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

2006－HPC－106 (7)
2006／6／12

Sherman-Morrison公式による前処理行列計算の精度評価とその再構成について

張 臨傑[†] 森屋 健太郎[††] 野寺 隆[†††]

近年，Sherman-Morrison 公式を用いて，近似逆行列を計算し，それを前処理行列とする前処理法は Bru ら [*SIAM J. Sci. Comput.*, 25 (2003), pp. 701-715] によって提案された．ただし，この近似逆行列を計算する過程の中で，演算コストを削減するため，Dropping を行う必要がある．本稿では，Dropping は前処理の効果に影響を与える一番の要素であることを述べ，Dropping による誤差について解析し，低いコストで，前処理行列を再構成する手法を提案する．

Error Evaluation for the Preconditioner Computed with Sherman-Morrison Formula and Its Re-construction

LINJIE ZHANG ,[†] KENTARO MORIYA [††] and TAKASHI NODERA[†††]

Bru et al.[*SIAM J. Sci. Comput.*, 25 (2003), pp. 701-715] have proposed a new preconditioner, which computing the approximate inverse matrix with Sherman-Morrision formula. However, in order to reduce the computational cost, dropping is necessary. In this paper, we show that the error caused by dropping influences the preconditioning's effect most. We analyze the error caused by dropping and propose a easy way for the re-construction of the preconditioner.

1. はじめに

大型で疎な連立1次方程式

$$Ax = b, \quad A \in R^{n \times n}, \quad x, b \in R^n$$

の解法として，クリロフ部分空間法はよく利用される算法である．しかし，問題によって，クリロフ部分空間法の収束が停滞したり，収束しないなどの欠点がある．そこで，クリロフ部分空間法の収束を改善するために，行列の前処理はよく使われている．例えば，右前処理は係数行列 A の右側に前処理行列 M を掛けて，次のような線形系にクリロフ部分空間法を適用する方法である．

$$AMy = b, \quad x = My \qquad (1)$$

行列の前処理には，このような右側前処理のほかに，左側前処理と両側から前処理行列を掛ける前処理もある．通常，前処理行列 M は，AM が A より良い固有値の分布になるようなものを選ぶことになる．一般的に，AM が単位行列 I_n に近いほうが望ましい．しかし，多くの問題において，AM の固有値が原点から離れる所でクラスタリングする場合にも良い収束が得られる．行列 AM の良いスペクトル分布のほか，前処理行列 M が簡単に求められることや低いコストで実装できることも必要である．

式 (1) 前処理行列 M を求める様々な方法が提案されている．例えば，(i) 係数行列 A を下三角行列 L と上三角行列 U の積に近似分解し，$(LU)^{-1}$ を前処理行列にする ILU 分解法．(ii)$I_n - AM$ のノルム（例えば，Frobenius ノルム）を最小にするように前処理行列 M を求める SPAI 法と MR 法[4),5)]．それから，(iii)$WAZ^T \approx D$ を満たす疎な上三角行列 W，Z と対角行列 D を求めて，$ZD^{-1}W^T$ を前処理行列として使う SAINV 法[1),2)] などがある．

上記のほか，Sherman-Morrison 公式による前処理も提案されている[3)]．それは $B^{-1} - A^{-1}$ を $U\Omega^{-1}V^T$ に分解し，それを前処理行列にする前処理法である．ただし，行列 B^{-1} は既に知られているか，又は簡単に求められるものである．行列 U，V は一般行列，Ω は対角行列である．しかし，たとえ行列 A が疎な行列としても，U，V は疎な行列になることは極めて少ない．U，V の疎な性質を保つため，つまり前処理コストを削減するために，Dropping を行う必要がある．本稿では，行列要素の Dropping は前処理を影響する

† 慶應義塾大学大学院理工学研究科
Graduate School of Science and Technology, Keio University
†† 青山学院大学理工学部
Faculty of Science and Technology, Aoyama Gakuin University
††† 慶應義塾大学理工学部
Faculty of Science and Technology, Keio University

一番の要因であることを述べ，Droppingによる誤差を解析する．この誤差解析に基づいて，低いコストで誤差を減少させる近似逆行列の再構成手法を提案する．

2章では，まずSherman-Morrison公式による前処理について簡単に述べる．その後，Droppingは前処理に影響する一番の要因であることについて述べる．3章では，Droppingによる誤差について解析し，これに基づいて前処理行列の再構成を提案する．4章では，提案した手法の数値実験について述べ，その有効性を示す．

2. Sherman-Morrison公式による前処理

本章では，まずSherman-Morrison公式とSherman-Morrison公式による前処理を簡単に述べる．その次に，Droppingは前処理の効果に影響する一番の要因であることについて述べる．

2.1 Sherman-Morrison公式

[Sherman-Morrison公式] 与えられた正則行列 $B \in R^{n\times n}$ とベクトル $x \in R^n, y \in R^n$ に対して，もし $r = 1 + y^T B^{-1} x \neq 0$ なら，行列 $A = B + xy^T$ も正則行列であり，その逆行列は

$$A^{-1} = B^{-1} - r^{-1} B^{-1} x y^T B^{-1} \tag{2}$$

となる．

Sherman-Morrison公式は，様々な問題に利用できる．例えば，方程式(1)の係数行列が変化した際の新しい近似解の算出，又は逆行列の更新などである．この公式を利用して，行列 $B^{-1} - A^{-1}$ の行列分解もできる．ただし，U, V は一般行列であり，Ω は対角行列である．B^{-1} の逆行列は既に知られているか，又は簡単に求められるものである．

2.2 Sherman-Morrison公式による前処理

ベクトル $\{x_k\}_{k=1}^n \in R^n$, $\{y_k\}_{k=1}^n \in R^n$ と行列 $A_0 \in R^{n\times n}$ が次式を満たすものとする．

$$A = A_0 + \sum_{k=1}^{n} x_k y_k^T$$

ここで，$A_k = A_0 + \sum_{k=1}^{k} x_k y_k^T$ を定義すると，$A_{k+1} = A_k + x_k y_k^T$, $A_n = A$ となる．

もし x_k, y_k と A_{k-1} がSherman-Morrison公式の条件を満たすならば，Sherman-Morrison公式(2)を適用して，A_k の逆行列は次の式で計算できる．

$$A_k^{-1} = A_{k-1}^{-1} - r_k^{-1} A_{k-1}^{-1} x_k y_k^T A_{k-1}^{-1}$$

故に，次式が得られる．$A^{-1} = A_n^{-1} = A_0^{-1} - \sum_{k=1}^{n} r_k^{-1} A_{k-1}^{-1} x_k y_k^T A_{k-1}^{-1}$

この式を整理すると

$$A_0^{-1} - A^{-1} = \sum_{k=1}^{n} r_k^{-1} A_{k-1}^{-1} x_k y_k^T A_{k-1}^{-1} \tag{3}$$

となる．次に，式(3)の右辺を行列方式で書き換えると

$$A_0^{-1} - A^{-1} = \Phi \Omega^{-1} \Psi^T \tag{4}$$

になる．ただし，$\Phi = [A_0^{-1}x_1\ A_1^{-1}x_2\ \ldots A_{n-1}^{-1}x_n]$,

$$\Omega^{-1} = \begin{bmatrix} r_1^{-1} & & \\ & \ddots & \\ & & r_n^{-1} \end{bmatrix}, \Psi^T = \begin{bmatrix} y_1^T A_0^{-1} \\ \vdots \\ y_n^T A_{n-1}^{-1} \end{bmatrix}$$

$\Phi\Omega^{-1}\Psi^T$ を計算するには，A_k^{-1} を明確に計算し，保存する必要がある．しかし，Bruら[3]は，A_k^{-1} の計算を回避する方法を提案している．．具体的に，新たなベクトル列 u_k, v_k を

$$u_k := x_k - \sum_{i=0}^{k-1} \frac{v_i^T A_0^{-1} x_k}{r_i} u_i,$$

$$v_k := y_k - \sum_{i=0}^{k-1} \frac{y_k^T A_0^{-1} u_i}{r_i} v_i$$

というように定義すれば，以下の式が成り立つ．

$$A_{k-1}^{-1} x_k = A_0^{-1} u_k, \tag{5}$$

$$y_k^T A_{k-1}^{-1} = v_k^T A_0^{-1} \tag{6}$$

$$r_k = 1 + y_k^T A_0^{-1} u_k = 1 + v_k^R A_0^{-1} x_k \tag{7}$$

ここで，式(5)と式(6)を式(4)に代入すると，

$$A_0^{-1} - A^{-1} = A_0^{-1} U \Omega^{-1} V^T A_0^{-1}. \tag{8}$$

ただし，$U = (u_1, u_2, \ldots, u_n)$, $V = (v_1, v_2, \ldots, v_n)$ である．

A_0, x_k, y_k の選び方について，Bruら[3]は，次の選択を提案した．

$$A_0 = sI_n, \quad s > 0,$$

$$x_k = e_k,$$

$$y_k = (a^k - a_0^k)^T.$$

ただし，$I_n \in R^{n\times n}$ は単位行列であり，$e_k \in R^n$ は I_n の k 番目の列ベクトルである．a^k と a_0^k は行列 A と A_0 の k 番目の行ベクトルである．A_0, x_k, y_k を式(8)に代入すると

$$sI_n - A^{-1} = s^{-2} U \Omega^{-1} V^T. \tag{9}$$

u_k, v_k は，次式のようになる．

$$u_k = x_k - \sum_{i=0}^{k-1} \frac{(v_i)_k}{s r_i} u_i, \quad v_k = y_k - \sum_{i=0}^{k-1} \frac{y_k^T u_i}{s r_i} v_i$$

このように A_0, x_k, y_k を選べば，A_0^{-1} と u_k, v_k が簡単に求められるだけではなく，行列 U の正則性も保証できる．さらに，$s \notin \sigma(A)$ なら，行列 V も正則である．ただし，$\sigma(A)$ は A のスペクトル半径である．

式(9)の分解が得られたなら，それを前処理行列として利用することになる．ここで，前処理された係数行列のスペクトル半径は $\sigma(AM) \approx \sigma(A)/s - 1$ となる．もし $s > \rho(A)$ なら，行列 AM の固有値が複数平面の左半面に含まれ，クリロフ部分空間法にとって良い固有値分布である．式(9)の分解を求めるアルゴリズムを図1で示す．

しかし，たとえ行列 A が疎な行列であっても，できあがりの行列 U と V は密行列になる可能性は大きい．そこで，U と V は疎な性質を維持するために，つま

```
1   Set x_k = e_k, y_k = (a^k - se^k)^T,
      (k = 1, ..., n)
2   for k = 1, ..., n
3     u_k = y_k
4     v_k = y_k
5     for i = 1, ..., k - 1
6       u_k = u_k - ((v_i)_k / (s r_i)) u_i
7       v_k = v_k - ((y_k^T u_i) / (s r_i)) v_i
8     end for
9     r_k = 1 + (u_k)_k / s
10    dropping u_k, v_k
11  end for
12  Return U = [u_1, u_2, ..., u_n],
      V = [v_1, v_2, ..., v_n],
      and Ω = diag(r_1, r_2, ..., r_n)
```

図1 式(9)の分解を計算するアルゴリズム

り前処理のコストを削減するために，Droppingを行なう必要がある．図1の10行目にてDroppingを行う．Bruら[3]は行列AがM行列なら，Droppingによって，行列U, V, Ωの正則性が変らないことを証明した．この前処理の有効性はBruら[3]により示されている．

2.3 収束に影響するパラメータ

前章で紹介したSherman-Morrison公式による前処理を実装する際に，sとDroppingの閾値を設定する必要がある．ここで，この2つのパラメータの収束に対する影響を考察する．

まず，パラメータsの影響を調べることにする．Bruら[3]の数値実験によると，sが大きくなると，収束は速くなることが報告されている．しかし，sを大きくすることにつれ，非ゼロ要素の数も増える．sの影響か，非ゼロ要素の数の影響かをはっきりさせるために，非ゼロ要素の数を増えさせず，sだけ大きくする実験を行うことにする．

ここで，非ゼロ要素の数を増加させないために，以下の関係を利用する[3]．

$$u_k = u_k^*$$
$$v_k = v_k^* - (s-1)w_k$$
$$w_k = x_k - \sum_{i=1}^{k-1} \frac{(y_i^*)^T u_k^*}{r_i^*} w_i$$
$$r_k = r_k^*/s$$

ただし，u_k^*, v_k^*, r_k^*は$s = 1.0$の時のu_k, v_k, r_kである．Droppingを行った後のベクトルu_k^*, v_k^*, r_k^*とw_kを予めて保存すれば，sを大きくしても，非ゼロ要素が増えることはない．しかも，u_k, v_k, r_kを簡単に計算できる．この方法を使って，Vの非ゼロ要素の数を固定し，4章の数値例を使って，sだけの影響を調べる．ただし，メッシュ幅hを1/33とし，Dhを10^{-7}とする．実験環境と条件については，4章を参考してほしい．GMRES(50)を使用した場合，sを$1.5\|A\|_\infty$から$1500\|A\|_\infty$に大幅に増大させても，反復回数は4040から4042に変わるだけで，ほぼ影響ないことが分かった．

以上のことから，s自身は収束にほぼ影響を与えないことが分かる．ただし，sが大きくなると，Vの要素の絶対値が大きくなり，閾値より大きい要素が増える．つまり，Droppngする要素の数が減り，収束が速くなる．もし，非ゼロ要素の数を固定すれば，sを大きくしても，収束はほぼ変わらない．従って，Droppingは前処理の効果に影響する一番の要因である．

次の章でDroppingによる誤差について解析し，低いコストで誤差を減少させる手法を提案する．

3. Droppingによる誤差と再構成

3.1 Droppingによる誤差

まず，行列UのDroppingによる誤差を考える．Droppingしない場合の行列U, Vの列ベクトルをu_k, v_kとし，行列Ωの対角要素をr_kとする．Droppingする前の列ベクトルを$\dot{u}_k$, $\dot{v}_k$とする．Droppingした後の列ベクトルを$\bar{u}_k$, $\bar{v}_k$とし，行列$\bar{\Omega}$の対角要素を$\bar{r}_k$とする．$\bar{U} = (\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_n)$, $\bar{V} = (\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_n)$, $\bar{\Omega} = \mathrm{dial}\ (\bar{r}_1, \bar{r}_2, \ldots, \bar{r}_n)$. $n(X, i)$は行列Xのi番目の列ベクトルのDroppingする要素の数を表し，Droppingするj番目の要素のインデックスを$d(X, i, j)$で表す．

$$\dot{u}_k - u_k = \sum_{i=1}^{k-1} \left(\frac{(v_i)_k}{s r_i} u_i - \frac{(\bar{v}_i)_k}{s \bar{r}_i} \bar{u}_i \right)$$

係数行列AがM行列，かつ$s > \max_{i=1,\ldots,n}\{a_{ii}\}$の場合，

$$\begin{cases} U \geq \bar{U} \geq 0 \\ \bar{r}_i \geq r_i > 0, \quad i = 1, \ldots, n \\ 0 \geq \bar{V} \geq V \end{cases}$$

を満たすため[3]，$\frac{(v_i)_k}{s r_i} u_i - \frac{(\bar{v}_i)_k}{s \bar{r}_i} \bar{u}_i \leq 0$. 従って，

$$\|\bar{u}_k - u_k\|_1 = \|\bar{u}_k - \dot{u}_k + \dot{u}_k - u_k\|_1 \qquad (10)$$
$$= \sum_{i}^{n(U,\,k)} u_{i,\,d(U,\,k,\,i)} + \sum_{i=1}^{k-1} \left(\frac{(\bar{v}_i)_k}{s \bar{r}_i} \bar{u}_i - \frac{(v_i)_k}{s r_i} u_i \right)$$

式(10)一番目の項目$\sum_i^{n(U,\,k)} u_{i,\,d(U,\,k,\,i)}$は$\dot{u}_k$からDroppingした要素の和である．同様に，次式が成立する．

$$\dot{v}_k - v_k = \sum_{i=1}^{k-1} \left(\frac{y_k^T u_i}{s r_i} v_i - \frac{y_k^T \bar{u}_i}{s \bar{r}_i} \bar{v}_i \right),$$
$$\left(\frac{y_k^T u_i}{s r_i} v_i - \frac{y_k^T \bar{u}_i}{s \bar{r}_i} \bar{v}_i \right) \geq 0, \quad i = 1, \ldots, n$$

$$\|\bar{v}_k - v_k\|_1 = \|\bar{v}_k - \hat{v}_k + \hat{v}_k - v_k\|_1 \qquad (11)$$
$$= \sum_{i}^{n(V,\,k)} -v_{i,\,d(V,\,k,\,i)} + \sum_{i=1}^{k-1}\left(\frac{y_k^T u_i}{s r_i} v_i - \frac{y_k^T \bar{u}_i}{s \bar{r}_i}\bar{v}_i\right)$$

ここで，閾値を小さくすれば，式 (10) と (11) の 2 つの項目両方を減少することができるが，u_i, v_i, $i = 1, \ldots, k-1$ の非ゼロ要素の数が増えることによって，$\bar{u}_k$,$\bar{v}_k$ の計算するコストは指数的に増え膨大になり，前処理のコストは高くなる．しかし，以下のように，式 (10) と (11) の一番目の項だけ減少させるのはとても簡単である．

まず，$\bar{u}_i$, $\bar{v}_i$, $i = 1, \ldots, k-1$ を使って，$\hat{u}_k$, $\hat{v}_k$ を計算する．次に，$\hat{u}_k$, $\hat{v}_k$ に Dropping を行い，$\bar{u}_k$ と $\bar{v}_k$ を得る．次に，Dropping した要素の一部を $\bar{u}_k$, $\bar{v}_k$ に加えて，$\tilde{u}_k$ と $\tilde{v}_k$ を得る．得られる $\tilde{U}\tilde{\Omega}^{-1}\tilde{V}^T$ を前処理行列として使う．明らかに，次式が成立する．

$$\|u_k - \tilde{u}_k\|_1 \le \|u_k - \bar{u}_k\|_1,$$
$$\|u_k - \tilde{v}_k\|_1 \le \|v_k - \bar{v}_k\|_1.$$

つまり，$\tilde{U}\tilde{\Omega}^{-1}\tilde{V}^T$ の Dropping による誤差は $\bar{U}\bar{\Omega}^{-1}\bar{V}^T$ より小さい．本稿では，$\bar{U}$, $\bar{V}$ から $\tilde{U}$, $\tilde{V}$ を構成する過程を再構成と定義する．又，ここでは，簡単のため，ベクトルの 1 ノルムを使っているが，2 ノルムも成り立つ．次の章において，再構成の実装について述べる．

3.2 近似逆行列の再構成の方法

Dropping によって，行列 U と V は疎な行列になる．アルゴリズム (図 1) を実装する際に，行列の非ゼロ要素の値とインデックスだけを保存する（例えば，Sparse-Row フォーマット）のは一般的である．こうすると，行列を保存するためのメモリ量を削減するだけではなく，$y_k^T u_i$ の計算と u_k, v_k の更新コストは U と V の非ゼロ要素の数だけに依存することになる．本稿では，Sparse-Row フォーマットを例にして近似逆行列の再構成手法について述べる．その他の非ゼロ要素の値とインデックスだけ保存するフォーマットにも適用する．

まず，図 2 で示した 10 次元の三重対角行列を使って，Sparse-Row フォーマットについて簡単に述べる．三重対角にある要素の値はそれぞれ a, b, c である．

Sparse-Row フォーマットは図 3 で示したように行列を格納する．nnzrow[ ] は各行ベクトルの非ゼロ要素の数を記録する配列，ja[ ][ ] は行ごとの非ゼロ要素のインデックスを格納する配列のアドレスを記録する配列であり，ma[ ][ ] は行ごとの非ゼロ要素の値を格納する配列のアドレスを記録する配列である．ただし，図 3 では配列のインデックスは 1 から．始めることにする．

行列 U^T, V^T を Sparse-Row フォーマットで格納するとする．図 1 の 10 行目に Dropping を行うと同時に，残される要素の数を統計して，nnzrow[ ] の適切な場所に書き込む．その後，非ゼロ要素の値とインデックスを保存できるスペースを動的に請求し，その

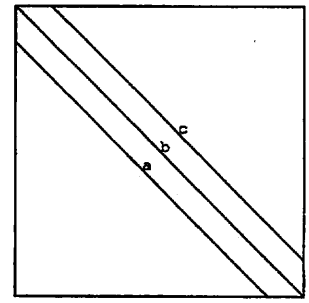


図 2 3 重対角行列

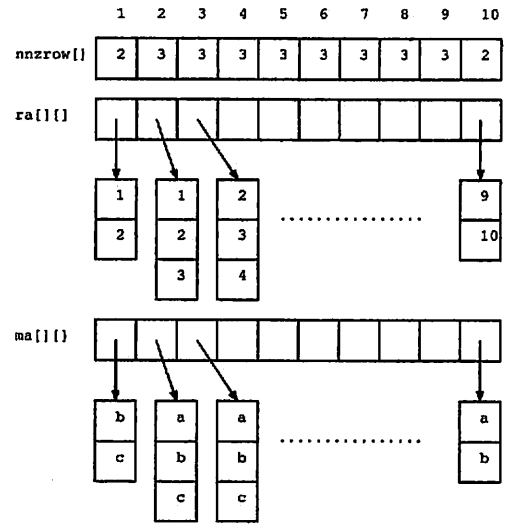


図 3 Sparse-Row フォーマットで 3 重対角行列を保存

スペースのアドレスを ma と ja に格納する．したがって，Dropping を行う際に，ベクトルのすべての値を計算しなければいけない．その後，絶対値が閾値より小さい要素を捨てる．

再構成を行う際に，行列 $\tilde{U}$, $\tilde{V}$ と $\bar{U}$, $\bar{V}$ を別々に保存する必要はない．図 4 で示した簡単な 5 重対角行列を使って $\tilde{U}$ の再構成について述べる．対角 a, b, c の値は閾値より大きいものとし，d, e の値は閾値より小さいが，再構成の際に使うものとする．5 重対角以外の要素はすべて捨てる．つまり，a, b, c を含む 3 重対角行列は前節の $\bar{U}$ になる．a, b, c, d, e を含む 5 重対角行列は前節の $\tilde{U}$ になる．アルゴリズム (図 1) の過程の中，図 5 と図 6 で示した構造で $\tilde{U}$ と再構成するための情報を保存する．こうすると，アルゴリズム (図 1) において，$\tilde{U}$ はすべての計算に影響しない．アルゴリズム (図 1) の過程が終った後，再構成を行う．つまり，図 5 の配列 nnzrow[ ] に図 6 の配列の値を足して，$\tilde{U}$ (図 7) を得る．$\tilde{V}$ も同じ方法で計算

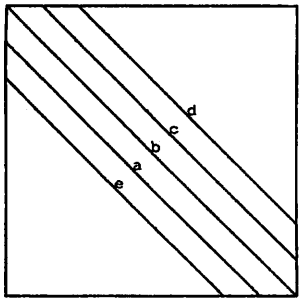
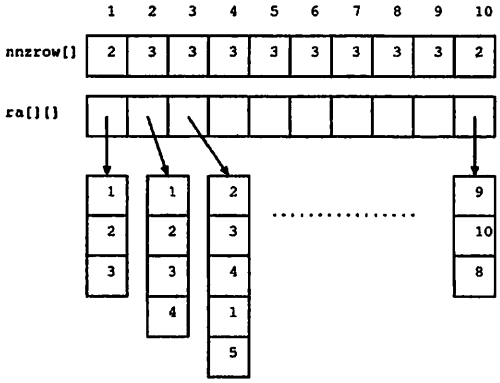


図 4　5 重対角行列



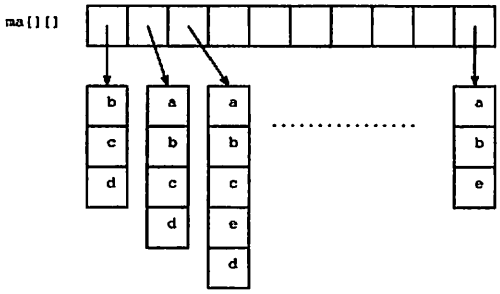


図 5　再構成を行う前の行列要素の保存

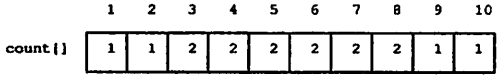
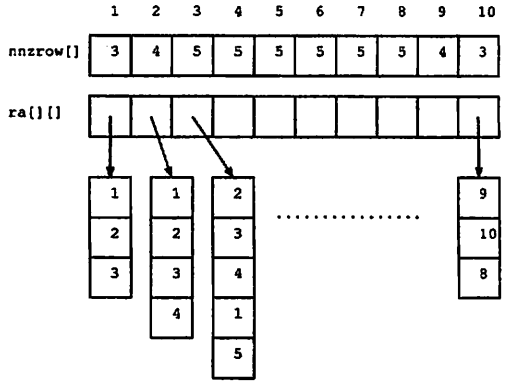


図 6　再構成のために保存した非ゼロ要素の数

することができる．

提案した手法の有効性は 4 章の数値実験で示すことにする．また，この手法は Dropping を行うほかの近似行列分解，近似逆行列，または近似逆行列分解にも適用できる．



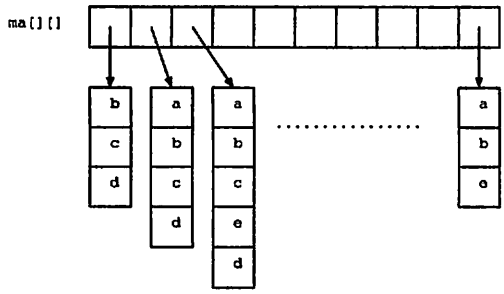


図 7　再構成を行った後の行列要素の保存

4. 数値実験

前章で提案した前処理行列の再構成について，その有効性示すために，数値実験を行った．ただし，近似解は右側前処理を利用する GMRES(m) 法[6] によって計算するものとする．反復回数については，GMRES の中の Arnoldi 過程の 1 反復につき 1 回と数える．計算時間については，Clock() 関数で求めた値を秒単位で表す．数値実験は以下の環境と条件で行なう．

- 計算機：DELL PowerEdge 1750
- OS：Red Hat Linux7.2
- CPU：3.02 GHz ×1 インテル (R) Xeon(R)
- メモリ：4GB
- 収束判定条件：$\| r_k \|_2 / \| r_0 \|_2 \leq 1.0 \times 10^{-12}$
- 初期値ベクトル：$x_0 = (0, 0, \ldots, 0)^T$
- プログラム言語：C 言語
- 計算精度：倍精度

要素の Dropping については，Bru ら[3] の手法を利用する．つまり，U に絶対閾値 $tolU$ を使用する．V に相対閾値 $\|A\|_\infty \times tolV$ を使用する．Bru ら[3] の数値実験では，$tolU = tolV$ を殆ど 0.1 に設定した．しかし，場合によって，0.1 で非ゼロ要素の数が少なすぎで，0.01 に設定したケースもある．このことを踏ま

表 1 非ゼロ要素の数

Dh	tol	再構成	nnz(U)	nnz(V)	T
2^{-5}	0.1	しない	23800	81416	10
2^{-5}	0.1	する	43143	136405	10
2^{-5}	0.01	しない	1437701	3550026	58
2^{-6}	0.1	しない	23815	81427	10
2^{-6}	0.1	する	43158	136651	10
2^{-6}	0.01	しない	1444466	3550522	58
2^{-7}	0.1	しない	23815	81423	10
2^{-7}	0.1	する	43158	137273	10
2^{-7}	0.01	しない	1453389	3564194	58

えて，本稿では，閾値を 0.1 に設定し，閾値 ×0.1 より大きい要素を再構成のために保存する．また，s の値は収束にほぼ影響しないため，$1.5\|\|A\|\|_\infty$ を使用する．

[例題] 次の領域 $\Omega = [0,1] \times [0,1]$ における偏微分方程式境界値問題を 5 点中心差分法で離散化する．

$$-u_{xx} - u_{yy} + D((y - \frac{1}{2})u_x) +$$
$$(x - \frac{1}{3})(x - \frac{2}{3})u_y)) - 43\pi^2 u = G(x,y)$$
$$u|_{\partial\Omega} = 1 + xy,$$

ただし，$G(x,y)$ は真の解が $u = 1 + xy$ になるように決める．メッシュ幅 h は 1/65 に設定したので，得られる線形方程式の次元は 4096 である．Dh の値を 2^{-5}, 2^{-6}, 2^{-7} に変化して，異なる線形システムを得る．ただし，ここで得られる係数行列は M 行列ではないが，M 行列に近い L 行列であるため，殆どのベクトル $\bar{u}_k$ と $\bar{v}_k$ に対して，式 (10) と式 (11) は成り立つ．つまり，再構成による誤差の減少は期待できる．

GMRES(m) 法のリスタート周期を 30, 40, 50 に設定し，数値実験を行った．まず再構成しない場合と再構成する場合のそれぞれの U と V の非ゼロ要素の数と計算時間を表 1 で示す．ただし，nnz(U), nnz(V) はそれぞれ行列 U と V の非ゼロ要素の数を意味する．T は前処理行列の算出時間（秒）である．表 1 から以下のことが分かる．

1. 近似逆行列の再構成の計算コストは増える非ゼロ要素を保存するだけで，計算時間は変わらない

2. 閾値を小さくすると，非ゼロ要素の数が増え，前処理のコストは高くなる．

次に，近似逆行列を再構成しない（従来の）前処理行列と再構成を行う前処理行列を GMRES(m) 法に適用して，離散化した線形方程式を解く．その結果を表 2 と表 3 に示す．ただし，I_g は GMRES(m) 法の収束するまでの反復回数，T_g は GMRES(m) 法の処理時間 (秒) を意味する．表 2 と表 3 から，反復回数と計算時間は約 50% ～ 20% 削減でき，再構成によって収束が改善されることが確認できた．また，再構成を行う場合，一回反復の計算時間は若干増える．それは非ゼロ要素が増えたためである．並列計算の場合，この影響を軽減することができる．ちなみに，閾値を 0.01 にして得られる前処理行列を使うと，反復回数は再構成する場合の 50% ぐらいまで減少するものの，GMRES(m) の処理時間はほぼ 4 倍かかった．前処理の演算コストは高すぎるので，実用的とは言えない．

表 2 再構成しない場合の数値実験の結果

Dh	GMRES(30)		GMRES(40)		GMRES(50)	
	I_g	T_g	I_g	T_g	I_g	T_g
2^{-5}	3025	16	2809	16	2572	17
2^{-6}	2796	15	2636	16	2576	17
2^{-7}	3820	20	2519	15	2711	18

表 3 再構成する場合の数値実験の結果

Dh	GMRES(30)		GMRES(40)		GMRES(50)	
	I_g	T_g	I_g	T_g	I_g	T_g
2^{-5}	2306	13	1805	11	1713	12
2^{-6}	2112	12	1742	11	1779	12
2^{-7}	1857	11	1640	10	1871	13

以上のことから，本稿で提案した近似逆行列を再構成する方法は Dropping による誤差を減少させる簡単且つ有効な手法と思われる．

5. おわりに

本稿では，Sherman-Morrison 公式による前処理を簡単に述べた．Dropping による誤差を解析し，それを減少させる簡単かつ有効な再構成手法を提案した．一般行列，又は，その他の Dropping を行う算法への適用については今後の課題である．

参考文献

1) Benzi, M., Cullum, J. K., and Tůma, M., Robust approximate inverse preconditioning for the conjugate gradient method, *SIAM J. Sci. Comput.*, 22 (2000), pp. 1318-1332.
2) Benzi, M., Haws, J. C., and Tůma, M., Preconditioning highly indefinite and nonsymmetric matrices, *SIAM J. Sci. Comput.*, 22 (2000), pp. 1333-1353.
3) Bru, R., Cerdán, J., Marín, J., and Mas, J., Preconditioning sparse nonsymmetric linear systems with the Sherman-Morrison formula, *SIAM J. Sci. Comput.*, 25 (2003), pp. 701-715.
4) Grote, M. J. and Huckel, T., Parallel preconditioning with sparse approximate inverses, *SIAM J. Sci. Comput.*, 18 (1997), pp. 838-853.
5) Huckel, T., Approximate sparsity patterns for the inverse of a matrix and preconditioning, *Appl. Numer. Math.*, 30 (1999), pp. 291-303.
6) Saad, Y., and Schultz, M. H., GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Stat. Comput.*, 7 (1986), pp. 856-869.