

フィルインの選択に基づく改良版 ABRB 順序付け法 による並列化 ICCG 法の性能評価

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な解法として広く利用されている。一方、並列計算機の性能向上と普及に伴い、並列化技法の研究が今後一層重要になる。ここでは、並列化技法の代数ブロック化赤-黒 (ABRB) 順序付け法に着目した。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 ICCG 法の本来の収束性が低下する。そこで、ABRB 法により並べ変え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、数値実験にてその有効性を検証する。

Estimation of parallelized ICCG method by means of improved ABRB ordering algorithm based on selection of fill-in

SEIJI FUJINO[†]

In this article, parallelism by the original ABRB (Algebraic Blocking 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 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 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⁶⁾。さらに、共有メモリ型並列計算機の性能向上と普及に伴って、その並列化実行をさせるために様々な技法が提案されている¹⁾⁸⁾。その一つに代数ブロック化赤-黒順序付け法がある。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 と呼ばれる⁴⁾。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。しかしながら、ABRB 法では、係数行列の非零要素のみを不完全分解において使用するため、並列化による台数効果は十分得ることが多い反面、本来の ICCG 法の収束性が低下することがある。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を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すればそ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。第 2 節で CG 法の前処理である不完全コレスキー分解とその加速について記述し、第 3 節で元の ABRB 順序付け法の概要およびフィルインを選択する改良版 ABRB 順序付け法の記述、すなわち、変換後の行列において、不完全分解の過程で発生するフィルインの影響を考慮できるブロックとそれができないブロックの選択法について記述する。第 4 節では数値実験結果を報告し、最後に第 5 節でまとめを行う。

[†] 九州大学情報基盤センター

Computing and Communications Center, Kyushu University

2. 不完全コレスキー (IC) 分解とその加速

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。

$$A = U^T U - R - R^T. \quad (1)$$

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。また、行列 U の要素の中で強制的に値を零にする指標 (i, j) の集合 φ を $\varphi = \{(i, j) \mid a_{ij} = 0\}$ とおき、以下の手順により行列 U の要素 u_{ij} を求める。

for $i = 1, \dots, n$

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

$$a_{ij}^* = a_{ij} - \sum_{k=1}^{i-1} u_{ik} u_{jk} \quad (2)$$

end for

$$u_{ii} = \left(a_{ii} - \sum_{k=1}^{i-1} u_{ik}^2 \right)^{1/2}, \quad (3)$$

$$u_{ij} = \begin{cases} a_{ij}^* / u_{ii} & \text{for } (j < i \leq n) \wedge (i, j) \notin \varphi \\ 0 & \text{for } (j < i \leq n) \wedge (i, j) \in \varphi \end{cases}$$

end for

数値実験では、分解を行う前に対角要素の値を $a_{ii} = \gamma a_{ii}$ ($\gamma > 1.0$) と置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。

3. 代数ブロック化赤-黒順序付け (ABRB) 法

3.1 非零要素パターンとプロセッサへの割り当て

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⁴⁾ も有用な並列化手法の一つである。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換え、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 のときの、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り当ての模式図を示す。

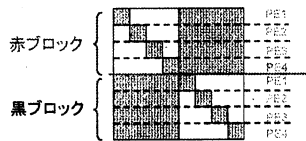


図 1 ABRB 法の適用後の係数行列の非零要素パターン (並列度 4)

3.2 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。 $\bar{A}\bar{x} = \bar{b}$ を ABRB 順序付け法で並び換えられた連立一次方程式とする。解ベクトル $\bar{x}$ を

$$\bar{x} = (\bar{x}_{r_1} \bar{x}_{r_2} \bar{x}_{r_3} \bar{x}_{r_4} \bar{x}_{b_1} \bar{x}_{b_2} \bar{x}_{b_3} \bar{x}_{b_4}) \quad (5)$$

のように分割すると、係数行列 $\bar{A}$ は、

$$\bar{A} = \left(\begin{array}{c|c} \bar{A}_1 & \bar{A}_2 \\ \hline \bar{A}_3 & \bar{A}_4 \end{array} \right) = \left(\begin{array}{c|c} \bar{A}_1 & \begin{matrix} \bar{A}_{r_1,B} \\ \bar{A}_{r_2,B} \\ \bar{A}_{r_3,B} \\ \bar{A}_{r_4,B} \end{matrix} \\ \hline \begin{matrix} \bar{A}_{b_1,R} \\ \bar{A}_{b_2,R} \\ \bar{A}_{b_3,R} \\ \bar{A}_{b_4,R} \end{matrix} & \bar{A}_4 \end{array} \right) \quad (6)$$

$$\bar{A}_1 = \begin{pmatrix} \bar{A}_{r_1,r_1} & & & 0 \\ & \bar{A}_{r_2,r_2} & & \\ & & \bar{A}_{r_3,r_3} & \\ 0 & & & \bar{A}_{r_4,r_4} \end{pmatrix} \quad (7)$$

$$\bar{A}_4 = \begin{pmatrix} \bar{A}_{b_1,b_1} & & & 0 \\ & \bar{A}_{b_2,b_2} & & \\ & & \bar{A}_{b_3,b_3} & \\ 0 & & & \bar{A}_{b_4,b_4} \end{pmatrix} \quad (8)$$

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 (6) における $\bar{A}_1, \bar{A}_2, \bar{A}_3, \bar{A}_4$ は小行列である。このとき、係数行列 $\bar{A}$ の不完全分解行列 $\bar{L}$ は次のように書ける。

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

$$\bar{L}_1 = \begin{pmatrix} \bar{L}_{r_1,r_1} & & & 0 \\ & \bar{L}_{r_2,r_2} & & \\ & & \bar{L}_{r_3,r_3} & \\ 0 & & & \bar{L}_{r_4,r_4} \end{pmatrix} \quad (10)$$

$$\bar{L}_4 = \begin{pmatrix} \bar{L}_{b_1,b_1} & & & 0 \\ & \bar{L}_{b_2,b_2} & & \\ & & \bar{L}_{b_3,b_3} & \\ 0 & & & \bar{L}_{b_4,b_4} \end{pmatrix} \quad (11)$$

ただし、行列 $\bar{A}$ は次のように分解される。

$$\bar{A} \approx \bar{L}\bar{L}^T \quad (12)$$

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

$$\begin{aligned} \bar{y}_{r_1} &= \bar{L}_{r_1, r_1}^{-1} \bar{r}_{r_1}, & \bar{y}_{r_2} &= \bar{L}_{r_2, r_2}^{-1} \bar{r}_{r_2}, \\ \bar{y}_{r_3} &= \bar{L}_{r_3, r_3}^{-1} \bar{r}_{r_3}, & \bar{y}_{r_4} &= \bar{L}_{r_4, r_4}^{-1} \bar{r}_{r_4} \end{aligned} \quad (13)$$

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 ($i = 1, \dots, 4$) に対する代入計算

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

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。また、並列度が 4 のとき、係数行列が (6)-(8) 式のように分割されるとき、非零要素の集合 P_{B_4} を小行列の集合として表現すると次のように表される。

$$P_{B_4} = \{\bar{A}_{r_i, r_i}\} \cup \{\bar{A}_{b_i, b_i}\} \cup \{\bar{A}_{r_i, B}\} \cup \{\bar{A}_{B, r_i}\}, \quad (i = 1, \dots, 4).$$

また、非零要素の複数ブロックの中に含まれる添字 (i, j) の集合として表現すると、非零要素の集合 P_{B_4} は以下のように表せる。

$$P_{B_4} = \{(i, j) \mid ((i \in r_i) \wedge (j \in r_i)) \cup ((i \in b_i) \wedge (j \in b_i)) \cup ((i \in r_i) \wedge (j \in B)) \cup ((i \in b_i) \wedge (j \in R)), i = 1, \dots, 4\}.$$

4. Fillin の選択に基づく ABRB 法

元の ABRB 法は係数行列の非零要素と同じ位置にあるフィルインだけを採用する不完全分解法であるため、解くべき問題によっては ICCG 法の収束性があまり向上しないときがある。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 法について考える。図 2 に、並列度 2 のときの ABRB 法による変換後の行列の非零要素 (図中の“○”印) の分布を示す。図 2 に不完全分解により分解中に発生したフィルイン (“●”印) および元の行列の非零要素 (“○”印) の分布を示す。ただし、ここで扱う行列が対称であることから下三角行列の部分だけを示す。元の ABRB 法では図中の“●”印の 5 つのフィルインすべてが棄却される。

一方、フィルインを選択する ABRB 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図 3 に示す。枠で囲った網掛けの部分はフィルインを採用する部分の添字 (i, j) の集合 P_{B_2} を表す。集合 P_{B_2} は以下のように定義される。

$$P_{B_2} = \{(i, j) \mid ((i \in r_i) \wedge (j \in r_i)) \cup ((i \in b_i) \wedge (j \in b_i)) \cup ((i \in r_i) \wedge (j \in B)) \cup ((i \in b_i) \wedge (j \in R)), i = 1, 2\}.$$

発生したフィルインの中で、集合 P_{B_2} に含まれるフィルイン (図中で赤の実線で指定された 3 つの“●”印) は前処理行列の要素として採用し、集合 P_{B_2} に含まれないフィルイン (図中で緑の点線で指定された 2 つの“●”印) は要素として棄却する。すなわち、フィルインの発生する位置により選択される。

さらに、一般に並列度 M のとき、上記の集合 P_{B_M} 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 τ ($0 \leq \tau \leq 1$) を導入して、或る閾値 τ よりもフィルインが大きい値のときのみ集合 P_{B_M} に含まれるフィルインを前処理行列の要素として採用する。このフィルインの選択により ICCG 法の収束性の改善を期待する。

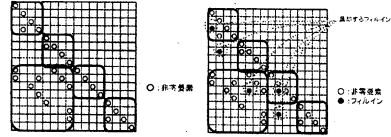


図 2 (左) ABRB 法による変換後の行列の非零要素 (“○”印) の分布 (並列度 2 のとき) (右) 変換後の行列の非零要素のうち棄却されるフィルイン (図中の 5 個の“●”印の要素が該当する)

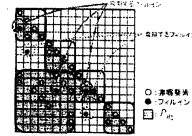


図 3 発生する位置によるフィルインの選択 (採用または棄却)

5. 数値実験

数値実験は、九州大学情報基盤センターの IBM p5 モデル 595 (CPU: POWER 5, 1.9GHz, 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 を使い、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: $\|r_{k+1}\|_2 / \|r_0\|_2$ の値が 10^{-8} 以下のときとした。方程式の右辺項は行列 “Msc10848”, “Ct20stif” は真の解 x がすべて 1 となるように定め、それ以外の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 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、行列は予め対角項を 1 にする正規化処理をした。また、行列は予め

Gibbs の初期値による RCM(Reverse Cuthill Mckee) オーダリングを施した²⁾³⁾。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、磁界解析の問題から 1 個、合計 11 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 1 テスト行列の主な特徴

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 ⁷⁾
Ct20stif	52,329	26.3	12	エンジンの応力解析 ⁷⁾
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 ⁵⁾
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)
Ccs_20	438,440	8.5	16	立方体磁性体の磁界解析

5.1 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 順序付け法のとときの使用メモリ量と改良版 ABRB 順序付け法のと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わかる。

- 速度向上率は 8.82 倍 (BEAM: 1 スレッドのとき) から 1.69 倍 (Msc10848: 6 スレッドのとき) とすべての場合で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 3 個の行列) のときに比べて全体に速度向上率が大きい。

- 改良版 ABRB 法のととき反復回数が大きく減った。さらに、表 2 と表 3 に示した 11 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 4-6 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。表から以下のことがわかる。
- 表 4 から、スレッド数が 1 と 2 のとき平均速度向上率は 4 倍以上、一方スレッド数が 8 と 16 のときでも同向上率は 3 倍以上と ICCG 法の収束性が大きく向上したことがわかる。
- また、スレッド数が多いとき速度向上率が小さいのは、改良版 ABRB 法のととき元の ABRB 法に比べて相対的に反復回数が増えたことによる。これは、スレッド数が多いとき程行列の対角ブロック部分、すなわち集合 $\{(i, j) \mid (i \in r_i) \wedge (j \in r_i)\}$ と $\{(i, j) \mid (i \in b_i) \wedge (j \in b_i)\}$ に含まれるフィルイン数が減少したためと思われる。
- 表 5 から、スレッド数が 1(2) のとき使用メモリ量の比の平均値は約 2.1 倍、一方スレッド数が 8(16) のときは約 1.8 倍であり変化が少ない。
- 表 6 から、反復回数の比の平均値はスレッド数が 1 と 2 のときは約 0.2、一方スレッド数が 8 と 16 のときは同比が約 0.4 と大きくなった。いずれの場合も反復回数は激減した。

表 4 スレッドごとの速度向上率の平均値

スレッド数						
1	2	4	(6)	8	(12)	16
4.75	4.32	2.98	(1.94)	3.36	(2.59)	3.12

表 5 使用メモリ量の比の平均値

スレッド数						
1	2	4	(6)	8	(12)	16
2.16	2.05	2.08	(2.19)	1.86	(1.86)	1.81

表 6 反復回数の比の平均値

スレッド数						
1	2	4	(6)	8	(12)	16
.210	.234	.382	(.693)	.446	(.427)	.398

6. ま と め

行列の非零要素だけを不完全分解で使用する ABRB 法順序付け法を基にして、フィルインが発生するブロッ

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

行列	スレッド	加速 係数	メモリ [MB]	反復 回数	前処理 時間 [s]	反復 時間 [s]	合計 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
	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.
	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
	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.
	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
	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	521	ϵ	63.4	63.4
	2	1.00	19.2	519	ϵ	49.1	49.2
	4	1.00	19.2	560	ϵ	26.5	26.5
	6	1.00	19.2	521	ϵ	19.2	19.2
SP_fine	1	1.00	24.8	521	ϵ	126.	126.
	2	1.00	24.8	519	ϵ	112.	112.
	4	1.00	24.8	560	ϵ	50.7	50.7
	8	1.00	24.8	515	ϵ	36.0	36.0
Ccs20	1	1.86	67.8	975	1.6	80.8	82.4
	4	1.86	67.8	973	1.6	23.6	25.1
	8	1.86	67.8	971	1.5	10.7	12.2
	16	1.86	67.8	1020	1.5	7.0	8.5

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検証した。

参 考 文 献

- 1) Benzi, M.: Preconditioning techniques for large linear systems : A Survey, *J. of Comput. Physics*, Vol. 189, pp. 418-477 (2002).
- 2) Cuthill, E., Mckee, J.: Reducing the band-

width of sparse symmetric matrices, The Proc. of 24th National Conference of the Associate for Computing Machinery, Brandon Press, NJ, pp.157-172 (1969).

- 3) Gibbs, N. E., Poole, W. G., 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).
- 4) Iwashita, T., Shimasaki, M.: Algebraic block red-black ordering method for parallelized

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

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
	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
	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
	4	1.00	0.01	85.2 (2.37)	314	2.33	4.01	6.34	2.89
	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	1.04	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
	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	0.05	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
	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)	143	1.0	32.9	33.8	1.88
	2	1.02	0.005	44.4 (2.31)	137	1.0	21.1	22.1	2.23
	4	1.02	0.005	44.4 (2.31)	608	1.1	11.7	12.8	1.79
	6	1.00	0.01	39.8 (2.07)	633	0.6	8.2	8.8	2.18
SP_fine	1	1.02	0.005	56.1 (2.26)	143	1.7	57.7	59.4	2.12
	2	1.02	0.005	56.6 (2.28)	137	1.7	39.2	40.9	2.74
	4	1.02	0.005	56.9 (2.29)	608	2.5	17.6	20.1	2.52
	8	1.00	0.01	51.2 (2.06)	645	0.9	11.9	12.8	2.81
Ccs20	1	1.02	0.01	191 (2.81)	163	5.0	22.9	27.9	2.95
	4	1.06	0.05	126 (1.86)	285	2.0	7.2	9.2	2.73
	8	1.06	0.05	128 (1.89)	294	2.2	3.4	5.6	2.18
	16	1.06	0.05	131 (1.93)	327	2.5	2.3	4.8	1.77

ICCG solver with fast convergence and low communication costs, *IEEE Trans. Magn.*, Vol. 39, pp. 1713–1716 (2003).

5) Kouhia, R., Sparse Matrices web page: <http://www.hut.fi/~kouhia/sparse.html>

6) Meijerink, J. A., 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).

7) University of Florida Sparse Matrix web page: <http://www.cise.ufl.edu/research/sparse/matrices/>

8) van der Vorst, H.A.: Iterative Krylov preconditionings for large linear systems, Cambridge University Press, Cambridge, 2003.