CSMA 2024
16ème Colloque National en Calcul des Structures
13-17 Mai 2024, Presqu'île de Giens (Var)

Préconditionneurs exploitant les blocs de rang faible pour les méthodes de décomposition de domaine primales

T. Gauthier[1,2], C. Bovet[1], P. Gosselet[2]

[1] *Université Paris-Saclay, ONERA, Matériaux et Structures, 92322 Châtillon, France, {theodore.gauthier, christophe.bovet}@onera.fr*
[2] *LaMcube, Univ. Lille / CNRS / Centrale Lille, F-59000, Lille, France, pierre.gosselet@univ-lille.fr*

Résumé — Les méthodes de factorisation approchées de type Block Low Rank sont utilisées pour construire des préconditionneurs à faible coût pour la méthode Balancing Domain Decomposition (BDD). L'influence du seuil de compression sur le taux de convergence et sur le calcul des noyaux locaux, est étudiée. Ces derniers équipent la méthode BDD d'un problème grossier indispensable à l'extensibilité. Le cas échéant, des méthodes de détections alternatives ou l'exploitation de méthodes basées sur le multipréconditionnement sont proposées.
Mots clés — BDD, AMPBDD, factorisation par blocs de rang faible (BLR).

1 Introduction

La méthode des éléments finis est aujourd'hui la méthode la plus répandue et versatile pour simuler la grande variété des problèmes rencontrés en mécanique des matériaux et des structures. Lorsque les modèles numériques sont de grandes tailles, le recours aux méthodes de calcul haute performance pour la construction des opérateurs et pour la résolution de systèmes linéaires devient indispensable. Les approches par décomposition de domaine, combinant des solveurs directs locaux et un solveur itératif d'interface, ont prouvé leur efficacité pour résoudre ces grands systèmes linéaires.

Les méthodes duales (FETI [1], AMPFETI [2, 3, 4]) disposent nativement de différents préconditionneurs à qualité et coût numérique variables, permettant de s'adapter au conditionnement du système à résoudre. Elles sont par contre mal adaptées à la simulation de problème de rupture (endommagement, propagation de fissures, etc.) du fait de leur plus grande sensibilité à la détection des noyaux des opérateurs locaux de rigidité. Sur ce point, les méthodes primales (BDD [5], BDDC [6]) font preuve d'une meilleure robustesse. Elles ne disposent par contre que du préconditionneur complet, nécessitant la factorisation des compléments de Schur locaux. Le coût mémoire de ces factorisations peut être pénalisant pour exploiter pleinement les nouveaux supercalculateurs. À titre d'exemple, les nœuds Milan du nouveau supercalculateur Topaze du TGCC, disposent uniquement de 2Go RAM par cœur.

Depuis quelques années, les solveurs directs (MUMPS [1], Pastix [2]) proposent des méthodes de *compression* pour réduire l'empreinte mémoire des factorisées. Par exemple, MUMPS exploite les blocs de rang faible (BLR) pour réaliser des résolutions approchées et accélérer les étapes de factorisation et de substitution. Ces nouvelles fonctionnalités offrent la possibilité de construire des préconditionneurs à faible coût pour la méthode BDD. Un compromis devra être trouvé entre le taux de compression et la rapidité de convergence du solveur itératif.

Enfin, la détermination des noyaux locaux équipe la méthode BDD d'un problème grossier indispensable à l'extensibilité. L'impact de la compression sur cette détection, des méthodes de détections alternatives [7, 8], ou l'exploitation des méthodes du multipréconditionnement seront étudiés.

Le papier est organisé comme suit. Dans un premier temps, le principe des factorisations avec blocs de rang faible est rappelé. Une première étude numérique met en évidence les avantages de ces factorisations approchées. Ensuite, la méthode BDD est rappelée et les méthodes BLR sont exploitées pour construire des préconditionneurs à faible coût. Les différentes configurations sont alors évaluées sur des problèmes académiques. Enfin, tous les résultats sont obtenus grâce à la suite éléments finis Z-set [3].

1. https://mumps-solver.org
2. https://gitlab.inria.fr/solverstack/pastix
3. http://www.zset-software.com/

2 Factorisation exploitant les blocs de rang faible

2.1 Matrices de rang faible et format Block Low Rank

Soit une matrice $\boldsymbol{A} \in \mathcal{M}_n(\mathbb{R})$, si $\boldsymbol{A}$ est de rang $k \in \mathbb{N}$, il existe deux matrices $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{M}_{n,k}(\mathbb{R})$ telles que $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^\top$. Plus généralement, on définit k le rang numérique d'une matrice $\boldsymbol{A}$ à la précision $\varepsilon > 0$ par :

$$k_\varepsilon = \min\{k \in \mathbb{N} \mid \exists\, \widetilde{\boldsymbol{A}} \in \mathcal{M}_n(\mathbb{R}) \text{ telle que } \mathrm{rang}(\widetilde{\boldsymbol{A}}) = k \text{ et } ||\boldsymbol{A} - \widetilde{\boldsymbol{A}}||_2 \leq \varepsilon\}$$

En pratique, $\widetilde{\boldsymbol{A}}$ est obtenu directement sous la forme de produit extérieur $\boldsymbol{X}\boldsymbol{Y}^\top$ par méthode RRQR tronquée. $\boldsymbol{A}$ est alors dite *compressée*. La matrice $\boldsymbol{A}$ (ou $\widetilde{\boldsymbol{A}}$) est dite de rang faible ou *low rank* dès lors que $k_\varepsilon(m+n) \leq mn$. Dans ce cas, $\boldsymbol{X}$ et $\boldsymbol{Y}$ sont moins coûteuses que $\widetilde{\boldsymbol{A}}$ à stocker et les multiplications successives par $\boldsymbol{Y}^\top$ puis $\boldsymbol{X}$ nécessitent moins d'opérations que la multiplication par $\widetilde{\boldsymbol{A}}$.

Nous dirons qu'une matrice de $p \times p$ blocs $\boldsymbol{F} = (\boldsymbol{F}_{i,j})$ est au format Blocks Low Rank (BLR) quand elle est approchée par $\widetilde{\boldsymbol{F}}$ donnée par :

$$\widetilde{\boldsymbol{F}} = \begin{pmatrix} \boldsymbol{F}_{1,1} & \widetilde{\boldsymbol{F}}_{1,2} & \dots & \widetilde{\boldsymbol{F}}_{1,p} \\ \widetilde{\boldsymbol{F}}_{2,1} & \ddots & \dots & \vdots \\ \vdots & \dots & \ddots & \widetilde{\boldsymbol{F}}_{p-1,p} \\ \widetilde{\boldsymbol{F}}_{p,1} & \dots & \widetilde{\boldsymbol{F}}_{p,p-1} & \boldsymbol{F}_{p,p} \end{pmatrix}$$

où seuls les blocs extra-diagonaux sont éventuellement compressés au format low rank. Typiquement, le rang numérique des blocs extra-diagonaux décroît exponentiellement en fonction de la distance géométrique entre les éléments pour une matrice de rigidité issue d'un problème elliptique (voir [9] et [10]).

2.2 Méthodes multifronales et factorisation BLR pour la factorisation $LDL^\top$

Les méthodes de factorisation BLR consistent à imposer le format BLR aux matrices frontales $\boldsymbol{F}_{i,j}$ issues des méthodes de factorisation multifrontales de matrices creuses. Ces dernières consistent à analyser le graphe d'élimination d'une matrice creuse $\boldsymbol{K}$, de sorte à établir un plan d'élimination de ses pivots à l'aide d'un arbre - dit d'élimination - dont les nœuds correspondent aux pivots à éliminer. L'arbre est parcouru de bas en haut sur différentes branches indépendantes de façon parallèle.

À chaque nœud est associée une matrice frontale $\boldsymbol{F}$ dense correspondant à différents pivots à éliminer. Une fois un nœud atteint, une factorisation partielle de $\boldsymbol{F}$ permet de calculer la contribution des pivots éliminés à cette étape aux nœuds situés plus haut dans l'arbre, nécessaire pour construire leurs matrices frontales associées. Une fois la racine de l'arbre atteinte et traitée, la factorisation est terminée.

Les méthodes de factorisation BLR consistent à effectuer ces factorisations partielles avec des matrices frontales $\boldsymbol{F}$ écrites en format BLR et en compressant éventuellement les blocs de contributions obtenus. Toutes les opérations habituelles sont encore possibles sous ce format sans nécessiter obligatoirement de décompression des blocs extra-diagonaux de $\widetilde{\boldsymbol{F}}$ avec des gains en temps et des compromis sur la précision déjà étudiés (voir [10]). Ces méthodes ont d'ores et déjà été implémentées dans différentes bibliothèques de solveurs creux : MUMPS[1] [10] et PaStiX[2] [11]. Puisque les factorisations sont approchées, la qualité des solutions obtenues est, au besoin, améliorée par raffinement itératif.

La méthode BLR implémentée dans MUMPS 5.5.1 offre un choix sur différents paramètres. En premier lieu la précision $\varepsilon_{BLR} > 0$ à laquelle sont calculées les rangs numériques des blocs extra-diagonaux des matrices frontales, ci-après désigné par seuil de compression BLR. Deux variantes de la factorisation sont proposées par MUMPS, UFSC et UCFS, qui diffèrent en ce que la seconde peut permettre un gain de temps en effectuant davantage d'opérations sous format compressé au prix d'un résidu potentiellement dégradé.

2.3 Paramètres étudiées et tests numériques en séquentiel

Gain mémoire, précision en fonction du seuil, des variantes et du raffinement itératif Des résolutions en séquentiel ont d'abord été réalisées avec MUMPS afin d'avoir une idée du comportement de

la factorisation BLR pour des problèmes de traction en élasticité linéaire isotrope. Le problème est discrétisé par $\boldsymbol{Ku} = \boldsymbol{f}$ via la méthode des éléments finis. Les domaines étudiés sont des cubes en damier (cf. figure 2 à gauche) composés de deux matériaux distincts afin d'introduire et moduler l'hétérogénéité $h = E_r/E_b$. Le coefficient de Poisson est de 0.3. Les paramètres du problème étudié sont, le nombre de degrés de liberté allant de 50 000 à 400 000, le nombre de cases du damier, le facteur d'hétérogénéité de $h = 1$ (aucune hétérogénéité) à $h = 10^6$. Le maillage est composé d'hexaèdres linéaires ou quadratiques.

Les effets de la variante employée (UCFS ou UFSC), du choix d'activer ou non le raffinement itératif, et surtout du seuil de précision ε_{BLR} ont été observés sur le temps de la factorisation, son empreinte mémoire et le comportement du résidu relatif $||\boldsymbol{Ku} - \boldsymbol{f}||/||\boldsymbol{f}||$. Les tendances observées sont similaires pour les différents cas étudiés, nous présentons ici seulement le cas quadratique avec 400 000 *ddls* et 16^3 cases de damier. Les résultats sont résumés dans la figure 1 ci-dessous.

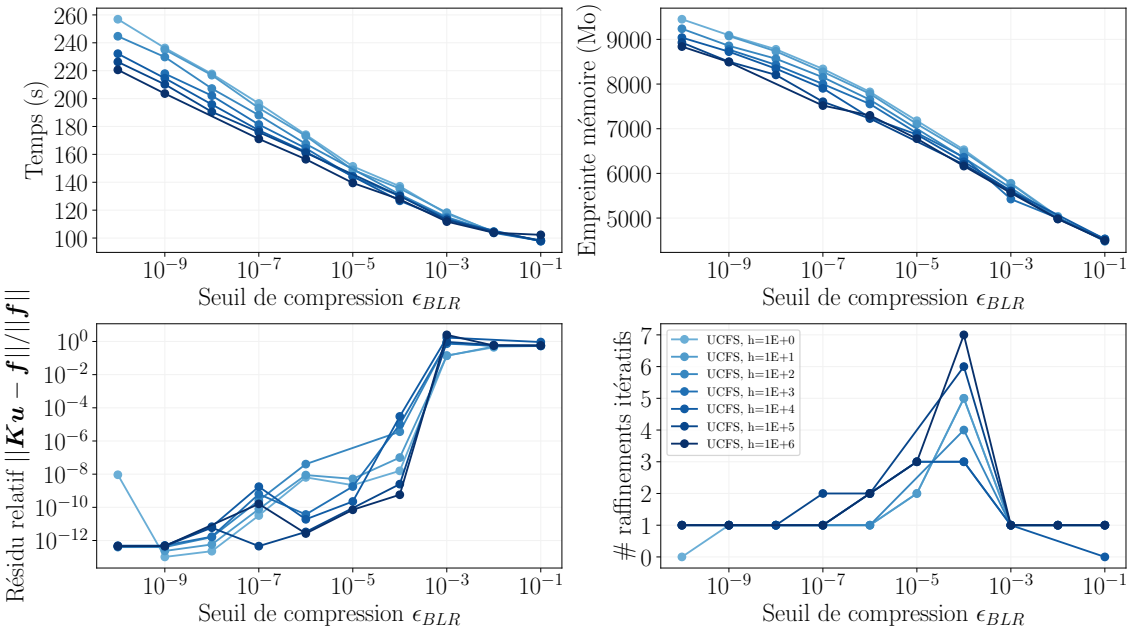

FIGURE 1 – Benchmark séquentiel : traction d'un cube en élasticité linéaire isotrope, éléments quadratiques, 400 000 degrés de liberté, 16^3 cases de damier pour différentes hétérogénéités h.

Les résultats obtenus sont conformes à l'intuition et à la littérature. L'augmentation du taux de compression réduit l'empreinte mémoire et le temps de calcul de la factorisée. En l'absence de raffinement itératif, la qualité du résidu relatif $||\boldsymbol{Ku} - \boldsymbol{f}||/||\boldsymbol{f}||$ devient rapidement inacceptable pour $\varepsilon_{BLR} > 10^{-6}$ ne permettant pas de profiter pleinement de ces gains. Le surcoût en temps et mémoire du raffinement itératif est négligeable pour les problèmes étudiés ici et permet de sélectionner des seuils $\varepsilon_{BLR} \geq 10^{-4}$. Au-delà de ce seuil, MUMPS était incapable d'améliorer la solution obtenue. Le comportement de la factorisation diffère peu en fonction l'hétérogénéité ou de la variante UCFS ou UFSC sélectionnée, du type d'éléments choisi, et de la taille des problèmes. Le seuil $\varepsilon_{BLR} = 10^{-4}$ marque un changement de régime vis-à-vis du raffinement itératif. Une compression plus faible requiert une seule itération de raffinement itératif pour converger. Le raffinement itératif est désactivé par MUMPS pour des compressions plus fortes, car le raffinement diverge. La factorisation BLR permet alors de façon consistante un gain en temps et en empreinte mémoire supérieur à 30% par rapport à une factorisation classique.

Factorisation approchée et calcul de noyaux Pour permettre l'extensibilité de la méthode BDD, il sera nécessaire de résoudre un problème grossier fondé sur le calcul des noyaux locaux. La compression BLR a naturellement un impact sur l'existence des noyaux et sur la bonne capacité de MUMPS à les détecter. Cette question sera détaillée lors des benchmarks parallèles.

3 Méthode de décomposition de domaine primale

3.1 Rappel sur la méthode BDD

On considère la résolution d'un problème d'élasticité linéaire discrétisé par la méthode des éléments finis en déplacement. Cela conduit à résoudre un système linéaire discret symétrique défini positif de la forme $\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}$.

On introduit une partition du domaine d'étude en N sous-domaines constitués par des ensembles connexes d'éléments. On note Γ l'interface entre les sous-domaines. En séparant les degrés de liberté internes (indice i) des degrés de bord (indice b), et en introduisant des opérateurs booléens d'assemblage sur l'interface $(\boldsymbol{A}^s)$, on obtient la formulation :

$$\begin{pmatrix} \boldsymbol{K}_{ii}^1 & & & \boldsymbol{K}_{ib}^1 \boldsymbol{A}^{1^T} \\ & \ddots & & \vdots \\ & & \boldsymbol{K}_{ii}^N & \boldsymbol{K}_{ib}^N \boldsymbol{A}^{N^T} \\ \boldsymbol{A}^1 \boldsymbol{K}_{bi}^1 & \dots & \boldsymbol{A}^N \boldsymbol{K}_{bi}^N & \sum_s \boldsymbol{A}^s \boldsymbol{K}_{bb}^s \boldsymbol{A}^{s\top} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_i^1 \\ \vdots \\ \boldsymbol{u}_i^N \\ \boldsymbol{u}_\Gamma \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_i^1 \\ \vdots \\ \boldsymbol{f}_i^N \\ \sum_s \boldsymbol{A}^s \boldsymbol{f}_b^s \end{pmatrix}. \tag{1}$$

Cette formulation est dite primale, car on a conservé pour inconnue principale le déplacement de l'interface $\boldsymbol{u}_\Gamma$.

Il est possible de formellement éliminer en parallèle les degrés de liberté internes, et d'obtenir la formulation dite condensée :

$$\sum_s \boldsymbol{A}^s \underbrace{\left(\boldsymbol{K}_{bb}^s - \boldsymbol{K}_{bi}^s \boldsymbol{K}_{ii}^{s^{-1}} \boldsymbol{K}_{ib}^s\right)}_{\boldsymbol{S}^s} \boldsymbol{A}_\Gamma^{s\top} = \sum_s \boldsymbol{A}^s \underbrace{\left(\boldsymbol{f}_b^s - \boldsymbol{K}_{bi}^s \boldsymbol{K}_{ii}^{s^{-1}} \boldsymbol{f}_i^s\right)}_{\boldsymbol{b}^s}. \tag{2}$$

On reconnaît le complément de Schur $\boldsymbol{S}^s$ et le second membre condensé $\boldsymbol{b}^s$. Le problème global se réécrit donc comme un assemblage de contributions locales. Le problème à l'interface hérite des propriétés du problème initial, en particulier l'opérateur $\boldsymbol{S}_{BDD} = \sum_s (\boldsymbol{A}^s \boldsymbol{S}^s \boldsymbol{A}^{s\top})$ est symétrique défini positif.

Cette formulation se prête naturellement à une résolution itérative par un solveur de Krylov, ici l'algorithme du gradient conjugué. Un préconditionneur de choix est disponbile sous la forme d'un assemblage pondéré des contributions inverses :

$$\boldsymbol{M}_{BDD}^{-1} = \sum_{s=1}^{N_d} \tilde{\boldsymbol{A}}^s \boldsymbol{S}^{s\dagger} \tilde{\boldsymbol{A}}^{s\top}. \tag{3}$$

Dans cette expression, les opérateurs d'assemblage pondérés $(\tilde{\boldsymbol{A}}^s)$ satisfont à l'équation suivante :

$$\sum_s \boldsymbol{A}^s \tilde{\boldsymbol{A}}^{s\top} = \boldsymbol{I}, \tag{4}$$

alors que la notation $\boldsymbol{S}^{s\dagger}$ correspond à une pseudo-inverse de $\boldsymbol{S}^s$. En effet, pour un sous-domaine flottant (non connecté à une condition de Dirichlet), les problèmes locaux sont mal posés du fait de la présence de modes rigides, et ne sont résolubles qu'orthogonalement à ceux-ci.

De manière à s'assurer de toujours travailler dans des champs qui n'excitent pas les modes rigides, un projecteur $\boldsymbol{P}$ est introduit. On note $\boldsymbol{R}^s$ une base du noyau de $\boldsymbol{S}^s$, qui correspond à la trace des modes rigides sur la frontière du sous-domaine. On pose $\boldsymbol{G} = \left[\dots \tilde{\boldsymbol{A}}^s \boldsymbol{R}^s \dots\right]$ la concaténation des modes rigides sur l'interface. De fait, $\boldsymbol{M}_{BDD}^{-1}$ préconditionne le système suivant à la base de la méthode BDD [5] :

$$\begin{aligned} \boldsymbol{S}_{BDD} \boldsymbol{P} \tilde{\boldsymbol{u}}_\Gamma &= \boldsymbol{b} - \boldsymbol{S}_{BDD} \boldsymbol{u}_{\Gamma 0} \\ \text{avec } \boldsymbol{u}_\Gamma &= \boldsymbol{P} \tilde{\boldsymbol{u}}_\Gamma + \boldsymbol{u}_{\Gamma 0} \\ \boldsymbol{u}_{\Gamma 0} &= \boldsymbol{G} \left(\boldsymbol{G}^\top \boldsymbol{S}_{BDD} \boldsymbol{G}\right)^{-1} \boldsymbol{G}^\top \boldsymbol{b} \\ \boldsymbol{P} &= \boldsymbol{I} - \boldsymbol{G} \left(\boldsymbol{G}^\top \boldsymbol{S}_{BDD} \boldsymbol{G}\right)^{-1} \boldsymbol{G}^\top \boldsymbol{S}_{BDD}. \end{aligned} \tag{5}$$

Ce système est symétrique défini positif et travaille dans l'espace orthogonal aux modes rigides.

3.2 Multipréconditionnement

Récemment, cette idée [12] a été appliquée aux méthodes de décomposition de domaine [2] comme alternative aux problèmes grossiers spectraux [13] pour rendre les méthodes plus robustes. Dans [14] la proximité théorique entre les méthodes est d'ailleurs établie.

L'idée est d'exploiter la structure additive du préconditionneur pour générer à chaque itération autant de directions de recherche que de sous-domaines. Si le vecteur $\boldsymbol{r}_i$ représente le résidu à l'itération i du gradient conjugué, on calcule classiquement le vecteur résidu préconditionné $\boldsymbol{z}_i$ par :

$$\boldsymbol{z}_i = \boldsymbol{M}_{BDD}^{-1}\boldsymbol{r}_i = \sum_s \tilde{\boldsymbol{A}}^s \boldsymbol{S}^{s^\dagger} \tilde{\boldsymbol{A}}^{s^\top} \boldsymbol{r}_i \tag{6}$$

À la place, le multipréconditionnement engendre un sous-espace de la forme :

$$\boldsymbol{Z}_i = \left[\dots \tilde{\boldsymbol{A}}^s \boldsymbol{S}^{s^\dagger} \tilde{\boldsymbol{A}}^{s^\top} \boldsymbol{r}_i \dots \right]. \tag{7}$$

Ce sous-espace est exploité à la manière des solveurs blocs pour les membres de droite multiples.

De manière à réduire le coût de calcul qui croît rapidement avec le nombre de sous-domaines, une stratégie de sélection des directions les plus pertinentes a été proposée dans [15]. Des techniques de recombinaison des directions sont également possibles [4].

3.3 Préconditionnement BLR

Afin de réduire les coûts de préconditionnement, on envisage l'utilisation de la factorisation BLR lors de l'évaluation du terme $\boldsymbol{S}^{s^\dagger}$. Pour ce faire, il est crucial de maîtriser l'effet de la factorisation approchée sur le spectre des opérateurs. En effet, il faut préserver la positivité du préconditionneur pour continuer à utiliser un algorithme de gradient conjugué.

Ensuite, il faut étudier la pertinence du problème grossier : pour garantir le sens de la résolution, il faut que le projecteur utilise le noyau véritable de l'opérateur approché par la factorisation BLR. Cependant, le sens physique et le principe de Saint Venant nous enjoignent à utiliser les véritables modes rigides.

3.4 Méthode robuste pour le calcul des noyaux locaux

Estimer correctement le noyau d'opérateurs mal conditionnés est un challenge car il est difficile de faire la distinction entre les pivots nuls et les pivots petits. Pour des problèmes très hétérogènes par exemple, la détection obtenue automatiquement par MUMPS est souvent fausse [8]. Il est possible de corriger cela en ajustant le paramètre numérique de MUMPS, mais le domaine de validité de ce paramètre diminue fortement avec le conditionnement du système.

C'est encore plus le cas lorsque l'opérateur est compressé. Nos observations numériques ont montré que MUMPS ne détecte plus le noyau pour un seuil de compression ε_{BLR} supérieur à 10^{-5}, même pour un problème homogène. La méthode BDD perd alors son problème grossier et ne peut plus être extensible.

Nous avons précédemment proposé une méthode robuste pour estimer correctement ces noyaux. Cette méthode se fonde sur une factorisation incomplète de l'opérateur et sur l'analyse des valeurs singulières d'un complément de Schur bien choisi. Le choix des inconnues à condenser utilise la notion de centralité de graphes. Nous renvoyons à [8] pour plus de détails. Cette méthode est implémentée dans le solveur Rugged [4] de Z-set. Rugged exploite la factorisation incomplète obtenue par MUMPS, il est donc entièrement compatible avec l'approche BLR. Cela permet d'obtenir la bonne taille de noyau quelle que soit la compression, les vecteurs du noyau seront par contre les vecteurs associés aux valeurs propres les plus faibles de l'opérateur compressé.

4 Résultats numériques

Les résultats d'une étude paramétrique sont présentés ici pour évaluer les performances des approches proposées. Le problème considéré est un cube en damier hétérogène (voir figure 2). Le comportement est élastique linéaire isotrope ($\nu = 0.3$) et trois valeurs d'hétérogénéité sont testées $E_r/E_b \in$

4. RobUst Graph based Generalized inverse and kErnel hanDler.

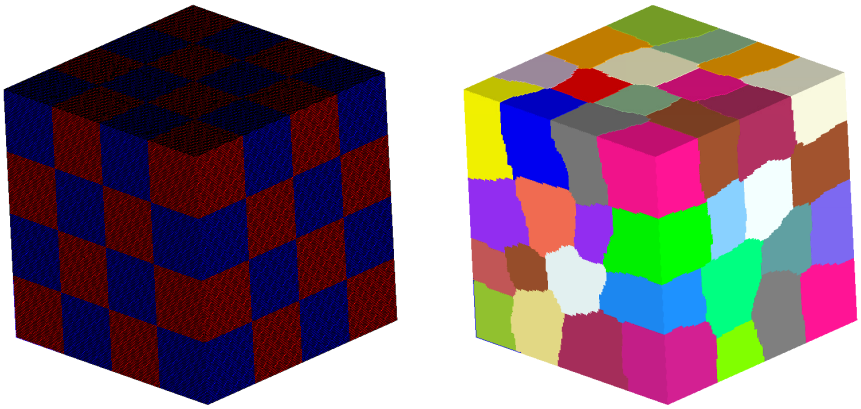

FIGURE 2 – Damier cubique hétérogène, maillage et décomposition automatique

$\{10^0, 10^2, 10^4\}$. Une face est encastrée et un déplacement dans les trois directions est imposé sur la face opposée. Le maillage est réglé, composé de 4 096 000 d'éléments c3d8 et de 4 173 281 de nœuds. Le problème global comporte donc approximativement 12.5 millions d'inconnues. Une décomposition automatique en 64 sous-domaines est utilisée. Les sous-domaines comportent en moyenne 64 000 éléments et 209250 *ddls* et le ratio h/H est de 40. Le nombre de cases du damier est également de 64 ce qui permet d'obtenir des cas pathologiques avec des hétérogénéités traversant les sous-domaines.

Les sous-domaines sont relativement grands pour mettre en évidence l'apport de la compression BLR. Le parallélisme est de type hydride MPI–multithreading, 6 cœurs sont affectés par sous-domaine ce qui fait un total de 384 cœurs. Les simulations sont réalisées avec Z-set, elles exploitent les nœuds Cascade Lake [5] du supercalculateur Sator de l'Onera. La version de MUMPS utilisée est la 5.5.1, celle de MPI est Intel MPI 22.2. Pour toutes les simulations, la méthode BDD est équipée du *scaling* rigidité et le seuil de convergence est relatif $\|\boldsymbol{r}\|_2/\|\boldsymbol{r}_0\|_2 \leq 10^{-6}$.

L'étude paramétrique a porté sur la variante BLR (UCFS ou UFSC), le seuil de compression, le mode de calcul des noyaux (MUMPS ou Rugged), l'utilisation du multipréconditionnement. Pour le cas multipréconditionné, un critère d'adaptation global ($\tau = 10^{-2}$) est utilisé avec 32 agrégats de multipréconditionnement (voir [4]). La variante BLR a peu d'influence sur les performances macroscopiques présentées ici. Afin de limiter la quantité de résultats, nous ne présentons donc que les résultats obtenus avec la variante UCFS.

Cas homogène Les résultats sont résumés dans la table 1. Pour BDD-CG, même avec un faible taux de compression, MUMPS ne détecte plus la présence de noyaux ce qui pénalise logiquement la convergence. Par contre, le nombre d'itérations le plus élevé est obtenu avec le taux de compression le plus faible ce qui paraît contre-intuitif. L'opérateur peu compressé est singulier et la non-détection des noyaux préjudiciable ici. Les résultats sont plus classiques avec Rugged, la compression augmente le nombre d'itérations et réduit l'empreinte mémoire. Le meilleur résultat $\varepsilon_{BLR} = 10^{-3}$ réduit le coût mémoire de 30% sans dégrader la convergence. Pour BDD-AMPCG, l'erreur de MUMPS sur les noyaux est compensée par le multipréconditionnement et l'influence de la compression redevient naturelle.

Hétérogénéité modérée $E_r/E_b = 10^2$ Les résultats sont résumés dans la table 2.

Pour BDD-CG, la nécessité de bien calculer les noyaux est encore bien visible. Le meilleur résultat $\varepsilon_{BLR} = 10^{-5}$, réduit le coût mémoire de 15% et le temps de calcul de l'ordre de 5%. Même avec BDD-AMPCG, un seuil de compression $\varepsilon_{BLR} = 10^{-1}$ n'atteint pas la convergence en moins de 500 itérations.

Hétérogénéité forte $E_r/E_b = 10^4$ Les résultats sont résumés dans la table 3. Pour BDD-CG, même sans compression le calcul des noyaux est erroné et la convergence n'est pas atteinte en moins de 500 itérations. Avec Rugged, seule la version sans compression atteint la convergence.

5. Intel Xeon Cascade Lake - 6240R, 24 cœurs, 2,4 GHz, 35,75MB cache. La technologie réseau est Intel Omnipath.

Hétérogénéité $E_r/E_b = 10^0$				**Solver CG**			**Solver AMPCG**		
BLR	ε_{BLR}	Noyaux	$\#\boldsymbol{G}$	# itér.	t (s)	Mém. (Mo)	# itér.	t (s)	Mém. (Mo)
		MUMPS	192	65	123.3	2525	59	120.9	2542
UCFS	10^{-1}	MUMPS	0	169	157.3	1433	168	159.0	1412
UCFS	10^{-3}	MUMPS	0	116	127.1	1778	113	125.6	1783
UCFS	10^{-5}	MUMPS	0	257	228.8	2183	70	116.1	2129
		Rugged	192	65	145.8	2462	59	139.6	2462
UCFS	10^{-1}	Rugged	192	167	193.5	1306	165	198.4	1349
UCFS	10^{-3}	Rugged	192	66	115.9	1727	65	119.5	1683
UCFS	10^{-5}	Rugged	192	65	119.0	2057	59	114.9	2085

TABLE 1 – Cas homogène : récapitulatif des résultats. La colonne t (s) représente le temps total de la simulation. La colonne Mémoire est la mesure de l'empreinte mémoire de la factorisée du préconditionneur local $\boldsymbol{S}^{s\dagger}$, $\#\boldsymbol{G}$ est la taille du problème grossier.

Hétérogénéité $E_r/E_b = 10^2$				**Solver CG**			**Solver AMPCG**		
BLR	ε_{BLR}	Noyaux	$\#\boldsymbol{G}$	# itér.	t (s)	Mém. (Mo)	# itér.	t (s)	Mém. (Mo)
		MUMPS	192	142	195.4	2541	102	168.1	2541
UCFS	10^{-1}	MUMPS	0	>500		1375	>500		1410
UCFS	10^{-3}	MUMPS	0	347	293.2	1778	238	240.3	1750
UCFS	10^{-5}	MUMPS	0	335	290.3	2120	131	168.8	2142
		Rugged	192	142	228.0	2455	102	199.2	2461
UCFS	10^{-1}	Rugged	192	>500		1346	>500		1349
UCFS	10^{-3}	Rugged	192	251	270.2	1726	182	239.0	1797
UCFS	10^{-5}	Rugged	192	142	179.7	2095	100	165.8	2087

TABLE 2 – Hétérogénéité $E_r/E_b = 10^2$: récapitulatif des résultats.

Pour BDD-AMPCG, l'erreur de MUMPS sur les noyaux conduit à une divergence pour le cas sans compression. Avec un seuil de compression inférieur à 10^{-3}, AMPCG arrive à convergence. Pour ce cas à forte hétérogénéité, les variantes compressées sont moins performantes que la variante BDD-AMPCG-Rugged sans compression.

5 Conclusion et perspectives

Cet article présente la première utilisation d'un solveur BLR dans l'étape de préconditionnement de la méthode de décomposition de domaine BDD. L'objectif est de disposer d'un préconditionneur plus léger en mémoire et en temps de calcul, à l'instar du préconditionneur *lumped* de FETI.

Une difficulté majeure apparaît dans la perte des noyaux lors de la compression BLR « agressive. Il apparaît que celle-ci peut-être compensée par la détection préalable des modes rigides via le solveur *Rugged*, ou par l'usage du multipréconditionnement. Lors de la conférence, l'étude sera enrichie et des exemples plus massifs seront présentés.

Hétérogénéité $E_r/E_b = 10^4$				**Solver CG**			**Solver AMPCG**		
BLR	ε_{BLR}	Noyaux	$\#G$	# itér.	t (s)	Mém. (Mo)	# itér.	t (s)	Mém. (Mo)
		MUMPS	149	>500		2541	Div.		2541
UCFS	10^{-1}	MUMPS	0	>500		1419	>500		1424
UCFS	10^{-3}	MUMPS	0	>500		1738	335	394.4	1733
UCFS	10^{-5}	MUMPS	0	>500		2120	138	212.9	2121
		Rugged	192	393	491.3	2456	108	256.1	2459
UCFS	10^{-1}	Rugged	192	>500		1339	>500		1343
UCFS	10^{-3}	Rugged	192	>500		1688	343	483.6	1739
UCFS	10^{-5}	Rugged	192	>500		2029	212	342.1	2089

TABLE 3 – Hétérogénéité $E_r/E_b = 10^4$: récapitulatif des résultats.

Références

[1] C. Farhat and F. X. Roux. The dual Schur complement method with well-posed local Neumann problems. *Contemporary Mathematics*, 157 :193, 1994.

[2] Pierre Gosselet, Daniel Rixen, François-Xavier Roux, and Nicole Spillane. Simultaneous FETI and block FETI : Robust domain decomposition with multiple search directions. *International Journal for Numerical Methods in Engineering*, 104(10) :905–927, 2015. nme.4946.

[3] Christophe Bovet, Augustin Parret-Fréaud, Nicole Spillane, and Pierre Gosselet. Adaptive multipreconditioned FETI : Scalability results and robustness assessment. *Computers & Structures*, pages 1–20, 2017.

[4] Christophe Bovet, Augustin Parret-Fréaud, and Pierre Gosselet. Two-level adaptation for adaptive multipreconditioned feti. *Advances in Engineering Software*, 152 :102952, 2021.

[5] Jan Mandel. Balancing domain decomposition. *Communications in Numerical Methods in Engineering*, 9(3) :233, 1993.

[6] Clark R. Dohrmann. A Preconditioner for Substructuring Based on Constrained Energy Minimization. *SIAM Journal on Scientific Computing*, 25(1) :246–258, January 2003. Publisher : Society for Industrial and Applied Mathematics.

[7] Charbel Farhat and Michel Géradin. On the general solution by a direct method of a large-scale singular system of linear equations : application to the analysis of floating structures. *International Journal for Numerical Methods in Engineering*, 41(4) :675–696, 1998.

[8] Christophe Bovet. On the use of graph centralities to compute generalized inverse of singular finite element operators : Applications to the analysis of floating substructures. *International Journal for Numerical Methods in Engineering*, 124(9) :1933–1964, 2022.

[9] Mario Bebendorf. Hierarchical matrices. *Lecture notes in computational science and engineering, v.63 (2008)*, 63, 01 2008.

[10] Théo Mary. *Block Low-Rank multifrontal solvers : complexity, performance, and scalability*. PhD thesis, Université de Toulouse, 2017.

[11] Grégoire Pichon, Eric Darve, Mathieu Faverge, Pierre Ramet, and Jean Roman. Sparse supernodal solver using block low-rank compression : Design, performance and analysis. *International Journal of Computational Science and Engineering*, 27 :255–270, 2018.

[12] Robert Bridson and Chen Greif. A multipreconditioned conjugate gradient algorithm. *SIAM J. Matrix Anal. Appl.*, 27(4) :1056–1068 (electronic), 2006.

[13] Nicole Spillane and Daniel J. Rixen. Automatic spectral coarse spaces for robust FETI and BDD algorithms. *International Journal for Numerical Methods in Engineering*, 95(11) :953–990, 2013.

[14] Michael C. Leistner, Pierre Gosselet, and Daniel J. Rixen. Recycling of Solution Spaces in Multi-Preconditioned FETI Methods Applied to Structural Dynamics. *International Journal for Numerical Methods in Engineering*, 2018.

[15] Nicole Spillane. An Adaptive Multipreconditioned Conjugate Gradient Algorithm. *SIAM J. Sci. Comput.*, 38(3) :A1896–A1918, 2016.