対角要素を事前に補償する非対称行列用前処理の提案

藤原 牧[†] 藤野 清次[††]

本論文は，最近提案された非対称行列用の Crout 版の ILU 前処理の改良について報告するものである．すなわち，元の Crout 版の ILU 前処理が持つ収束の不安定性を改善するために，棄却したフィルインに対応する対角要素を事前に補償する新しい前処理を提案する．また，数値実験において，Florida 疎行列データベースから選出した 10 題の行列に対して，提案した改良版 ILUC 前処理を適用し，元の ILUC 前処理では収束しなかったあるいは収束が不安定だった例題に対して顕著な改善効果を示すことを明らかにした．

A Proposal of Preconditioning with Compensation in Advance of Diagonal Entries for Nonsymmetric Matrix

Maki Fujiwara[†] and Seiji Fujino[††]

This paper presents improvement of the Crout version of Incomplete LU decomposition. We embody a particular strategy for remedying convergence instability of ILUC decomposition by means of compensation in advance of diagonal entry due to dropped fillin. Numerical experiments for ten matrices from Florida sparse matrix collection show that ILUC decomposition with compensation in advance of diagonal entry work well in view of effectiveness, convergence safety of iterative method.

1. はじめに

大型で疎な非対称行列を係数行列 A として持つ連立一次方程式は前処理つきの双共役勾配法（以下，BiCG 法[4] と略す）系統の反復法によって解かれることが多い．今まで様々な反復法が提案され，より効率的で収束が安定な反復法が研究されてきた[6),18)~20)]．その中でも，一般化積型 BiCG 法（以下，GPBiCG 法[21] と呼ぶ）は，多くの実際問題の求解に適用され実績をあげてきた．また，著者らも BiCGSafe 法を提案し，収束性の良さと収束安全性を明らかにしてきた[6)]．

一方，前処理（preconditioning）は反復法の収束性を大幅に向上させる有力な技法である．従来多くの前処理が提案されてきたが，本研究では最近提案された非対称行列用の Crout 版の Incomplete LU 分解（以下，ILUC 分解と呼ぶ）に注目した[14),18)]．この前処理は分解過程において分解の対象とする行と列の大きさがつねに同一であり，そのため非対称行列の上三角行列 U に存在する非零要素の個数と下三角行列 L に存在する非零要素の個数との間に大きな差異があるときでも安定な分解ができることが知られている[14),15)]．また，最近ではピボットつきの ILUC 分解についても研究がなされてきた[17)]．

一般に，対称正定値行列の場合には，元の係数行列 A の対角要素を修正する R（obust）IC 分解がよく知られている[1)]．また，柿原ら[12] により，対角緩和係数の導入による RIC 分解の高速版が誕生し，そのロバスト性の証明もなされた．さらに，前処理行列の近似精度を 2 次精度に高め安定な分解ができる RIC2S（Stabilized）分解が Kaporin によって開発された[13)]．

一方，非対称行列の場合には，よりロバストな前処理行列を生成するために，元の係数行列 A の対角要素を修正する分解法が Chow ら[3] によって研究され，さらにその後いろいろな改良がなされてきた[7)~9)]．

そこで，本研究では，ILUC 分解の収束性をさらに向上させるべく，棄却したフィルインに対応した対角要素を事前に補償する ILUC 前処理を提案し，Florida 疎行列データベースから選出した 10 題の行列に対して改良版 ILUC 前処理を適用し，元の ILUC 前処理では収束しなかったあるいは収束が不安定だった例題に対してその有効性を示すことにする．

† ソニーグローバルソルーションズ株式会社
SONY Global Solutions Inc.
†† 九州大学情報基盤センター
Computing and Communications Center, Kyushu University

本論文の構成は次のとおりである．2 章で前処理の概要について述べる．次に 3 章で，Crout 版 ILU 分解の更新処理の概要について記述する．4 章で対角要素を事前に補償する新しい ILUC 前処理を提案する．5 章では，数値実験を通じて，元の ILUC 分解と新しい改良版 ILUC 分解を比較し，後者の改良版 ILUC 前処理の特徴とその有用性を検証する．6 章でまとめと今後の課題について述べる．

2. 前 処 理

解くべき連立一次方程式を $A\boldsymbol{x} = \boldsymbol{b}$ とする．ここで，行列 $A = (a_{ij})$ は正則で $n \times n$ の実数非対称行列，$\boldsymbol{x}$，$\boldsymbol{b}$ は n 次元の解ベクトルと右辺ベクトルとする．一般に，連立一次方程式を反復法で解くとき，行列 A の固有値分布によって収束の速さが決まるとされる．すなわち，固有値ができるだけ密集している方が収束が速い．そこで，行列 A の適当な近似行列 M の逆行列を方程式の左から掛けて

$$M^{-1}A\boldsymbol{x} = M^{-1}\boldsymbol{b} \tag{1}$$

と変形し，これに反復法を適用して解く．この操作により，係数行列 $M^{-1}A$ は行列 A そのものよりも単位行列に近くなり，収束解が得られるまでの反復回数が元のまま方程式を解くよりも大幅に減少することが多い．このような操作は一般に**前処理**と呼ばれる．

2.1 不完全 LU 前処理

係数行列 A が非対称行列のとき有用な前処理の 1 つに次の不完全 LU 分解前処理がある．この前処理は下三角行列 L と上三角行列 U を用いて，

$$A \approx M = LU \tag{2}$$

と行列 A を次のように不完全分解する．

$$A = LU + R. \tag{3}$$

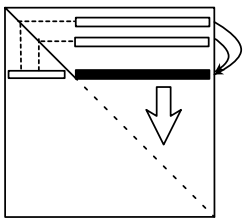
ここで，行列 R は行列 L と行列 U の疎（スパース）性を保持するための誤差行列を表す．この分解において，元の行列 A では値が零の要素であったものが分解過程で零でなくなった要素は**フィルイン**（fillin）と呼ばれる．一般に，数学的に厳密に LU 分解する完全 LU 分解に対して，上のような不完全な分解や前処理は，**ILU 分解**または **ILU 前処理**と呼ばれる．

3. ILUC 前処理

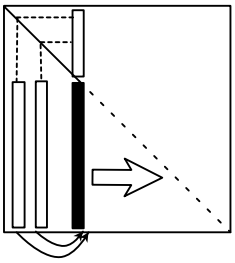
3.1 ILUC 前処理の分解手順とその算法

図 1 に，ILUC 前処理における分解手順と処理が進む方向を矢印で示す．

Crout 版の ILU 前処理（以下，ILUC 前処理と略す）は，下三角行列 L の i 列目と上三角行列 U の i 行目を同時に更新する算法である．そのとき，下三角行列 L の 1 列目から $i-1$ 列目までの要素および上三角行列 U の 1 行目から $i-1$ 行目までの要素が i 列目と i 行目の更新で各々参照される．図 1 (a) は上三角行列 U における分解手順と進行の様子，図 1 (b) は下三角行列 L における分解手順と進行の様子を各々表す．図 1 (a) 中の黒く塗った行は更新の対象とする行列 U の i 行目を指し，図 1 (b) 中の黒く塗った列は同じく更新の対象とする行列 L の i 列目を指している．このような更新処理が，上三角行列 U では上から下に向かって進行し，下三角行列 L では左から右に向かって進行する．



(a) 上三角行列 U の場合



(b) 下三角行列 L の場合

図 1 ILUC 前処理における上三角行列 U と下三角行列 L における分解手順と処理が進行する方向

Fig. 1 Schmatic decomposition procedure of upper triangular matrix U and lower triangular matrix L in ILUC decomposition.

次に ILUC 前処理の算法を以下に示す．算法中で，式 (6) の左辺の $z_{k:n}$ の表記は配列 $z(j)$ において $j = k, \ldots, n$ と変化させることを意味し，同様に，式 (9) 中の左辺の $w_{k+1:n}$ の表記は配列 $w(j)$ において $j = k+1, \ldots, n$ と変化させることを意味する．

ILUC 前処理の算法

For $k = 1, n$ Do

$z_{1:k-1} = 0,$ (4)

$z_{k:n} = a_{k,k:n}$ (5)

For $i = 1, k-1$ and if $l_{k,i} \neq 0$, Do

$z_{k:n} = z_{k:n} - l_{k,i} u_{i,k:n}$ (6)

End Do

$w_{1:k} = 0,$ (7)

$w_{k+1:n} = a_{k+1:n,k}$ (8)

For $i = 1, k-1$ and if $u_{i,k} \neq 0$, Do

$w_{k+1:n} = w_{k+1:n} - u_{i,k} l_{k+1:n,i}$ (9)

End Do

Apply a dropping rule to z (10)

Apply a dropping rule to w (11)

$u_{k,k:n} = z_{k:n}$ (12)

$l_{k+1:n,k} = w_{k+1:n} / u_{k,k}$ (13)

End Do

上の ILUC 算法中の式 (4) と式 (7) は各々作業用配列 z と w の初期化処理であり，このように分解が終了した行と列にも零を代入することで，後の棄却処理でのフィルインと閾値との判定処理が行いやすくなる．フィルインに対して 2 種類の棄却（ドロッピング）処理は式 (10) と式 (11) で以下のように行う．

(1) まず，式 (6) の配列 $z(j)$ と式 (9) 中の配列 $w(j)$ の要素の中で，その絶対値があらかじめ定めた閾値 τ よりも小さいとき，そのフィルインを棄却する．

(2) 次に，行列 L と行列 U ごとに，分解対象の各行各列で絶対値が大きい方から Lfil 個のフィルインを前処理行列の要素として採用する．原論文 14) に従い，以下の算出式で個数の上限値 Lfil を定める．

$$\mathrm{Lfil} = \frac{nnz}{2 \times n} \times m \qquad (14)$$

ここで，n は行列の次元数，nnz は総非零要素数，m は倍率を各々表す．

3.2 ILUC 前処理の棄却処理の実装

行列の非零要素の格納は，下三角行列 L については CCS（Compressed Column Storage）格納法，そして上三角行列 U については CRS（Compressed Row Storage）格納法とする[2)]．このとき，下三角行列 L の更新手順は，上三角行列 U の場合と基本的に同じであるので，以下では行列 U の更新手順だけを記述する．

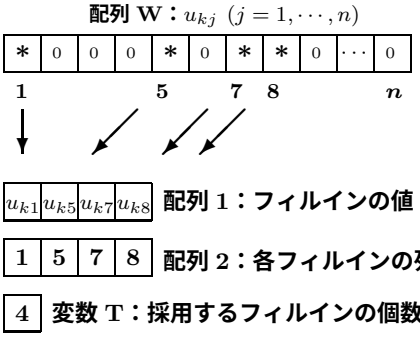


図 2 採用するフィルインだけを，サイズ（$= n$）の固定長の配列からサイズの小さな配列に抽出する様子

Fig. 2 Extraction of adopted fillins into short array from fixed long array of size n.

ここでは，上三角行列 U の第 k 行目の更新のときを考える．**図 2** に，採用するフィルインだけを次元数 n と同じ大きさの作業用配列 W から 2 つの配列（図中の配列 1，2）と 1 つの変数 T に抽出する様子を示す．図に示す配列 1 にはフィルインの値，同配列 2 には各フィルインの列番号，そして同変数 T にはその行で採用されたフィルインの個数の合計が各々収められる．配列 1，2 の大きさは最大で次元数 n と同じだけ用意すればよい．図において配列 u_{kj} 中の「$*$」印は非零要素を「0」は零要素を各々表す．この図は第 k 行目のフィルインとして，4 個 u_{k1}，u_{k5}，u_{k7}，u_{k8} が採用されたことを表す．

また，作業用配列 W は各行各列の処理で再利用される．一方，配列 1 と配列 2 に収められたフィルインの値自体と列番号は，別の大きな配列に各々移される．必要な配列の大きさは行列全体で採用されたフィルインの個数分である．また，変数 T の値も別の次元数 n と同じ大きさの配列に移される．移された後，反復法の繰返し計算中で前処理行列として使用される．

そして，分解過程の各行（または各列）の処理が終わった段階では，配列 W にはその行（または列）で発生したすべてのフィルインが，行列での列番号 (または行番号）と同じ位置に収納されている．次に，行（または列）の暫定的に Lfil 個のフィルインを大きい順に並べて配列 1，2 に収めておき，それらよりも絶対値の大きなフィルインが現れるつど，配列 1，2 の該当する位置に新しいフィルインを挿入する．それにともない，小さなフィルインが配列 1，2 から棄却される．この判定と挿入および棄却処理がその行（または列）の最後のフィルインまで続けられ，最終的にその行（または列）の絶対値の大きなフィルインが最大で Lfil 個だけ配列にセットされ反復法の前処理行列として使われる．

最近の論文 10) でも議論されているように，大型の疎行列の場合，フィルインを格納している配列から非零要素だけを取り出す処理に多くの時間を要するときがある．上で述べた抽出方法の場合，フィルイン u^*_{kj} が発生するごとに，配列 W の同じ位置 j にそのフィルイン u^*_{kj} を収納するだけで済むので手間が非常に少なく，そして行（または列）ごとに最後に 1 度だけ絶対値が大きい方から Lfil 個のフィルインを抽出するという計算コストのかかる処理をすればよいことになる．なお，本論文で採用した方法は，論文 10) では固定長型前処理と呼ばれており，フィルインが現れるごとに手間のかかる抽出処理を行う可変長型前処理に比べて有効なことが多い．

3.3 棄却処理による影響を受ける要素について

非対称行列において，前処理中に棄却された要素に基づいて対角要素を補償する ILUC 前処理について考える．すなわち，係数行列 A を

$$A = LU - R - D \tag{15}$$

と分解する．ここで，行列 L と U は分解後の下三角行列と上三角行列を各々表し，行列 R は不完全分解で生じた誤差行列を表す．対角行列 D で対角要素の補償の処理を行う．また，分解中に発生する上三角行列 U のフィルイン $u^*_{i,j}$ は次のように表せる．

$$u^*_{i,j} = u_{i,j} - \sum_{k=1}^{i-1} l_{i,k} u_{k,j} \tag{16}$$

同様に，下三角行列 L のフィルイン $l^*_{h,s}$ についても次のように表せる．

$$l^*_{h,s} = l_{h,s} - \sum_{k=1}^{s-1} l_{h,k} u_{k,s}. \tag{17}$$

フィルイン $u^*_{i,j}$ に関する式 (16) から分かるように，その算出には $u_{i,j}$ および $u_{1,j}$，$u_{2,j}$，…，$u_{i-1,j}$ が使われる．したがって，フィルイン $u^*_{i,j}$ が棄却されたとき，$u_{k,j}$（$k = i+1,\ i+2,\ \ldots,\ j$）がその影響を被ることになる．同様に，下三角行列 L のフィルイン $l^*_{h,s}$ についても $\ell_{h,k}$（$k = s+1,\ s+2,\ \ldots,\ h$）がその影響を被ることになる．**図 3** に，フィルイン $u^*_{i,j}$ と $l^*_{h,s}$（図中の大きな白い○印）が棄却された場合に，$i+1$ 行より下方および $s+1$ 列より右に位置する要素の分解処理において棄却の影響を受ける部分（図中の小さい白い ∘ 印と大きな黒い●印）を示す．その結果，対角要素 $u_{j,j}$ の値が厳密な LU 分解のときと比べて変動し，異常な値（零に非常に近い値あるいは零自体）になることがありうる．

次に，対角要素 $u_{j,j}$（$j = 1,\ \cdots,\ n$）が異常な値

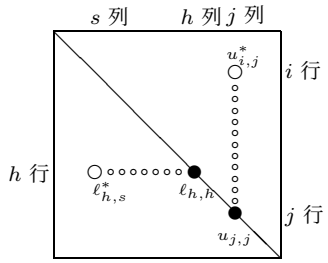


図 3 フィルイン $u^*_{i,j}$ と $l^*_{h,s}$ の棄却の影響を受ける当該の行と列の摸式図

Fig. 3 A diagram of the row and column affected by dropping for small fillins $u^*_{i,j}$ and $l^*_{h,s}$.

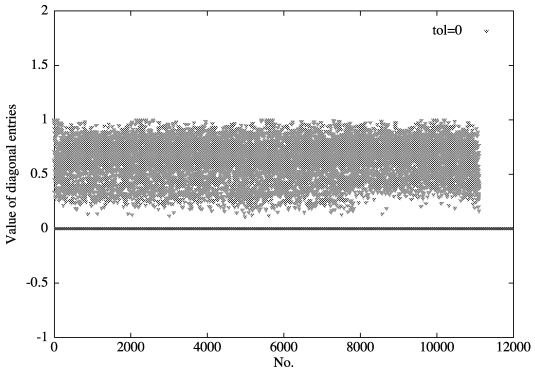


図 4 完全 LU 分解後の対角要素 u_{jj} の値のプロット（行列 K3PLATES のとき）

Fig. 4 Plot of diagonal entries u_{jj} after complete LU decomposition for matrix K3PLATES.

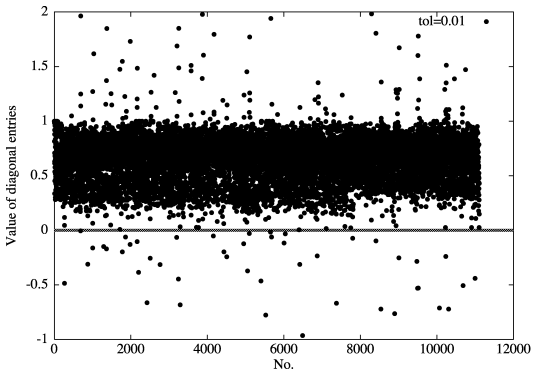


図 5 不完全 LU 分解後の対角要素 u_{jj} の値のプロット（行列 K3PLATES のとき）

Fig. 5 Plot of diagonal entries u_{jj} after incomplete LU decomposition for matrix K3PLATES.

になる 1 つの例を示す．**図 4** に完全 LU 分解後（閾値 = 0 のときに相当）の上三角行列 U の対角要素の値をプロットしたものを灰色の三角点で示す．行列は K3PLATES（後述の表 1 参照）の場合である．同様に，同じ行列に対して，**図 5** に，不完全 LU 分解終了

後（閾値 = 0.01 のとき）の上三角行列 U の対角要素の値をプロットしたものを黒い点で示す．各図の横軸は対角要素の番号，縦軸は対角要素の値を各々表す．また，水平線は値が零の線を表す．これらの図から，不完全 LU 分解のとき，対角要素の値が実際に零あるいは零に非常に近い値になることが分かる．しかし，3.1 節に示した ILUC 前処理の算法中の式 (13) の右辺の分母から分かるように，対角要素の値が零の場合は分解が破綻し，対角要素の値が零に非常に近い場合も近似解の精度を保つ観点から好ましいことではない．

4. 対角要素を事前に補償する前処理

対角要素が零になるのを防ぐために，フィルイン $u^*_{i,j}$ が棄却されたとき，上三角行列 U の対角要素 $u_{j,j}$ を以下のように補償することが考えられる．

$$u_{j,j} = u_{j,j} + |u^*_{i,j}|. \tag{18}$$

同様にして，$l^*_{h,s}$ が棄却されたとき，下三角行列 L の対角要素 $l_{h,h}$ を以下のように補償する．

$$l_{h,h} = l_{h,h} + |l^*_{h,s}|. \tag{19}$$

図 6 に，フィルイン $u^*_{i,j}$ と $l^*_{h,s}$ が棄却されたときの対角要素 $u_{j,j}$ と $l_{h,h}$ との対応関係を表す．

前述の論文 10) においては，分解前に対角要素に 1 より大きな値を掛けることによって分解の破綻を防止し反復法の収束を安定化させた．この考え方は，最初に Manteuffel によって提唱された[16]．一方，本論文では，分解過程で小さなフィルインを棄却するたびに対角要素（行列の対角要素 a_{ii} はあらかじめ 1 に正規化されている）にフィルインの絶対値を加算することにより，分解の破綻を防止し反復法の収束を安定化させている．対角要素 l_{ii} は分解中 1.0 に固定される．一方，対角要素 u_{ii} は対角要素 a_{ii}（分解の最初で 1.0 に正規化処理済み）が代入され，その後の補償処理において対角要素 u_{ii} は次第に大きくなるだけであるので，つねに 1.0 以上の値になり値が零になることはない．次の ca-ILUC 前処理の算法中の式 (20) と式 (21)，さらに補償の式 (26) と式 (33) を参照されたい．

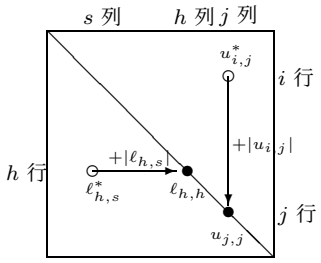


図 6 ca-ILUC 前処理におけるフィルイン $u^*_{i,j}$ と $l^*_{h,s}$ が棄却されたときの対角要素 $u_{j,j}$，$l_{h,h}$ との対応関係

Fig. 6 Relationship between fillins $u^*_{i,j}$, $l^*_{h,s}$ and its corresponding diagonal entries $u_{j,j}$, $l_{h,h}$ in case of ca-ILU preconditioning.

以上のように，棄却した要素に対応する対角要素を該当する行（列）の分解処理が始まる前に事前に補償する前処理を，事前的補償（compensated in advance ILUC，**ca-ILUC**）前処理と呼ぶ．ca-ILUC 前処理の算法を以下に示す．

ca-ILUC 前処理の算法

$$z_k = a_{k,k} \ (k = 1, \ldots, n) \tag{20}$$
$$l_{k,k} = 1.0 \ (k = 1, \ldots, n) \tag{21}$$
For $k = 1, n$ Do
  $z_{1:k-1} = 0$ (22)
  $z_{k+1:n} = a_{k,k+1:n}$ (23)
  For $i = 1, k-1$ and if $l_{k,i} \neq 0$, Do
    $z_{k:n} = z_{k:n} - l_{k,i} u_{i,k:n}$ (24)
  End Do
  For $j = k+1, n$ Do
    If $|z_j| <$ tol then (25)
      $z_k = z_k + |z_j|$, (26)
      $z_j = 0$ (27)
    End If (28)
  End Do
  $w_{1:k} = 0$ (29)
  $w_{k+1:n} = a_{k+1:n,k}$ (30)
  For $i = 1, k-1$ and if $u_{i,k} \neq 0$, Do
    $w_{k+1:n} = w_{k+1:n} - u_{i,k} l_{k+1:n,i}$ (31)
  End Do
  For $j = k+1, n$ Do
    If $|w_j| <$ tol then (32)
      $l_{j,j} = l_{j,j} + |w_j|$, (33)
      $w_j = 0$ (34)
    End If (35)
  End Do
  $u_{k,k:n} = z_{k:n}$ (36)
  $l_{k+1:n,k} = w_{k+1:n} / u_{k,k}$ (37)
End Do

5. 数 値 実 験

5.1 計算機環境と計算条件

数値実験は，CPU：Pentium 4（クロック周波数 3.2 GHz），メインメモリ：2 GB 搭載の HP 社 workstation xw4100 を使用して行った．コンパイラは Intel Fortran Compiler version 7.1，最適化オプションは -O3 を使用した．計算はすべて倍精度演算で行っ

表 1 テスト行列の仕様
Table 1 Specification of test matrices.

行列	次元数 ($=n$)	非零要素 ($=nnz$)	非零要素/行	解析分野
PDE2961	2,961	14,585	4.9	偏微分方程式
K3PLATES	11,107	378,927	34.1	構造解析
EX19	12,005	259,879	21.6	熱流体解析
OLAFU	16,146	1,015,156	62.9	構造工学
LI	22,695	1,350,309	59.5	磁性流体
SME3DA	12,504	874,887	70.0	3D 構造力学
EPB1	14,734	95,053	6.5	熱交換器
NMOS3	18,588	386,594	20.8	集積回路問題
VENKAT25	62,424	1,717,792	27.5	2D オイラー問題
Poisson3db	85,623	2,374,949	27.7	3D ポアソン問題

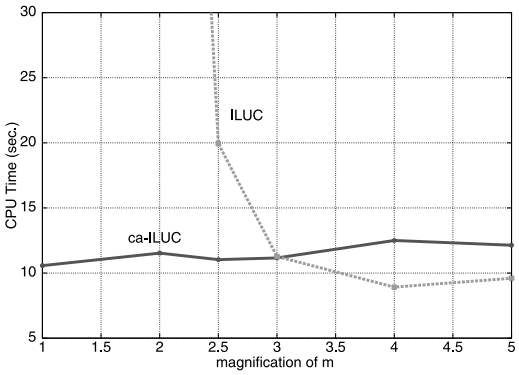


図 7 倍率 m を変化させたときの ILUC 前処理と ca-ILUC 前処理の CPU 時間比較（反復解法：BiCGSafe 法，行列 SME3DA，閾値 $= 0.005$ のとき）
Fig. 7 Convergence property of BiCGSafe method with ILUC and ca-ILUC preconditioning in view of dependence on magnification m of parameter Lfil for matrix SME3DA in case of fixed tolerance of 0.005.

た．反復法の収束判定は残差ベクトル $\boldsymbol{r}_k$ の 2 ノルムの比：$\|\boldsymbol{r}_k\|_2/\|\boldsymbol{r}_0\|_2$ が 10^{-7} 以下となったときとした．初期近似解 $\boldsymbol{x}_0$ はすべて零とした．最大反復回数は 10,000 回とした．また，行列はあらかじめ対角要素をすべて 1 にする正規化処理をした．また方程式の右辺項は物理的に課せられた条件から得られたものを使用した．初期シャドウ残差ベクトル $\boldsymbol{r}_0^*$ には一様乱数を用いた．反復法は BiCGStab 法[19]，GPBiCG 法[21] と BiCGSafe 法[6] を採用した．**表 1** に用いたテスト行列の仕様を表す[11]．

5.2 ILUC 前処理と ca-ILUC 前処理の評価

まず，行列 U と行列 L の各々で絶対値が大きい方から Lfil 個のフィルインの個数の決め方について検討した[5),14)]．**図 7** に，行列 SME3DA の場合，ILUC 前処理および ca-ILUC 前処理の過程で行列 U と行列 L の各々でフィルインの個数の最大値：Lfil $= \frac{nnz}{2\times n}\times m$ の式において，倍率 m を 1.0 から 5.0 まで変化させたときの BiCGSafe 法の収束までの計算時間を示す．閾値 tol は 0.005 に固定した．この図から，ILUC 前処理では反復法の収束が倍率 m に大きく依存し，倍率 m が小さい値（2.5 以下）のとき収束が悪く最大反復回数まで反復を繰り返しても収束しなかったことが分かる．一方，ca-ILUC 前処理は倍率 m の変化に対して非常に安定で，計算時間についても倍率 $m = 1.0$ のとき 10.58 秒，倍率 $m = 5.0$ のとき 12.33 秒であった．これらの結果から，以下の実験では，収束の安全性を優先し，各行各列での絶対値が大きい方から Lfil 個のフィルインを決める算出式中の倍率 m は 5.0 に固定した．

次に，**表 2** と**表 3** に，ILUC 前処理と ca-ILUC 前処理を施した反復法の数値実験結果を示す．前処理の閾値は 0.1，0.05，0.01，0.005 の 4 通りについて調べ，最も早く収束した閾値について数値実験結果を示す．表 2 は，元の ILUC 前処理つき反復法では 4 通りの閾値で収束しなかった 5 つの行列の結果を表す．一方，表 3 は 2 つの前処理（元の ILUC 前処理および ca_ILUC 前処理）でどちらも収束した残りの 5 つの行列に対する結果を表す．表中の「ILUC」の欄の「元」は元の ILUC 前処理を表し，「ca-」は ca-ILUC 前処理を示す．「tol」はフィルインの棄却用の閾値，「fillin」はフィルインの個数を表し，その単位は 10^4 である．「max」は反復計算が最大反復回数までで収束しなかったことを表す．「Itr.」は反復回数，「pre-t」は前処理行列の作成に要した時間，「CG-t」は CG 法の反復に要した時間，「tot-t」は前処理時間と CG 法の反復時間の合計時間を各々表す．時間の単位はすべて秒である．表中の合計時間の欄で太字の数字は各行列で合計時間が最も少なかったものを表す．

表 2 から，以下のことが分かる．

- 3 つの行列 PDE2961，EX19，OLAFU では，BiCGStab 法は ILUC 前処理または ca-ILUC 前処理を施しても，調べたすべての閾値に対して収束しなかった．
- 一方，GPBiCG 法と BiCGSafe 法は，行列 PDE2961 では 2 つの前処理でともに収束し，行列 EX19 と OLAFU では，ca-ILUC 前処理の場合だけが収束した．
- 以上のことから，BiCGStab 法は GPBiCG 法や BiCGSafe 法と比べたとき，収束性が不安定な解法であるといえる．
- また，4 つの行列 K3PLATES，EX19，OLAFU，LI では，ILUC 前処理つき反復法はすべての閾値に対して収束しなかった．

表 2 行列 PDE2961, K3PLATES, EX19, OLAFU, LI に対する BiCGStab, GPBiCG, BiCGSafe 法の収束性
Table 2 Convergence of BiCGStab, GPBiCG and BiCGSafe methods for matrices PDE2961, K3PLATES, EX19, OLAFU and LI.

method	ILUC	tol	fillin	Itr.	pre-t	CG-t	tot-t
BiCGStab	元	—	—	max	—	—	—
	ca-	—	—	max	—	—	—
GPBiCG	元	0.1	2.4	293	0.01	0.21	**0.22**
	ca-	0.01	4.0	248	0.02	0.20	**0.22**
BiCGSafe	元	0.1	2.4	337	0.02	0.22	0.24
	ca-	0.01	4.0	248	0.02	0.20	**0.22**

(a) matrix：PDE2961

method	ILUC	tol	fillin	Itr.	pre-t	CG-t	tot-t
BiCGStab	元	—	—	max	—	—	—
	ca-	0.01	37.7	67	0.27	0.46	0.73
GPBiCG	元	—	—	max	—	—	—
	ca-	0.005	44.4	50	0.23	0.41	0.71
BiCGSafe	元	—	—	max	—	—	—
	ca-	0.005	44.4	49	0.31	0.38	**0.69**

(b) matrix：K3PLATES

method	ILUC	tol	fillin	Itr.	pre-t	CG-t	tot-t
BiCGStab	元	—	—	max	—	—	—
	ca-	—	—	max	—	—	—
GPBiCG	元	—	—	max	—	—	—
	ca-	0.005	35.4	1800	0.29	12.13	12.42
BiCGSafe	元	—	—	max	—	—	—
	ca-	0.005	35.4	1237	0.29	7.86	**8.15**

(c) matrix：EX19

method	ILUC	tol	fillin	Itr.	pre-t	CG-t	tot-t
BiCGStab	元	—	—	max	—	—	—
	ca-	—	—	max	—	—	—
GPBiCG	元	—	—	max	—	—	—
	ca-	0.005	83.1	3908	0.95	64.80	65.75
BiCGSafe	元	—	—	max	—	—	—
	ca-	0.005	83.1	2435	0.94	38.96	**39.90**

(d) matrix：OLAFU

method	ILUC	tol	fillin	Itr.	pre-t	CG-t	tot-t
BiCGStab	元	—	—	max	—	—	—
	ca-	0.05	73.5	92	1.00	1.67	2.67
GPBiCG	元	—	—	max	—	—	—
	ca-	0.05	73.5	72	1.02	1.47	2.49
BiCGSafe	元	—	—	max	—	—	—
	ca-	0.05	73.5	74	1.03	1.42	**2.45**

(e) matrix：LI

表 3 行列 SME3DA, EPB1, NMOS3, VENKAT25, Poisson3db に対する BiCGStab, GPBiCG, BiCGSafe 法の収束性
Table 3 Convergence of BiCGStab, GPBiCG and BiCGSafe methods for matrices SME3DA, EPB1, NMOS3, VENKAT25 and Poisson3db.

method	ILUC	tol	fillin	Itr.	pre-t	CG-t	tot-t
BiCGStab	元	0.005	163.8	281	4.19	7.18	11.37
	ca-	0.005	144.5	543	3.71	12.91	16.62
GPBiCG	元	0.005	163.8	220	4.14	5.69	9.83
	ca-	0.01	94.9	597	1.79	11.58	13.37
BiCGSafe	元	0.005	163.8	209	4.15	5.37	**9.52**
	ca-	0.01	94.9	551	1.78	10.55	12.33

(a) matrix：SME3DA

method	ILUC	tol	fillin	Itr.	pre-t	CG-t	tot-t
BiCGStab	元	0.005	23.0	18	0.33	0.07	0.40
	ca-	0.005	22.4	20	0.29	0.07	**0.36**
GPBiCG	元	0.005	23.0	18	0.34	0.09	0.43
	ca-	0.005	22.4	20	0.28	0.10	0.38
BiCGSafe	元	0.005	23.0	18	0.34	0.08	0.42
	ca-	0.005	22.4	21	0.28	0.10	0.38

(b) matrix：EPB1

method	ILUC	tol	fillin	Itr.	pre-t	CG-t	tot-t
BiCGStab	元	0.005	46.9	3885	0.61	37.17	37.78
	ca-	0.005	46.5	3994	0.61	38.51	39.12
GPBiCG	元	0.005	46.9	2706	0.63	29.20	29.83
	ca-	0.005	46.5	3696	0.63	40.18	40.81
BiCGSafe	元	0.005	46.9	2668	0.63	27.09	**27.72**
	ca-	0.005	46.5	3379	0.62	34.65	35.27

(c) matrix：NMOS3

method	ILUC	tol	fillin	Itr.	pre-t	CG-t	tot-t
BiCGStab	元	0.01	471.0	72	12.84	4.43	17.27
	ca-	0.01	405.9	77	8.66	4.22	12.88
GPBiCG	元	0.01	471.0	66	13.01	4.43	17.44
	ca-	0.05	169.2	201	5.20	7.91	13.11
BiCGSafe	元	0.01	471.0	66	12.92	4.23	17.15
	ca-	0.05	169.2	187	5.131	6.71	**11.84**

(d) matrix：VENKAT25

method	ILUC	tol	fillin	Itr.	pre-t	CG-t	tot-t
BiCGStab	元	0.05	135.5	78	10.63	9.65	20.28
	ca-	0.05	127.5	91	10.61	11.11	21.72
GPBiCG	元	0.05	135.5	79	10.57	9.91	20.48
	ca-	0.05	127.5	77	10.66	9.57	20.23
BiCGSafe	元	0.05	135.5	75	10.65	9.26	**19.91**
	ca-	0.05	127.5	80	10.65	9.99	20.64

(e) matrix：Poisson3db

- 一方，ca-ILUC 前処理つき GPBiCG 法と BiCGSafe 法は上の 4 つの行列に対してすべての閾値に対して収束した．以上のことから，ca-ILUC 前処理つき反復法は従来の ILUC 前処理つき反復法と比べて，安全な前処理であることが分かる．

さらに，表 3 に示した結果から，

- 行列 EPB1，VENKAT25 では，ca-ILUC 前処理の方が ILUC 前処理よりも早く収束し，行列 Poisson3db では，2 つの前処理の性能は同程度であった．
- 一方，行列 SME3DA，NMOS3 では，ca-ILUC 前処理よりも ILUC 前処理の方が早く収束した．

これは，ca-ILUC 前処理において，対角要素を過剰に修正したために，係数行列 A に対する前処理行列 M の近似度が悪くなったためと考えられる．

5.2.1 閾値を変えた場合の両前処理の収束傾向

表 3 の結果は，4 通りの閾値の中で最も合計時間の少なかった場合の比較であり，ここでは様々な閾値で

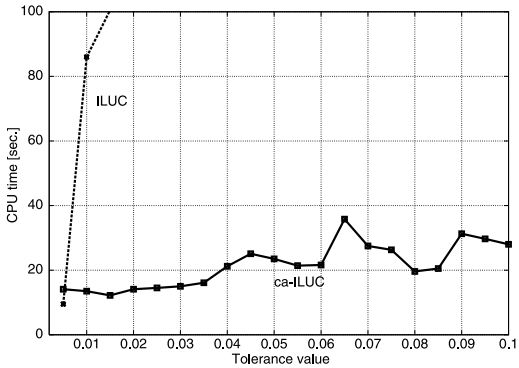


図 8 閾値を変えたときの 2 種類の前処理つきの BiCGSafe 法の収束性（行列 SME3DA のとき）

Fig. 8 Convergence behavior of BiCGSafe method with two kinds of preconditioning for matrix SME3DA when tolerance values are varied.

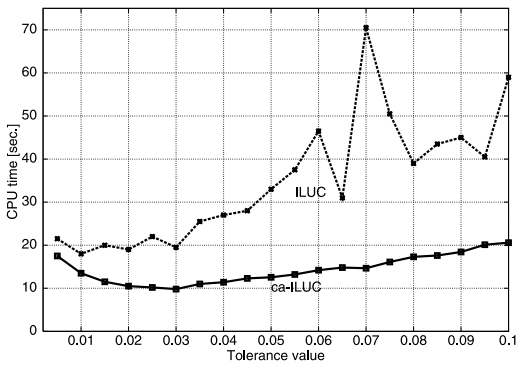


図 9 閾値を変えたときの 2 種類の前処理つきの BiCGSafe 法の収束性（行列 VENKAT25 のとき）

Fig. 9 Convergence behavior of BiCGSafe method with two kinds of preconditioning for matrix VENKAT25 when tolerance values are varied.

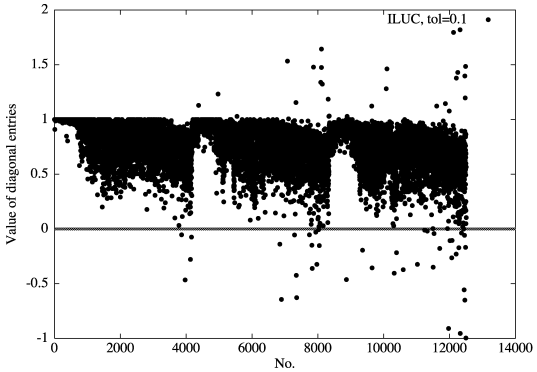


図 10 元の ILUC 前処理における対角要素 u_{ii} の値のプロット（行列 SME3DA，閾値 $= 0.1$ のとき）

Fig. 10 Plot of diagonal entries at tolerance value of 0.1 for matrix SME3DA in case of the original ILUC preconditioning.

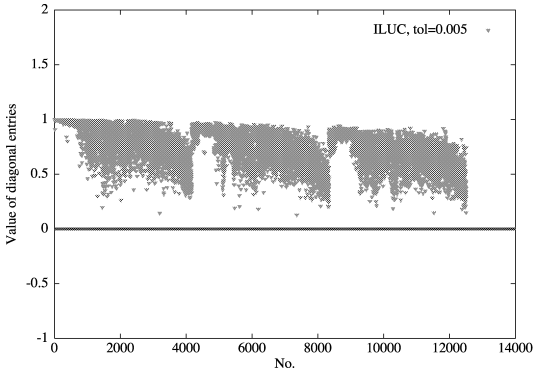


図 11 元の ILUC 前処理における対角要素 u_{ii} の値のプロット（行列 SME3DA，閾値 $= 0.005$ のとき）

Fig. 11 Plot of diagonal entries at tolerance value of 0.005 for matrix SME3DA in case of the original ILUC preconditioning.

2 つの前処理が持つ特徴やその傾向を明らかにする．そこで，行列 SME3DA について，ILUC-BiCGSafe 法と ca-ILUC-BiCGSafe 法の閾値による収束性の変化を調べた．**図 8** にその結果を示す．図の横軸は閾値を，縦軸は合計時間 [単位：秒] を表す．閾値は 0.005 から 0.1 まで 0.005 刻みで変化させ，全部で 20 ケース調べた．また，図中の点線は ILUC-BiCGSafe 法の結果を，実線は ca-ILUC-BiCGSafe 法の結果を各々プロットした．

図 8 の結果から分かるように，ca-ILUC-BiCGSafe 法がすべての閾値で収束したのに対して，ILUC-BiCGSafe 法では閾値が 0.005，0.01 の 2 ケースを除き，閾値がもっと大きな値では収束しなかった．したがって，前処理に対する評価は，収束までの時間が最も少なく効率が良いという点だけでなく，閾値の変化に対してよりロバストで安全な収束性が得られる，という評価も必要である．

さらに，行列 VENKAT25 についても，ILUC-BiCGSafe 法と ca-ILUC-BiCGSafe 法の閾値による収束性の変化を調べた．**図 9** にその結果を示す．調べた閾値などの諸条件は行列 SME3DA のときと同じである．図に示した結果から分かるように，すべての閾値で ca-ILUC-BiCGSafe 法が ILUC-BiCGSafe 法よりも早く収束した．また，閾値を大きくしたとき，ILUC-BiCGSafe 法の収束性は急激に悪くなるのに対して，ca-ILUC-BiCGSafe 法は徐々に収束性が悪くなる程度にとどまっていることが分かる．

5.2.2 収束性と対角要素の値との関係

さらに，図 8 で示した行列 SME3DA の場合，ILUC-BiCGSafe 法は閾値が 0.005 のとき収束したが，閾値が 0.1 のときは収束しなかった．そこで，**図 10** と **図 11** に ILUC 前処理における対角要素 u_{ii} の値をす

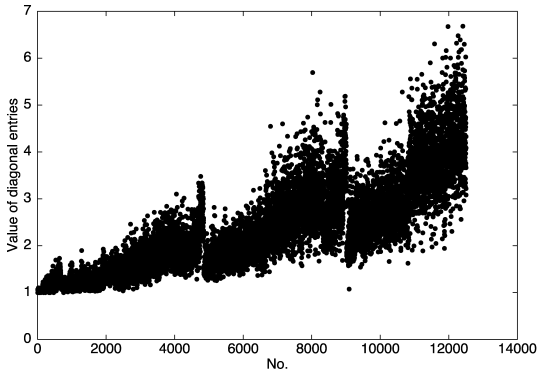


図 12 ca_ILUC 前処理における対角要素 u_{ii} の値のプロット（行列 SME3DA，閾値 = 0.1 のとき）

Fig. 12 Plot of diagonal entries at tolerance value of 0.1 for matrix SME3DA in case of the ca_ILUC preconditioning.

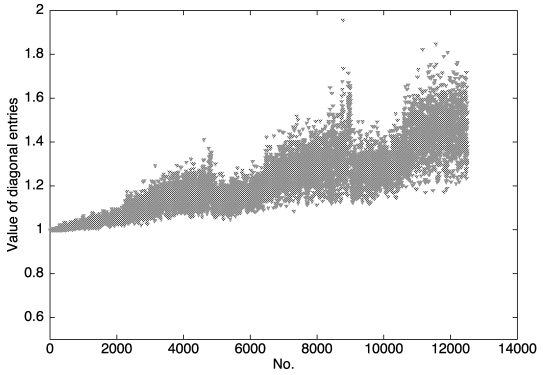


図 13 ca_ILUC 前処理における対角要素 u_{ii} の値のプロット（行列 SME3DA，閾値 = 0.005 のとき）

Fig. 13 Plot of diagonal entries at tolerance value of 0.005 for matrix SME3DA in case of the ca_ILUC preconditioning.

べてプロットした．図 10 が閾値が 0.1 の場合，図 11 が閾値が 0.005 の場合である．図の横軸は対角要素の番号，縦軸は対角要素 u_{ii} の値を各々表す．

閾値が 0.1 のとき（図 10 参照）は，閾値が 0.005 のとき（図 11 参照）と比べて，角要素の値が全般に零に近いものが多数あることが分かる．このことが，閾値が 0.005 のとき収束し，閾値が 0.1 のとき収束しなかった要因の 1 つと考えられる．

また，**図 12** と **図 13** に ca_ILUC 前処理における対角要素 u_{ii} の値をすべてプロットした．図 12 が閾値が 0.1 の場合，図 13 が閾値が 0.005 の場合である．図の横軸は対角要素の番号，縦軸は対角要素 u_{ii} の値を各々表す．元の ILUC 前処理の場合と異なり，閾値が 0.1 の場合の対角要素の最大値が 6 を超えているのに対して，閾値が 0.005 の場合の対角要素の最大値は 2 以下に抑えられているという違いはあるが，いずれの場合も対角要素 u_{ii} の値が全部 1 以上である．このように，ca_ILUC 前処理では反復法の収束性が安定化する．

6. ま と め

本論文では，ILUC 分解が持つ収束の安定性をさらに向上させるために，棄却したフィルインに応じて対角要素を該当する行（列）の分解処理が始まる前に事前に補償する新しい ca-ILUC 前処理を提案し，その性能を検証した．その結果，ca-ILUC 前処理は，従来の ILUC 前処理に比べて，反復法の収束の安全性，安定性を実現する優れた前処理であることが分かった．今後の課題はより多くの行列に対して ca-ILUC 前処理を応用し，その適用範囲を拡大することにある．

参 考 文 献

1) Ajiz, M.A. and Jennings, A.: A robust incomplete Choleski-conjugate gradient algorithm, *Int. J. Numer. Methods Engrg.*, Vol.20, pp.949–966 (1984).
2) Barrett, R., et al.: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, *SIAM* (1994).
3) Chow, E. and Saad, Y.: Experimental study of ILU preconditioners for general sparse matrices, *J. Comput. Appl. Math.*, Vol.86, pp.387–414 (1997).
4) Fletcher, R.: Conjugate Gradient preconditioning for indefinite systems, Lecture Notes in Mathematics, No.506, pp.73–89 (1976).
5) 藤野清次，藤原 牧：非対称行列用の対角要素補償型前処理の性能評価，日本応用数理学会平成 18 年度研究部会連合発表会，早稲田大学 (2006.3).
6) 藤野清次，藤原 牧，吉田正浩：準残差の最小化に基づく積型 BiCG 法，日本計算工学会論文集，pp.145–152 (2006). インターネット論文 http://save.k.u-tokyo.ac.jp/jsces/trans/trans2005/No.20050028.pdf
7) 藤原 牧，吉田正浩，藤野清次：棄却した要素に基づいて対角項を修正する Crout 版 ILU-BiCGSafe 法の収束性評価，第 9 回環瀬戸内シンポジウム予稿集，pp.58–63, 金沢大学 (2005).
8) 藤原 牧，吉田正浩，藤野清次：対角要素を修正する Crout 版 ILU 前処理の収束性評価，九州大学大学院システム情報科学紀要，Vol.11, No.1, pp.45–50 (2006).
9) 藤原 牧：非対称行列に対する不完全分解前処理と反復法の評価，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).
10) 藤原 牧，吉田正浩，藤野清次：収束の三重の安全鍵を与える Crout 版 ILU 分解つき BiCGSafe

法，情報処理学会論文誌：コンピューティングシステム，Vol.47, No.SIG7 (ACS14), pp.52–60 (2006).
11) University of Florida Sparse Matrix web page. http://www.cise.ufl.edu/research/sparse/matrices
12) 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：コンピューティングシステム，Vol.46, No.SIG4 (ACS9), pp.45–55 (2005).
13) Kaporin, I.E.: High quality preconditioning of a general symmetric positive definite matrix based on its $U^TU+U^TR+R^TU$-decomposition, *Numer. Lin. Alg. Appl.*, Vol.5, pp.483–509 (1998).
14) Li, N., Saad, Y. and Chow, E.: Crout version of ILU for general sparse matrices, *SIAM J. Sci. Comput.*, Vol.25, pp.716–728 (2003).
15) 巻幡憲俊，宇都宮智昭，渡邊英一：波浪解析問題のための境界要素法への ILUC の適用，応用力学論文集，Vol.7, pp.279–284, 土木学会 (2004).
16) Manteuffel, T.A.: An incomplete factorization technique for positive definite linear systems, *Math. Comp.*, Vol.31, pp.473–497 (1980).
17) Mayer, J.: ILUCP: A Crout ILU preconditioner with pivoting, *Numerical Linear Algebra with Applications*, Vol.12, pp.941–955 (2006).
18) Saad, Y.: *Iterative Methods for Sparse Linear Systems*, SIAM Philadelphia (2003).
19) van der Vorst, H.A.: Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, *SIAM J. Sci. Stat. Comput.*, Vol.13, pp.631–644 (1992).
20) van der Vorst, H.A.: *Iterative Krylov preconditionings for large linear systems*, Cambridge University Press, Cambridge (2003).
21) Zhang, S.-L.: GPBi-CG: Generalized product-type preconditionings based on Bi-CG for solving nonsymmetric linear systems, *SIAM J. Sci. Comput.*, Vol.18, pp.537–551 (1997).

(平成 18 年 4 月 25 日受付)
(平成 18 年 7 月 29 日採録)

藤原　牧

2004 年 3 月九州大学工学部電気情報工学科卒業．2006 年 3 月九州大学大学院システム情報科学府修士課程修了．2006 年 4 月（株）ソニーグローバルソルーションズ入社．非対称行列用の前処理と反復法に興味を持つ．

藤野　清次（正会員）

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．日本応用数理学会，計算工学会各会員．