2006－HPC－107（47）
2006／8／2

ICCG 法のデュアルコア PC 上での並列性能評価

染原 一仁 *　南里 豪志 **　藤野 清次 **

* 九州大学大学院システム情報科学府　** 九州大学情報基盤センター

ICCG 法は本質的に逐次性の高い代入計算を必要とするため，収束性を保持しかつ高い並列化性能を発揮させるには容易ではない．そのため，ICCG 法の並列化手法について数多くの考案がなされてきた．IC 分解においてプロセッサ間に跨る行列要素を無視することにより代入計算の並列化を可能とするブロック ICCG 法や係数行列の情報に基づき，未知変数を並べ替えることにより並列化を行う ABRB 法などが例として挙げられる．また，近年の並列計算機の性能向上は著しく，現在では 1 つのプロセッサパッケージに 2 つの実行コアを搭載したデュアルコアプロセッサが発表されている．そこで本論文では，デュアルコア PC にて並列化 ICCG 法の数値実験を行い，その並列効果について検討する．

Parallel performance of ICCG method on Dual-Core PC

Kazunori SOMEHARA *　Takeshi NANRI **　Seiji FUJINO **

* Graduate School of Information Science and Electrical Engineering, Kyushu University
** Computing and Communications Center, Kyushu University

The ICCG method requires successive substitutions in the iteration kernel, which makes its parallel processing difficult. Various types of parallel ICCG methods have been proposed. Block ICCG method and ABRB method are well known as useful methods. Lately, as for CPU of the performance enhancement, there have been many studies on multithreading and multi-core processor. This paper examines the performance of the parallel ICCG methods on Dual-Core Processor-based PC.

1　はじめに

大規模な疎行列を係数行列とする連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ の解法には，反復法が広く用いられている．特に，係数行列 A が対称正定値行列であるとき，共役勾配 (Conjugate Gradient，CG) 法が最も一般的な解法として知られる．さらに，CG 法の収束性を改善するための前処理手法としては不完全コレスキー分解 (Incomplete Cholesky Factorization，IC) などがよく使用され，連立一次方程式を高速に解く研究が盛んに行われている．

近年，CPU はクロック周波数の向上から 1 つのダイの中に 2 つ以上のプロセッサコアを搭載したマルチコア・プロセッサのマルチスレッディングによる性能向上が進められ，並列計算を容易に行うことができる環境が整ってきた．また，科学技術計算の分野ではスーパーコンピュータなどの導入により並列計算という概念が用いられ，ICCG 法の並列化についても数多くの考案がなされてきた [9]．その例として IC 分解においてプロセッサ間に跨る行列要素を無視することにより代入計算を同期・通信することなく並列に処理することができるブロック ICCG 法 [2] や連立一次方程式における係数行列の情報のみに基づき，未知変数を赤と黒のブロックに割り当て並べ替えることにより，前進後退代入計算を並列化することができる代数ブロック化赤-黒順序付け (Algebraic Block Red-Black ordering，ABRB) 法 [6] がある．

本論文では，ICCG 法のアルゴリズムと並列化について触れ，デュアルコアプロセッサを搭載した HPC-P4 と高性能演算サーバ IBM eServer p5 モデル 595 上における並列化 ICCG 法の数値実験結果を記載し，その考察を行う．

2　ICCG 法の並列化

解くべき方程式を $A\boldsymbol{x} = \boldsymbol{b}$ とする．ここで行列 $A = (a_{ij})$ は正則で $n \times n$ の実数対称正定値行列，$\boldsymbol{x}$，$\boldsymbol{b}$ は n 次元の解ベクトルと右辺ベクトルとする．このとき連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ を ICCG 法で解くことを考える．以下に，ICCG 法のアルゴリズムを示す．

$\boldsymbol{x}_0$ is an initial solution.

$$\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0,\ \boldsymbol{p}_0 = (LL^T)^{-1}\boldsymbol{r}_0 \tag{1}$$

for $k = 0, 1, \cdots$

$$\alpha_k = \frac{(\boldsymbol{r}_k, (LL^T)^{-1}\boldsymbol{r}_k)}{(\boldsymbol{p}_k, A\boldsymbol{p}_k)} \tag{2}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k \tag{3}$$

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k \tag{4}$$

if $||\boldsymbol{r}_{k+1}||_2/||\boldsymbol{r}_0||_2 \leq \epsilon$ stop

$$\beta_k = \frac{(\boldsymbol{r}_{k+1}, (LL^T)^{-1}\boldsymbol{r}_{k+1})}{(\boldsymbol{r}_k, (LL^T)^{-1}\boldsymbol{r}_k)} \tag{5}$$

$$\boldsymbol{p}_{k+1} = (LL^T)^{-1}\boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k \tag{6}$$

end for

$\boldsymbol{r}_k$ は残差ベクトル，$\boldsymbol{p}_k$ は探索ベクトル，ϵ は収束判定値，α_k 及び β_k は補助変数を表す．行列 L は下三角行列であり，

上付き添え字 T は転置を表す．また，行列 L は係数行列 A を IC 分解することによって得られる．アルゴリズム中の演算 $(LL^T)^{-1}\boldsymbol{r}$ は，計算量とメモリ量の観点から (LL^T) の逆行列を求めて計算するのではなく，$(LL^T)\boldsymbol{y}=\boldsymbol{r}$ を用いて逐次的に求める．この計算は前進 (後退) 代入計算と呼ばれる．ICCG 法には 1 回の反復計算で以下に示す処理が必要となる．

(a) 行列とベクトルの積　(b) ベクトルの内積

(c) ベクトルの加減　(d) 前進 (後退) 代入

ここで，ICCG 法の並列化について考える．(a) 行列とベクトルの積 $A\boldsymbol{p}$ は，行列 A を行ごとに分割し，ベクトル $\boldsymbol{p}$ を各プロセッサで共有することで，並列計算を容易に行うことができる．同様に，(b) ベクトルの内積，(c) ベクトルの加減についてもその並列化は容易である．

一方，$(LL^T)^{-1}\boldsymbol{r}$ を求めるために必要となる (d) 前進 (後退) 代入計算は，本質的に逐次計算であるため，その並列化は大変難しい．ここで $(LL^T)\boldsymbol{y}=\boldsymbol{r}$ において $\boldsymbol{z}=L^T\boldsymbol{y}$ とおくと，$\boldsymbol{z}=L^{-1}\boldsymbol{r}$ を求める前進代入計算は，

$$\begin{aligned}&\textbf{for } i=1,\cdots,n\\&\qquad z_i=\frac{1}{l_{ii}}(r_i-\sum_{k=1}^{i-1}l_{ik}z_k)\\&\textbf{end for}\end{aligned}\tag{7}$$

で表される．式 (7) を並列で実行するのは，その逐次性のため様々な工夫が必要になる．

3　前進 (後退) 代入計算の並列化手法

ここでは，ICCG 法の算法中に現れる前進 (後退) 代入計算の並列化について考える．まず，ブロック ICCG 法，ABRB 法による並列化，ABRB 法適用後の行列のブロック構造を利用してフィルインの影響を考慮し収束性を向上させた PBFRIC 分解，そしてフィルインの考慮領域を拡大した AMB 法について述べる．

3.1　ブロック ICCG 法

ブロック ICCG 法は主にバンド行列を対象とし，IC 分解において各プロセッサ間にまたがる行列要素を無視し，並列化する手法である．

ここで並列台数が M のときブロック化による IC 分解によって得られた下三角行列 L は

$$L=\begin{pmatrix}L_1 & & & \mathbf{0}\\ & L_2 & & \\ & & \ddots & \\ \mathbf{0} & & & L_M\end{pmatrix}\tag{8}$$

となり，各プロセッサでは

$$\boldsymbol{y}_i=(L_i^T)^{-1}\left\{(L_i)^{-1}\boldsymbol{r}_i\right\}\tag{9}$$

を計算する．この手法は各プロセッサ間で通信を行うことなく代入計算を並列に行うことができる．ただし，並列台数が多くなるに従って無視される行列要素が単調増加し，収束性が悪くなるという問題点もある．

3.2　代数ブロック化赤-黒順序付け (ABRB) 法

ABRB 法は，係数行列 A の非零要素の行と列番号だけの情報を元に未知変数を赤と黒の 2 色のブロックに分け，同色のブロック間では互いに依存関係のないように並べ替えることにより，各色でブロックごとに前進 (後退) 代入計算を並列化した技法である．

次に 2 並列の場合を考える．$\tilde{A}\tilde{\boldsymbol{x}}=\tilde{\boldsymbol{b}}$ を順番を並べ替えた方程式とする．このとき，解ベクトル $\tilde{\boldsymbol{x}}$ を

$$\tilde{\boldsymbol{x}}=(\tilde{\boldsymbol{x}}_{r_1}\ \tilde{\boldsymbol{x}}_{r_2}\ \tilde{\boldsymbol{x}}_{b_1}\ \tilde{\boldsymbol{x}}_{b_2})\tag{10}$$

と分割すると，行列 $\tilde{A}$ は

$$\tilde{A}=\left(\begin{array}{cc|cc}\tilde{A}_{r_1,r_1} & \mathbf{0} & \tilde{A}_{r_1,B} & \\ \mathbf{0} & \tilde{A}_{r_2,r_2} & \tilde{A}_{r_2,B} & \\ \hline \tilde{A}_{b_1,R} & & \tilde{A}_{b_1,b_1} & \mathbf{0}\\ \tilde{A}_{b_2,R} & & \mathbf{0} & \tilde{A}_{b_2,b_2}\end{array}\right)\tag{11}$$

と表せる．ただし，下付き添え字 R，B，r_j，b_j は赤ブロック全体，黒ブロック全体，j 番目の赤ブロック，j 番目の黒ブロックを各々表す．また，$\tilde{A}_{r_i,r_i}$ は解ベクトル $\tilde{x}_{r_i}$ と同ベクトル $\tilde{x}_{r_i}$ の関係を表す小行列とする．このとき，下三角行列 $\tilde{L}$ は次のように表せる．

$$\tilde{L}=\left(\begin{array}{cc|cc}\tilde{L}_{r_1,r_1} & \mathbf{0} & \mathbf{0} & \\ \mathbf{0} & \tilde{L}_{r_2,r_2} & & \\ \hline \tilde{L}_{b_1,R} & & \tilde{L}_{b_1,b_1} & \mathbf{0}\\ \tilde{L}_{b_2,R} & & \mathbf{0} & \tilde{L}_{b_2,b_2}\end{array}\right)\tag{12}$$

これより j 番目の赤ブロックにおける前進代入は

$$\tilde{L}_{r_j}^{-1}\boldsymbol{z}_{r_j}=\boldsymbol{r}_{r_j},\ \boldsymbol{z}_{r_j}=\tilde{L}_{r_j}\boldsymbol{r}_{r_j}\tag{13}$$

で与えられる．赤ブロックに対して各プロセッサは上式の前進代入計算を並列に行う．次に各プロセッサによって得られた結果をすべてのプロセッサに送信して共有し，その後，黒ブロックにおける前進代入についても，赤ブロックのときと同様に，並列して計算する．同様に，後退代入計算についても，$\boldsymbol{z}=\tilde{L}^T\boldsymbol{y}$ を用いて，黒ブロック，赤ブロックの順で並列化できる．また，各代入計算に必要となる同期回数は 1 回である．

3.3　PBFRIC 分解

前述の ABRB 法では，IC 分解過程で発生したフィルインを考慮しない．そのため，行列によっては，ICCG 法の収束性が悪くなることがある．そこで，フィルインを考慮する並列版 PBFRIC 分解が提案された [4]．この分解では行列 A を次のように分解する．

$$A=LL^T-R-R^T-D.\tag{14}$$

ここで，行列 L は下三角行列，行列 R は代入計算の並列化及び行列 L のスパース性を保つための形式的な行列を各々表す．また，行列 D は対角行列を表し，行列 L の要素が棄却されたとき，対角要素が負にならないように対角要素を修正するための行列である．

次に，並列数が M の場合を考える．ABRB 法によって並び替えられた行列 A は一般にブロック化された構造を持つ．このとき，ブロック構造に含まれる添字 (i,j) の集合を P_{B_M} とする．ABRB 法により解ベクトル $\tilde{\boldsymbol{x}}$ が式 (10) のように分割されるとき，集合 P_{B_M} は

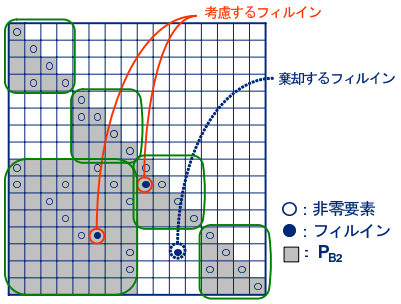

図 1: PBFRIC 分解においてフィルインを考慮する領域とフィルインをすべて棄却する領域の模式図 (2 並列の場合)

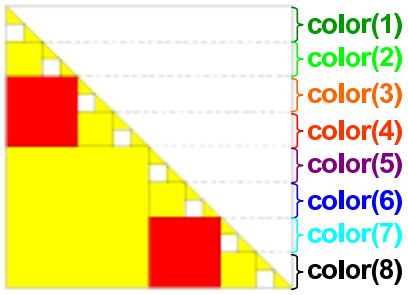

図 2: AMB 法 8 色 (2 並列の場合) のときのフィルインを考慮する領域

$$
\begin{aligned}
P_{B_M} = \quad & \{(i,j) | i \in r_k \wedge j \in r_k, k = 1, \cdots, M\} \\
& \cup \{(i,j) | i \in b_k \wedge j \in b_k, k = 1, \cdots, M\} \\
& \cup \{(i,j) | i \in R \wedge j \in B\} \\
& \cup \{(i,j) | i \in B \wedge j \in R\} \qquad (15)
\end{aligned}
$$

と表せる．PBFRIC 分解では，分解過程で現れたフィルインのうち，集合 P_{B_M} に含まれるフィルインは閾値による判定処理の後閾値より大きなフィルインだけが採用される．一方，集合 P_{B_M} に含まれないフィルインはすべて棄却される．図 1 に，2 並列の場合の PBFRIC 分解においてフィルインを考慮する領域とフィルインをすべて棄却する領域の模式図を示す．図中の ∘ 印は元の行列 A の非零要素，• 印は分解過程で発生したフィルインを各々表す．また，灰色で塗ったブロックは，式 (15) で表した集合 P_{B_2} で発生したフィルインを考慮できる領域を表す．

さらに，PBFRIC 分解におけるフィルイン l^*_{ij} に対する棄却処理手順を以下に示す．ここで，ω は乗算緩和係数，tol は閾値を，$\bar{a}_{ii}$ と $\bar{a}_{ii}$ はともに分解過程における対角要素を表す．

$$\xi = |l^*_{ij}| / \sqrt{\bar{a}_{ii}\bar{a}_{jj}} \qquad (16)$$

if $((i,j) \notin P_{B_2}) \vee (\xi \leq \mathrm{tol})$ **then**

$$l_{ij} = 0,$$

$$\bar{a}_{ii} = (1 + \omega\xi)\bar{a}_{ii} \qquad (17)$$

$$\bar{a}_{jj} = (1 + \omega\xi)\bar{a}_{jj} \qquad (18)$$

end if

3.4 代数マルチブロック (AMB) 順序つけ法

代数マルチブロック (AMB) 法は PBFRIC 分解におけるフィルインの考慮領域を拡張し，前処理行列の近似精度を高めることを目的とした並列化手法である．この手法は，ABRB 法と同様に，係数行列 A の非零要素の行と列番号だけの情報を元に未知変数をブロックに分割し，並べ替えを行う．また，AMB 法では並列計算における同期コストの軽減を図るために，すべてのブロックの大きさを揃えている．AMB 法の各ブロックの大きさ BS は色数 N_{color}，並列台数 M のとき

$$BS = n/(M \times N_{\mathrm{color}}) \qquad (19)$$

で与えられる．また，AMB 法 n 色の場合，前述のブロック構造に含まれる添字 (i,j) の集合 P_{B_M} は，

$$
\begin{aligned}
P_{B_M} \quad & = \{(i,j) | i,j \in color(1,k), k = 1, \cdots, M\} \\
& \cup \{(i,j) | i,j \in color(2,k), k = 1, \cdots, M\} \\
& \vdots \\
& \cup \{(i,j) | i,j \in color(n,k), k = 1, \cdots, M\} \\
& \cup \{(i,j) | i \in C(1) \wedge j \notin C(1)\} \\
& \cup \{(i,j) | i \in C(2) \wedge j \notin C(2)\} \\
& \vdots \\
& \cup \{(i,j) | i \in C(n) \wedge j \notin C(n)\} \qquad (20)
\end{aligned}
$$

と表される．ただし，$color(i,k)$ は k 番目の $color(i)$ ブロック，$C(i)$ は $color(i)$ ブロック全体を表すものとする．この領域拡張により，前処理行列の近似精度を高めることができ，ICCG 法の収束性の向上が期待される．図 2 に，8 並列の場合で AMB 法 4 色のときのフィルインを考慮する領域を示す．

ただし，AMB 法では ABRB 法と同様に，前進 (後退) 代入計算は各色ごとに並列化して行うことができるが，1 回の代入計算ごとに $N_{\mathrm{color}} - 1$ 回の同期を必要とする．

4 数値実験

4.1 並列計算機環境と計算条件

数値実験には 2 つの計算機を使用し，並列化ライブラリとして OpenMP を用いた．計算機の主な仕様は以下のとおりである．

1. HPC-P4
   - CPU：Intel PentiumD 3.2 GHz (Dual Core)
   - メモリ容量 ： 2 GB
   - OS ： RedHat Enterprise Linux WS 3 (linux kernel 2.4.21-37.EL-smp)
   - コンパイラ：Intel C/Fortran Compiler version 9.1

- オプション：-O3 -tpp7 -xW -openmp

2. IBM eServer p5 モデル 595
   - CPU ： IBM POWER5 1.9 GHz
   - メモリ容量 ： 48 GB
   - OS：IBM AIX 5L
   - コンパイラ：IBM XL Fortran version 9.1
   - オプション：-O3 -qarch=pwr5 -qtune=pwr5 -qsmp=omp -qhot

計算はすべて倍精度演算で行った．反復解法には CG 法を用いた．並列計算機上の経過時間の計測は関数:systemclock を用いて行った．また，表に掲載した計算時間は 3 回計測したときの平均値である．時間の単位はすべて秒とする．前処理つき CG 法における収束判定条件は，残差の 2 ノルムの比：$||\boldsymbol{r}_{k+1}||_2/||\boldsymbol{r}_0||_2$ の値が 10^{-12} 以下になったときとした．初期近似解 $\boldsymbol{x}_0$ はすべて 0 とした．CG 法における最大反復回数は行列の次元数と同じ値とした．行列はあらかじめ対角スケーリングにより対角項の値をすべて 1.0 に正規化した．AMB 法における色数は，8 色，16 色の 2 ケースを調べた．並列化手法として， ブロック ICCG 法，ABRB 法，AMB 法を用いた．それぞれの並列化手法には前処理 PBFRIC 分解を適用し，ABRB 法については前処理に VRIC(ω) 分解を用いた場合についても記した [3]．また，比較のために対角スケーリングのみを適用した DIAGCG 法と前処理に対角緩和準ロバスト IC 分解を適用した DRRICCG 法 [7] についても実験を行った．閾値 (=tol) 及び緩和係数 (=ω) は各行列に対して計算時間が最短であった場合のみ記載した．

4.2 テスト行列

数値実験でテストに用いた問題は，疎行列データベースから選出した [8] [10]．表 1 にテスト行列の主な特徴を示す．

表 1: テスト行列の主な特徴

行列	次元数	下三角の非零要素	非零要素/行	解析分野
S3DKT3M2	90,449	1,921,955	21.2	シリンダシェル
TUBE1	134,856	4,830,936	35.8	応力解析
MODEL3	374,229	11,165,692	29.8	構造解析

4.3 実験結果

表 2，表 3，表 4 には行列 S3DKT3M2 について，表 5，表 6，表 7 には行列 TUBE1 について表 8，表 9，表 10 には行列 MODEL3 について各手法における係数行列 A 及び行列 L の特徴とその性能評価を記す．表中の “Th” はスレッド数，“ω” は前処理 DRRIC，VRIC，PBFRIC 分解における乗算緩和係数，“計算時間” は係数行列の並べ替え時間，IC 分解必要とした前処理時間及び反復時間の合計を表す．“行列 A の帯幅” には ABRB 法及び AMB 法による並べ替えを行った係数行列 $\tilde{A}$ も含む．以下，DIAGCG 法は DIAG，DRRIC 分解つき CG 法は DRRIC，Block-PBFRIC 分解つき CG 法は Block，ABRB-VRIC(ω) 分解つき CG 法は ABRB(V)，ABRB-PBFRIC 分解つき CG 法は ABRB(PBF)，AMB-PBFRIC 分解つき CG 法は AMB(8)，AMB(16) と略す．AMB 法の () 内は色数を表す．ただし，DRRIC のプログラムでは並列化手法を用いていないため，前進 (後退) 代入計算については並列して処理することはできない．また，DRRIC 分解及び PBFRIC 分解で用いた閾値 (=tol) は行列 S3DKT3M2，行列 MODEL3 に対して 0.005，行列 TUBE1 に対しては 0.01 を適用した．

4.4 考察

表 2，表 8 において DIAG，DRRIC，Block についてはオーダリングを行っていないため，係数行列 A の帯幅は一定である．しかし，Block では前進後退代入計算を並列処理するために行列 L においてスレッド間に跨る要素は無視するので，DRRIC と比べて閾値や緩和係数が同じ場合でも PBFRIC 分解によって得られる行列 L の非零要素数は少なくなる．また，ABRB 及び AMB 法ではそれぞれスレッド数に応じたオーダリングを行うため，係数行列 A の帯幅はスレッド数により異なる．そのため，閾値を用いた IC 分解によって得られる行列 L の要素数も異なる．表 2 に示す AMB(8) では 1 スレッドのとき係数行列 A の帯幅は 1,229，行列 L の非零要素数は 2,752,826 となったが，2 スレッドのときは帯幅は 158,311，行列 L の非零要素数は 2,775,412 となった．

表 3，表 6，表 9 におけるデュアルコア PC 上の結果ではどの手法も 2 並列による十分な並列効果が現れなかった．特に表 6 における Block，2 スレッドでは前処理行列 LL^T の近似精度が悪くなったため，1 スレッドのときと比べると反復回数が 790 回から 1243 回に増大し，2 並列における計算時間の方が長くなった．これらの原因としてはメモリバンド幅，メモリアクセス衝突や false sharing によるキャッシュミスなどが考えられる．デュアルコア PC 上において比較すると，行列 S3DKT3M2 では ABRB(V)，行列 TUBE1 では DRRIC，行列 MODEL3 では ABRB(PBF) の計算時間が最小となり，収束性がよく，同期コストの少ない手法が有効であった．

表 4，表 7，表 10 における Power5 上での結果では各手法ともに高い並列効果が現れている．特に表 4 では DIAG，及び並列化 ICCG 法である ABRB，AMB において台数以上の並列効果が得られている．これは各スレッドに割り当てられるデータ量が減り，キャッシュメモリに保持した状態で各プロセッサにおいて実行できる演算量が増えたためと思われる．Power5 にて比較すると，行列 S3DKT3M2 と行列 MODEL3 では ABRB(PBF)，行列 TUBE1 では AMB(16) の計算時間が最小となり，収束性がよく，前進 (後退) 代入計算を並列化した手法が有効であった．また，今回用いた 3 つの行列において 2 並列では ABRB 法が有効であることが分かった．

5 まとめ

今回の 3 つの行列に対してデュアルコア PC 及び高性能演算サーバ上で OpenMP より実装した並列化 ICCG のプログラムを用いて性能評価を行った．前処理によって異なるが，2 並列では ABRB 法が収束性及び通信コストの点で有効であることが分かった．今後は，キャッシュヒット率や FLOPS 値などを用いて並列化 ICCG 法の性能評価に対して検証を行い，より効果的な並列化手法について研究していきたい．

表 2: 行列 A 及び行列 L の特徴 (行列 S3DKT3M2)

手法	ω	Th	行列 A の帯幅	行列 L の非零要素数
DIAG	-	1	1,229	-
		2		
DRRIC	0.1	1	1,229	2,752,826
		2		
Block	0.1	1	1,229	2,752,826
		2		2,724,617
ABRB(V)	0.0	1	1,229	1,921,955
		2	90,451	
ABRB(PBF)	0.1	1	1,229	2,752,826
		2	90,451	2,775,284
AMB(8)	0.1	1	1,229	2,752,826
		2	158,311	2,775,412
AMB(16)	0.1	1	1,229	2,752,826
		2	169,619	2,775,412

表 3: デュアルコアでの性能評価 (行列 S3DKT3M2)

手法	ω	Th	反復回数	計算時間	並列効果
DIAG	-	1	50316	620.26	1.00
		2	50307	572.45	1.08
DRRIC	0.1	1	2526	83.58	1.00
		2	2527	81.18	1.03
Block	0.1	1	2526	80.72	1.00
		2	3133	83.07	0.97
ABRB(V)	0.0	1	2967	77.23	1.00
		2	3239	74.71	1.03
ABRB(PBF)	0.1	1	2526	81.75	1.00
		2	2645	73.97	1.11
AMB(8)	0.1	1	2528	83.15	1.00
		2	2648	76.66	1.08
AMB(16)	0.1	1	2527	82.90	1.00
		2	2648	77.12	1.07

表 4: Power5 での性能評価 (行列 S3DKT3M2)

手法	ω	Th	反復回数	計算時間	並列効果
DIAG	-	1	50302	944.38	1.00
		2	50305	273.77	3.45
DRRIC	0.1	1	2529	133.04	1.00
		2	2528	99.48	1.34
Block	0.1	1	2529	154.32	1.00
		2	3133	79.37	1.94
ABRB(V)	0.0	1	2973	124.80	1.00
		2	3223	58.23	2.14
ABRB(PBF)	0.1	1	2528	146.93	1.00
		2	2652	67.83	2.17
AMB(8)	0.1	1	2530	135.65	1.00
		2	2650	65.50	2.07
AMB(16)	0.1	1	2528	138.90	1.00
		2	2650	65.99	2.10

表 5: 行列 A 及び行列 L の特徴 (行列 TUBE1)

手法	ω	Th	帯幅	行列 L の非零要素数
DIAG	-	1	7,853	-
		2		
DRRIC	0.2	1	7,853	4,868,856
		2		
Block	0.2	1	7,853	4,868,856
		2		4,804,800
ABRB(V)	0.3	1	7,853	4,830,936
		2	134,855	
ABRB(PBF)	0.2	1	7,853	4,868,856
		2	134,855	4,967,588
AMB(8)	0.2	1	7,853	4,874,120
		2	235,603	4,969,429
AMB(16)	0.2	1	8,361	4,888,671
		2	252,515	4,996,792

表 6: デュアルコアでの性能評価 (行列 TUBE1)

手法	ω	Th	反復回数	計算時間	並列効果
DIAG	-	1	10583	315.27	1.00
		2	10578	278.81	1.13
DRRIC	0.2	1	790	55.78	1.00
		2	785	54.11	1.03
Block	0.2	1	790	54.73	1.00
		2	1243	69.20	0.79
ABRB(V)	0.3	1	1012	66.97	1.00
		2	1242	69.79	0.96
ABRB(PBF)	0.2	1	790	56.46	1.00
		2	923	56.71	1.00
AMB(8)	0.2	1	875	61.89	1.00
		2	918	57.32	1.08
AMB(16)	0.2	1	869	61.56	1.00
		2	910	56.43	1.09

表 7: Power5 での性能評価 (行列 TUBE1)

手法	ω	Th	反復回数	計算時間	並列効果
DIAG	-	1	10567	508.52	1.00
		2	10570	272.63	1.87
DRRIC	0.2	1	791	98.64	1.00
		2	789	82.23	1.20
Block	0.2	1	787	114.63	1.00
		2	1247	83.20	1.38
ABRB(V)	0.3	1	1010	137.47	1.00
		2	1243	78.69	1.75
ABRB(PBF)	0.2	1	787	109.30	1.00
		2	920	67.36	1.62
AMB(8)	0.2	1	872	117.97	1.00
		2	921	64.98	1.82
AMB(16)	0.2	1	874	114.32	1.00
		2	909	63.49	1.80

表 8: 行列 A 及び行列 L の特徴 (行列 MODEL3)

手法	ω	Th	行列 A の帯幅	行列 L の非零要素数
DIAG	-	1	20,975	-
		2		
DRRIC	0.1	1	20,975	10,674,334
		2		
Block	0.1	1	20,975	10,674,334
		2		10,422,467
ABRB(V)	0.1	1	20,975	11,165,692
		2	374,229	
ABRB(PBF)	0.1	1	20,975	10,674,334
		2	374,229	10,759,969
AMB(8)	0.1	1	80,835	10,675,009
		2	654,353	10,750,499
AMB(16)	0.1	1	40,665	10,679,849
		2	654,689	10,752,265

表 9: デュアルコアでの性能評価 (行列 MODEL3)

手法	ω	Th	反復回数	計算時間	並列効果
DIAG	-	1	9408	657.8	1.00
		2	9406	603.5	1.09
DRRIC	0.1	1	832	136.0	1.00
		2	832	131.9	1.03
Block	0.1	1	832	134.0	1.00
		2	1105	145.5	0.92
ABRB(V)	0.1	1	2160	339.5	1.00
		2	2179	293.8	1.16
ABRB(PBF)	0.1	1	832	137.6	1.00
		2	835	119.7	1.15
AMB(8)	0.1	1	832	136.8	1.00
		2	835	121.3	1.13
AMB(16)	0.1	1	832	136.6	1.00
		2	837	122.0	1.12

表 10: Power5 での性能評価 (行列 MODEL3)

手法	ω	Th	反復回数	計算時間	並列効果
DIAG	-	1	9406	1126.7	1.00
		2	9405	579.4	1.94
DRRIC	0.1	1	832	260.4	1.00
		2	832	217.2	1.20
Block	0.1	1	832	295.7	1.00
		2	1105	193.3	1.53
ABRB(V)	0.1	1	2160	727.2	1.00
		2	2179	371.0	1.96
ABRB(PBF)	0.1	1	832	289.1	1.00
		2	835	150.3	1.92
AMB(8)	0.1	1	832	277.8	1.00
		2	836	156.3	1.78
AMB(16)	0.1	1	832	278.3	1.00
		2	837	155.6	1.79

参考文献

[1] M.A. Ajiz, A. Jennings: A robust incomplete Choleski-conjugate gradient algorithm, *Int. J. Numer. Methods Engrg.*, Vol.20, pp.949-966(1984)

[2] M.L. Barton: Three-dimensional magnetic field computation on a distributed memory parallel processor, *IEEE Trans. Magn.*, Vol.26-2, pp.834-836(1990)

[3] 井上明彦，畑崎隆雄，柿原正伸，南里豪志，藤野清次：ダブルオーダリングによる VRIC(ω)-CG 法の並列化，第 8 回環瀬戸内応用数理研究部会 シンポジウム講演予稿集，pp.113-118(2005)

[4] 井上明彦，藤野清次：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，情報処理学会論文誌コンピュータシステム Vol.46 No.SIG16 (ACS12)，Vol.46-16，pp.119-127(2005)

[5] 井上明彦，藤野清次，柿原正伸：行列の非零要素だけを使う修正 RIC(0)-CG 法の収束性について，日本計算機工学会論文集 Vol.8，pp.63-71(2006)

[6] T. Iwashita, M. Shimasaki: Algebraic Block Red-Black Ordering Method for Parallelized ICCG Solver in with Fast Convergence and Low Communication Costs, *IEEE Trans. Magn.*, Vol.39-3, pp.1713-1716(2003)

[7] 柿原正伸, 藤野清次: 緩和係数 ω を自動決定する対角緩和準ロバスト ICCG 法の収束性, 情報処理学会論文誌 コンピュータシステム, Vol.46 No.SIG4(ACS9), pp.45-55, (2005)

[8] Kouhia, R. Sparse Matrices web page: http://www.hut.fi/ kouhia/sparse.html

[9] K. Nakajima, K. Okuda, Parallel Iterative Solvers for Finite-Element Methods using a Hybrid Programming Model on SMP Cluster Architectures, RIST/Tokyo GeoFEM Report, Vol.3, pp.2-4(2003)

[10] University of Florida Sparse Matrix web page: http://www.cise.ufl.edu/research/sparse/matrices/