A–直交過程に基づく RIF 前処理の効率化について

池 田 優 介[†]　藤 野 清 次[††]
柿 原 正 伸[†]　井 上 明 彦[†]

近年，A–直交化過程に基づいて近似逆行列分解を行う前処理が注目され，その中でも特に安定化近似逆行列分解前処理が CG 法の収束の安定化と効率化を実現する有用な方法として評価されてきた．そして，その分解過程において “副産物” として得られる不完全分解因子を利用する RIF 前処理が，従来の不完全コレスキー分解因子では十分持ち得なかった分解のロバスト性を兼ね備えていることが分かってきた．そこで，本研究では，行列のスパース性の保持のための dropping 処理に着目し，このロバスト性をより発揮させるために，dropping 処理を分解過程で二重に行うことによって，RIF 前処理の有効性をよりいっそう高める方法を提案し，数値実験によりその有効性の検証を行う．

An Enhancement of Efficiency for Robust Incomplete Factorization Preconditioning Based on A-orthogonalization Process

Yusuke Ikeda,[†] Seiji Fujino,[††] Masanobu Kakihara[†]
and Akihiko Inoue[†]

A sparse stabilized approximate inverse preconditioning is used as an explicit and effective proconditioning technique for the conjugate gradient computation. The lower triangular matrix L of matrix A can be obtained as a by-product of the A–orthogonalization process, at no extra cost. In this paper, we adopt double dropping strategy in the process of the A–orthogonalization for enhancement of effectiveness of RIF preconditioning for the CG method. Our experiments indicate that this preconditioning strategy can insure quick convergence of the PCG iteration.

1. はじめに

大型の疎行列を係数行列 A に持つ連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ は，前処理つき反復法によって解かれることが多い．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[6),16)]．現在，前処理行列の構成方法は，大きく 2 つの考え方に分けることができる．1 つは係数行列 A を近似する不完全分解，もう 1 つは逆行列 A^{-1} を近似する近似逆行列分解である．前者では不完全コレスキー分解がよく知られている[13)]．後者では，A–直交化過程に基づいて逆行列 A^{-1} を近似する近似逆行列分解（Approximate INVerse，以下 **AINV** と略す）が最近知られてきた[1)]．AINV では，前処理行列のスパース性を保つために，dropping 処理と呼ばれる操作が行われる．dropping 処理とは，あらかじめある閾値を設定し，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大きいときのみそれを残し，小さいときは零と見なして以降の分解を行うという処理を指す．dropping 処理は，計算量やメモリ量の削減という観点から AINV では不可欠な操作である[1)]．しかし，この操作によって，前処理行列の正定値性が失われ，CG 法の収束性に悪影響を及ぼすことがある[2)]．そこで，分解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安定化近似逆行列（Stabilized-AINV，以下 **SAINV** と略す）前処理である[2),10)]．SAINV では，分解に必要な計算量を増加させることなく，分解の安定性を実現している．

近年，SAINV の分解過程において得られた逆行列 A^{-1} の近似分解因子が，行列 A の通常の不完全分解因子 L と数学的に等しい，という事実を利用した新しい前処理が提案された[3)]．この前処理においても SAINV の優れた特徴を受け継いで安定した不完全分解ができる．そのため，この前処理は **RIF**（Robust Incomplete Factorization）と呼ばれる．この前処理では，余計な計算を付加せずに不完全分解因子 L が得られ，元の近似分解因子を用いた前処理に比べて，よりいっそう有効な前処理になることが知られている[3)]．

† 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

我々は前報（文献 7)）において，SAINV 前処理の性能向上のために，近似分解中に dropping 処理を二重に行う double dropping 技法を提案し，その有効性を数値実験で実証した[7),8)]．そこで，本研究では，double dropping 技法を RIF 前処理に適用し，その有用性を検証する．さらに，Matrix Market などの行列データベースに収納された問題だけでなく，エンジニアリングの分野において用いられた，実際のコンクリート橋の梁（BEAM）やケーブル定着部における応力解析から生じた行列に対しても double dropping つき RIF 前処理を適用し，より具体的な問題に対する有効性の検証を試みる．

本論文の構成は以下のとおりである．まず，2 章で不完全分解および近似逆行列分解による前処理つき CG 法について記述する．3 章では，A–直交化に基づく近似逆行列分解について述べる．続く 4 章で，近似逆行列分解からロバスト不完全分解が導かれる過程を示す．そして，5 章で数値実験結果を示しその考察を行う．最後に，6 章でまとめを述べる．

2. 前処理つき共役勾配法

正定値対称行列を係数行列 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 法で解くことを考える．ここで，A は $n \times n$ の正方行列，$\boldsymbol{x}$，$\boldsymbol{b}$ は次元数 n の解および右辺ベクトルとする．不完全分解を用いた前処理では，係数行列 A を

$$A \simeq LD_{ic}L^t \tag{2}$$

のように分解する．ここで，L は対角要素が 1 の下三角行列，D_{ic} は対角行列，上付き添字 t は転置を表す．また，下付き添字 ic は対角行列が不完全コレスキー分解の因子であることを表す．このとき，不完全分解前処理つき CG 法の算法は以下のように表すことができる．$\boldsymbol{x}_0$ は初期近似解，ε は収束判定値である．

算法 1　不完全分解前処理つき CG 法

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

一方，近似逆行列による前処理は，逆行列 A^{-1} を，

$$A^{-1} \simeq ZD_{ainv}^{-1}Z^t \tag{3}$$

のように近似する．ここで，Z は対角項が 1 の上三角行列，D_{ainv} は対角行列とする．不完全分解のときと同様に，下付き添字 $ainv$ は対角行列が近似逆行列の因子であることを表す．近似逆行列分解を用いた前処理つき CG 法の算法は以下のように表される．

算法 2　近似逆行列分解前処理つき CG 法

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

3. 逆行列分解

3.1 近似逆行列分解

A–直交化過程に基づく逆行列分解（**AINV**）では，以下に示す分解によって Z および D_{ainv} を構成する[1)]．

分解 1　近似逆行列分解

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

上記の近似逆行列分解において，$\boldsymbol{z}_j^{(i)}$ 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_{ainv} 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．このとき，d_j の計算では指標 j の動く範囲が $i \le j \le n$ であることから，反復 i 回目の分解では D_{ainv} の第 i 番目の要素 d_i が確定することが分かる．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 Z 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．上記の分解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．

処理 1　dropping 処理

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

AINV では，このようにして近似逆行列因子 Z および対角行列 D_{ainv} を構成し，前処理行列として前節で示した近似逆行列分解前処理つき CG 法に適用する．

3.2 分解の安定化

AINV では，dropping 処理の影響で対角行列 D_{ainv} の要素が負の値になることがある．そこで，分解の安定化を図るため，安定化近似逆行列（**SAINV**）が提案された[2),10)]．SAINV の分解は次のように表せる．分解中の下線部分が前述の AINV と異なる部分である．

分解 2　安定化近似逆行列分解

$$
\begin{array}{l}
\textbf{for}\ \ j=1,\cdots,n \\
\quad \boldsymbol{z}_j^{(0)} = \boldsymbol{e}_j \\
\textbf{end for} \\
\textbf{for}\ \ i=1,\cdots,n \\
\quad \underline{\boldsymbol{v} = \mathrm{A}\boldsymbol{z}_i^{(i-1)}} \\
\quad \underline{\textbf{for}\ \ j=i,\cdots,n} \\
\qquad \underline{d_j = \boldsymbol{v}^t \boldsymbol{z}_j^{(i-1)}} \\
\quad \textbf{end for} \\
\quad \textbf{for}\ \ j=i+1,\cdots,n \\
\qquad \boldsymbol{z}_j^{(i)} = \boldsymbol{z}_j^{(i-1)} - \dfrac{d_j}{d_i} \boldsymbol{z}_i^{(i-1)} \\
\quad \textbf{end for} \\
\textbf{end for}
\end{array}
$$

ここで，$\boldsymbol{v}$ は中間ベクトルである．A–直交化過程に基づく近似逆行列分解において，対角 d_j の計算式 $\boldsymbol{a}_i^t \boldsymbol{z}_j^{(i-1)}$ は，$(\boldsymbol{z}_i^{(i-1)})^t A \boldsymbol{z}_j^{(i-1)}$ と等しいことが分かっている．そこで，SAINV では，$(\boldsymbol{z}_i^{(i-1)})^t A \boldsymbol{z}_j^{(i-1)}$ を用いて d_j を計算する．このとき，$d_i = (\boldsymbol{z}_i^{(i-1)})^t A \boldsymbol{z}_i^{(i-1)}$ となり，係数行列 A が正定値行列ならば，任意の $\boldsymbol{z}_i^{(i-1)}$ に対して対角項 d_i はつねに正の値となり安定して分解が行われる．実際の計算は，中間ベクトル $\boldsymbol{v}$ を使って，上に示した安定化近似逆行列分解の中の下線部に示すように 2 段階に分けて行われる．

3.3 Double Dropping つき SAINV

我々は，前報（文献 7)）において SAINV の分解過程で二重の dropping を行い，CG 法の収束性を大幅に改善させる改良型の（Improved）SAINV を提案した[7),8)]．以下において，この前処理を **ISAINV** と呼ぶ．この二重の dropping（以下，double dropping と呼ぶ）処理では，小さな値の要素を棄却する従来の dropping 処理のほかに，ベクトル $\boldsymbol{z}_j^{(i-1)}$ を更新するかど うかを判定するための新たな dropping 処理を追加する．そして，追加された dropping 処理のために閾値 tol_dd を導入し，更新の可否判定には $\boldsymbol{z}_i^{(i-1)}$ に掛ける比率 $\frac{d_j}{d_i}$ の絶対値が使用される．したがって，$\boldsymbol{z}_j^{(i-1)}$ の更新を表す式 (4) は式 (5) に置き換わる．ここで，下線部分が double dropping にあたる部分である．

処理 2　double dropping 処理

$$
\begin{array}{l}
\underline{\textbf{if}\ \ |\frac{d_j}{d_i}| > \mathrm{tol\_dd}} \\
\quad \textbf{for}\ \ k=1,\cdots,i \\
\qquad \textbf{if}\ \ |z_{kj}^{(i-1)} - \frac{d_j}{d_i} z_{ki}^{(i-1)}|\ > \mathrm{tol} \\
\qquad\quad z_{kj}^{(i)} = z_{kj}^{(i-1)} - \frac{d_j}{d_i} z_{ki}^{(i-1)} \\
\qquad \textbf{else} \\
\qquad\quad z_{kj}^{(i)} = 0 \\
\qquad \textbf{end if} \\
\quad \textbf{end for} \\
\underline{\textbf{end if}}
\end{array} \qquad (5)
$$

SAINV では，係数行列 A の正定値性を利用して任意の $\boldsymbol{z}_i^{(i-1)}$ に対して対角項 d_i の値を正に保ち，分解の安定性を実現している．したがって，double dropping 処理によって $z_j^{(i)}$ が変化しても，SAINV と同様に安定して分解を行うことができる．

4. 近似逆行列からロバスト不完全分解へ

4.1 完 全 分 解

本節では，係数行列 A とその逆行列 A^{-1} を完全分解することを考える．完全分解では，式 (2) と式 (3) は各々次のように表すことができる．

$$A = \bar{L}\bar{D}_{ic}\bar{L}^t, \quad (6)$$
$$A^{-1} = \bar{Z}\bar{D}_{ainv}^{-1}\bar{Z}^t. \quad (7)$$

ここで，$\bar{L}$，$\bar{D}_{ic}$，$\bar{Z}$，$\bar{D}_{ainv}$ は完全分解因子をそれぞれ表す．ただし，$\bar{L}$ および $\bar{Z}$ の対角項は 1 となるように $\bar{D}_{ic}$ と $\bar{D}_{ainv}$ を定める．特に，式 (6) は完全コレスキー分解に相当し，その逆行列は，

$$A^{-1} = \bar{L}^{-t}\bar{D}_{ic}^{-1}\bar{L}^{-1} \quad (8)$$

となり，式 (7) および式 (8) が同じ構成をしていることに気がつく[9),14)]．したがって，式 (6) で示した完全コレスキー分解においては，分解が一意に定まることから，完全分解因子と逆行列因子の間には次のような関係が成り立っていることが分かる．

$$\bar{Z}^t = \bar{L}^{-1}, \quad (9)$$
$$\bar{D}_{ainv} = \bar{D}_{ic}. \quad (10)$$

これらの等式から，式 (6) と式 (7) は両方とも次のように変形できる[5)]．

$$\bar{L} = A\bar{Z}\bar{D}_{ainv}^{-1} \quad (\text{または } \bar{L}\bar{D}_{ainv} = A\bar{Z}). \quad (11)$$

この式 (11) は，数学的に $A\bar{Z}\bar{D}_{ainv}^{-1}$ と完全分解因子 $\bar{L}$ が同じであること，すなわち**逆行列因子 $\bar{Z}$ から完全分解因子 $\bar{L}$ が計算可能であること**を意味している．この関係を不完全分解に対して適用すると，不完全分解因子 L を近似逆行列因子 Z から，

$$L \leftarrow AZD_{ainv}^{-1} \quad (\text{または } LD_{ainv} \leftarrow AZ). \quad (12)$$

のようにして計算する方法が考えられる．A–直交化過程に基づく不完全分解とは，式 (12) を基に近似逆行列因子 Z から不完全分解因子 L を計算し，前処理行列を構成する方法である[3),4)]．

4.2 A–直交化に基づくロバスト不完全分解（RIF）

ここで，近似逆行列分解中に現れる $\frac{d_j}{d_i}(= \frac{\boldsymbol{a}_i^t \boldsymbol{z}_j^{i-1}}{d_i})$ は，$AZD_{ainv}^{-1}(= L)$ の要素に等しいことに注目する．分解中，近似逆行列因子 Z の代わりに $\frac{d_j}{d_i}$ の値を保存すれば，不完全分解因子 L を構成することができる．この分解は，従来の不完全コレスキー分解とは異なる考えに基づくため，A–直交化過程に基づく不完全分解と呼ばれる．

また，この不完全分解因子 L に対応する対角行列 D_{ic} は，近似逆行列分解における対角行列 D_{ainv} に等しい☆．したがって，AINV 分解ではなく SAINV 分解において不完全分解因子 L を構成することで，$Dainv$ と同様に D_{ic} の要素もつねに正の値となり，安定して分解を行うことができる．そのため，SAINV を通して得られる不完全分解は **RIF**（Robust Incomplete Factorization）と呼ばれる[3)]．具体的には，因子 L の要素の保存は，次の式 (13) 中に示した下線部のように行われる．このとき，以下のように不完全分解因子 L の要素となる $\frac{d_j}{d_i}$ に対しても dropping 処理を適用し，L の疎性を保つ必要がある．

処理 3　　dropping 処理（RIF）

$$\begin{array}{l}
\textbf{if } \ |\frac{d_j}{d_i}| \ > \text{tol} \\
\quad \underline{l_{ji} = \frac{d_j}{d_i}} \\
\textbf{end if} \\
\textbf{for } \ k = 1, \cdots, i \\
\quad \textbf{if } \ |z_{kj}^{(i-1)} - \frac{d_j}{d_i} z_{ki}^{(i-1)}| \ > \text{tol} \\
\quad\quad z_{kj}^{(i)} = z_{kj}^{(i-1)} - \frac{d_j}{d_i} z_{ki}^{(i-1)} \\
\quad \textbf{else} \\
\quad\quad z_{kj}^{(i)} = 0 \\
\quad \textbf{end if} \\
\textbf{end for}
\end{array} \quad (13)$$

ここで，要素 l_{ji} は不完全分解因子 L の j 行 i 列の要素を表す．ただし，この l_{ji} を保存する操作において，指標 j の動く範囲は $i+1 \leq j \leq n$ であることに注意する．したがって，$i = j$ のとき，すなわち不完全分解因子 L の対角項 l_{ii} は上記の過程では得られないことが分かる．ここでは，RIF では，逆行列因子 $\bar{Z}$ とコレスキー因子 $\bar{L}$ の関係式 (9) を利用し，L の対角項を Z の対角項と同様に 1 として不完全分解因子 L を構成する．

4.3 改良型（double dropping つき）RIF

提案する改良型の (Improved) **RIF** を **IRIF** と呼ぶ．**IRIF** では，A–直交化過程に基づく不完全分解の過程において double dropping を適用する．したがって，**IRIF** では，$\boldsymbol{z}_j^{(i-1)}$ の更新の処理は次の式 (14)，そしてその double dropping 処理は下線部分で示される．

処理 4　　double dropping 処理（IRIF）

$$\begin{array}{l}
\textbf{if } \ |\frac{d_j}{d_i}| \ > \text{tol} \\
\quad l_{ji} = \frac{d_j}{d_i} \\
\textbf{end if} \\
\underline{\textbf{if } \ |\frac{d_j}{d_i}| > \text{tol_dd}} \\
\quad \textbf{for } \ k = 1, \cdots, i \\
\quad\quad \textbf{if } \ |z_{kj}^{(i-1)} - \frac{d_j}{d_i} z_{ki}^{(i-1)}| \ > \text{tol} \\
\quad\quad\quad z_{kj}^{(i)} = z_{kj}^{(i-1)} - \frac{d_j}{d_i} z_{ki}^{(i-1)} \\
\quad\quad \textbf{else} \\
\quad\quad\quad z_{kj}^{(i)} = 0 \\
\quad\quad \textbf{end if} \\
\quad \textbf{end for} \\
\underline{\textbf{end if}}
\end{array} \quad (14)$$

☆ RIF の導出については，本文で述べた方法とは異なる別の導出方法が考えられ，その方がより理解がしやすいと思われる．詳細は付録 A.1 参照．

上の式 (14) の中で，$\frac{d_j}{d_i}$ の値は，double dropping において，$\boldsymbol{z}_j^{(i-1)}$ を更新するかどうかの判定をするための値であると同時に，不完全分解因子 L の要素自体の値にもなっていることに注意を要する．閾値 tol_dd は要素の更新を行うかどうかの判定に使用し，閾値 tol は小さな値の棄却の判定に使用する．新しい **IRIF** では，このようにして得られた因子 L および対角行列 D_{ic} を用いて前処理行列を構成し，それらを不完全分解前処理つき CG 法の反復計算で使用する．

5. 数値実験

5.1 計算機環境と計算条件

数値実験は，1 つの問題に対して前処理なしの CG 法と，フィルインを考慮しない不完全コレスキー分解，SAINV，ISAINV，RIF，**IRIF** の 5 つの前処理つき CG 法の合わせて 6 種類の CG 法を適用し，その収束性を比較した．計算機環境は以下のとおりである．

- 計算機：富士通 Prime Power 850
- 使用 PE 数（総 PE 数）：1（16）
- CPU（クロック周波数）：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

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 としそこで計算を中断させた．収束判定値 ε は相対残差 L_2 ノルム $||\boldsymbol{r_m}||_2/||\boldsymbol{r_0}||_2$ の値が 10^{-9} 以下になったときとした．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．メモリ量は，近似分解の反復ごとに前処理に必要なメモリ量が異なるため，最もメモリ量を必要とした反復における配列を合計し算出した．ただし，フィルインを考慮しない不完全コレスキー分解に必要なメモリ量は，対角成分を格納する次元数 n の配列のみである．

また，閾値 tol の範囲は文献 2)，3) において有効であると報告された値（0.1 前後）を参考に次のように定めた．すなわち，SAINV と RIF においては，閾値 tol は 0.01〜0.16 まで 0.01 刻みで 16 通り変化させた．一方，ISAINV と **IRIF** では，閾値 tol_dd は閾値 tol の値の 1.0〜5.0 倍を 0.5 刻みで全部で 9 通り変化させ合計 144 通り（$16 \times 9 = 144$）の閾値の場合の CG 法の収束性を調べた．その試行数 144 は以下の問題 1 と問題 2 で共通である．

表 1 問題 1 のテスト行列の特徴
Table 1 Description of tested matrices for Problem 1.

行列	次元数	非零要素数	問題（対象物）
BCSSTK24	3562	81736	固有値問題
BCSSTK35	30237	740200	自動車座席の構造解析
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

表 2 問題 2 のテスト行列の特徴
Table 2 Description of tested matrices for Problem 2.

	行列	
項目	BEAM	CABLE
次元数	10626	59002
非零要素数	233268	1986094
平均バンド幅	576	1741
総ノード数	1977	20194
総要素数	2832	16084

5.2 テスト行列

以下の 2 つの問題をテスト用の問題として取り上げた．行列は全部で 6 種類である．

- 問題 1：3 つの疎行列データベースから選んだ行列 4 種類．
- 問題 2：実際のコンクリート橋梁の応力解析で使用された行列 2 種類．

5.2.1 問 題 1

問題 1 で数値実験に用いた行列 4 種類の特徴と解いた問題（対象物）を**表 1** に示す．行列 BCSSTK24 は Matrix Market[12] から，行列 BCSSTK35 と NASASRB はフロリダ大学の疎行列データベース[18] から，行列 TUBE1-2 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．また，右辺項は厳密解がすべて 1 となるように定めた．

5.2.2 問 題 2

問題 2 では，より現実的な問題に対する評価をするために，実際にコンクリート橋の設計に使用された応力解析の問題を取り上げ，そこで得られた 2 つの行列を数値実験で使用した．また，右辺項は設定された加重条件を元に離散化によって得られる値を用いた．2 つの行列を各々BEAM および CABLE と呼ぶ．その特徴を**表 2** に示す．

まず，行列 BEAM は橋梁の間にトラック荷重を課したときの応力解析の問題で，解析モデルはシェル要素（プレート要素，ビーム要素，トラス要素）のみで構成される．一般に，シェル要素による離散化行列の解析は難しく，収束までに多くの反復回数が必要になるとされる[15),19)]．一方，行列 CABLE はコンクリー

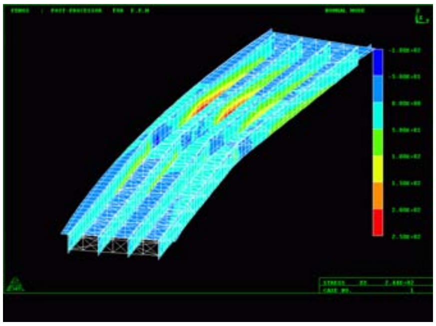

図 1　応力（x-y 成分）分布（行列 BEAM）
Fig. 1　Distribution of stress of x-y component for matrix BEAM.

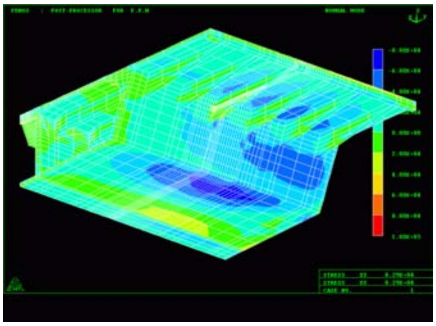

図 2　応力（x-y 成分）分布（行列 CABLE）
Fig. 2　Distribution of stress of x-y component for matrix CABLE.

ト橋におけるケーブル定着部の応力解析で生じた問題で，ソリッド要素だけで離散化が行われた．ソリッド要素による離散化の場合は，シェル要素の場合と比較して，問題は解きやすいことが多く，反復法の収束も早いとされる．しかし，行列の次元数は一般に多くなる傾向がある[17)]．

図 1 に行列 BEAM のときの応力（x-y 成分）分布の様子を，**図 2** に行列 CABLE のときの応力（x-y 成分）分布の様子を各々示す．これらの図は，トラックによる荷重やケーブル定着部の荷重点と近い位置に大きな応力が発生していること，その応力が設計強度の範囲内であることを表している．特に図 2 からは，荷重点となるケーブル定着部から離れた位置に逆方向の応力が見られ，解析結果が妥当であることが分かる．

5.3　実験結果と考察

5.3.1　問　題　1

表 3 に，問題 1 の 4 種類の行列に対して 6 つの前処理をそれぞれ適用した CG 法の実験結果を示す．ただし，行列はすべて対角項を 1 に正規化してあるため，前処理なし CG 法は実質的に対角スケーリング CG 法にあたる．また，フィルインを考慮しない IC 分解では正定値性を保つために，対角項がすべて正の値になるまで対角項の値を修正して分解を行った．SAINV，ISAINV，RIF，IRIF の 4 つの前処理つき CG 法については，調べた閾値の範囲において最も計算時間が少なかったときの実験結果である．

表 3 において，“tol”，“tol_dd” は各閾値の値，“メモリ量” とは前処理で必要なメモリ量（単位：MByte）の最大値，“非零要素比” とは各前処理によって得られた前処理行列 Z または L が持つ非零要素数と元の係数行列が持つ非零要素数との比，“時間比” とは RIF つき CG 法の収束までの計算時間を 1.0 としたときの各前処理つき CG 法の計算時間の比を各々表す．表中，“∞” 印は最大反復回数までで収束しなかったことを指す．また，“前処理時間” とは不完全分解に要した時間を表し，“CG 時間” とは，CG 法の算法と，各反復で実行される前処理行列とベクトルの積の合計時間である．時間の単位はすべて秒である．表の中で太字表記の数字は改善効果が著しかったときの数字を指す．

表 3 の結果から，以下のことが分かる．

- 前処理なしの CG 法や IC 分解つき CG 法は，2 種類の行列で収束せず，収束した場合でも多くの反復回数や計算時間が必要になる．
- A–直交化に基づく 4 つの前処理つき CG 法は，調べた 4 種類の行列で収束し，前処理なし CG 法や IC 分解つき CG 法より計算時間が少ない．
- すべての行列において，提案する **IRIF** は 6 つの反復法の中で計算時間が最も少なく，従来の RIF と比べても計算時間の比が 0.42〜0.57 と半減する．
- メモリ量も，double dropping つき ISAINV と **IRIF** は，それぞれ従来のものに比べて多くの場合で分解に必要なメモリ量が少なくて済む．

5.3.2　問　題　2

表 4 に，問題 2 の行列に対する 6 つの前処理つき CG 法の数値実験結果を表す．表 4 において，各項目の表記や表中に用いた記号の意味は表 3 と同じである．

表 4 の結果から，以下のことが観察される．

- 行列 BEAM では，前処理なしの CG 法が収束せず，IC 分解つき CG 法も収束までに多くの反復回数が必要である．
- 問題 1 のときと同様に，**IRIF** 前処理つき CG 法が 6 つの反復法の中で計算時間が最も少ない．
- 分解に必要なメモリ量も，**IRIF** は従来の RIF に比べて少ない．

表 3　問題 1 の行列に対する 6 つの前処理つき CG 法の数値実験結果
Table 3 Numerical results of several kinds of preconditioned CG methods for Problem 1.

行列	前処理	tol	tol_dd	メモリ量（MB）	非零要素比	反復回数	前処理時間	CG 時間	合計時間	時間比
BCSSTK24	なし	–	–	–	–	∞	–	–	–	–
	IC	–	–	0.03	1.00	∞	0.07	–	–	–
	SAINV	0.10	–	7.00	0.45	1061	0.84	2.29	3.13	1.59
	ISAINV	0.13	0.455	2.26	0.12	1044	0.16	1.64	1.80	0.91
	RIF	0.10	–	6.90	0.22	666	0.85	1.12	1.97	1.00
	IRIF	0.04	0.100	3.40	0.42	289	0.38	0.56	0.94	**0.48**
BCSSTK35	なし	–	–	–	–	∞	–	–	–	–
	IC	–	–	0.23	1.00	15176	1.66	727	729	4.73
	SAINV	0.13	–	27.9	0.43	13848	12.0	426	438	2.84
	ISAINV	0.14	0.420	21.3	0.24	11199	5.46	272	277	1.80
	RIF	0.02	–	53.5	1.07	1991	59.2	94.4	154	1.00
	IRIF	0.01	0.030	34.9	1.25	1098	31.2	57.4	88.5	0.57
NASASRB	なし	–	–	–	–	14605	–	468	468	1.80
	IC	–	–	0.42	1.00	5477	1.93	491	493	1.93
	SAINV	0.12	–	43.8	0.34	7761	16.6	392	408	1.56
	ISAINV	0.15	0.750	36.6	0.07	7923	13.1	285	298	1.15
	RIF	0.08	–	46.8	0.40	4155	18.4	241	259	1.00
	IRIF	0.02	0.100	50.9	0.86	1405	20.5	111	132	**0.51**
TUBE1-2	なし	–	–	–	–	13832	–	146	146	2.24
	IC	–	–	0.16	1.00	5746	0.77	167	168	2.58
	SAINV	0.13	–	35.3	0.32	4134	25.9	68.0	93.8	1.44
	ISAINV	0.05	0.250	15.0	0.51	2050	4.48	42.0	46.5	0.71
	RIF	0.15	–	35.7	0.15	2558	27.0	38.0	65.1	1.00
	IRIF	0.04	0.200	15.5	0.53	1094	5.26	22.7	27.9	**0.42**

表 4　問題 2 の行列に対する 6 つの前処理つき CG 法の数値実験結果
Table 4 Numerical results of several kinds of preconditioned CG methods for Problem 2.

行列	前処理	tol	tol_dd	メモリ量（MB）	非零要素比	反復回数	前処理時間	CG 時間	合計時間	時間比
BEAM	なし	–	–	–	–	∞	–	–	–	–
	IC	–	–	0.08	1.00	7620	0.3	102	102	6.46
	SAINV	0.11	–	11.8	0.47	3166	2.68	28.6	31.2	1.97
	ISAINV	0.08	0.280	6.83	0.23	1468	0.85	10.4	11.2	0.71
	RIF	0.11	–	11.5	0.25	1735	2.70	13.1	15.8	1.00
	IRIF	0.02	0.070	9.33	0.56	**447**	1.67	4.49	6.16	**0.39**
CABLE	なし	–	–	–	–	7330	–	314	314	2.92
	IC	–	–	0.45	1.00	3024	2.24	376	378	3.41
	SAINV	0.07	–	75.3	0.27	1470	36.0	96.6	133	1.20
	ISAINV	0.06	0.060	58.7	0.32	1388	23.9	96.7	121	1.10
	RIF	0.04	–	88.1	0.38	786	49.2	61.9	111	1.00
	IRIF	0.04	0.080	61.2	0.37	785	28.9	61.3	90.2	0.81

特に，一般に解きにくいとされるシェル要素による離散化で生じた行列 BEAM に対して，**IRIF** が大きな改善効果があったということは注目に価する．

図 3 に，行列 BEAM において，収束した 5 つの前処理つき CG 法の相対残差の履歴を示す．横軸は反復回数，縦軸は相対残差（常用対数目盛）で表す．図から分かるように，**IRIF** の収束は他の前処理に比べて格段に早い．

図 4 に，行列 BEAM において閾値 tol の値が適切であった場合（0.01，0.02 のとき）および不適切であった場合（0.15，0.16 のとき）について，**IRIF** の CPU 時間を示す．図において，横軸に閾値 tol_dd（閾値 tol に対する倍率），縦軸に CPU 時間（単位：秒）をとった．また，閾値 tol_dd が零のときの結果は double dropping を行わない RIF の実験結果にあたる．同様に，**図 5** に行列 CABLE に対する CPU 時間を示す．

図から分かるように，閾値 tol の値が適切な場合，**IRIF** は閾値 tol_dd の値が 2〜5 倍のとき計算時間が激減する．逆に閾値 tol の値が不適切であった場合，閾値 tol_dd の値が 1〜2 倍の範囲でしか計算時間が少なくならず，それ以上閾値 tol_dd の値を大きく設定すると計算時間が増加することが分かる．

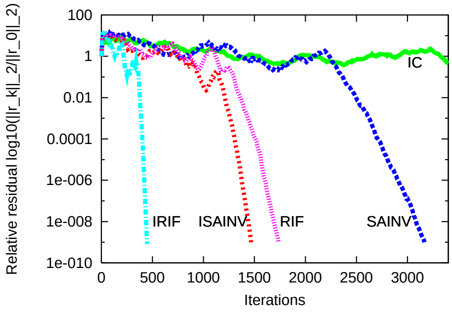

図 3 IC 分解，SAINV，ISAINV，RIF，**IRIF** 前処理つき CG 法の収束履歴（行列 BEAM）
Fig. 3 Convergence history of CG methods with IC, SAINV, ISAINV, RIF and **IRIF** preconditioning for matrix BEAM.

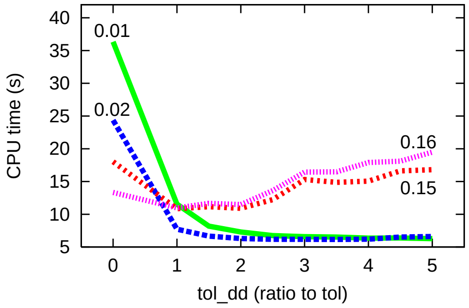

図 4 4 つの閾値 tol に対して閾値 tol_dd を変化させたときの **IRIF** つき CG 法の CPU 時間（行列 BEAM）
Fig. 4 CPU time of CG method with **IRIF** preconditioning for matrix BEAM.

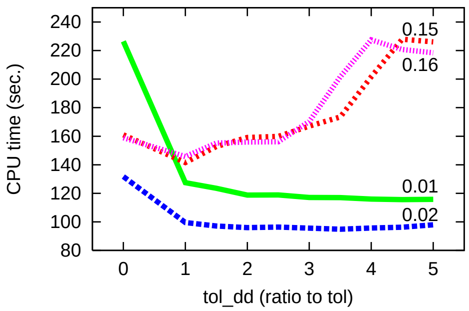

図 5 4 つの閾値 tol に対して閾値 tol_dd を変化させたときの **IRIF** つき CG 法の CPU 時間（行列 CABLE）
Fig. 5 CPU time of CG method with **IRIF** preconditioning for matrix CABLE.

5.3.3 考　　察

表 3 と表 4 から分かるように，テストに用いたすべての行列に対して **IRIF** は ISAINV よりも計算時間が少なかった．これは，逆行列分解の逆行列因子 $\bar{Z}$ の非零要素数が完全コレスキー分解の完全分解因子 $\bar{L}$ の非零要素数に比べてきわめて多く，dropping によって前処理行列の非零要素を棄却した場合，不完全分解因子 L が近似分解因子 Z と比べて相対的に厳密な分解となるためである．実際に，行列 BEAM において閾値 tol を 0.0 として完全分解を行うと，SAINV では非零要素比が 645，RIF では同比 69.3 となり，Z の非零要素が L よりも非常に多いことが分かる．この傾向は他の 5 種類の行列でも同様であった．一方で，近似逆行列分解は反復に逐次計算を含まない前処理であるという利点があり，そのため，現在では並列化およびその応用研究が精力的に行われている[4]．

また，表 3 と表 4 から，**IRIF** において有効な閾値 tol は，RIF の場合と比べて小さい値（0.01〜0.04）に多いことが分かる．したがって，**IRIF** は従来の RIF に比べてより厳密な分解であるといえる．これは，**IRIF** の非零要素比が RIF よりも大きく，非零要素数が多いことからも分かる．このことから，**IRIF** では，従来の RIF では不完全分解に時間がかかるため，計算時間の短縮という観点から有効でなかった小さな値の閾値 tol による分解が，double dropping によって現実的になったためであると考えられる．その結果，**IRIF** つき CG 法では，収束性が大幅に改善し，計算時間が減少したと推測される．このことは，図 4 と図 5 から，小さな値（0.01，0.02 のとき）の閾値 tol による分解と大きな値（0.15，0.16 のとき）の閾値 tol による分解の前処理つき CG 法の計算時間が，double dropping によって逆転していることからも読み取ることができる．

ただし，**IRIF** と同様に double dropping を適用した ISAINV では，最適な閾値 tol の値が従来の SAINV と同様に 0.1 前後の値であることが多く，非零要素比が小さくなっているにもかかわらず反復回数が減少する例（行列 BCSSTK35，行列 BEAM）が見られた．このことから，double dropping には，上記の “小さな値の閾値 tol による分解が現実的となる” 以外にも CG 法の収束性を改善する数学的性質が内在していると考えられ，その理論的な解明は今後の課題である☆．

6. ま と め

本研究では，分解過程で dropping 処理を二重に行うことで，RIF 前処理の有効性をよりいっそう高める **IRIF** 前処理を提案した．そして数値実験によりその

☆ double dropping 処理で用いられる比 $|\frac{d_j}{d_i}|$ の意味については付録 A.2 参照．

有効性を検証した．その結果，**IRIF** は従来の RIF に比べて CG 法の収束性を大幅に改善することが分かった．また，従来解きにくいとされてきたシェル要素の離散化で生じた行列に対して，**IRIF** 前処理つき CG 法による収束性の改善が大きいことが分かった．

謝辞 有限要素法解析に関して有用なご教示をいただいた（株）ホクトシステム原田義明氏および有益なご助言をいただいた匿名の査読者に心より感謝の意を表します．

参 考 文 献

1) 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).
2) 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).
3) Benzi, M. and Tuma, M.: A robust incomplete factorization preconditioner for positive definite matrices, *Numer. Lin. Alg. Appl.*, Vol.10, pp.385–400 (2003).
4) Benzi, M.: Preconditioning techniques for large linear systems: A survey, *J. of Comput. Physics*, Vol.182, pp.418–477 (2002).
5) Bollhöfer, M. and Saad, Y.: On the relations between ILUs and factored approximate inverses, *SIAM J. Matrix Anal. Appl.*, Vol.24 pp.219–237 (2002).
6) Hestenes, M.R. and Stiefel, E.: Methods of conjugate gradients for solving linear systems, *J. of Research of the National Bureau of Standards*, Vol.49, pp.409–436 (1952).
7) 池田優介，藤野清次：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，情報処理学会論文誌：コンピューティングシステム，Vol.45, No.SIG1(ACS4), pp.10–17 (2004).
8) 池田優介，藤野清次：計算機の特性に応じた改良型安定化近似逆行列前処理の有効な利用法，*INFORMATION*（印刷中）
9) 伊理正夫：岩波講座応用数学線形代数 I，岩波書店 (1993).
10) Kharchenko, S.A., Kolotilina, L.Y., Nikishin, A.A., 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) 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, *Math. Comput.*, Vol.31, pp.148–162 (1977).
14) 森 正武，杉原正顕，室田一雄：岩波講座応用数学線形計算，岩波書店 (1994).
15) 日本機械学会（編）：シェルの振動と座屈ハンドブック，技報堂出版 (2003).
16) 戸川隼人：共役勾配法，教育出版 (1977).
17) 矢川元基，金山 寛：コンピュータ機械工学，コロナ社 (1990).
18) University of Florida Sparse Matrix. http://www.cise.ufl.edu/research/sparse/matrices/
19) 有限要素法による構造解析システム：FEMLEEG ユーザガイド，ホクトシステム (2003).

付 録

A.1 RIF の別の導出方法

近似逆行列分解では，逆行列 A^{-1} を $ZD_{ainv}^{-1}Z^t$ のように近似した．このとき，$ZD_{ainv}^{-1}Z^t$ の逆行列は行列 A の不完全分解と見なせる．その結果，

$$A \simeq Z^{-t}D_{ainv}Z^{-1} \tag{15}$$

のように表せる．この式 (15) は，Z^{-t} を計算することで A の不完全分解が構成できることを意味する．一般に，逆行列を求める計算は多くの演算を必要とするため，直接 Z^{-t} を求めるのは現実的ではない．しかし，式 (15) を変形すると Z^{-t} は

$$Z^{-t} \simeq AZD_{ainv}^{-1} \tag{16}$$

と近似的に表せる．この式 (16) の右辺は，近似逆行列分解中に現れる $\frac{d_j}{d_i}$ $(= \frac{\boldsymbol{a}_i^t \boldsymbol{z}_j^{(i-1)}}{d_i})$ に等しいことが分かる．したがって，近似逆行列分解の反復過程において，$\frac{d_j}{d_i}$ を保存すれば Z^{-t} が得られる．通常，不完全分解においては，不完全分解因子の対角行列 D_{ic} と近似逆行列分解因子の対角項 D_{ainv} が等しくなるとは限らない．しかし，RIF では，不完全分解因子 L と近似逆行列分解因子 Z^{-t} が等しくなるように，L を AZD_{ainv}^{-1} によって求める．そのため，対応する対角行列 D_{ic} も D_{ainv} に等しいことが分かる．

A.2 double dropping 処理の中の比 $|\frac{d_j}{d_i}|$ の意味

A–直交過程に基づく近似逆行列分解では，上三角行列 Z の各列ベクトル $\boldsymbol{z}_i$ を，行列 A に対して互いに A–直交，すなわち $j\ (\neq i)$ に対して

$$\boldsymbol{z}_j^t A \boldsymbol{z}_i = 0 \tag{17}$$

となるように構成する．このとき，$Z^t AZ$ を計算すると，対角項のみに非零要素が現れるため，

$$Z^t AZ = D \tag{18}$$

と表すことができる．そして，dropping によって A–直交過程が不完全に行われた場合，式 (18) の右辺項は厳密に対角行列とはならないため，

$$Z^t AZ = D_{ainv} + E \tag{19}$$

と表すことができる．ここで，E は A–直交化の不完全さを表す誤差行列とする．この誤差行列 E を零行列と見なして式 (19) を変形すると，$ZD_{ainv}^{-1}Z^t$ のように逆行列を近似することができる．

このとき，SAINV の算法中に現れる $d_j(= \boldsymbol{z}_j^t A\boldsymbol{z}_i)$ と $d_i(= \boldsymbol{z}_i^t A\boldsymbol{z}_i)$ は，E と D_{ainv} の要素の値に各々等しいことに注目する．すなわち，double dropping 処理中の $|\frac{d_j}{d_i}|$ は，誤差行列 E の要素を対角行列 D_{ainv} の値で正規化された値と見なせる．double dropping では，$|\frac{d_j}{d_i}|$ の値が閾値 tol_dd より大きいときのみ A–直交化（すなわち，$\boldsymbol{z}_j^{(i-1)}$ の更新）処理を追加した．従来の dropping が前処理行列の疎性の保持が目的であるのに対して，新たな dropping は誤差行列 E の要素の値を小さくし近似分解精度の向上が目的である．

(平成 15 年 10 月 10 日受付)
(平成 16 年 1 月 26 日採録)

池田　優介
1979 年生．2004 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．2004 年 4 月（株）東芝デジタルメディアネットワーク社入社．共役勾配法の前処理ついて研究を行う．

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．日本応用数理学会会員．

柿原　正伸
1981 年生．2002 年 3 月九州大学工学部情報工学科卒業．九州大学大学院システム情報科学府修士課程在籍中．共役勾配法の不完全分解前処理に興味を持つ．

井上　明彦（正会員）
1980 年生．2002 年 3 月九州大学工学部情報工学科卒業．九州大学大学院システム情報科学府修士課程在籍中．共役勾配法の並列化と乱数に興味を持つ．