INSTITUT FRANÇAIS DU PÉTROLE
Division Technologie Informatique Scientifique et Mathématiques Appliquées

MÉTHODES DE RÉSOLUTION DE SYSTÈMES LINÉAIRES POUR UN PROTOTYPE DE CALCUL DE L'INJECTION DE CO_2 DANS DES MILIEUX POREUX

LOOS Florian (étudiant de MASTER MATHÉMATIQUES,
Université de Marne-La-Vallée)

Responsables de stage : TRENTY Laurent (R1150), TILLIER Elodie (R1150),
EYMARD Robert (Université de Marne-La-Vallée)

RESUMÉ :
La modélisation du stockage géologique de CO_2 débouche sur un très grand système d'équations couplées. En cherchant sa solution, on est amené à résoudre des systèmes linéaires de taille importante.
Pour trouver leur solution rapidement, on se sert de diverses techniques appelées des méthodes itératives.
L'objectif de ce stage est de mettre en place un solveur itératif, la méthode BICGSTAB avec préconditionnement, dont les résultats sont comparés avec ceux obtenus par un solveur direct.

Mots-Clés : Sparse, Red & Black, préconditionnement, ILU(0), solveur, BICGSTAB

TABLE DES MATIÈRES

REMERCIEMENTS

D'abord, je veux remercier Mr Robert EYMARD (professeur de l'Université de Marne-La-Vallée) qui m'a cherché ce stage et grâce qui j'ai eu l'occasion de travailler à l'IFP.

En plus, je remercie surtout Mlle Elodie TILLIER qui s'est occupée de moi, qui m'a guidé et conseillé pendant les semaines de mon stage.

Merci à Mr Laurent TRENTY, Mr Anthony MICHEL, Mr Eric FLAURAUD et Mr Roland MASSON pour leur aide et leurs conseils.

J'ai bien aimé travailler avec l'équipe de l'IFP dans une ambiance très agréable.

INTRODUCTION

Pour limiter les émissions de gaz à effet de serre dans l'atmosphère, l'une des solutions envisagées est la capture et le stockage géologique du CO_2. La modélisation du stockage débouche sur un système couplé d'équations aux dérivées partielles non linéaires et d'équations algébriques locales.
Pour trouver la solution de ce problème, on utilise une méthode de Newton et on doit donc résoudre des systèmes linéaires $Ax = b$ où A est une matrice de grande taille, x la solution du système linéaire et b le second membre.
Comme généralement la matrice A obtenue est très grande mais souvent creuse, on perd beaucoup de mémoire en stockant la matrice A entièrement. Au lieu de stocker toutes les valeurs de cette matrice, on se sert d'une méthode qui ne stocke que les valeurs non nulles.
De plus, il n'est pas efficace de résoudre de tels systèmes linéaires par un solveur direct comme l'algorithme de Gauss. L'emploi de méthodes itératives est un moyen qui permet de trouver leur solution rapidement. BICGSTAB est un solveur itératif qui résout des systèmes linéaires de façon peu couteuse.
Finalement, les résultats obtenus par BICGSTAB sont comparés avec ceux obtenus par la résolution avec Gauss et avec ceux obtenus par BICGSTAB avec préconditionnement.

1 STOCKAGE DES MATRICES CREUSES

1.1 Red and Black (R&B)

La discrétisation des équations sur le maillage donne une matrice A creuse, par bloc, et un second membre b.
La structure de la matrice A dépend de la numérotation des mailles du maillage. Si les mailles sont numérotées en ordre ascendant, A est une matrice par bande.
Au lieu de numéroter les mailles en ordre ascendant, on les numérote en échiquier avec les couleurs rouge et noir. Comme ça, chaque point rouge n'a que des voisins noirs et inversement.

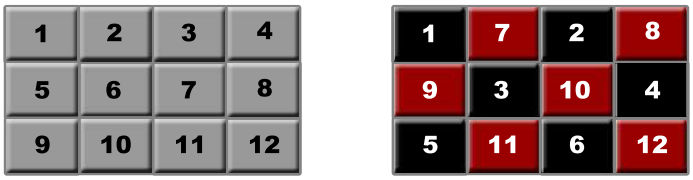

FIG. 1 – Numérotation d'un maillage par ordre ascendant et par R&B

La forme de la matrice A après numérotation par «Red and Black» se présente de la façon suivante :

$$A = \left(\begin{array}{c|c} D_1 & B \\ \hline R & D_2 \end{array}\right)$$

B et R sont des matrices bandes par bloc où B est affectée aux points rouges, R aux points noirs. D_1 et D_2 sont des matrices diagonales par bloc :

$$D_1 = \begin{pmatrix} \boxed{A_1} & & \\ & \ddots & \\ & & \boxed{A_{n_1}} \end{pmatrix} \quad D_2 = \begin{pmatrix} \boxed{A_{n_1+1}} & & \\ & \ddots & \\ & & \boxed{A_n} \end{pmatrix}$$

Si on multiplie la totalité du système linéaire par une matrice $\hat{A}$ contenant les inverses des blocs diagonaux de D_1 et D_2, on obtient un nouveau système linéaire qui est «plus facile» à résoudre.

$$\tilde{A} = \hat{A}A = \left(\begin{array}{c|c} D_1^{-1} & \theta \\ \hline \theta & D_2^{-1} \end{array}\right)\left(\begin{array}{c|c} D_1 & B \\ \hline R & D_2 \end{array}\right)$$

$$\tilde{b} = \hat{A}b = \left(\begin{array}{c|c} D_1^{-1} & \theta \\ \hline \theta & D_2^{-1} \end{array}\right)\left(\begin{array}{c} b_1 \\ \hline b_2 \end{array}\right)$$

Il reste à résoudre le nouveau système lineáre (voir **2.3**) :

$$\tilde{A}x = \tilde{b}$$

$$\left(\begin{array}{c|c} Id1 & \tilde{B} \\ \hline \tilde{R} & Id2 \end{array}\right)\left(\begin{array}{c} x_1 \\ \hline x_2 \end{array}\right) = \left(\begin{array}{c} \tilde{b}_1 \\ \hline \tilde{b}_2 \end{array}\right)$$

$Id1$ et $Id2$ sont des matrices unités, $Id1$ est de la même taille que $D1$, $Id2$ de la même taille que $D2$.

1.2 Compressed Sparse Row (CSR)

Comme la plupart des blocs de A sont des blocs nuls, on utilisait de la mémoire inutile en stockant la matrice A entièrement. C'est la raison pour laquelle on se sert d'un format de stockage qui s'appelle «Compressed Sparse Row» (CSR), on dit souvent simplement «stockage morse».
Au lieu de stocker toutes les valeurs de la matrice A, on ne stocke que les valeurs non nulles avec leur position dans la matrice.
La structure des données consiste en trois parties :

- Un tableau AA qui contient les valeurs non nulles de la matrice A, stockées colonne par colonne de la première colonne à la dernière. La taille de ce tableau correspond au nombre de valeurs non nulles (noté Nz).

– Un tableau JA de taille Nz contenant les numéros de colonne des éléments non nuls de la matrice A stockés dans AA.
– Un tableau IA de pointeurs de longeur $nbln+1$ (où $nbln$ est le nombre de lignes). Le i-ème pointeur pointe sur le numéro de la valeur dans AA et JA qui contient le premier élément non nul de la ligne i dans A. La valeur $IA[nbln+1]$ indique la fin des valeurs non nulles dans AA et JA.

Exemple :

La matrice

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

est représentée par

$$\begin{array}{rl} AA = & \boxed{1\ 3\ 4\ 2\ 3\ 1\ 2} \\ JA = & \boxed{0\ 3\ 1\ 3\ 1\ 0\ 1} \\ IA = & \boxed{0\ 2\ 4\ 5\ 7} \end{array}$$

Il était mentionné que la matrice A est une matrice par bloc où tous les blocs sont de la même taille. Pour conserver et profiter de cette structure, on a adapté le stockage CSR :
Chaque valeur de l'array AA contient une matrice de la taille d'un bloc où au moins une valeur est non nulle. Donc, AA est un array de matrices dont la taille de l'array correspond au nombre de blocs non nuls.
Si la matrice A de l'exemple précédent est divisée en quatre blocs de taille 2×2, elle est stockée de la façon suivante :

$$A = \left(\begin{array}{cc|cc} 1 & 0 & 0 & 3 \\ 0 & 4 & 0 & 2 \\ \hline 0 & 3 & 0 & 0 \\ 1 & 2 & 0 & 0 \end{array}\right)$$

$$\begin{array}{rl} AA = & \boxed{\begin{array}{cc} 1 & 0 \\ 0 & 4 \end{array}} \boxed{\begin{array}{cc} 0 & 3 \\ 0 & 2 \end{array}} \boxed{\begin{array}{cc} 0 & 3 \\ 1 & 2 \end{array}} \\ JA = & \boxed{0\ 1\ 0} \\ IA = & \boxed{0\ 2\ 3} \end{array}$$

2 RÉSOLUTION DES SYSTÈMES LINÉAIRES

Afin de résoudre des système linéaires de grande taille

$$Ax = b \tag{2.1}$$

on se sert souvent des méthodes iteratives.
Depuis les années 1960s, quelques solveurs itératifs ont été découverts et ils sont très utiles pour beaucoup d'applications, surtout lorsque les systèmes sont linéaires, creux et de grande taille.
Des systèmes creux demandent des méthodes capables de les résoudre très vite ce qui n'est pas le cas des méthodes directes comme l'algorithme de Gauss. C'est la raison pour laquelle il est intéressant d'utiliser et d'implémenter des méthodes itératives comme BiCGStab.
La notion de «préconditionnement» définit une technique qui permet de tranformer la matrice pour obtenir de bonnes propriétés comme un meilleur conditionnement ou une convergence plus rapide.

2.1 Préconditionnement par factorisation incomplète (ILU(0))

En général, le préconditionnement est une transformation d'un système linéaire qui permet d'appliquer une méthode itérative «plus facilement».
Comme la convergence d'une méthode itérative dépend du conditionnement de la matrice A (noté $K(A)$), on essaie de trouver une matrice P «facile à inverser» telle que $K(P^{-1}A) \ll K(A)$. Ensuite, on résout le système linéaire transformé

$$P^{-1}Ax = P^{-1}b. \tag{2.2}$$

Une méthode de préconditionnement particulièrement intéressante est la factorisation incomplète de la matrice A, notée par ILU.
On calcule une matrice triangulaire inférieure creuse à diagonale unité L et une matrice triangulaire supérieure creuse U telles que la matrice résidu $R = LU - A$ satisfasse certaines conditions.
Dans notre cas, on s'est servi de la factorisation ILU(0), pour laquelle les valeurs de la matrice résidu R sont nulles à tous les endroits où la matrice A est non nulle.
La factorisation incomplète de degré zéro (notée ILU(0)) consiste à chercher deux matrices L et U telles que L et U soient nulles à tous les endroits où A est nulle.
La décomposition de A en L et U avec les conditions indiquées n'est pas unique. Nous avons programmé l'ILU(0) standard en utilisant l'algorithme qui est présenté en ANNEXE 1.
Le système lineáire est préconditionné par le produit de L et U ($P = L * U$).

Exemple :

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

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1/2 & 0 & 1 & 0 \\ 1/2 & 1/6 & 0 & 1 \end{pmatrix} \quad U = \begin{pmatrix} 2 & 3 & 0 & 1 \\ 0 & 3 & 0 & 2 \\ 0 & 0 & 2 & 1/2 \\ 0 & 0 & 0 & 13/6 \end{pmatrix}$$

$$L * U = \begin{pmatrix} 2 & 3 & 0 & 1 \\ 0 & 3 & 0 & 2 \\ 1 & 3/2 & 2 & 1 \\ 1 & 2 & 0 & 3 \end{pmatrix}$$

2.2 Méthodes itératives (BICGSTAB)

Pour résoudre le système linéaire d'origine (2.1) ou le système linéaire préconditionné (2.2), il faut choisir un solveur qui trouve la solution assez rapidement sans gaspiller trop de mémoire.
Les méthodes de Krylov sont des méthodes de résolution qui, généralement, convergent vite sans être couteuse en temps calcul.
Parmi les méthodes de Krylov, on a choisi la méthode du gradient bi-conjugé stabilisé (BICGSTAB). BICGSTAB converge normalement très vite sans utiliser beaucoup de mémoire (en comparaison avec d'autres méthodes de Krylov).
Les algorithmes du BICGSTAB et du BICGSTAB préconditionné sont présentés en annexe.

2.3 Résolution du système transformé obtenu par R&B

Dans le paragraphe **1.1**, il était décrit comment on transforme la matrice A du système d'origine pour obtenir une matrice de la forme :

$$\tilde{A} = \left(\begin{array}{c|c} Id1 & \tilde{B} \\ \hline \tilde{R} & Id2 \end{array}\right)$$

Le but est alors de résoudre le système modifié

$$\tilde{A}x = \tilde{b}.$$

$$\left(\begin{array}{c|c} Id1 & \tilde{B} \\ \hline \tilde{R} & Id2 \end{array}\right) \left(\begin{array}{c} x_1 \\ \hline x_2 \end{array}\right) = \left(\begin{array}{c} \tilde{b}_1 \\ \hline \tilde{b}_2 \end{array}\right)$$

$$\begin{array}{rl} x_1 + \tilde{B}x_2 = \tilde{b}_1 & (I) \\ \tilde{R}x_1 + x_2 = \tilde{b}_2 & (II) \\ \hline \tilde{R}x_1 + \tilde{R}\tilde{B}x_2 = \tilde{R}\tilde{b}_1 & (I^{'}) \\ \tilde{R}x_1 + x_2 = \tilde{b}_2 & (II) \\ \hline x_1 + \tilde{B}x_2 = \tilde{b}_1 & (I) \\ \tilde{R}\tilde{B}x_2 - x_2 = \tilde{R}\tilde{b}_1 - \tilde{b}_2 & (II^{'}) \end{array}$$

Finalement, il faut résoudre le système $(\tilde{R}\tilde{B} - I)x_2 = \tilde{R}\tilde{b}_1 - \tilde{b}_2$ et calculer ensuite $x_1 = b_1 - \tilde{B}x_2$.
Pour la résolution de ce système dont la taille a été divisée par 2, on peut se servir des méthodes itératives et du préconditionnement décrits précedement.

3 TESTS ET RÉSULTATS

Dans ce paragraphe, on compare d'abord les résultats obtenus par GAUSS avec ceux obtenus par BICGSTAB et BICGSTAB préconditionné. Ensuite, plusieurs méthodes itératives sont considérées et on les compare par rapport au nombre d'itérations et à la vitesse de convergence.

3.1 Comparaison de la qualité des résultats sur un exemple

Sur un cas 1D horizontal avec une chimie simplifiée (cas test CO_2) où le maillage est divisé en 500 mailles, la qualité de la solution obtenue par BICGSTAB et BICGSTAB préconditionné par ILU(0) est examinée. En faisant varier la tolérance de BICGSTAB, on calcule les erreurs rélatives.

	Qualité de la solution						
	Précision	1.0e-03	1.0e-04	1.0e-05	1.0e-06	1.0e-07	1.0e-08
BICGSTAB	Itérations	35	519	1846	1914	1996	2053
	Conditionnement	1.86e+07					
	Erreur relative	8.54e-01	2.84e-02	5.35e-03	3.83e-04	5.17e-05	9.49e-07
BICGSTAB préconditionné	Itérations	8	100	134	156	167	179
	Conditionnement	9.92e+06					
	Erreur relative	7.34e-01	1.2e-01	7.42e-02	1.37e-02	6.27e-03	2.50e-03

	Qualité de la solution						
	Précision	1.0e-09	1.0e-10	1.0e-11	1.0e-12	1.0e-13	1.0e-14
BICGSTAB	Itérations	2189	2786	2817	2877	3113	3422
	Conditionnement	1.86e+07					
	Erreur relative	7.32e-07	1.62e-06	1.60e-06	1.60e-06	1.60e-06	1.60e-06
BICGSTAB préconditionné	Itérations	181	186	192	221	222	268
	Conditionnement	9.92e+06					
	Erreur relative	4.26e-04	3.09e-04	3.19e-04	3.20e-04	3.20e-04	3.20e-04

	Qualité de la solution						
	Précision	1.0e-15	1.0e-16	1.0e-17	1.0e-18	1.0e-19	1.0e-20
BICGSTAB	Itérations	3561	3582	3626	3678	-1	-1
	Conditionnement	1.86e+07					
	Erreur relative	1.60e-06	1.60e-06	1.60e-06	1.60e-06	1.60e-06	1.60e-06
BICGSTAB préconditionné	Itérations	-1	-1	-1	-1	-1	-1
	Conditionnement	9.92e+06					
	Erreur relative	3.20e-04	3.20e-04	3.20e-04	3.20e-04	3.20e-04	3.20e-04

FIG. 2 – Test de la qualité des solutions sur un exemple

Dans ce test, le nombre maximal d'itérations de BICGSTAB s'élève à 50000. -1 comme valeur pour le nombre d'itérations signifie que BICGSTAB s'arrête car l'un des critères nécessaires à la bonne continuation de l'algo-

rithme n'est pas vérifié.

Conclusion

BICGSTAB donne sur notre exemple choisi de très bonnes solutions. En plus, on voit que par préconditionnement, le conditionnement des matrices est amélioré, ce qui entraîne une convergence plus rapide.
Malgré cela, il faut faire attention avec BICGSTAB. Si la tolérance est trop faible, BICGSTAB souvent ne converge pas. Ce comportement peut être prévu avec des «méthodes de Look-Ahead» qui ne sont pas présentées dans ce rapport.

3.2 Comparaison de la vitesse de convergence

En considérant plusieurs examples, on compare les cinq méthodes suivantes :

- GAUSS
- BICGSTAB
- BICGSTABpre : BICGSTAB avec préconditionnement par ILU(0)
- BICRB : Le système linéaire après élimination par R & B est résolu avec BICGSTAB
- BICRBpre : Le système linéaire après élimination par R & B est résolu avec BICGSTAB préconditionné par ILU(0)

Les exemples sur lesquels les méthodes sont appliquées représentent un cas 1D horizontal (avec écoulement par différence de pression) sur la minéralogie de l'Alberta. La fin du nom des exemples correspond au nombre de mailles. Par exemple, horiz100 indique que le maillage est divisé en 100 mailles.
Le nombre des équations sur chaque maille est 10. Par conséquent, pour le cas horiz100, la dimension de la matrice du système linéaire est 100×100.
La tolérance de BICGSTAB pour tous les exemples est de $1.0e-10$ et le nombre maximal d'itérations est 99999.

	Nombre d'Itérations					
	horiz50	horiz100	horiz200	horiz300	horiz400	horiz500
BICGSTAB	-1	99999	99999	99999	99999	99999
BICGSTABpre	-1	648	1848	2140	2836	4244
BICRB	25	64	114	166	216	269
BICRBpre	13	22	43	65	84	102

	Nombre d'Itérations				
	horiz600	horiz700	horiz800	horiz900	horiz1000
BICGSTAB	99999	99999	99999	99999	99999
BICGSTABpre	7055	8776	8732	12259	16674
BICRB	317	395	483	521	559
BICRBpre	120	137	155	176	191

	Temps de calcul (en s)					
	horiz50	horiz100	horiz200	horiz300	horiz400	horiz500
GAUSS	2,5	16	124	416	915	1795
BICGSTAB	54	133	514	788	996	1317
BICGSTABpre	1,1	3,8	19	34	59	116
BICRB	0,6	1,0	2,5	5,3	8,2	16
BICRBpre	0,5	1,0	2,4	5,1	7,9	14

	Temps de calcul (en s)				
	horiz600	horiz700	horiz800	horiz900	horiz1000
GAUSS	3079	4882	7272	10345	14170
BICGSTAB	1510	1798	2060	2311	2703
BICGSTABpre	239	339	410	654	869
BICRB	23	34	42	54	70
BICRBpre	22	32	37	50	65

	Erreur relative					
	horiz50	horiz100	horiz200	horiz300	horiz400	horiz500
BICGSTAB	5.70e+140	2.52e+90	9.96e-01	9.98e-01	8.18e+86	9.98e-01
BICGSTABpre	1.66e-10	8.97e-11	2.25e-08	4.18e-10	5.36e-10	3.04e-10
BICRB	1.95e-10	5.82e-08	6.18e-12	5.37e-08	7.42e-08	9.06e-08
BICRBpre	1.56e-10	2.20e-08	2.25e-12	2.13e-08	7.42e-08	8.47e-08

	Erreur relative				
	horiz600	horiz700	horiz800	horiz900	horiz1000
BICGSTAB	9.99e-01	9.99e-01	2.90e+08	4.83e+76	1.35e+08
BICGSTABpre	6.31e-10	3.25e-08	1.54e-08	2.89e-08	7.72e-08
BICRB	4.58e-08	6.66e-08	4.83e-08	1.58e-07	1.26e-07
BICRBpre	2.99e-08	1.06e-07	3.73e-08	3.82e-08	1.29e-07

FIG. 3: Comparaison de plusieurs méthodes en nombre d'itérations, en temps de calcul et en erreur relative

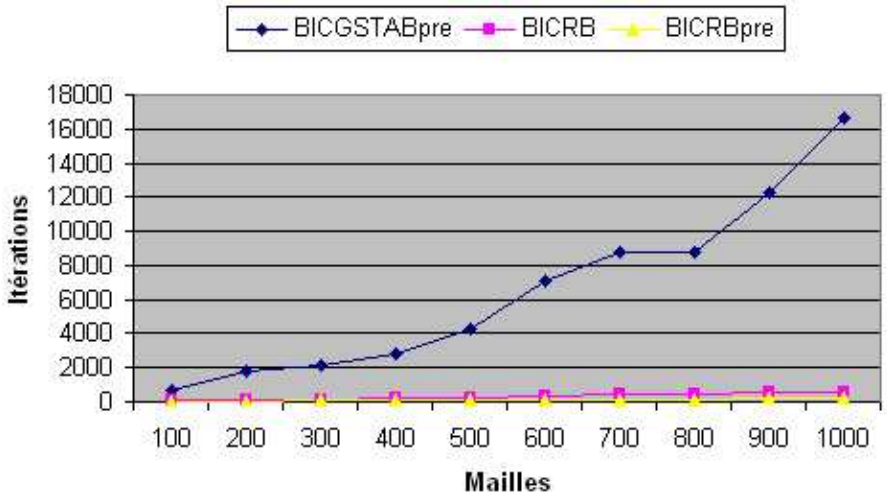

FIG. 4: Nombre d'itérations de BICGSTAB, BICRB et BICRBpre

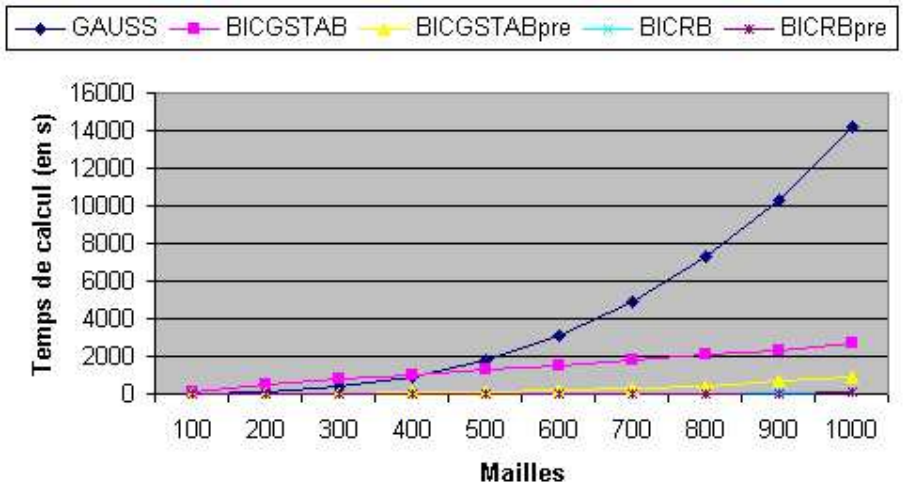

FIG. 5: Temps de calcul de GAUSS, BICGSTAB, BICGSTABpre, BICRB et BICRBpre

Conclusion

Comme on a des matrices très mal conditionnées (par exemple dans le cas horiz50 : $K(A) = 9.015766e + 13$), BICGSTAB ne donne pas de bons résultats sur la plupart des exemples indiqués.
Cependant, les solutions de BICGSTAB préconditionné et du système linéaire après élimination par R & B sont très proches de la solution exacte. Le gain de temps, surtout avec BICRB et BICRBpre, est très important en comparaison de la résolution avec GAUSS.

CONCLUSION

On a vu que BICGSTAB et les diverses méthodes de préconditionnement sont un moyen efficace pour résoudre des grands systèmes linéaires creux. Notamment la combinaison de Red & Black puis BICGSTAB préconditionné par ILU(0) s'est montrée très efficace parce qu'elle est capable de résoudre des systèmes linéaires beaucoup plus rapidement que des solveurs directs. Il serait intéressant d'examiner d'autres préconditionnement comme ILU(p) et de tester leur influence sur la vitesse de convergence et sur l'exactitude de la solution.

Pendant les deux mois de stage à l'IFP, j'ai programmé en C/C++ l'algorithme standard de ILU(0) et la résolution du système linéaire après Red & Black. Toutes les matrices sont stockées au format CSR et les algorithmes sont adaptés au stockage des matrices creuses. Le code de l'algorithme BICGSTAB a été pris d'un site internet [2].

BIBLIOGRAPHIE

[1] Yousef Saad, «Iterative Methods for Sparse Linear Systems», 2ème Edition, Boston(2000)

[2] *http ://www.mathematik.uni-freiburg.de/IAM/Research/ porjectskr/lin_solveur*

[3] *http ://home.vrweb.de/ benjamin.bihler/bicgstab.pdf*

[4] *http ://iamlasun8.mathematik.uni-karlsruhe.de/parallel/skript*

[5] *http ://www.dorn.org/uni/sls*

[6] Pierre Kirner, «Étude comparative de différents préconditionnements par découplage dans le simulateur de réservoir ATHOS», Rapport IFP 56 442, décembre 2001

ANNEXES

ANNEXE 1 : Algorithme de ILU(0)

Décomposition de la matrice A (taille $n \times n$) en une matrice triangulaire inférieure à diagonale unité L et une matrice triangulaire supérieure U qui sont nulles où A est nulle. $CoNz(A)$ contient les cordonnées des valeurs non nulles dans A.

pour $i = 1, \ldots, n-1$
  pour $k = 0, \ldots, i-1$ et $(i,k) \in CoNz(A)$ faire :
    $a_{ik} = a_{ik}/a_{kk}$
    $pour j = k+1, \ldots, n-1$ et $(i,j) \in CoNz(A)$ faire :
      $a_{ij} = a_{ik}a_{kj}$
    Retour
  Retour
Retour

ANNEXE 2 : Algorithme de BICGSTAB

$r_0 = b - Ax_0$
r_0^* choisi arbitraiement
$p_0 = r_0$
pour $j = 0, 1, \ldots$ jusqu'à convergence faire :

$\alpha_j = \frac{(r_j, r_0^*)}{(Ap_j, r_0^*)}$

$s_j = r_j - \alpha_j A p_j$

$w_j = \frac{(As_j, s_j)}{(As_j, As_j)}$

$x_{j+1} = x_j + \alpha_j p_j + w_j s_j$

$r_{j+1} = s_j - w_j A s_j$

$\beta_j = \frac{(r_{j+1}, r_0^*)}{(r_j, r_0^*)} * \frac{\alpha_j}{w_j}$

$p_{j+1} = r_{j+1} + \beta_{j+1}(p_j - w_j A p_j)$

Retour

ANNEXE 3 : Algorithme de BICGSTAB préconditionné

$r_0 = b - Ax_0$
r_0^* choisi arbitraiement
$p_0 = r_0$
pour $j = 0, 1, \ldots$ jusqu'à convergence faire :

résoudre $P\hat{p} = p_j$

$\alpha_j = \frac{(r_j, r_0^*)}{(A\hat{p}, r_0^*)}$

$s_j = r_j - \alpha_j A\hat{p}$

résoudre $P\hat{s} = s_j$

$w_j = \frac{(A\hat{s}, \hat{s})}{(A\hat{s}, A\hat{s})}$

$x_{j+1} = x_j + \alpha_j \hat{p} + w_j \hat{s}$

$r_{j+1} = \hat{s} - w_j A\hat{s}$

$\beta_j = \frac{(r_{j+1}, r_0^*)}{(r_j, r_0^*)} * \frac{\alpha_j}{w_j}$

$p_{j+1} = r_{j+1} + \beta_{j+1}(\hat{p} - w_j A\hat{p})$

Retour