
1. Introduction

Nous nous intéressons ici à la résolution des systèmes linéaires de la forme

$$Ax = b \quad (1)$$

sur des architectures parallèles, avec $A \in \mathbb{R}^{n \times n}$, x et $b \in \mathbb{R}^n$. Nous supposons que A est une matrice creuse, non symétrique, et qu'elle est très grande (en taille et en nombre d'éléments non nuls). Dans ce contexte, les méthodes itératives basées sur les sous-espaces de Krylov tel GMRES [1] sont assez efficaces. Mais pour être robustes, ces méthodes doivent résoudre un système préconditionné à gauche 2.(a) ou à droite 2.(b).

$$M^{-1}Ax = M^{-1}b \quad (a) \quad AM^{-1}Mx = b \quad (b) \quad (2)$$

Les matrices M ou M^{-1} n'existent pas explicitement. De façon générale, l'opérateur de préconditionnement M^{-1} permet, sans être coûteux à appliquer, d'améliorer le conditionnement du système modifié et donc d'accélérer la convergence de la méthode itérative. Pour un environnement de calcul distribué, les préconditionneurs basés sur les méthodes de décomposition de domaines sont bien adaptés. Elles permettent de diviser le système initial en sous-systèmes, de les résoudre sur les différents processeurs, et de construire la solution globale à partir des solutions de sous-systèmes.

Dans ce travail, nous considérons, à partir du graphe de la matrice, une décomposition en sous-domaines avec recouvrement. Dans ce cas, le préconditionneur associé est équivalent à une méthode itérative de type Schwarz additif ou Schwarz multiplicatif. Des travaux récents ont exhibé une formulation explicite associée à une itération de Schwarz multiplicatif [2]. Pour faire usage au maximum des opérations parallèles, cette forme explicite est utilisée comme préconditionneur pour la version parallèle de GMRES [3]. Cette approche offre de bonnes performances sur des cas-tests pratiques [4, 5]. Mais en augmentant le nombre de sous-domaines, la convergence de la méthode s'en trouve détériorée. Cela est généralement vrai pour les méthodes de décomposition de domaine appliquées aux problèmes fortement non elliptiques [6]. Pour garantir la convergence de la méthode, il est nécessaire de limiter le nombre de sous-domaines. Cependant, réduire le nombre de sous-domaines entraîne une augmentation de la taille des systèmes associés aux sous-domaines et par conséquent de leur temps de résolution.

Dans cet article, nous introduisons un deuxième niveau de parallélisme pour résoudre les sous-systèmes issus de la décomposition précédente. Nous privilégions ici les méthodes directes car elles permettent d'obtenir au final un préconditionneur plus robuste et plus performant. Pour des nœuds de calcul multicœurs, ce deuxième niveau de parallélisme permet d'utiliser de façon efficace la puissance de calcul disponible. Cette approche n'est pas nouvelle ; elle est utilisée le plus souvent pour augmenter la scalabilité des méthodes basées sur les préconditionneurs de type Schwarz additif [7, 8].

La suite de cet article est organisée comme suit : dans la section 2, nous rappelons les grandes étapes de GMRES préconditionné par Schwarz multiplicatif. La section 3 décrit le mécanisme pour résoudre de façon parallèle les systèmes associés aux sous-domaines. En fonction de la mémoire disponible sur les nœuds de calcul, cette résolution peut se faire sur les cœurs d'un seul nœud ou sur un groupe de nœuds distincts. Nous présentons dans la section 4 les performances de cette approche à partir des matrices issues d'applications réelles. Quelques remarques pour la suite de ce travail sont proposées à la fin du document.

2. GMRES préconditionné par Schwarz multiplicatif

Les méthodes basées sur Schwarz multiplicatif permettent de résoudre itérativement le système 1 en utilisant une décomposition de domaines avec recouvrement. Mais leur convergence n'est garantie que pour des matrices symétriques et des M-matrices [9]. Pour cette raison, elles sont plutôt utilisées comme préconditionneurs pour les méthodes de type Krylov. Dans cette section, nous rappelons la forme explicite du préconditionneur associé à une itération de Schwarz multiplicatif. Ensuite, nous montrons brièvement comment l'utiliser dans un environnement parallèle avec un accélérateur de type GMRES.

2.1. Forme explicite de Schwarz multiplicatif

Considérons une décomposition de A en p partitions A_i qui se recouvrent ; on note C_i la matrice de recouvrement entre A_i et A_{i+1} ($i = 1, \dots, p-1$). Ici, chaque partition A_i a au plus deux voisins (voir figure 1.a). Un tel partitionnement peut être obtenu directement à partir du graphe d'adjacence de la matrice [10]. On définit \bar{A}_i et \bar{C}_i respectivement les matrices A_i et C_i complétées par l'identité à la taille de A (voir figure 1.b).

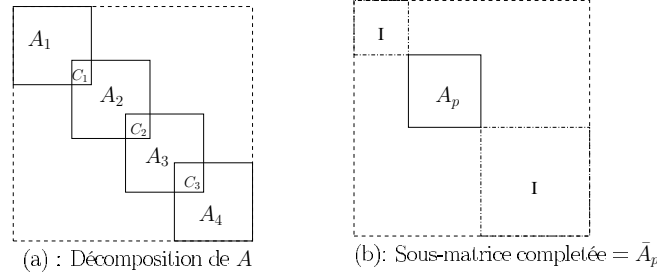


Figure 1. Partitionnement de A en quatre sous-domaines

Si \bar{A}_i et \bar{C}_i ($i = 1, \dots, p-1$) sont des matrices non-singulières, alors la forme explicite de Schwarz multiplicatif est définie [2] par :

$$M^{-1} = \bar{A}_p^{-1} \bar{C}_{p-1} \bar{A}_{p-1}^{-1} \bar{C}_{p-2} \dots \bar{A}_2^{-1} \bar{C}_1 \bar{A}_1^{-1} \quad (3)$$

Contrairement à la formulation classique, cette forme explicite permet de calculer de façon indépendante l'approximation courante et le nouveau vecteur résidu. Cependant, elle n'enlève pas la dépendance intrinsèque entre les sous-domaines. Par exemple, l'opération $M^{-1}x$ est séquentielle par rapport aux sous-domaines, à moins d'utiliser les méthodes classiques de coloriage des sous-domaines. Cependant, il est possible d'obtenir un parallélisme de type pipeline si on applique successivement M^{-1} à une suite de vecteurs. Dans la section suivante, nous montrons brièvement comment obtenir ce pipeline lors de la construction des vecteurs de la base du sous-espace de Krylov.

2.2. Application de Schwarz multiplicatif à GMRES

La méthode GMRES(m) préconditionnée minimise le résidu $r_m = M^{-1}(b - Ax_m)$ dans le sous-espace de Krylov $x_0 + \mathcal{K}_m$ où x_0 est une approximation initiale. La nouvelle approximation est de la forme $x_m = x_0 + V_m y_m$ où V_m est une base orthonormale de \mathcal{K}_m et y_m tout vecteur qui minimise le résidu r_m . La version parallèle de GMRES [3] permet de calculer V_m par le procédé d'Arnoldi parallèle, ceci en évitant les communications globales (de type produit scalaire) entre les processeurs. Les vecteurs de V_m peuvent

donc être formés dans un pipeline à travers tous les processeurs participants. Pour une meilleure stabilité, cette approche utilise la base de Newton définie par

$$\tilde{V}_{m+1} = [\mu_0 v, \mu_1 (M^{-1}A - \lambda_1 I)v, \dots, \mu_m \prod_{j=1}^m (M^{-1}A - \lambda_j I)v] \quad (4)$$

avec λ_i ($i = 1, \dots, m$) les valeurs propres approchées de $M^{-1}A$ ordonnées selon l'ordre de Leja [11] et μ_i les coefficients de normalisation. La construction de V_m se fait en trois étapes :

1. Construction a priori de la base de Krylov :

$$V_{m+1} = [v, (M^{-1}A - \lambda_1 I)v, \dots, \prod_{j=1}^m (M^{-1}A - \lambda_j I)v] \quad (5)$$

2. Détermination des μ_j : $\sigma_{j+1} = 1/|(M^{-1}A - \lambda_j I)\tilde{v}_j|$, $\mu_{j+1} = \sigma_{j+1}\sigma_j \dots \sigma_0$

3. Factorisation QR de la base : $V_{m+1} = \hat{Q}_{m+1}\hat{R}_{m+1}$

À partir de $M^{-1}AV_m = V_{m+1}T_m$, où T_m est une matrice bidiagonale, on peut écrire

$$M^{-1}A\hat{Q}_m = \hat{Q}_{m+1}\hat{R}_{m+1}T_m\hat{R}_m^{-1} = \hat{Q}_{m+1}\bar{H}_m \quad (6)$$

L'équation (6) est une relation d'Arnoldi avec \bar{H}_m une matrice de Hessenberg supérieure. Si la matrice \hat{R}_m^{-1} est mal conditionnée, l'algorithme risque d'être instable. De ce fait, la projection est faite en utilisant plutôt la relation

$$M^{-1}AV_m = \hat{Q}_{m+1}\hat{R}_{m+1}T_m = \hat{Q}_{m+1}\bar{C}_m \quad (7)$$

où \bar{C}_m est aussi une matrice de Hessenberg. La nouvelle approximation est donc $x_m = x_0 + V_m y_m$, où y_m est solution de $\min\|\beta e_1 - \bar{C}_m y_m\|$.

Pour le calcul des étapes 1 et 2 ci-dessus, considérons le découpage de la figure 1. Chaque processeur détient un sous-domaine A_p . Un tel découpage en blocs de ligne est effectué aussi pour les vecteurs de V_m . Si chaque processeur p calcule la suite de vecteurs $v_{i+1}^p = M^{-1}A v_i^p$ alors il en résulte une double récurrence : une récurrence pour le calcul des différents vecteurs de la base et une récurrence entre les sous-domaines pour le calcul des sous-vecteurs. La figure 2 présente un exemple de construction de trois vecteurs de Krylov avec quatre sous-domaines. L'opération $\bar{C}_p A_p^{-1} y^p$ se fait de façon séquentielle entre les sous-domaines. Mais dès que le processeur $p+2$ a terminé cette opération, le processeur p peut commencer le calcul d'un nouveau vecteur. Les coefficients μ_i de l'étape 2 sont aussi obtenus dans le pipeline. Pour cela, dès qu'un processeur a sa version finale de v_i^p , il calcule son produit scalaire et l'envoie au processeur suivant avec la partie recouverte du sous-vecteur. Le dernier processeur calcule la norme globale et le diffuse aux autres processeurs. Pour plus de détails sur l'implémentation de cette méthode, consulter [4, 12].

3. Résolution des systèmes associés aux sous-domaines

Pour les méthodes de décomposition de domaines, l'approche classique est d'associer un sous-domaine à un seul processeur. Lors de l'application de l'opérateur M^{-1} , chaque processeur p résout un sous-système de la forme $A_p v = y$. En tant que préconditionneur, la solution à ce sous-système est obtenue de façon « exacte » (factorisation LU) ou approximative (factorisation incomplète, inverse approchée, méthode itérative) :

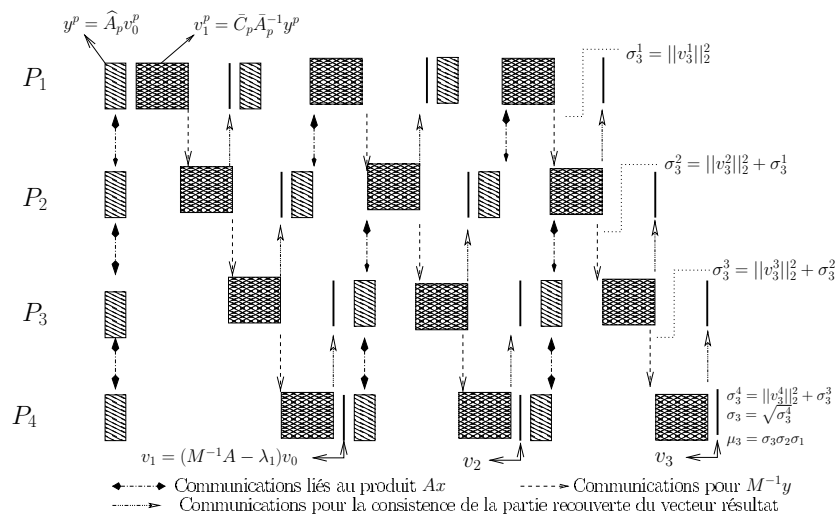


Figure 2. Construction de la base de Krylov

– Une résolution « exacte » permet d’avoir un préconditionneur plus robuste ; en revanche elle requiert un espace-mémoire assez important pour chaque sous-domaine. Ici, il faut stocker la sous-matrice A_p ainsi que les facteurs L et U .

– Pour une résolution approximative, le préconditionneur devient moins robuste. Le processus itératif a plus de risques de stagner pour la plupart des problèmes complexes.

Ici, notre approche est de garantir une certaine robustesse en privilégiant une méthode directe pour les sous-systèmes. Cette résolution est accélérée à l’intérieur de chaque sous-domaine en introduisant un deuxième niveau de parallélisme. Ceci nous conduit aussi à exploiter efficacement la puissance de calcul offerte sur les nœuds de calcul.

Pour les nœuds de calcul SMP (figure 3.a), supposons qu’un processeur ait besoin de plus de mémoire que ce qui lui est allouée pour traiter un sous-domaine. Alors il faudra réserver plusieurs processeurs dans le nœud SMP pour avoir plus de mémoire. Dans ce cas, un seul processeur sera actif à un moment donné ; c’est le cas de la figure (3.b). Avec notre approche, tous les processeurs du nœud SMP sont actifs car ils résolvent le sous-système en parallèle (3.c). Il est aussi possible que la mémoire disponible sur tout le nœud de calcul ne soit pas suffisante ; dans ce cas, la sous-matrice peut être distribuée sur plus d’un nœud de calcul SMP. C’est le cas de la figure (3.d).

Bien que la résolution du sous-système puisse se faire sur un nœud SMP, la communication se fait par passage de messages. Ceci nous permet de distribuer la sous-matrice sur plus d’un nœud de calcul. Cette approche exige qu’il existe plusieurs groupes et communicateurs MPI. Un premier groupe ou communicateur qui gère l’échange entre les différents sous-domaines. Ensuite, pour chaque sous-domaine, un second communicateur est créé pour l’échange de messages à l’intérieur du sous-domaine.

4. Application

Dans cette section, nous présentons quelques résultats numériques pour valider l’efficacité de cette approche. Les tests sont effectués sur le cluster *Paramount* de la grille expérimentale GRID’5000 [15]. Le cluster est composé de nœuds SMP dual-socket quadri-

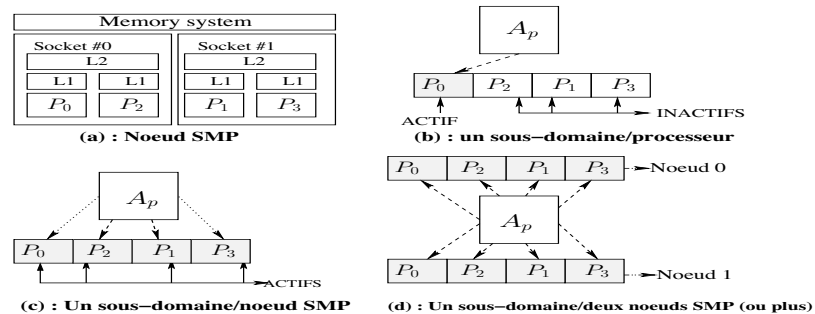


Figure 3. Distribution d'un sous-domaine sur des nœuds SMP

core (Intel Xeon cadencé à 2.33GHz). Chaque nœud possède 8Go de mémoire vive. Les nœuds sont interconnectés par un switch myrinet à 10Gb/sec. La bibliothèque MPICH2 est utilisée pour les communications MPI. MPICH2-Nemesis est utilisé pour le canal de communication intra-nœuds. Les matrices de tests sont présentées dans le tableau 1. Toutes ces matrices sont non-symétriques. La matrice CASE_07 est issue d'une discrétisation en volumes finies d'équations de Navier Stokes paramétrisées.

Matrice	taille	nonzeros	Type	Origine
XENON1	48 600	1 181 120	materials problem	UFL[13]
CASE_07	233 786	11 762 405	Fluid dynamics simulation	FLUOREM[14]
ATMOSMODL	1 489 752	10 319 760	Fluid dynamics simulation	UFL[13]

Tableau 1. Matrices de tests

4.1. Tests de convergence

Dans un premier temps, nous étudions la vitesse de convergence de GMRES pour résoudre ces systèmes. L'erreur relative à la convergence est fixée à 10^{-9} . La taille de la base de Krylov est de 64 pour la matrice CASE_07, de 32 pour la matrice XENON1 et de 16 pour ATMOSMODL. Cette taille est la plus petite possible qui empêche l'algorithme de stagner. Sur la figure 4, l'historique de la convergence est donnée en fonction du nombre d'itérations pour 4, 8 et 16 sous-domaines. On remarque bien que la méthode converge vers la solution du système. Cependant, le nombre d'itérations croît avec le nombre de sous-domaines. Il est donc nécessaire de limiter le nombre de sous-domaines pour garantir la convergence.

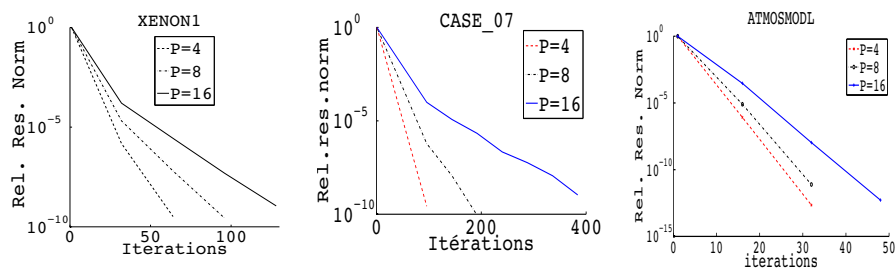


Figure 4. Convergence de la méthode

4.2. Temps d'exécution

Dans cette partie, nous regardons le temps d'horloge requis pour converger vers la solution. On associe un sous-domaine à chaque nœud de calcul du cluster. Ici, nous comparons la stratégie présentée dans cet article (*Two-level*) à l'approche classique (*One-level*). Avec cette dernière, la solution dans le sous-domaine est obtenue par un seul processeur dans le nœud SMP. Le solveur séquentiel SuperLU [16] est utilisé pour cette résolution. Lors de la factorisation *LU*, les noyaux de calcul BLAS sont multithreadés grâce à Goto-BLAS. Cela permet d'utiliser dans une certaine mesure les autres coeurs du nœud SMP. Dans la stratégie *Two-level*, La résolution du sous-système se fait en parallèle à l'intérieur du nœud de calcul. Ici, nous utilisons SuperLU_DIST pour cette résolution ; les quatre coeurs du nœud SMP étant considérés comme des processeurs, chacun avec sa zone de mémoire propre. Cependant, les communications intra-nœuds sont optimisées grâce au canal de communication MPICH2-Nemesis. Également, le placement des processus sur les coeurs peut être géré automatiquement par le système d'exploitation ou manuellement grâce à l'outil MPICH2-Hydra.

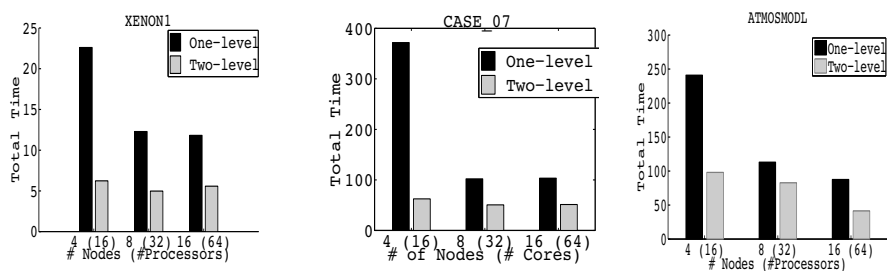


Figure 5. Temps d'exécution

La figure 5 montre le temps d'exécution de ces deux stratégies pour les matrices de la table 1. Le nombre de nœuds est équivalent au nombre de sous-domaines. Le nombre total de processeurs est donné entre parenthèses. Dans ces exécutions, chaque sous-système est résolu par quatre processeurs. On constate d'abord que le temps global diminue avec le nombre de processeurs quelle que soit la stratégie utilisée. Bien plus, en considérant deux niveaux de parallélisme, on diminue considérablement le temps global d'exécution. Ceci est bien visible avec quatre sous-domaines. Pour les cas XENON1 et CASE_07, lorsque l'on augmente le nombre de sous-domaines, le temps d'exécution reste sensiblement le même pour huit et seize sous-domaines. Cela est dû au fait que le nombre d'itérations croît considérablement comme indiqué sur la figure 4. Pour ATMOSMODL, le nombre d'itérations croît de façon linéaire. On observe dans ce cas une bonne scalabilité de la méthode avec les deux niveaux de parallélisme.

5. Conclusion

Dans cet article, nous avons présenté une méthode hybride pour la résolution des systèmes linéaires sur une architecture distribuée. Cette méthode utilise un parallélisme à deux niveaux : un premier niveau de parallélisme sous forme de pipeline pour la construction de la base de Krylov. Le deuxième niveau de parallélisme est utilisé à l'intérieur de chaque sous-domaine pour le calcul des sous-systèmes linéaires issus de la décomposition. Nous avons montré que cette approche permet d'utiliser pleinement tous les processeurs d'un nœud de calcul SMP. Les résultats numériques montrent un gain de temps

important sur des systèmes issus d'applications réelles. Cependant, plusieurs améliorations sont possibles pour ce travail ; par exemple, un placement plus « judicieux » des processus sur les cœurs pourrait aboutir à une utilisation plus efficace du cache-mémoire. Enfin, il serait intéressant de comparer, pour les sous-systèmes de petite taille, une résolution en mémoire distribuée et en mémoire partagée. Pour les sous-systèmes qui ne peuvent pas tenir dans la mémoire d'un seul noeud, on pourrait aussi considérer une factorisation out-of-core. Ceci peut facilement être intégré dans notre solveur en utilisant un solveur direct ayant déjà cette fonctionnalité.

Remerciements : Ce travail a été effectué dans le cadre du projet LIBRAERO financé par l'ANR-RNTL. Les tests numériques ont été effectués sur la grille de calcul Grid'5000[15].

6. Bibliographie

- [1] Y. SAAD, « Iterative methods for sparse linear systems », SIAM, 2003.
- [2] G.-A. ATENEKENG K., E. KAMGNIA, B. PHILIPPE. « An explicit formulation of the multiplicative Schwarz preconditioner ». *Applied Numerical Mathematics*, vol. 57 p. 1197-1213, 2007
- [3] J. ERHEL, « A parallel GMRES version for general sparse matrices ». *Electronic Transaction on Numerical Analysis*, 3 :160-176, 1995.
- [4] G.-A. ATENEKENG-K. « Parallélisation de GMRES préconditionné par une itération de Schwarz multiplicatif », *PhD thesis, University of Rennes 1 and University of Yaounde I*, 2008.
- [5] D. NUENTSA WAKAM, J. ERHEL, É. CANOT, G.-A. ATENEKENG K. « A comparative study of some distributed linear solvers on systems arising from fluid dynamics simulations ». *In Proc. of Int. Conf. on Parallel Computing*, Lyon, Sept. 2009.
- [6] A. TOSELLI, O. WIDLUND « Domain Decomposition Methods - Algorithms and Theory. Springer series in computational mathematics », vol. 34 Springer, 2005
- [7] L. GIRAUD, A. HAIDAR, L. T. WATSON. « Parallel scalability study of hybrid preconditioners in three dimensions ». *Parallel Computing*, 34 :363-379, 2008.
- [8] A. HAIDAR « On the parallel scalability of hybrid linear solvers for large 3D problems ». *PhD thesis, Institut National Polytechnique de Toulouse*, 2008.
- [9] M. BENZI, A. FROMMER, R. NABBEN, D. B. SZYLD, « Algebraic theory of Multiplicative Schwarz methods ». *Numerische Mathematik*, 89(4) :605-639, 2001
- [10] G.-A. ATENEKENG KAHOU, L. GRIGORI, M. SOSONKINA, « A partitioning algorithm for block-diagonal matrices with overlap. » *Parallel Computing*, 34 :332-344, 2008
- [11] Z. BAI, D. HU, L. REICHEL, « A Newton basis GMRES implementation. » *IMA J. Numer. Anal*, 14(4) :563-581, 1994
- [12] I. SOUOPGUI, « Parallélisation d'une double récurrence dans GMRES », *DEA report, University of Yaoundé I*, Nov. 2005
- [13] T. DAVIS, University of Florida sparse matrix collection, <http://www.cise.ufl.edu/~davis/sparse>
- [14] FLUOREM S.A., The Fluorem Matrix Collection, LIB0721 2.0 / FP-SA, <http://www.fluorem.com>, 2009
- [15] ALADDIN-G5K, « GRID'5000 experimental testbed », <https://grid5000.fr>, 2009
- [16] X. S. LI AND J. W. DEMMEL. « SuperLU_DIST : A Scalable Distributed-Memory Sparse Direct Solver for Unsymmetric Linear Systems » *ACM Transactions on Mathematical Software*, 29(2) :110-140, 2003