MINI-élément et factorisation incomplètes pour la parallelisation d'un solveur de Stokes 2D : application au forgeage

Etienne Perchat

THÈSE

présentée à

L'ÉCOLE NATIONALE SUPÉRIEURE DES MINES DE PARIS

par

Etienne Perchat

en vue de l'obtention du titre de

DOCTEUR

en

SCIENCES ET GÉNIE DES MATÉRIAUX

MINI-Élément et factorisations incomplètes pour la parallélisation d'un solveur de Stokes 2D. Application au forgeage.

---

soutenue le 11 Juillet 2000, devant le jury composé de :

MM.	Michel DEVILLE	Président
	Denis AUBRY	Rapporteur
	François-Xavier ROUX	Rapporteur
	Jean-Loup CHENOT	Examinateur
	Thierry COUPEZ	Examinateur
	Etienne WEY	Examinateur
	Lionel FOURMENT	Directeur de thèse

Remerciements

Mes remerciements vont tout d'abord aux différents membres du jury qui ont bien voulu consacrer du temps à la lecture et à l'examen de ce travail : M. Deville, J. L. Chenot, T. Coupez, L. Fourment, ainsi que D. Aubry et F. X. Roux qui ont accepté d'être rapporteurs.

J'exprime toute ma gratitude à mon directeur de thèse Lionel Fourment qui m'a guidé avec beaucoup de compétence et de patience durant ces trois années. Sa gentillesse et ses qualités humaines ont largement contribué à faire de ces années une expérience enrichissante.

Je remercie aussi tout particulièrement Thierry Coupez pour l'intérêt qu'il a porté à mon travail, pour ses conseils et sa précieuse relecture.

Cette thèse est le résultat des trois années que j'ai passé au Cemef, centre de mise en forme des matériaux, au sein du groupe Thermo-Mécanique et Plasticité (TMP). Je tiens vivement à remercier la direction de L'École Nationale Supérieure des Mines de Paris de m'avoir accepté au sein de sa formation doctorale. Je suis aussi très reconnaissant à Jean-Loup Chenot, Jean-François Agassant et Michel Bellet de m'avoir accueilli au sein du Cemef, pour la confiance qu'ils m'ont accordé et pour les importants moyens qu'ils ont mis à ma disposition.

Cette thèse a été financée dans le cadre du projet européen Esprit OPTIMUM. Je remercie la CEE et tous les partenaires industriels pour leur soutien financier et l'intérêt porte à cette étude.

Ce document est bien sûr le fruit d'une réflexion personnelle mais est aussi issu d'un travail d'équipe et d'une culture très "cemefienne" de la simulation des procédés de mise en forme. Tout au long de ce travail, ma réflexion s'est nourrie des échanges et discussions scientifiques que j'ai eu notamment avec S. Marie, H. Digonnet, K. Traore et D. Vieilledent. Je les remercie chaleureusement pour leur disponibilité et leurs encouragements.

J'adresse une mention spéciale à Serge Algarotti pour avoir veillé avec tant de soin sur mes calculs ainsi qu'à Hervé et Lolo qui ont si souvent répondu à mes questions. Je leur en suis profondément reconnaissant.

Je me dois enfin de remercier tous mes collègues et amis du Cemef et d'ailleurs, pour tous les échanges si enrichissants, les conseils qu'ils m'ont donnés et pour leur amitié. Eux aussi m'ont beaucoup donné et beaucoup appris: Oliskou et Radka, Olivier, Olof, David et Mag, Erwan, Marianne, Sylvie, Charlie, Mihaela, Nicolas, Mike, Roland, Katia ...

Je tiens enfin à exprimer une profonde gratitude envers ma famille pour son soutien sans faille pendant toutes ces années.

À Frédérique et à Thomas

Table des matières

Table des figures

Liste des tableaux

Introduction

L'objectif principal de ce travail est d'accélérer, grâce au calcul parallèle, les performances d'un code éléments finis 2D utilisé dans l'industrie: Forge2®. Ce logiciel, développé au CEMEF depuis la fin des années 80, est commercialisé par la société de valorisation TRANSVALOR. Il est dédié à la simulation du procédé de forgeage de pièces présentant des symétries 2D, telles que les pièces axisymétriques. Quand elles sont possibles, les simulations 2D ont l'avantage de présenter des temps de calculs bien inférieurs aux géométries 3D. Ces temps restent toutefois trop élevés. En effet, pour un utilisateur averti, la recherche du procédé optimal pour concevoir une nouvelle pièce, demande environ cinq simulations nécessitant de une à trois heures chacune, auxquelles on doit encore ajouter les étapes préliminaires de mise en place des fichiers de données, le maillage des pièces, la définition des outillages ainsi que les étapes finales d'exploitation des résultats. Ainsi, la conception d'une pièce, et plus encore d'une gamme de forgeage, c'est à dire l'ensemble des opérations de forgeage successives permettant de réaliser la pièce finale, peut prendre de un à quelques jours pour des problèmes de petite taille (maillages de 1000 à 2000 noeuds). Ces coûts deviennent bien plus importants pour des problèmes complexes, lorsque l'on s'intéresse par exemple aux déformations des outillages, qui sont alors maillés, ou encore à l'étude des chocs thermiques (trempe de rails par exemple). Les maillages peuvent alors être de l'ordre de 5000 à 10000 noeuds voire 20000, et une seule simulation peut nécessiter plus de 24 heures.

C'est pour répondre à ces besoins que le projet européen ESPRIT OPTIMUM, dans le cadre duquel cette thèse s'est déroulée, s'est mis en place. Il implique le CEMEF et TRANSVALOR, ainsi que les forgerons espagnols GSB et FOVISA, l'entreprise allemande FORD WERKE et la société espagnole LABEIN.

Les forgerons attendent de ce projet une forte réduction des temps de simulation nécessaires à la conception d'une gamme de forgeage. Cet objectif est important car il concerne une grande classe des utilisations du logiciel Forge2®et la majorité de ses utilisateurs. Il est aussi ambitieux dans la mesure où des résultats sont espérés sur des problèmes de petite taille, pour lesquels les méthodes de résolution les plus efficaces ont peu de parallélisme intrinsèque. Un second bénéfice non négligeable est d'obtenir, pour les mêmes temps de calcul, des solutions de bien meilleure qualité grâce à des maillages beaucoup plus fins. Ce projet offre donc la perspective de traiter des problèmes de très grosse taille dans des temps non prohibitifs. Cet objectif est, a priori, plus aisé à atteindre, car pour les gros problèmes les méthodes de résolution efficaces sont mieux adaptées au parallélisme.

Une parallélisation efficace nécessite souvent de modifier en profondeur les codes de calcul. Les algorithmes doivent être revisités et adaptés sous peine d'être purement et simplement réformés. Dans ce cas d'autres algorithmes, possédant plus de parallélisme mais parfois moins efficace en séquentiel, doivent alors être mis en place. Comme nous le verrons, paralléliser un code de calcul reviens en définitive à rechercher des compromis entre la dépendance globale du problème et, la localisation des opérations et des données nécessaires à l'efficacité parallèle.

A l'origine, le logiciel FORGE2® était basé sur une discrétisation des équations en vitesse/pression avec un élément finis $P2P0$ et utilisait une formulation en vitesse avec pénalisation de la condition d'incompressibilité. On résolvait alors un problème uniquement en vitesse, pour éventuellement en déduire ensuite la pression. Ce type d'approche a l'avantage de réduire le nombre de degrés de liberté mais au détriment du conditionnement des systèmes linéaires. Une méthode directe était alors nécessaire pour les inverser. Comme nous le verrons au chapitre II , ces solveurs sont difficilement envisageables pour la résolution de problème de très grosse taille. En effet, en 2D avec les méthodes directes le temps de calcul est en $\mathcal{O}\left(Nbnoe^2\right)$, où $Nbnoe$ est le nombre de noeuds du maillage. Ainsi, passer d'un problème de 1000 noeuds à un problème de 5000 noeuds augmente le temps de calcul d'un facteur 25, et d'un facteur 400 pour 20000 noeuds. Les limites de ces méthodes, en temps mais aussi en place mémoire, font que dans la majorité des codes de calcul développé ces dernières années, on leur a préféré des solveurs itératifs. De plus, la parallélisation des méthodes directes est techniquement délicate et ne donne de bons résultats que pour des architectures partagées et un petit nombre de processeurs (inférieur à huit). Une alternative réside dans les méthodes de décomposition de domaine que nous présenterons au chapitre III . Ces méthodes sont basées sur la résolution itérative d'un problème condensé à l'interface entre les sous-domaines et nécessitent à chaque itération la résolution de problèmes locaux, de préférence avec un solveur direct. Toutefois, même si le problème condensé est mieux conditionné que le problème global avec une formulation pénalisée, il resterait dans notre cas de mauvaise, voire très mauvaise, qualité au point de compromettre la convergence de la méthode. Le préconditionnement deviendrait alors un point crucial sans que nous soyons assurés d'obtenir un solveur robuste.

Ces risques techniques nous ont donc incités à nous tourner vers une nouvelle discrétisation éléments finis basée sur le MINI-élément P1+P1. Cette formulation est similaire à celle implantée dans le code de calcul FORGE3® dédié à la simulation du forgeage en 3D. Elle a l'avantage de fournir des systèmes linéaires beaucoup mieux conditionnés. Une version parallèle de ce logiciel a été développée au CEMEF par *S. Marie* [57] selon une méthode de partitionnement de domaine qui vise, à l'inverse des méthodes de décomposition de domaine, à résoudre directement le problème global sans passer par des sous-problèmes locaux. Cette méthode que nous avons implantée sera détaillée au chapitre III .

Cette approche, quoique impliquant de modifier en profondeur le code de calcul original, nous assure d'obtenir une formulation éléments finis robuste, ayant déjà fait ses preuves. Elle nous permet de bénéficier de l'expérience acquise et du travail effectué lors de la parallélisation de FORGE3®, et apporte la perspective d'une réflexion supplémentaire sur les techniques employées et sur leurs possibles améliorations. En définitive cela s'est traduit, par un travail sur le préconditionnement séquentiel et parallèle. En effet, en 2D un solveur itératif avec un simple préconditionnement diagonal ne peut concurrencer les méthodes directes que pour des problèmes de grosse taille et fortement non-linéaires. Nous nous sommes donc penchés vers des préconditionneurs par factorisation incomplète, présentés au chapitre II , qui ont donné d'excellent résultats en séquentiel. La parallélisation de ce type de préconditionnement est par contre très délicate. Une implémentation directe mène à des récursions nécessitant de rassembler des données globales sur un seul processeur, ce qui n'est pas adapté aux architectures parallèles. Nous présenterons au chapitre IV les diverses solutions que nous avons envisagées pour construire un préconditionneur parallèle par factorisation incomplète satisfaisant. Le dernier chapitre sera consacré à des exemples et applications de notre travail sur des problèmes issus de l'industrie.

Chapitre I

Formulation mixte P1+P1 axisymétrique pour le problème de forgeage

1 Viscoplasticité et problème de Stokes généralisé

1.1 Problème de forgeage: Présentation et Formulation mécanique

L'expression mécanique du problème de forgeage est maintenant très classique. Nous nous contenterons ici de présenter les grandes lignes et les équations de base. Le lecteur intéressé pourra se reporter aux thèses de *Y. Germain* (1985) [46], *G. Surdon* (1986) [76] ou encore *N. Soyris* (1990) [75].

Mécanique des milieux continus:

On assimile le métal chaud à un milieu continu et homogène. Son écoulement lors de la mise en forme se modélise à partir des principes fondamentaux de la mécanique des milieux continus. Ainsi, la matière forgée vérifie :

- la conservation de la masse,
- la conservation de la quantité de mouvement.

Ces principes se traduisant mathématiquement par :

- l'équation de continuité : $$\frac{\partial \rho}{\partial t} + \mathrm{div}(\rho v) = 0 \qquad (1)$$
- l'équation de l'équilibre dynamique : $$\rho \frac{dv}{dt} - \mathrm{div}\sigma = 0 \qquad (2)$$

où t désigne le temps, ρ la masse volumique de la matière, v la vitesse d'une particule matérielle et σ le tenseur des contraintes.

Hypothèses principales :

Dans le cadre du forgeage à chaud, il est classique de faire les hypothèses suivantes :

- le matériau est supposé purement viscoplastique : les déformations élastiques sont négligées devant les déformations plastiques,
- le matériau est incompressible,
- les forces de gravité et d'inertie sont négligeables devant les forces visqueuses.

Lois de conservation :

Compte-tenu de ces hypothèses, les équations de conservation de la masse (1) et d'équilibre (2) se simplifient en :

$$\begin{cases} \mathrm{div}\sigma = 0 \\ \mathrm{div}v = 0 \end{cases} \tag{3}$$

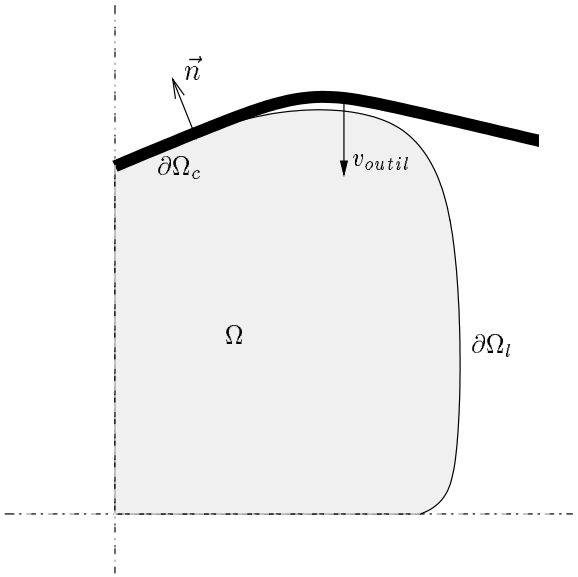


FIG. 1 – *Notations pour le problème de forgeage*

1.2 Lois de comportement

Pour fermer le système (3), il faut le compléter par la relation de comportement entre contraintes et déformations et par des conditions limites. Nous adoptons une description en Lagrangien réactualisé du mouvement et effectuons les hypothèses de petites déformations.

Loi de comportement de Norton-Hoff :

Le tenseur des contraintes σ peut être décomposé en la somme de sa partie sphérique et de sa partie déviatorique, *i.e.* d'une pression hydrostatique:

$$p = -\frac{1}{3}\,\mathrm{Trace}\,(\sigma)$$

et d'un tenseur de trace nulle, le déviateur des contraintes s :

$$\sigma = s - pI \tag{4}$$

Finalement, la loi de comportement traduit la relation entre le déviateur des contraintes s et la vitesse v. Pour des fluides viscoplastiques la loi de Norton-Hoff relie s au tenseur des vitesses de déformation $\dot{\varepsilon}\,(v)$:

$$\dot{\varepsilon}\,(v) = \frac{1}{2}\left(\nabla v + {}^{T}\nabla v\right) \tag{5}$$

Et s'écrit :

$$s = 2K\left(\sqrt{3}\,\dot{\bar{\varepsilon}}\right)^{m-1}\dot{\varepsilon}\,(v) \tag{6}$$

où m est l'indice de sensibilité (de la viscosité) à la vitesse de déformation, K représente la consistance du matériau et $\dot{\bar{\varepsilon}}$ la vitesse de déformation:

$$\dot{\bar{\varepsilon}} \equiv \dot{\bar{\varepsilon}}(v) = \sqrt{\frac{2}{3}\,\dot{\varepsilon}(v) : \dot{\varepsilon}(v)}$$

Par la suite, nous noterons (6) de la manière suivante:

$$s = 2\eta\left(|\dot{\varepsilon}(v)|\right)\dot{\varepsilon}(v) \tag{7}$$

où η représente la viscosité du fluide.

Loi de frottement de Norton :

Nous utilisons une loi de frottement qui exprime l'existence d'un potentiel dissipatif dont dérive le vecteur cission de frottement τ (composante tangentielle du vecteur contrainte à l'interface outil matière).

$$\tau(v) = \sigma\, n - \sigma_n n$$

Où $\sigma_n = \sigma\, n \cdot n$ est la pression de contact.
Les procédés de forgeage étant caractérisés par de très fortes pressions de contact, nous utilisons une loi de frottement dérivant d'un potentiel de Norton. Elle fait intervenir la sensibilité à la vitesse relative de glissement, Δv_g, entre la pièce et l'outil:

$$\Delta v_g = v - v_{outil} - \left[(v - v_{outil}) \cdot n\right] n$$

Où v_{outil} est la vitesse de l'outil en contact avec le matériau et n la normale extérieure (sortante) à la matière (cf. figure 1). Le vecteur cission de frottement à l'interface s'exprime alors selon la loi puissance:

$$\tau(v) = -\alpha_f K_f \,||\Delta v_g||^{p_f - 1}\,\Delta v_g \tag{8}$$

Où α_f est le coefficient de frottement et p_f la sensibilité à la vitesse de glissement.
En pratique, on choisit généralement $p_f = m$ et $K_f = K$. Le frottement est alors uniquement fonction du paramètre α_f qui est déterminé expérimentalement.

Conditions limites :

Contact :

On considère un contact unilatéral qui exprime la non pénétration des noeuds de la pièce dans l'outillage. Celui-ci est décrit par les conditions de *Signorini*:

$$\begin{cases} (v - v_{outil}) \cdot n & \leq & 0 & \textit{non pénétration} \\ \sigma_n & \leq & 0 & \textit{condition mécanique de contact} \\ \left[(v - v_{outil}) \cdot n\right]\sigma_n & = & 0 & \textit{glissement tangentiel} \end{cases} \tag{9}$$

Surface libre :

Sur le bord libre, la pièce n'est soumise à aucune contrainte, ce que l'on exprime par:

$$\sigma \cdot n = 0 \quad \text{sur} \quad \partial\Omega_l \tag{10}$$

Cas particulier : le Comportement Newtonien:

Si l'on considère que le matériau est un fluide newtonien $m = p_f = 1$, le problème à résoudre devient un problème de Stokes simple. Ce problème reste toutefois non-linéaire en raison du contact, sauf si celui-ci est supposé bilatéral.

1.3 Problème fort

Finalement, en notant Ω le domaine matériel et en tenant compte de (4) (7) et des conditions limites de frottement (8), de contact (9) et de surface libre (10), le problème d'écoulement relatif a la mise en forme à chaud (3) est donné par le problème de **Stokes généralisé** suivant:

Trouver $v(x,t) \in I\!R^2 \times (0,T)$ et $p(x,t) \in I\!R \times (0,T)$ solutions du problème:

$$\begin{cases} \operatorname{div}\left(2\eta(|\dot{\varepsilon}(v)|)\dot{\varepsilon}(v)\right) - \nabla p = 0 & \text{dans } \Omega & \text{(11a)} \\ \operatorname{div} v = 0 & \text{dans } \Omega & \text{(11b)} \\ \sigma \cdot n = 0 & \text{sur } \partial\Omega_l & \text{(11c)} \\ (v - v_{outil}) \cdot n \leq 0 & \text{sur } \partial\Omega_C & \text{(11d)} \\ \text{Si } \sigma n \cdot n \leq 0\ , & & \\ \quad \tau(v) = -\alpha(|\Delta v_g|)\,\Delta v_g & \text{sur } \partial\Omega_C & \text{(11e)} \end{cases}$$

Nous effectuons une discrétisation temporelle par un schéma d'Euler explicite. Cela nous mène, à chaque incrément, à la résolution par la méthode des éléments finis d'un problème stationnaire non-linéaire.

2 Formulation faible et discrétisation

Nous nous plaçons dans le domaine matériel noté Ω supposé ouvert borné. On note $L^2(\Omega)$ l'espace de Lebesgue des fonctions de carré sommable, associé au produit scalaire $(.,.)$ et à la norme $\|.\|_0$.

Soit $\mathcal{V}$ un sous-espace de $L^2(\Omega)$. On introduit alors les espaces fonctionnels des vitesses cinématiquement admissibles $\mathcal{V}^{ca}$ (i.e. vérifiant les conditions limites en vitesses) et des vitesses cinématiquement admissibles à zéro $\mathcal{V}_0^{ca}$ (*i.e.* vérifiant les conditions limites homogènes en vitesse), ainsi que l'espace $\mathcal{Q}$ des pressions. On notera que sans les conditions limites sur la pression (11c), l'espace $\mathcal{Q}$ ne serait plus défini qu'à une constante additive près.

$$\begin{aligned} \mathcal{V}^{ca} &= \left\{ v \in \mathcal{V}^2 \mid (v - v_{out}) \cdot n \leq 0 \text{ sur } \partial\Omega_c \right\} \\ \mathcal{V}_0^{ca} &= \left\{ v \in \mathcal{V}^2 \mid v \cdot n \leq 0 \text{ sur } \partial\Omega_c \right\} \qquad (12) \\ \mathcal{Q} &= L^2(\Omega) \end{aligned}$$

Dans le cadre général de la loi de Norton-Hoff (7), l'espace $\mathcal{V}$ est alors l'espace de Sobolev :

$$\mathcal{V} = W^{1,m+1}(\Omega) = \left\{ v \in L^{m+1}(\Omega)\,;\, \frac{\partial v}{\partial x_i} \in L^{m+1}(\Omega)\ \forall i = 1,2 \right\} \qquad (13)$$

où $0 < m \leq 1$ est la sensibilité à la vitesse de déformation.

On remarquera que dans le cas d'une rhéologie newtonienne ($m = 1$) on retrouve l'espace $H^1(\Omega) \equiv W^{1,2}(\Omega)$, des fonctions de $L^2(\Omega)$ à dérivée de carré sommable:

$$\mathcal{V} = H^1(\Omega) = \left\{ v \in L^2(\Omega)\,;\, \frac{\partial v}{\partial x_i} \in L^2(\Omega)\ \forall i = 1,2 \right\} \qquad (14)$$

Que nous munirons de la norme $\|.\|_1$ avec:

$$\|v\|_1 = \|\nabla v\|_0 + \|v\|_0$$

2.1 Formulation faible continue

On obtient alors une formulation faible du problème mixte en vitesse pression (11) en multipliant par des fonctions tests l'équation de comportement (11a) et l'équation d'incompressibilité (11b) et en intégrant selon une formule de Green. Le problème devient :

$$\begin{array}{l} Trouver\ (v,p) \in \mathcal{V}^{ca} \times \mathcal{Q} \quad tel\ que : \\ \left\{ \begin{array}{lll} \displaystyle\int_\Omega 2\eta\left(|\dot{\varepsilon}(v)|\right)\dot{\varepsilon}(v) : \dot{\varepsilon}(v^*)\ dV & \displaystyle - \int_\Omega p\,\mathrm{div} v^*\, dV - \int_{\partial\Omega_c} \tau(v)\cdot v^*\, d\Gamma & = 0 \quad \forall v^* \in \mathcal{V}_0^{ca} \\ & \displaystyle - \int_\Omega p^*\,\mathrm{div} v\, dV & = 0 \quad \forall p^* \in \mathcal{Q} \end{array} \right. \end{array} \tag{15}$$

Le problème peut alors s'énoncer sous la forme d'un point selle ou comme un problème de minimisation sous contraintes.

Supposons Ω soit borné et connexe. Dans le cas continu, la relation suivante est toujours vérifiée.

Condition de stabilité LBB (Ladyzhenskaya Brezzi Babuska) :

Il existe une constante $\xi > 0$ satisfaisant la condition *inf-sup* :

$$\sup_{w\in\mathcal{V}} \frac{(p, \mathrm{div} w)}{\|w\|_1} \geq \xi \|p\|_0 \quad \forall p \in \mathcal{Q} \tag{16}$$

Cette condition est démontrée, par exemple, dans le livre de *Brezzi et Fortin* [13].

Existence et unicité des solutions

Dans le cas linéaire, l'existence et l'unicité des solutions du problème (15) découlent de la propriété (16), et de la $\mathcal{V}$-ellipticité de la forme bilinéaire

$$a(u,v) = \int_\Omega 2\eta\dot{\varepsilon}(u) : \dot{\varepsilon}(v)\ dV$$

et se démontre aisément par le Lemme de Lax-Milgram.

Dans le cas non-linéaire, on prouve l'existence et l'unicité à l'aide de l'analyse convexe. Nous renvoyons à *Baranger et Najib* pour les démonstrations [7].

2.2 Discrétisation

Espaces d'approximation :

Soit $\mathcal{T}_h(\Omega)$ une triangulation du domaine Ω en sous-éléments Ω_e et h une taille de maille caractéristique.

On notera :

$$\mathcal{T}_h(\Omega) = \bigcup_{e\in\mathcal{E}} \Omega_e$$

Un problème éléments finis discret est construit en approchant les espaces de dimension infinie où vivent les solutions par des espaces discrets de dimension finie. Soit $\mathcal{V}_h \subset \mathcal{V}^{ca}$ et $\mathcal{Q}_h \subset \mathcal{Q}$ des sous-espaces de dimension finie. En exprimant (15) dans ces sous-espaces et en effectuant le choix de Galerkin pour les fonctions tests on obtient le problème discret suivant :

Formulation faible discrète :

$$Trouver\ (u_h, p_h) \in \mathcal{V}_h \times \mathcal{Q}_h\ tel\ que\ \forall\ (u^*, p^*) \in \mathcal{V}_h^0 \times \mathcal{Q}_h\ on\ ait :$$

$$\begin{cases} \displaystyle\int_\Omega 2\eta\left(|\dot{\varepsilon}\left(u_h\right)|\right)\dot{\varepsilon}(u_h) : \dot{\varepsilon}(u^*)\, dV \quad - \int_\Omega p_h \,\mathrm{div} u^*\, dV + \int_{\partial\Omega_c} \tau(u_h)\cdot u^*\, d\Gamma = 0 & (17a) \\ \qquad\qquad - \displaystyle\int_\Omega p^* \,\mathrm{div} u_h\, dV = 0 & (17b) \end{cases}$$

Par abus de notation, nous définissons les intégrales sur le domaine Ω, mais ces intégrales sont en fait évaluées sur le domaine discrétisé $\mathcal{T}_h(\Omega)$.
Il est maintenant bien connu que la convergence d'une méthode mixte ne dépend pas seulement du choix des espaces d'approximation $\mathcal{V}_h$ et $\mathcal{Q}_h$, mais aussi de l'expression discrète de la condition de stabilité *inf-sup* (16):

Condition LBB discrète :

Il existe une constante $\gamma > 0$ indépendante de h telle que :

$$\sup_{u_h \in \mathcal{V}_h} \frac{(p, \mathrm{div} u_h)}{\|\nabla u_h\|_{\mathcal{V}}} \geq \gamma \|p\| \quad \forall p \in \mathcal{Q} \tag{18}$$

C'est une condition **suffisante** d'existence et d'unicité de la solution discrète de (17).

Un exemple classique de méthodes instables est celui des méthodes utilisant, sur une même grille, des interpolations de même ordre (par exemple P1P1) pour les vitesses et les pressions. L'espace des pressions est trop riche comparé à celui des vitesses et, on peut alors observer un phénomène de "pression en damier" résultant en des modes de pressions et des vitesses pouvant êtres perturbées. Nous renvoyons à la thèse de *C. Aliaga* [2] pour une présentation plus complète et des exemples dans le cas d'une discrétisation P1P1.

On trouve dans la bibliographie de nombreux éléments finis mixtes satisfaisant la condition *inf-sup*. Nous renvoyons par exemple aux articles de *M. Fortin et A. Fortin* [38, 39] ou au livre de *O. Pironneau* [64] pour une revue des éléments utilisés dans le cadre du problème de Stokes. Pour des problèmes spécifiques à la mise en forme, on pourra trouver une comparaison en 3D des éléments P1+P1, P2P0 et P2P1 dans la thèse de *C. Gay* [45] ainsi que dans les thèse de *L. Gaston* [44] et *M. Miles*.

3 Formulation P1P1 stabilisée

Le principe de la méthode est très simple et permet de conserver des interpolations de faible ordre. Il consiste à perturber (ou à relaxer) l'équation d'incompressibilité (17b). On remplace alors le problème (17) par un problème dit stabilisé, que l'on peut considérer comme une simple perturbation du précédent. L'idée de base consiste à introduire un terme **de second ordre**, pour perturber l'équation d'incompressibilité **sans changer l'ordre de la méthode**. Cela a le double avantage de fournir une formulation éléments finis stable et des systèmes linéaires beaucoup mieux conditionnés. L'inconvénient est que diverses perturbations auront le même ordre de convergence et donc qu'il y aura numériquement un paramètre à ajuster.

On associe aux espaces fonctionnels précédents les sous espaces de dimension finie $\mathcal{V}_h \subset \mathcal{V}^{ca}$ et $\mathcal{Q}_h \subset \mathcal{Q}$ où :

$$\begin{aligned} \mathcal{V}_h &= \{ v_h \in \mathcal{V}^{ca} \cap C^0(\overline{\Omega}_h) \mid v_{h|\Omega_e} \in P^1(\Omega_e) \ \forall e \in \mathcal{E} \} \\ \mathcal{Q}_h &= \left\{ p_h \in L^2(\Omega_h) \cap C^0(\overline{\Omega}_h) \mid p_{h|\Omega_e} \in P^1(\Omega_e) \ \forall e \in \mathcal{E} \right\} \end{aligned} \tag{19}$$

et le problème stabilisé est alors :

$$\begin{array}{l} Trouver\ (u_h, p_h) \in \mathcal{V}_h \times \mathcal{P}_h \quad tel\ que : \\ \left\{ \begin{array}{lllll} (\, 2\eta(\,|\dot{\varepsilon}(u_h)|\,)\,\dot{\varepsilon}(u_h)\,,\,\dot{\varepsilon}(u^*)\,) & - & (\,p_h\,,\,div\,u^*\,) & = & (\,f\,,\,u^*\,) \qquad \forall\, u^* \in \mathcal{V}_h^0 \\ -\,(\,q^*\,,\,div\,u_h\,) & - & \beta^2 C_h(\,q^*\,,\,p_h\,) & = & 0 \qquad \forall\, q^* \in \mathcal{P}_h \end{array} \right. \end{array} \tag{20}$$

Où $C_h(.,.)$ une forme bilinéaire symétrique semi-définie positive vérifiant la condition faible de stabilité :

$$(p, \mathrm{div} v) = 0 \implies C_h(p,p) \neq 0 \tag{21}$$

Et β est le paramètre de stabilisation. Celui-ci devant respecter certaines conditions pour que le problème stabilisé (20) admette une unique solution.

Dans le cadre d'une interpolation linéaire et continue de la pression, il est classique de choisir une formulation globalement stabilisée [59] avec :

$$C_h(\,p\,,\,q\,) = \sum_{K \in \mathcal{T}_h(\Omega)} h_K^2 \int_K \nabla p \nabla q \, d\omega \tag{22}$$

Où h_K est une taille caractéristique de l'élément K, par exemple le diamètre du cercle inscrit. Pour ce type de stabilisation, on peut montrer que tout choix de β résulte en une méthode LBB-stable [72] et on a l'estimation d'erreur :

$$\|u - u_h\|_{\mathcal{V}_h} + \|p - p_h\|_{\mathcal{Q}_h} \leq C_1 h^2 + C_2 h \tag{23}$$

On aboutit alors au système algébrique:

$$\begin{pmatrix} A & B^T \\ B & -\beta^2 C \end{pmatrix} \begin{pmatrix} u_h \\ p_h \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix} \tag{24}$$

La matrice C est symétrique semi-définie positive et donc $-C$ est définie négative. Le système algébrique est donc soluble par une méthode itérative adaptée (MINRES ou SYMMLQ si l'on se réfère à la bibliographie).

Remarques :

Dans le cadre d'une formulation globalement stabilisée le choix de la valeur β est souvent considéré comme un point critique [59]. On considère généralement qu'une valeur optimale est $\beta^* \approx \frac{1}{10}$.

La stabilisation du problème de Stokes revient à autoriser numériquement une légère compressibilité locale.

Une méthode permettant d'éviter d'avoir à ajuster un paramètre numérique consiste à effectuer cette stabilisation par l'apport d'une fonction "bulle".

4 Discrétisation par le MINI-élément P1+P1

Présentation :

L'élément P1+P1 ou encore MINI-élément est l'élément mixte compatible de plus petit degré d'interpolation. Il a été introduit par *Arnold, Brezzi et Fortin* [3] pour des écoulements de Stokes incompressibles. C'est un élément linéaire P1P1 dont on enrichis l'interpolation de la vitesse par l'ajout d'un degré de liberté interne dit "bulle", qui s'annule sur la frontière. Les champs de vitesse et de pression sont les suivant :

- la pression est linéaire et continue,
- la vitesse se décompose en une partie linéaire v et une partie bulle b. Cette décomposition est **unique**.

Cette composante bulle sert essentiellement à contrôler la condition d'incompressibilité. Elle permet à l'élément de vérifier la condition de compatibilité (18) [3] et est interpolée par une fonction bulle valant 1 au centre de l'élément et 0 sur sa frontière.

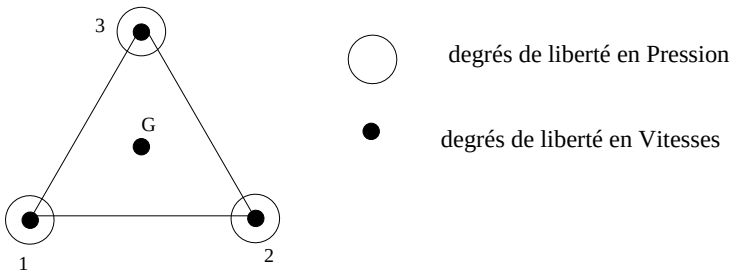


FIG. 2 – *Le MINI-élément P1+P1*

On définit $\mathcal{B}_h$ l'espace d'interpolation des fonctions bulle :

$$\mathcal{B}_h \;=\; \left\{\, b_h \;\in H_0^1(\Omega_h) \cap C^0(\overline{\Omega}_h) \,/\, b_{h|\Omega_e} \;\in\; H_0^1(\Omega_e) \cap P^k(\Omega_e) \;\forall\, e \in \mathcal{E} \,\right\} \tag{25}$$

Et l'on interpole alors la vitesse dans l'espace :

$$\mathcal{X}_h = \mathcal{V}_h \oplus \mathcal{B}_h \tag{26}$$

Dans le travail de *Arnold et al.* [3] ceux-ci ont montré que le choix d'une bulle cubique (k=3 en 2D) menait à une méthode stable avec le même ordre de convergence que la formulation stabilisée (23).

Dans le cadre d'éléments linéaires, le choix d'une bulle de degré élevé est source de problèmes numériques. Elle nécessite une intégration numérique plus précise, fait craindre l'apparition d'instabilités dues à l'association de fonctions linéaires et cubiques et engendre un surcoût de calcul important pour une simple correction de vitesse. Soit $w_h = v_h + b_h$ l'interpolée de la vitesse dans $\mathcal{X}_h$ alors :

$$b_e = w_h(G_e) - \frac{v_1 + v_2 + v_3}{3} \qquad \forall \Omega_e \in \mathcal{T}_h(\Omega)$$

Où G_e est le centre de gravité de l'élément Ω_e et b_e le vecteur bulle.

R. Pierre [62, 63] a généralisé la preuve de *Arnold et al.* en montrant que la condition :

$$\forall\, \Omega_e \in \mathcal{T}_h,\; \phi_e \in H_0^1(\Omega_e),\; \phi_e = 0 \; sur \; \Omega_h \backslash \Omega_e \tag{27}$$

est suffisante pour avoir un élément bulle conforme LBB-stable. Il a de plus montré qu'une approximation non-conforme était possible sous la condition que l'élément vérifie la propriété d'orthogonalité de la fonction bulle aux constantes.

Propriété d'orthogonalité:

$$\forall e \in \mathcal{E} \quad \int_{\Omega_e} \dot{\varepsilon}\left(v_h\right) : \dot{\varepsilon}\left(b_h\right) \, d\Omega = 0 \qquad \forall v_h \in P^1(\Omega_e) \tag{28}$$

Cette généralisation autorise donc d'effectuer un raffinement h plutôt que p. On interpole alors les fonctions bulle par des fonctions linéaires par morceaux vérifiant (27). Celle-ci a été implantée avec succès dans le code de calcul FORGE3® par *T. Coupez* [20, 21].

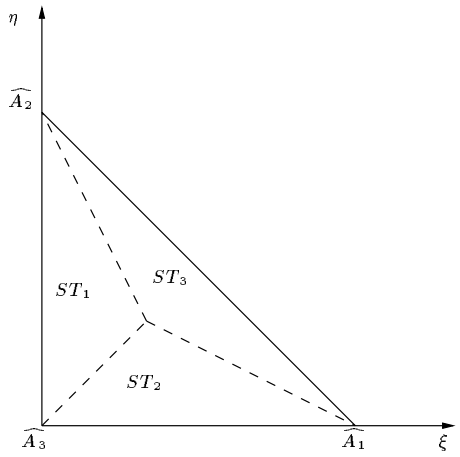


FIG. 3 – *Décomposition en sous-triangles*

La fonction hiérarchique est alors la fonction continue, linéaire par sous-triangles, valant 1 au centre de gravité et 0 sur le bord du triangle. On peut l'écrire sur le triangle de référence:

$$\phi_e(\xi,\eta) = \begin{cases} 3\,\xi & \text{dans } ST_1 \\ 3\,\eta & \text{dans } ST_2 \\ 3\,(1-\xi-\eta) & \text{dans } ST_3 \end{cases} \tag{29}$$

Cela nous permet d'avoir le même ordre d'intégration pour les fonctions bulle que pour les vitesses et les pressions.

4.1 Le MINI-élément en 2D plan et en 3D :

Dans un premier temps nous nous contenterons d'une présentation dans le cas Newtonien, avant d'étudier l'extension de notre formulation au cas non-linéaire.

4.1.1 Cas Newtonien (m=1):

On pose $u_h = v_h + b_h$. Le problème faible discret s'écrit alors :

Trouver $(u_h, p_h) \in (\mathcal{V}_h \oplus \mathcal{B}_h) \times \mathcal{Q}_h$ *solutions de* :

$$\left\{ \begin{array}{lll} \displaystyle\int_\Omega 2K\dot{\varepsilon}\left(u_h\right) : \dot{\varepsilon}\left(u^*\right)\, dV + \int_{\partial\Omega_c} \tau(u_h)\cdot u^*\, d\Gamma & - \displaystyle\int_\Omega p_h \mathrm{div} u^*\, dV & = 0 \quad \forall u^* \in ({\mathcal{V}_h}^0 \oplus \mathcal{B}_h) \\ & - \displaystyle\int_\Omega p^* \mathrm{div} u_h\, dV & = 0 \qquad\qquad \forall p^* \in \mathcal{Q} \end{array} \right. \tag{30}$$

Et, comme $b_h \in H_0^1(\Omega)$ les contributions bulle sont éliminées des termes de bords et on peut développer le système précédent en:

Trouver $(v_h, b_h, p_h) \in \mathcal{V}_h \times \mathcal{B}_h \times \mathcal{Q}_h$ *solutions de* :

$$\left\{ \begin{array}{llll} \displaystyle\int_\Omega 2K\dot{\varepsilon}\left(v_h\right) : \dot{\varepsilon}\left(v^*\right)\, dV & + \displaystyle\int_\Omega 2K\dot{\varepsilon}\left(b_h\right) : \dot{\varepsilon}\left(v^*\right)\, dV & - \displaystyle\int_\Omega p_h \mathrm{div} v^*\, dV & \\ & & + \displaystyle\int_{\partial\Omega_c} \tau(v_h)\cdot v^*\, d\Gamma & = 0 \quad \forall v^* \in \mathcal{V}_h^0 \\ \displaystyle\int_\Omega 2K\dot{\varepsilon}\left(b_h\right) : \dot{\varepsilon}\left(b^*\right)\, dV & + \displaystyle\int_\Omega 2K\dot{\varepsilon}\left(v_h\right) : \dot{\varepsilon}\left(b^*\right)\, dV & - \displaystyle\int_\Omega p_h \mathrm{div} b^*\, dV & = 0 \quad \forall b^* \in \mathcal{B}_h^0 \\ & - \displaystyle\int_\Omega p^* \mathrm{div} v_h\, dV & - \displaystyle\int_\Omega p^* \mathrm{div} b_h\, dV & = 0 \quad \forall p^* \in \mathcal{Q} \end{array} \right. \tag{31}$$

Propriété :
L'élément bulle vérifie la condition d'orthogonalité (28).

preuve :

$$\begin{aligned} \int_\Omega \dot{\varepsilon}\left(v_h\right) : \dot{\varepsilon}\left(b_h\right)\, dV &= \sum_{e\in\mathcal{E}} \int_{\Omega_e} \dot{\varepsilon}\left(v_h\right) : \dot{\varepsilon}\left(b_h\right)\, d\omega \\ &= \sum_{e\in\mathcal{E}} \left(- \int_{\Omega_e} \mathrm{div}\dot{\varepsilon}\left(v_h\right) b_h\, d\omega + \underbrace{\int_{\partial\Omega_e} \dot{\varepsilon}\left(v_h\right)\cdot n\; b_h\, d\Gamma}_{=0 \; car \; b_h \in H_0^1(\Omega_e)} \right) = 0 \\ &\text{Car } v_h \in P^1(\Omega_e),\ \dot{\varepsilon}\left(v_h\right) \in P^0(\Omega_e) \text{ et donc } \mathrm{div}\dot{\varepsilon}\left(v_h\right) = 0 \end{aligned}$$

□

On obtient alors le problème faible discret :

Trouver $(v_h, b_h, p_h) \in \mathcal{V}_h \times \mathcal{B}_h \times \mathcal{Q}_h$ *solutions de* :

$$\left\{ \begin{array}{llll} \displaystyle\int_\Omega 2K\dot{\varepsilon}\left(v_h\right) : \dot{\varepsilon}\left(v^*\right)\, dV & + \displaystyle\int_{\partial\Omega_c} \tau(v_h)\cdot v^*\, d\Gamma & - \displaystyle\int_\Omega p_h \mathrm{div} v^*\, dV & = 0 \quad \forall v^* \in \mathcal{V}_h^0 \\ \displaystyle\int_\Omega 2K\dot{\varepsilon}\left(b_h\right) : \dot{\varepsilon}\left(b^*\right)\, dV & & - \displaystyle\int_\Omega p_h \mathrm{div} b^*\, dV & = 0 \quad \forall\, b^* \in \mathcal{B}_h \\ & - \displaystyle\int_\Omega p^* \mathrm{div} v_h\, dV - & \displaystyle\int_\Omega p^* \mathrm{div} b_h\, dV & = 0 \quad \forall p^* \in \mathcal{Q}_h \end{array} \right. \tag{32}$$

Ce qui amène localement le système algébrique suivant :

$$\begin{pmatrix} H_{vv} & 0 & H_{vp}^T \\ 0 & H_{bb} & H_{bp}^T \\ H_{vp} & H_{bp} & 0 \end{pmatrix} \begin{pmatrix} v_h \\ b_h \\ p_h \end{pmatrix} = \begin{pmatrix} f \\ 0 \\ g \end{pmatrix} \tag{33}$$

Le second membre venant de l'imposition des conditions limites, en vitesse, par une méthode d'élimination.
La bulle étant locale à chaque triangle, *i.e.* $b_h \in H_0^1(\Omega_e)\ \forall e \in \mathcal{V}$, on peut la substituer par le procédé classique de condensation statique:

$$\forall e \in \mathcal{E} \qquad b^e = -(H_{bb}^e)^{-1} H_{bp}^T p^e$$

Et aboutir finalement à:

$$\begin{pmatrix} H_{vv} & H_{vp}^T \\ H_{vp} & -C \end{pmatrix} \begin{pmatrix} v_h \\ p_h \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \tag{34}$$

Avec $C = (H_{bp}) H_{bb}^{-1} H_{bp}^T$ matrice semi-définie positive.

4.1.2 Cas non-linéaire :

L'apparition des termes non-linéaires rend le procédé de condensation de la bulle beaucoup plus lourd à mettre en oeuvre. On fait, par exemple, appel à des techniques de reconstruction de la fonction bulle. Méthodes que l'on peut à nouveau considérer comme excessives pour une simple correction de vitesse.

On effectue alors une approximation, nous permettant de simplifier les calculs, tout en conservant les termes de stabilisation obtenus dans le cas linéaire.
Posons :

$$D = \dot{\varepsilon}(v) \quad \text{et} \quad \eta(|D|) = K\left(\sqrt{3}\,\overline{D}\right)^{m-1} \quad \text{où} \quad \overline{D} = \sqrt{\frac{2}{3} D : D}$$

On peut alors écrire une formulation faible à trois champs du problème viscoplastique continu :

Trouver $(w, p, D) \in \mathcal{V}^{ca} \times \mathcal{Q} \times \mathcal{D}$ tel que :

$$\begin{cases} \displaystyle\int_\Omega 2\eta(|D|)\ \dot{\varepsilon}(w) : \dot{\varepsilon}(v^*)\ d\Omega - \int_\Omega p\, div\, v^*\, d\Omega = 0 & \forall v^* \in \mathcal{V}^0 \\ \qquad\qquad - \displaystyle\int_\Omega p^* div\, w\, d\Omega = 0 & \forall p^* \in \mathcal{Q} \\ \displaystyle\int_\Omega (D - \dot{\varepsilon}(w)) : D^*\, d\Omega = 0 & \forall D^* \in \mathcal{D} \end{cases} \tag{35}$$

Ce problème étant équivalent au problème non-linéaire classique, il admet une unique solution.
On discrétise ce problème comme précédemment :

$$\text{Soit}\ \ w_h \in P^{1+}(\Omega_e), \qquad p_h \in P^1(\Omega_e) \qquad \forall e \in \mathcal{E}$$

On interpole alors D sur l'espace des constantes par éléments par analogie avec une formulation $P1P1$ stabilisée. On pose donc:

$$D_h \in P^0(\Omega_e) \tag{36}$$

On montre alors aisément que la troisième équation de (35) implique:

$$D_h = \dot{\varepsilon}(v_h) \text{ au sens fort}$$

preuve:

$$\begin{aligned}
\forall e \in \mathcal{E} \qquad & \int_{\Omega_e} (D_h - \dot{\varepsilon}(w_h)) : D^* \, d\omega = 0 \qquad \forall D^* \in (P^0(\Omega_e))^4 \\
\Longleftrightarrow \qquad & \int_{\Omega_e} (D_h - \dot{\varepsilon}(v_h)) : D^* \, d\omega = 0 \\
\text{Car:} \qquad & \int_{\Omega_e} \dot{\varepsilon}(b_h) : D^* \, d\omega = 0 \qquad \text{(même preuve que (4.1.1))}
\end{aligned}$$

□

La contribution de la bulle ayant été éliminée du terme non-linéaire, il ne reste plus qu'à raisonner comme dans le cas linéaire.
L'approximation $D_h \in P^0(\Omega_e)$ permet ainsi de construire, et d'amener de façon naturelle, le terme de stabilisation $H_{bp} H_{bb}^{-1} H_{bp}^T$ de la formulation P1P1.

4.2 Cadre axisymétrique

Formulaire :

Soit le vecteur vitesse : $v = \begin{pmatrix} v_r \\ 0 \\ v_z \end{pmatrix}$ et le tenseur des contraintes : $\sigma = \begin{pmatrix} \sigma_{rr} & 0 & \sigma_{rz} \\ 0 & \sigma_{\theta\theta} & 0 \\ \sigma_{rz} & 0 & \sigma_{zz} \end{pmatrix}$

On a alors :

$$\dot{\varepsilon}(v) = \begin{pmatrix} \dfrac{\partial v_r}{\partial r} & 0 & \dfrac{1}{2}\left(\dfrac{\partial v_r}{\partial z} + \dfrac{\partial v_z}{\partial r}\right) \\ 0 & \dfrac{v_r}{r} & 0 \\ \dfrac{1}{2}\left(\dfrac{\partial v_r}{\partial z} + \dfrac{\partial v_z}{\partial r}\right) & 0 & \dfrac{\partial v_z}{\partial z} \end{pmatrix}$$

et :

$$\mathrm{div}\sigma = \begin{pmatrix} \dfrac{\partial \sigma_{rr}}{\partial r} + \dfrac{\partial \sigma_{rz}}{\partial z} + \dfrac{\sigma_{rr} - \sigma_{\theta\theta}}{r} \\ 0 \\ \dfrac{\partial \sigma_{rz}}{\partial r} + \dfrac{\partial \sigma_{zz}}{\partial z} + \dfrac{\sigma_{rz}}{r} \end{pmatrix} \tag{37}$$

La forme particulière du tenseur des contraintes, ainsi que l'expression de la divergence axisymétrique font que la propriété d'orthogonalité (28) doit être remplacée par:

$$\int_{\Omega_e} \dot{\varepsilon}(v) : \dot{\varepsilon}(b) \, d\omega = \int_{\Omega_e} \left[\frac{1}{r}\left(\frac{\partial v_r}{\partial r} - \frac{v_r}{r}\right) b_r + \frac{1}{2r}\left(\frac{\partial v_r}{\partial z} + \frac{\partial v_z}{\partial r}\right) b_z\right] 2\pi \, r \, dr \, dz \tag{38}$$

Le fait que cette propriété ne soit plus vérifiée naturellement ne remet pas en cause l'élément P1+P1 axisymétrique. En effet, cette condition n'est pas nécessaire à l'existence et à la convergence d'une méthode conforme. En revanche cela rend le procédé de condensation

statique plus coûteux et on peut s'interroger sur l'utilité des termes supplémentaires à rajouter par rapport au cas plan.

Formulation faible :

On suppose que l'on peut encore effectuer l'hypothèse que la contribution de la bulle aux non-linéarités est négligeable. Cette hypothèse ne semble pas a priori très forte car une petite perturbation de vitesse sur chaque élément influencera peu la viscosité. La formulation faible discrète est alors :

Trouver $(v_h, b_h, p_h) \in \mathcal{V}_h \times \mathcal{B}_h \times \mathcal{Q}_h$ *solutions de* :

$$\left\{ \begin{array}{lll} \displaystyle\int_\Omega 2\eta\left(|\dot{\varepsilon}\left(v_h\right)|\right)\dot{\varepsilon}(v_h+b_h):\dot{\varepsilon}(v^*)\,dV+ & \displaystyle\int_{\partial\Omega_c}\tau(v_h)\cdot v^*\,d\Gamma-\int_\Omega p_h \mathrm{div}\, v^*\,dV & =0 \quad \forall v^*\in\mathcal{V}_h^0 \\ \displaystyle\int_\Omega 2\eta\left(|\dot{\varepsilon}\left(v_h\right)|\right)\dot{\varepsilon}(v_h+b_h):\dot{\varepsilon}(b^*)\,dV & \displaystyle-\int_\Omega p_h \mathrm{div} b^*\,dV & =0 \quad \forall b^*\in\mathcal{B}_h \\ & \displaystyle-\int_\Omega p^* \mathrm{div}\, v_h\,dV-\int_\Omega p^* \mathrm{div} b_h\,dV & =0 \quad \forall p^*\in\mathcal{Q}_h \end{array} \right. \tag{39}$$

Formulation menant à des systèmes linéaires de la forme :

$$\begin{pmatrix} H_{vv} & H_{vb}^T & H_{vp}^T \\ H_{vb} & H_{bb} & H_{bp}^T \\ H_{vp} & H_{bp} & 0 \end{pmatrix} \begin{pmatrix} v_h \\ b_h \\ p_h \end{pmatrix} = \begin{pmatrix} f \\ 0 \\ g \end{pmatrix} \tag{40}$$

Aucune condition limite n'étant appliquée aux termes bulle puisque ceux-ci s'annulent toujours sur la frontière
La localité des fonctions bulle à chaque triangle permet encore d'appliquer le procédé de condensation statique et aboutit finalement à :

$$\begin{pmatrix} H_{vv}-A & H_{vp}^T-B^T \\ H_{vp}-B & -C \end{pmatrix} \begin{pmatrix} v_h \\ p_h \end{pmatrix} = \begin{pmatrix} f' \\ g' \end{pmatrix} \tag{41}$$

Avec : $C = H_{bp}H_{bb}^{-1}H_{bp}^T$ matrice semi-définie positive

$$A = H_{vb}^T H_{bb}^{-1} H_{vb} \quad \text{et} \quad B = H_{vb}H_{bb}^{-1}H_{bp}^T$$

Nous nous référerons par la suite à cette formulation par le nom de **formulation complète**.

4.3 Expressions discrètes

Nous nous contenterons ici d'expliciter les termes issus des discrétisations des fonctions bulle qui représentent notre contribution et renvoyons par exemple à l'article de *L. Fourment* dans les *Séminaires de Plasticité* [41] pour la dérivation et l'expression discrète des termes de comportement, et à [42] pour les termes de frottement et le traitement du contact.

Soit ξ, η les coordonnées barycentriques. On notera : $X^k = (r_k, z_k)$ et $V^k = \left(V_r^k, V_z^k\right)$ les vecteurs coordonnées et vitesses du noeud k. Soit $Nbnoe$ le nombre de noeuds et $Nbelt$ le nombre d'éléments associés à la triangulation du domaine. On pose :

$$v_h = \sum_{k=1}^{Nbnoe} N_k\, V_k \qquad \text{et} \qquad p_h = \sum_{k=1}^{Nbnoe} N_k\, P_k \tag{42}$$

N_k est la fonction de base linéaire associée au noeud k, engendrée par $(\xi, \eta, 1-\xi-\eta)$

$$b_h = \sum_{e=1}^{Nbelt} \phi_e \, b_e \quad \text{avec} \quad b_e = \begin{pmatrix} b_e^r \\ b_e^z \end{pmatrix} \tag{43}$$

Où ϕ_e est la fonction hiérarchique définie en (29) page 11. Nous écrirons donc la vitesse d'un point $M(r,z)$ de l'élément Ω_e de la manière suivante :

$$v_M(r,z) \, = \, \sum_{k=1}^{3} N_k(r,z) \, V_k \, + \, \phi_e(r,z) \, b_e$$

Résolution non-linéaire:

Sous l'hypothèse (36), la résolution du problème non-linéaire (39) entraîne la résolution à chaque incrément de calcul du système d'équations :

$$R\,(v,b,p) = F \Longleftrightarrow \begin{cases} R_V(v,b,p) & = f & 2 \times \text{Nbnoe équations} \\ R_B(v,b,p) & = 0 & 2 \times \text{Nbelt équations} \\ R_P(v,b) & = g & \text{Nbnoe équations} \end{cases} \tag{44}$$

Qui, en tenant compte des conditions aux limites sur le champs vitesse, s'écrit sous forme matricielle:

$$\begin{pmatrix} \widetilde{H}_{vv}(v) & H_{vb}^T & H_{vp}^T \\ H_{vb} & H_{bb} & H_{bp}^T \\ H_{vp} & H_{bp} & 0 \end{pmatrix} \begin{pmatrix} v \\ b \\ p \end{pmatrix} = \begin{pmatrix} f \\ 0 \\ g \end{pmatrix} \tag{45}$$

Les non-linéarités sont uniquement issues des termes en vitesse/vitesse. En effet l'hypothèse (36) conjuguée à la propriété de localité des fonctions bulle à leurs éléments respectifs (27) fait disparaître les non-linéarités de tous les termes associés aux fonctions bulle. La mise en oeuvre de la méthode de Newton-Raphson consiste alors, à partir d'une solution initiale (v,b,p), à rechercher les corrections $(\Delta v, \Delta b, \Delta p)$ solutions de:

$$R\,(v+\Delta v,\; b+\Delta b,\; p+\Delta p) = 0$$

La linéarisation de cette équation menant finalement à:

$$\begin{pmatrix} \dfrac{\partial R_V}{\partial V} & H_{vb}^T & H_{vp}^T \\ H_{vb} & H_{bb} & H_{bp}^T \\ H_{vp} & H_{bp} & 0 \end{pmatrix} \begin{pmatrix} \Delta v \\ \Delta b \\ \Delta p \end{pmatrix} = F - \begin{pmatrix} R_V(v,b,p) \\ R_B(v,b,p) \\ 0 \end{pmatrix} \tag{46}$$

On applique alors la technique de condensation statique de la bulle et des conditions limites pour aboutir à l'une des formulations présentées précédemment.

Ainsi nous obtenons finalement:

$$\begin{aligned} f' &= f - \left(\widetilde{H}_{vv}(v) - H_{vb}^T H_{bb}^{-1} H_{vb}\right) v - \left(H_{vp} - H_{vb}^T H_{bb}^{-1} H_{bp}\right) p \\ &= f - \left(\widetilde{H}_{vv}(v) - A\right) v - \left(H_{vp}^T - B^T\right) p \\ g' &= g - \left(H_{vp} - H_{bp} H_{bb}^{-1} H_{vb}^T\right) v + H_{bp}^T H_{bb}^{-1} H_{bp} p \\ &= g - \left(H_{vp} - B\right) v + Cp \end{aligned}$$

Calcul des contributions des termes bulle/pression :

Le calcul et l'assemblage de ces termes est aisé. Il suffit de remarquer que:

Propriété:

Soit $G(r_e, z_e)$ le centre de gravité de l'élément Ω_e, on a:

$$\int_\Omega b\,\nabla p\,dV = 2\pi \sum_{e\in\mathcal{E}} r_e b_e\,\nabla p^e \frac{|\Omega_e|^{plan}}{3} \tag{47}$$

Où b_e est la valeur bulle sur l'élément Ω_e et ∇p^e le gradient de pression à l'intérieur de l'élément. ∇p^e est une constante puisque la pression est linéaire par éléments.

preuve:

$$\begin{aligned}\int_\Omega b\,\nabla p\,dV &= \sum_{e\in\mathcal{E}} \int_{\Omega_e} b\,\nabla p\; 2\pi\; r\,dr\,dz \\ &= \sum_{e\in\mathcal{E}} \left[\sum_{ST\in\mathcal{E}} \int_{ST} \phi_e\; 2\pi\; r\,dr\,dz\right] b_e \nabla p^e\end{aligned}$$

Car ∇p^e est une constante par éléments.

On utilise alors une formule d'intégration au milieu des cotés des sous-triangles, licite puisque $r\phi_e$ est de degré deux.

Comme par définition du centre de gravité $\forall i = 1,3\ |ST_i|^{plan} = \dfrac{|\Omega_e|^{plan}}{3}$

On tire finalement:

$$\int_\Omega b\,\nabla p\,dV = 2\pi \sum_{e\in\mathcal{E}} b_e\,\nabla p^e \frac{|\Omega_e|^{plan}}{9} \left(\frac{3r_e + r_1 + r_2 + r_3}{2}\right)$$

□

Calcul des contributions des termes bulle/bulle:

Soit b^* les fonctions tests du champs bulle:

$$b^* = \begin{pmatrix} \phi_e \\ 0 \end{pmatrix} \quad \text{ou bien} \quad b^* = \begin{pmatrix} 0 \\ \phi_e \end{pmatrix}$$

La matrice peut s'écrire comme une somme sur chaque sous-triangle de matrices locales 2×2:

$$H^e_{bb} = \int_{\Omega_e} 2\eta(v)\dot\varepsilon(b_e\phi_e) : \dot\varepsilon(b^*)\; 2\pi\; r\,dr\,dz = 2\pi \sum_{i=1}^{3} \int_{ST_i} 2\eta(v)\,\dot\varepsilon(b_e\phi_e) : \dot\varepsilon(b^*)\,r\,dr\,dz \tag{48}$$

Que l'on décompose de la manière suivante:

$$H^e_{bb} = H^{e\theta}_{bb} + H^{erz}_{bb} \tag{49}$$

Avec :

$$\begin{aligned} H^{erz}_{bb} &= 2\pi \sum_{i=1}^{3} \begin{pmatrix} \displaystyle\int_{ST_i} \frac{\partial\phi_e}{\partial r}\frac{\partial\phi_e}{\partial r} + \frac{1}{2}\frac{\partial\phi_e}{\partial z}\frac{\partial\phi_e}{\partial z}\,r\,dr\,dz & \displaystyle\int_{ST_i} \frac{1}{2}\frac{\partial\phi_e}{\partial r}\frac{\partial\phi_e}{\partial z}\,r\,dr\,dz \\ \displaystyle\int_{ST_i} \frac{1}{2}\frac{\partial\phi_e}{\partial z}\frac{\partial\phi_e}{\partial r}\,r\,dr\,dz & \displaystyle\int_{ST_i} \frac{\partial\phi_e}{\partial z}\frac{\partial\phi_e}{\partial z} + \frac{1}{2}\frac{\partial\phi_e}{\partial r}\frac{\partial\phi_e}{\partial r}\,r\,dr\,dz \end{pmatrix} \\ H^{e\theta}_{bb} &= 2\pi \sum_{i=1}^{3} \begin{pmatrix} \displaystyle\int_{ST_i} \frac{\phi_e}{r}\frac{\phi_e}{r}\,r\,dr\,dz & 0 \\ 0 & 0 \end{pmatrix} \end{aligned}$$

On peut calculer analytiquement les termes de la matrice H_{bb}^{erz}. En effet comme ϕ_e est linéaire par sous-triangles ses dérivées sont constantes et peuvent être aisément déduites. Le calcul de $H_{bb}^{e\theta}$ est beaucoup moins évident et l'on préfère utiliser un schéma d'intégration à trois points de Gauss pour calculer ces contributions. Ce qui par rapport au cas plan ou 3D, où l'on obtenait uniquement des termes linéaires ou constants, augmente encore le coût de l'assemblage. C'est aussi le cas pour les termes croisés en bulle/vitesse.

Calcul des contributions des termes bulle/vitesse:

Une solution élégante pour assembler cette matrice consiste à utiliser la même procédure que pour l'assemblage de la matrice en vitesse/vitesse. En effet ces termes sont exactement de la même forme *i.e.* intégrale du produit contracté de tenseurs de vitesse de déformation. Il faut, dans ce cas, appeler trois fois cette procédure (une fois par sous-triangle). Si l'on considère la décomposition en sous-triangles présentée sur la figure 3 page 11 avec le noeud central toujours numéroté en dernier.
Soit N_i la fonction de base associée au noeud i, N_4 étant la fonction bulle hiérarchique associé au noeud central et $N_4^* = (N_4, 0)$ ou $(0, N_4)$ la fonction bulle test. On impose que les indices $(i, j, k) = \sigma\,(1, 2, 3)$ soient toujours obtenus par des permutations circulaires positives sur les entiers 1, 2 et 3. Les contributions croisées s'écrivent:

$$H_{vb}^e = \int_{\Omega_e} 2K\dot{\varepsilon}\,(v) : \dot{\varepsilon}\,(b^*)\; d\omega = 2\pi \sum_{i=1}^{3} \int_{ST_i} \dot{\varepsilon}\,(v_j N_j + v_k N_k + v_4 N_4) : \dot{\varepsilon}\,(N_4^*)\; r\,dr\,dz$$

Et en considérant toujours les permutations circulaires des indices, notons:

$$\widetilde{H}_j^i = \int_{ST_i} \dot{\varepsilon}\,(N_j) : \dot{\varepsilon}\,(N_4^*)\; r\,dr\,dz$$

Comme $v_4 = \dfrac{v_1 + v_2 + v_3}{3}$, la matrice s'obtient alors en assemblant:

$$H_{vb}^e = 2\pi \sum_{i=1}^{3} \left[\left(\widetilde{H}_j^i + \frac{1}{3}\widetilde{H}_4^i \right) v_j + \left(\widetilde{H}_k^i + \frac{1}{3}\widetilde{H}_4^i \right) v_k + \frac{1}{3}\widetilde{H}_4^i v_i \right]$$

Tandis que la matrice H_{bb} est obtenue en assemblant sur chaque sous-triangles les contributions du noeud internes sur lui-même *i.e.* :

$$H_{bb}^e = \int_{\Omega_e} 2K\dot{\varepsilon}\,(b) : \dot{\varepsilon}\,(b^*)\; d\omega = 2\pi \sum_{i=1}^{3} \left[\widetilde{H}_4^i \right] b_e$$

Il apparaît donc qu'effectuer ces calculs rend les termes bulle trois fois plus chers à assembler que les termes standards en vitesse. Le coût total de l'assemblage étant alors multiplié par un facteur quatre, pour une simple correction de vitesse.

4.4 Formulations simplifiées

Comme nous venons de le voir, l'assemblage de la formulation complète (41) représente un surcoût très important. On peut s'interroger sur notre conception de la formulation P1+P1 :

Est-ce plus qu'une formulation stabilisée?

Dans ce cas on peut soit envisager d'assembler les termes supplémentaires, soit tenter de construire un élément bulle spécifique à l'axisymétrique. Une piste que nous avons

longtemps suivie consistait à envisager de différentes manières les interpolations des vitesses et des fonctions bulle. Par exemple, en utilisant une interpolation des vitesses en (r^2, z) qui mène à un champ de vitesse à divergence constante par éléments. Dans tous les cas envisagés (*i.e.* interpolation axi (r^2, z) ou plane (r, z) en vitesse, interpolation axi ou plane voire "exotique" pour les fonctions bulle) nous n'avons jamais pu retrouver la propriété d'orthogonalité. Nous ne sommes même pas parvenus à trouver une formulation qui tende vers la formulation plane quand $r \to +\infty$. Alors qu'avec chaque formulation nous étions capables, pour les vitesses, de trouver des formulations intégrales équivalentes entre plan et axi, nous ne sommes pas parvenus à faire de même pour les fonctions bulle. De ce point de vue, la formulation axisymétrique est particulièrement spécifique.

Ou bien est-ce juste une technique élégante permettant d'obtenir la matrice de stabilisation ? Dans cette perspective, les approximations déjà effectuées dans le cadre non-linéaire nous incitent à ne considérer que les contributions de la bulle amenant ce terme et à négliger les autres.

A partir de cette dernière idée, nous avons en plus de la formulation complète, développé deux autres formulations :

Formulation simplifiée :

On néglige les termes croisés en *bulle/vitesse*, *i.e.* termes apparus dans (38), ce qui revient à négliger les termes contenus dans les matrices A et B de la formulation complète. On aboutit alors au système linéaire :

$$\begin{pmatrix} H_{vv} & H_{vp}^T \\ H_{vp} & -C \end{pmatrix} \begin{pmatrix} v_h \\ p_h \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \tag{50}$$

Formulation simplifiée stabilisée:

Les fonctions bulle sont utilisées pour construire la matrice $-C$. Comme la formulation obtenue est en fait équivalente à une formulation stabilisée, on peut donc envisager d'introduire un paramètre de stabilisation β^2, que l'on ajustera de manière à contrôler la précision et l'efficacité de la formulation. Des idées similaires ont été développées par *A. Wathen et al.* [83] dans le cadre de formulations P1P1 globalement stabilisée. Ceux-ci ont recherché la valeur optimale du paramètre β^2 minimisant le nombre d'itérations de leur méthode itérative de résolution sans détériorer la qualité de la solution.

$$\begin{pmatrix} H_{vv} & H_{vp}^T \\ H_{vp} & -\beta^2 C \end{pmatrix} \begin{pmatrix} v_h \\ p_h \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \tag{51}$$

5 Comparaisons et validation des formulations

Quatre formulations seront comparées ici :

P1P1 stabilisé (24)
P1+P1 complet (41)
P1+P1 simplifié (50)
P1+P1 simplifié stabilisé (51)

L'objectif principal de cette étude est de trouver la formulation éléments finis fournissant les meilleurs résultats en terme de précision, mais aussi de temps de calcul et de qualité des résultats. Par ailleurs, une solution globalement plus précise mais présentant localement des oscillations n'est pas souhaitée.

Le premier point est de prouver numériquement qu'effectuer l'hypothèse d'orthogonalité était encore licite en axisymétrique. Dans l'affirmative, le second point est de choisir entre une formulation simplifiée stabilisée et une formulation simplifiée non stabilisée. Deux raisons nous poussent à nous intéresser fortement à la formulation simplifiée stabilisée.

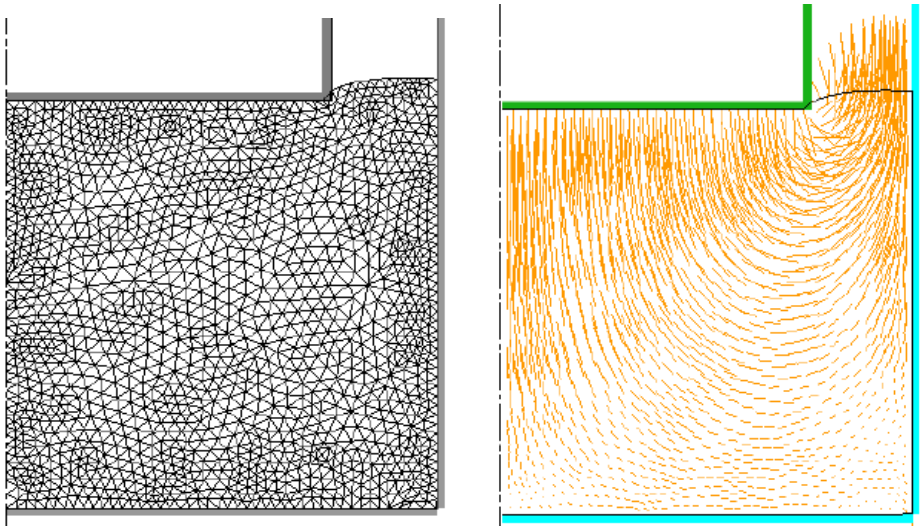
1. Qualitativement, les isovaleurs semblent meilleures ou du moins plus lisses. Cela est principalement le cas pour les isovaleurs de pression ou encore des contraintes. Les vitesses et vitesses de déformations sont par contre de qualité équivalente pour toutes les formulations excepté les formulations stabilisées pour lesquelles on note des différences.
2. Le temps de calcul est significativement plus faible. Cela se comprend en considérant que la relaxation de la condition d'incompressibilité rend le problème moins raide à résoudre sans modifier significativement la qualité de la solution obtenue.

Comparaison qualitative :

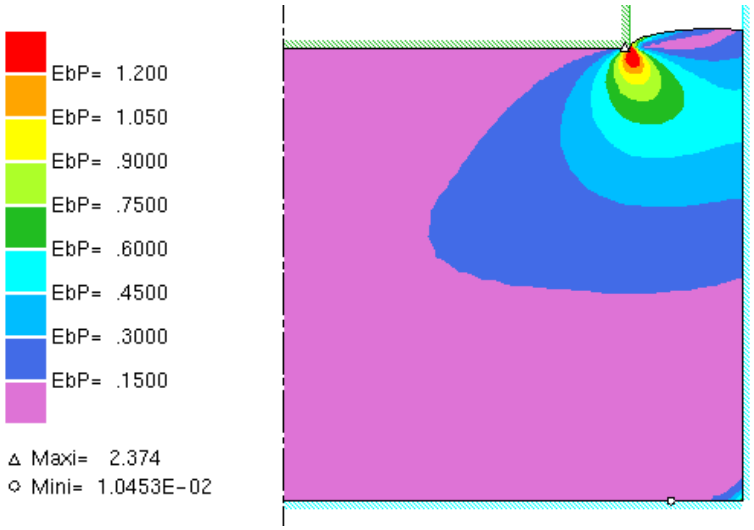
Nous allons effectuer une comparaison des solutions obtenues pour un cas de filage arrière sur un cylindre de 10 mm de rayon et de 10 mm de hauteur. Le rapport de filage *i.e.* le rapport de la longueur de la pièce sur la longueur de la filière est de 4. Le maillage utilisé est un maillage de 1581 noeuds. Une vitesse constante de $1mm.s^{-1}$ est imposée à l'outillage. Le coefficient de frottement est $\alpha = 0.3$. Nous considérons une consistance $K = 8MPa.s^{-1}$ (donc un matériau peu visqueux). La figure 4 illustre le problème et nous présentons les champs de vitesse et de vitesse de déformation obtenus pour la formulation simplifiée dans le cas newtonien. Sur le premier exemple (figure 5), la rhéologie considérée est newtonienne ($m = p = 1$) et nous présentons les différences en pression hydrostatique selon la formulation utilisée. Sur le second (figure 6) la rhéologie est viscoplastique $m = p = 0.1$.

On voit sur ces figures que les solutions issues des formulations "bulle" sont d'une qualité équivalente, les formulations simplifiées stabilisées opérant un lissage des solutions. On remarque de plus qu'il n'existe quasiment aucune différence entre les résultats issus de la formulation complète et ceux issus de la formulation simplifiée.

La formulation stabilisée avec $\beta^2 = 0.01$ fournit une solution nettement différente des autres formulations. Le choix $\beta^2 = 0.1$ détériore encore plus la solution, tandis que des choix de $\beta^2 < 0.01$ l'améliorent mais compromettent en général la convergence de la résolution non-linéaire.

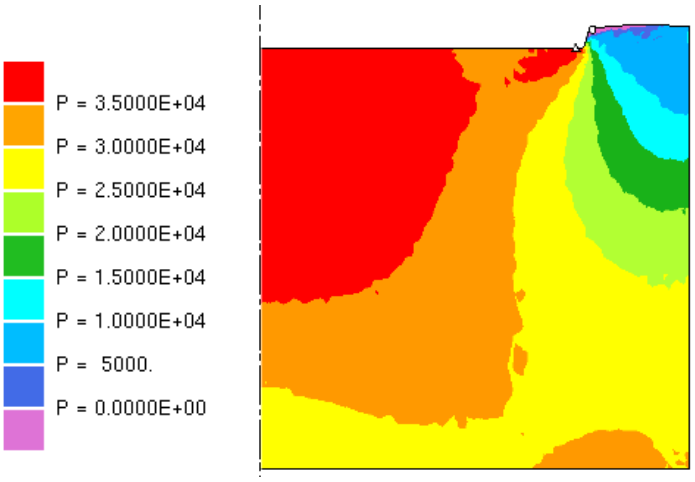


(a) maillage (b) champs de vitesse

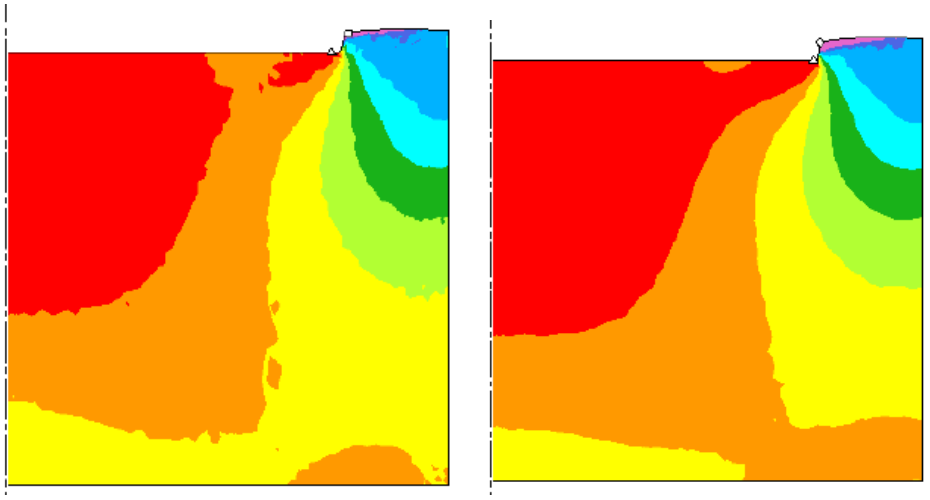


(c) vitesse de déformation

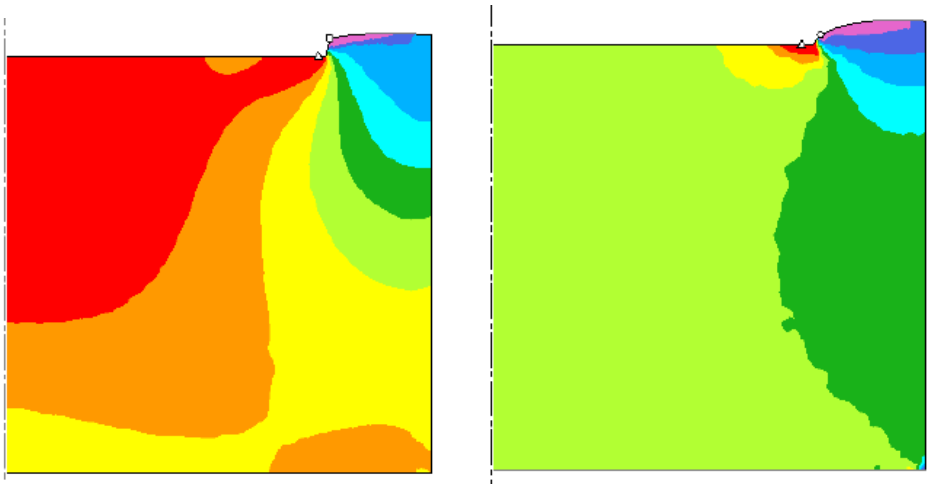
FIG. 4 – *Cas de filage arrière. Maillage de 1581 noeuds, champs de vitesse et de vitesse de déformation*



(a) échelle (b) Formulation simplifiée $\beta = 1$

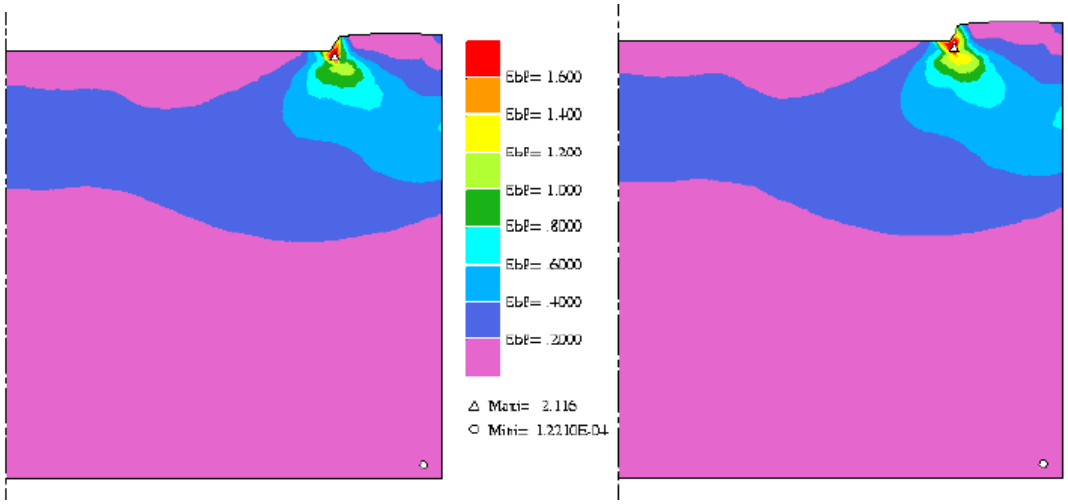


(c) Formulation complète (d) Formulation simplifiée $\beta = 3$



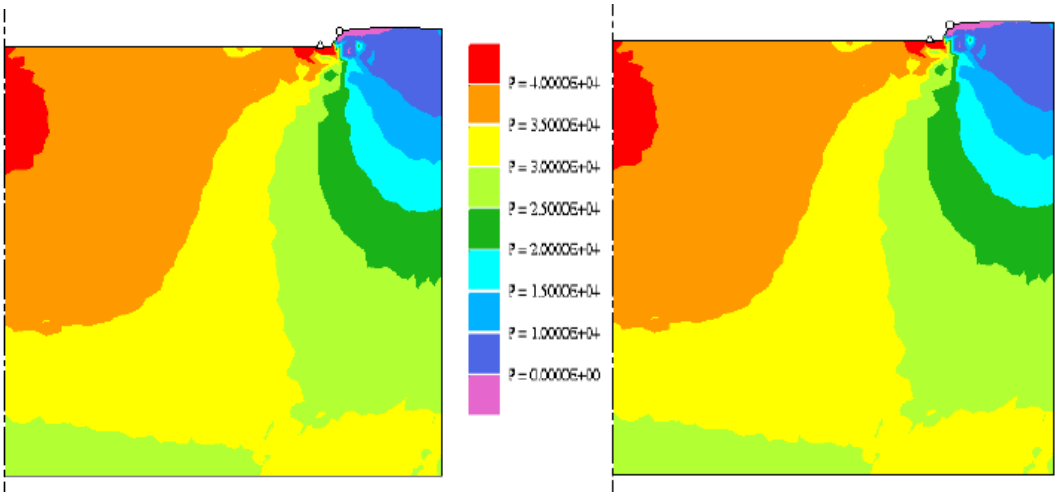
(e) Formulation simplifiée $\beta = 5$ (f) Formulation stabilisée $\beta^2 = 0.01$

FIG. 5 – *Isovaleurs de pression hydrostatique en kPa pour un comportement newtonien*



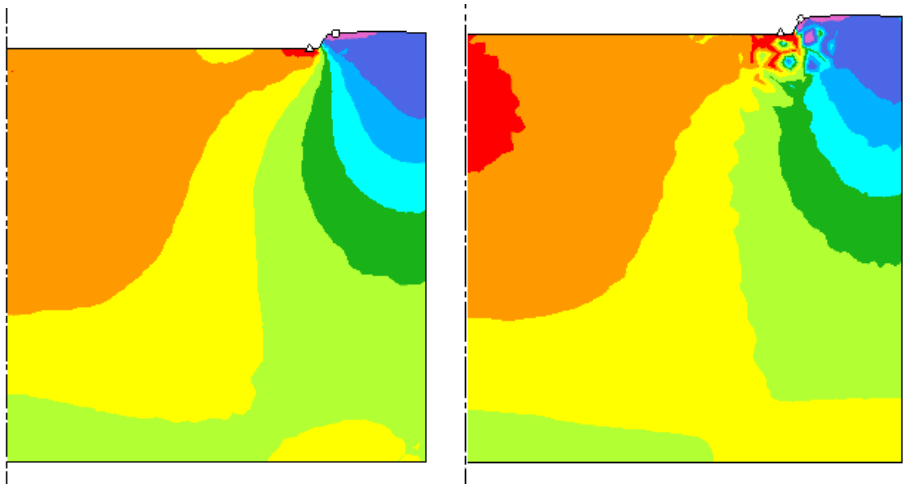
(a) Formulation simplifiée $\beta = 3$ isovaleurs de $\dot{\bar{\varepsilon}}$

(b) Formulation complète isovaleurs de $\dot{\bar{\varepsilon}}$



(c) Formulation simplifiée $\beta = 1$

(d) Formulation complète



(e) Formulation simplifiée $\beta = 3$

(f) Formulation stabilisée $\beta^2 = 0.01$

FIG. 6 – *Isovaleurs de pression hydrostatique en kPa et de vitesse de déformation pour un comportement viscoplastique* $m = p = 0.1$

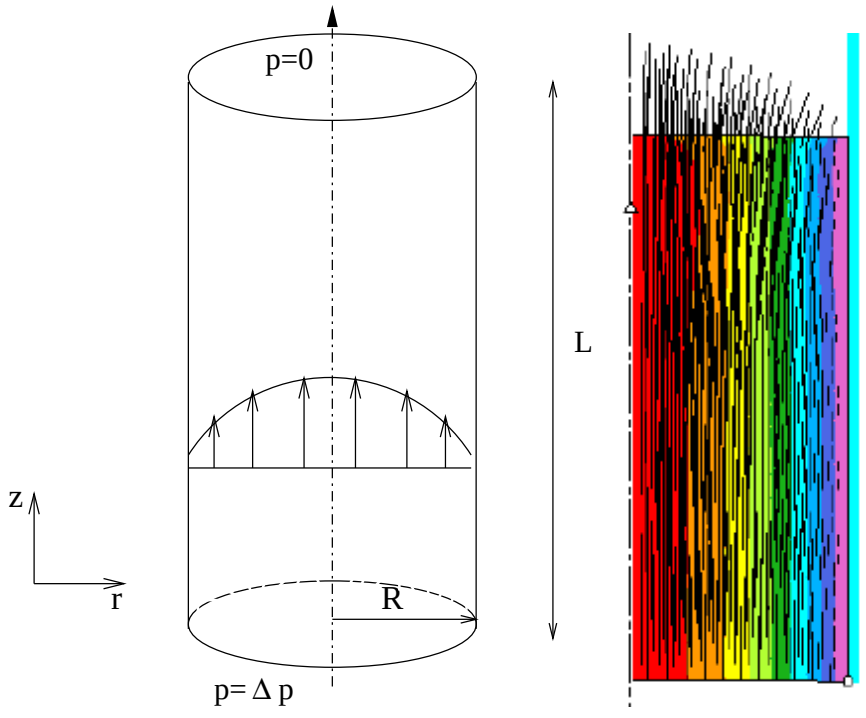
5.1 L'écoulement de Poiseuille tube : comparaison à des solutions analytiques

On trouvera une description très complète de cet écoulement dans le livre de *J. F. Agassant et al.* [1]. L'écoulement de Poiseuille est celui d'un fluide pseudo-plastique dans un tube de rayon R et de longueur L, aux extrémités duquel on impose une perte de charge ΔP (cf. figure 7).
On effectue l'hypothèse que le champ de vitesse peut s'écrire sous la forme :

$$v = \begin{pmatrix} 0 \\ 0 \\ w(r) \end{pmatrix}$$

Cette hypothèse n'est valable qu'en retrait du front de matière. En effet, elle n'autorise pas le remplissage du tube près des parois. Au voisinage du front de matière on est confronté à l'effet fontaine qui implique que le vecteur vitesse possède une composante selon r non nulle. En retrait du front on a, les solutions analytiques suivantes pour un fluide pseudo-plastique d'indice m:

$$\begin{aligned} \text{Débit:} \quad & Q = \pi \frac{m}{3m+1} \left[\frac{1}{2K}\frac{\Delta p}{L}\right]^{\frac{1}{m}} R^{\frac{1+3m}{m}} & \overline{V} &= \frac{Q}{\pi R^2} \\ & w(r) = \frac{3m+1}{m+1}\left[1 - \left(\frac{r}{R}\right)^{\frac{1+m}{m}}\right]\overline{V} & Pression(z) &= \Delta P - \frac{\Delta P}{L} z \end{aligned} \tag{52}$$



(a) (b)

FIG. 7 – *Écoulement de Poiseuille dans un tube. Champs de vitesse avec effet fontaine*

On estime la taille de maille moyenne h à l'aide de la formule:

$$h = \sqrt{\frac{Aire}{Nbelt}}$$

On effectue plusieurs simulations pour des maillages allant de 33 noeuds ($h = 3,61.10^{-2}$) à 21645 noeuds ($h = 1,21.10^{-3}$). Pour chaque maillage on effectue un seul incrément et on calcule les normes d'erreurs relatives pour les champs de vitesse et de pression obtenus en effectuant les intégrations numériques:

$$\begin{aligned} ||q_h - \bar{q}||_0 &= \int_\Omega (q_h - \bar{q})^2 \, dV = \sum_{e \in \mathcal{E}} \int_{\Omega_e} (q_h - \bar{q})^2 \, d\omega \\ &= \sum_{e \in \mathcal{E}} \left(\sum_{int} w(int) (q_h(int) - \bar{q}(int))^2 \right) \end{aligned} \tag{53}$$

Où $\bar{q}$ est la solution analytique. Celle-ci est un polynôme de degré 1 en pression et de degré $\frac{1+m}{m}$ en vitesse. Dans le cas d'un fluide newtonien nous pourrons donc intégrer exactement la solution analytique (polynôme de degré 2) tandis que dans le cas d'un fluide viscoplastique l'intégration numérique sera inexacte. En effet, dans le cas d'un métal à chaud $m \equiv 0.1$ il faudrait être capable d'intégrer des polynômes de degré 22 ! Le schéma d'intégration utilisé est un schéma à sept points permettant d'intégrer exactement des polynômes de degré 5 ([22] p. 781) ce qui implique que dans le cas viscoplastique $m \neq 1$ la solution analytique n'est jamais intégrée exactement.

Les tableaux 1 et 2 donnent les taux de convergence mesurés en vitesse et en pression pour des écoulements newtonien et viscoplastique ($m = 0.1$). Les pentes ont été obtenues par des régressions linéaires effectuées sur des points considérés comme significatifs (les solutions grossières n'ont généralement pas été utilisées).
On estime donc C et p tels que: $||q_h - \bar{q}|| = Ch^p$. L'évolution des erreurs relatives en fonction de la taille de maille est présentée à la figure 9. Les résultats complets figurent en Annexe 1.

		Complète	Simplifiée $\beta = 1$	Simplifiée $\beta = 3$	Simplifiée $\beta = 5$	Stabilisée $\beta^2 = 0.1$	Stabilisée $\beta^2 = 0.01$
Newtonien	p	1.00	1.01	1.01	1.05	1.01	1.00
	C	1.85	1.86	2.02	2.64	1.93	1.87
Viscoplastique	p	1.04	1.04	1.03	1.03	1.05	1.01
	C	34.7	34.7	32.8	32.8	38.1	30.41

TAB. 1 – *Écoulement de Poiseuille dans un tube. Taux de convergence en vitesse. On estime C et p tels que* $\frac{||v_h - \bar{v}||_1}{||\bar{v}||} = Ch^p$

On peut tirer plusieurs observations de ces résultats:

1. Les résultats obtenus sont en accord avec la théorie et, nous retrouvons l'estimation à priori:
$$\|v - v_h\|_{1,\Omega} + \|p - p_h\|_{0,\Omega} \sim \mathcal{O}(h)$$
Dans le cas linéaire, on note une super convergence en pression en raison de cet l'écoulement particulier (pression linéaire).
2. Les écarts sont plus sensibles en pression qu'en vitesse et la viscoplasticité aplatit toutes les différences en vitesse (cf. figure 9(c)).

		Complète	Simplifiée	Simplifiée $\beta = 3$	Simplifiée $\beta = 5$	Stabilisée $\beta^2 = 0.1$	Stabilisée $\beta^2 = 0.01$
Newtonien	p	1.40	1.44	1.52	1.48	1.36	1.34
	C	2.9	4.0	7.7	10.5	2.5	2.8
Viscoplastique	p	1.06	1.06	0.93	0.84	1.15	0.76
	C	8.46	8.38	3.38	1.76	16.1	1.65

TAB. 2 – *Écoulement de Poiseuille dans un tube. Taux de convergence en pression. On estime* C *et* p *tels que* $\frac{||p_h - \bar{p}||_0}{||\bar{p}||} = Ch^p$

3. Les formulations $P1 + P1$ complètes et simplifiées ($\beta = 1$) fournissent des résultats et un taux de convergence quasi-identique.
4. Maintenant si l'on regarde la précision relative de ces formulations, on note de fortes disparités entre elles. Ainsi la formulation simplifiée stabilisée avec $\beta = 5$ accuse un écart significatif par rapport aux autres formulations (*cf.* figures 9(a),9(b)). Cette tendance s'inversant sur l'erreur en pression dans le cas d'un écoulement viscoplastique (*cf.* figure 8).

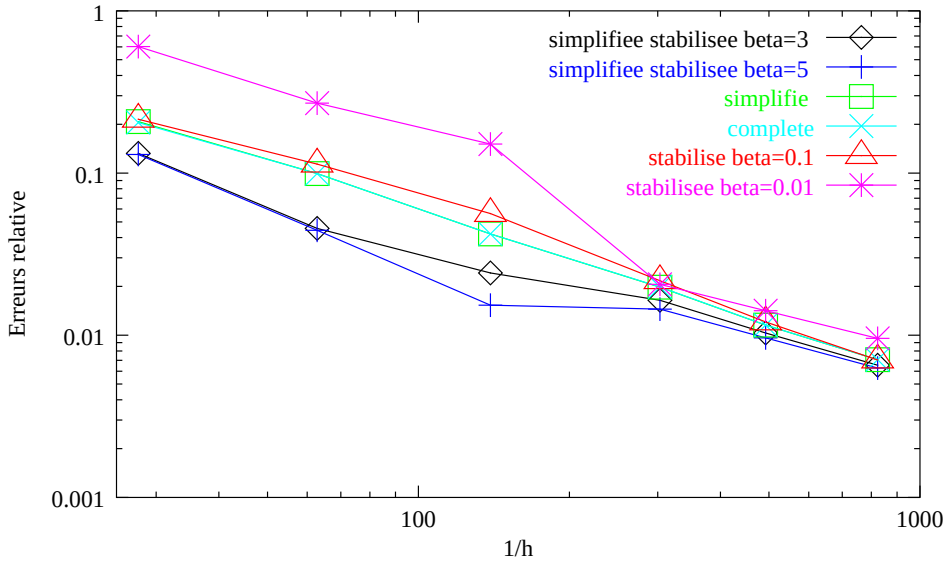
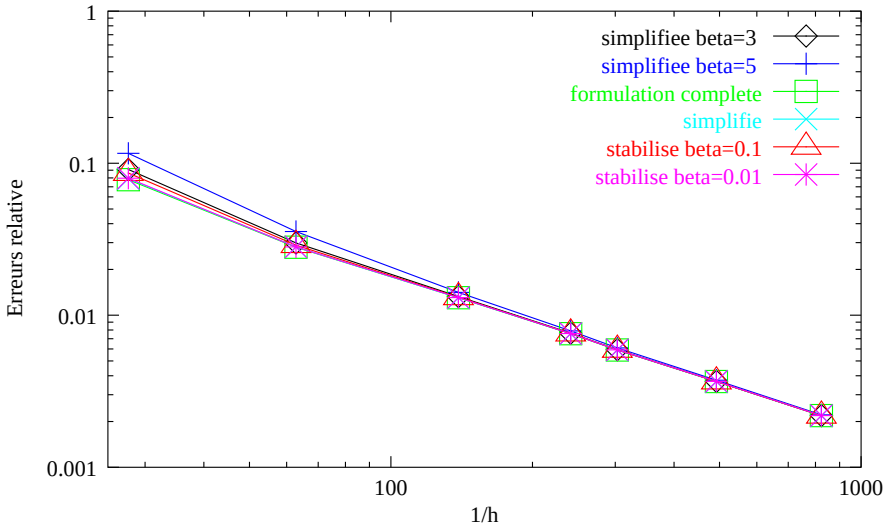
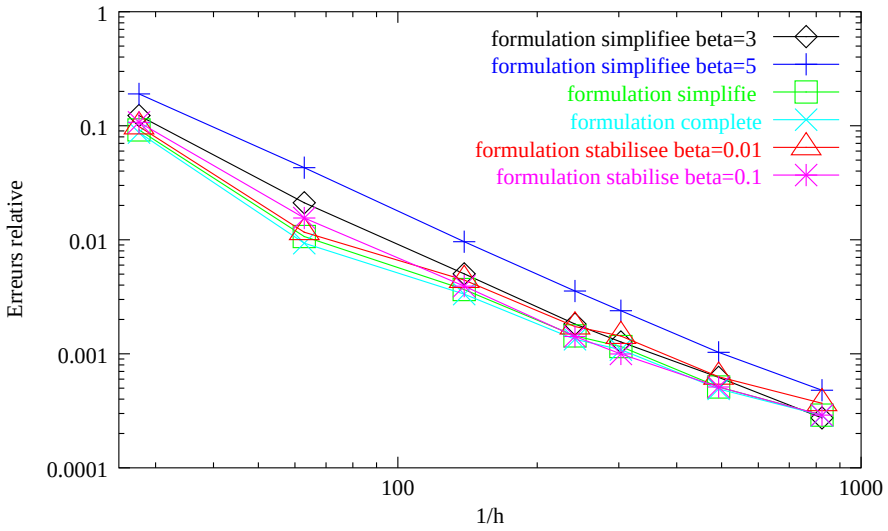


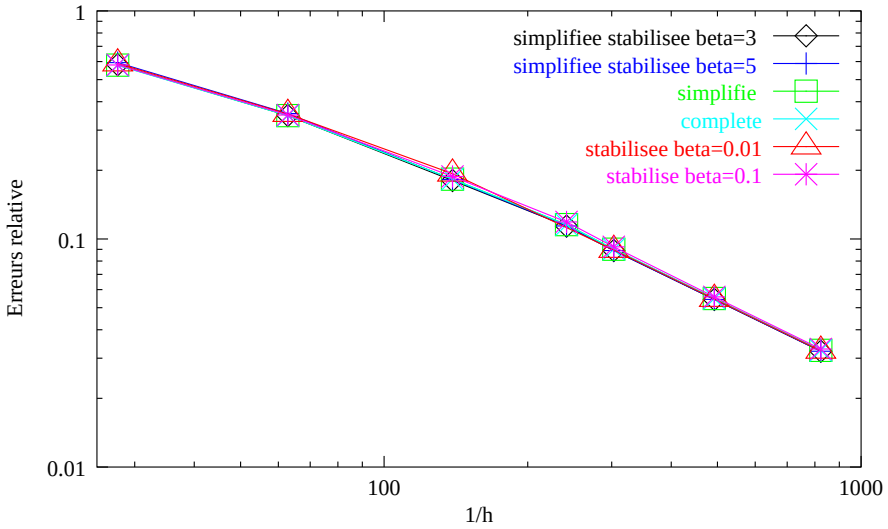
FIG. 8 – *Comportement viscoplastique. Erreur relative en pression en norme 0*



(a) Comportement newtonien. Erreur relative en vitesse en norme 1



(b) Comportement newtonien. Erreur relative en pression en norme 0



(c) Comportement viscoplastique. Erreur relative en vitesse en norme 1

FIG. 9 – *Écoulements de Poiseuille. Évolution des erreurs relatives en fonction de la taille de maille*

5.2 Prise en compte du contact : Écrasement d'un cylindre

5.2.1 Calcul d'erreur numérique :

Dans la majorité des cas, nous n'avons pas à notre disposition de solutions analytiques. L'erreur est alors calculée par rapport à une solution de référence obtenue sur un maillage très fin. Dans ce but nous avons choisi comme solution de référence la solution obtenue par la formulation complète sur un maillage de 31639 noeuds (h=0.21). Nous avons alors effectué des simulations sur cinq maillages allant de 129 à 19685 noeuds(h=3.745 à h=0.285) et ce pour chacune des formulations considérées. Les champs solutions en vitesse et en pression ont alors été transportés en chaque point d'intégration du maillage de référence. Le transport est effectué à l'aide des fonctions de base du maillage courant. (Une procédure préliminaire ayant repéré à quel élément du maillage courant appartient chaque point d'intégration du maillage de référence.)

5.2.2 Résultats

Les tableaux 3 et 4 donnent les taux de convergence mesurés en vitesse et en pression pour des écoulements newtonien et viscoplastique ($m = 0.1$). Les pentes ont été obtenues par des régressions linéaires effectuées sur l'ensemble des maillages sauf celui de 19685 noeuds car l'écart des tailles de maille est alors trop faible pour être significatif. Les résultats sont présentée sur les figure 10 et 11.

		Complète	Simplifiée $\beta = 1$	Simplifiée $\beta = 3$	Simplifiée $\beta = 5$	Stabilisée $\beta^2 = 0.1$	Stabilisée $\beta^2 = 0.01$
Newtonien	p	0.92	0.92	0.91	0.94	1.47	1.16
	C	0.0034	0.0034	0.0033	0.0034	0.01	0.0052
Viscoplastique	p	1.06	1.06	1.11	1.13	xxx	1.02
	C	0.012	0.012	0.077	0.009	xxx	0.031

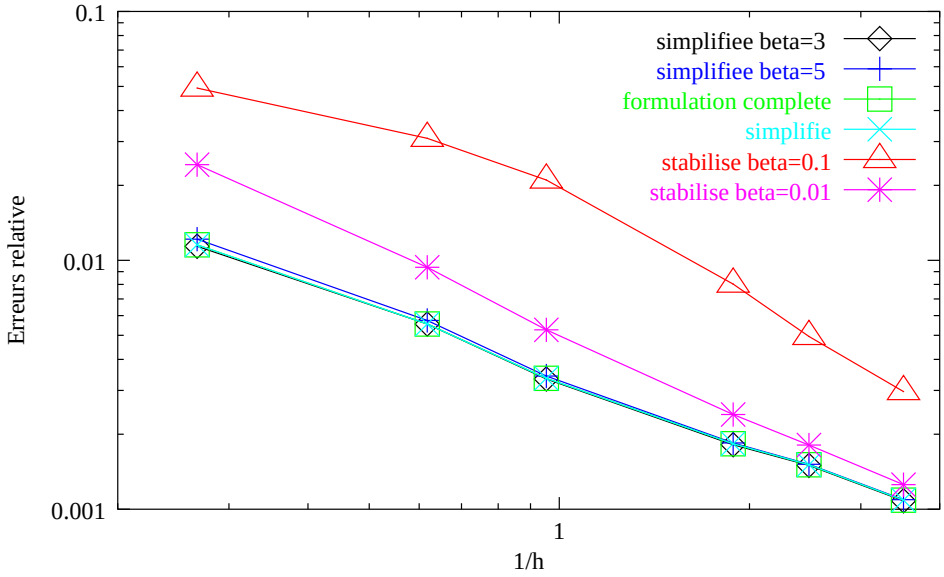
TAB. 3 – *Écrasement d'un cylindre. Taux de convergence sur la vitesse* $\frac{||v_h - \bar{v}||_1}{||\bar{v}||} = ch^p$

		Complète	Simplifiée	Simplifiée $\beta = 3$	Simplifiée $\beta = 5$	Stabilisée $\beta^2 = 0.1$	Stabilisée $\beta^2 = 0.01$
Newtonien	p	1.12	1.12	1.04	1.08	1.26	1.12
	C	0.015	0.015	0.009	0.011	0.089	0.024
Viscoplastique	p	1.06	1.06	1.11	1.14	xxx	1.02
	C	0.012	0.012	0.008	0.009	xxx	0.031

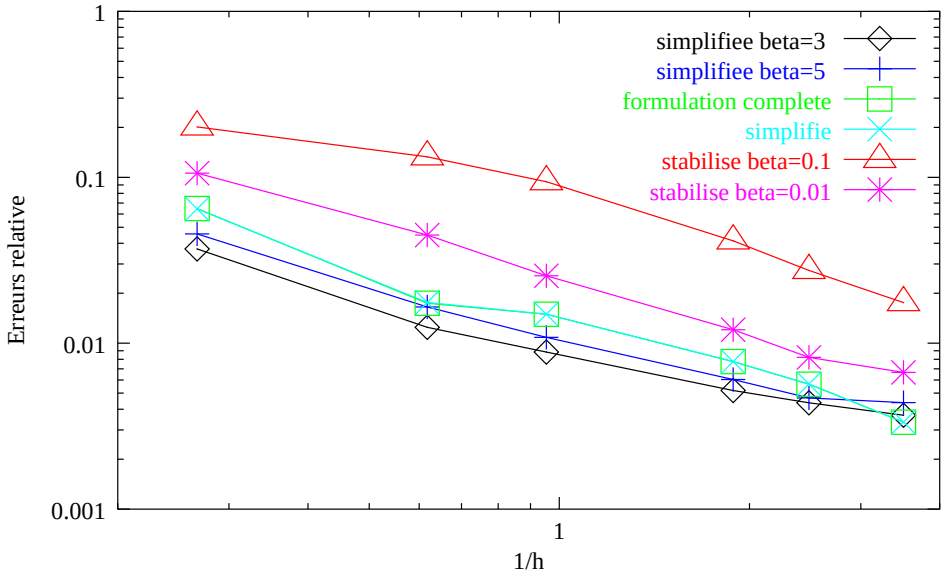
TAB. 4 – *Écrasement d'un cylindre. Taux de convergence sur la pression* $\frac{||p_h - \bar{p}||_0}{||\bar{p}||} = ch^p$

Ces résultats confirment les impressions dégagées déjà sur le cas analytique. Les taux de convergence théoriques sont à nouveaux retrouvés et nous ne constatons pas de différences notables entre le comportement newtonien et le comportement viscoplastique.
Les formulations stabilisées ne doivent leur bon taux de convergence qu'à de moins bonnes solutions sur les maillages grossiers.

Les formulations bulle complète et simplifiée donnent des résultats quasi-identique. Les formulations simplifiées stabilisées fournissent aussi des résultats équivalents voire un peu meilleurs.

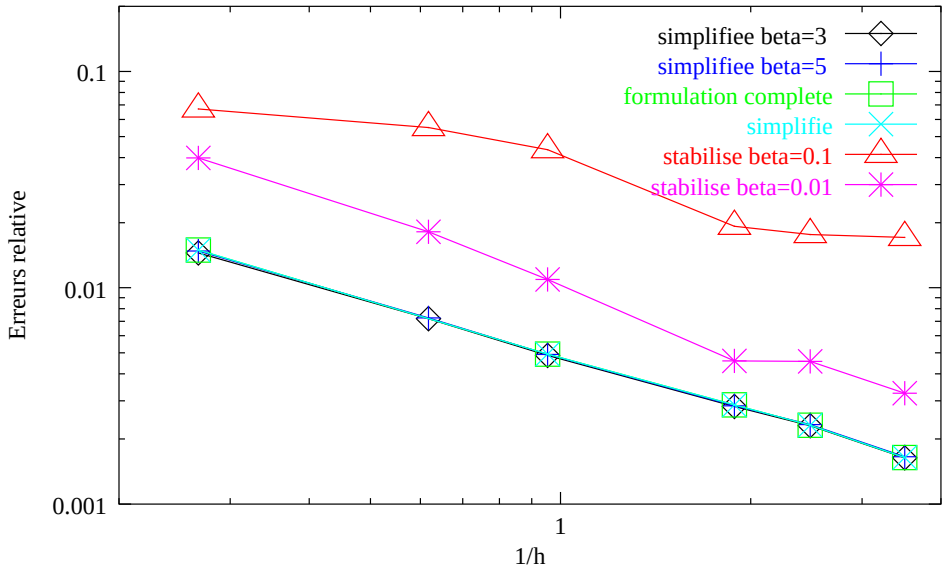


(a) Comportement newtonien. Norme 1 en vitesse

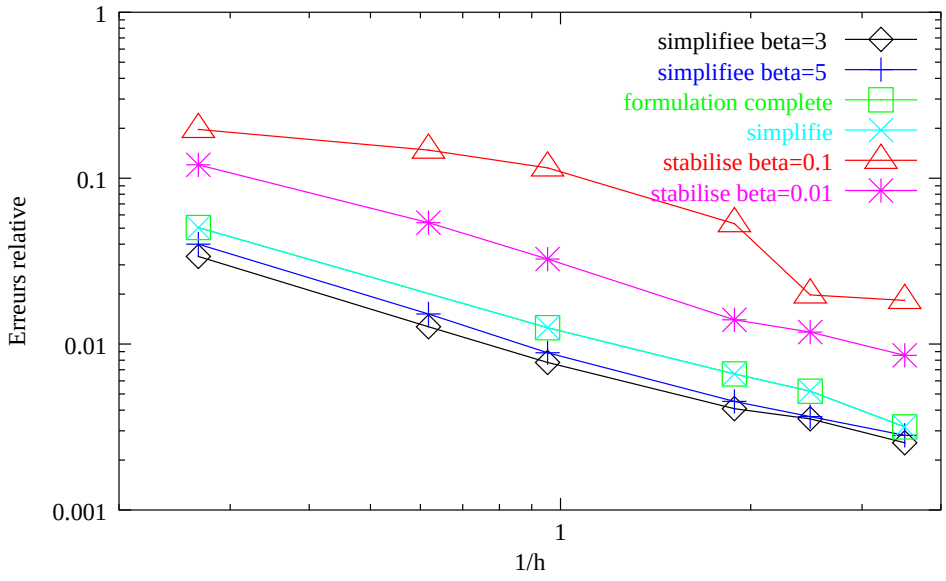


(b) Comportement newtonien. Norme 0 en pression

FIG. 10 – *Écrasement d'un cylindre. Évolution de l'erreur relative en fonction de la taille de maille. Comportement newtonien*



(a) Comportement viscoplastique. Norme 1 en vitesse



(b) Comportement viscoplastique. Norme 0 en pression

FIG. 11 – *Écrasement d'un cylindre. Évolution de l'erreur relative en fonction de la taille de maille. Comportement viscoplastique*

6 Conclusions

Nous avons présenté dans ce chapitres les diverses formulations qui ont été envisagées pour la discrétisation du problème de Stokes généralisé (11). Ces formulations étaient basées sur le MINI-élément P1+P1, qui avait déjà prouvé en 3D son efficacité et sa robustesse. Toutefois, le cadre axisymétrique rend sa mise en oeuvre plus délicate. Il fait apparaître des termes supplémentaires non négligeables à priori dont le coût de calcul est très élevé pour de simple corrections de vitesses. Comme l'introduction d'une bulle est motivé par la nécessité de stabiliser la formulation, il était naturel de s'interroger sur l'utilité de ces termes pour la stabilisation. Notre contribution a principalement consisté en une réflexion sur les hypothèses et simplifications envisageables en axisymétrique. Les formulations envisagées ont ensuite été implémentées dans le logiciel FORGE2®de manière à valider nos hypothèses et comparer les formulations. Il est vite apparu que les formulations globalement stabilisées étaient peu précises et ne pouvaient pas fournir de résultats satisfaisants. La mise en oeuvre de l'élément P1+P1 nous a mené à plusieurs formulations. La formulation complète est apparue très coûteuse sans apporter de précision supplémentaire par rapport à la formulation simplifiée. Au vu de nos nombreux tests numériques, la formulation simplifiée s'est imposée comme la plus satisfaisante. Elle est aussi précise que la formulation complète, beaucoup moins coûteuse et plus proche du 2D plan et du 3D. De plus on peut encore l'optimiser par l'ajout d'un paramètre de stabilisation faisant disparaître les oscillations en pression et améliorant le conditionnement du système linéaire sans détériorer la précision.

Chapitre II

Résolution itérative du problème de Stokes

Nous allons dans ce chapitre nous intéresser à la résolution des systèmes linéaires de la forme:

$$\begin{pmatrix} A & B^T \\ B & -\beta^2 C \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix} \tag{1}$$

Avec: $A \in I\!R^{n\times n}$ symétrique définie positive
$B \in I\!R^{n\times m}$ de rang maximal ($i.e.\ rang(B) = m$)
$C \in I\!R^{m\times m}$ symétrique semi-définie positive
β constante positive ou nulle

Cette matrice est symétrique mais non définie positive puisque $-\beta^2 C$ est semi-définie négative. C'est un problème classique, que l'on rencontre couramment en mécanique des fluides, en élasticité linéaire incompressible et quasi incompressible ou encore dans le contexte de l'optimisation et des problèmes de point selle.

Une résolution efficace de ces problèmes est cruciale et, dès 1984 *O. C. Zienkiewicz et al.* [87] se sont intéressé à différentes stratégies de résolution itérative de ce problème en comparant les méthodes de Gradient Conjugué, du résidu minimal et de la plus grande pente.

En préambule, le premier paragraphe présente les méthodes que l'on rencontre dans la bibliographie. La motivation de cette revue étant de montrer que deux grands types d'approches itératives sont possibles, avec chacune leurs avantages et leurs inconvénients, et que les questions quant au(x) choix à effectuer se posent tant en programmation séquentielle qu'en parallèle.

1 Méthodes itératives de résolution d'un problème mixte

1.1 Méthodes d'Uzawa et de Lagrangien

Pour une revue bien plus complète de ces méthodes nous renvoyons le lecteur à *M. Fortin* et *R. Glowinsky* [40].

Soit la fonctionnelle:

$$J(v) = \frac{1}{2}(Av, v) - (f, v) \tag{2}$$

Le problème (1) est équivalent au problème de minimisation sous contraintes:

$$\min_{v \in \mathcal{K}} J(v) \qquad \text{où} \quad \mathcal{K} = \left\{ v \in \mathcal{V} \text{ tels que } Bv - \beta^2 Cp = 0 \right\} \tag{3}$$

Il peut donc se réinterpréter comme un problème de point selle associé au Lagrangien $\mathcal{L}$ ou au Lagrangien augmenté $\mathcal{L}_r$ [40]:

$$\begin{aligned} \mathcal{L}(v, q) &= \frac{1}{2}(Av, v) - (f, v) + \left(q, Bv - \beta^2 Cq\right) \\ \mathcal{L}_r(v, q) &= \mathcal{L}(v, q) + \frac{r}{2}\left|Bv - \beta^2 Cq\right|^2 \end{aligned} \tag{4}$$

Où q est le paramètre de Lagrange associée à la contrainte d'incompressibilité et r un coefficient de pénalisation. La pression hydrostatique est alors égale à q.

Les méthodes de résolution les plus anciennes du problème (1) sont issues de la théorie de l'optimisation et basées sur la résolution des problèmes de point selle associés aux Lagrangiens (4). Les méthodes de Gradient Projeté sont particulièrement adaptées à la résolution de ces problèmes. Toutefois, la construction de l'opérateur de projection étant souvent délicate, on leur préfère généralement une méthode de type Uzawa. La mise en oeuvre de l'algorithme d'Uzawa revient à découpler, au prix de deux résolutions imbriquées, les résolutions en vitesse et en pression. Cet algorithme, appliqué au Lagrangien augmenté (4), se présente ainsi :

Soit $k = 0$; $\quad p^0$ fixé :

Tant que non convergence faire :

$$\begin{aligned} &\left(A + rB^T B\right) u^{k+1} = \omega_k \left(f - B^T p^k\right) \\ &p^{k+1} = p^k + \rho_k \left[Bu^{k+1} - \beta^2 Cp^k\right] \\ &k = k + 1 \end{aligned} \tag{5}$$

Fin tant que

Le paramètre ρ_k est généralement choisi égal à r, et le paramètre ω_k est un paramètre de relaxation souvent pris égal à 1. Cette méthode nécessite donc de déterminer, a priori, deux paramètres avec les inconvénients que cela comporte (*cf.* page suivante).

En pratique, on applique souvent la méthode d'Uzawa en faisant apparaître dans l'équation en pression la matrice du complément de Schur $BA^{-1}B^T$. Le principe est de découpler les variables de manière à obtenir un système symétrique défini positif en pression. Comme A est symétrique définie positive, la première équation de (1) peut encore s'écrire:

$$u = A^{-1}\left(f - B^T p\right) \tag{6}$$

et, en substituant u dans la seconde équation de (1), on obtient:

$$\left(BA^{-1}B^T + \beta^2 C\right) p = BA^{-1} f \tag{7}$$

Comme la matrice B est de rang maximal, la matrice $BA^{-1}B^T$ est encore symétrique définie positive.

L'algorithme d'Uzawa est en fait une simple méthode de gradient appliquée à la fonctionnelle duale $J^* = \min_v \mathcal{L}(v, q)$. On peut donc envisager de le préconditionner pour accélérer la convergence.

Propriété :

On peut montrer [13] que pour tout élément finis stable *i.e.* vérifiant la condition *inf-sup* (chapitre I (18) page 8) :
il existe deux constantes γ et Γ telles que :

$$\gamma^2 \leq \frac{p^T(BA^{-1}B^T + \beta^2 C)p}{p^T M_p p} \leq \Gamma^2 \qquad \forall p \in I\!R^n - \{0\} \tag{8}$$

Où M_p est la matrice de masse en pression, dont le conditionnement est indépendant de la taille de maille [43] et γ la constante issue de la condition *inf-sup*.

Cet encadrement implique que le conditionnement de la matrice $BA^{-1}B^T + \beta^2 C$ (et donc le nombre d'itérations nécessaires pour l'inverser) est indépendant de la taille de maille. Ainsi tout système basé sur cette matrice peut être rapidement inversé par un gradient conjugué ou par tout autre méthode itérative. L'algorithme de correction de pression proposé par *J. Atanga et D. Silvester* [5] est une extension de ces schémas type Uzawa modifié. Celui-ci vise à résoudre (7) par un gradient conjugué avec un simple préconditionnement diagonal et à en déduire u solution de (6). Les auteurs montrent que pour de faibles valeurs du coefficient de stabilisation β ($\beta \leq 1$) cet algorithme donne de meilleurs résultats que celui d'Uzawa.

Ces méthodes sont toutes basées sur la résolution de problèmes imbriqués; elles nécessitent à chaque itération la résolution de problèmes de la forme $(A + rBB^T)z = b \quad r \geq 0$ pour lesquels une méthode directe est performante. Nous qualifierons ce second niveau de résolution d'interne. L'augmentation, ces dernières années, de la taille des problèmes à résoudre a rendu les solveurs directs de plus en plus pénalisants. La mise en oeuvre de ces méthodes a alors entraîné l'imbrication de deux processus itératifs nécessitant un bon préconditionneur pour le système interne. En effet, peu de choses peuvent être faites pour le problème externe car le complément de Schur est déjà très bien conditionné (8). En revanche la convergence du problème interne est cruciale pour l'efficacité de la méthode, la matrice A ayant un conditionnement en $O(h^{-2})$.

Inconvénients des méthodes emboîtées:

Ces méthodes sont très intéressantes lorsqu'un solveur direct est utilisé pour inverser ces problèmes. Dans ce cas, on effectue une factorisation (coûteuse) en début de résolution puis à chaque itération les problèmes internes sont résolus par une descente/remontée dont le coût est équivalent à celui de deux produits matrice/vecteur. Mais, dans le cas où la méthode directe s'avère trop coûteuse, un second ensemble d'itérations doit être effectué. Nous obtenons alors un solveur avec des itérations intérieures et extérieures pour la résolution de (7).

Un second inconvénient réside dans la détermination des paramètres supplémentaires ω_k et ρ_k. En effet, leur calcul est souvent d'un coût prohibitif et l'on se contente généralement de les estimer. Une valeur de ρ_k élevée favorise la convergence de l'algorithme d'Uzawa mais induit un mauvais conditionnement du système linéaire interne, tandis qu'une faible valeur permet une résolution itérative aisée mais au détriment de la convergence d'Uzawa.

1.2 Méthodes alternatives à un seul niveau

Gradient Conjugué:

J. H. Bramble et J. E. Pasciak [10] ont développé une approche originale en reformulant le problème (1) en un problème symétrique défini positif inversible ne nécessitant plus la mise en oeuvre de méthodes itératives imbriquées.

Ils introduisent la matrice de reformulation ou de préconditionnement :

$$\mathcal{T} = \begin{pmatrix} A_0^{-1} & 0 \\ BA_0^{-1} & -I \end{pmatrix} \tag{9}$$

où la matrice A_O vérifie :

$$\exists\, \alpha > 0 \text{ tel que} \quad 0 < ((A - A_0)\, u, u) < \alpha\, (Au, u) \quad \forall\, u \in \mathcal{V}_h - \{0\} \tag{10}$$

En prémultipliant le problème de Stokes (1) par $\mathcal{T}$, soit $\mathcal{M} = \mathcal{T}\mathcal{A}$, on aboutit au problème équivalent :

$$\mathcal{M} \begin{pmatrix} u \\ p \end{pmatrix} \equiv \begin{pmatrix} A_0^{-1}A & A_0^{-1}B^T \\ BA_0^{-1}A - B & BA_0^{-1}B^T + \beta^2 C \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} A_0^{-1}f \\ BA_0^{-1}f \end{pmatrix} \tag{11}$$

et (10) nous permet de définir le $\mathcal{T}$ produit scalaire sur $\mathcal{V}_h \times \mathcal{P}_h$:

$$\left[\begin{pmatrix} u \\ v \end{pmatrix}, \begin{pmatrix} w \\ x \end{pmatrix} \right]_{\mathcal{T}} \equiv ((A - A_0)u, w) + (v, x) \tag{12}$$

les auteurs montrent alors que la matrice $\mathcal{M}$ est symétrique définie positive pour ce produit scalaire (*Théorème 1* [10]). La méthode du gradient conjugué peut donc s'appliquer pour résoudre le système (11) en considérant le $\mathcal{T}$-produit scalaire (12). On notera toutefois que cela demande une "mise à l'échelle" du préconditionneur A_0^{-1} qui nécessite d'évaluer la plus petite valeur propre de $A_0^{-1}A$. Cette méthode est appliquée, par exemple, dans *H. Elman* [30], qui calcule alors A_0^{-1} à partir d'un incrément d'un V-cycle multigrille dérivé du laplacien discret.

Résidu minimal:

La seconde approche repose sur une résolution par la méthode du résidu minimal encore connue sous le nom de MINRES. Cette méthode introduite par *Paige et Saunders* en 1975 [60] est la méthode de Krylov équivalente au gradient conjugué pour des problèmes symétriques non définis positifs [4]. *T. Rusten et R. Winther* [68] l'ont appliquée au problème (1) avec $\beta = 0$, tandis que *A. Wathen et D. Silvester* [83, 84] l'ont utilisée pour la résolution du problème de Stokes stabilisé (1). On trouvera une étude théorique sur le comportement asymptotique de la méthode dans *Wathen et al.* [82].

Discussion:

Une comparaison avec la méthode de la correction de pression est effectuée par *A. Wathen et A. Ramage* [65]. Les expérimentations numériques faites par les auteurs montrent que la méthode du résidu minimal est la plus performante, même si ce n'est pas le cas en théorie. *H. Elman* a présenté dans [30] une comparaison de plusieurs méthodes (Uzawa, Gradient Conjugué Préconditionné appliqué au système (11), MINRES) pour des systèmes linéaires de la forme (1) issus de discrétisations éléments finis et différences finies d'un problème de Stokes sur un carré. De ses nombreuses conclusions, on peut retenir que les performances des méthodes sont très proches. La méthode du résidu minimal semble coûter légèrement plus cher que les deux autres, mais a l'avantage par rapport à Uzawa et au gradient conjugué de ne pas nécessiter la détermination de paramètres.

2 La méthode du résidu minimal

Nous nous contenterons d'une présentation succincte de cette méthode et nous renvoyons le lecteur à la thèse de *S. Marie* [57] ou encore à la taxonomie établie par *S. Ashby et al.* [4] pour une description plus complète.

2.1 Présentation

On considère le système linéaire:

$$\mathcal{A}x = b \tag{13}$$

avec $\mathcal{A} \in I\!R^{N\times N}$ symétrique non définie positive, et $x,\ b \in I\!R^N$.

Nous utiliserons la même notation par bloc que dans (1) pour la matrice $\mathcal{A}$.

La méthode du gradient conjugué repose sur la minimisation de la fonctionnelle,

$$J(x) = \frac{1}{2}(\mathcal{A}x, x) - (b, x)$$

dont la stricte convexité (qui entraîne l'existence et l'unicité d'un minimum) n'est assurée que si $\mathcal{A}$ est symétrique définie positive.

La méthode du résidu minimal repose sur la minimisation de la fonctionnelle d'erreur:

$$E(r) = (r, r) \tag{14}$$

où: $\quad r \equiv r(x) = b - \mathcal{A}x \qquad$ est le résidu.

On vérifie que E est encore strictement convexe lorsque $\mathcal{A}$ est symétrique non définie positive.

La méthode du résidu minimal est une méthode de descente classique: à chaque itération on détermine une pente p_k et un scalaire α_k qui nous permettent de calculer l'itérée suivante:

$$x_{k+1} = x_k + \alpha_k p_k$$

et d'en déduire le résidu:

$$r_{k+1} = r_k - \alpha_k \mathcal{A} p_k$$

Les valeurs de α_k et p_k sont données dans l'algorithme II .1 page 38.

Préconditionnement:

Dans de nombreux cas, les méthodes itératives convergent lentement ou ne convergent pas du tout si le conditionnement de la matrice $\kappa(A)$ (le rapport de la valeur propre de plus grand module sur celle de plus petit module) est trop élevé. Le remède usuel consiste à introduire une matrice de préconditionnement $\mathcal{M}$ et à remplacer (13) par:

$$\mathcal{M}^{-1}\mathcal{A}x = \mathcal{M}^{-1}b \qquad \text{ou encore} \qquad \left\{ \begin{array}{l} \mathcal{M}_1^{-1}\mathcal{A}\mathcal{M}_1^{-T} y = \mathcal{M}_1^{-1} b \\ \mathcal{M}_1^T x = y \\ \text{avec } \mathcal{M} = \mathcal{M}_1 \mathcal{M}_1^T \end{array} \right. \tag{15}$$

Le problème est alors de trouver un préconditionneur $\mathcal{M}$ satisfaisant les contraintes suivantes:

1. $\mathcal{M}$ doit être une bonne approximation de $\mathcal{A}$, de manière à ce que le conditionnement du système préconditionné soit bien meilleur que celui du système original *i.e.*:

$$\kappa\left(\mathcal{M}^{-1}\mathcal{A}\right) << \kappa\left(\mathcal{A}\right)$$

2. La construction de $\mathcal{M}$ doit être peu coûteuse comparée à celle de $\mathcal{A}$,
3. Le système $\mathcal{M}y = z$ doit être beaucoup plus facile à résoudre que le problème original.

La méthode du résidu minimal préconditionné revient alors à minimiser la fonctionnelle:

$$E(r) = (\mathcal{M}^{-1}r, r) \tag{16}$$

On vérifie que cette fonctionnelle est strictement convexe si et seulement si le préconditionneur $\mathcal{M}$ est symétrique défini positif. On a de plus la relation:

$$E(r^{k+1}) = E(r^k) - \frac{\left(r_k, \mathcal{M}^{-1}\mathcal{A}p_k\right)^2}{(\mathcal{M}^{-1}\mathcal{A}p_k, \mathcal{A}p_k)} \tag{17}$$

Cette relation montre clairement que si le préconditionneur $\mathcal{M}$ est symétrique non défini positif la décroissance de l'erreur à chaque itération n'est plus assurée.

Algorithme :

La méthode consiste en un calcul de la pente de l'itérée suivante par conjugaison dans le plan formé par r_{k+1} et p_k. On détermine alors un paramètre optimal de descente β_k [57] et on pose:

$$p_{k+1} = \mathcal{M}^{-1}r_{k+1} + \beta_k p_k$$

Le critère d'arrêt est que la norme du résidu relatif soit suffisamment petite:

$$\frac{||r_k||}{||r_0||} \leq \epsilon_{iter}$$

L'algorithme du résidu minimal (MINRES) est alors:

Algorithme II .1 Méthode du résidu minimal préconditionné

$r_0 = b$ *résidu initial*
$p_0 = \mathcal{M}^{-1}r_0$ *pente initiale*
$z_0 = \mathcal{A}p_0$

Tant que $||r_k|| \geq \epsilon_{Iter}\,||r_0||$ répète:

$\alpha_k = \dfrac{(r_k, \mathcal{M}^{-1}z_k)}{(\mathcal{M}^{-1}z_k, z_k)}$ *pas optimal*

$x_{k+1} = x_k + \alpha_k p_k$ *itérée suivante*

$r_{k+1} = r_k - \alpha_k z_k$ *réactualisation du résidu*

$\beta_k = -\dfrac{(\mathcal{A}\mathcal{M}^{-1}r_{k+1}, \mathcal{M}^{-1}z_k)}{(\mathcal{M}^{-1}z_k, z_k)}$ *paramètre optimal de descente*

$p_{k+1} = \mathcal{M}^{-1}r_{k+1} + \beta_k p_k$ *calcul de la direction de descente*

$z_{k+1} = \mathcal{A}\mathcal{M}^{-1}r_{k+1} + \beta_k z_k$

$k = k + 1$

fin Tant que

fin algorithme

Celui-ci nécessite à chaque itération 4 produits scalaires, 2 étapes de préconditionnement et 1 produit matrice/vecteur. Dans cet algorithme, on ne calcule qu'une seule pente par

itérations. Il peut se généraliser au choix de plusieurs pentes avec l'algorithme **Orthomin**. On recherche alors la direction de descente p_{k+1} comme combinaison linéaire du résidu r_{k+1} et des pentes précédentes. Pour des raisons pratiques on tronque en général cette combinaison aux $s+1$ pentes précédentes. C'est alors l'algorithme **Orthomin(s)**. Dans certain cas, cet algorithme peut stagner. Une alternative consiste à utiliser la méthode robuste **Orthodir**, générant des pentes orthogonales, ou sa version tronquée **Orthodir(s)** qui malheureusement peut à nouveau diverger. Une solution élégante consiste alors à utiliser une forme hybride **Orthomin/Orthodir**, introduite par *Fletcher* [37], qui en exploitant la colinéarité des pentes calculées par les deux méthodes assure efficacité et robustesse.

Dans le cas de préconditionneurs symétriques définis positifs nous utiliserons la forme hybride à une seule pente car elle a prouvé son efficacité et sa robustesse. Dans le cas de préconditionneurs non symétriques définis positifs, **Orthodir** échoue à trouver des pentes orthogonales et la méthode hybride diverge. Nous utiliserons alors la méthode du résidu minimal présentée dans l'algorithme II .1 qui s'avère robuste.

3 Le préconditionnement

Le choix classique de préconditionneurs pour le problème de Stokes ou pour des problèmes de point-selle est généralement celui de matrices symétriques définies positives de la forme:

$$\mathcal{M} = \begin{pmatrix} \hat{A} & 0 \\ 0 & \hat{C} \end{pmatrix} \tag{18}$$

Le choix de $\hat{C}$ est guidé par la nécessité d'avoir $\mathcal{M}$ définie positive.

Dans les pages qui suivent, nous allons aussi considérer le cas de préconditionneurs non symétriques définis positifs. En effet, alors que la théorie tend à prouver que ces préconditionneurs ne sont pas fiables, la pratique a montré qu'ils s'avéraient efficaces et rapides, du moins pour nos problèmes de forgeage [57]. Toutefois ce choix n'est pas sans risques et nécessite la mise en place de méthodes alternatives en cas de divergence ou de stagnation. Nous aborderons ce point au paragraphe 4.4.
Nous terminerons par **P** (pour positif) les noms des préconditionneurs définis positifs.

3.1 Les préconditionneurs mis en place

3.1.1 Préconditionneurs diagonaux et bloc diagonaux

De part leur simplicité les préconditionneurs diagonaux et blocs diagonaux sont d'un grand intérêt pour le calcul parallèle. Ils sont peu coûteux à assembler et à inverser et, ils sont aisément parallélisables (cf. chapitre III ou encore [57]).

Le préconditionneur le plus simple consiste à prendre la diagonale de la matrice des vitesses A et **l'opposé** de la diagonale en pression. On obtient ainsi un préconditionneur symétrique défini positif puisque A et C sont définies positives. Nous le noterons **DiagP**:

$$\mathbf{DiagP} \equiv |diag(\mathcal{A})| = \begin{pmatrix} diag(A) & 0 \\ 0 & \beta^2\, diag(C) \end{pmatrix}$$

Un préconditionneur, plus proche de $\mathcal{A}$ mais non défini positif peut être construit en

prenant simplement la diagonale de la matrice $\mathcal{A}$, *i.e.*

$$\mathbf{Diag} \equiv diag(\mathcal{A}) = \begin{pmatrix} diag(A) & 0 \\ 0 & -\beta^2\ diag(C) \end{pmatrix}$$

On peut, de manière similaire, construire des préconditionneurs plus efficaces et à peine plus coûteux en prenant les blocs diagonaux. On relie alors les deux composantes nodales de la vitesse en blocs 2×2 et en conservant encore l'inverse de la diagonale de la pression [21]:

$$\mathbf{VipP} = \begin{pmatrix} \begin{pmatrix} \ddots & 0 & 0 \\ 0 & \begin{pmatrix} a_{i,i} & a_{i,i+1} \\ a_{i+1,i} & a_{i+1,i+1} \end{pmatrix} & 0 \\ 0 & \vdots & \ddots \end{pmatrix} & 0 \\ 0 & \beta^2\ diag(C) \end{pmatrix}$$

On peut, de la même manière, construire deux autres préconditionneurs, plus proches de $\mathcal{A}$ mais non symétriques définis positifs:

Vip: On relie les blocs vitesse 2×2 avec la diagonale de la pression.

Tot: On inverse directement les blocs 3×3 reliant les deux vitesses et la pression de chaque noeud.

Comme nous le verrons aux paragraphes 3.2.3 et 3.3, la méthode itérative s'avère beaucoup moins efficace en 2D qu'en 3D. Il y a deux raisons à cela:

1. Les problèmes 2D étant de bien plus petite taille que les problèmes 3D, les méthodes directes sont plus performantes pour les inverser.
2. En terme de comportement asymptotique, passer d'un solveur direct à un solveur itératif est bien plus favorable en 3D qu'en 2D. Cela est mis en évidence au paragraphe 3.2.3 et à la figure 3.

On obtient donc un solveur décevant, incapable de concurrencer une méthode directe sauf sur des problèmes de très grosse taille. De plus, les performances du code parallèle que nous présentons au chapitre III montrent que même en parallèle, la majorité des problèmes seraient résolus en plus de temps que par la méthode directe. Ainsi, la nécessité de mettre au point un solveur itératif plus performant nous a amené à nous tourner vers des préconditionneurs par factorisations incomplètes, plus efficaces mais aussi plus complexes à développer.

3.1.2 Préconditionneurs par factorisation incomplète

Principe :

Le principe de ces méthodes est de construire une factorisation incomplète de la matrice en calculant seulement les termes initialement non nuls, sans se soucier de savoir si l'on retrouve les valeurs nulles lors de la factorisation. On calcule ainsi une factorisation que l'on stockera le plus souvent de manière compacte dans un tableau identique à celui utilisé pour la matrice $\mathcal{A}$ (adressé avec les mêmes pointeurs).

On définit l'ensemble S des indices (i, j) des termes non nuls de la matrice $\mathcal{A}$,

$$S = \{(i,j), \quad 1 \leq i,j \leq N \ \text{ tels que } \ a_{ij} \neq 0\}$$

On construit une factorisation incomplète $M = LDL^T$ de la matrice $\mathcal{A}$ en identifiant les coefficients l_{ij} et d_i des matrices L et D tels que:

$$m_{ij} = a_{ij} \quad \forall\, (i,j) \in S \tag{19}$$

$$\text{Où:} \quad \begin{cases} m_{ij} = \displaystyle\sum_{k=1}^{N} l_{ik} d_{kk} l_{kj} \\ l_{ij} = 0 \qquad \text{si } (i,j) \notin S \end{cases} \tag{20}$$

Cette factorisation n'est pas unique, elle dépend de la numérotation ainsi que de l'algorithme choisi pour effectuer l'identification [1]. Par abus de notation, nous noterons $IC(0)$ cette factorisation LDL^T. Le (0) signifie que l'on n'accorde aucun niveau de remplissage supplémentaire par rapport à celui de $\mathcal{A}$. En effet, on peut de la même manière construire des factorisations incomplètes autorisant un remplissage supplémentaire. Par exemple, le calcul d'un terme supplémentaire par lignes aboutit à une méthode $IC(1)$. D'autres variantes sont encore possibles et nous renvoyons à *I. Gustafsson* [48] et à *T. Chan et H. van der Vorst* [16] pour plus de détails.

Une méthode populaire dans la bibliographie consiste à effectuer une factorisation incomplète modifiée $MIC(0)$ (ou $MLDL(0)$). Cette méthode procure un algorithme plus robuste permettant de garantir l'existence de la factorisation, ce qui n'est pas le cas des factorisations incomplètes simples. De plus, elle permet de conserver une partie de l'information contenue dans tous les termes non nuls de la factorisation que l'on a décidé de ne pas stocker. Par un procédé de condensation, ceux-ci sont utilisés pour renforcer la diagonale du préconditionneur ce qui assure la robustesse de l'algorithme de factorisation et celle du préconditionneur dans les cas limites. La condition (19) est alors remplacée par:

$$\begin{cases} \displaystyle\sum_{j=1}^{N} m_{ij} = \sum_{j=1}^{N} a_{ij} & \forall 1 \le i \le N \\ m_{ij} = a_{ij} & \text{si } i \neq j \text{ et } (i,j) \in S \end{cases} \tag{21}$$

En pratique, la méthode $IC(0)$ offre souvent le meilleur compromis. Les méthodes $IC(n)$, $n > 0$ ou $MIC(n)$ s'avèrent plus complexes à mettre en oeuvre et génèrent des surcoûts en mémoire et en temps de calcul (factorisation, descente-remontée) qui ne sont pas toujours justifiés. Par la suite, nous ne considérerons que des factorisations de la forme LDL^T sans remplissage supplémentaire et le (0) sera implicite.

Choix effectués :

Selon le même principe que pour les préconditionneurs diagonaux nous avons construit plusieurs types de préconditionneurs par factorisation incomplète.
Un préconditionneur symétrique défini positif de la forme (18) est construit en choisissant pour $\hat{A}$ la factorisation incomplète de A et l'opposé de la diagonale en pression pour $\hat{C}$. Nous noterons ce préconditionneur **IcP**, et **MicP** celui construit à partir d'une factorisation incomplète modifiée de A. De manière similaire, on construit aussi:

IcV: la factorisation incomplète de la matrice A avec la diagonale de la pression.

1. L'identification des termes de la factorisation incomplète est un processus similaire au calcul d'une factorisation complète pour lesquelles il existe au moins six implémentations différentes (cf. [47], p. 98-100). Dans le cas d'une factorisation incomplète, ces six implémentations produisent, a priori, six préconditionneurs différents.

IC: la factorisation incomplète de la matrice complète vitesse/pression $\mathcal{A}$.

Nos essais numériques ont montré que les préconditionneurs symétriques non définis positifs construits à partir des méthodes **MIC** ne sont pas efficaces et conduisent à la divergence ou à la stagnation du solveur itératif. Nous ne les présenterons donc pas ici.

3.2 Estimations du coût et de la complexité des méthodes

Soit $Nbnoe$ le nombre de noeuds du maillage, $Nbelt$ le nombre d'éléments, et h une taille de maille caractéristique. On estime généralement:

$$\begin{array}{ll} \text{En 2D} & \left\{ \begin{array}{rcl} h & \sim & Nbnoe^{-\frac{1}{2}} \\ Nbelt & \sim & 2 \times Nbnoe \end{array} \right. \\ \text{En 3D} & \left\{ \begin{array}{rcl} h & \sim & Nbnoe^{-\frac{1}{3}} \\ Nbelt & \sim & 5 \times Nbnoe \end{array} \right. \end{array} \tag{22}$$

Soit N le nombre de degrés de liberté associés au problème et d la dimension de l'espace.

3.2.1 Le stockage :

Stockage méthode directe :

Dans le cadre d'un solveur direct, il est nécessaire de stocker l'ensemble du profil de la matrice (par exemple avec un stockage dit ligne de ciel) de manière à pouvoir ranger la factorisation dans le même tableau. On peut approcher le profil de la matrice par le nombre d'éléments de de la bande.

Soit lbg la largeur de bande géométrique et soit:

$$lb_2 = (d+1)lbg \quad \text{la demi-largeur de bande de la matrice}$$

On estime généralement:

$$\left\{ \begin{array}{ll} lbg \sim \sqrt{Nbnoe} & \text{en 2D} \\ lbg \sim Nbnoe^{2/3} & \text{en 3D} \end{array} \right. \quad \text{ce qui revient à} \quad lbg \sim h^{1-d} \tag{23}$$

Comme $N = (d+1)Nbnoe$, nous avons alors :

$$Prof(A) \sim Nlb_2 \sim (d+1)^2 h^{1-2d} \tag{24}$$

Stockage méthode itérative :

Pour une méthode itérative de type gradient conjugué, on peut se contenter de stocker uniquement les termes non nuls de la matrice (celle-ci n'étant utilisée que pour des produits Matrice/Vecteur). On utilise alors un stockage compact ou morse, appelé CRS (Compressed Row Storage) dans la bibliographie anglo-saxonne.

Notons n_z le nombre de valeurs non nulles de la matrice $\mathcal{A}$.

Soit $\overline{v} \sim \left\{ \begin{array}{ll} 6 & \text{en 2D} \\ 12 & \text{en 3D} \end{array} \right.$ le nombre moyen de voisins d'un noeud du maillage, qui est **indépendant de la taille de maille**.
On approche n_z par:

$$n_z \sim (d+1)\overline{v}\, N = (d+1)^2\, \overline{v}\, Nbnoe \tag{25}$$

Comparaison des coûts de stockage:

Le tableau 1 résume l'espace mémoire nécessaire en 2D et 3D selon le mode de stockage utilisé en fonction du nombre de noeuds du maillage. Ces valeurs sont tracées sur la figure 1. Nous illustrons ces valeurs dans le tableau 2 pour deux tailles de mailles (grossière et fine) représentatives des calculs envisagés.

	Méthode directe Stockage profil		Méthode itérative Stockage morse	
	2D	3D	2D	3D
Stockage	$9Nbnoe^{3/2}$	$16Nbnoe^{5/3}$	$54Nbnoe$	$192Nbnoe$

TAB. 1 – *Coûts de stockage en 2D et en 3D*

	h=1/25			h=1/100		
	Nbnoe	Profil	Morse	Nbnoe	Profil	Morse
2D	676	1.07 Mo	0.25 Mo	10201	69 Mo	4.11 Mo
3D	17576	149 Mo	23 Mo	$\sim 1.10^6$	1192 Go	1.46 Go

TAB. 2 – *Espace mémoire nécessaire pour stocker la matrice selon la taille de maille relative à la discrétisation d'un carré ou d'un cube de coté 1*

L'évolution linéaire de la quantité de stockage pour un stockage morse à la place d'une évolution polynomiale pour un stockage profil met clairement en évidence l'intérêt d'avoir un stockage compact, impossible avec un solveur direct. L'utilisation d'un solveur itératif permet donc de traiter, sur une machine donnée, des problèmes de bien plus grande taille.

Il apparaît ainsi clairement que les méthodes directes ne sont pas envisageables en 3D compte tenu du nombre de degrés de libertés nécessaires. De plus en 3D le coût de stockage du profil est d'une puissance polynomiale supérieure à celui du 2D. Pour le 2D, les conclusions ne sont pas aussi marquées. Même si le gain est très important, le stockage n'est cependant pas un argument décisif en faveur des méthodes itératives. En effet des processeurs avec 256 voire 512 Mo de RAM sont déjà disponibles sur le marché et permettent de traiter des problèmes de très grosse taille. On préfère toutefois ne pas les utiliser dans l'optique du calcul parallèle en raison de deux inconvénients majeurs: ils ne sont réellement efficaces que sur des architectures partagées et, leurs performances s'écroulent au-delà de 8 processeurs.

3.2.2 Complexité

Nous nous plaçons dans le cadre de systèmes à $(d+1)$ degrés de liberté par noeuds, d étant la dimension de l'espace.

3.2.2.1 Méthode directe

On rappelle le coût de calcul d'une méthode directe:

$$\textbf{Factorisation:} \qquad \frac{N \times lb_2^2}{2} = \frac{(d+1)^3 \times Nbnoe \times lb_g^2}{2} \tag{26}$$

$$\textbf{Descente-remontée:} \qquad 2Nlb_2 = 2(d+1)^2 \times Nbnoe \times lb_g \tag{27}$$

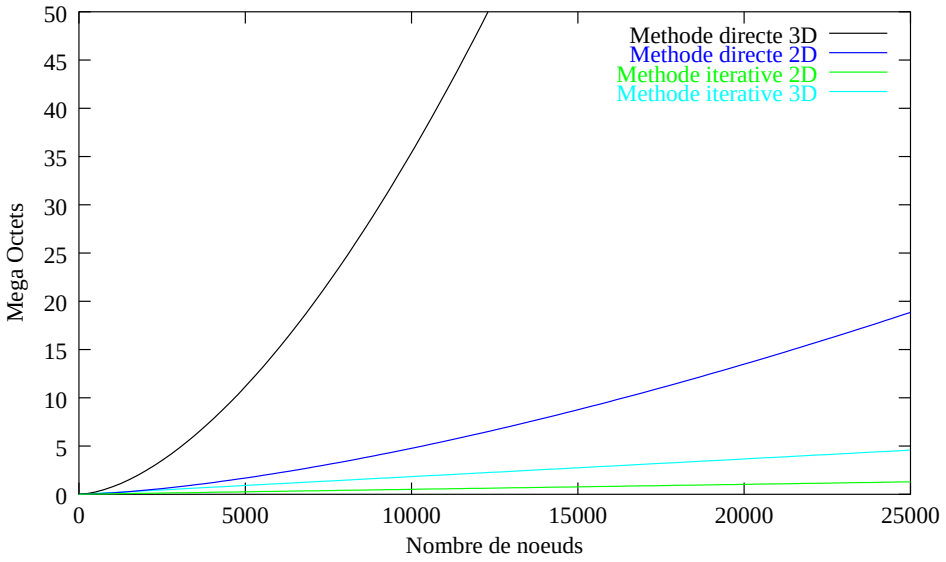


FIG. 1 – *Coûts de stockage en Mo des méthodes directes et itératives en 2D et en 3D à nombre de noeuds fixés*

3.2.2.2 Méthode itérative :

La méthode du résidu minimal présentée dans l'algorithme (II .1), nécessite à chaque itération:

1 produit matrice/vecteur nécessitant $4n_z$ opérations pour un stockage morse symétrique [57], *i.e.* $72N$ opérations en 2D et $192N$ en 3D,

4 produits scalaires (trois pour la résolution plus un pour le critère d'arrêt) de coût: $4 \times 2N$ opérations en 2D, et $4 \times 3N$ en 3D,

2 étapes de préconditionnement dont le coût est détaillé ci-après.

Le coût d'une itération de MINRES non préconditionné est donc: $\begin{cases} 240\ Nbnoe & \text{en 2D} \\ 816\ Nbnoe & \text{en 3D} \end{cases}$

3.2.2.3 Coûts de préconditionnement

Préconditionneurs blocs diagonaux: Leur construction nécessite d'assembler sur chaque élément du maillage les blocs nodaux diagonaux pour ensuite les inverser. Une estimation du nombre d'opérations nécessaires est présenté dans le tableau suivant (pour plus de détails nous renvoyons à *Marie* [57]):

	2D	3D
Assemblage	$12\ Nbelt$	$28\ Nbelt$
Calcul de M^{-1}	$9\ Nbnoe$	$30\ Nbnoe$
Précond. $M^{-1}x$	$2 \times Nbit \times 9\ Nbnoe$	$2 \times Nbit \times 16\ Nbnoe$
Total	$33\ Nbnoe + 18\ Nbnoe \times Nbit$	$170\ Nbnoe + 32 Nbnoe \times\ Nbit$

TAB. 3 – *Coût d'assemblage et par itérations pour le préconditionnement bloc diagonal* **VipP** *en 2D et en 3D. Nbit est le nombre d'itérations nécessaires à MINRES pour converger*

Factorisations incomplètes

Construction: Le préconditionnement par factorisations incomplètes nécessite de construire une factorisation de la matrice de la forme LDL^T avec L triangulaire inférieure. On peut approcher le nombre d'opérations nécessaires pour effectuer cette construction en s'inspirant de la formule (26) et en remplaçant la largeur de bande lbg par $\overline{v}$.

On estime la construction du préconditionneur **Ic** à : $\dfrac{N \times ((d+1)\bar{v})^2}{2}$

On utilise la même formule dans le cas du préconditionneur défini positif **IcP** mais appliqué seulement à la matrice bloc des vitesses et en ajoutant la construction et l'inversion de la diagonale en pression.

Pour la factorisation incomplète modifiée *i.e.* **MicP**, on doit en outre considérer le calcul, *sur chaque ligne*, des termes non stockés devant renforcer la diagonale.

On estime donc le coût de construction (toujours en s'inspirant de (26)) à : $\dfrac{d^3 Nbnoe \times \bar{v} \times lbg}{2}$

Préconditionnement: A chaque étape, on résout des problèmes de la forme:

$$Mz \equiv LDL^T z = y$$

Cette résolution est effectuée en 3 étapes:

$$\begin{array}{ll} 1/ & w = L^{-1} y \\ 2/ & x = D^{-1} w \\ 3/ & z = L^{-T} x \end{array}$$

Le coût de l'opération de descente remontée dans le cas d'une factorisation incomplète est égal à celui d'un produit matrice/vecteur (étapes 1/ et 3/) plus une division par un vecteur (étape 2/). Toutefois, en raison de l'opération de remontée, nécessitant de prendre les pointeurs à l'envers, il serait plus juste de considérer ce coût comme 1,5 fois celui d'un produit matrice/vecteur.

Une synthèse des coûts de préconditionnement est présentée au tableau suivant:

	2D	3D
Ic	$486\ Nbnoe + 2 \times 327\ Nbnoe \times Nbit$	$4608\ Nbnoe + 2 \times 1156\ Nbnoe \times Nbit$
IcP	$145\ Nbnoe + 2 \times 147\ Nbnoe \times Nbit$	$1945\ Nbnoe + 2 \times 652\ Nbnoe \times Nbit$
MicP	$24\ Nbnoe^{1.5} + 2 \times 147\ Nbnoe \times Nbit$	$162\ Nbnoe^{1.66} + 2 \times 652\ Nbnoe \times Nbit$

TAB. 4 – *Complexité du préconditionnement par factorisations incomplètes; construction et coût par itérations.*

On remarque donc que le coût d'une itération **IcP** est de l'ordre de huit fois celui d'une itération **Vipp**. Il faut donc que ce préconditionneur entraîne au moins huit fois moins d'itérations pour être compétitif. Finalement, il reste à estimer le comportement asymptotique de MINRES selon le préconditionneur utilisé.

3.2.3 Taux de convergence de MINRES

3.2.3.1 Taux de convergence théorique

Le comportement asymptotique de MINRES pour des problèmes de Stokes stabilisés à été étudié par *A. Wathen et D. Silvester* [83, 84]. Le taux de convergence dépend de la

distribution des valeurs propres du système préconditionné (15), qui est dissymétrique par rapport à l'origine, et de la forme:

$$[-a;-b\alpha]\cup\left[c\alpha^2;d\right]$$

avec $a,b,c,d>0$ et $\alpha>0$ petit.

Les études menées par les auteurs sur la distribution de ces valeurs propres pour les préconditionneurs symétriques définis positifs ont abouti aux estimations suivantes du taux de convergence (*i.e.* le nombre d'itérations nécessaires à la convergence de la méthode):

- Préconditionneurs diagonaux et blocs diagonaux **VipP, DiagP**:

$$Nbit\sim\mathcal{O}\left(h^{-3/2}\right)\quad i.e.\begin{cases} Nbit\sim\mathcal{O}\left(Nbnoe^{0.75}\right) & \text{en 2D}\\ Nbit\sim\mathcal{O}\left(Nbnoe^{0.5}\right) & \text{en 3D}\end{cases}$$

  Ce taux de convergence a été numériquement vérifié sur une formulation 3D identique à la notre par *T. Coupez et S. Marie* [21].

- Préconditionneurs par factorisations incomplètes modifiées **MicP**:

$$Nbit\sim\mathcal{O}\left(h^{-3/4}\right)\quad i.e.\begin{cases} Nbit\sim\mathcal{O}\left(Nbnoe^{0.375}\right) & \text{en 2D}\\ Nbit\sim\mathcal{O}\left(Nbnoe^{0.25}\right) & \text{en 3D}\end{cases}$$

  Comme nous le verrons au paragraphe suivant ce taux de convergence n'est pas retrouvé en pratique.

Aucune estimation n'est connue (voire possible) pour les préconditionneurs symétriques non définis positifs.

3.2.3.2 Estimations numériques du comportement asymptotique

Nous présentons dans le tableau 5 le nombre d'itérations nécessaires pour les différentes méthodes en fonction du nombre de noeuds des maillages. La courbe correspondante est tracée sur la figure 2.

$Nbnoe$	n_z	VipP	IcP	Ic	MicP
1517	48450	1235	384	128	357
3017	96294	1968	530	194	499
5757	186714	3248	834	306	738
10042	324969	4917	1258	435	1243
13722	446409	6220	1513	460	1415
19660	642327	7951	1913	669	1819
31639	1037616	11797	2605	949	(3632)

	VipP	IcP	Ic	MicP
Vitesse de convergence asymptotique (p)	0.75	0.65	0.65	0.65
Constantes (C)	4.5	3.	0.7	2.75

TAB. 5 – *Comportement asymptotique de MINRES en 2D pour des maillages non déformés et selon le préconditionneur utilisé. Nombre d'itérations nécessaires pour inverser un système linéaire en fonction du nombre de noeuds du maillage. Interpolation de* $Nbit = C\,Nbnoe^p$.

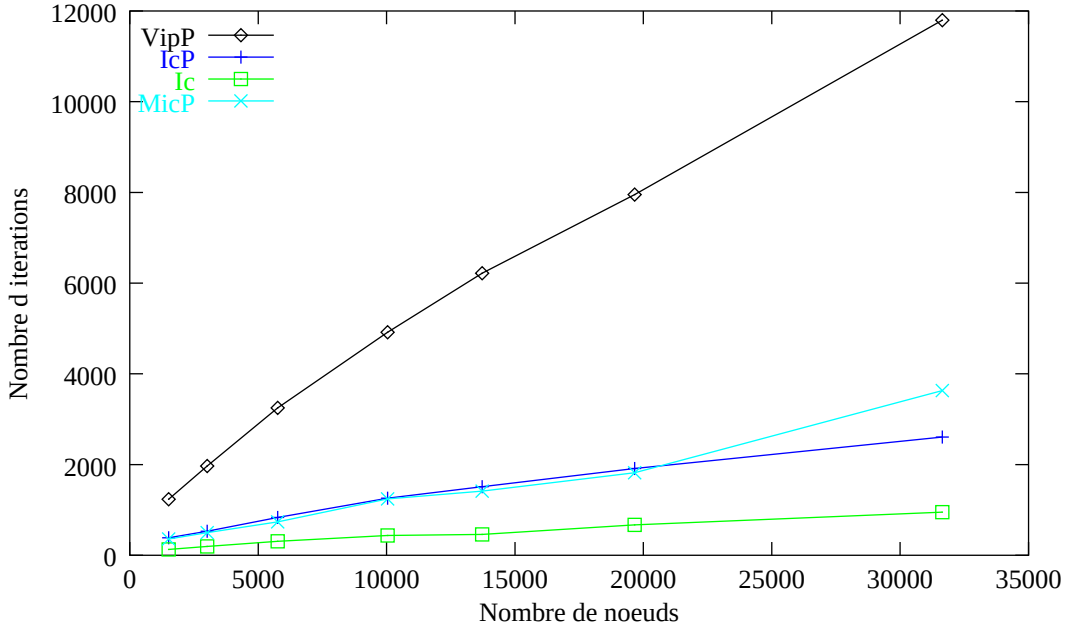


FIG. 2 – *Comportement asymptotique des préconditionneurs pour des maillages non déformés. Nombre d'itérations nécessaires pour inverser un système linéaire en fonction du nombre de noeuds du maillage.*

En 3D, *K. Mocellin* [58] a mesuré un nombre d'itérations en $\mathcal{O}\left(Nbnoe^{0,34}\right)$ pour les préconditionneurs par factorisations incomplètes. Cette mesure montre que le comportement du solveur est plus proche de ce que l'on attend en 3D. Nous ne disposons malheureusement pas d'une estimation des constantes.

3.2.3.3 Synthèse :

Nous présentons dans le tableau 6 une synthèse des estimations effectuées en 2D (en tenant compte des résultats du tableau 5) et à une constante près de celles en 3D. Il apparaît

	2D	3D
Direct	$18\ Nbnoe^{1.5} + 13.5\ Nbnoe^{2}$	$32\ Nbnoe^{1.66} + 32\ Nbnoe^{2.33}$
VipP	$34\ Nbnoe + 4.5 \times 276\ Nbnoe^{1.75}$	$171\ Nbnoe + 880 \times \mathcal{O}\left(Nbnoe^{1.5}\right)$
Ic	$486\ Nbnoe + 0.7 \times 678\ Nbnoe^{1.65}$	$4608\ Nbnoe + 2360 \times \mathcal{O}\left(Nbnoe^{1.34}\right)$
IcP	$145\ Nbnoe + 3 \times 438\ Nbnoe^{1.65}$	$1945\ Nbnoe + 1688\ Nbnoe \times \mathcal{O}\left(Nbnoe^{1.34}\right)$
MicP	$24\ Nbnoe^{1.5} + 2.75 \times 438\ Nbnoe^{1.65}$	$162\ Nbnoe^{1.66} + 1688\ Nbnoe \times \mathcal{O}\left(Nbnoe^{1.34}\right)$

TAB. 6 – *Coût total en nombre d'opérations du solveur direct et de MINRES pour les divers préconditionneurs en 2D et en 3D*

dans ce tableau et clairement sur la figure 3 que le passage d'un solveur direct à un solveur itératif est bien plus favorable en 3D qu'en 2D. En effet, pour les préconditionneurs diagonaux, nous passons en 3D d'un $\mathcal{O}\left(N^{2.33}\right)$ à $\mathcal{O}\left(N^{1.5}\right)$ tandis qu'en 2D nous passons d'un $\mathcal{O}\left(N^{2}\right)$ à $\mathcal{O}\left(N^{1.75}\right)$. On note aussi qu'en terme de complexité, la méthode itérative avec préconditionneur diagonal reste beaucoup plus coûteuse que la méthode directe et, devient

compétitive avec le préconditionneur **Ic** pour des maillages supérieurs à 25000 noeuds. En pratique, la limite se situe plutôt autour de 15000 noeuds en raison d'effets de cache beaucoup plus favorables à un stockage compact.

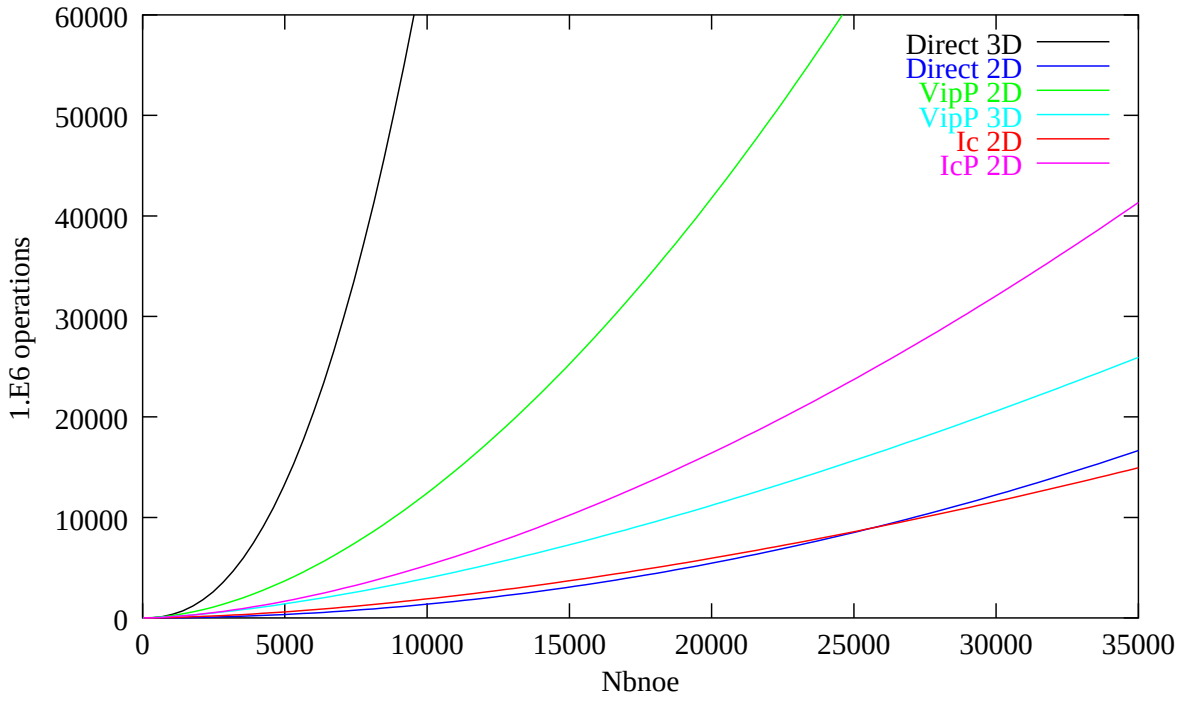


FIG. 3 – *Nombre d'opérations asymptotiques en 2D et 3D selon le solveur utilisé. Pour le préconditionneur* **VipP** *en 3D, la constante à été choisie en première approximation égale à celle déterminée en 2D*

3.3 Premiers résultats numériques

Nous nous plaçons dans le cadre d'un matériau newtonien ($m = 1$); le cas viscoplastique sera principalement abordé au paragraphe 4.

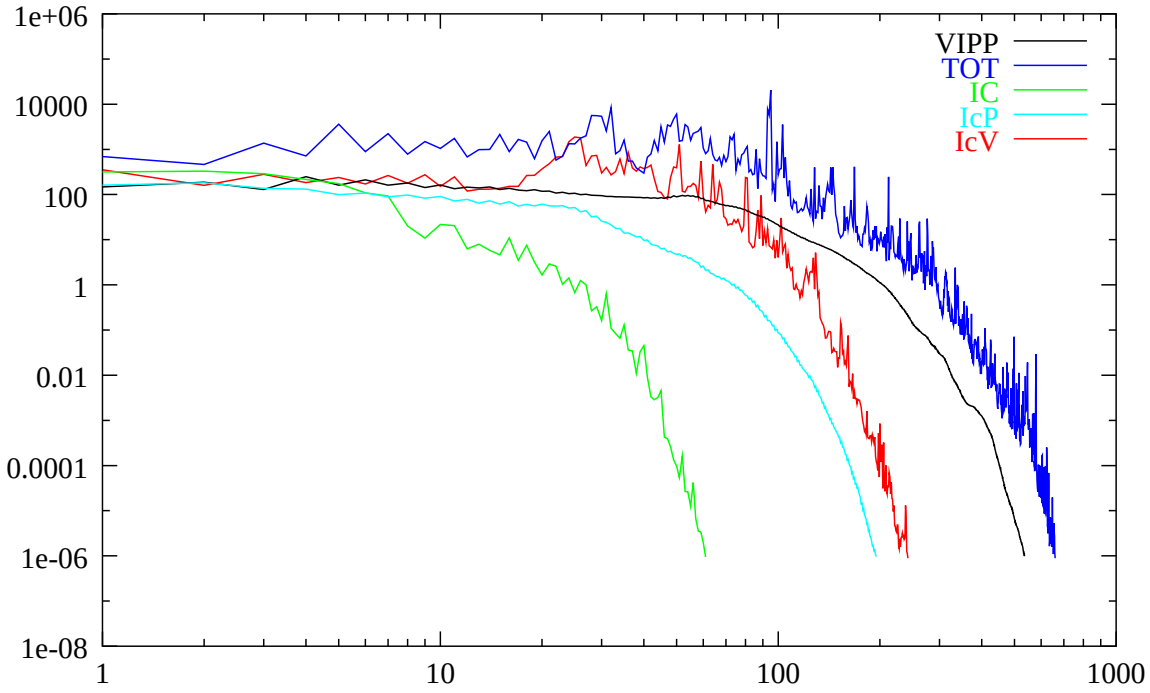
Comparaisons des préconditionneurs:

Considérons un problème de petite taille, 625 noeuds, relatif à un filage arrière loin de l'axe de symétrie et similaire au problème présenté au chapitre précédent (cf. figure 4 page 21). Le critère d'arrêt imposé aux solveurs itératifs (méthode hybride dans le cas des préconditionneurs symétriques définis positifs et méthode du résidu minimal sinon) est $||r_k|| \leq 1.10^{-6}\,||r_0||$. On reporte dans le tableau (a) figure 4 le nombre d'itérations ayant été nécessaire à la convergence des solveurs. En terme de nombre d'opérations, et aussi de temps de calcul (*cf.* paragraphe suivant), les factorisations incomplètes donnent les meilleurs résultats. Le préconditionneur symétrique non défini positif **Ic** est le plus efficace.

La figure 4(b) illustre la décroissance chaotique du résidu quand les préconditionneurs sont non défini positif. On remarque que le comportement de **Ic** est moins perturbé que celui des deux autres préconditionneurs non définis positifs, mais cela est seulement dû à un effet d'échelle. Le comportement observé n'est pas sans rappeler celui des méthodes de gradient conjugué carré (CGS) ou de bigradient conjugué et bigradient conjugué stabilisé (BiCG et BiCG-stab) pour la résolution de systèmes linéaires non symétriques. Des techniques de lis-

Préconditionneur	Diag	DiagP	Vip	VipP	Tot	IC	IcV	IcP	MicP
nombre d'itérations	825	545	775	536	660	61	242	195	175

(a) Nombre d'itérations de MINRES selon le préconditionneur utilisé



(b) Évolution de l'erreur relative au cours des itérations de MINRES selon que le préconditionneur est symétrique défini positif ou non

FIG. 4 – *Comparaison des préconditionneurs et de leur comportement pour le cas de filage arrière*

sage (smoothing) sont alors pratiquées et il pourrait être envisageable de nous en inspirer par exemple en suivant les travaux de *H. Vorst et G. Sleijpen* [73] ou de *L. Zhou et al.* [86].

On notera que pour les préconditionneurs diagonaux et blocs diagonaux, on observe généralement que les préconditionneurs symétrique défini positif **DiagP** et **VipP** sont les plus efficaces. Cela n'est plus vrai si l'on modifie la structure de la matrice en ajoutant des termes diagonaux de fort poids pour le traitement du contact par une méthode de pénalisation. Il a alors été observé que les préconditionneurs **Vip** et **Tot** sont les plus performants [57].

Temps de calcul :

On reprend le problème de filage arrière pour trois maillages ayant entre 1581 noeuds et 5757 noeuds. Les temps de calcul mesurés pour l'assemblage des préconditionneurs et l'inversion de trois systèmes linéaires successifs sont présentés dans le tableau 7. Il apparaît que quelle que soit la taille des problèmes traités, les préconditionneurs diagonaux ne sont pas compétitifs par rapport au solveur direct. Les factorisations incomplètes améliorent significativement les performances de la méthode itérative sans être compétitives pour moins de 6000 noeuds.

Nbnoe	VipP	MicP	IcP	Ic	Direct
1581	22 s	15 s	15 s	6 s	2 s
3017	58 s	40 s	41 s	18 s	7 s
5757	234 s	152 s	145 s	68 s	44 s
Accélération par rapport à **VipP**	1	1.54	1.61	3.44	5.31

TAB. 7 – *Comparaisons de temps de calcul entre méthode itérative préconditionnée et méthode directe pour l'inversion de trois systèmes linéaires consécutifs*

Conclusions:

Le taux de convergence théorique est parfaitement retrouvé pour le préconditionneur bloc diagonal (il en est de même pour le préconditionneur diagonal): 0.75.

Les factorisations incomplètes "simples" sont les meilleurs préconditionneurs avec un ordre de convergence de 0.65 sur des maillages non structurés. Les résultats obtenus pour les factorisations incomplètes sont très éloignés des estimations théoriques trouvées dans *Wathen et al.* [84] à savoir 0.375. Sur des maillages structurés, avec un outil plat et des conditions de contact collant (Dirichlet) et de surface libre, nous obtenons encore les mêmes taux de convergence. Nous n'avons pas, sur ce point, d'explications particulières, mais pouvons nous interroger sur l'influence de la formulation axisymétrique sur la distribution des valeurs propres dont sont issues les estimations théoriques.

Le préconditionneur par factorisation incomplète modifiée **MicP** s'avère décevant. Celui-ci nécessite un nombre d'itérations comparable à celui du préconditionneur **IcP**. Son taux de convergence est quasi identique si l'on considère que la dernière ligne du tableau 5 est un accident. Il peut s'avérer intéressant lorsque **Ic** n'est pas utilisable.

Les mêmes pentes sont obtenues pour les préconditionneurs **IcP** et **Ic** mais avec un facteur trois de différence sur la constante. Au niveau du temps de calcul cette différence est plutôt de l'ordre d'un facteur deux, car la descente remontée effectuée à chaque étape de préconditionnement porte seulement sur le bloc des vitesses, dans le cas de **IcP**, et non sur la matrice entière comme pour **Ic**.

Des études similaires ont été effectuées pour d'autre type d'écoulements et pour des rhéologies non linéaires. Les tendances observées sont les mêmes que celles présentées ici et **le préconditionneur Ic apparaît donc comme le meilleur** quoique ses performances soient encore insuffisantes par rapport à la méthode directe pour des maillages de moins de 6000 noeuds.

Comme nous venons de le voir, le travail mené sur le préconditionnement par factorisations incomplètes a grandement amélioré les performances de la méthode du résidu minimal, en réduisant les temps de calcul d'un facteur trois et en améliorant sa vitesse de convergence. Cela reste toutefois insuffisant. Envisager des solveurs encore plus performants est une première perspective qui nous mènerait naturellement aux méthodes multigrilles dont la complexité théorique est en $\mathcal{O}(N\,logN)$. Cette direction, explorée par *K. Mocellin* [58] dans sa thèse a donné d'assez bons résultats en 3D. Toutefois, leurs mise en place est très délicate en raison des nombreux remaillages devant être effectués au cours d'une simulation. De plus même si asymptotiquement elles sont très performantes, leur coût important par itérations les rend intéressantes seulement pour des problèmes de grosse taille. Les autres pistes pouvant être suivies pour améliorer les performances du solveur itératif font l'objet du paragraphe suivant.

4 Solveur Non-linéaire et optimisations du temps de calcul

4.1 Paramètre de stabilisation et nombre d'itérations

Comme nous l'avons vu au chapitre I , plusieurs formulations éléments finis sont envisageables; les plus attrayantes sont les formulations simplifiées et simplifiées stabilisées. Pour ces dernières, nous sommes parvenus à la conclusion qu'un paramètre de stabilisation $\beta = 3$ assurait une bonne précision des calculs. L'idée ici est de comparer les différentes formulations du point de vue du nombre d'itérations et du CPU et non de la précision (elles sont relativement équivalentes) qui a déjà été abordée. Cette étude s'inspire des travaux de *Wathen et al.* [83, 84, 82] qui ont déjà observé l'influence importante du paramètre de stabilisation β^2 sur le comportement de MINRES. Comme une valeur trop élevée de ce paramètre peut toutefois dégrader la précision de la formulation (*cf.* paragraphe 5) nous ne nous intéresserons pas à des valeurs de $\beta > 5$.

Exemple pour une rhéologie newtonienne :

Nous reprenons le cas de filage arrière précédent (cf. tableau 7) et regardons l'influence de la formulation éléments finis sur les performances de MINRES avec le préconditionneur **Ic**.

$Nbnoe$	P1+P1 simplifié	P1+P1 stabilisé $\beta = 3$	P1+P1 stabilisé $\beta = 5$	Direct P1+P1 simplifié
1581	295 (6s)	226 (5 s)	236 (5s)	2 s
3017	437 (18 s)	266 (13 s)	273 (13 s)	7 s
5757	630 (68 s)	501 (51 s)	530 (54 s)	44 s

TAB. 8 – *Nombre d'itérations et temps de calcul pour inverser trois systèmes linéaires consécutifs pour un problème de filage newtonien. Influence du paramètre de stabilisation β sur la résolution itérative par MINRES préconditionnée par* **Ic**

Sur cet exemple, il apparaît clairement que l'introduction d'un paramètre de stabilisation améliore nettement le conditionnement des systèmes linéaires, et permet une diminution de l'ordre de 25 à 30 % du nombre d'itérations de MINRES.

Exemple pour une rhéologie viscoplastique :

Toujours pour le même problème de filage arrière, nous nous plaçons maintenant dans le cadre d'une rhéologie non-newtonienne avec $m = p = 0.11$. Les résultats présentés au tableau 9 montrent encore l'influence du paramètre de stabilisation sur les temps de calcul à la fois sur le nombre moyen d'itérations de MINRES ainsi que sur le nombre d'itérations du solveur non-linéaire. La valeur $\beta = 5$, à l'inverse des résultats précédents, donne les meilleures performances. On observe en général que plus le paramètre est élevé plus la convergence de MINRES est rapide. Cela est d'autant plus vrai que l'écoulement est complexe. Ce phénomène est lié à l'augmentation du poids des termes bloc diagonaux en pression qui rend les problèmes linéaires et non-linéaires moins raides à résoudre.

$Nbnoe$	P1+P1 simplifié	P1+P1 stabilisé $\beta = 3$	P1+P1 stabilisé $\beta = 5$	Direct P1+P1 simplifié
1581	43 s (2271 it.) 21 (2.04 s)	41 s (1837 it.) 28 (1.46 s)	39 s (1782 it.) 27 (1.44 s)	29 s 21 (1.38 s)
3017	465 s(11609 it.) 47 (9.89 s)	203 s (5102 it.) 41 (4.95 s)	133 s (3258 it.) 32 (4.15 s)	181 s 47 (3.85 s)
5757	960 s (13191 it.) 48 (20. s)	533 s (6775 it.) 41 (13. s)	474 s (5919 it.) 39 (12.1 s)	897 s 48 (18.6 s)

TAB. 9 – *Influence du paramètre de stabilisation sur la résolution itérative préconditionnée par* **Ic** *pour effectuer 5 incréments du problème de filage viscoplastique. Première ligne: Temps de calcul (nombre total d'itérations de MINRES). Deuxième ligne: nombre total d'itérations de Newton-Raphson (temps moyen par itérations de Newton)*

Conclusions:

1. Comme constaté par d'autres auteurs, les formulations stabilisées sont plus faciles à résoudre tant en linéaire qu'en non-linéaire.
2. Le cadre non-linéaire rend les méthodes itératives plus compétitives par rapport au solveur direct que dans le cadre linéaire.
3. Sur le problème non-linéaire, on constate également une diminution du nombre d'itérations de Newton-Raphson. On explique cela par le fait que l'ajout du terme de stabilisation rend le problème moins raide à résoudre. C'est particulièrement vrai dans le cas étudié, représentatif d'une situation de fin de forgeage, où l'écoulement est très confiné (souvent la partie de la simulation la plus gourmande en CPU). Dans d'autres cas, quand l'écoulement n'est pas confiné, on ne constate pas toujours une diminution du nombre d'itérations non-linéaires.
4. Les formulations stabilisées sont globalement meilleures tant du point de vue de la précision que du temps CPU.
5. Sur cet exemple, le solveur itératif devient compétitif par rapport au solveur direct pour un maillage d'environ 3000 noeuds.
6. $\beta = 3$ offre un bon compromis entre précision des calculs et performance de la résolution itérative. C'est la solution que l'on retient pour la suite.

4.2 Optimisation des opérations élémentaires du solveur itératif

La résolution itérative met en jeu deux types d'opérations très gourmandes en temps de calcul: les produits matrices/vecteurs et les descentes/remontées. Une implémentation imprudente des routines associées à ces opérations peut très vite s'avérer désastreuse. On pourra trouver une discussion sur le sujet dans le cas d'un gradient conjugué dans *M. Field* [35]. L'idée de base est de tenir compte dans l'écriture des programmes de la manière dont fonctionnent les processeurs utilisés, par exemple les RS10000 sur architecture IBM.

Nous ne détaillerons pas l'optimisation du produit matrice/vecteur que nous avons implanté et qui est présentée dans [35]. Nous avons procédé au même type d'optimisations pour traiter la descente-remontée. Pour la présenter, il est nécessaire de décrire le stockage morse utilisé. Celui-ci consiste à ne stocker que les termes non nuls de la matrice et à adresser les termes de ce tableau à l'aide de deux pointeurs: un pour les lignes noté pl (de taille $N+1$) qui indique l'adresse du premier terme non nul de la ligne dans le pointeur colonne pc (de taille

n_z) donnant le numéro de la colonne correspondante à chaque terme de la matrice. Ainsi sur l'exemple de la matrice 4×4 suivante, on aurait en stockant la partie triangulaire inférieure:

$$A = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 2 & -4 & 0 & 1 \\ 0 & 0 & 3 & 0 \\ 0 & 1 & 0 & 4 \end{pmatrix}$$

A(1:6)	1	2	-4	3	1	4

pc(1:6)	1	1	2	3	2	4

pl(0:4)	0	1	3	4	6

La ligne i possédant $pl(i+1) - pl(i)$ termes et $A(pl(i))$ est le terme diagonal.

Une implémentation immédiate de la résolution de $x = \left(LDL^T\right)^{-1} b$ est:

```
 DO i = 1, N                              DESCENTE de  L x = B
   S = 0
   DO j = pl(i-1) + 1, pl(i) - 1
      k = pc(j)
      S  = S  + L(j) * X(k)
   ENDDO
   X(i) = B(i) - S
END DO

DO i = 1, N                               DIAGONALE de D  x = x
   X(i) = X(i) * D(i)
END DO

DO i = 1, N
   Baux(i) = 0.D0
ENDDO
                                                        t
DO i = N, 1, -1                           REMONTE  de  L  x = x
   X(i) = X(i) - Baux(i)

   DO j = pl(i-1) + 1, pl(i)- 1
      k = pc(j)
      Baux(k) = Baux(k) + L(j) * X(i)
   ENDDO
ENDDO
```

Un point décisif pour l'optimisation de notre routine réside dans l'utilisation de la mémoire cache. Il y a trois niveaux d'accès à la mémoire: le cache, qui est une zone mémoire de petite taille directement connectée au processeur, la mémoire vive, qui est la zone mémoire courante, et le "swap" *i.e.* les accès disque, qui est le pire mode d'accès. Le temps nécessaire pour accéder à des données situées dans le cache est de l'ordre d'un cycle d'horloge, tandis que si elles ne sont pas dans le cache, une douzaine de cycle sont nécessaires pour retrouver les données dans la mémoire vive. Il faut donc utiliser au mieux la mémoire cache et cela peut être effectué en tenant compte de notre structure matricielle faite de blocs 3×3. Il nous suffit alors de récrire les procédures non plus lignes par lignes, mais blocs par blocs. Nous présentons la première partie de la routine optimisée, *i.e.* la descente.

```
c                  ~
c   DESCENTE de  L x = B
c
      DO i = 1, N, 3
         Sa = 0 ; Sb = 0 ; Sc = 0.D0

         pt1a = pl(i-1) ; pt2a = pl(i) - 1
         pt1b= pt2a+1   ; pt2b= pl(i+1) -1
         pt1c= pt2b+1   ; pt2c= pl(i+2) -1

         b1 = B(i) ; b2 = B(i+1) ; b3= B(i+2)

         DO k = 1, pt2a-pt1a, 3
            col = pc(pt1a+k)
            x1 = X(col) ;  x2 = X(col+1) ;  x3 = X(col+2)

            Sa = Sa + L(pt1a+k  ) * X1 ; Sb = Sb + L(pt1b+k  ) * X1
            Sa = Sa + L(pt1a+k+1) * X2 ; Sb = Sb + L(pt1b+k+1) * X2
            Sa = Sa + L(pt1a+k+2) * X3 ; Sb = Sb + L(pt1b+k+2) * X3

            Sc = Sc + L(pt1c+k  ) * X1
            Sc = Sc + L(pt1c+k+1) * X2
            Sc = Sc + L(pt1c+k+2) * X3
         ENDDO
         auxa  = b1 - Sa ; Sb= Sb + L(pt2b) * auxa ;  auxb= b2 - Sb

         Sc= Sc + L(pt2c-1) * auxa ; Sc= Sc + L(pt2c)*auxb ; auxc= b3 - Sc

         X(i)   = auxa
         X(i+1) = auxb
         X(i+2) = auxc
      END DO
```

On présente, dans le tableau 10, les temps de calcul mesurés pour la résolution de trois systèmes linéaires successifs avec ou sans optimisation des routines de produit matrice/vecteur et de descente/remontée. Pour 3000 noeuds, le gain obtenu est supérieur à 20 % du temps de calcul et celui-ci augmente avec la taille des systèmes linéaires.

maillages	Ic $\beta = 3$ non optimisé	Ic $\beta = 3$ optimisé	Direct
1581	5 s	4 s	2 s
3017	13 s	9 s	7 s
5757	51 s	36 s	44 s

TAB. 10 – *Temps de calcul nécessaire pour inverser 3 systèmes linéaires successifs avec des sous-programmes optimisés ou non*

4.3 Résolutions inexactes et solveur itératif

Les optimisations présentées précédemment n'utilisent pas le fait que la résolution soit elle-même incluse dans un algorithme itératif. On peut exploiter cette résolution extérieure par Newton-Raphson en indexant la précision demandée au solveur itératif interne (linéaire) sur la précision courante de la résolution externe (non-linéaire).

En effet, durant les premières itérations du solveur non-linéaire, le résidu courant peut être très grand. Il est alors inutile de résoudre très précisément le système linéaire. Par exemple, si le résidu est de l'ordre de 1000, calculer une correction à une précision de 1.10^{-6} est superflu.

Méthode de Newton-Raphson :

L'algorithme pour la résolution d'un problème non linéaire

$$R(v) = 0$$

s'écrit:

Algorithme II .2 Méthode de Newton-Raphson

$k = 0$, v^0 fixé

$r^0 = ||R(v^0)||$

Tant que $\dfrac{r^k}{r^0} \geq \epsilon_{NR}$ répète:

Calcul de la matrice tangente

Assemblage de $H \equiv \dfrac{\partial R(v^k)}{\partial v}$

Assemblage du préconditionneur P

Résolution par MINRES algorithme II .1 :

On cherche Δv solution de : $H\Delta V = -R(v^k)$ avec une précision ϵ_{it} sur $\frac{||r_k||}{||r_0||}$

Recherche linéaire :

On cherche α minimisant $||R(v^k + \alpha\Delta v)||$

Réactualisation :

$v^{k+1} = v^k + \alpha\Delta v$

$k = k + 1$

fin Tant que

fin algorithme

Nous choisissons en général $\epsilon_{NR} = 1.10^{-5}$ *i.e.* une réduction du résidu initial d'un facteur cent mille.
Pour obtenir cette précision, à la convergence, il faut $\epsilon_{it} < \epsilon_{NR}$ que l'on impose en choisissant $\epsilon_{it} = 0.1\epsilon_{NR}$, de manière à calculer la correction avec une précision suffisante lorsque l'on est proche de la solution. Par contre, lorsque l'on est éloigné de la solution, on effectue alors des **résolutions inexactes**, en ajustant ϵ_{it} selon la formule:

$$\epsilon_{it} = \max\left(\epsilon_{NR} * 0.1, \frac{r^k}{r^0} * \epsilon_{inexact}\right)$$

Nous avons constaté numériquement que pour des valeurs de $\epsilon_{inexact}$ supérieures à 1.10^{-2}, nous n'étions plus assurés de la convergence de Newton-Raphson. On choisit donc généralement $1.10^{-2} \leq \epsilon_{inexact} \leq 1.10^{-4}$. Pour cet intervalle de valeurs nous obtenons systématiquement la convergence de Newton-Raphson et donc la même solution, à la précision ϵ_{NR} près, que pour les résolutions "exactes". Le choix de la valeur optimale est délicat et, les expérimentations numériques que nous avons menées ont montré qu'elle dépendait du type d'écoulement ainsi que du maillage. Par la suite, nous utiliserons toujours la valeur $\epsilon_{inexact} = 1.10^{-2}$. Pour de plus amples détails sur ces techniques nous renvoyons, par exemple, à *Coupez* [21] ou à *Issman* [53].

Remarques:

- Il est important de noter que les résolutions non linéaires inexactes ne sont possibles qu'avec des solveurs itératifs et ne puissent être envisagées avec un solveur direct.
- On pourrait aussi envisager de ne pas construire H et P à chaque fois (méthode de Quasi-Newton), mais l'expérience à montré en 3D que l'on ne gagnait guère en temps de calcul et que l'on perdait en robustesse. La raison principale est que les événements de contacts peuvent avoir une forte influence et modifier de manière significative les termes de la matrice.

Résultats :

On présente dans le tableau 11 les temps de calcul mesurés lors des cinq premiers incréments de la simulation du filage arrière pour un matériau viscoplastique ($m = p = 0.1$). Le choix de ces cinq incréments a été motivé par le fait (rare) que, dans tous les cas, le même nombre de systèmes ont été inversés. On désigne par OPT 1 les calculs effectués après optimisation des routines les plus coûteuses (*cf.* 4.2) et par OPT 2 les calculs effectués avec à la fois les résolutions inexactes et l'optimisation des routines. Le calcul non-optimisé est effectué avec $\epsilon_{it} = 1.10^{-6}$. Les temps de calcul sont donnés pour une formulation P1+P1 simplifiée stabilisée avec $\beta = 3$. Entre parenthèses sont donnés les temps pour une formulation simplifiée non stabilisée ($\beta = 1$). Nous présentons dans le tableau 12 un second jeu de

		P1+P1 simplifié $\beta = 3$ (entre parenthèses $\beta = 1$)			
maillages Nbnoe	Nombre total d'inversions	Solveur direct	Non Optimisé	OPT 1	OPT 2
3676	56	510 s	888 s (1805 s)	660 s (1426 s)	260 s (412 s)
6006	41	665 s	1620 s	1248 s	366 s

TAB. 11 – *Synthèse des optimisations effectuées. Temps de calcul pour 5 incréments du problème de filage non-linéaire, pour les différents niveaux d'optimisation: stabilisation, opérations élémentaires, solveur non-linéaire.*

résultats pour un problème non linéaire issu d'un cas fourni par l'industriel FORD WERKE. On trouvera une description de ce cas test au chapitre V page 116. La rhéologie est viscoplastique $m = p = 0.141$, le coefficient de frottement est $\alpha = 0.2$ et la consistance est $K = 3.464.10^6$ KPa. Nous donnons les mesures effectuées après 10 itérations de Newton. Ce cas test est complémentaire et intéressant car, en début de procédé, l'écoulement est moins favorable aux formulations stabilisées car les possibilités d'écoulement sont nombreuses (*i.e.* le problème n'est pas très raide du point de vue de l'écoulement aussi la stabilisation est-elle moins efficace).

	P1+P1 simplifié				P1+P1 stabilisé $\beta = 3$		
nbnoe	Solveur Direct	Non optimisé	OPT 1	OPT 2	Non optimisé	OPT 1	OPT 2
1789	82 s 62 (1.3 s)	258 s (15313 it.) 62 (4.2 s)	196 s (15681 it.) 62 (3.2 s)	164 s (12542 it.) 62 (2.6 s)	185 s (10683 it.) 68 (2.7 s)	144 s (10726 it.) 68 (2.1 s)	101 s (6923 it.) 64 (1.6 s)
3591	261 s 53 (4.9 s)	838 s (28787 it.) 53 (15.8 s)	603 s (22626 it.) 53 (11.4 s)	472 s (17532 it.) 53 (8.9 s)	514 s (14424 it.) 57 (9. s)	404 s (14531 it.) 57 (7.1 s)	268 s (8845 it.) 57 (4.7 s)
5616	712 s 53 (13.4 s)	2041 s (38406 it.) 53 (38.5 s)	1481 s (32717 it.) 53 (27.9 s)	1031 s (22351 it.) 52 (19.8 s)	1242 s (20147 it.) 57 (21.8 s)	942 s (20246 it.) 57 (16.5 s)	589 s (11680 it.) 55 (10.7 s)

TAB. 12 – *Temps de calcul en secondes pour le problème de forgeage de FORD WERKE. Première ligne: Temps de calcul. Deuxième ligne: (nombre total d'itérations de MINRES). Troisième ligne: nombre total d'itérations de Newton-Raphson (temps moyen par itérations de Newton)*

Le tableau 11 montre que pour ce cas test (favorable), la combinaison de toutes les optimisations permet d'avoir un solveur compétitif pour moins de 3000 noeuds. Le gain le plus important est donné par l'optimisation du solveur non-linéaire.
Le second problème considéré est moins raide et donc les optimisations, mis à part celles des routines élémentaires, sont moins efficaces (tableau 12). Les meilleurs gains sont obtenus par l'introduction du paramètre de stabilisation. Le solveur est compétitif pour des maillages de l'ordre de 3500 noeuds, et dès 1800 noeuds les résultats sont comparables.

4.4 Fiabilisation du solveur

Les nombreux tests et cas industriels étudiés ont montré que le préconditionneur **Ic** est robuste. Il stagne parfois mais plutôt exceptionnellement (de l'ordre de 1% des inversions) et, très rarement diverge. Cela se produit généralement quand un élément est proche de la dégénérescence ou encore juste après un remaillage.

Plusieurs stratégies pour fiabiliser le solveur ont été mises en places. La méthode la plus simple consiste à effectuer une reprise à partir de la meilleure solution obtenue, en recommençant les calculs avec cette fois un préconditionneur bloc diagonal symétrique défini positif **VipP**.

Dans le cas où il y aurait stagnation et où le résidu aurait été obtenu avec une précision inférieure à la précision courante de Newton-Raphson, une seconde stratégie consiste à passer à l'itérée suivante de Newton-Raphson. Cette méthode donne de bons résultats quoiqu'un peu dangereuse car susceptible d'entraîner une divergence de Newton-Raphson.

Enfin, pour améliorer la robustesse nous avons tenté d'introduire une méthode de continuation sur le paramètre de stabilisation β^2. On démarre la résolution avec $\beta \geq 5$, puis au fur et à mesure de la convergence de Newton-Raphson, on fait décroître cette valeur. Cette méthode permet d'accélérer les calculs quand la surface libre est peu importante. Par contre, si l'écoulement est peu contraint elle mène à une augmentation significative du nombre d'itérations de Newton-Raphson et perd son intérêt.

La figure 5 montre le comportement de MINRES avec le préconditionneur **Ic** au cours

d'une simulation complète du cas FORD. Elle a été effectuée sur un maillage de 1781 noeuds, pour une formulation P1+P1 simplifiée en utilisant une méthode de continuation de $\beta = 3$ à $\beta = 1$ à chaque résolution non-linéaire. Cette simulation demande environ 10 heures de calcul et nécessite quelques 89 remaillages. MINRES préconditionnée par **Ic** n'a pas convergé à seulement 3 reprises, sur un total d'environ 12500 inversions et a dépassé 1000 itérations à seulement 6 reprises. Ces phénomènes sont chaque fois liés soit à un événement de contact soit à un remaillage. Pour le même problème, mais cette fois avec une formulation simplifiée stabilisée ($\beta = 3$), le temps de calcul chute à 3h30 (3h00 pour le solveur direct) sans qu'aucun accident dans la convergence de MINRES ne soit à déplorer.

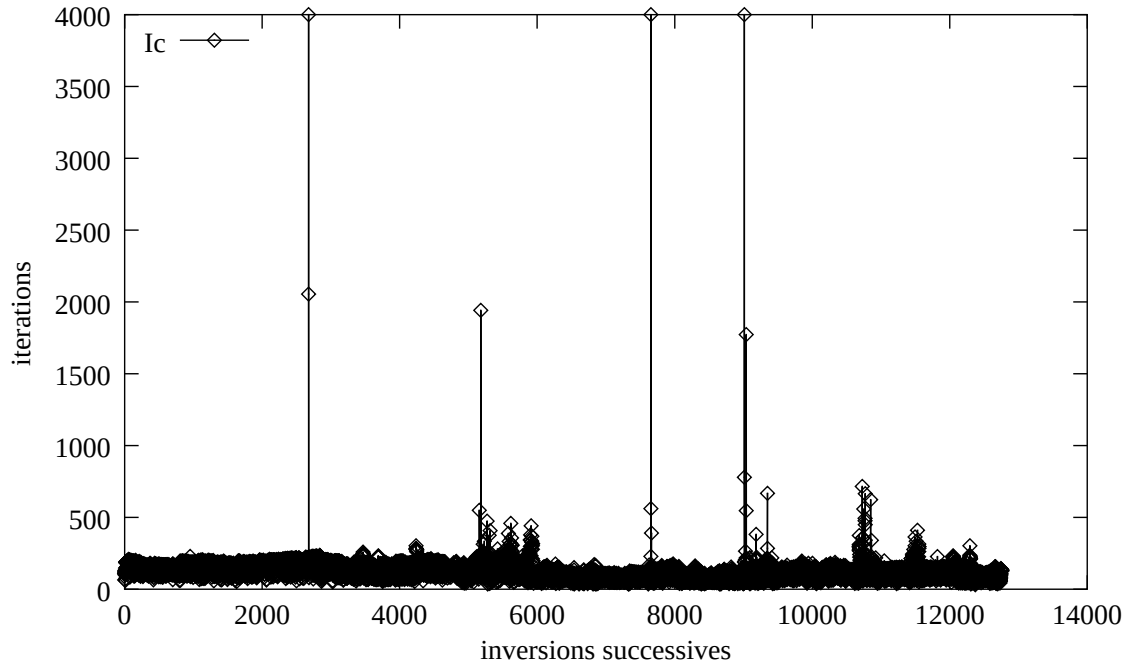


FIG. 5 – *Itérations de MINRES préconditionnée par* **Ic** *au cours de la simulation du cas FORD sur un maillage de 1781 noeuds. (Nombre maximum d'itérations = 4000)*

5 Conclusions

Dans ce chapitre, nous avons présenté les diverses possibilités pour résoudre itérativement les systèmes linéaires issus de la discrétisation du problème de Stokes. Le choix de la méthode du résidu minimal, déjà employé en 3D, nous assure de pouvoir développer un code parallèle efficace. Toutefois les spécificités des problèmes 2D font que le solveur itératif est moins performant qu'en 3D. Notre premier effort a donc consisté à développer un solveur séquentiel performant. Cela est passé par l'introduction de préconditionneurs par factorisations incomplètes, bien plus efficaces que les préconditionneurs diagonaux et blocs diagonaux précédemment utilisés. Nous avons conclu que les versions symétriques non définies positives de ces préconditionneurs étaient les plus efficaces: d'un facteur trois en nombre d'itérations et d'un facteur deux en temps de calcul. Ce sont donc ces versions non définies positives qui sont désormais utilisées dans notre code de calcul. Nous renvoyons au dernier chapitre V consacré aux applications industrielles pour d'autres illustrations des gains obtenus.

L'introduction de ces préconditionneurs est un point clé mais il ne suffit pas pour obtenir

un solveur compétitif par rapport aux méthodes directes. L'utilisation de la formulation éléments finis simplifiée stabilisée a aussi une grande importance, permettant de diminuer de près de 30 % le nombre d'itérations nécessaires à la convergence sans pour autant détériorer la qualité des solutions. Nous sommes parvenus à la conclusion qu'un paramètre de stabilisation $\beta = 3$ était un bon compromis entre précision et temps de calcul. Enfin, une écriture optimisée des algorithmes les plus coûteux à encore permis d'améliorer les performances de la méthode de résolution.

Le denier point et le plus crucial, a été le couplage du solveur itératif à la résolution non-linéaire par Newton-Raphson. L'ajustement de la précision du solveur en fonction de la distance à la solution du problème non-linéaire a permis de réduire encore fortement le nombre total d'itération et les temps de calcul. Nous sommes finalement parvenus à un solveur itératif capable de concurrencer les méthodes directes, et ce pour des maillages de taille moyenne (de l'ordre de 3000 noeuds) et pour des maillages de l'ordre de 1500 noeuds dans certains cas favorables. Enfin, une dernière amélioration, que nous n'avons pas effectué, concerne la gestion du contact. *S. Marie* de la société TRANSVALOR a montré qu'un traitement du contact par une méthode de pénalisation améliorait significativement les temps de calcul.

Nous avons donc obtenu un solveur itératif compétitif en séquentiel. Ces résultats nous permettent donc d'espérer pouvoir développer un solveur parallèle plus rapide que le solveur direct séquentiel, même si les préconditionneurs par factorisations incomplètes sont plus difficiles à paralléliser.

Chapitre III

Le solveur parallèle

Il existe une très grande variété de calculateurs parallèles avec des architectures pouvant être fondamentalement différentes. Le choix et l'implémentation d'un algorithme parallèle est fortement lié au type de machine sur lequel celui-ci va s'exécuter. D'excellents algorithmes pour une architecture donnée peuvent s'avérer peu efficaces sur d'autres. Il est donc essentiel de choisir des algorithmes dépendant le moins possible des architectures avant d'envisager la parallélisation d'un code de calcul. La connaissance de ces architectures, de leurs spécificités ainsi que des contraintes liées à celles-ci, est une condition préalable à toute implémentation. La portabilité, c'est à dire la capacité d'un logiciel à s'exécuter sur différentes machines avec peu de changements, est une nécessité pour tout code commercial aussi les outils utilisés pour sa parallélisation doivent être, autant que possible, indépendants des architectures.

1 Généralités sur le parallélisme :

1.1 Le Hardware :

L'entité élémentaire d'une machine parallèle, appelée **noeud**, est constituée d'une unité de calcul(scalaire ou vectorielle), d'une unité de communication et d'une mémoire. Ces noeuds sont reliés entre eux par un réseau d'interconnections qui conditionne les communications entre les processeurs. Une machine parallèle peut se définir à l'aide de trois caractéristiques: la granularité, la topologie et le mode de contrôle des données.

1. **Granularité :**
   La granularité est le rapport du nombre de processeurs sur la puissance de chaque processeur.

   On parle ainsi de **granularité fine**, ou encore de **parallélisme massif**, pour des architectures possédant un très grand nombre de processeurs ayant chacun une faible puissance de calcul. Sur ce type de machines, la parallélisation d'un code éléments finis se situe au niveau des éléments voire des degrés de liberté.
   Citons à titre d'exemple les Connection Machine CM5 (512 processeurs) et CM2 (16^3 à 16^4 processeurs !), le Parangon XP/S ou encore l'Intel iPSC-860/128.

   Inversement on parle de **granularité grossière (coarse)** pour des machines possédant un petit nombre de processeurs avec chacun une forte puissance de calcul. Le parallélisme se situe dans ce cas au niveau des sous-domaines.
   Des exemples de ce type de machine sont les CRAY-2/4, T3E, X-MP, ou encore l'Alliant FX-8.

2. **Topologie :**
   La topologie définit les connexions entre les noeuds et reflète les flux de données entre les processeurs.
   On rencontre divers types de topologies: en anneaux, en crossbar (tous les processeurs sont connectés entre eux), en hypercubes ... L'organisation de la mémoire est intrinsèquement liée à la topologie. Celle-ci peut être **partagée** si tous les processeurs partagent une mémoire commune, ou **distribuée** si chaque processeur dispose d'une mémoire locale.
   Il existe également des machines à mémoire **hiérarchique**, où la mémoire est partagée mais où chaque processeur dispose en plus d'une mémoire locale. On peut toutefois considérer que la majorité des calculateurs à mémoire partagée ont une mémoire hiérarchique, puisque chaque processeur possède une mémoire cache qui lui est propre.
   Pour des raisons techniques, les machines à mémoire partagée sont généralement des machines à grains grossiers. En effet, au-dessus d'un petit nombre de processeurs les accès mémoire sont susceptibles d'engendrer une saturation du bus. En pratique, ces machines sont aussi plus faciles à programmer, mais les meilleures performances, en nombre d'opérations par secondes ("Flops"), sont en général obtenues sur des machines massivement parallèles à mémoire distribuée.
   Dans le cadre de calculs issus d'une discrétisation par une méthode éléments finis, le choix d'une structure de données adaptée à la topologie est crucial [31].
3. **Contrôle :**
   Le contrôle décrit la manière dont les processeurs contrôlent les données et comment le travail est divisé et synchronisé. Celui-ci peut s'effectuer de deux manières :

   S.I.M.D. : (Single Instruction flow Multiple Data flow) **un seul et unique programme** s'exécute sur la machine parallèle. Chaque processeur effectue le même travail sur des données différentes. Les processeurs sont **synchrones**.
   Ces machines sont généralement à grains fins. La Connection Machine CM2 est une machine SIMD à mémoire distribuée.

   M.I.M.D. : (Multiple Instruction flow Multiple Data flow) **des copies séparées** du programme s'exécutent sur chaque processeur qui effectue un travail différent sur des données différentes. Les processeurs sont **asynchrones**.
   Ce sont généralement des machines à grains grossiers ou moyens. Les Parangon XP/S, IBM SP2, CM5 sont des machines MIMD à mémoire distribuée tandis que les CRAY X-MP, SUN Enterprise 4000 ou encore la majorité des PC multi processeurs sont des machines MIMD à mémoire partagée.

1.2 Stratégies de programmation et cadre de notre travail

Nous nous plaçons dans le cadre de machines MIMD à granularité grossière et à mémoire partagée.

Ce choix est essentiellement motivé par la nécessité de développer un code de calcul utilisable par de nombreux industriels et pour un investissement informatique relativement faible. À l'heure actuelle, les plateformes parallèles les moins onéreuses sont les PC multiprocesseurs, qui sont des machines MIMD à granularité grossière et à mémoire partagée. Il est toutefois obligatoire pour un code commercial d'être portable sur un maximum d'architectures différentes. Aussi, nous veillerons à n'utiliser que des méthodes et des outils pouvant s'étendre à des machines à mémoire distribuée. On distingue deux types de parallélisme, chacun étant associé à un type de programmation: le parallélisme de données ("data parallel") et le paral-

lélisme de contrôle, selon qu'il s'agisse respectivement d'extraire du parallélisme en exécutant une suite d'opérations parallèles ou bien d'exécuter en parallèle des séquences d'instructions. Les langages à parallélisme de données se rencontrent traditionnellement -mais non exclusivement- sur les machines SIMD à grains fins. Nous renvoyons le lecteur à l'article de *S. L. Johnsson et K. K. Mathur* [55] sur l'implémentation d'une méthode éléments finis sur la CM2 et sur la structure de données associée, ainsi qu'aux publications très claires de *A. Basermann et al.* [9, 8] sur l'implémentation de solveurs dissymétriques sur un Parangon XP/S. Pour une présentation complète et une bibliographie conséquente sur la programmation "data parallel", nous renvoyons à la thèse de *C. Weill-Duflos*(1994) [85]. Ce mode de programmation n'est pas très attractif pour nous car il est peu portable.

La tendance actuelle, dans le cadre de méthodes éléments finis implicites, repose sur un paradigme de partitionnement de maillage / "message-passing" (*C. Farhat et F. X. Roux* [34]). La programmation par "message-passing" ou par échange de messages a été initialement développée pour des architectures MIMD à mémoire distribuée. C'est le programmeur qui prévoit la distribution des données et les communications entre les processeurs. Cette approche a l'avantage de renforcer la localité des données, et est donc adaptée aux architectures parallèles en général. Il existe plusieurs librairies permettant d'effectuer le "message-passing", les plus connues étant PVM (Parallel Virtual Machine) et MPI (Message Passing Interface). On notera toutefois que le "message-passing" est moins un outil de programmation qu'un modèle de programmation. Il peut tout à fait être envisagé sur une machine à mémoire partagée, sans que des messages soient explicitement échangés, mais par exemple comme un simple accès à une zone mémoire partagée (*cf.* [34]).

Le choix d'un modèle de programmation par "message-passing" nous permet donc d'implémenter une méthode parallèle aisément portable sur de nombreuses architectures.

1.3 Modèles de mesures de performances

La qualité d'une implémentation parallèle est généralement donnée par la mesure de l'accélération (ou speed-up) et de l'efficacité atteinte.
Notons $T(N,p)$ le temps d'exécution d'un algorithme parallèle sur p processeurs pour un problème de taille N. L'accélération d'un programme parallèle s'exécutant sur p processeurs est le rapport du temps d'exécution du programme parallèle sur un processeur, sur le temps d'exécution du programme parallèle sur p processeurs:

$$\textbf{Accélération :} \qquad Sp\,(N,p) = \frac{T\,(N,1)}{T\,(N,p)}$$

L'efficacité étant le rapport de l'accélération sur le nombre de processeurs utilisés:

$$\textbf{Efficacité :} \qquad E\,(N,p) = \frac{Sp\,(N,p)}{p}$$

Nous dirons d'un algorithme qu'il est **scalable** si il a la capacité de conserver une bonne efficacité parallèle lorsque le nombre de processeurs augmente.

Ces deux mesures indiquent seulement la qualité de l'implémentation parallèle, sans mesurer le véritable gain occasionné par l'exécution parallèle. En effet, la parallélisation d'un logiciel conduit souvent à développer des algorithmes spécifiques qui ne sont pas nécessairement aussi performants en séquentiel que les algorithmes originaux. Nous introduisons donc pour la suite l'accélération absolue $\widetilde{Sp}\,(N,p)$ comme le rapport du meilleur temps séquentiel

sur le temps parallèle.

$$\widetilde{Sp}\,(N,p) = \frac{T_{seq}\,(N)}{T\,(N,p)}$$

Un algorithme parfaitement parallèle présenterait une accélération égale au nombre de processeurs utilisés. En pratique cela n'est (presque) jamais le cas. On peut recenser cinq causes majeures de perte d'efficacité parallèle.

1. La **fraction séquentielle** est la partie non parallélisable (ou non parallélisée) de l'algorithme. On peut décomposer le temps de calcul de tout code parallèle en la somme d'une partie séquentielle incompressible et d'une partie parallélisable, *i.e.*

   $$T(N,p) = t^{seq} + \frac{t^{//}}{p}$$

   Son influence est mise en évidence par la *loi d'Amdhal*. Cette loi établit que l'accélération maximale pouvant être atteinte par une machine parallèle exécutant un algorithme sur p processeurs est limitée par la fraction $\alpha \leq 1$ de temps de calcul devant être effectué en séquentiel:

   $$T(N,p) = \alpha + \frac{1-\alpha}{p} \Longrightarrow Sp\,(N,p) = \frac{1}{\alpha + \frac{1-\alpha}{p}} \leq \frac{1}{\alpha}$$

2. Le **déséquilibre des charges** peut être la plus grande cause d'inefficacité. Le temps d'exécution d'un algorithme parallèle est limité par le temps d'exécution du processeur ayant la charge de calcul la plus importante. Dès que la charge de travail est mal répartie entre les processeurs, certains se retrouvent "désoeuvrés" (idle) tandis que les autres continuent de calculer. Un équilibre des charges de calcul *tout au long* de la simulation est indispensable à l'efficacité parallèle.
3. Les **communications** entre processeurs contribuent en totalité à la diminution de l'efficacité de l'algorithme. Nous détaillerons ce point important au paragraphe suivant.
4. Les **contraintes algorithmiques** sont de deux types:
   L'algorithme séquentiel optimal peut être impossible à paralléliser tel quel. Sa parallélisation implique alors de le modifier en un algorithme moins performant.
   De manière à éviter de trop nombreuses communications, il est courant d'effectuer plusieurs fois le même calcul sur plusieurs processeurs plutôt que de le faire sur un seul processeur qui redistribue ensuite les résultats aux autres. Même s'ils sont moins coûteux que des communications, ces *calcul redondants* contribuent en totalité à la perte d'efficacité.
5. **L'implémentation parallèle** induit des surcoûts par rapport au code séquentiel: procédures et opérations additionnelles, utilisation de registres et tableaux supplémentaires, partitionnement du problème, gestion des interfaces, etc.

Nous sommes ici principalement concernés par les quatre derniers points. Nous veillerons ainsi à mettre en place des algorithmes qui se parallélisent bien, et à effectuer une implémentation minimisant les communications et les opérations additionnelles.

1.3.1 Coûts de communications

Nous nous plaçons dans le cadre de machines MIMD, chaque processeur ayant sa propre mémoire (virtuelle ou physique) interconnectée aux autres par un réseau de communication. Chaque fois qu'un processeur a besoin de données localisées sur un autre noeud, un surcoût est occasionné pour deux raisons:

Synchronisations:
Il est possible que les données nécessaires ne soient pas disponibles. Les processeurs doivent alors attendre d'être synchronisés avant de pouvoir effectuer le transfert des données. Les tâches doivent donc être équilibrées, de façon à minimiser ces périodes où un processeur attend. Ainsi dans le cadre d'une décomposition du maillage en sous-domaines, la construction d'une partition équilibrée est cruciale.

Transfert des données :
Le modèle classique de temps de transfert d'un message de N octets d'un processeur à un autre est:

$$t_{comm} = t_{latence} + t_{pack} + N\, t_{debit} \quad (+ (d-1)\, t_h\ )$$

Où:

$t_{latence}$ est la latence *i.e.* le temps d'initialisation de la communication,
t_{pack} le temps de chargement du message à transmettre dans une mémoire tampon
t_{debit} le temps de transmission d'un octet.

Dans le cadre de machines à mémoire distribuée, si deux processeurs sont à d connexions l'un de l'autre, on doit en plus considérer le temps "per-hop" t_h, caractérisant le délai nécessaire à un message pour passer d'un noeud à son voisin immédiat. Dans le cas d'une mémoire partagée, on a $d = 1$.

En raison de la latence et du chargement, on a tout intérêt à envoyer un message long plutôt que plusieurs messages courts.
Notons que la latence et le débit dépendent des caractéristiques de la machine, du réseau de communication et du "message passing" employé.

2 Les méthodes itératives parallèles

C'est un domaine très vaste ayant fait l'objet de très nombreuses recherches ces quinze dernières années. Nous ne pourrons ici qu'effectuer un bref survol du sujet et nous conseillons vivement la lecture de trois articles de synthèse, tous parus en 1994, et relatifs aux méthodes parallèles pour des schémas éléments finis implicites. *T. F. Chan et T. P. Mathew* [15] effectuent une revue des méthodes de Schwarz et du complément de Schur primal. Ils présentent le cadre abstrait avec une synthèse des principaux résultats relatifs à la convergence de ces algorithmes. *C. Farhat et F.X. Roux* [34] présentent une monographie très complète et orientée sur la mise en oeuvre de ces méthodes dans le cadre de la mécanique des structures, mais dont le fond reste général aux problèmes elliptiques. Les modèles de programmation parallèle et les solveurs directs parallèles sont présentés et discutés. La méthode du complément de Schur primale y est décrite et, de manière plus détaillée, la méthode duale FETI dont ils sont les initiateurs. Ils montrent les principaux résultats obtenus avec ces méthodes, avec de nombreuses comparaisons numériques sans, que les détails pratiques de l'implémentation ne soient perdus de vue. La monographie de *P. Le Tallec* [77] décrit de manière très détaillée des méthodes primales et fournit de nombreux résultats avec leurs démonstrations.

Pour paralléliser un grand code de calcul sur des machines à grains grossiers ou moyens,

une stratégie de type **Diviser pour régner** ("**Divide and Conquer**") est généralement recommandée [31, 34]. Le maillage est partitionné en sous-domaines, alloués chacun à un processeur. Ceux-ci effectuent les calculs éléments finis locaux à leur domaine, de manière concurrentielle et sans la moindre communication. La globalité des solutions est assurée par les techniques de résolution du ou des systèmes linéaires. Les algorithmes utilisés à cette fin peuvent s'envisager selon deux philosophies :

Une approche locale:

Nous parlerons alors de **méthode de décomposition de domaine**.
Cette stratégie consiste à diviser le problème global en plusieurs sous-problèmes locaux indépendants, pouvant être résolus en parallèle, puis à recombiner les solutions locales via la résolution itérative d'un problème condensé aux interfaces des sous-domaines. Ce problème est mieux conditionné que le problème global, ce qui favorise, voire rend possible, une résolution itérative. À chaque itération, des problèmes locaux indépendants sont résolus de manière directe ou itérative.

Une approche globale:

Nous qualifierons cette approche de **méthode de partitionnement de domaine**.
Celle-ci vise à résoudre directement, à l'aide de tous les processeurs, le système linéaire global. Cela n'est possible que si le système linéaire global est suffisamment bien conditionné pour être directement résolu par une méthode itérative. Cette méthode a l'avantage de présenter une très grande scalabilité, une grande simplicité d'implémentation et de ne pas introduire de résolution spécifique pour le problème parallèle. Dans le cas d'un préconditionneur parfaitement parallélisable, comme le préconditionnement diagonal, elle ne dépend pas du nombre de domaines.

Avant de décrire ces différentes approches, présentons d'abord les notations utilisées dans la suite de cet exposé.

Notations

Domaine :

Ω	le domaine supposé borné,
$Nbelt$	le nombre d'éléments de la triangulation de Ω,
$Nbnoe$	le nombre de noeuds associés,
h	une taille de maille caractéristique
N	le nombre de degrés de liberté associés au problème

Sous-domaines :

Np	le nombre de sous-domaines (et de processeurs),
Ω_i	le ième sous-domaine, $\Omega = \bigcup_{i=1}^{i=Np} \Omega_i$
H	un diamètre caractéristique des sous-domaines,
$Nbnoe_i$	le nombre de noeuds du domaine i,
N_i	le nombre de degrés de liberté associés,

Interfaces :

S_{ij} l'interface entre les domaines i et j: $S_{ij} = \Omega_i \cap \Omega_j, i \neq j$

S_i l'interface du domaine i: $S_i = \bigcup_{j=1}^{j=Np} S_{ij}$

S l'interface "totale": $S = \bigcup_{i,j=1;i<j}^{i,j=Np} S_{ij}$

On suppose, par souci de simplicité de la présentation, qu'en numérotation globale les noeuds de l'interface S sont numérotés en dernier. De même, en numérotation locale, on suppose que les noeuds des interfaces S_i $i = 1, \cdots N_p$ sont aussi numérotés en derniers.

Opérateurs :

R_i l'opérateur de restriction au sous domaine Ω_i,
R_i^T l'opérateur d'extension du sous domaine Ω_i à Ω
$\widehat{R_i}$ l'opérateur de restriction à l'interface de Ω_i,
$\widehat{R_i^T}$ l'opérateur d'extension de S_i à Ω

Vecteurs:

x un vecteur global assemblé sur le domaine Ω,
x_i la restriction de x à Ω_i,
x_{Ω_i} le vecteur de $\mathbb{R}^{N_i}$ des contributions de Ω_i au vecteur x, en numérotation locale,
$\widehat{x_i}$ le vecteur de $\mathbb{R}^N$ des contribution de Ω_i au vecteur x, complété par des zéro

Matrices:

$\mathcal{A}$ une matrice globale assemblée sur le domaine Ω,

$$\mathcal{A} = \begin{pmatrix} A_{11} & 0 & \cdots & A_{1S} \\ 0 & A_{22} & (0) & A_{2S} \\ \vdots & (0) & \ddots & \vdots \\ A_{1S}^T & A_{2S}^T & \cdots & A_{SS} \end{pmatrix}$$

$\mathcal{A}_{\Omega_i} = \begin{pmatrix} A_{ii} & A_{iS} \\ A_{Si} & A_{SS}^{\Omega_i} \end{pmatrix}$ la contribution de Ω_i à la matrice $\mathcal{A}$ en numérotation locale

$\mathcal{A}_i = \begin{pmatrix} A_{ii} & A_{iS} \\ A_{Si} & A_{SS}^{i} \end{pmatrix}$ la restriction à Ω_i de la matrice $\mathcal{A}$ en numérotation locale

$\widehat{\mathcal{A}}_{\Omega_i} = R_i^T \mathcal{A}_{\Omega_i} R_i$ la contribution de Ω_i à $\mathcal{A}$ en numérotation globale

$A_{SS}^{\Omega_i}$ la contribution du sous-domaine à la sous-matrice d'interface A_{SS},
A_{SS}^{i} la restriction à S_i de la sous-matrice d'interface.

On a alors les formules:

$$\begin{array}{rclcrcl} \mathcal{A}_i & = & R_i \mathcal{A} R_i^T & & x_i & = & R_i x \\ \mathcal{A} & = & \sum_{i=1}^{Np} \widehat{\mathcal{A}}_{\Omega_i} = \sum_{i=1}^{Np} R_i^T \mathcal{A}_{\Omega_i} R_i & & x & = & \sum_{i=1}^{N_p} R_i^T x_{\Omega_i} = \sum_{i=1}^{N_p} \widehat{x_i} \end{array} \quad (1)$$

Exemple:
Dans le cas de 2 domaines nous avons :

$$\begin{aligned}\mathcal{A} &= \begin{pmatrix} A_{11} & 0 & A_{1S} \\ 0 & A_{22} & A_{2S} \\ A_{S1} & A_{S2} & A_{SS} \end{pmatrix} \text{ avec } A_{SS} = A_{SS}^{\Omega_1} + A_{SS}^{\Omega_2} \\ &= R_1^T \begin{pmatrix} A_{11} & A_{1S} \\ A_{S1} & A_{SS}^{\Omega_1} \end{pmatrix} R_1 + R_2^T \begin{pmatrix} A_{22} & A_{2S} \\ A_{S2} & A_{SS}^{\Omega_2} \end{pmatrix} R_2 \end{aligned} \tag{2}$$

Où :

$$R_1 = \begin{pmatrix} I & 0 & 0 \\ 0 & 0 & I \end{pmatrix} \text{ et } R_2 = \begin{pmatrix} 0 & I & 0 \\ 0 & 0 & I \end{pmatrix}$$

2.1 Les méthodes de décomposition de domaine primales

Historiquement, la première méthode de décomposition de domaine a été développée par *H. A. Schwarz* en 1870 pour prouver des résultats d'existence sur des domaines complexes. Il avait alors partitionné le domaine global en une réunion de sous-domaines simples se recouvrant, et développé une méthode itérative permettant de calculer la solution globale à partir des solutions locales. Avec l'émergence du calcul parallèle, cette méthode a été redécouverte et appliquée avec succès à de nombreux problèmes. Dans le même temps, d'autres méthodes présentant moins de difficultés techniques et basées sur des décompositions sans recouvrement sont apparues, et se sont généralement imposées. Notons que l'utilisation de ces méthodes pour des problèmes séquentiels peut s'avérer également plus efficace (pour un petit nombre de domaines) car elles permettent de réduire la complexité des calculs.

Nous allons présenter, les trois grands types de méthodes de décomposition de domaines, dans le cas simple d'une décomposition en deux sous-domaines. Nous renvoyons à la bibliographie citée pour les généralisations, généralement non triviales, à un grand nombre de sous-domaines.
Nous illustrerons cette présentation à partir du problème suivant:

On cherche $u \in \mathcal{V}$ solution du problème :

$$\left\{ \begin{array}{rcl} Lu \equiv -\text{div}\left(2\eta\dot{\varepsilon}\left(u\right)\right) & = f & \text{dans } \Omega \\ u & = 0 & \text{sur } \partial\Omega \end{array} \right. \tag{3}$$

dont la discrétisation aboutit au système linéaire:

$$\mathcal{A}u = b \tag{4}$$

2.1.1 La méthode alternative de Schwarz :

Cette méthode opère des recouvrements de taille δ (*cf.* figure 1) entre les sous-domaines. Elle vise à résoudre le problème global en découplant les problèmes locaux et en effectuant un point fixe sur le recouvrement.
Soit respectivement u_1 et u_2 les vecteurs d'inconnues nodales sur les sous-domaines respectifs Ω_1 et Ω_2 et soit u^k la kième itérée du vecteur solution.
On pose:

$$u^0 = \left\{ \begin{array}{ll} u_1^0 & \text{sur } \Omega_1 \\ u_2^0 & \text{sur } \Omega \setminus \Omega_1 \end{array} \right. \qquad \text{avec } u^0 \equiv 0 \text{ sur } \partial\Omega$$

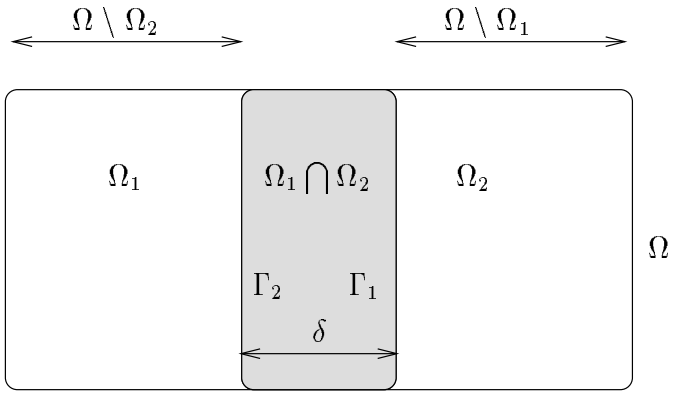


FIG. 1 – *Décomposition en deux sous-domaines avec recouvrements*

Supposant u^k connu, on construit une séquence d'approximations de la solution en résolvant d'abord sur Ω_1 :

$$\left\{\begin{array}{lll} Lu_1^{k+1} = f & \text{dans } \Omega_1 & \\ u_1^{k+1} = 0 & \text{sur } \partial\Omega_1 \setminus \Gamma_1 & \text{Alors :} \quad u^{k+1/2} = \left\{\begin{array}{ll} u_1^{k+1} & \text{dans } \Omega_1 \\ u^k & \text{dans } \Omega \setminus \Omega_1 \end{array}\right. \\ u_1^{k+1} = u_{|\Gamma_1}^k & \text{sur } \Gamma_1 \quad (CL1) & \end{array}\right. \tag{5}$$

puis sur Ω_2 :

$$\left\{\begin{array}{lll} Lu_2^{k+1} = f & \text{dans } \Omega_2 & \\ u_2^{k+1} = 0 & \text{sur } \partial\Omega_2 \setminus \Gamma_2 & \text{Et :} \quad u^{k+1} = \left\{\begin{array}{ll} u_2^{k+1} & \text{dans } \Omega_2 \\ u^{k+1/2} & \text{dans } \Omega \setminus \Omega_2 \end{array}\right. \\ u_2^{k+1} = u^{k+1/2} & \text{sur } \Gamma_2 \quad (CL2) & \end{array}\right. \tag{6}$$

Ce schéma correspond à une **méthode multiplicative de Schwarz**. Le problème est résolu à chaque demi-itération sur seulement un sous-domaine, sur lequel on impose la valeur de la demi-itérée précédente à l'interface avec l'autre sous-domaine. Le problème de cette méthode est qu'elle est peu scalable puisque les deux résolutions ne sont pas indépendantes. Sa parallélisation peut toutefois s'envisager en utilisant des techniques de coloriage.

Une variante importante, et beaucoup plus adaptée au parallélisme, consiste à prendre comme condition limite la valeur de l'itérée précédente. Cela revient à remplacer la condition limite (CL2) du second problème par u^k:

$$u_2^{k+1} = u^k \equiv u_2^k \qquad \text{sur } \Gamma_2 \tag{7}$$

On obtient alors une **méthode additive de Schwarz**.

Formulation discrète :
L'expression discrète de ces algorithmes mène aux schémas itératifs suivant [15, 77]:

Méthode multiplicative de Schwarz :

$$\left\{\begin{array}{l} u^{k+1/2} = u^k - R_1^T A_1^{-1} R_1 (b - Au^k) \\ u^{k+1} = u^{k+1/2} - R_2^T A_2^{-1} R_2 (b - Au^{k+1/2}) \end{array}\right. \tag{8}$$

qui conduit au point fixe:

$$\left(u^{k+1} - u\right) = \left(I - R_2^T A_2^{-1} R_2 A\right)\left(I - R_1^T A_1^{-1} R_1 A\right)\left(u^k - u\right)$$

On montre que cela correspond à une généralisation de la méthode de Gauss-Seidel (*cf.* par exemple *Le Tallec* [77]).

Méthode additive de Schwarz :

$$\begin{cases} u^{k+1/2} = u^k - R_1^T A_1^{-1} R_1 (b - Au^k) \\ u^{k+1} = u^{k+1/2} - R_2^T A_2^{-1} R_2 (b - Au^k) \end{cases} \tag{9}$$

qui équivaut au point fixe:

$$\left(u^{k+1} - u\right) = \left(I - R_2^T A_2^{-1} R_2 A - R_1^T A_1^{-1} R_1 A\right) \left(u^k - u\right)$$

On montre que cette méthode équivaut à une méthode de Jacobi par blocs.

Ces méthodes étant des points fixes, on montre que leur convergence est géométrique [15], ce qui signifie qu'elle est au plus linéaire. Il est relativement aisé de les accélérer en utilisant, par exemple, dans le cas symétrique, un gradient conjugué préconditionné.

Pour la suite, on définit le préconditionneur additif de Schwarz comme:

$$P_{as}^{-1} = \sum_{i=1}^{Np} R_i^T A_i^{-1} R_i \tag{10}$$

On montre alors (*T.F. Chan et T.P. Mathew* [15] ou *P. Le Tallec* [77]) que, pour des opérateurs elliptiques, le conditionnement de la méthode additive est :

$$\kappa\left(P_{as}^{-1}\mathcal{A}\right) = \mathcal{O}\left(H^{-2}\left(1 + \left(\frac{H}{\delta}\right)^2\right)\right) \tag{11}$$

Nous voyons donc que le conditionnement est indépendant de la taille de maille et que, pour une taille suffisante de recouvrement, il est en $\mathcal{O}\left(H^{-2}\right)$. Cela signifie que, pour un grand nombre de sous-domaines, (*c.àd.* lorsque $H \to 0$) la méthode risque de se détériorer.

2.1.2 Méthodes du complément de Schur:

Les méthodes du complément de Schur sont aussi appelées méthodes primales par opposition aux méthodes de complément de Schur duales. Dans la pratique, il existe de très nombreuses formulations du problème condensé aux interfaces, selon le type de conditions limites imposées aux frontières des problèmes locaux.

On considère la résolution du problème modèle (3) avec une partition de Ω en deux sous-domaines ne se recouvrant pas (*cf.* figure 2).

$$\Omega = \Omega_1 \cup \Omega_2 \quad \text{et} \quad \Omega_1 \cap \Omega_2 = S$$

Soit: $u = (u_1, u_2, u_S)$ la solution du problème global où $u_i,\ i = 1, 2$ et u_S sont respectivement les restrictions de u à $\Omega_i,\ i = 1, 2$ et à S.

u est solution du problème (3) si et seulement si :

$$u_i \text{ est solution de } \begin{cases} Lu_i = f & \text{dans} \quad \Omega_i \\ u_i = 0 & \text{sur} \quad \partial\Omega_i \setminus S \quad \text{pour } i = 1, 2 \\ u_i = u_S & \text{sur} \quad S \end{cases} \tag{12}$$

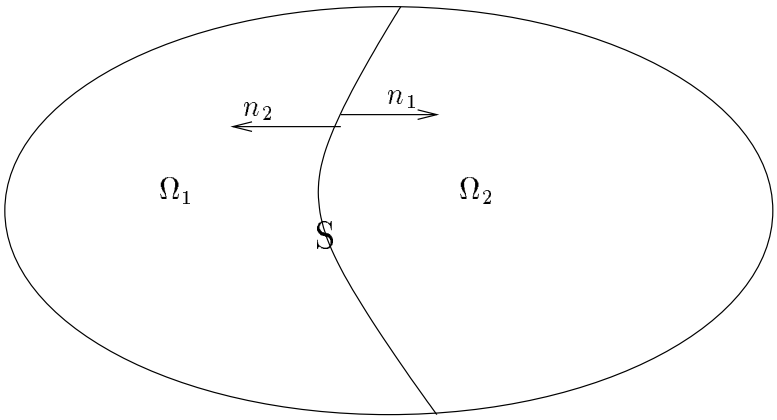


FIG. 2 – *Décomposition en deux sous-domaines sans recouvrement*

et satisfait la condition de continuité:

$$g_S \equiv \frac{\partial u_1}{\partial n_1} + \frac{\partial u_2}{\partial n_2} = 0 \qquad \text{sur} \quad S \tag{13}$$

où n_1 et n_2 sont respectivement les normales extérieures aux sous-domaines Ω_1 et Ω_2.

La méthode consiste alors à calculer, via une méthode itérative, la valeur de u_S. Cela nécessite de résoudre à chaque itération les problèmes locaux (12) qui sont bien posés dès lors que u_S est fixé. Le résidu à l'interface étant fourni par le déséquilibre des flux g_S.

Formulation discrète

L'inconnue u et le second membre f se décomposent respectivement en $u = (u_1, u_2, u_S)$ et $f = (f_1, f_2, f_S)$.
Le système linéaire (4) prend la forme:

$$\begin{pmatrix} A_{11} & 0 & A_{1S} \\ 0 & A_{22} & A_{2S} \\ A_{1S}^T & A_{2S}^T & A_{SS} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_S \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_S \end{pmatrix} \tag{14}$$

où:

$$A_{SS} = A_{SS}^1 + A_{SS}^2$$

est la somme des contributions des deux domaines à la matrice d'interface.
On peut, comme dans le cas continu, transformer ce problème en deux sous-problèmes:

$$\begin{aligned} u_1 &= A_{11}^{-1} \left(f_1 - A_{1S} u_S\right) \\ u_2 &= A_{22}^{-1} \left(f_2 - A_{2S} u_S\right) \end{aligned} \tag{15}$$

qui admettent une solution unique si u_S est fixé.
En substituant ces deux équations dans la troisième (14), on obtient le problème condensé à l'interface:

$$\widetilde{\mathcal{S}} u_S = \widetilde{f}_S \tag{16}$$

Avec :

$$\widetilde{\mathcal{S}} \equiv A_{SS} - A_{1S}^T A_{11}^{-1} A_{1S} - A_{2S}^T A_{22}^{-1} A_{2S}$$

$$\widetilde{f}_S = f_S - A_{1S}^T A_{11}^{-1} f_1 - A_{2S}^T A_{22}^{-1} f_2$$

où la matrice $\widetilde{\mathcal{S}}$ est appelée le complément de Schur de la matrice A_{SS}.
C'est une matrice dense mais mieux conditionnée que $\mathcal{A}$. On montre, [77] que pour des problèmes elliptiques du second ordre du type (3), on a:

$$\kappa(\widetilde{\mathcal{S}}) = \mathcal{O}\left(h^{-1}H^{-1}\right) \quad \text{et} \quad \kappa\left(\mathcal{A}\right) = \mathcal{O}\left(h^{-2}\right) \tag{17}$$

Il est alors courant d'inverser le problème (16) par un gradient conjugué, dont la convergence sera plus rapide sur le problème condensé que sur le problème global. Dans la pratique, il n'est pas nécessaire d'assembler explicitement la matrice $\widetilde{\mathcal{S}}$. En effet, (16) peut encore s'écrire:

$$\widetilde{\mathcal{S}} u_S \equiv \left(\widetilde{\mathcal{S}}_1 + \widetilde{\mathcal{S}}_2\right) u_s = \widetilde{f}_S \tag{18}$$

avec:

$$\widetilde{\mathcal{S}}_1 \equiv A^1_{SS} - A^T_{1S} A^{-1}_{11} A_{1S}$$
$$\widetilde{\mathcal{S}}_2 \equiv A^2_{SS} - A^T_{2S} A^{-1}_{22} A_{2S}$$

À chaque itération, le produit matrice/vecteur est calculé en effectuant, en parallèle, des produits matrice/vecteur locaux à chaque domaine, avant d'additionner les valeurs aux interfaces. Ces produits locaux nécessitent la résolution de problèmes internes de la forme $A_{ii} z = y$, $i = 1, 2$. Selon la taille des domaines, la résolution se fait avec un solveur direct ou un solveur itératif.

Un algorithme possible est, par exemple (*Le Tallec* [77]):

Algorithme III .1 Méthode du complément de Schur primale: GC sur le problème condensé

$r^0 = b$ *résidu initial*
$z^0 = 0$
u^0_S fixé

Tant que non convergence répète:

Calcul de la direction de descente préconditionnée: $p^n = P^{-1} r^n$

Calcule : $z^n = p^n + \dfrac{(r^n, p^n)}{(r^{n-1}, p^{n-1})} p^{n-1}$

Pour tous les domaines $k = 1, \cdots, N_p$ faire :

Résoudre en parallèle le problème de Dirichlet :

$A_{kk} u_k = -A^T_{kS} \widehat{R_k} z^n$

Calcule:

$$s^n = \sum_{k=1}^{N_p} \widehat{R_k}^T \left(A^k_{SS} \widehat{R_k} z^n - A_{kS} u_k\right) \equiv \widetilde{\mathcal{S}} z^n$$
$$\alpha_n = \frac{(r^n, p^n)}{(s^n, z^n)}$$
$$u^{n+1}_S = u^n_S + \alpha_n z^n$$
$$r^{n+1} = r^n - \alpha_n s^n$$

fin Pour

fin Tant que

fin algorithme

La résolution de (16) nécessite un préconditionnement adapté car le conditionnement de $\widetilde{\mathcal{S}}$ est encore important.

Les préconditionneurs dits de "Dirichlet-Neumann" et "Neumann-Neumann" font partie des premiers à avoir été proposés dans la bibliographie. Leur nom est issu de la remarque suivante: inverser une matrice de complément de Schur sur un sous-domaine équivaut à résoudre, sur ce même sous-domaine, un problème avec conditions limites de Neumann.

Préconditionneurs Dirichlet-Neumann :
Une première technique de préconditionnement consiste à considérer que comme dans le cas ou les domaines sont parfaitement symétriques on a:

$$\widetilde{\mathcal{S}}_1 = \widetilde{\mathcal{S}}_2 = \frac{1}{2}\widetilde{\mathcal{S}}$$

on peut choisir indifféremment $\widetilde{\mathcal{S}}_1$ ou $\widetilde{\mathcal{S}}_2$ comme préconditionneurs.

Préconditionneurs Neumann-Neumann :
Une seconde méthode consiste à approcher l'inverse de la somme par la somme des inverses, à un facteur multiplicatif près. On a alors :

$$\widetilde{P}_{NN}^{-1} = \widetilde{\mathcal{S}}_1^{-1} + \widetilde{\mathcal{S}}_2^{-1}$$

On montre que ([77] et référence citée):

$$\kappa(P_{NN}^{-1}\widetilde{\mathcal{S}}) = \mathcal{O}\left(1 + \log^2 \frac{H}{h}\right)$$

Ce préconditionneur est très intéressant car il s'applique à toute forme de domaine et est robuste. Toutefois, il est très coûteux puisqu'il nécessite deux fois plus de résolutions a chaque itération.

2.1.3 La sous-structuration

Le problème des algorithmes présentés précédemment est que le transfert d'information se fait par "proximité" (*i.e.* par les interfaces). Il est donc local. Or, dans le cadre d'un problème elliptique, on sait que la dépendance du problème au domaine est globale (*i.e.* une perturbation locale a un effet global). Ainsi, il manque un moyen de transmettre, à faible coût, l'information au travers de tout le domaine.
Une technique très populaire consiste à utiliser une méthode de sous-structuration en introduisant un espace grossier.

2.1.3.1 Méthodes avec recouvrement

On construit des préconditionneurs additifs de Schwarz (10) à deux niveaux:

$$P_{asSS}^{-1} = R_H A_H^{-1} R_H + \sum_{i=1}^{Np} R_i^T A_i^{-1} R_i = \sum_{i=0}^{Np} R_i^T A_i^{-1} R_i \tag{19}$$

où $A_H \equiv A_0$ est la matrice assemblée sur l'espace grossier et $R_H \equiv R_0$ l'opérateur de restriction à ce sous-espace.
Le conditionnement du problème ne dépend alors ni de H, ni de h. Ce préconditionneur est **optimal** mais très coûteux. On peut toutefois envisager de le calculer de manière approchée en effectuant des résolutions inexactes, par exemple avec des factorisations incomplètes, une méthode multigrilles ou des méthodes SOR (de type Gauss-Seidel).

2.1.3.2 Méthodes sans recouvrement

On construit des préconditionneurs de Jacobi par blocs à deux niveaux; on utilise l'espace grossier pour transmettre l'information et on se contente de préconditionner par la diagonale du complément de Schur.

$$P_{SS}^{-1} = \widehat{R}_H \widetilde{\mathcal{S}}_H^{-1} \widehat{R}_H + \sum_{i=1}^{Np} \widehat{R}_i^T \, Diag(\widetilde{\mathcal{S}}_i)^{-1} \widehat{R}_i \tag{20}$$

où $\widetilde{\mathcal{S}}_H \equiv \widetilde{\mathcal{S}}_0$ est le complément de Schur assemblé sur l'espace grossier et $\widehat{R}_H \equiv \widehat{R}_0$ l'opérateur de restriction à l'intersection entre l'interface S et l'espace grossier.
On montre que ([77] [15] et référence citée):

$$\kappa(P_{SS}^{-1}\widetilde{\mathcal{S}}) = \mathcal{O}\left(\frac{1}{H^2}\left(1+\log\frac{H}{h}\right)^2\right)$$

Bramble, Pasciak et Schatz [11] ont proposé une variante consistant à utiliser directement A_H^{-1} dans le préconditionneur, plutôt que $\widetilde{\mathcal{S}}_H^{-1}$. On montre alors que ce préconditionneur, connu sous le nom de BPS, permet d'obtenir un taux de convergence en $\mathcal{O}\left(1+\log^2 H/h\right)$ (*i.e.* un taux équivalent à celui de "Neumann-Neumann" pour un coût bien moindre).

2.2 Les méthodes de décomposition de domaines duales

On peut définir les méthodes de complément de Schur duales comme des méthodes traitant la contrainte de continuité aux interfaces par l'ajout de multiplicateurs de Lagrange. Différentes techniques peuvent êtres employées pour formuler un problème dual de (12) (13), et nous renvoyons à *Le Tallec* et aux références citées par celui-ci pour leur description.

D'autre part et par abus de langage, de nombreux auteurs qualifient de duale toute méthode de décomposition de domaine qui remplace les conditions aux limites de Dirichlet à l'interface par des conditions de Neumann. Cela conduit à remplacer (12) (13) par:

$$u_i \text{ est solution de } \left\{ \begin{array}{rlll} Lu_i & = f & \text{dans} & \Omega_i \\ u_i & = 0 & \text{sur} & \partial\Omega_i \setminus S \quad \text{pour } i=1,2 \\ \frac{\partial u_i}{\partial n_i} & = \pm\,\lambda & \text{sur} & S \end{array} \right. \tag{21}$$

où $\pm$ signifie que les normales n_1 et n_2 sont opposées.
et satisfait la condition de continuité:

$$u_1 - u_2 = 0 \qquad \text{sur } S \tag{22}$$

Une méthode duale couramment utilisée est la méthode FETI (Finite Element Tearing and Interconnecting) proposée par *C. Farhat* et *F. X. Roux* en 1991 [33].

Principe variationnel:

Soit la fonctionelle:

$$J(v) = \frac{1}{2}(Lv, v) - (f, v) \tag{23}$$

L'application du lemme de Lax-Milgram au problème modèle (3) montre que celui-ci est équivalent au problème :

Trouver $u \in \mathcal{V}$ solution de:

$$\min_{v \in \mathcal{V}} J(v) \tag{24}$$

qui, si Ω est partitionné en deux sous-domaines, équivaut à:

Trouver $(u_1, u_2) \in \mathcal{V}_1 \times \mathcal{V}_2$ solutions de :

$$\min_{v_1 \in \mathcal{V}_1} J(v_1) \quad \text{et} \quad \min_{v_2 \in \mathcal{V}_2} J(v_2) \tag{25}$$

avec :

$$J(v_1) = \frac{1}{2}(Lv_1, v_1) - (f, v_1) \quad \text{et} \quad J(v_2) = \frac{1}{2}(Lv_2, v_2) - (f, v_2)$$

sous la contrainte de continuité à l'interface:

$$v_1 = v_2 \quad \text{sur } S \tag{26}$$

Résoudre ce problème sous contrainte est équivalent à résoudre un problème de point-selle associé au Lagrangien:

$$\mathcal{L}(v_1, v_2; \mu) = J_1(v_1) + J_2(v_2) + (v_1 - v_2, \mu)_S \tag{27}$$

Trouver $(u_1, u_2; \lambda) \in \mathcal{V}_1 \times \mathcal{V}_2 \times \mathcal{Q}$ solutions de :

$$\inf_{(v_1, v_2)} \sup_{\mu} \mathcal{L}(v_1, v_2; \mu) \tag{28}$$

La condition de sup assure la continuité à l'interface tandis que la condition d'inf assure de trouver parmi toutes les paires de solutions admissibles celle minimisant l'énergie sur les deux domaines. Les multiplicateurs de Lagrange peuvent être considérés comme des forces d'interaction s'appliquant le long des interfaces entre les sous-domaines. Ceux-ci assurent "l'interconnection" des domaines. Une discrétisation par une méthode de Galerkin standard des espaces $\mathcal{V}$ et $\mathcal{Q}$ en des espace $\mathcal{V}_h$ et $\mathcal{Q}_h$ nous permet d'exprimer le Lagrangien discret comme:

$$\begin{aligned} \mathcal{L}(v_1, v_2; \mu) &= \frac{1}{2}\sum_{k=1}^{2} v_k^T A_{kk} v_k - \sum_{k=1}^{2} v_k^T b_k + \mu^T \sum_{k=1}^{2} \widehat{R}_k v_k \\ &= \frac{1}{2}\sum_{k=1}^{2} v_k^T A_{kk} v_k - \sum_{k=1}^{2} v_k^T \left(b_k - \widehat{R}_k^T \mu\right) \end{aligned} \tag{29}$$

où $\widehat{R}_k$ est l'opérateur de restriction à l'interface du sous-domaine k. Les conditions d'Euler impliquent que le couple (u, λ) est solution de:

$$\left\{ \begin{array}{rcl} A_1 u_1 &=& b_1 - \widehat{R}_1^T \lambda \\ A_2 u_2 &=& b_2 - \widehat{R}_2^T \lambda \\ \widehat{R}_1 u_1 - \widehat{R}_2 u_2 &=& 0 \end{array} \right. \tag{30}$$

Ce système définit le problème FETI.

Dans le cas de sous-domaines singuliers, *c.àd.* quand les sous-problèmes de Neumann (21) (22) sont mal posés et les matrices associées ne sont pas inversibles, il est nécessaire d'introduire les inverses généralisées des matrices. Le problème ainsi obtenu est alors résolu par un Gradient Conjugué Projeté, présentant une grande scalabilité [33, 34].

Dans le cas où les sous-domaines Ω_1 et Ω_2 ne sont pas singuliers, les matrices A_1 et A_2 sont inversibles et le problème (30) peut alors s'écrire:

$$\left\{ \begin{array}{rcl} u_1 & = & A_1^{-1}\left(b_1 - \widehat{R}_1^T\lambda\right) \\ u_2 & = & A_2^{-1}\left(b_2 - \widehat{R}_2^T\lambda\right) \\ D\lambda & = & \widehat{R}_2 A_2^{-1} b_2 - \widehat{R}_1 A_1^{-1} b_1 \end{array} \right. \tag{31}$$

où :

$$D = \left(\widehat{R}_1 A_1^{-1}\widehat{R}_1^T + \widehat{R}_2 A_2^{-1}\widehat{R}_2^T\right)$$

On peut alors appliquer une méthode de gradient quelconque sur le problème en λ, qui nécessite à chaque itération la résolution de problèmes locaux.

b
Soit λ^k la kième approximation des multiplicateurs de Lagrange, le résidu du problème à l'interface (30c) s'écrit:

$$D\lambda^k + \widehat{R}_1 A_1^{-1} b_1 - \widehat{R}_1 A_1^{-1} b_1 = \widehat{R}_1 u_1^k - \widehat{R}_2 u_2^k$$

Un algorithme possible est alors [66]:

Algorithme III .2 Méthode du complément de Schur dual

λ^0 fixé
$A_i u_i^0 = b_i \ i = 1, 2$ *initialisations des solutions locales*
$r^0 = u_1^0 - u_2^0$ sur S *résidu initial*
$p^0 = r^0$

Tant que non convergence répète:

Résolution en parallèle des problèmes de Neumann locaux:
$A_i v_i^k = \widehat{R}_i^T p^k \ i = 1, 2$

Calcul du produit par le complément de Schur dual:
$Dp^k = v_1^k - v_2^k$ sur S

Mise à jour :

$$\rho^k = -\frac{(r^k, p^k)}{(Dp^k, p^k)}$$
$$\lambda^{k+1} = \lambda^k + \rho^k p^k$$
$$r^{k+1} = r^k + \rho^k Dp^k$$
$$\gamma^k = -\frac{(r^{k+1}, Dp^k)}{(Dp^k, p^k)}$$
$$p^{k+1} = r^{k+1} + \gamma^k p^k$$

Mise à jour des solutions:
fin Tant que

fin algorithme

Des techniques de reconjuguaison des pentes peuvent être utilisées pour accélérer la convergence de la méthode [67]. Celles-ci permettent de recycler les pentes et de les réintroduire aux itérations ultérieures du Gradient Conjugué. Le préconditionnement de la méthode

FETI peut s'effectuer avec les même types de préconditionneurs que ceux présentés pour la méthode primale. On montre ainsi que le conditionnement du problème FETI avec un préconditionneur de Dirichlet est $\log^2 \frac{H}{h}$ et qu'il croît lentement quand $h \to 0$ (*C. Farhat et al.* [32]).

2.3 Applications au problème de Stokes :

La stratégie classique de résolution du problème de Stokes par une méthode de décomposition de domaine, repose sur l'élimination des vitesses de l'équation en pression (*cf.* chapitre précédent paragraphe 1 page 33). À chaque itération d'un algorithme de type Uzawa, le produit par le complément de Schur de la matrice en pression $BA^{-1}B^T$ doit être effectué. Les produits par B et B^T sont effectués par des techniques classiques en calcul parallèle: écriture du produit matrice/vecteur comme une somme sur tous les domaines de produits matrices/vecteurs locaux, puis somme des vecteurs aux interfaces. En revanche, le produit par A^{-1} est effectué par la méthode du complément de Schur primale ou par la méthode duale FETI.
Cette manière de procéder a plusieurs avantages:

1. La propriété (8), page 35, vue au chapitre précédent, nous assure que le problème en pression est d'ordre 0, *c.àd.* qu'il est indépendant de la taille de maille et de celle des sous domaines. Ainsi le nombre d'itérations nécessaires à sa résolution ne varie théoriquement pas avec le nombre de domaines. Elle permet également de travailler avec la matrice A qui a de bonnes propriétés.
2. L'utilisation de ces méthodes avec un préconditionneur adapté fournit une méthode de résolution du problème d'interface, indépendante ou peu dépendante, de la partition et du nombre de domaines.

Toutefois, on peut déplorer certains inconvénients:

1. Le préconditionnement de la méthode FETI peut s'avérer coûteux, ce qui la rend intéressante seulement pour des problèmes de grosse taille ou pour un nombre élevé de sous-domaines.
2. Le préconditionnement de la méthode primale peut s'effectuer à moindre coût. Toutefois il dépend du nombre de domaines et nécessite de recourir aux techniques de sous-structuration; ces techniques sont délicates à mettre en place sur des maillages non structurés.
3. Le couplage de l'algorithme d'Uzawa avec une méthode de décomposition de domaine constitue un procédé itératif à trois niveaux: à chaque itération de l'algorithme d'Uzawa, il faut résoudre itérativement un problème d'interface nécessitant la résolution directe ou itérative de problèmes locaux indépendants.

2.4 Discussion

De nombreuses choses sont à retenir des travaux sur les méthodes de décomposition de domaines. Sur la base des méthodes de Jacobi par blocs et additives de Schwarz ainsi que du lemme de partition de l'énergie, elles fournissent un cadre abstrait valable pour toutes les méthodes de partitionnement de domaine.
On retiendra que pour des problèmes 2D avec un maillage non régulier les méthodes avec ou sans recouvrements sont toutes susceptibles d'être optimales ou quasi optimales. C'est aussi le cas pour les méthodes hiérarchiques type BPX ([12] ou références précédemment

citées), qui sont l'équivalent des méthodes multigrilles pour le calcul parallèle. Par contre, pour les problèmes 3D hétérogènes, seules fonctionnent les méthodes sans recouvrements avec des préconditionneur type Neumann-Neumann ou encore FETI et, dès que le nombre de domaine augmente la sous-structuration devient une nécessité.

En définitive, quelle que soit la technique choisie, le point clé qui assure une bonne efficacité parallèle et la "scalabilité" (*i.e.* la capacité à rester efficace quand le nombre de processeurs augmente) est le préconditionnement. Celui-ci s'avère souvent délicat à mettre en place et est susceptible de devenir très coûteux. La méthode de partitionnement de domaine est, de ce point de vue très intéressante. En effet, dans le cas d'un préconditionnement par la diagonale, **le nombre d'itérations est indépendant du nombre de domaines** pour un coût d'assemblage du préconditionneur négligeable. Cette méthode, a été implantée dans le code 3D FORGE3®[57], et nous avons choisi de l'utiliser. Néanmoins les spécificités du 2D ont montré que les préconditionneurs diagonaux n'étaient pas assez performants (*cf.* chapitre II). Ainsi, nous sommes confrontés au problème classique: comment construire à moindre coût un préconditionneur efficace? De nombreuses techniques décrites dans le cadre des méthodes de décomposition de domaine peuvent être appliquées à notre approche ou être sources d'inspiration.

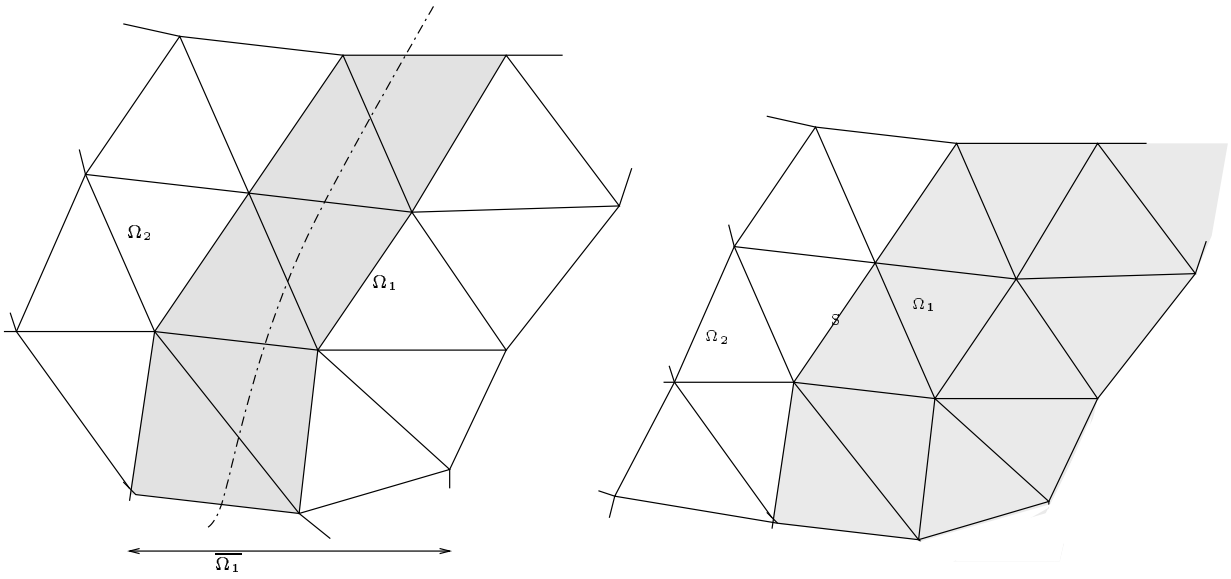
3 Stratégie de parallélisation SPMD

Comme nous l'avons vu, une large partie du travail de parallélisation de FORGE2® a consisté à modifier le solveur du code originel de manière à le rendre parallélisable. La nouvelle formulation éléments finis, présentée au chapitre I , nous permet d'utiliser un solveur itératif. À l'instar de FORGE3® tous les autres algorithmes utilisés dans FORGE2®, mis à part la résolution du système linéaire, peuvent aisément se localiser à chaque sous-domaine [57]. **Ainsi, le point clé de la stratégie de parallélisation réside dans la résolution parallèle des systèmes linéaires.**

La stratégie SPMD consiste à exécuter, sur tous les processeurs, les mêmes séquences d'instructions mais sur des données différentes. Le maillage est décomposé en sous-domaines. Une version complète du code s'exécute sur chaque sous-domaine, **avec les conditions aux limites du problème global**. Par rapport à la version séquentielle, deux structures de données spécifiques sont ajoutées. La première est relative à la **description du processus parallèle** et contient le nombre de processeurs (et de domaines) $Nproc$, le numéro de chaque domaine $Numdom$ et les noms des processeurs pour le cas d'une architecture distribuée. La seconde contient la description des interfaces. Étant dépendante de la partition choisie, elle sera présentée aux paragraphes qui lui sont dédiés. Cette décomposition du maillage peut, en effet, s'effectuer de différentes manières (*cf.* figure 3): soit par noeuds, chaque noeud du maillage appartenant à un et à un seul sous-domaine, soit par éléments, chaque élément du maillage appartenant alors à un et à un seul sous-domaine. En terme de nombre d'opérations et de coût de communication, ces deux stratégies sont équivalentes. Elles affectent toutefois la stratégie SPMD par la manière d'effectuer les opérations parallèles et la structure de données. La facilité d'implémentation est alors le principal critère de choix entre les deux stratégies et dépend principalement du type de préconditionneurs parallèles employés. Dans le cadre de ce travail, nous avons choisi une décomposition par éléments. Nous présenterons toutefois succinctement la stratégie par noeuds au paragraphe 3.2.

La résolution itérative d'un système linéaire par MINRES met en jeu quatre types d'opérations qui doivent être parallélisées:

- Produit matrice/vecteur
- Produit scalaire
- Préconditionnement
- Réactualisation de vecteur



(a) Décomposition par noeuds (b) Décomposition par éléments

FIG. 3 – *Décomposition en sous-domaines*

3.1 Méthode SPMD par éléments:

C'est la méthode la plus souvent utilisée et c'est en particulier celle utilisée dans FORGE3®.

Structure de données:

Pour chaque sous-domaine on construit une structure de donnée d'interface contenant:

- le nombre de sous-domaines voisins (et aussi le nombre d'interfaces),
- le numéro des sous-domaines voisins,
- le nombre de noeuds de chaque interface,
- la numérotation des noeuds d'interface.

Pour une interface donnée, les seules informations communes aux deux processeurs situés de part et d'autre de celle-ci sont, le nombre de noeuds appartenant à cette interface et leur numérotation. Ainsi, pour ces deux processeurs le nième noeud de l'interface correspond au même noeud du maillage global, quel que soit leur numérotation locale. Cela permet alors d'implémenter aisément les sommes aux interfaces.

Produit matrice/vecteur :

La matrice globale $\mathcal{A}$ s'écrit naturellement comme la somme des matrices locales *i.e.* :

$$\mathcal{A} = \sum_{i=1}^{N_p} \widehat{\mathcal{A}}_{\Omega_i} = \sum_{i=1}^{N_p} R_i^T \mathcal{A}_{\Omega_i} R_i \tag{32}$$

ainsi, le produit matrice/vecteur peut s'écrire:

$$\begin{aligned} \mathcal{A}x &= \left(\sum_{i=1}^{N_p} \widehat{\mathcal{A}}_{\Omega_i} \right) x = \left(\sum_{i=1}^{N_p} R_i^T \mathcal{A}_{\Omega_i} R_i \right) x \\ &= \sum_{i=1}^{N_p} \left(R_i^T \mathcal{A}_{\Omega_i} x_{|\Omega_i} \right) \end{aligned} \tag{33}$$

Ainsi, la connaissance des contributions du sous-domaine à la matrice globale suffit pour effectuer le produit matrice/vecteur, à condition bien entendu de connaître la restriction du vecteur au sous-domaine.

La philosophie d'une partition par éléments est donc de connaître sur chaque sous-domaine les restrictions de tous les vecteurs et seulement la contribution du sous-domaine à la matrice globale.

Pour l'assemblage des matrices et des vecteurs éléments finis ainsi que pour les produits matrice/vecteur, les calculs locaux sont faits en premier. Les communications sont effectuées dans un second temps pour assembler les restrictions des vecteurs. Ces communications sont de la taille des interfaces, puisque seules les contributions des noeuds communs sont incomplètes (d'où l'avantage d'utiliser des éléments finis de faible ordre).

Préconditionnement :

Les préconditionneurs diagonaux sont assimilés à des vecteurs et reçoivent donc un traitement similaire: assemblage de ceux-ci sur chaque sous-domaine, puis échange de données pour calculer leur restriction au sous-domaine. Ensuite, comme l'inverse d'une matrice bloc diagonale est la matrice formée par l'inverse des blocs diagonaux, on peut calculer la restriction du préconditionneur au sous-domaine.

$$\begin{aligned} P^{-1}x &= \left(\sum_{i=1}^{N_p} R_i^T P_i^{-1} R_i \right) x \\ &= \sum_{i=1}^{N_p} R_i^T P_i^{-1} x_{|\Omega_i} \end{aligned} \tag{34}$$

On remarque que pour tous les noeuds de l'interface, on est amené à calculer plusieurs fois l'inverse de la diagonale ou du bloc diagonal.

Produit scalaire :

Comme sur chaque domaine la restriction des vecteurs est connue, les produits scalaires nécessitent un traitement particulier pour éviter d'ajouter plusieurs fois la même contribution. On introduit donc un vecteur de pondération, noté $Pond$, valant 1 pour tout noeud interne (n'appartenant qu'au domaine considéré) et égal à l'inverse du nombre de domaines partageant le noeud si celui-ci est d'interface. On écrit alors:

$$(u, v)_\Omega = \sum_{i=1}^{N_p} \left(\sum_{noe=1}^{Nbnoe_i} Pond(noe)\ u_{|\Omega_i}(noe)\ v_{|\Omega_i}(noe) \right) \tag{35}$$

Sur chaque domaine, on effectue donc les produits scalaires pondérés avant de faire la somme sur tous les domaines et de rapatrier la valeur globale sur chaque processeur.

Réactualisations :

Les opérations d'addition entre vecteurs, ou de multiplication par un scalaire sont effectuées concurrentiellement par chaque processeur. Ces opérations ne nécessitent aucune communication puisque les données utilisées (restriction de vecteur et nombre issus d'un produit scalaire) sont globales.

Chacune de ces quatre opérations induit des calculs redondants aux interfaces, qui réduisent les coûts de communication en augmentant légèrement, mais de manière négligeable, la perte d'efficacité parallèle.

La méthode ainsi développée est optimale dans la mesure où quel que soit le nombre de domaines, les opérations effectuées sont les mêmes qu'en séquentiel et aboutissent donc au même nombre d'itérations de résolution. Par contre, pour chaque produit matrice/vecteur et pour chaque produit/scalaire une synchronisation des processeurs est nécessaire. Il est donc clair que l'équilibrage des charges de travail entre les processeurs est une condition nécessaire à l'efficacité de la parallélisation. Le partitionnement du maillage est donc un point crucial et doit permettre d'obtenir des domaines équilibrés avec des interfaces de taille minimale. Cette conclusion s'applique à la fois pour la partition par noeuds et pour la partition par éléments. Le problème de l'équilibrage des charges est abordé au paragraphe 4.1.

L'algorithme parallèle s'écrit alors:

Algorithme III .3 Méthode SPMD par éléments

$V^k = (v_k, p_k)$ fixé

Pour tous les processeurs faire :

Assemblage éléments finis :

Calcule H_{Ω_i}, G_{Ω_i} *matrice et second membre locaux*

Extrait le préconditionneur bloc diagonal D_{Ω_i}

Communications: Sommes aux interfaces :

$$G_i = \sum_{i=1}^{N_p} R_i G_{\Omega_i} \qquad \textit{construction du vecteur global}$$

$$D_{|\Omega_i} = \sum_{i=1}^{N_p} R_i D_{\Omega_i} R_i^T \qquad \textit{construction du préconditionneur global}$$

Calcul de l'inverse du préconditionneur

$D_{|\Omega_i}^{-1}$

Résolution du système linéaire $H\Delta V = -G$: MINRES algorithme (II .1)

Produits matrice/vecteur *équation (33)*

Produits scalaires *équation (35)*

Préconditionnement diagonal *équation (34)*

Réactualisations concurrentielles

Fin résolution

$V_{|\Omega_i}^{k+1} = V_{|\Omega_i}^k + \Delta V_{|\Omega_i}$ *réactualisations*

fin Pour

fin algorithme

3.2 Méthode SPMD par noeuds

La philosophie d'une partition par noeuds est de considérer qu'un noeud donné du maillage appartient à un processeur et un seul. En pratique, il s'avère toutefois beaucoup plus facile de construire la partition en opérant un recouvrement d'une bande d'éléments entre les sous-domaines (*cf.* figure 3(a)). Une décomposition par noeuds se construit donc à partir d'une décomposition par éléments avec un recouvrement d'une bande d'éléments. Nous renvoyons à *E. Issman et G. Degrez* [54, 53] pour plus de détails sur ce type de décomposition.

On notera pour cette section :

Ω_i	la partition des N_i noeuds alloués au processeur i
$\overline{\Omega}_i$	l'ensemble des $\overline{N}_i$ noeuds connus par le processeur i
$\Omega_i^E = \overline{\Omega}_i \bigcap \Omega_i$	l'ensemble des N_E noeuds extérieurs à Ω_i

Dans le cadre d'une méthode éléments finis, l'intérêt d'effectuer des recouvrements apparaît lors de l'assemblage. En effet, la connaissance de tous les éléments connectés aux noeuds de la partition permet d'assembler, **sans communication et de manière exacte**, la restriction d'un vecteur global à la partition.

Ainsi, soit v un vecteur global. En supposant que les noeuds de Ω_i^E ont été numérotés en

dernier, on écrit:

$$v_i = \begin{pmatrix} v_{\Omega_i} & 3 \times N_i \ termes\ exact \\ v_E & 3 \times N_E \ termes\ inexact \end{pmatrix} \tag{36}$$

où v_i représente les composantes du vecteur v calculées par le processeur i et v_E la partie du vecteur v connue mais non à la charge du processeur i.

De la même manière, soit H la matrice de raideur globale et H_i la matrice de dimension $N_i \times N_i$ assemblée par le processeur i.

$$H_i = \left(\begin{array}{c|c} H_{ii} & H_{iE}^T \\ \hline H_{iE} & H_{EE} \end{array} \right) \tag{37}$$

Grâce à la connaissance de tous les éléments connectés aux noeuds de la partition, seule la matrice H_{EE} est incomplète.

A l'inverse de la stratégie par éléments où l'on échange les informations **après** avoir effectué des calculs sur des vecteurs globaux, la philosophie par noeuds consiste à communiquer **avant** d'effectuer les calculs en **écrasant les valeurs inexactes**. La stratégie SPMD se déroule alors ainsi:

Produit matrice/vecteur:

1. Communication et synchronisation:
   On écrase la valeur extérieure v_E du vecteur v par sa valeur exacte
2. On effectue le produit matrice/vecteur *pour les valeurs internes*

$$u = H.v \text{ où } u_i = \begin{pmatrix} u_{\Omega_i} = H_{ii} v_i + H_{iE}^T v_E \\ u_E \text{ non calculé} \end{pmatrix} \tag{38}$$

Autres opérations:
On effectue les calculs concurrentiellement uniquement sur les noeuds internes, les valeurs externes n'étant nécessaires que pour les produits matrice/vecteur. Les produits scalaires n'ont pas besoin d'être pondérés.

La méthode par noeuds demande exactement le même montant de communications que la méthode par éléments, la différence étant le moment où celles-ci sont effectuées. Elle nécessite une structure de données un peu plus complexe de manière à gérer les deux types de noeuds (internes et externes) pour chaque sous-domaine. Elle implique aussi des calculs redondants plus nombreux lors de l'assemblage éléments finis en raison des recouvrements effectués. Elle à l'avantage de contenir sur chaque domaine la restriction de la matrice globale, ce qui peut-être décisif dans la construction d'un préconditionneur.

4 Résolution parallèle

4.1 Algorithme de partitionnement

Le travail présenté dans ce paragraphe s'est effectué en étroite collaboration avec *H. Digonnet* qui est impliqué dans le projet ESPRIT de recherche à long terme DRAMA [18] visant à effectuer les opérations de remaillage et repartitionnement parallèle de manière dynamique.

Une résolution parallèle efficace nécessite que les temps de synchronisation entre les processeurs soient les plus court possible. Pour cela, il faut que tous les processeurs aient la même charge de travail ou plus généralement une charge proportionnelle à leur puissance de calcul. Dans le cas de processeurs ayant tous la même puissance de calcul, on considère généralement qu'une distribution équilibrée des charges sur chaque processeur est obtenue si les partitions ont le même nombre d'éléments. Une seconde condition est que le nombre et les temps de communications soient minimum. Il faut alors minimiser la taille et le nombre d'interfaces entre les sous-domaines. Ce problème qui se formalise grâce à la théorie des graphes est un problème Np-complet. Il n'existe donc pas, pour l'instant, d'algorithme général permettant de le résoudre. Toutefois, de nombreux algorithmes heuristiques ont été étudiés par différentes équipes et aboutissent à des solutions de très bonne qualité, voir quasi-optimales. On peut citer entre autre *Jostle* développé à l'université de Greenwhich par *C. Walshaw et al.* [81], *Metis* développé à l'université du Minesota par *G. Karypis et al.* [56] ou encore *Chaco* [50]. Nous utilisons un algorithme dit "glouton" (greedy), qui utilise uniquement la topologie du maillage. Celui-ci consiste en trois étapes distinctes:

1. **Sélection de N_p "éléments-germes"**
   On détermine en premier lieu un élément par sous-domaine, appelé "germe", à partir duquel la procédure "glouton" se propagera. On cherche pour cela à maximiser la distance minimale entre ceux-ci. On sélectionne autant "d'éléments-germes" que l'on désire de sous-domaines. Si k, $k < N_p$, "éléments-germes" $\bar{e}_1, \bar{e}_2, \cdots, \bar{e}_k$ sont déjà choisis, on recherche le $k+1$ ième tel que:

   $$\bar{e}_{k+1} \text{ maximise } \min_{1 \leq j < k} d(\bar{e}_{k+1}, \bar{e}_j)$$

   que nous écrivons par abus de notation: $\bar{e}_{k+1} = \max_{e \in \mathcal{E}} \min_{1 \leq j < k} d(e, \bar{e}_j)$

   la distance $d(.,.)$ étant définie récursivement par:

   $d(e_1, e_2) = 1$ si les éléments ont un noeud commun
   $d(e_1, e_2) = 2$ si $d(e_1, e_2) \neq 1$ et si il existe au moins un élément $\bar{e}$ tel que:
   $d(e_1, \bar{e}) = 1$ et $d(e_2, \bar{e}) = 1$

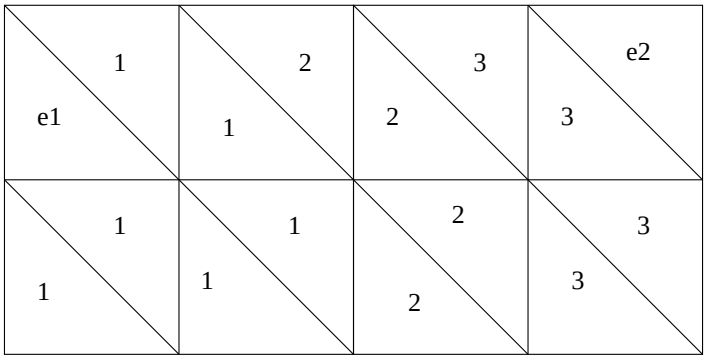


FIG. 4 – *Exemple de calcul de la distance entre deux éléments*

2. **Procédure "glouton"**
   Les Np germes étant choisis, on attribue une couleur différente à chacun d'entre eux. Pour chaque élément coloré, on propage alors de manière récursive sa couleur à tous les éléments voisins (situés à une distance de 1) non colorés (voir figure 5). Lorsque tous les éléments sont colorés, on a une première partition que l'on essaie ensuite d'améliorer.

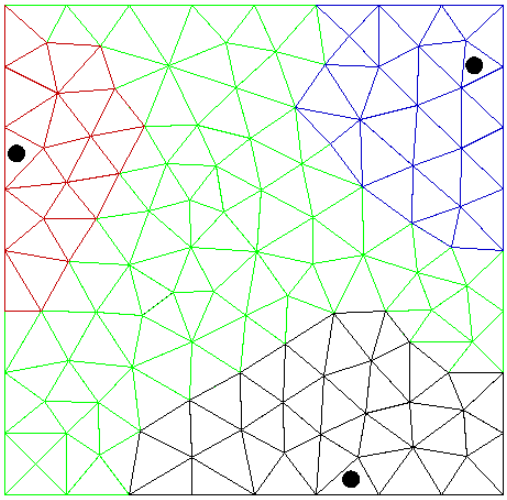


FIG. 5 – *Exemple de développement de trois sous-domaines par un algorithme glouton. Les germes sont représentés par les trois points noirs. Les éléments de couleur verte n'ont pas encore été attribués.*

3. **Régularisation des partitions ainsi générées**
   Lors de cette étape, on minimise par un processus itératif les interfaces (communications) en effectuant des changements locaux de couleur. Cette étape s'effectue en revisitant la couleur de chaque élément du maillage. Un triangle à au plus trois éléments adjacents (selon une arète) de une à trois couleurs possibles. On modifie la couleur du triangle visité afin d'optimiser une fonction coût tenant compte du nombre d'éléments par couleurs (charge des processeurs) et du nombre de communications dont l'unité est ici le nombre d'arètes partagées par des éléments de couleurs différentes. Nous renvoyons à [19] pour l'expression de cette fonction coût. Cet algorithme minimise les communications **locales** du triangle puisqu'il y a communication entre deux éléments voisins s'ils sont de couleurs différentes. L'algorithme itératif se déroule ainsi : on visite un élément, on optimise les communications locales en modifiant ou non sa couleur, on passe à un élément non encore visité et qui ne fait pas partie des voisins d'un élément déja visité et l'on recommence l'optimisation, jusqu'à ce que l'on ait visité tous les groupes d'éléments.

La programmation directe de cet algorithme aboutit parfois à des partitions peu satisfaisantes (*cf.* figure 6(a)) et ce pour plusieurs raisons:

– Une fois un germe choisi, celui-ci n'est pas remis en cause à la lumière du choix des autres germes. Pourtant, une configuration optimale pour un nombre p de germes ne le seras plus pour $p+1$ germes. Les germes sélectionnés sont donc de bons points de départ mais ne sont pas optimaux. Par exemple, sur la figure 6(a), les deux premiers éléments sélectionnés sont situés aux coins inférieur gauche et supérieur droit du carré. Le troisième "élément-germe" choisi par l'algorithme est alors situé dans le coin inférieur

droit, ce qui rend les choix précédents non optimaux. Le meilleur choix étant celui d'éléments situés sur trois des cotés et formant le plus grand triangle équilatéral possible.

– La procédure "glouton" (étape 2) est basée sur la distance que nous avons définit précédemment. Cela peut défavoriser certains germes et conduire à des partitions très déséquilibrées. Par exemple, si un germe est situé dans un coin, il se développera moins vite qu'un autre germe situé sur une arête ou au centre du maillage. Comme la régularisation est **locale** elle ne parvient à rééquilibrer les charges qu'en détériorant la qualité de la partition, par exemple en effectuant un "passage en force" (*cf.* figure 6(a)).

L'algorithme de partitionnement a donc été amélioré en rajoutant les critères suivants:

1. **Remise en cause des germes déjà sélectionnés :**
   chaque fois qu'un nouveau germe est sélectionné, les choix des germes précédents sont reconsidérés un à un de la manière suivante:
   Si $\bar{e}_{k+1}$ est choisis, on recherche, $\bar{e}_1, \bar{e}_2, \cdots, \bar{e}_k$ tels que:

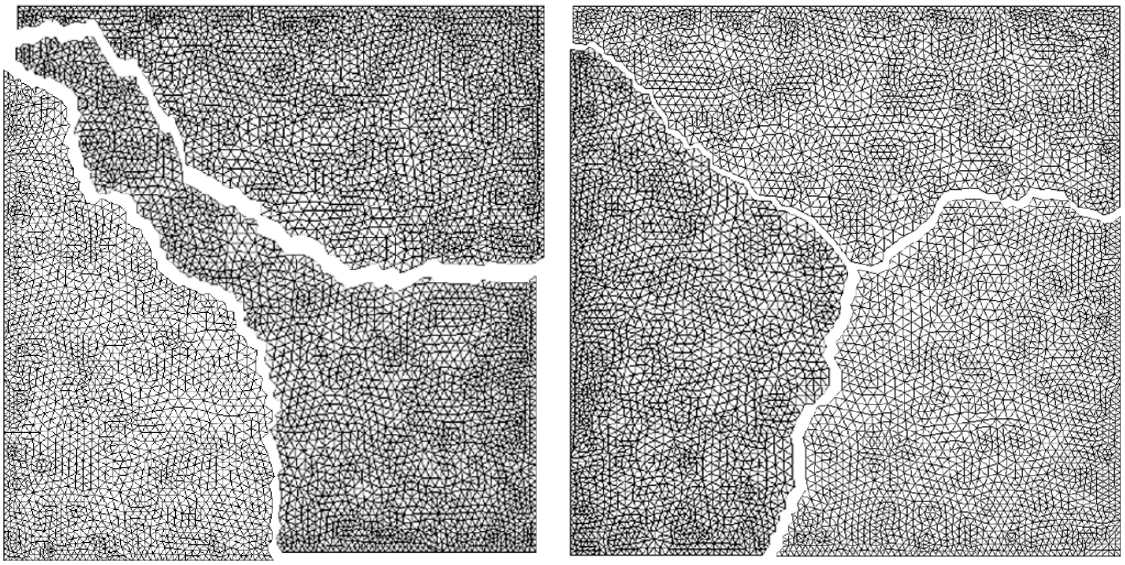
$$\bar{e}_j = \max_{e \in \mathcal{E}} \min_{l<j \text{ et } l=k+1} d(e, \bar{e}_l)$$

2. **Introduction d'un nombre maximum et d'un nombre minimum** d'éléments pouvant être sélectionnés à chaque étape de l'algorithme glouton. Cela permet d'éviter le développement de domaines en "corridor" au prix d'un déséquilibre de charge parfois plus important.
3. **Modification de l'algorithme de régularisation** traitant la fonction coût de façon à pouvoir traiter un fort déséquilibre initial. On introduit une notion de distance à l'interface initiale comme critère supplémentaire pour la migration d'éléments d'une couleur à l'autre. Celà permet d'éviter une certaine dépendance à la numérotation notée précédemment lorsque un élément devait être choisis parmi n possibles. Cet algorithme se rapproche alors des techniques de repartitionnement développée dans le cadre du projet DRAMA [19].

Ces deux améliorations permettent d'obtenir des partitions de bien meilleure qualité, comme on le voit sur la figure 6(b). C'est celles que nous utilisons dans la suite.

4.2 Remaillage et repartitionnement parallèle

Nous avons adopté une approche très simple qui consiste à rapatrier tous les maillages déformés sur un processeur "maître". Pendant que les "esclaves" sont inactifs, le maître reconstitue le maillage déformé global, effectue le remaillage avant de repartitionner le nouveau maillage. Il distribue ensuite les nouveaux sous-maillages aux autres processeurs qui reprennent les calculs éléments finis. Cette approche n'est pas optimale et va à l'encontre de la loi d'Amdhal. Elle a été motivée par sa simplicité et par le faible coût des opérations de remaillages et de repartitionnement en 2D. Par ailleurs, effectuer ces opérations en parallèle nécessite un traitement dynamique qui est très délicat à mettre en place. Ce problème fait l'objet du projet de recherche DRAMA [18]. Nous renvoyons donc le lecteur aux travaux de *H. Digonnet* pour l'approfondissement de ces aspects. Un exemple de simulation du procédé de filage arrière sur un maillage grossier incluant de très nombreuses opérations de remaillage et repartitionnement est présenté sur la figure 7.



(a) Ancienne partition - 1969, 1973 et 1995 noeuds

(b) Nouvelle partition - 1956, 1940 et 1960 noeuds

FIG. 6 – *Partitions d'un maillage de 5757 noeuds*

4.3 Comportement du solveur parallèle

On trouvera dans la thèse de *S. Marie* [57] un grand nombre de résultats sur le comportement du solveur parallèle en 3D et ce pour de nombreuses plateformes parallèles. Par exemple, pour la simulation complète sur une machine IBM SP2 du forgeage d'une ailette de réacteur, maillée avec 23000 tétraèdres et 7077 noeuds, et incluant des opérations de remaillages et de repartitionnement, il obtient, sur 10 processeurs une accélération de 9.5, soit une efficacité de 95 % et sur 20 processeurs une accélération de 14.2 soit une efficacité de 71 %. Sur le même problème, et pour un maillage de 58972 éléments et 14743 noeuds, une efficacité de 53% est obtenue sur huit stations de travail DEC Alpha connectées par un réseau f.d.d.i..

Les résultats que nous présentons dans la suite ont été obtenus à partir de la formulation éléments finis P1+P1 simplifiée stabilisée (*cf.* chapitre I). Le paramètre de stabilisation est $\beta = 3$. Nous nous contenterons d'illustrer notre propos sur quelques exemples simples et renvoyons au dernier chapitre ou à [57] pour des résultats sur des simulations de forgeage complètes.

4.3.1 Plateformes parallèles

Les plateformes parallèles dont nous disposons au CEMEF et que nous utilisons pour ce projet, sont des calculateurs SUN à mémoire partagée.

- SUN Enterprise 4000/5000 avec 3 Go de mémoire vive et 5,2 Go de mémoire virtuelle, équipé de 14 processeurs Sun UltraSPARC-II à 248 MHz.
- SUN Enterprise 450 avec 2 Go de mémoire vive et 4,5 Go de mémoire virtuelle équipé de 4 processeurs Sun UltraSPARC-II à 400 MHz.

Les noeuds de ces calculateurs sont connectés selon une topologie en hypercube par un réseaux Ethernet rapide à 100 $Mo.s^{-1}$. Les processeurs possèdent une mémoire cache de premier niveau (accès très rapide) de 16 Ko et, une mémoire cache de second niveau (accès rapide) de

(a) Partition initiale

(b) incrément 100

(c) incrément 450

(d) Fin de calcul incrément 750

FIG. 7 – *Exemple de simulation du problème de filage arrière sur 4 processeurs. Isovaleurs de vitesse*

4 *Mo*. Sur ces deux machines, les exécutables sont obtenus via le compilateur *f77* et l'option d'optimisation *-fast*.

Il faut noter que ces calculateurs sont des serveurs de calcul utilisés par de nombreuses autres personnes, ce qui s'est avéré être une source de perturbation de nos mesures de temps de calculs. En raison des accès disques gérés automatiquement, des perturbations sont encore possibles même lorsque les calculateurs sont entièrement réservés pour nos calculs. Cela est

d'autant plus vrai que le nombre de processeurs requis est élevé. Aussi, même si tous les temps de calcul présentés ont été obtenus lorsque les calculateurs étaient entièrement réservés pour ces mesures, nous ne pouvons, à partir de 10 processeurs, garantir que ceux-ci sont parfaitement reproductibles. C'est pourquoi nous ne présenterons pas de mesures de temps CPU pour un nombre plus élevé de processeurs.

4.3.2 Efficacité de la parallélisation 2D

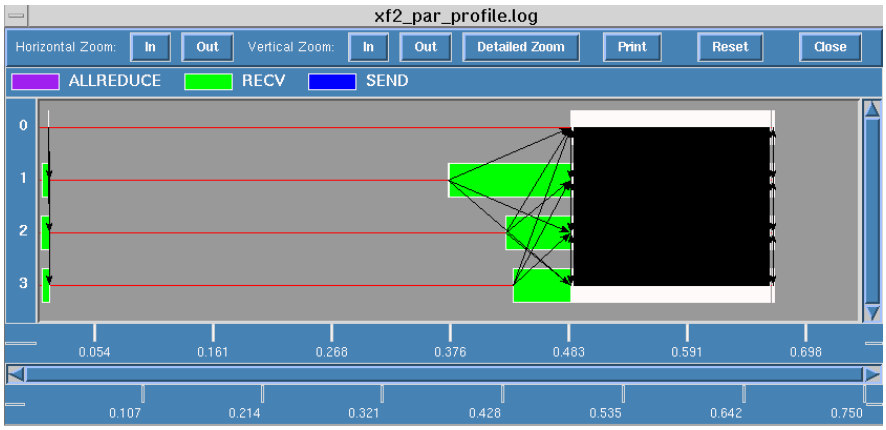
On peut visualiser la manière dont se déroule l'exécution parallèle par exemple à l'aide de l'utilitaire *upshot*. On représente sur la figure 8, les opérations parallèles effectuées par quatre processeurs lors du premier incrément de la simulation de l'écrasement d'un petit maillage de 120 noeuds. Cet utilitaire traite un fichier généré par un exécutable compilé en mode non optimisé. La simulation a été effectuée pendant une utilisation courante du calculateur aussi, les mesures de temps de calcul sont-elles perturbées et la figure est-elle uniquement indicative. La figure 8(a) montre l'exécution globale du logiciel. Au tout début le processeur 0, "maître", distribue les tâches aux autres processeurs placés en attente. Ils effectuent alors les assemblages éléments finis de manière concurrentielle, pour ensuite se synchroniser et communiquer les restrictions du second membre et du préconditionneur bloc diagonal. Ils abordent alors la résolution itérative parallèle. La figure 8(b) est un zoom effectué sur la résolution itérative. Celle-ci se décompose en des communications globales (MPI_ALL_REDUCE) relatives au calcul de produits scalaires et nécessitant une synchronisation de tous les processeurs et, en des envois et réceptions de messages, deux à deux, à la fin de chaque produit matrice/vecteur.

Premier cas test: cas linéaire

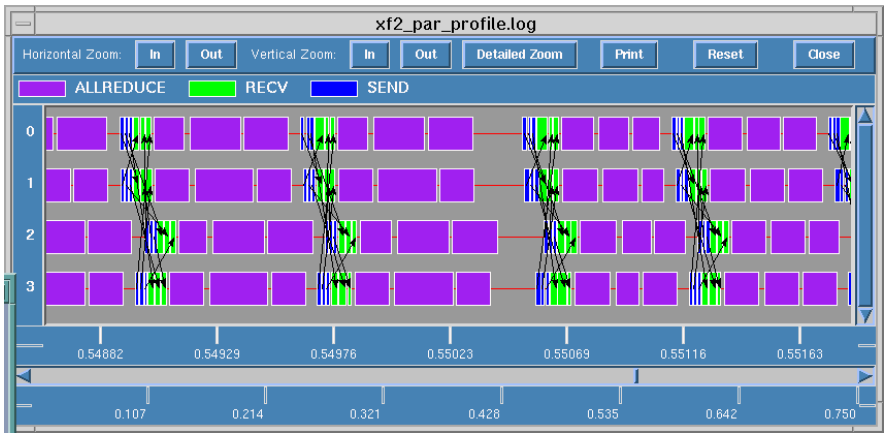
Le premier cas test étudié est présenté sur la figure 9. Il correspond à l'écrasement d'un cylindre, de 8 cm de rayon et 4 cm de hauteur maillé de manière régulière et contenant 6780 noeuds (113×60). Des partitions en deux, quatre et huit sous-domaines sont effectuées en "crayons", selon des droites parallèles à l'axe des ordonnées. Ces partitions sont parfaitement équilibrées en nombre d'éléments et en nombre de noeuds. Elles minimisent le nombre d'interfaces, un processeur ayant au plus deux voisins. Nous avons utilisé deux outillages différents et considéré une rhéologie newtonienne. Le premier outil est un tas plat, ainsi tous les processeurs ont exactement la même charge de travail lors de l'assemblage éléments finis. Le second est un outil arrondi présenté sur la figure 9. Pour cet outil des événements de contact auront lieu uniquement sur les domaines situés aux extrémités. Nous avons effectué 10 incréments de calcul et reporté les temps mesurés dans le tableau 1 pour l'écrasement entre tas plats et, dans le tableau 2 pour l'écrasement avec l'autre outil arrondi.

	séquentiel	2 Proc.	4 Proc.	8 Proc.
$Nbnoe$	6780	2×3420	4×1740	8×900
CPU	787 s (168 s)	468.6 s	220.9 s	107.3 s
Accélération	1	1.7	3.6	7.3
Efficacité	100 %	84 %	89 %	91 %
Accélération absolue Solveur Direct: 64 s	0.08	0.14	0.29	0.60

TAB. 1 – *Efficacité du solveur parallèle pour 10 incréments de calcul et une rhéologie newtonienne, sur un maillage structuré de 6780 noeuds* (113×60) *et une décomposition en crayons. Entre parenthèses temps séquentiel avec le préconditionneur* **Ic**. *Écrasement entre tas plats.*



(a) Premier incrément de calcul. Initialisations, assemblage éléments finis et inversion du système linéaire (en noir)



(b) Zoom sur l'inversion du système linéaire

FIG. 8 – *Visualisation de l'exécution parallèle avec l'utilitaire upshot. Les flèches matérialisent les communications.*

Nbnoe	séquentiel 6780	2 Proc. 2×3420	4 Proc. 4×1740	8 Proc. 8×900
CPU	877 s (201 s)	526 s	246 s	120 s
Accélération	1	1.7	3.6	7.3
Efficacité	100 %	83.3 %	89 %	91 %
Accélération absolue Solveur Direct: 64 s	0.07	0.12	0.26	0.53

TAB. 2 – *Efficacité du solveur parallèle pour 10 incréments de calcul et une rhéologie newtonienne, sur un maillage structuré de 6780 noeuds* (113×60) *et une décomposition en crayons. Entre parenthèses temps séquentiel avec le préconditionneur* **Ic**. *Écrasement par un outil arrondi.*

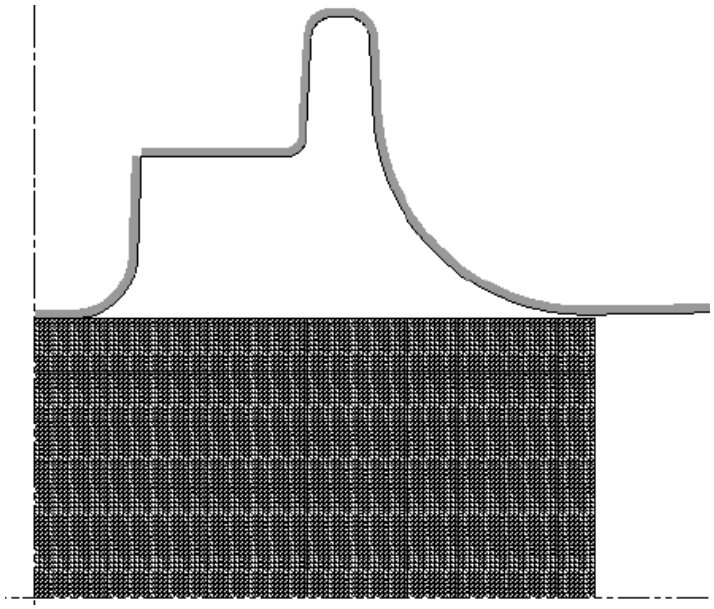


FIG. 9 – *Premier cas test: maillage structuré de 6780 noeuds*

Ces mesures amènent plusieurs remarques:

- Les résultats obtenus pour les deux types d'outillages sont similaires. L'efficacité parallèle et les accélérations obtenues sont très satisfaisantes bien que un peu décevantes sur deux processeurs.
- L'influence des calculs supplémentaires dus au contact sur l'efficacité parallèle est faible.
- L'influence du contact sur l'accélération absolue est significative. Celui-ci dégrade le conditionnement des systèmes linéaires, ce qui augmente légèrement le coût de la résolution itérative mais pas celui de la résolution directe.
- L'accélération absolue est catastrophique. Sur huit processeurs, le solveur parallèle est encore deux fois plus lent qu'en séquentiel avec le solveur direct. Cela vient en grande partie du fait que l'on ne profite pas en Newtonien, de l'optimisation du solveur non-linéaire.
- L'augmentation de l'efficacité avec le nombre de processeurs est étonnante, le comportement contraire est en général observé. Nous étudierons des explications de ce phénomène à la fin de ce chapitre.
- Sur huit processeurs MINRES avec le préconditionneur diagonal est plus rapide qu'en séquentiel avec le préconditionneur **Ic**.

Second cas test: Filage Non-Newtonien 5757 noeuds:

On reprend le problème de filage arrière présenté aux chapitres précédents. On considère les 6 premiers incréments de calcul pour deux rhéologies viscoplastique. La première est faiblement non-linéaire avec $m = p = 0.75$, tandis que la seconde et fortement non-linéaire avec $m = p = 0.15$. Des résolutions tronquées (*cf.* paragraphe 4.3 page 55), permettant une optimisation du solveur non-linéaire, sont effectuées uniquement pour la seconde rhéologie, celles-ci n'étant pas efficaces dans le premier cas. Le solveur itératif effectue environ 27800 itérations pour 19 itérations de Newton-Raphson dans le premier cas (tableau 3) et, 46700 itérations pour 41 itérations non linéaires dans le second cas (tableau 4).

	1 Proc	2 Proc	3 Proc	4 Proc.	6 Proc.	8 Proc.
CPU	573 s	332 s	217 s	159 s	103 s	81 s
Accélération	1	1.7	2.6	3.6	5.6	7.1
Efficacité	100%	86 %	88 %	90 %	92 %	88 %
Accélération absolue Direct 277 s	0.48	0.83	1.27	1.74	2.68	3.42

TAB. 3 – *Efficacité du solveur parallèle pour un problème de filage arrière sur un maillage non-structuré de 5757 noeuds* $m = p = 0.75$

	1 Proc	2 Proc	3 Proc	4 Proc.	6 Proc.	8 Proc.
CPU	508 s	295 s	192 s	141 s	92 s	73 s
Accélération	1	1.7	2.6	3.6	5.5	6.95
Efficacité	100%	86 %	88 %	90 %	92 %	87 %
Accélération absolue Direct 527 s	1.0	1.8	2.7	3.7	5.7	7.2

TAB. 4 – *Efficacité du solveur parallèle pour un problème de filage arrière sur un maillage non-structuré de 5757 noeuds* $m = p = 0.15$

Ces résultats confirment les observations effectuées sur le premier cas test:

- L'efficacité parallèle ainsi que les accélérations sont très bonnes et, dans les trois cas envisagés, les performances du solveur parallèle sont remarquablement stables.
- L'accélération absolue est excellente dans le cas fortement non-linéaire, mais elle reste faible dans l'autre cas.
- La baisse de l'efficacité entre six et huit processeurs peut aisément être attribuée aux synchronisations et donc à l'équilibrage des charges de calcul. De ce point de vue les résultats obtenus sur six processeurs reflètent alors une partition de très bonne qualité.

Évolution de l'efficacité:

Pour essayer de comprendre cette évolution, nous avons repris le dernier cas de filage arrière non linéaire $m = p = 0.15$ (tableau 4) et placé des horloges entre les principales opérations de la résolution. Ces horloges sont basées sur la fonction MPI_WTIME [74]. On constate que l'ajout de ces horloges perturbe les mesures, de l'ordre de 5% de variation par rapport au temps de calcul mesuré sans. On peut donc se demander si l'incertitude liée à la mesure du temps CPU n'est pas un des facteurs dominants. Toutefois, nous observons malgré ces perturbations la même tendance générale. Les mesures effectuées sont reportées sur le tableau 5. Le coût des étapes d'assemblage éléments finis est relativement stable, compris entre 4 et 5% du temps total, et n'est pas considéré ici. L'assemblage et l'inversion du préconditionneur bloc diagonal est quant à lui négligeable (inférieur à 0.1 %).

Np	Temps total MINRES	Efficacité	Produits scalaires	Produits matrice/vecteur	Précond. diagonal
1	485 s	100 %	101 s (21 %)	258 s (53 %)	53 s (11 %)
2	273 s	88 %	74 s (27 %)	137 s (50 %)	31 s (11 %)
3	175 s	92 %	50 s (28 %)	90 s (51.7 %)	19.5 s (11 %)
4	130 s	93 %	39 s (29 %)	67 s (51.7 %)	14.7 s (11 %)
6	83 s	97 %	30 s (36 %)	41 s (49 %)	9.9 s (12 %)
8	66 s	92 %	25 s (37 %)	32 s (49 %)	7.5 s (11%)

TAB. 5 – *Temps passé dans les différentes opérations de la résolution itérative pour le cas de filage arrière avec* $m = p = 0.15$ *également présenté au tableau 4*

Il semble qu'il y ait toujours une augmentation de l'efficacité entre deux et six processeurs. Cela est ici clairement dû à un effet de cache: plus le nombre de domaines augmente, plus la taille des vecteurs et des matrices diminue, et donc plus la proportion de termes directement accessibles en mémoire cache de premier niveau croît. En revanche, la proportion de temps passé dans les produits scalaires augmente avec le nombre de processeurs. Cela vient des synchronisations globales entre les processeurs. On peut aisément prévoir que pour un nombre encore plus élevé de processeurs, les produits scalaires deviendront l'opération la plus coûteuse de la résolution itérative. Ce problème est classique lorsque l'on utilise un nombre important de processeurs. Il est difficile à résoudre et nécessite en général de choisir d'autres méthodes de résolution.

Influence de la décomposition:

Pour évaluer l'influence de la décomposition, nous avons comparé le comportement du solveur parallèle sur les deux partitions présentées au paragraphe 4.1 (figure 6, page 87). La première partition (figure 6(a)), que nous noterons (a), est équilibrée mais ne minimise pas les communications. La seconde (figure 6(b)), que nous noterons (b), est équilibrée avec un nombre total de noeuds en interface plus faible: 180 pour la partition (a) et 123 pour (b). Nous avons simulé l'écrasement des maillages, par un outil plat de manière à ce que les événements de contacts ne perturbent pas cette étude, et effectué 15 inversions de systèmes linéaires avec une précision de 1.10^{-6}. Nous avons alors constaté un écart de 1% entre les deux simulations.

Partition	Temps total	Assemblages	Temps MINRES	Produits scalaires	Produits matrice/vecteur	Précond. diagonal
(a)	238.0 s	4.46 s	230.2	66.7 s	118.6 s	25.7
(b)	235.6 s	4.46 s	227.9	65.4 s	117.3 s	25.2

TAB. 6 – *Temps passé dans les opérations de la résolution itérative pour le cas de filage arrière avec* $m = p = 0.15$ *présenté dans le tableau 4 et pour deux partitions de maillage différentes*

Les temps d'assemblage des matrices éléments finis sont équivalents. La différence se fait dans la résolution itérative et principalement lors des produits matrice/vecteur où le volume des sommes aux interfaces est plus important pour la première partition. On note aussi une légère augmentation du coût des produits scalaires et du préconditionnement en raison des calculs redondants plus nombreux. L'influence de la décomposition reste toutefois négligeable, pour le moins dans ce cas. Nous verrons au chapitre suivant que cette influence est plus grande sur le solveur avec factorisations incomplètes parallèles. Ces observations vont dans le même sens que celles réalisée pas *S. Marie* dans sa thèse [57].

5 Conclusions

Les résultats obtenus montrent le grand intérêt de l'approche SPMD, tout d'abord développée au laboratoire dans le code FORGE3®, puis adoptée ici. Nous avons mis en évidence l'efficacité et la scalabilité de la méthode, avec par exemple des accélérations de 7 sur 8 processeurs. La méthode semble aussi peu dépendante de la qualité de la décomposition, aussi longtemps que celle-ci est bien équilibrée. Cela constitue un gage de robustesse pour un code de calcul simulant de grandes déformations sur des problèmes industriels complexes. C'est une méthode parallèle optimale au sens où le nombre d'itérations du solveur ne dépend, ni des partitions, ni du nombre de domaines. Nous sommes ainsi capables d'obtenir des accélérations importantes, par rapport au solveur direct séquentiel initial, sur des problèmes fortement non-linéaires et des maillages de 5000 noeuds ou plus. Toutefois, les performances ne sont plus aussi bonnes sur des problèmes de petite taille où le solveur direct est bien adapté et très rapide, ni pour les problèmes quasi-linéaires (mais cela tombe en dehors du cadre de la simulation du forgeage pour lequel le matériau a toujours un comportement fortement non-linéaire). De plus, il existe des procédés très sensibles aux petites perturbations et dont la simulation ne permet pas d'utiliser les résolutions tronquées lors de la résolution du problème non-linéaire. Un exemple est le laminage d'anneaux (voir chapitre V).

Par ailleurs, le coût des produits scalaires, comme pour beaucoup d'autres méthodes parallèles, pose le problème de la scalabilité de la méthode sur un grand nombre de domaines. Comme nous l'avons vu, ceux-ci sont susceptibles de devenir l'opération la plus coûteuse de l'algorithme. Pour diminuer leur poids, une solution consiste à diminuer le nombre d'itérations et, à augmenter le coût de chaque itération par des opérations ne nécessitant pas de synchronisations globales, mais deux à deux comme pour les sommes aux interfaces.

Les factorisations incomplètes ont donné de très bons résultats en séquentiel avec des accélérations d'un facteur trois par rapport aux préconditionneurs diagonaux. Elles permettent de plus une diminution importante du nombre d'itérations, d'un facteur dix. La perspective d'une version parallèle de ce préconditionneur est donc très attrayante. Nous allons donc nous intéresser au chapitre suivant aux possibilités de développer un préconditionneur parallèle basé sur les factorisations incomplètes.

Chapitre IV

Préconditionnement incomplet parallèle

DANS ce chapitre, nous nous proposons d'étudier l'extension au calcul parallèle des préconditionneurs par factorisations incomplètes, présentés au chapitre II (paragraphe 3, page 39). La nature séquentielle des factorisations incomplètes rend leur parallélisation délicate et source de moindre efficacité, que ce soit lors de leur construction ou lors de l'étape de préconditionnement proprement dite. Ce problème a motivé de nombreux travaux en raison de l'efficacité de ce préconditionneur et du recours de plus en plus courant au calcul parallèle [49, 16, 6, 28, 80, 61]. Les difficultés rencontrées ont conduit beaucoup de chercheurs à reformuler les préconditionneurs, ou encore à s'intéresser à d'autres types plus adaptés aux calculateurs parallèles. Pour une revue complète des diverses approches du préconditionnement parallèle dans le cadre des méthodes itératives de Krylov nous renvoyons le lecteur aux articles de *Y. Saad* [70, 71] et de *Duff, Vorst et al* [27].

1 Position du problème

Soit D un préconditionneur diagonal ou bloc diagonal. La méthode parallèle SPMD présentée au chapitre précédent nous permet de **calculer explicitement la restriction** de ce préconditionneur à chaque sous-domaine. En effet dans le cas de matrices diagonales ou blocs diagonales, la restriction de l'inverse est égale a l'inverse de la restriction *i.e.* :

$$\left(D^{-1}\right)_{|\Omega_i} = \left(D_{|\Omega_i}\right)^{-1} \tag{1}$$

Ainsi, une fois le préconditionneur calculé, l'étape de préconditionnement proprement dite ne nécessite plus aucune communication entre les processeurs, les restrictions des vecteurs et du préconditionneur étant connues par chacun. En théorie le préconditionneur parallèle est identique au préconditionneur séquentiel. En pratique, les erreurs d'arrondis se traduisent par de légères différences. Celui-ci ne dépend donc ni de la décomposition ni du nombre de domaines.

Le préconditionnement par factorisation incomplète quant a lui, consiste en deux opérations toutes deux séquentielles par nature:

1. Le calcul, une seule fois par résolution, du préconditionneur *i.e.* de la factorisation LDL^T où L est une matrice triangulaire inférieure.
2. L'étape de préconditionnement proprement dite consistant en la résolution par descente/remontée de systèmes de la forme: $z = (LDL^T)^{-1}y$.

Une implémentation directe de ces opérations mène à des récursions, nécessitant de rassembler des données globales sur un processeur, qui ne sont pas adaptées aux architectures parallèles. Le principal problème réside dans les descentes/remontées. Pour une inconnue donnée, elles nécessitent que les inconnues précédentes (en terme de numérotation), ou suivantes (dans le cas des remontées), aient déjà été calculées. Le même problème se rencontre pour le calcul des coefficients des matrices triangulaires. Toutefois, ce calcul n'ayant lieu qu'une seule fois par résolution, on pourrait accepter de l'effectuer au prix d'une perte d'efficacité, voire de le localiser sur un seul processeur qui redistribuerait ensuite l'information aux autres.

Une première approche consiste à extraire le maximum de parallélisme de la méthode séquentielle *sans changer* ses propriétés numériques. Dans ce cas, on cherche à changer l'ordre des calculs de manière à rendre le calcul d'un ensemble d'inconnues indépendant d'un autre. Ces méthodes sont bien adaptées pour des matrices issues d'un schéma par différences finies mais ne permettent généralement pas d'atteindre un degré suffisant de parallélisme. Pour plus de détails nous renvoyons à [27]. Des approches utilisant des décompositions algébriques des matrices en blocs ont aussi été envisagées par *Saad* [70] ou encore par *Sturler* [24].

Une seconde approche consiste à modifier la méthode séquentielle pour augmenter le degré de parallélisme possible, ce qui *peut changer* ses propriétés numériques et mener à de nouveaux types de préconditionneurs. Comme les performances des méthodes par factorisations incomplètes sont dépendantes de la numérotation des lignes et des colonnes (*Duff et Meurant* [26], *Eijkhout* [29]) on peut trouver des stratégies de numérotation améliorant les performances de ces méthodes et/ou augmentant leur degré de parallélisme. D'un autre coté, certaines numérotations peuvent mener à un haut niveau de parallélisme mais au prix d'une détérioration importante du taux de convergence. On effectue ces renumérotations en faisant appel, par exemple, à des techniques de coloriage. L'exemple le plus classique est celui à partir de deux couleurs (généralement référencées comme rouge et noir) apparaissant naturellement lors de la discrétisation par différences finies centrées d'un Laplacien 2D ou 3D. Nous renvoyons à [27, 16, 70] et aux références citées pour de plus amples détails.

Une troisième possibilité consiste tout simplement à choisir une autre méthode plus adaptée au calcul parallèle. On citera entre autre:

- les méthodes de préconditionnement polynomiaux (*Saad* [69]) (ou plus généralement les méthodes de développement en séries [79]) et les méthodes de préconditionnement SOR (Gauss Seidel relaxé) [25]. Ces méthodes réduisent la proportion de produits scalaires et d'opérations sur les vecteurs au prix d'une augmentation du nombre d'itérations.
- les méthodes de préconditionnement éléments par éléments (EBE) (*Dayde et al.* [23]) qui font appel à des techniques de coloriage pour le calcul parallèle,
- les techniques SPAI (SParse Approximate Inverse) d'approximation des inverses des matrices creuses, qui ouvrent des perspectives intéressantes, mais peu de résultats concrets ont été obtenus à ce jour [17].

Généralement les préconditionneurs basés sur les renumérotations, le multi-coloriage ou encore sur les développements en séries offrent un haut niveau de parallélisme mais au prix d'une détérioration du taux de convergence. Ils sont donc surtout intéressants sur des machines à grains moyens ou fins. Dans le cadre d'un parallélisme à grain grossier, il semble que ce soit les méthodes par domaines qui offrent les meilleurs compromis. Celles-ci, à l'inverse des méthodes algébriques, permettent de prendre en compte de l'information relative à la physique du problème.

1.1 Méthodes par domaines

Di Brozzolo et Y. Robert [14, 36] ont étudié la résolution parallèle de systèmes non-symétriques par la méthode du gradient conjugué CGS (Conjugate Gradient Squared) et par celle du résidu minimal généralisé GMRES(k). Soit $\mathcal{A}$ la matrice globale, $\overline{P}$ le préconditionneur incomplet séquentiel et P le préconditionneur parallèle. A partir d'une approche algébrique, les auteurs ont proposé deux manières de construire un préconditionneur $ILU(0)$ parallèle, en considérant ou non de légers recouvrements entre les sous-domaines:

1. **Méthode globale**: Un seul processeur calcule $\overline{P}$, la factorisation incomplète de la matrice $\mathcal{A}$. Celui-ci redistribue ensuite la restriction de $\overline{P}$ à chacun des sous-domaines attaché à un processeur.
2. **Méthode locale**: Chaque processeur calcule indépendamment et concurrentiellement une factorisation incomplète locale à partir des sous-matrices $\mathcal{A}_{|\Omega_i}$ (*i.e.* les restrictions de $\mathcal{A}$ au sous-domaine i)(*cf.* figure 1). Dans ce cas, l'information ne transite plus du haut vers le bas de la matrice, comme lors d'une construction séquentielle, mais seulement du haut vers le bas de chaque bloc. Ainsi, un haut niveau de parallélisme est obtenu au prix de la construction d'un préconditionneur de moindre qualité : l'information contenue dans celui-ci devient locale et non plus globale.

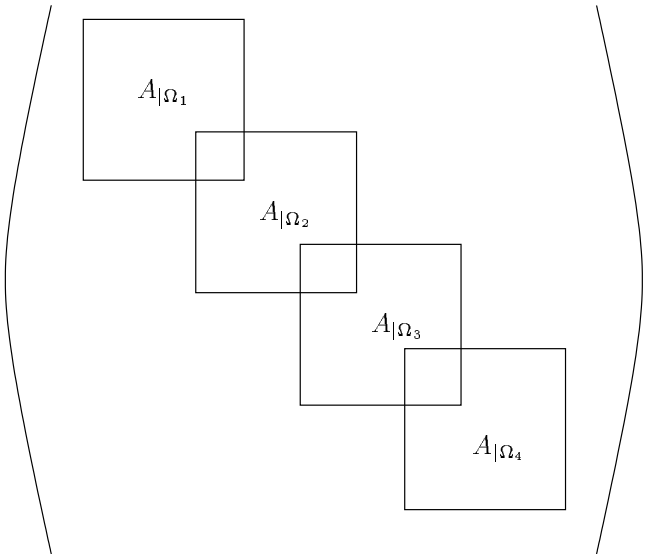


FIG. 1 – *Schématisation de la partition de la matrice*

Ces deux approches permettent de construire Np factorisations $L_i D_i L_i^T$ sur chacun des sous-domaines. L'étape de préconditionnement proprement dite:

$$x = P^{-1} w$$

est alors effectuée de la manière suivante:

1. Découpage de w en Np vecteurs $w_{|\Omega_i}$
2. Résolution en parallèle des Np systèmes : $L_i D_i L_i^T x_i = w_{|\Omega_i}$
3. La solution globale x est alors égale à x_i à l'intérieur de chaque domaine et à la moyenne des valeurs aux interfaces ou dans les zones de recouvrement.

Quelle que soit la méthode de calcul des préconditionneurs locaux P_i, le préconditionneur

parallèle peut s'écrire de la manière suivante :

$$P^{-1} = \sum_{i=1}^{Np} Q_i^T P_i^{-1} R_i \tag{2}$$

où Q_i est la matrice associée à l'opérateur de moyenne aux interfaces et sur les recouvrements éventuels. Elle peut encore être la matrice associée à un opérateur de régularisation [51], R_i est l'opérateur de restriction au sous-domaine Ω_i, défini au chapitre précédent.

Les auteurs observent alors, sans surprise, que le préconditionneur global est meilleur que le local. Ils notent toutefois, qu'en général, cela ne suffit pas à compenser la perte d'efficacité engendrée par le calcul sur un seul processeur pendant que les autres sont en attente. Le préconditionneur local est donc préférable.
Une étude sur la pertinence d'avoir des recouvrements entre les blocs est effectuée. Elle montre que de petits recouvrements entre les blocs sont nécessaires, même si la convergence de la méthode de résolution peut-être obtenue sans recouvrements. En effet, ceux-ci permettent d'obtenir un nombre d'itérations proche de celui obtenu en séquentiel. Dans [36], les auteurs préconisent des recouvrements de la taille de la largeur de bande locale, de manière à ce qu'aucun coefficient de la factorisation globale ne soit "laissé de côté" lors de la factorisation locale.

Remarque :

Dans notre cas où les systèmes linéaires sont symétriques, nous ne pouvons utiliser cette méthode ainsi que d'autres semblables rencontrées dans la bibliographie. En effet, la présence de la matrice Q_i, rend le préconditionneur P non-symétrique et n'est donc pas envisageable avec MINRES. Le passage à un solveur non symétrique, type GMRES, serait source d'une augmentation importante du temps de résolution. On peut s'interroger sur la possibilité de symétriser ce type de préconditionneur, en considérant par exemple $Q_i \equiv R_i$.

2 Préconditionneurs ILU(0) locaux

Nous considérons le préconditionneur additif de Schwarz de la forme:

$$P^{-1} = \sum_{i=1}^{Np} R_i^T P_i^{-1} R_i \tag{3}$$

où P_i est le préconditionneur local au ième sous-domaine que nous voulons construire à partir de factorisations incomplètes.

En pratique, il est important de noter qu'avec ce type de préconditionneur, il sera nécessaire d'effectuer une somme aux interfaces après chaque préconditionnement, comme dans le cas d'un produit matrice/vecteur. En effet, nous ne disposerons sur chaque sous-domaine que de la contribution de celui-ci au préconditionneur global.

Le problème consiste donc à construire, **à un coût acceptable**, un préconditionneur parallèle **optimal**, ou du moins quasi optimal, *i.e.* dont les performances ne dépendent ni de la partition ni du nombre de domaines. Dans le cadre de systèmes globalement couplés ou diffusifs, comme le sont les EDP elliptiques, le problème réside dans la recherche d'un compromis entre le parallélisme, qui préfère la localité de l'information, et le besoin d'un bon taux de convergence qui nécessite de transmettre une information globale. Les techniques de

sous-structuration, que nous avons évoquées au chapitre précédent, apportent des éléments de réponse à ces problèmes. En introduisant une grille grossière, elles autorisent des transferts d'information entre des domaines diamétralement opposés et peuvent aboutir à des préconditionneurs optimaux ou quasi optimaux. Mais ces méthodes ne sont applicables que dans un second temps, une fois qu'un préconditionneur déjà performant a été mis en place. Nous pourrons donc les envisager, soit pour améliorer le préconditionneur que nous allons développer, soit pour compléter le préconditionneur diagonal, si les méthodes envisagées s'avéraient trop coûteuses pour être appliquées.

Le premier préconditionneur de type (3) que nous avons étudié consiste à construire localement les préconditionneurs P_i sans communication, à partir de l'information immédiatement disponible sur chaque processeur. On construit donc des factorisations incomplètes ou complètes des matrices locales $\mathcal{A}_{\Omega_i}$ (matrices des contributions du sous-domaine Ω_i à la matrice globale $\mathcal{A}$). Soit:

$$\begin{array}{rcll} P_i^1 & = & L_i^1 D_i^1 (L_i^1)^T \approx \mathcal{A}_{\Omega_i} & \text{factorisations incomplètes} \\ P_i^2 & = & L_i^2 D_i^2 (L_i^2)^T = \mathcal{A}_{\Omega_i} & \text{factorisations complètes} \end{array} \tag{4}$$

Dans les deux cas MINRES ne converge pas pour nos problèmes de forgeage. Toutefois, d'autres essais numériques nous ont montré qu'avec cette approche MINRES converge parfaitement si les matrices sont à diagonale strictement dominante. On peut remarquer que si l'on réinterprète cette approche en terme des équations de départ, cela reviendrait, si l'on ne communique pas, à construire un préconditionneur adapté pour un problème à frontière libre aux interfaces.
Un second essai infructueux a consisté à effectuer les factorisations ci-dessus pour ensuite remplacer la matrice diagonale D_i^1 (ou D_i^2) par la restriction de la diagonale globale *i.e.* par $D_{|\Omega_i}$. Nous avons alors observé le même comportement que précédemment.

Ces essais numériques montrent que l'information contenue dans les seules matrices $\mathcal{A}_{\Omega_i}$ est insuffisante pour nous permettre de construire un préconditionneur parallèle satisfaisant. Ainsi, de l'information doit transiter entre les processeurs. Notons $C_{\Omega_i}^j$ l'information relative à la j^{eme} interface du domaine Ω_i et qui doit être transmise au processeur voisin (selon l'interface j). La méthode SPMD est alors de la forme présentée dans l'algorithme IV .1. La question à laquelle nous allons tenter d'apporter une réponse est: quel est(sont) le(s) meilleur(s) choix pour $C_{\Omega_i}^j$? On peut en effet envisager d'apporter cette information supplémentaire d'au moins deux manières:

- On peut suivre l'approche de *Di Brozzolo et Y. Robert* citée précédemment. Ils raisonnent à partir de la **matrice globale** et travaillent sur les matrices $\mathcal{A}_{|\Omega_i}$, *i.e.* les restrictions à chaque sous-domaine de la matrice $\mathcal{A}$. L'information supplémentaire est alors apportée par les recouvrements des sous matrices (*cf.* figure 1). Cette méthode peut s'interpréter comme une approche algébrique du problème. L'inconvénient principal est qu'elle nécessite un nombre important de communications pour compléter les matrices sur chaque sous-domaine, sans compter les opérations redondantes si l'on effectue des recouvrements entre les sous-maillages. L'implémentation est délicate et une décomposition par noeuds est dans ce cas préférable.
- On peut raisonner à partir des **matrices locales** $\mathcal{A}_{\Omega_i}$: les contributions des sous-domaines à la matrice globale. Cette approche reste très proche des éléments finis. Dans le cadre d'une décomposition par éléments, l'implémentation est aisée. Des communications additionnelles sont nécessaires pour enrichir l'information disponible. On utilise alors l'information la plus aisément accessible, qui est contenue dans les blocs diagonaux des matrices locales.

Algorithme IV .1 Une itération de Newton-Raphson de la méthode SPMD par éléments avec préconditionnement additif de Schwarz basé sur des factorisations incomplètes (équation (3))

Pour tous les processeurs faire :

Assemblage éléments finis :

Calcule H_{Ω_i}, G_{Ω_i} *matrice et second membre locaux*

Pour $j = 1$ à $Nbinterfaces$ faire :

Assemble $C^j_{\Omega_i}$ *Contributions du domaine aux préconditionneurs*

Le contenu des matrices $C^j_{\Omega_i}$ sera précisé dans les paragraphes suivant

fin Pour

Calcul de la restriction du second membre à chaque sous-domaine

$$G_{|\Omega_i} = \sum_{j=1}^{N_p} R_i G_{\Omega_j}$$ *Somme aux interfaces*

Construction du préconditionneur :

Communications: Envoi et réception des C^j

$H_{\Omega_i} = H_{\Omega_i} + C_j$ *stockage de l'information dans la matrice*

$P_i = L_i D_i L_i^T \approx H_{\Omega_i}$ *construction de la factorisation incomplète locale*

$H_{\Omega_i} = H_{\Omega_i} - C_j$ *libère la matrice des informations supplémentaires*

Résolution du système linéaire $H\Delta V = -G$: MINRES algorithme (II .1)

Produits matrice/vecteur *équation (33)*

Produits scalaires *équation (35)*

Préconditionnement: Calcul de $z = P^{-1}r$

Calcul concurrentiel de $z_{\Omega_i} = P_i^{-1} r_{|\Omega_i}$

Sommes aux interfaces: $z_{|\Omega_i} = \sum_{j=1}^{N_p} R_i z_{\Omega_j}$

Réactualisations concurrentielles

Fin résolution

réactualisations

fin Pour

fin algorithme

2.1 Préconditionnement à partir des restrictions $\mathcal{A}_{|\Omega_i}$

Notons **LDL** le préconditionneur additif de Schwarz construit sur chaque sous-domaine Ω_i à partir de la factorisation complète de $\mathcal{A}_{|\Omega_i}$:

$$\textbf{LDL} \qquad P_i = L_i D_i L_i^T = \mathcal{A}_{|\Omega_i} \tag{5}$$

et, notons **LDL(0)** le préconditionneur construit sur chaque sous-domaine Ω_i à partir de la factorisation incomplète de $\mathcal{A}_{|\Omega_i}$.

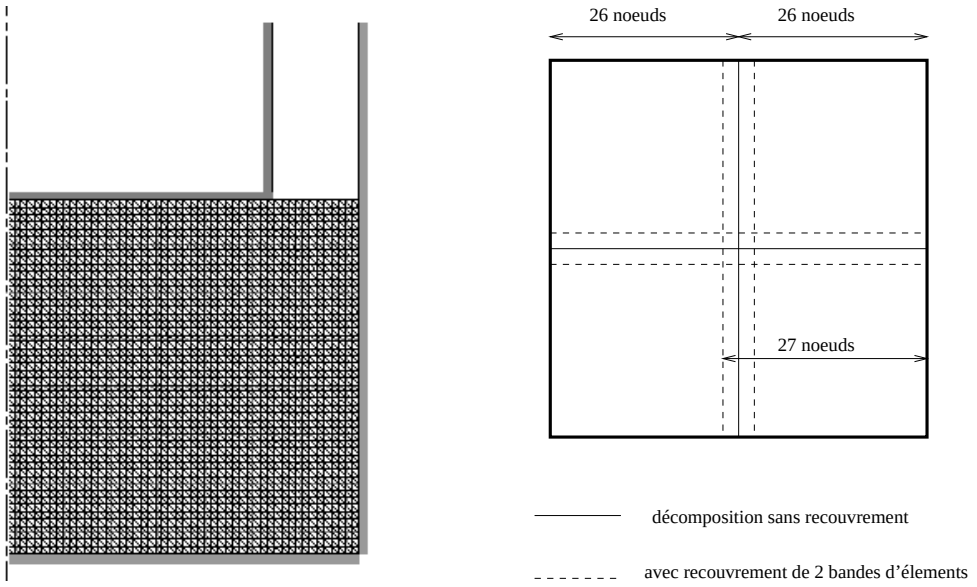
$$\textbf{LDL(0)} \qquad P_i = L_i D_i L_i^T \approx \mathcal{A}_{|\Omega_i} \tag{6}$$

Dans le cadre d'une décomposition par éléments, le coût d'assemblage de ces préconditionneurs est élevé. Ils nécessitent en effet la connaissance, sur chaque domaine, de la restriction de la matrice globale. Ceci engendre des communications de la taille de la matrice d'interface qui sont en $\mathcal{O}\left(\frac{1}{2}(N^{S_i})^2\right)$. Dans le cadre d'une décomposition par noeuds, le coût en communications est négligeable, de l'ordre de celui de la méthode avec préconditionneur diagonal, mais au prix d'une augmentation des opérations redondantes (aussi en $\mathcal{O}\left(\frac{1}{2}(N^{S_i})^2\right)$) lors de l'assemblage des matrices élémentaires.

A priori, le préconditionneur **LDL** n'est pas envisageable et, nous sert surtout à illustrer notre propos en nous affranchissant partiellement des incertitudes liées aux factorisations incomplètes (existence et unicité de la factorisation). En effet, sa construction est beaucoup plus coûteuse que celle de **LDL(0)**, de l'ordre de $121\,Nbnoe^2$ au lieu de $486Nbnoe$ (*cf.* chapitre II , page 45). L'étape de préconditionnement á chaque itération est elle aussi beaucoup plus chère en $93Nbnoe^{1.5}$ au lieu de $435Nbnoe$ pour **LDL(0)**. Ainsi pour un maillage de 1000 noeuds et, en ne considérant que les descentes/remontées, il faut que MINRES préconditionné par **LDL** effectue 6.7 fois moins d'itérations qu'avec **LDL(0)** pour avoir un coût équivalent.

2.1.1 Cas test FIL2601:

On considère une nouvelle fois le problème de filage arrière présenté au chapitre I , page 21, pour un maillage structuré de 2601 noeuds (51 noeuds par cotés) présenté sur la figure 2(a). Ce cas test illustre bien les tendances observées sur d'autres cas étudiés. Nous le dénommerons FIL2601. Le matériau est supposé Newtonien. Les partitions sont effectuées en carrés ou en rectangles, selon des droites parallèles et/ou perpendiculaires à l'axe des abscisses. La partition en 2 domaines est faite de 2 rectangles de 26×52 noeuds, celle en 4 domaines en 4 carrés de 26×26 noeuds, celle en 6 domaines de 4 rectangles de 18×26 noeuds et deux rectangles de 17×26 noeuds et celle en 9 domaines de 4 carrés de 18×18 noeuds, 2 rectangles de 17×18 noeuds, 2 rectangles de 18×17 noeuds et un carré de 17×17 noeuds.
Dans le cas où les sous-domaines se recouvrent, les recouvrements suivent naturellement la géométrie des interfaces construites (*cf.* exemple figure 2(b)).



(a) Cas test FIL2601

(b) Décomposition en 4 sous-domaines équilibrés

FIG. 2 – *Cas test FIL2601: carré de* 51×51 *noeuds. Exemple de décomposition en 4 sous-domaines*

2.1.2 Comportement des préconditionneurs avec ou sans recouvrements

Nous avons étudié le comportement de ces deux préconditionneurs sur le cas test FIL2601, sans recouvrement et avec des recouvrements de deux bandes d'éléments. Les résultats sont donnés dans le tableau 1. Nous présentons sur la figure 5 le comportement de MINRES pour les préconditionneurs **LDL** et **LDL(0)** pour des partitions en 2, 4, 6 et 9 domaines sans recouvrement. Sur la figure 6, nous étudions le comportement de MINRES avec les préconditionneurs **LDL** et **LDL(0)** pour une décomposition en 4 sous-domaines avec différents niveaux de recouvrements.

	Sans recouvrement			Avec recouvrement de deux bandes d'éléments		
Nombre de domaines	Itérations **LDL(0)**	Itérations **LDL**	N_{noe}^S	Itérations **LDL(0)**	Itérations **LDL**	N_{noe}^S
1	141	1	0	141	1	0
2	179	63	51	167	37	153
4	207	118	101	200	68	297
6	243	147	151	211	88	441
9	264	164	200	217	105	576

TAB. 1 – *Évolution du nombre d'itérations pour les préconditionneurs* **LDL** *et* **LDL(0)** *en fonction du nombre de domaines et de la présence ou de l'absence de recouvrements. Cas FIL2601* ($\epsilon_{it} = 1.10^{-6}$). N_{noe}^S *représente le nombre total de noeuds appartenant à plus d'un sous-domaine.*

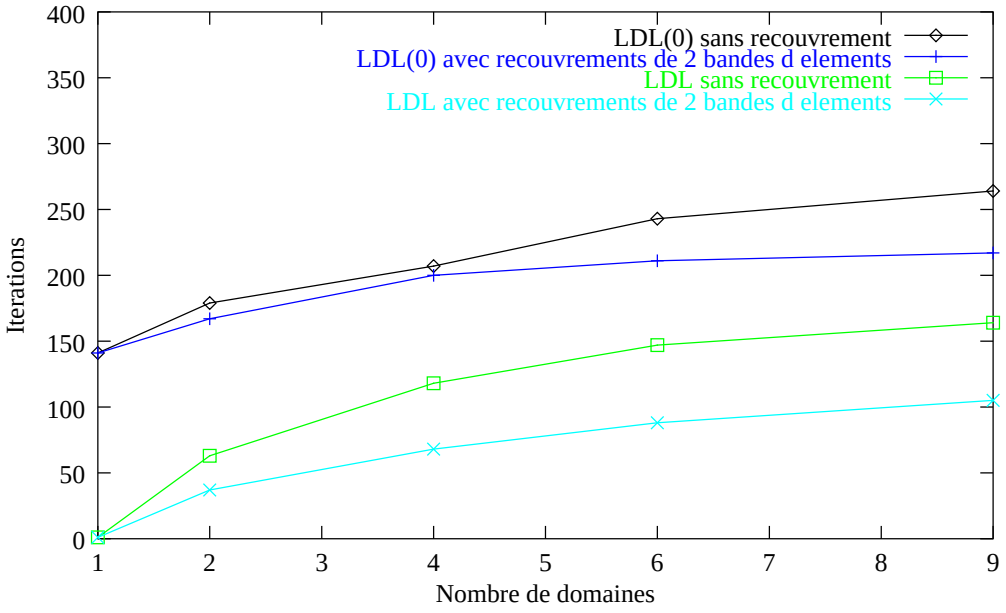


FIG. 3 – *Évolution du nombre d'itérations pour les préconditionneurs* **LDL** *et* **LDL(0)** *en fonction du nombre de domaines*

Ces résultats montrent que:

MINRES converge avec les deux préconditionneurs, que des recouvrements soient ou non effectués.
Le préconditionneur **LDL** nécessite moins d'itérations que **LDL(0)** surtout si des recouvrements sont effectués. On remarque toutefois que l'écart, en nombre total d'itérations, diminue quand le nombre de domaines augmente. Ainsi, dans le cas avec recouvrements, cet écart passe d'un facteur 4.5 sur deux domaines à un facteur 2. sur 9 domaines. On note que même sur deux domaines, cet écart n'est pas suffisant pour justifier l'utilisation des factorisations complètes. Le préconditionneur **LDL(0)** est le meilleur.

Le comportement de MINRES préconditionné est similaire à celui observé en séquentiel.
Nous observons sur les figures 5 et 6 une convergence "chahutée" de MINRES, similaire à celle remarquée en séquentiel au chapitre II . Le préconditionneur parallèle (3) non symétrique défini positif a le même comportement que les préconditionneurs non s.d.p. séquentiels.

Dépendance au nombre de domaines:
Que des recouvrements soient pratiqués ou non, le nombre d'itérations dépend de manière flagrante du nombre de domaines. Plus le nombre de domaines est élevé moins le préconditionneur est efficace. Pour **LDL(0)** sans recouvrement, on passe ainsi de 141 itérations en séquentiel à 264 sur 9 domaines, soit une augmentation de 85%. Toutefois, l'écart le plus important vient du passage du solveur séquentiel au solveur parallèle avec une augmentation de 27% du nombre d'itérations entre 1 et 2 processeurs. Il progresse ensuite plus lentement. On notera que, même si le nombre d'itérations double presque entre 1 et 9 domaines, l'efficacité parallèle n'est pas vraiment compromise, puisqu'une accélération proche de 4.5 est estimée. Le nombre d'itérations semble aussi croître linéairement avec le nombre de noeuds à l'interface (*cf.* figure 4).

Les recouvrements améliorent la convergence.
Les recouvrements, quelle que soit leur taille, contribuent à diminuer le nombre d'itérations de la méthode parallèle et améliorent donc la qualité du préconditionneur.

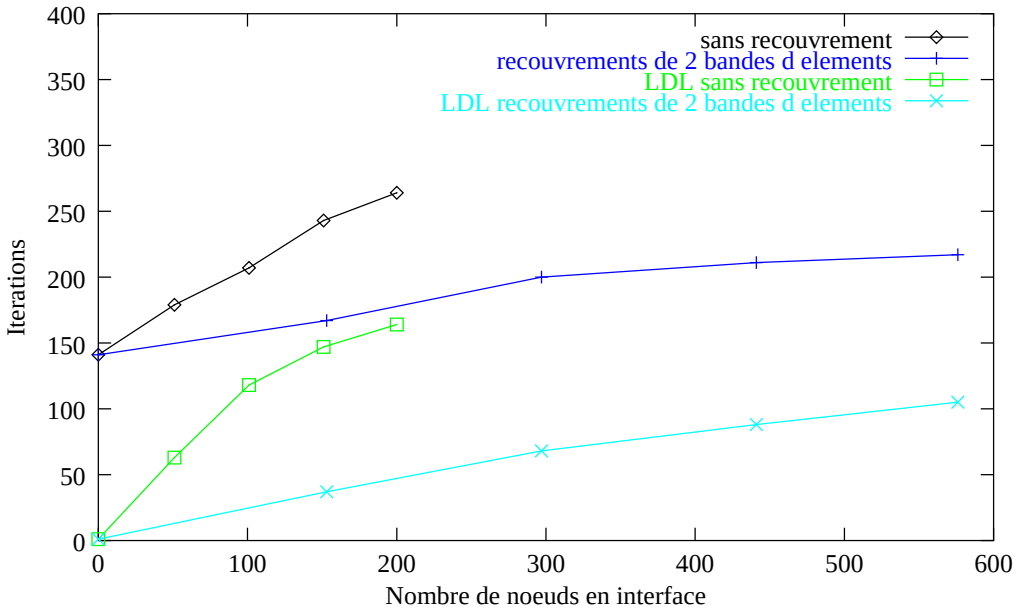
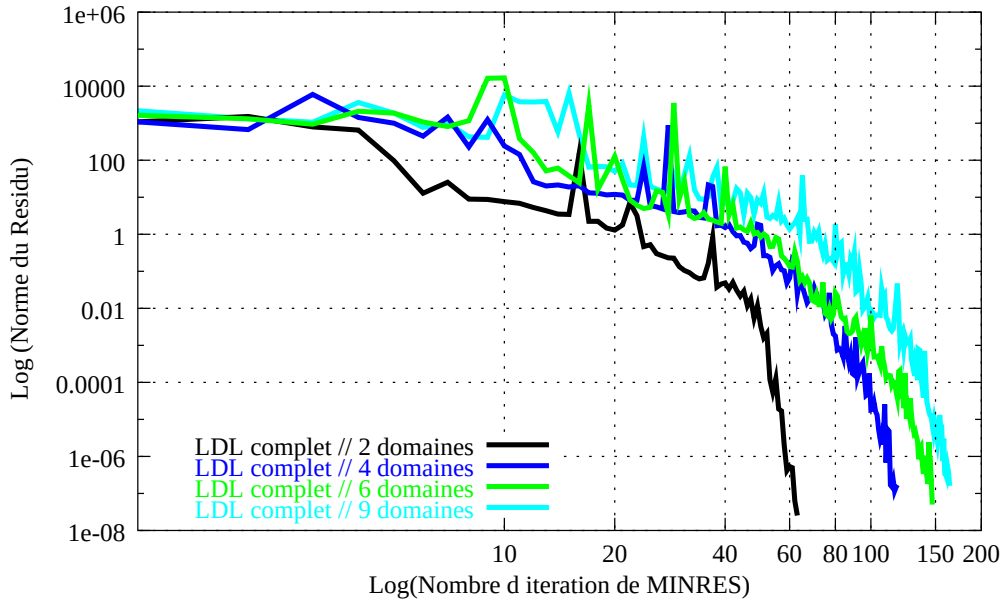


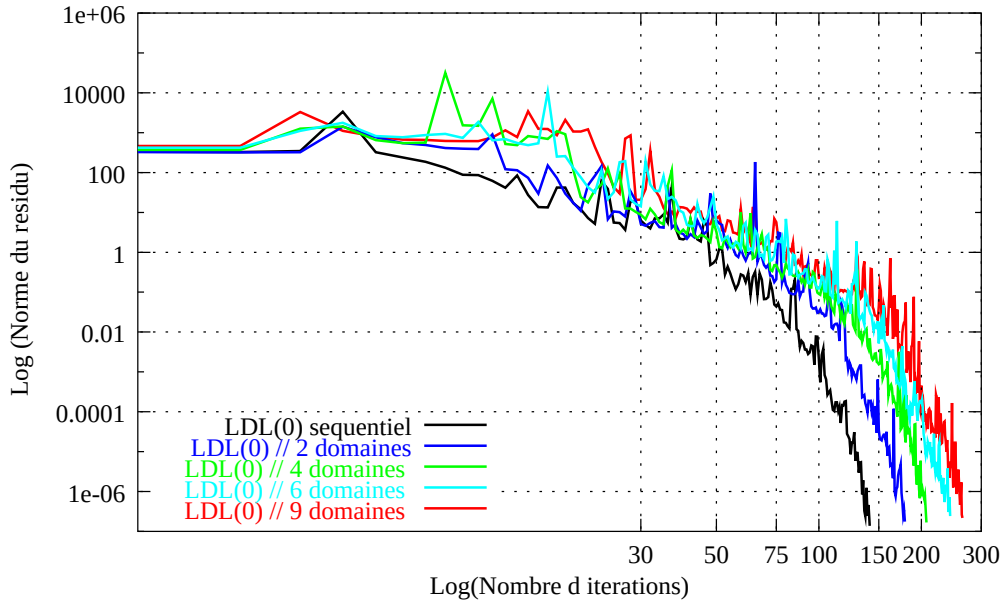
FIG. 4 – *Évolution du nombre d'itérations pour les préconditionneurs en fonction de N_S nombre total de noeuds situés à l'interface.*

Il apparaît sur les figures 5 et 6 que le nombre d'itérations séquentielles est une limite inférieure au nombre d'itérations que l'on peut espérer atteindre en parallèle. On vérifie d'ailleurs qu'un recouvrement complet du domaine mène, pour le préconditionneur **LDL**, à la convergence en une itération (que ce soit sur 2 ou 4 domaines). De même, pour le préconditionneur **LDL(0)**, on obtient un nombre d'itérations très proche de celui obtenus en séquentiel. Dans le cas de recouvrements de deux bandes d'éléments (tableau 1), l'influence sur le préconditionneur **LDL** semble être stable, réduisant de 40 % le nombre d'itération, tandis que pour le préconditionneur **LDL(0)** celle-ci augmente avec le nombre de sous-domaines. Le préconditionneur **LDL** est aussi beaucoup plus sensible à la taille des recouvrements (*cf.* figure 6) que le préconditionneur **LDL(0)**. Pour celui-ci, il semble qu'un simple recouvrement d'une bande d'éléments est suffisant. Par ailleurs, la partition par noeuds fait l'effet d'une sorte "d'anti-recouvrement". En effet, le nombre d'itérations semble augmenter plus rapidement avec le nombre de domaines que pour la partition par éléments. Si l'on effectue une partition par noeuds avec un recouvrement d'une bande d'éléments on retrouve alors des résultats très proches de ceux obtenus avec une partition par éléments sans recouvrement.

La construction de préconditionneurs additif de Schwarz de la forme (3) à partir de factorisations complètes ou incomplètes des matrices $\mathcal{A}_{|\Omega_i}$ aboutit à des préconditionneurs assez satisfaisants. Que des recouvrements soient ou non pratiqués, MINRES converge en un nombre acceptable d'itérations pour les deux types de préconditionneurs. Bien que **LDL(0)** soit efficace sans que des recouvrements soient effectués, il semble qu'un recouvrement d'une bande d'éléments soit utile, surtout si le nombre de domaines est élevé.

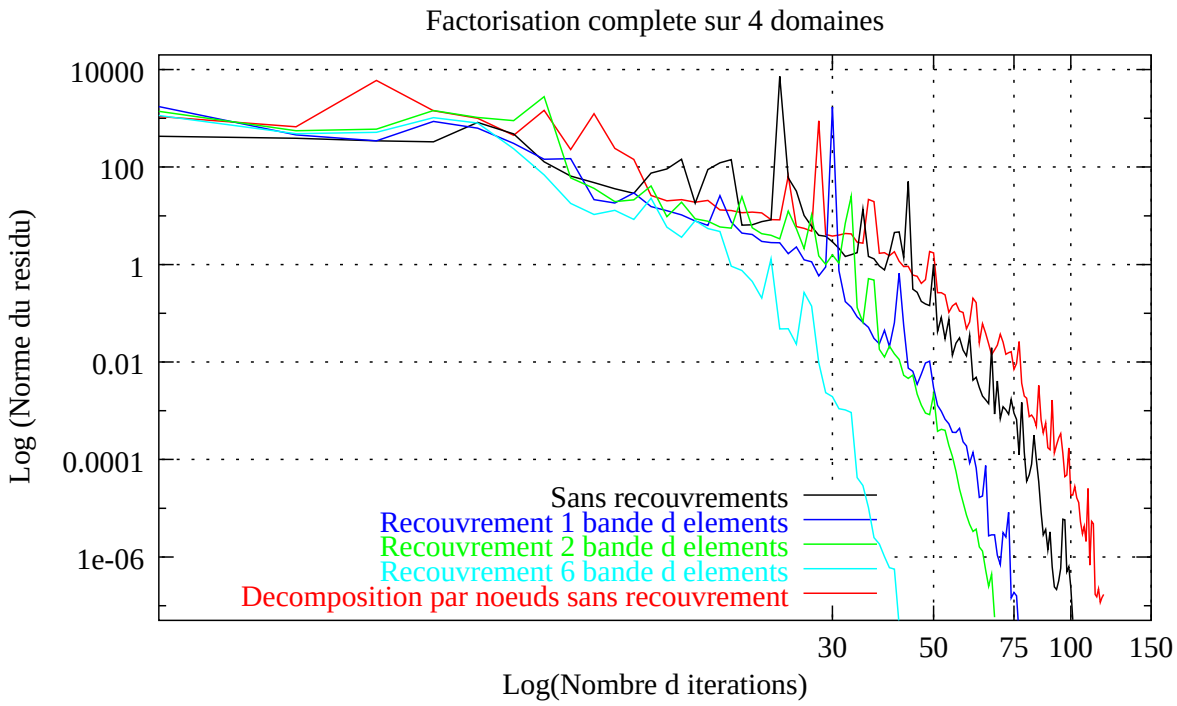


(a) Préconditionneur **LDL** parallèle

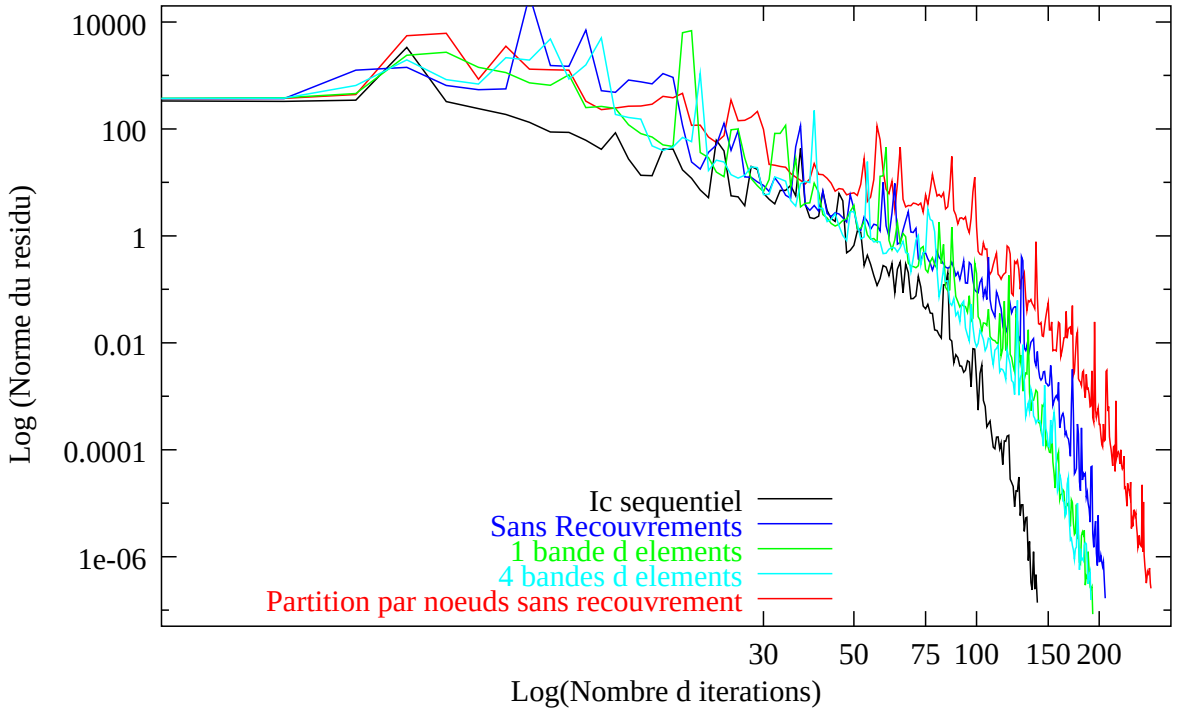


(b) Préconditionneur **LDL(0)** parallèle

FIG. 5 – *Préconditionnement par factorisations complètes et incomplètes sans recouvrement pour une décomposition par éléments*



(a) Préconditionneur **LDL** sur 4 domaines



(b) Préconditionneur **LDL(0)** sur 4 domaines

FIG. 6 – *Influence des recouvrements sur les préconditionneurs. Cas FIL2601.*

2.2 Préconditionnement à partir des matrices $\mathcal{A}_{\Omega_i}$

Nous venons de voir que nous étions capables de construire un préconditionneur parallèle assez satisfaisant sur la base des matrices $\mathcal{A}_{|\Omega_i}$. Toutefois, l'information contenue sur les restrictions des matrices n'est pas aisément accessible d'un processeur à l'autre. On cherche donc à trouver un préconditionneur aussi efficace, mais qui se construise à partir des matrices $\mathcal{A}_{\Omega_i}$. Dans ce paragraphe nous allons considérer 2 types de factorisations:

- $\overline{\mathbf{IC}}$: Les factorisations sont obtenues en complétant la matrice aux interfaces, soit juste sur sa diagonale, soit sur les blocs diagonaux (BD) 3×3 que nous notons BD_{Ω_i} et $BD_{|\Omega_i}$.

$$\overline{\mathbf{IC}}\ \mathbf{Diag} \quad P_i = \mathcal{A}_{\Omega_i} - D_{\Omega_i} + D_{|\Omega_i} \tag{7}$$

$$\overline{\mathbf{IC}}\ \mathbf{Bloc\ Diag} \quad P_i = \mathcal{A}_{\Omega_i} - BD_{\Omega_i} + BD_{|\Omega_i} \tag{8}$$

- $\overline{\mathbf{IC}}(\omega)$: Les factorisations sont obtenues à partir des précédentes, relaxées aux interfaces. L'idée est d'augmenter le poids des termes d'interfaces.

$$\overline{\mathbf{IC}}(\omega)\ \mathbf{Diag} \quad P_i = \mathcal{A}_{\Omega_i} - D_{\Omega_i} + W_i\, D_{|\Omega_i} \tag{9}$$

$$\overline{\mathbf{IC}}(\omega)\ \mathbf{Bloc\ Diag} \quad P_i = \mathcal{A}_{\Omega_i} - BD_{\Omega_i} + W_i\, BD_{|\Omega_i} \tag{10}$$

Où W_i est une matrice diagonale de dimension $N_i \times N_i$ telle que :

$$W_i = \begin{pmatrix} \omega_i(1) & & & (0) \\ & \omega_i(2) & & \\ & & \ddots & \\ (0) & & & \omega_i(N_i) \end{pmatrix} \tag{11}$$

où: $$\omega_i(r) = \begin{cases} 1 & \text{si} \quad r \in \Omega_i \backslash S_i \\ \alpha\, NbDom_r & \text{si} \quad r \in S_i \end{cases} \qquad \forall 1 \leq r \leq N_i$$

$NbDom_r$ est le nombre de domaines connectés au ddl r
N_i est le nombre de noeuds du domaineΩ_i
α est un coefficient de relaxation

Un parallèle entre cette fonction de pondération et la moyenne aux interfaces effectuée entre autres par *Di Brozzolo et al.* [14] peut se faire. En effet, une manière d'approcher la moyenne aux interfaces que nous ne pouvons faire, consiste à multiplier les termes de la matrice par le nombre de domaines connectés au domaine courant. Ainsi, lors de l'inversion du préconditionneur, chaque terme de l'interface sera pondéré par le nombre de domaines connectés à celui-ci.
Ce procédé a de plus l'avantage très important, si les ω_i sont assez grands, de procurer des préconditionneurs locaux inversibles et donc des factorisations praticables !
Nous choisissons pour le moment $\alpha = 1$, et étudierons au paragraphe 2.2.2 l'influence de sa valeur.

Par défaut, si rien n'est précisé, ce sont les versions par blocs qui sont utilisées car a priori plus performantes.

2.2.1 Comparaison des préconditionneurs $\overline{\mathbf{IC}}$ et $\overline{\mathbf{IC}}(\omega)$

Pour un matériau Newtonien, on effectue 7 incréments de filage arrière avec un maillage non structuré de 5757 noeuds. La partition en 3 domaines de ce maillage est présentée sur

la figure 6(b), page 87. Le nombre d'itérations obtenues avec les différents préconditionneurs montrent que les factorisations relaxées $\overline{\mathbf{IC}}(\omega)$ sont de bien meilleurs préconditionneurs (voir tableau 2 et figure 7). De plus, l'écart est d'autant plus marqué que le nombre de domaines augmente. Cela tend à prouver que la méthode choisie pour renforcer la diagonale est une bonne manière de faire. On remarque que l'augmentation du nombre d'itérations se fait

Préconditionneur	2 Processeurs	3 Proc	4 Proc	8 Proc	12 Proc
$\overline{\mathbf{IC}}$**Diag**	2322	2364	2303	2805	3091
$\overline{\mathbf{IC}}$**Bloc Diag**	2220	2220	2261	2735	3476
$\overline{\mathbf{IC}}(\omega)$**Diag**	2085	2068	1999	2316	2787
$\overline{\mathbf{IC}}(\omega)$**Bloc Diag**	2182	2014	2019	2327	2766

TAB. 2 – *Itérations pour les 4 préconditionneurs: 1542 itérations en séquentiel. Cas de filage pour un maillage de 5757 noeuds.*

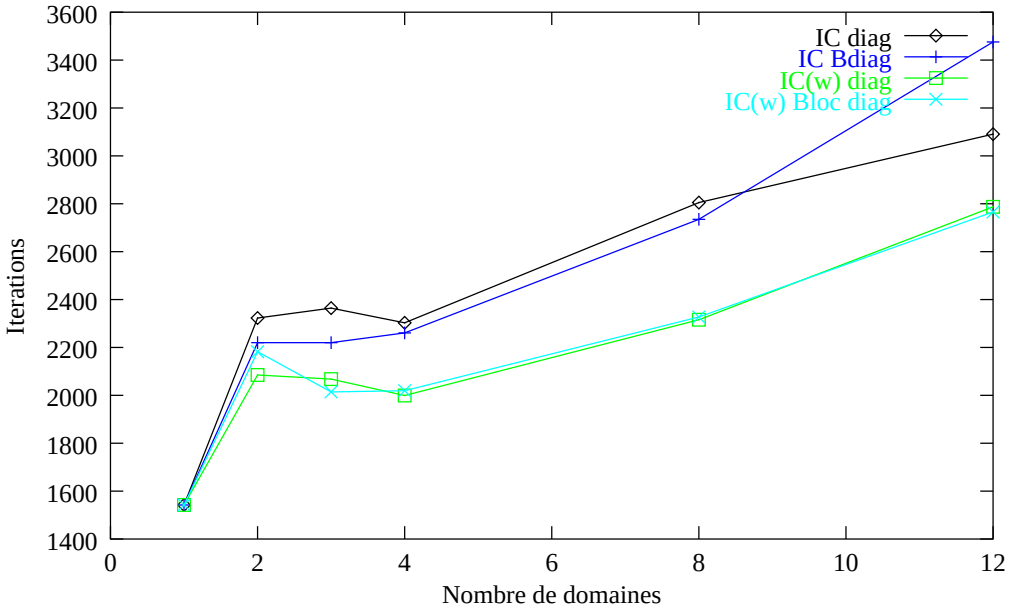


FIG. 7 – *Comportement des préconditionneurs par contributions sur un problème de filage arrière et pour un maillage de 5757 noeuds*

essentiellement lors du passage du séquentiel à deux processeurs. Ainsi, pour les préconditionneurs $\overline{\mathbf{IC}}(\omega)$ le nombre d'itérations augmente de 35 %, tandis qu'il ne progresse ensuite que de 27 % quand on passe de 2 à 9 processeurs. Sur cet exemple, on peut estimer une accélération du solveur parallèle de l'ordre de 1,5 sur 2 processeurs et de 5 sur 9 processeurs.

Il est aussi remarquable de voir que, sur cet exemple, entre 2 et 4 processeurs, les performances du solveur parallèle sont quasiment stables !

2.2.2 Choix du coefficient de relaxation

Sur un maillage non structuré de 340 noeuds, pour 35 itérations de Newton-Raphson, et sur 2 et 4 domaines, nous avons comparé le nombre total d'itérations de MINRES pour différentes valeurs du coefficient α variant entre $0.25 < \alpha < 5$. Les résultats sont présenté sur la figure 8.

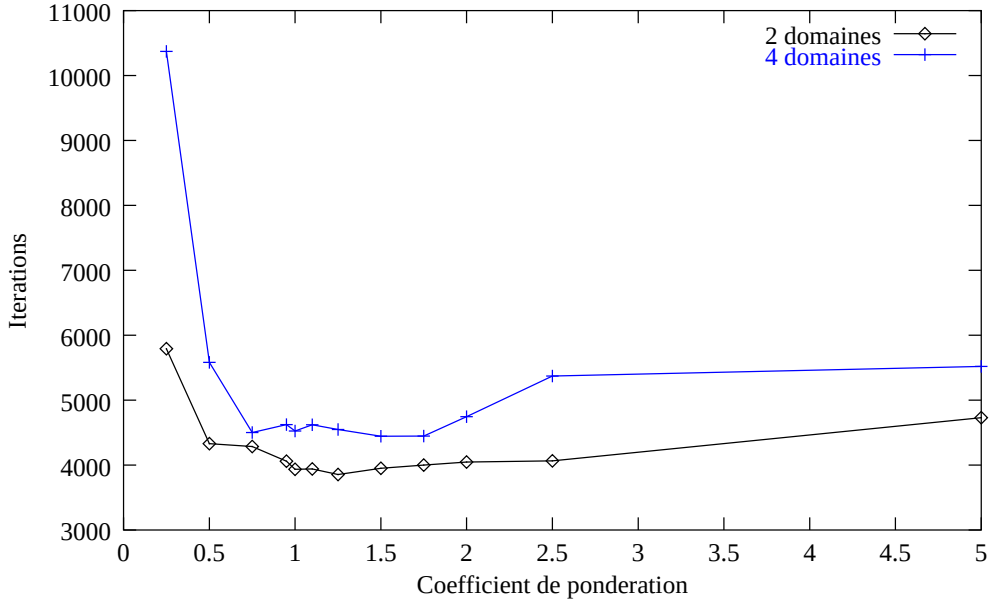


FIG. 8 – *Évolution du nombre d'itérations selon le facteur de relaxation aux interfaces*

Il apparaît que le coefficient α doit être choisi supérieur ou égal à un. On remarque une très faible dépendance du nombre d'itérations aux valeurs de α quand celui-ci varie entre un et deux. Il semble qu'une légère sur-relaxation soit favorable et, dans la suite nous utiliserons $\alpha = 1.25$.

2.3 Comparaison des préconditionneurs $\overline{\mathbf{IC}}(\omega)$ et LDL(0)

Le premier cas test sur lequel nous avons comparé les deux types de préconditionneurs est le cas FIL2601 présenté page 102. Nous avons considéré les cinq premiers incréments de calcul, avec $\epsilon_{it} = 1.10^{-6}$ fixé. Les résultats obtenus sont présentés dans le tableau 3.

Nombre de domaines	2	4	6	9
$\overline{\mathbf{IC}}(\omega)$**Diag**	809	876	950	1029
$\overline{\mathbf{IC}}(\omega)$**Bloc Diag**	775	888	965	1076
LDL(0)	815	976	1136	1242
LDL(0) avec recouvrements de 2 bandes d'éléments	760	917	987	1073

TAB. 3 – *Nombre total d'itérations pour les différents préconditionneurs pour 5 inversions de systèmes avec* $\epsilon_{it} = 1.10^{-6}$ *du cas test FIL2601*

Il est remarquable que, pour un coût de calcul bien moindre, le préconditionneur $\overline{\mathbf{IC}}(\omega)$ donne des résultats équivalents au préconditionneur LDL(0) utilisant une décomposition avec recouvrements des maillages.

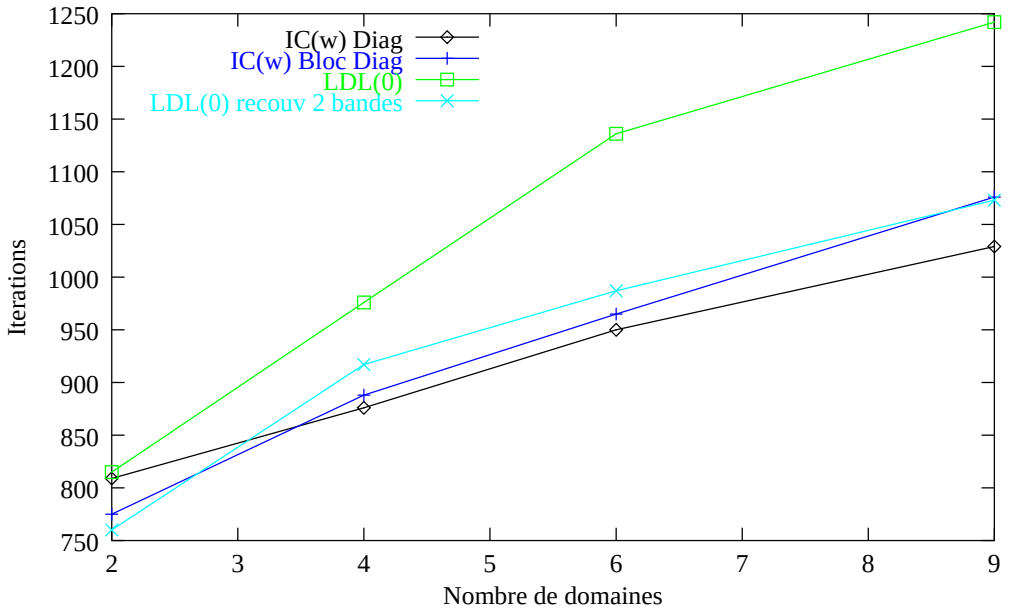


FIG. 9 – *Comparaison des préconditionneurs parallèles sur le cas FIL2601 pour 5 incréments Newtoniens. Évolution du nombre d'itérations avec le nombre de domaines*

Le second cas test que nous avons utilisé est le problème d'écrasement d'un maillage régulier de 6780 noeuds par un outil arrondi. Ce cas présenté également au chapitre précédent, figure 9, page 91. Les décompositions sont toujours effectuées en bandes. Nous avons utilisé deux rhéologies: une newtonienne et une viscoplastique $m = p = 0.15$ et considéré les quatre premiers incréments de calcul, soit 13 inversions de système dans le cas linéaire et 52 inversions dans le cas non-linéaire. Les résultats sont présentés dans le tableau 4.

	cas non-linéaire $m = p = 0.15$		cas linéaire $m = p = 1$	
Nombre de domaines	2	4	2	4
$\overline{\mathbf{IC}}(\omega)$**Bloc Diag**	7234 (139)	8277 (153)	4518 (347)	4126 (317)
$\overline{\mathbf{IC}}(\omega)$**Diag**	6978 (134)	8496 (151)	4027 (309)	3711 (285)
LDL(0)	14237 (302)	15594 (294)	3241 (249)	2916 (224)
LDL(0) recouv 2	14076 (287)	14464 (289)	2423 (186)	2108 (162)

TAB. 4 – *Nombre d'itérations de MINRES pour les différents préconditionneurs pour environ 52 inversions de systèmes dans le cas non-linéaire et 13 inversions dans le cas linéaire (entre parenthèses nombre moyen d'itérations par inversion). Second cas test.*

Pour la rhéologie newtonienne, les systèmes linéaires sont assez bien conditionnés. Il en résulte que le préconditionneur **LDL(0)** est généralement le meilleur. En non linéaire, les systèmes sont moins bien conditionnés. Dans ce cas, ce sont les préconditionneurs $\overline{\mathbf{IC}}(\omega)$ qui sont, et de beaucoup, les plus performants. Les méthodes par restriction semblent donc être beaucoup plus sensibles au conditionnement. L'influence de la qualité des décompositions est à noter. En effet, dans certains cas, le nombre d'itérations peut être moindre sur 4 domaines que sur 2. En général, on observe toutefois que la tendance est à l'augmentation du nombre d'itérations avec le nombre de domaines. Pour les problèmes, fortement non-linéaires auxquels nous sommes confrontés pour la simulation du procédé de forgeage, le préconditionneur $\overline{\mathbf{IC}}(\omega)$ est le meilleur choix. Il n'est pas trop coûteux à calculer et MINRES semble bien se comporter

lorsque les systèmes linéaires sont mal conditionnés.

2.4 Temps de calcul

Nous considérons à nouveau le cas de filage arrière sur un maillage de 5757 noeuds et pour une rhéologie viscoplastique $m = p = 0.15$. Des mesures de temps de calcul ont déjà été effectuées au chapitre précédent (tableau 4, page 92). Nous reportons les performances mesurées dans le tableau 5 pour le préconditionneur $\overline{\mathbf{IC}}(\omega)$ pour 6 incréments de calculs. Ces résultats sont délicats à interpréter. En effet, en raison des résolutions tronquées le nombre d'itérations de Newton-Raphson varie fortement avec les décompositions (voir fin de chapitre). On effectue ainsi en séquentiel 35 itérations avec le solveur direct et 25 avec MINRES tronqué. En parallèle nous en effectuons 35 sur 2, 4 et 6 processeurs, 34 sur 3 et 26 sur 8.

	1 Proc	2 Proc	3 Proc	4 Proc.	6 Proc.	8 Proc.
CPU $\overline{\mathbf{IC}}(\omega) BlocDiag$	134 s	150 s	90 s	65 s	50 s	26 s
$It_{moy}(MINRES)$	63	107	107	100	123	128
Accélération	1	0.89	1.49	2.1	2.68	5.15
Efficacité	100%	45 %	49 %	51 %	45 %	64 %
Accélération par rapport au solveur direct (527 s)	3.93	3.51	5.85	8.1	10.5	20.3
CPU **VipP**	508 s	295 s	192 s	141 s	92 s	73 s
Accélération obtenue par rapport à **VipP**	3.79	1.96	2.13	2.16	1.84	2.8

TAB. 5 – *Efficacité du solveur parallèle pour un problème de filage arrière sur un maillage non-structuré de 5757 noeuds* $m = p = 0.15$. It_{moy} *nombre moyen d'itérations de MINRES par inversion de système. Comparaison avec les temps de calcul du solveur parallèle avec le préconditionneur bloc diagonal* **VipP**.

Sur deux domaines, nous constatons une décélération par rapport au temps mesuré en séquentiel avec le préconditionneur **Ic**. La raison de cette contre-performance est due à la combinaison de l'augmentation du nombre moyen d'itérations du solveur, qui était attendue, et de celui du nombre d'itérations de Newton-Raphson, plus difficile à prévoir (et qui nécessite sans doute une adaptation des critères de troncature dans le cas du calcul parallèle). Le temps de calcul reste toutefois très inférieur à celui obtenu sur deux processeurs avec le préconditionneur **VipP**. À partir de trois domaines, les accélérations sont bien supérieures à 1, pour une efficacité relativement faible, principalement en raison des très bonnes performances séquentielles. Ainsi, sur quatre processeurs nous sommes deux fois plus rapides que précédemment avec le préconditionneur diagonal et presque six fois plus rapides que la méthode directe séquentielle. Nous obtenons d'excellents résultats sur huit processeurs où nous sommes 20 fois plus rapide que la méthode directe. De plus, le filage est plutôt un cas difficile à simuler et on peut s'attendre à de meilleures performances sur des problèmes de forgeage plus classiques. Nous renvoyons au chapitre suivant pour des exemples de ce type. L'évolution du nombre d'itérations de MINRES est aussi un point très positif. Comme nous l'avons observé sur les exemples précédents, un saut important a lieu du séquentiel au parallèle sur deux processeurs susceptible de compromettre la méthode sur deux processeurs. Pour un nombre plus élevé de processeurs, cette évolution est beaucoup plus douce ce qui nous assure des accélérations substantielles et nous permet d'obtenir une efficacité parallèle relativement stable.

Poids des calculs:

De la même manière qu'au chapitre précédent (tableau 4, page 92) et pour le cas test ci-dessus, nous avons mesuré le temps passé dans les diverses étapes de la résolution itérative.

NProc	Temps MINRES ($It_{moy}(MINRES)$)	Produits scalaires	produits matrice/vecteur	préconditionnement descentes/remontées
1	100 s (63)	7.7 (7.7 %)	20.7 (20.7 %)	62.0 (62 %)
2	110 s (107)	14.7 (13 %)	22.7 (21 %)	66.3 (60 %)
3	65.7 s (107)	9.45 (14 %)	14.8 (22.5 %)	38.0 (57 %)
4	45.7 s (100)	6.7 (15 %)	10.7 (23 %)	26.0 (57 %)
6	36.6 s (123)	6.48 (18 %)	8.5 (23 %)	20.4 (56 %)
8	19.9 s (128)	3.65 (18 %)	4.8 (24 %)	10.7 (53 %)

TAB. 6 – *Temps passé dans les différentes opérations de la résolution itérative pour le cas de filage arrière avec* $m = p = 0.15$ *présenté dans le tableau 5*

Pour le préconditionneur $\overline{\mathbf{IC}}(\omega)$, ce sont les descentes remontées qui s'avèrent être l'opération la plus coûteuse. On estime généralement leur coût comparable à celui d'un produit matrice/vecteur. Il apparaît que celui-ci est plutôt 1.5 fois supérieur. Leur part au temps total diminue avec l'augmentation du nombre de domaines tandis que celle des produits scalaires et des produits matrice/vecteur augmente. Le coût des produits scalaires double de un à deux processeurs en raison des synchronisations. On remarque toutefois que la part des produits scalaires au temps total est bien plus faible qu'avec le préconditionneur diagonal. Cela est dû à la forte diminution du nombre total d'itérations de MINRES. On peut espérer, si le nombre moyen d'itérations n'augmente pas trop vite avec le nombre de domaines, une meilleure scalabilité de la méthode pour un nombre élevé de domaines.

Influence de la décomposition

Comme au chapitre précédent (page 93), nous avons évalué l'influence de la partition. Les mêmes partitions (présentées page 87) ont été considérées. Les mesures sont reportées dans le tableau 7. Les temps d'assemblages correspondent à la construction des matrices et vecteurs (incluant les synchronisations), ainsi que le calcul du préconditionneur. Nous remarquons une influence plus importante de la partition que pour le préconditionneur diagonal: de l'ordre de 5 % du temps de calcul. Cela s'explique par une proportion plus grande des opérations nécessitant des communications (produit matrice/vecteur une fois par itération et descente/remontée deux fois par itérations). Le volume moindre de communications avec la partition de bonne qualité se traduit sur les temps nécessaires à ces deux opérations. L'écart entre les descentes/remontées et les produits matrice/vecteur est consistant puisque l'on retrouve un facteur 2.5. Nous attribuons la différence sur les produits scalaires aux calculs redondants plus importants sur la partition (a).

Partition	Temps total	Assemblages	Temps MINRES	Produits scalaires	Produits matrice/vecteur	descentes/ remontées
(a)	59.5 s	6.1 s	50.0	7.3 s	11.4 s	28.8
(b)	56.6 s	6.1 s	47.1	6.9 s	10.6 s	27.3

TAB. 7 – *Temps passé dans les opérations de la résolution itérative pour le cas de filage arrière avec* $m = p = 0.15$ *présenté dans le tableau 5 pour deux partitions distinctes du domaine.*

3 Conclusions

Dans ce chapitre, nous avons envisagé deux manières de construire un préconditionneur additif de Schwarz basé sur des factorisations incomplètes de matrices locales. La méthode basée sur l'utilisation des blocs diagonaux de la matrice globale pour compléter l'information contenue dans les matrices locales semble la plus intéressante. La méthode itérative avec le préconditionneur $\overline{\mathbf{IC}}(\omega)$ converge en général en moins d'itérations et, est moins sensible à l'augmentation du nombre de sous-domaines. Elle est de plus beaucoup plus simple à programmer que la première méthode envisagée.

La parallélisation permet d'accélérer les calculs sur tous les problèmes où en séquentiel nous avons pu rendre MINRES compétitif, c'est à dire à partir de 3000 noeuds, voire 1500 dans les cas favorables. Un second point positif réside dans la plus grande scalabilité, a priori, de la méthode retenue. En effet, en réduisant la proportion du temps passé dans les produits scalaires, nous avons réduit de manière significative le nombre de synchronisations. Ainsi, pour une taille de problème donnée, le nombre limite de processeurs à partir duquel les produits scalaires deviennent le facteur limitant de l'efficacité parallèle s'est accru par rapport à la méthode initiale.

Toutefois, le problème de la scalabilité a été seulement transposé vers celui de l'augmentation du nombre d'itérations de MINRES avec le nombre de domaines. Nous sommes en effet convaincus qu'à partir d'un nombre donné de domaines celle-ci sera trop importante pour être compensée par l'exécution parallèle. Ce phénomène couramment rencontré dans le cadre des méthodes de décomposition de domaine, est lié à la contradiction entre l'utilisation d'une information locale pour le calcul parallèle et le caractère global du problème physique. Même si nous rencontrons ce problème de manière moins accentuée, puisque la localisation de l'information ne se restreint qu'au préconditionneur, il fixe une limite au nombre maximum de processeurs utilisables. Pour échapper à cet écueil, il serait nécessaire d'améliorer le préconditionnement de manière à transmettre plus vite l'information entre des domaines éloignés. Une solution réside dans les techniques de sous-structuration développées dans le cadre des méthodes de décomposition de domaine. L'ajout, sur chaque processeur, d'une grille grossière globale sur laquelle on peut construire un bon préconditionneur permet de transmettre des informations de basse fréquence entre des domaines diamétralement opposés alors que MINRES se charge des informations de haute fréquence. Leur mise en oeuvre, de manière géométrique ou algébrique, sur des maillages non structurés est toutefois assez complexe.

Chapitre V

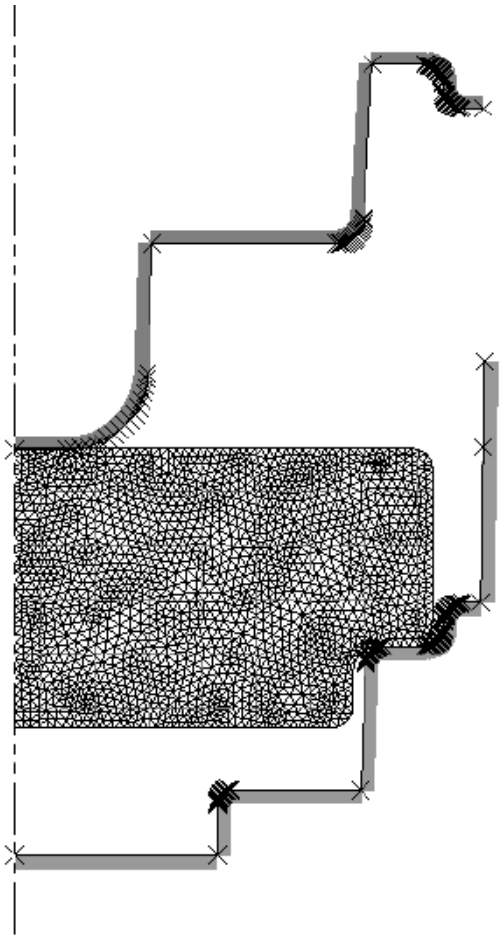
Tests et applications à des cas industriels

1 Applications à des problèmes industriels 2D

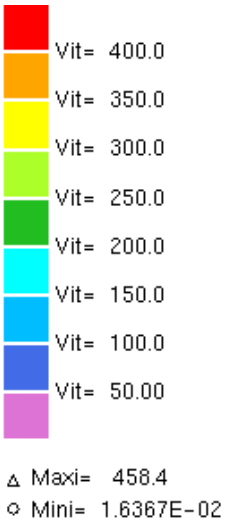
Pour évaluer les performances de notre solveur parallèle, nous avons retenu deux cas tests représentatifs du forgeage. Le premier que nous présentons consiste en une mise en forme finale. L'écoulement qui lui est associé est complexe et, la majorité des simulations ont été effectuées pour une formulation non stabilisée qui, comme nous l'avons vu, ne facilite ni la convergence de Newton-Raphson, ni celle de MINRES. Il nous permet de valider les choix effectués et de mettre en évidence la scalabilité de notre méthode ainsi que sa robustesse. Le second correspond au forgeage d'une préforme et se caractérise par un écoulement plus simple; il nous permettra mettre en évidence les très bonnes accélérations mesurées et l'efficacité de notre solveur parallèle.

1.1 Cas de FORD WERKE

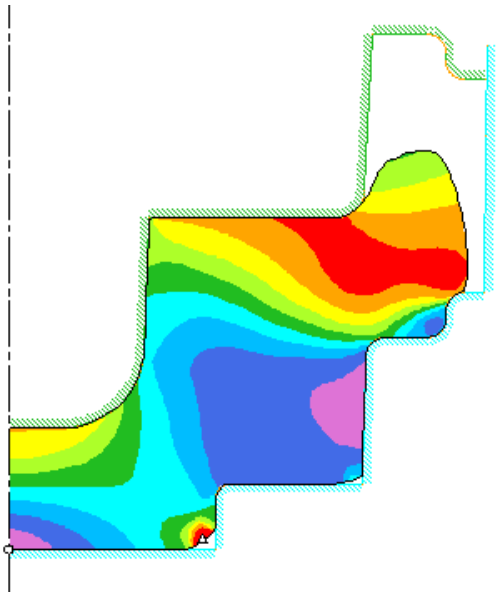
Ce cas test, a été proposé par l'industriel allemand FORD WERKE dans le cadre du projet ESPRIT OPTIMUM. Il est relatif au forgeage d'une pièce automobile. C'est un problème difficile car l'écoulement en fin de procédé est complexe et confiné. Ainsi, les quinze derniers pour cent de la simulation représentent-ils plus de cinquante pour cent du temps de calcul. L'outil est piloté par une presse mécanique et sa vitesse maximale est de l'ordre de $1,2$ m.s^{-1}. Nous présentons les mesures de temps CPU effectuées lors des simulations sur trois maillages: un grossier dont le nombre de noeuds varie entre 1700 et 2200 noeuds (*cf.* figure 2 et 1), un de taille moyenne dont le nombre de noeuds varie entre 3500 et 5500 noeuds et un fin dont le nombre de noeuds varie entre 9500 et 14000 noeuds.



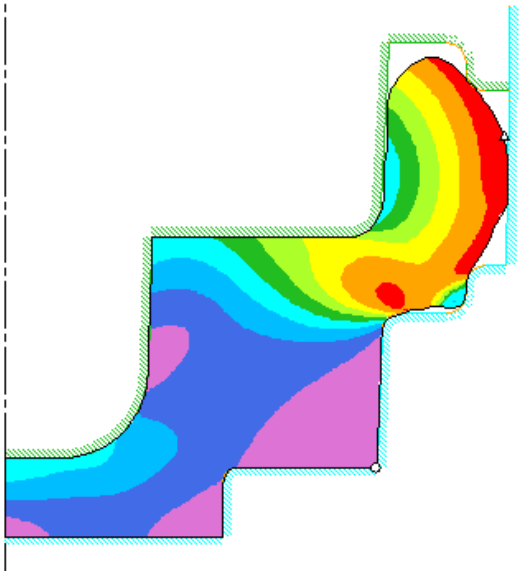
(a) Cas test FORD WERKE. Maillage grossier de 1789 noeuds au début des calculs.



(b) échelle des vitesses



(c) Isovaleurs de vitesses en milieu de calcul



(d) Isovaleurs de vitesses un peu avant la fin des calculs

FIG. 1 – *Cas test FORD WERKE: Maillage grossier et isovaleurs de vitesses*

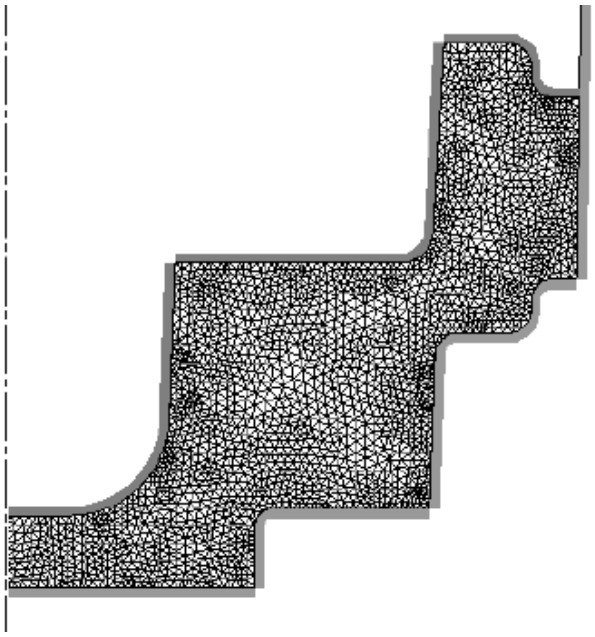


FIG. 2 – *Cas test FORD WERKE: Maillage grossier en fin de calcul (2226 noeuds).*

1.1.1 Meilleure robustesse du préconditionneur $\overline{\mathbf{IC}}(\omega)$ Bloc Diag

Nous avons effectué de nombreuses simulations de ce cas test, principalement pour une formulation P1+P1 simplifiée non stabilisée ($\beta = 1$). Nous avons vu, au chapitre II , page 51, que les systèmes linéaires issus de cette formulation sont beaucoup moins bien conditionnés qu'avec la formulation stabilisée ($\beta = 3$). MINRES peine davantage à converger et diverge parfois. Au cours des itérations de Newton-Raphson, nous avons systématiquement effectué une continuation de $\beta = 3$ jusqu'à $\beta = 1$. Les valeurs successives sont $\beta = 3$, 2, 1.5, 1.25 et 1.

Pour cette formulation et, *dans tous les cas testés*, le préconditionneur $\overline{\mathbf{IC}}(\omega)$ **Bloc Diag** s'est avéré plus robuste que $\overline{\mathbf{IC}}(\omega)$ **Diag**. Sur la figure 3, nous avons tracé pour les deux préconditionneurs, l'évolution du nombre d'itérations de MINRES au cours de la simulation complète sur deux domaines du problème grossier.

Pour 2 et 4 domaines, nous indiquons dans le tableau 1 :

It_{moy}: le nombre moyen d'itérations de MINRES effectuées sur l'ensemble de la simulation (entre 11500 et 12500 inversions de systèmes),

Nb_{sup} : le nombre de fois où le MINRES a effectué plus de 1000 itérations,

Nb_{div} : le nombre de fois où MINRES n'a pas convergé ($Nbit \geq 4000$) et a du être redémarré.

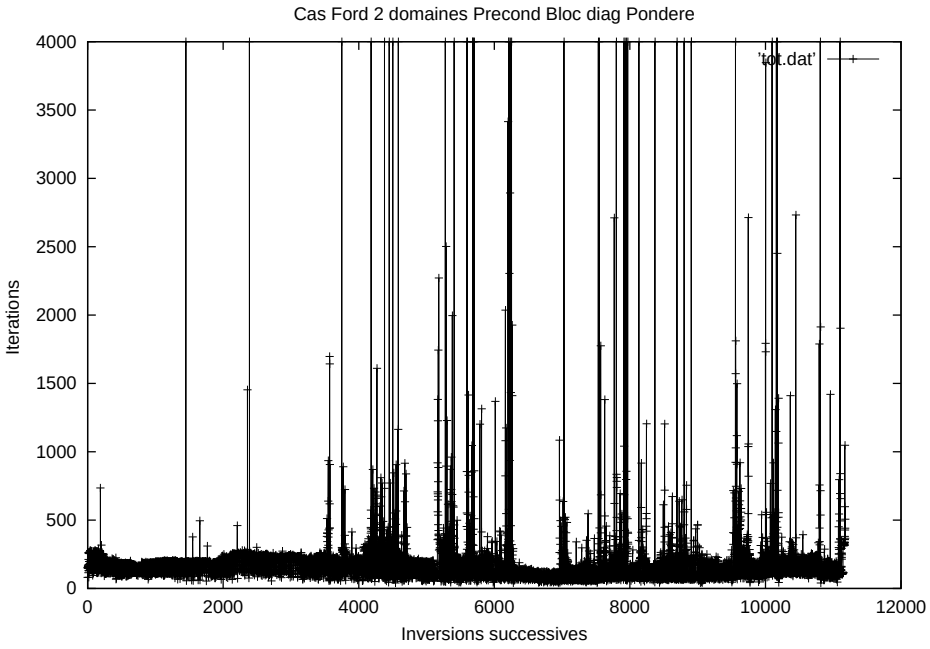
	$\overline{\mathbf{IC}}(\omega)$ **Diag**			$\overline{\mathbf{IC}}(\omega)$**Bloc Diag**		
	It_{moy}	Nb_{sup}	Nb_{div}	It_{moy}	Nb_{sup}	Nb_{div}
2 Proc	213	114	91	180	47	40
4 Proc	222	80	67	211	75	57

TAB. 1 – *Comportement du solveur itératif selon le préconditionneur utilisé lors de la simulation complète du cas FORD WERKE sur le maillage grossier et pour une formulation P1+P1 non stabilisée*

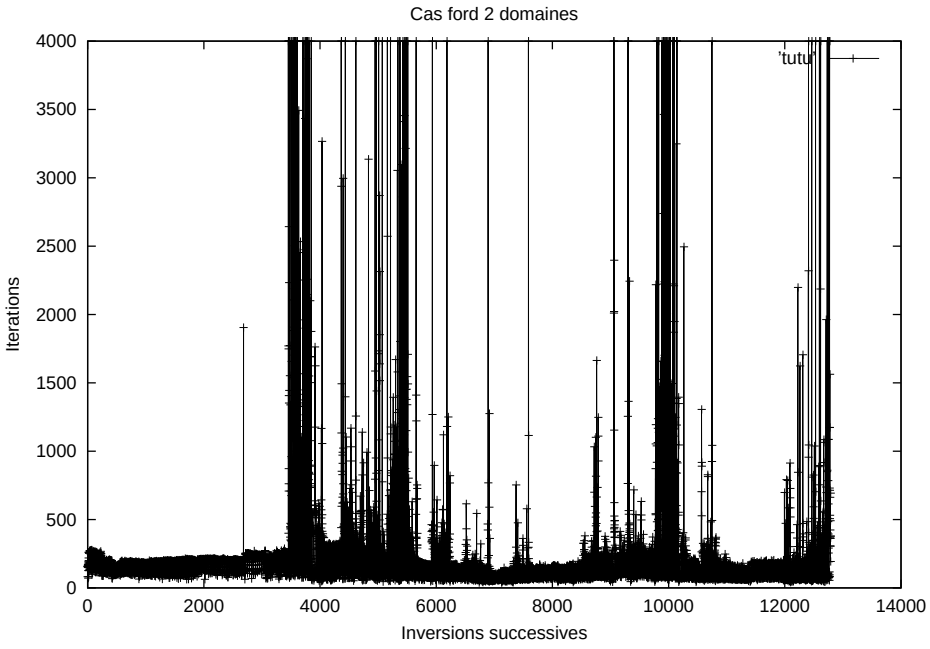
La tendance générale est à l'augmentation du nombre d'échecs et des convergences difficiles

quand le nombre de domaines augmente. Cela ne se retrouve pas dans le tableau ci-dessus pour le préconditionneur $\overline{\mathbf{IC}}(\omega)Diag$ essentiellement en raison de son très mauvais comportement sur 2 domaines. Au prix d'une augmentation du nombre d'itérations de Newton-Raphson, il est possible de réduire le nombre d'échecs en effectuant une continuation plus douce sur le paramètre β= 3, 2, 1.5, 1.25, 1.125, 1.0625. Par exemple, sur 4 domaines et pour le préconditionneur par blocs, la moyenne passe à 208 itérations et le solveur n'échoue plus qu'à 25 reprises.

Sur la figure 3, on remarque aussi le bien meilleur comportement de Newton-Raphson avec le préconditionneur par blocs. On effectue environ 10 % moins d'inversions de systèmes qu'avec le préconditionneur $\overline{\mathbf{IC}}(\omega)$ **Diag**. En raison des résolutions itératives tronquées, les chemins de convergence varient d'un préconditionnement à l'autre et, il semble que celui par blocs fournisse généralement des solutions plus précises permettant à Newton-Raphson de converger plus rapidement.



(a) Préconditionneur $\overline{\mathbf{IC}}(\omega)$ **Bloc Diag** sur 2 domaines. 47 calculs de plus de 1000 itérations. Temps total: 6h40'; $It_{moy} = 180$



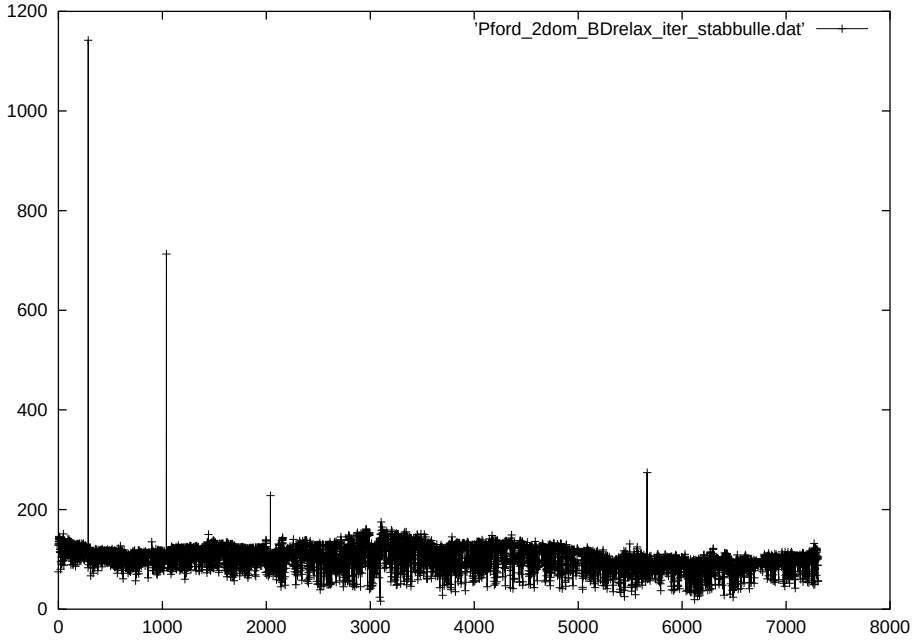
(b) Préconditionneur $\overline{\mathbf{IC}}(\omega)$**Diag** sur 2 domaines. 114 calculs de plus de 1000 itérations. Temps total: 7h51'; $It_{moy} = 213$

FIG. 3 – *Cas test FORD WERKE: Influence des préconditionneurs sur le comportement de MINRES tout au long de la simulation du cas FORD WERKE sur deux domaines pour le maillage grossier et avec la formulation P1+P1 simplifiée mais non stabilisée. Solveur direct 3h02*

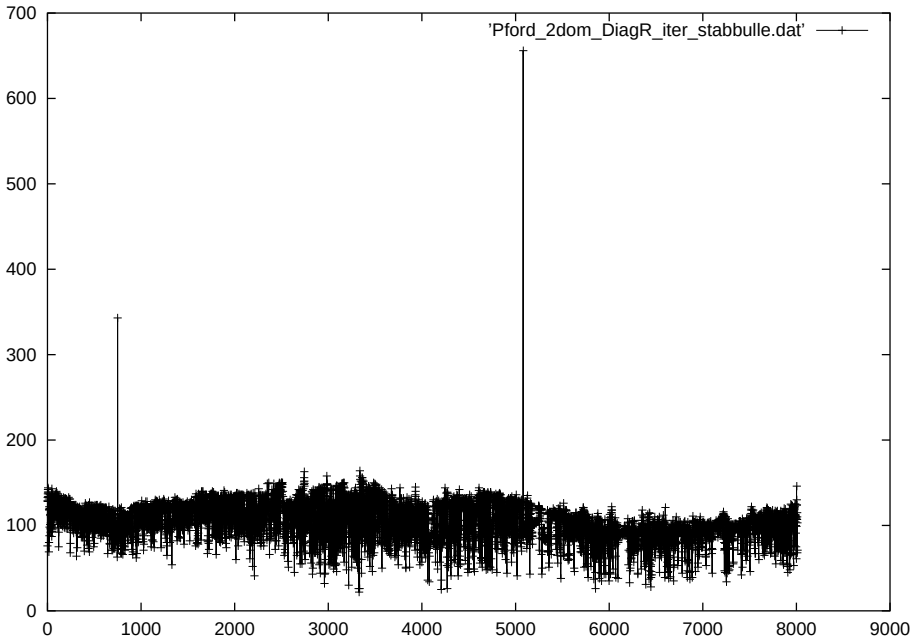
Influence de la formulation stabilisée

On effectue à nouveau les simulations précédentes mais cette fois, pour une formulation P1+P1 simplifiée stabilisée ($\beta = 3$). On constate une nouvelle fois l'influence très importante de cette formulation sur le conditionnement des systèmes linéaires et, sur la résolution non-linéaire. Pour les préconditionneurs $\overline{\mathbf{IC}}(\omega)$, MINRES converge alors sur tous les systèmes linéaires avec une moyenne de l'ordre de 103 itérations par inversion de système. La résolution non-linéaire est aussi grandement facilitée et, on observe une diminution de près de 40 % des itérations de Newton-Raphson. Cette différence importante s'explique par l'effet conjugué de la formulation et de l'abandon de la continuation qui contribuait à augmenter le nombre d'itérations non-linéaires. Cela se traduit finalement par une diminution du temps CPU d'un facteur 2.5 ! Ce qui nous permet d'avoir, sur le maillage grossier, des temps de calculs équivalents à ceux obtenus avec le solveur direct (qui est le plus rapide) dès deux processeurs et d'obtenir des accélérations à partir de trois processeurs.

Nous constatons à nouveau que pour une même formulation éléments finis, c'est avec le préconditionneur $\overline{\mathbf{IC}}(\omega)$ **Bloc Diag** que Newton-Raphson réalise le moins d'itérations *cf.* figure 4.



(a) Préconditionneur $\overline{\mathbf{IC}}(\omega)$**Bloc Diag** sur 2 domaines. Temps total: 2h40' $It_{moy} = 103$



(b) Préconditionneur $\overline{\mathbf{IC}}(\omega)Diag$ sur 2 domaines. Temps total: 2h55' $It_{moy} = 104$

FIG. 4 – *Cas test FORD WERKE: Influence de la formulation stabilisée sur le comportement du solveur parallèle sur deux domaines pour le maillage grossier et avec la formulation P1+P1 simplifiée et non stabilisée. Solveur direct 2h40'.*

1.1.2 Temps de calcul et performances

Cas moyen:

Pour ce cas test (les maillages varient entre 3500 et 5500 noeuds), nous avons reporté les mesures de temps de calculs dans le tableau ci-dessous. Sur 4 processeurs, avec une formulation non stabilisée, nous obtenons une accélération de 1.5 par rapport à la méthode directe. Avec la formulation stabilisée le solveur est compétitif en séquentiel et, nous obtenons la même accélération dès deux processeurs. En extrapolant à un nombre plus important de processeurs, on peut estimer, pour ce cas test, une accélération par rapport au solveur direct de 3 sur quatre processeurs et de 5.5 sur huit.

Nombre de domaines	Solveur direct	1	2	4	8
temps de calcul	9h17' (9h58')	9h12' (20h42')	6h03' (11h12')	(6h44')	(3h47')
Accélération		1	1.52 (1.84)	(3.07)	(5.47)
Efficacité		100 %	76 % (92 %)	(77 %)	(68 %)
Accélération absolue	1	1 (0.48)	1.52 (0.88)	(1.47)	(2.63)

TAB. 2 – *Temps de calcul pour le cas FORD WERKE sur les maillages moyens et pour une formulation P1+P1 simplifiée stabilisée* $\beta = 3$. *(Entre parenthèses temps de calculs avec une formulation simplifiée non stabilisée* $\beta = 1$ *avec continuation depuis* $\beta = 3$.*)*

Cas très fin: 9500 à 14000 noeuds

Pour des problèmes de cette taille nous n'avons pu effectuer que des simulations parallèles. En effet le solveur direct devient beaucoup trop coûteux et une simulation séquentielle aurait demandé plus de 100 heures de calcul. Nous avons donc effectué ces simulations sur 4 et 10 processeurs, pour une formulation $P1 + P1$ non stabilisée. Les résultats obtenus sont très satisfaisants. L'augmentation moyenne du nombre d'itérations n'est que de 5 %, alors que le nombre de domaines double. Cette faible augmentation, peut-être combinée à des effets de cache, nous permet d'obtenir une excellente accélération de 2.5 entre quatre et dix processeurs. Ces mesures nous permettent encore une fois de mettre en évidence la très bonne scalabilité de notre méthode parallèle.

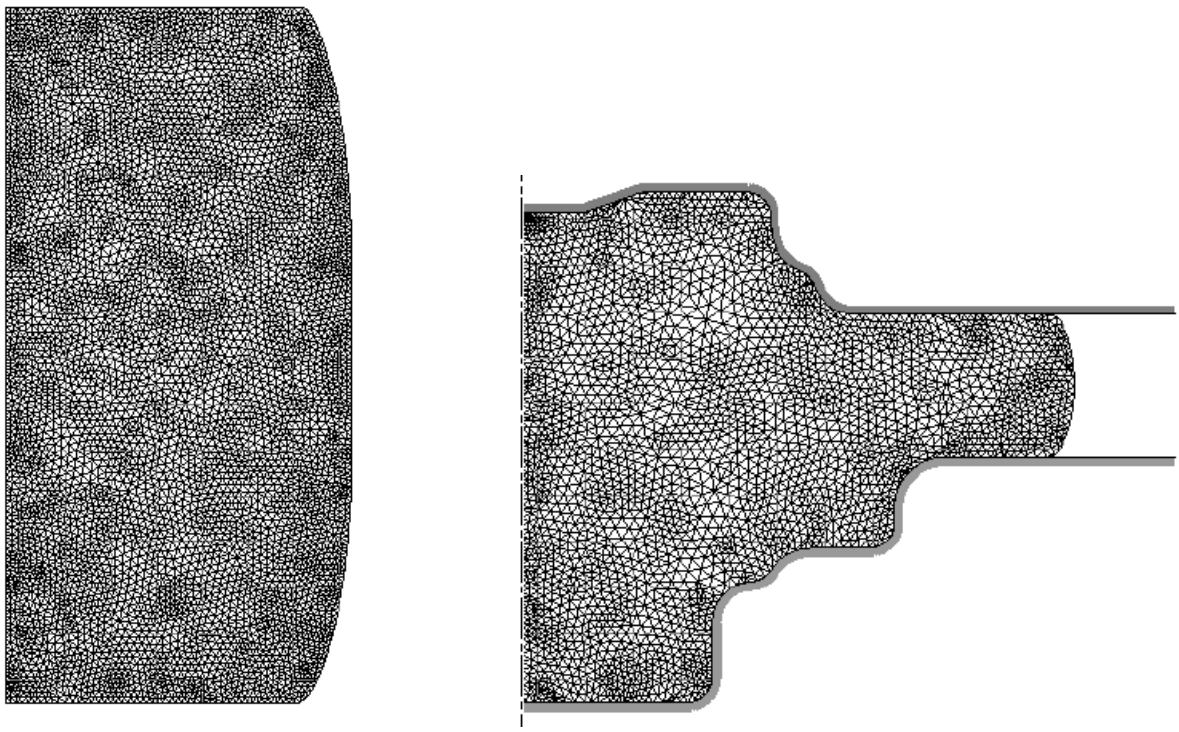
Nombre de domaines	4	10
temps de calcul	37h17'	14h53'
It_{moy}	388	406
Accélération	1	2.5

TAB. 3 – *Temps de calcul pour le cas FORD WERKE sur les maillages moyens et pour une formulation P1+P1 simplifiée non stabilisée* $\beta = 1$ *avec continuation depuis* $\beta = 3$

On peut raisonnablement, au vu des résultats obtenus précédemment, étendre cette conclusion à la formulation stabilisée. On pourrait donc s'attendre à obtenir sur 10 processeurs des temps CPU inférieurs à 10 heures.

1.2 Cas de GSB

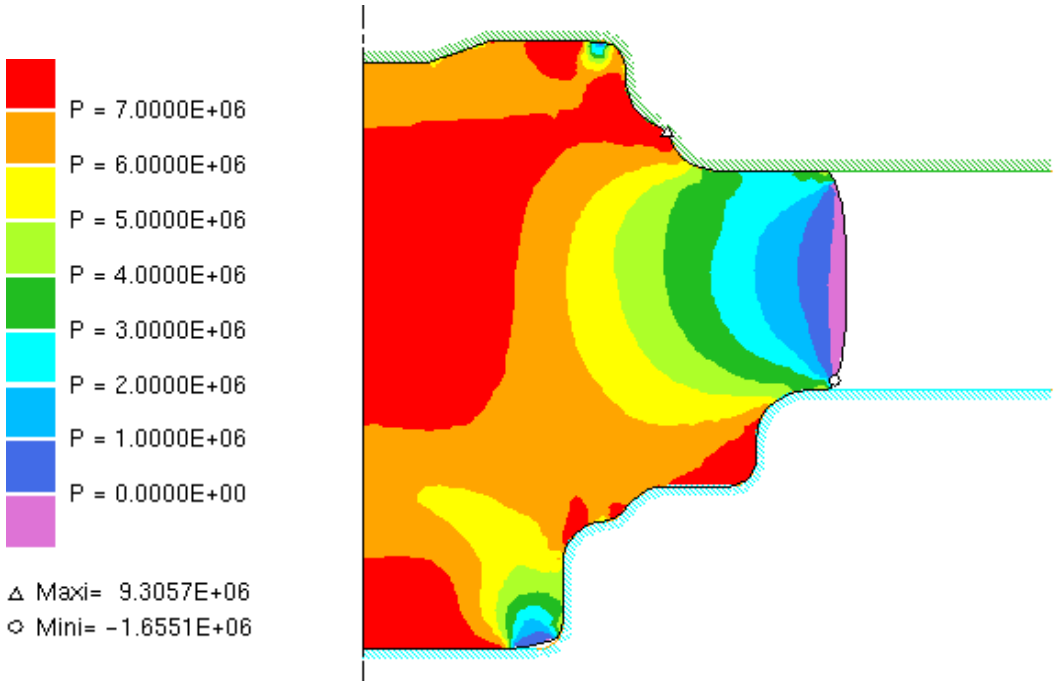
Ce cas test fourni par le forgeron espagnol GSB, consiste en trois passes de forgeage. La première est un simple écrasement entre tas plats, menant à ce que nous considérons comme la forme initiale (figure 5(a)). La seconde est une opération de préforme, objet de notre étude. La troisième, la finition, présente une qualité d'écoulement semblable au cas précédent (très contraint). Pour le forgeage de cette préforme, l'outillage (figure 5(b)) est piloté par une presse mécanique, dont la vitesse maximale est de l'ordre de $1\,m.s^{-1}$. Les simulations ont été effectuées sur des partitions du maillage initial de 5885 noeuds en deux, quatre et dix domaines. Au cours des remaillages, le nombre total de noeuds diminue pour être de l'ordre de 3500 noeuds durant la plus grande partie de la simulation. Une formulation simplifiée stabilisée ($\beta = 3$) est utilisée. Des accélérations sont obtenues par rapport aux solveurs directs et itératifs. L'efficacité parallèle est très satisfaisante, avec par exemple une accélération de 3.2 sur 4 processeurs. Les résultats sur 10 processeurs sont encore très bons, ce qui n'allait pas de soi. En effet, dans ce cas les décompositions produisent des maillages très petits, de l'ordre de 400 noeuds, avec lesquels l'efficacité de nos algorithmes restait à démontrer.



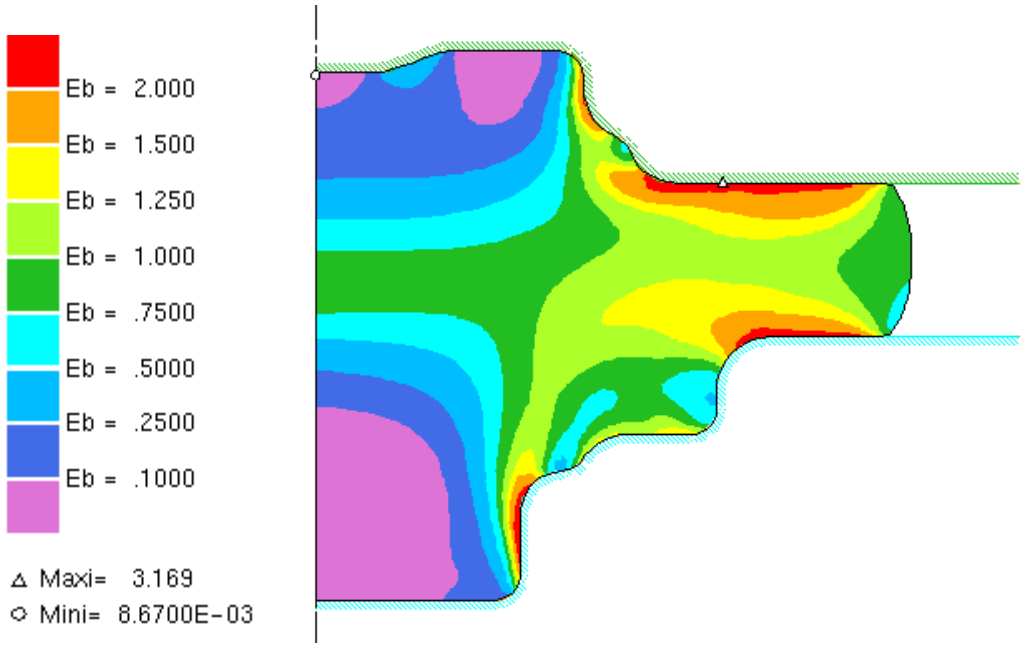
(a) Maillage initial de 5885 noeuds et 11432 éléments

(b) maillage et outil en fin de forgeage

FIG. 5 – *Cas GSB*



(a) Échelle (b)



(c) Échelle (d)

FIG. 6 – *Cas GSB. Isovaleurs de $\bar{\varepsilon}$ et de pression hydrostatique à des temps différents*

Nombre de domaines	Solveur direct	1	2	4	10
temps de calcul	9h28'	8h01'	5h24'	2h31'	1h22'
IT_{moy}		81	99	105	126
Itérations de NR	8530	8215	9037	8150	7969
Accélération	0.84	1	1.48	3.18	5.86
Efficacité	xxx	100 %	74 %	79.5 %	59 %
Accélération/direct	1.	1.18	1.74	3.75	6.91

TAB. 4 – *Temps de calcul pour le cas GSB pour une formulation P1+P1 simplifiée stabilisée* $\beta = 3$.

1.3 Conclusion sur le 2D

Les résultats obtenus prouvent que nous avons en grande partie atteint nos objectifs. L'étude que nous avons menée a permis de montrer que le choix d'une formulation P1+P1 stabilisée, avec $\beta = 3$, associée au préconditionneur $\overline{\mathbf{IC}}(\omega)$ était le meilleur. Sur des problèmes classiques de forgeage de préformes, tel le cas de GSB, nous sommes à même de concurrencer le solveur direct pour des maillages de 2500 noeuds voire moins et d'obtenir grâce à la parallélisation des accélérations conséquentes par rapport aux solveurs séquentiels. Sur des problèmes avec des écoulements plus complexes, tel le cas de FORD Werke, le seuil se situe plutôt vers 3500 noeuds. Le travail présenté dans ce document concerne, plus généralement, la parallélisation d'un solveur de Stokes en formulation mixte. Il s'applique aussi bien à des problèmes 3D, où les formulations et les méthodes de résolution sont similaires. Un exemple d'application est présenté au paragraphe suivant.

En particulier, le préconditionneur $\overline{\mathbf{IC}}(\omega)$ s'est avéré plus robuste que les simples préconditionneurs diagonaux. Nous sommes ainsi capables d'effectuer des simulations autrefois impossibles en raison de la non convergence de MINRES. Nous pouvons citer entre autre:

- les problèmes de contact matière/matière,
- les calculs avec des outils flottants, *i.e.* la position de certains outils devient une inconnue du problème,
- les méthodes multigrilles

Dans ces trois cas, une gestion du contact par une méthode de pénalisation, amène des termes extra diagonaux de poids important qui déconditionnent les systèmes linéaires. Les préconditionneurs par factorisations incomplètes permettent d'obtenir la convergence des solveurs ou de retrouver une efficacité des calculs comme sur l'exemple suivant.

2 Applications à des problèmes industriels 3D

Nous allons présenter dans ce paragraphe des résultats obtenus avec le code de calcul FORGE3®. Le préconditionneur $\overline{\mathbf{IC}}(\omega)$ a été introduit dans ce code qui, par ailleurs utilise des solveurs identiques à ceux de FORGE2®.

2.1 Simulation du laminage circulaire

Les résultats présentés ici sont le fruit des travaux de *K. Traore* qui a utilisé nos solveurs dans le cadre de sa thèse portant sur la simulation thermomécanique du procédé de laminage circulaire.

Le procédé (voir figure 7) consiste à comprimer un anneau entre deux mandrins tournants. Cela a pour effet de réduire son épaisseur, d'augmenter son diamètre et de conférer à la matière une anisotropie favorisant de bonnes propriétés mécaniques. C'est en général une opération intermédiaire de mise en forme, précédant le forgeage. Le mandrin extérieur "main roll" est fixe en translation et entraîne l'anneau et le mandrin intérieur "mandrel" dans son mouvement de rotation. Des galets latéraux de maintien "guide roll" ainsi que des cônes axiaux "axial roll" permettent de stabiliser l'évolution de la géométrie de l'anneau et de contrôler l'élargissement et l'ovalisation.

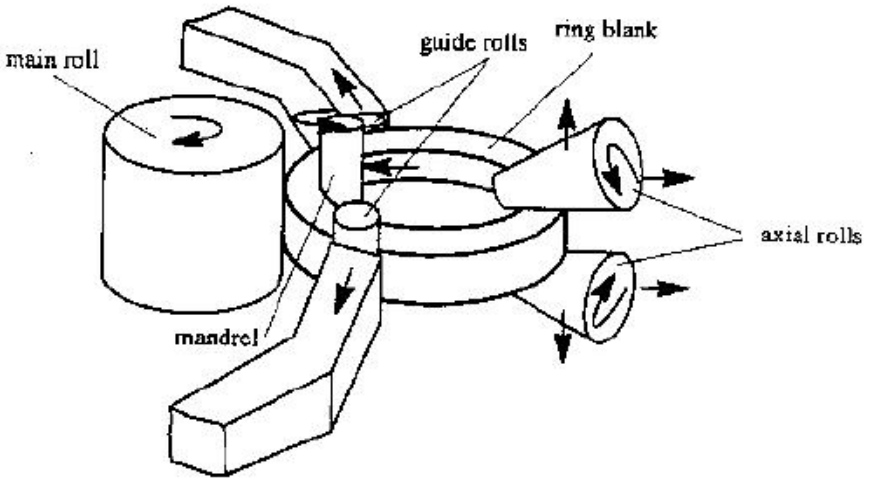


FIG. 7 – *Schéma d'un laminoir d'anneau Hu et al. [52]*

C'est un procédé fortement non-linéaire et extrêmement coûteux à simuler. En effet il réclame environ 15 tours et sa simulation complète demande environ 3500 incréments de calculs. La sensibilité du problème aux événements de contact est telle que les systèmes linéaires doivent être résolus avec beaucoup de précision, ce qui ne permet pas d'effectuer des résolutions itératives tronquées.

Le coût élevé de chaque incrément de calcul ne permet pas d'utiliser de maillages aussi fins que l'on voudrait. Si on adopte une description Lagrangienne du mouvement avec un maillage structuré de 14430 tétraèdres et 4650 noeuds (*cf.* figure 8), partitionné de manière angulaire en 4 sous-domaines et, si l'on utilise le préconditionneur bloc diagonal non défini positif **Tot** (le plus efficace), la simulation des $\frac{2}{5}$ du procédé (soit 7,5 tours) sur 4 processeurs est estimée à 60 jours de calcul ! L'utilisation du préconditionneur parallèle $\overline{\mathbf{IC}}(\omega)$, avec la possibilité de reprise par préconditionneur type **VipP** en cas de divergence ramène ce temps de calcul à 20 jours (estimés) soit une accélération d'un facteur 3. Il est à noter que si la partition est de mauvaise qualité ou non connexe, MINRES ne converge plus avec le préconditionneur $\overline{\mathbf{IC}}(\omega)$.

FIG. 8 – *Maillage régulier d'un anneau pour une description Lagrangienne de son mouvement. 14430 tétraèdres et 4650 noeuds.*

Ces temps de calculs étant encore bien trop importants, *K. Traore* a adopté une description ALE (Arbitrairement Eulerienne-Lagrangienne) du mouvement [78], et utilisé des maillages non structurés. Cette description permet de ne mailler finement que la partie de l'anneau en contact et réduit le nombre de noeuds d'un facteur 3,6. Ainsi sur un maillage de 3066 éléments et 1275 noeuds (figure 9), avec 4 processeurs, la simulation **complète** du procédé avec le préconditionneur **Tot** se réduit à 10 jours de calculs et, avec le préconditionneur $\overline{\mathbf{IC}}(\omega)$, à 5 jours. On observe donc dans ce cas une accélération d'un facteur 2 par rapport à la précédente version parallèle.

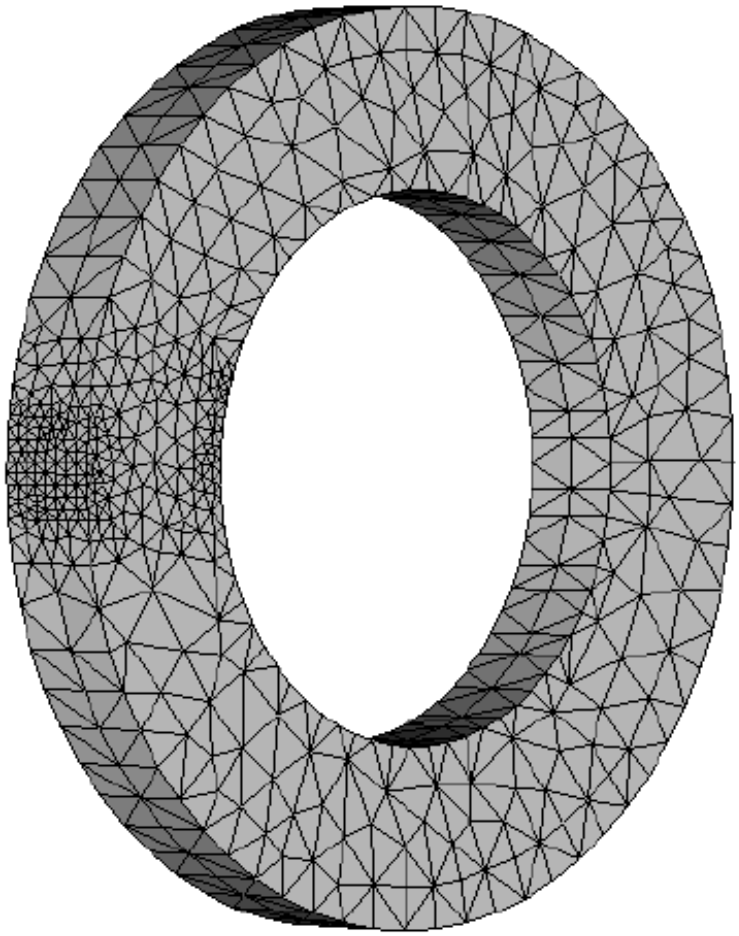


FIG. 9 – *Maillage régulier d'un anneau pour une description ALE de son mouvement. 3066 éléments et 1275 noeuds*

Conclusion et perspectives

La première étape de ce travail a consisté à rechercher une formulation mixte en vitesse et pression adaptée au contexte axisymétrique et aboutissant à des systèmes linéaires inversibles par une méthode itérative. Plusieurs formulations ont ainsi été implémentées et testées. Les comparaisons effectuées avec des solutions analytiques et numériques nous ont permis de valider certaines hypothèses proposées. Nous avons ainsi montré que la formulation que nous avons nommée simplifiée réalisait un bon compromis entre précision et coût de calcul. Nous avons, en outre, constaté que l'introduction d'un paramètre de stabilisation β^2, loin de dégrader la qualité des solutions, permet de les améliorer en terme de précision et d'allure des résultats, pour une même vitesse de convergence. Ce paramètre de stabilisation s'est avéré avoir une grande influence sur la vitesse de convergence de la résolution itérative par MINRES (et dans une moindre mesure sur la résolution non-linéaire), en permettant de réduire les temps de calculs de 25% à plus de 50% grâce à une amélioration du conditionnement des systèmes linéaires.

Toutefois, en raison de la taille des systèmes à résoudre dans le contexte 2D, le solveur itératif avec préconditionnement diagonal s'est avéré incapable de concurrencer le solveur direct. En indexant le critère de convergence de la méthode itérative sur la précision recherchée au court des itérations de l'algorithme de Newton-Raphson, il a été possible d'optimiser la résolution non-linéaire et de réduire de manière significative le nombre total d'itérations de MINRES. Cela nous a ainsi permis de concurrencer la méthode directe sur des problèmes de grande taille, de plus de 5000 noeuds, ou fortement non-linéaires. Afin de concurrencer le solveur direct sur une plus large classe de problèmes, nous avons cherché à construire des préconditionneurs plus efficaces. Nous nous sommes alors tournés vers les techniques de factorisations incomplètes ILU(0). Dans le cadre d'une formulation en vitesse et pression, plusieurs méthodes de construction de préconditionneurs basées sur ces factorisations étaient envisageables. Les tests effectués ont montré qu'une factorisation incomplète de la matrice en vitesse et pression, amenant une matrice non définie positive, aboutissait à un très bon préconditionneur que nous avons nommé **Ic**. Il s'est en effet avéré robuste et très performant, MINRES échouant parfois mais très rarement. Ici aussi, le paramètre de stabilisation β^2 a montré son importance en réduisant de manière significative le taux d'échecs du solveur par rapport à une formulation non stabilisée. En outre, nous avons fiabilisé ce solveur en introduisant la possibilité, en cas d'échec, de redémarrer MINRES avec un préconditionneur bloc diagonal défini positif. Ces développements en séquentiel permettent de concurrencer le solveur direct à partir de 2500 ou 3000 noeuds. De plus, par rapport à l'ancienne formulation pénalisée en vitesse avec son stockage profil, l'introduction d'un stockage morse nous permet d'effectuer des simulations avec des maillages de très grande taille (supérieurs à 10000 noeuds). Nous avons ainsi montré que pour des problèmes 2D non-linéaires un solveur itératif peut être très efficace.

Par une approche SPMD et une méthode de partitionnement de domaine inspirée de travaux antérieurs en 3D, nous avons alors développé un solveur parallèle efficace. Les excellentes

performances obtenues en séquentiel avec le préconditionneur **Ic** nous ont incité, malgré les obstacles, à tenter de le paralléliser. Aussi avons-nous testé différentes manières de construire en parallèle un préconditionneur additif de Schwarz basé sur ces factorisations. Deux directions ont été plus particulièrement envisagées : la construction de factorisations incomplètes à partir des restrictions à chaque sous-domaine de la matrice globale, ou la construction de factorisations incomplètes à partir des matrices locales complétées d'une information additionnelle aux interfaces. La seconde s'est avérée réaliser le meilleur compromis, avec des coûts additionnels faibles et une bonne robustesse. Pour un nombre fixé de processeurs et pour des problèmes de l'ordre de 5000 noeuds, notre solveur parallèle avec ce préconditionneur est deux ou trois fois plus rapide qu'avec le préconditionneur bloc diagonal. Par contre, le nombre d'itérations de MINRES dépend dorénavant de la partition et du nombre de domaines, d'où une moins bonne efficacité parallèle. Cela se traduit par une augmentation importante du nombre d'itérations quand on passe du séquentiel à deux processeurs. Ensuite, à mesure que le nombre de processeurs augmente, cette évolution est beaucoup plus douce et permet, au moins jusqu'à dix processeurs, d'obtenir des accélérations. Ainsi, pour des problèmes de forgeage de 5000 noeuds ou plus, nous obtenons des accélérations de l'ordre de 3 sur quatre processeurs et de 5 voire 5.5 sur huit. Sur dix processeurs ou moins, nous sommes aussi en mesure de réduire les temps CPU pour des maillages de l'ordre de 3000 noeuds voire moins dans les cas favorables. Nous estimons que pour des problèmes de moins de 1500 ou 2000 noeuds il est préférable d'effectuer des simulations séquentielle avec un solveur direct. Par contre, pour des problèmes de taille supérieure, la méthode itérative devient compétitive et le code parallèle susceptible de fournir de bonnes accélérations.

Les applications de travail sur le préconditionnement parallèle, ne se limitent pas simplement à la simulation du forgeage en 2D mais peuvent s'étendre à de nombreux autres problèmes et aux autres dimensions de l'espace. Ces résultats ont contribué à faire progresser le code FORGE3®: les préconditionneurs séquentiels et parallèles basés sur les factorisations incomplètes ont en effet permis d'accélérer les calculs d'un facteur compris entre deux et trois par rapport à la la version précédente. Des simulations difficiles, voire impossibles, à réaliser jusqu'alors ont ainsi pu être effectuées, telle, par exemple celles de laminage circulaire.

Par nos travaux, nous avons pu répondre à certaines interrogations quant à la validité de notre formulation éléments finis et la possibilité de réduire les temps de calculs sur des maillages relativement fins. Nous avons montré qu'une résolution parallèle efficace de problèmes 2D était possible et, que la stratégie gagnante résidait dans la résolution itérative parallèle de problèmes non-linéaires. La formulation éléments finis P1+P1 simplifiée stabilisée s'est montrée très satisfaisante. Elle a été étendue avec succès par d'autres personnes aux équations de l'élastoplasticité. D'un point de vue plus rigoureux il serait intéressant d'étudier son champ d'application en dehors de problèmes très visqueux, ainsi que la validité théorique de nos hypothèses simplificatrices.

Le préconditionnement par des matrices non symétriques définies positives soulève aussi de nombreuses interrogations et les seules réponses que nous avons pu fournir sont numériques. Un travail additionnel de lissage du comportement de MINRES pour ces préconditionneurs, de manière à bien obtenir une décroissance du résidu à chaque itération, permettrait sans doute de gagner encore en robustesse. Le préconditionneur additif de Schwarz que nous avons mis en place donne de bons résultats mais il peut encore progresser. Le saut important du séquentiel au parallèle ainsi que l'augmentation du nombre d'itérations de MINRES avec le nombre de domaines, limitent le nombre de processeurs utilisables. D'autres techniques de construction du préconditionneur pourraient être explorées, notamment à partir des techniques développées dans le cadre des méthodes de décomposition de domaines. Parmi ces dernières, les techniques de sous-structuration semblent les plus prometteuses.

Annexe 1

Cette annexe contient les mesures d'erreurs relatives effectuées pour tracer les courbes de convergence éléments finis présentées au chapitre I . Les pentes p et les constantes C sont estimées telles que

- en vitesse : $\dfrac{||v_h - \bar{v}||_1}{||\bar{v}||} = Ch^p$
- en pression : $\dfrac{||p_h - \bar{p}||_0}{||\bar{p}||} = Ch^p$

Écoulement de Poiseuille pour un comportement Newtonien

Taille de maille (Nbnoe)	Complète	Simplifiée	Simplifiée $\beta = 3$	Simplifiée $\beta = 5$	Stabilisée $\beta^2 = 0.1$	Stabilisée $\beta^2 = 0.01$
3.618D-02 (33)	7.795	7.919	9.096	11.6	8.619	7.966
1.591D-02 (145)	2.803	2.813	2.987	3.557	2.892	2.811
7.188D-03 (651)	1.302	1.304	1.327	0.141	1.312	1.305
4.145D-03 (1913)	0.756	0.756	0.762	0.790	0.758	0.757
3.299D-03 (2982)	0.591	0.591	0.594	0.610	0.592	0.592
2.030D-03 (7809)	0.367	0.367	0.367	0.374	0.367	0.367
1.214D-03 (21645)	0.219	0.218	0.219	0.221	0.218	0.219
Pentes	1.00	1.01	1.01	1.05	1.01	1.00
Constantes	1.85	1.86	2.02	2.64	1.93	1.87

TAB. 1 – *Poiseuille newtonien évolution de l'erreur relative en % sur la vitesse en norme 1 en fonction de la taille de maille. Taux de convergence en vitesse.*

Taille de maille	Complète	Simplifiée	Simplifiée $\beta = 3$	Simplifiée $\beta = 5$	Stabilisée $\beta^2 = 0.1$	Stabilisée $\beta^2 = 0.01$
3.618D-02	8.704	9.192	12.3	19.0	10.7	9.843
1.591D-02	0.933	1.069	2.109	4.292	1.552	1.164
7.188D-03	0.333	0.367	0.502	0.961	0.392	0.444
4.145D-03	0.134	0.144	0.182	0.355	0.142	0.174
3.299D-03	0.109	0.116	0.127	0.238	0.0996	0.143
2.030D-03	0.049	0.051	0.062	0.103	0.051	0.069
1.214D-03	0.029	0.029	0.027	0.048	0.029	0.037
Pentes	1.40	1.44	1.52	1.48	1.36	1.34
Constantes	2.9	4.0	7.7	10.5	2.5	2.8

TAB. 2 – *Poiseuille newtonien évolution de l'erreur relative en % sur la pression en norme 0 en fonction de la taille de maille. Taux de convergence en pression.*

Écoulement de Poiseuille pour un comportement viscoplastique

Taille de maille	Complète	Simplifiée	Simplifiée $\beta = 3$	Simplifiée $\beta = 5$	Stabilisée $\beta^2 = 0.1$	Stabilisée $\beta^2 = 0.01$
3.618D-02	57.7	57.8	58.2	59.3	57.9	58.80
1.591D-02	34.7	34.8	34.8	35.5	35.0	35.30
7.188D-03	18.3	18.3	18.1	18.2	18.8	19.33
4.145D-03	11.6	11.6	11.4	11.4	11.9	11.46
3.299D-03	9.02	9.02	8.89	8.90	9.24	8.959
2.030D-03	5.48	5.48	5.43	5.44	5.57	5.458
1.214D-03	3.25	3.24	3.22	3.22	3.29	3.235
Pentes	1.04	1.04	1.03	1.03	1.05	1.01
Constantes	34.7	34.7	32.8	32.8	38.1	30.41

TAB. 3 – *Poiseuille viscoplastique évolution de l'erreur relative en % sur la vitesse en norme 1 en fonction de la taille de maille. Taux de convergence en vitesse.*

Taille de maille	Complète	Simplifiée	Simplifiée $\beta = 3$	Simplifiée $\beta = 5$	Stabilisée $\beta^2 = 0.1$	Stabilisée $\beta^2 = 0.01$
3.618D-02	20.5	20.8	13.2	13.0	21.5	60.30
1.591D-02	9.92	9.92	4.56	4.44	11.4	26.96
7.188D-03	4.21	4.21	2.42	1.53	5.63	15.08
3.299D-03	1.98	1.98	1.64	1.45	2.17	2.059
2.030D-03	1.15	1.15	1.03	0.96	1.21	1.418
1.214D-03	0.71	0.71	0.65	0.63	0.70	0.957
Pentes	1.06	1.06	0.93	0.84	1.15	0.76
Constantes	8.46	8.38	3.38	1.76	16.1	1.65

TAB. 4 – *Poiseuille viscoplastique évolution de l'erreur relative en % sur la pression en norme 0 en fonction de la taille de maille. Taux de convergence en pression.*

Écrasement entre tas plats pour un comportement Newtonien

Taille de maille	Complète	Simplifiée	Simplifiée $\beta = 3$	Simplifiée $\beta = 5$	Stabilisée $\beta^2 = 0.1$	Stabilisée $\beta^2 = 0.01$
3.745	1.155	1.154	1.138	1.216	4.926	2.425
1.618	0.555	0.555	0.555	0.573	3.092	0.937
1.049	0.336	0.336	0.334	0.343	2.102	0.526
0.531	0.183	0.183	0.182	0.185	0.802	0.240
0.403	0.151	0.151	0.150	0.152	0.495	0.181
0.285	0.109	0.109	0.108	0.109	0.297	0.126
Pente	0.92	0.92	0.91	0.94	1.47	1.16
Constantes	0.0034	0.0034	0.0033	0.0034	0.01	0.0052

TAB. 5 – *Tas plat Newtonien frottant évolution de l'erreur relative en % sur la vitesse en norme 1 en fonction de la taille de maille. Taux de convergence en vitesse.*

Taille de maille	Complète	Simplifiée	Simplifiée $\beta = 3$	Simplifiée $\beta = 5$	Stabilisée $\beta^2 = 0.1$	Stabilisée $\beta^2 = 0.01$
3.745	6.475	6.453	3.702	4.560	20.12	10.59
1.618	1.743	1.760	1.247	1.654	13.28	4.484
1.049	1.495	1.493	0.885	1.085	9.414	2.547
0.531	0.776	0.775	0.518	0.604	4.140	1.206
0.403	0.566	0.568	0.437	0.468	2.744	0.822
0.285	0.334	0.335	0.368	0.438	1.756	0.667
Pente	1.12	1.12	1.04	1.08	1.26	1.12
Constantes	0.015	0.015	0.009	0.011	0.089	0.024

TAB. 6 – *Tas plat Newtonien frottant évolution de l'erreur relative en % sur la pression en norme 0 en fonction de la taille de maille. Taux de convergence en pression.*

Écrasement entre tas plats pour un comportement viscoplastique

Taille de maille	Complète	Simplifiée	Simplifiée $\beta = 3$	Simplifiée $\beta = 5$	Stabilisée $\beta^2 = 0.1$	Stabilisée $\beta^2 = 0.01$
3.745	1.495	1.493	1.448	1.478	6.713	3.985
1.618	0.732	0.732	0.719	0.725	5.508	1.815
1.049	0.494	0.494	0.487	0.491	4.350	1.092
0.531	0.287	0.287	0.283	0.284	1.925	0.459
0.403	0.231	0.231	0.230	0.233	1.762	0.457
0.285	0.164	0.164	0.164	0.166	1.708	0.326
Pente	1.06	1.06	1.11	1.13		1.02
Constantes	0.012	0.012	0.077	0.009		0.031

TAB. 7 – *Tas plat visco frottant évolution de l'erreur relative en % sur la vitesse en norme 1 en fonction de la taille de maille. Taux de convergence en vitesse.*

Taille de maille	Complète	Simplifiée	Simplifiée $\beta = 3$	Simplifiée $\beta = 5$	Stabilisée $\beta^2 = 0.1$	Stabilisée $\beta^2 = 0.01$
3.745	5.046	5.028	3.372	4.006	19.66	12.05
1.618	1.518	1.529	1.272	1.519	14.76	5.397
1.049	1.256	1.255	0.774	0.887	11.53	3.258
0.531	0.661	0.661	0.408	0.451	5.322	1.400
0.403	0.520	0.521	0.354	0.366	1.978	1.181
0.285	0.317	0.317	0.254	0.282	1.832	0.856
Pente	1.06	1.06	1.11	1.14	x	1.02
Constantes	0.012	0.012	0.008	0.009	x	0.031

TAB. 8 – *Tas plat visco frottant évolution de l'erreur relative en % sur la pression en norme 0 en fonction de la taille de maille. Taux de convergence en pression.*

Références bibliographiques

[1] J.F. Agassant, P. Avenas, J.P. Sergent, B. Vergnes, and M. Vincent. *La mise en forme des matières plastiques.* Lavoisier Tec & Doc, 3 edition, 1996.

[2] C. Aliaga. *Simulation numérique par éléments finis en 3D du comportement thermomécanique au cours du traitement thermique des aciers: application à la trempe de pièces forgées ou coulées.* Thèse de doctorat, Ecole Nationale Supérieure des Mines de Paris, 2000.

[3] D.N. Arnold, F. Brezzi, and M. Fortin. A stable finite element for Stokes equations. *Calcolo*, 21:337–344, 1984.

[4] S. F. Ashby, T. A. Manteuffel, and P. E. Saylor. A taxonomy for conjugate gradient methods. *SIAM J. Numer. Anal.*, 27(6):1542–1568, 1990.

[5] J. Atanga and D. Silvester. Iterative methods for stabilised mixed velocity-pressure finite elements. *International Journal for numerical methods in fluids*, 14:71–81, 1992.

[6] R. E. Bank and C. Wagner. Multilevel ILU decomposition. *Numer. Math.*, 82:543–576, 1999.

[7] J. Baranger and K. Najib. Analyse numérique des écoulements quasi-newtoniens dont la viscosité obéit à la loi puissance ou la loi de carreau. *Numerische Mathematik*, 58:35–49, 1990.

[8] A. Basermann. QMR and TFQMR methods for sparse nonsymmetric problems on massively parallel systems. *Lectures in Applied Math.*, 32:59–76, 1996.

[9] A. Basermann, B. Reichel, and C. Schelthoff. Preconditionned CG methods for sparse matrices on massively parallel machines. *Parallel Computing*, 23:381–398, 1997.

[10] J. H. Bramble and J. E. Pasciak. A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems. *Mathematics of Computation*, 50(181):1–17, 1988.

[11] J. H. Bramble, J. E. Pasciak, and A. Schatz. The construction of preconditioneers for elliptic problems by substructuring, iv. *Mathematics of Computation*, 53:1–24, 1989.

[12] J. H. Bramble, J. E. Pasciak, and J. Xu. Parallel multilevel preconditioneers. *Mathematics of Computation*, 55:1–22, 1990.

[13] F. Brezzi and M. Fortin. *Mixed and hybrid finite element methods.* Springer-Verlag, 1991.

[14] G. Radicati Di Brozolo and Y. Robert. Conjugate gradient-like algorithms on a vector multiprocessor. *Parallel Computing*, 11:223–239, 1989.

[15] T. F. Chan and T. P. Mathew. Domain decomposition algorithms. *Acta Numerica*, pages 61–143, 1994.

[16] T. F. Chan and H. A. van der Vorst. Approximate and incomplete factorisations. Technical Report 871, University of Utrecht, Department of Mathematics, 1994.

[17] E. Chow and Y. Saad. Approximate inverse preconditioners via sparse-sparse iterations. *SIAM J. Sci. Comput.*, 19(3):995–1023, 1998.

[18] The DRAMA Consortium. Project Homepage. http://www.cs.kuleuven.ac.be/cwis/research/natw/DRAMA.html.

[19] The DRAMA Consortium. Final DRAMA Cost Model. Deliverable D1.1b, DRAMA Project, 1999.

[20] T. Coupez. Stable-stabilized finite element for 3D forming calculation. CEMEF,rapport interne, 1996.

[21] T. Coupez and S. Marie. From a direct solver to a parallel iterative solver in 3D forming simulation. *International Journal of Supercomputer and Applications*, 11:205–211, 1997.

[22] R. Dautray and J. L. Lions. *Analyse mathématique et calcul numérique pour les sciences et les techniques*, volume 6. CEA-INSTN Collection Enseignement, Eds Masson, 1988.

[23] M. J. Daydé, J. Y. L'Excellent, and N. I. M. Gould. Element-by-Element preconditioners for large partially separable optimization problems. *SIAM J. Sci. Computing*, 18(6):1767–1787, 1997.

[24] E. de Sturler. *Iterative methods on distributed memory computers.* Phd thesis, Delft University of Technology, Delft, the Netherlands, 1994.

[25] M. DeLong and J. Ortega. SOR as a preconditioner. *Applied Numerical Matematics*, 18:431–440, 1995.

[26] I. S. Duff and G. A. Meurant. The effect of ordering on preconditioned conjugate gradients. *BIT*, 29:635–657, 1989.

[27] I. S. Duff and H. A. van der Vorst. Developments and trends in the parallel solution of linear systems. Technical Report 1091, Departement of Mathematics, Utrecht University, 1999.

[28] L. C. Dutto and W. G. Habashi. Parallelization if the ILU(0) preconditioner for CFD problems on shared-memory computers. *Int. J. Numer. Meth. Fluids*, 30:995–1008, 1999.

[29] V. Eijkhout. Analysis of parallel incomplete point factorizations. *Linear Algebra And Its Applications*, 154-156:723–740, 1991.

[30] H. C. Elman. Multigrid and Krylov subspace methods for the discrete Stokes equations. *International Journal for numerical methods in fluids*, 22:755–770, 1996.

[31] C. Farhat. Which parallel finite element algorithm for which architecture and which problem? *Eng. Comput.*, 7:186–195, September 1990.

[32] C. Farhat, J. Mandel, and F. X. Roux. Optimal convergence properties of the feti domain decomposition method. *Comput. Methods Appl. Mech. Engrg.*, 115:365–385, 1994.

[33] C. Farhat and F. X. Roux. A method of finite element tearing and interconnecting and its parallel solution algorithm. *J. Comput. Systems Engng.*, 2:149–156, 1991.

[34] C. Farhat and F. X. Roux. Implicit parallel processing in structural mechanics. *Computational Mechanics Advances*, 2:1–124, 1994.

[35] M. R. Field. Optimizing a parallel conjugate gradient solver. *SIAM J. Sci. Comput.*, 19(1):27–37, 1998.

[36] S. Filippone, M. Marrone, and G. Radicati di Brozolo. Parallel preconditioned conjugate-gradient type algorithms for general sparsity structures. *International Journal Computer Mathematics*, 40:159–167, 1992.

[37] R. Fletcher. Conjugate gradient methods for indefinite systems. In Springer Verlag, editor, *Lecture Notes Math.*, volume 506, pages 73–89. Berlin, New-York, 1976.

[38] M. Fortin and A. Fortin. Experiments with several elements for viscous incompressible flows. *Int. Jour. for Num. Meth. in Fluids*, 5:911–928, 1985.

[39] M. Fortin and A. Fortin. Newer and newer elements for incompressible flows. In J.T. Oden, editor, *Finite Element in Fluids*, volume 6, pages 171–187. Wiley and Sons, 1985.

[40] M. Fortin and R. Glowinsky. *Méthodes de Lagrangien augmenté: application à la résolution numérique de problèmes aux limites*. Dunod, 1982.

[41] L. Fourment. Simulation des écoulements viscoplastiques par éléments finis: application au forgeage à chaud. In *Séminaire de Plasticité. Éléments finis et mise en forme des métaux*, volume 2. Ecole Nationale Supérieure des Mines de Paris - Centre de Mise en forme des matériau - CEMEF, 1994.

[42] L. Fourment, J. L. Chenot, and K. Mocellin. Numerical formulations and algorithms for solving contact problems in metal forming simulation. *Int. J. for Num. Meth. in Engn.*, pages 1435–1462, 1999.

[43] I. Fried. The l_2 and l_∞ conditions numbers of the finite element stifness and mass matrices and the pointwise convergence of the method. In London Academic Press, editor, *The Mathematics of Finite Elements and Applications*, pages 163–174. J. R. Whiteman, 1973.

[44] L. Gaston. *Simulation numérique par éléments finis bidimensionnels du remplissage de moules de fonderie et étude expérimentale sur maquette hydraulique*. Thèse de doctorat, Ecole Nationale Supérieure des Mines de Paris, 1997.

[45] C. Gay. *Contribution à la simulation numérique tridimensionnelle du forgeage à froid*. Thèse de doctorat, Ecole Nationale Supérieure des Mines de Paris, 1995.

[46] Y. Germain. *Modélisation par éléments finis d'écoulement viscoplastique avec frottement.Application au forgeage à chaud*. Thèse de doctorat, Ecole Nationale Supérieure des Mines de Paris, 1985.

[47] G. H. Golub and C. F. Van Loan. *Matrix Computation*. The Johns Hopkins University Press, Baltimore, MD, 3 edition, 1996.

[48] I. Gustafsson. A class of first order factorisation methods. *BIT*, 18:142–156, 1978.

[49] G. Haase. Parallel incomplete Cholesky preconditioners based on the non-overlapping data distribution. *Parallel Computing*, 24:1685–1703, 1998.

[50] B. Hendrickson and R. Leland. The Chaco user's guide, v.1.0. Technical report, Sandia National Laboratories, 1993.

[51] D. C. Hodgson and P. K. Jimack. A domain decomposition preconditioner for a parallel finite element solver on distributed unstructured grids. *Parallel Computing*, 23:1157–1181, 1997.

[52] Z. M. Hu, I. Pillinger, P. Hartley, S. McKenzie, and P. J. Spence. Thermoplastic finite element modelling of the rolling of a hot titanium ring. In S. F. Shen and P. R. Dawson, editors, *International Conference on Numerical Methods in Industrial Forming Processes*, pages 941–946. Ithaca 1995, Balkema Press, Rotterdam, 1995.

[53] E. Issman. *Implicit solution strategies for compressible flow equations on unstructured meshes*. Ph.d. thesis, Université Libre de Bruxelle, 1997.

[54] E. Issman and G. Degrez. Non-overlapping preconditioners for a parallel implicit Navier-Stokes solver. *Future generation Computer Systems*, 13:303–313, 1997/98.

[55] S. L. Johnsson and K. K. Mathur. Data structure and algorithms for the finite element method on a data parallel supercomputer. *Int. J. Num. Meth. Eng.*, 29:881–908, 1990.

[56] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical Report 95-035, Department of Computer Science, University of Minnesota, 1995.

[57] S. Marie. *Un modèle de parallélisation S.P.M.D. pour la simulation numérique de procédés de mise en forme des matériaux*. Thèse de doctorat, Ecole Nationale Supérieure des Mines de Paris, 1997.

[58] K. Mocellin. *Contribution à la simulation numérique tridimensionnelle du forgeage à chaud: Étude du contact et calcul multigrille*. Thèse de doctorat, Ecole Nationale Supérieure des Mines de Paris, 1999.

[59] S. Norburn and D. Silvester. Stabilised vs. stable mixed methods for incompressible flow. *Computer Methods in Applied Mechanics and Engineering*, 166:131–141, 1998.

[60] C. C. Paige and M.A. Saunders. Solutions of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 121:617–629, 1975.

[61] M. Pakzad, J. L. Lloyd, and C. Philips. Independent columns: A new parallel ilu preconditioner for the PCG method. *Parallel Comp.*, 23:637–647, 1997.

[62] R. Pierre. Simple C^0-approximations for the computation of incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 68:205–227, 1988.

[63] R. Pierre. Optimal selection of the bubble function in the stabilisation of the P1-P1 element for the Stokes problem. *SIAM Journal of Numerical Analysis*, 32:1201–1224, 1995.

[64] O. Pironneau. *Méthodes des éléments finis pour les fluides*. P. G. Ciarlet et J. L. Lions, recherche en mathématiques appliquées, Masson edition, 1988.

[65] A. Ramage and A. J. Wathen. Iterative solution techniques for the Stokes and Navier-Stokes equations. *International Journal for numerical methods in fluids*, 19:67–83, 1994.

[66] F. X. Roux. Domain decomposition methods and distributed memory machines. Technical report, ONERA,?

[67] F. X. Roux. Acceleration of the outer conjugate gradient by reorthogonalization for a domain decomposition method for structural analysis problems. In T. Chan, R. Glowinsky, J. Periaux, and O. B. Wildlund, editors, *Proceedings of the Third International Symposium on Domain Decomposition Methods*, pages 314–319. SIAM, Philadelphie, 1990.

[68] T. Rusten and R. Winther. A preconditioned iterative method for saddlepoint problems. *SIAM J. Matrix Anal. Appl.*, 13(3):887–904, 1992.

[69] Y. Saad. Practical use of polynomial preconditionings for the conjugate gradient method. *SIAM J. Sci. Stat. Computing*, 6(4):865–881, 1985.

[70] Y. Saad. Krylov subspace methods on supercomputers. *SIAM J. Sci. Stat. Comput.*, 10(6):1200–1232, 1989.

[71] Y. Saad. Highly parallel preconditioners for general sparse matrices. In G. Golub, M. Luskin, and A. Greenbaum, editors, *Recent Advances in Iterative Methods, IMA Volumes in Mathematics and Its Applications*, volume 60, pages 165–199, New York, 1994. Springer Verlag.

[72] D. J. Silvester and N. Kechkar. Stabilised bilinear-constant velocity-pressure finite elements for the conjugate gradient solution of the Stokes problem. *Computer Methods in Applied Mechanics and Engineering*, 79:71–86, 1990.

[73] G. L. G. Sleijpen and H. A. Van der Vorst. Reliable updated residual in hybrid Bi-CG methods. Preprint 886, University Utrecht, Dept. Math., 1994.

[74] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra. *MPI: The complete reference*. Scientific and Engineering Computation, Janusz Kovalik edition.

[75] N. Soyris. *Modélisation tridimensionnelle du couplage thermique en forgeage à chaud.* Thèse de doctorat, Ecole Nationale Supérieure des Mines de Paris, 1990.

[76] G. Surdon. *Simulation numérique du forgeage tridimensionnel à chaud.* Thèse de doctorat, Ecole Nationale Supérieure des Mines de Paris, 1986.

[77] P. Le Tallec. Domain decomposition methods in computational mechanics. *Computational Mechanics Advances*, 1:121–220, 1994.

[78] K. Traore. Méthodes Arbitrairement Euleriennes-Lagrangiennes. Rapport bibliographique, Ecole Nationale Supérieure des Mines de Paris, 1998.

[79] H. A. van der Vorst. A vectorizable variant of some ICCG methods. *SIAM J. Sci. Stat. Computing*, 3(3):350–356, 1982.

[80] C. Vuik, R.R.P. van Nooyen, and P. Wesseling. Parallelism in ILU-preconditioned GMRES. *Parallel Comp.*, 24:1927–1946, 1998.

[81] C. Walshaw, M. Cross, M. Everett, S. Johnson, and K. McManus. Partitioning and mapping of unstructured meshes to parallel machine topologies. In *Irregular '95: Parallel Algorithms for Irregularly Structured Problems*, volume 980, pages 121–126. Springer, 1995.

[82] A. Wathen, B. Fischer, and D. Silvester. The convergence rate of the minimal residual method for the Stokes problem. *Numer. Math.*, 71:121–134, 1995.

[83] A. Wathen and D. Silvester. Fast iterative solution of stabilised Stokes systems Part I: Using simple diagonal preconditioners. *SIAM J. Numer. Anal.*, 30(3):630–649, 1993.

[84] A. Wathen and D. Silvester. Fast iterative solution of stabilised Stokes systems Part II: Using general block preconditioners. *SIAM J. Numer. Anal.*, 31(5):1352–1387, 1994.

[85] C. Weill-Duflos. *Optimisation de méthodes de résolution itérative de grands systèmes linéaires creux sur machines massivement parallèles.* Informatique, Université de Paris 6, 1994.

[86] L. Zhou and H. F. Walker. Residual smothing techniques for iterative methods. *SIAM J. Sci. Comput.*, 15(2):297–312, 1994.

[87] O. C. Zienkiewicz, J. P. Pilote, S. Toyoshima, and S. Nakazawa. Iterative methods for constrained and mixed approximation. An inexpensive improvement of F.E.M. performance. *Comput. Meth. in Appl. Mech. and Eng.*, 51:3–29, 1985.

Résumé

Nous présentons dans cette contribution les techniques que nous avons mises en oeuvre pour paralléliser un code éléments finis 2D dédié à la simulation du forgeage de pièces axisymétriques. Les modèles de comportement conduisent à résoudre des équations de type Stokes généralisé, exprimées sous forme mixte en vitesse et pression. La discrétisation spatiale est effectuée par une méthode éléments finis originale basée sur une stabilisation du MINI-élément P1+P1.

Cette approche mène à des systèmes linéaires symétriques non définis positifs que l'on peux inverser avec un solveur itératif. L'introduction de préconditionneurs par factorisations incomplètes LDL(0) ainsi que l'optimisation de la résolution non-linéaire nous permet de concurrencer une méthode directe sur des maillages de plus de 3000 noeuds.

Une stratégie de parallélisation SPMD couplée au solveur itératif avec préconditionnement diagonal aboutit à un solveur parallèle simple et efficace, ne dépendant ni de la partition ni du nombre de domaines. Différentes stratégies sont envisagées pour développer des factorisations incomplètes parallèles. Un préconditonneur additif de Schwarz est notamment proposé. Celui-ci est construit à partir des matrices locales, complétées sur leur diagonale aux interfaces et avec un coefficient de sur-relaxation.

Des résultats sur des simulations industrielles sont donnés pour une machine parallèle à mémoire partagée. Ceux-ci, obtenus sur des problèmes 2D et 3D, prouvent la pertinence de notre approche.

Les stratégies développées permettent ainsi de réduire de manière significative les temps de simulation de la majorité des cas industriels. Elles permettent aussi d'élargir les champs d'application des codes de calculs à des simulations industrielles très complexes ou avec des maillages de plus de 15000 noeuds en 2D.

MOTS CLES: Problème de Stokes, formulation axisymétrique, MINI-élément P1+P1, factorisations incomplètes, Calcul parallèle, préconditionneurs parallèles

MINI-element and incompletes factorisations for the parallelisation of a 2D stokes solver. Application to the forging process.

Abstract

In this contribution, we present the parallelisation of a 2D finite element code dedicated to the simulation of hot metal forging of axisymetrical workpieces. The constitutives equations leads to a generalised Stokes problem. The spatial discretisation is carried out by an original mixed finite element approach based on a stabilisation of the so-called MINI-element P1+P1.

This approach allows to solve the associated symmetric non positive definite systems with an iterative solver. The introduction of incomplete factorisation preconditionner LDL(0) and the optimisation of the non-linear resolutions allow the sequential code to compete with a direct solver on meshes larger than 3000 nodes. A SPMD parallelisation strategy combined with the iterative solver with diagonal scaling produce a simple parallel code that do not depend of the partition nor of the number of domains. We consider various strategies to develop parallel incomplete factorisations. We propose an additive Schwarz preconditionner, built up from the local matrices completed at the interfaces on their diagonal.

We present the results obtained on some 2D and 3D industrial applications carried out on a shared memory parallel computer. It proves the relevance of our approach. The strategy we developped reduce significantly the simulation time for most of the industrials applications. This allow to extend the applications of our code to very complex industrial simulations or to problems with meshes larger than 15000 nodes in 2D.

KEYWORDS: Stokes problem, axisymetrical formulation, MINI-element P1+P1, incompletes factorisations, parallel computing, parallel preconditionning