二重ドロッピングによる安定化近似逆行列前処理の改良

池田優介† 藤野清次††

共役勾配法はその収束性を向上させるために不完全コレスキー分解のような前処理と併用して使われることが多い．しかし，行列の性質が極端に悪い場合，分解過程でピボットの値が負の値になって分解が中断したり，あるいは共役勾配法の収束性に改善があまり見られなかったりすることがよく起こる．このような現象を回避するため，近年近似逆行列による前処理が注目を集めている．近似逆行列では，前処理行列のスパース性を保つために dropping と呼ばれる処理が施される．本研究では，スパース性の保持にとって鍵となる dropping の処理に注目し，この処理を近似分解中で二重に行い，収束性を改良した安定化近似逆行列前処理を提案する．そして，数値実験によりその有効性を検証する．

An Improvement of Stabilized Approximate INVerse Preconditioner by Double Dropping

Yusuke Ikeda† and Seiji Fujino††

Preconditioning based on incomplete factorization of the matrix A is among the best known and most popular methods for solving a linear system of equations with symmetric positive definite coefficient matrix. However, the existence of an incomplete factorization is a delicate issue which must be addressed if one has a desire to design reliable preconditioner. Recently stabilized AINV (Approximate INVerse) preconditioners with single dropping have been proposed. Dropping procedure is a key to improvement of efficiency of computation. In this paper, we propose stabilized AINV precondioner with double dropping. Numerical results show that the proposed stabilized AINV is very effective for realizing a high-performance computation as compared with the standard SAINV preconditioners.

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は前処理つきの BiCG 法系統の反復法によって解かれることが多く，特に係数行列が対称正定値行列のとき，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）分解が広く用いられる．IC 分解では前処理行列 M を $M = U^tU$ のように行列と行列の積の形に分解する．しかし，理論的に IC 分解が有効であるのが分かっているのは M–行列のときだけである[14]．それ以外の行列に対して IC 分解を適用すると，分解過程の途中で，ピボットの値が負になり分解が中断されたり，仮に分解はできたとしても M–行列のときのように大幅な収束性の改善が望めなかったりすることが多い[9]．

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[2),3),5)]．AINV とは，A-直交化（A-orthogonalization）に基づき逆行列 A^{-1} そのものを近似する前処理行列 M を構成する方法である．

逆行列を求める分解は，完全に行うと演算量とメモリ量が急増し現実的でない．そこで，分解を近似的に行い，演算量とメモリ量の増加を抑える必要がある．具体的には，ある閾値をあらかじめ設定し，分解過程で生まれた非零要素に対して，その値が閾値よりも小さいときはそれを零と見なし，閾値よりも大きいとき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} をこのように近似分解して前処理に用いるのが Benzi らによる AINV 前処理である[3]．

さらに，近似分解の中の計算方法を工夫し，CG 法の収束の安定化を図ったものが同じ著者らによる安定

† 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

化近似逆行列[5]（Stabilized-AINV，以下 **SAINV** と略す）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[10] 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．

近似逆行列による前処理はいろいろな分野の問題に適用され始めている．たとえば，固体力学や構造解析分野では文献 7)，特に解きにくいとされるシェル構造問題に対しては文献 4) が参考になる．また，最近は実数係数行列だけでなく複素係数行列に対する前処理としてもその有効性が報告されてきた．たとえば，電磁気問題では文献 1)，8) があげられる．さらに，多孔質媒体の浸透流解析にも応用されその有効性が示された[13]．この問題の係数行列は複素かつ密行列になる．

さらに，上述の dropping については，近似分解終了後にもう一度 dropping を行う post-filteration（近似分解後の濾過）と呼ばれる方法がほぼ同時期に提案されている[6),10]．しかし，その改善効果の度合いは小さく，開発者たちも post-filteration を積極的に推奨していない．そこで，post-filteration のように近似分解が終った後ではなく，近似分解過程で要素の dropping 処理を行う直前に，その要素が含まれる列全体に対して要素の更新を行うかどうかを判定する dropping 処理をここでは提案する．したがって，dropping 処理は二重構造の形になる．

本論文の構成は以下のとおりである．まず，2 章で前処理 U^tU 分解つき CG 法について記す．続く 3 章では，近似逆行列と安定化近似逆行列について述べる．3.1 節で近似逆行列前処理つき CG 法，3.2 節で近似逆行列分解の安定化について言及する．さらに，3.3 節で従来の post-filteration の考え方，3.4 節で提案する二重 dropping による近似分解について各々記述する．そして，4 章で数値実験結果を示し，その考察を行う．最後に，5 章でまとめと今後の課題について述べる．

2. U^tU 分解前処理つき CG 法

対称正定値行列を係数行列 A に持つ連立一次方程式

$$A\boldsymbol{x} = \boldsymbol{b} \tag{1}$$

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．まず，係数行列 A を近似分解して，以下のように前処理行列 M を構成する．

$$M = U^tU \ (\simeq A) \ . \tag{2}$$

ここで，U は上三角行列，上付き添字 t は転置を表す．この U および U^t を用いて，方程式 (1) を

$$(U^{-t}AU^{-1})\ (U\boldsymbol{x}) = U^{-t}\boldsymbol{b} \tag{3}$$

のように変形する．この操作は前処理と呼ばれる．前処理によって，式 (3) は，$U^{-t}AU^{-1}$，$U\boldsymbol{x}$，$U^{-t}\boldsymbol{b}$ をそれぞれ係数行列，解ベクトル，右辺ベクトルとする新たな方程式となる．前処理行列 M が元の係数行列 A をよく近似していれば，前処理後の係数行列 $U^{-t}AU^{-1}$ は単位行列に近づき CG 法の収束性が改善すると予想される．そこで，方程式 (3) と等価な方程式を導出し，その方程式に対して CG 法を適用する．U^tU 分解前処理つき CG 法の算法は以下のように表される．$\boldsymbol{x}_0$ は初期近似解，ε は収束判定値である．

算法 1　U^tU分解前処理つき CG 法

$\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0,\ \boldsymbol{p}_0 = (U^tU)^{-1}\boldsymbol{r}_0$
for $m = 1, 2, \ldots$
$\quad \alpha_m = \dfrac{(\boldsymbol{r}_{m-1}, (U^tU)^{-1}\boldsymbol{r}_{m-1})}{(\boldsymbol{p}_{m-1}, A\boldsymbol{p}_{m-1})}$
$\quad \boldsymbol{x}_m = \boldsymbol{x}_{m-1} + \alpha_m \boldsymbol{p}_{m-1}$
$\quad \boldsymbol{r}_m = \boldsymbol{r}_{m-1} - \alpha_m A\boldsymbol{p}_{m-1}$
$\quad$ if $||\boldsymbol{r}_m||_2/||\boldsymbol{r}_0||_2 \leq \varepsilon$ stop
$\quad \beta_m = \dfrac{(\boldsymbol{r}_m, (U^tU)^{-1}\boldsymbol{r}_m)}{(\boldsymbol{r}_{m-1}, (U^tU)^{-1}\boldsymbol{r}_{m-1})}$
$\quad \boldsymbol{p}_m = (U^tU)^{-1}\boldsymbol{r}_m + \beta_m \boldsymbol{p}_{m-1}$
end for

3. 近似逆行列と安定化近似逆行列

3.1 近似逆行列前処理つき CG 法

前処理行列 M として，式 (2) のように係数行列 A を近似せずに，逆行列 A^{-1} を直接近似することを考える．近似逆行列（**AINV**）では，次のように A^{-1} を近似分解する．ここで，Z は上三角行列，D は対角行列とする．

$$M = ZD^{-1}Z^t \ \ (\simeq A^{-1}). \tag{4}$$

AINV による前処理においても，U^tU 分解と同様に前処理後の係数行列は対称性を保つ必要がある．そこで，さらに対角行列 D を分解し

$$ZD^{-1}Z^t = (ZD^{-\frac{1}{2}})(D^{-\frac{1}{2}}Z^t) = \tilde{Z}\tilde{Z}^t \tag{5}$$

とする．この分解は対角行列 D の要素がすべて正であれば可能である．そして，Z ではなく $\tilde{Z}$ を用いて方程式 (1) を

$$(\tilde{Z}^tA\tilde{Z})\ (\tilde{Z}^{-1}\boldsymbol{x}) = \tilde{Z}^t\boldsymbol{b} \tag{6}$$

のように変形する．前節の U^tU 分解による前処理と同様に，式 (6) は $\tilde{Z}^tA\tilde{Z}$，$\tilde{Z}^{-1}\boldsymbol{x}$，$\tilde{Z}^t\boldsymbol{b}$ をそれぞれ係数行列，解ベクトル，右辺ベクトルとする新たな方程式となる．ただし，式 (6) は “前処理つき CG 法において，前処理行列 M は $\tilde{Z}\tilde{Z}^t$ の形に分解できる必要がある” ことを示しているだけで，実際の反復計算では

$ZD^{-1}Z^t$ を用いても問題はない．したがって，AINV 前処理つき CG 法の算法は以下のように表せる．

算法 2　AINV 前処理つき CG 法

$\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0,\ \boldsymbol{p}_0 = ZD^{-1}Z^t\boldsymbol{r}_0$
for $m = 1, 2, \ldots$
$\alpha_m = \dfrac{(\boldsymbol{r}_{m-1}, ZD^{-1}Z^t\boldsymbol{r}_{m-1})}{(\boldsymbol{p}_{m-1}, A\boldsymbol{p}_{m-1})}$
$\boldsymbol{x}_m = \boldsymbol{x}_{m-1} + \alpha_m\boldsymbol{p}_{m-1}$
$\boldsymbol{r}_m = \boldsymbol{r}_{m-1} - \alpha_m A\boldsymbol{p}_{m-1}$
if $||\boldsymbol{r}_m||_2/||\boldsymbol{r}_0||_2 \leq \varepsilon$ stop
$\beta_m = \dfrac{(\boldsymbol{r}_m, ZD^{-1}Z^t\boldsymbol{r}_m)}{(\boldsymbol{r}_{m-1}, ZD^{-1}Z^t\boldsymbol{r}_{m-1})}$
$\boldsymbol{p}_m = ZD^{-1}Z^t\boldsymbol{r}_m + \beta_m\boldsymbol{p}_{m-1}$
end for

AINV では，A-直交化に基づいて逆行列因子 Z および D を構成する．AINV の算法は以下のとおりである[3]．

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

上の算法の中において，$\boldsymbol{z}_j^{(i)}$ は反復 i 回目における行列 Z の第 j 番目の列ベクトル，$\boldsymbol{e}_j$ は第 j 番目の要素が 1 である単位ベクトル，d_j は対角行列 D の第 j 番目の対角要素，$\boldsymbol{a}_i^t$ は行列 A の第 i 番目の行ベクトルをそれぞれ表す．また，下線をつけた部分は $\boldsymbol{z}_j^{(i-1)}$ の更新を行う箇所であり，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，あらかじ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 処理が行われると以下のようになる．ただし，dropping 処理は要素ごとの操作となるため，列ベクトル $\boldsymbol{z}_j^{(i-1)}$ の第 k 番目の要素を $z_{kj}^{(i-1)}$ と表記することにする．

for $k = 1, 2, \cdots, i$
　if $|z_{kj}^{(i-1)} - \frac{d_j}{d_i}z_{ki}^{(i-1)}| > \mathrm{tol}$
　　$z_{kj}^{(i)} = z_{kj}^{(i-1)} - \frac{d_j}{d_i}z_{ki}^{(i-1)}$
　else (7)
　　$z_{kj}^{(i)} = 0$
　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，この処理により行列 D の対角要素が正の値になるとは限らないので，CG 法の収束性に影響が出る可能性がある．

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，安定化近似逆行列（**SAINV**）が提案された[5]．SAINV の算法は以下のように表せる．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 $j = 1, 2, \cdots, n$
　$\boldsymbol{z}_j^{(0)} = \boldsymbol{e}_j$
end for
for $i = 1, 2, \cdots, n$
　$\underline{\boldsymbol{v} = \mathrm{A}\boldsymbol{z}_i^{(i-1)}}$
　for $j = i, i+1, \cdots, n$
　　$\underline{d_j = \boldsymbol{v}^t\boldsymbol{z}_j^{(i-1)}}$
　end for
　for $j = i+1, i+2, \cdots, n$
　　$\boldsymbol{z}_j^{(i)} = \boldsymbol{z}_j^{(i-1)} - \dfrac{d_j}{d_i}\boldsymbol{z}_i^{(i-1)}$
　end for
end for

ここで，$\boldsymbol{v}$ は中間列ベクトルである．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，AINV の算法中に示した式 $\boldsymbol{a}_i^t\boldsymbol{z}_j^{(i-1)}$ で行われる．一方，この対角要素 d_j は，数学的に $(\boldsymbol{z}_i^{(i-1)})^tA\boldsymbol{z}_j^{(i-1)}$ に等しいことが分かっている[5]．しかし，dropping によって要素が棄却されたとき，この等号の関係は成り立たなくなる．閾値の選び方によっては，d_j の値が負になり，正定値性が崩れてしまう．そこで，SAINV では，元の計算式 $\boldsymbol{a}_i^t\boldsymbol{z}_j^{(i-1)}$ の代わりに，数学的に等価な計算式 $(\boldsymbol{z}_i^{(i-1)})^tA\boldsymbol{z}_j^{(i-1)}$ 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 $i \leq j \leq n$ であることに注意を要する．$j = i$ のとき，$d_i = (\boldsymbol{z}_i^{(i-1)})^tA\boldsymbol{z}_i^{(i-1)}$ となり，対称正定値を表す定義式そのものとなる．したがって，対角要素 d_i はつねに正の値をとり，前処理行列 M の正定値性が保たれる．実際の計算は，中間列ベクトル $\boldsymbol{v}$ を使って，上

の算法中の下線部に示すように 2 段階に分けて行われる．このようにして，SAINV では対角行列 D の要素が正の値であることが保証され，閾値 tol の値の大小にかかわ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，前処理の計算の負荷も比較的少ない．

3.3 post-filteration

さらに，SAINV の算法によって得られた行列 Z に対して，近似分解が終了した後（反復ステップ数は n になる）に以下のような filtering と呼ばれる処理を施すことが提案された[6),10)]．この処理は，近似分解後に行うという意味から，特に post-filteration（近似分解後の濾過）と呼ばれている．すなわち，反復計算で得られた最終的なベクトル $\boldsymbol{z}_j^{(n)}$ だけが濾過の対象となる．行列 Z が上三角行列であることに注意すると，post-filteration は次のように表すことができる．

```
for j = 1, 2, ···, n
  for k = 1, 2, ···, j
    if |z_kj^(n)| > tol_pf
      z_kj = z_kj^(n)
    else
      z_kj = 0
    end if
  end for
end for
```
(8)

ここで，tol_pf は post-filteration に用いる新たに導入した閾値である．この処理により，行列 Z の非零要素の個数が減少し，近似行列はより不完全になる．したがって，CG 法の反復回数の増加が予想されるが，反復 1 回あたりの演算量は減少するため，収束までの計算時間は両者の兼ね合いで決まる．以下，この処理を適用した SAINV を SAINV-pf と略す．

3.4 二重の dropping 処理

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）では近似分解後ではなく，近似分解の過程の中で，$\boldsymbol{z}_j^{(i-1)}$ 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．新たに導入した閾値 tol_dd と従来の dropping 処理に使われる閾値 tol が果たす役割はそれぞれ次のように表される．

(1) **閾値 tol_dd により，ベクトル $\boldsymbol{z}_j^{(i-1)}$ を更新するかどうかを判定する．判定には，$\boldsymbol{z}_i^{(i-1)}$ に掛ける比率 $\frac{d_j}{d_i}$ の絶対値が使用される．**

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

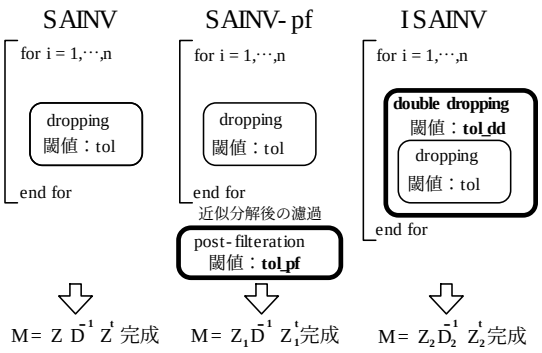

図 1 3 つの安定化近似逆行列前処理の手順の概略
Fig. 1 Procedure of three stabilized Approximate INVerse preconditionings.

閾値 tol_dd により $\boldsymbol{z}_j^{(i-1)}$ を更新するかどうかの判定は，式 (9) に示す処理の下線部分で行われる．

```
if |d_j/d_i| > tol_dd
  for k = 1, 2, ···, i
    if |z_kj^(i-1) − (d_j/d_i) z_ki^(i-1)| > tol
      z_kj^(i) = z_kj^(i-1) − (d_j/d_i) z_ki^(i-1)
    else
      z_kj^(i) = 0
    end if
  end for
end if
```
(9)

3 つの安定化近似逆行列前処理：SAINV，post-filteration を行う SAINV-pf，double dropping を行う **ISAINV** の手順の概略は**図 1** のように表せる．図中で太い実線で囲まれた部分が新たに追加された処理および変更の該当部分を表す．一方，細い実線で囲まれた部分は共通部分である．また，SAINV と SAINV-pf の対角行列 D は同じになる．しかし，**ISAINV** の対角行列 D はそれとは異なる．一方，行列 Z は 3 つの前処理で各々異なる．

4. 数 値 実 験

4.1 計算機環境と計算条件

数値実験は以下の計算機環境で行った．

- 計算機：富士通 Prime Power 850
- PE 数：16
- CPU（クロック数）：SPARC64（1.3 GHz）
- メモリ：24 Gbyte
- 使用言語：C 言語
- コンパイラ：富士通 C compiler version 5.3
- 最適化オプション：-Kfast

表 1 テストに用いた問題の仕様
Table 1 Specification of tested problems.

行列	次元数	非零要素数	問題（対象物）
BCSSTK24	3,562	81,736	固有値問題
BCSSTK35	30,237	740,200	自動車座席の構造解析
NASA4704	4,704	54,730	NASA 建築物
NASASRB	54,870	1,366,097	シャトルロケットブースタ
TUBE1-2	21,498	459,277	タイヤチューブの構造解析
SMT	25,710	1,889,477	トランジスタ

上記の計算機の 1PE を使用した．計算はすべて倍精度演算で行い，最大反復回数は各行列の次元数 n とし，そこで計算を打ち切った．収束判定条件 ε は相対残差 L_2 ノルム $||\boldsymbol{r_m}||_2/||\boldsymbol{r_0}||_2$ の値が 10^{-9} 以下とした．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 $\boldsymbol{x}_0$ はすべて 0 とした．また，行列は対角スケーリングを用いて対角項を 1 に正規化した．各前処理に必要なメモリ量の算出は，近似分解の反復ごとに必要となる配列の要素数から行い，その最大値をとった．

SAINV では閾値 tol を 0.05〜0.15 まで 0.01 刻みで 11 通り変化させた．一方，SAINV-pf と **ISAINV** では，閾値 tol は SAINV と同じ値にして，第 2 番目の閾値，tol_pf と tol_dd の値を，近似分解中に用いる第 1 番目の閾値 tol 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，1 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から問題を選び，数値実験に用いた．6 個のテスト行列の仕様と解いた問題を**表 1** に示す．行列 BCSSTK24 は Matrix Market [12] から，行列 BCSSTK35，NASA4704，NASASRB はフロリダ大学の疎行列データベース[15] から，行列 TUBE1-2，SMT は R. Kouhia [11] 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 2 に前処理なし CG 法とフィルインを考慮しない IC 分解つき CG 法の実験結果を示す．ただし，前述のとおり行列は対角項を 1 に正規化した．したがって，前処理なし CG 法は実質的にいわゆる対角スケーリング CG 法にあたる．表中において “∞” 印は最大反復回数までで収束しなかったことを意味しており，前処理時間，反復時間，合計時間の単位はすべて秒である．

表 2 前処理なし CG 法とフィルインを考慮しない不完全コレスキー分解つき CG 法の数値実験結果
Table 2 Numerical results of non-preconditioned CG and ICCG without fill-in.

行列	前処理	反復回数	前処理時間	反復時間	合計時間
BCSSTK24	なし	∞	0.0	–	–
	IC	∞	0.01	–	–
BCSSTK35	なし	∞	0.0	–	–
	IC	∞	0.03	–	–
NASA4704	なし	∞	0.0	–	–
	IC	∞	0.01	–	–
NASASRB	なし	13331	0.0	391	391
	IC	∞	0.05	–	–
TUBE1-2	なし	13826	0.0	139	139
	IC	∞	0.03	–	–
SMT	なし	3488	0.0	134	134
	IC	∞	0.06	–	–

表 2 から分かるように，IC 分解つき CG 法はすべての問題に対して収束しなかった．CG 法では半分にあたる 3 つの問題で収束せず，また収束した場合でも収束までに多くの反復回数がかかったことが分かる．

表 3 に，各行列に対して閾値：tol および tol_pf，tol_dd を変化させたとき計算時間が最も少なかった場合の 3 つの前処理つき CG 法の実験結果を示す．表中において，tol，tol_pf，tol_dd は各閾値の値，メモリ量は前処理に必要になったメモリ量（単位：Mbyte）の最大値，メモリ比は前処理 SAINV に必要なメモリ量を 1.0 としたときの他の前処理のメモリ量の比，同様に，時間比は SAINV が収束するまでの合計時間を 1.0 としたときの他の前処理の計算時間の比をそれぞれ表す．計算時間の単位はすべて秒である．

近似逆行列前処理におけるその有用性の判断基準は，計算時間の短縮とメモリ量の削減の度合いである．そして，閾値の選び方がこれらに大きな影響を及ぼす．しかし，実際に問題を解く前に最適な閾値の値を知ることは不可能である．そこで，次善の策として，できるだけ多くの問題において，最適な閾値が存在する可能性がある範囲を知っておくことは実際の問題を解くとき重要であると思われる．また，その平均値は改善効果の見積りをするうえで有用な目安になる．

そこで，閾値の組合せを変えた合計 99 通りのケースについて最適な閾値を調べた．**表 4** にその探索結果を示す．ただし，SAINV では閾値を変化させて合計 11 通りについて最適な閾値を調べた．表中には，メモリ量，反復回数，計算時間の平均値および比率などを示す．また，太字で表した数字は顕著な効果が見られるケースを意味する．

表 3 と表 4 の結果から，double dropping を施し

表 3 各行列に対する 3 つの前処理つき CG 法の最少の計算時間のときの数値実験結果（閾値，メモリ量（MB），反復回数，合計計算時間および比率など）
Table 3 Numerical results (dropping tolerance values, amount of memory (MB), iterations, total computation time (in sec.) and ratios) in case of the least computation time.

行列	前処理	tol	tol_pf	tol_dd	メモリ量	メモリ比	反復回数	前処理時間	反復時間	合計時間	時間比
	SAINV	.10	–	–	7.00	1.00	1061	0.83	2.31	3.14	1.00
BCSSTK24	SAINV-pf	.10	.10	–	7.00	1.00	1061	0.84	2.29	3.13	0.99
	ISAINV	.13	–	.455	2.26	**0.32**	1044	0.16	1.64	1.80	0.57
	SAINV	.13	–	–	27.9	1.00	13848	12.0	426	438	1.00
BCSSTK35	SAINV-pf	.06	.15	–	35.4	1.27	12035	19.1	381	400	0.91
	ISAINV	.14	–	.42	21.3	0.76	11199	5.46	272	277	0.63
	SAINV	.12	–	–	2.26	1.00	1471	0.25	2.80	3.05	1.00
NASA4704	SAINV-pf	.12	.12	–	2.26	1.00	1471	0.25	2.79	3.04	0.99
	ISAINV	.12	–	.48	1.78	0.79	925	0.16	1.29	1.45	**0.48**
	SAINV	.12	–	–	43.8	1.00	7761	16.6	392	408	1.00
NASASRB	SAINV-pf	.12	.30	–	43.8	1.00	9630	16.2	376	393	0.96
	ISAINV	.15	–	.75	36.6	0.83	7923	13.1	285	298	0.73
	SAINV	.13	–	–	35.3	1.00	4134	25.9	68.0	93.8	1.00
TUBE1-2	SAINV-pf	.13	.13	–	35.3	1.00	4134	25.9	68.0	93.8	1.00
	ISAINV	.05	–	.25	15.0	**0.42**	2050	4.48	42.0	46.5	**0.50**
	SAINV	.06	–	–	220	1.00	930	61.0	54.5	115.4	1.00
SMT	SAINV-pf	.07	.07	–	204	0.93	1057	57.1	58.1	115.2	0.99
	ISAINV	.05	–	.075	96.1	**0.44**	831	24.5	47.1	71.6	0.62

表 4 各行列に対して閾値を変化させたときの 3 つの前処理つき CG 法の数値実験結果（平均メモリ量（MB），平均反復回数，平均計算時間およびその比率など）
Table 4 Numerical results (average amount of memory (MB), average iterations, average computation time (in sec.) and ratios) of three preconditioned CG method when varying dropping tolerance values.

行列	前処理	メモリ量	メモリ比	反復回数	前処理時間	反復時間	合計時間	時間比
	SAINV	7.46	1.00	1101	1.13	2.55	3.69	1.00
BCSSTK24	SAINV-pf	7.46	1.00	2122	1.14	3.72	4.86	1.32
	ISAINV	2.58	**0.35**	969	0.22	1.80	2.02	**0.55**
	SAINV	30.7	1.00	12421	15.4	444	459	1.00
BCSSTK35	SAINV-pf	30.7	1.00	21366	15.4	534	550	1.20
	ISAINV	23.1	0.75	11853	6.45	345	352	0.77
	SAINV	2.54	1.00	1380	0.30	2.95	3.25	1.00
NASA4704	SAINV-pf	2.54	1.00	2453	0.31	3.70	4.00	1.23
	ISAINV	1.97	0.78	1122	0.20	1.97	2.17	0.67
	SAINV	46.9	1.00	9063	18.0	526	544	1.00
NASASRB	SAINV-pf	46.9	1.00	12634	17.6	528	545	1.00
	ISAINV	40.2	0.86	7638	14.2	361	375	0.69
	SAINV	43.2	1.00	3609	37.8	67.8	106	1.00
TUBE1-2	SAINV-pf	43.2	1.00	5806	37.8	82.6	120	1.13
	ISAINV	14.2	**0.33**	3486	4.38	56.3	60.7	**0.57**
	SAINV	188	1.00	1286	58.4	65.6	124	1.00
SMT	SAINV-pf	188	1.00	2066	58.5	90.0	149	1.20
	ISAINV	58.1	**0.31**	1610	10.6	72.4	83.0	0.67

た **ISAINV** は，テストしたどの行列に対しても，他の 2 つの近似逆行列より全体の計算時間が少ないことが分かる．すなわち，**ISAINV** は合計計算時間を大幅（表 3 では時間比：**0.48**〜0.73，表 4 では **0.55**〜0.77）に減少させた．一方，SAINV-pf では計算時間の短縮が見られないことが多かった．

まず，表 3 における **ISAINV** による改善の効果について述べる．行列 BCSSTK24 では，反復回数の減少の度合いは小さいけれども，前処理行列の非零要素数が少なくなったため，全体の計算時間（比：**0.57**）は激減した．行列 TUBE1-2 では，反復回数が 4134 回から 2,050 回に半減した．また，行列 SMT では，反復計算の時間はあまり（54.5 秒が 47.1 秒に減少）改善されなかったが，前処理の時間は 61.0 秒が 24.5

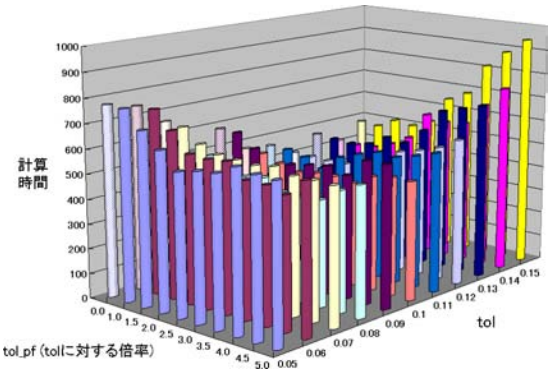

図 2 閾値 tol，tol_pf（tol に対する倍率で表示）を変化させたときの前処理 SAINV-pf つき CG 法の計算時間（単位：秒）の傾向（行列 NASASRB のとき）

Fig. 2 Tendency of computation time (in sec.) of CG method with SAINV-pf for matrix NASASRB.

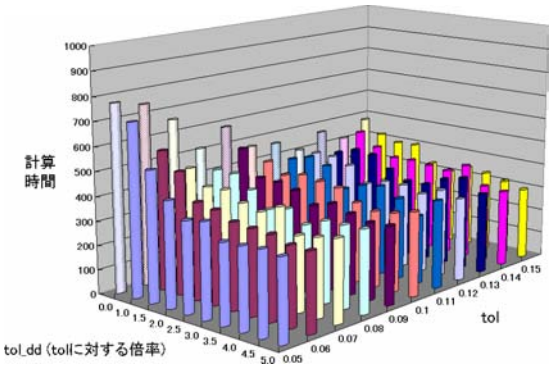

図 3 閾値 tol，tol_dd（tol に対する倍率で表示）を変化させたときの提案した前処理 **ISAINV** つき CG 法の計算時間（単位：秒）の傾向（行列 NASASRB のとき）

Fig. 3 Tendency of computation time (in sec.) of CG method with **ISAINV** for matrix NASASRB.

秒へと大きく減少した．さらに，テストした 6 個の行列すべてにおいて，近似分解に必要なメモリ量がかなりの割合（比：**0.32**〜0.83）で削減された．

次に，表 4 における **ISAINV** による改善の効果について述べる．ここでも，行列 BCSSTK24 と TUBE1-2 において，**ISAINV** の計算時間（比：**0.55** と **0.57**）が減少したことが分かる．また，メモリ量についても，行列 BCSSTK24，TUBE1-2，SMT において，大きく（比：**0.31**〜**0.35**）削減されたことが分かる．

数値実験で調べた閾値 tol の範囲は，文献 5)，6) を参考に定めた．この範囲内において，最も計算時間が大きかった最悪のケースは，**ISAINV** の合計計算時間が，行列 BCSSTK24 において 2.31 秒（比：0.73），行列 NASA4704 では 2.67 秒（比：0.87）の場合であった．その他の行列についても同程度の比率が観察された．このことから，前処理 **ISAINV** は，従来の前処理 SAINV および SAINV-pf と比較して，ほとんどの場合において収束性の改善に大幅な効果があることが分かる．

図 2 に，行列 NASASRB に対する前処理 SAINV-pf つき CG 法の閾値 tol と tol_pf の値（tol に対する倍率で表示）ごとの計算時間の傾向を示す．同様に，**図 3** に，行列 NASASRB に対する提案した前処理 **ISAINV** つき CG 法の閾値 tol と tol_dd の値（tol に対する倍率で表示）ごとの計算時間の傾向を示す．いずれの図でも，高さ方向は計算時間（単位：秒）を表す．ここで，tol_pf，tol_dd の値が 0.0 の場合（図の中で左側奥に位置する）は SAINV の実験結果にあたる．

図 2 より，SAINV-pf では，近似分解で用いる閾値 tol の値が小さいとき，閾値 tol_pf の値を変えても計算時間はあまり変動しないこと，一方，閾値 tol の値が大きいとき，閾値 tol_pf の値の選び方によって計算時間が大きく変動することが分かる．最少の計算時間のケースは，図の左中奥の方に位置する閾値 tol = 0.12 と tol_pf = 2.5（倍率）の値のときに得られた．また，計算時間が比較的少ないのは，閾値 tol_pf の値が閾値 tol の値のおよそ 1.0 倍から 2.0 倍の大きさの値のときであることが分かる．

図 3 より，**ISAINV** では，閾値 tol の値にかかわらず全体の計算時間が SAINV-pf に比べて全般的に少ないことが分かる．特に，その中でも計算時間が少ないのは，閾値 tol_dd の値を閾値 tol の値の 3.5〜5 倍に選んだときである．これは，実験で調べたその他の行列の場合も同程度の値であった．

このため，上で得られた倍率は，実際の問題に **ISAINV** つき CG 法を適用するとき，閾値 tol_dd の妥当な値を選ぶうえで非常に参考になると思われる．以上のことから，double dropping を行う **ISAINV** 前処理は，調べたすべてのテスト行列に対して，CG 法の収束までの計算時間の短縮およびメモリ量の削減に非常に有効な前処理であることが分かった．また，double dropping で新たに導入された閾値 tol_dd の値の決定は，実験で得られた 1 つの目安（閾値 tol の 3 倍程度）に従って行えばよいことが分かった．

5. ま と め

安定化近似逆行列による近似分解の中で二重に dropping 処理を行う **ISAINV** 前処理を提案した．その結果，**ISAINV** は，近似分解に要する時間と CG 法の収束までの時間との合計時間において，時間の短縮に

きわめて有効な前処理であることが分かった．また，時間の短縮だけでなく，必要なメモリ量の削減にも大きな効果があることも分かった．さらに，**ISAINV** では，double dropping 処理で使用する 2 つの閾値（tol，tol_dd）のうち，後者の新しく導入された閾値 tol_dd の値は，従来の閾値 tol の値の 3.5〜5 倍程度の値を与えればより少ない計算時間で済むケースが多いことが分かった．したがって，適切な閾値の決定に要する手間は従来の SAINV 前処理の場合とほとんど同程度であり，このため **ISAINV** は実用的な前処理であるといえる．

今後は，**ISAINV** 前処理を実際のアプリケーションの問題に適用し性能評価を行う予定である．

謝辞 安定化近似逆行列について，多くの有益なるご助言をいただいた M. Benzi 助教授および M. Tuma 博士に心より感謝の意を表す．また，論文を丁寧に読んでいただき有益なコメントをいただいた匿名の査読者に感謝の意を表する．

参 考 文 献

1) Alléon, G., Benzi, M. and Giraud, L.: Sparse approximate inverse preconditioning for dense linear systems arising in computational electromagnetics, *Numerical Algorithms*, Vol.16, pp.1–15 (1997).
2) Benzi, M.: A direct row-projection method for sparse linear systems, Ph.D. Thesis, North Carolina State University (1993).
3) Benzi, M., Meyer, C.D. and Tuma, M.: A sparse approximate inverse preconditioner for the conjugate gradient method, *SIAM J. on Scientific Computing*, Vol.17, pp.1135–1149 (1996).
4) Benzi, M., Kouhia, R. and Tuma, M.: An assessment of some preconditioning techniques in shell problems, *Commun. Numer. Meth. Engrg.*, Vol.14, pp.897–906 (1998).
5) Benzi, M., Cullum, J.K. and Tuma, M.: Robust approximate inverse preconditioning for the conjugate gradient method, *SIAM J. on Scientific Computing*, Vol.22, pp.1318–1332 (2000).
6) Benzi, M. and Tuma, M.: A robust incomplete factorization preconditioner for positive definite matrices, *Numer. Lin. Alg. Appl.*, Vol.10, pp.385–400 (2003).
7) Benzi, M., Kouhia, R. and Tuma, M.: Stabilized and block approximate inverse preconditioners for problems in solid and structual mechanics *Comput. Methods Appl. Mech. Engrg.*, Vol.190, pp.6533–6554 (2001).
8) Carpentieri, B., Duff, I.S. and Giraud, L.: Sparse pattern selection strategies for robust Frobenius norm minimization preconditioners in electromagnetism, *Numer. Lin. Alg. Appl.*, Vol.7, pp.667–685 (2000).
9) 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).
10) Kharchenko, S.A., Kolotilina, L.Y., Nikishin, A.A. and Yeremin, A.Yu.: A robust AINV-type method for constructing sparse approximate inverse preconditioners in factored form, *Numer. Lin. Alg. Appl.*, Vol.8, pp.165–179 (2001).
11) Kouhia, R. Sparse Matrices web page. http://www.hut.fi/∼kouhia/sparse.html
12) Matrix Market web page. http://math.nist.gov/MatrixMarket/
13) Mazzia, A. and Pini, G.: Numerical performance of preconditioning techniques for the solution of complex sparse linear systems, *Commun. Numer. Meth. Engrg.*, Vol.19, pp.37–48 (2003)
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, *Mathematics of Computation*, Vol.31, pp.148–162 (1977).
15) University of Florida Sparse Matrix web page. http://www.cise.ufl.edu/research/sparse/matrices/

(平成 15 年 5 月 12 日受付)
(平成 15 年 9 月 25 日採録)

池田 優介

1979 年生．2001 年九州大学工学部情報工学科卒業．現在同大学院修士課程在籍．共役勾配法の前処理と並列化に興味を持つ．

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．著書：『反復法の数理』（共著），朝倉書店．日本応用数理学会会員．