Commissariat à l'Énergie Atomique

Rapport de stage de fin d'études

Comparaison de différentes méthodes de mise à l'échelle de matrices type système et comparaison de différents solveurs/préconditionneurs Petsc dans un context HPC

Bada Gaye

Mathématiques Appliquées et Calcul Scientifique
Sup'Galilée

Encadrants :
Mr Florian Caro, CEA,
Mme Laura Grigori, INRIA.

Tuteur Sup'Galilée :
Mr Hakim Boumaza

Paris, le 24 septembre 2012

Remerciements

Je tiens en premier lieu à remercier Florian Caro, ingénieur au département de Modélisation des Systèmes et Structures, pour m'avoir donné la possibilité d'éffectuer mon stage de fin d'études au CEA Saclay. Je lui suis très reconnaissant pour son encadrement et pour son entière disponibilité, ainsi que pour son écoute attentive et ses précieux conseils.

Je tiens à remercier les autres personnels du service qui m'ont bien accueilli au sein de leur équipe et également les stagiaires avec lesquels, on passait des bons moments.

Je remercie aussi Laura Grigori pour m'avoir accueilli en premier temps dans l'équipe d'INRIA Saclay et son aide théorique. Je remercie Riadh Fezzani pour son aide aussi bien théorique que pratique qu'il m'a apporté durant le stage et tout le personnel de l'équipe d'INRIA Saclay pour leur accueil chaleureux.

J'adresse toute ma gratitude et tout mon respect aux professeurs de la MACS ainsi qu'aux professeurs des cours communs de Sup'Galilée qui durant ces trois années, m'ont formé et accompagné. Je remercie en particulier le responsable de ma formation Mr Olivier Lafitte qui m'a donné confiance, Mr Pascal Omnes qui m'a mis en contact avec mon tuteur de stage et Mr Hakim Boumaza pour la lecture du rapport avec ses précieux remarques et suggestions.

Enfin, une pensée particulière à mes camarades de formation MACS qui durant ces trois années, m'ont su aider, me supporter et me faire rire.

Sommaire

1 Introduction

1.1 Présentation du problème

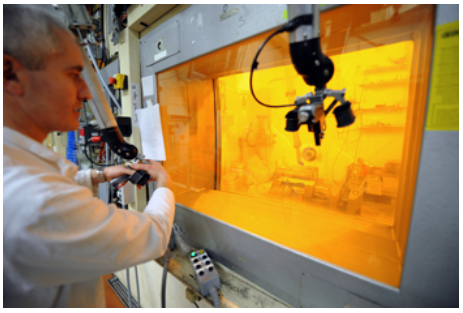
Dans le cadre des applications étudiées au commissariat à l'énergie atomique (dégradation des bétons sous irradiation, simulation d'écoulements diphasiques en milieu poreux dans les sites de stockage de déchets radioactifs, calculs de thermo-hydraulique dans les cœurs de réacteur...), les simulations numériques produisent des matrices de grandes tailles généralement mal conditionnées. La résolution performante des systèmes linéaires liées à ces matrices est donc un enjeu majeur pour le CEA.

La plupart des préconditionneurs existants, comme la factorisation incomplète ILU, ont des problèmes de scalabilité. Ceci signifie qu'en augmentant le nombre de processeurs ou la taille du problème, le nombre d'itérations augmente. Ceci est dû à la présence de quelques modes basses fréquences. Des travaux ont été effectués afin de concevoir des préconditionneurs satisfaisant une propriété de filtrage. Ceux-ci permettent d'éliminer les modes basses fréquences et d'accélérer de manière significative la convergence de la méthode itérative.

Le but de ce projet consiste dans un premier temps à comprendre les méthodes numériques développées dans le cadre du projet PETALh. Ensuite, nous devons tester et comparer la scalabilité des différentes méthodes développées sur des matrices issues de cas d'application du CEA. Enfin, nous devons faire une étude des solveurs et préconditionneurs de Petsc sur un cas de thermo-hydraulique dans le code Trio_U.

Après avoir présenté le commissariat à l'énergie atomique, nous décrivons la méthode de filtrage de préconditionnement en présentant sa construction et ses avantages par rapport à la méthode ILU. Ensuite, nous nous intéressons à la mise à l'échelle des matrices en évoquant les différentes méthodes développées et en faisant leur comparaison. Nous terminons par une étude des solveurs en l'occurrence BICGSTAB et GCP et préconditionneurs de Petsc sur une matrice issue d'un problème thermo-hydraulique sur un grand nombre de processeurs.

1.2 Le commissariat à l'énergie atomique



Le commissariat à l'énergie atomique(CEA) a été créé en 1945 sous l'impulsion de l'ex président de la république, le Général de Gaulle et Frédéric Joliot. Il est composé d'environ 16000 chercheurs, ingénieurs et collaborateurs qui se consacrent à cette mission qui couvre le court, le moyen et le long terme. En 2010, le budget du CEA était de l'ordre de 3.5 milliards d'euros.

Le CEA intervient dans quatre grands domaines : l'énergie, la défense et la sécurité globale, les technologies pour l'information et les technologies pour la santé, associés à une recherche fondamentale d'excellence, et sur la conception et l'exploitation des très grandes infrastructures de recherche.

Organisme public singulier, le CEA occupe une place à part dans le paysage français de la recherche où il est désormais l'opérateur majeur de la recherche scientifique et technologique dans le champ des énergies à bas taux de carbone en complément des missions qui lui avaient été déjà confiées antérieurement. Le CEA étudie des palettes de solutions scientifiques et techniques afin que les décideurs, pouvoirs publics et industriels soient en mesure de prendre, en toute connaissance de cause et en tout moment, les décisions les mieux adaptées pour le présent et pour l'avenir.

Le commissariat à l'énergie atomique est dirigé par le haut commissaire à l'énergie atomique et l'administrateur général. Les activités scientifiques et technologiques de l'entreprise sont partagées en quatre grands pôles

- Le pôle recherche fondamentale,
- La direction des recherches technologiques,
- La direction de l'énergie nucléaire,
- La direction des applications militaires.

Chaque direction est divisée en départements et unités qui peuvent être rattachées à des directions d'objectifs. Dans le centre de Saclay, la direction est composée des

- Département réacteurs et services nucléaires,
- Département patrimoine et infrastructures,
- Département de sécurité et de protection,
- Département de physico-chimie,
- Département de modélisation des systèmes et structures.

Dans chaque département, on a plusieurs services qui se subdivisent en cinq ou six laboratoires d'une quinzaine de personnes chacun.

1.3 La direction de l'Energie Nucléaire

Le pôle nucléaire du CEA (Direction de l'Energie Nucléaire) est en charge de la R&D consacrée à l'industrie nucléaire. Ces programmes sont faits à la demande de l'Etat ou des industriels et sont liés aux grands enjeux du développement de l'Energie Nucléaire : face aux défis climatiques d'aujourd'hui, le nucléaire, énergie à bas taux de carbone, est un acteur de tout premier plan.

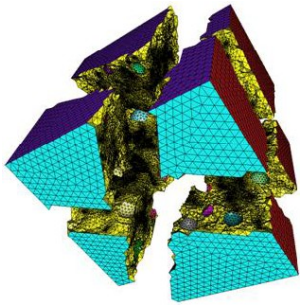
Encadrés par la loi d'orientation du 13 juillet 2005 les programmes de la DEN concernent : le soutien à l'industrie nucléaire nationale, le développement des combustibles nucléaires innovants, la gestion durable des déchets nucléaires et le développement des technologies des réacteurs nucléaires du futur (fission ou fusion).

Pour la satisfaction de ses objectifs, la DEN dispose d'une part d'installations expérimentales polyvalentes (réacteurs, maquettes critiques), et de laboratoires spécifiques au domaine nucléaire (dits « labos chauds »). D'autre part, à côté de ses moyens expérimentaux, la DEN utilise et développe des outils de simulation numérique, qui permettent d'une part l'exploration par le calcul de domaines difficilement atteignables par l'expérimentation et d'autre part contribuent à la réduction des coûts d'investissements et réduisent les durées de développement.

Ces programmes sont menés au niveau national et dans le cadre de nombreuses coopérations européennes et internationales. Les activités du pôle nucléaire du CEA se déploient principalement sur trois sites

- **Saclay**, où se trouvent la Direction déléguée aux Activités Nucléaires de Saclay (DANS) tournée plus particulièrement vers la recherche amont sur les matériaux, le calcul intensif et la simulation numérique, ainsi que l'Itésé, Institut en charge des études d'évaluation de technico-économie des systèmes énergétiques
- **Cadarache**, orienté vers la recherche sur les réacteurs (physique des cœurs, conception, technologie, sûreté) et sur leurs combustibles (comportement en service et en situation accidentelle, nouveaux concepts),
- **Marcoule** qui constitue un pôle scientifique de pointe sur le cycle du combustible, dans les domaines amont et aval avec des outils expérimentaux uniques (laboratoire Atalante) et des compétences reconnues en chimie des actinides.

1.4 Le Département de Modélisation des Systèmes et Structures



Le Département de modélisation des systèmes et structures(DM2S) est constitué de 250 personnes et appartient à la Direction de l'Énergie Nucléaire de Saclay. Le domaine d'activité du DM2S est centré autour de la simulation, la conception et l'évaluation des systèmes nucléaires et plus largement des systèmes énergétiques.

Le DM2S conduit des actions de recherche théoriques et expérimentales et développe des méthodologies de simulation dans les domaines suivants :
- mécanique des structures,
- physique des cœurs de réacteur,
- calculs de criticité, de fluence, de protection,
- écoulements diphasiques incompressibles et compressibles,
- transferts dans les milieux géologiques,
- fonctionnement d'ensemble des procédés.

Le DM2S construit ces méthodologies en s'appuyant sur des plateformes logicielles développées en interne ou en partenariat avec d'autres unités ou organismes et sur des essais appropriés.

Il les met en œuvre dans le cadre des études qui lui sont confiées, notamment dans les domaines de la tenue et intégrité des structures des installations nucléaires, le comportement du cœur et la protection contre les rayonnements, la simulation de situations incidentelles, le rendement des procédés d'enrichissement (ultracentrifugation et séparation isotopique par laser). Il élabore des codes et normes de conception ; il conçoit des équipements et en assure la maîtrise d'œuvre pour les départements de la DEN ou du CEA.

2 Méthode BFD (Block Filtering Decomposition)

Nous présentons dans cette section la méthode BFD (Block Filtering Decomposition) développée dans le cadre du projet ANR PETALh.

Pour cela, considérons un système linéaire

$$A \times x = b, \tag{1}$$

où A est une matrice carrée inversible de taille $n \times n$.

Étant donné une matrice M inversible d'ordre n, le système

$$M^{-1}Ax = M^{-1}b, \tag{2}$$

admet la même solution que l'équation (1). Dans la suite, on dira que M est la matrice de préconditionnement associée au système (2).

La convergence d'une méthode itérative dépendra beaucoup des propriétés spectrales de la nouvelle matrice $M^{-1}A$. En fait, dans certaines méthodes itératives, on a seulement besoin de connaitre le produit matrice-vecteur. Lorsque le système est préconditionné, l'opération matrice-vecteur correspond à

$$w = M^{-1}Ax \longleftrightarrow Mw = Ax, \tag{3}$$

par conséquent,

$$Mw = b. \tag{4}$$

Savoir choisir un préconditionneur est une question fort délicate qui, à ce jour, n'a pas de solution universelle. Ainsi, on construit notre propre préconditionneur en tenant compte de l'aspect système de notre domaine.

2.1 Présentation de la méthode

On cherche à construire un préconditionneur par bloc de filtrage M qui satisfait la condition de filtrage suivante

$$(M - A) \times t = 0, \tag{5}$$

où t est un vecteur de filtrage.

Pour construire le vecteur t, on a choisi les petites valeurs propres de la matrice A.

Soit A une matrice de taille $n \times n$ partitionnée en $N \times N$ blocs A_{ij} de taille $p \times p$ avec $N = n/p$ où p est le nombre d'équations de notre système

$$A = \begin{pmatrix} A_{11} & \cdots & A_{1N} \\ A_{21} & \cdots & A_{2N} \\ \vdots & \ddots & \vdots \\ A_{N1} & \cdots & A_{NN} \end{pmatrix} \tag{6}$$

Une factorisation exacte par bloc de type LDU est donnée par

$$A = \begin{pmatrix} D_{11} & & & \\ L_{21} & D_{22} & & \\ \vdots & \ddots & \ddots & \\ L_{N1} & \cdots & L_{N,N-1} & D_{NN} \end{pmatrix} \begin{pmatrix} D_{11}^{-1} & & & \\ & \ddots & & \\ & & & D_{NN}^{-1} \end{pmatrix} \begin{pmatrix} D_{11} & U_{12} & \cdots & U_{1N} \\ & D_{22} & \ddots & \vdots \\ & & \ddots & U_{N-1,N} \\ & & & U_{N,N} \end{pmatrix}, \tag{7}$$

où $D_{ii}, i = 1, ..., N$ sont des matrices carrées inversibles de taille $p \times p$.

Soient

$$L = \begin{pmatrix} 0 & & & \\ L_{21} & \ddots & & \\ \vdots & \ddots & \ddots & \\ L_{N1} & \cdots & L_{N,N-1} & 0 \end{pmatrix}, D = \begin{pmatrix} D_{11} & & \\ & \ddots & \\ & & D_{N,N} \end{pmatrix} \text{ et } U = \begin{pmatrix} 0 & U_{12} & \cdots & U_{1N} \\ & 0 & \ddots & \vdots \\ & & \ddots & U_{N-1,N} \\ & & & 0 \end{pmatrix}. \quad (8)$$

La factorisation (7) peut alors s'écrire

$$A = (L + D)D^{-1}(U + D). \quad (9)$$

Dans la suite de cette partie, on se réfère aux blocs des matrices L, D, U et de la matrice C définie par

$$C_{ij} = \begin{cases} L_{ij}, & \text{si} \quad i > j, \\ U_{ij}, & \text{si} \quad i < j, \\ D_{ij}, & \text{si} \quad i = j. \end{cases} \quad (10)$$

En d'autre termes, la matrice C peut s'écrire comme $C = L + D + U$. Les blocs de L, D et U sont calculés en utilisant la formule (11) pour $i, j = 1, ..., N$

$$C_{ij} = \begin{cases} A_{ij}, & i = 1 \quad \text{ou} \quad j = 1, \\ A_{ij} - \sum_{k=1, L_{ik} \neq 0, U_{ik} \neq 0}^{\min(i,j)-1} L_{ik} D_{ik}^{-1} U_{kj}, & i > 1 \quad \text{ou} \quad j > 1. \end{cases} \quad (11)$$

En pratique, même si la matrice A est très creuse, les facteurs L, D et U peuvent être plus denses. En particulier, le terme $L_{ik}D_{ik}^{-1}U_{kj}$ peut introduire une quantité importante d'éléments non nuls sur les facteurs. Dans ce travail, le but est d'approximer l'inverse du bloc diagonal $D_{kk}^{-1}, k = 1, ..., n$ par une matrice creuse telle que $L_{ik}D_{ik}^{-1}U_{kj}$ reste creuse. Dans ce contexte de filtrage de décomposition, notre approche consiste à approximer D_{kk}^{-1} par une matrice creuse $\bar{F}$ choisie de telle sorte que la condition de filtrage soit satisfaite.

Dans ce qui suit, nous expliquons la construction du bloc de filtrage de préconditionneur M. Nous donnons d'abord sa définition et nous expliquons en détail les raisons qui nous mènent à sa construction. Dans la section 2.2, nous discutons sur la construction de l'approximation $\bar{F}$ de l'inverse du bloc de matrices diagonales.

Définition 1. *Soit A, une matrice de taille $n \times n$. Pour $k = 1, ..., N$, soit L_k une matrice de taille $n \times p$, D_k une matrice inversible de taille $p \times p$, et U_k une matrice de taille $p \times n$. Soit M la matrice définie par*

$$M - A = \sum_{k=1}^{N} L_k D_k^{-1} U_k - \sum_{k=1}^{N} L_k F_k U_k. \quad (12)$$

Une méthode de filtrage est la construction de $(F_k)_{1,N}, k = 1, ..., N$, vérifiant la relation

$$F_k U_k t \quad = \quad D_k^{-1} U_k t, \qquad \forall k = 1, ..., N, \quad (13)$$

où t est le vecteur de filtrage.

Définition 2. *Soit t un vecteur de filtrage de taille n et soit A une matrice de taille $n \times n$ partitionnée en blocs de matrices de taille $N \times N$. Une décomposition de filtrage de A est définie par la matrice M donnée par*

$$M = \begin{pmatrix} \bar{D}_{11} & & & \\ \bar{L}_{21} & \bar{D}_{22} & & \\ \vdots & \ddots & \ddots & \\ \bar{L}_{N1} & \cdots & \bar{L}_{N,N-1} & \bar{D}_{NN} \end{pmatrix} \begin{pmatrix} \bar{D}_{11}^{-1} & & & \\ & \ddots & & \\ & & & \bar{D}_{NN}^{-1} \end{pmatrix} \begin{pmatrix} \bar{D}_{11} & \bar{U}_{12} & \cdots & \bar{U}_{1N} \\ & D_{22} & \ddots & \vdots \\ & & \ddots & \bar{U}_{N-1,N} \\ & & & \bar{U}_{N,N} \end{pmatrix}, \quad (14)$$

où $(\bar{D}_{kk})_{1,N}, k = 1, ..., N$ sont des matrices carrées inversibles de taille p. Sous une forme plus compacte, $M = (\bar{L}+\bar{D})\bar{D}^{-1}(\bar{U}+\bar{D})$ où $M, \bar{L}, \bar{D}, \bar{U}$ sont des blocs de matrices de taille $N \times N$. Soit $\bar{C} = \bar{L} + \bar{D} + \bar{U}$ et $(t_1, t_2, ..., t_N)^T$, le bloc correspondant représentant le vecteur t. Les blocs sont calculés de la manière suivante

$$\bar{C}_{ij} = \begin{cases} A_{ij}, & i = 1 \quad ou \quad j = 1 \\ A_{ij} - \sum_{k=1, \bar{L}_{ik} \neq 0, \bar{U}_{ik} \neq 0}^{min(i,j)-1} \bar{L}_{kj} \bar{F}_{ik} \bar{U}_{kj}, & i > 1 \quad ou \quad j > 1 \end{cases} \quad (15)$$

où $\bar{F}_{kj}$ est une approximation de $\bar{D}_{kk}^{-1}$ satisfaisant

$$\bar{F}_{kj}\bar{U}_{kj}t_j \quad = \quad \bar{D}_{kk}^{-1}\bar{U}_{kj}t_j, \qquad \forall k = 1, ..., \min(i,j) - 1 \text{ avec } \bar{L}_{ik} \neq 0, \bar{U}_{ik} \neq 0. \quad (16)$$

Remarque 1. *Si le vecteur $\bar{U}_{kj}t_j$ ne possède pas d'éléments nuls, une matrice $\bar{F}_{kj}$ satisfaisant la condition de l'équation (16) peut se calculer par*

$$\bar{F}_{kj} = \text{Diag}((\bar{D}_{kk}^{-1}\bar{U}_{kj}t_j)./\bar{U}_{kj}t_j). \quad (17)$$

où ./ est la division ponctuelle. Pour des matrices très creuses, la matrice $\bar{U}_{kj}$ peut contenir des lignes nulles donc le résultat de $\bar{U}_{kj}t_j$ peut être un vecteur nul.

Dans la section qui suit, nous présentons une construction de $\bar{F}_{kj}$ qui permet de résoudre ce problème.

La principale idée dans la conception du préconditionneur de la définition 2 est d'assurer que chaque bloc satisfait la condition du filtrage appropriée $M_{ij}t_j = A_{ij}t_j$ telle que la condition du filtrage globale $Mt = At$ soit satisfaite où $t = (t_1, t_2, ..., t_N)^T$. Si on note $B = M - A$ alors nous voulons nous assurer que pour chaque bloc, nous avons $B_{ij}t_j = 0$. Cette différence vient de l'approche utilisée pour les blocs tri diagonaux du système, où $B = M - A$ est une matrice diagonale. Cette matrice $B = M - A$ est formée par $(B_{ij})_{1 \leqslant i,j \leqslant N}$ avec

$$B_{ij} = \bar{C}_{ij} + \sum_{k=1, \bar{L}_{ik} \neq 0, \bar{U}_{ik} \neq 0}^{\min(i,j)-1} \bar{L}_{kj}\bar{D}_{kk}^{-1}\bar{U}_{kj} - A_{ij}. \quad (18)$$

La construction de M assure que pour chaque bloc B_{ij}, pour chaque terme $\bar{L}_{kj}\bar{D}_{kk}^{-1}\bar{U}_{kj}$ de la somme de l'équation (18), $\bar{F}_{kj}$ est choisie de telle sorte que la condition de filtrage est satisfaite. Nous avons

$$\bar{L}_{kj}\bar{D}_{kk}^{-1}\bar{U}_{kj}t_j \quad = \quad \bar{L}_{kj}\bar{F}_{kj}\bar{U}_{kj}t_j. \quad (19)$$

A partir de cette formule, on déduit $\bar{F}_{kj}$ de l'équation (19). Ceci permet d'assurer que la condition de filtrage de l'ensemble de la matrice est satisfaite. On note qu'il existe une matrice $\bar{F}_{kj}$ pour chaque bloc $\bar{U}_{kj}$ non nul qui est l'approximation du bloc diagonal dépendant sur les blocs hors diagonaux de $\bar{U}$. Nous donnons une preuve formelle du lemme 1.

Lemme 1. *Considèrons une matrice A de taille $n \times n$ et un vecteur de filtrage t de taille n. Si le préconditionneur de bloc de filtrage M défini dans la définition 1 existe alors il satisfait la propriété de filtrage $Mt = At$.*

Démonstration. Le préconditionneur M satisfait la propriété de filtrage si pour chaque bloc B_{ij} non nul, nous avons $B_{ij}t_j = 0$, où B_{ij} est de taille $p \times p$ et t_j est un vecteur de p éléments. Dans l'expression des B_{ij} de l'équation (16), on remplace l'expression de $\bar{C}_{ij}$ de l'équation (15), nous obtenons

$$B_{ij}t_j = \left[\sum_{k=1,\bar{L}_{ik}\neq 0,\bar{U}_{ik}\neq 0}^{min(i,j)-1} \bar{L}_{kj}\bar{D}_{kk}^{-1}\bar{U}_{kj} - \sum_{k=1,\bar{L}_{ik}\neq 0,\bar{U}_{ik}\neq 0}^{min(i,j)-1} \bar{L}_{kj}\bar{F}_{kj}\bar{U}_{kj} \right] t_j \tag{20}$$

$$= \left[\sum_{k=1,\bar{L}_{ik}\neq 0,\bar{U}_{ik}\neq 0}^{min(i,j)-1} \bar{L}_{kj}\bar{D}_{kk}^{-1}(I - \bar{D}_{kk}\bar{F}_{kj})\bar{U}_{kj} \right] t_j = 0 \tag{21}$$

□

2.2 Construction de l'approximation

Nous décrivons la construction de l'approximation $\bar{F}_{kj}$. On note l'élément en position (i, j) de la matrice A comme $A(i, j)$ et l'élément en position i du vecteur v comme $v(i)$.

Le préconditionneur de filtrage par bloc défini à la définition 2 exige la construction de la matrice $\bar{F}_{kj}$ satisfaisant l'équation (16), c'est à dire

$$\bar{D}_{kk}^{-1}\bar{U}_{kj}t_j = \bar{F}_{kj}\bar{U}_{kj}t_j. \tag{22}$$

Pour simplifier, on note dans ce qui suit, $\bar{U}_{kj}t_j = v_{kj}$ et $\bar{D}_{kk}^{-1}\bar{U}_{kj}t_j = u_{kj}$ où v_{kj} et u_{kj} sont des vecteurs de taille p. Par la suite nous avons $\bar{F}_{kj}v_{kj} = u_{kj}$.

Dans ce qui suit pour faciliter la compréhension, nous simplifions la notation et discutons sur la relation $\bar{F}_{kj}v = u$, avec $v = v_{kj}$ et $u = u_{kj}$.

Si nous souhaitons que $\bar{F}_{kj}$ soit proche de la diagonale, nous la construisons de la manière suivante avec une tolérance ϵ de l'ordre de 10^{-7}.

$$\bar{F}_{kj} = \begin{cases} \frac{u(i)}{v(i)} & \text{si} \quad i = j \quad \text{et} \quad v(i) > \epsilon \\ \frac{u(i)}{v(i)} & \text{si} \quad v(i) < \epsilon \quad \text{et} \quad j = \text{argmax}(v) \\ 0 & \text{sinon} \end{cases} \tag{23}$$

où argmax(v) est l'argument du maximum de v.

En effet, pour une matrice creuse, le vecteur v peut posséder des éléments nuls ou quasi nuls. Nous utilisons ainsi ϵ comme tolérance pour éviter de diviser par des trop petites valeurs de v.

Si $||v||_\infty \leqslant \epsilon$, nous approximons v avec un vecteur nul. Dans ce cas, u est aussi nul et la relation $\bar{F}v = u$ est satisfaite.

Nous discutons dans ce qui suit le cas lorsque il y a moins d'éléments dans v ayant une grande valeur. Soit j l'indice de la plus grande valeur des éléments de v. Si $|v(i)| < \epsilon$, on prend $\bar{F}(i,j) = u(i)/v(j)$.

Un exemple de construction de $\bar{F}$ avec $j = 5$ est donné par

$$\begin{pmatrix} 0 & & & u(1)/v(5) & & \\ & u(2)/v(2) & & & & \\ & & u(3)/v(3) & & & \\ & & & u(4)/v(5) & & \\ & & & u(5)/v(5) & & \\ & & & u(6)/v(5) & 0 & \end{pmatrix} \begin{pmatrix} 0 \\ v(2) \\ v(3) \\ \frac{\epsilon}{2} \\ v(5) \\ \epsilon \end{pmatrix} = \begin{pmatrix} u(1) \\ u(2) \\ u(3) \\ u(4) \\ u(5) \\ u(6) \end{pmatrix} \tag{24}$$

2.3 La pertinence du BFD pour le parallélisme

Le bloc de préconditionneur de filtrage vu ci-dessus, est défini dans le cas général. Avec une passation de commande convenable, un préconditionneur parallèle peut être obtenu. Sur ce projet, on se focalise sur des matrices partitionnées utilisant la dissection emboitée. Le partitionnement mène à des algorithmes qui peuvent être implémentés en parallèle. On fait une bièvre description de la passation de commande.

La dissection emboitée considère un graphe G non orienté d'une matrice A symétrique. Elle identifie un séparateur S qui partitionne le graphe G en deux graphes G_1 et G_2 discontinus. La matrice A est permutée, les sommets correspondant au séparateur S sont ordonnés après les sommets correspondant aux graphes G_1 et G_2. Le même partionnement est appliqué sur les deux graphes discontinus. On procède ainsi de manière récursive sur les graphes obtenus jusqu'à ce que le nombre de parties indépendantes désirées après dissection soit atteint.

Considèrons une matrice A de taille $n \times n$ partionnée utilisant la dissection emboitée dans un bloc de matrices de taille $N \times N$. L'exemple qui suit montre le résultat obtenu après l'application de deux pas de la dissection emboitée qui donne un bloc de matrices de taille 7×7 où P est une matrice de permutation.

$$PAP^T = \begin{pmatrix} A_{11} & & A_{13} & & & & A_{17} \\ & A_{22} & A_{23} & & & & A_{27} \\ A_{31} & A_{32} & A_{33} & & & & A_{37} \\ & & & A_{44} & & A_{46} & A_{47} \\ & & & & A_{55} & A_{56} & A_{57} \\ & & & A_{64} & A_{65} & A_{66} & A_{67} \\ A_{71} & A_{72} & A_{73} & A_{74} & A_{75} & A_{76} & A_{77} \end{pmatrix}. \tag{25}$$

Le préconditionneur $M = (\bar{L}+\bar{D})\bar{D}^{-1}(\bar{U}+\bar{D})$ associé à PAP^T est obtenu par le calcul de

$$\bar{C} = \bar{L}+\bar{D}+\bar{U} = \begin{pmatrix} \bar{D}_{11} & & \bar{U}_{13} & & & & \bar{U}_{17} \\ & \bar{D}_{22} & \bar{U}_{23} & & & & \bar{U}_{27} \\ \bar{L}_{31} & \bar{L}_{32} & \bar{D}_{33} & & & & \bar{U}_{37} \\ & & & \bar{D}_{44} & & \bar{U}_{46} & \bar{U}_{47} \\ & & & & \bar{D}_{55} & \bar{U}_{56} & \bar{U}_{57} \\ & & & \bar{L}_{64} & \bar{L}_{65} & \bar{D}_{66} & \bar{U}_{67} \\ \bar{L}_{71} & \bar{L}_{72} & \bar{L}_{73} & \bar{L}_{74} & \bar{L}_{75} & \bar{L}_{76} & \bar{D}_{77} \end{pmatrix}, \tag{26}$$

où chaque bloc des facteurs $\bar{L}, \bar{D}$ et $\bar{U}$ peut être calculé suivant la définition 2. La matrice $\bar{C}$ préserve la structure de la dissection emboitée de PAP^T. Ainsi avec cette partition, le pré conditionneur et le processus itératif peuvent être implémentés en parallèle.

2.4 Résultats numériques

Pour valider notre approche de préconditionneur, on considère différents critères :
- Le nombre d'itérations obtenu par la méthode de GMRES de Matlab
- La précision mesurant l'erreur relative de la solution x_{gmres} de GMRES comparée à la solution exacte x_{exacte} définie par $x_{\text{exacte}} = \text{random}$.

$$Err \quad = \quad \frac{||x_{\text{exacte}} - x_{\text{gmres}}||_2}{||x_{\text{exacte}}||_2} \tag{27}$$

Considèrons le système suivant

$$\begin{cases} \eta(x) + div(a(x)u) - div(k(x)\nabla u) = f & \text{dans } \Omega \\ u = 0 & \text{sur } \partial\Omega_D \\ \frac{\partial u}{\partial n} = 0 & \text{sur } \partial\Omega_N \end{cases}, \tag{28}$$

où $\Omega = [0,1]^m (m = 2 \text{ en } 2D \text{ et } 3 \text{ en } 3D), \partial\Omega_N = \partial\Omega/\partial\Omega_D$ et $\partial\Omega_D = [0.1]\times[0.1]$ en $2D$ et $[0,1]\times[0,1]\times[0,1]$ en $3D$.

Nous étudions le cas suivant

$$\begin{cases} \eta(x) = 0, \\ a(x) = 2\pi(x_2 - 0.5, x_1 - 0.5)^T \\ k(x) = 0 \end{cases} \tag{29}$$

On obtient les résultats suivants

	ILUO		BFD	
Ordre	Nombre d'itérations Itér	Erreur	Nombre d'itérations	Erreur
D.E.16	400	7.4E-5	47	3.9E-8
D.E.32	400	7.2E-4	46	5.4E-8
D.E.64	400	7.4E-4	50	3.1E-8

TABLE 1 – Comparaison de ILUO et BFD dans le cas homogène

où D.E est la dissection emboitée réordonnée avec 16, 32 et 64 parties indépendantes.

Le cas d'un problème non homogène avec un saut important sur les coefficients est donné dans l'annexe A.

On note que le nombre d'itérations dans le cas de la méthode BFD est largement inférieur à celui ILUO, on peut dire que le conditionnement dans le cas de la méthode de BFD est inférieur à celui d' ILUO.

En terme d'évolutivité parallèle, on note que le nombre d'itérations de GMRES par BFD n'augmente pas avec le nombre de parties indépendantes de la dissection emboitée. Ceci nous permet d'affirmer qu'une implémentation en parallèle de notre méthode est possible.

2.5 L'impact du BFD sur le conditionnement

Nous discutons de l'impact du BFD sur le conditionnement de la matrice de notre système qui correspond au rapport entre la plus grande et plus petite valeur propre de la matrice en module. Les matrices sont réordonnées en utilisant la dissection emboitée avec 16 parties indépendantes. Le BFD préconditionneur a pour but principal d'éliminer les valeurs propres trop petites de la matrice. La plus petite valeur propre de la matrice préconditionnée avec BFD est toujours plus grande que la plus petite valeur propre de la matrice originale et de la matrice préconditionnée utilisant la méthode de la décomposition incomplète ILU.

Nous observons que le BFD produit en général un meilleur conditionnement de matrice que ILU. Cependant, il existe des cas particuliers où le conditionnement donné par BFD n'est pas meilleur.

3 Mise à l'échelle d'une matrice

L'équilibrage ou la mise en échelle d'une matrice consiste à multiplier la matrice originale à gauche et à droite par deux matrices diagonales pour avoir un équilibrage des éléments de la matrice.

Considèrons le problème suivant : étant donné une matrice $A \in \mathbb{R}^{n\times n}$, trouver deux matrices diagonales R et C telles que la matrice équilibrée $\overline{A}$

$$\overline{A} = R \times A \times C, \tag{30}$$

avec $C \in \mathbb{R}^{n\times n}$ et $R \in \mathbb{R}^{n\times n}$ est un meilleur conditionnement que la matrice A.

Les matrices A et $\overline{A}$ ont même ordre de grandeur dans certaines normes telles que la norme infinie, la norme euclidienne et la norme L^2 [12]. La norme infinie $||x||_\infty$ est utilisée pour trouver les matrices R et C.

Dans cette section, nous allons parler des différentes méthodes de mise à l'échelle développées et ensuite faire leur comparaison en terme de nombre d'itérations, d'erreur relative et de conditionnement.

3.1 Présentation des différentes méthodes de mise à l'échelle (scaling) utilisées

Ici, nous allons développer quatre méthodes de mise à l'échelle de matrice.

3.1.1 Scaling global (SG)

Soit A une matrice carrée de taille $n\times$n.

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & & \vdots \\ a_{n-1,1} & a_{n-1,2} & \cdots & a_{n-1,n} \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \tag{31}$$

Dans chaque ligne de A, on cherche le maximum des éléments en valeur absolue, on calcule son inverse et on le stocke dans la diagonale de R.

$$\forall i \in [1,n], \quad \mathrm{Diag}(R)_i = 1/\max(|a_{i,j}|)_{j=1,\dots,n}, \tag{32}$$

et dans chaque colonne de A, on cherche le maximum des éléments en valeur absolue, on calcule son inverse et on le stocke dans la diagonale de C.

$$\forall j \in [1,n], \quad \mathrm{Diag}(C)_j = 1/\max(|a_{i,j}|)_{i=1,\dots,n}. \tag{33}$$

Pour ne pas diviser par zéro ou quand le maximum et le minimum sont quasi identiques dans ces deux cas, on remplace les matrices diagonales R et C par l'identité.

Il est important de constater que les éléments des matrices R et C sont nuls en dehors des diagonales, donc on peut seulement stocker lors de l'implémentation leurs diagonales pour économiser de la mémoire.

La structure de l'algorithme est donnée dans l'algorithme 1.

Algorithm 1 Scaling Global

Entree :
A : Une matrice de taille $n \times n$
Sorties :
R : Une matrice diagonale de taille $n \times n$
C : Une matrice diagonale de taille $n \times n$
$\overline{A}$: Une matrice de taille $n \times n$ vérifiant $\overline{A} = R \times A \times C$
Description :

1: On fixe un seuil
2: On calcule l'inverse du maximum sur les lignes et on obtient R
3: Test si le rapport du minimum et maximum de R est grand par rapport au seuil
4: Si oui, R est une matrice d'identité de taille $n \times n$
5: Calcul du produit $\overline{A} = R \times A$
6: Calcul de l'inverse du maximum sur les colonnes de $\overline{A}$ donnant C
7: Test si le rapport du minimum et maximum de C est grand par rapport au seuil
8: Si oui, C est une matrice d'identité de taille $n \times n$
9: La matrice équilibrée est donnée par $\overline{A} := \overline{A} \times C$

3.1.2 Scaling système utilisant les blocs diagonaux (SSD)

Dans cette partie, on subdivise la matrice A en blocs de matrices de tailles égales.

Soit A, une matrice de taille $n \times n$ représentée ci-dessous.

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & & \vdots \\ a_{n-1,1} & a_{n-1,2} & \cdots & a_{n-1,n} \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \tag{34}$$

La matrice A peut être représentée comme ceci

$$A = \begin{pmatrix} a_{1,1} \cdots a_{1,p} & \cdots & a_{1,j} \cdots & a_{1,p} & \cdots & a_{1,n-p} \cdots a_{1,n} \\ a_{p,1} \cdots a_{p,p} & \cdots & a_{p,j} \cdots & a_{p,p} & \cdots & a_{p,n-p} \cdots a_{p,n} \\ \vdots & & \ddots & & & \vdots \\ a_{n-p,1} \cdots a_{n-p,p} & \cdots & a_{n-p,j} \cdots & a_{n-p,p} & \cdots & a_{n-p,n-p} \cdots a_{n-p,n} \\ a_{n,1} \cdots a_{n,p} & \cdots & a_{n,j} \cdots & a_{n,p} & \cdots & a_{n,n-p} \cdots a_{n,n} \end{pmatrix}, \tag{35}$$

c'est à dire

$$A = (A_{ij})_{1,N}, \text{ avec } A_{ij} \in \mathbb{R}^{p \times p}. \tag{36}$$

Avant de faire la mise en échelle, on effectue une permutation de A, nommée $\tilde{A}$ donnée par

$$(\tilde{A}_{ij})_{kl} = (A_{kl})_{ij}, \text{ avec } \tilde{A}_{ij} \in \mathbb{R}^{N \times N} \tag{37}$$

Ainsi, on fait l'équilibrage de $\tilde{A}$ appliquant le même principe que précédemment mais sur les blocs diagonaux de la matrice (37).

Si on note $C_1, .., C_N$, les inverses des maximums sur les lignes respectives des matrices $\tilde{A}_{1,1}, ..., \tilde{A}_{N,N}$ et $R_1, ..., R_N$ les inverses des maximums sur les colonnes des matrices $\tilde{A}_{1,1}, ..., \tilde{A}_{N,N}$, nous obtenons

$$C = \begin{pmatrix} C_1 & & & \\ & C_2 & & \\ & & \ddots & \\ & & & C_N \end{pmatrix}, \text{ et } R = \begin{pmatrix} R_1 & & & \\ & R_2 & & \\ & & \ddots & \\ & & & R_N \end{pmatrix}, \tag{38}$$

et donc, la matrice $\overline{A}$ de mise à l'échelle est donnée par

$$\overline{A} = R \times \tilde{A} \times C. \tag{39}$$

Après la mise à l'échelle, il est nécessaire de faire une permutation inverse pour se ramener aux positions initiales des éléments de la matrice A.

La description de l'algorithme est donnée dans l'algorithme 2.

Algorithm 2 Scaling Système Bloc Diagonal

Entrees :
A : Une matrice de taille $n \times n$
N : Le nombre d'équations du système considéré
Sorties :
R : Une matrice diagonale de taille $n \times n$
C : Une matrice diagonale de taille $n \times n$
$\overline{A}$: Une matrice de taille $n \times n$ telle que $\overline{A} = R \times A \times C$
Description :

1: On fixe un seuil
2: On subdivise la matrice A en blocs de matrices de tailles $p \times p$
3: Calculer la permutation de A et on obtient $\tilde{A}$
4: Pour chaque i de 1 à N
5: Calculer l'inverse du maximum sur les lignes du bloc $\tilde{A}_{i,i}$ et on obtient R_i
6: Test si le rapport du minimum et maximum de R_i est grand par rapport au seuil
7: Si oui, R_i est vecteur de taille p qui contient que des 1
8: Fin pour i
9: R est la matrice diagonale dont la diagonale est composée par des vecteurs R_i
10: Calcul du produit $\overline{A} = R \times \tilde{A}$
11: Pour chaque i de 1 à N
12: Calculer l'inverse du maximum sur les colonnes du bloc $\overline{A}_{i,i}$ et on obtient C_i
13: Test si le rapport du minimum et maximum de C_i est grand par rapport au seuil
14: Si oui, C_i est vecteur de taille p qui contient que des 1
15: Fin pour i
16: C est la matrice diagonale dont la diagonale est composée par des vecteurs C_i
17: La matrice équilibrée est donnée par $\overline{A} := \overline{A} \times C$

3.1.3 Scaling système tenant compte tous les blocs de la matrice (SSB)

Dans cette partie, on met d'abord la matrice A sous la forme (37) de la section 3.1.2.

On fait l'équilibrage sur tous les blocs de la matrice. On note $\tilde{A}_{i,j}$ le bloc de la matrice de ligne i de colonne j. Pour chaque ligne i de $\tilde{A}$, on équilibre chacun des p blocs de matrice et on fait la moyenne sur les vecteurs trouvés.

Soit $R_{i,j}$, l'inverse du maximum en valeur absolue sur les lignes du bloc de la matrice $\bar{A}_{i,j}$ et R_i, la moyenne des $R_{i,j}$ pour i fixé

$$R_i = \frac{1}{p}\sum_{j=1}^{p} R_{i,j}, \qquad \forall i \text{ fixé} \tag{40}$$

et on obtient la matrice R définie par

$$R = \begin{pmatrix} R_1 & 0 & \cdots & 0 \\ 0 & R_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & R_N \end{pmatrix}. \tag{41}$$

On procède de la même manière en choisissant l'inverse du maximum en valeur absolue sur les colonnes pour trouver la matrice C

$$C = \begin{pmatrix} C_1 & 0 & \cdots & 0 \\ 0 & C_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & C_N \end{pmatrix}. \tag{42}$$

Ainsi, la matrice équilibrée $\overline{A}$ est donnée par

$$\overline{A} = \begin{pmatrix} R_1 & 0 & \cdots & 0 \\ 0 & R_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & R_N \end{pmatrix} \times \begin{pmatrix} \tilde{A}_{1,1} & \tilde{A}_{1,2} & \cdots & \tilde{A}_{1,N} \\ \tilde{A}_{2,1} & \tilde{A}_{2,2} & \cdots & \tilde{A}_{2,N} \\ \vdots & \ddots & & \vdots \\ \tilde{A}_{N,1} & \tilde{A}_{N,2} & \cdots & \tilde{A}_{N,N} \end{pmatrix} \times \begin{pmatrix} C_1 & 0 & \cdots & 0 \\ 0 & C_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & C_N \end{pmatrix}. \tag{43}$$

La description de l'algorithme est donnée par l'algorithme 3.

Algorithm 3 Scaling Système tenant compte tous les blocs (SSB)

Entrees :
A : Une matrice de taille $n \times n$
N : est le nombre d'équations du système considéré
Sorties :
R : Une matrice diagonale de taille $n \times n$
C : Une matrice diagonale de taille $n \times n$
$\overline{A}$: Une matrice de taille $n \times n$ telle que $\overline{A} = R \times A \times C$
Description :

On fixe un seuil
On subdivise la matrice A en blocs de matrices de taille $p \times p$
Calculer la permutation de A et on obtient $\tilde{A}$
Pour chaque i de 1 à N
    Calcul de l'inverse du maximum sur les lignes de $\tilde{A}_{i,j}$ pour obtenir $R_{i,j}$
    Calculer $R_i = \frac{1}{p}\sum_{j=1}^{p} R_{i,j}$
    Test si le rapport du minimum et maximum de R_i est grand par rapport au seuil
    Si oui, R_i est un vecteur de taille p qui contient que des 1
Fin pour i
R est la matrice diagonale dont la diagonale est composée par des vecteurs R_i
Calcul du produit $\overline{A} = R \times \tilde{A}$
Pour chaque i de 1 à N
    Calcul de l'inverse du maximum sur les colonnes de $\overline{A}_{i,j}$ pour obtenir $C_{i,j}$
    Calculer $C_i = \frac{1}{p}\sum_{j=1}^{p} C_{i,j}$
    Test si le rapport du minimum et maximum de C_i est grand par rapport au seuil
    Si oui, C_i est un vecteur de taille p qui contient que des 1
Fin pour i
C est la matrice diagonale dont la diagonale est composée par des vecteurs C_i
La matrice équilibrée est donnée par $\overline{A} := \overline{A} \times C$

3.1.4 Scaling système itératif (SSI)

Ici, on présente une méthode itérative qui fait une mise à l'échelle de façon asymptotique sur matrice originale en utilisant la norme infinie$||.||_\infty$ (norme utilisée dans les cas précédents). Cette stratégie d'équilibrage montre des propriétés optimales permettant de préserver la symétrie de la matrice [14].

3.1.4.a Description

Soit A une matrice réelle de taille $n \times n$. On note
- $r_i = a_{i.}^T \in \mathbb{R}^{n\times 1}, i = 1, ..., n$, les vecteurs lignes issus de A
- $c_j = a_{.j}^T \in \mathbb{R}^{n\times 1}, j = 1, ..., n$, les vecteurs colonnes issus de A
- R et C des matrices diagonales de tailles $n \times n$ données par

$$\forall i \in [1, n], \quad \text{Diag}(R)_i = 1/(\sqrt{||r_i||_\infty})_{i=1,...,n}, \tag{44}$$

$$\forall j \in [1, n], \quad \text{Diag}(C)_j = 1/(\sqrt{||c_j||_\infty})_{j=1,...,n}. \tag{45}$$

Si une ligne (ou respectivement une colonne) de la matrice originale A est nulle, on remplace la matrice diagonale R (respectivement C) par l'identité.

La matrice d'équilibrage de A est donnée par

$$\overline{A} = R \times A \times C. \tag{46}$$

La méthode itérative proposée est donnée dans l'algorithme 4 et se fait de manière récursive.

Algorithm 4 Scaling système itératif (SSI)

Entree :
A : Une matrice de taille $n \times n$
Sorties :
R : Une matrice diagonale de taille $n \times n$
C : Une matrice diagonale de taille $n \times n$
$\overline{A}$: Une matrice de taille $n \times n$ telle que $\overline{A} = R \times A \times C$
Description :
1: $D_1^{(0)} = I_n$
2: $D_2^{(0)} = I_n$
3: for $k = 0, 1, 2, ...,$ jusqu'à convergence faire
4: $\quad \text{Diag}(R)_i = 1/(\sqrt{||r_i^{(k)}||_\infty})_{i=1,...,n}$
5: $\quad \text{Diag}(C)_i = 1/(\sqrt{||c_i^{(k)}||_\infty})_{i=1,...,n}$
6: $\quad D_1^{(k+1)} = D_1^{(k)} \times R$
7: $\quad D_2^{(k+1)} = D_2^{(k)} \times C$
8: $\quad \overline{A}^k = D_1^{(k+1)} \times A \times D_2^{(k+1)}$

La convergence est obtenue lorsque

$$\max_{1 \leqslant i \leqslant m} \{|(1 - ||r_i^{(k)}||_\infty)|\} \leqslant \epsilon \text{ et } \max_{1 \leqslant j \leqslant n} \{|(1 - ||c_i^{(k)}||_\infty)|\} \leqslant \epsilon \tag{47}$$

pour une valeur $\epsilon > 0$ donnée. Les propriétés de cet algorithme 4 ainsi que le taux de convergence sont discutés dans les paragraphes qui suivent.

3.1.4.b Convergence linéaire et propriétés

La première propriété de cet algorithme est qu'il préserve la symétrie de la matrice. C'est l'une des principales raisons pour la construction de cette méthode itérative. En effet, si la matrice A est symétrique, les matrices diagonales R et C le sont aussi et donc par conséquent la matrice $\overline{A}$ de (46) reste symétrique comme c'est le cas pour les matrices $\overline{A}^k$ à n'importe quelle itération dans l'algorithme 4. Cette propriété n'est pas toujours vérifiée pour beaucoup de méthodes d'équilibrage qui font une mise en échelle d'abord sur les lignes puis sur les colonnes ou vice-versa.

Le cas particulier où la matrice A possède des lignes et colonnes avec la norme infinie égale à 1 est fixé dans les itérations de l'algorithme. Si A est une matrice carrée dans laquelle la

valeur absolue de chaque élément de la diagonale est plus importante ou égale à la valeur absolue de tout autre élément correspondant à une ligne ou colonne, alors il est facile de voir que l'algorithme converge en une seule itération, avec une matrice résultante $\overline{A}^{(1)}$ possédant que des 1 sur la diagonale.

Concernant le taux de convergence, dans le cas général, nous montrons que l'algorithme converge avec un taux linéaire asymptotique égale à $\frac{1}{2}$.

Dans la démonstration, le premier point est de noter qu'après la première itération de l'algorithme, tous les éléments de la matrice $\overline{A}^{(1)}$ sont inférieurs ou égaux à 1 en valeur absolue. Il est facile de voire que tous les éléments a_{ij} de A sont divisés par les racines carrées des deux nombres $||r_i^{(k)}||_\infty$ et $||c_i^{(k)}||_\infty$ qui sont supérieures ou égaux à $|a_{ij}|$.

Ainsi, pour chaque itération ($k \geqslant 1$), on considère la norme infinie du vecteur ligne $r_i^{(k)}$ ou colonne $c_i^{(k)}$. Si on note $a_{ij_0}^{(k)}$ l'élément sur la ligne i pour la quelle l'égalité $a_{ij_0}^{(k)} = r_i^{(k)}$ est vérifiée et $a_{i_0j}^{(k)}$ l'élément sur la colonne j pour la quelle l'égalité $a_{i_0j}^{(k)} = r_i^{(k)}$ est vérifiée, on peut facilement vérifier que les éléments $a_{ij_0}^{(k+1)}$ et $a_{i_0j}^{(k+1)}$ de la matrice équilibrée $\overline{A}^{(k+1)}$ sont supérieurs en valeur absolue à la valeur de la racine carrée correspondant à l'itération k et restent inférieurs à 1. En effet, on peut écrire

$$1 \geqslant |a_{ij_0}^{(k+1)}| = \frac{|a_{ij_0}^{(k)}|}{\sqrt{||r_i^{(k)}||_\infty}\sqrt{||c_i^{(k)}||_\infty}} = \frac{\sqrt{|a_{ij_0}^{(k)}|}}{\sqrt{||c_i^{(k)}||_\infty}} \geqslant \sqrt{|a_{ij_0}^{(k)}|} \tag{48}$$

pour $|a_{ij_0}^{(k)}| = ||r_i^{(k)}||_\infty$ et $||c_i^{(k)}||_\infty \leqslant 1$ pour tout $k \geqslant 1$.

Il est facile de montrer que

$$\sqrt{|a_{ij_0}^{(k)}|} \leqslant |a_{i_0j}^{(k+1)}| \leqslant 1 \qquad \forall k \geqslant 1. \tag{49}$$

Ainsi, nous pouvons écrire finalement que les itérations de l'algorithme donnent des matrices équilibrées $\overline{A}^{(k)}, k = 1, 2, ...$ avec les propriétés suivantes

$$\forall k \geqslant 1, \quad 1 \leqslant i \leqslant n, \qquad \sqrt{||r_i^{(k)}||_\infty} \leqslant |a_{ij_0}^{(k+1)}| \leqslant \sqrt{||r_i^{(k+1)}||_\infty} \leqslant 1 \tag{50}$$

et

$$\forall k \geqslant 1, \quad 1 \leqslant j \leqslant n, \qquad \sqrt{||c_j^{(k)}||_\infty} \leqslant |a_{i_0j}^{(k+1)}| \leqslant \sqrt{||c_j^{(k+1)}||_\infty} \leqslant 1 \tag{51}$$

Les inéquations (50) et (51) montrent que les deux normes doivent converger vers 1.

Pour conclure la démonstration, nous avons juste besoin de vérifier que

$$1 - ||r_i^{(k+1)}||_\infty \quad = \quad \frac{1 - ||r_i^{(k+1)}||_\infty^2}{1 + ||r_i^{(k+1)}||_\infty} \quad \leqslant \quad \frac{1 - ||r_i^{(k)}||_\infty}{1 + ||r_i^{(k+1)}||_\infty} \tag{52}$$

Ces résultats restent valables pour la norme infinie sur les colonnes, $||c_i^{(k+1)}||_\infty$ qui respecte la preuve de la convergence linéaire de l'algorithme (4) avec un taux asymptotique égale à $\frac{1}{2}$ [15].

3.2 Comparaison des différentes méthodes de scaling

3.2.1 Comparaison de SG et SSD sur une matrice *Couplex Gaz*

Nous discutons sur l'efficacité des méthodes de SG et SSD en comparant les nombres d'itérations, les erreurs relatives, les temps totaux et les conditionnements dans les cas de ILUO et BFD.

Considérons la matrice *Couplex Gaz* de taille 52876×52876 avec 1420316 éléments non nuls. Cette matrice est issue d'un problème 2D d'écoulements diphasiques en milieu poreux pour le problème de stockage des déchets radioactifs. Elle vient d'un benchmark proposé par l'ANDRA en 2006.

En notant respectivement Nparts, Cdt, NbrIter, Tps et Erreur, le nombre de parties indépendantes désirées après dissection, le conditionnement de la matrice, le nombre d'iterations pour effectuer les calculs, le temps total du calcul exprimé en seconde et la norme de l'erreur relative entre la solution calculée et la solution exacte, le tableau 2 récapitule les réultats pour ces deux méthodes où SS est la matrice sans mise à l'échelle.

Nparts	16				32		
	Cdt	NbrIter	Tps(s)	Erreur	NbrIter	Tps(s)	Erreur
ILUO							
SG	1.14e+04	56	3	3.5e-08	51	12	1.6e-09
SSD	1.12e+06	56	3	2.54e-06	56	3.2	1.86e-06
SS	1.05e+09	59	0.4	2.21e-06	78	20	3.94e-07
BFD							
SG	1.14e+04	42	8.1	1.65e-09	51	12	1.60e-09
SSD	1.12e+06	67	12	1.6e-07	69	16	1.41e-07
SS	1.05e+09	65	14	7.11e-08	78	20	3.94e-07

TABLE 2 – Comparaison de SG et SSD sur la matrice *Couplex Gaz*

Le tableau 2 montre que dans les deux méthodes (SG et SSD), nous avons des erreurs petites (environ 10^{-7}) donc, on peut s'assurer une convergence de nos méthodes sur cette application.

Pour mieux analyser le tableau 2, on représente les résultats de ce tableau sous forme de histogrammes.

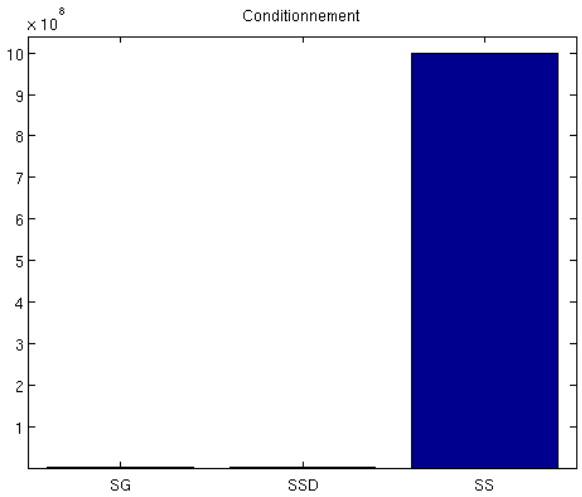


FIGURE 1 – Conditionnement de SG et SSD sur la matrice*Couplex Gaz*

En regardant la figure 1, il est clair que le conditionnement avec mise à l'échelle de matrice est bien meilleur que sans mise à l'échelle où on obtient un conditionnement égal à environ à 10^9.

Pour les deux méthodes de mise à l'échelle, le scaling global est un peu meilleur que celle scaling système utilisant les blocs diagonaux.

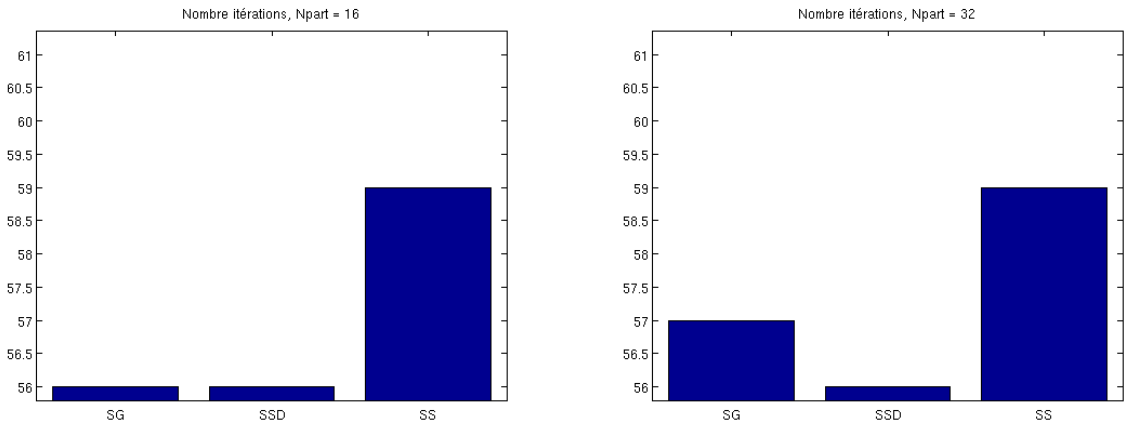


FIGURE 2 – Nombre d'itérations de SG et SSD dans le cas ILUO

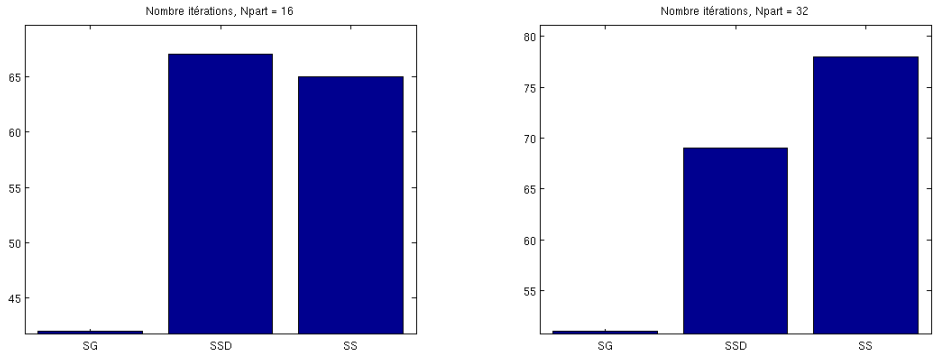


FIGURE 3 – Nombres d'itérations de SG et SSD dans le cas BFD

Vu les résultats des deux figures 2 et 3 sur le nombre d'itérations, on peut dire que la méthode SG coûte moins chére que celle de SSD. Cependant, dans le cas de la méthode ILUO les deux méthodes sont quasiment identiques en nombre d'itérations.

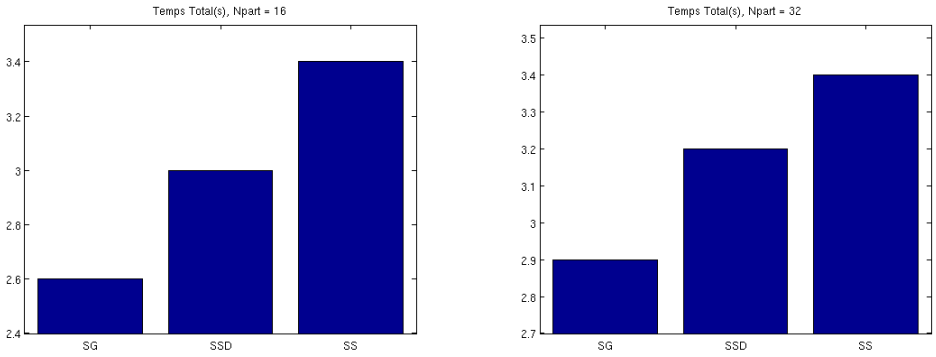


FIGURE 4 – Temps total de SG et SSD dans le cas ILUO

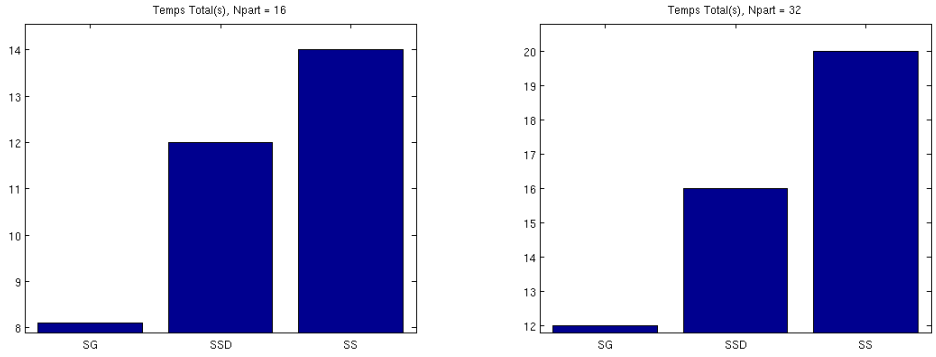


FIGURE 5 – Temps total de SG et SSD dans le cas BFD

Au niveau du temps total, c'est le même scénario que le nombre d'itérations ce qui peut être jugé normal.

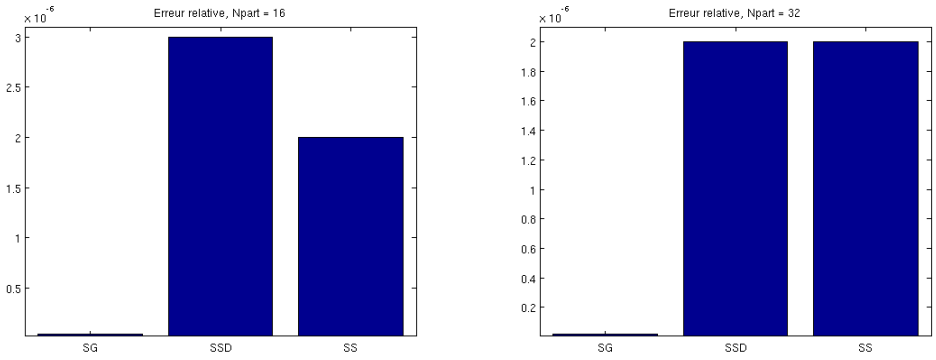


FIGURE 6 – Erreur de SG et SSD dans le cas ILUO

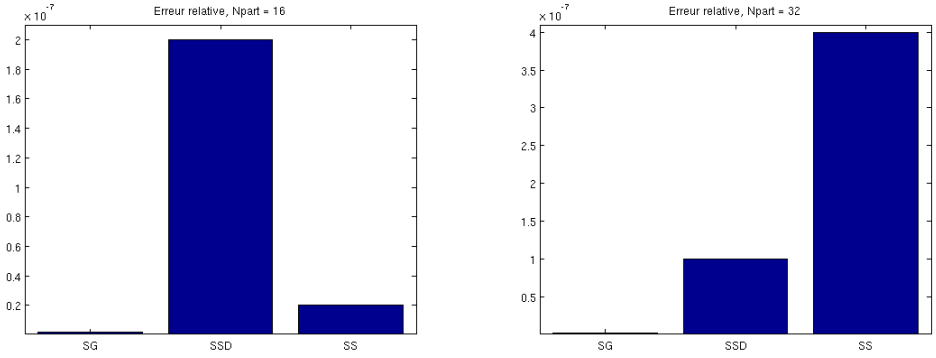


FIGURE 7 – Erreur de SG et SSD dans le cas BFD

En comparant les erreurs, on peut dire que l'erreur dans le cas de SG est négligeable devant l'erreur dans le cas SSD et celle de sans scaling matrice.

On peut dire d'une manière générale que la méthode de SG est meilleure que celle de SSD en conditionnement, nombre d'itérations, du temps du calcul et d'erreur. Pour le cas d'autres exemples de matrice, voir annexe B. Cette conclusion nous a amené à penser une nouvelle méthode de mise à l'échelle de matrice, le scaling système tenant compte tous les blocs diagonaux de la matrice.

3.2.2 Comparaison de SG, SSD et SSB sur 2 matrices CEA (*Couplex Gaz* et calcul de thermo-hydraulique)

Ici, nous allons comparer trois méthodes d'équilibrage de matrice à savoir la SG, SSD et la SSB. Nous allons tester par deux cas de matrices. Nous utilisons les deux méthodes de préconditionneurs en l'occurrence les méthodes ILUO et BFD.

3.2.2.a Application 1 : Calcul de thermo-hydraulique

On prend ici une matrice A de taille 270×270. Cette matrice est assez irrégulière. Les résultats obtenus sont affichés sur le tableau 3.

Nparts	16				32		
	Cdt	NbrIter	Tps(s)	Erreur	NbrIter	Tps(s)	Erreur
ILUO							
SG	1.58e+05	91	0.28	6.85e-11	91	0.14	3.42e-11
SSD	8.03e+16	37	0.10	909.07	38	0.03	594.882
SSB	3.20e+17	37	0.28	606.05	38	0.04	396.59
SS	4.87e+17	42	0.12	486.86	44	2.3	378.14
BFD							
SG	1.58e+05	32	0.93	7.47e-12	32	2.4	3.56e-12
SSD	8.03e+16	47	0.86	4.20e+07	48	2.4	2.20e+08
SSB	3.20e+17	28	0.84	1001.53	26	2.2	498.04
SS	4.87e+17	50	0.94	2.30e+06	42	2.3	1.42e+06

TABLE 3 – Comparaison de SG, SSD et SSB sur la matrice Thermo-hydraulique

La méthode de Scaling global est largement meilleure en terme de conditionnement que les autres méthodes de mise à l'échelle. On observe que les deux méthodes de scaling système (SSD et SSB) ont des conditionnements un peu similaires au cas où on ne fait pas de mise à l'échelle de la matrice.

Sur cette application, il y a une explosion des erreurs des deux méthodes de scaling système (SSD et SSB) et l'erreur sans faire une mise à l'échelle de la matrice. Donc, il n'est pas judicieux de parler de nombre d'itérations ou de temps de calcul avec ces méthodes car on ne trouve pas la solution exacte.

Il est à noter qu'on trouve dans le cas de la méthode SG, des erreurs petites et donc des bonnes solutions obtenues.

Dans ce paragraphe, on peut dire qu'avec la méthode de scaling système tenant compte de tous les blocs de la matrice (SSB), on a fait moins d'itérations dans les deux cas de méthodes de préconditionneurs (ILUO et BFD). Cependant, les erreurs trouvées avec cette méthode sont très grandes donc elle ne donne pas de bonnes solutions.

La méthode de mise à l'échelle globale (SG) donne un meilleur conditionnement et des erreurs très petites et donc une meilleur méthode de notre application.
Avec cette méthode, les nombres d'itérations sont plus petits dans le cas de la méthode de BFD que dans celle d'ILUO.

Remarque 2. *Si on ne fait pas de mise à l'échelle de la matrice (SS), on fait plus d'itérations dans le cas du BFD que dans le cas de la méthode ILUO et l'erreur et le conditionnement sont très grands.*

3.2.2.b Application 2 : *Couplex Gaz*

Considérons comme dans le cas de la section 3.4 la matrice *Couplex Gaz* de taille 52876 × 52876 avec 1420316 éléments non nuls. Les résultats sont présentés sur le tableau 4.

Nparts	16				32		
	Cdt	NbrIter	Tps(s)	Erreur	NbrIter	Tps(s)	Erreur
ILUO							
SG	1.14e+04	56	3	3.5e-08	51	12	1.6e-09
SSD	1.12e+06	56	3	2.54e-06	56	3.2	1.86e-07
SSB	1.91e+06	56	3	3.70e-06	56	4.6	1.89e-09
SS	1.05e+09	59	0.4	2.21e-06	78	20	3.94e-07
BFD							
SG	1.14e+04	42	8.1	1.65e-09	51	12	1.60e-09
SSD	1.12e+06	67	12	1.60e-07	69	16	1.41e-07
SSB	1.12e+06	18	4.8	1.59e-07	23	7.4	1.82e-07
SS	1.05e+09	65	14	7.11e-08	78	20	3.94e-07

TABLE 4 – Comparaison de SG, SSD et SSB sur la matrice *Couplex Gaz*

Le tableau 4 montre que le conditionnement est un peu meilleur dans la méthode de SG que dans les deux méthodes de scaling système (SSD et SSB).

Dans le cas de la méthode d'ILUO, les nombres d'itérations sont quasiment identiques sur les trois méthodes. Les erreurs sont plus petites dans la méthode de scaling global comparé aux autres méthodes qui restent faibles (2×10^{-6}) environ.

Dans le cas de la méthode de BFD, nous avons construit les figures 8 et 9 pour mieux voir les résultats.

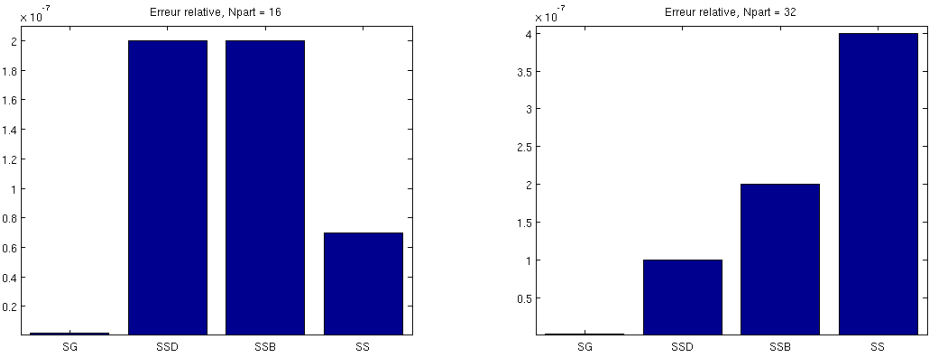


FIGURE 8 – Erreur de SG, SSD et SSB dans le cas BFD

Comme dans le cas de la méthode ILUO, on trouve que l'erreur commise dans la méthode de SG est plus faible que celles trouvées dans les deux autres méthodes. Cependant, il est à signaler que les erreurs des méthodes de scaling système restent aussi faibles (10^{-7}).

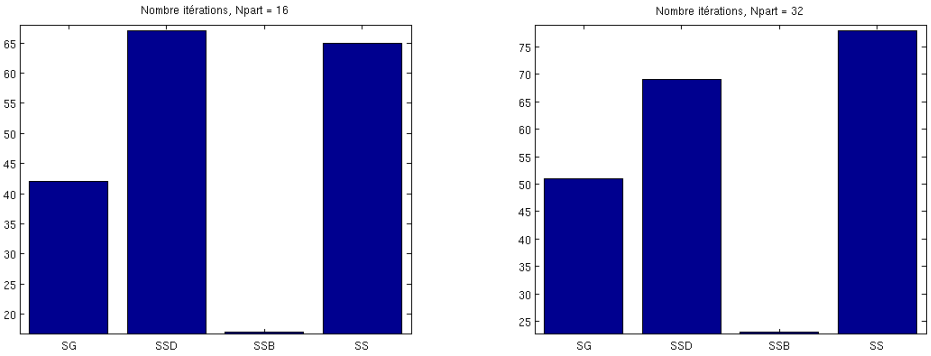


FIGURE 9 – Nombre d'itérations de SG, SSD et SSB dans le cas BFD

En observant la figure 9 ci-dessus, on voit que la méthode de SSB est meilleure en terme de nombre d'itérations que les deux autres méthodes. On gagne plus de la moitié dans le cas de la méthode de scaling global et beaucoup plus (plus de 3 fois) par rapport à la méthode de scaling système utilisant les blocs diagonaux.

Les temps totaux évoluent dans le même sens que les nombres d'itérations. C'est à dire plus le nombre d'itérations est important, plus le temps total augmente pour chacune de ces méthodes.

Nous pouvons dire sur cette application que la méthode de SSB est meilleure que les méthodes de SG et de SSD en terme d'itérations et du temps de calcul. Néanmoins l'erreur commise dans le cas du SG est moins importante que celle commise dans le cas de SSB même si cette dernière reste petite (environ 10^{-7}).

Au vue des deux applications ci-dessus, nous pouvons dire que la méthode de scaling système tenant compte de tous les blocs de la matrice est meilleure en terme d'itérations. Cependant, dans le cas de l'application de la matrice de thermo-hydraulique, on voit que son erreur est trop grande, donc ne donne pas une bonne solution. Et donc, il est difficile de dire que cette méthode est meilleure d'une manière générale que les autres méthodes en particulier la méthode SG où on voit des erreurs très petites dans les deux applications et des conditionnements largement en dessous de ceux obtenus avec les autres méthodes de mise à l'échelle.

3.2.3 Comparaison de SG, SSD, SSB et SSI sur deux matrices issues d'un problème d'écoulements diphasiques en milieux poreux (*Couplex Gaz* et *FORGE*)

On fait une comparaison de toutes les méthodes de mise à l'échelle développées.

3.2.3.a Application 1 : *Couplex Gaz*

Comme dans les cas des comparaisons ci-dessus, on prend la matrice *Couplex* creuse de taille 52876×52876 avec 1420316 éléments non nuls.

Nparts	16				32		
	Cdt	NbrIter	Tps(s)	Erreur	NbrIter	Tps(s)	Erreur
ILUO							
SG	1.14e+04	56	3	3.50e-08	56	12	1.60e-09
SSD	1.12e+06	56	3	2.54e-06	56	3.2	1.86e-07
SSB	1.91e+06	56	3	3.70e-06	56	4.6	1.89e-09
SSI	3.02e+03	57	5	2.30e-06	57	5	9.10e-09
SS	1.05e+09	59	5	2.21e-06	78	5	3.94e-07
BFD							
SG	1.14e+04	42	8.1	1.65e-09	51	12	1.60e-09
SSD	1.12e+06	67	12	1.60e-07	69	16	1.41e-07
SSB	1.91e+06	18	4.8	1.59e-07	23	7.4	1.82e-07
SSI	3.02e+03	32	11	5,20e-09	42	16	9.01e-09
SS	1.05e+09	65	14	7.11e-08	78	20	3.94e-07

TABLE 5 – Comparaison de SG, SSD, SSB et SSI sur la matrice *Couplex Gaz*

Comme dans les cas précédents, on représente les données sous forme d'histogrammes

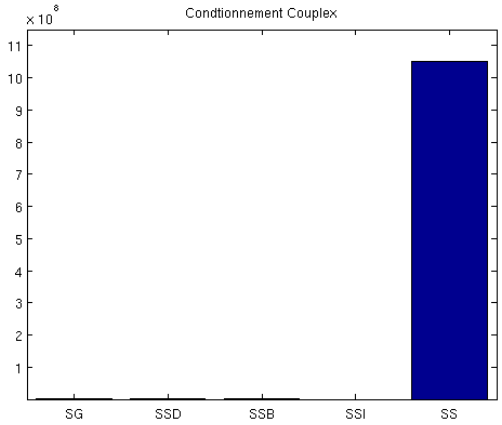


FIGURE 10 – Conditionnement de SG, SSD, SSB sur la matrice *Couplex Gaz*

Le tableau 5 révèle qu'on a un meilleur conditionnement dans le cas de la méthode de scaliong système tenant compte de tous les blocs de la matrice (SSI) qui est égal à environ 10^3. La figure 10 nous montre l'importance de faire une mise à l'échelle en terme de conditionnement. Il y a une diminution du conditionnement de l'ordre de 10^5 environ.

Préconditionneur ILUO

On voit un temps de calcul plus important dans le cas de la méthode de scaling système itératif et sans faire de mise à l'échelle de la matrice que les autres méthodes.

Concernant les erreurs, elles sont petites dans toutes les méthodes et avec sans mise à l'échelle de la matrice. Cependant, elles sont plus petites dans le cas des méthodes SG et SSI.

Dans cette méthode, les nombres d'itérations sont pratiquement identiques pour toutes les méthodes de mise à l'échelle. Il y a une augmentation de nombre d'itérations si la matrice n'est pas mise à l'échelle quand le nombre de parties de dissection emboitée augmente comme la figure 11 la montre.

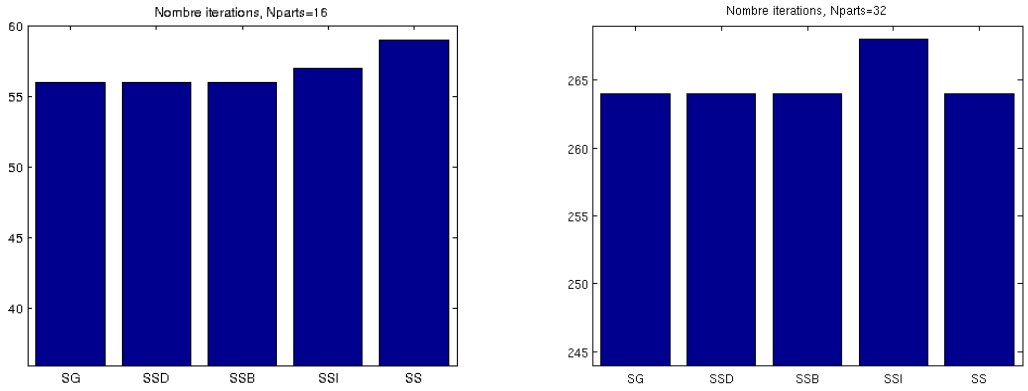


FIGURE 11 – Nombre d'itérations de SG, SSD, SSB et SSI dans le cas ILU

Préconditionneur BFD

Dans ce cas, on considère les figures 13, 12 et 14.

Les erreurs se comportent comme dans le cas de la méthode ILUO. On voit qu'elles sont plus importantes dans le cas des méthodes SSB et SSD même si elles peuvent être jugées petites (10^{-7}).

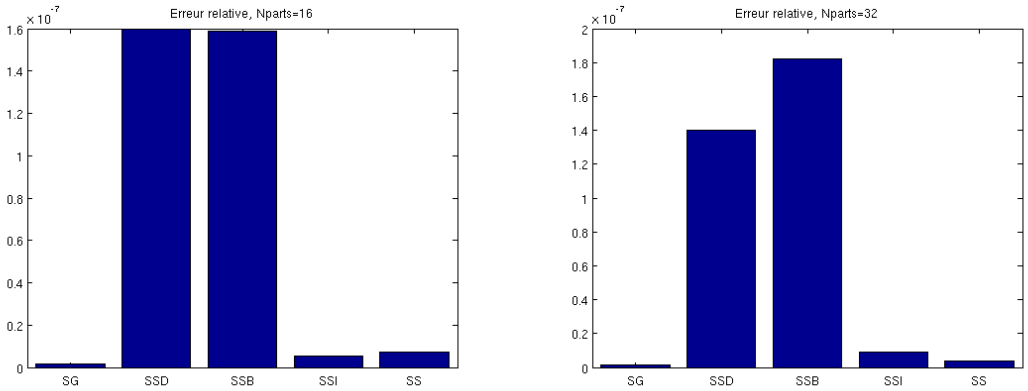


FIGURE 12 – Erreur de SG, SSD, SSB et SSI dans le cas BFD

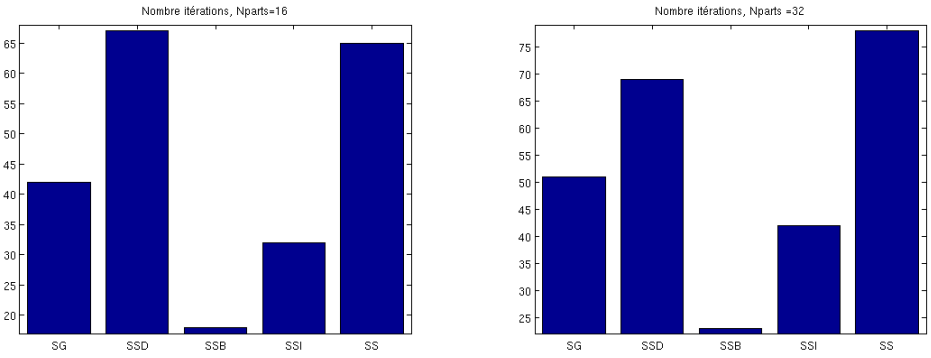


FIGURE 13 – Nombre d'itérations de SG, SSD, SSB et SSI dans le cas BFD

Les nombres d'itérations sont beaucoup moins importants avec la méthode de SSB et un peu moins important avec la méthode itérative (SSI). Si on ne fait pas d'équilibrage de matrice, le nombre d'itérations est grand et il va de même avec la méthode de scaling système utilisant les blocs diagonaux (SSD).

Cette situation peut s'expliquer entre autre par le fait qu'on a un meilleur conditionnement de la matrice et du dernier bloc de la matrice dans le cas de la cas de la méthode de scaling système itératif (SSI).

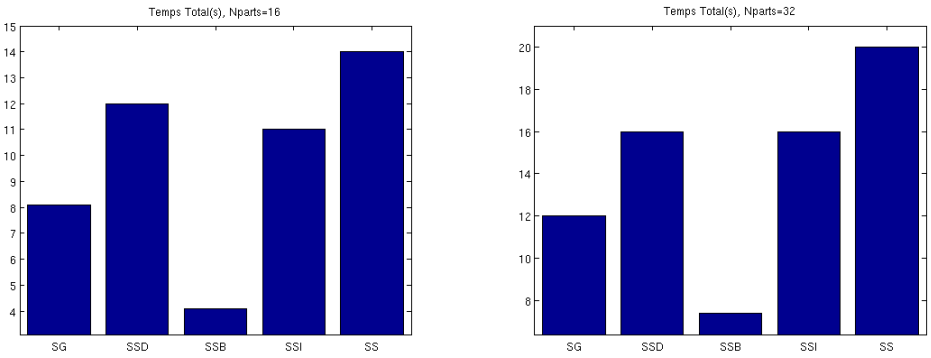


FIGURE 14 – Temps total de SG, SSD, SSB et SSI

Les temps totaux évoluent dans le même sens que les nombres d'itérations, c'est à dire plus le nombre d'itérations est grand, plus le temps est important. Cette situation peut être jugée normale.

En conclusion sur cette application, on peut affirmer que la méthode de scaling système itératif (SSI) est meilleure que les autres méthodes de mise à l'échelle de matrice. L'erreur trouvée avec cette méthode est toujours plus petite dans les deux cas de préconditionneurs (BFD et ILUO), le conditionnement est plus petit et les nombres d'itérations sont parmi les moins importants dans les deux cas de figure.

3.2.3.b Application 2 : *FORGE*

Ici, on fait l'étude d'une matrice de taille 17646 × 17646 avec 456132 éléments non nuls. Comme la matrice *Couplex Gaz*, elle est issue d'un problème 2D d'écoulements diphasiques en milieu poreux pour le problème de stockage des déchets radioactifs. Elle provient d'un benchmark défini dans le cadre du projet européen FORGE débuté en février 2009.
Les résultats obtenus sont donnés par le tableau 6.

Nparts	16				32		
	Cdt	NbrIter	Tps(s)	Erreur	NbrIter	Tps(s)	Erreur
ILUO							
SG	9e+5	240	21	5e-5	264	12	6e-5
SSD	2e+5	240	11	5e-7	264	12	5e-7
SSB	2e+7	264	27	1e+2	264	15	1e-2
SSI	3e+5	266	12	1e-6	268	12	1e-6
SS	5e+8	240	11	5e-5	264	12	6e-5
BFD							
SG	9e+5	46	4.4	5e-5	71	9.4	3e-5
SSD	2e+5	46	4	5e-7	71	9.5	3e-7
SSB	2e+7	160	28	8e-1	400	51	3e+1
SSI	3e+5	82	6.8	2e-7	119	14	2e-7
SS	5e+8	51	4.4	5e-6	81	10	2e-6

TABLE 6 – Comparaison de SG, SSD, SSB et SSI sur la matrice *FORGE*

Les erreurs sont plus petites dans le cas de scaling système itératif (environ 10^{-7}) et son explosion dans le cas de la méthode SSB où elle atteignent 30 pour le préconditionneurs BFD. Donc, on peut dire que la méthode de SSB ne fournit pas une bonne solution pour ce cas de matrice.

Préconditionneur ILUO

Le tableau 6 montre que les méthodes de mise à l'échelle ont des résultats un peu similaires dans le cas où le nombre de parties indépendantes de la dissection emboitée (Nparts) est égal à 32 et des nombres d'itérations plus élevés pour les méthodes SSB et SSI dans le cas où Nparts est égal à 16, voir figure 15.

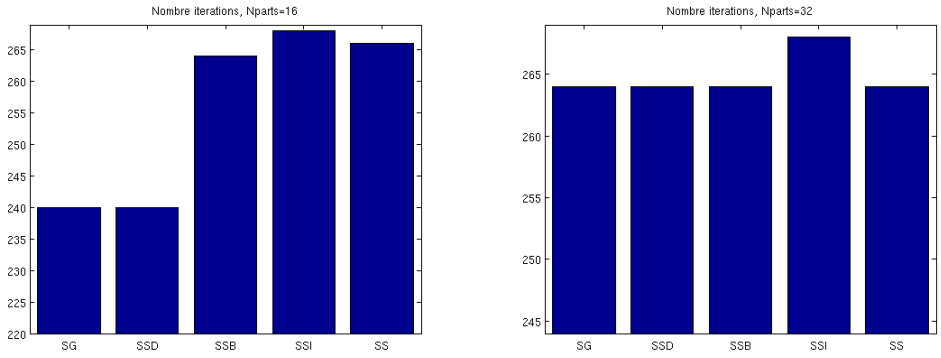


FIGURE 15 – Nombre d'itérations de SG, SSD, SSB et SSI dans le cas ILUO

Préconditionneur BFD

On observe une explosion de la méthode de scaling système tenant compte tous les blocs de la matrice (SSB) en nombre d'itérations pour Nparts égal à 32 où ils atteignent les 400 itérations qui est le maximum à faire, voir figure 16.

Les nombres d'itérations sont moins importants avec la méthode de scaling système global (SG) et un peu moins avec la méthode de scaling système en utilisant les blocs diagonaux.

Concernant les temps de calcul, ils évoluent dans le même sens que les nombres d'itérations, c'est à dire plus le nombre d'itérations est important, plus le temps de calcul augmente. D'autres cas de matrices sont traitées dans l'annexe D.

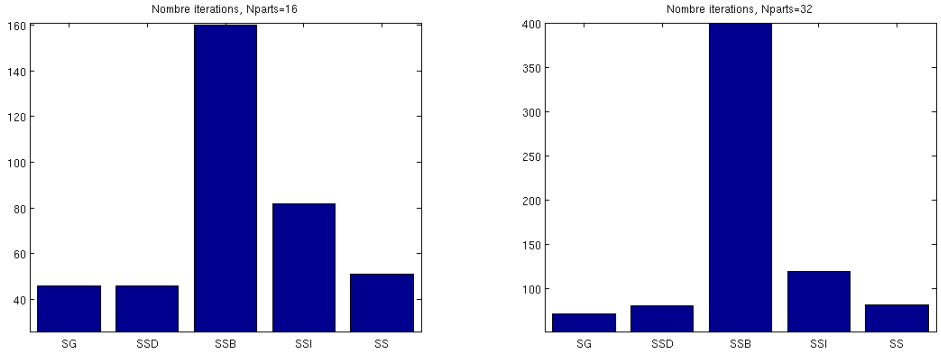


FIGURE 16 – Nombre d'itérations de SG, SSD, SSB et SSI dans le cas BFD

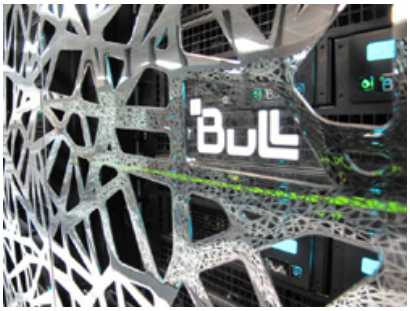
Sur cette section, on peut dire que la méthode de scaling global (SG) est la meilleure méthode de mise à l'échelle de cette matrice *Forge*. Nous avons les nombres d'itérations les moins importants dans les deux cas de préconditionneurs avec des erreurs qui peuvent être acceptées. Le conditionnement également fait partir de ceux qui sont les plus faibles.

4 Étude des solveurs de Petsc sur une matrice issue d'un problème thermo-hydraulique (code Trio_U) sur un grand nombre de processeurs

Les solveurs et préconditionneurs testés dans ce présent document sont ceux disponibles dans la bibliothèque Hypre [8].

Nous présentons la machine curie sur laquelle les calculs ont été effectués et ensuite nous décrivons brièvement les méthodes de résolutions testées dans ce document. Nous avons utilisé deux solveurs BICGSTAB et GCP à l'aide des préconditionneurs ILU(0), PILUT, DIAG, SSOR et SPAI (Sparse Approximate Inverse) ainsi qu'une méthode Multi-Grille.

4.1 Curie



Le supercalculateur Curie appartient à Genci et est hébergé dans les locaux du CEA au TGCC. Il s'agit de la première machine ouverte aux scientifiques européens dans le cadre de la participation française aux instructions de la recherche Prace.

Curie propose trois différents types de ressources de calcul basé sur une architecture X86-64 s'adressant à un large panel de challenges scientifiques et offre une puissance de calcul crête globale de 2Pflops.

1. Spécifications de Curie Fat nodes

   Machine

   360 nœuds S6010 bullx

   Pour chaque nœud : 4 processeurs octo-cœurs Intel® Xeon®, 128 Go,

   1 disque local de 2To.

   105 téraflops crête.

   Processeur

   1440 processeurs octo-cœurs Intel® Nehalem-EX X7560 cadencés à 2.26 GHz

   Mi 2012, grâce à l'intégration par groupe de 4 nœuds, cette configuration pourra disposer de 90 nœuds de 128 cœurs et 512 Go de mémoire/nœud.

   Ces nœuds sont dédiés au passage de codes parallèles hybride (MPI OpenMP),

codes nécessitant une grande capacité mémoire et / ou des capacités de multithreading, ainsi que de pré et post traiter des tâches.

2. Spécifications de Curie noeuds hybrides

Machine(Intégration durant l'été 2011)

16 châssis bullx B équipés chacun de 9 lames hybrides GPUs
B505 2 Intel® Westmere® 2.66 GHz/ 2 Nvidia M2090 T20A,
total de 288 processeurs Intel® + 288 processeurs Nvidia.
+192 téraflops crête.

3. Spécifications de Curie Thin nodes

Machine

5040 nœuds B510 bullx
Pour chaque n œud : 2 processeurs octo-cœurs Intel® Sandy Bridge EP (E5-2680) 2.7 GHz, 64 Go, 1 disque local SSD.

Processeur

10080 processeurs octo-cœurs Intel® Xeon® nouvelle génération soit 80640 cœurs.

Ces nœuds sont plus dédiés au passage de codes parallèles MPI.
disposer de 90 nœuds de 128 cœurs et 512 Go de mémoire/nœud.

En juin 2012, Curie est devenue le 9ème supercalculateur au monde, source [17] et voici son schéma sur la figure 17.

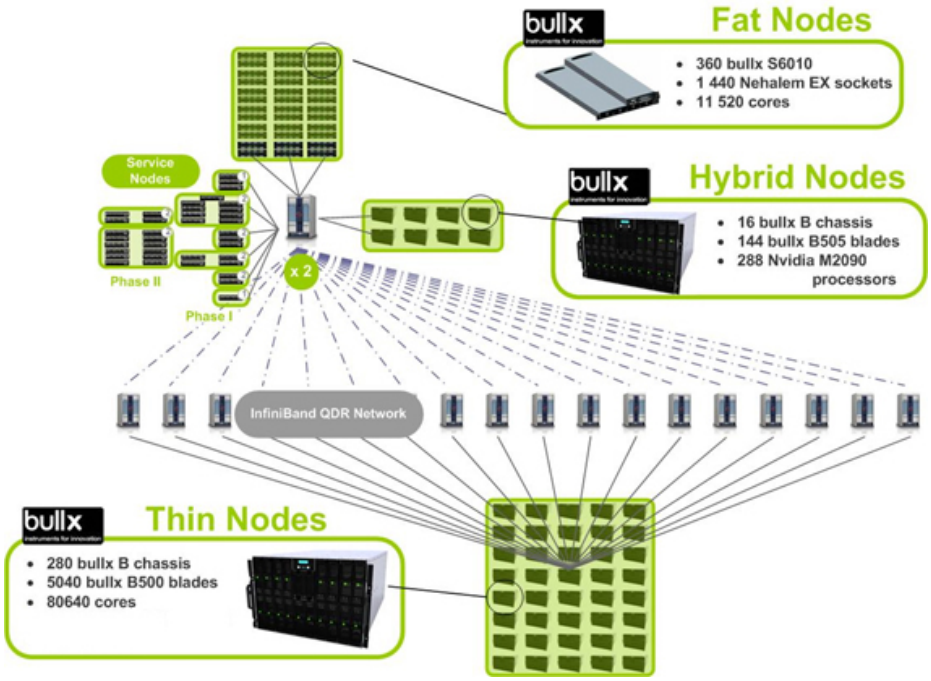


FIGURE 17 – Schéma principe de Curie phase 1 et 2

4.2 Solveur BICGSTAB

L'algorithme de résolution BICGSTAB du système linéaire $Ax = b$ s'écrit de la façon suivante (voir [4] ou [5])

Algorithm 5 Bicgstab

1: Calcul $r_0 = b - Ax_0$, $p_0 = r_0$ et $\tilde{r}_0$
2: Pour $k = 0, 1, 2, ...,$ jusqu'à convergence faire
3: $\alpha_k = \frac{(r_k, \tilde{r}_0)}{(Ap_k, \tilde{r}_0)}$,
4: $s_k = r_k - \alpha_k A p_k$,
5: $\omega_k = \frac{(As_k, s_k)}{(As_k, As_k)}$,
6: $x_{k+1} = x_k + \alpha_k p_k + \omega_k s_k$,
7: $r_{k+1} = s_k - \omega_k A s_k$,
8: $\beta_{k+1} = \frac{(r_{k+1}, \tilde{r}_0)}{(r_k, \tilde{r}_0)} \frac{\alpha_k}{\omega_k}$,
9: $p_{k+1} = r_{k+1} + \beta_{k+1}(p_k - \omega_k A p_k)$,

où $(\cdot, \cdot)$ désigne le produit scalaire euclidien $(u, v) = \sum_i u_i v_i$.

4.3 Solveur GCP

Considérons A une matrice symétrique définie positive. Soit M le préconditionneur associé à A. Le système préconditionné peut être résolu par (3) et l'algorithme de résolution GCP du système linéaire $Ax = b$ est donné par [7].

Algorithm 6 Preconditioned Conjugate Gradient

1: Calcul $r_0 = b - Ax_0, z_0 = M^{-1}r_0,$ et $p_0 = z_0$
2: Pour $k = 0, 1, ...,$ jusqu'à convergence faire
3: $\alpha_k = \frac{(r_k, z_k)}{(Ap_k, p_k)}$,
4: $x_{k+1} = x_k + \alpha_k p_k$
5: $r_{k+1} = r_k - \alpha_k A p_k$,
6: $z_{k+1} = M^{-1} r_{k+1}$,
7: $\beta_k = \frac{(r_{k+1}, z_{k+1})}{(r_k, z_k)}$
8: $p_{k+1} = z_{k+1} + \beta_k p_k$

4.4 Préconditionneurs

4.4.1 ILU

Ce préconditionneur est basé sur une factorisation incomplète LU. L'objectif est de construire une matrice triangulaire inférieure L et une matrice triangulaire supérieure U de telle sorte que la matrice résiduelle $R = LU - A$ vérifie certaines contraintes, comme par exemple forcer certains éléments à être nuls.

4.4.2 PILUT

Ce préconsitionneur est basé sur un algorithme de factorisation incomplète de type ILU mais dans une version parallélisée. Historiquement, la version originelle de PILUT a été construite par M. Karypis et M. Kumar dans [3]. La bibliothèque Petsc propose une version modifiée en utilisant la bibliothèque MPI à la place de la bibliothèque SHMEM.

Rappelons qu'une factorisation incomplète de type ILUT est une factorisation de type ILU où certains éléments sont ignorés durant la factorisation. La version parallèle de ILUT est destinée aux ordinateurs à mémoire distribuée. Tout d'abord, PILUT utilise un algorithme de partitionnement de manière à obtenir une décomposition de domaine selon les processeurs. Chaque processeur calcule une factorisation des nœuds intérieurs de type ILUT. Dans un deuxième temps, des ensembles indépendants sont calculés de manière itérative jusqu'à ce que tous les nœuds aux interfaces soient factorisés.

4.4.3 SPAI

L'idée pour ce type de préconditionneur est de trouver une matrice de préconditionnement M en minimisant la quantité $AM - I$ au sens de la norme de Fröbenius

$$\|AM - I\|_F^2 = \sum_{k=1}^{n} \|Am_k - e^k\|_2^2, \tag{53}$$

où m^k et e^k désignent la $k$ième colonne de M et de la matrice identité. Le problème de minimisation revient à minimiser les n problèmes indépendants

$$\min_{m^k} \|Am_k - e^k\|_2, \quad \forall k = 1, n.$$

La méthode utilisée ici est la méthode proposée par Huckle et Grote [6] et nous renvoyons à [5] ou [7] pour plus de détails.

4.4.4 SSOR

Les méthodes de Jacobi et de Gauss-Seidel sont des méthodes itératives de résolution d'un système matriciel de la forme $Ax = b$. Elle sont données par

$$Mx_{k+1} = Nx_k + b = (M - A)x_k + b, \tag{54}$$

où

$$A = M - N, \tag{55}$$

avec $M = D$ pour Jacobi, $M = D - E$ pour Gauss Seidel avancée et $M = D - F$ pour Gauss Seidel retardée. Un méthode itération de la forme (54) peut se définir pour n'importe quelle partition de (55) où M est non singulière.

Une surrelaxation est basée sur la partition [7]

$$wA = (D - wE) - (wF + (1 - w)D) \tag{56}$$

et la méthode de surrelaxation successive (Successive Over Relaxation (SOR) en anglais) est donnée par l'occurrence (57)

$$(D - wE)x_{k+1} = [wF + (1 - w)D]x_k + wb. \tag{57}$$

Une SOR symétrique (SSOR) est donnée par les formules (58) et (59)

$$(D - wF)x_{K+1/2} = [wE + (1 - w)D]x_k + wb, \tag{58}$$

$$(D - wF)x_{K+1} = [wE + (1 - w)D]x_{k+1/2} + wb. \tag{59}$$

4.4.5 DIAG

Le préconditionneur diagonal est obtenu en formant une matrice diagonale avec les éléments de la diagonale de A.

Définition 3. *Polynôme de Neumann*
La série de Neuman permet d'approximer l'inverse d'une matrice A. D la matrice diagonale composée des coefficients diagonaux de A. Alors on pose $A = D - C = (I - CD^{-1})D$.
En utilisant série de Neuman on obtient l'expression suivante

$$A^{-1} = D^{-1}(I - CD^{-1})^{-1}, \tag{60}$$

$$A^{-1} = D^{-1}(I + CD^{-1} + (CD^{-1})^2 + (CD^{-1})^3 + ...). \tag{61}$$

En tronquant l'expression au terme CD^{-1} de puissance n, on obtient le polynôme de Neumann de degré n. Le préconditionneur diagonal correspond au polynôme de Neumann de degré 0.

4.5 Méthode Multi-Grille

Il s'agit d'une méthode Multi-Grille algébrique provenant de la librairie Petsc. C'est un préconditionneur dans Petsc. Nous renvoyons à [5] pour une description complète de cette méthode. La mise en œuvre dans Petsc est celle décrite dans [9].

4.6 Résultats numériques

Dans cette section, on présente les performances des deux solveurs et préconditionneurs de Petsc vus ci-dessus. Les tableaux récapitulatifs des résultats sont donnés par 7 et 8 et les figures associées 18 et 19.

BICGSTAB	SSOR	DIAG	SPAI	ILU	PILUT	BOOMERANG
Itérations	24598	24222	24022	23939		
Temps CPU(s)	169.4	160.74	159.35	158.35		

TABLE 7 – Solveur BICGSTAB

BICGSTAB	SSOR	DIAG	SPAI	ILU	PILUT	BOOMERANG
Itérations	24433		24331	24087		
Temps CPU(s)	155.65		162.9	172.82		

TABLE 8 – Solveur GCP

On remarque d'abord sur ces deux tableaux (7 et 8), qu'il n'y a pas d'itérations ni de temps de calcul pour les préconditionneurs PILUT et BOOMERANG. Ceci est dû à une non convergence de ces deux méthodes.

Concernant la méthode DIAG dans le cas du solveur GCP, ces absences de résultats sont causées par un épuisement du quota de temps de calcul qu'on nous avait octroyé pour la matrice Curie.

On représente les données sur les figures 18 et 19.

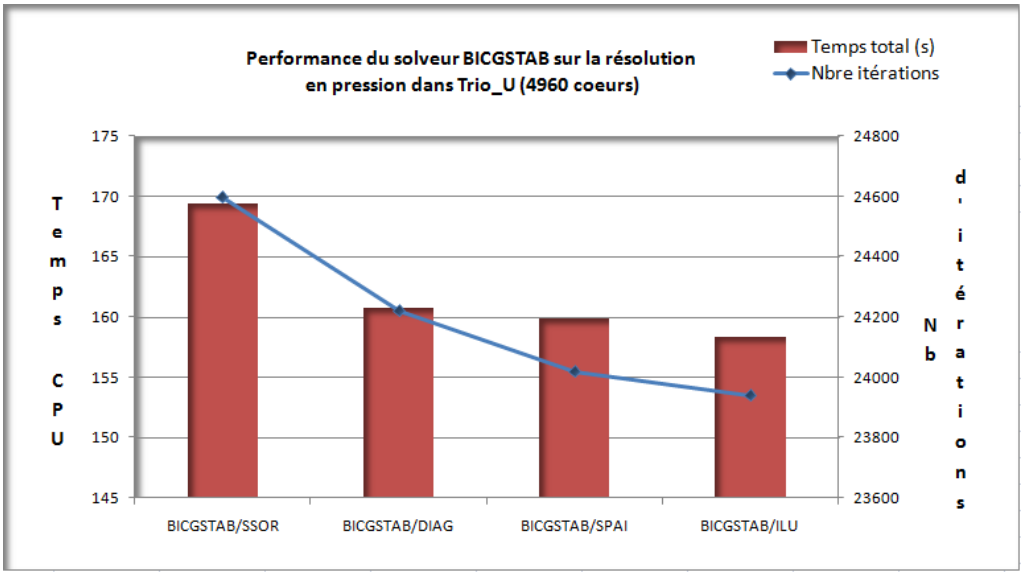


FIGURE 18 – Performance du solveur BICGSTAB

On constate sur cette figure 18 que lorsque le nombre d'itérations diminue, le temps de calcul diminue aussi. Avec le préconditionneur ILU, on fait moins d'itérations et moins de temps de calcul.

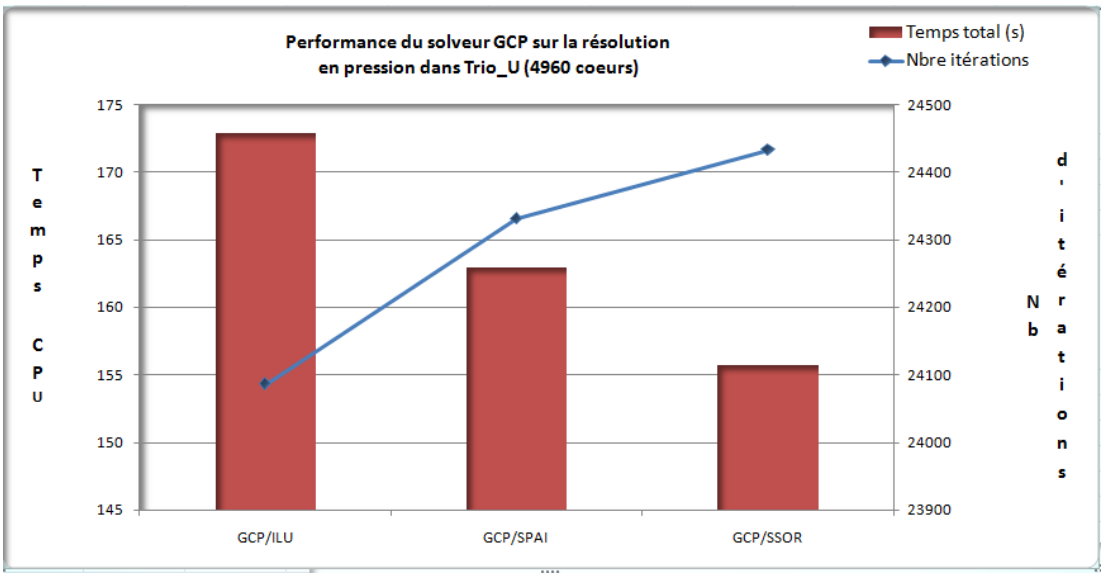


FIGURE 19 – Performance du solveur GCP

Cette figure 19 montre dans le cas du solveur GCP, malgré une augmentation du nombre d'itérations, le temps de calcul diminue.

On voit sur cette méthode que le temps CPU et le nombre d'itérations ne vont pas dans le même sens contrairement à la méthode de BICGSTAB où on observe une diminution de ces deux éléments en même temps. Cette situation peut s'expliquer par le fait que dans la méthode de GCP, les erreurs de calculs font perdre l'orthogonalité des directions de descente alors que la convergence est rapide et donc peut exiger plus d'itérations même si le parallélisme est incorporé pour réduire les calculs intermédiaires. Cette augmentation peut s'expliquer également par les échanges faits entre le solveur et le préconditionneur.

5 Conclusion

Durant ce stage, nous avons vu que la méthode du block filtering décomposition (BFD) présente plusieurs avantages par rapport à la méthode d'incomplète LU (ILUO). L'erreur relative est beaucoup moins importante et le nombre d'itérations obtenu est largement plus petit. Cependant, il existe des cas particuliers où le conditionnement de la matrice par BFD n'est pas meilleur que celui donné par ILUO.

Nous avons vu également que la mise à l'échelle de la matrice offre beaucoup plus d'avantages. Le nombre d'itérations diminue considérablement (3 fois moins d'itérations dans certains cas), le conditionnement de la matrice aussi (10^5 moins) et l'erreur est presque dans tous les cas plus petite.

Parmi les méthodes de mise à l'échelle développées, la méthode de scaling système itératif est meilleure en terme de conditionnement et d'erreur. Cependant en terme de nombre d'itérations, il est difficile de différencier les méthodes scaling global (SG), scaling système en utilisant tous les blocs de la matrice (SSB) et scaling système itératif (SSI).

Le solveur BICGSTAB donne des résultats plus concluants que ceux de GCP. Les nombres d'itérations diminuent et les temps de calcul aussi vont dans le même sens. Pour les préconditionneurs, la méthode ILU offre moins d'itérations et de temps de calcul dans le cas du solveur BICGSTAB et moins d'itérations malgré un temps de calcul plus élevé dans le cas du solveur GCP.

En perspective de ce projet, il serait intéressant de pouvoir programmer la méthode de Block Filtering Decomposition (BFD) en calcul parallèle pour l'intégrer dans la bibliothèque de Petsc et de la comparer avec les méthodes existantes. Cette méthode donne des résultats plus satisfaisants que celle d'ILU dans le cas non parallèle alors que cette dernière semble meilleure dans Petsc.

Outre l'aspect d'équipe et organisationnel, ce stage m'a permis de voir des aspects techniques de l'analyse numérique comme le préconditionnement et la mise en échelle d'une matrice ainsi que l'utilité du calcul parallèle pour résoudre les grandes matrices. Il est ma première expérience significative sur le plan professionnel et en particulier dans le domaine de la recherche en calcul scientifique et en analyse numérique.

6 Annexes

A Cas non homogéne

Le cas non homogéne avec

$$\begin{cases} \eta(x) = 0, \\ a(x) = 0, \end{cases} \tag{62}$$

Le tenseur k est isotrope et discontinue. Il fait un saut à partir de 10^3 dans le domaine $2^{-\frac{3}{2}} \leqslant |x - c| \leqslant 2^{-1}, \quad c = (0.5, 0.5)^T$ à 1 en dehors.

	ILUO		BFD	
Ordre	Nbre-Itér	Erreur	Nbre-Itér	Erreur
D.E.16	400	3.8E-4	44	6.6E-8
D.E.32	400	4.1E-4	45	4.0E-8
D.E.64	400	3.0E-4	45	4.8E-8

TABLE 9 – Résultats de ILUO et BFD dans le cas non homongéne

B Application : Cas de thermo-hydraulique

On étudie le cas d'une matrice de taille 270×270 avec nombre d'éléments non nuls égale à 4575

Nparts	16				32		
	Cdt	NbrIter	Tps(s)	Erreur	NbrIter	Tps(s)	Erreur
ILUO							
SG	2e+5	91	0.2	7.48e-12	91	0.14	3e-11
SSD	8e+16	37	0.1	909.07	38	0.03	6e+2
SS	5e+17	37	0.1	909.1	38	0.03	6e+2
BFD							
SG	2e+05	32	0.7	7.48e-11	32	1.9	4e-12
SSD	8e+16	47	0.87	4.2e+07	48	2.4	2e+08
SS	5e+17	47	0.89	4.17+06	48	2.9	1.33e+06

TABLE 10 – Résultats de SG, SSD, SSB, cas matrice Thermo-hydraulique

C Erreurs de SG, SSD et SSB cas *Coulpex Gaz*

FIGURE 20 – Erreur de SG, SSD et SSB cas matrice *Couplex Gaz*

D Comparaison de SG, SSD,SSB et SSI avec d'autres matrices

Matrice :Forge_benchmark_bench1 : 17646 × 17646, nombre éléments non nuls :456132.

Nparts	16				32		
	Cdt	NbrIter	Tps(s)	Erreur	NbrIter	Tps(s)	Erreur
ILUO							
SG	2e+1	1	0.16	2e-8	1	0.1	2e-11
SSD	9e+1	1	0.18	2e-10	1	0.1	2e-10
SSB	1e+5	1	0.16	1e-4	1	1.8	1e-4
SSI	9e+0	1	0.17	2e-7	1	2.4	2e-12
SS	6e+9	1	0.21	2e-8	1	0.1	2e-8
BFD							
SG	2e+1	1	1.4	9e-12	1	2.1	2e-11
SSD	9e+1	1	1	9e-14	1	2.2	2e-13
SSB	1e+5	1	0.8	9e-6	1	1.8	2e-5
SSI	9e+0	1	1.5	1e-12	1	2.4	2e-12
SS	6e+9	1	1.4	1e-11	1	2.1	2e-1

TABLE 11 – Resultats de SG, SSD, SSB et SSI cas matrice FORGE

Matrice : two_phase_dissolution_tough2_model : 4504 × 4504, nombre éléments non nuls :113216.

Nparts	16				32		
	Cdt	NbrIter	Tps(s)	Erreur	NbrIter	Tps(s)	Erreur
ILUO							
SG	3e+2	14	0.14	4e-9	13	0.09	1e-8
SSD	3e+4	14	0.17	1e-7	14	0.09	1e-9
SSB	5e+5	13	0.17	1e-6	13	1.09	9e-7
SSI	5e+2	13	0.19	1e-11	13	0.09	6e-8
SS	7e+5	14	0.28	8e-8	14	0.09	6e-8
BFD							
SG	3e+2	9	0.63	4e-9	9	1.9	5e-9
SSD	3e+4	139	4.8	9e-9	400	27	2e-5
SSB	5e+5	9	0.73	5e-7	10	2	8e-9
SSI	5e+2	8	0.62	3e-12	9	2	1e-12
SS	7e+5	17	1	6e-8	17	2.2	1e-7

TABLE 12 – Resultats de SG, SSD, SSB et SSI cas matrice two-phase-dissolution-tough2-model

Matrice : Pleine d'éléments aleatoires : 1020×1020

Nparts	16				32		
	Cdt	NbrIter	Tps(s)	Erreur	NbrIter	Tps(s)	Erreur
ILUO							
SG	4e+4	1	0.29	2e-13	1	0.18	2e-15
SSD	2e+5	1	0.24	2e-15	1	0.17	9e-15
SSI	1e+4	1	0.17	4e-13	1	0.18	4e-14
SS	1e+7	1	0.25	1e-12	1	0.18	2e-12
BFD							
SG	4e+4	27	0.96	1e-9	23	2.3	8e-9
SSD	2e+5	29	0.91	7e-9	34	2.7	1e-7
SSI	1e+4	27	0.82	1e-8	25	2.4	5e-8
SS	1e+7	27	1.1	2e-5	25	2.6	1e-3

TABLE 13 – Resultats de SG, SSD, SSB et SSI cas Matrice aleatoire pleine

Références

[1] F. Caro and E. Laucoin, "Description du module MPCube permettant la simulation numérique des écoulements diphasiques en milieu poreux", CEA (2009) SFME/LSET/RT/09-006/A

[2] `http://www-trio-u.cea.fr/`, Trio_U

[3] G. Karpis and V. Kumar,"Parallel thershold-based ILU factorization", University of Minnesota,061(1998),Department of Computer Science/Army HPC Research Center, Minneapolis, MN 5455

[4] H. A. Van Der Lost, "BiCGSTAB : a fast and smoothly converging variant of BiCG for the solution of nonsymetric linear systems", SIAM J. Sci. Comp.13(1992) 631-644

[5] G. Meurant, "Computer solution of large linear systems", Studies in mathematics and its applications, Elsevier, 1999

[6] T. Huckle and M. Grote, "A new approach to parallel preconditioning with sparse approximate inverses", Computer Science Dept., Standford University, (1994) SCCM-94-03

[7] Y. Saad, "Iterative methods for sparse linear systems", Computer science/Numerical method,PWS publishing compan, 1996

[8] `http://www.llnl.gov/CASC/linear_solvers/`

[9] V. E. Henson and U. M. Yang, "BoomerAMG : a parallel algebraic multigid solver and preconditioner", Devlopments and trends in iterative methods for large systems of equations – in memoriam Redriger Weiss, Applied Numerical Mathematics, 41(2002) 155–177

[10] M.v Golitschek, U.G.Rothblum and H.Schneider," A conforming decomposition theorem, a picewise linear theorem of the alternative, and scalings of matrices satisfying lower and upper bounds", Mathematical Programming 27(1983) 291-306

[11] R.Sinkhorn and P.Knopp, " concerning nonnegative matrices and doubly stochastic matrices ", Pacific Journal of Mathematics 21 (1967) 343-348

[12] W.Marshall and I.Olkin, " Scaling of matrices to achieve specified row and column sums", Numerische Mathematik 12(1968) 83-90

[13] M.V Menon, " Reduction of a matrix with positive elements to a doubly stochastic matrix", Proceedings of the American Mathematical Society 18 (1967)244-247

[14] M Daniel Ruiz " A Scaling Algorithm to Equilibrate Both Rows and Columns Norms in Matrices", RAL -TR -2001-034

[15] E.ACHILLES, (1993), "Implications of Convergence Rates in Sinkhorn Balancing, Lenear Algebra and its Applications", 187-109-112

[16] R.BUNCH, (1971), "Equilbration of Symmetric Matrices in the Max-Norm ",J.ACM-18-566-572

[17] `http://www.journaldunet.com/solutions/cloud-computing/ top-500-des-super calculateurs-0612.shtml`