数理解析研究所講究録
第1573巻 2007年 124-138

Inverse-based ドロッピング手法つき ICCG 法の性能評価

塩出 亮 * 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

概要. クラウト (Crout) 版 ILU 前処理には，上三角行列 U の逆行列 U^{-1} の列ごとのノルムの大きさを見積もるドロッピング技法を採用することができるという利点がある．本論文では，対称正定値行列を係数に持つ線形方程式に対して，上記の逆行列をベースにしたドロッピング技法をどのように拡張するかについて記述する．さらに，数値実験を通じて，逆行列をベースにしたドロッピング技法の持つ優れた特長を明らかにする．

Performance estimation of ICCG method with Inverse-based dropping

Akira Shiode* Seiji Fujino**
*Graduate School of Information Science and Electrical Eng., Kyushu Univ.
**Computing and Communications Center, Kyushu University

Abstract. The Crout variant of ILU preconditioner devised recently has been to have some advantages over conventional row-based ILU preconditioner. One of advantages is to be able to adopt a dropping strategy for estimating column norm of inverse U^{-1}. This paper shows how to extend this inverse-based dropping strategy to solve a linear system of equations with symmetric positive definite matrix. Furthermore this paper reveals significant characteristics of inverse-based dropping strategy through numerical experiments.

1. はじめに

連立一次方程式 $Ax = b$ を解くことを考える．ここで A は対称正定値行列 $A = (a_{i,j}) \in R^{n\times n}$，解ベクトル $x \in R^n$，右辺ベクトル $b \in R^n$ とする．このような連立一次方程式を反復法で解く場合，前処理として IC(Incomplete Cholesky，以下 IC と略す) 分解が広く用いられている．IC 分解では，連立一次方程式の係数行列 A をよく近似した前処理行列 $M = U^T U(\approx A)$ を生成する (ただし，U は上三角行列 $U = (u_{i,j}) \in R^{n\times n}$ とする)．さらに，閾値による IC 分解 (IC with tolerance : 以下，IC(tol) 分解と略す) [14] では，フィルインに対して，予め定めた閾値 tol より小さいフィルインに対してはそれをドロッピングしながら，分解行列 U を計算していく．そして，生成した前処理行列の逆行列 $M^{-1} = (U^T U)^{-1}$ を係数行列に作用させて係数行列の対称性を保存するように変換し，変換された後の方

程式

$$(U^{-T}AU^{-1})(Ux) = U^{-T}b \tag{1.1}$$

に対して反復法を適用する．その際，閾値の大きさによっては，反復法の収束性に大きなばらつきが生じることがある．この理由は，従来のドロッピングつき IC(*tol*) 分解では，「行列 U の誤差行列 $\tilde{X}$ が小さい場合には，その逆行列 U^{-1} の誤差行列 X もおのずと小さいであろう」という前提で不完全分解がなされてきたことに起因する [3] [4] [5]．しかし，その理論的な必然性はなく，逆行列 U^{-1} の誤差行列 X が偶然大きい場合に，その影響により前処理つき反復法の収束性が悪化している危険性があった．しかし，逆行列 U^{-1} の誤差行列 X を計算する適当な計算手段が今まで存在しなかった．

近年，非対称行列用の前処理として Crout 版 ILU 分解 (以下，ILUC 分解と略す) が提案された [7]．原論文では，従来の ILUT(dual Threshold ILU) 分解 [12] [13] との収束性の違いが明らかにされ，さらに逆行列 U^{-1} の誤差行列 X の具体的な評価方法について Inverse-based ドロッピング手法 と呼ばれる方法が提案された．元々逆行列 L^{-1} の行ノルムあるいは逆行列 U^{-1} の列ノルムを利用したドロッピング手法は最初文献 [3] で発表された．その後，ILUC 分解は，文献 [8] [9] などでピボッティング操作が組み入れられるなど盛んに研究がなされている．

本研究の目的は，Inverse-based ドロッピング手法の考え方を対称行列に適用すること，さらに原論文では必ずしも十分明らかにされなかった Inverse-based ドロッピング手法が持つ収束特性をより明確にすることである．

2. IC(*tol*) 分解について

IC(*tol*) 分解 [14] は次のように表される．

$$A = U^T U + R + R^T.$$

ただし，行列 U，R は各々上三角行列とし，行列 R は分解の不完全さを表す形式的な行列で，分解途中で棄却 (ドロッピング) された分解行列 U の要素を含む．また，実際の前処理つき反復法のアルゴリズム中には現れない．IC(*tol*) 分解を行うとき，予め閾値として用いるパラメータ *tol* を設定し，分解の対象の行列 U の要素の大きさが閾値 *tol* よりも大きいときはその要素の計算を実行し，逆に小さいときはその要素をドロッピングして (要素の値を零とおいて) 計算を進めることにする．ただし，実際は，係数行列 A はスパース行列のための圧縮列格納形式 (以下，CCS 形式と略す) [1] で保存されるので，CCS 形式によく対応した U^T を計算により求めることにする．以降，アルゴリズム中では，便宜上行列 U^T を下三角行列 $L = (l_{j,k})$ で表現する．

IC(*tol*) 分解では，下三角行列 U^T の各列 $k = 1, \ldots, n$ に対して，次の手順で計算を行う．ただし，要素 $a^*_{j,k}$ は分解過程においては作業用配列として使われ，分解が終了した後は行

列 U^T の非対角要素になる要素を表す．一方，従来の IC(tol) 分解では，対角要素 $\overline{a_{j,j}}$ は $\overline{a_{j,j}} = a_{j,j}$ と置かれる．

[IC(tol) 分解の手順]

$$a^*_{j,k} = a_{j,k} - \sum_{m=1}^{k-1} l_{j,m} l_{k,m}, \quad (j = k+1, \ldots, n)$$

$$l_{k,k} = \sqrt{\overline{a_{k,k}} - \sum_{m=1}^{k-1} l^2_{k,m}},$$

$$l_{j,k} = \begin{cases} a^*_{j,k} \,/\, l_{k,k}, & (\frac{|a^*_{j,k}|}{\sqrt{(a_{j,j})(a_{k,k})}} > tol \text{ のとき}) \ (j = k+1, \ldots, n) \\ 0. & (\frac{|a^*_{j,k}|}{\sqrt{(a_{j,j})(a_{k,k})}} \leq tol \text{ のとき}) \ (j = k+1, \ldots, n) \end{cases}$$

しかし，場合によっては，式 (2.1) の右辺の平方根の中の式の値が負になり，分解が破綻することがある．分解が破綻することを防ぐための手法として，シフト IC(tol) 分解がある．シフト IC(tol) 分解では，シフト量と呼ばれる一定のスカラ値 α として，予め係数行列 A の対角項全てに α を加える．シフト IC(tol) 分解は次のように表される．

$$\tilde{A} = A + \alpha I = U^T U + R + R^T. \tag{2.1}$$

ただし，I は単位行列とする．

3. IC(tol) 分解で生じる誤差に対する考察

IC(tol) 分解つき反復法では，解くべき方程式を式 (1.1) に変換して解くのが一般的である．このとき，行列 A を完全 (コレスキー) 分解して (閾値 tol を 0 にして) 得られた上三角行列を $\bar{U}$ とするとき，IC(tol) 分解により得られた分解行列 U は誤差行列 $\tilde{X}$ を用いて次のように表せる．

$$U^T = \bar{U}^T + \tilde{X}^T, \quad U = \bar{U} + \tilde{X}. \tag{3.1}$$

また，行列 U の逆行列 U^{-1} は別の誤差行列 X を用いて次のように表せる．

$$U^{-T} = \bar{U}^{-T} + X^T, \quad U^{-1} = \bar{U}^{-1} + X. \tag{3.2}$$

これらの式 (3.1)，(3.2) を使うと，前処理後の係数行列 $U^{-T}AU^{-1}$ は次のように表せる．

$$U^{-T}AU^{-1} = I + \bar{U}X + X^T\bar{U}^T + X^TAX. \tag{3.3}$$

ここで，式 (3.3) 中に式 (3.1) で定義した誤差行列 $\tilde{X}$ が現れていないことに注意を要する．したがって，一般に行列 U の誤差行列 $\tilde{X}$ が小さい場合に逆行列 U^{-1} の誤差行列 X も小さ

いという保証はまったくなく，式 (3.3) から，もし逆行列 U^{-1} の誤差行列 X が大きい場合には前処理つき反復法の収束性に影響を及ぼす可能性があることが分かる．

さらに，より実用的なシフト IC(tol) 分解で生じる誤差についても考える．式 (2.1) の行列 $\tilde{A}$ を完全分解して得られた上三角行列を $\bar{U}_\alpha$，シフト量に関する誤差行列 Y とするとき，シフト量を加える前の係数行列 A は次のように表せる．

$$A = \bar{U}^T\bar{U} = \bar{U}_\alpha^T\bar{U}_\alpha + Y(= \tilde{A} + Y). \tag{3.4}$$

シフト IC(tol) 分解により得られた分解行列 U_α は式 (3.1) と同様に，誤差行列 $\tilde{X}_\alpha$ を用いて次のように表せる．

$$U_\alpha^T = \bar{U}_\alpha^T + \tilde{X}_\alpha^T, \quad U_\alpha = \bar{U}_\alpha + \tilde{X}_\alpha \tag{3.5}$$

また，行列 U_α の逆行列 U_α^{-1} は式 (3.2) と同様に，別の誤差行列 X_α を用いて次のように表せる．

$$U_\alpha^{-T} = \bar{U}_\alpha^{-T} + X_\alpha^T, \quad U_\alpha^{-1} = \bar{U}_\alpha^{-1} + X_\alpha. \tag{3.6}$$

これらの式 (3.5)，(3.6) を使うと，前処理後の係数行列 $U_\alpha^{-T}AU_\alpha^{-1}$ は次のように表せる．

$$\begin{aligned} U_\alpha^{-T}AU_\alpha^{-1} &= (\bar{U}_\alpha^{-T} + X_\alpha^T)(\tilde{A} + Y)(\bar{U}_\alpha^{-1} + X_\alpha) \\ &= I + \bar{U}_\alpha X_\alpha + X_\alpha^T\bar{U}_\alpha^T + X_\alpha^T\tilde{A}X_\alpha \\ &\quad + \bar{U}_\alpha^{-T}Y\bar{U}_\alpha^{-1} + \bar{U}_\alpha^{-T}YX_\alpha + X_\alpha^TY\bar{U}_\alpha^{-1} + X_\alpha^TYX_\alpha. \end{aligned}$$

シフト IC(tol) 分解では，誤差行列 X_α が大きくなれば，誤差行列 Y の与える影響が大きくなることが分かる．

以上から，ドロッピング操作が前処理行列の逆行列 U^{-1} に与える影響の見積もり，すなわち，U^{-T} の行のノルム (U^{-1} の列のノルム) をできるだけ正確に見積もり，それに基づきドロッピングする IC(tol) 分解の方が，より適切なドロッピングができると考えられる．

4. Inverse-based ドロッピング手法に基づく IC(tol) 分解

Inverse-based ドロッピング手法に基づく IC(tol) 分解 (以下，ib_IC(tol) 分解と略す) は，前節で述べたように，前処理行列のノルムの見積もりにより適切なドロッピングを行う IC(tol) 分解である．Table 1 に従来の IC(tol) 分解と ib_IC(tol) 分解における誤差行列およびドロッピングのとき対象となる行列の違いを示す．さて，ib_IC(tol) 分解について，これから具体的に説明を行っていく．ただし，便宜上，行列 $\bar{U}^T = (\bar{u}_{j,i})$ を下三角行列 $\bar{L} = (\bar{l}_{i,j})$ を用いて表現する．

まず，行列 L_k を，1 から k 列目まで前処理行列 L の 1 から k 列目までと同じ要素を持ち，$k+1$ から n 列目まで要素 $\overline{a_{i,i}}$ $(1 \le i \le n)$ を対角要素に持つ対角行列と同じ要素を持つ

Table 1. Error matrix and object matrix of dropping strategy in the conventional IC(*tol*) and ib_IC(*tol*) decompositions.

decompositions	error matrix $\tilde{X}$ of U and error matrix X of U^{-1}	matrices for dropping
IC(*tol*)	$U \equiv \bar{U} + \tilde{X}$, $U^{-1} \equiv \bar{U}^{-1} + X$	$U = \bar{U} + \tilde{X}$
ib_IC(*tol*)	same as above	$U^{-1} = \bar{U}^{-1} + X$

次のような下三角行列を考える．

$$L_k = \begin{pmatrix} l_{1,1} & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ l_{2,1} & \ddots & \ddots & & & & \vdots \\ \vdots & \vdots & l_{k,k} & \ddots & & & \vdots \\ \vdots & \vdots & \vdots & \overline{a_{k+1,k+1}} & \ddots & & \vdots \\ \vdots & \vdots & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ l_{n,1} & \cdots & l_{n,k} & 0 & \cdots & 0 & \overline{a_{n,n}} \end{pmatrix}.$$

行列 L の k 列目を生成するとき，初めてフィルイン $l_{j,k}$ $(j > k)$ が棄却されたと仮定する．このとき，行列 L_k は次のように表現できる．ただし，行列 $\bar{L}_k$ は，行列 L の k 列目を生成するときにフィルイン $l_{j,k}$ $(j > k)$ を棄却しなかった場合に相当する行列である．すなわち，行列 $\bar{L}_k$ は係数行列 A を k 列目まで完全分解した行列である．また，単位ベクトル e_i $(1 \le i \le n)$ は i 行目が 1 で，i 以外の行はすべて 0 の列ベクトルとする．

$$L_k = \bar{L}_k - l_{j,k} e_j e_k^T.$$

$j > k$ より，$\bar{L}_k e_j = \overline{a_{j,j}} e_j$ であることに注意すると，

$$L_k = \bar{L}_k - l_{j,k} e_j e_k^T = \bar{L}_k (I - l_{j,k} e_j e_k^T / \overline{a_{j,j}}) \tag{4.1}$$

である．式 (4.1) より，行列 L_k^{-1} を次のように表現できる．

$$\begin{aligned} L_k^{-1} &= (I - l_{j,k} e_j e_k^T / \overline{a_{j,j}})^{-1} \bar{L}_k^{-1} \\ &\approx \bar{L}_k^{-1} + l_{j,k} e_j e_k^T \bar{L}_k^{-1} / \overline{a_{j,j}}. \end{aligned}$$

したがって，**行列 L_k の逆行列 L_k^{-1} は $l_{j,k}/\overline{a_{j,j}}$ と $\bar{L}_k^{-1}$ の k 行目の積の値だけ影響を受ける**ことが分かる．すなわち，$\|l_{j,k} e_j e_k^T \bar{L}_k^{-1} / \overline{a_{j,j}}\|_\infty = |l_{j,k}/\overline{a_{j,j}}| \|e_j e_k^T \bar{L}_k^{-1}\|_\infty$ でドロッピングを行う

ことにより，ドロッピングにより影響を受ける逆行列の行のノルムを限定することができる．ただし，$\|A\|_\infty$ は行列 A の最大ノルムで，

$$\|A\|_\infty = \max_{\boldsymbol{x}\neq 0} \frac{\|A\boldsymbol{x}\|_\infty}{\|\boldsymbol{x}\|_\infty} = \max_i \sum_{j=1}^{n} |a_{i,j}| \tag{4.2}$$

のように定義される．また，$\|\boldsymbol{e}_j \boldsymbol{e}_k^T \bar{L}_k^{-1}\|_\infty$ はすべての要素が 0 でないあるベクトル $\boldsymbol{b}$ を用いて，次のように近似できる．ここで，行列 $\bar{L}_k^{-1}$ の k 行目の要素は，行列 $\bar{L}^{-1}$ の k 行目の要素と各々一致し，$\|\boldsymbol{e}_j \boldsymbol{e}_k^T \bar{L}_k^{-1}\|_\infty$ と $\|\boldsymbol{e}_j \boldsymbol{e}_k^T \bar{L}^{-1}\|_\infty$ は各々 k 行目の要素の絶対値の総和であることに注意を要する．

$$\begin{aligned} \|\boldsymbol{e}_j \boldsymbol{e}_k^T \bar{L}_k^{-1}\|_\infty &= \|\boldsymbol{e}_j \boldsymbol{e}_k^T \bar{L}^{-1}\|_\infty \\ &= \max_{\boldsymbol{b}\neq 0} \frac{\|\boldsymbol{e}_j \boldsymbol{e}_k^T \bar{L}^{-1}\boldsymbol{b}\|_\infty}{\|\boldsymbol{b}\|_\infty}. \end{aligned} \tag{4.3}$$

したがって，$\|\boldsymbol{e}_j \boldsymbol{e}_k^T \bar{L}^{-1}\|_\infty$ を推定するためには，式 (4.3) を満たすようなベクトル $\boldsymbol{b}(\neq \boldsymbol{0})$ を決定すればよい．最大ノルムの定義式 (4.2) において，$i = m$ のとき，

$$\sum_{j=1}^{n} |a_{i,j}| \tag{4.4}$$

が最大になったとすると，その最大値を与えるベクトル $\boldsymbol{x}$ の第 j 成分は，

$$x_j = \mathrm{sign}(a_{m,j}), \quad (j = 1, 2, \ldots, n) \tag{4.5}$$

となることが知られている [10] [11]．ただし，sign は，

$$\mathrm{sign}(t) = \begin{cases} +1, & (t \geq 0) \\ -1 & (t < 0) \end{cases}$$

で定義される符号関数である．しかし，式 (4.3) では，行列 $\bar{L}^{-1}$ の計算をせず，ベクトル $\boldsymbol{b}$ の要素の符号を決定できない．そこで，

$$\begin{aligned} \boldsymbol{b} &= (-b_1, -b_2, \ldots, -b_n)^T, \\ & (b_1, b_2, \ldots, b_n = \pm 1) \end{aligned} \tag{4.6}$$

の中から，$\|\boldsymbol{e}_j \boldsymbol{e}_k^T \bar{L}^{-1}\boldsymbol{b}\|_\infty$ をできるだけ大きくするベクトル $\boldsymbol{b}$ を探すことを考える．このようなベクトル $\boldsymbol{b}$ を見つければ，次の近似式が成り立つ．

$$\|\boldsymbol{e}_j \boldsymbol{e}_k^T \bar{L}^{-1}\|_\infty \approx \frac{\|\boldsymbol{e}_j \boldsymbol{e}_k^T \bar{L}^{-1}\boldsymbol{b}\|_\infty}{\|\boldsymbol{b}\|_\infty}. \tag{4.7}$$

式 (4.7) の右辺項は，$\bar{L}\boldsymbol{\xi} = \boldsymbol{b}$ の解 $\boldsymbol{\xi}$ の第 k 成分で評価でき，ベクトル $\boldsymbol{\xi}$ の第 k 成分 ξ_k は，右辺ベクトル $\boldsymbol{b}$ の第 k 成分を用いた次の式で求められる．ただし，$\boldsymbol{\xi}_k = (\xi_1, \xi_2, \ldots, \xi_k, 0, \ldots, 0)^T$ とする．

$$\xi_k = (b_k - \boldsymbol{e}_k^T \bar{L}_{k-1} \boldsymbol{\xi}_{k-1}) / l_{k,k}.$$

以上から，$\|e_j e_k^T \bar{L}^{-1}\|_\infty$ を見積もる次のアルゴリズムが得られる．ただし，アルゴリズム中の ξ_k と ν_k は，$\|e_j e_k^T \bar{L}^{-1}\|_\infty$ と $e_k^T \bar{L}_{k-1} \xi_{k-1}$ に各々対応する．

Algorithm 1: Estimating the norms $\|e_j e_k^T \bar{L}^{-1}\|_\infty$

set $\xi_1 = 1/l_{1,1}$; $\nu_i = 0$ $(i = 1, \ldots n)$
for $k = 2, n$
  $temp_+ = 1 - \nu_k$
  $temp_- = -1 - \nu_k$
  if $|temp_+| > |temp_-|$ then
    $\xi_k = temp_+ / l_{k,k}$
  else
    $\xi_k = temp_- / l_{k,k}$
  end if
  for $j = k+1, n$ $(l_{j,k} \neq 0)$
    $\nu_j = \nu_j + \xi_k l_{j,k}$
  end for
end for.

従来のドロッピング手法では，フィルイン $l_{j,k}$ に対して，

$$|l_{j,k}| / \sqrt{a_{j,j}} = |a^*_{j,k}| / \sqrt{(a_{j,j})(a_{k,k})} \leq tol \tag{4.8}$$

のとき棄却する．一方，Inverse-based ドロッピング手法では，Algorithm 1 に従いフィルイン $l_{j,k}$ に対して，

$$|l_{j,k}/\overline{a_{j,j}}| \|e_j e_k^T \bar{L}^{-1}\|_\infty = |l_{j,k}||\xi_k| / |\overline{a_{j,j}}| = |a^*_{j,k}||\xi_k| / (|\overline{a_{j,j}}| \sqrt{a_{k,k}}) \leq tol \tag{4.9}$$

のとき棄却する．

5. 数値実験

テスト行列は，Kouhia の疎行列データベース [6] から行列 S3DKQ4M2，S3DKT3M2 の合計 2 種類の実対称正定値行列を選出した．これらのテスト行列の主な特徴を Table 2 に示す．なお，行列 S3DKQ4M2 の問題はシリンダー (薄肉モデル) を四角形状シェル要素でメッシュ分割して得られたものであり，一方，行列 S3DKT3M2 はシリンダーを三角形状シェル要素でメッシュ分割して得られたものである [2] [6].

数値実験は，Fortran90 でプログラムを実装し，計算はすべて実数倍精度演算，コンパイラの最適化オプションは-O3，そして，CPU に Pentium4 (クロック周波数 3.8GHz) を

搭載したPCで行った．初期近似解 $\boldsymbol{x}_0 = \boldsymbol{0}$ とした．最大反復回数は行列の次元数と同じ値にした．2種類のテスト行列は予めそれらの対角項を1に揃える正規化を各々行った．収束判定値は相対残差の2ノルム：$\|\boldsymbol{r}_m\|_2/\|\boldsymbol{r}_0\|_2 \leq 10^{-8}$ とした．シフトIC(*tol*)分解，シフトib_IC(*tol*)分解において閾値 *tol* は0.005から0.15まで0.005刻み(合計30通り)で変化させ，シフト量 α は0.0から0.2まで0.02刻み(合計11通り)で変化させた($\alpha = 0$ のときIC(*tol*)分解，ib_IC(*tol*)分解に各々相当する)．

行列S3DKQ4M2，S3DKT3M2に対するシフトIC(*tol*)分解つきCG法(以下，SIC(*tol*)-CG法と略す)とシフトib_IC(*tol*)分解つきCG法(以下，S ib_IC(*tol*)-CG法と略す)の反復回数，および計算時間をTable 3，4に各々示す．ただし，計算時間“time”は前処理行列の生成に要した時間とCG法の反復計算に要した時間の合計とし，“min”，“max”は各々反復回数(計算時間)の最小値，最大値，“ave.”は反復回数(計算時間)を総試行数(分解が破綻したときは除く)で割った値とする．

また，Fig. 1-4に行列S3DKQ4M2，S3DKT3M2に対するSIC(*tol*)-CG法とS ib_IC(*tol*)-CG法の閾値 *tol* とシフト量 α 変化毎の反復回数を各々示す．ただし，棒グラフが表示されていないところは分解が途中で破綻したことを表す．

さらに，Table 5，6に行列S3DKQ4M2，S3DKT3M2に対するSIC(*tol*)-CG法とS ib_IC(*tol*)-CG法のシフト量 α 変化毎の合計時間とフィルイン数を各々示す．ただし，“break”は分解が途中で破綻したことを表す．

Table 2. Specification of tested real symmetric positive definite matrices.

matrix	dimensions	total nnz	max bandwidth	ave. bandwidth	analysis
S3DKQ4M2	90,449	2,455,670	614	607.87	cylindrical shells
S3DKT3M2	90,449	1,921,955	614	607.87	same as above

[行列 S3DKQ4M2 の問題]

まず，Table 3，Fig. 1，2から次のことが観察できる．

- 反復回数の最小値，最大値，平均値について，S ib_IC(*tol*)-CG法の方がSIC(*tol*)-CG法より少なかった．また，SIC(*tol*)-CG法では，最大反復回数(行列の次元数)まで達し収束しない場合があった．
- 計算時間の最小値，最大値，平均値について，最小値はSIC(*tol*)-CG法の方がS ib_IC(*tol*)-CG法より短く，最大値，平均値はS ib_IC(*tol*)-CG法の方がSIC(*tol*)-CG法より短かった．
- SIC(*tol*)-CG法の反復回数は *tol*，α の変化に対して不安定に変化した．すなわち，*tol* = 0.05，0.055，α = 0.06 のときと *tol* = 0.105，α = 0.14 のとき収束せず，

Table 3. Camparison of convergence property of shifted_IC(*tol*)-CG and shifted ib_IC(*tol*)-CG methods for matrix S3DKQ4M2.

method	iterations			time[s]		
	min	max	ave.	min	max	ave.
SIC(*tol*)-CG	2573	90449	10280	**80.07**	1931.07	208.90
S ib_IC(*tol*)-CG	**2470**	**12779**	**7879**	98.33	**291.52**	**172.54**

Table 4. Camparison of convergence property of shifted_IC(*tol*)-CG and shifted ib_IC(*tol*)-CG methods for matrix S3DKT3M2.

method	iterations			time[s]		
	min	max	ave.	min	max	ave.
SIC(*tol*)-CG	3155	47758	10578	**80.60**	793.50	179.45
S ib_IC(*tol*)-CG	**3024**	**12083**	**9288**	108.45	**251.55**	**170.31**

$tol = 0.05$, $\alpha = 0.05$, 0.07 のとき，$tol = 0.1$，$\alpha = 0.12$, 0.14, 0.16, 0.18 のとき，$tol = 0.105$，$\alpha = 0.16$ のとき反復回数は極端に増加した．

- S ib_IC(*tol*)-CG 法の反復回数は *tol*，α の変化に対して安定に変化した．また，収束する *tol*，α の範囲は S ib_IC(*tol*)-CG 法の方が広かった．

次に，Table 5 から次のことが観察できる．

- フィルイン数について，SIC(*tol*)-CG 法と S ib_IC(*tol*)-CG 法はともに，α の変化に対して単調に減少した．また，同じ *tol* 下で見ると，SIC(*tol*)-CG 法の方が S ib_IC(*tol*)-CG 法よりもフィルイン数は少なく，
  - $tol = 0.01$ のとき，$\alpha = 0.02$ では 40% 程度，$\alpha = 0.2$ では 15% 程度少なかった．
  - $tol = 0.05$ のとき，$\alpha = 0.06$ では 45% 程度，$\alpha = 0.2$ では 30% 程度少なかった．
  - $tol = 0.1$ のとき，$\alpha = 0.12$ では 55% 程度，$\alpha = 0.2$ では 45% 程度少なかった．
  - $tol = 0.15$ のとき，$\alpha = 0.12$ では 65% 程度，$\alpha = 0.2$ では 50% 程度少なかった．
- 合計時間について，S ib_IC(*tol*)-CG 法は α の変化に対してほぼ単調に増加したが，SIC(*tol*)-CG 法は α の変化に対して不安定に変化した．すなわち，
  - $tol = 0.01$ のとき，$\alpha = 0.12$ では極端に長くなった．
  - $tol = 0.05$ のとき，$\alpha = 0.06$, 0.08, 0.12 では極端に長くなった．特に $\alpha = 0.06$ では収束しなかったことに注意を要する．
  - $tol = 0.1$ のとき，$\alpha = 0.2$ 以外では極端に長くなった．
  - $tol = 0.15$ のとき，α 変化毎に単調に減少した．

[行列 S3DKT3M2 の問題]

Table 5. Computational costs and fill-ins of shifted_IC(*tol*)-CG and shifted ib_IC(*tol*)-CG methods for matrix S3DKQ4M2.

method		$tol \setminus \alpha$	0	0.02	0.04	0.06	0.08	0.1
SIC(*tol*)-CG	time[s]	0.01	break	**94.18**	106.75	128.70	146.36	162.97
		0.05	break	break	break	1931.07	629.24	193.81
		0.1	break	break	break	break	break	break
		0.15	break	break	break	break	break	break
	fill-ins	0.01	break	2284825	2157016	2133905	2086374	2052945
		0.05	break	break	break	966971	940822	921011
		0.1	break	break	break	break	break	break
		0.15	break	break	break	break	break	break
S ib_IC(*tol*)-CG	time[s]	0.01	break	117.27	142.82	161.40	176.17	190.36
		0.05	break	124.59	127.79	129.65	143.47	151.68
		0.1	break	break	break	break	188.43	180.92
		0.15	break	break	break	break	break	break
	fill-ins	0.01	break	3907263	3355900	3061154	2891956	2770816
		0.05	break	2279139	1939862	1748260	1640113	1557717
		0.1	break	break	break	break	1180192	1084304
		0.15	break	break	break	break	break	break

method		$tol \setminus \alpha$	0.12	0.14	0.16	0.18	0.2
SIC(*tol*)-CG	time[s]	0.01	229.55	169.86	198.72	207.36	218.22
		0.05	231.13	163.86	164.57	170.57	172.46
		0.1	399.13	441.14	388.12	374.27	199.54
		0.15	217.57	208.76	206.85	206.12	204.54
	fill-ins	0.01	2036012	2015739	1987880	1964536	1940514
		0.05	915335	905667	898599	886802	866996
		0.1	431208	422579	416182	404241	363236
		0.15	263978	259769	254602	249731	243289
S ib_IC(*tol*)-CG	time[s]	0.01	200.33	210.77	219.77	228.20	237.06
		0.05	164.41	172.06	179.21	187.18	190.27
		0.1	169.26	170.28	177.73	186.10	196.75
		0.15	185.31	189.28	190.90	193.91	199.46
	fill-ins	0.01	2637831	2534709	2423589	2362957	2292991
		0.05	1471841	1386521	1316804	1252897	1202796
		0.1	997380	927628	863435	811100	772448
		0.15	714328	652406	597272	542731	489235

まず，Table 4，Fig. 3，4 から次のことが観察できる．

- 反復回数の最小値，最大値，平均値について，S ib_IC(*tol*)-CG 法の方が SIC(*tol*)-CG 法より少なかった．
- 計算時間の最小値，最大値，平均値について，最小値は SIC(*tol*)-CG 法の方が S ib_IC(*tol*)-CG 法より短く，最大値，平均値は S ib_IC(*tol*)-CG 法の方が SIC(*tol*)-CG 法より短かった．
- SIC(*tol*)-CG 法の反復回数は *tol*，α の変化に対して不安定に変化した．すなわち，$tol = 0.045$，$\alpha = 0.04$ のとき，$tol = 0.05$，0.095，$\alpha = 0.06$ のとき，$tol = 0.095$，0.1，$\alpha = 0.08$ のとき，$tol = 0.14$，$\alpha = 0.12$ のとき反復回数は極端に増加した．

Table 6. Computational costs and fill-ins of shifted_IC(*tol*)-CG and shifted ib_IC(*tol*)-CG methods for matrix S3DKT3M2.

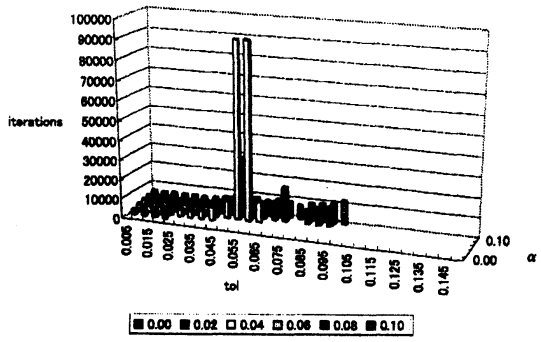
method		*tol* \ α	0	0.02	0.04	0.06	0.08	0.1
SIC(*tol*)-CG	time[s]	0.01	break	**90.56**	117.63	138.04	155.89	173.80
		0.05	break	break	break	261.91	159.73	158.77
		0.1	break	break	break	break	793.50	184.99
		0.15	break	break	break	break	break	break
	fill-ins	0.01	break	2051303	1955988	1897577	1871980	1839974
		0.05	break	break	break	960819	915296	888583
		0.1	break	break	break	break	578355	541842
		0.15	break	break	break	break	break	break
S ib_IC(*tol*)-CG	time[s]	0.01	break	121.94	150.68	171.30	189.20	201.53
		0.05	break	break	126.92	141.25	152.61	162.98
		0.1	break	break	break	150.57	156.93	165.80
		0.15	break	break	break	break	173.92	170.95
	fill-ins	0.01	break	3504019	2977610	2755614	2580402	2464296
		0.05	break	break	1782462	1616048	1536827	1463320
		0.1	break	break	break	1324728	1139248	1047882
		0.15	break	break	break	break	933093	888609

method		*tol* \ α	0.12	0.14	0.16	0.18	0.2
SIC(*tol*)-CG	time[s]	0.01	208.49	201.58	212.06	225.26	225.47
		0.05	159.88	166.82	160.43	168.32	173.32
		0.1	151.64	152.70	146.93	159.79	172.62
		0.15	245.00	244.57	239.16	206.24	178.56
	fill-ins	0.01	2223229	1806878	1801091	1792978	1785248
		0.05	889801	832307	801650	780472	771371
		0.1	495302	452591	421242	418544	416116
		0.15	325045	319906	314087	299998	297778
S ib_IC(*tol*)-CG	time[s]	0.01	215.21	227.34	238.66	247.24	256.09
		0.05	173.24	181.78	189.97	193.92	202.66
		0.1	173.14	181.66	184.53	188.20	199.14
		0.15	177.46	183.34	187.89	186.81	194.93
	fill-ins	0.01	2364873	2276295	2198236	2122333	2022427
		0.05	1379713	1304799	1244842	1181661	1152946
		0.1	995259	946366	907664	867144	830925
		0.15	835606	737033	674317	640193	609422

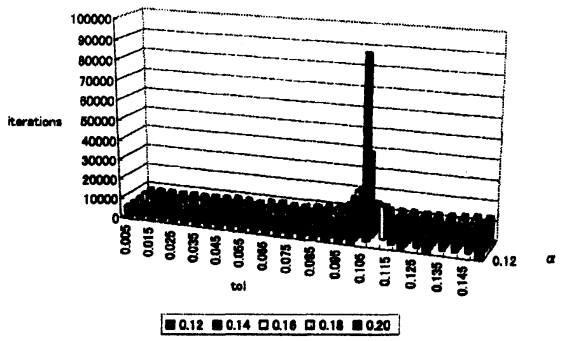
- S ib_IC(*tol*)-CG 法の反復回数は *tol*, α の変化に対して安定に変化した．また，収束する *tol*, α の範囲は S ib_IC(*tol*)-CG 法の方が広かった．

次に，Table 6 から次のことが観察できる．

- フィルイン数について，SIC(*tol*)-CG 法と S ib_IC(*tol*)-CG 法はともに，α の変化に対して単調に減少した．また，同じ *tol* 下で見ると，SIC(*tol*)-CG 法の方が S ib_IC(*tol*)-CG 法よりもフィルイン数は少なく，
  - *tol* = 0.01 のとき，α = 0.02 では 40% 程度，α = 0.2 では 10% 程度少なかった．
  - *tol* = 0.05 のとき，α = 0.06 では 40% 程度，α = 0.2 では 35% 程度少なかった．

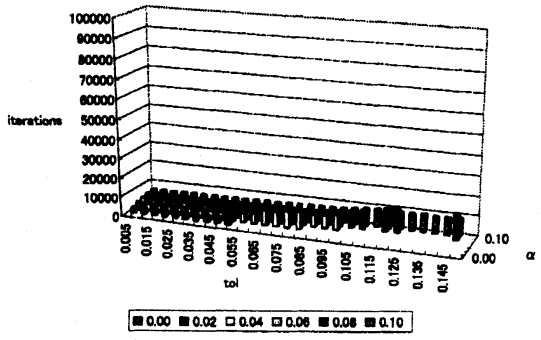


(a) $0 \leq \alpha \leq 0.1$

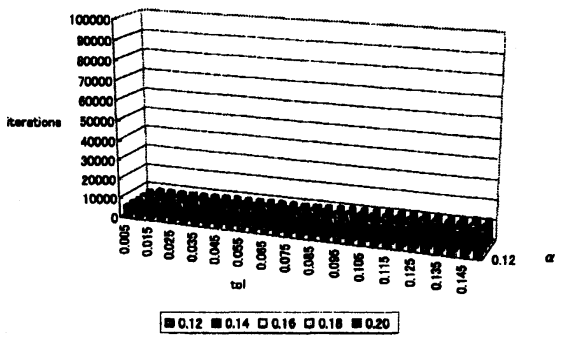


(b) $0.12 \leq \alpha \leq 0.2$

Fig. 1. Iterations versus tolerance value and shifted value of shifted_IC(*tol*)-CG method for matrix S3DKQ4M2.



(a) $0 \leq \alpha \leq 0.1$



(b) $0.12 \leq \alpha \leq 0.2$

Fig. 2. Iterations versus tolerance value and shifted value of shifted ib_IC(*tol*)-CG method for matrix S3DKQ4M2.

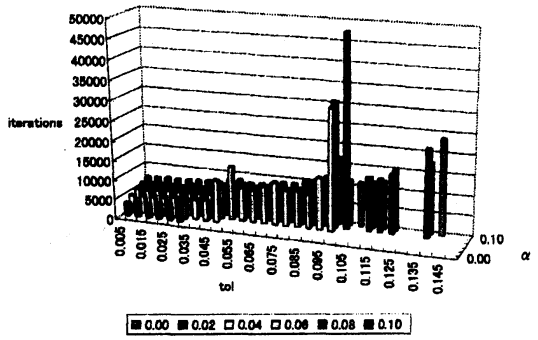
- $tol = 0.1$ のとき，$\alpha = 0.08$ では 50% 程度，$\alpha = 0.2$ では 50% 程度少なかった．
- $tol = 0.15$ のとき，$\alpha = 0.12$ では 60% 程度，$\alpha = 0.2$ では 50% 程度少なかった．

• 合計時間について，S ib_IC(*tol*)-CG 法は α の変化に対してほぼ単調に増加したが，SIC(*tol*)-CG 法は α の変化に対して不安定に変化した．すなわち，

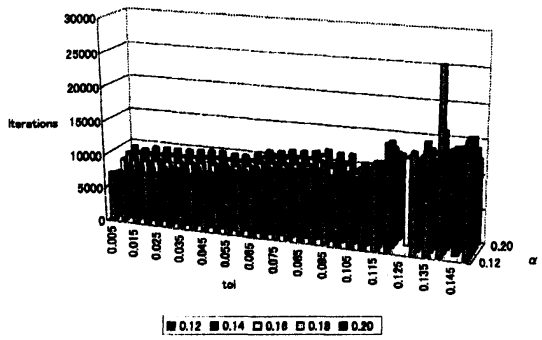
- $tol = 0.01$ のとき，ほぼ単調に増加した．
- $tol = 0.05$ のとき，$\alpha = 0.06$ では極端に長くなった．
- $tol = 0.1$ のとき，$\alpha = 0.08$ では極端に長くなった．
- $tol = 0.15$ のとき，α 変化毎に単調に減少した．

SIC(*tol*)-CG 法は，*tol*，α 変化毎に反復回数が不安定に変化する傾向が多く見られた．特に行列 S3DKQ4M2 の問題では，最大反復回数内に収束しない現象も見られた．

一方，S_ib_IC(*tol*)-CG 法は，*tol*，α が大きくなる毎に，反復回数は単調に増加する傾向が見られ，安定に変化することが分かった．また，必要メモリ量の観点から見ると，Table 5，6 では，α が大きくなる毎に，SIC(*tol*)-CG 法に対する S_ib_IC(*tol*)-CG 法の必要メモリ量の比は小さくなる傾向が見られた．ここで，行列 S3DKQ4M2，S3DKT3M2 の問題

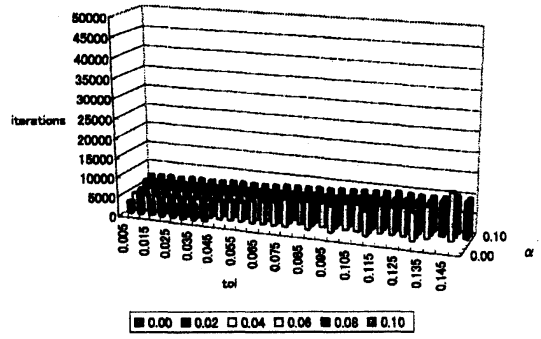


(a) $0 \leq \alpha \leq 0.1$

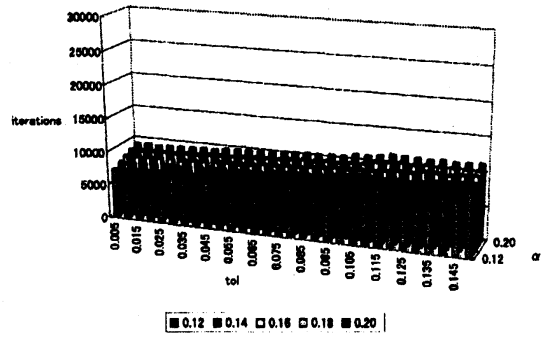


(b) $0.12 \leq \alpha \leq 0.2$

Fig. 3. Iterations versus tolerance value and shifted value of shifted_IC(*tol*)-CG method for matrix S3DKT3M2.



(a) $0 \leq \alpha \leq 0.1$



(b) $0.12 \leq \alpha \leq 0.2$

Fig. 4. Iterations versus tolerance value and shifted value of shifted ib_IC(*tol*)-CG method for matrix S3DKT3M2.

に対して，二つの解法の必要メモリ量が同程度になるまで α を大きくしたところ，Fig. 5 に示すような結果が各々得られた．ただし，棒グラフ，および折れ線グラフは各々反復回数 (左側縦軸)，フィルイン数 (右側縦軸) を表し，表示されていないところは分解が途中で破綻したことを表す．また，*tol* は最も必要メモリ量の少ない $tol = 0.15$ のときとした．

Fig. 5 (a) から次のことが観察できる．

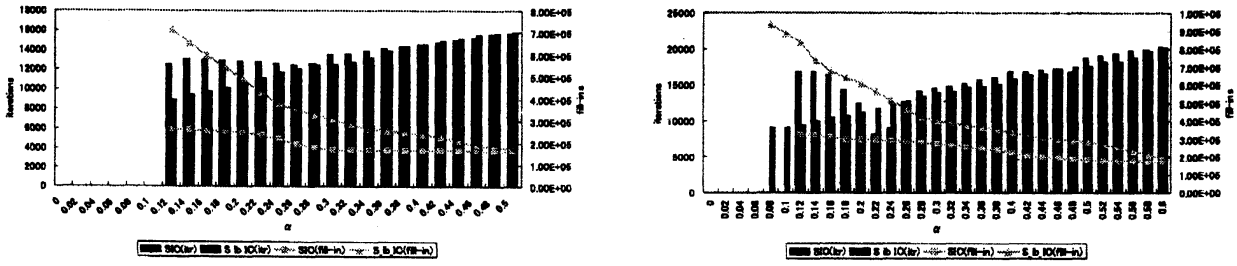
- SIC(*tol*)-CG 法の反復回数は，$\alpha = 0.14$ から $\alpha = 0.28$ までは単調に減少し，以降の $\alpha = 0.3$ から $\alpha = 0.5$ までは単調に増加した．
- SIC(*tol*)-CG 法のフィルイン数は，α 変化毎に非常に緩やかに減少した．
- S ib_IC(*tol*)-CG 法の反復回数は，α 変化毎に単調に増加した．
- S ib_IC(*tol*)-CG 法のフィルイン数は，α 変化毎に単調に減少した．
- 二つの解法の必要メモリ量は $\alpha = 0.5$ のとき，ほぼ同程度となった．

また，Fig. 5 (b) から次のことが観察できる．

- SIC(*tol*)-CG 法の反復回数は，$\alpha = 0.16$ から $\alpha = 0.22$ までは急激に減少し，$\alpha = 0.26$

で急激に増加し，以降の $\alpha = 0.28$ から $\alpha = 0.6$ まではほぼ単調に増加した．

- SIC(*tol*)-CG 法のフィルイン数は，α 変化毎に非常に緩やかに減少した．
- S ib_IC(*tol*)-CG 法の反復回数は，α 変化毎に単調に増加した．
- S ib_IC(*tol*)-CG 法のフィルイン数は，α 変化毎に単調に減少した．
- 二つの解法の必要メモリ量は $\alpha = 0.6$ のとき，ほぼ同程度となった．



(a) matrix S3DKQ4M2　　(b) matrix S3DKT3M2

Fig. 5. Iterations and fill-ins versus shifted value of shifted_IC(*tol*)-CG and shifted ib_IC(*tol*)-CG methods for test matrices(*tol* = 0.15).

以上から，S_ib_IC(*tol*)-CG 法の反復回数が，*tol*，α の変化に対して安定に変化することは必要メモリ量の変化に対して安定に変化することと言い換えることもできる．これは，式 (3.6) で示したように，S_ib_IC(*tol*) 分解が誤差行列 X_α のノルムの大きさを見積もり，行列 U_α の要素をドロッピングすることにより，前処理後の係数行列 $U_\alpha^{-T} A U_\alpha^{-1}$ の反復法の収束性に関する性質，すなわち条件数が *tol*，α の変化に対して徐々に大きくなったためであると考えられる．また，α 変化毎に S_ib_IC(*tol*)-CG 法のフィルイン数が単調に減少したのは，そのドロッピングの式 (4.9) が，SIC(*tol*)-CG 法のドロッピングの式 (4.8) よりも α の変化の影響を受け易いためであると考えられる．これは，α が大きくなる毎に式 (4.9) の $|\xi|$ が小さくなり，そしてその分母が式 (4.8) の分母よりも大きくなることから予測できる．したがって，S_ib_IC(*tol*)-CG 法において，SIC(*tol*)-CG 法のように必要メモリの大幅な削減を目指すには，α を積極的に大きく設定すればよいことが分かる．

6. まとめ

IC(*tol*) 分解を適用した CG 法の収束性に大きなばらつきが生じることに着目し，その問題点を克服するために非対称行列用の手法として提案された Inverse-based ドロッピング手法の考え方を IC(*tol*) 分解に適用することを試みた．そして，Inverse-based ドロッピング手法に基づく IC(*tol*) 分解を ib_IC(*tol*) 分解と名付け，より実用的なシフト ib_IC(*tol*) 分解と併せて，それらの評価を様々な角度から行った．その結果，当手法の収束性は，従来型のシフト IC(*tol*) 分解つき CG 法の収束性と比較して，閾値 *tol* とシフト量 α，そしてフィルイン数の変化に対して安定に変化することが分かった．

参考文献

[1] Barret, R., M., Chan, T., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., van der Vorst, H., Templates for the solution of linear systems: Building blocks for iterative methods, SIAM, (1994). (邦訳) 春日里美, 長谷川秀彦, 藤野清次, 反復法 Templates, 朝倉書店, 東京, (1996).

[2] Benzi, M., Kouhia, R., Tuma, M., Stabilized and block approximate inverse preconditioners for problems in solid and structural mechanics, Comput. Methods Appl. Mech. Engrg. 190(2001), 6533-6554.

[3] Bollhoefer, M., A robust ILU with pivoting based on monitoring the growth of the inverse factors, Linear Algebra and its Applications, 338(2001), 201-218.

[4] Bollhoefer, M., Saad, Y., A factored approximate inverse preconditioner with pivoting, SIAM J. on Matrix Analysis and Applications, 23(2001), 692-705.

[5] Bollhoefer, M., A robust and efficient ILU that incorporates the growth of the inverse triangular factors, SIAM J. Sci. Comput., 25(2003), 86-103.

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

[7] Li, N., Saad, Y., Chow, E., Crout version of ILU for general sparse matrices, SIAM J. Sci. Comput., 25(2003), 716-728.

[8] Li, N., Saad, Y., Crout versions of the ILU factorization with pivoting for sparse symmetric matrices, ETNA, 20(2005), 75-85.

[9] Mayer, J., ILUCP: a Crout ILU preconditioner with pivoting, Numerical Linear Algebra with Applications, 12(2005), 941-955.

[10] 森正武, 数値解析第 2 版, 共立出版, 東京, (2002).

[11] 名取亮, すうがくぶっくす 線形計算, 朝倉書店, 東京, (1993).

[12] Saad, Y., ILUT: a dual threshold incomplete LU factorization, Numerical Linear Algebra with Applications, 1(1994), 387-402.

[13] Saad, Y., Iterative Methods for Sparse Linear Systems, SIAM Philadelphia, (2003).

[14] Tuff A. D., Jennings, A., An iterative method for large systems of linear structural equations, *Int. J. Numer. Meth. Engrg.*, 7(1973), 175-183.