社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2003－HPC－95 (4)
2003／8／4

改良型安定化近似逆行列の最適な閾値の探索について

池田優介† 藤野清次 ††

安定化近似逆行列による前処理つき共役勾配法の収束性は，dropping 処理で使われる閾値と呼ばれるパラメータに大きく影響を受ける．しかし，近似分解の前に最適な閾値を知る，あるいは予見することは難しく，実際に閾値を変えて CG 法の収束の様子を調べるしかよい手段がない．本稿では，まず dropping 処理を二重に行なうことで CG 法の収束性を改善させた改良型安定化近似逆行列を提案する．次に，最適な閾値が分布する様子を分析し，計算機が持つ特性を考慮した閾値の探索法について考察する．そして，数値実験によってこれらの有効性を検証する．

A proposal of Improved Stabilized Approximate INVerse and strategy for optimal dropping tolerance value

YUSUKE IKEDA † and SEIJI FUJINO ††

Convergence of the CG method with Stabilized Approximate INVerse preconditioner is affected greatly by dropping tolerance value. It is extremely difficult, however, to foresee an optimal value in advance. Therefore it is necessary for dropping tolerance value to be chosen on the basis of a trial-and-error strategy. In this article, we propose Improved SAINV preconditioner. Moreover, we consider a strategy for choosing optimum dropping tolerance value according to characteristics of computers. Numerical results show that Improved SAINV performs well in comparison with the standard SAINV preconditioner.

1. はじめに

大規模な疎行列を係数行列 A に持つ連立一次方程式の解法には，前処理つき反復法，特に A が正定値対称行列のとき，共役勾配法 (Conjugate Gradient method，以下 CG 法と略す) がよく用いられる．そして，その前処理には不完全コレスキー (Incomplete Cholesky，以下 IC と略す) 分解が広く採用されることが多い．しかし，IC 分解が有効なのは M-行列に限られ，それ以外の行列では CG 法の収束性が悪化する場合が報告されている[4]．また，その計算過程に逐次処理が含まれるため並列化も困難が伴うことが多い．

そこで，本研究では，近似逆行列 (Approximate INVerse，以下 **AINV** と略す) に基づく前処理法に着目した[1)2)]．AINV では，A–直交化に基づき，逆行列 A^{-1} そのものを近似する前処理行列 M を構成する．この前処理を用いた CG 法は，計算が行列とベクトルの積のみで行えるため，並列化が容易である．AINV では，発生した非零要素に対して，あらかじめ定めた閾値よりも小さい要素を零とみなし，大きい要素のみ残すという処理を行う．この処理は **dropping** (棄却，選別除去) と呼ばれる．逆行列 A^{-1} 自体をこのように近似する前処理が M. Benzi らの提案による AINV である[1)]．さらに，近似分解中の計算を工夫し，CG 法の収束の安定化を図ったものが同じ著者らによる **SAINV** (安定化近似逆行列，Stabilized-AINV)[2)] と呼ばれる前処理である．また，彼らとは独立に，Kharchenko らも AINV-A 法[5)] と呼ばれる前処理を提案しているが，その計算法は SAINV 法とほとんど同じである．これらの前処理を用いた場合，CG 法の収束性は閾値によって大きく影響を受ける．しかし，前処理の分解を行う前に最適な閾値を推測することは難しいため，実際に閾値の値を変えて方程式を解き，最適な閾値を探す必要がある．

本研究では，近似分解の過程において，通常の dropping 処理を行うと同時に，要素の更新可否判定を行う dropping 処理を新たに追加する．この処理は，CG 法の計算時間の短縮や，分解に必要なメモリ量の削減など目指した戦略の一つである．また，効率的な閾値の探索法について考察するために，複数の計算機を用いて数値実験を行う．本論文では，新しい **ISAINV** (Improved SAINV) の近似分解の考え方と数値実験の結果を報告し，その有効性を検証する．さらに，計算機の持つ特性を考慮した閾値の探索法についても報

† 九州大学大学院システム情報科学府
Graduate School of Information Science and Electrical Engineering, Kyushu University
†† 九州大学情報基盤センター
Computing and Communications Center, Kyushu University

告する．

2. 前処理つき共役勾配法

対称正定値行列を係数行列に持つ連立一次方程式

$$A\boldsymbol{x} = \boldsymbol{b} \qquad (1)$$

を前処理つきCG法で解くことにする．ここで，A は大きさ $n \times n$ の正方行列，$\boldsymbol{x}$，$\boldsymbol{b}$ は大きさ n の解および右辺ベクトルとする．このとき，前処理つきCG法の算法は以下のように表される．ここで，$\boldsymbol{x}_0$ は初期近似解，ε は収束判定値，M は前処理行列である．

[**Algorithm of preconditioned CG**]
$\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, $\boldsymbol{p}_0 = M\boldsymbol{r}_0$
for $k = 0, 1, \cdots$
$\alpha_k = (\boldsymbol{r}_k, M\boldsymbol{r}_k)/(\boldsymbol{p}_k, A\boldsymbol{p}_k)$
$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$
$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k$
if $||\boldsymbol{r}_{k+1}||_2/||\boldsymbol{r}_0||_2 \leq \varepsilon$ stop
$\beta_k = (\boldsymbol{r}_{k+1}, M\boldsymbol{r}_{k+1})/(\boldsymbol{r}_k, M\boldsymbol{r}_k)$
$\boldsymbol{p}_{k+1} = M\boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k$
end for

行列 MA の固有値分布が密集するように前処理行列 M を選べば，CG法の収束性が大幅に改善する．

3. 近似逆行列とその安定化および改良

3.1 近似逆行列分解

前処理行列 M として，係数行列 A の逆行列の近似を選ぶと，MA は単位行列に近づき，固有値は1付近に密集すると予想される．そこで，A-直交化に基づき，次のように行列 A の逆行列を近似分解する．ここで，Z は上三角行列，D は対角行列，Z^t は Z の転置行列とする．

$$M = Z^t D^{-1} Z \quad (\simeq A^{-1}) \qquad (2)$$

近似逆行列（**AINV**）の算法は以下のように表せる[1]．

[**Algorithm of AINV**]
for $i = 0, 1, \cdots, n-1$
$\boldsymbol{z}_i^{(0)} = \boldsymbol{e}_i$
end for
for $i = 0, 1, \cdots, n-1$
for $j = i, i+1, \cdots, n-1$
$d_j = \boldsymbol{a}_j^t \boldsymbol{z}_j^{(i)}$
end for
for $j = i+1, j+2, \cdots, n-1$
$\boldsymbol{z}_j^{(i+1)} = \boldsymbol{z}_j^{(i)} - \dfrac{d_j}{d_i}\boldsymbol{z}_i^{(i)}$
end for
end for

上の算法の中で，$\boldsymbol{z}_i^{(i)}$ は行列 Z の第 i 番目の列ベクトル，d_i は対角行列 D の第 i 番目の対角要素，$\boldsymbol{a}_j^t$ は行列 A の第 j 行ベクトルを表す．$\boldsymbol{e}_i$ は単位ベクトルの第 i 番目の要素を表す．$\boldsymbol{z}_j^{(i)}$ の上つき添字 (i) は繰り返し回数を表す．また，下線をつけた部分は $z_j^{(i)}$ の更新の計算であり，最終的に逆行列の近似分解の因子が得られる．このとき，行列のスパース性を保つために，予め設定した閾値の値よりも小さい要素は捨てられる．この処理を **dropping**，閾値のことを **drop tolerance value**（以下，tolと略す）と呼ぶ．上記の算法中で下線をつけた部分は，**dropping** 処理が行なわれると以下のようになる．ただし，ここでは要素ごとの処理になるため，列ベクトル $\boldsymbol{z}_k^{(i)}$ の第 j 番目の要素を $z_{kj}^{(i)}$ と表記することにする．

for $k = 0, \cdots, i$
if $|z_{kj}^{(i)} - \frac{d_j}{d_i} z_{ki}^{(i)}|$ > tol-1
$z_{kj}^{(i+1)} = z_{kj}^{(i)} - \frac{d_j}{d_i} z_{ki}^{(i)}$
else (3)
$z_{kj}^{(i+1)} = 0$
end if
end for

この近似分解によって得られた行列 Z と D を用いて，前処理行列 M は構成される．ただし，この分解では行列 D の対角要素が正の値になるとは限らないので，CG法の収束性が悪化する場合がある．

3.2 近似逆行列分解の安定化

そこで，収束の安定化を図るために，以下に示す安定化近似逆行列（**SAINV**）が提案された[2]．算法中の下線部分が前述の **AINV** と異なる部分である．

[**Algorithm of SAINV**]
for $i = 0, \cdots, n-1$
$\boldsymbol{z}_i^{(0)} = \boldsymbol{e}_i$
end for
for $i = 0, \cdots, n-1$
$\underline{\boldsymbol{v} = \mathrm{A}\boldsymbol{z}_i^{(i)}}$
for $j = i, \cdots, n-1$
$\underline{d_j = \boldsymbol{v}^t \boldsymbol{z}_j^{(i)}}$
end for
for $j = i+1, \cdots, n-1$
$\boldsymbol{z}_j^{(i+1)} = \boldsymbol{z}_j^{(i)} - \dfrac{d_j}{d_i}\boldsymbol{z}_i^{(i)}$
end for
end for

ここで，$\boldsymbol{v}$ は中間ベクトルである．A-直交化に基づく分解においては，対角要素 d_j の計算は簡略化された式 $\boldsymbol{a}_j^t \boldsymbol{z}_j^{(i)}$ で行なわれる．一方，この対角要素 d_j は，数学的に $\boldsymbol{z}_i^{(i)} A \boldsymbol{z}_j^{(i)}$ に厳密に等しいことがわかっている[2]．しかし，droppingによって要素が棄却されると，この等式は成り立たなくなり，閾値の選び方によっては，d_j の値が負になり，正定値性が崩れる可能性がある．そこで，安定化近似逆行列SAINVでは，簡略な計算式 $\boldsymbol{a}_j^t \boldsymbol{z}_j^{(i)}$ の代わりに，厳密な計算式 $\boldsymbol{z}_i^{(i)} A \boldsymbol{z}_j^{(i)}$ を

用いることによって対角要素 d_j を計算する．ここで，指標 j の動く範囲が $i \leq j \leq n-1$ であることに注意を要する．$j=i$ のとき，$d_i = \boldsymbol{z}_i^{(i)} A \boldsymbol{z}_i^{(i)}$ となり，対角要素 d_i は常に正の値をとり，前処理行列 M の正定値性が保たれる．実際の計算は，中間ベクトル $\boldsymbol{v}$ を使って，算法中の下線部に示すように2段階に分けて行なわれる．このようにして，安定化近似逆行列では対角行列 D の要素が正の値をとることが保証され，閾値 tol-1 の値の大小に拘らず分解が安定して実行できる．また，行列のスパース性も保たれるので，前処理の計算量が大きく増加することはない．

3.3 double dropping

本論文で提案する改良型のSAINV（以下，Improved SAINV を略して，**ISAINV** と呼ぶ）では，近似分解の過程の中で $\boldsymbol{z}_j^{(i)}$ の更新のときに二重に dropping 処理を行う．以下，この処理を double dropping と呼ぶ．double dropping 処理の果たす役割とそこで使われる閾値 tol-1 と tol-2 は次のように記述される．

(1) **閾値 tol-2 により，$z_j^{(i)}$ を更新するかどうかを判定する．判定の対象は $z_{ki}^{(i)}$ に掛ける比率 $\frac{d_j}{d_i}$ の絶対値とする．**

(2) 更新すると判定されたとき，閾値 tol-1 により，小さい要素 $z_{kj}^{(i+1)}$ は棄却される．すなわち，その値に 0 が代入される．

閾値 tol-2 による $\boldsymbol{z}_j^{(i)}$ の更新するかどうかの判定のタイミングは以下の下線部分で示される．

$$
\begin{array}{l}
\textbf{if } |\frac{d_j}{d_i}| > \text{tol-2} \\
\quad \textbf{for } k = 0, \cdots, i \\
\quad\quad \textbf{if } |z_{kj}^{(i)} - \frac{d_j}{d_i} z_{ki}^{(i)}| > \text{tol-1} \\
\quad\quad\quad z_{kj}^{(i+1)} = z_{kj}^{(i)} - \frac{d_j}{d_i} z_{ki}^{(i)} \\
\quad\quad \textbf{else} \\
\quad\quad\quad z_{kj}^{(i+1)} = 0 \\
\quad\quad \textbf{end if} \\
\quad \textbf{end for} \\
\underline{\textbf{end if}}
\end{array}
\tag{4}
$$

ISAINV では，double dropping を適用した近似分解によって逆行列因子 Z および対角行列 D を得ることになる．

4. 数値実験

4.1 計算機環境と計算条件

数値実験は異なる CPU を持つ 4 種類の計算機で行った．計算機の仕様を表 1 に示す．いずれの計算機においても，コンパイルは最適化オプションを使用せず行った．以下において，各 CPU を搭載した計算機を Itanium2，SPARC64，Pentium4，Alpha21264 と各々表記する．CG 法の収束判定条件は相対残差 L_2 ノルム: $||\boldsymbol{r}_k||_2/||\boldsymbol{r}_0||_2$ の値が 10^{-9} 以下のときとした．右辺項は厳密解がすべて 1 となるように定め，初期近似解 $\boldsymbol{x}_0$ は全て 0 とした．最大反復回数は行列の次元数と同じ値とし，そこで計算を打ち切った．行列は全て対角項を 1 に正規化した．以下において，従来の近似逆行列前処理は SAINV，提案する double dropping を行なう近似逆行列前処理は **ISAINV** と各々表記する．

表 1 数値実験で用いた計算機の仕様．
Table 1 Specifications of computers used in numerical experiments.

CPU	クロック数	メモリ	OS
Itanium2	0.9 GHz	0.5 GB	Red Hat Linux
SPARC64	1.3 GHz	24.0 GB	Solaris 8
Pentium4	2.0 GHz	1.5 GB	Windows2000
Alpha21264	0.73 GHz	64.0 GB	Tru64UNIX

表 2 テストに用いた行列の特徴と問題．
Table 2 Description on tested matrices.

行列	次元数	非零要素	問題（対象物）
BCSSTK24	3562	81736	固有値問題
BCSSTK35	30237	740200	車の座席と車体の間の剛性行列
NASA4704	4704	54730	NASA 建築物
NASASRB	54870	1366097	シャトルロケットブースタ
TUBE1-2	21498	459277	タイヤチューブの構造解析
SMT	25710	1889477	トランジスタの熱応力

SAINV では閾値 tol-1 を 0.05～0.15 まで 0.01 刻みで 11 通り変化させた．**ISAINV** では，閾値 tol-1 は上と同じ値にして，第 2 番目の閾値 tol-2 の値は，近似分解中に用いる第 1 番目の閾値 tol-1 の値の倍率 1.0～5.0 倍までの値を 0.5 刻みで全部で 9 通り変化させ，一つのテスト行列に対して合計 99 通りの閾値のときの各前処理つき CG 法の計算時間について調べた．

4.2 テスト行列

テスト行列は主として構造解析の分野の行列を用いた．6 個のテスト行列の特徴と解いた問題を表 2 に示す．行列 BCSSTK24 は Matrix Market[7] から，行列 BCSSTK35，同 NASA4704，同 NASASRB はフロリダ大学の疎行列データベース[8] から，行列 TUBE1-2，同 SMT は R. Kouhia[6] の home page から各々ダウンロードして数値実験で使用した．

4.3 実験結果と考察

表 3 に SPARC64 上における対角スケーリングおよびフィルインを考慮しない不完全コレスキー分解つき CG 法 (以下，ICCG 法と略す) の実験結果を示す．"∞" 印は各行列の次元数 n と同じ反復回数までに収束しなかったことを意味する．Diag. は対角スケーリング CG 法，IC は ICCG 法の結果である．表中において，precond. は前処理，it. は反復回数，p-t は前処理の計算時間，it-t は CG 法の収束までの計算時

間，tot-t は前処理と CG 法の反復の合計時間を各々表す．計算時間の単位はすべて秒である．表 3 より，ICCG 法はすべての問題に対して収束しなかった．対角スケーリングつき CG 法は半数の問題で収束せず，また収束した場合でも収束までに多くの反復回数が必要であった．他の計算機においても同様に，ICCG 法は収束せず，対角スケーリング CG 法も収束までに多くの反復回数が必要であった．

表 3 計算機 SPARC64 における対角スケーリング CG 法と ICCG 法の数値実験結果．

Table 3 Numerical results of Diagonal scaling CG method and ICCG method without fill-in on SPARC64.

matrix	precond.	it.	p-t	it-t	tot-t
BCSSTK24	Diag.	∞	0.0	–	–
	IC	∞	0.01	–	–
BCSSTK35	Diag.	∞	0.0	–	–
	IC	∞	0.05	–	–
NASA4704	Diag.	∞	0.0	–	–
	IC	∞	0.01	–	–
NASASRB	Diag.	13331	0.0	1392	1392
	IC	∞	0.08	–	–
TUBE1-2	Diag.	13826	0.0	488	488
	IC	∞	0.03	–	–
SMT	Diag.	3488	0.0	472	472
	IC	∞	0.10	–	–

表 4〜表 7 に，各計算機，行列に対して閾値：tol-1 と tol-2 を両方とも変化させたとき計算時間が最も短かった場合の二つの前処理つき CG 法の実験結果を計算機毎に示す．各表中において，tol-1，tol-2 は二つの閾値の値，ra1， ra2 は各々行列 A の格納および前処理 SAINV で必要なメモリ量に対する比，ra-t は SAINV が収束するまでの計算時間を 1.0 としたときの時間の比を表す．また，表 5 中の ra-diag には，表 3 で示した対角スケーリング CG 法の全体の計算時間との比率を表す．それ以外の項目は表 3 と同じである．表中の太字の数字はその値が顕著であることを表す．

これらの表の結果から，テストしたすべての行列と計算機に対して，提案した **ISAINV** 前処理は，従来の近似逆行列 SAINV より全体の計算時間の短縮，近似分解に必要なメモリの量の削減に効果的であった．具体的には，表 4 では時間比 0.47〜0.75，表 5 では同比 0.49〜0.74，表 6 では同比 0.46〜0.77，表 7 では同比 0.37〜0.74 に短縮した．メモリについても同様に，表 4 ではメモリ比 0.41〜0.91，表 5 では同比 0.41〜0.86，表 6 では同比 0.25〜0.92，表 7 では同比 0.33〜0.86 に減少した．また，表 5 より，SAINV および ISAINV は，対角スケーリングや IC 分解よりも収束までの計算時間が少ないことがわかった．

さらに，表 4〜表 7 の結果からは，調べたすべての計算機で共通に観察される **ISAINV** による改善の様子についても触れることができる．まず，行列 TUBE1-2 では反復回数が半減した．一方，行列 SMT では，反復計算時間はあまり減らなかったが，前処理の時間は大きく減少した．さらに，行列 BCSSTK24 では，SPARC64 の場合を除き反復回数は増加したが，前処理行列の非零要素が減少したため合計時間が減少した．

表 8 に，SAINV における CG 法の反復の計算時間を 1.0 としたときの前処理の計算時間の比，閾値 tol-1 を示す．表中において，ra-pre は前処理と反復計算の時間比，それ以外の項目は，表 4〜表 7 の場合と同じである．表中の太字の数字は各行列で最も比率が大きいものを表す．

さらに，図 1 は，行列 SMT に対する各計算機で最も短い全体の計算時間を 1.0 としたときの，閾値ごとの合計時間の比率をプロットしたものである．横軸は閾値 tol-1 の値，縦軸はその比率を表す．

まず表 8 より，Pentium4 の比率は他の三機種のそれに比べて，前処理の比率が大きいことがわかる．したがって，Pentium4 のような計算機では，閾値 tol-1 を大きく設定したとき合計時間が少なくて済む，あるいは閾値 tol-1 が最適値より大きくても，全体の計算時間は急激に増加しないと見積もることができる．実際，図 1 に示した比率のプロットからわかるように，Pentium4 では最適な値より閾値を大きくしても，他の計算機に比べて全体の計算時間が増加する割合が緩いことがわかる．また，一般に閾値を大きくすると dropping 処理によって多くの非零要素が落されるため，Pentium4 のような計算機では計算時間の増加を最小限に押えてメモリ量の大幅削減が可能になる．

一方，それ以外の計算機のように前処理の比率が小さいときには，閾値 tol-1 を小さくして合計計算時間が少なくなるように，最適閾値を探す必要がある．

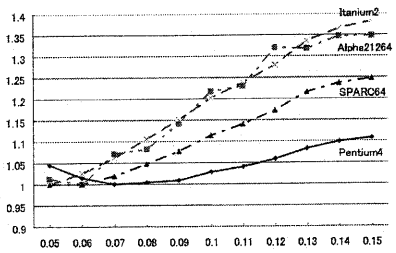

図 1 行列 SMT に対する各計算機で最も短い計算時間を 1.0 としたときの計算時間の比率のプロット．

Fig. 1 Plot of ratios of computation time for matrix SMT on four computers.

図 2 に，行列 NASA4704 に対する，Itanium2 上での閾値 tol-1 と tol-2 の値（tol-1 に対する倍率で表示）ごとの全体の計算時間の分布を示す．高さ方向は計算時間（秒）を表す．また，閾値 tol-1 は値をそのまま目盛にとり，閾値 tol-2 は閾値 tol-1 に対する倍率を目盛にとった．ここで，閾値 tol-2 の値が零の場

表 4　計算時間が最も短かったときのメモリ量の比と計算時間. (計算機 : Itanium2)
Table 4　Numerical results in case of the least computation time on Itanium2.

matrix	precond.	tol-1	tol-2	ra1	ra2	it.	p-t	it-t	tot-t	ra-t
BCSSTK24	SAINV	0.09	–	8.01	1.00	893	1.44	4.80	6.24	1.00
	ISAINV	0.14	0.280	2.58	**0.45**	944	0.21	3.69	3.90	0.64
BCSSTK35	SAINV	0.07	–	3.94	1.00	9843	18.8	726	746	1.00
	ISAINV	0.05	0.125	3.26	0.79	6644	9.04	529	538	0.69
NASA4704	SAINV	0.12	–	3.61	1.00	1469	0.21	5.62	5.83	1.00
	ISAINV	0.13	0.390	2.89	0.60	996	0.14	3.10	3.24	**0.49**
NASASRB	SAINV	0.12	–	2.80	1.00	7767	12.5	748	761	1.00
	ISAINV	0.08	0.280	2.56	0.91	6107	10.0	557	567	0.75
TUBE1-2	SAINV	0.14	–	6.84	1.00	4183	41.5	130	172	1.00
	ISAINV	0.05	0.250	2.85	**0.42**	2048	4.43	76.2	80.6	**0.47**
SMT	SAINV	0.05	–	10.8	1.00	803	61.4	93.7	155	1.00
	ISAINV	0.05	0.075	4.45	**0.41**	831	23.5	93.5	117	0.75

表 5　計算時間が最も短かったときのメモリ量の比と計算時間. (計算機 : SPARC64)
Table 5　Numerical results in case of the least computation time on SPARC64.

matrix	precond.	tol-1	tol-2	ra1	ra2	it.	p-t	it-t	tot-t	ra-t	ra-diag
BCSSTK24	SAINV	0.09	–	8.01	1.00	893	2.62	8.98	11.6	1.00	–
	ISAINV	0.05	0.100	3.63	**0.45**	588	0.82	6.65	7.47	0.64	–
BCSSTK35	SAINV	0.06	–	4.18	1.00	9497	38.1	1138	1176	1.00	–
	ISAINV	0.05	0.125	3.26	0.79	6648	18.0	792	810	0.69	–
NASA4704	SAINV	0.06		4.79	1.00	1029	0.88	11.1	12.0	1.00	–
	ISAINV	0.13	0.455	2.86	0.60	896	0.29	5.59	5.88	**0.49**	–
NASASRB	SAINV	0.12	–	2.80	1.00	7761	25.9	1161	1187	1.00	0.85
	ISAINV	0.11	0.495	2.40	0.86	6830	18.7	856	874	0.74	0.63
TUBE1-2	SAINV	0.13	–	6.71	1.00	4134	45.9	208	254	1.00	0.52
	ISAINV	0.05	0.250	2.85	**0.42**	2050	10.4	118	128	**0.50**	0.26
SMT	SAINV	0.05	–	10.8	1.00	804	138	148	286	1.00	0.61
	ISAINV	0.05	0.075	4.45	**0.41**	831	41.9	149	190	0.66	0.40

表 6　計算時間が最も短かったときのメモリ量の比と計算時間. (計算機 : Pentium4)
Table 6　Numerical results in case of the least computation time on Pentium4.

matrix	precond.	tol-1	tol-2	ra1	ra2	it.	p-t	it-t	tot-t	ra-t
BCSSTK24	SAINV	0.09	–	8.01	1.00	895	1.16	2.70	3.86	1.00
	ISAINV	0.13	0.455	2.42	**0.30**	1045	0.14	1.99	2.13	**0.55**
BCSSTK35	SAINV	0.13	–	3.23	1.00	13646	21.6	372	394	1.00
	ISAINV	0.14	0.420	2.51	0.78	11195	5.72	261	267	0.68
NASA4704	SAINV	0.12	–	3.61	1.00	1470	0.22	3.78	4.00	1.00
	ISAINV	0.13	0.455	2.86	0.79	895	0.13	1.72	1.84	**0.46**
NASASRB	SAINV	0.12	–	2.80	1.00	7769	19.4	361	380	1.00
	ISAINV	0.07	0.280	2.58	0.92	6141	16.3	276	292	0.77
TUBE1-2	SAINV	0.10	–	7.37	1.00	3611	24.5	63.9	88.4	1.00
	ISAINV	0.05	0.250	2.85	**0.39**	2022	5.11	38.1	43.3	**0.49**
SMT	SAINV	0.07	–	9.42	1.00	1057	67.0	48.3	115	1.00
	ISAINV	0.14	0.280	2.37	**0.25**	1509	17.4	56.5	73.9	0.64

合 (図の中で最も奥) は従来の SAINV の結果を表している.

図 2 より, **ISAINV** では, 閾値 tol-1 の値に関わらず全般的に計算時間が短いことがわかる. 特に計算時間が短いのは, 閾値 tol-2 の値を閾値 tol-1 の値の 3.5 倍 ~4 倍のときである. これ以外の行列の場合でも, 閾値 tol-2 の値は閾値 tol-1 の値の 3 倍前後で計算時間が最も短くなる場合が多かった.

5. おわりに

安定化近似逆行列による分解過程の途中で dropping を二重に行う **ISAINV** 前処理を提案し, その閾値の効率的な探索について言及した. そして, **ISAINV** は, 近似分解に要する前処理に要する時間と CG 法の収束までの合計計算時間の短縮に極めて有効であった. さらに, 必要なメモリ量の削減にも大きな効果が

表 7　計算時間が最も短かったときのメモリ量の比と計算時間.（計算機：Alpha21264）
Table 7　Numerical results in case of the least computation time on Alpha21264.

matrix	precond.	tol-1	tol-2	ra1	ra2	it.	p-t	it-t	tot-t	ra-t
BCSSTK24	SAINV	0.09	–	7.99	1.00	893	1.12	3.87	4.98	1.00
	ISAINV	0.09	0.270	2.68	**0.33**	920	0.30	2.93	3.23	0.65
BCSSTK35	SAINV	0.13	–	3.30	1.00	13848	20.8	1306	1327	1.00
	ISAINV	0.12	0.420	2.53	0.77	11009	8.78	894	903	0.68
NASA4704	SAINV	0.12	–	3.61	1.00	1471	0.32	4.63	4.95	1.00
	ISAINV	0.13	0.455	2.86	0.79	895	0.13	1.72	1.84	**0.37**
NASASRB	SAINV	0.12	–	2.80	1.00	7761	28.5	1430	1459	1.00
	ISAINV	0.11	0.495	2.40	0.86	6830	23.3	1019	1043	0.71
TUBE1-2	SAINV	0.12	–	6.97	1.00	3888	18.7	200	218	1.00
	ISAINV	0.05	0.250	2.85	**0.41**	2050	7.32	123	130	0.60
SMT	SAINV	0.06	–	10.2	1.00	930	92.0	222	314	1.00
	ISAINV	0.05	0.075	4.45	**0.44**	831	47.4	183	231	0.74

表 8　CG 法の反復の計算時間を 1.0 としたときの前処理の計算時間の比.
Table 8　Ratios of computation time of preconditioning versus iteration time needed for convergence.

matrix	Itanium2		SPARC64		Pentium4		Alpha21264	
	ra-pre	tol-1	ra-pre	tol-1	ra-pre	tol-1	ra-pre	tol-1
BCSSTK24	0.300	0.09	0.292	0.09	**0.430**	0.09	0.289	0.09
BCSSTK35	0.026	0.07	0.032	0.06	**0.058**	0.13	0.016	0.13
NASA4704	0.037	0.12	**0.079**	0.06	0.058	0.12	0.069	0.12
NASASRB	0.017	0.12	0.022	0.12	**0.054**	0.12	0.020	0.12
TUBE1-2	0.319	0.14	0.221	0.13	**0.383**	0.10	0.094	0.12
SMT	0.655	0.05	0.932	0.05	**1.387**	0.07	0.414	0.06

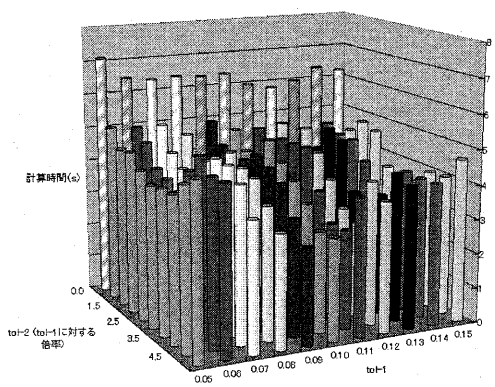

図 2　行列 NASA4704 に対する閾値 tol-1 と tol-2（tol-1 に対する倍率）ごとの CG 法の計算時間（秒）の分布.（計算機：Itanium2）
Fig. 2　Distribution of computation time (in sec.) on Itanium2 for matrix NASA4704.

あった．閾値は，前処理の比率が大きい計算機（例えば，Pentium4）では閾値 tol-1 を大きく設定し，逆に前処理の比率が小さい計算機（例えば，Itanium2, SPARC64, Alpha21264）では tol-1 を小さく設定したときに，全体の計算時間が短くなることがわかった．さらに，**ISAINV** では，dropping で使用する二つの閾値のうち，閾値 tol-2 の値は閾値 tol-1 の値の 3 倍程度の値を与えればよいことがわかった．

謝辞

多くの有益なるご助言を戴いた M. Benzi 助教授および M. Tuma 博士に心より感謝の意を表す．

参 考 文 献

1) Benzi, M., Meyer, C.D., Tuma, M.: A sparse approximate inverse preconditioner for the conjugate gradient method, *SIAM J. on Scientific Computing*, Vol. 17, pp. 1135–1149 (1996).
2) Benzi, M., Cullum, J.K., Tuma, M.: Robust approximate inverse preconditioning for the conjugate gradient method, *SIAM J. on Scientific Computing*, Vol. 22, pp. 1318–1332 (2000).
3) Benzi, M., Tuma, M.: A robust incomplete factorization preconditioner for positive definite matrices, *Numer. Lin. Alg. Appl.*, Vol. 99, pp. 1–20 (2001).
4) 池田優介：Matrix Market における共役勾配法の収束性評価，九州大学工学部卒業論文，2002.
5) Kharchenko, S.,et al.: A robust AINV-type method for constructing sparse approximate inverse preconditioners in factored form, *Numer. Lin. Alg. Appl.*, Vol.8, pp.165–179 (2001).
6) http://www.hut.fi/~kouhia/sparse.html
7) http://math.nist.gov/MatrixMarket/
8) http://www.cise.ufl.edu/research/sparse/matrices/