フィルインの選択に基づく改良版 ABRB 順序付け法による ICCG 法の並列化

井 上 明 彦[†] 藤 野 清 次[††]

不完全コレスキー分解つき共役勾配法は対称正定値行列を係数行列に持つ連立一次方程式の有用で効率の良い解法として広く利用されている．一方，共有メモリ型並列計算機の性能向上と普及にともない，その並列化のための技法の研究が今後ますます重要になると思われる．そこで並列化技法の 1 つである代数ブロック化赤-黒順序付け（ABRB）法に着目した．この順序付け法では，係数行列の非零要素だけを使うためフィルインが考慮されず，そのため並列化による台数効果は得ることができる反面，ICCG 法の収束性の良さが十分発揮されないときがある．そこで，ABRB 法により順序付けられた行列の非零要素パターンのブロック構造に注目し，フィルインを選択する改良版 ABRB 順序付け法を ICCG 法に適用し，数値実験によってその有効性を検証する．

Improved ABRB Ordering Algorithm Based on Selection of Fill-in for Parallelized ICCG Method

Akihiko Inoue[†] and Seiji Fujino[††]

In this paper, parallelism by the original ABRB (Algebraic Block Red Black) ordering algorithm for the ICCG method without fill-in is considered for the purpose of improvement of convergence rate of ICCG method. The blocking structure of entries of the reordered matrix by the ABRB ordering algorithm is an underlying factor for realizing the ICCG method with fill-in. Numerical experiments indicates that the proposed parallel blocking with fill-in ABRB ordering algorithm improves efficiency of the ICCG method on parallel computer with shared memory.

1. はじめに

前処理つき共役勾配（Conjugate Gradient，以下 CG と略す）法は疎で対称正定値な係数行列を持つ連立一次方程式の有効な解法として知られる．特に行列が M-行列のとき，不完全コレスキー（Incomplete Cholesky）分解つき CG（以下，ICCG と略す）法が非常に有効であることが知られている[14]．また，共有メモリ型並列計算機の性能向上と普及にともなって，（前進，後退）代入計算をともなう ICCG 法の並列化に関して様々な研究がなされてきた[1),21)]．ここでは，以下の主な 4 つの並列化技法の特徴を各々記述し，ICCG 法の並列化について概観する．

(1) レベルに基づくフィルインの選択

(2) 閾値に基づくフィルインの選択

(3) 選択的（selective）ブロック化によるフィルインの選択

(4) 要素番号の並べ替え（カラーオーダリング）

第 1 の「レベルに基づくフィルインの選択」においては，不完全分解中に現れるフィルインを分解過程のレベルに応じて，前処理行列の要素として採用あるいは棄却を行う方法である．これは ICCG 法の開発初期の段階からある考え方で，たとえば，元の係数行列の非零要素だけを使う分解法は IC(0) 法と呼ばれ，レベルの深さに応じて，IC(1) 法，IC(2) 法などがよく用いられる．プログラムの実装の容易さという特長を有する反面，CG 法の収束性向上には自ずと限界があるとされる[1),14),21)]．また最近は近似逆行列前処理への応用も研究されている[2)]．

第 2 の「閾値に基づくフィルインの選択」においては，分解過程で発生するフィルインをその大きさがあらかじめ指定した閾値より小さいとき棄却し，大きいとき前処理行列の要素として採用する方法であり，一

† 株式会社三菱電機情報ネットワーク
Mitsubishi Electric Information Network Corporation
†† 九州大学情報基盤センター
Computing and Communications Center, Kyushu University

般に様々な変種や拡張形がありよく使用される方法である[1),12)]．

第3の「選択的ブロック化によるフィルインの選択」においては，フィルインの発生するブロックや位置に応じて，いくつかの分解処理手順の中から適切なものを選択（select）するという比較的新しい考え方である[3),9),15),16),18),19)]．たとえば，文献 15)，16) では，接触問題において接触するグループごとに格子点を並べ替えて1つのグループとしてまとめ，全体行列中のその局所小行列ブロックでは完全 LU 分解する方法が研究された．この選択的ブロックにより ICCG 法の高収束性が得られたとされる．

最後に，第4の「要素番号の並べ替え（カラーオーダリング）」においては，互いにデータの依存関係がない未知変数ごとに1つの色と見なす．色の順に未知変数を並べ替えることによって，前進・後退の代入計算が各色ごとに並列化される[21)]．色数については，色数が少ない（たかだか8色程度）場合は従来から数多くの研究がなされており，この場合の利点は並列計算機での通信（同期）回数の少なさにあるとされる．一方，色数が多い（およそ 100 色程度）場合を最初に取り扱ったのは，著者の知る限り，文献 6) においてである．その後，同論文で指摘されたベクトル計算機上でベクトル長が非常に長くとれるという優れた特長が注目され，その拡張版が NEC 社 SX シリーズ機を使って次々と発表されてきた[5),17),22)]．ただ，色数が少ない場合に比べて通信（同期）回数が多くなるので，実際の実効性能は計算機への依存性が高い．

このほかにも並列版の前処理の有望な方法に Multi-Grid 法や Algebraic MultiGrid（AMG）法などがある（それらの最新成果については文献 23) に特集が組まれており，参照されたい）．

本論文で取り上げるのは，上記の主な4つの並列化技法において，第4の「要素番号の並べ替え（カラーオーダリング）」手法をベースに，第2の「閾値に基づくフィルインの選択」と本論文の最大の特徴である第3の「選択的ブロック化によるフィルインの選択」の3つの考え方を取り入れた複合的な技法である．

カラーオーダリングとしては，反復法の1反復あたりの同期回数の少なさに着目して，代数ブロック化赤–黒順序付け法を採用した[10)]．この並列化技法は，岩下らにより提案された元の係数行列の不規則分布の非零要素のブロック構造を利用した技法であり，代数ブロック化赤–黒順序付け（Algebraic Block Red-Black ordering，略して **ABRB**）法と呼ばれる[10),11)]．ABRB 法は非構造な疎行列を対象にし，行列の非零要素の行と列番号だけの情報で非零要素を並列化のためにブロック化し，本質的に逐次計算になる前進（後退）代入計算を各色でブロックごとに並列化する手法である．文献 11) では比較的小規模な並列計算機環境（原論文では “small-scale parallel computation environment” と記述されており，4 並列程度）の場合，より有用な並列化手法とされる．そこで，ABRB 法がより有効に働くと思われる計算機環境として，九州大学情報基盤センターに設置された共有メモリ型並列計算機 IBM p5 モデル 595 を選んだ．この計算機を構成するマルチチップモジュールは 8 個の POWER 5 のコアと 8 通信路（ウエイ）を持つ．そのため，8 並列の場合が最も性能が発揮するように設計されており，たとえばシステム側で用意された線形方程式に対する直接解法のライブラリでも 8 並列程度の場合が性能が良いとの報告がなされている[8)]．

本研究では，ABRB 順序付け法により変換された後の行列の非零要素パターンが持つブロック構造に注目し，行列のあるブロックの中では不完全分解の過程で発生するフィルインを選択（採用するかあるいは棄却するか）すればフィルインの影響を考慮できることを見出した．そこでフィルインを選択する改良版 ABRB 順序付け法を提案する．そして数値実験を通してその有効性を検証する．

本論文の構成は以下のとおりである．2 章で元の ABRB 順序付け法の概要およびフィルインをブロックごとに選択する改良版 ABRB 順序付け法について記述する．3 章で数値実験結果を報告し，最後に 4 章でまとめを行う．

2. 代数ブロック化赤–黒順序付け法とその改良

2.1 非零要素パターンとプロセッサへの割当て

ICCG 法では反復中の前進（後退）代入計算が本質的に逐次計算になるため，並列化するためには工夫が必要である．代数ブロック化赤–黒順序付け法[11)] も有用な並列化手法の1つである．ABRB 法は，一般の不規則疎行列を対象に行列の非零要素の行と列番号だけの情報をもとにブロック化する手法である．まず，未知変数をブロックに分け，同色のブロック間には互いに依存関係がないように，ブロックごとに赤，黒の色を割り当てる．次に，同じ色のブロックごとに未知変数を並べ替え，その結果上の代入計算が各色でブロックごとに並列化が可能になる．**図 1** に，並列度 4 の場合の，ABRB 法の適用後の係数行列の非零要素パターンとプロセッサへの割当ての模式図を示す．

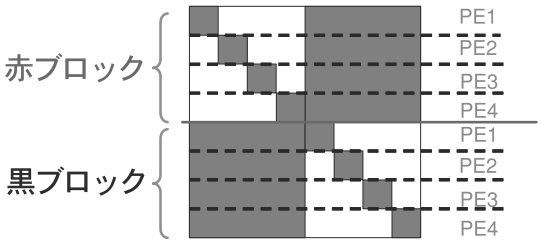


図 1 ABRB 法の適用後の係数行列の非零要素パターンの模式図（並列度 4 のとき）

Fig. 1 Diagram of nonzero entries pattern of a coefficient matrix A reordered by the ABRB ordering algorithm in case of four processors.

2.2 ABRB 法におけるブロック化と小行列

ここでは，ABRB 法における前進（後退）代入計算の手順の概要を図 1 に示した並列度 4 のときを例にとって記述する．$\tilde{A}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}}$ を ABRB 順序付け法で並べ替えられた連立一次方程式とする．解ベクトル $\tilde{\boldsymbol{x}}$ を

$$\tilde{\boldsymbol{x}} = (\tilde{\boldsymbol{x}}_{r_1}\ \tilde{\boldsymbol{x}}_{r_2}\ \tilde{\boldsymbol{x}}_{r_3}\ \tilde{\boldsymbol{x}}_{r_4}\ \ \tilde{\boldsymbol{x}}_{b_1}\ \tilde{\boldsymbol{x}}_{b_2}\ \tilde{\boldsymbol{x}}_{b_3}\ \tilde{\boldsymbol{x}}_{b_4}) \tag{1}$$

のように分割すると，係数行列 $\tilde{A}$ は，

$$\tilde{A} = \left(\begin{array}{c|c} \tilde{A}_1 & \tilde{A}_2 \\ \hline \tilde{A}_3 & \tilde{A}_4 \end{array}\right) = \left(\begin{array}{c|c} \tilde{A}_1 & \begin{matrix}\tilde{A}_{r_1,B}\\ \tilde{A}_{r_2,B}\\ \tilde{A}_{r_3,B}\\ \tilde{A}_{r_4,B}\end{matrix} \\ \hline \begin{matrix}\tilde{A}_{b_1,R}\\ \tilde{A}_{b_2,R}\\ \tilde{A}_{b_3,R}\\ \tilde{A}_{b_4,R}\end{matrix} & \tilde{A}_4 \end{array}\right) \tag{2}$$

$$\tilde{A}_1 = \begin{pmatrix} \tilde{A}_{r_1,r_1} & & & 0 \\ & \tilde{A}_{r_2,r_2} & & \\ & & \tilde{A}_{r_3,r_3} & \\ 0 & & & \tilde{A}_{r_4,r_4} \end{pmatrix} \tag{3}$$

$$\tilde{A}_4 = \begin{pmatrix} \tilde{A}_{b_1,b_1} & & & 0 \\ & \tilde{A}_{b_2,b_2} & & \\ & & \tilde{A}_{b_3,b_3} & \\ 0 & & & \tilde{A}_{b_4,b_4} \end{pmatrix} \tag{4}$$

と書ける．ここで，小行列の下付き添字 r_i，b_i は i 番目の赤ブロックおよび黒ブロックを各々表す．同様に，下小行列の付き添字 R，B は赤ブロック全体，黒ブロック全体を各々表す．式 (2) における $\tilde{A}_1$，$\tilde{A}_2$，$\tilde{A}_3$，$\tilde{A}_4$ は小行列である．ここで，式 (2)–(4) を使って以下のような小行列の和集合 φ_4 を考える．並列度は 4 とする．

$$\varphi_4 = \{\tilde{A}_{r_i,r_i}\} \cup \{\tilde{A}_{b_i,b_i}\} \cup \{\tilde{A}_{r_i,B}\} \cup \{\tilde{A}_{b_i,R}\}.$$

- 図 1 の左上小行列集合：$\{\tilde{A}_{r_i,r_i}\}$ $(r_i = 1,\ldots,4)$
- 同図の右上小行列集合：$\{\tilde{A}_{r_i,B}\}$ $(r_i = 1,\ldots,4)$
- 同図の左下小行列集合：$\{\tilde{A}_{b_i,R}\}$ $(b_i = 1,\ldots,4)$
- 同図の右下小行列集合：$\{\tilde{A}_{b_i,b_i}\}$ $(b_i = 1,\ldots,4)$

もっと並列度が多い一般に並列度が m のときも小行列の和集合 φ_m は同様に記述できる．

2.3 ABRB 法における不完全コレスキー分解とブロック化によるフィルインの選択

不完全コレスキー（IC）分解は，元の係数行列 $A(=(a_{ij})) \in R^{n\times n}$ を以下のように不完全分解する方法である．ここで，$U = (u_{ij})$ は上三角行列，R は狭義上三角行列，上付き添え字 T は行列の転置を各々表す．

$$A = U^T U - R - R^T. \tag{5}$$

行列 R は分解の不完全さを表し，分解途中で棄却された U の要素を表す．行列 U の要素 u_{ij} を求めるアルゴリズムを以下に示す．ここでは，ブロックごとにフィルインを採用するかまたは棄却するか選択することになる．

for $i = 1,\ldots,n$

for $j = i+1,\ldots,n$

$$a^*_{ij} = a_{ij} - \sum_{k=1}^{i-1} u_{ik}u_{jk} \tag{6}$$

end for

$$u_{ii} = \left(a_{ii} - \sum_{k=1}^{i-1} u^2_{ik}\right)^{1/2}, \tag{7}$$

$$u_{ij} = \begin{cases} a^*_{ij}/u_{ii} & \textbf{for}\ \ (j < i \le n) \wedge (i,j) \\ & \quad \in \text{blocks with } \textbf{adopted} \text{ fillin} \\ 0 & \textbf{for}\ \ (j < i \le n) \wedge (i,j) \\ & \quad \in \text{blocks with } \textbf{rejected} \text{ fillin} \end{cases} \tag{8}$$

end for

上記のアルゴリズムにおいて，(1) 元の ABRB 法および (2) 本論文で提案する改良版 ABRB 法（フィルインをブロックごとに選択する）では以下のように処理内容が異なる．

(1) 元の ABRB 法では，すべてのフィルインは棄却する．すなわち，前処理行列の要素として採用しない．

(2) 一方，本論文の ABRB 法では，発生したフィルインの添字がこれらの小行列の添字の動く範囲に含まれる場合，前処理行列の要素として採用し，含まれない場合は棄却する．

なお，後述の数値実験では，分解を行う前に対角要素の値を $a_{ii} = \gamma a_{ii}$ $(\gamma > 1.0)$ とおくことで分解の破

綻を防ぐ IC 分解（以後，加速係数つき IC 分解と呼ぶ）を用いる．

2.4 ABRB 法における前進（後退）代入計算

ここでは，ABRB 法における前進（後退）代入計算について考える．係数行列 $\tilde{A}$ の不完全分解行列を $\tilde{L}$ とし，並列度は 4 の場合を考える．

$$\tilde{L} = \left(\begin{array}{c|c} \tilde{L}_1 & \mathbf{0} \\ \hline \tilde{L}_3 & \tilde{L}_4 \end{array}\right) = \left(\begin{array}{c|c} \tilde{L}_1 & 0 \\ \hline \begin{array}{c}\tilde{L}_{b_1,R}\\ \tilde{L}_{b_2,R}\\ \tilde{L}_{b_3,R}\\ \tilde{L}_{b_4,R}\end{array} & \tilde{L}_4 \end{array}\right) \tag{9}$$

$$\tilde{L}_1 = \begin{pmatrix} \tilde{L}_{r_1,r_1} & & & 0 \\ & \tilde{L}_{r_2,r_2} & & \\ & & \tilde{L}_{r_3,r_3} & \\ 0 & & & \tilde{L}_{r_4,r_4} \end{pmatrix} \tag{10}$$

$$\tilde{L}_4 = \begin{pmatrix} \tilde{L}_{b_1,b_1} & & & 0 \\ & \tilde{L}_{b_2,b_2} & & \\ & & \tilde{L}_{b_3,b_3} & \\ 0 & & & \tilde{L}_{b_4,b_4} \end{pmatrix} \tag{11}$$

ただし，行列 $\tilde{A}$ は次のように分解される．

$$\tilde{A} \simeq \tilde{L}\tilde{L}^T \tag{12}$$

このとき，前進代入計算 $\tilde{L}\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{r}}$ は以下のように行うことができる．まず，赤ブロック（$r_i = 1, \ldots, 4$）に対する代入計算は，

$$\begin{aligned} &\tilde{\boldsymbol{y}}_{r_1} = \tilde{L}^{-1}_{r_1,r_1}\tilde{\boldsymbol{r}}_{r_1},\quad \tilde{\boldsymbol{y}}_{r_2} = \tilde{L}^{-1}_{r_2,r_2}\tilde{\boldsymbol{r}}_{r_2},\\ &\tilde{\boldsymbol{y}}_{r_3} = \tilde{L}^{-1}_{r_3,r_3}\tilde{\boldsymbol{r}}_{r_3},\quad \tilde{\boldsymbol{y}}_{r_4} = \tilde{L}^{-1}_{r_4,r_4}\tilde{\boldsymbol{r}}_{r_4} \end{aligned} \tag{13}$$

で与えられる．各プロセッサはいくつかの赤ブロックに対する代入計算 (13) を同時に並列して行う．次に，黒ブロック ($b_i = 1, \ldots, 4$) に対する代入計算

$$\begin{aligned} \tilde{\boldsymbol{y}}_{b_1} &= \tilde{L}^{-1}_{b_1,b_1}(\tilde{\boldsymbol{r}}_{b_1} - \tilde{L}_{b_1,r_1}\tilde{\boldsymbol{y}}_{r_1}),\\ \tilde{\boldsymbol{y}}_{b_2} &= \tilde{L}^{-1}_{b_2,b_2}(\tilde{\boldsymbol{r}}_{b_2} - \tilde{L}_{b_2,r_2}\tilde{\boldsymbol{y}}_{r_2}),\\ \tilde{\boldsymbol{y}}_{b_3} &= \tilde{L}^{-1}_{b_3,b_3}(\tilde{\boldsymbol{r}}_{b_3} - \tilde{L}_{b_3,r_3}\tilde{\boldsymbol{y}}_{r_3}),\\ \tilde{\boldsymbol{y}}_{b_4} &= \tilde{L}^{-1}_{b_4,b_4}(\tilde{\boldsymbol{r}}_{b_4} - \tilde{L}_{b_4,r_4}\tilde{\boldsymbol{y}}_{r_4}) \end{aligned} \tag{14}$$

がブロック単位で並列に行われる．後退代入計算についても同様である．赤ブロック数と黒ブロック数が 4 よりも多い一般の場合も同様に表せる．

2.5 模式図による選択的ブロックによるフィルインの選択

ABRB 順序付け法により要素番号を並べ替えられた行列の非零要素の値をそのまま使用する IC(0) 分解では，解くべき問題によっては ICCG 法の収束性の良さがあまり発揮できないときがある．そこで，不完全分解過程で発生するフィルインを選択して用いる ABRB 順序付け法について考える．

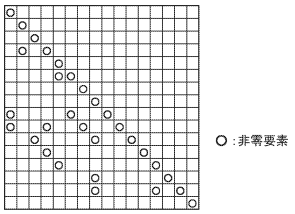


図 2 ABRB 法により並べ替えられた行列の非零要素（“○” 印）の分布
Fig. 2 A pattern of nonzero entries (“○” symbols) reordered by the ABRB ordering.

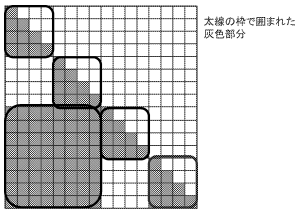


図 3 ABRB 法によって並べ替えられた非零要素の典型的なパターンの模式図（並列度 2 のとき）
Fig. 3 A typical pattern of nonzero entries reordered by the ABRB ordering in case of two threads.

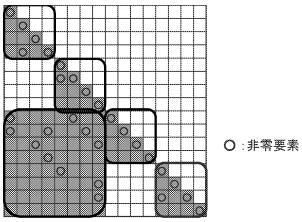


図 4 ABRB 法による変換後の行列の非零要素（“○” 印）の分布（図は図 2 と図 3 の重ね合わせで表現）
Fig. 4 Distribution of nonzero entries (“○” symbols) of the ABRB ordered matrix.

図 2 に ABRB 法により並べ替えられた行列の非零要素（“○” 印）の分布を示す．また，**図 3** に並列度 2 のときの ABRB 法によって並べ替えられた非零要素の典型的なパターンの模式図を示す．**図 4** に ABRB 法による変換後の行列の非零要素（“○” 印）の分布を示す．このように，行列の非零要素はこれらの太線の枠で囲まれたブロックおよび灰色部分（行列の対称部分のみ表示）に配置される．さらに，**図 5** に元の ABRB 法では棄却するフィルイン（“●” 印）を示す．一方，**図 6** にフィルインを選択する ABRB 順序付け法におけるフィルインが発生する位置により選択される様子を示す．すなわち，フィルインが灰色部分に現れたときそのフィルインは採用され，灰色部分以外の

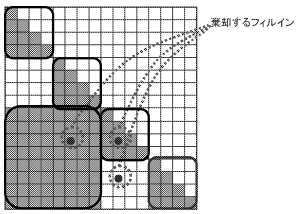


図 5 ABRB 法のときの IC(0) 分解で棄却するフィルイン（“•” 印）
Fig. 5 Rejected fill-in (“•” symbols) in the IC(0) decomposition reordered by the ABRB ordering.

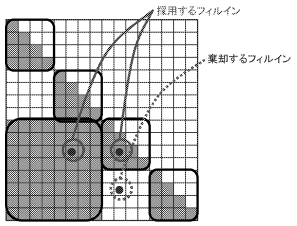


図 6 改良版 ABRB 法のとき採用するフィルイン（実線で指定された 2 個の “•” 印）と棄却するフィルイン（点線で指定された 1 個の “•” 印）の選択
Fig. 6 Selection of fill-in (adopted “•” symbols and rejected “•” symbol) in the improved ABRB ordering.

位置に現れたときそのフィルインは棄却される．したがって，たとえば，実線で指定した 2 個の “•” 印のフィルインは採用され，点線で指定した 1 個の “•” 印のフィルインは棄却されることになる．ただし，ここで表した 3 個のフィルインは採用または棄却の説明用のもので，図 2 に示した非零要素から分解の処理で直接生じたフィルインを表したものでないこと付記する．

さらに，一般に並列度 M のとき，上記の和集合に含まれるフィルインをすべて採用すると計算量が多くなるので，閾値 tol $(0 \leq \text{tol} \leq 1)$ を導入して，この閾値 tol よりもフィルインが大きい値のときのみ和集合に含まれるフィルインを前処理行列の要素として採用し，ICCG 法の収束性の改善を期待する．

今までの議論から，フィルインを選択する ABRB 順序付け法の処理手順は次のようにまとめられる．

(1) 元の係数行列に対して，通常の ABRB 順序付け法により要素番号の並べ替えを行う．
(2) 並べ替えられた行列に対して，不完全コレスキー分解の処理を行う．
(3) (2) の分解中に現れるフィルインのうち，その位置が図 3 の灰色部分に存在しかつあらかじめ設定した閾値よりも大きいとき，そのフィルインは不完全分解行列の要素として採用し，それ以外のときは棄却する．

表 1 テスト行列の主な特徴
Table 1 Characteristcs of tested matrices.

行列	次元数	非零要素/行	スレッド最大数	解析内容
Beam	10,626	22.0	8	橋梁の応力解析
Msc10848	10,848	57.2	6	自動車部品の構造解析[20]
Ct20stif	52,329	26.3	12	エンジンの応力解析[20]
Wire	104,576	17.7	16	橋梁ワイヤの応力解析
Engine	143,571	16.9	12	エンジン熱応力解析[13]
Truss	187,866	36.7	8	橋梁トラスの応力解析
Pillar	352,496	38.4	16	橋梁支柱の応力解析
RC	374,229	29.8	16	コンクリート応力解析
SP_coarse	89,586	13.7	6	重合メッシュ（粗い分割）
SP_fine	115,248	13.8	8	同上（細い分割）

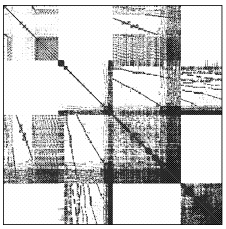


図 7 フィルインを選択する改良版 ABRB 法のとき前処理行列の要素として採用されたフィルインの分布の例（行列 Engine）
Fig. 7 Adopted fill-in in the matrix Engine reordered by the improved ABRB ordering.

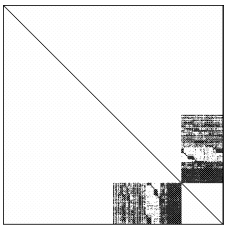


図 8 フィルインを選択する改良版 ABRB 法のとき，ブロック単位でそのブロックに含まれるフィルインがすべて棄却された例（行列 Engine）．ただし，対角線上の点列は除く
Fig. 8 Rejected fill-in in the matrix Engine reordered by the improved ABRB ordering except for diagonal.

3. 数値実験

数値実験は，九州大学情報基盤センターに設置された IBM p5 モデル 595（CPU：POWER 5，クロック周波数：1.9 GHz，OS：IBM AIX 5L）の 16 スレッドを使用して行った．コンパイラは IBM XL Fortran version 9.1 を用い，最適化オプションは-O3 -qarch=pwr5 -qtune=pwr5 -qhot，並列化ライブラリは OpenMP を用いた．計算はすべて倍精度実数演算で行った．CG 法の収束判定条件は，相対残差 l_2 ノルム：$||\boldsymbol{r}_{k+1}||_2/||\boldsymbol{r}_0||_2$ の値が 10^{-8} 以下のときとした．方程式の右辺項は行列 Msc10848 と行列 Ct20stif は真の解$\boldsymbol{x}$がすべて 1 となるように定め，それ以外の

表 2 元の ABRB 順序付け法による加速係数つき ICCG 法の数値実験結果
Table 2 Numerical results for accelerated ICCG method with the original ABRB ordering algorithm.

行列	スレッド	加速	メモリ	反復	前処理	反復	合計
		係数	[MB]	回数	時間 [s]	時間 [s]	時間 [s]
Beam	1	2.48	3.28	8102	0.08	23.1	23.2
	2	2.48	3.28	8096	0.08	13.8	13.9
	4	2.48	3.28	8115	0.08	7.09	7.17
	8	2.48	3.28	8182	0.08	3.95	4.03
Msc10848	1	2.62	7.72	1569	0.19	11.3	11.5
	2	2.48	7.72	1593	0.17	7.03	7.21
	4	2.48	7.72	1683	0.17	3.81	3.99
	6	2.58	7.72	1579	0.19	2.39	2.57
Ct20stif	1	2.44	18.7	11249	0.51	256.7	257.2
	2	2.44	18.7	11255	0.50	124.0	124.5
	4	2.44	18.7	10499	0.50	58.2	58.7
	8	2.44	18.7	11132	0.50	33.6	34.0
	12	2.44	18.7	10715	0.51	22.9	23.4
Wire	1	3.36	27.1	10545	1.2	257.	258.
	2	3.36	27.1	10545	1.2	198.	199.
	4	3.36	27.1	10527	1.2	91.4	92.6
	8	3.36	27.1	10624	1.2	45.0	51.1
	16	3.36	27.1	10774	1.2	28.2	29.4
Engine	1	1.68	36.0	1570	0.50	72.9	73.4
	2	1.68	36.0	1566	0.51	38.8	39.4
	4	1.68	36.0	1633	0.51	17.8	18.3
	8	1.68	36.0	1623	0.51	9.5	10.0
	12	1.68	36.0	1627	0.52	6.8	7.3
Truss	1	1.90	89.6	1043	2.4	158.	160.
	2	1.90	89.6	1040	2.4	92.4	94.8
	4	1.90	89.6	1033	2.3	43.2	45.4
	8	1.90	89.6	1014	2.1	18.1	20.2
Pillar	1	1.44	175	1685	2.3	567.	569.
	2	1.44	175	1690	2.2	342.	344.
	4	1.44	175	1809	2.2	162.	164.
	8	1.44	175	1846	2.1	82.7	84.8
	16	1.48	175	2003	2.2	48.5	50.7
RC	1	1.56	149	3070	2.2	771.2	773.4
	2	1.56	149	3086	2.2	466.3	468.5
	4	1.56	149	3094	2.2	220.8	223.0
	8	1.56	149	3130	2.2	121.9	124.1
	16	1.62	149	2911	2.6	59.5	62.1
SP_coarse	1	1.00	19.2	3845	ε	103.3	103.3
	2	1.00	19.2	3993	ε	52.8	52.8
	4	1.00	19.2	3971	ε	27.4	27.4
	6	1.00	19.2	4048	ε	19.7	19.7
SP_fine	1	1.00	24.8	5868	ε	211.7	211.7
	2	1.00	24.8	5997	ε	125.1	125.1
	4	1.00	24.8	6024	ε	56.2	56.2
	8	1.00	24.8	6332	ε	37.2	37.2

行列では物理的な荷重条件から得られる値を用いた．初期近似解 $\boldsymbol{x}_0$ はすべて 0 とした．最大反復回数は行列の次元数と同じ値とし，行列はあらかじめ対角項を 1 にする正規化処理をした．また，行列はあらかじめ Gibbs の初期値による RCM（Reverse Cuthill Mckee）オーダリングを施した[4),7)]．**表 1** に数値実験に用いたテスト行列の主な特徴を示す．表中で「スレッド最大数」とは RCM オーダリング後の行列に対して ABRB 法を適用したときの最大のスレッド数（=並列度）を表す．構造解析の問題から 8 個，重合メッシュの問題から 2 個，合計 10 個の行列をテストした．3 つの行列 Msc10848，Ct20stif，Engine は行列データベースから選出し，それ以外の 8 個の行列は実際の解析で使用された行列である．

表 3 改良版 ABRB 順序付け法による加速係数つき ICCG 法の数値実験結果
Table 3 Numerical results for accelerated ICCG method with the improved ABRB ordering algorithm.

行列	スレッド	加速係数	閾値	メモリ（比）[MB]	反復回数	前処理時間 [s]	反復時間 [s]	合計時間 [s]	速度向上率
Beam	1	1.02	0.01	5.7 (1.74)	1138	0.09	2.54	2.63	**8.82**
	2	1.02	0.01	5.8 (1.77)	1083	0.10	1.50	1.60	**8.69**
	4	1.02	0.01	5.9 (1.80)	1232	0.10	0.93	1.03	**6.96**
	8	1.02	0.01	5.9 (1.80)	1433	0.11	0.64	0.75	**5.37**
Msc10848	1	1.02	0.01	16.5 (2.14)	257	0.60	1.59	2.19	**5.25**
	2	1.02	0.01	17.3 (2.24)	266	0.94	1.18	2.11	**3.42**
	4	1.02	0.01	17.7 (2.29)	335	1.11	0.78	1.88	**2.12**
	6	1.02	0.01	17.8 (2.31)	270	1.11	0.41	1.52	**1.69**
Ct20stif	1	1.02	0.01	38.4 (2.05)	1987	0.85	33.7	34.5	**7.46**
	2	1.02	0.01	39.0 (2.09)	2009	0.93	20.9	21.9	**5.68**
	4	1.02	0.01	39.7 (2.12)	2275	1.06	12.8	13.9	**4.22**
	8	1.02	0.01	39.7 (2.12)	3012	1.10	8.82	9.92	**3.43**
	12	1.02	0.01	39.3 (2.10)	3221	1.12	6.64	7.76	**3.02**
Wire	1	1.02	0.01	56.1 (2.16)	1442	1.03	43.1	44.0	**5.86**
	2	1.02	0.01	56.4 (2.08)	1390	1.02	28.0	29.0	**6.86**
	4	1.02	0.01	56.8 (2.10)	1396	1.05	12.5	13.6	**6.81**
	8	1.02	0.01	57.7 (2.13)	1543	1.12	7.52	8.64	**5.94**
	16	1.02	0.01	58.3 (2.15)	1806	1.33	4.90	6.13	**4.82**
Engine	1	1.00	0.005	94.0 (2.61)	211	2.30	13.3	15.6	**4.71**
	2	1.00	0.01	83.3 (2.31)	311	1.76	9.69	11.5	**3.43**
	4	1.00	0.01	85.2 (2.37)	314	2.33	4.01	6.34	**2.89**
	*8	1.00	0.01	86.2	389	2.14	3.14	5.28	注 1
	*12	1.00	0.01	86.4	433	2.48	2.09	4.57	注 2
	8	1.00	0.05	57.9 (1.61)	978	0.58	4.36	4.94	**2.02**
	12	1.00	0.05	57.9 (1.61)	899	0.60	2.79	3.39	**2.15**
Truss	1	1.02	0.01	173 (1.93)	260	5.42	30.3	35.7	**4.48**
	2	1.02	0.01	174 (1.94)	255	5.71	18.8	24.5	**3.87**
	4	1.02	0.01	176 (1.96)	265	6.13	9.91	16.0	**2.84**
	*8	1.02	0.01	176	310	4.16	5.43	9.59	注 3
	8	1.04	0.05	121 (1.35)	660	1.40	5.83	7.23	**2.85**
Pillar	1	1.00	0.01	339 (1.94)	486	10.7	126.	137.	**4.15**
	2	1.00	0.01	340 (1.94)	490	11.1	81.2	92.3	**3.73**
	4	1.00	0.01	342 (1.95)	502	11.8	37.8	49.6	**3.31**
	8	1.00	0.01	346 (1.98)	521	13.2	20.3	33.5	**2.53**
	*16	1.00	0.01	338	617	12.9	10.0	22.9	注 4
	16	1.00	0.05	239 (1.37)	1321	3.0	12.8	15.8	**3.21**
RC	1	1.02	0.01	263 (1.77)	927	5.5	163.	168.	**4.60**
	2	1.02	0.01	264 (1.77)	922	5.6	114.	119.	**3.94**
	4	1.02	0.01	265 (1.78)	947	5.9	56.1	61.9	**3.60**
	8	1.02	0.01	267 (1.79)	1058	6.3	33.6	39.9	**3.11**
	16	1.02	0.01	264 (1.77)	1229	6.5	16.5	23.1	**2.68**
SP_coarse	1	1.02	0.005	44.3 (2.31)	1512	0.97	46.8	47.7	**2.17**
	2	1.02	0.005	44.4 (2.31)	1521	1.01	21.6	22.7	**2.33**
	4	1.02	0.005	44.4 (2.31)	1713	1.09	14.0	15.1	**1.81**
	6	1.02	0.005	45.0 (2.34)	1797	1.71	8.73	10.4	**1.89**
SP_fine	1	1.02	0.005	56.1 (2.26)	1926	1.64	79.5	81.2	**2.61**
	2	1.02	0.005	56.6 (2.28)	1928	1.72	38.6	40.4	**3.10**
	4	1.02	0.005	56.9 (2.29)	2014	2.46	18.0	20.5	**2.74**
	8	1.00	0.005	60.5 (2.44)	1878	1.74	11.9	13.6	**2.74**

3.1 改良版 ABRB 法における採用（棄却）フィルインの分布の例

図 7 にフィルインを選択した改良版 ABRB 法のとき前処理行列の要素として採用されたフィルインの分布の例（行列 Engine）を示す．同様に，**図 8** にフィルインを選択する改良版 ABRB 法のとき，ブロック単位でそのブロックに含まれるフィルインがすべて棄却されたの例（行列は同一）を示す．ただし，対角線

上の点列は行列を表示するためだけに描いたもので，棄却されたフィルインを表すものではないことに注意されたい．いずれの場合も，スレッドの数は 2，閾値は 0.01 のときの結果である．図 7 の中の行列の右下の対称な 2 つの小ブロック部分のフィルインが図 8 に示したように棄却されたことが分かる．まさにこれが小ブロック部分ごとのフィルインの選択（採用または棄却）である．

3.2 実験結果と考察

表 2 に元の ABRB 順序付け法による加速係数つき ICCG 法を，**表 3** に改良版 ABRB 順序付け法による加速係数つき ICCG 法の数値実験結果を各々示す．並列計算機での経過時間の計測には関数 gettimeofday を用いた．表中の「ε」印は計測可能な最小時間単位（0.01 秒）以下の微小な時間だったことを表す．表 3 の最右側の欄の「速度向上率」とは表 2 の元の ABRB 順序付け法による加速係数つき ICCG 法の実行時間を 1.0 としたときの改良版 ABRB 順序付け法による加速係数つき ICCG 法の実行時間の速度向上の倍率を意味する．時間の単位はすべて秒である．同様に，「メモリ（比）」は元の ABRB 順序付け法のときの使用メモリ量と改良版 ABRB 順序付け法のときの使用メモリ量およびそれらの比率を表す．加速係数つき IC 分解における加速係数はまず 1.00 に設定し（このとき IC(0) 分解に相当），対角項の値が負になったら加速係数を 0.02 ずつ増加させ，対角項の値が正になるまで繰り返した．さらに，改良版 ABRB 順序付け法による加速係数つき IC 分解における棄却値 tol の値は 0.0005，0.001，0.005，0.01，0.05 の 5 通りを調べ，最も計算時間が短いものを選んだ．表 2 と表 3 の結果から以下のことが分かる．

- 速度向上率は 8.82 倍（行列 Beam で 1 スレッドの場合）から 1.69 倍（行列 Msc10848 で 6 スレッドの場合）とすべての場合において収束速度が向上した．
- 特に構造解析の問題（合計 8 個の行列）のとき，他の解析分野の問題（合計 2 個の行列）のときに比べて全体に速度向上率が大きい．ただし，後者の問題の数が少ないので，結論づけるのはまだ早いと思われる．
- 改良版 ABRB 順序付け法のとき反復回数が大きく減った．

特に，表 3 の 3 つの行列 Engine，Truss，Pillar において，8 スレッド以上の場合，閾値が 1〜4 のスレッド数の場合と同じ値 0.01 のときの実験結果を追加記載した（表中において「注 1」から「注 4」をつけた合計 4 行を指す）．これらの結果から以下の観察や考察ができる．

- 8 スレッド以上の場合でも，反復回数の増加の程度は 1〜4 のスレッドの場合と比べて特に急なものではない．
- 8 スレッド（表中の「注 1」と「注 3」をつけた 2 行の結果）の場合，同一スレッド数の最速のときの合計時間に比べてその差はそれほど大きくはない．たとえば，行列 Engine で 8 スレッドの場合，閾値 0.01 のとき 5.28 秒，閾値 0.05 のとき 4.94 秒でその比は 1.058 であった．
- 行列 Engine と Pillar の場合，8 スレッド以上の場合の合計時間が最速のときのデータは，閾値を 0.01 から 0.05 にいずれの場合も大きくしたときに得られた．しかし，閾値を大きくしたため，収束までの反復回数は大幅に増加した．
- 行列 Truss の場合は，閾値と加速係数の値を同時に両方とも変化しているので収束性の分析は複雑で難しい．
- 一方，スレッド数がもっと多い場合，すなわち 12 および 16 スレッドの場合（表中の「注 2」と「注 4」をつけた 2 行の結果）は最速のときの合計時間と今回追加した結果の合計時間との差は拡大した．たとえば，行列 Pillar で 16 スレッドの場合，

表 4 2 つのタイプの ABRB 順序付け法による加速係数つき ICCG 法のスレッドごとの速度向上率の平均値
Table 4 Comparison of average speedup ratio of parallelized ICCG methods with two types of ABRB ordering algorithms.

スレッド数						
1	2	4	(6)	8	(12)	16
5.01	4.51	3.73	(1.79)	3.50	(2.59)	3.57

表 5 2 つのタイプの ABRB 順序付け法による加速係数つき ICCG 法のメモリ量の比の平均値
Table 5 Comparison of average ratio of used amount of memory of parallelized ICCG methods with two types of ABRB ordering algorithms.

スレッド数						
1	2	4	(6)	8	(12)	16
2.09	2.07	2.10	(2.33)	1.90	(1.86)	1.76

表 6 2 つのタイプの ABRB 順序付け法による加速係数つき ICCG 法の反復回数の比の平均値
Table 6 Comparison of average ratio of iteration counts of parallelized ICCG methods with two types of ABRB ordering algorithms.

スレッド数						
1	2	4	(6)	8	(12)	16
.209	.208	.227	(.367)	.253	(.334)	.278

閾値 0.01 のとき 22.9 秒，閾値 0.05 のとき 15.8 秒でその比は 1.449 であった．このように差が広がった原因は，使用した並列計算機の各モジュールが 8 個の POWER5 のコアと 8 個の通信路で構成されているためであると思われる．

さらに，表 2 と表 3 に示した 10 個のテスト行列に対する個別の実験結果をまとめて集計したものが**表 4**，**表 5**，**表 6** である．表 4 は 2 つのタイプの ABRB 順序付け法による加速係数つき ICCG 法のスレッドごとの平均速度向上率の比較である．同様に，表 5 に使用メモリ量の比の平均値を，表 6 に反復回数の比の平均値をスレッドごとにまとめたものを各々示す．ただし，スレッド数が 6 と 12 の場合は，いずれもサンプル数が 2 と少なかったので参考数字の意味で括弧で括った．これらの表から以下のことが分かる．

- 表 4 から，スレッド数が 1 と 2 の場合，平均速度向上率は 4.5 倍以上，一方スレッド数が 8 と 16 の場合でも同向上率は 3.5 倍以上と ICCG 法の収束性が大きく向上した．
- ただし，スレッド数が多い場合に速度向上率が小さいのは，改良版 ABRB 順序付け法のとき元の ABRB 順序付け法に比べて相対的に反復回数が若干増えたことによる．
- 表 5 から，スレッド数が 1 と 2 の場合，使用メモリ量の比の平均値は 2.1 倍程度，一方スレッド数が 8 と 16 の場合は同平均値は 1.8 倍程度であり，あまり変化がない．
- 表 6 から，反復回数の比の平均値はスレッド数が 1 と 2 の場合は 0.2 を少し超える程度の値，一方スレッド数が 8 と 16 の場合は同比が 0.3 前後の値に大きくなった．いずれの場合も反復回数は激減した．

4. ま と め

行列の非零要素だけを不完全分解で使用する ABRB 法順序付け法をもとにして，フィルインが発生するブロックの構造を利用してフィルインを選択するという改良版 ABRB 順序付け法を提案した．そして，改良版 ABRB 順序付け法を ICCG 法の並列化に適用し，その速度向上率からその有効性を確かめた．そして，今回の提案した方法は比較的小規模（8 スレッド程度）な並列度の場合に，より有効な並列化法であることを検証した．

今後の課題として，スレッド数が多くなった場合，改良版 ABRB 順序付け法による ICCG 法の反復回数の増加の割合を低く抑えることも含めて，中規模以上の並列度の場合に，より有用な ICCG 法の並列化技法の研究する必要があると思われる．また，解析分野の異なるより多くの行列に対して，改良版 ABRB 順序付け法により並列化した ICCG 法の有効性を検証する必要もある．さらに，色数がより多くなるマルチカラー技法への提案法の応用にも挑戦する予定である．

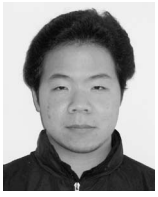
謝辞 本論文に対して貴重かつ本質的な多くの助言をいただいた匿名の査読者に心より感謝の意を表します．

参 考 文 献

1) Benzi, M.: Preconditioning techniques for large linear systems: A Survey, *J. Comput. Phys.*, Vol.189, pp.418–477 (2002).
2) Benzi, M., Marin, J. and Tuma, M.: A two-level parallel preconditioner based on sparse approximate inverse, In D. Kincaid (Ed.), *Iterative Methods in Scientific computation IV*, IMACS series in Comput. and Appl. Math., Vol.5, IMACS, pp.167–178 (1999).
3) Carpentieri, B., Duff, I. and Giraud, L.: Sparse pattern selection strategies for robust Frobenius-norm minimization preconditioners in electromagneticsm, *Numer. Linear Algebra Appl.*, Vol.7, pp.667–686 (2000).
4) Cuthill, E. and Mckee, J.: Reducing the bandwidth of sparse symmetric matrices, *Proc. 24th National Conference of the Associate for Computing Machinery*, pp.157–172, Brandon Press, NJ (1969).
5) Doi, S. and Hoshi, A.: Large-numbered multicolor MILU preconditioning on SX-3/14, *Int. J. Comput. Math.*, Vol.44, pp.143–152 (1992).
6) Fujino, S. and Doi, S.: Optimizing multicolor ICCG methods on some vector computers, *Proc. IMACS Int. Sym. on Iterative methods in Linear Algebra*, Beauwens, R.(Ed.), North-Holland, pp.349–358 (1991).
7) Gibbs, N.E., Poole, W.G. and Stockmeyer, P. K.: An algorithm for reducing the bandwidth and profile of a sparse matrix, *SIAM J. Numer. Anal.*, Vol.13, pp.236–250 (1976).
8) Gupta, A.: Recent advances in direct methods for solving unsymmetric sparse systems of linear equations, *IBM Research Report* RC22039, April, 2001, *ACM Trans. Mathematical Software*, Vol.28, pp.301–324 (2002).
9) Henon, P., Pellegrini, F., Ramet, P. and Roman, A.: Blocking issues for an efficient parallel block ILU preconditioner, *Abstracts of International Workshop Preconditioning 2005*, Atlanta U.S., May (2005).

10) 岩下武史ほか：同期点の少ない並列化 ICCG 法のためのブロック化赤–黒順序付け，情報処理学会論文誌，Vol.43，pp.893–904 (2002).
11) Iwashita, T. and Shimasaki, M.: Algebraic block red-black ordering method for parallelized ICCG solver with fast convergence and low communication costs, *IEEE Trans. Magn.*, Vol.39, pp.1713–1716 (2003).
12) Karypis, G. and Kumar, V.: Parallel threshold-based ILU factorization, Technical Report No.061, Department of Computer Science, University of Minnesota (1998).
13) Kouhia, R.: Sparse Matrices web page. http://www.hut.fi/~kouhia/sparse.html
14) Meijerink, J.A. and van der Vorst, H.A.: An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix, *Math. Comput.*, Vol.31, pp.148–162 (1977).
15) 中島研吾：GeoFEM における接触問題用安定化ソルバーの開発，日本計算工学会計算工学講演会論文集，Vol.7, pp.629–630 (2002).
16) Nakajima, K. and Okuda, H.: Parallel iterative solvers with selective blocking preconditioning for simulations of fault-zone contact, *Numer. Linear Algebra Appl.*, Vol.11, pp. 831–852 (2004).
17) 襲田 勉，丸山訓英，鷲尾 巧，土肥 俊，山田 進：非構造メッシュ用 BILU 前処理付き反復法のベクトル・並列化手法，情報処理学会論文誌：ハイパフォーマンスコンピューティングシステム，Vol.41, No.SIG 8(HPS 2)，pp.92–100 (2000).
18) Raghavan, P.: Efficient parallel triangular solution with selective inversion, *Parallel Processing Letters*, Vol.8, pp.29–40 (1998).
19) Raghavan, P., Teranishi, K. and Ng, E.: A latency tolerant hybrid sparse solver using incomplete Cholesky factorization, *Numer. Linear Algebra Appl.*, Vol.10, pp.541–560 (2003).
20) University of Florida Sparse Matrix web page. http://www.cise.ufl.edu/research/sparse/matrices/
21) van der Vorst, H.A.: *Iterative Krylov preconditionings for large linear systems*, Cambridge University Press, Cambridge (2003).
22) 鷲尾 巧：Overlapped Multicolor Preconditioning，第 23 回数値解析シンポジウム講演予稿集，pp.115–119 (1994).
23) Yavneh, I. (Ed.): Special Issue: Multigrid methods, *Numer. Linear Algebra Appl.*, Vol.11, No.2–3, (2004).

(平成 17 年 4 月 28 日受付)
(平成 17 年 8 月 15 日採録)



井上 明彦（正会員）

1980 年生．2003 年 3 月九州大学工学部電気情報工学科卒業．2005 年 3 月九州大学大学院システム情報科学府修士課程修了．2005 年 4 月（株）三菱電機情報ネットワーク入社．共役勾配法の並列化と乱数に興味を持つ．

藤野 清次（正会員）

1950 年生．1974 年京都大学理学部卒業．1993 年博士（工学），東京大学．2001 年九州大学情報基盤センター研究部教授．現在に至る．その間共役勾配法系統の反復法とその前処理の研究を行う．日本応用数理学会会員．