数理解析研究所講究録
1505巻 2006年 20-33

近似逆行列前処理と不完全コレスキ分解を融合したCG法の並列用前処理の提案

A proposal of Approximate INVerse preconditioner united with IC decomposition for the parallelism of the CG method

吉田 正浩 (Masahiro Yoshida) (九州大学大学院, 現 日本電気株式会社)
藤野 清次 (Seiji Fujino) (九州大学情報基盤センター)

1 はじめに

一般に, 共役勾配 (Conjugate Gradient : 以下, CG と略す) 法の前処理行列に望まれる性質として, 元の行列の疎 (スパース) 性の保持とその近似度の高さが上げられる. さらに, 反復計算の高い並列性を実現するには, 前処理行列のタイプも重要になってくる. 代表的な前処理の1つに, 不完全コレスキー (Incomplete Cholesky : 以下, IC と略す) 分解のように, 連立1次方程式の係数行列を三角行列どうしの積の形に不完全分解する方法がよく知られている. しかしながら, このタイプの前処理では, その計算法が本質的に持つ演算の逐次性のため, CG 法の算法中の前処理部分の並列化が非常に困難であるため今まで数多くの提案がなされてきた [11].

本論文では, IC 分解によって不完全分解された三角行列の逆行列を近似的に計算することで前処理行列として用いることを提案する [13]. 具体的には, 前処理つき CG 法の反復計算中に現れる前進 (後退) 代入計算を行列とベクトルの積の計算に置き換えられることを示す. これにより, IC 分解の手軽さと近似逆行列前処理の並列化の容易さを同時に満たす前処理が可能になる.

本論文の構成は以下のとおりである. 第2節では, 反復法の各種前処理の特徴と相互の関係について言及する. 第3節では, 前処理の並列化について述べる. ここでは, ブロック IC 分解, 近似逆行列型の前処理, そして新しい前処理の導出と考え方について記述する. 第4節では, 並列計算機を使った数値実験結果を報告する. 第5節ではまとめと今後の課題を述べる.

2 各種前処理の特徴

実数対称正定値行列 $A(=(a_{ij}))\in R^{n\times n}$ を持つ線形方程式

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

を反復法の1つである共役勾配 (Conjugate Gradient : 以下, CG と略す) 法で解くことを考える. x, b は n 次元の解ベクトルおよび右辺ベクトルとする. 特に, 行列 A が大型で非零要素が非常に少ないいわゆる疎行列の場合, 線形方程式 (2.1) は前処理つき CG 法で解かれることが多い. ここで, 前処理 (preconditioning) とは, $K\approx A^{-1}$ となるような適当な行列 K を定めることを指し, そして

$$KAx = Kb \tag{2.2}$$

のように, 変換された線形方程式を反復法で解くとき反復法の収束性が高められることがよく知られている. 具体的には, CG 法の反復計算の中に現れる前処理行列 K とベクトルとの積を計算することになり, 一般に収束までの反復回数が大幅に減少することが多い. 代表的な前処理の1つである IC 分解では, 前処理行列 K を直接求めず, その代わり次のように分解した上三角行列 $U(=(u_{ij}))$ を求める.

$$A\approx U^TU = K^{-1}. \tag{2.3}$$

ここで, 上付き添え字 "T" は転置を表す. このように行列 A を上三角行列 U とその転置 U^T の積の形で表す分解を**不完全分解型の前処理**と呼ぶ. さらに, CG 法の反復ループ中に現れる前処理行列とベクトルの積の計算は, 逆行列を経由して直接求めるのではなく, 前進代入と後退代入という二段階で順に求めることになる.

2.1 IC 分解

IC 分解は (完全) コレスキー分解を閾値などを使って不完全に分解するもので, いろいろな変形版や改良版がある. 例えば, IC 分解にフィルインの棄却 (dropping) と呼ばれる処理を加えた算法は, 前処理行列の疎性の保持と前処理行列の生成時間の短縮を主な目的に開発された処理法である. 具体的には, 生成中の前処理行列の各要素の絶対値が予め定めた閾値よりも小さいとき零とみなすものである. IC 分解の算法を以下に示す.

- **IC 分解の算法**

$$
\begin{aligned}
&\text{for } i = 1, \ldots, n \\
&\quad u_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} u_{ki}^2} \\
&\quad \text{for } j = i+1, \ldots, n \\
&\quad\quad u_{ij} = (a_{ij} - \sum_{k=1}^{i-1} u_{ki}^2)/u_{ii} \\
&\quad\quad \text{apply dropping rule} \\
&\quad \text{end for} \\
&\text{end for}
\end{aligned}
\tag{2.4}
$$

IC 分解は CG 法の前処理として最もよく用いられる手法である. しかし, まだ計算の安定化あるいは並列計算の観点から幾つかの課題が残っている. その一つが, たとえ行列が対称正定値行列の場合でも式 (2.4) の右辺の根号の中が零または負の数になり分解が途中で破綻 (breakdown) する可能性がある点である.

そこで, 破綻を防ぐために, 行列の対角項だけに定数値を加えてシフトさせるシフト IC 分解と呼ばれる分解法が考案された. シフトする量を α とおくとき, シフト IC 分解は行列表現:

$$A + \alpha I \approx U^T U \tag{2.5}$$

で表せる. ここで I は単位行列である. シフト IC 分解は適切なパラメータ α を定められれば, 前処理の破綻を回避できると同時に高い収束性をもたらすが, パラメータを適切に決めるのは一般に難しいとされる.

一方, 閾値だけを使って, 対称正定値行列に対して安定性を保証する前処理として, **RIC (Robust IC) 分解**がある. これは, フィルインの棄却を行うと同時に前処理行列の対角項に正の値を加えることで破綻を未前に防ぐ分解である. ただ, 元の RIC 分解では, 理論的保証を行いながら対角項の修正を行うために過剰に修正してしまい, 収束性の向上が見込めない場合がある. このタイプの前処理の 1 つに次に述べる RIF (Robust Incomplete Factorization) 前処理がある [4][7].

2.2 RIF 前処理

ここではRIF 前処理を簡単に導出する. グラム・シュミットの直交化の手順を用いて

$$\bar{Z}_i{}^T A \bar{Z}_j = \begin{cases} 1 & (i=j), \\ 0 & (i \neq j) \end{cases}$$

となる上三角行列 $\bar{Z}(=(\bar{z}_{ij}))$ を構成することができる. このとき, $\bar{Z}^T A \bar{Z} = I$ の関係式から

$$(\bar{Z}^T)^{-1} = A\bar{Z} \tag{2.6}$$

が成り立ち, $A\bar{Z}$ が下三角行列であることがわかる. さらに,

$$\begin{aligned} (A\bar{Z})(A\bar{Z})^T &= A\bar{Z}\bar{Z}^T A^T \\ &= A \end{aligned} \tag{2.7}$$

であるので, $\bar{U}^T = A\bar{Z}$ とおくと, $\bar{U}^T\bar{U} = A$ と分解するコレスキー分解が得られる. 以上の議論から, グラム・シュミットの直交化過程の中で行列 $\bar{Z}$ に対してフィルインの棄却を行う, という高速版の不完全分解法が得られた. この前処理は対称正定値行列に対して破綻しないことが理論的に保証されており [4], さらに必要パラメータは閾値だけであり, これもRIF 前処理の特徴の1つである. ただし, RIF 前処理は前述のIC 分解と比較して一般に前処理時間が長くかかるという弱点を持つ.

2.3 SAINV 前処理

次に, SAINV 前処理の導出について考える. $\bar{Z}^T A \bar{Z} = I$ の関係より, 行列 $\bar{Z}$ を保存すると,

$$\bar{Z}\bar{Z}^T = A^{-1} \tag{2.8}$$

とできる. すなわち, 行列 $\bar{Z}$ のフィルインを棄却しながら不完全分解し, 得られた行列を Z とするとき, ZZ^T を前処理行列 K として用いることができる. すなわち,

$$A^{-1} \approx ZZ^T \tag{2.9}$$

と近似できる. この前処理はBenzi らによって最初提案され, **SAINV** (Stabilized Approximate INVerse) **前処理**と呼ばれた [3]. このSAINV 前処理も, RIF 前処理と同じく, 行列が対称正定値行列ならば分解中に破綻を起こさないことが保証されている. 式 (2.9) のように, 三角行列とその転置行列の積で逆行列を近似する方法は一般に**近似逆行列型の前処理**と呼ばれる.

3 前処理の並列化

3.1 IC 分解の並列化

ここでは並列化の観点からIC 分解を考える. 前処理行列の作成部分およびCG 法の反復ループの中の前進 (後退) 代入計算は本質的に逐次的なアルゴリズムである. そのため, IC 分解の並列化は非常に難しいことが知られている. この特徴はIC 分解だけでなく, シフトIC 分解, RIC 分解も共通して持つ. ただし, 前処理行列の生成に要する時間は, CG 法などの反復計算に要する時間に比べて非常に少ないことが多いので, 以下では主に反復計算の並列化を中心に考えることにする.

不完全分解型で前処理行列を生成した場合, CG 法の反復計算の中で, 前進 (後退) 代入計算が必要となる. しかし, この代入計算は演算順序が逐次的であるため, その並列化は難しいことが多い. そのため, 前進 (後退) 代入計算の並列化の研究が盛んに行われてきたが, 現在でも活発に議論されている. 次に, IC 分解の並列化手法の 1 つであるブロック IC 分解について述べる.

3.2 ブロック IC 分解の手順

ブロック IC 分解は IC 分解の並列化手法として提案され, 前進 (後退) 代入の並列化が可能である [2]. 図 1 と図 2 にブロック IC 分解の手順の概念を示す. ここでは並列計算の単位をノードとして考える. 図 1 は, 4 ノードでの並列計算時の前処理行列を構成する下三角行列 U^T とノードの配置を表す. また図 2 は, 8 ノードを用いる場合の同行列 U^T とノードの配置を表す. 図中の矢印「行方向」と「列方向」は, 各々行列の行番号または列番号が増加する方向を示す. 前進 (後退) 代入の依存性をなくすため, 三角行列 U^T の複数のノードに跨る要素を零とみなし, 依存関係を消失させる. 「依存関係をなくす」とは, 図中の斜線部の領域にはフィルインを認めないこと, すなわち, この領域の値はすべて零とみなすことを意味する. このとき, 図 1 の場合に比べて図 2 の場合の方が許容されるフィルインの個数が少ないことがわかる. また, ノード数が増加すればする程, 前処理行列の近似精度が低下し, 反復回数も増加する可能性が高くなる. これがブロック IC 分解の短所とされる. 一方, 前進 (後退) 代入を各ノードが独立に実行でき, 1 回の前進代入または後退代入の間に各ノードが通信を行う必要がないという長所も持つ.

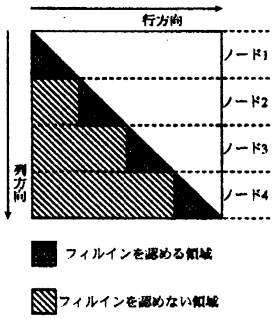

図 1: ブロック IC 分解においてフィルインを認める領域とフィルインを棄却する領域の配置 (ノード数が 4 の場合)

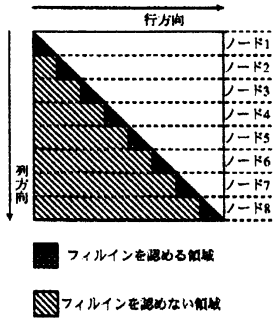

図 2: ブロック IC 分解においてフィルインを認める領域とフィルインを棄却する領域の配置 (ノード数が 8 の場合)

3.3 近似逆行列型の前処理の並列化

一方, SAINV 前処理などの近似逆行列型の前処理の場合, CG 法の反復ループ中に現れる前処理行列は行列 Z とベクトルの積の計算を行なえば求めることができる. また, 一般に行列とベクトルの積の計算は並列化が非常に容易で, 大型の行列に対しても十分な台数効果を期待できることが多い. 図 3 に行列 Z^T とベクトル $\boldsymbol{p}$ の積 $\boldsymbol{q} = Z^T\boldsymbol{p}$ の計算の場合の 4 並列のときの各ノードへの割り付けを示す. しかし, 一般に SAINV 前処理では前処理時間が長くかかり, 並列化によって反復時間が短くなるに従って逆に前処理時間の方が顕在化するという新たな問題点が出てくる.

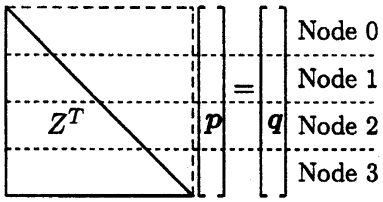

図 3: 4 並列のとき行列 Z^T とベクトル $\boldsymbol{p}$ の積 $\boldsymbol{q} = Z^T\boldsymbol{p}$ の計算

3.4 新しい前処理の導出とその並列化

ここでは, 前処理時間が SAINV よりも短くかつ反復計算中で前処理行列の計算が前進 (後退) 代入計算ではなく, 行列とベクトルの積で扱える近似逆行列分解型の新しい前処理の並列化について考える.

まず, 行列 Z と U の関係を明らかにする. 前節で述べた SAINV 前処理と RIF 前処理の議論から, $AZ \approx U^T$ とおくと, $A \approx U^TU$ の関係を使い,

$$\begin{aligned} U^TUZ &\approx U^T, \\ UZ &\approx I, \\ Z &\approx U^{-1} \end{aligned} \tag{3.10}$$

となることがわかる. したがって, 近似逆行列分解を構成する行列 Z の算出は行列 U の逆行列 U^{-1} を計算することと同じであることがわかる. したがって, 上三角行列 U の逆行列を具体的にどのように求めるかが並列化の善し悪しを決める鍵になる.

ここではガウス・ジョルダン (Gauss-Jordan, : 以下, GJ 法と略す) 法と同様の手順を採用する. ただし, GJ 法は線型方程式求解のための数値解法ではなく, あくまで逆行列算出用の数値解法と位置づける [5] [6] また, GJ 法の並列化の研究もよくなされている [9][10]. GJ 法で逆行列を求めるとき, 行列 U と逆行列 U^{-1} が各々上三角行列であることから, 作業用の行列 S を用いて, 次のように三角行列の逆行列計算の算法が得られる.

- **三角行列の逆行列計算の算法**

$$S = I \tag{3.11}$$

for $j = 1, \ldots, n$

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

    for $k = 1, \ldots, i$

$$s_{ik} = s_{ik} - \frac{u_{ij}}{u_{ii}} s_{jk} \tag{3.12}$$

end for

end for

end for

$U^{-1} = S$ (3.13)

ここで, 式 (3.11) と式 (3.13) は行列表現とする. また, 行列 U の逆行列 U^{-1} は行列 U 自体が疎行列であっても一般に密行列となるため, その生成過程でフィルインの棄却を行う. したがって, 行列 Z は厳密な逆行列でなく近似された逆行列が得られることに注意を要する. 以上から, IC 分解をベースとした近似逆行列分解を得る算法が以下のように得られる. この算法を **ICAInv**(IC Approximate Inverse) 分解と呼ぶ.

- **ICAInv 分解の算法**

for $i = 1, \ldots, n$

$$u_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} u_{ki}^2}$$

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

$$u_{ij} = (a_{ij} - \sum_{k=1}^{i-1} u_{ki}u_{kj})/u_{ii} \quad (3.14)$$

if $|u_{ij}| \leq \mathrm{tol}$ then $u_{ij} = 0$ (3.15)

end for

end for

$Z = I$

for $j = 1, \ldots, n$

$z_{ii} = 1$

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

for $k = 1, \ldots, i$

$$z_{ik} = z_{ik} - \frac{u_{ij}}{u_{ii}} z_{jk} \quad (3.16)$$

if $|z_{ik}| \leq \mathrm{tol}$ then $z_{ik} = 0$ (3.17)

end for

end for

end for

ICAInv 分解の算法中の式 (3.16) は GJ 法による行列 Z の更新部分を表す. また式 (3.15) と式 (3.17) は閾値 (=tol) によるフィルインの棄却の判定部分である. さらに, フィルイン u_{ij} を棄却するとき対角要素を修正する RIC 分解 [1] を取り入れた分解を **RICAInv** (Robust IC Approximate Inverse) 分解と呼ぶ. ここでは, 式 (3.15) においてフィルイン u_{ij} を棄却するとき, 2 つの対角要素 u_{ii}, u_{jj} を以下のように修正する.

$$
\begin{array}{l}
\text{if } (|u_{ij}| \leq \text{tol}) \text{ then } \ u_{ij} = 0 \\
\quad u_{ii} = (1 + |u_{ij}|/\sqrt{u_{ii}u_{jj}})u_{ii} \\
\quad u_{jj} = (1 + |u_{ij}|/\sqrt{u_{ii}u_{jj}})u_{jj} \\
\text{end if}
\end{array}
$$

図 4 にフィルイン u_{ij} と対応する対角要素 u_{ii}, u_{jj} の配置の様子を示す. 補償の対象は対角要素 u_{ii} だけでなく, 行列の対称性から, フィルイン u_{ji} に対する対角要素 u_{jj} も補償の対象になる.

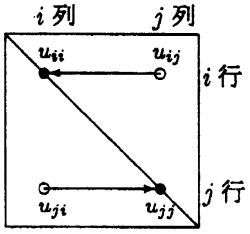

図 4: RIC 分解におけるフィルインと修正される対角要素の対応

同様に, 式 (3.17) においてフィルイン z_{ik} を棄却するとき, 2 つの対角要素 z_{ii}, z_{kk} を以下のように修正する. これらの処置により不完全分解過程での破綻が防止される.

$$
\begin{array}{l}
\text{if } (|z_{ik}| \leq \text{tol}) \text{ then } \ z_{ik} = 0 \\
\quad z_{ii} = (1 + |z_{ik}|/\sqrt{z_{ii}z_{kk}})z_{ii} \\
\quad z_{kk} = (1 + |z_{ik}|/\sqrt{z_{ii}z_{kk}})z_{kk} \\
\text{end if}
\end{array}
$$

4 数値実験

4.1 計算環境と計算条件

表 1 に計算機環境を示す. 計算はすべて倍精度演算で行なった. CG 法の収束判定条件は反復第 k 回目の残差ベクトル $\boldsymbol{r}_k$ の 2 ノルムが, $||\boldsymbol{r}_k||_2/||\boldsymbol{r}_0||_2 < 10^{-12}$ を満たしたとき収束とした. 各前処理の閾値 (=tol) は 0.05 に固定した. 閾値が 0.1 の場合も測定したが同様の傾向であったことを付記する. 初期近似解 $\boldsymbol{x}_0$ はすべて 0 とした. 係数行列は対角スケーリングを行い対角項を 1.0 に揃える正規化処理をした. 表 2 にテストに用いた 8 つの行列を示す. そのうち TUBE1-2 は R. Kouhia の疎行列データベースから, NASASRB と ENGINE は Florida 大学の疎行列データベースから選んだ [8][12]. それ以外の 5 つの行列は実際の工学の解析で現れた行列である.

4.2 並列性能評価

並列ライブラリ OpenMP を使用し, スレッド数を 1, 2, 4, 8, 16 と変化させて並列性能を調べた. スレッド数の上限を 16 としたのは計算機が設置された情報基盤センターの運用上の制約である.

表 1: 計算機環境

計算機	eServer p5 モデル 595
設置場所	九州大学情報基盤センター
CPU	Power5 (1.9GHz)
CPU 数	16
メモリ	48GB
OS	AIX 5L
コンパイラ	XL Fortran 9.1
並列化ライブラリ	OpenMP

表 2: テスト行列の特徴

行列	次元数	非零要素	非零要素/次元数	分野
TUBE1-2	21,498	459,277	21.4	応力解析 [8]
NASASRB	54,870	1,366,097	24.9	建築計算 [12]
TCASE	134,856	4,830,936	35.8	応力解析
ENGINE	143,571	2,424,822	35.8	応力解析 [12]
SBEAM	352,496	13,528,771	38.4	応力解析
MODEL3	374,229	11,165,692	29.8	応力解析
GRID_w	634,335	22,385,096	35.3	設計計算
GRID_s	838,395	13,566,835	16.2	設計計算

経過時間の測定は合計 3 度行いその平均値を表に掲載した. また経過時間の測定は Fortran90 の組み込み関数 SYSTEM_CLOCK を用いて行なった. 本研究では, 閾値以外のパラメータを必要としない 3 つの前処理, すなわち, ブロック ICCG 法 (以下, BIC-CG 法と略す) , SAInv 前処理つき CG 法 (以下, SAInv-CG 法と略す), RICAInv 前処理つき CG 法 (以下, RICAInv-CG 法と略す) の並列性能を評価した. また, 行列は対称であるため一般には下三角部分のみ保持すればよいが, 並列計算では経過時間の観点から全非零要素を保持する方が時間が短かったので, 数値実験では行列の全非零要素を行列ベクトルの積の計算で使用した.

表 3 から表 10 に各行列に対する実験結果を示す. 表中で, 「precond.」は前処理, 「th」はスレッド数, 「fillin ($\times 10^6$)」は前処理行列で使われたフィルインの個数を表す. 単位は 10^6 である. 「Itr.」は反復回数, 「pre-t」は前処理に要した時間, 「CG-t」は CG 法の反復計算の時間, 「tot-t」はそれらの合計時間, 「speedup」は台数効果を各々表す. 時間の単位はすべて秒とする. 台数効果は, スレッド数が 1 のケースの合計時間を 1.00 としたときに, 各スレッド数で並列化によって得られた速度向上の比を指す. また各表の合計時間の欄で太字の数字はスレッド数が 16 のとき経過時間が最も短かったものを指す. 表 8 の行列 MODEL3 の前処理 SAINV で, 反復回数の欄で「max」は最大反復回数までで CG 法が収束しなかったことを表す.

これらの表から以下の観察ができる.

- 16 スレッドのとき, 合計時間が最も短かったのは RICAInv 前処理のときが 4 ケース (行列 : TUBE1-2, NASASRB, TCASE, GRID_w), BIC 分解のときが 3 ケース (行列 : ENGINE, SBEAM, MODEL3) そして SAInv 前処理のときが 1 ケース (行列 : GRID_s) であった.

表 3: 行列 TUBE1-2 に対する並列版前処理つき CG 法の性能

pre-cond.	th	fillin ($\times 10^6$)	Itr.	pre-t [s]	CG-t [s]	tot-t [s]	speed-up
BIC	1	.149	4360	0.07	23.61	23.68	1.00
	2	.143	4506	0.07	12.49	12.56	1.89
	4	.137	4685	0.06	6.54	6.60	3.59
	8	.129	4971	0.06	3.72	3.78	6.27
	16	124	5266	0.06	3.11	3.17	7.47
SAInv	1	.397	2552	2.04	19.26	21.30	1.00
	2	.397	2552	2.03	9.72	11.75	1.81
	4	.397	2506	2.08	4.88	6.95	3.06
	8	.397	2554	2.05	2.60	4.65	4.58
	16	.397	2553	2.09	1.58	3.67	5.80
RIC-AInv	1	.124	5265	0.21	29.11	29.31	1.00
	2	.124	5265	0.21	14.93	15.14	1.94
	4	.124	5263	0.21	7.53	7.73	3.79
	8	.124	5263	0.21	4.06	4.27	6.87
	16	.124	5265	0.21	2.92	**3.13**	9.36

表 4: 行列 NASASRB に対する並列版前処理つき CG 法の性能

pre-cond.	th	fillin ($\times 10^6$)	Itr.	pre-t [s]	CG-t [s]	tot-t [s]	speed-up
BIC	1	.47	5987	0.2	125.7	126.0	1.00
	2	.47	5989	0.2	54.7	54.9	2.29
	4	.46	6034	0.2	25.2	25.4	4.96
	8	.45	6209	0.2	13.6	13.9	9.10
	16	.43	6516	0.2	11.1	11.3	11.12
SAInv	1	1.22	13398	7.9	468.9	476.8	1.00
	2	1.22	13391	8.1	219.6	227.6	2.09
	4	1.22	13381	7.9	82.4	90.4	5.28
	8	1.22	13370	8.3	38.7	47.0	10.14
	16	1.22	13362	7.9	20.5	28.4	16.79
RIC-AInv	1	.34	7349	0.7	157.8	158.5	1.00
	2	.34	7352	0.7	62.0	62.7	2.53
	4	.34	7349	0.7	30.0	30.7	5.17
	8	.34	7348	0.7	15.0	15.7	10.08
	16	.34	7347	0.7	8.0	**8.7**	18.18

表 5: 行列 TCASE に対する並列版前処理つき CG 法の性能

pre-cond.	th	fillin ($\times 10^6$)	Itr.	pre-t [s]	CG-t [s]	tot-t [s]	speed-up
BIC	1	1.91	3477	1.2	313.0	314.2	1.00
	2	1.89	3473	1.2	155.2	156.4	2.01
	4	1.81	4133	1.1	76.3	77.4	4.06
	8	1.73	5092	1.1	42.2	43.2	7.27
	16	1.55	6672	1.0	26.9	27.8	11.30
SAInv	1	4.79	4680	57.3	610.1	667.4	1.00
	2	4.79	4694	57.4	317.1	374.5	1.78
	4	4.79	4674	57.2	160.7	218.0	3.06
	8	4.79	4682	57.0	69.8	126.8	5.26
	16	4.79	4680	56.9	29.4	86.3	7.73
RIC-AInv	1	1.02	5272	3.6	424.9	428.5	1.00
	2	1.02	5279	3.6	220.7	224.4	1.91
	4	1.02	5271	3.6	93.2	96.9	4.42
	8	1.02	5267	3.6	41.8	45.4	9.43
	16	1.02	5278	3.6	18.3	**21.9**	19.58

表 6: 行列 ENGINE に対する並列版前処理つき CG 法の性能

pre-cond.	th	fillin ($\times 10^6$)	Itr.	pre-t [s]	CG-t [s]	tot-t [s]	speed-up
BIC	1	.99	1953	0.4	96.5	97.0	1.00
	2	.82	2506	0.4	51.3	51.7	1.88
	4	.72	2762	0.3	26.0	26.4	3.68
	8	.67	2831	0.3	13.3	13.6	7.11
	16	.63	2885	0.3	8.3	**8.6**	11.28
SAInv	1	1.56	1010	12.6	61.8	74.4	1.00
	2	1.56	1010	12.6	30.2	42.8	1.74
	4	1.56	1010	12.7	14.2	26.9	2.77
	8	1.56	1010	13.3	12.5	25.8	2.88
	16	1.56	1010	13.1	4.3	17.5	4.26
RIC-AInv	1	.78	2127	1.4	110.0	111.4	1.00
	2	.78	2126	1.4	52.7	54.1	2.06
	4	.78	2126	1.4	21.7	23.0	4.84
	8	.78	2126	1.4	12.8	14.2	7.84
	16	.78	2126	1.4	7.7	9.1	12.29

表 7: 行列 SBEAM に対する並列版前処理つき CG 法の性能

pre-cond.	th	fillin $\times 10^6$	Itr.	pre-t [s]	CG-t [s]	tot-t [s]	speed-up
BIC	1	2.38	1951	1.8	411.2	413.0	1.00
	2	2.35	1957	1.8	212.1	213.9	1.93
	4	2.29	2175	1.8	123.0	124.8	3.31
	8	2.19	2422	1.7	72.5	74.3	5.56
	16	1.98	2688	1.7	40.3	**42.0**	9.84
SAInv	1	3.83	1207	35.5	285.5	321.0	1.00
	2	3.83	1207	35.6	147.7	183.3	1.75
	4	3.83	1206	35.5	75.3	110.8	2.90
	8	3.83	1206	35.6	38.7	74.3	4.32
	16	3.83	1206	35.6	20.0	55.6	5.77
RIC-AInv	1	1.13	3111	3.3	649.7	653.0	1.00
	2	1.13	3104	3.3	326.9	330.2	1.98
	4	1.13	3101	3.3	170.2	173.5	3.76
	8	1.13	3093	3.3	87.0	90.3	7.23
	16	1.13	3093	3.3	45.0	48.3	13.52

表 8: 行列 MODEL3 に対する並列版前処理つき CG 法の性能

pre-cond.	th	fillin ($\times 10^6$)	Itr.	pre-t [s]	CG-t [s]	tot-t [s]	speed-up
BIC	1	2.23	4767	1.5	891.6	893.1	1.00
	2	2.20	4831	1.5	454.9	456.4	1.96
	4	2.16	4933	1.5	238.1	239.6	3.73
	8	2.08	5090	1.5	119.3	120.7	7.40
	16	1.90	5473	1.4	53.9	**55.2**	16.17
SAInv	1	-	max	-	-	-	-
	2	-	max	-	-	-	-
	4	-	max	-	-	-	-
	8	-	max	-	-	-	-
	16	-	max	-	-	-	-
RIC-AInv	1	1.58	5779	3.3	1066.1	1069.4	1.00
	2	1.58	5777	3.3	550.4	553.7	1.93
	4	1.58	5775	3.3	286.6	289.9	3.69
	8	1.58	5775	3.3	124.2	127.5	8.39
	16	1.58	5774	3.3	64.1	67.4	15.87

表 9: 行列 GRID_w に対する並列版前処理つき CG 法の性能

pre-cond.	th	fillin ($\times 10^6$)	Itr.	pre-t [s]	CG-t [s]	tot-t [s]	speed-up
BIC	1	5.44	4052	3.7	1542.5	1546.2	1.00
	2	5.41	4265	3.7	800.9	804.6	1.92
	4	5.38	4270	3.7	437.0	440.7	3.51
	8	5.16	4464	3.6	224.9	228.4	6.77
	16	4.97	4635	3.5	144.8	148.2	10.43
SAInv	1	9.07	1954	103.7	834.9	938.6	1.00
	2	9.07	1953	103.2	441.7	544.9	1.72
	4	9.07	1953	103.4	230.4	333.9	2.81
	8	9.07	1953	103.4	117.5	220.9	4.25
	16	9.07	1953	103.3	63.4	166.7	5.63
RIC-AInv	1	2.84	4435	8.2	1596.4	1604.6	1.00
	2	2.84	4426	8.2	781.0	789.2	2.03
	4	2.84	4427	8.1	407.9	416.0	3.86
	8	2.84	4426	8.1	210.2	218.3	7.35
	16	2.84	4426	8.2	119.7	**127.9**	12.55

表 10: 行列 GRID_s に対する並列版前処理つき CG 法の性能

pre-cond.	th	fillin ($\times 10^6$)	Itr.	pre-t [s]	CG-t [s]	tot-t [s]	speed-up
BIC	1	7.97	28578	3.4	9213.6	9217.0	1.00
	2	7.95	34436	3.4	5535.3	5538.7	1.66
	4	7.91	38448	3.4	3350.9	3354.3	2.75
	8	7.85	42203	3.4	1748.1	1751.5	5.26
	16	7.79	44668	3.3	970.1	973.4	9.47
SAInv	1	16.91	9098	131.4	3951.3	4082.7	1.00
	2	16.91	9096	127.9	1999.7	2127.5	1.92
	4	16.91	9097	128.7	1032.8	1161.6	3.51
	8	16.91	9096	128.1	534.2	662.3	6.16
	16	16.91	9096	128.1	305.7	**433.8**	9.41
RIC-AInv	1	6.89	34329	13.0	10897.0	10909.9	1.00
	2	6.89	34321	13.0	5513.9	5526.8	1.97
	4	6.89	34316	13.0	2827.0	2840.0	3.84
	8	6.89	34326	13.0	1408.7	1421.7	7.67
	16	6.89	34329	13.0	744.9	757.8	14.40

- スレッド数が増加したときの収束までの反復回数について, BIC 分解では徐々に増加するのに対して, SAInv 前処理と RICAInv 前処理ではほぼ一定回数であった.
- RICAInv 前処理で必要になった前処理行列のフィルインの個数が相対的に他の前処理のときよりも少ないことが多い.
- SAInv 前処理では表 8 のときのように収束しなかったり, 表 10 のときのように速く収束したり, 収束が非常に不安定である.

解析結果をより詳細に検討するために, 表 11 に並列版前処理つき CG 法の 16 スレッドを使用した場合の台数効果の比をまとめた. 表中の太字の数字は各行列で最も台数効果が高かったものを示す. 表 11 の結果から, RICAInv 前処理が他の 2 つの前処理に比べて台数効果が高いことがわかる.

同様に, 16 スレッドを使用した場合, 表 12 に SAInv 前処理と RICAInv 前処理において, 前処理つき CG 法における前処理時間と CG 法の反復時間の関係をまとめた. 表中の比 (ratio) は CG 法の反復時間を 1 としたときの前処理時間の比率である. また表中の太字の数字は前処理時間の方が CG 法の反復時間よりも短かった行列を表す. SAInv 前処理のときの前処理時間に比べて RICAInv 前処理のときの同時間が大幅 (約 10%前後) に短くなったことがわかる. その結果, SAInv 前処理のときのように, 多数のスレッドを使用したときに起こる前処理時間が顕在化し, 台数効果が得られないという傾向は消失した.

表 11: 並列版前処理つき CG 法の台数効果の評価 (16 スレッドのとき)

Matrix	preconditioning		
	BIC	SAInv	RICAInv
TUBE1-2	7.47	5.80	**9.36**
NASASRB	11.12	16.79	**18.18**
TCASE	11.30	7.73	**19.58**
ENGINE	11.28	4.26	**12.29**
SBEAM	9.84	5.77	**13.52**
MODEL3	**16.17**	-	15.87
GRID_w	10.43	5.63	**12.55**
GRID_s	9.47	9.41	**14.40**

5 おわりに

IC 分解を元に近似的に逆行列を計算するという並列計算用の高速な近似逆行列前処理を提案した. 提案した前処理は, 従来の SAINV 前処理に比べて, (i) 前処理に要する時間が短くなった, (ii)CG 法の収束性が安定化した, (iii) 並列台数に係わらずほぼ一定の反復回数で収束し台数効果が得やすい, などの優れた特徴を有する. 今後の課題は前処理部分の並列化である.

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).

表 12: 前処理時間と反復時間の関係 (16 スレッドのとき)

Matrix	SAInv			RICAInv		
	pre-t [s]	CG-t [s]	ratio	pre-t [s]	CG-t [s]	ratio
TUBE1-2	2.09	1.58	1.32	0.21	2.92	**0.072**
NASASRB	7.9	20.5	**0.385**	0.70	8.0	**0.088**
TCASE	56.9	29.4	1.935	3.6	18.3	**0.197**
ENGINE	13.1	4.3	3.047	1.4	7.7	**0.182**
SBEAM	35.6	20.0	1.78	3.3	45.0	**0.073**
MODEL3	-	-	-	3.3	64.1	**0.051**
GRID_w	103.3	63.4	1.629	8.2	119.7	**0.069**
GRID_s	128.1	305.7	**0.419**	13.0	744.9	**0.017**

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

[3] M. Benzi, J.K. Cullum and M. Tuma, "Robust Approximate Inverse Preconditioning for the Conjugate Gradient Method", SIAM J. Sci. Comput., Vol.22, pp. 1318-1332 (2000).

[4] M. Benzi, M. Tuma. "A Robust Incomplete Factorization Preconditioner for Positive Definite Matrices", Numerical Linear Algebra with Applications, Vol.10, pp. 385-400 (2003).

[5] P. Bogacki, "HINGES - An illustration of Gauss-Jordan reduction", MathDL: J. of Online Math. and its Appli., (2006).

[6] K. Chen, D. Evans, "An efficient variant of Gauss-Jordan type algorithms for direct and parallel solution of dense linear systems", Int. J. of Computer Math., Vol.76, pp. 387-410 (2000).

[7] 池田 優介, 藤野 清次, 柿原正伸, 井上明彦, "A-直交過程に基づく RIF 前処理の効率化について", 情報処理学会論文誌: コンピューティングシステム, Vol.46, No.SIG4(ACS9), pp. 95-104 (2005).

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

[9] N. Melab, S. Petiton and E.-G. Talbi, "A new parallel adaptive block-based Gauss-Jordan algorithm", Publication Interne AS-180, LIFL, Université de Lillel, Fév (1998).

[10] N. Melab, "A parallel adaptive Gauss-Jordan algorithm", The Journal of Supercomputing, Vol.17, pp. 167-185, (2000).

[11] Y. Saad, "Iterative methods for sparse linear systems 2nd ed.", SIAM Philadelphia (2003).

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

[13] 吉田正浩, 不完全分解の逆行列を用いた前処理つき CG 法の並列化, 九州大学大学院システム情報科学府情報工学専攻, 修士論文, 2006.3.