Sherman-Morrison 法の並列化による近似逆行列の計算

森屋 健太郎[†] 張 臨傑[††] 野寺 隆[†††]

近年，Sherman-Morrison 法は，大規模な連立 1 次方程式の前処理技法の中で近似逆行列の計算法の 1 つとして，Bru ら［SIAM J. Sci. Comput., 25(2003), pp.701–715］によって提案され，その有効性が報告されている．しかし，行列分解で得られる列ベクトルの計算に依存関係があるので，並列化して計算速度を向上させることはそれほど簡単ではない．本稿では Naik［IBM System J., 34(1995), pp.273–291］の実装法に基づき，Sherman-Morrison 法における行列分解の計算を部分的に並列化する手法を提案する．本稿で提案した手法は，PE 6 台を搭載した PC クラスタ上で 3 倍以上の速度向上を可能にした．また，この前処理行列は，MR 法や ILU 分解による前処理行列を用いるときよりも，GMRES(m) 法の残差ノルムの収束を向上させる場合があることを示す．

The Computation of the Approximate Inverse by Parallelizing the Sherman-Morrison Formula

Kentaro Moriya,[†] Lingie Zhang[††] and Takashi Nodera[†††]

The Sherman-Morrison formula is one of schemes that compute the approximate inverse preconditioner for the large linear systems of equations. This formula is proposed by Bru, et al. [SIAM J. Sci. Comput., 25(2003), pp.701–715]. It is also reported that the preconditioner computed by this formula performs fairly well. However, it is not so easy to parallelize this formula and improve its computation time since the column vectors are dependent of each other. In this paper, we improve the performance of the Sherman-Morrison preconditioner by parallelizing the matrix factorization based on Naik [IBM System J., 34(1995), pp.273–291] implementation. The proposed technique makes the speedup of more than 3 times on the PC cluster system with 6 processors. It is also verified that this preconditioner often enables the residual norm of GMRES(m) method to converge more rapidly than the MR or ILU preconditioner.

1. はじめに

大型で疎な $n \times n$ の非対称正則行列 A を係数とする連立 1 次方程式

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

について考える．方程式 (1) の近似解を非定常反復法により求めるには，通常，反復回数を減少させるために行列の前処理が用いられる．特にその中でも，近似逆行列はよく知られた前処理行列の 1 つである．今，前処理行列を M^{-1} とすると $M^{-1} \cong A^{-1}$ となるように計算することになる．ここで，前処理とは方程式 (1) を

$$\begin{cases} AM^{-1}\boldsymbol{y} = \boldsymbol{b} \\ \boldsymbol{x} = M^{-1}\boldsymbol{y} \end{cases} \tag{2}$$

もしくは

$$M^{-1}A\boldsymbol{x} = M^{-1}\boldsymbol{b} \tag{3}$$

のように変形する技法のことをいうが，本稿では方程式 (2) の形式で前処理を実装する．

近年，Bru ら[12] は Sherman-Morrison 法を用いて近似逆行列を求めることを提案し，この手法による行列の前処理の有効性を示している．しかし，Sherman-Morrison 法による近似逆行列は，行列の列ベクトルどうしに依存関係があるので，並列化には不向きであることが指摘されている．このような難点を改善するために，本稿では Naik[4] の提案した実装法を利用して前処理行列の計算を部分的に並列化することを考える．彼は，ブロック三重対角行列を係数とする線形システム（BT Bench Mark という）を LU 分解で

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

解く過程をパイプライン制御により部分的に並列化している．それに対して，本稿ではこのような手法をSherman-Morrison 法による近似逆行列の計算に漸化式を適用し，この計算過程を部分的に並列化する手法を提案する．

数値実験では，6 個の PE を持つ PC クラスタシステムに，提案した手法を用いて Sherman-Morrison 法による近似逆行列計算の実装を行い，この前処理行列を計算するときの PE 台数効果を計測した．また，この前処理行列を GMRES(m) 法[1] に適用し，残差ノルムの収束の様子を評価した．その結果，提案した手法が部分的な並列化であるにもかかわらず，一定の速度向上を発揮することを示す．また，ILU 分解[3] や MR 法[7] による行列の前処理を用いる場合よりも残差ノルムが安定して収束する場合があることを示す．

2 章ではそれぞれ ILU 分解，MR 法および Sherman-Morrison 法による前処理行列の計算法について簡単に述べる．3 章では Sherman-Morrison 法を並列化する方法について提案する．4 章では PC クラスタによる数値実験の結果を示し，5 章において数値結果から得られた結論を述べる．なお，本稿では PE（Processing element）1 台を CPU 1 台として考えることにする．

2. 前処理行列の計算法とその並列化

本章では，不完全 LU 分解（ILU 分解ともいう）や近似逆行列を用いる前処理の各算法とその並列実装法について考える．

2.1 ILU 分解とその並列化

ILU 分解による前処理行列の計算は，行列 A を $A = LU$ と分解するとき

$$M^{-1} \cong (LU)^{-1} \tag{4}$$

となるように構成することである．ただし，L，U はそれぞれ下三角行列と上三角行列である．この前処理行列の計算では計算量を少なくするために，係数行列 A の非零要素の位置から左右それぞれ k 個目までの fill-in のみを許している．一般に，このような ILU 分解のことを ILU(k) 法と表記する．この前処理行列を反復法に適用するときは，$\boldsymbol{w} = (LU)^{-1}\boldsymbol{v}$ の計算が必要になる．通常，この $\boldsymbol{w}$ の計算は

$$L\tilde{\boldsymbol{v}} = \boldsymbol{v} \tag{5}$$

$$U\boldsymbol{w} = \tilde{\boldsymbol{v}} \tag{6}$$

とおき，それぞれの方程式を前進代入と後退代入で解くことになる．しかし，これらの処理には列ベクトルの間に依存関係があるので，並列処理を行うにはひと工夫必要である．

ここで，l 番目の PE が担当する L，U とベクトルを行ごとに均等に m 行ずつブロック分割すると，l 番目の PE は式 (5) を次式のように変形して計算することができる．

$$\tilde{v}_i = \frac{1}{L_{i,i}}\left(v_i - \sum_{j=1}^{i-1} L_{i,j}\tilde{v}_j\right), \quad i = lm+1, \ldots, (l+1)m \tag{7}$$

ただし，この式は通常 $\tilde{v}_j$（$j = 0, \ldots, lm$）を別の PE から受信しないと計算できない．しかし，各 PE が自分の担当する $\boldsymbol{v}_j$ をすべて計算せずに，途中で通信を行うことで部分的な並列計算が可能となる．たとえば，最初に 0 番目の PE のみが $\tilde{v}_1$，$\tilde{v}_2$ のみを計算して，これらを別の PE に通信すれば，次の段階で 1 番目以降の PE はこれらを使って式 (7) の計算ができ，同時に PE0 は $\tilde{v}_3$，$\tilde{v}_4$ の計算をすることができる．なお，式 (6) も同様にして部分的な並列計算をすることができる．本稿では，式 (5) と式 (6) において，1 回に通信するベクトル $\tilde{\boldsymbol{v}}$，$\boldsymbol{w}$ の要素数を $\tilde{m}$ とおき，これをブロック数と呼ぶことにする．このように $\tilde{m}$ を定義すると，たとえば，$m = 16$ でベクトル要素の通信回数を 2 回にしたい場合には，$\tilde{m} = 8$ とすればよいことになる．なお，この行列の前処理の実装についての詳細は，森屋ら[10] を参照してほしい．

2.2 MR 法による近似逆行列の計算

MR 法による近似逆行列の計算法は，Grote ら[5] と Huckel[6] によって提案されたものであり，反復解法を利用して前処理行列を計算する代表的な算法の 1 つである．この手法は前処理行列 M^{-1} の各列をまったく独立に計算することができるので，並列化が非常に容易である．ここで，$\boldsymbol{e}_j$，$\boldsymbol{m}_j$ をそれぞれ単位行列 I，前処理行列 M^{-1} の j 番目の列ベクトルであるとする．このとき，MR 法は最急降下法を用いて

$$\boldsymbol{m}_j^{(k)} = \boldsymbol{m}_j^{(k-1)} + \alpha_j^{(k)}\boldsymbol{r}_j^{(k)}, \quad j = 1,\ 2,\ \ldots,\ n \tag{8}$$

となるような漸化式によって実装を行う．ただし，$\boldsymbol{m}_j^{(k)}$ は最急降下法の k 回目の反復における $\boldsymbol{m}_j$ を表し，かつ $\boldsymbol{r}_j^{(k)} = \boldsymbol{e}_j - A\boldsymbol{m}_j^{(k)}$ である．$\alpha_j^{(k)}$ は最急降下法の各ステップ k において，最小 2 乗問題：

$$\min_{m_j^{(k)}} \|\boldsymbol{e}_j - A\boldsymbol{m}_j^{(k)}\|_2^2, \quad j = 1,\ 2, \ldots, n \tag{9}$$

を解くことに帰着できる．n 個の漸化式 (8) が各 PE に均等に割り当てられて他の PE とはまったく独立に計算できるので，この手法は並列化に適している算法といえる．また，この手法には計算量を抑えるための工夫がいくつか提案されている[6),7),9)]．$\boldsymbol{m}_j$ の非零要

素に関しては，その絶対値がある閾値 tol より小さい場合は切り捨てられるし，MR 法の反復もある一定の回数 imax で打ち切るように設定することになる．

2.3 Sherman-Morrison 法による近似逆行列の計算

ある正則行列 B と 0 でないベクトル $\boldsymbol{p}$，$\boldsymbol{q}$ を用いて正則行列 A を

$$A = B + \boldsymbol{p}\boldsymbol{q}^T \tag{10}$$

のように分割することを考える．このとき A の逆行列は

$$A^{-1} = B^{-1} - r^{-1}B^{-1}\boldsymbol{p}\boldsymbol{q}^T B^{-1} \tag{11}$$

のように表すことができる．ただし，$r = 1 + \boldsymbol{q}^T B^{-1}\boldsymbol{p} \neq 0$ である．式 (11) のことを Sherman-Morrison の公式[12] という．これ以降，Sherman-Morrison の公式 (11) を用いた行列 A の近似逆行列の計算法について述べる．ここで，$A_0,\ A_1, \ldots, A_n$ をいずれも正則行列とするとき，次の漸化式を定義する．

$$A_k = A_{k-1} + \boldsymbol{p}_k\boldsymbol{q}_k^T, \quad k = 1,\ 2,\ \ldots, n \tag{12}$$

式 (12) に対して Sherman-Morrison の公式 (11) を適用すると

$$A_k^{-1} = A_{k-1}^{-1} - r_k^{-1}A_{k-1}^{-1}\boldsymbol{p}_k\boldsymbol{q}_k^T A_{k-1}^{-1} \tag{13}$$

を導くことができる．ただし，$r_k = 1 + \boldsymbol{q}_k^T A_{k-1}^{-1}\boldsymbol{p}_k \neq 0$ である．$A^{-1} = A_n^{-1}$ として，A_n^{-1} を式 (13) を用いて展開すると次式を得ることができる．

$$A_0^{-1} - A^{-1} = \Psi\Omega^{-1}\Xi^T \tag{14}$$

ただし，式 (14) の右辺の行列はそれぞれ

$$\Psi = \left[A_0^{-1}\boldsymbol{p}_1,\ A_1^{-1}\boldsymbol{p}_2, \ldots, A_{n-1}^{-1}\boldsymbol{p}_n\right],$$
$$\Omega = \mathrm{diag}\left[r_1,\ r_2, \ldots, r_n\right],$$
$$\Xi = \left[\boldsymbol{q}_1^T A_0^{-1},\ \boldsymbol{q}_2^T A_1^{-1}, \ldots, \boldsymbol{q}_n^T A_{n-1}^{-1}\right]$$

である．また，対角行列 Ω の対角成分 r_k は，次のようになる．

$$r_k = 1 + \boldsymbol{q}_k^T A_{k-1}^{-1}\boldsymbol{p}_k \tag{15}$$

さらに，

$$\boldsymbol{u}_k := \boldsymbol{p}_k - \sum_{i=1}^{k-1} \frac{(\boldsymbol{v}_i^T, A_0^{-1}\boldsymbol{p}_k)}{r_i}\boldsymbol{u}_i \tag{16}$$

$$\boldsymbol{v}_k := \boldsymbol{q}_k - \sum_{i=1}^{k-1} \frac{(\boldsymbol{q}_k^T, A_0^{-1}\boldsymbol{u}_i)}{r_i}\boldsymbol{v}_i \tag{17}$$

と定義すると，$A_{k-1}^{-1}\boldsymbol{p}_k = A_0^{-1}\boldsymbol{u}_k$，$\boldsymbol{q}_k^T A_{k-1}^{-1} = \boldsymbol{v}_k^T A_0^{-1}$ が成り立つので，式 (14) と式 (15) はそれぞれ

$$A_0^{-1} - A^{-1} = A_0^{-1}U\Omega^{-1}V^T A_0^T \tag{18}$$

$$r_k = 1 + \boldsymbol{q}_k^T A_0^{-1}\boldsymbol{u}_k \tag{19}$$

と書き直すことができる[12]．ただし，行列 U と V は，それぞれ次のように定義する．

$$U = [\boldsymbol{u}_1,\ \boldsymbol{u}_2,\ \ldots,\ \boldsymbol{u}_n],$$
$$V = [\boldsymbol{v}_1,\ \boldsymbol{v}_2,\ \ldots,\ \boldsymbol{v}_n]$$

次に，初期値として A_0，$\boldsymbol{p}_k$，$\boldsymbol{q}_k$ を設定することを考える．Bru ら[12] によれば，$A_0^{-1} - A^{-1}$ が A の近似逆行列 M^{-1} となる最も簡単なとり方は

$$A_0 = sI,\ \boldsymbol{p}_k = \boldsymbol{e}_k,\ \boldsymbol{q}_k = (\boldsymbol{a}^k - s\boldsymbol{e}_k)^T \tag{20}$$

である．ただし，$\boldsymbol{a}^k$ は A の k 番目の列ベクトル，s は非負の実数値である．ここで，式 (20) のように値を設定し，式 (18) を M^{-1} とおくと，M^{-1} は A^{-1} と対角成分のみが異なることになる．式 (18) で表される前処理行列 M^{-1} と式 (20) による A_0 の設定から，A と M^{-1} の乗算は，次式のようになる．

$$AM^{-1} = s^{-1}A - I \tag{21}$$

ここで，AM^{-1}，A の固有値をそれぞれ $\lambda(AM^{-1})$，$\lambda(A)$ とすると，式 (21) から次式が成立する．

$$\lambda(AM^{-1}) = \frac{\lambda(A)}{s} - 1 \tag{22}$$

さらに，$\rho(A)$ を A のスペクトル半径とすると，式 (22) より $\rho(A) < s$ なら AM^{-1} の最大固有値の実部は負になるので，AM^{-1} の他のすべての固有値の実部もまた負になり複素平面の左半分に分布することになる．このことにより，桁落ちのしやすい原点近傍に存在する AM^{-1} の固有値は，すべて複素平面の左半分に移ることになり，これらの固有値の絶対値は 0 から遠ざかることになる．よって，これは非定常反復法における残差ノルムの収束に良い影響をもたらすことになる．Bru ら[12] は $\rho(A) < s$ を満たすとり方の 1 つとして，$s = 1.5 \parallel A \parallel_\infty$ と選択することを提案している．A_0，$\boldsymbol{p}_k$，$\boldsymbol{q}_k$ を式 (20) のようにとると，式 (16)，(17)，(19) はそれぞれ

$$\boldsymbol{u}_k = \boldsymbol{p}_k - \sum_{i=1}^{k-1} \frac{(v_i)_k}{sr_i}\boldsymbol{u}_i \tag{23}$$

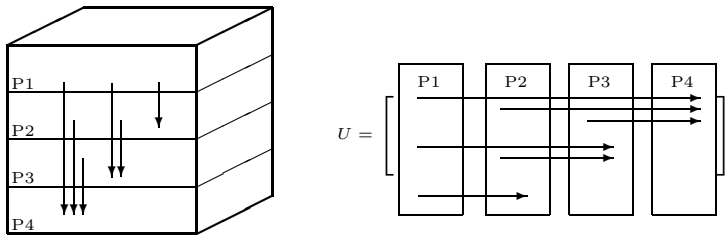
$$\boldsymbol{v}_k = \boldsymbol{q}_k - \sum_{i=1}^{k-1} \frac{(\boldsymbol{q}_k, \boldsymbol{u}_i)}{sr_i}\boldsymbol{v}_i \tag{24}$$

$$r_k = 1 + (\boldsymbol{v}_k)_k/s \tag{25}$$

となり，式 (18) から前処理行列 M^{-1} は

$$M^{-1} = s^{-1}I - A^{-1} = s^{-2}U\Omega^{-1}V^T \tag{26}$$

となる．ただし，$(v_i)_k$ はベクトル $\boldsymbol{v}_i$ の k 番目の要素である．式 (25) による r_k の計算は，$\boldsymbol{u}_k$ と $\boldsymbol{v}_k$ の計算が終了してから行うことになる．したがって，Sherman-Morrison 法に基づいて近似逆行列を計算するには，式 (23) と式 (24) より $\boldsymbol{u}_k$ と $\boldsymbol{v}_k$ を計算してから式 (25) によって r_k を求め，式 (26) の右辺のように行列分解を行えばよい．式 (26) は，真ん中の式に着目すると，M^{-1} と A^{-1} が理論上は対角成分のみが異なることを意味している．ただし，実際には計



(a) 領域の場合 (b) 行列の場合

図 2 1D 分割の様子
Fig. 2 The aspect of 1D partitioning.

Sherman-Morrison formula

for $k = 1$ **to** n **do**
  $\boldsymbol{p}_k = \boldsymbol{e}_k$, $\boldsymbol{q}_k = (\boldsymbol{a}^k - s\boldsymbol{e}_k)^T$
  $\boldsymbol{u}_k = \boldsymbol{p}_k$, $\boldsymbol{v}_k = \boldsymbol{q}_k$
  for $i = 1$ **to** $k-1$ **do**
    $\boldsymbol{u}_k = \boldsymbol{u}_k - \{(v_i)_k \ / \ (sr_i)\}\boldsymbol{u}_i$
    $\boldsymbol{v}_k = \boldsymbol{v}_k - \{(\boldsymbol{q}_k,\ \boldsymbol{u}_i) \ / \ (sr_i)\}\boldsymbol{v}_i$
  endfor
  for $i = 1$ **to** n **do**
    if $|(u_k)_i| <$ tolU **dropoff** $|(u_k)_i|$
    if $|(v_k)_i| <$ tolV **dropoff** $|(v_k)_i|$
  endfor
  $r_k = 1 + (v_k)_k/s$
endfor

図 1 Sherman-Morrison 法の実装
Fig. 1 The implementation of the Sherman-Morrison formula.

算量を少なくするために U と V の非零要素すべてを求めることはせず，適当な閾値を設けてそれよりも絶対値の小さい非零要素の切り捨てを行う．これらをまとめると近似逆行列を計算する Sherman-Morrison 法の実装は，**図 1** のようになる．図 1 の 9，10 行目が非零要素の切り捨ての処理である．つまり，U，V の k 列目のベクトルにおける i 番目の要素 $(u_k)_i$，$(v_k)_i$ の絶対値がそれぞれ閾値 tolU，tolV よりも小さいとき，これらの要素は切り捨てられることになる．Sherman-Morrison 法の実装で問題となるのは，式 (23) と式 (24) において，$\boldsymbol{u}_k$ と $\boldsymbol{v}_k$ が $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$ および $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ を必要とするので，この並列化はそれほど簡単ではない．

3. Sherman-Morrison 法の並列化とその実装

3.1 行列分解の問題点

近似逆行列を計算する Sherman-Morrison 法の実装では，式 (23) と式 (24) による $\boldsymbol{u}_k$ と $\boldsymbol{v}_k$ の計算に $\boldsymbol{u}_1$，$\boldsymbol{u}_2, \ldots, \boldsymbol{u}_{k-1}$，$\boldsymbol{v}_1$，$\boldsymbol{v}_2, \ldots, \boldsymbol{v}_{k-1}$ および r_1，$r_2, \ldots, r_{k-1}$ が必要となる．したがって，これらのベクトル列が完全に独立して計算できるわけではないので，並列化することはそれほど簡単ではない．本稿では，Naik[4] の提案した実装方法を用いて，すなわち列ベクトルの計算を他の PE との通信を交互に行うことで，式 (23) と式 (24) の実装を部分的に並列化し，行列 U，V，Ω の計算の高速化を実現する．

3.2 列ベクトルの割当て

Naik[4] の論文で扱われている BT Bench Mark では，問題の対象を 3 次元偏微分方程式の境界値問題に焦点を当て，領域を 1D から 3D までの 3 通りの方法で分割することを考えている．これらの分割はそれぞれ，1 方向，2 方向，3 方向の分割となる．彼の手法では，これら 3 通りの方法により分割した領域の各格子点上における解を LU 分解によって求めている．本稿では 1D 分割法を用いて，Sherman-Morrison 法の行列分解で出現する行列を列方向に分割させることにする．ここで，1D 分割とは**図 2** の (a) のように領域をある 1 方向のみに分割する方法である．この図における矢印で示すように，ある領域の格子点の情報は，自分より上の領域に属する格子点の情報を受信することではじめて求めることができる．

Sherman-Morrison 法を用いて近似逆行列を求める場合，行列 U，V，Ω を計算することが必要となる．本稿では，図 2 の (b) で示すように，行列を横へ 1 方向のみに分割することを行い，各 PE に均等に分割することを考える．このように行列の分割を行うと，矢印で示すように右の列ベクトルは自分よりも左の列ベクトルの情報に依存して計算することになる．ここで，行列の次元数と使用する PE 台数がそれぞれ n，d である場合，行列 U の列ベクトルの分割は

$$U = \{\underbrace{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m}_{\text{PE0}}, \underbrace{\boldsymbol{u}_{m+1}, \ldots, \boldsymbol{u}_{2m}}_{\text{PE1}},$$

$$\ldots\ldots, \underbrace{\boldsymbol{u}_{n-m+1}, \ldots, \boldsymbol{u}_n}_{\mathrm{PE}(d-1)}\}$$

となるように行う．ただし，m は各 PE の担当する列ベクトルの個数であり，$m = n/d$ かつ n は d で割り切れるものとする．したがって，l 番目の PE は $lm+1$ から $(l+1)m$ 番目までの列ベクトルを担当することになる．特に，行列の列数が PE に均等に割り当てられない場合，$l = \mathrm{mod}(n,\ d)$ とすると 0 から $l-1$ 番目までの PE は $\mathrm{integer}(n/d)+1$ 列だけを担当して，l 番目以降の PE は $\mathrm{interger}(n/d)$ 列だけを担当することになる．ただし，$\mathrm{integer}(n/d)$ は n/d の整数部分を表すものとする．行列 V と Ω の列ベクトルも同様にして割り当てられるが，Ω は対角行列であるため対角要素 r_i のみが各 PE に割り当てられる．

3.3 近似逆行列計算の並列化

式 (23) と式 (24) より，ベクトル $\boldsymbol{u}_k$，$\boldsymbol{v}_k$ の計算は $\boldsymbol{u}_1$，$\boldsymbol{u}_2, \ldots, \boldsymbol{u}_{k-1}$，$\boldsymbol{v}_1$，$\boldsymbol{v}_2, \ldots, \boldsymbol{v}_{k-1}$ および r_1，$r_2, \ldots, r_{k-1}$ に依存していることになる．このような依存関係があるので，添字の順番どおりに $\boldsymbol{u}_k$ と $\boldsymbol{v}_k$ を計算すると l 番目の PE は，0 番目から $l-1$ 番目までの PE の計算が終わらないと計算を開始することができない．しかし，各 PE の担当する列ベクトルの計算と他 PE との通信を交互に行うことで，式 (23) と式 (24) を部分的に並列化することは可能である．ここで，U，V，Ω の列ベクトルが 3.2 節で述べたように分割されているとする．このとき式 (23) と式 (24) はローカルに計算できる部分と通信によって計算できる部分の 2 つに分けることができる．今，l（$l = 1, 2, \ldots, d-1$）番目の PE が担当する U，V，Ω の列ベクトルは，$lm+1$ から $(l+1)m$ 番目までの m 個であるとする．ただし，ここでは簡単のため，$m = n/d$ であり，n は d で割り切れるものとして考える．今，式 (23) を次式のように分割する．

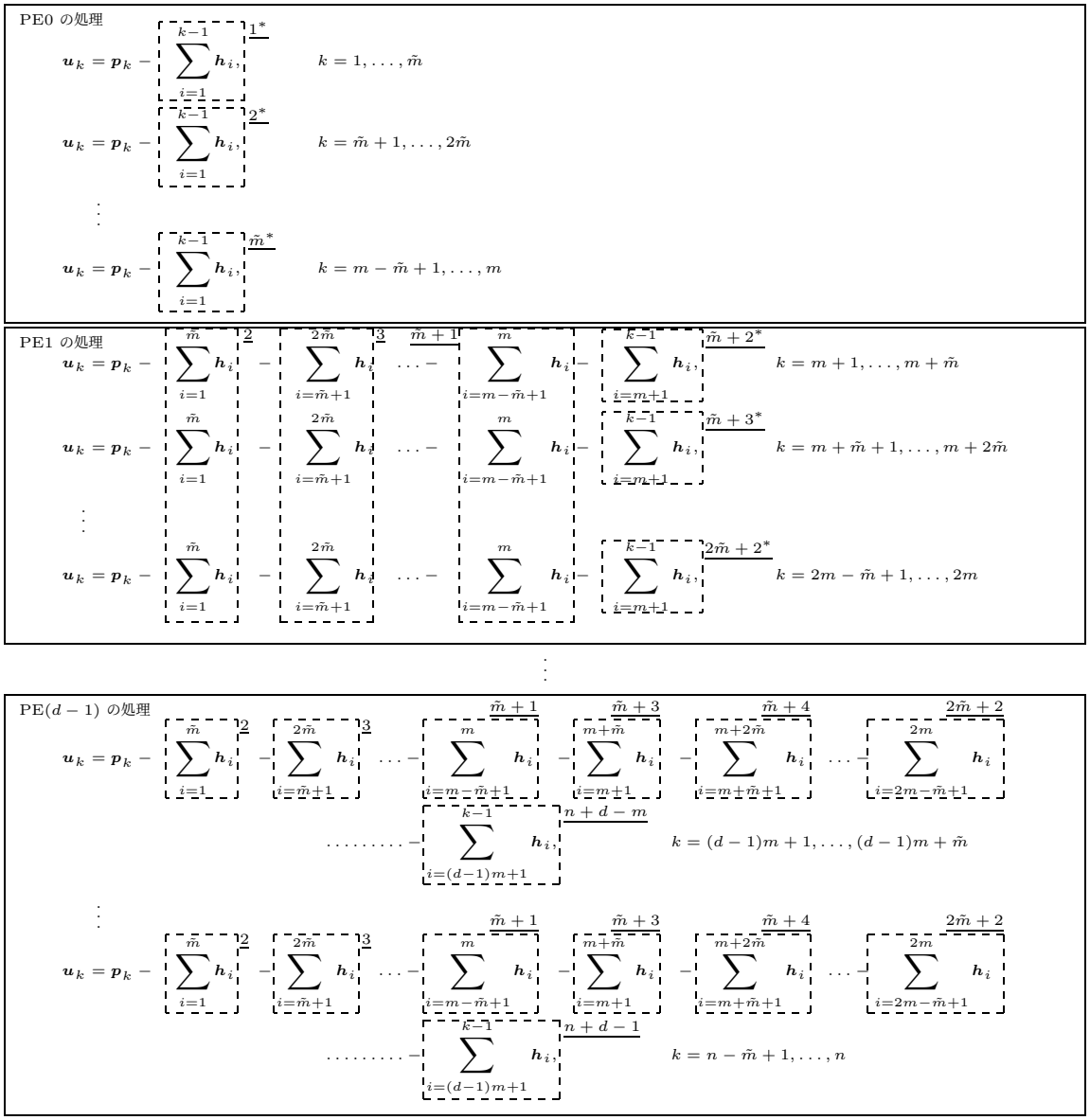
$$\boldsymbol{u}_k = \boldsymbol{p}_k - \sum_{i=1}^{lm} \frac{(v_i)_k}{sr_i}\boldsymbol{u}_i - \sum_{i=lm+1}^{k-1} \frac{(v_i)_k}{sr_i}\boldsymbol{u}_i \tag{27}$$

この式 (27) において，他の PE からのデータ受信を完了することで計算できるようになる部分は右辺第 2 項の部分であり，この計算は他の PE からデータを受信することによってはじめて計算可能となる．それに対して，ローカルに計算できる部分は右辺第 3 項の部分であり，これに対応するデータは自分の PE が担当しているので，他の PE からデータを受信しなくても計算することができる．ここで問題となるのは，右辺第 2 項で用いられているベクトルを別の PE から受信していないと第 3 項の計算を開始できないことである．そこで，ある PE が自分の担当する U，V，Ω の列ベクトルをいくつか計算して，すべての計算が終わらないうちに途中で列ベクトルを別 PE に送信することを考える．このとき，データを送信した PE には，まだ計算すべき列ベクトルが残っているので，以降はこの PE とこれからデータを受信した PE とで並列に動作することができる．つまり，計算とデータの通信を交互に行うことで部分的な並列動作が可能となる．たとえば，各々の PE が U，V，Ω の列ベクトルをそれぞれ 8 個ずつ担当している場合を考える．この場合，もし PE0 が 8 個すべて計算してから通信をすると通信回数は 1 回で済むが，PE0 の計算が終了しないと 1 番目以降の PE は動作できないことになる．それに対して，PE0 が 2 個だけ計算してデータを残りの PE に送信すると，次のステップでは，1 番目以降の PE は受信したデータを使って自分の担当する列ベクトルを更新でき，PE0 は次の 2 個の列ベクトルを計算できる．したがって，最初のステップでは並列に動作していないが，それ以降のステップでは並列計算が行われることになる．このことは，式 (24) にもあてはまり，同様にして並列化することができる．

ここで，各 PE の担当する U，V，Ω の列ベクトルの組数を m，1 回に通信する列ベクトルの組数を $\tilde{m}$，使用する PE 台数を d とおくことにする．これらのことを前提とし，式 (27) を次式のように分割することを考える．

$$\begin{aligned}
\boldsymbol{u}_k = \boldsymbol{p}_k &- \underbrace{\sum_{i=1}^{\tilde{m}} \boldsymbol{h}_i \ldots - \sum_{i=m-\tilde{m}+1}^{m} \boldsymbol{h}_i}_{\text{PE0 から通信}} \\
&\underbrace{- \sum_{i=m+1}^{m+\tilde{m}} \boldsymbol{h}_i \ldots - \sum_{i=2m-\tilde{m}+1}^{2m} \boldsymbol{h}_i}_{\text{PE1 から通信}} \\
&\vdots \\
&\underbrace{- \sum_{i=(l-1)m+1}^{(l-1)m+\tilde{m}} \boldsymbol{h}_i \ldots - \sum_{i=lm-\tilde{m}+1}^{lm} \boldsymbol{h}_i}_{\text{PE}l-1 \text{ から通信}} \\
&\underbrace{- \sum_{i=lm+1}^{k-1} \boldsymbol{h}_i}_{\text{通信なしで計算可能}}
\end{aligned} \tag{28}$$

ただし，$\boldsymbol{h}_i = \{(v_i)_k \ / \ sr_i\}\boldsymbol{u}_i$ である．式 (28) は式



注：同じ番号の項を同時に計算する．∗ がついている項は，計算後に $\boldsymbol{u}_k$ を他 PE に送信する．

図 3 $\boldsymbol{u}_k$ を並列計算するしくみ

Fig. 3 The mechanism of parallel computating for $\boldsymbol{u}_k$.

(27) の 2 項目をより細かく分割したものであり，各 PE から列ベクトル $\boldsymbol{u}_i$，$\boldsymbol{v}_i$，r_i を 1 回につき $\tilde{m}$ 組受信するたびに $\sum$ の項を 1 つ計算することができる．たとえば，PE0 から $i = 1, \ldots, \tilde{m}$ に対応した $\boldsymbol{u}_i$，$\boldsymbol{v}_i$，r_i を受信すると，式 (28) の右辺における $\sum_{i=1}^{\tilde{m}} \boldsymbol{h}_i$ の項を計算することが可能となる．このことをふまえて式 (28) を並列化すると**図 3** のようになる．この図の点線で囲まれた部分において，アンダーライン付きの番号が同じである項が並列に処理できる．“∗” は，該当する項がローカルに計算できることを表し，この項が計算し終わった後に対応する $\boldsymbol{u}_k$ を他 PE に送信する．たとえば，PE0 の $\underline{1^*}$ の指している項を計算してから，$\boldsymbol{u}_k$（$k = 1, \ldots, \tilde{m}$）が他 PE に送信できる．その後，PE1 から PE $(d-1)$ の $\underline{2}$ の指す項と PE0 の $\underline{2^*}$ の指す項が同時に処理できることになる．このような処理により，すべての PE がすべてのステップで並列に動作できるとは限らないが，いくつかの PE が部分的には並列に動作することは可能である．$\boldsymbol{v}_k$ の計算も同様にして並列化することができる．図 3 にならって $\boldsymbol{u}_k$ の計算を行うことにし，たとえば，$d = 4$,

$\tilde{m}=2$，$m=4$ である場合を考える．この場合，PE は 4 台使用することになり，$\boldsymbol{u}_k$，$\boldsymbol{v}_k$，r_k に関しては，PE0 が $k=1,\ldots,4$，PE1 が $k=5,\ldots,8$，PE2 が $k=9,\ldots,12$，PE3 が $k=13,\ldots,16$ に対応するベクトルをそれぞれ担当することになる．この例では，次のようなステップに分けて，式 (28) における $\boldsymbol{u}_k$ の並列計算を実行することができる．ただし，$\boldsymbol{v}_k$ と r_k についても同様な過程によって計算するものとする．

（ステップ 1） PE0 が $\boldsymbol{u}_k$，$\boldsymbol{v}_k$，r_k（$k=1,2$）を計算し，これらを PE1 から PE3 まで送信する．このステップでは 0 番目の PE 以外は稼働していない．

（ステップ 2） PE0 は $\boldsymbol{u}_k$，$\boldsymbol{v}_k$，r_k（$k=3,4$）を計算し，これらを PE1 から PE3 まで送信する．同時に，PE1 から PE3 までは，（ステップ 1）で受信したデータを用いて，自分の担当する $\boldsymbol{u}_k$ を更新する．ここまでで，PE0 の担当する計算は終了する．

（ステップ 3） PE1 から PE3 が，（ステップ 2）で受信したデータを用いて自分の担当する $\boldsymbol{u}_k$ を更新する．さらに，PE1 は $\boldsymbol{u}_k$，$\boldsymbol{v}_k$，r_k（$k=5,6$）のローカルで計算できる部分を計算して，これらを PE2 と PE3 へ送信する．

（ステップ 4） PE1 は $\boldsymbol{u}_k$，$\boldsymbol{v}_k$，r_k（$k=7,8$）のローカルで計算できる部分を計算し，これらを PE2 と PE3 に送信する．同時に PE2 と PE3 は，（ステップ 3）で受信したデータを用いて，自分の担当する $\boldsymbol{u}_k$，$\boldsymbol{v}_k$ を更新する．ここまでで PE1 の担当する計算は終了する．

（ステップ 5 以降） これ以降は PE2 と PE3 のみが起動しているが，前ステップまでと同様にして計算と通信を行うことができる．

図 3 では 1PE の担当する G_k の個数 m が $\tilde{m}$ で割り切れる場合を想定しているが，$m/\tilde{m}$ が通信回数に相当する．通信回数が多くなれば 1 ステップ目における PE0 の計算が早く終わるので，全 PE の動作する 2 ステップ目の開始時間が早くなるが通信回数も増大することになる．したがって，1 ステップごとに計算する列ベクトルの個数である $\tilde{m}$ を小さくしても必ずしも並列処理効率が向上するとは限らない．つまり，$\tilde{m}$ を増加させることが必ずしも PE の台数効果を向上させることに結び付くとは限らない．2.1 節の ILU 分解の前処理と同様，本稿では $\tilde{m}$ を以降ブロック数と呼ぶことにする．このような並列化を取り入れて Sherman-Morrison 法による近似逆行列の計算を行うと，l 番目の PE の処理は**図 4** のようになる．ただし，この算法はスレッド並列でも実装が可能であるので，この図のタイトルには PE とプロセス両方の表現を用いている．1 行目から 3 行目までは，各 PE の担当している列ベクトル $\boldsymbol{u}_k$，$\boldsymbol{v}_k$ の初期値を設定している．4 行目から 11 行目までは，各 PE が別の PE から計算に必要となる $\boldsymbol{u}_j$，$\boldsymbol{v}_j$，r_j を受信してから，各 PE の担当する $\boldsymbol{u}_k$，$\boldsymbol{v}_k$ を更新している．これらの計算は，式 (27) における右辺第 2 項に対応している．これらの部分の計算では，0 番目を除いた各 PE は受信が完了しないと 7 行目の過程で処理が待ち状態になる．残りの 12 行目から 24 行目までも各 PE の担当する $\boldsymbol{u}_k$，$\boldsymbol{v}_k$ を更新しているが，この部分の計算では PE 間通信なしにローカルに計算することになる．これらの部分の計算は，式 (27) における右辺第 3 項目に相当する．また，途中の 21 行目から 23 行目までは，$\boldsymbol{u}_k$，$\boldsymbol{v}_k$，r_k がそれぞれ $\tilde{m}$ 個だけ計算が終了すると，これらのデータを他の PE に送信している．また，m が $\tilde{m}$ で割り切れない場合，たとえば $m=501$ で $\tilde{m}=100$ なら通信は全部で 6 回必要であるが，最初の 5 回の通信で G_k をそれぞれ 100 個ずつ他 PE に送信し，最後の 6 回目の通信で余りの 1 個だけ送信を行うことになる．

4. 数値実験

3 台のノードをネットワークで接続し PC クラスタを構築した．各々のノードおよび数値実験の環境は以下のようになる．

ノード：IBM Xseries 346（× 3）
PE：Pentium Xeon 3.6 GHz（× 2）
メモリ容量：1 ノードにつき 1 GB
OS：Linux Fedora Core 4
コンパイラ：gcc 4.0.0（オプションいっさいなし）
通信ライブラリ：LAM/MPI 7.1.1 [14]

本章では 3 題の数値例を取り上げ，以下の 4 つの数値実験を行った．

(1) Sherman-Morrison 法による前処理の計算で通信回数を変化させて，最適なブロック数 $\tilde{m}$ を決定する．同様にして ILU 分解における最適な $\tilde{m}$ の値を決定する．
(2) 行列 U，V の非零要素数を計測する．
(3) PE 台数を 1，2，4，6 台と変更して，Sherman-Morrison 法による前処理行列を計算したときの並列度を計測する．
(4) Sherman-Morrison 法で計算した近似逆行列を用いたときの残差ノルムの収束の評価を行う．

実験項目 (3) を除いて，PE を 6 台用いて実験を

The parallel implementation

for $k = lm+1$ **to** $(l+1)m$ **do**
  $\boldsymbol{p}_k = \boldsymbol{e}_k, \quad \boldsymbol{q}_k = (\boldsymbol{a}^k - s\boldsymbol{e}_k)^T, \quad \boldsymbol{u}_k = \boldsymbol{p}_k, \quad \boldsymbol{v}_k = \boldsymbol{q}_k$
endfor
for $i = 0$ **to** $l-1$ **do**
  for $k = mi+1$ **to** $m(i+1)$ **do**
    if $\mathrm{mod}(k,\ \tilde{m}) = 1$ **then**
      Receive $\boldsymbol{u}_j,\ \boldsymbol{v}_j,\ r_j,\ (j = k, \ldots, k+\tilde{m}-1)$ **from** PE_i
    endif
    $\boldsymbol{u}_k = \boldsymbol{u}_k - \{(v_i)_k \ / \ (sr_i)\}\boldsymbol{u}_i, \quad \boldsymbol{v}_k = \boldsymbol{v}_k - \{(\boldsymbol{q}_k,\ \boldsymbol{u}_i) \ / \ (sr_i)\}\boldsymbol{v}_i$
  endfor
endfor
for $k = lm+1$ **to** $(l+1)m$ **do**
  for $i = lm+1$ **to** $k-1$ **do**
    $\boldsymbol{u}_k = \boldsymbol{u}_k - \{(v_i)_k \ / \ (sr_i)\}\boldsymbol{u}_i, \quad \boldsymbol{v}_k = \boldsymbol{v}_k - \{(\boldsymbol{q}_k,\ \boldsymbol{u}_i) \ / \ (sr_i)\}\boldsymbol{v}_i$
  endfor
  for $i = 1$ **to** $i = n$ **do**
    if $|(u_k)_i| < \mathrm{tolU}$ **dropoff** $|(u_k)_i|$
    if $|(v_k)_i| < \mathrm{tolV}$ **dropoff** $|(v_k)_i|$
  endfor
  $r_k = 1 + (v_k)_k/s$
  if $\mathrm{mod}(k,\ \tilde{m}) = 0$ **then**
    Send $\boldsymbol{u}_i,\ \boldsymbol{v}_i,\ r_i (i = k-\tilde{m}+1, \ldots, k)$ to $\mathrm{PE}_{l+1}, \ldots, \mathrm{PE}_{d-1}$
  endif
endfor

図 4 Sherman-Morrison 法の並列実装（l 番目の PE もしくはプロセスの場合）
Fig. 4 The parallel implementation of the Sherman-Morrison formula (in case of l-th PE or process).

行った．また，残差ノルムの収束の評価は，Sherman-Morrison 法による前処理行列と MR 法もしくは ILU 分解による前処理行列を GMRES(m) 法[1] に適用して，それぞれの前処理行列の性能を比較した．数値例として取り扱う問題の係数行列は，いずれも実数かつ非対称とした．ここで，連立 1 次方程式の初期近似解を零ベクトルとし，さらに GMRES(m) 法の反復終了条件を次式のように設定する．

$$\|\boldsymbol{r}_i\|_2 \ / \ \|\boldsymbol{b}\|_2 < \ 1.0 \times 10^{-12} \qquad (29)$$

ただし，$\boldsymbol{r}_i$ は GMRES(m) 法の i 回目の反復における残差ベクトルである．前処理行列の計算に関する各種の値として，tolU，tolV には 0.1 もしくは 0.01 の値を使用した．また，スカラ s には，Bru ら[12] の設定した $s = 1.5\,\|A\|_\infty$ のほかに，$1.5\,\|A\|_2 \times 10^1$ から $1.5\,\|A\|_2 \times 10^3$ までの値を利用し，s を変化させたときの速度向上や残差ノルムの収束に及ぼす影響を計測した．ここで，便宜上 $\tilde{s} = s \ / \ (1.5\,\|A\|_\infty)$ を定義し，数値実験で s を変化させるときは $\tilde{s}$ を用いて行うものとする．MR 法には，非零要素の切り捨てを行う閾値を tol とし，0.1 もしくは 0.01 の値を利用した．MR 法の反復の初期値となる行列には単位行列 I を選択し，この前処理計算法の反復回数を imax とし 1 から 3 の値を利用した．ILU 分解では，fill-in をまったく許さない ILU(0) と，非零要素の両サイドにそれぞれ 1 ないしは 2 個の要素の fill-in を許す ILU(1), ILU(2) の 3 通りの方法により前処理行列を計算した．

前処理行列の計算の実装に関していえば，Sherman-Morrison 法は 3.3 節で記述した手法で実装を行った．また，MR 法による前処理行列の計算では Sherman-Morrison 法の前処理行列の計算と同様にして，前処理行列の列を各 PE にブロック分割して均等に割り当てた．それに対して，ILU 分解による前処理行列では，

表 1 数値例 1：Sherman-Morrison 法のブロック数と計算時間の関係
Table 1 Example 1: The relation between the number of blocks and the computation time in the Sherman-Morrison formula.

(a) tolU，tolV = 0.1

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	6,144	3,072	1,536	768	384	192	96	48
計算時間（秒）	84.0	63.0	59.0	58.0	57.0	57.0	57.0	56.0

(b) tolU，tolV = 0.01

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	6,144	3,072	1,536	768	384	192	96	48
計算時間（秒）	591.0	460.0	405.0	392.0	390.0	389.0	389.0	384.0

表 2 数値例 1：ILU(2) 法のブロック数と計算時間の関係
Table 2 Example 1: The relation between the number of blocks and the computation time in the ILU(2) method.

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	6,144	3,072	1,536	768	384	192	96	48
計算時間（秒）	0.42	0.41	0.41	0.42	0.42	0.44	0.46	0.52

表 3 数値例 1：Sherman-Morrison 法による前処理行列の非零要素数
Table 3 Example1: Nonzero entries of the Sherman-Morrison preconditioner.

tolU / tolV	$\tilde{s}$							
	10^0		10^1		10^2		10^3	
	U	V	U	V	U	V	U	V
0.1	138,004	155,308	138,004	943,205	138,004	3,171,837	138,004	7,027,265
0.01	1,051,329	13,164,320	1,051,329	35,531,595	1,051,329	66,748,783	1,051,329	103,157,063

行列 L と U の行をブロック分割して各 PE に均等に割り当てることにした．

［数値例 1］ 正方領域 $\Theta = [0,\ 1]^2$ における以下のような偏微分方程式の境界値問題を考える[8]．

$$-\frac{d}{dx}\left[\{\exp(-xy)\}u_x\right] - \frac{d}{dy}\left[\{\exp(xy)\}u_y\right] + 10.0(u_x + u_y) - 60.0u = f(x,y)$$

$$u(x,\ y)|_{\partial\Theta} = 1 + xy$$

この偏微分方程式を 5 点中心差分を用いて格子点数 192×192 で離散化を行い，36,864 次元の連立 1 次方程式を得た．ただし，$f(x,y)$ は厳密解が $u(x,y) = 1+xy$ となるように設定した．この方程式における係数行列 A は 5 本の非零対角成分を持ち，各対角成分の値はすべて異なっている．最初に，ブロック数 $\tilde{m}$ を変化させて Sherman-Morrison 法により近似逆行列を計算するのにかかる時間を計測した．**表 1** より閾値が tolU，tolV = 0.1 と tolU，tolV = 0.01 のいずれの場合も通信回数が 128 回のとき，つまり $\tilde{m} = 48$ のときに M^{-1} の計算時間は最小になった．同様にして，ILU 分解で式 (5) と式 (6) の計算にかかる時間を計測した．その結果を**表 2** に示すが，通信回数が 4 回，つまり $\tilde{m} = 1{,}536$ のとき計算時間は最小になった．したがって，この事実をもとに残差ノルムの収束評価を行う数値実験で Sherman-Morrison 法と ILU(2) 法のブロック数には，それぞれ $\tilde{m} = 48$，1,536 の値を利用した．

次に，s の値を変化させて Sherman-Morrison 法の行列分解から得られた行列 U，V の非零要素数を計測し，その結果を**表 3** に示した．V の非零要素は s に比例して増加する傾向にあった．特に，s が 3 桁違うと非零要素数は最大で約 10 倍の差がみられた．したがって，U，V に必要となるメモリの容量は s の大きさに比例することになる．また，s を変化させることが速度向上率に影響を及ぼすかを検証するために，$\tilde{s} = 10^0$，10^1 のときに Sherman-Morrison 法による M^{-1} の計算時間を PE 台数を 1 台から 6 台まで変化させて台数効果を計測した．ただし，各 PE 台数とも通信回数が 128 回となるようにブロック数の値を設定している． **表 4** による計測結果から，PE を 6 台使用すると台数効果は 4 台までのときに比べて低下しているものの，5.6 倍などといったように理想の 90% 程度の速度向上を得ることができた．また，$\tilde{s} = 10.0$ の場合は $\tilde{s} = 1.0$ のときに比べて，PE 数が 4 台と 6 台のときに速度は若干向上している．これは，V の非零要素数が増加したことで計算量が相対的に増えたため

表 5 数値例 1：計算時間と反復回数（time：計算時間（秒），iter：反復回数）
Table 5 Example 1: The computation time and iteration count (time: Computation time (s), iter: Iteration count).

前処理の種類	前処理行列の計算	非定常反復法					
		GMRES(20)		GMRES(30)		GMRES(40)	
	time	time	iter	time	iter	time	iter
なし	0.0	–	–	–	–	–	–
SM 法（tolU = tolV = 0.1, $\tilde{s} = 10^0$）	56.0	–	–	514.0	12,125	289.0	5,839
SM 法（tolU, tolV = 0.1, $\tilde{s} = 10^1$）	142.0	158.0	338	164.0	453	165.0	473
SM 法（tolU, tolV = 0.01, $\tilde{s} = 10^0$）	384.0	452.0	1,160	420.0	565	409.0	382
SM 法（tolU, tolV = 0.01, $\tilde{s} = 10^1$）	1,461.0	1,470.0	57	1,468.0	39	1,467.0	33
ILU(0) 法	0.21	–	–	–	–	171.0	3,066
ILU(1) 法	0.31	–	–	63.0	1,044	78.0	1,313
ILU(2) 法	0.41	–	–	38.0	625	41.0	588
MR 法（tol = 0.1, imax = 3）	173.0	–	–	–	–	–	–
MR 法（tol = 0.1, imax = 2）	100.0	–	–	–	–	–	–
MR 法（tol = 0.1, imax = 1）	45.0	–	–	–	–	–	–
MR 法（tol = 0.01, imax = 3）	189.0	–	–	–	–	–	–
MR 法（tol = 0.01, imax = 2）	103.0	–	–	–	–	–	–
MR 法（tol = 0.01, imax = 1）	45.0	–	–	–	–	214.0	6,021

–：2 時間以内に残差ノルムが収束しなかった場合
SM 法：Sherman-Morrison 法，　imax：MR 法の反復回数

表 4 数値例 1：Sherman-Morrison 法による前処理行列の計算の速度向上（tolU，tolV = 0.01）
Table 4 Example 1: Speedup of computation of the Sherman-Morrison preconditioner (tolU, tolV = 0.01).

PE 数	$\tilde{s}$	
	10^0	10^1
1	1.0	1.0
2	1.9	1.9
4	3.8	3.9
6	5.6	5.8

である．

次に，3 種類の前処理行列をそれぞれ GMRES(m) 法に適用し，この偏微分方程式を離散化して得られた連立 1 次方程式を解く実験を行った．**表 5** には GMRES(m) 法の残差ノルムが式 (29) の収束判定条件を満たすまでにかかった計算時間と反復回数を示している．また，この表における GMRES(m) 法の計算時間には，各前処理行列を計算するのにかかった時間も含まれており，2 列目に示されている値がその計算時間である．MR 法による前処理行列を適用した場合，残差ノルムを収束させることができたのは，tol = 0.1 かつ imax = 1 のときの GMRES(40) 法だけであった．この前処理行列の計算では，imax を増加させても前処理行列の近似度が上がらず，残差ノルムの収束を改善させるには至らなかった．ILU 分解による前処理行列を適用すると Sherman-Morrison 法による前処理行列を適用したときよりも残差ノルムの収束が良い場合もある．また，Sherman-Morrison 法よりも少ない計算で ILU 分解の前処理行列を求めることができる．しかし，fill-in をまったく許さなかったり GMRES(m) 法のリスタート周期が小さかったりすると収束せず，安定性に欠けることがあった．それに対して，Sherman-Morrison 法による前処理行列を適用した場合，GMRES(20) 法の 1 ケースを除いたすべての場合で残差ノルムは収束した．したがって，この数値例からは，ILU 分解の方が Sherman-Morrison 法より前処理行列として優れた性能を発揮することがある．しかし，後者の前処理行列の方が計算に時間がかかるものの，前者の前処理行列よりも多くの場合で安定して残差ノルムが収束している．したがって，残差ノルムの収束の安定性は，Sherman-Morrison 法で計算した前処理行列の場合が一番良いといえる．さらに，この数値例では表 4 のように PE 台数を変化させて，Sherman-Morrison 法による近似逆行列前処理を用いた GMRES(40) 法の計算時間を計測して，その台数効果を測定した．ただし，tolU，tolV = 0.01 かつ $\tilde{s} = 10^0$ の場合とする．その結果，台数効果は表 4 の $\tilde{s} = 10^0$ の場合とまったく同じ結果となった．これは，この GMRES(40) 法の計算時間 409 秒のうち，前処理行列の計算時間が 384 秒であり，前者にしめる後者の割合が 90%以上と高いために同じ結果になったと考えることができる．

最後に，2.3 節で述べた $\rho(A) < s$ の関係が成立するかを確認した．ただし，GMRES(m) 法の過程で 100 次元の Hessenberg 行列を計算し，この行列の固有値を係数行列 A の近似固有値とした．その結果，$\rho(A) = 1.9947$ かつ $s = 3.0006$ であったので，上記

表 6 数値例 2：Sherman-Morrison 法のブロック数と計算時間の関係
Table 6 Example 2: The relation between the number of blocks and the computation time in the Sherman-Morrison formula.

(a-1) ecl32, tolU, tolV = 0.1

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	8,666	4,333	2,167	1,084	542	271	136	68
計算時間（秒）	310.0	230.0	216.0	210.0	207.0	206.0	209.0	209.0

(a-2) ecl32, tolU, tolV = 0.01

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	8,666	4,333	2,167	1,084	542	271	136	68
計算時間（秒）	1,606.0	1,287.0	1,144.0	1,075.0	1,047.0	1,038.0	1,042.0	1,049.0

(b-1) af23560, tolU, tolV = 0.1

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	3,927	1,964	982	491	246	123	62	31
計算時間（秒）	1,835.0	1,465.0	1,284.0	1,185.0	1,175.0	1,223.0	1,228.0	1,234.0

表 7 数値例 2：ILU(2) 法のブロック数と計算時間の関係
Table 7 Example 2: The relation between the number of blocks and the computation time in the ILU(2) method.

(a) ecl32

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	8,666	4,333	2,167	1,084	542	271	136	68
計算時間（秒）	2.17	2.14	2.14	2.13	2.14	2.17	2.23	2.34

(b) af23560

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	3,927	1,964	982	491	246	123	62	31
計算時間（秒）	2.12	2.12	2.11	2.10	2.10	2.10	2.12	2.20

の不等式が成立することを確認した．

[数値例 2] Florida Sparse Matrix Collection [13] のデータベースにある行列を係数とする連立 1 次方程式を解く問題を考える．係数行列として扱ったものは，“ecl32”，“af23560” の 2 種類である．これらの係数行列の次元数と非零要素数はそれぞれ “ecl32” が 51,993, 347,097 であり，“af23560” が 23,560，460,456 である．これらの係数行列の非零要素は数値例 1 で扱ったものより不規則に配置されている．どちらの例でも厳密解の要素がすべて 1.0 となるように連立 1 次方程式の右辺ベクトル $\boldsymbol{b}$ を設定した．数値例 2 の係数行列では，必ずしも次元数 n が PE 台数 $d = 6$ で割り切れるとは限らない．この場合，たとえば “af23560” では次元数は 23,560 なので，integer$(n/d) = 3{,}926$, mod$(n,\ d) = 4$ となる．したがって，PE0 から PE3 まで担当するベクトルの次元数は $m = 3{,}927$ であり，PE4 から PE5 までは $m = 3{,}926$ となる．また，数値例 2 では $\tilde{m}$ に必ずしも m を割り切ることができない値を設定している．この場合，3.3 節で述べたように，最後に m を $\tilde{m}$ で割った余りの分だけを送信することになる．これらのことをふまえて Sherman-Morrison 法と ILU(2) 法の最適なブロック数 $\tilde{m}$ の計測結果をそれぞれ**表 6** と**表 7** に示した．“ecl32” の場合には，Sherman-Morrison 法では tolU，tolV = 0.1 と tolU，tolV = 0.01 のいずれの場合も通信回数が 32 回，つまり $\tilde{m} = 271$ のとき計算時間が最小となった．ILU(2) 法では $\tilde{m} = 8$ のとき最小となった．また，“af23560” の場合には，Sherman-Morrison 法では tolU，tolV = 0.01 のとき 2 時間以内に前処理行列を計算できなかったが，tolU，tolV = 0.1 かつ $\tilde{m} = 246$ のときに計算時間は最小となった．この例において，ILU(2) 法の計算時間が最小となったのは，$\tilde{m} = 32$ のときであった．したがって，残差ノルムの収束評価の実験で Sherman-Morrison 法による前処理行列を用いるときは，“ecl32” では $\tilde{m} = 271$ を利用し，“af23560” では $\tilde{m} = 246$ を用いることにした．また，ILU 分解においては ILU(2) 法での計測結果をもとにして，“ecl32” と “af23560” では，それぞれ $\tilde{m} = 8$ と $\tilde{m} = 32$ を用いた．

次に，これら 2 つの行列について，数値例 1 同様に

表 8 数値例 2：Sherman-Morrison 法による前処理行列の非零要素数
Table 8 Example2: Nonzero entries of the Sherman-Morrison preconditioner.

(a) ecl32

tolU / tolV	$\tilde{s}$							
	10^0		10^1		10^2		10^3	
	U	V	U	V	U	V	U	V
0.1	204,069	2,255,976	204,069	3,762,618	204,069	5,604,401	204,069	7,775,226
0.01	1,535,930	11,331,526	1,535,930	17,858,501	1,535,930	25,564,994	1,535,930	33,866,053

(b) af23560

tolU / tolV	$\tilde{s}$							
	10^0		10^1		10^2		10^3	
	U	V	U	V	U	V	U	V
0.1	551,740	32,207,303	551,740	61,057,101	551,740	93,002,049	551,740	124,318,919

表 9 数値例 2：Sherman-Morrison 法による前処理行列の計算の速度向上（af23560 かつ tolU，tolV = 0.1）
Table 9 Example 2: Speedup of computation of the Sherman-Morrison preconditioner (af23560 and tolU, tolV = 0.1).

PE 数	$\tilde{s}$	
	10^0	10^1
1	1.0	1.0
2	1.7	1.9
4	2.8	3.0
6	3.8	4.0

s を変化させて U，V の非零要素数を計測した．ただし，“af23560” の tolU，tolV = 0.01 では，計算が完了しなかったので計測を省略した．**表 8** から，これらの例においても s に比例して V の非零要素数が増加する傾向があることが分かる．特に，“af23560” では $\tilde{s} = 10^3$ のとき $\tilde{s} = 10^0$ のときの約 4 倍の非零要素数が得られた．この場合，非零要素 1 つを 8 バイトの倍精度の浮動小数点型で記憶していることを考えると，$8 \times 124{,}318{,}919 = 994{,}551{,}352$ なので，V の非零要素の記憶には 994 MB 必要となる．したがって，3 つのノードを使用していることから 1 ノードあたり V の非零要素に約 331 MB 程度の記憶領域が必要となる．このことを考慮すると，必ずしも s を大きくとることが有効とは限らない．ここで，$\tilde{s}$ がそれぞれ 10^0，10^1 のときの台数効果を**表 9** に示した．この例では，数値例 1 ほどの台数効果は望めないものの，6 台使用したときでも 4 倍近い速度向上が観測されており，理想の 2/3 程度の性能が得られた．また，s の桁数を変化させても速度向上率はほぼ同じであり，この変化による影響は限定的であった．**表 10** には “ecl32” と “af23560” をそれぞれ係数とする連立 1 次方程式について，収束判定条件を満たすまでに要した GMRES(m) 法の計算時間と反復回数を示した．MR 法による前処理を適用しても収束は改善されなかった．ILU 分解による前処理では，ILU(0) 法を適用した GMRES(40) 法の場合のみ残差ノルムは収束した．この場合，非常に多くの反復回数が必要となるが，前処理行列の計算時間は短いので，総合的には Sherman-Morrison 法と同じくらいの時間で残差ノルムは収束している．Sherman-Morrison 法で計算した前処理行列を使用すると，ILU 分解で残差ノルムが収束した 1 例に比べて計算時間はかかっているものの，閾値 tolU，tolV の値を 0.01 に設定すれば，すべての GMRES(m) 法で残差ノルムは収束している．特に，この前処理行列の計算には MR 法で反復を 3 回にした場合よりも計算時間が多く必要になるが，MR 法で前処理行列を計算しても残差ノルムが収束しなかったことを考慮すると，前処理行列の計算のコストが割高となっても Sherman-Morrison 法を用いる価値があるといえる．“af23560” の場合，閾値が 0.01 では計算が 2 時間以内に終わらず前処理を GMRES(m) 法に適用することができなかったが，閾値が 0.1 のときにはすべての GMRES(m) 法で残差ノルムの収束を確認することができた．それに対して，他の前処理行列を適用しても残差ノルムは収束することはなかった．したがって，このように係数行列の疎構造が比較的不規則な問題に対しては，ILU 分解による前処理行列よりも Sherman-Morrison 法による近似逆行列を使う方が有効であると考えられる．

［数値例 3］ $n = 240{,}000$ とするとき，次のようなブロック対角行列

$$A = \mathrm{diag}\,[A_1, A_2, \ldots, A_{\tilde{n}}] \in R^{n \times n} \qquad (30)$$

を係数とする連立 1 次方程式を考える[2]．ただし，

$$A_j = \begin{bmatrix} \lambda_{re,j} & \lambda_{im,j} \\ -\lambda_{im,j} & \lambda_{re,j} \end{bmatrix} \in R^{2\times 2}, \quad j = 0,\ 1, \ldots, \tilde{n}$$

かつ $\tilde{n} = 120{,}000$ である．厳密解の要素は，すべて

表 10 数値例 2：計算時間と反復回数（time：計算時間（秒），iter：反復回数）
Table 10 Example 2: The computation time and iteration count (time: Computation time (s), iter: Iteration count).

(a) ecl32

前処理の種類	前処理行列の計算	非定常反復法					
		GMRES(20)		GMRES(30)		GMRES(40)	
	time	time	iter	time	iter	time	iter
なし	0.0	–	–	–	–	–	–
SM 法（tolU，tolV = 0.1，$\tilde{s} = 10^0$）	208.0	–	–	–	–	–	–
SM 法（tolU，tolV = 0.1，$\tilde{s} = 10^1$）	290.0	–	–	–	–	–	–
SM 法（tolU，tolV = 0.01，$\tilde{s} = 10^0$）	1,038.0	1,128.0	790	1,090.0	408	1,065.0	231
SM 法（tolU，tolV = 0.01，$\tilde{s} = 10^1$）	1,455.0	1,570.0	819	1,519.0	415	1,485.0	221
ILU(0) 法	0.48	–	–	–	–	1,188.0	22,553
ILU(1) 法	1.14	–	–	–	–	–	–
ILU(2) 法	2.13	–	–	–	–	–	–
MR 法（tol = 0.1，imax = 3）	402.0	–	–	–	–	–	–
MR 法（tol = 0.1，imax = 2）	238.0	–	–	–	–	–	–
MR 法（tol = 0.1，imax = 1）	99.0	–	–	–	–	–	–
MR 法（tol = 0.01，imax = 3）	544.0	–	–	–	–	–	–
MR 法（tol = 0.01，imax = 2）	277.0	–	–	–	–	–	–
MR 法（tol = 0.01，imax = 1）	99.0	–	–	–	–	–	–

(b) af23560

前処理の種類	前処理行列の計算	非定常反復法					
		GMRES(20)		GMRES(30)		GMRES(40)	
	time	time	iter	time	iter	time	iter
なし	0.0	–	–	–	–	–	–
SM 法（tolU，tolV = 0.1，$\tilde{s} = 10^0$）	1,175.0	1,344.0	759	1,328.0	621	1,302.0	549
SM 法（tolU，tolV = 0.1，$\tilde{s} = 10^1$）	2,304.0	2,562.0	740	2,526	649	2,478.0	544
ILU(0) 法	0.62	–	–	–	–	–	–
ILU(1) 法	1.31	–	–	–	–	–	–
ILU(2) 法	2.10	–	–	–	–	–	–
MR 法（tol = 0.1，imax = 3）	180.0	–	–	–	–	–	–
MR 法（tol = 0.1，imax = 2）	111.0	–	–	–	–	–	–
MR 法（tol = 0.1，imax = 1）	48.0	–	–	–	–	–	–
MR 法（tol = 0.01，imax = 3）	254.0	–	–	–	–	–	–
MR 法（tol = 0.01，imax = 2）	126.0	–	–	–	–	–	–
MR 法（tol = 0.01，imax = 1）	48.0	–	–	–	–	–	–

–：2 時間以内に計算が完了しなかった場合
SM 法：Sherman-Morrison 法，　imax：MR 法の反復回数

[0, 1] の範囲の乱数で決定し右辺 $\boldsymbol{b}$ を設定した．$\lambda_{im,j}$ は [−1, 1] の範囲の乱数を利用して設定し，$\lambda_{re,j}$ は $[10^{-3}, 10^{-2}]$ の範囲の乱数を利用した．他の数値例同様，最初に通信回数を変化させて，6 台の PE で Sherman-Morrison 法と ILU 分解による前処理行列の計算にかかる時間を計測し，その結果をそれぞれ**表 11** と**表 12** に示した．Sherman-Morrison 法では $\tilde{m} = 625$ のとき，ILU 分解では $\tilde{m} = 20{,}000$ のときにそれぞれ計算時間が最小になったので，残差ノルムの収束評価の実験においてもブロック数としてこれらの値を用いた．次に，前の 2 つの数値例と同様にして s を変化させて非零要素数を計測し，その結果を**表 13** に示した．この数値例では，s を変化させても V の非零要素数に大幅な変化は見られなかった．また，数値例 1 と 2 と同様の手順で PE 台数を変化させて，Sherman-Morrison 法による前処理行列の計算時間を計測した．**表 14** から，この例では PE 6 台で 3 倍以上の速度向上が得られた．さらに，GMRES(m) 法の残差ノルムが収束するまでに要した計算時間と反復回数を**表 15** に示した．この数値例では，ILU 分解による前処理行列を使用することが最も効果的であり，反復回数はいずれも 1 回で残差ノルムは収束している．その理由は，ILU 分解がすべて完全な LU 分解になってしまったためである．つまり，この例では完全な LU 分解が可能であり，この前処理行列を適用した時点で解が求まっていることになる．この数値例の係数行列は三重対角行列でもあり，かつ

表 11 数値例 3：Sherman-Morrison 法のブロック数と計算時間の関係
Table 11 Example 3: The relation between the number of blocks and the computation time in the Sherman-Morrison formula.

(a) tolU，tolV = 0.1

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	40,000	20,000	10,000	5,000	2,500	1,250	625	313
計算時間（秒）	1,512.0	1,146.0	1,055.0	1,034.0	1,030.0	1,027.0	1,027.0	1,028.0

(b) tolU，tolV = 0.01

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	40,000	20,000	10,000	5,000	2,500	1,250	625	313
計算時間（秒）	1,513.0	1,146.0	1,056.0	1,034.0	1,031.0	1,027.0	1,026.0	1,027.0

表 12 数値例 3：ILU(2) 法のブロック数と計算時間の関係
Table 12 Example 3: The relation between the number of blocks and the computation time in the ILU(2) method.

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	40,000	20,000	10,000	5,000	2,500	1,250	625	313
計算時間（秒）	0.78	0.76	0.77	0.79	0.88	1.07	1.49	2.85

表 13 数値例 3：Sherman-Morrison 法による前処理行列の非零要素数
Table 13 Example 3: Nonzero entries of the Sherman-Morrison preconditioner.

tolU / tolV	$\tilde{s}$							
	10^0		10^1		10^2		10^3	
	U	V	U	V	U	V	U	V
0.1	359,937	479,930	359,937	479,931	359,937	479,931	359,937	479,931
0.01	359,990	479,984	359,990	479,984	359,990	479,984	359,990	479,984

表 14 数値例 3：Sherman-Morrison 法による前処理行列の計算の速度向上（tolU，tolV = 0.01）
Table 14 Example 3: Speedup of computation of the Sherman-Morrison preconditioner (tolU, tolV = 0.01).

PE 数	$\tilde{s}$	
	10^0	10^1
1	1.00	1.00
2	1.39	1.39
4	2.38	2.39
6	3.12	3.14

$$a_{1,1} \neq 0, \begin{vmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{vmatrix} \neq 0,$$

$$\begin{vmatrix} a_{1,1} & a_{1,2} & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} \\ 0 & a_{3,2} & a_{3,3} \end{vmatrix} \neq 0,$$

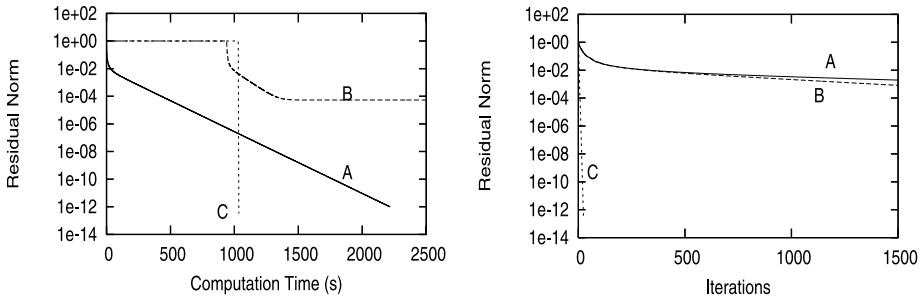
$$\ldots\ldots, \det(A) \neq 0$$

の条件を満たす．一般に，このような条件を満たす三重対角行列は fill-in をいっさい起こさずに完全な LU 分解が可能であることが知られている[11]．したがって，このような係数行列を持つ連立 1 次方程式は直接法で解くことも可能である．しかし，Sherman-Morrison 法による前処理行列を使用する場合でも，前処理を行わない GMRES(m) 法や MR 法による前処理行列を用いる場合に比べて，残差ノルムの効果的な収束が得られた．この場合，閾値が 0.1 と 0.01 の場合，どちらでも前処理行列の計算時間はほぼ同じであった．これは，異なる閾値を設定したにもかかわらず，表 13 で示したように，前処理行列の非零要素数がほぼ同じになったためであると考えられる．また，s の値を変化させても非零要素数が変化しなかったため，この値が速度向上に影響を及ぼすことはなかった．残差ノルムの収束に関しては，前処理なしでも収束するものの，その反復回数は Sherman-Morrison 法による前処理行列を適用したときよりも約 1,000 倍余計にかかっている．また，MR 法による前処理行列を適用しても残差ノルムの収束を改善することはできず，かえって収束を悪化させている．一方，Sherman-Morrison 法で行列の前処理を行う場合，前処理行列の計算だけで 20 分近くかかっている．しかし，この計算時間を埋め合わせるだけの反復回数の減少がみられており，前処理行列の計算時間を考慮しても残差ノルムの収束にかかる計算時間は，前処理を行わない場合よりも少ない．また，この数値例では s を変化させると反復回数は変わるも

表 15 数値例 3：計算時間と反復回数（time：計算時間（秒），iter：反復回数）
Table 15 Example 3: The computation time and iteration count (time: Computation time (s), iter: Iteration count).

前処理の種類	前処理行列の計算	非定常反復法					
		GMRES(20)		GMRES(30)		GMRES(40)	
	time	time	iter	time	iter	time	iter
なし	0.0	1,891.0	24,083	2,065.0	23,455	2,216.0	23,131
SM 法（tolU，tolV = 0.1，$\tilde{s} = 10^0$）	1,027.0	1,035.0	24	1,034.0	24	1,034.0	24
SM 法（tolU，tolV = 0.1，$\tilde{s} = 10^1$）	1,026.0	1,027.0	9	1,027.0	9	1,027.0	9
SM 法（tolU，tolV = 0.01，$\tilde{s} = 10^0$）	1,026.0	1,034.0	24	1,034.0	24	1,035.0	24
SM 法（tolU，tolV = 0.01，$\tilde{s} = 10^1$）	1,027.0	1,028.0	9	1,028.0	9	1,028.0	9
ILU(0) 法	0.76	1.0	1	1.0	1	1.0	1
ILU(1) 法	0.76	1.0	1	1.0	1	1.0	1
ILU(2) 法	0.76	1.0	1	1.0	1	1.0	1
MR 法（tol = 0.1，imax = 1）	636.0	–	–	–	–	–	–
MR 法（tol = 0.01，imax = 1）	636.0	–	–	–	–	–	–
MR 法（tol = 0.1，imax = 2）	1,271.0	–	–	–	–	–	–
MR 法（tol = 0.01，imax = 2）	1,274.0	–	–	–	–	–	–

–：2 時間以内に計算が完了しなかった場合
SM 法：Sherman-Morrison 法，　imax：MR 法の反復回数



(a) 残差ノルム vs. 計算時間　　(b) 残差ノルム vs. 反復回数

図 5 数値例 3：GMRES(40) 法の残差ノルムの収束する様子，A：前処理なし，B：MR 法（tol = 0.01，imax = 1），C：Sherman-Morrison 法（tolU，tolV = 0.01，$\tilde{s} = 1.0$）
Fig. 5 Example 2: The behavior of convergence of the residual norm in GMRES(40) algorithm, A: no preconditioning, B: The MR method (tol = 0.01, imax = 1), C: The Sherman-Morrison formula (tolU, tolV = 0.01, $\tilde{s} = 1.0$).

のの，残差ノルムの収束にかかる計算時間に大きな変化はみられなかった．ここで，**図 5** に GMRES(40) 法の残差ノルムの収束する様子を示す．前処理を行わなくても残差ノルムは計算時間に対して一定の速度で収束している．一方，Sherman-Morrison 法による前処理を適用した GMRES(40) 法の残差ノルムは，途中の 1,000 秒過ぎから急激に 1.0×10^{-12} 付近まで下降している．これは 1,000 秒付近まで前処理行列の計算をしており，この後 24 回の反復回数ですぐに収束したためである．なお，MR 法による前処理を適用した場合，前処理行列の計算が終了した 900 秒くらいから急激に残差ノルムが減少しているものの，その収束は長続きせず 1,500 秒付近から停滞しており前処理の効果が現れていない．

ここで，数値例全体の台数効果を比較するため，**表 16** に各 PE の稼働率と担当した非零要素数とを示す．ただし，送受信の時間とは，各 PE が送信と受信に要した時間の合計であり，これらの通信での待ち時間も含まれる．また，PE0 以外の PE の最初に要素を受信するまでの受信待ち時間も含まれている．同期の時間とは，各 PE の担当する非零要素をすべて計算した後，別の PE の計算が終了するまでの待ち時間のことである．稼働率の計算は

“稼働率=計算時間 / (計算時間
　　+送受信の時間+同期の時間)”

によって行った．また，担当した非零要素数とは，各

表 16 各 PE の稼働率と担当した非零要素数
Table 16 The operating ratio and the number of nonzero entries that each PE covers.

(a) 数値例 1，tolU，tolV = 0.01，$\tilde{s} = 10^0$

PE 番号	担当した非零要素数	計算時間	送受信の時間	同期の時間	稼働率（%）
0	2,125,451	36.0	14.0	334.0	9.0
1	2,433,024	101.0	18.0	265.0	26.0
2	2,452,044	156.0	33.0	195.0	41.0
3	2,363,144	212.0	49.0	123.0	56.0
4	2,445,060	266.0	64.0	54.0	70.0
5	2,396,926	324.0	60.0	0.0	85.0

(b) 数値例 2，af23560 かつ tolU，tolV = 0.1，$\tilde{s} = 10^0$

PE 番号	担当した非零要素数	計算時間	送受信の時間	同期の時間	稼働率（%）
0	2,057,346	40.0	6.0	1,129.0	3.0
1	7,026,059	242.0	15.0	918.0	21.0
2	11,468,891	589.0	77.0	509.0	50.0
3	8,740,500	856.0	210.0	109.0	73.0
4	2,627,566	771.0	401.0	23.0	56.0
5	838,681	752.0	413.0	0.0	66.0

(c) 数値例 3，tolU，tolV = 0.01，$\tilde{s} = 10^0$

PE 番号	担当した非零要素数	計算時間	送受信の時間	同期の時間	稼働率（%）
0	139,996	11.0	78.0	937.0	1.0
1	139,997	180.0	95.0	751.0	17.0
2	139,993	342.0	122.0	562.0	33.0
3	139,999	484.0	167.0	375.0	47.0
4	139,995	628.0	213.0	185.0	62.0
5	139,994	768.0	258.0	0.0	75.0

稼働率=計算時間 / (計算時間+送受信の時間+同期の時間)

PE が担当した列ベクトル $\boldsymbol{u}_k$ と $\boldsymbol{v}_k$ に含まれていた非零要素数の合計値のことである．数値例 1 の各 PE の稼働率は，他の 2 つに比べて高いということが分かった．数値例 2 では，各 PE の計算した非零要素数にバラつきがあった．これは，係数行列が数値例 1 と 3 に比べて元々不規則な構造になっているためである．そのため，数値例 2 では各 PE の計算した非零要素数にはバラつきがあり，PE の稼働率も数値例 1 より低下したと考えられる．それに対して，数値例 3 では数値例 2 ほど非零要素数のバラつきはなく，数値例 1 同様に各 PE で比較的均一であった．しかし，数値例 1 と比べて各 PE の計算した非零要素数は少なかったために，計算が比較的早く終了してしまい，その後の別の PE との同期に数値例 1 よりも多くの時間がかかったからである．以上の理由から，数値例 2 と 3 は数値例 1 ほどの並列処理効率が出なかったと考えられる．

5. 結　　論

本稿では，Naik[4] の手法を用いて，Sherman-Morrison 法による前処理行列としての近似逆行列を求める計算手順を並列化する方法を提案した．列ベクトルを各 PE に分割し，これらのベクトルを $\tilde{m}$ 個だけ計算するたびに通信を行い，計算と通信の過程を交互に繰り返すことにより部分的な並列化を可能とした．また，このような並列化により一定の速度向上が得られた．さらに，本稿で提案した手法で計算を行った近似逆行列前処理は，MR 法を並列化して計算したものよりも残差ノルムの収束を改善できることを確認した．したがって，本稿で提案した手法は，PC クラスタシステムの環境で Sherman-Morrison 法による近似逆行列の計算を高速化することができるといえる．

参 考 文 献

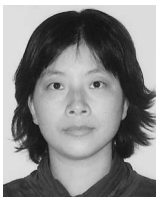
1) Saad, Y. and Schultz, M.H.: GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems, *SIAM J. Sci. Stat. Comput.*, No.7, pp.856–869 (1986).
2) Gutknecht, M.H.: Variants of BiCGStab for Matrices with Complex Spectrum, *SIAM J. Sci. Comput.,* Vol.14, pp.1020–1033 (1993).
3) Bruaset, A.M.: A Survey of Preconditioned

Iterative Methods, *Pitman Research Note in Mathematics*, No.32, Longman Scientific & Technical, U.K. (1995).
4) Naik, V.K.: A Scalable Implementation of the NAS Benchmark BT on Distributed Memory Systems, *IBM Systems Journal*, Vol.34, No.2, pp.273–291 (1995).
5) Grote, M. and Huckel, T.: Parallel Preconditioning with Sparse Approximate Inverses, *SIAM J. Sci. Comput.*, Vol.18, No.3, pp.838–853 (1997).
6) Huckel, T.: Efficient Computation of Sparse Approximate Inverses, *Numer. Linear Algebra Appl.*, Vol.5, pp.57–71 (1998).
7) Huckel, T.: Approximate Sparsity Patterns for the Inverse of a Matrix and Preconditioning, *Appl. Numer. Math.*, No.30, pp.291–303 (1999).
8) Simoncini, Y.: On the Convergence of Restarted Krylov Subspace Method, *SIAM J. Matrix. Anal. Appl.*, Vol.22, No.2, pp.430–452 (2000).
9) Montero, G., González, L., Flórez, E., García, M.D. and Suárez, A.: Approximate Inverse Computation using Frobenius Inner Product, *Numer. Linear Algebra Appl.*, Vol.9, pp.239–247 (2002).
10) 森屋健太郎，野寺　隆：並列性を考慮した大規模な線形システムの前処理，応用数理，Vol.12, No.1, pp.14–28 (2002).
11) 山本哲朗：数値解析入門・増訂版，サイエンス社 (2003).
12) Bru, R., Credán, J., Marín, J. and Mas, J.: Preconditioning Sparse Nonsymmetric Linear Systems with the Sherman-Morrison Formula, *SIAM J.Sci.Comput.,* Vol.25, No.2, pp.701–715 (2003).
13) University of Florida Sparse Matrix Collection. [Online] http://www.cise.ufl.edu/research/sparse/matrices
14) MPICH Home Page. [Online] http://www-unix.mcs.anl.gov/mpi/mpich1/

(平成 18 年 5 月 22 日受付)
(平成 18 年 12 月 7 日採録)

森屋健太郎（正会員）

1999 年富士通株式会社入社．2003 年 4 月より青山学院大学理工学部助手に就任し現在に至る．その間，2000 年 9 月慶應義塾大学大学院理工学研究科後期博士課程（基礎理工学専攻）に入学し，2002 年 3 月同大学大学院同研究科後期博士課程修了．工学博士．並列計算機による大規模計算に興味を持つ．



張　臨傑

1997 年中国西安交通大学コンピュータ学部コンピュータ科学 & 工学学科卒業．2004 年慶應義塾大学大学院理工学研究科基礎理工学専攻博士前期課程修了．現在，同大学院理工学研究科基礎理工学専攻博士後期課程に在学中．計算機による大規模計算に興味を持つ．

野寺　隆（正会員）

1982 年慶應義塾大学大学院工学研究科博士課程（数理工学専攻）修了．現在，同大学教授．その間，1986 年より 1 年間米国スタンフォード大学客員教授．大規模な行列計算の算法の研究開発に従事．ハイパフォーマンス・コンピューティングや文書処理に興味を持つ．著書に『楽々LaTeX』（共立出版）等がある．工学博士．エッセイスト．SIAM，日本応用数理学会各会員．