Au-delà de la discrétisation stratégies de résolution du système poroélastique

Dans les chapitres précédents, une méthode de discrétisation pour le système de poroélasticité a été construite, étudiée et améliorée pour répondre aux problématiques spécifiques aux applications. Ce chapitre aborde le système poroélastique à travers un autre angle d'approche : celui de la résolution du système linéaire issu de la discrétisation. Ainsi, jusqu'à la fin de ce chapitre, la discrétisation est simplifiée en utilisant le schéma volumes finis deux points plutôt que le schéma multipoints pour la partie écoulement. Ces travaux sont motivés par l'incapacité des méthodes classiques à résoudre efficacement les systèmes obtenus dans des cas réels tels que celui illustrant le chapitre 5. Ce point est détaillé dans la première section du présent chapitre, qui donne également un aperçu des stratégies envisageables. Parmi celles-ci, la méthode séquentielle dite fixed-stress est détaillée dans la section 6.2. En réinterprétant celle-ci comme un problème d' annulation de résidu, la section 6.3 en propose une variante accélérée. Le principal intérêt derrière cette méthode étant de pouvoir choisir un sous-solveur pour chaque sous-problème, la section 6.4 est dédiée à la recherche de deux solveurs, chacun performant sur l'un des deux sous-problèmes physiques. On qualifie ainsi une méthode multigrille algébrique pour le sous-problème en pression, et une méthode de décomposition de domaine multi-niveaux pour le sous-problème en déplacement. Ces deux méthodes sont finalement mises en place et utilisées en tant que sous-solveurs dans le cadre de l'algorithme fixed-stress dans la section 6.5.

Sommaire

6.1 Panorama des différentes méthodes

Différentes techniques vont être abordées dans ce chapitre, parfois de manière indépendante, parfois en tant qu'élément d'une approche englobante. Pour faciliter la compréhension des concepts manipulés et de leurs interactions, cette section propose d'entrée de jeu un aperçu des stratégies envisageables. Le choix de la stratégie à suivre est justifié en fin de section à partir du cas d'application tridimensionnel proposé dans le chapitre précédent.

6.1.1 Méthode directe ou méthode itérative

Du point de vue le plus proche de la résolution, et quelle que soit la méthode utilisée en amont, le cœur de l'algèbre linéaire consiste à résoudre un système linéaire réel de la forme $Ax = b$ avec $A \in \mathbb{R}^{n\times n}$, $x \in \mathbb{R}^n$ et $b \in \mathbb{R}^n$. On classifie en général les algorithmes de résolution en deux grandes catégories : les méthodes *directes* et les méthodes *itératives* [AK08].

Avec une méthode directe, la solution "exacte" du système peut être déterminée en un nombre fini d'opérations. Les guillemets sont utilisés pour souligner le fait que même si l'algorithme donne la vraie solution en arithmétique exacte, l'utilisation de l'outil numérique introduit inévitablement des erreurs d'arrondis lors de la manipulation de coefficients réels (de l'ordre de 10^{-16} par opération sur une machine standard). Un algorithme célèbre est la factorisation LU, qui consiste à réécrire la matrice A comme le produit d'une matrice triangulaire inférieure avec une matrice triangulaire supérieure en suivant des étapes inspirées de la méthode de Gauss, pour ensuite obtenir la solution en résolvant deux systèmes triangulaires (ce qui est une opération relativement peu coûteuse par rapport à la factorisation). Les méthodes directes sont les plus précises et sont également robustes : leur efficacité dépend relativement peu des coefficients de la matrice. En revanche, elles sont coûteuses en nombre d'opérations – de l'ordre de $\mathcal{O}(n^3)$ pour une matrice dense – et exigent une grande quantité de mémoire. De plus, la parallélisation de ces méthodes est délicate.

En pratique, les méthodes directes deviennent rapidement inutilisables lorsque la taille du système augmente. On leur préfère pour cette raison les méthodes dites itératives. Les méthodes itératives consistent à appliquer une procédure un certain nombre de fois sur une solution approchée jusqu'à ce que celle-ci soit suffisamment proche de la solution exacte. Ces méthodes ne donnent pas la solution exacte, mais une approximation de la solution, l'écart entre ces deux grandeurs pouvant être contrôlé à l'aide du critère d'arrêt de la méthode. Il existe une grande variété de méthodes itératives [Saa03], qui sont en général bien plus

économes en consommation de mémoire (du moins pour les systèmes creux tels que ceux issus de la discrétisation de problèmes physiques), moins coûteuses en nombre d'opérations et facilement parallélisables. L'inconvénient principal de ces méthodes est qu'elles sont bien moins robustes que les méthodes directes : le nombre d'itérations nécessaires à la convergence de ces méthodes dépend fortement des coefficients de la matrice. En réalité, une méthode itérative est toujours utilisée avec un préconditionnement : étant donné une matrice M inversible, on résout le système $M^{-1}Ax = M^{-1}b$ dont la solution est la même que celle du système $Ax = b$. Tout l'intérêt est de choisir une technique de construction de M de sorte que la résolution de ce système soit plus rapide ou plus robuste que celle du système original, ce qui est un sujet de recherche en soi. Ce point sera développé dans les sections suivantes. En pratique, il est possible d'appliquer le préconditionnement à la droite de la matrice plutôt qu'à sa gauche : le système effectivement résolu est alors $AM^{-1}y = b$, et la solution est déduite selon $x = M^{-1}y$. Cette approche est à préférer pour comparer deux préconditionneurs car avec celle-ci, le critère d'arrêt ne dépend pas du préconditionnement.

6.1.2 Résolution monolithique ou résolution séquentielle

En prenant du recul par rapport aux système linéaires et en se plaçant du point de vue plus éloigné de la méthode numérique étudiée, la discrétisation du problème poroélastique s'achève avec l'obtention du système

$$\begin{bmatrix} \mathcal{A} & -\mathcal{B} \\ \mathcal{B}^T & \mathcal{F} \end{bmatrix} \begin{bmatrix} \mathcal{U} \\ \mathcal{P} \end{bmatrix} = \begin{bmatrix} \mathcal{L}^u \\ \mathcal{L}^p + \mathcal{L}^c \end{bmatrix} \tag{6.1}$$

où l'on cherche à déterminer les déplacements et la pression inconnus au temps t^n en fonction du second membre, qui fait intervenir ces quantités au temps t^{n-1} (connues) ainsi que les données du problème. On parlera de résolution *externe* pour désigner le processus permettant de déterminer ces inconnues et donc de passer au pas de temps suivant de la simulation. Différentes stratégies peuvent être adoptées pour la résolution externe, catégorisées en deux familles principales : les stratégies de résolution *monolithique* et les stratégies de résolution *séquentielle*. Il est évidement souhaitable que toutes ces stratégies produisent au final une même solution à l'issue du temps t^n, *a minima* à une certaine tolérance près.

Résolution monolithique

La stratégie monolithique, aussi appelée *fully coupled*, est la plus naturelle. Elle consiste à voir (6.1) comme un grand système linéaire $Ax = b$ dont l'inconnue est un vecteur stockant les déplacements et la pression. Cette stratégie ne nécessite qu'une seule itération externe pour valider un pas de temps : la résolution de ce système. Même s'il n'y a dans ce cas pas de confusion possible, on parle de sous-solveur pour désigner la méthode de résolution. Celle-ci peut être l'une ou l'autre des méthodes développées dans la section 6.1.1 : directe, auquel cas il y une seule itération *interne*, ou itérative, auquel cas le sous-solveur effectue plusieurs itérations internes.

Résolution séquentielle

La seconde approche, la stratégie séquentielle, consiste à reformuler le système (6.1) de manière à découpler les deux équations. L'idée est de retomber sur deux sous-problèmes dont les inconnues sont respectivement les déplacements et la pression. Ces deux sous-problèmes sont résolus de manière alternée jusqu'à ce que la solution converge vers celle du système couplé. À chaque pas de temps, le processus de résolution va donc effectuer un certain nombre d'itérations externes, chacune d'entre elles nécessitant la résolution des sous-problèmes. À nouveau, on parle de sous-solveurs pour désigner les méthodes utilisées

	Sous-solveurs directs			Sous-solveurs itératifs			Combinaisons		
Stratégie monolithique	$l = 1$	$s = 1$		$l = 1$	$s > 1$		$\emptyset$		
Stratégie séquentielle	$l > 1$	$s^u = 1$	$s^p = 1$	$l > 1$	$s^u > 1$	$s^p > 1$	$l > 1$ ou $l > 1$	$s^u = 1$ $s^u > 1$	$s^p > 1$ $s^p = 1$

TAB. 6.1 : Nombre d'itérations externes l et internes s en fonction du couple stratégie – sous-solveur choisi. Dans le cas de la stratégie séquentielle, le sous-solveur utilisé pour la pression peut être différent du sous-solveur utilisé pour les déplacements.

pour résoudre ceux-ci. Comme dans le cas précédent, cette résolution peut être directe ou itérative, sachant qu'il y a cette fois deux sous-solveurs à choisir (un pour chaque problème). On parlera donc d'itérations internes en déplacement et d'itérations internes en pression. Au passage, il est possible de mélanger les types des sous-solveurs en optant par exemple pour une méthode directe en pression et pour une méthode itérative en déplacement, même si ce choix n'est pas fréquent. Il peut en revanche être intéressant d'utiliser deux méthodes itératives mais d'adapter le solveur ou le préconditionneur au sous-problème considéré. La stratégie séquentielle permet en particulier de réutiliser des codes de calcul existant pour chaque sous-problème et donc de réduire l'effort de développement. Il existe différentes approches pour découpler les deux équations, qui ne présentent pas toutes les mêmes propriétés de stabilité ou de vitesse de convergence (voir par exemple [MW13]). Parmi ces approches, on s'intéressera particulièrement à la méthode *fixed-stress split* qui a l'avantage d'être inconditionnellement stable et qui a récemment fait l'objet de nombreux travaux.

Les différentes combinaisons entre la stratégie de résolution et la méthode employée pour le ou les sous-solveurs sont résumées dans le tableau 6.1.

6.1.3 Choix de la stratégie

La plateforme Arcane est interfacée avec la bibliothèque *open source* PETSc[1] qui rassemble différents solveurs linéaires. Ceux-ci sont accessibles via une interface commune, de sorte qu'il suffit d'indiquer dans le jeu de données le solveur à utiliser et ses éventuels paramètres (critère d'arrêt, nombre maximum d'itérations, préconditionneur, etc.).

Dans les cas tests présentés jusqu'à maintenant, la résolution du système linéaire a été réalisée en choisissant l'approche conceptuellement la plus simple : la stratégie monolithique avec résolution directe du système. Cependant, les limites de cette approche apparaissent rapidement, par exemple lors des calculs sur le cas physique tridimensionnel réaliste de la section 5.4. Pour ce système couplé totalisant environ 300 milliers d'inconnues, la résolution par méthode directe est laborieuse avec les deux solveurs testés. Avec le premier solveur direct, SuperLU [Li05], la résolution échoue ; avec le second solveur direct, MUMPS [ADLK01], la résolution est un succès mais au prix d'une consommation importante de mémoire vive (plus de 20 Go) et d'un temps de résolution extrêmement long (environ 1h30 sur un cœur cadencé à 3,6 GHz).

Pour accélérer la résolution, le plus simple serait de conserver la stratégie de résolution monolithique tout en utilisant une méthode itérative pour résoudre le système couplé. La matrice n'étant pas symétrique, on utilise la méthode du BiCGStab avec un critère d'arrêt fixé à 10^{-8} et un nombre d'itérations maximal de 3000. Plusieurs préconditionnements sont testés pour ce problème : les méthodes ILU(k) avec $k = 0$ et $k = 4$

1. https ://www.mcs.anl.gov/petsc/

	Block-ILU(0)			Block-ILU(4)			AMG		
	N_{it}^{int}	T_{it}^{ext}	T_{it}^{int}	N_{it}^{int}	T_{it}^{ext}	T_{it}^{int}	N_{it}^{int}	T_{it}^{ext}	T_{it}^{int}
1p	1027	161	0,16	21	564	27			
2p	1716	179	0,10	684	795	1,16	En échec		
4p	1947	137	0,07	1225	615	0,50			
8p	2518	112	0,04	1667	386	0,23			

Tab. 6.2 : Sur le cas géosciences, en fonction du préconditionneur et du nombre de cœurs utilisés : nombre d'itérations N_{it}^{int} et temps T_{it}^{ext} (en secondes) nécessaires à la convergence de l'algorithme BiCGStab, et temps passé dans chaque itération interne $T_{it}^{int} = T_{it}^{ext}/N_{it}^{int}$. Les valeurs sont des moyennes obtenues sur 10 pas de temps.

(lorsque k augmente, la qualité du préconditionnement est meilleure mais le coût de construction du préconditionneur est plus élevé), et la méthode multigrille AMG via son implémentation dans la bibliothèque Hypre [2]. Pour une introduction à ces méthodes de préconditionnement, voir par exemple [Saa03]. Les méthodes sont utilisées avec leurs paramètres par défaut, et le préconditionneur est appliqué à droite de la matrice. Sur cet exemple, la simulation est stoppée à mi-parcours (après 10 pas de temps). Pour chaque pas de temps, le nombre d'itérations externe est de 1 puisque la méthode est monolithique. On relève la durée de résolution du système linéaire, c'est-à-dire la durée nécessaire à une itération externe T_{it}^{ext}, le nombre d'itérations internes du BiCGStab N_{it}^{int} et le temps nécessaire à une itération interne du solveur $T_{it}^{int} = \frac{T_{it}^{ext}}{N_{it}^{int}}$. Toutes ces valeurs sont ensuite moyennées sur les 10 pas de temps. Cette procédure est répétée en faisant varier le nombre de cœurs utilisés pour le calcul, et les résultats sont présentés dans le tableau 6.2. On peut observer que les préconditionneurs ILU(k) ne sont pas stables vis-à-vis du parallélisme : lorsque le nombre de cœurs augmente, le nombre d'itérations augmente également. En particulier, on observe un écart important entre l'utilisation séquentielle et parallèle, par exemple avec le préconditionneur ILU(4) qui converge en 21 itérations en séquentiel contre presque 700 sur deux cœurs. Même si le temps passé dans une itération interne diminue grâce au parallélisme, la durée d'une itération externe (équivalente ici à la durée nécessaire à la validation d'un pas de temps) peut être plus élevée en parallèle qu'en séquentiel. Pour le préconditionneur AMG, le comportement est encore pire puisqu'en séquentiel comme en parallèle, le solveur atteint la limite des 3000 itérations sans satisfaire le critère d'arrêt, et ce dès le premier pas de temps. Ce cas illustre donc un frein à l'utilisation d'un solveur itératif avec la stratégie monolithique : les préconditionneurs classiques ne sont pas efficaces pour ce problème couplé. Cette limite est en réalité connue, et on compte de nombreux travaux visant à en proposer des nouveaux adaptés à ce système, tels que [BFG07 ; WB11 ; HOL12].

À moins de construire un préconditionneur efficace pour le système couplé, la stratégie monolithique est donc insuffisante. On choisit donc de se tourner vers une stratégie séquentielle, en faisant le pari suivant : certes, pour chaque pas de temps, plusieurs itérations externes seront nécessaires, mais on espère

1. que le nombre d'itérations externes reste raisonnable,
2. que pour chaque sous-problème, le sous-solveur employé soit robuste et rapide,

de manière à être compétitif par rapport à la méthode monolithique, en particulier lors de simulations sur

2. https ://computation.llnl.gov/projects/hypre-scalable-linear-solvers-multigrid-methods

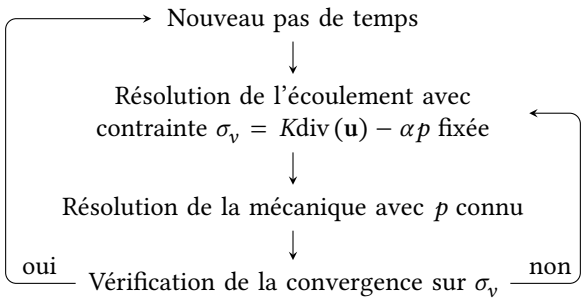


FIG. 6.1 : **Idée générale de la méthode fixed-stress.** Le coefficient K intervenant dans l'expression de la contrainte σ_v peut être ajusté de différentes manières.

plusieurs cœurs.

6.2 Stratégie séquentielle fixed-stress

Pour mettre en place une stratégie séquentielle, on considère la méthode appelée fixed-stress. Introduite dans sa formulation historique puis réinterprétée de manière à faire le lien avec l'algorithme de point fixe sous-jacent, différentes variantes de la stratégie fixed-stress sont décrites et comparées dans cette section.

6.2.1 Formulation de l'algorithme fixed-stress

Historiquement, cette manière de découpler les deux équations est motivée par la physique du problème [SM94]. L'idée générale est décrite par l'organigramme présenté sur la figure 6.1, en utilisant ici les variables et opérateurs continus. À chaque pas de temps, on alterne entre la résolution d'un problème d'écoulement en supposant qu'une certaine quantité est fixée (cette quantité est assimilable à une contrainte, d'où le nom de l'algorithme), et la résolution d'un problème de mécanique en prenant pour la pression la valeur issue de l'étape précédente. Le paramètre K intervenant dans la définition de la contrainte σ_v peut être ajusté : d'abord choisi de manière à rester au plus proche de la physique, d'autres valeurs ont ensuite été proposées dans le but d'accélérer la convergence de la méthode. Pour une comparaison des différentes valeurs admissibles, voir par exemple [BK19]. Dans ce chapitre, on se contentera du choix guidé par la physique en prenant $K = \lambda$, ce paramètre ayant été défini dans la section 2.1.

Avec la méthode fixed-stress, les matrices et seconds membres sont tout de même modifiés par rapport à ceux qui seraient issus de problèmes initialement non couplés. Habituellement (voir par exemple [GWAD19]), on fait apparaître dans l'équation d'écoulement la contrainte moyenne $\sigma_v = \lambda \mathrm{div}(\mathbf{u}) - \alpha p$: formellement, en repartant de l'équation semi-discrète,

$$\left(c_0 + \frac{\alpha^2}{\lambda}\right) p^n + \frac{\alpha}{\lambda} \underbrace{\left(\lambda \mathrm{div}(\mathbf{u}^n) - \alpha p^n\right)}_{\sigma_v^n} + \Delta t \, \mathrm{div}\left(-\overline{\kappa} \boldsymbol{\nabla} p^n\right) = \Delta t q^n + c_0 p^{n-1} + \alpha \mathrm{div}\left(\mathbf{u}^{n-1}\right).$$

Le système est ensuite résolu par une méthode de point fixe en plaçant σ_v^n au membre de droite : en notant l les itérations de point fixe, l'équation d'écoulement devient

$$\left(c_0 + \frac{\alpha^2}{\lambda}\right) p^{n,l} + \Delta t \, \mathrm{div}\left(-\overline{\kappa} \boldsymbol{\nabla} p^{n,l}\right) = \Delta t q^n + c_0 p^{n-1} + \alpha \mathrm{div}\left(\mathbf{u}^{n-1}\right) - \frac{\alpha}{\lambda} \underbrace{\left(\lambda \mathrm{div}\left(\mathbf{u}^{n,l-1}\right) - \alpha p^{n,l-1}\right)}_{\sigma_v^{n,l-1}}. \tag{6.2}$$

Ainsi, seule la grandeur $p^{n,l}$ est une inconnue dans l'équation d'écoulement. Pour l'équation de mécanique, il est clair le terme $p^{n,l}$ venant d'être déterminé, seuls les déplacements sont à calculer :

$$-\mathbf{div}\left(\overline{\overline{C}}\,\overline{\epsilon_{u^{n,l}}}\right) = \mathbf{f} - \alpha\boldsymbol{\nabla} p^{n,l}. \tag{6.3}$$

On propose ici une manière différente d'aboutir aux sous-systèmes, qui se base sur la formulation matricielle du système original couplé (6.1) et qui sera utile pour formuler différentes variantes de la méthode fixed-stress. Cette approche consiste à reformuler le système linéaire en introduisant l'inconnue supplémentaire discrète $\mathcal{S}_v$ (représentant la contrainte σ_v) et l'équation qui lui est associée : puisque $\mathcal{B}^T(\phi_k, \boldsymbol{\varphi}_j^l) = \alpha_K \int_K \mathrm{div}\left(\boldsymbol{\varphi}_j^l\right)$ d'après (4.38), à chaque pas de temps,

$$\mathcal{S}_v = \frac{\lambda_K}{\alpha_K|K|}\mathcal{B}^T\mathcal{U} - \alpha_K\mathcal{P}. \tag{6.4}$$

L'abus de notation $\cdot_K$ dans cette équation fait référence au fait que la ligne k des matrices, qui correspond à la cellule K, est multipliée par les coefficients réels $\cdot_K$ dans le cas où ceux ci ne sont pas constants sur l'ensemble du domaine. La notation $\mathcal{I}$ désigne la matrice identité de taille card($\mathfrak{M}$). Ajouter cette équation au système couplé permet de reformuler la seconde ligne selon

$$\mathcal{L}^p + \mathcal{L}^c = \mathcal{B}^T\mathcal{U} + \mathcal{F}\mathcal{P} = \frac{\alpha_K|K|}{\lambda_K}\mathcal{S}_v + \left(\mathcal{F} + \frac{\alpha_K^2|K|}{\lambda_K}\mathcal{I}\right)\mathcal{P}$$

et donc d'obtenir le système issu des trois équations

$$\begin{bmatrix} \mathcal{I} & -\frac{\lambda_K}{\alpha_K|K|}\mathcal{B}^T & \alpha_K\mathcal{I} \\ 0 & \mathcal{A} & -\mathcal{B} \\ \frac{\alpha_K|K|}{\lambda_K}\mathcal{I} & 0 & \tilde{\mathcal{F}} \end{bmatrix}\begin{bmatrix} \mathcal{S}_v \\ \mathcal{U} \\ \mathcal{P} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathcal{L}^u \\ \mathcal{L}^p + \mathcal{L}^c \end{bmatrix} \quad \text{où} \quad \tilde{\mathcal{F}} = \mathcal{F} + \frac{\alpha_K^2|K|}{\lambda_K}\mathcal{I}. \tag{6.5}$$

La technique du complément de Schur permet alors de définir deux formulations différentes de l'algorithme fixed-stress. En effet, (6.5) est de la forme

$$\begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix}\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 0 \\ G_2 \end{bmatrix}$$

avec

$$Q_{11} = \mathcal{I},\ Q_{12} = \begin{bmatrix} -\frac{\lambda_K}{\alpha_K|K|}\mathcal{B}^T & \alpha_K\mathcal{I} \end{bmatrix},\ Q_{21} = \begin{bmatrix} 0 \\ \frac{\alpha_K|K|}{\lambda_K}\mathcal{I} \end{bmatrix},\ Q_{22} = \begin{bmatrix} \mathcal{A} & -\mathcal{B} \\ 0 & \tilde{\mathcal{F}} \end{bmatrix},\ X_1 = \mathcal{S}_v \text{ et } X_2 = \begin{bmatrix} \mathcal{U} \\ \mathcal{P} \end{bmatrix},$$

où Q_{11} et Q_{22} sont inversibles, et peut donc être réécrit sous la forme du système

$$\begin{aligned} X_1 &= -Q_{12}X_2 \\ X_2 &= Q_{22}^{-1}(G_2 - Q_{21}X_1). \end{aligned} \tag{6.6}$$

Éliminer l'inconnue X_1 dans (6.6) revient à travailler sur un problème dont le vecteur d'inconnues est $X_2 = [\mathcal{U}, \mathcal{P}]^T$:

$$X_2 = Q_{22}^{-1}\left(G_2 + Q_{21}Q_{12}X_2\right) := \mathcal{C}(X_2). \tag{6.7}$$

Inversement, éliminer l'inconnue X_2 dans (6.6) revient à travailler sur un problème dont le vecteur d'inconnues est $X_1 = [\varsigma_v]$:

$$X_1 = -Q_{12}Q_{22}^{-1}\left(G_2 - Q_{21}X_1\right) := \tilde{C}(X_1). \tag{6.8}$$

Il est donc possible de travailler sur l'une ou l'autre des deux variables pour les itérations externes de la méthode fixed-stress. Néanmoins, quelle que soit l'alternative choisie, l'étape au centre de l'algorithme consiste à résoudre des systèmes linéaires dont la matrice associée est Q_{22}.

6.2.2 Interprétation de la stratégie fixed-stress comme une méthode de point fixe

La méthode classique pour progresser dans la résolution externe consiste à appliquer une méthode de point fixe : selon la variable de travail choisie, pour une valeur initiale X_1^0 – resp. X_2^0 donnée, l'opération $X_1^l = \tilde{C}(X_1^{l-1})$ – resp. $X_2^l = C(X_2^{l-1})$ est appliquée jusqu'à convergence. Cette méthode est détaillée dans l'algorithme 3. Dans le cas où la variable de travail est $X_2 = \begin{bmatrix}\mathcal{U} & \mathcal{P}\end{bmatrix}^T$, cet algorithme est équivalent à la

Algorithm 3 Stratégie fixed-stress vue comme une méthode de point fixe

1 Choose initial unknown $X_1^0 = \varsigma_v^0$	1 Choose initial unknown $X_2^0 = \begin{bmatrix}\mathcal{U}^0 & \mathcal{P}^0\end{bmatrix}^T$
2 Compute initial residual r_0	2 Compute initial residual r_0
3 **repeat**	3 **repeat**
4 $X_1^l = \tilde{C}(X_1^{l-1})$:	4 $X_2^l = C(X_2^{l-1})$:
5 Compute r	5 Compute r
6 $X_1^{l-1} = X_1^l$	6 $X_2^{l-1} = X_2^l$
7 **until** $r \leq r_0\epsilon^{\text{ext}}$ or $l > N_{\max}$	7 **until** $r \leq r_0\epsilon^{\text{ext}}$ or $l > N_{\max}$
8 **function** $\tilde{C}(X)$	8 **function** $C(X)$
9 $X_1 = X$	9 $X_2 = X$
10 $X_2 = Q_{22}^{-1}(G_2 - Q_{21}X_1)$	10 $X_1 = -Q_{12}X_2$
11 $X_1 = -Q_{12}X_2$	11 $X_2 = Q_{22}^{-1}(G_2 - Q_{21}X_1)$

formulation historique de la méthode fixed-stress : en effet, (6.7) se réécrit

$$\begin{bmatrix}\mathcal{U}^l \\ \mathcal{P}^l\end{bmatrix} = \begin{bmatrix}\mathcal{A} & -\mathcal{B} \\ 0 & \tilde{\mathcal{F}}\end{bmatrix}^{-1}\begin{bmatrix}\mathcal{L}^u \\ \mathcal{L}^p + \mathcal{L}^c + \mathcal{L}^{\sigma,l-1}\end{bmatrix} \quad \text{avec} \quad \mathcal{L}^{\sigma,l-1} = -\mathcal{B}^T\mathcal{U}^{l-1} + \frac{\alpha_K^2|K|}{\lambda_K}\mathcal{P}^{l-1},$$

et n'est autre que la formulation matricielle issue de la discrétisation des équations (6.2)–(6.3). Finalement, par rapport à un problème fluide indépendant, l'équation en pression est modifiée en ajoutant au second membre la quantité $\mathcal{L}^c + \mathcal{L}^\sigma$ (calculée à partir de l'itération fixed-stress précédente pour $\mathcal{L}^\sigma$ et du pas de temps précédent pour $\mathcal{L}^c$), et à la matrice $\mathcal{F}$ la quantité $\frac{\alpha_K^2|K|}{\lambda_K}\mathcal{I}$. L'équation de mécanique est résolue dans un second temps en utilisant la pression qui vient d'être calculée : $\mathcal{A}\mathcal{U}^l = \mathcal{L}^u + \mathcal{B}\mathcal{P}^l$.

Dans le cas où la variable de travail est $X_1 = \varsigma_v$, le lien avec la méthode fixed-stress historique est soumis à condition : (6.7) se réécrit

$$\varsigma_v^l = \begin{bmatrix}\frac{\lambda_K}{\alpha_K|K|}\mathcal{B}^T & -\alpha_K\mathcal{I}\end{bmatrix}\begin{bmatrix}\mathcal{A} & -\mathcal{B} \\ 0 & \tilde{\mathcal{F}}\end{bmatrix}^{-1}\begin{bmatrix}\mathcal{L}^u \\ \mathcal{L}^p + \mathcal{L}^c + \mathcal{L}^{\sigma,l-1}\end{bmatrix} \quad \text{avec} \quad \mathcal{L}^{\sigma,l-1} = -\frac{\alpha_K|K|}{\lambda_K}\varsigma_v^{l-1}.$$

La résolution du système linéaire donne donc deux variables temporaires $\mathcal{U}^l$ et $\mathcal{P}^l$, correspondant à X_2 dans la partie gauche de l'algorithme 3 (l 10), et utilisées pour mettre à jour ς_v^l. *A priori*, les deux versions

de la méthode fixed-stress produisent des grandeurs itérées différentes à chaque itération ; cependant, dans le cas particulier où

$$\varsigma_v^0 = -Q_{12}\begin{bmatrix}\mathcal{U}^0 & \mathcal{P}^0\end{bmatrix}^T, \tag{6.9}$$

et donc où les variables sont compatibles à l'instant initial, les deux versions de l'algorithme produisent en réalité à chaque itération les mêmes grandeurs X_1^l et X_2^l. Pour montrer cette propriété, on utilise la définition des fonctions $\tilde{C}$ et C dans l'algorithme 3 (l9-l13) en notant $\tilde{X}_1$ et $\tilde{X}_2$ les grandeurs de la version itérant avec la fonction $\tilde{C}$ pour les différencier.

La démonstration s'effectue par récurrence : supposons qu'à l'instant $l-1$, $\tilde{X}_1^{l-1} = -Q_{12}X_2^{l-1}$. Alors

$$\tilde{X}_2^l = Q_{22}^{-1}(G_2 - Q_{21}\tilde{X}_1^{l-1}) = Q_{22}^{-1}(G_2 + Q_{21}Q_{12}X_2^{l-1}) = Q_{22}^{-1}(G_2 - Q_{21}X_1^l) = X_2^l$$

puis

$$\tilde{X}_1^l = -Q_{12}\tilde{X}_2^l = -Q_{12}X_2^l = X_1^l.$$

En tenant compte de l'hypothèse initiale (6.9) $\tilde{X}_1^0 = -Q_{12}X_2^0$, la seconde égalité établit la récurrence et montre que à chaque itération l, $\tilde{X}_1^l = X_1^l$ et $\tilde{X}_2^l = X_2^l$. Pour conserver l'équivalence de ces deux variantes, on prendra donc comme valeur initiale la grandeur issue du pas de temps précédent :

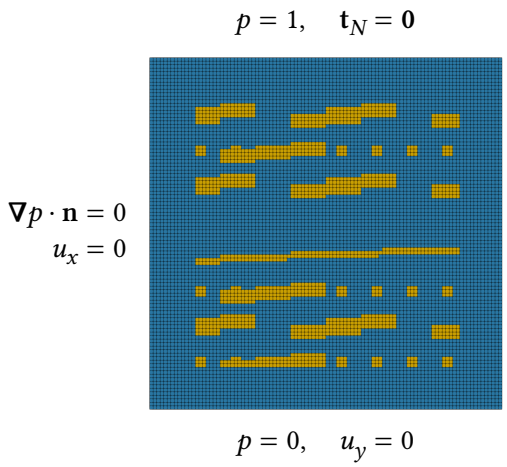
$$\begin{aligned} X_2^{n,0} &= \begin{bmatrix}\mathcal{U}^{n-1} & \mathcal{P}^{n-1}\end{bmatrix}^T && \text{pour la formulation en } X_2 \\ X_1^{n,0} &= \varsigma_v^{n-1} = -Q_{12}\begin{bmatrix}\mathcal{U}^{n-1} & \mathcal{P}^{n-1}\end{bmatrix}^T && \text{pour la formulation en } X_1. \end{aligned}$$

Pour achever la description de la méthode, il reste à choisir la méthode à utiliser pour calculer les résidus r_0 et r. Le choix standard pour un algorithme de point fixe consiste à calculer la différence entre deux itérations successives :

$$\begin{aligned} r(X_2) &= \left\| C\left(\begin{bmatrix}\mathcal{U}^{l-1} \\ \mathcal{P}^{l-1}\end{bmatrix}\right) - \begin{bmatrix}\mathcal{U}^{l-1} \\ \mathcal{P}^{l-1}\end{bmatrix}\right\|_{L^2} && \text{et } r_0 = \|C(0)\|_{L^2} \text{ pour la formulation en } X_2 \\ r(X_1) &= \|\tilde{C}(\varsigma_v^{l-1}) - \varsigma_v^{l-1}\|_{L^2} && \text{et } r_0 = \|\tilde{C}(0)\|_{L^2} \text{ pour la formulation en } X_1. \end{aligned}$$

Dans notre cas, lorsque la variable de travail utilisée est X_2, d'autres possibilités sont offertes. Il est par exemple possible avant et après l'itération de calculer les contraintes ς_v^{l-1} et ς_v^l selon la relation (6.4). On peut alors calculer le résidu sur la différence entre la variable de contrainte plutôt que sur celle entre les variables primaires selon $r = \|\varsigma_v^l - \varsigma_v^{l-1}\|_{L^2}$. À nouveau, on normalise en appliquant la méthode au vecteur nul : $r_0 = \|\tilde{C}(0)\|_{L^2}$. Calculer le résidu sur la contrainte alors que les itérations sont effectuées sur les variables primaires est une approche parfois adoptée dans la littérature, et qui trouve ici sa justification dans la formulation en $X_1 = \varsigma_v$ de la méthode. Une troisième possibilité, pouvant être utile à des fins de tests, consiste à assembler la matrice monolithique M^{mono} et le second membre monolithique rhs^{mono} même en cas de résolution séquentielle, et à utiliser ces quantités pour calculer un résidu selon $r = \|M^{\text{mono}}\begin{bmatrix}\mathcal{U}^l & \mathcal{P}^l\end{bmatrix}^T - \text{rhs}^{\text{mono}}\|_{L^2}$. Dans ce cas, la quantité utilisée pour normaliser est calculée avec la même formule appliquée au vecteur nul de sorte que $r_0 = \|\text{rhs}^{\text{mono}}\|_{L^2}$.

Lorsque la variable de travail utilisée est X_1, il n'y a pas vraiment d'autre moyen de définir un résidu en dehors de celui propre à la méthode de point fixe défini plus haut. En effet, on ne sait pas remonter aux vecteurs $\mathcal{U}$ et $\mathcal{P}$ à partir d'un vecteur ς_v donné. Il est notamment impossible de réutiliser le système monolithique comme dans le cas précédent.



	Mat1	Mat2
k (m^2)	1	10^3
μ_f (Pa·s)	10^{-3}	10^{-3}
E (Pa)	10	1
ν (-)	0.22	0.22
c_0 (Pa^{-1})	1	0.1
α (-)	0.9	0.9

FIG. 6.2 : **Domaine hétérogène bidimensionnel** et caractéristiques physiques des deux matériaux.

6.2.3 Comparaison des différentes variantes

Pour une première illustration de l'algorithme fixed-stress, de l'influence du choix de la variable inconnue et de celui du critère d'arrêt, on considère un nouveau cas test bidimensionnel hétérogène inspiré de [BV16] et [CKHT19]. Le carré unité, domaine de calcul, est constitué de deux matériaux et est représenté sur la figure 6.2. Celui-ci est discrétisé par 400×400 mailles carrées (pour plus de lisibilité, la figure ne représente qu'un maillage de taille 100×100). Les constantes mécaniques des matériaux sont également indiquées sur la figure. À l'instant initial $t = 0$, la pression est nulle. L'évolution de ce problème est due à ses conditions aux limites, représentées sur la figure. Le système est résolu au bout d'un unique pas de temps $\Delta t = T = 5$ secondes.

Le nombre d'inconnues est suffisamment faible pour résoudre les sous-problèmes de manière directe. La tolérance de l'algorithme fixed-stress est pour sa part fixée à 10^{-6}. Dans un premier temps, on compare les trois manières de calculer le résidu (et donc le critère d'arrêt) en utilisant la formulation en variables $(\mathbf{u}, p)$. L'évolution du résidu en fonction du nombre d'itérations externes est représentée à gauche de la figure 6.3. Les deux méthodes qui calculent la norme L^2 de la différence entre deux itérations fixed-stress successives donnent un résultat proche : l'algorithme converge en 13 itérations lorsque la différence est directement calculée sur $(\mathbf{u}, p)$ et en 14 itérations lorsqu'on déduit σ_v à partir des variables primaires pour calculer la différence. La méthode faisant appel au système monolithique donne des résultats différents, et semble converger plus rapidement. Cela dit, la comparaison est difficile puisque dans ce dernier cas la valeur du résidu est affectée par la matrice M^{mono}, et sa normalisation est différente : on observe sur le graphe un écart de deux ordres de grandeurs. Pour ce cas, on dispose d'une solution de référence calculée avec la stratégie monolithique–résolution directe. Il est donc possible de vérifier en fin de calcul l'écart sur les variables $\mathbf{u}$ et p entre la solution de référence et l'approche séquentielle. Pour les deux premières manières de calculer le résidu, l'écart est de l'ordre de 10^{-7} alors qu'il est de l'ordre de 10^{-5} avec la troisième. Dans ce dernier cas, la méthode s'est donc stoppée plus tôt (d'où le nombre d'itérations plus faible) mais a atteint une solution moins précise. Quoi qu'il en soit, cette troisième méthode n'est pas vraiment utilisable dans une application réelle puisqu'elle demande l'assemblage du système monolithique en plus des sous-systèmes et est donc plus coûteuse. Dans un second temps, on compare le choix de la variable utilisée en tant que quantité à faire converger par la méthode de point fixe. Le résidu est calculé selon la première méthode, et son évolution en fonction du nombre d'itérations externes est représentée à droite de la figure

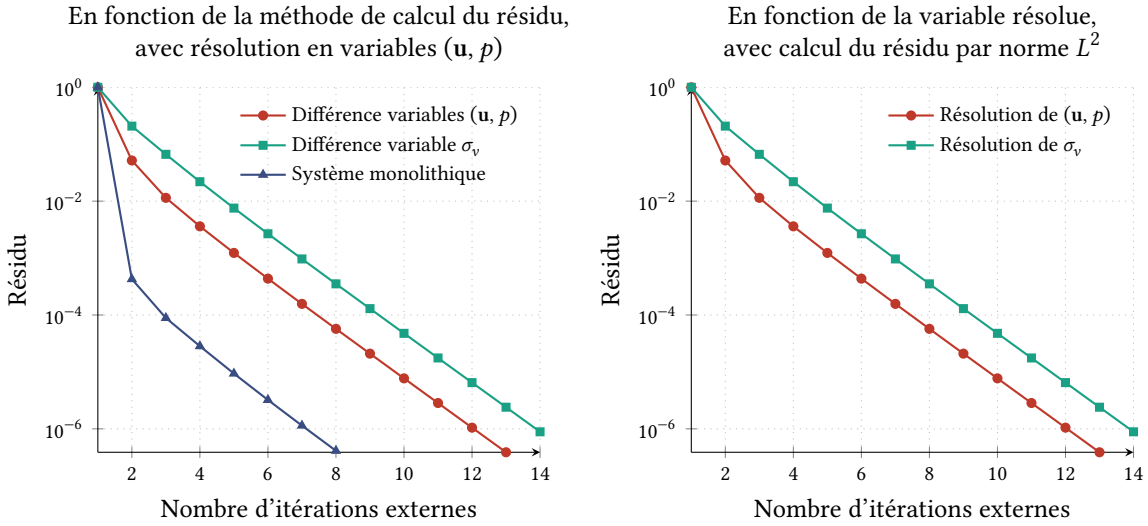


FIG. 6.3 : **Comportement de différentes variantes de l'algorithme fixed-stress** implémenté en tant que méthode de point fixe sur le cas hétérogène 2D.

6.3. Le nombre d'itérations des deux formulations est presque identique. On retrouve également le fait que l'utilisation de la variable σ_v avec son calcul de résidu standard (courbe verte à droite) donne les mêmes résultats que l'utilisation des variables $(\mathbf{u}, p)$ avec calcul de résidu sur σ_v (courbe verte à gauche), ceci étant dû à l'équivalence des deux formulations. Compte tenu de la proximité des deux approches, il peut être préférable de privilégier la formulation en variable σ_v, qui exécute l'algorithme de point fixe sur un vecteur de taille plus petite (une inconnue par cellule contre une inconnue par cellule plus d inconnues par nœud pour la formulation $(\mathbf{u}, p)$). Il faut toutefois être conscient que la charge de calcul est majoritairement portée par la résolution des sous-problèmes, et donc ne pas attendre de ce choix une amélioration significative des performances.

Pour achever cette section, on relève la durée de simulation de la stratégie séquentielle : pour ses 14 itérations, la méthode de point fixe a consommé 260 secondes, soit moins de 20 secondes pour chaque itération de point fixe. À titre de comparaison, l'approche monolithique nécessite sur ce cas 80 secondes pour résoudre le système couplé. Résoudre deux "petits" systèmes découplés est donc plus rapide que résoudre un "grand" système couplé. La méthode fixed-stress est au final plus lente en raison de son nombre important d'itérations, qui dépend d'ailleurs de la précision exigée. Même si la robustesse reste l'objectif prioritairement recherché lors de la mise en place de l'approche séquentielle (voir le paragraphe 6.1.3), il est toujours souhaitable d'améliorer la vitesse de la méthode.

6.3 Accélération de la stratégie fixed-stress

Une des pistes envisageables pour réduire le temps de calcul de la stratégie fixed-stress consiste à faire diminuer le nombre d'itérations de celle-ci. Il s'agit de l'objet de cette section, où l'algorithme est réécrit dans le formalisme des méthodes de Krylov.

6.3.1 Interprétation de la stratégie fixed-stress comme une méthode de Krylov

Dans la section précédente, la stratégie fixed-stress a été présentée comme un algorithme visant à résoudre le problème $\mathcal{C}(X) = X$, avec X désignant la variable de travail choisie, *cf* paragraphe 6.2.1. La méthode alors adoptée pour résoudre ce problème, en accord avec la présentation classique de l'algorithme fixed-stress, était une méthode de point fixe (voir l'algorithme 3). Or, on peut reformuler le même problème comme la résolution du problème linéaire $\mathcal{R}(X) = 0$ avec $\mathcal{R}(X) = \mathcal{C}(X) - X$, et utiliser une méthode de Krylov pour en calculer la solution.

Dans les ouvrages, les méthodes de Krylov sont souvent formulées dans le but de résoudre un système linéaire $Ay = b$ et font donc intervenir la matrice A et le second membre b. Une première étape consiste donc à réécrire ces algorithmes dans leur version "matrix-vector product free", c'est-à-dire en ne faisant intervenir que la fonction résidu $\mathcal{R}$ (alors définie par $\mathcal{R}(y) = b - Ay$ dans le cas de ce système). Pour la méthode du BiCGStab, ce travail est effectué dans l'algorithme 4. Dans celui-ci, la notation (u, v) désigne le produit scalaire discret de L^2. Dans cet algorithme, il existe un lien entre le résidu r^k utilisé dans le test

Algorithm 4 BiCGStab dans sa version originale (gauche) et matrix-vector product free (droite)

X^0 given	X^0 given
$r^0 = b - AX^0$	$r^0 = \mathcal{R}(X^0); b = \mathcal{R}(0)$
$\hat{r}^0 = p^0 = r^0$	$\hat{r}^0 = p^0 = r^0$
repeat	**repeat**
$\alpha = (r^k, \hat{r}^0)/(Ap^k, \hat{r}^0)$	$\alpha = (r^k, \hat{r}^0)/(b - \mathcal{R}(p^k), \hat{r}^0)$
$s^k = r^k - \alpha Ap^k$	$s^k = r^k - \alpha(b - \mathcal{R}(p^k))$
$\omega = \frac{(As^k, s^k)}{\|As^k\|^2}$	$\omega = \frac{(b - \mathcal{R}(s^k), s^k)}{\|b - \mathcal{R}(s^k)\|^2}$
$X^k \leftarrow X^k + \alpha p^k + \omega s^k$	$X^k \leftarrow X^k + \alpha p^k + \omega s^k$
$r^{k+1} = s^k - \omega As^k$	$r^{k+1} = s^k - \omega(b - \mathcal{R}(s^k))$
$\beta = \alpha(r^{k+1}, \hat{r}^0)/\omega(r^k, \hat{r}^0)$	$\beta = \alpha(r^{k+1}, \hat{r}^0)/\omega(r^k, \hat{r}^0)$
$p^k \leftarrow r^{k+1} + \beta(p^k - \omega Ap^k)$	$p^k \leftarrow r^{k+1} + \beta(p^k - \omega(b - \mathcal{R}(p^k)))$
until $\|r^{k+1}\| \leq \epsilon^{\text{ext}}\|r_0\|$ or $k > k_{\max}$	**until** $\|r^{k+1}\| \leq \epsilon^{\text{ext}}\|r_0\|$ or $k > k_{\max}$

d'arrêt et l'opérateur résidu $\mathcal{R}$ utilisé dans la boucle retranscrit par la proposition suivante :

Proposition 6.1 *Si l'opérateur $\mathcal{R}$ est affine* ie $\forall X, \mathcal{R}(X) = \mathcal{R}(0) + \mathcal{R}^{lin}(X)$ *avec $\mathcal{R}^{lin}$ linéaire, alors*

$$r^k = \mathcal{R}(X^k)$$

dans l'algorithme du BiCGStab.

La démonstration s'effectue par récurrence : par définition, $r^0 = \mathcal{R}(X^0)$. En supposant que la proposition est vrai au rang k,

$$\begin{aligned} r^{k+1} &= r^k - \alpha(\mathcal{R}(0) - \mathcal{R}(p^k)) - \omega(\mathcal{R}(0) - \mathcal{R}(s^k)) \\ &= \mathcal{R}(X^k) - \alpha(\mathcal{R}(0) - \mathcal{R}(p^k)) - \omega(\mathcal{R}(0) - \mathcal{R}(s^k)) \\ &= \mathcal{R}(0) + \mathcal{R}^{\text{lin}}(X^k) + \alpha\mathcal{R}^{\text{lin}}(p^k) + \omega\mathcal{R}^{\text{lin}}(s^k) \\ &= \mathcal{R}(0) + \mathcal{R}^{\text{lin}}(X^{k+1}) = \mathcal{R}(X^{k+1}). \end{aligned}$$

Ce lien permet d'assurer que lorsque le critère d'arrêt de la méthode de Krylov est satisfait, le problème $\mathcal{R}(X) = 0$ est bien résolu (à la tolérance près).

De la même manière, la méthode Gmres [Saa03, section 6.5], autre méthode de Krylov, est implémentée dans sa version sans produit matrice vecteur afin de l'utiliser pour résoudre le problème de minimisation du résidu fixed-stress. Cette procédure est décrite dans l'algorithme 5. Par rapport au BiCGStab, cette méthode

Algorithm 5 Algorithme Gmres dans sa version matrix-vector product free

1 X^0 given
2 $r = \mathcal{R}(X^0)\,; \quad b = r^0 = \mathcal{R}(0)\,;$
3 β, c and s lists of size $k_{\max}+1$ ▷ Initialise empty arrays
4 V array of $k_{\max}$ vectors sized as X
5 $\beta_0 = \|r\|_{L^2}\,; \quad V^0 = r/\|r\|_{L^2}$
6 **repeat**
7 $\quad \omega = b - \mathcal{R}(V^k)$
8 $\quad$ **for** $i = 0, \cdots, k$ **do** ▷ Build an orthonormalized basis
9 $\quad\quad H_{ik} = (\omega, V^i)$
10 $\quad\quad \omega \mathrel{-}= H_{ik} V^i$
11 $\quad H_{k+1,k} = \|\omega\|_{L^2}$
12 $\quad$ **if** $\|\omega\|_{L^2} \neq 0$ **then** $V^{k+1} = \omega/\|\omega\|_{L^2}$
13 $\quad$ **for** $i = 0, \cdots, k-1$ **do** ▷ Apply Given's rotations to triangulate H
14 $\quad\quad \begin{aligned} H_{ik} &= c_i H_{ik} + s_i H_{i+1,k} \\ H_{i+1,k} &= -s_i H_{ik} + c_i H_{i+1,k} \end{aligned}$
15 $\quad c_k = \frac{H_{kk}}{\sqrt{H_{kk}^2 + H_{k+1,k}^2}} \quad \text{and} \quad s_k = \frac{H_{k+1,k}}{\sqrt{H_{kk}^2 + H_{k+1,k}^2}}$
16 $\quad\quad \begin{aligned} H_{kk} &= c_k H_{kk} + s_k H_{k+1,k} \\ H_{k+1,k} &= 0 \end{aligned}$
17 $\quad\quad \begin{aligned} \beta_{k+1} &= -s_k \beta_k \\ \beta_k &= c_k \beta_k \end{aligned}$
18 $\quad \|r\| = |\beta_{k+1}|$ ▷ Get the residual norm for free !
19 **until** $\|r\| \leq \epsilon^{\text{ext}} \|r^0\|$ or $k > k_{\max}$
20 Solve upper triangular system $Hy = \beta$ ▷ Gmres is converged, now let's recover X
21 $X = X^0 + \sum_{i=0}^{k} y_i V^i$

consomme plus mémoire puisqu'à l'itération k, elle requiert tous les vecteurs $V^i, i < k$ pour construire la base orthonormalisée (l8) et doit donc conserver ces vecteurs de taille égale à celle de X. En revanche, elle a l'avantage de n'appeler qu'une seule fois la procédure de calcul du résidu $\mathcal{R}$ (l7). Cet avantage est dû à l'utilisation des rotations de Given (l13) pour résoudre le problème des moindres carrés $Hy = \beta$, qui permet de connaître la norme du résidu $\|r\|_{L^2}$ gratuitement *avant* d'avoir calculé la solution X (l18). Une version moins sophistiquée consisterait à résoudre à chaque itération le problème des moindres carrés, déterminer X^k puis appeler $\mathcal{R}(X^k) = r$ pour vérifier si l'algorithme a convergé ou non, mais cette solution nécessite deux appels à $\mathcal{R}$ par itération.

Dans le cas de la stratégie séquentielle fixed-stress, il suffit donc d'appliquer la version sans produit matrice vecteur de l'algorithme BiCGStab ou de l'algorithme Gmres en utilisant la procédure $\mathcal{C}$ définie par (6.7) ou (6.8) selon le choix de la variable X lors des appels à la fonction résidu $\mathcal{R}(X) = \mathcal{C}(X) - X$. Comme dans la méthode du point fixe, le vecteur initial X^0 est calculé à partir des inconnues au pas de temps précédent et, lors du test d'arrêt, la valeur r_0 utilisée pour normaliser correspond à $\mathcal{R}(0)$. Compte

tenu des besoins en mémoire plus importants de Gmres, mais directement proportionnels à la taille du vecteur inconnu, il est préférable d'utiliser la formulation en variable σ_v qui est de taille bien inférieure au vecteur d'inconnues $(\mathbf{u}, p)$ puisque les deux formulations sont équivalentes d'après la section 6.2. Enfin, on note que lorsque la méthode séquentielle s'arrête, le dernier appel à la procédure C a été effectué avec une variable temporaire : p^k dans le cas du BiCGStab et V^k dans le cas de Gmres. Dans l'implémentation effectuée, il faut rappeler C sur le dernier vecteur X^k pour mettre à jour correctement les déplacements et la pression.

6.3.2 Comparaison numérique des algorithmes

Les stratégies fixed-stress en tant que méthode de Krylov ou en tant qu'algorithme de point fixe sont comparées dans les conditions du cas test de la section 6.2.3. Pour les trois approches, la variable de calcul choisie est $X = \varsigma_v$ et le résidu est calculé par l'approche naturelle à chaque méthode. À gauche de la figure 6.4, l'évolution du résidu est représentée au fil des itérations de la méthode fixed-stress jusqu'à satisfaire le critère de convergence avec une tolérance ϵ^{ext} fixée à 10^{-6}. Contrairement à la méthode de point fixe, dont la convergence est linéaire, les méthodes de Krylov atteignent plus rapidement le critère de convergence : 5 itérations pour le BiCGStab et 8 pour Gmres. Le temps moyen passé dans chaque itération est de 19 secondes pour la méthode de point fixe, 38 secondes pour l'algorithme du BiCGStab et 21 secondes pour la méthode Gmres. Comme attendu, la méthode du BiCGStab est pénalisée par le fait qu'elle appelle deux fois la résolution des sous-systèmes, partie la plus coûteuse, à chaque itération ; si bien qu'au final, la durée totale de la méthode fixed-stress est moins longue avec Gmres qu'avec le BiCGStab.

Les écarts entre la méthode de point fixe et les méthodes de Krylov se creusent lorsque la tolérance ϵ^{ext} est abaissée, comme illustré par le graphique de droite sur la figure 6.4 qui représente le nombre d'itérations des trois méthodes en fonction de la tolérance. Si l'on tient compte du fait que la méthode du BiCGStab appelle deux fois la fonction résidu pour chaque itération (courbe en pointillés verts), la méthode Gmres semble être la plus économe. L'écart entre les deux méthodes de Krylov reste toutefois faible par rapport à l'écart entre une méthode de Krylov et l'algorithme de point fixe.

Cette accélération permet donc de réduire le nombre d'itérations de la méthode fixed-stress. Il s'agit également d'un moyen d'améliorer la fiabilité de l'algorithme fixed-stress : si la méthode de point fixe peut parfois ne pas converger, les méthodes de Krylov sont connues pour être plus robustes. Pour compléter cette optimisation, il reste à choisir des sous-solveurs itératifs robustes pour faire face aux difficultés relevées dans la section 6.1.3 et remplacer les sous-solveurs directs qui ne sont pas utilisables dans les cas réels.

6.4 Recherche de sous-solveurs efficaces pour les sous-problèmes

On cherche dans cette section à qualifier les sous-solveurs à utiliser pour la résolution des sous-problèmes de la méthode fixed-stress. Pour effectuer cette sélection, on sort du cadre poroélastique en considérant des problèmes découplés.

6.4.1 Préconditionneurs ILU et AMG

Problème en pression

Le problème découplé d'écoulement est créé à partir du cas hétérogène décrit dans la section 6.2.3. Le maillage et la répartition en deux matériaux sont conservés, avec les mêmes conditions aux limites et initiales pour la pression. La simulation court toujours sur un unique pas de temps ($\Delta t = 5$ s), et la

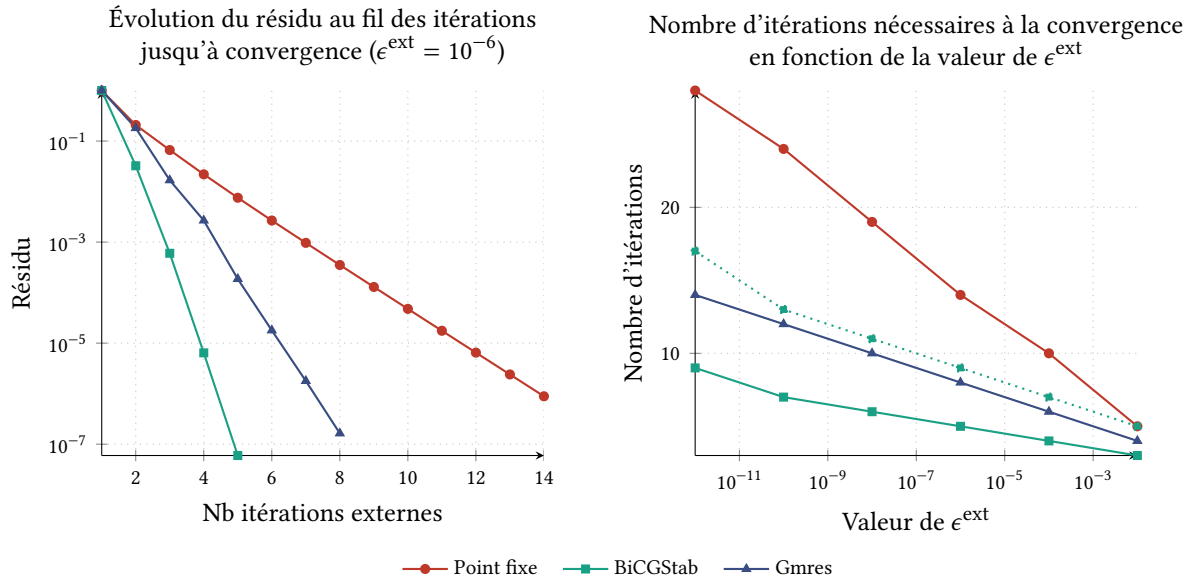


FIG. 6.4 : **Comparaison de la méthode de point fixe et des méthodes de Krylov** pour faire converger l'algorithme fixed-stress. À droite en pointillés, les itérations du BiCGStab sont multipliées par 2 pour tenir compte du double appel à la fonction $\mathcal{R}$ spécifique à cette méthode. Sur le cas synthétique hétérogène 2D, les méthodes de Krylov nécessitent moins d'itérations.

grandeur physique variant selon le domaine est la mobilité $\overline{\kappa}$ (supposée isotrope, $\overline{\kappa} = \kappa \overline{I_2}$) : dans le cas dit homogène, $\kappa = 10^{-3}$ m^2(Pa·s)$^{-1}$ dans tout le domaine, tandis que dans le cas dit hétérogène, $\kappa = 10^{-3}$ m^2(Pa·s)$^{-1}$ dans la zone bleue et $\kappa = 1$ m^2(Pa·s)$^{-1}$ dans la zone jaune. Dans les deux cas, $c_0 = 1$ Pa^{-1} dans l'ensemble du domaine. Le problème est résolu par le solveur itératif BiCGStab avec une tolérance ϵ^{int} de 10^{-8} et un préconditionnement à droite. À nouveau, les préconditionneurs ILU(k), $k = 0$ ou $k = 4$ ainsi que le préconditionneur AMG sont testés pour un calcul séquentiel ou parallèle jusqu'à 8 cœurs. Le nombre d'itérations du solveur selon la configuration est rassemblé dans le tableau 6.3. Dans le cas homogène, en dehors de ILU(4) qui souffre du passage en parallèle, tous les préconditionneurs testés sont relativement robustes lorsque le nombre de cœurs augmente : leur nombre d'itérations reste plus ou moins stable. En revanche, dans le cas hétérogène, la résolution séquentielle est déjà très difficile pour les préconditionneurs ILU, qui effectuent quasiment 10 fois plus d'itérations que dans le cas homogène, et devient impossible pour ceux-ci à partir d'un certain nombre de cœurs. Le préconditionnement par la méthode AMG s'avère au contraire extrêmement robuste pour ce problème. Le nombre d'itérations reste constant quel que soit la configuration et est de plus extrêmement faible par rapport aux autres méthodes. Ce préconditionneur semble donc utilisable pour la résolution du problème d'écoulement de fluide dans le cadre de la méthode séquentielle.

Problème en déplacement

Pour créer un problème découplé de mécanique à partir du cas hétérogène couplé, il faut modifier les conditions aux limites ou le chargement qui étaient tous les deux nuls, de manière à créer une déformation non nulle. On impose donc sur la face supérieure une traction $\mathbf{t}_N = -0.1\mathbf{e}_y$, les autres conditions aux limites étant inchangées. À l'instar du problème de fluide, les constantes des matériaux sont ajustables de manière

	homogène				hétérogène			
	1p	2p	4p	8p	1p	2p	4p	8p
Aucun	389	431	391	390	x	x	x	x
Block-ILU(0)	110	127	121	131	1424	>	x	x
Block-ILU(4)	36	115	121	120	360	1899	1360	x
AMG	3	3	3	3	4	3	3	3

Tab. 6.3 : Comparaison du nombre d'itérations du BiCGStab sur le problème d'écoulement (160K inconnues) en fonction du préconditionnement. Le symbole (x) indique un échec de la résolution, tandis que le symbole (>) signifie que le nombre d'itérations maximum (2000) est dépassé. Le préconditionneur AMG est robuste pour ce problème.

	homogène				hétérogène			
	1p	2p	4p	8p	1p	2p	4p	8p
Aucun	>	>	1989	>	x	>	x	x
Block-ILU(0)	352	455	434	456	1519	x	>	>
Block-ILU(4)	132	422	442	464	368	>	x	1964
AMG	124	126	127	120	365	355	372	341

Tab. 6.4 : Comparaison du nombre d'itérations du BiCGStab sur le problème mécanique (321K inconnues) en fonction du préconditionnement. Le symbole (x) indique un échec de la résolution, tandis que le symbole (>) signifie que le nombre d'itérations maximum (2000) est dépassé. Aucun des préconditionneurs n'est robuste pour ce problème.

à obtenir un problème homogène ou hétérogène. C'est ici le module de Young qui permet d'aboutir à ces deux situations, en le fixant à 1 Pa dans l'ensemble du domaine dans le cas homogène ou bien en le fixant à 1 Pa dans le matériau bleu et 100 Pa dans le matériau jaune dans le cas hétérogène. Le coefficient de Poisson $\nu = 0.25$ reste homogène dans les deux cas. On note qu'il n'y a en mécanique plus de notion de pas de temps puisque le problème est statique. Comme dans le cas précédent, les préconditionneurs ILU(0), ILU(4) et AMG sont évalués avec le solveur BiCGStab sur un ou plusieurs cœurs. Le critère d'arrêt est cette fois fixé à 10^{-6} pour accélérer les tests. Le nombre d'itérations du solveur selon la configuration est retranscrit dans le tableau 6.4. Par rapport au problème d'écoulement, on note que le problème mécanique est bien plus difficile à résoudre. Même dans le cas homogène, où tous les préconditionneurs convergent, le nombre d'itérations est élevé dans l'absolu. En particulier, le préconditionneur AMG, qui était très efficace pour le problème de fluide, nécessite ici autour de 120 itérations. C'est certes moins que les méthodes ILU(k), mais cela reste assez coûteux. Cette quantité a toutefois le mérite de rester quasiment constante lorsque le nombre de cœurs augmente. En revanche, même le préconditionneur AMG supporte mal le passage au cas hétérogène, qui se traduit par un triplement du nombre de ses itérations. À la différence du problème d'écoulement, ce préconditionneur n'est pas stable vis-à-vis des coefficients physiques. On note toutefois que celui-ci a été utilisé avec ses options par défaut, et que ses résultats auraient pu être améliorés avec un paramétrage plus fin, par exemple en indiquant les vecteurs associés aux mouvements de corps rigide (*cf* section 2.1.1) qui sont dans le noyau de la matrice. Les préconditionneurs ILU(k) sont quand à eux presque systématiquement en échec sur les cas parallèles et donc insuffisamment fiables pour

être appliqué à la résolution du problème de mécanique. La conclusion de ce test diffère donc de celle du précédent : les préconditionneurs classiquement utilisés ne semblent pas assez robustes pour envisager leur usage comme sous-solveur de la partie mécanique dans la stratégie séquentielle.

6.4.2 Méthode de décomposition de domaine pour l'élasticité

Au tout début de ce chapitre, deux catégories de méthodes de résolution de système linéaire ont été mentionnées : les méthodes directes et les méthodes itératives. Pour trouver un préconditionneur robuste pour le système d'élasticité, les méthodes à présent investiguées appartiennent à une troisième catégorie visant à combiner les avantages des deux précédentes : les méthodes hybrides (catégorie à laquelle appartiennent également les méthodes multigrilles algébriques, soit dit en passant). Il s'agit ici de méthodes de décomposition de domaine, décrites en détails dans [DJN15], qui combinent des solveurs directs (robustes) utilisés sur des sous-domaines pour construire le préconditionneur (naturellement parallèle) d'une méthode itérative. L'étude de la robustesse de ces méthodes a fait l'objet de plusieurs travaux, parmi lesquels la thèse de N. Spillane [Spi14], traitant entre autres le système d'élasticité discrétisé par éléments finis. Il est ainsi connu que les méthodes de décomposition de domaine à un niveau peuvent souffrir de problèmes de robustesse, mais qu'il existe un cadre théorique permettant de corriger ce comportement. Ce problème de robustesse est dû à l'échange limité d'informations entre les sous-domaines, celles-ci n'étant transmises que de voisin à voisin. Le remède prend donc la forme de la résolution d'un problème dit grossier, commun à tous les sous-domaines, et soigneusement choisi afin de représenter une approximation du problème initial. On parle de méthode de décomposition de domaine à deux niveaux pour désigner l'utilisation de ce problème grossier. Dans cette section, on décrit brièvement les méthodes utilisées pour la création d'un préconditionneur utilisant un ou deux niveaux de décomposition de domaine. Une illustration similaire à celle de la section précédente est ensuite proposée.

Méthode à un niveau

Le préconditionneur à un niveau est issu de la méthode de Schwarz additive restrictive (voir [Gano8] pour une description et un historique des méthodes de Schwarz), sous-catégorie populaire des méthodes de décomposition de domaine, et sa description fait appel à la terminologie suivante.

Le maillage τ_h est décomposé en N_p sous-maillages $\tau_{h,i}$ pouvant éventuellement se recouvrir de sorte que $\tau_h = \cup_{i=1}^{N_p} \tau_{h,i}$. Cette décomposition permet de définir naturellement sur chaque sous-domaine un espace d'approximation local $V_{h,i}$. On obtient donc une décomposition des degrés de liberté globaux, notés ici $\mathcal{N}$, en degrés de liberté locaux $\mathcal{N}_i$ définis sur l'espace d'approximation associé au sous-domaine $\tau_{h,i}$: $\mathcal{N} = \cup_{i=1}^{N_p} \mathcal{N}_i$
L'opérateur permettant de passer d'un vecteur global $\mathcal{U} \in \mathbb{R}^{\#\mathcal{N}}$ à un vecteur local $\mathcal{U}_i \in \mathbb{R}^{\#\mathcal{N}_i}$ est appelé opérateur de restriction. Il s'agit d'une matrice rectangulaire de taille $\text{card}(\mathcal{N}_i) \times \text{card}(\mathcal{N})$ notée R_i dont les coefficients sont booléens. Inversement, la transposée R_i^T de cette matrice est l'opérateur permettant de passer d'un vecteur local à un vecteur global par ajout de zéros et est appelé opérateur de prolongement. Enfin, on définit sur chaque sous-domaine une matrice diagonale D_i de taille $\text{card}(\mathcal{N}_i)$ de manière à ce que la famille $(D_i)_{1 \leq i \leq N_p}$ forme une partition de l'unité :

$$I = \sum_{i=1}^{N_p} R_i^T D_i R_i \quad \text{avec } I \text{ matrice identité de taille } \text{card}(\mathcal{N}).$$

Il existe plusieurs moyens de définir les matrices D_i, qui diffèrent par le traitement appliqué aux degrés de liberté dupliqués appartenant à k sous-domaines ($k > 1$). Pour ces degrés de liberté, on peut par exemple

choisir la valeur $\frac{1}{k}$ pour le coefficient associé dans chacune des matrice D_i partageant ce degré de liberté, ou bien imposer arbitraitement 1 dans l'une de ces matrices et 0 dans toutes les autres.

Ces différents éléments permettent de construire sur chaque sous-domaine une matrice appelée matrice de Dirichlet locale A_i^D à partir de la matrice globale A selon

$$A_i^D = R_i A R_i^T. \tag{6.10}$$

Le préconditionneur à un niveau de la méthode de Schwarz additive restrictive (RAS, [CS99]) s'obtient grâce à l'équation (6.11) qui fait intervenir l'inverse des matrices de Dirichlet locales :

$$M_1^{-1} = \sum_{i=1}^{N_p} R_i^T D_i (A_i^D)^{-1} R_i. \tag{6.11}$$

En pratique, les inverses des matrices locales ne sont pas effectivement calculés ; ce sont des systèmes $A_i^D y = b$ qui sont résolus. Si un nombre suffisant de sous-domaines est utilisé, ces systèmes sont de taille raisonnable et peuvent être résolus de manière directe.

Méthode à deux niveaux

Une méthode à deux niveaux se construit à partir d'une méthode à un niveau et fait intervenir deux ingrédients supplémentaires principaux : un problème grossier, construit de manière à englober tous les sous-domaines pour améliorer la robustesse tout en étant peu coûteux à résoudre, et une formule de correction combinant ce problème grossier au préconditionneur à un niveau. Concernant le choix du problème grossier, on s'intéresse à une méthode algébrique permettant de construire l'espace grossier indépendamment de la décomposition en sous-domaines et de la distribution des coefficients physiques du problème, particulièrement adaptée aux problèmes hétérogènes : la méthode GenEO (***Gen**eralized **E**igenvalue in the **O**verlap*, [Spi+14]), qui construit son espace grossier à partir de la résolution de problèmes aux valeurs propres. En plus des matrices introduites précédemment, la méthode fait intervenir sur chaque sous-domaine une seconde matrice locale appelée matrice de Neumann et notée A_i^N. Il s'agit de la matrice qui aurait été obtenue si le problème physique était discrétisé uniquement sur le sous-domaine en question, en utilisant des conditions aux limites de traction nulles sur les bords du sous-domaine (en dehors des bords correspondant à une vraie frontière physique, qui conservent eux leurs conditions aux limites originales), d'où le nom de cette matrice. La méthode nécessite la résolution, dans chaque sous-domaine, du problème aux valeurs propres généralisé

$$A_i^N v = \lambda (D_i A_i^D D_i) v. \tag{6.12}$$

Si les valeurs propres de ce problème sont ordonnées dans l'ordre croissant, l'espace grossier local au sous-domaine i est l'espace Z_i engendré par les vecteurs propres associés aux n_i premières valeurs propres, cet entier pouvant être ajusté. L'espace grossier global est engendré par les colonnes de la matrice rectangulaire Z, de taille $\text{card}(\mathcal{N}) \times N_c$ avec $N_c = \sum_i n_i$, qui s'obtient à partir des espaces Z_i selon

$$Z = \left[R_1^T D_1 Z_1,\ R_2^T D_2 Z_2,\quad \cdots \quad , R_{N_p}^T D_{N_p} Z_{N_p} \right]$$

et permet de définir la matrice grossière $A_c = Z^T A Z$ de taille $N_c \times N_c$. Ainsi, lorsque les n_i augmentent, la correction apportée par l'espace grossier est plus importante, mais la taille du problème associé est plus grande. Une fois cet espace grossier construit et la matrice A_c obtenue, il reste à appliquer une formule de correction pour obtenir le préconditionneur de la méthode à deux niveaux. La correction employée dans la suite est la méthode *Balancing Neumann-Neumann* [Man93], qui calcule le préconditionneur selon

$$M_2^{-1} = (I - QA) M_1^{-1} (I - QA) + Q \quad \text{avec } Q = Z A_c^{-1} Z^T. \tag{6.13}$$

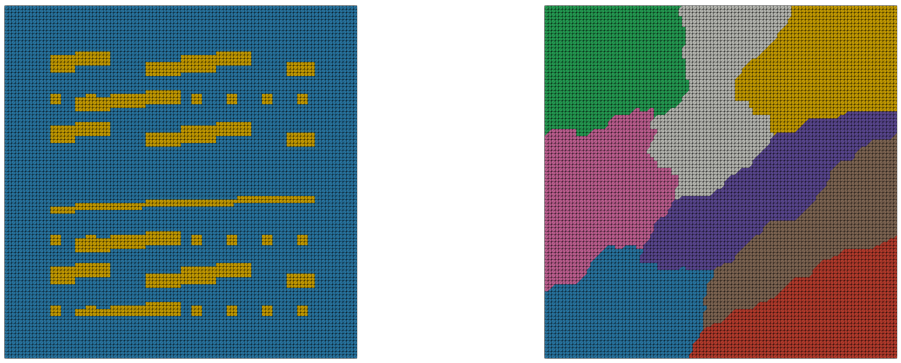


FIG. 6.5 : **Discrétisation en triangles du domaine hétérogène bidimensionnel** et exemple de partitionnement en sous-domaines, généré par Metis.

	homogène					hétérogène				
	2p	4p	8p	12p	16p	2p	4p	8p	12p	16p
1 niveau	84	102	132	148	161	139	241	345	430	482
2 niveaux	35	38	43	39	38	52	79	65	68	68

TAB. 6.5 : Comparaison du nombre d'itérations du solveur Gmres sur le problème mécanique en fonction du nombre de sous-domaines. Les méthodes de décomposition de domaine à deux niveaux semblent bien plus robustes que les préconditionneurs précédemment évalués sur ce problème.

Application au cas de mécanique hétérogène

Pour évaluer ces méthodes de décomposition de domaine, le logiciel FreeFem++ est à nouveau utilisé. En effet, celui-ci embarque un module de décomposition de domaine[3] qui permet d'utiliser simplement ces méthodes, la construction des matrices et du préconditionneur étant gérée par l'outil. Le cas précédemment utilisé pour l'évaluation des solveurs appliqués au problème de mécanique demande quelques adaptations pour être utilisé avec FreeFem++ : les 400×400 mailles carrées de la grille hétérogène sont chacune divisées en deux triangles de manière à obtenir un maillage triangulaire structuré tel que représenté sur la figure 6.5 (pour plus de lisibilité, la figure ne représente qu'un maillage de $100 \times 100 \times 2$ triangles). La discrétisation du problème se fait par éléments finis P1, qui sont ici équivalents à la méthode des éléments virtuels puisque la grille est triangulaire (*cf* section 3.3). Les valeurs des coefficients physiques sont les mêmes que lors des simulations précédentes. En revanche, la méthode itérative appliquée pour résoudre le système est l'algorithme Gmres au lieu de l'algorithme du BiCGStab. Le critère de convergence de 10^{-6} est néanmoins conservé, tout comme le fait de préconditionner le système à droite pour comparer le nombre d'itérations. Le nombre d'itérations nécessaire à la convergence de la méthode Gmres, avec application du préconditionneur à un (6.11) ou à deux (6.13) niveaux est rassemblé dans le tableau 6.5. Dans le second cas, le nombre de vecteurs propres utilisés pour la construction de l'espace grossier est de 10 par sous-domaine.

Ce cas illustre bien le manque de robustesse de la méthode à un niveau : dans le cas hétérogène, le nombre d'itérations augmente fortement avec le nombre de sous-domaines. Cette augmentation se re-

3. https ://doc.freefem.org/documentation/ffddm/

trouve d'ailleurs dans le cas homogène, même si elle est pour celui-ci d'intensité plus tenue. Dans les deux cas, l'amélioration apportée par le problème grossier est significative : le nombre d'itérations du solveur est extrêmement stable dans le cas homogène, et relativement stable dans le cas hétérogène (seules les simulations sur 2 ou 4 sous-domaines donnent un résultat légèrement plus éloigné de la moyenne). Cet apport ne coûte pas plus de temps de calcul : la résolution est plus rapide avec la méthode à deux niveaux qu'avec la méthode à un niveau. Dans le cas hétérogène à 8 sous-domaines, la durée totale de résolution est de 67 secondes pour la méthode à un niveau contre 13 secondes pour la méthode à deux niveaux. Pour celle-ci, la résolution du problème grossier est négligeable (moins de 0.02 s), et le temps passé à résoudre les problèmes aux valeurs propres (5 s) est largement compensé par celui économisé dans le solveur Gmres (5 secondes, contre 64 pour la méthode à un niveau). Enfin, dans l'absolu, le nombre d'itérations de la méthode à deux niveaux est non seulement inférieur à celui de la méthode à un niveau, mais il est également largement inférieur à celui qui avait été obtenu avec le préconditionnement AMG (voir tableau 6.4), même si la différence entre les solveurs et les maillages d'un cas sur l'autre rend cette comparaison délicate.

6.5 Utilisation des sous-solveurs spécialisés parallèles dans le cadre de la stratégie fixed-stress

La section précédente, en considérant des problèmes découplés, a permis de qualifier deux sous-solveurs prometteurs pour la résolution des sous-problèmes. Il reste à mettre effectivement en place ces solveurs dans le cadre de la stratégie séquentielle, et à vérifier leur comportement sur les sous-problèmes issus de cas couplés.

6.5.1 Utilisation de la librairie hpddm pour la décomposition de domaine

Cette partie, dont le sujet est à cheval entre les sections 6.4 et 6.5, explique comment utiliser effectivement les sous-solveurs dans la plateforme Arcane dans le cadre de la stratégie séquentielle. Puisque la méthode qualifiée pour le sous-problème d'écoulement, le préconditionnement AMG, est déjà accessible via PETSc, on se concentre en réalité sur la mise en place des méthodes de décomposition de domaine pour l'élasticité.

Pour mettre en place ces méthodes, on passe par la librairie hpddm[4] [JHNP13] qui en propose une implémentation parallèle performante. Si à terme l'objectif est de connecter cette librairie au simulateur, on ne dispose aujourd'hui que d'une installation indépendante, accessible via une interface python. Néanmoins, l'usage de l'interface est très proche de celui qui sera fait de l'API de la librairie lorsqu'elle sera connectée : dans les deux cas, la majorité des développements est à effectuer en amont pour préparer les objets attendus par la librairie (matrices de Dirichlet, de Neumann, partition de l'unité, etc.). Le processus actuel est donc le suivant : dans un premier temps, une simulation sur N_p cœurs est lancée dans le code de calcul, qui se charge du partitionnement et qui, durant l'assemblage du système linéaire, exporte un certain nombre d'objets dans des fichiers. Dans un second temps, un script python est lancé en parallèle sur N_p cœurs. Celui-ci lit les informations préalablement stockées, et appelle les fonctions contenues dans hpddm pour résoudre le système.

4. https ://github.com/hpddm/hpddm

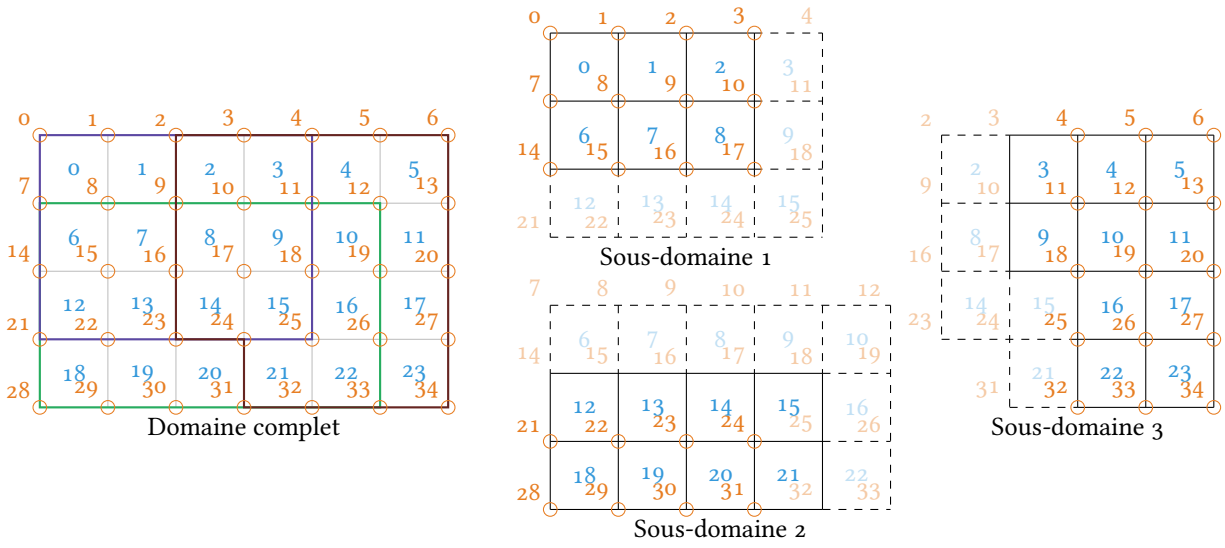


FIG. 6.6 : **Schématisation du partitionnement en sous-domaines.** Les cellules et les nœuds sont numérotés de manière globale. Dans les sous-domaines, les éléments own sont représentés par des cercles pour les nœuds et par des traits pleins pour les cellules. Les autres éléments affichés sont connus mais ghost.

Décomposition en sous-domaines

La décomposition en sous-domaines est déléguée à Metis et aux routines d'Arcane. En raison du recouvrement, certains nœuds ou cellules se retrouvent dans plusieurs sous-domaines. Pour différencier ces éléments dupliqués, il existe une notion d'appartenance : chaque sous-domaine dispose de sa liste d'éléments propres (own) et d'éléments connus mais propres à un sous-domaine voisin (ghost). Cette situation est schématisée sur la figure 6.6. Dans cet exemple, le sous-domaine 3 possède 15 nœuds propres, et "voit" 6 nœuds propres au sous-domaine 1 (d'indices globaux 2, 3, 9, 10, 16 et 17) ainsi que 3 nœuds propres au sous-domaine 2 (d'indices 23, 24 et 31). En plus de cette numérotation globale, il existe une numérotation locale à chaque sous-domaine. L'objet de type dictionnaire permettant de passer de la numérotation locale à la numérotation globale est exploité pour construire un ingrédient de base requis par hpddm : la liste des nœuds connus chez chaque voisin, dans la numérotation locale du voisin. Pour ce faire, le processus P_i réceptionne le dictionnaire de chaque autre processus P_j envoyé à l'aide de mpi_bcast. Si l'intersection entre les clés de son dictionnaire et celles du dictionnaire reçu de P_j est non nulle, alors les deux sous-domaines sont voisins. Dans ce cas, l'ensemble des valeurs du dictionnaire reçu pour les clés communes est la liste de connectivité demandée par hpddm. La notion de own et ghost permet également de construire les matrices de partition de l'unité D_i. Dans chaque sous-domaine, l'entrée de D_i correspondant à un nœud donné vaut 1 si le nœud en question est own, et 0 sinon.

Remarque. En pratique, on ne travaille pas sur des nœuds mais sur des degrés de liberté : dans les dictionnaires, une clé (nœud numéroté globalement) est associé à un tableau de degrés de liberté, qui compte au plus 3 éléments, mais qui peut en avoir moins lorsqu'une ou plusieurs composantes du déplacement sont imposées par des conditions aux limites de Dirichlet sur ce nœud. De même, la matrice D_i comporte jusqu'à trois entrées par nœud.

Matrices locales aux sous-domaines

Les sous-matrices les plus simples à construire sont les matrices de Neumann A_i^N. Il s'agit en effet des matrices qui auraient été obtenues si le problème n'était défini que sur le sous-domaine, avec des conditions aux limites de Neumann sur les frontières créées par la partition. Puisque ces conditions aux limites ne modifient pas la matrice, il suffit en réalité d'utiliser la procédure d'assemblage standard (voir algorithme 1) en utilisant cette fois la numérotation locale pour construire ces matrices.

La construction des sous-matrices de Dirichlet A_i^D est moins directe. Celles-ci sont définies comme la restriction de la matrice globale A au sous-domaine : de manière grossière, il s'agit de supprimer toutes les lignes et colonnes de la matrice globale qui ne sont pas associées à un degré de liberté connu du sous-domaine. Une solution consiste d'ailleurs à partir de la matrice globale, que l'on sait construire en parallèle, et d'en extraire les matrices A_i^D au prix de quelques manipulations des différentes indexations. Si cette solution est applicable via un script externe après écriture de la matrice dans un fichier, elle est en revanche délicate à mettre en place au sein du simulateur : on ne peut en effet pas accéder simplement aux coefficients de la matrice globale une fois celle-ci assemblée. L'approche décrite ici permet de construire à la volée les matrices A_i^D en même temps que les matrices A_i^N. Dans un premier temps, les matrices locales sont utilisées pour assembler les sous-matrices de Dirichlet, à ceci près que l'assemblage n'est réalisé que pour les nœuds own. Les entrées des matrices associées aux nœuds ghost vont en effet récupérer les valeurs issues des autres sous-domaines. Pour ce faire, dans un second temps, chaque processus P_i reçoit des autres processus P_j une liste de nœuds own dans le domaine i et ghost dans le domaine j. Après quoi, le processus P_i parcourt sa matrice, vérifie quelles lignes correspondent à un nœud présent dans la liste demandée, et envoie en retour l'intégralité de ces lignes au processus P_j, dans une numérotation globale. Il reste au processus P_j à vérifier, pour chaque ligne reçue, quelles colonnes correspondent également à un nœud présent dans son domaine (own ou ghost), puis à ajouter dans sa propre matrice A_i^D les coefficients retenus après avoir converti leur numérotation dans son indexation locale. Ces opérations sont effectuées en utilisant des conteneurs de données optimisés pour la recherche d'éléments, et les échanges se font par communications non bloquantes (mpi_sendRecv). Sur l'exemple de la figure 6.6, le processus 3 recevrait la liste [4, 11, 18, 25] du processus 1. Il renverrait en retour au processus 1 (parmi d'autres lignes) les lignes de coefficients associées au nœud 4, soit, en supposant pour simplifier qu'il n'y ait qu'un seul degré de liberté par nœud, les coefficients $[A_3^D(4,3), A_3^D(4,4), A_3^D(4,5), A_3^D(4,10), A_3^D(4,11), A_3^D(4,12)]$, le reste de cette ligne étant nul. Pour finir, le processus 1 aurait à filtrer cette liste pour ignorer les coefficients $A_3^D(4,5)$ et $A_3^D(4,12)$, puisque les nœuds 5 et 12 lui sont totalement inconnus, et à ajouter les autres à sa matrice A_1^D.

Enfin, il reste à construire sur chaque sous-domaine un second membre local b_i et une solution initiale locale X_i^0. Pour ces quantités, l'extraction depuis les vecteurs globaux b et x_0 pendant la simulation étant possible, c'est cette approche qui est utilisée. Il suffit donc pour chaque processus d'aller chercher dans les vecteurs globaux les coefficients correspondant aux nœuds présents dans son sous-domaine.

Passage dans hpddm

Une fois ces objets construits, l'utilisation de l'interface de hpddm est relativement simple et est résumée dans la figure 6.7. L'option *schwarz_coarse_correction* déclenche la prise en compte du second niveau dans la méthode de décomposition de domaine, le paramètre *nu* servant à régler le nombre de valeurs propres à conserver dans chaque sous-domaine. Après résolution, chaque processus connaît la solution locale à son sous-domaine stockée dans le vecteur *sol*. Pour reconstruire la solution globale, la contribution de chaque sous-domaine aux nœuds partagés est pondérée à l'aide de la partition de l'unité. Ainsi, avec les choix effectués précédemment, c'est le sous-domaine propriétaire d'un nœud qui fixe la solution

```
import hpddm
A = hpddm.schwarzCreate(AiD, neighbors, connectivity)
hpddm.schwarzInitialize(A, Di)
if hpddm.optionSet(opt, b'schwarz_coarse_correction'):
    hpddm.schwarzSolveGEVP(A, AiN)
    hpddm.initializeCoarseOperator(hpddm.schwarzPreconditioner(A), nu)
    hpddm.schwarzBuildCoarseOperator(A, mpi_communicator)
hpddm.schwarzCallNumfact(A)

hpddm.solve(A, rhs, sol, mpi_communicator)
```

FIG. 6.7 : **Utilisation de l'interface de la libraire hpddm** pour la résolution parallèle d'un problème par décomposition de domaine. Pour plus de lisibilité, les procédures ou arguments non essentiels ne figurent pas sur cet extrait.

sur celui-ci. Le rapatriement des solutions locales vers un vecteur solution globale s'effectue à l'aide de la fonction mpi_gather.

6.5.2 Comportement sur le cas synthétique hétérogène

Les sous-solveurs qualifiés pour la résolution des sous-problèmes étant maintenant utilisables, il reste à régler leurs paramètres, typiquement leur tolérance ϵ^{int}, et à vérifier leur comportement dans le cadre de la stratégie séquentielle fixed-stress. On considère donc à nouveau le problème hétérogène synthétique de la section 6.2.3 pour lequel, jusqu'à maintenant, les sous-problèmes étaient résolus par une méthode directe.

Pour régler les paramètres des sous-solveurs, l'algorithme utilisé est la version point fixe avec itération sur la variable σ_v. Pour les deux sous-problèmes, le sous-solveur est le BiCGStab préconditionné par la méthode ILU(0). Même si ces choix sont loin d'être optimaux du point de vue des performances et de la robustesse (voir sections précédentes), on n'accorde pas d'importance à cet aspect dans cette étude préliminaire. Pour la même raison, les expériences sont menées en séquentiel. La tolérance de la méthode fixed-stress ϵ^{ext} étant fixée à 10^{-6}, on étudie l'impact de la tolérance des sous-solveurs ϵ^{int} sur le nombre d'itérations externes et sur la qualité de la solution. Pour mesurer cette dernière, les déplacements et la pression après convergence sont comparés aux valeurs calculées par la résolution monolithique. Enfin, on mesure le nombre moyen d'itérations des sous-solveurs pour une itération externe. Les résultats sont rassemblés dans le tableau 6.6. Si la tolérance interne est trop permissive par rapport à la tolérance externe, alors la méthode fixed-stress converge très rapidement vers une mauvaise solution. Lorsque la tolérance interne diminue, la convergence de la méthode fixed-stress nécessite plus d'itérations mais la précision des solutions s'améliore. En toute logique, ces deux quantités tendent vers un palier correspondant aux résultats obtenus avec un sous-solveur direct pour les sous-problèmes. Diminuer la tolérance interne améliore la solution, mais augmente le coût global de la résolution : non seulement le nombre d'itérations externes est plus élevé, mais le nombre d'itérations internes augmente également. Au vu de ces résultats, la tolérance interne doit être au moins de deux ordres de grandeur inférieure à la tolérance externe. En revanche, à partir de 4 ordres de grandeurs d'écart, l'amélioration de la solution semble insuffisante vis-à-vis de l'augmentation du nombre d'itérations des sous-solveurs. Dans la suite, on maintiendra donc trois ordres de

ϵ^{int}	Itérations externes	Écarts sur p	Écarts sur $\mathbf{u}$	Itérations internes P	Itérations internes U
10^{-4}	4	$1.4 \cdot 10^{-1}$	$1.8 \cdot 10^{-1}$	15	106
10^{-6}	6	$4.8 \cdot 10^{-3}$	$7.8 \cdot 10^{-3}$	110	283
10^{-8}	9	$5.5 \cdot 10^{-5}$	$8.2 \cdot 10^{-5}$	197	324
10^{-10}	13	$9.8 \cdot 10^{-7}$	$1.8 \cdot 10^{-6}$	373	458
10^{-12}	14	$2.4 \cdot 10^{-7}$	$5.4 \cdot 10^{-7}$	714	545
Référence	14	$2.4 \cdot 10^{-7}$	$5.3 \cdot 10^{-7}$		

Tab. 6.6 : Nombre d'itérations de la méthode fixed-stress avec $\epsilon^{\text{ext}} = 10^{-6}$ selon la tolérance des sous-solveurs ϵ^{int}. Le tableau indique également les écarts avec les solutions du schéma monolithique, et le nombre d'itérations internes des sous-solveurs pour chaque itération externe (valeur moyenne).

grandeur d'écart entre la tolérance externe et les tolérances internes.

Par ailleurs, à chaque itération externe, les sous-solveurs itératifs internes nécessitent un vecteur pour initier leur algorithme. Fournir comme vecteur initial la solution issue de l'itération externe précédente $\mathbf{u}^{n,l-1}$ ou $p^{n,l-1}$ permet aux sous-solveurs internes de converger de plus en plus rapidement : la différence entre deux itérations externes successives diminue de plus en plus lorsque l'algorithme fixed-stress est sur le point de converger. Par exemple, dans le cas précédent avec $\epsilon^{\text{int}} = 10^{-8}$, le sous-solveur de l'écoulement nécessite 485 itérations internes pour converger pendant la première itération externe, mais converge en 2 itérations pendant la dernière itération externe. Pour le sous-solveur des déplacements, on passe de 503 à 2 itérations internes.

L'étude préliminaire étant à présent terminée, on sait comment paramétrer les sous-solveurs : ceux-ci doivent démarrer leurs itérations sur la solution de l'itération externe précédente et avoir un critère d'arrêt de trois ordres de grandeur inférieur à la tolérance externe. Il subsiste une limitation due à l'usage externalisé de la librairie hpddm : il n'est actuellement pas possible de faire remonter la solution de hpddm vers le simulateur et donc de l'utiliser pour valider l'itération externe. Pour évaluer tout de même les sous-solveurs spécialisés, la stratégie retenue est la suivante : en même temps que ceux-ci résolvent les sous-problèmes, deux sous-solveurs directs sont utilisés pour effectuer la même résolution. Ce sont les solutions issues des sous-solveurs directs qui permettent l'avancée des itérations de la méthode séquentielle, les sous-solveurs spécialisés n'étant utilisés que pour étudier leur comportement. Ainsi, on utilise pour la pression la méthode du BiCGStab préconditionnée par AMG (via une interface interne à Arcane), et pour les déplacements la méthode Gmres préconditionnée par la méthode de décomposition de domaine à deux niveaux (via l'interface externalisée). Dans les deux cas, la tolérance est fixée à $\epsilon^{\text{int}} = 10^{-9}$. Selon le nombre de cœurs utilisés pour la résolution, le nombre d'itérations des sous-solveurs pour chacune des 14 itérations nécessaires à la convergence de l'algorithme fixed-stress est représenté sur la figure 6.8. Même dans le cadre du problème couplé, le sous-solveur pour la pression reste extrêmement robuste : ce n'est que pour deux sous-systèmes que le nombre d'itérations varie selon le nombre de sous-domaines, l'écart n'étant alors que de une itération. Pour le sous-problème en déplacement, la méthode de décomposition de domaine à deux niveaux reste relativement robuste puisque le solveur converge dans tous les cas, avec une variation de son nombre d'itérations qui reste faible par rapport à celles qui avaient pu être relevées dans les cas qualifiés de problématique (tableaux 6.2 et 6.4).

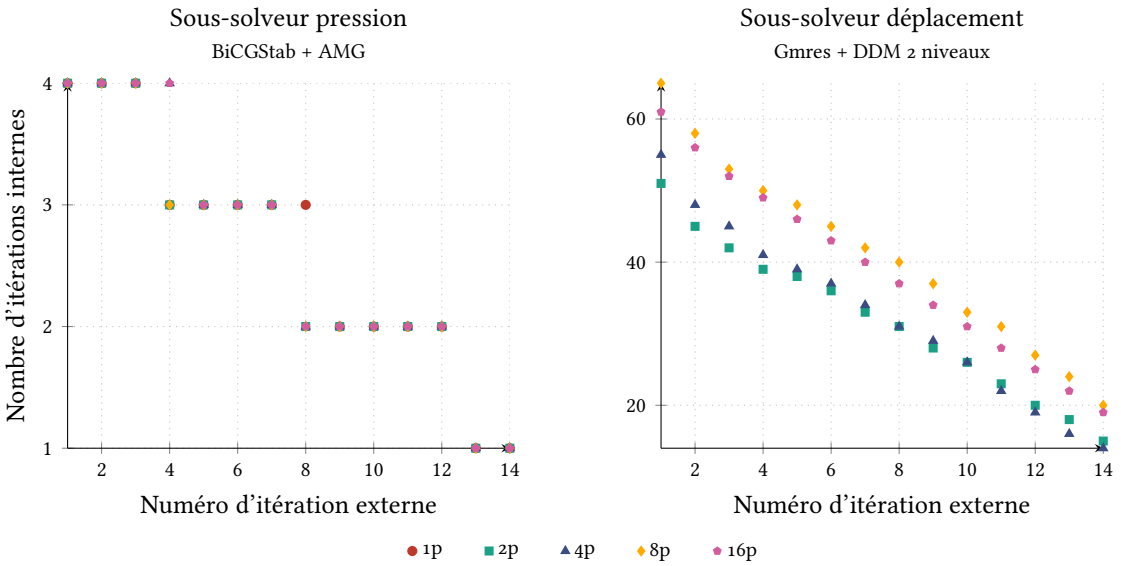


FIG. 6.8 : **Nombre d'itérations des sous-solveurs pour chaque itération externe sur le cas hétérogène 2D, selon le nombre de cœurs employés.** Même dans le cadre du couplage, les sous-solveurs restent robustes.

6.5.3 Application au problème tridimensionnel réaliste

Les différentes méthodes détaillées dans ce chapitre ainsi que les observations effectuées sont à présent mises à contribution pour tenter de résoudre le problème réel tridimensionnel, qui avait été utilisé dans la section 6.1.3 pour choisir la stratégie à adopter.

Le système est donc résolu par stratégie séquentielle, accélérée par la méthode de Krylov Gmres, en itérant sur la variable σ_v. La tolérance externe est fixée à 10^{-6}. Comme dans le cas synthétique hétérogène de la section précédente, les sous-solveurs utilisés sont l'algorithme du BiCGStab préconditionné par AMG pour la pression, et la méthode Gmres préconditionnée par la décomposition de domaine à deux niveaux pour les déplacements. Pour ces deux méthodes, la tolérance interne est fixée à 10^{-9} et le vecteur initial est pris nul. À nouveau, pour s'affranchir de l'intégration partielle de la librairie hpddm, des sous-solveurs directs permettent d'avancer la résolution. Pour la même raison ainsi que pour diminuer le temps de calcul, seul le premier pas de temps du système est résolu. Sur ce premier pas de temps, l'accélération de la méthode fixed-stress permet à celle-ci de converger en 11 itérations. À titre indicatif, 24 itérations sont nécessaires avec l'algorithme de point fixe. Comme dans le cas précédent, le nombre d'itérations des sous-solveurs pour chacune de ces 11 itérations est représenté sur la figure 6.9, chaque courbe correspondant à un certain nombre de sous-domaines. Comme pour le cas synthétique présenté dans la section précédente, le sous-solveur en pression (préconditionné par AMG) est extrêmement stable : pour une itération externe (et donc pour un système) donnée, son nombre d'itérations est le même quel que soit le nombre de sous-domaines utilisés. Ce nombre varie également très peu au fil des itérations externes puisqu'il reste fixé à 4 (excepté pour la première résolution), ce qui est faible : la convergence de la méthode est très rapide. Pour la résolution du sous-problème élastique par décomposition de domaine, on observe cette même stabilité au fil des itérations externes pour un nombre de sous-domaines donné. L'écart selon le nombre de cœurs utilisés est sensiblement identique à celui qui avait été obtenu sur le cas bidimensionnel précédent, de même

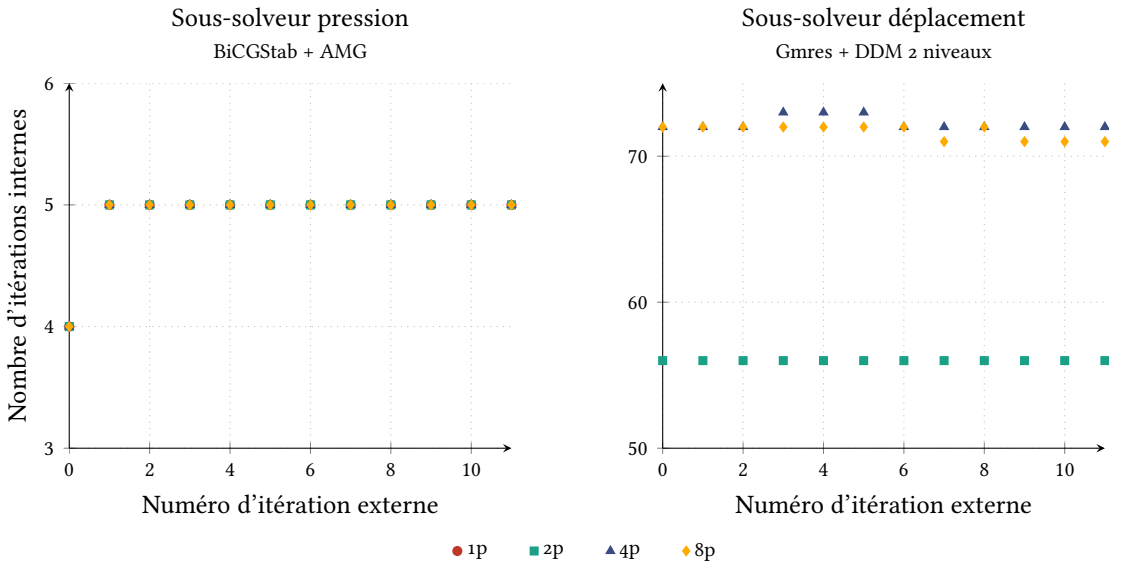


FIG. 6.9 : **Nombre d'itérations des sous-solveurs pour chaque itération externe sur le cas réaliste 3D, selon le nombre de cœurs employés.** Les conclusions sont semblables à celles du cas synthétique 2D : le sous-solveur en pression est extrêmement robuste, le sous-solveur en déplacement relativement robuste.

que la valeur du nombre d'itérations : le passage au cas réel n'entraîne pas de changements significatifs quant à la robustesse de la méthode.

Pour revenir au pari effectué en début de chapitre, les résultats obtenus sur ce cas sont plutôt encourageants. Du point du vue du nombre de systèmes à résoudre, l'accélération par méthode de Krylov de la stratégie fixed-stress a permis de diminuer le nombre d'itérations externes. Du point de vue de la robustesse, le sous-problème des déplacements, sur lequel a été concentrée la majorité des efforts, semble pouvoir être résolu de manière robuste par les méthodes de décomposition de domaine à deux niveaux. Pour le sous-problème d'écoulement, le solveur préconditionné par AMG demeure très robuste et devrait donc être suffisant. Dans tous les cas, il reste à poursuivre les efforts d'intégration de la librairie hpddm au simulateur pour répondre à l'aspect performances de la question : soit de manière directe, soit en attendant son implémentation dans une bibliothèque telle que PETSc.