592191	0.0594166	0.0601436	0.000625	0.0586861	0.0588837	0.059611	6.25e-05

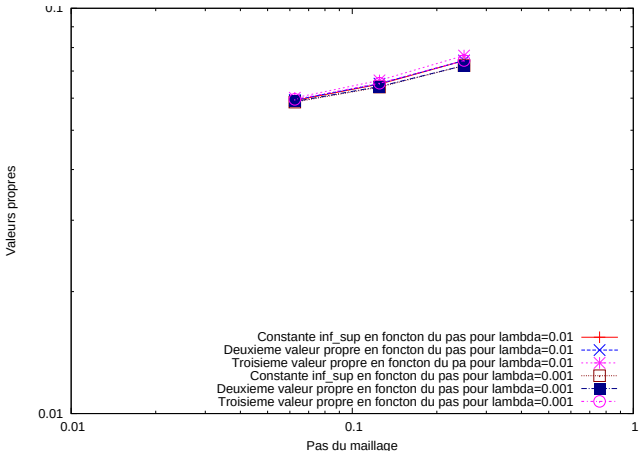
Maillage triangles centres de de gravité $\gamma = 1$

	$\lambda = 0.01$				$\lambda = 0.001$			
h	vp_1	vp_2	vp_3	vp_h	vp_1	vp_2	vp_3	vp_h
0.25	0.0106025	0.0748489	0.0756145	0.0025	0.00837041	0.072741	0.073508	0.00025
0.125	0.00527679	0.0659056	0.0664308	0.00125	0.00415629	0.064845	0.0653707	0.000125
0.0625	0.00264341	0.0600573	0.0601663	0.000625	0.00208205	0.0595247	0.0596337	6.25e-05

Maillage rectangle bidomaine $\gamma = 1$



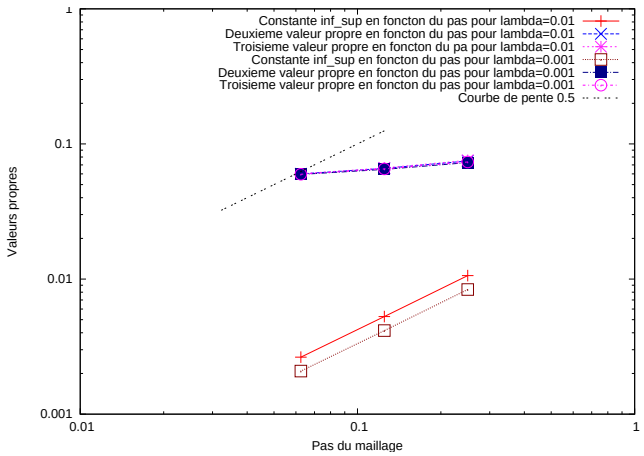
Maillage triangles $\gamma = 1$



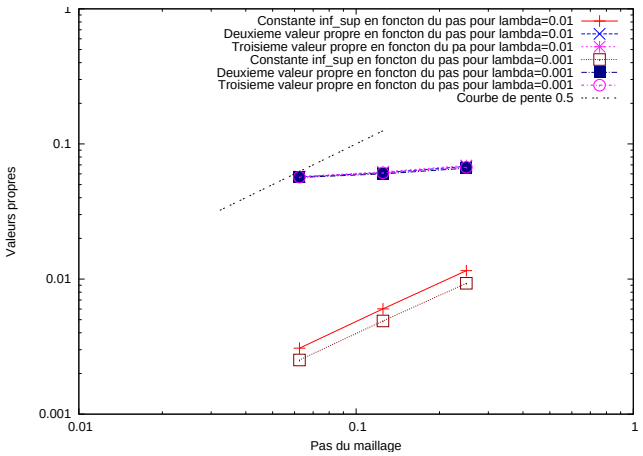
Maillage triangles centres de gravité $\gamma = 1$

	$\lambda = 0.01$				$\lambda = 0.001$			
h	vp_1	vp_2	vp_3	vp_h	vp_1	vp_2	vp_3	vp_h
0.25	0.0115343	0.0682501	0.0692971	0.0025	0.00930428	0.0661308	0.0671796	0.00025
0.125	0.00601736	0.0611335	0.062135	0.00125	0.00489767	0.0600687	0.0610711	0.000125
0.0625	0.00307783	0.0569924	0.0573534	0.000625	0.0025167	0.0564584	0.0568195	6.25e-05

Maillage localement raffiné $\gamma = 1$



Maillage rectangle bidomaine $\gamma = 1$



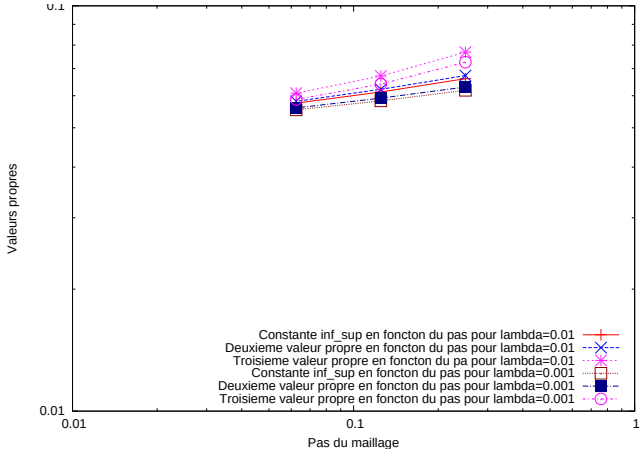
Maillage localement raffiné $\gamma = 1$

	$\lambda = 0.01$				$\lambda = 0.001$			
h	vp_1	vp_2	vp_3	vp_h	vp_1	vp_2	vp_3	vp_h
0.25	0.0661224	0.0673011	0.0768364	0.005	0.0618673	0.0630502	0.0726189	0.0005
0.125	0.0613077	0.0622212	0.06711	0.00353553	0.0582905	0.0592063	0.0641074	0.000353553
0.0625	0.0575313	0.0581217	0.0608805	0.0025	0.0553928	0.0559843	0.0587481	0.00025

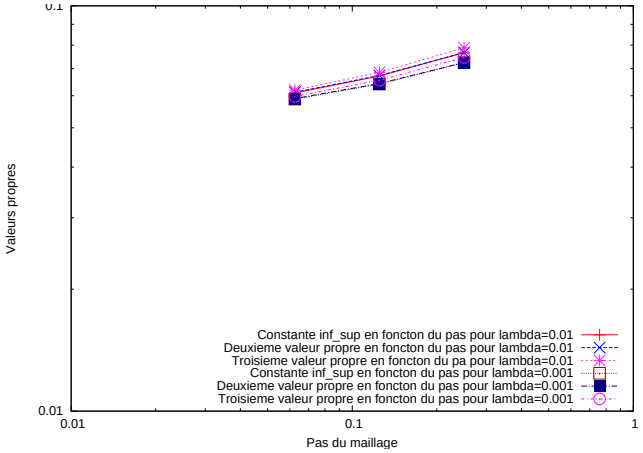
Maillage triangles $\gamma = 0.5$

	$\lambda = 0.01$				$\lambda = 0.001$			
h	vp_1	vp_2	vp_3	vp_h	vp_1	vp_2	vp_3	vp_h
0.25	0.0766401	0.0766761	0.0787855	0.005	0.0724219	0.072458	0.0745746	0.0005
0.125	0.067128	0.0673223	0.068533	0.00353553	0.0641255	0.0643203	0.0655339	0.000353553
0.0625	0.060996	0.0611932	0.0619191	0.0025	0.0588637	0.0590613	0.0597886	0.00025

Maillage triangles centres de de gravité $\gamma = 0.5$



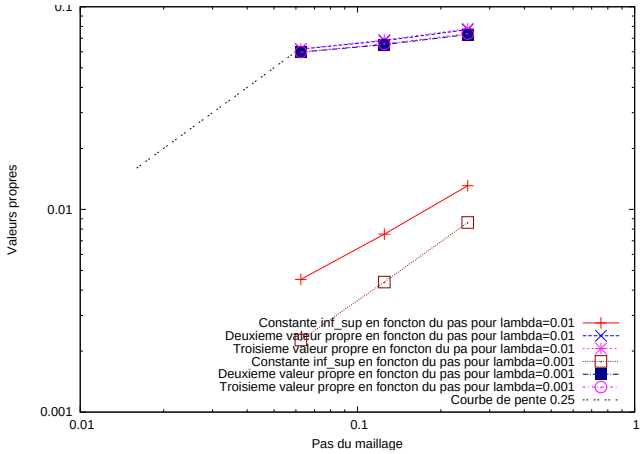
Maillage triangles $\gamma = 0.5$



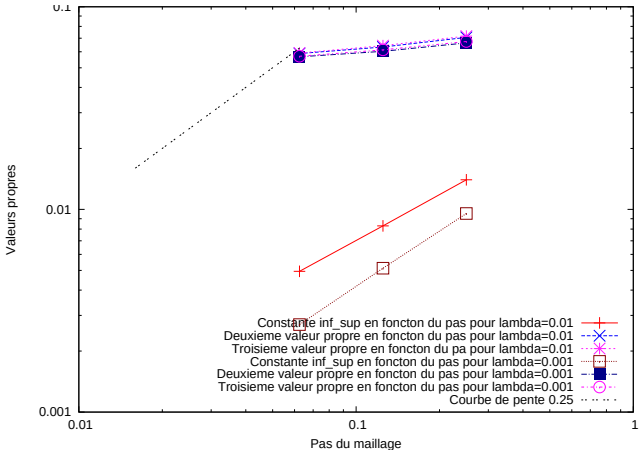
Maillage triangles centres de gravité $\gamma = 0.5$

	$\lambda = 0.01$				$\lambda = 0.001$			
h	vp_1	vp_2	vp_3	vp_h	vp_1	vp_2	vp_3	vp_h
0.25	0.0130826	0.0771915	0.0779557	0.005	0.00861841	0.0729752	0.073742	0.0005
0.125	0.00755323	0.0680606	0.068585	0.00353553	0.00438393	0.0650604	0.065586	0.000353553
0.0625	0.00451465	0.061833	0.0619417	0.0025	0.00226917	0.0597023	0.0598112	0.00025

Maillage rectangle bidomaine $\gamma = 0.5$



Maillage rectangle bidomaine $\gamma = 0.5$



Maillage localement raffiné $\gamma = 0.5$

	$\lambda = 0.01$				$\lambda = 0.001$			
h	vp_1	vp_2	vp_3	vp_h	vp_1	vp_2	vp_3	vp_h
0.25	0.0140122	0.0706054	0.0716504	0.005	0.00955206	0.0663662	0.0674149	0.0005
0.125	0.00829214	0.0632972	0.0642969	0.00353553	0.00512514	0.060285	0.0612872	0.000353553
0.0625	0.00494826	0.0587727	0.059133	0.0025	0.00270374	0.0566364	0.0569975	0.00025

Maillage localement raffiné $\gamma = 0.5$

Conclusion

Ces résulats sont donc en accord avec l'estimation d'erreur du théorème 4.2 lorsque $\gamma = 1$ la constante Inf-Sup se comporte en $h^{\frac{1}{2}}$ et en $h^{\frac{1}{4}}$ lorsque $\gamma = 0.5$ dans les cas où elle tend vers 0.
On a donc pas réellement d'amélioration de la constante Inf-Sup lorsque celle-ci tendait vers 0. Ceci suggère alors qu'il faudrait plutôt s'intéresser à d'autres types de stabilisation.

IV.3 Stabilisation de type Brezzi-Pitkäranta

Dans cette section nous avons seulement eu le temps de faire une petite étude de ce type de stabilisation et nous n'avons pas, pour le moment, de résultats numériques.

Le problème discret

Définition 4.1
On définit un opérateur du second ordre discret, noté $\Delta^{\mathfrak{D}} : p^{\mathfrak{D}} \in \mathbb{R}^{\mathfrak{D}} \mapsto \Delta^{\mathfrak{D}} p^{\mathfrak{D}} \in \mathbb{R}^{\mathfrak{D}}$, *comme suit :*

$$\Delta^{\mathcal{D}} p^{\mathfrak{D}} = \frac{1}{|\mathcal{D}|} \sum_{\mathfrak{s}=\mathcal{D}|\mathcal{D}' \in \mathcal{E}_{\mathcal{D}}} \frac{d_{\mathcal{D}}^2 + d_{\mathcal{D}'}^2}{d_{\mathcal{D}}^2} (p^{\mathcal{D}'} - p^{\mathcal{D}}), \forall \mathcal{D} \in \mathfrak{D}.$$

Ainsi, le problème discret devient :

$$\begin{cases} \text{Trouver } u^{\mathcal{T}} \in (\mathbb{R}^2)^{\mathcal{T}} \text{ et } p^{\mathfrak{D}} \in \mathbb{R}^{\mathfrak{D}} \text{ tels que,} \\ \forall \mathcal{K} \in \mathfrak{M}, \sum_{\mathcal{D}_{\sigma,\sigma^*} \in \mathfrak{D}_{\mathcal{K}}} F_{\mathcal{K},\mathcal{D}_{\sigma,\sigma^*}}(u^{\mathcal{T}}, p^{\mathfrak{D}}) = |\mathcal{K}| f_{\mathcal{K}}, \\ \forall \mathcal{K}^* \in \mathfrak{M}^* \cup \partial\mathfrak{M}^*, \sum_{\mathcal{D}_{\sigma,\sigma^*} \in \mathfrak{D}_{\mathcal{K}^*}} F_{\mathcal{K}^*,\mathcal{D}_{\sigma,\sigma^*}}(u^{\mathcal{T}}, p^{\mathfrak{D}}) = |\mathcal{K}^*| f_{\mathcal{K}^*}, \\ \forall \mathcal{D} \in \mathfrak{D}, \ \mathrm{div}^{\mathcal{D}} u^{\mathcal{T}} - \lambda d_{\mathcal{D}}^2 \Delta^{\mathcal{D}} p^{\mathfrak{D}} = 0, \\ \sum_{\mathcal{D} \in \mathfrak{D}} |\mathcal{D}| p^{\mathcal{D}} = 0. \end{cases} \tag{IV.9}$$

On a alors (voir [Kre10, Th. V.13 p122]) une meilleure estimation d'erreur que la précédente :

Théorème 4.3
On suppose que la solution (u, p) *du problème de Stokes appartient à* $(H^2(\Omega))^2 \times H^1(\Omega)$. *On suppose que* $\lambda > 0$.
Soit $(u^{\mathcal{T}}, p^{\mathfrak{D}}) \in (\mathbb{R}^2)^{\mathcal{T}} \times \mathbb{R}^{\mathfrak{D}}$ *la solution du schéma DDFV stabilisé* (IV.9). *Alors, il existe une constante* $C > 0$ *dépendant uniquement de* Ω, $\mathcal{T}$, λ, $||u||_{H^2}$ *et* $||p||_H^1$, *telle que :*

$$||\nabla u - \nabla^{\mathfrak{D}} u^{\mathcal{T}}||_2 \leq C size(\mathcal{T}),$$

et

$$||p - p^{\mathfrak{D}}||_2 \leq C size(\mathcal{T}).$$

Problème sous forme matricielle

Le terme de stabilisation revient à rajouter une matrice $S \in \mathcal{M}_{N_p,N_p}$ dans notre système, c'est à dire que l'on résoud maintenant le système matriciel suivant :

$$\text{Trouver } U \text{ et } P \text{ tels que : } \begin{cases} RU + B^T P = F \\ BU - SP = 0 \end{cases}.$$

Pour construire cette matrice utilisons le fait que :

$$-\lambda d_{\mathcal{D}}^2 \Delta^{\mathcal{D}} p^{\mathfrak{D}} = \frac{2\lambda size(\mathcal{T})^2}{|\mathcal{D}|} \sum_{\substack{\mathcal{D}' \in \mathfrak{D} \\ \mathcal{D} \cap \mathcal{D}' \neq \emptyset}} (p^{\mathcal{D}} - p^{\mathcal{D}'}).$$

Sachant que chaque diamant a 2 voisins s'il est sur le bord et 4 sinon, on construit la matrice S comme suit.
Pour un diamant $\mathcal{D}$ donné, appelons $\mathcal{D}_1$, $\mathcal{D}_2$ ses diamants toujours voisins et $\mathcal{D}_3$, $\mathcal{D}_4$ les deux autres s'il n'est pas sur le bord.
On parcourt là encore le maillage par arête (donc par diamamnt), lorque l'on est sur la ligne correspondant à l'arête $\mathcal{D}$, on ajoute :

- $2(2\lambda size(\mathcal{T})^2)$ à la colonne correspondant au diamant $\mathcal{D}$ si celui est sur le bord,
- $4(2\lambda size(\mathcal{T})^2)$ à la colonne correspondant au diamant $\mathcal{D}$ si celui n'est pas sur le bord,
- $-(2\lambda size(\mathcal{T})^2)$ à la colonne correspondant au diamant $\mathcal{D}_1$,
- $-(2\lambda size(\mathcal{T})^2)$ à la colonne correspondant au diamant $\mathcal{D}_2$,
- $-(2\lambda size(\mathcal{T})^2)$ à la colonne correspondant au diamant $\mathcal{D}_3$ si $\mathcal{D}$ sur le bord,
- $-(2\lambda size(\mathcal{T})^2)$ à la colonne correspondant au diamant $\mathcal{D}_4$ si $\mathcal{D}$ sur le bord.

Ce qui nous intéresse c'est de voir comment varie la constante Inf-Sup lorque l'on modifie λ.

Conclusion et perspectives

Ces tests numériques nous ont donc permis de réfuter certaines hypothèses et de poser certaines conjectures.

– Dans le cas des éléments finis standards :
  • L'élément fini $\mathbb{P}_1$bulle$/\mathbb{P}_1$ ne satisfait clairement pas la condition Inf-Sup uniforme lorsque l'on raffine non uniformément le maillage. De plus, on remarque que lorsque la constante Inf-Sup tend vers 0 c'est également le cas des deux suivantes. Il semble donc difficile de remédier à ceci.
  • Pour les éléments finis $\mathbb{P}_2/\mathbb{P}_1$ et $\mathbb{P}_1$iso$\mathbb{P}_2/\mathbb{P}_1$, les différents tests effectués semblent montrer que la constante Inf-Sup est uniformément minorée même lors des raffinements non uniforme.
  Il faudrait maintenant essayer de prouver ce résultat.

– En ce qui concerne les éléments finis multiniveaux et le raffinement CHARMS, on peut conjecturer le fait que la stabilité de la constante Inf-Sup est robuste à cette stratégie de raffinement.
  En effet :
  • quelles que soient les zones que l'on a décidé de raffiner,
  • quel que soit le pas du maillage initial utilisé,
  • quelle que soit la taille de la zone que l'on raffine,
  • que l'on utilise ou pas la règle "au-plus-un-niveau-de-difference" ;
  la constante Inf-Sup était bien uniformément minorée.

– Dans le cas du schéma DDFV, on a montré qu'il existe des maillages dont la constante Inf-Sup tend vers 0 alors que le schéma converge. Or, dans ces cas là, les valeurs propres suivantes (contrairement aux cas $\mathbb{P}_1$bulle$/\mathbb{P}_1$ ou $\mathbb{Q}_1/\mathbb{P}_0$) semblent uniformément minorées ce qui peut, peut-être, expliquer le bon comportement du schéma malgré tout.
  Dans ce cadre là, il resterait à étudier de manière plus approfondie le cas de la stabilisation. Tout d'abord, dans le cas d'un terme de stabilisation proportionnel à la pression (ce que nous avons commencé à regarder) puis dans le cas de la stabilisation de type Brezzi-Pitkäranta. On pourrait ainsi regarder l'influence des différents paramètres de stabilisation dans l'évolution de la constante Inf-Sup pour les maillages où l'on a vu qu'elle tendait vers 0.

Bibliographie

[BEH04] Philippe Blanc, Robert Eymard, and Raphaèle Herbin, *An error estimate for finite volume methods for the stokes equations on equilateral triangular meshes*, Num. Meth. P.D.E **20** (2004), 907–918.

[EG04] A. Ern and J.-L. Guermond, *Theory and practice of finite elements*, Applied Mathematical Sciences, vol. 159, Springer-Verlag, New York, 2004. MR MR2050138 (2005d :65002)

[EGH00] Robert Eymard, Thierry Gallouët, and Raphaèle Herbin, *Finite volume methods*, Handb. Numer. Anal., VII, North-Holland, Amsterdam, 2000.

[Kre10] Stella Krell, *Schémas volumes finis en mécanique des fluides complexes*, Ph.D. thesis, Université de Provence, Marseille, France, 2010, http ://tel.archives-ouvertes.fr/tel-00524509/fr/.

[LT94] P. Lascaux and R. Théodor, *Analyse numérique matricielle appliquée à l'art de l'ingénieur. Tome 2*, second ed., Masson, Paris, 1994, Méthodes itératives. [Iterative methods]. MR 1286939 (95f :65002)

[Min10] Sebastian Minjeaud, *Raffinement local adaptatif et méthodes multiniveaux pour la simulation d'écoulements multiphasiques*, Ph.D. thesis, Université de Provence, Marseille, France, 2010, http ://tel.archives-ouvertes.fr/tel-00535892/fr/.

[Saa92] Youcef Saad, *Numerical methods for large eigenvalue problems*, Algorithms and Architectures for Advanced Scientific Computing, Manchester University Press, Manchester, 1992. MR 1177405 (93h :65052)

Annexe A

Listing des programmes

Programme sortie_matrices_M_R_B.edp

```
/////////////////
//// Donnees ////
/////////////////

include "donnees.edp";
include "espace_Xh_Mh.edp";



//////////////////////////
//// Pas du maillage ////
//////////////////////////

fespace Ph(Th,P0);
Ph diam=hTriangle;
cout <<"pas du maillage=" << diam[].max << endl;



/////////////////
//// Espaces ////
/////////////////

Xh [u1h,u2h],[v2h,v1h];

Mh ph,qh;


//////////////////////////////
//// Construction matrices ////
//////////////////////////////


// matrice de masse //

varf mat1(ph,qh)=int2d(Th)(ph*qh);
matrix<real> M=mat1(Mh,Mh);

// matrice de rigidite //

varf mat2([u1h,u2h],[v1h,v2h])=int2d(Th)(dx(u1h)*dx(v1h)+dy(u1h)*dy(v1h)+dx(u2h)
*dx(v2h)+dy(u2h)*dy(v2h))+on(1,2,3,4,u1h=1,u2h=1);
// on doit mettre 1 comme CL pour pouvoir enlever les dl du bord

matrix<real> R=mat2(Xh,Xh);

// matrice B //

varf mat3([u1h,u2h],[ph,qh])=int2d(Th)(-ph*(dx(u1h)+dy(u2h)));

matrix<real> B=mat3(Xh,Mh);



//////////////////////////////////
//// Enregistrement matrices ////
//////////////////////////////////

 {
  ofstream file(matricemasse);
  file << M << endl;
 }

 {
  ofstream file(matricerigidite);
```

Programme sortie_matrices_M_R_B.edp

```
  file << R << endl;
 }

 {
  ofstream file(matricedivergence);
  file << B << endl;
 }


////////////////////////////////////
//// Degres de liberte du bord ////
////////////////////////////////////

real[int] ddl = mat2(0,Xh);

{
  ofstream file(dlaubord);
  file << ddl << endl;
 }



//////////////////////////////
//// Affichage des donnees ////
//////////////////////////////


{
ofstream file(maillage);

file << "Pas du maillage " << endl;
file << diam[].max << endl;

file << "Nombre d'elements en vitesse "<< endl;
file << Xh.nt << endl;

file << "Nombre de degres de liberte en vitesse "<< endl;
file << Xh.ndof <<endl;

file << "Nombre d'elements en pression "<< endl;
file << Mh.nt <<endl;

file << "Nombre de degres de liberte en pression "<< endl;
file <<  Mh.ndof <<endl;
}
```

Programme schema_EF.sce

```
////////////////////////////////////////////////////////
/////                                              ///////
/////             Schema de elements finis         ///////
/////                pour pb Stokes 2D             ///////
/////                                              ///////
/////                                              ///////
/////                                              ///////
////////////////////////////////////////////////////////



//*************************************************************************
// On efface toutes les variables et on charge les fonctions utiles
//*************************************************************************


clear;
clc;
lines(0);
stacksize('max');
warning('off');

exec 'sortie_matrices.sce';
exec 'suppression_dl_au_bord.sce';
exec 'valeur_propre_puissance_inverse.sce';
exec 'valeur_propre_QR.sce';
exec 'decompositionQR.sci';

//*************************************************************************
// Saisie par l'utilisateur des donnees du calcul
//*************************************************************************


// Choix des espaces

printf('=======================================\n');
printf('Choix de l element fini :\n');
printf('1) P2_P1\n');
printf('2) P1b_P1\n');
printf('3) P1isoP2_P1\n');
choix_element_fini=-1;

while ((choix_element_fini<1) | (choix_element_fini>3))
    choix_element_fini=input("Faites votre choix :");
end;

select choix_element_fini

case 1 then
    fic=mopen('espace_Xh_Mh.edp','w');
    mfprintf(fic,'fespace Xh(Th,[P2,P2]);\n fespace Mh(Th,P1);\n');
    mclose(fic);

    nom_espace='P2_P1';
    choix_espace=sprintf('string espace=%s;','""P2_P1""');

case 2 then
    fic=mopen('espace_Xh_Mh.edp','w');
    mfprintf(fic,'fespace Xh(Th,[P1b,P1b]);\n fespace Mh(Th,P1);\n');
    mclose(fic);

    nom_espace='P1b_P1';
    choix_espace=sprintf('string espace=%s;','""P1b_P1""');

case 3 then
    fic=mopen('espace_Xh_Mh.edp','w');
```

Programme schema_EF.sce

```
    mfprintf(fic,'mesh Th1=trunc(Th,1,split=2);\n fespace Xh(Th1,[P1,P1]);\n fes
pace Mh(Th,P1);\n');
    mclose(fic);

    nom_espace='P1isoP2_P1';
    choix_espace=sprintf('string espace=%s;','""P1isoP2_P1""');

end


// Choix du maillage

printf('=======================================\n');
printf('Choix du maillage :\n');
printf('1) Carre uniforme \n');
printf('2) Carre non uniforme \n');
printf('3) Cercle non uniforme \n');
printf('4) Couronne non uniforme \n');

choix_maillage=-1;
while ((choix_maillage<1) | (choix_maillage>4))
    choix_maillage=input("Faites votre choix :");
end;


// Choix de raffinement

printf('=======================================\n');
printf('Choix du raffinement minimum :\n');
kmin=-1;
while (int(kmin)<1)
    kmin=input("Faites votre choix :");
end;

printf('=======================================\n');
printf('Choix du raffinement maximum :\n');
kmax=-1;
while (int(kmax)<1)
    kmax=input("Faites votre choix :");
end;

// Choix de la methode utilisee

printf('=======================================\n');
printf('Choix de la fonction utiliser :\n');
printf('1) Methode de la puissance inverse \n');
printf('2) Methode QR \n');

choix_methode=-1;
while ((choix_methode<1) | (choix_methode>2))
    choix_methode=input("Faites votre choix :");
end;

select choix_methode
case 1 then
    nom_methode='methode_puissance_inverse';
case 2 then
    nom_methode='methode_QR';
end

Donnees=zeros(kmax,5);

for k=kmin:kmax

    select choix_maillage
```

Programme schema_EF.sce

```
    case 1 then
        donnees_maillage=sprintf('include ""raffinement_uniforme_carre.edp"";\n'
);

        donnees_vp=sprintf('donnee_trace_vp_maillage_unif_carre_%s_%s_%g.dat',no
m_espace,nom_methode,k);
        valeur_vp=sprintf('valeur_vp_maillage_unif_carre_%s_%s_de_%g_a_%g.dat',n
om_espace,nom_methode,kmin,kmax);
        fic3=mopen('donnees_maillage.dat','w');
        mfprintf(fic3,'maillage=''donnee_trace_vp_maillage_unif_carre_%s_%s_%g.d
at'';\n k=%g;\n nom_trace=''trace_maillage_carre_unif_%s_%s_%g.eps''',nom_espace
,nom_methode,k,k,nom_espace,nom_methode,k);
        mclose(fic3);
        nom_trace=sprintf('Valeurs_propres_en_fonction_du_pas_maillage_unif_carr
e_%s',nom_espace);
        maillage=sprintf('donnee_maillage_carre_unif_%s_%g.dat',nom_espace,k);

        matrice_masse=sprintf('matrice_masse_carre_unif_%s_%g.dat',nom_espace,k)
;
        matrice_rigidite=sprintf('matrice_rigidite_carre_unif_%s_%g.dat',nom_esp
ace,k);
        matrice_divergence=sprintf('matrice_divergence_carre_unif_%s_%g.dat',nom
_espace,k);
        nom_DL=sprintf('dl_au_bord_carre_unif_%s_%g.dat',nom_espace,k);


    case 2 then
        donnees_maillage=sprintf('include ""raffinement_non_uniforme_carre.edp""
;\n');

        donnees_vp=sprintf('donnee_trace_vp_maillage_non_unif_carre_%s_%s_%g.dat
',nom_espace,nom_methode,k);
        valeur_vp=sprintf('valeur_vp_maillage_non_unif_carre_%s_%s_de_%g_a_%g.da
t',nom_espace,nom_methode,kmin,kmax);
        fic3=mopen('donnees_maillage.dat','w');
        mfprintf(fic3,'maillage=''donnee_trace_vp_maillage_non_unif_carre_%s_%s_
%g.dat'';\n k=%g;\n nom_trace=''trace_maillage_carre_non_unif_%s_%s_%g.eps''',no
m_espace,nom_methode,k,k,nom_espace,nom_methode,k);
        mclose(fic3);
        nom_trace=sprintf('Valeurs_propres_en_fonction_du_pas_maillage_non_unif_
carre_%s',nom_espace);
        maillage=sprintf('donnee_maillage_carre_non_unif_%s_%g.dat',nom_espace,k
);

        matrice_masse=sprintf('matrice_masse_carre_non_unif_%s_%g.dat',nom_espac
e,k);
        matrice_rigidite=sprintf('matrice_rigidite_carre_non_unif_%s_%g.dat',nom
_espace,k);
        matrice_divergence=sprintf('matrice_divergence_carre_non_unif_%s_%g.dat'
,nom_espace,k);
        nom_DL=sprintf('dl_au_bord_carre_non_unif_%s_%g.dat',nom_espace,k);


    case 3 then
        donnees_maillage=sprintf('include ""raffinement_non_uniforme_cercle.edp"
";\n');

        donnees_vp=sprintf('donnee_trace_vp_maillage_non_unif_cercle_%s_%s_%g.da
t',nom_espace,nom_methode,k);
        valeur_vp=sprintf('valeur_vp_maillage_non_unif_cercle_%s_%s_de_%g_a_%g.d
at',nom_espace,nom_methode,kmin,kmax);
        fic3=mopen('donnees_maillage.dat','w');
        mfprintf(fic3,'maillage=''donnee_trace_vp_maillage_non_unif_cercle_%s_%s
_%g.dat'';\n k=%g;\n nom_trace=''trace_maillage_cercle_non_unif_%s_%s_%g.eps''',
```

Programme schema_EF.sce

```
nom_espace,nom_methode,k,k,nom_espace,nom_methode,k);
        mclose(fic3);
        nom_trace=sprintf('Valeurs_propres_en_fonction_du_pas_maillage_non_unif_
cercle_%s',nom_espace);
        maillage=sprintf('donnee_maillage_cercle_non_unif_%s_%g.dat',nom_espace,
k);

        matrice_masse=sprintf('matrice_masse_cercle_non_unif_%s_%g.dat',nom_espa
ce,k);
        matrice_rigidite=sprintf('matrice_rigidite_cercle_non_unif_%s_%g.dat',no
m_espace,k);
        matrice_divergence=sprintf('matrice_divergence_cercle_non_unif_%s_%g.dat
',nom_espace,k);
        nom_DL=sprintf('dl_au_bord_cercle_non_unif_%s_%g.dat',nom_espace,k);

    case 4 then
        donnees_maillage=sprintf('include ""raffinement_non_uniforme_couronne.ed
p"";\n');

        donnees_vp=sprintf('donnee_trace_vp_maillage_non_unif_couronne_%s_%s_%g.
dat',nom_espace,nom_methode,k);
        valeur_vp=sprintf('valeur_vp_maillage_non_unif_couronne_%s_%s_de_%g_a_%g
.dat',nom_espace,nom_methode,kmin,kmax);
        fic3=mopen('donnees_maillage.dat','w');
        mfprintf(fic3,'maillage=''donnee_trace_vp_maillage_non_unif_couronne_%s_
%s_%g.dat'';\n k=%g;\n nom_trace=''trace_maillage_couronne_non_unif_%s_%s_%g.eps
''',nom_espace,nom_methode,k,k,nom_espace,nom_methode,k);
        mclose(fic3);
        nom_trace=sprintf('Valeurs_propres_en_fonction_du_pas_maillage_non_unif_
couronne_%s',nom_espace);
        maillage=sprintf('donnee_maillage_couronne_non_unif_%s_%g.dat',nom_espac
e,k);

        matrice_masse=sprintf('matrice_masse_couronne_non_unif_%s_%g.dat',nom_es
pace,k);
        matrice_rigidite=sprintf('matrice_rigidite_couronne_non_unif_%s_%g.dat',
nom_espace,k);
        matrice_divergence=sprintf('matrice_divergence_couronne_non_unif_%s_%g.d
at',nom_espace,k);
        nom_DL=sprintf('dl_au_bord_couronne_non_unif_%s_%g.dat',nom_espace,k);

    end



    //*************************************************************************
    // Recuperation des matrices
    //*************************************************************************


    tic();

    // Taille du maillage

    fic1=mopen('taille.edp','w');
    mfprintf(fic1,'int k=%g;\n',k);
    mclose(fic1);


    // Maillage choisi

    fic2=mopen('donnees.edp','w');
    mfprintf(fic2,donnees_maillage);
    mclose(fic2);
```

Programme schema_EF.sce

```
    fic2prime=mopen('choix_maillage.edp','w');
    mfprintf(fic2prime,choix_espace);
    mclose(fic2prime);


    // FreeFem

    [x,ierr] = fileinfo('DL_au_bord/'+nom_DL);

    if size(x)==0
        unix_s('FreeFem++ sortie_matrices_M_R_B.edp');
        unix_s('mv '+matrice_masse+' Matrice_masse/');
        unix_s('mv '+matrice_rigidite+' Matrice_rigidite/');
        unix_s('mv '+matrice_divergence+' Matrice_divergence/');
        unix_s('mv '+nom_DL+' DL_au_bord/');
        unix_s('mv '+maillage+' donnees_maillage_EF/');
    end


    // Enregistrement matrices

    [Mat_Masse,Mat_Rigidite,Mat_Div]=matrices_M_R_B(k,matrice_masse,matrice_rigi
dite,matrice_divergence);

    t1=toc();



    //***************************************************************************
    // Modification des matrices
    //***************************************************************************


    tic();
    // Suppression des degres de liberte du bord

    [Mat_Rigidite,Mat_Div,J]=suppression_dl_au_bord_R_B(k,nom_DL,Mat_Rigidite,Ma
t_Div);

    // Supression des pressions constantes

    b=size(Mat_Div);
    Mat_Div_tild=Mat_Div(1:b(1)-1,:);
    btild=size(Mat_Div_tild);



    //***************************************************************************
    // Calcul des valeurs propres
    //***************************************************************************

    // Choix du trace residu et lambda en fonction iterations

    printf('=========================================\n');
    printf('Choix de tracer \lambda et rÃ©sidu en fonction itÃ©rations :\n');
    printf('1) Oui \n');
    printf('2) Non \n');

    choix_trace=-1;
    while ((choix_trace<1) | (choix_trace>2))
        choix_trace=input("Faites votre choix :");
    end;

    // Donnees du maillage
```

Programme schema_EF.sce

```
    fic5=mopen('donnees_maillage_EF/'+maillage,'r');
    Donnees1=mgetl(fic5);
    mclose(fic5);


    for l=2:2:10
        Donnees(k,(l/2))=evstr(Donnees1(l));
    end

    vecteur_pas(k)=Donnees(k,1);

    printf('=========================================\n');
    printf('k=%g pas du maillage=%g \n',k,Donnees(k,1));
    printf('Nb elements vitesse=%g Nb ddl vitesse =%g \nNb elements pression=%g
Nb ddl pression=%g\n',Donnees(k,2),Donnees(k,3),Donnees(k,4),Donnees(k,5));

    if choix_methode==1 then
        [c_inf_sup,vp_2,vp_3,it1,it2,it3,vpropre]=valeur_propre_puissance_invers
e(k,Mat_Rigidite,Mat_Div_tild,Mat_Masse);

        // Nombre d iterations

        vecteur_it1(k)=it1;
        vecteur_it2(k)=it2;
        vecteur_it3(k)=it3;

        printf('Nb iterations=%g  cte inf_sup=%8.5e \n' ,it1,c_inf_sup);
        printf('Nb iterations=%g  Deuxieme valeur propre=%8.5e \n' ,it2,vp_2);
        printf('Nb iterations=%g  Troisieme valeur propre=%8.5e \n' ,it3,vp_3);

    else
        [c_inf_sup,vp_2,vp_3,it,vpropre]=valeur_propre_QR(k,Mat_Rigidite,Mat_Div
_tild,Mat_Masse);
        vecteur_it(k)=it;

        printf('Nb iterations=%g\ncte inf_sup=%8.5e \n' ,it,c_inf_sup);
        printf('Deuxieme valeur propre=%8.5e \n' ,vp_2);
        printf('Troisieme valeur propre=%8.5e \n' ,vp_3);

    end
    t2=toc();
    // Valeur des valeurs propres

    vecteur_c_inf_sup(k)=c_inf_sup;
    vecteur_vp2(k)=vp_2;
    vecteur_vp3(k)=vp_3;

    if choix_trace==1

        // Fichier contenant valeur de la constante inf-sup au carre

        valeur=sprintf('valeur_lambda_%g.dat',k);
        fic4=mopen(valeur,'w');
        mfprintf(fic4,'lambda=.17%g;\n',c_inf_sup^2);
        mclose(fic4);


        // Gnuplot

        unix_s('gnuplot trace_residu_EF.gp');

    end

end
```

Programme schema_EF.sce

```
//************************************************************************
// Mode propre
//************************************************************************


printf('=======================================\n');
printf('Choix de tracer le mode propre :\n');
printf('1) Oui \n');
printf('2) Non \n');

choix_mode=-1;
while ((choix_mode<1) | (choix_mode>2))
    choix_mode=input("Faites votre choix :");
end;


if choix_mode==1

    P=(full(Mat_Masse))^(-1/2)*vpropre;

    deletefile('vect_propre.dat');
    write('vect_propre.dat',[length(P);real(P)]);

    unix_s('FreeFem++ trace_mode_propre.edp');

end



//************************************************************************
// Affichage des donnees
//************************************************************************


// Temps de calcul

printf('=======================================\n');
printf(' temps enregistrement matrice=%g\n temps calcul algorithme avec astuce=%
g\n',t1 , t2);




//************************************************************************
// Trace
//************************************************************************

// Construction du fichier vecteurs a trace //
if choix_methode==1 then

    fichier=mopen(valeur_vp,'w');
    mfprintf(fichier,'%g & \t %g & \t %g & \t %g & \t %g & \t %g \\ \n',
 vecteur_pas,vecteur_c_inf_sup,vecteur_vp2,vecteur_vp3,int(vecteur_it1),int(vect
eur_it2),int(vecteur_it3));
    mclose(fichier);


else

    fichier=mopen(valeur_vp,'w');
    mfprintf(fichier,'%g & \t %g & \t %g & \t %g & \t %g \\ \n', vecteur_pas , v
ecteur_c_inf_sup , vecteur_vp2 , vecteur_vp3 , int(vecteur_it));
    mclose(fichier);
```

Programme schema_EF.sce

```
end


printf('=======================================\n');
printf('Choix de tracer valeur propre en fonction du pas :\n');
printf('1) Oui \n');
printf('2) Non \n');

choix_trace_vp=-1;
while ((choix_trace_vp<1) | (choix_trace_vp>2))
    choix_trace_vp=input("Faites votre choix :");
end;

if choix_trace_vp==1

    fic=mopen('donnees_maillage_EF.dat','w');
    mfprintf(fic,'maillage=''%s''\n nom_trace=''%s_%s_de_%g_a_%g.eps''',valeur_v
p,nom_trace,nom_methode,kmin,kmax);
    mclose(fic);

    unix_s('gnuplot trace_vp_EF.gp');
end



printf('==============  FIN  ==================\n');

//************************************************************************
//          FIN
//************************************************************************
```

Programme construction_VF_Stokes.sce

```
// Toutes ces fonctions prennent deux parametres :
//      m        : un maillage
//      donnees  : un jeu de donnees
//


//***************************************************************************
//    Le schema VF4
//    programme a l'aide d'operations vectorielles pour des raisons d'efficacite

//    Prise en compte des CL de Dirichlet non-homogene (_LABEL=-1)
//                        des CL de Neumann homogene  (_LABEL<-1)
//
//***************************************************************************

function [Mat_Rigidite,Mat_Div,Mat_Masse,F]=const_schema_VF4(m,donnees)

    printf("Assemblage VF4\n");

    // On evalue le terme source aux centres des mailles

    source_u=eval_fonction_u(m.centres,donnees.source);
    source_u1=source_u(1:m.nb_vol);
    source_u2=source_u(m.nb_vol+1:$);
    source_p=zeros(m.nb_are,1);

    indiA=[];
    indjA=[];
    valA=[];

    indiB_x=[];
    indjB_x=[];
    valB_x=[];

    indiB_y=[];
    indjB_y=[];
    valB_y=[];

    indiM=[];
    valM=[];

    indib=[];
    indjb=[];
    valb1=[];
    valb2=[];

    // Calcul de la transmissivite de chaque arete

    tauKL=m.aretes(:,_MES)./m.aretes(:,_DKL);

    // Calcul des coefficients de la matrice de divergence

    coord_segment_centre_deb=m.sommets(m.aretes(:,_DEB),:)-m.centres(m.aretes(:,
_K),:);

    coeff_x=(m.sommets(m.aretes(:,_FIN),_Y)-m.sommets(m.aretes(:,_DEB),_Y));
    coeff_y=-(m.sommets(m.aretes(:,_FIN),_X)-m.sommets(m.aretes(:,_DEB),_X));

    pr_scal=coord_segment_centre_deb(:,_X).*coeff_x+coord_segment_centre_deb(:,_
Y).*coeff_y;
    v_rectif=find(pr_scal<0);

    rectification=ones(pr_scal);
    rectification(v_rectif)=-1;
    coeff_x=rectification.*coeff_x;
```

Programme construction_VF_Stokes.sce

```
    coeff_y=rectification.*coeff_y;


    // Calcul des coefficients de la matrice de masse

    coeff_M=(m.aretes(:,_DKL).*m.aretes(:,_MES))/2;


    /// Les aretes du bord

    /// Recherche des aretes pour la condition de Dirichlet non homogene
    temp=find(m.aretes(:,_LABEL)==-1);

    // Evaluation des donnees au bord
    ubord=zeros(2*m.aretes(temp,_K));


    indiA=m.aretes(temp,_K);
    indjA=indiA;
    valA=tauKL(temp);

    indiB_x=temp';
    indjB_x=indiA;
    valB_x=coeff_x(temp);

    indiB_y=temp';
    indjB_y=indiA;
    valB_y=coeff_y(temp);

    indiM=temp';
    valM=coeff_M(temp);

    indib=[indib; m.aretes(temp,_K)];
    valb1=[valb1; tauKL(temp).*ubord];
    valb2=[valb2; tauKL(temp).*ubord];

    /// Toutes les autres aretes du bord sont supposees
    /// correspondre a des CL de Neumann non homogene
    /// Donc elles ne contribuent pas au schema


    /// La partie du terme source qui concerne tout le monde

    indib=[indib; m.aretes(:,_K)];
    valb1=[valb1; (m.aretes(:,_MES_K_DEB)+m.aretes(:,_MES_K_FIN)).*source_u1(m.a
retes(:,_K))];
    valb2=[valb2; (m.aretes(:,_MES_K_DEB)+m.aretes(:,_MES_K_FIN)).*source_u2(m.a
retes(:,_K))];

    /// On traite maintenant les aretes interieures

    temp=find(m.aretes(:,_L)>0);

    indiA=[indiA; m.aretes(temp,_K)];
    indjA=[indjA; m.aretes(temp,_K)];
    valA=[valA; tauKL(temp)];

    indiA=[indiA; m.aretes(temp,_K)];
    indjA=[indjA; m.aretes(temp,_L)];
    valA=[valA; -tauKL(temp)];

    indiA=[indiA; m.aretes(temp,_L)];
    indjA=[indjA; m.aretes(temp,_K)];
    valA=[valA; -tauKL(temp)];
```

Programme construction_VF_Stokes.sce

```
    indiA=[indiA; m.aretes(temp,_L)];
    indjA=[indjA; m.aretes(temp,_L)];
    valA=[valA; tauKL(temp)];


    indiB_x=[indiB_x;temp'];
    indjB_x=[indjB_x;m.aretes(temp,_K)];
    valB_x=[valB_x;coeff_x(temp)];

    indiB_y=[indiB_y;temp'];
    indjB_y=[indjB_y;m.aretes(temp,_K)];
    valB_y=[valB_y;coeff_y(temp)];

    indiB_x=[indiB_x;temp'];
    indjB_x=[indjB_x;m.aretes(temp,_L)];
    valB_x=[valB_x;-coeff_x(temp)];

    indiB_y=[indiB_y;temp'];
    indjB_y=[indjB_y;m.aretes(temp,_L)];
    valB_y=[valB_y;-coeff_y(temp)];

    indiM=[indiM;temp'];
    valM=[valM;coeff_M(temp)];


    indib=[indib; m.aretes(temp,_L)];
    valb1=[valb1; (m.aretes(temp,_MES_L_DEB)+m.aretes(temp,_MES_L_FIN)).*source_
u1(m.aretes(temp,_L))];
    valb2=[valb2; (m.aretes(temp,_MES_L_DEB)+m.aretes(temp,_MES_L_FIN)).*source_
u2(m.aretes(temp,_L))];

    /// Construction proprement dite

    indjb=ones(indib);

    A=sparse([indiA indjA], valA);
    Mat_Rigidite=[A spzeros(A);spzeros(A) A];

    B_x=sparse([indiB_x indjB_x], valB_x);
    B_y=sparse([indiB_y indjB_y], valB_y);
    Mat_Div=[B_x B_y];

    Mat_Masse=sparse([indiM indiM], valM);

    indjb=ones(indib);
    b1=sparse([indib indjb], valb1);
    b2=sparse([indib indjb], valb2);
    F=[b1;b2;source_p];
    F=full(F);


endfunction



//===========================================================================
//    Le schema VF4 version Eymard_Gallouet_Herbin
//    programme a l'aide d'operations vectorielles pour des raisons d'efficacite

//    Prise en compte des CL de Dirichlet non-homogene (_LABEL=-1)
//                      des CL de Neumann homogene  (_LABEL<-1)
//
//===========================================================================
```

Programme construction_VF_Stokes.sce

```
function [Mat_Rigidite,Mat_Div,Mat_Masse,F]=const_schema_VF4_EGH(m,donnees)

    printf("Assemblage VF4 version EGH\n");

    // On evalue le terme source aux centres des mailles

    source_u=eval_fonction_u(m.centres,donnees.source);
    source_u1=source_u(1:m.nb_vol);
    source_u2=source_u(m.nb_vol+1:$);
    source_p=zeros(m.nb_som,1);

    indiA=[];
    indjA=[];
    valA=[];

    indiB_x=[];
    indjB_x=[];
    valB_x=[];

    indiB_y=[];
    indjB_y=[];
    valB_y=[];

    indiM=[];
    indjM=[];
    valM=[];

    indib=[];
    indjb=[];
    valb1=[];
    valb2=[];

    // Calcul de la transmissivite de chaque arete

    tauKL=m.aretes(:,_MES)./m.aretes(:,_DKL);

    // Calcul des coefficients de la matrice de divergence

    coord_segment_centre_deb=m.sommets(m.aretes(:,_DEB),:)-m.centres(m.aretes(:,
_K),:);

    coeff_x=(m.sommets(m.aretes(:,_FIN),_Y)-m.sommets(m.aretes(:,_DEB),_Y));
    coeff_y=-(m.sommets(m.aretes(:,_FIN),_X)-m.sommets(m.aretes(:,_DEB),_X));

    pr_scal=coord_segment_centre_deb(:,_X).*coeff_x+coord_segment_centre_deb(:,_
Y).*coeff_y;
    v_rectif=find(pr_scal<0);

    rectification=ones(pr_scal);
    rectification(v_rectif)=-1;
    coeff_x=rectification.*coeff_x;
    coeff_y=rectification.*coeff_y;


    // Calcul des coefficients de la matrice de masse

    coeff_M=(m.aretes(:,_DKL).*m.aretes(:,_MES))/2;


    /// Les aretes du bord

    /// Recherche des aretes pour la condition de Dirichlet non homogene
    temp=find(m.aretes(:,_LABEL)==-1);

    // Evaluation des donnÃ©es au bord
```

Programme construction_VF_Stokes.sce

```
    ubord=zeros(2*m.aretes(temp,_K));

    indiA=m.aretes(temp,_K);
    indjA=indiA;
    valA=tauKL(temp);

    indiB_x=m.aretes(temp,_DEB);
    indjB_x=indiA;
    valB_x=coeff_x(temp)/2;

    indiB_x=[indiB_x;m.aretes(temp,_FIN)];
    indjB_x=[indjB_x;m.aretes(temp,_K)];
    valB_x=[valB_x;coeff_x(temp)/2];


    indiB_y=m.aretes(temp,_DEB);
    indjB_y=indiA;
    valB_y=coeff_y(temp) /2;

    indiB_y=[indiB_y;m.aretes(temp,_FIN)];
    indjB_y=[indjB_y;m.aretes(temp,_K)];
    valB_y=[valB_y;coeff_y(temp) /2];

    indib=[indib; m.aretes(temp,_K)];
    valb1=[valb1; tauKL(temp).*ubord];
    valb2=[valb2; tauKL(temp).*ubord];

    /// Toutes les autres aretes du bord sont supposees
    /// correspondre a des CL de Neumann non homogene
    /// Donc elles ne contribuent pas au schema


    /// La partie du terme source qui concerne tout le monde

    indib=[indib; m.aretes(:,_K)];
    valb1=[valb1; (m.aretes(:,_MES_K_DEB)+m.aretes(:,_MES_K_FIN)).*source_u1(m.a
retes(:,_K))];
    valb2=[valb2; (m.aretes(:,_MES_K_DEB)+m.aretes(:,_MES_K_FIN)).*source_u2(m.a
retes(:,_K))];

    /// On traite maintenant les aretes interieures

    temp=find(m.aretes(:,_L)>0);

    indiA=[indiA; m.aretes(temp,_K)];
    indjA=[indjA; m.aretes(temp,_K)];
    valA=[valA; tauKL(temp)];

    indiA=[indiA; m.aretes(temp,_K)];
    indjA=[indjA; m.aretes(temp,_L)];
    valA=[valA; -tauKL(temp)];

    indiA=[indiA; m.aretes(temp,_L)];
    indjA=[indjA; m.aretes(temp,_K)];
    valA=[valA; -tauKL(temp)];

    indiA=[indiA; m.aretes(temp,_L)];
    indjA=[indjA; m.aretes(temp,_L)];
    valA=[valA; tauKL(temp)];


    indiB_x=[indiB_x;m.aretes(temp,_DEB)];
    indjB_x=[indjB_x;m.aretes(temp,_K)];
    valB_x=[valB_x;coeff_x(temp) /2];
```

Programme construction_VF_Stokes.sce

```
    indiB_x=[indiB_x;m.aretes(temp,_FIN)];
    indjB_x=[indjB_x;m.aretes(temp,_K)];
    valB_x=[valB_x;coeff_x(temp) /2];

    indiB_y=[indiB_y;m.aretes(temp,_DEB)];
    indjB_y=[indjB_y;m.aretes(temp,_K)];
    valB_y=[valB_y;coeff_y(temp) /2];

    indiB_y=[indiB_y;m.aretes(temp,_FIN)];
    indjB_y=[indjB_y;m.aretes(temp,_K)];
    valB_y=[valB_y;coeff_y(temp) /2];

    indiB_x=[indiB_x;m.aretes(temp,_DEB)];
    indjB_x=[indjB_x;m.aretes(temp,_L)];
    valB_x=[valB_x;-coeff_x(temp) /2];

    indiB_x=[indiB_x;m.aretes(temp,_FIN)];
    indjB_x=[indjB_x;m.aretes(temp,_L)];
    valB_x=[valB_x;-coeff_x(temp) /2];

    indiB_y=[indiB_y;m.aretes(temp,_DEB)];
    indjB_y=[indjB_y;m.aretes(temp,_L)];
    valB_y=[valB_y;-coeff_y(temp) /2];

    indiB_y=[indiB_y;m.aretes(temp,_FIN)];
    indjB_y=[indjB_y;m.aretes(temp,_L)];
    valB_y=[valB_y;-coeff_y(temp) /2];

    indib=[indib; m.aretes(temp,_L)];
    valb1=[valb1; (m.aretes(temp,_MES_L_DEB)+m.aretes(temp,_MES_L_FIN)).*source_
u1(m.aretes(temp,_L))];
    valb2=[valb2; (m.aretes(temp,_MES_L_DEB)+m.aretes(temp,_MES_L_FIN)).*source_
u2(m.aretes(temp,_L))];

    /// Construction proprement dite

    indjb=ones(indib);

    A=sparse([indiA indjA], valA);
    Mat_Rigidite=[A spzeros(A);spzeros(A) A];

    B_x=sparse([indiB_x indjB_x], valB_x);
    B_y=sparse([indiB_y indjB_y], valB_y);
    Mat_Div=[B_x B_y];

    indjb=ones(indib);
    b1=sparse([indib indjb], valb1);
    b2=sparse([indib indjb], valb2);
    F=[b1;b2;source_p];
    F=full(F);

    //// Construction de la matrice de masse ////

    // On trouve les sommets de chaque maille
    [m]=calcul_sommets_triangles(m);

    //Coefficient de la matrice de l element de reference
    m1=1/12;
    m2=1/24;

    // Construction

    indiM=m.centres(:,_S1);
    indjM=m.centres(:,_S1);
    valM=2*m1*m.centres(:,_MES_K);
```

Programme construction_VF_Stokes.sce

```
    indiM=[indiM;m.centres(:,_S1)];
    indjM=[indjM;m.centres(:,_S2)];
    valM=[valM;2*m2*m.centres(:,_MES_K)];

    indiM=[indiM;m.centres(:,_S1)];
    indjM=[indjM;m.centres(:,_S3)];
    valM=[valM;2*m2*m.centres(:,_MES_K)];

    indiM=[indiM;m.centres(:,_S2)];
    indjM=[indjM;m.centres(:,_S1)];
    valM=[valM;2*m2*m.centres(:,_MES_K)];

    indiM=[indiM;m.centres(:,_S2)];
    indjM=[indjM;m.centres(:,_S2)];
    valM=[valM;2*m1*m.centres(:,_MES_K)];

    indiM=[indiM;m.centres(:,_S2)];
    indjM=[indjM;m.centres(:,_S3)];
    valM=[valM;2*m2*m.centres(:,_MES_K)];

    indiM=[indiM;m.centres(:,_S3)];
    indjM=[indjM;m.centres(:,_S1)];
    valM=[valM;2*m2*m.centres(:,_MES_K)];

    indiM=[indiM;m.centres(:,_S3)];
    indjM=[indjM;m.centres(:,_S2)];
    valM=[valM;2*m2*m.centres(:,_MES_K)];

    indiM=[indiM;m.centres(:,_S3)];
    indjM=[indjM;m.centres(:,_S3)];
    valM=[valM;2*m1*m.centres(:,_MES_K)];

    Mat_Masse=sparse([indiM indjM], valM);
endfunction



//***********************************************************************
//    Le schema DDFV pour une diffusion scalaire
//    programme a l'aide d'operations vectorielles
//    Prise en compte des CL de Dirichlet non-homogene (_LABEL=-1)
//                        des CL de Neumann homogene  (_LABEL<-1)
//***********************************************************************


function [Mat_Rigidite,Mat_Div,Mat_Masse,F]=const_schema_DDFV(m,donnees)

    printf("Assemblage DDFV\n");

    // On evalue le terme source aux centres et aux sommets
    source_centres=eval_fonction_u(m.centres,donnees.source);
    source_centres1=source_centres(1:m.nb_vol);
    source_centres2=source_centres(m.nb_vol+1:$);

    source_sommets=eval_fonction_u(m.sommets,donnees.source);
    source_sommets1=source_sommets(1:m.nb_som);
    source_sommets2=source_sommets(m.nb_som+1:$);

    source_p=zeros(m.nb_are,1);

    centres_aretes=(m.sommets(m.aretes(:,_DEB),:)+m.sommets(m.aretes(:,_FIN),:))
/2;
```

Programme construction_VF_Stokes.sce

```
    // On calcule la mesure des diamants
    md=m.aretes(:,_MES_K_DEB)+m.aretes(:,_MES_K_FIN)+m.aretes(:,_MES_L_DEB)+m.ar
etes(:,_MES_L_FIN);

    // On calcule le produit scalaire entre la normale primale et la normale dua
le
    // Ce coefficient est nul si le maillage est orthogonal
    N_KLscaNS_KL=m.aretes(:,_N_KL_X).*m.aretes(:,_NS_DEBFIN_X)+m.aretes(:,_N_KL_
Y).*m.aretes(:,_NS_DEBFIN_Y);

    // On calcule maintenant les coefficients qui serviront dans la construction
 du schema
    flux_NN=0.5.*m.aretes(:,_MES).*m.aretes(:,_MES)./md;
    flux_NNS=0.5.*m.aretes(:,_MES).*m.aretes(:,_DKL)./md.*N_KLscaNS_KL;
    flux_NSN=flux_NNS;
    flux_NSNS=0.5.*m.aretes(:,_DKL).*m.aretes(:,_DKL)./md;

    // Ces coefficients interviennent de la facon suivante
    // Dans le flux numerique
    // m.aretes(:,_MES)*k_sig*grad(u).N_KL=flux_NN*(u_L-u_K)+flux_NNS*(u_FIN-u_D
EB)
    // m.aretes(:,_DKL)*k_sig*grad(u).NS_KL=flux_NSN*(u_L-u_K)+flux_NSNS*(u_FIN-
u_DEB)

    // On a besoin de reperer les sommets au bord pour assembler le terme source
    // som_au_bord vaudra 1 a l'interieur et 0 au bord

    som_au_bord=ones(m.nb_som,1);
    temp=find(m.aretes(:,_L)<=0);
    som_au_bord(m.aretes(temp,_DEB))=0;
    som_au_bord(m.aretes(temp,_FIN))=0;

    // On evalue la donnee de Dirichlet sur tous les sommets

    ub_som=zeros(m.nb_som,1);
    ub_som=eval_fonction_u(m.sommets(:,[_X _Y]),donnees.ubordD);


    //// Calcul des coefficients de la matrice de divergence :
    //// terme mes(sigma)n_KL et mes(sigma_star)n_K_star_L_star

    // coordonnees du vecteur x_k x_k_star
    coord_segment_centre_deb=m.sommets(m.aretes(:,_DEB),:)-m.centres(m.aretes(:,
_K),:);

    // normale sortante a K (non rectifiee)
    coeff_x=(m.sommets(m.aretes(:,_FIN),_Y)-m.sommets(m.aretes(:,_DEB),_Y));
    coeff_y=-(m.sommets(m.aretes(:,_FIN),_X)-m.sommets(m.aretes(:,_DEB),_X));

    // normale sortante a K_star (non rectifiee)

    rectif_centre_bord=zeros(m.centres(m.aretes(:,_K),:));

    rectif_centre_bord(temp,:)=(1/2)*(m.sommets(m.aretes(temp,_DEB),:)+m.sommets
(m.aretes(temp,_FIN),:));

    temp_rectif=find(m.aretes(:,_L)>0);
    rectif_centre_bord(temp_rectif,:)=m.centres(m.aretes(temp_rectif,_L),:);

    coeff_x_star=(rectif_centre_bord(:,_Y)-m.centres(m.aretes(:,_K),_Y));
    coeff_y_star=-(rectif_centre_bord(:,_X)-m.centres(m.aretes(:,_K),_X));

    // calcul produit scalaire vecteur x_k x_k_star avec normale sortante a K
    // pour rectifier si ce n est pas positif
```

Programme construction_VF_Stokes.sce

```
    pr_scal=coord_segment_centre_deb(:,_X).*coeff_x+coord_segment_centre_deb(:,_
Y).*coeff_y;
    v_rectif=find(pr_scal<0);

    // calcul produit scalaire vecteur x_k x_k_star avec normale sortante a K
    // pour rectifier si ce n est pas negatif
    pr_scal_star=coord_segment_centre_deb(:,_X).*coeff_x_star+coord_segment_cent
re_deb(:,_Y).*coeff_y_star;
    v_rectif_star=find(pr_scal_star>0);

    // rectification
    rectification=ones(pr_scal);
    rectification_star=ones(pr_scal_star);

    rectification(v_rectif)=-1;
    rectification_star(v_rectif_star)=-1;

    coeff_x=rectification.*coeff_x;
    coeff_y=rectification.*coeff_y;

    coeff_x_star=rectification_star.*coeff_x_star;
    coeff_y_star=rectification_star.*coeff_y_star;

    // Debut de la construction de la matrice et du second membre

    indiA=[];
    indjA=[];
    valA=[];

    indiB_x=[];
    indjB_x=[];
    valB_x=[];

    indiB_y=[];
    indjB_y=[];
    valB_y=[];

    indib=[];
    indjb=[];
    valb1=[];
    valb2=[];

    // On cherche les aretes du bord Dirichlet non homogene
    temp=find(m.aretes(:,_LABEL)==-1);

    // On evalue la donnee de Dirichlet au centre de ces aretes
    //ubord=eval_fonction_u(centres_aretes(temp,:),donnees.ubordD);
    t=size(centres_aretes(temp,:),1)
    ubord=zeros(t,1)

    indiA=m.aretes(temp,_K);
    indjA=indiA;
    valA=flux_NN(temp);

    indiB_x=temp';
    indjB_x=indiA;
    valB_x=(coeff_x(temp))/2;

    indiB_y=temp';
    indjB_y=indiA;
    valB_y=(coeff_y(temp))/2;

    indib=m.aretes(temp,_K);

    valb1=flux_NN(temp).*ubord- flux_NNS(temp).*ub_som(m.aretes(temp,_DEB)) ...
```

Programme construction_VF_Stokes.sce

```
    + flux_NNS(temp).*ub_som(m.aretes(temp,_FIN));
    valb2=flux_NN(temp).*ubord- flux_NNS(temp).*ub_som(m.aretes(temp,_DEB)) ...
    + flux_NNS(temp).*ub_som(m.aretes(temp,_FIN));

    // Les autres aretes du bord sont supposees correspondre a une condition
    // de Neumann homogene qui ne necessite aucun traitement particulier

    // On traite maintenant les aretes interieures.

    temp=find(m.aretes(:,_L)>0);

    // Contributions a l'equation de la maille K

    indiA=[indiA;m.aretes(temp,_K)];
    indjA=[indjA;m.aretes(temp,_K)];
    valA=[valA;flux_NN(temp)];

    indiA=[indiA; m.aretes(temp,_K)];
    indjA=[indjA; m.aretes(temp,_L)];
    valA=[valA; -flux_NN(temp)];

    indiA=[indiA; m.aretes(temp,_K)];
    indjA=[indjA; m.aretes(temp,_DEB)+m.nb_vol];
    valA=[valA; flux_NNS(temp)];

    indiA=[indiA; m.aretes(temp,_K)];
    indjA=[indjA; m.aretes(temp,_FIN)+m.nb_vol];
    valA=[valA; -flux_NNS(temp)];

    indiB_x=[indiB_x;temp'];
    indjB_x=[indjB_x;m.aretes(temp,_K)];
    valB_x=[valB_x;(coeff_x(temp))/2];

    indiB_y=[indiB_y;temp'];
    indjB_y=[indjB_y;m.aretes(temp,_K)];
    valB_y=[valB_y;(coeff_y(temp))/2];

    indib=[indib;m.aretes(:,_K)];
    valb1=[valb1; (m.aretes(:,_MES_K_DEB)+m.aretes(:,_MES_K_FIN)).*source_centre
s1(m.aretes(:,_K))];
    valb2=[valb2; (m.aretes(:,_MES_K_DEB)+m.aretes(:,_MES_K_FIN)).*source_centre
s2(m.aretes(:,_K))];

    // Contributions a l'equation de la maille L

    indiA=[indiA; m.aretes(temp,_L)];
    indjA=[indjA; m.aretes(temp,_K)];
    valA=[valA; -flux_NN(temp)];

    indiA=[indiA; m.aretes(temp,_L)];
    indjA=[indjA; m.aretes(temp,_L)];
    valA=[valA; flux_NN(temp)];

    indiA=[indiA; m.aretes(temp,_L)];
    indjA=[indjA; m.aretes(temp,_DEB)+m.nb_vol];
    valA=[valA; -flux_NNS(temp)];

    indiA=[indiA; m.aretes(temp,_L)];
    indjA=[indjA; m.aretes(temp,_FIN)+m.nb_vol];
    valA=[valA; flux_NNS(temp)];

    indiB_x=[indiB_x;temp'];
    indjB_x=[indjB_x;m.aretes(temp,_L)];
    valB_x=[valB_x;-(coeff_x(temp))/2];
```

Programme construction_VF_Stokes.sce

```
    indiB_y=[indiB_y;temp'];
    indjB_y=[indjB_y;m.aretes(temp,_L)];
    valB_y=[valB_y;-(coeff_y(temp))/2];

    indib=[indib; m.aretes(temp,_L)];
    valb1=[valb1; (m.aretes(temp,_MES_L_DEB)+m.aretes(temp,_MES_L_FIN)).*source_
centres1(m.aretes(temp,_L))];
    valb2=[valb2; (m.aretes(temp,_MES_L_DEB)+m.aretes(temp,_MES_L_FIN)).*source_
centres2(m.aretes(temp,_L))];

    // Contributions a l'equation du sommet DEB
    // En prenant garde aux sommets du bord

    indiA=[indiA; m.aretes(:,_DEB)+m.nb_vol];
    indjA=[indjA; m.aretes(:,_DEB)+m.nb_vol];
    valA=[valA; flux_NSNS(:).*som_au_bord(m.aretes(:,_DEB))+1-som_au_bord(m.aret
es(:,_DEB))];

    indiA=[indiA; m.aretes(temp,_DEB)+m.nb_vol];
    indjA=[indjA; m.aretes(temp,_FIN)+m.nb_vol];
    valA=[valA; -flux_NSNS(temp).*som_au_bord(m.aretes(temp,_DEB))];

    indiA=[indiA; m.aretes(temp,_DEB)+m.nb_vol];
    indjA=[indjA; m.aretes(temp,_K)];
    valA=[valA; flux_NSN(temp).*som_au_bord(m.aretes(temp,_DEB))];

    indiA=[indiA; m.aretes(temp,_DEB)+m.nb_vol];
    indjA=[indjA; m.aretes(temp,_L)];
    valA=[valA; -flux_NSN(temp).*som_au_bord(m.aretes(temp,_DEB))];

    vect_are=1:m.nb_are;
    indiB_x=[indiB_x;vect_are'];
    indjB_x=[indjB_x; m.aretes(:,_DEB)+m.nb_vol];
    valB_x=[valB_x;(coeff_x_star/2).*som_au_bord(m.aretes(:,_DEB))];

    indiB_y=[indiB_y;vect_are'];
    indjB_y=[indjB_y; m.aretes(:,_DEB)+m.nb_vol];
    valB_y=[valB_y;(coeff_y_star/2).*som_au_bord(m.aretes(:,_DEB))];

    // Le second membre est different si on est au bord ou a l'interieur
    test1= (m.aretes(:,_MES_K_DEB)+m.aretes(:,_MES_L_DEB)).*source_sommets1(m.ar
etes(:,_DEB)) ...
    .*som_au_bord(m.aretes(:,_DEB)) ...
    +(1-som_au_bord(m.aretes(:,_DEB))).*ub_som(m.aretes(:,_DEB)) ;

    test2= (m.aretes(:,_MES_K_DEB)+m.aretes(:,_MES_L_DEB)).*source_sommets2(m.ar
etes(:,_DEB)) ...
    .*som_au_bord(m.aretes(:,_DEB)) ...
    +(1-som_au_bord(m.aretes(:,_DEB))).*ub_som(m.aretes(:,_DEB)) ;

    indib=[indib; m.aretes(:,_DEB)+m.nb_vol];
    valb1=[valb1;test1 ];
    valb2=[valb2;test2 ];

    // Contributions a l'equation du sommet FIN
    // En prenant garde aux sommets du bord

    indiA=[indiA; m.aretes(:,_FIN)+m.nb_vol];
    indjA=[indjA; m.aretes(:,_FIN)+m.nb_vol];
    valA=[valA; flux_NSNS(:).*som_au_bord(m.aretes(:,_FIN))+1-som_au_bord(m.aret
es(:,_FIN))];

    indiA=[indiA; m.aretes(temp,_FIN)+m.nb_vol];
    indjA=[indjA; m.aretes(temp,_DEB)+m.nb_vol];
    valA=[valA; -flux_NSNS(temp).*som_au_bord(m.aretes(temp,_FIN))];
```

Programme construction_VF_Stokes.sce

```
    indiA=[indiA; m.aretes(temp,_FIN)+m.nb_vol];
    indjA=[indjA; m.aretes(temp,_K)];
    valA=[valA; -flux_NSN(temp).*som_au_bord(m.aretes(temp,_FIN))];

    indiA=[indiA; m.aretes(temp,_FIN)+m.nb_vol];
    indjA=[indjA; m.aretes(temp,_L)];
    valA=[valA; flux_NSN(temp).*som_au_bord(m.aretes(temp,_FIN))];

    indiB_x=[indiB_x;vect_are'];
    indjB_x=[indjB_x; m.aretes(:,_FIN)+m.nb_vol];
    valB_x=[valB_x;-(coeff_x_star/2).*som_au_bord(m.aretes(:,_FIN))];

    indiB_y=[indiB_y;vect_are'];
    indjB_y=[indjB_y; m.aretes(:,_FIN)+m.nb_vol];
    valB_y=[valB_y;-(coeff_y_star/2).*som_au_bord(m.aretes(:,_FIN))];

    // Le second membre est different si on est au bord ou a l'interieur
    test1= (m.aretes(:,_MES_K_FIN)+m.aretes(:,_MES_L_FIN)).*source_sommets1(m.ar
etes(:,_FIN)) ...
    .*som_au_bord(m.aretes(:,_FIN)) ...
    +(1-som_au_bord(m.aretes(:,_FIN))).*ub_som(m.aretes(:,_FIN));

    test2= (m.aretes(:,_MES_K_FIN)+m.aretes(:,_MES_L_FIN)).*source_sommets2(m.ar
etes(:,_FIN)) ...
    .*som_au_bord(m.aretes(:,_FIN)) ...
    +(1-som_au_bord(m.aretes(:,_FIN))).*ub_som(m.aretes(:,_FIN));

    indib=[indib; m.aretes(:,_FIN)+m.nb_vol];
    valb1=[valb1; test1];
    valb2=[valb2; test2];


    // Construction proprement dite


    A=sparse([indiA indjA], valA);
    Mat_Rigidite=[A zeros(A);zeros(A) A];

    B_x=sparse([indiB_x indjB_x], valB_x);
    B_y=sparse([indiB_y indjB_y], valB_y);

    Mat_Div=[B_x B_y];

    s=size(Mat_Div,1);

    Mat_Masse=diag(md);
    Mat_Masse=sparse(Mat_Masse)


    indjb=ones(indib);
    b1=sparse([indib indjb], valb1);
    b2=sparse([indib indjb], valb2);
    F=[b1;b2;source_p];
    F=full(F);

endfunction



//*************************************************************************
//    Le schÃ©ma DDFV stabilise pour une diffusion scalaire
//    programme a l'aide d'operations vectorielles
//    Prise en compte des CL de Dirichlet non-homogene (_LABEL=-1)
//                     des CL de Neumann homogene  (_LABEL<-1)
```

Programme construction_VF_Stokes.sce

```
//*****************************************************************************


function [Mat_Rigidite,Mat_Div,Mat_Masse,Mat_Stab,F]=const_schema_DDFVs(m,donnees,lambda_stab)

    printf("Assemblage DDFVs\n");

    // On evalue le terme source aux centres et aux sommets
    source_centres=eval_fonction_u(m.centres,donnees.source);
    source_centres1=source_centres(1:m.nb_vol);
    source_centres2=source_centres(m.nb_vol+1:$);

    source_sommets=eval_fonction_u(m.sommets,donnees.source);
    source_sommets1=source_sommets(1:m.nb_som);
    source_sommets2=source_sommets(m.nb_som+1:$);

    source_p=zeros(m.nb_are,1);

    centres_aretes=(m.sommets(m.aretes(:,_DEB),:)+m.sommets(m.aretes(:,_FIN),:))/2;


    // On calcule la mesure des diamants
    md=m.aretes(:,_MES_K_DEB)+m.aretes(:,_MES_K_FIN)+m.aretes(:,_MES_L_DEB)+m.aretes(:,_MES_L_FIN);

    // On calcule le produit scalaire entre la normale primale et la normale duale
    // Ce coefficient est nul si le maillage est orthogonal
    N_KLscaNS_KL=m.aretes(:,_N_KL_X).*m.aretes(:,_NS_DEBFIN_X)+m.aretes(:,_N_KL_Y).*m.aretes(:,_NS_DEBFIN_Y);

    // On calcule maintenant les coefficients qui serviront dans la construction du schema
    flux_NN=0.5.*m.aretes(:,_MES).*m.aretes(:,_MES)./md;
    flux_NNS=0.5.*m.aretes(:,_MES).*m.aretes(:,_DKL)./md.*N_KLscaNS_KL;
    flux_NSN=flux_NNS;
    flux_NSNS=0.5.*m.aretes(:,_DKL).*m.aretes(:,_DKL)./md;

    // Ces coefficients interviennent de la facon suivante
    // Dans le flux numerique
    // m.aretes(:,_MES)*k_sig*grad(u).N_KL=flux_NN*(u_L-u_K)+flux_NNS*(u_FIN-u_DEB)
    // m.aretes(:,_DKL)*k_sig*grad(u).NS_KL=flux_NSN*(u_L-u_K)+flux_NSNS*(u_FIN-u_DEB)

    // On a besoin de reperer les sommets au bord pour assembler le terme source
    // som_au_bord vaudra 1 a l'interieur et 0 au bord

    som_au_bord=ones(m.nb_som,1);
    temp=find(m.aretes(:,_L)<=0);
    som_au_bord(m.aretes(temp,_DEB))=0;
    som_au_bord(m.aretes(temp,_FIN))=0;

    // On evalue la donnee de Dirichlet sur tous les sommets

    ub_som=zeros(m.nb_som,1);
    ub_som=eval_fonction_u(m.sommets(:,[_X _Y]),donnees.ubordD);


    //// Calcul des coefficients de la matrice de divergence :
    //// terme mes(sigma)n_KL et mes(sigma_star)n_K_star_L_star

    // coordonnees du vecteur x_k x_k_star
```

Programme construction_VF_Stokes.sce

```
    coord_segment_centre_deb=m.sommets(m.aretes(:,_DEB),:)-m.centres(m.aretes(:,_K),:);

    // normale sortante a K (non rectifiee)
    coeff_x=(m.sommets(m.aretes(:,_FIN),_Y)-m.sommets(m.aretes(:,_DEB),_Y));
    coeff_y=-(m.sommets(m.aretes(:,_FIN),_X)-m.sommets(m.aretes(:,_DEB),_X));

    // normale sortante a K_star (non rectifiee)

    rectif_centre_bord=zeros(m.centres(m.aretes(:,_K),:));

    rectif_centre_bord(temp,:)=(1/2)*(m.sommets(m.aretes(temp,_DEB),:)+m.sommets(m.aretes(temp,_FIN),:));

    temp_rectif=find(m.aretes(:,_L)>0);
    rectif_centre_bord(temp_rectif,:)=m.centres(m.aretes(temp_rectif,_L),:);

    coeff_x_star=(rectif_centre_bord(:,_Y)-m.centres(m.aretes(:,_K),_Y));
    coeff_y_star=-(rectif_centre_bord(:,_X)-m.centres(m.aretes(:,_K),_X));

    // calcul produit scalaire vecteur x_k x_k_star avec normale sortante a K
    // pour rectifier si ce n est pas positif
    pr_scal=coord_segment_centre_deb(:,_X).*coeff_x+coord_segment_centre_deb(:,_Y).*coeff_y;
    v_rectif=find(pr_scal<0);

    // calcul produit scalaire vecteur x_k x_k_star avec normale sortante a K
    // pour rectifier si ce n est pas negatif
    pr_scal_star=coord_segment_centre_deb(:,_X).*coeff_x_star+coord_segment_centre_deb(:,_Y).*coeff_y_star;
    v_rectif_star=find(pr_scal_star>0);

    // rectification
    rectification=ones(pr_scal);
    rectification_star=ones(pr_scal_star);

    rectification(v_rectif)=-1;
    rectification_star(v_rectif_star)=-1;

    coeff_x=rectification.*coeff_x;
    coeff_y=rectification.*coeff_y;

    coeff_x_star=rectification_star.*coeff_x_star;
    coeff_y_star=rectification_star.*coeff_y_star;

    // Debut de la construction de la matrice et du second membre

    indiA=[];
    indjA=[];
    valA=[];

    indiB_x=[];
    indjB_x=[];
    valB_x=[];

    indiB_y=[];
    indjB_y=[];
    valB_y=[];

    indiM=[];
    valM=[];

    indib=[];
    indjb=[];
    valb1=[];
```

Programme construction_VF_Stokes.sce

```
valb2=[];

// On cherche les aretes du bord Dirichlet non homogene
temp=find(m.aretes(:,_LABEL)==-1);

// On evalue la donnee de Dirichlet au centre de ces aretes
//ubord=eval_fonction_u(centres_aretes(temp,:),donnees.ubordD);
t=size(centres_aretes(temp,:),1)
ubord=zeros(t,1)

indiA=m.aretes(temp,_K);
indjA=indiA;
valA=flux_NN(temp);

indiB_x=temp';
indjB_x=indiA;
valB_x=(coeff_x(temp))/2;

indiB_y=temp';
indjB_y=indiA;
valB_y=(coeff_y(temp))/2;

indiM=temp';
valM=md(temp);

indib=m.aretes(temp,_K);

valb1=flux_NN(temp).*ubord- flux_NNS(temp).*ub_som(m.aretes(temp,_DEB)) ...
+ flux_NNS(temp).*ub_som(m.aretes(temp,_FIN));
valb2=flux_NN(temp).*ubord- flux_NNS(temp).*ub_som(m.aretes(temp,_DEB)) ...
+ flux_NNS(temp).*ub_som(m.aretes(temp,_FIN));

// Les autres aretes du bord sont supposees correspondre a une condition
// de Neumann homogene qui ne necessite aucun traitement particulier

// On traite maintenant les aretes interieures.

temp=find(m.aretes(:,_L)>0);

// Contributions a l'equation de la maille K

indiA=[indiA;m.aretes(temp,_K)];
indjA=[indjA;m.aretes(temp,_K)];
valA=[valA;flux_NN(temp)];

indiA=[indiA; m.aretes(temp,_K)];
indjA=[indjA; m.aretes(temp,_L)];
valA=[valA; -flux_NN(temp)];

indiA=[indiA; m.aretes(temp,_K)];
indjA=[indjA; m.aretes(temp,_DEB)+m.nb_vol];
valA=[valA; flux_NNS(temp)];

indiA=[indiA; m.aretes(temp,_K)];
indjA=[indjA; m.aretes(temp,_FIN)+m.nb_vol];
valA=[valA; -flux_NNS(temp)];

indiB_x=[indiB_x;temp'];
indjB_x=[indjB_x;m.aretes(temp,_K)];
valB_x=[valB_x;(coeff_x(temp))/2];

indiB_y=[indiB_y;temp'];
indjB_y=[indjB_y;m.aretes(temp,_K)];
valB_y=[valB_y;(coeff_y(temp))/2];
```

Programme construction_VF_Stokes.sce

```
    indiM=[indiM;temp'];
    valM=[valM;md(temp)];

    indib=[indib;m.aretes(:,_K)];
    valb1=[valb1; (m.aretes(:,_MES_K_DEB)+m.aretes(:,_MES_K_FIN)).*source_centre
s1(m.aretes(:,_K))];
    valb2=[valb2; (m.aretes(:,_MES_K_DEB)+m.aretes(:,_MES_K_FIN)).*source_centre
s2(m.aretes(:,_K))];

    // Contributions a l'equation de la maille L

    indiA=[indiA; m.aretes(temp,_L)];
    indjA=[indjA; m.aretes(temp,_K)];
    valA=[valA; -flux_NN(temp)];

    indiA=[indiA; m.aretes(temp,_L)];
    indjA=[indjA; m.aretes(temp,_L)];
    valA=[valA; flux_NN(temp)];

    indiA=[indiA; m.aretes(temp,_L)];
    indjA=[indjA; m.aretes(temp,_DEB)+m.nb_vol];
    valA=[valA; -flux_NNS(temp)];

    indiA=[indiA; m.aretes(temp,_L)];
    indjA=[indjA; m.aretes(temp,_FIN)+m.nb_vol];
    valA=[valA; flux_NNS(temp)];

    indiB_x=[indiB_x;temp'];
    indjB_x=[indjB_x;m.aretes(temp,_L)];
    valB_x=[valB_x;-(coeff_x(temp))/2];

    indiB_y=[indiB_y;temp'];
    indjB_y=[indjB_y;m.aretes(temp,_L)];
    valB_y=[valB_y;-(coeff_y(temp))/2];

    indiM=[indiM;temp'];
    valM=[valM;md(temp)];

    indib=[indib; m.aretes(temp,_L)];
    valb1=[valb1; (m.aretes(temp,_MES_L_DEB)+m.aretes(temp,_MES_L_FIN)).*source_
centres1(m.aretes(temp,_L))];
    valb2=[valb2; (m.aretes(temp,_MES_L_DEB)+m.aretes(temp,_MES_L_FIN)).*source_
centres2(m.aretes(temp,_L))];

    // Contributions a l'equation du sommet DEB
    // En prenant garde aux sommets du bord

    indiA=[indiA; m.aretes(:,_DEB)+m.nb_vol];
    indjA=[indjA; m.aretes(:,_DEB)+m.nb_vol];
    valA=[valA; flux_NSNS(:).*som_au_bord(m.aretes(:,_DEB))+1-som_au_bord(m.aret
es(:,_DEB))];

    indiA=[indiA; m.aretes(temp,_DEB)+m.nb_vol];
    indjA=[indjA; m.aretes(temp,_FIN)+m.nb_vol];
    valA=[valA; -flux_NSNS(temp).*som_au_bord(m.aretes(temp,_DEB))];

    indiA=[indiA; m.aretes(temp,_DEB)+m.nb_vol];
    indjA=[indjA; m.aretes(temp,_K)];
    valA=[valA; flux_NSN(temp).*som_au_bord(m.aretes(temp,_DEB))];

    indiA=[indiA; m.aretes(temp,_DEB)+m.nb_vol];
    indjA=[indjA; m.aretes(temp,_L)];
    valA=[valA; -flux_NSN(temp).*som_au_bord(m.aretes(temp,_DEB))];

    vect_are=1:m.nb_are;
```

Programme construction_VF_Stokes.sce

```
    indiB_x=[indiB_x;vect_are'];
    indjB_x=[indjB_x; m.aretes(:,_DEB)+m.nb_vol];
    valB_x=[valB_x;(coeff_x_star/2).*som_au_bord(m.aretes(:,_DEB))];

    indiB_y=[indiB_y;vect_are'];
    indjB_y=[indjB_y; m.aretes(:,_DEB)+m.nb_vol];
    valB_y=[valB_y;(coeff_y_star/2).*som_au_bord(m.aretes(:,_DEB))];

    // Le second membre est different si on est au bord ou a l'interieur
    test1= (m.aretes(:,_MES_K_DEB)+m.aretes(:,_MES_L_DEB)).*source_sommets1(m.ar
etes(:,_DEB)) ...
    .*som_au_bord(m.aretes(:,_DEB)) ...
    +(1-som_au_bord(m.aretes(:,_DEB))).*ub_som(m.aretes(:,_DEB)) ;

    test2= (m.aretes(:,_MES_K_DEB)+m.aretes(:,_MES_L_DEB)).*source_sommets2(m.ar
etes(:,_DEB)) ...
    .*som_au_bord(m.aretes(:,_DEB)) ...
    +(1-som_au_bord(m.aretes(:,_DEB))).*ub_som(m.aretes(:,_DEB)) ;

    indib=[indib; m.aretes(:,_DEB)+m.nb_vol];
    valb1=[valb1;test1 ];
    valb2=[valb2;test2 ];

    // Contributions a l'equation du sommet FIN
    // En prenant garde aux sommets du bord

    indiA=[indiA; m.aretes(:,_FIN)+m.nb_vol];
    indjA=[indjA; m.aretes(:,_FIN)+m.nb_vol];
    valA=[valA; flux_NSNS(:).*som_au_bord(m.aretes(:,_FIN))+1-som_au_bord(m.aret
es(:,_FIN))];

    indiA=[indiA; m.aretes(temp,_FIN)+m.nb_vol];
    indjA=[indjA; m.aretes(temp,_DEB)+m.nb_vol];
    valA=[valA; -flux_NSNS(temp).*som_au_bord(m.aretes(temp,_FIN))];

    indiA=[indiA; m.aretes(temp,_FIN)+m.nb_vol];
    indjA=[indjA; m.aretes(temp,_K)];
    valA=[valA; -flux_NSN(temp).*som_au_bord(m.aretes(temp,_FIN))];

    indiA=[indiA; m.aretes(temp,_FIN)+m.nb_vol];
    indjA=[indjA; m.aretes(temp,_L)];
    valA=[valA; flux_NSN(temp).*som_au_bord(m.aretes(temp,_FIN))];

    indiB_x=[indiB_x;vect_are'];
    indjB_x=[indjB_x; m.aretes(:,_FIN)+m.nb_vol];
    valB_x=[valB_x;-(coeff_x_star/2).*som_au_bord(m.aretes(:,_FIN))];

    indiB_y=[indiB_y;vect_are'];
    indjB_y=[indjB_y; m.aretes(:,_FIN)+m.nb_vol];
    valB_y=[valB_y;-(coeff_y_star/2).*som_au_bord(m.aretes(:,_FIN))];

    // Le second membre est different si on est au bord ou a l'interieur
    test1= (m.aretes(:,_MES_K_FIN)+m.aretes(:,_MES_L_FIN)).*source_sommets1(m.ar
etes(:,_FIN)) ...
    .*som_au_bord(m.aretes(:,_FIN)) ...
    +(1-som_au_bord(m.aretes(:,_FIN))).*ub_som(m.aretes(:,_FIN));

    test2= (m.aretes(:,_MES_K_FIN)+m.aretes(:,_MES_L_FIN)).*source_sommets2(m.ar
etes(:,_FIN)) ...
    .*som_au_bord(m.aretes(:,_FIN)) ...
    +(1-som_au_bord(m.aretes(:,_FIN))).*ub_som(m.aretes(:,_FIN));

    indib=[indib; m.aretes(:,_FIN)+m.nb_vol];
    valb1=[valb1; test1];
    valb2=[valb2; test2];
```

Programme construction_VF_Stokes.sce

```
    //// Calcul de la matrice de stabilisation


    ////////////////// Matrice de stabilisation Brezzi-Pitkaranta
    //  // Coefficients de la matrice
    //  diam1=max(m.aretes(:,_DKL));
    //  diam2=max(m.aretes(:,_MES));
    //  diam_maillage=max(diam1,diam2);
    //  coeff_S=(2*lambda_stab*diam_maillage^2) ./ones(md);
    //
    //  indiS=[];
    //  indjS=[];
    //  valS=[];
    //
    //  // Pour tous les diamants
    //  indiS=vect_are';
    //  indjS=vect_are';
    //  valS=2*coeff_S;
    //
    //  indiS=[indiS;vect_are'];
    //  indjS=[indjS;m.diamants(:,_D1)];
    //  valS=[valS;-coeff_S];
    //
    //  indiS=[indiS;vect_are'];
    //  indjS=[indjS;m.diamants(:,_D2)];
    //  valS=[valS;-coeff_S];
    //
    //  // Pour les diamants interieurs
    //  temp=find(m.diamants(:,_D3)~=0);
    //
    //  indiS=[indiS;temp'];
    //  indjS=[indjS;temp'];
    //  valS=[valS;2*coeff_S(temp)];
    //
    //  indiS=[indiS;temp'];
    //  indjS=[indjS;m.diamants(temp,_D3)];
    //  valS=[valS;-coeff_S(temp)];
    //
    //  indiS=[indiS;temp'];
    //  indjS=[indjS;m.diamants(temp,_D4)];
    //  valS=[valS;-coeff_S(temp)];
    //
    //  Mat_Stab=sparse([indiS indjS], valS);
    //  Mat_Stab($,:)=spzeros(1,s(1))
    ////////////////////////////////////////////


    // Construction proprement dite

    A=sparse([indiA indjA], valA);
    Mat_Rigidite=[A zeros(A);zeros(A) A];

    B_x=sparse([indiB_x indjB_x], valB_x);
    B_y=sparse([indiB_y indjB_y], valB_y);

    Mat_Div=[B_x B_y];

    s=size(Mat_Div,1);

    Mat_Masse=sparse([indiM indiM], valM);

    indjb=ones(indib);
    b1=sparse([indib indjb], valb1);
```

Programme construction_VF_Stokes.sce

```
    b2=sparse([indib indjb], valb2);
    F=[b1;b2;source_p];
    F=full(F);


    Mat_Stab=lambda_stab*Mat_Masse;


endfunction
```

Programme schema_VF.sce

```
////////////////////////////////////////////////////////
/////         Divers schemas de volumes finis        ///////
/////                pour pb Stokes 2D               ///////
/////                                                ///////
/////               Programme principal              ///////
/////         permettant le calcul par VF4/DDFV      ///////
/////                                                ///////
////////////////////////////////////////////////////////



//*****************************************************************************
// On efface toutes les variables et on charge les fonctions utiles
//*****************************************************************************

clear
clc
stacksize('max');
warning('off');
lines(0);

exec('maillages2D.sci');
exec('construction_VF_Stokes.sce');
exec('donnees2D.sci');
exec ('valeur_propre_puissance_inverse.sce');
exec ('valeur_propre_QR.sce');
exec ('decompositionQR.sci');



//*****************************************************************************
// Saisie par l'utilisateur des donnees du calcul
//*****************************************************************************


// on ne trace pas residu et lambda en fonction des itÃ©rations
choix_trace=2;

// choix du cas test

printf('=========================================\n');
printf('Choix du cas test :\n');
for i=1:length(cas_test)
    printf('%d) %s\n',i,cas_test(i).nom);
end;

choix=-1;
while (int(choix)<=0) | (int(choix)>length(cas_test))
    choix=input("Faites votre choix :");
end;
choix=int(choix);

// Chargement des donnees

donnees=cas_test(choix);


// On regarde si les maillages sont imposes dans le jeu de donnees
if isfield(donnees,"maillages") then
    l=list();
    for a=donnees.maillages,
        for b=liste_maillages,
            if a==b.nom then
                l($+1)=b;
            end;
```

Programme schema_VF.sce

```
        end;
    end;
    liste_maillages=l;
end;

// On ne garde que les maillages qui ont plusieurs niveaux de raffinement

l=list();
for i=1:length(liste_maillages)
    if isfield(liste_maillages(i),'indice_min') then
        l($+1)=liste_maillages(i);
    end;
end
liste_maillages=l;


// On choisit si on fait la verification Scilab

printf('=========================================\n');
printf('Choix de verifier avec Scilab :\n');
printf('1) On fait la verification\n');
printf('2) On ne fait pas la verification\n');

choix_scilab=-1;
while (int(choix_scilab)<=0) | (int(choix_scilab)>2)
    choix_scilab=input("Faites votre choix :");
end;

// On fait un choix parmi les possibilites restantes

printf('=========================================\n');
printf('Choix du maillage :\n');
for i=1:length(liste_maillages)
    printf('%d) %s\n',i,liste_maillages(i).nom);
end;

choix=-1;
while (int(choix)<=0) | (int(choix)>length(liste_maillages))
    choix=input("Faites votre choix :");
end;

choix=int(choix);
maillage=liste_maillages(choix);


// choix des differentes tailles de maillage pour les valeurs propres
if isfield(maillage,"indice_min") then
    printf('=========================================\n');
    printf('Choix du niveau de raffinement entre %d et %d :\n',maillage.indice_m
in,maillage.indice_max);
    nb_dep=-1;
    while (int(nb_dep)<maillage.indice_min) | (int(nb_dep)>maillage.indice_max)
        nb_dep=input("Choix du niveau le plus grosier :");
    end;
    nb_fin=-1;
    while (int(nb_fin)<nb_dep) | (int(nb_fin)>maillage.indice_max)
        nb_fin=input("Choix du niveau le plus fin :");
    end;
end

NN=nb_fin-nb_dep+1;


// choix du schÃ©ma : VF4 ou DDFV
if isfield(donnees,"coeff_mat") then
```

Programme schema_VF.sce

```
    choix_schema=3;
    printf('=====================================\n');
    printf('Schema DDFV\n');
else
    printf('=====================================\n');
    printf('Choix du schÃ©ma Ã  utiliser :\n');
    printf('1) VF4\n');
    printf('2) VF4 version EGH\n');
    printf('3) DDFV\n');

    choix_schema=-1;
    while ((choix_schema<1) | (choix_schema>4))
        choix_schema=input("Faites votre choix :");
    end;
end;

// Choix de la methode utilisee

printf('=====================================\n');
printf('Choix de la fonction utiliser :\n');
printf('1) Methode de la puissance inverse \n');
printf('2) Methode QR \n');

choix_methode=-1;
while ((choix_methode<1) | (choix_methode>2))
    choix_methode=input("Faites votre choix :");
end;

select choix_methode
case 1 then
    nom_methode='methode_puissance_inverse';

case 2 then
    nom_methode='methode_QR';
end

for k=nb_dep:nb_fin     // boucle sur les diffÃ©rents maillages

    // Chargement du maillage
    nom_maillage=maillage.nom+'_'+string(k);
    m = lecture_maillage(rep_maillages+nom_maillage);

    printf('=====================================\n');
    printf('Calcul sur le maillage '+nom_maillage+'\n');

    //************************************************************************
    // Construction du schema et resolution du systeme lineaire
    //************************************************************************

    // Construction de la matrice et du second membre

    select choix_schema
    case 1
        [Mat_Rigidite,Mat_Div,Mat_Masse,F]=const_schema_VF4(m,donnees);
        Mat_Div_tild=Mat_Div(1:$-1,:);
        taille_div=size(Mat_Div);
        A=[Mat_Rigidite Mat_Div_tild'; Mat_Div_tild spzeros(taille_div(1)-1,tail
le_div(1)-1)];
        nom_methode='VF4';
    case 2
        [Mat_Rigidite,Mat_Div,Mat_Masse,F]=const_schema_VF4_EGH(m,donnees);
        Mat_Div_tild=Mat_Div(1:$-1,:);
        taille_div=size(Mat_Div);
        A=[Mat_Rigidite Mat_Div_tild'; Mat_Div_tild spzeros(taille_div(1)-1,tail
le_div(1)-1)];
```

Programme schema_VF.sce

```
        nom_methode='VF4_EGH';
    case 3
        [Mat_Rigidite,Mat_Div,Mat_Masse,F]=const_schema_DDFV(m,donnees);
        Mat_Div_tild=Mat_Div(1:$-1,:);
        taille_div=size(Mat_Div);
        A=[Mat_Rigidite 2*Mat_Div_tild'; 2*Mat_Div_tild spzeros(taille_div(1)-1,
taille_div(1)-1)];
        nom_methode='DDFV';
    end;

    b=taille_div;
    btild=size(Mat_Div_tild);

    if choix_scilab==1 then
        Atest=Mat_Masse^(-1)*Mat_Div*Mat_Rigidite^(-1)*Mat_Div';
        vecteur_vp=gsort(spec(full(Atest)));
        printf('=====================================\n');
        printf('cte inf_sup=%8.5e \n' ,sqrt(vecteur_vp($-1)));
        printf('Deuxieme valeur propre=%8.5e \n' ,sqrt(vecteur_vp($-2)));
        printf('Troisieme valeur propre=%8.5e \n' ,sqrt(vecteur_vp($-3)));

    end


    //************************************************************************
    // Calcul des valeurs propres
    //************************************************************************

    printf('=====================================\n');

        if choix_methode==1 then

            printf('Methode de la puissance\n');

            [c_inf_sup,vp_2,vp_3,it1,it2,it3,vpropre]=valeur_propre_puissance_in
verse(k,Mat_Rigidite,Mat_Div_tild,Mat_Masse);

            // Nombre d iterations

            vecteur_it1(k)=it1;
            vecteur_it2(k)=it2;
            vecteur_it3(k)=it3;

            printf('Nb iterations=%g  cte inf_sup=%8.5e \n' ,it1,c_inf_sup);
            printf('Nb iterations=%g  Deuxieme valeur propre=%8.5e \n' ,it2,vp_2
);
            printf('Nb iterations=%g  Troisieme valeur propre=%8.5e \n' ,it3,vp_
3);

        else

            printf('Methode QR\n');
            [c_inf_sup,vp_2,vp_3,it,vpropre]=valeur_propre_QR(k,Mat_Rigidite,Mat
_Div_tild,Mat_Masse);
            vecteur_it(k)=it;

            printf('Nb iterations=%g\ncte inf_sup=%8.5e \n' ,it,c_inf_sup);
            printf('Deuxieme valeur propre=%8.5e \n' ,vp_2);
            printf('Troisieme valeur propre=%8.5e \n' ,vp_3);
        end


        // Valeur des valeurs propres

        vecteur_c_inf_sup(k)=c_inf_sup;
```

Programme schema_VF.sce

```
        vecteur_vp2(k)=vp_2;
        vecteur_vp3(k)=vp_3;

        // Vecteur pas

        vecteur_pas(k)=max(m.aretes(:,_MES));
    end


//***************************************************************************
// Mode propre
//***************************************************************************


printf('=======================================\n');
printf('Choix de tracer le mode propre :\n');
printf('1) Oui \n');
printf('2) Non \n');

choix_mode=-1;
while ((choix_mode<1) | (choix_mode>2))
    choix_mode=input("Faites votre choix :");
end;


if choix_mode==1
    clf;
    P=(full(Mat_Masse))^(-1/2)*vpropre;
    scf(1);
    trace_fonction_duale(m,P,1,0);
    xtitle(['Mode propre '; donnees.nom; m.nom]);
    nom_mode_propre=sprintf('Mode_propre_pression_%s_%s',nom_maillage,nom_method
e );
    xs2png(1,nom_mode_propre)


end


//***************************************************************************
// Trace
//***************************************************************************

// Construction du fichier vecteurs a trace //
if choix_methode==1 then

    nom_fichier=sprintf('valeur_vp_%s_%s_de_%g_a_%g.dat',maillage.nom,nom_method
e,nb_dep,nb_fin);
    fichier=mopen(nom_fichier,'w');
    mfprintf(fichier,'%g & \t %g & \t %g & \t %g & \t %g & \t %g \\ \n',
 vecteur_pas,vecteur_c_inf_sup,vecteur_vp2,vecteur_vp3,int(vecteur_it1),int(vect
eur_it2),int(vecteur_it3));
    mclose(fichier);

else
    nom_fichier=sprintf('valeur_vp_%s_%s_de_%g_a_%g.dat',maillage.nom,nom_method
e,nb_dep,nb_fin);
    fichier=mopen(nom_fichier,'w');
    mfprintf(fichier,'%g & \t %g & \t %g & \t %g & \t %g \\ \n', vecteur_pas , v
ecteur_c_inf_sup , vecteur_vp2 , vecteur_vp3 , int(vecteur_it));
    mclose(fichier);

end

printf('=======================================\n');
```

Programme schema_VF.sce

```
printf('Choix de tracer valeur propre en fonction du pas :\n');
printf('1) Oui \n');
printf('2) Non \n');

choix_trace_vp=-1;
while ((choix_trace_vp<1) | (choix_trace_vp>2))
    choix_trace_vp=input("Faites votre choix :");
end;


if choix_trace_vp==1

    fic=mopen('donnees_maillage_VF.dat','w');
    mfprintf(fic,'maillage=''%s''\n nom_trace=''trace_vp_%s_%s_de_%g_a_%g.eps'''
,nom_fichier,maillage.nom,nom_methode,nb_dep,nb_fin);
    mclose(fic);

    unix_s('gnuplot trace_vp_VF.gp');
end


printf('==============  FIN  ==================\n');

//***************************************************************************
//          FIN
//***************************************************************************
```

Programme valeur_propre_puissance_inverse.sce

```
////////////////////////////////////////////////////////
/////                                                //////
/////             Methode de la puissance            //////
/////                    inverse                     //////
/////                                                //////
/////                                                //////
////////////////////////////////////////////////////////


function [c_inf_sup,vp_2,vp_3,it1,it2,it3,vpropre]=valeur_propre_puissance_inver
se(k,Mat_Rigidite,Mat_Div_tild,Mat_Masse)


    ////////////////////////////////////
    //// Initialisation des vecteurs ////

    iteration=[];
    lambda=[];
    lambda2=[];
    lambda3=[];
    res=[];


    ////////////////////////////
    //// Initialisation de q ////

    // On choisit un vecteur initial
    qq=rand(b(1),1);
    mes_Omega=ones(qq)'*Mat_Masse*ones(qq);


    // On le prend a moyenne nulle
    q=qq-ones(1,b(1))*Mat_Masse*qq/mes_Omega;
    moyenne = ones(1,b(1))*Mat_Masse*q;

    // On le normalise
    q=q/norm(q);

    ////////////////////
    //// Parametres ////

    err=1;
    err1=1;
    err2=1;
    i=1;
    eps=10^(-7);


    ////////////////////////
    //// Factorisation LU ////

    A=[Mat_Rigidite Mat_Div_tild' ; Mat_Div_tild spzeros(btild(1),btild(1))];

    lu_A=umf_lufact(A);


    ////////////////////////////////
    //// Methode de la puissance ////



    while err>eps

        i=i+1;
```

Programme valeur_propre_puissance_inverse.sce

```
        F=-Mat_Masse*q;
        F=F(1:$-1);

        q1=[zeros(b(2),1);F];

        p1=umf_lusolve(lu_A,q1);
        p1(1:b(2))=[];
        p=[p1;0];

        p=p-ones(1,b(1))*Mat_Masse*p/mes_Omega;


        // On stocke la valeur de lambda a chaque itÃ©ration
        lambda(i)=1/norm(p);

        err=abs(lambda(i)-lambda(i-1))/abs(lambda(i));

        q=p/norm(p);

        // On stocke les itÃ©rations de lambda a chaque itÃ©ration
        iteration(i-1)=i;

        // On stocke le residu
        res(i-1)=err;

    end
    lambda(1)=[];

    vpropre=q;


    ////////////////////////////////////////////
    //// Recherche de la deuxieme valeur propre ////


    ii=1;


    // On choisit un vecteur initial
    qq=rand(b(1),1);

    // On le prend a moyenne nulle
    q=qq-ones(1,b(1))*Mat_Masse*qq/mes_Omega;

    // On le place dans l'orthogonal du vecteur propre associÃ© Ã  lambda
    norme_vpropre=vpropre'*Mat_Masse*vpropre;
    q=q-(vpropre'*Mat_Masse*q)*(vpropre/norme_vpropre);

    // On le norme
    q=q/norm(q);


    while err1>eps

        ii=ii+1;

        F=-Mat_Masse*q;
        F=F(1:$-1);

        q1=[zeros(b(2),1);F];

        p1=umf_lusolve(lu_A,q1);
        p1(1:b(2))=[];
        p=[p1;0];
```

Programme valeur_propre_puissance_inverse.sce

```
        p=p-ones(1,b(1))*Mat_Masse*p/mes_Omega;

        lambda2(ii)=1/norm(p);

        err1=abs(lambda2(ii)-lambda2(ii-1))/abs(lambda2(ii));

        q=p/norm(p);
        q=q-(vpropre'*Mat_Masse*q)*(vpropre/norme_vpropre);
        q=q/norm(q);

    end
    vpropre2=q/norm(q);


    ////////////////////////////////////////////
    //// Recherche de la troisieme valeur propre ////


    i2=1;


    // On choisit un vecteur initial
    qq=rand(b(1),1);

    // On le prend a moyenne nulle
    q=qq-ones(1,b(1))*Mat_Masse*qq/mes_Omega;

    // On le place dans l'orthogonal du vecteur propre associÃ© Ã  lambda
    norme_vpropre2=vpropre2'*Mat_Masse*vpropre2;
    q=q-(vpropre'*Mat_Masse*q)*(vpropre/norme_vpropre)-(vpropre2'*Mat_Masse*q)*(
vpropre2/norme_vpropre2);

    // On le norme
    q=q/norm(q);


    while err2>eps

        i2=i2+1;

        F=-Mat_Masse*q;
        F=F(1:$-1);

        q1=[zeros(b(2),1);F];

        p1=umf_lusolve(lu_A,q1);
        p1(1:b(2))=[];
        p=[p1;0];

        p=p-ones(1,b(1))*Mat_Masse*p/mes_Omega;

        lambda3(i2)=1/norm(p);

        err2=abs(lambda3(i2)-lambda3(i2-1))/abs(lambda3(i2));

        q=p/norm(p);

        q=q-(vpropre'*Mat_Masse*q)*(vpropre/norme_vpropre)-(vpropre2'*Mat_Masse*
q)*(vpropre2/norme_vpropre2);
        q=q/norm(q);

    end


    ////////////////////////
```

Programme valeur_propre_puissance_inverse.sce

```
    //// VALEURS OBTENUES ////
    ////////////////////////


    c_inf_sup=sqrt(lambda($));
    vp_2=sqrt(lambda2($));
    vp_3=sqrt(lambda3($));



    ////////////////////////////
    //// NOMBRE D ITERATIONS///
    ////////////////////////////

    it1=i;
    it2=ii;
    it3=i2;

    if choix_trace==1


        // Construction du fichier vecteurs a trace

        deletefile(donnees_vp);
        write(donnees_vp,[iteration,lambda,res]);

    end

endfunction
```

Programme valeur_propre_QR.sce

```
////////////////////////////////////////////////////////
/////                                                //////
/////                    Methode QR                  //////
/////                                                //////
/////                                                //////
////////////////////////////////////////////////////////


function [c_inf_sup,vp_2,vp_3,it,vpropre]=valeur_propre_QR(k,Mat_Rigidite,Mat_Di
v_tild,Mat_Masse)


    //////////////////////////
    //// Factorisation LU ////

    A=[Mat_Rigidite Mat_Div_tild' ; Mat_Div_tild spzeros(btild(1),btild(1))];
    lu_A=umf_lufact(A);



    ////////////////////////////
    //// Initialisation de Q ////

    // On choisit 3 vecteurs indÃ©pendants alÃ©atoires
    q1=rand(b(1),1);
    q2=rand(b(1),1);
    q3=rand(b(1),1);

    // Mesure de \Omega
    mes_Omega=ones(q1)'*Mat_Masse*ones(q1);

    // On les prend a moyenne nulle
    q1=q1-ones(1,b(1))*Mat_Masse*q1/mes_Omega;
    q2=q2-ones(1,b(1))*Mat_Masse*q2/mes_Omega;
    q3=q3-ones(1,b(1))*Mat_Masse*q3/mes_Omega;

    // On les rend M-orthogonaux
    norme_q1=q1'*Mat_Masse*q1;
    q2=q2-(q1'*Mat_Masse*q2)*(q1/norme_q1);

    norme_q2=q2'*Mat_Masse*q2;
    q3=q3-(q1'*Mat_Masse*q3)*(q1/norme_q1)-(q2'*Mat_Masse*q3)*(q2/norme_q2);
    norme_q3=q3'*Mat_Masse*q3;

    // On les norme
    q1=q1/sqrt(norme_q1);
    q2=q2/sqrt(norme_q2);
    q3=q3/sqrt(norme_q3);


    // Matrice constituee des 3 vecteurs
    Q=[q1 q2 q3];


    ////////////////////
    //// Parametres ////

    err1=1;
    err2=1;
    err3=1;
    eps=10^-7;
    i=1;

    iteration=[];
    vp1=[];
```

Programme valeur_propre_QR.sce

```
    vp2=[];
    vp3=[];

    res1=[];
    res2=[];
    res3=[];


    while (err1>eps) | (err2>eps) | (err3>eps)

        i=i+1;

        F=-Mat_Masse*Q;
        F($,:)=[];

        Q1=[zeros(b(2),3);F];

        P1=umf_lusolve(lu_A,Q1);
        P1(1:b(2),:)=[];
        P=[P1;0 0 0];

        p1=(ones(1,b(1))*Mat_Masse*P(:,1))*ones(q1);
        p2=(ones(1,b(1))*Mat_Masse*P(:,2))*ones(q1);
        p3=(ones(1,b(1))*Mat_Masse*P(:,3))*ones(q1);

        P=P-[p1 p2 p3]/mes_Omega;

        [Qtild,Rtild]=decompositionQR(P,Mat_Masse,3);
        B=Qtild'*Mat_Masse*Q*Rtild^-1;
        [Y,D]=spec(B);

        Q=Qtild*Y;

        [vp,ordre_vp]=gsort([D(1,1) D(2,2) D(3,3)]);

        vp1(i)=vp(3);
        vp2(i)=vp(2);
        vp3(i)=vp(1);

        err1=abs(vp1(i)-vp1(i-1))/abs(vp1(i));
        err2=abs(vp2(i)-vp2(i-1))/abs(vp2(i));
        err3=abs(vp3(i)-vp3(i-1))/abs(vp3(i));

        // On stocke les itÃ©rations de lambda a chaque itÃ©ration
        iteration(i-1)=i;

        // On stocke le residu
        res1(i-1)=err1;
        res2(i-1)=err2;
        res3(i-1)=err3;

    end


    ////////////////////////
    //// VECTEUR PROPRE ////
    ////////////////////////

    vpropre=Q(:,ordre_vp(3));
    vpropre2=Q(:,ordre_vp(2));
    vpropre3=Q(:,ordre_vp(1));



    ////////////////////////
```

Programme valeur_propre_QR.sce

```
    //// VALEURS OBTENUES ////
    ////////////////////////


    c_inf_sup=sqrt(vp1($));
    vp_2=sqrt(vp2($));
    vp_3=sqrt(vp3($));



    ////////////////////////////
    //// NOMBRE D ITERATIONS////
    ////////////////////////////

    it=i;
    if choix_trace==1


        // Construction du fichier vecteurs a trace
        vp1(1)=[];


        deletefile(donnees_vp);
        write(donnees_vp,[iteration,vp1,res1,res2,res3]);

    end

endfunction
```

Programme schema_Q1P0.sce

```
////////////////////////////////////////////////////////
/////                                                //////
/////         Schema de elements finis Q1/P0         //////
/////               pour pb Stokes 2D                //////
/////                                                //////
/////                                                //////
////////////////////////////////////////////////////////



//************************************************************************
// On efface toutes les variables et on charge les fonctions utiles
//************************************************************************


clear;
clc
lines(0);
stacksize('max');
warning('off');

exec 'decompositionQR.sci';

//************************************************************************
// Saisie par l'utilisateur des donnees du calcul
//************************************************************************


// On prend pour Omega le carre unite

// On le divise en M*N cellules

M1= [4 8 16 32 64 128]; // Nombre de colonnes

// On choisit si on fait la verification Scilab

printf('=========================================\n');
printf('Choix de verifier avec Scilab :\n');
printf('1) On fait la verification\n');
printf('2) On ne fait pas la verification\n');

choix_scilab=-1;
while (int(choix_scilab)<=0) | (int(choix_scilab)>2)
    choix_scilab=input("Faites votre choix :");
end;


// Choix du maillage

printf('=========================================\n');
printf('Choix du maillage :\n');
printf('1) Carre uniforme \n');
printf('2) Carre non uniforme \n');

choix_maillage=-1;
while ((choix_maillage<1) | (choix_maillage>2))
    choix_maillage=input("Faites votre choix :");
end;

if  choix_maillage==1 then
    N1=M1; // Nombre de lignes
elseif  choix_maillage==2 then
    N1=[4 16 64 256];
end
```

Programme schema_Q1P0.sce

```
// Choix du raffinement

printf('=========================================\n');
printf('Choix du raffinement minimum :\n');
kmin=-1;
while ((int(kmin)<1) | (choix_maillage>length(N1)))
    kmin=input("Faites votre choix :");
end;

printf('=========================================\n');
printf('Choix du raffinement maximum :\n');
kmax=-1;
while ((int(kmax)<1) | (choix_maillage>length(N1)))
    kmax=input("Faites votre choix :");
end;


// Choix de la methode utilisee

printf('=========================================\n');
printf('Choix de la fonction utiliser :\n');
printf('1) Methode de la puissance inverse \n');
printf('2) Methode QR \n');

choix_methode=-1;
while ((choix_methode<1) | (choix_methode>2))
    choix_methode=input("Faites votre choix :");
end;

select choix_methode

case 1 then
    nom_methode='methode_puissance_inverse';

case 2 then
    nom_methode='methode_QR';

end

select choix_maillage

case 1 then
    nom_trace=sprintf('Trace_vp_en_fonction_pas_Q1_P0_uniforme_de_%g_a_%g_%s.eps
',kmin,kmax,nom_methode);
    valeur_vp=sprintf('valeur_vp_maillage_unif_Q1P0_de_%g_a_%g_%s.dat',kmin,kmax
,nom_methode);
    nom_pente='Courbe de pente 1';
    nom_mode=sprintf('Mode_propre_Q1_P0_unif_%g.png',kmin);
case 2 then
    nom_trace=sprintf('Trace_vp_en_fonction_pas_Q1_P0_non_uniforme_de_%g_a_%g_%s
.eps',kmin,kmax,nom_methode);
    valeur_vp=sprintf('valeur_vp_maillage_non_unif_Q1P0_de_%g_a_%g_%s.dat',kmin,
kmax,nom_methode);
    nom_pente='Courbe de pente 2.5';
    nom_mode=sprintf('Mode_propre_Q1_P0_non_unif_%g.png',kmin);
end


for t=kmin:kmax

    M=M1(t);
    N=N1(t);

    // Nombre de sommets
```

Programme schema_Q1P0.sce

```
Ns=(N+1)*(M+1);

// Nombre de sommets interne

Ns_int=Ns-2*(N+M);

// On choisi le pas du maillage

h1=1/M; // Pas en x
h2=1/N;  // Pas en y

printf('=========================================\n');
printf('Nombre de colonnes = %g Nombre de lignes = %g\n',M,N)
printf('Nombre de sommets internes = %g\n',Ns_int);
printf('Pas en x = %g Pas en y = %g\n',h1,h2)


//****************************************************************************
// Construction des matrices
//****************************************************************************


////////////////////////
//// Initialisation ////
////////////////////////


// Matrice du laplacien
M_laplacien=spzeros(Ns_int,Ns_int);

// Matrice de rigidite
Mat_Rigidite=spzeros(2*Ns_int,2*Ns_int);

// Matrice de divergence
Mat_Div=spzeros(M*N,2*Ns_int);



/////////////////////////////////////////////////////
//// Fonction qui a une cellule lui associe la place ////
////    dans la matrice de chacun de ses sommets     ////
/////////////////////////////////////////////////////

// Pour une composante de la vitesse

function [v]=place_matrice_vitesse(i,j)

    v=-ones(1,4);

    if i~=1 & j~=1
        v(1)=(j-1)+(M-1)*(i-2);
    end

    if i~=1 & j~=M
        v(2)=j+(M-1)*(i-2);
    end

    if i~=N & j~=M
        v(3)=j+(M-1)*(i-1);
    end

    if i~=N & j~=1
        v(4)=(j-1)+(M-1)*(i-1);
    end
```

Programme schema_Q1P0.sce

```
endfunction

// Pour la pression

function [v_p]=place_matrice_pression(i,j)

    v_p=j+M*(i-1);

endfunction



////////////////////////////////////////////////////
//// Creation des matrices de fonctions de bases ////
////////////////////////////////////////////////////


//// Coefficients souvent utilises ////

d=1/h1^2+1/h2^2; // sur la diagonale
d1=-1/h1^2+1/(2*h2^2);
d2=-1/2*d;
d3=1/(2*h1^2)-1/h2^2;

//// Construction ////

A=((h1*h2)/3)*[d d1 d2 d3 ; d1 d d3 d2 ; d2 d3 d d1 ; d3 d2 d1 d];

C_h1=(h1)/2*[-1;-1;1;1];
C_h2=(h2)/2*[-1;1;1;-1];



//////////////////////////////
//// Creation des matrices ////
//////////////////////////////

//// Matrice de masse en pression ////

Mat_Masse=h1*h2*speye(N*M,N*M);


//// Matrice de rigidite ////

// On se place dans la cellule (i,j)
for i=1:N
    for j=1:M

        [v]=place_matrice_vitesse(i,j);

        // On remplit les matrices
        for k=1:4
            for l=1:4

                if v(k)~=-1 & v(l)~=-1
                    M_laplacien(v(k),v(l))= M_laplacien(v(k),v(l))+A(k,l);

                end

            end
        end
    end
end

Mat_Rigidite(1:Ns_int,1:Ns_int)=M_laplacien;
```

Programme schema_Q1P0.sce

```
    Mat_Rigidite((Ns_int)+1:2*Ns_int,(Ns_int)+1:2*Ns_int)=M_laplacien;


    //// Matrice de divergence ////

    for i=1:N
        for j=1:M

            v_p=place_matrice_pression(i,j);
            [v_u]=place_matrice_vitesse(i,j);

            for l_p=1:4

                if v_u(l_p)~=-1

                    Mat_Div(v_p,v_u(l_p))=Mat_Div(v_p,v_u(l_p))-C_h2(l_p);
                    Mat_Div(v_p,Ns_int+v_u(l_p))=Mat_Div(v_p,Ns_int+v_u(l_p))-C_
h1(l_p);

                end

            end

        end
    end


    //// Mode en damier ////

    for i=1:N
        for j=1:M

            place_damier=place_matrice_pression(i,j);
            v_damier(place_damier)=(-1)^(i+j);

        end
    end

    norme_damier=norm(v_damier);


    //************************************************************************
    // Choix des tests effectues
    //************************************************************************


    //// test pour matrice

    //test_matrices='true';
    test_matrices='false';


    //// Erreur

    //  test_erreur='true';
    test_erreur='false';



    //************************************************************************
    // Si on teste que les matrices sont bonnes
    //************************************************************************


    if test_matrices=='true'
```

Programme schema_Q1P0.sce

```
        Mat_Champs=rand(2,2);


        function [z]=test(x,y)
            z=Mat_Champs*[x;y];
        endfunction


        for i_test=1:N-1
            for j_test=1:M-1

                u_test=test(j_test*h1,i_test*h2);
                U(j_test+(i_test-1)*(M-1))=u_test(1);
                U(Ns_int+j_test+(i_test-1)*(M-1))=u_test(2);

            end
        end

        Div=trace(Mat_Champs);

        val=Mat_Div*U;


    end



    //************************************************************************
    // Calcul des valeurs propres
    //************************************************************************

    if choix_scilab==1 then

        ///// VERIF SCILAB

        Atest=Mat_Masse^(-1)*Mat_Div*Mat_Rigidite^(-1)*Mat_Div';
        vecteur_vp=gsort(spec(full(Atest)));
        printf('=======================================\n');
        printf('cte inf_sup=%8.5e \n' ,sqrt(vecteur_vp($-2)));
        printf('Deuxieme valeur propre=%8.5e \n' ,sqrt(vecteur_vp($-3)));
        printf('Troisieme valeur propre=%8.5e \n' ,sqrt(vecteur_vp($-4)));

    end


    // Vecteur pas

    vecteur_pas(t)=sqrt(h1^2+h2^2);

    ////////////////////////
    //// Factorisation LU ////

    Mat_Div_tild=Mat_Div(1:$-2,:);

    btild=size(Mat_Div_tild);
    b=size(Mat_Div);

    Mat_Resolution=[Mat_Rigidite Mat_Div_tild' ; Mat_Div_tild spzeros(btild(1),b
tild(1))];

    lu_A=umf_lufact(Mat_Resolution);
```

Programme schema_Q1P0.sce

```
if choix_methode==1 then

    ////////////////////////////////////////
    //// METHODE DE LA PUISSANCE INVERSE ////
    ////////////////////////////////////////

    ////////////////////////////////////
    //// Initialisation des vecteurs ////

    iteration=[];

    lambda=[];
    lambda2=[];
    lambda3=[];

    res=[];


    ////////////////////////////
    //// Initialisation de q ////

    // On choisit un vecteur initial
    qq=rand(v_damier);
    mes_Omega=ones(qq)'*Mat_Masse*ones(qq);

    // On le prend a moyenne nulle
    q=qq-ones(v_damier)'*Mat_Masse*qq/mes_Omega;

    // On le rend orthogonal au damier
    q=q-(v_damier'*q)*v_damier/(norme_damier)^2;

    // On le normalise
    q=q/norm(q);


    ////////////////////
    //// Parametres ////

    err=1;
    err1=1;
    err2=1;

    i=1;
    ii=1;
    i2=1;

    eps=10^(-9);


    ////////////////////
    //// Algorithme ////


    while err>eps

        i=i+1;

        F=-Mat_Masse*q;

        F_tild=F(1:$-2);

        q1=[zeros(b(2),1);F_tild];

        p1=umf_lusolve(lu_A,q1);
        p1(1:b(2))=[];
```

Programme schema_Q1P0.sce

```
        p=[p1;0;0];

        p=p-ones(v_damier)'*Mat_Masse*p/mes_Omega;

        p=p-(v_damier'*p)*v_damier/(norme_damier)^2;

        // On stocke la valeur de lambda a chaque itÃ©ration
        lambda(i)=1/norm(p);

        err=abs(lambda(i)-lambda(i-1))/abs(lambda(i));

        q=p/norm(p);

        // On stocke les itÃ©rations de lambda a chaque itÃ©ration
        iteration(i-1)=i;

        // On stocke le residu
        res(i-1)=err;

    end
    lambda(1)=[];

    vpropre=q;



    ////////////////////////////////////////////////
    //// Recherche de la deuxieme valeur propre ////

    // On choisit un vecteur initial
    qq=rand(v_damier);

    // On le prend a moyenne nulle
    q=qq-ones(v_damier)'*Mat_Masse*qq/mes_Omega;

    // On le place dans l'orthogonal du vecteur propre associÃ© Ã  lambda
    norme_vpropre=vpropre'*Mat_Masse*vpropre;
    q=q-(vpropre'*Mat_Masse*q)*(vpropre/norme_vpropre);

    // On le rend orthogonal au damier
    q=q-(v_damier'*q)*v_damier/(norme_damier)^2;


    // On le norme
    q=q/norm(q);


    while err1>eps

        ii=ii+1;

        F=-Mat_Masse*q;

        F_tild=F(1:$-2);

        q1=[zeros(b(2),1);F_tild];

        p1=umf_lusolve(lu_A,q1);
        p1(1:b(2))=[];
        p=[p1;0;0];

        p=p-ones(v_damier)'*Mat_Masse*p/mes_Omega;

        p=p-(v_damier'*p)*v_damier/(norme_damier)^2;
```

Programme schema_Q1P0.sce

```
            lambda2(ii)=1/norm(p);

            err1=abs(lambda2(ii)-lambda2(ii-1))/abs(lambda2(ii));

            q=p/norm(p);
            q=q-(vpropre'*Mat_Masse*q)*(vpropre/norme_vpropre);
            q=q/norm(q);

        end
        lambda2(1)=[];

        vpropre2=q;


        ////////////////////////////////////////////////
        //// Recherche de la troisieme valeur propre ////

        // On choisit un vecteur initial
        qq=rand(v_damier);


        // On le prend a moyenne nulle
        q=qq-ones(v_damier)'*Mat_Masse*qq/mes_Omega;

        // On le place dans l'orthogonal du vecteur propre associÃ© Ã  lambda
        norme_vpropre2=vpropre2'*Mat_Masse*vpropre2;
        q=q-(vpropre'*Mat_Masse*q)*(vpropre/norme_vpropre)-(vpropre2'*Mat_Masse*
q)*(vpropre2/norme_vpropre2);

        // On le rend orthogonal au damier
        q=q-(v_damier'*q)*v_damier/(norme_damier)^2;

        // On le norme
        q=q/norm(q);


        while err2>eps

            i2=i2+1;

            F=-Mat_Masse*q;

            F_tild=F(1:$-2);

            q1=[zeros(b(2),1);F_tild];

            p1=umf_lusolve(lu_A,q1);
            p1(1:b(2))=[];
            p=[p1;0;0];

            p=p-ones(v_damier)'*Mat_Masse*p/mes_Omega;

            p=p-(v_damier'*p)*v_damier/(norme_damier)^2;

            lambda3(i2)=1/norm(p);

            err2=abs(lambda3(i2)-lambda3(i2-1))/abs(lambda3(i2));

            q=p/norm(p);
            q=q-(vpropre'*Mat_Masse*q)*(vpropre/norme_vpropre)-(vpropre2'*Mat_Ma
sse*q)*(vpropre2/norme_vpropre2);
            q=q/norm(q);

        end
        lambda3(1)=[];
```

Programme schema_Q1P0.sce

```
        vpropre3=q;


        //////////////////////////
        //// Valeurs obtenues ////

        c_inf_sup=sqrt(lambda($));
        vp_2=sqrt(lambda2($));
        vp_3=sqrt(lambda3($));


        ////////////////////////////
        //// Nombre d iterations ////

        vecteur_it1(t)=i;
        vecteur_it2(t)=ii;
        vecteur_it3(t)=i2;


        ////////////////////
        //// Affichage ////

        printf('=========================================\n');
        printf('Nb iterations=%g  cte inf_sup=%8.5e \n' ,i,c_inf_sup);
        printf('Nb iterations=%g  Deuxieme valeur propre=%8.5e \n' ,ii,vp_2);
        printf('Nb iterations=%g  Troisieme valeur propre=%8.5e \n' ,i2,vp_3);



        //*************************************************************************
****
        // Choix de tracer vp et residu en fonction des iterations
        //*************************************************************************
****

        printf('=========================================\n');
        printf('Choix de tracer \lambda et rÃ©sidu en fonction itÃ©rations :\n')
;
        printf('1) Oui \n');
        printf('2) Non \n');

        choix_trace=-1;
        while ((choix_trace<1) | (choix_trace>2))
            choix_trace=input("Faites votre choix :");
        end;


        if choix_trace==1

            // Construction du fichier vecteurs a trace //

            donnees_vp_Q1_P0=sprintf('donnee_trace_vp_maillage_Q1_P0_%g.dat',t);
            fic3=mopen('donnees_maillage.dat','w');
            mfprintf(fic3,'maillage=''donnee_trace_vp_maillage_Q1_P0_%g.dat'';\n
k=%g;...
            \n nom_trace=''trace_maillage_Q1_P0_%g.eps''',t,t,t);
            mclose(fic3);


            deletefile(donnees_vp_Q1_P0);
            write(donnees_vp_Q1_P0,[iteration,lambda,res]);


            // Fichier contenant valeur de la constante inf-sup au carre
```

Programme schema_Q1P0.sce

```
        valeur=sprintf('valeur_lambda_%g_Q1_P0.dat',t);
        fic4=mopen(valeur,'w');
        mfprintf(fic4,'lambda=.17%g;\n',lambda($));
        mclose(fic4);


        // Gnuplot

        unix_s('gnuplot trace_residu_Q1P0.gp');

    end

else

    ////////////////////
    //// METHODE QR ////
    ////////////////////



    ////////////////////
    //// Parametres ////

    err1=1;
    err2=1;
    err3=1;
    eps=10^-7;
    i=1;

    vp1=[];
    vp2=[];
    vp3=[];


    ////////////////////////////
    //// Initialisation de Q ////

    // On choisit 3 vecteurs indÃ©pendants alÃ©atoires
    q1=rand(b(1),1);
    q2=rand(b(1),1);
    q3=rand(b(1),1);

    // Mesure de \Omega
    mes_Omega=ones(q1)'*Mat_Masse*ones(q1);

    // On les prends a moyenne nulle
    q1=q1-ones(1,b(1))*Mat_Masse*q1/mes_Omega;
    q2=q2-ones(1,b(1))*Mat_Masse*q2/mes_Omega;
    q3=q3-ones(1,b(1))*Mat_Masse*q3/mes_Omega;

    // On les rend orthogonaux au damier
    q1=q1-(v_damier'*q1)*v_damier/(norme_damier)^2;
    q2=q2-(v_damier'*q2)*v_damier/(norme_damier)^2;
    q3=q3-(v_damier'*q3)*v_damier/(norme_damier)^2;

    // On les rend M-orthogonaux
    norme_q1=q1'*Mat_Masse*q1;
    q2=q2-(q1'*Mat_Masse*q2)*(q1/norme_q1);

    norme_q2=q2'*Mat_Masse*q2;
    q3=q3-(q1'*Mat_Masse*q3)*(q1/norme_q1)-(q2'*Mat_Masse*q3)*(q2/norme_q2);
    norme_q3=q3'*Mat_Masse*q3;

    q1=q1/sqrt(norme_q1);
```

Programme schema_Q1P0.sce

```
    q2=q2/sqrt(norme_q2);
    q3=q3/sqrt(norme_q3);

    // On construit le vecteur Q_0
    Q=[q1 q2 q3];

    while (err1>eps) | (err2>eps) | (err3>eps)

        i=i+1;

        F_tild=-Mat_Masse*Q;

        F_tild($-1:$,:)=[];

        Q1=[zeros(b(2),3);F_tild];

        P1=umf_lusolve(lu_A,Q1);
        U=P1(1:b(2),:);
        P1(1:b(2),:)=[];
        P=[P1;0 0 0;0 0 0];

        p1=(ones(1,b(1))*Mat_Masse*P(:,1))*ones(q1);
        p2=(ones(1,b(1))*Mat_Masse*P(:,2))*ones(q1);
        p3=(ones(1,b(1))*Mat_Masse*P(:,3))*ones(q1);

        P=P-[p1 p2 p3]/mes_Omega;

        p1=(v_damier'*P(:,1))*v_damier;
        p2=(v_damier'*P(:,2))*v_damier;
        p3=(v_damier'*P(:,3))*v_damier;

        P=P-[p1 p2 p3]/(norme_damier)^2;

        [Qtild,Rtild]=decompositionQR(P,Mat_Masse,3);
        B=Qtild'*Mat_Masse*Q*Rtild^-1;
        [Y,D]=spec(B);

        Q=Qtild*Y;

        [vp,ordre_vp]=gsort([D(1,1) D(2,2) D(3,3)]);


        vp1(i)=vp(3);
        vp2(i)=vp(2);
        vp3(i)=vp(1);

        err1=abs(vp1(i)-vp1(i-1))/abs(vp1(i));
        err2=abs(vp2(i)-vp2(i-1))/abs(vp2(i));
        err3=abs(vp3(i)-vp3(i-1))/abs(vp3(i));

    end

    vpropre=Q(:,ordre_vp(3));
    vpropre2=Q(:,ordre_vp(2));
    vpropre3=Q(:,ordre_vp(1));


    ////////////////////////
    //// Valeurs obtenues ////

    c_inf_sup=sqrt(vp1($));
    vp_2=sqrt(vp2($));
    vp_3=sqrt(vp3($));
```

Programme schema_Q1P0.sce

```
    ////////////////////////////
    //// Nombre d iterations///

    it=i;
    vecteur_it(t)=it;

    ////////////////////
    //// Affichage ////

    printf('=======================================\n');
    printf('Nb iterations=%g  cte inf_sup=%8.5e \n' ,i,c_inf_sup);
    printf('Deuxieme valeur propre=%8.5e \n' ,vp_2);
    printf('Troisieme valeur propre=%8.5e \n' ,vp_3);

end


// Valeur des valeurs propres

vecteur_c_inf_sup(t)=c_inf_sup;
vecteur_vp2(t)=vp_2;
vecteur_vp3(t)=vp_3;



//***************************************************************************
// Erreur entre solution exacte et approchee
//***************************************************************************

if test_erreur=='true'

    ////////////////////////
    //// Solution exacte ////
    ////////////////////////

    // Pour la vitesse

    function [u]=uexacte(x,y)

        u1=-2*%pi*(sin(%pi*x)).^2.*cos(%pi*y).*sin(%pi*y);
        u2= 2*%pi*(sin(%pi*y)).^2.*cos(%pi*x).*sin(%pi*x);

        u=[u1,u2];

    endfunction


    for m=1:M-1
        for n=1:N-1

            u=uexacte(m*h1,n*h2);
            U_ex(m+(n-1)*(M-1))=u(1);
            U_ex(Ns_int+m+(n-1)*(M-1))=u(2);

        end
    end


    // Pour la pression

    function [p]=pexacte(x,y)

        p=x+y-1;
```

Programme schema_Q1P0.sce

```
    endfunction

    for m_p=1:M
        for n_p=1:N

            P_ex(place_matrice_pression(n_p,m_p))=pexacte((h1/2)+h1*(m_p-1),
(h2/2)+h2*(n_p-1));

        end
    end


    // Vecteur solution exacte

    S_ex=[U_ex;P_ex];



    ////////////////////
    //// Terme source ////
    ////////////////////


    // Pour la vitesse

    function [f]=fonction_source(x,y)

        f1=-4*(%pi)^3*sin((%pi)*y).*cos((%pi)*y).*(3*(sin((%pi)*x)).^2-...
        (cos((%pi)*x)).^2)+1;
        f2= 4*(%pi)^3*sin((%pi)*x).*cos((%pi)*x).*(3*(sin((%pi)*y)).^2-...
        (cos((%pi)*y)).^2)+1;

        f=[f1,sqrt(lambda($))f2];

    endfunction

    for m=1:M-1
        for n=1:N-1

            f=fonction_source(m*h1,n*h2);
            F(m+(n-1)*(M-1))=f(1);
            F(Ns_int+m+(n-1)*(M-1))=f(2);

        end
    end


    // Terme source final

    L=[F;zeros(P_ex)];



    ////////////////////////////
    //// Solution approchee ////
    ////////////////////////////

    b=size(Mat_Div);
    Mat_Resolution=[Mat_Rigidite Mat_Div' ; Mat_Div spzeros(b(1),b(1))];

    lu_A=umf_lufact(Mat_Resolution);

    S_app=umf_lusolve(lu_A,L);

    U_app=S_app(1:2*Ns_int);
```

Programme schema_Q1P0.sce

```
        P_app=S_app(2*Ns_int+1:$);



        ////////////////
        //// Erreur ////
        ////////////////



        e_pression=P_ex-P_app;
        err1_pression=sqrt(e_pression'*Mat_Masse*e_pression);
        err2_pression=sqrt(P_ex'*Mat_Masse*P_ex);


        err_pression(t)=err1_pression/err2_pression;
        printf('M=%g N=%g Erreur pression =%8.5e\n',M,N,err_pression(t));

    end

end



//***************************************************************************
// Mode propre
//***************************************************************************


printf('=======================================\n');
printf('Choix de tracer le mode propre :\n');
printf('1) Oui \n');
printf('2) Non \n');

choix_mode=-1;
while ((choix_mode<1) | (choix_mode>2))
    choix_mode=input("Faites votre choix :");
end;


if choix_mode==1

    clf()
    mode_propre=real((full(Mat_Masse))^(-1/2)*vpropre2);
    f=gcf();
    nbcolor=255;
    Matplot((matrix(mode_propre,N,M)-min(mode_propre))/(max(mode_propre)-min(mod
e_propre))*nbcolor)
    f.color_map=hotcolormap(nbcolor);
    colorbar(min(mode_propre),max(mode_propre));
    f.children(2).axes_visible=["off" "off" "off"];
    xs2png(0,nom_mode)

    disp(ones(1,size(Mat_Masse,1))*Mat_Masse*mode_propre)
end



//***************************************************************************
// Trace valeur propre en fonction du pas
//***************************************************************************



printf('=======================================\n');
```

Programme schema_Q1P0.sce

```
printf('Choix de tracer valeur propre en fonction du pas :\n');
printf('1) Oui \n');
printf('2) Non \n');

choix_trace_vp=-1;
while ((choix_trace_vp<1) | (choix_trace_vp>2))
    choix_trace_vp=input("Faites votre choix :");
end;



//////////////////////////////////////////
//// Fichier enregistrant les resultats ////

if choix_methode==1 then

    fichier=mopen(valeur_vp,'w');
    mfprintf(fichier,'%g & \t %g & \t %g & \t %g & \t %g & \t %g & \t %g \\ \n',
 vecteur_pas,vecteur_c_inf_sup,vecteur_vp2,vecteur_vp3,int(vecteur_it1),int(vect
eur_it2),int(vecteur_it3));
    mclose(fichier);

else

    fichier=mopen(valeur_vp,'w');
    mfprintf(fichier,'%g & \t %g & \t %g & \t %g & \t %g \\ \n', vecteur_pas , v
ecteur_c_inf_sup , vecteur_vp2 , vecteur_vp3 , int(vecteur_it));
    mclose(fichier);
end


if choix_trace_vp==1;

    fic=mopen('donnees_maillage_Q1P0.dat','w');
    mfprintf(fic,'nom_maillage=''%s''\n nom_pente=''%s''\n nom_trace=''%s_%s_de_
%g_a_%g.eps'',valeur_vp,nom_pente,nom_trace,nom_methode,kmin,kmax);
    mclose(fic);

    unix_s('gnuplot trace_vp_Q1P0.gp');
end


printf('==============  FIN  ==================\n');

//***************************************************************************
//          FIN
//***************************************************************************
```