数理解析研究所講究録
1505巻 2006年 170-182

Sherman-Morrison法の部分的な並列化による近似逆行列計算の高速化について

† 青山学院大学理工学部　森屋 健太郎 (Kentaro Moriya)
†† 慶應義塾大学大学院理工学研究科　張臨傑 (Linjie Zhang)
††† 慶應義塾大学理工学部　野寺隆 (Takashi Nodera)
† Faculty of Science and Technology, Aoyama Gakuin University
†† Graduate school of Keio University
††† Faculty of Science and Technology, Keio University

概 要

近似逆行列は，大規模で疎な連立1次方程式に対する有効な前処理の1つとしてしばしば用いられる．近年，その計算手法としてSherman-Morrison法が注目を集めている．この手法は，行列分解により近似逆行列を求めている．しかし，この手法の行列分解の過程には逐次処理が含まれるので，それを並列化することはそれほど簡単ではない．本稿では，Sherman-Morrison法による近似逆行列の行列分解の過程を部分的に並列化する実装について提案し，Pentium Xeon 3.6GHzを6台搭載したPCクラスタシステムによる数値実験結果を報告する．

1 はじめに

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

$$Ax = b \tag{1}$$

を非定常反復法で解く場合，必要となる反復回数を減少させるために前処理を利用することが多い．特に，近年では近似逆行列を前処理として好んで使用する傾向があり，通常は前処理行列 M^{-1} が $M^{-1} \cong A^{-1}$ となるように計算される．近年，Bruら [5] はSherman-Morrison法を用いて近似逆行列を求めることを提案し，この手法によって計算された近似逆行列が前処理として有効であることを示している．しかし，この手法を用いた近似逆行列の計算には行列分解に逐次処理が含まれるので，並列化を行うには不向きであることが指摘されている．

本稿ではこのような難点を改善するために，Sherman-Morrison法による近似逆行列の計算を部分的に並列化する手法を提案する．Pentium Xeon 3.6GHzで構成されたPCクラスタシステムによる数値実験の結果から，提案した手法が優れた台数効果を発揮し，かつMR法 [3] による前処理以上に残差ノルムの収束性を改善できることを示す．

2 Sherman-Morrison法を用いた近似逆行列前処理

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

$$A = B + \boldsymbol{p}\boldsymbol{q}^T \tag{2}$$

のように分解することを考える．このとき A の逆行列は

$$A^{-1} = B^{-1} - r^{-1}B^{-1}\boldsymbol{p}\boldsymbol{q}^T B^{-1} \tag{3}$$

のように表わすことができる．ただし，$r = 1 + \boldsymbol{q}^T B^{-1}\boldsymbol{p} \neq 0$ である．式 (3) のことをSherman-Morrisonの公式 [5] という．以降，Sherman-Morrisonの公式 (3) を用いた行列 A の近似逆行列の計算法を述べる．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．

$$A_k = A_{k-1} + \boldsymbol{p}_k\boldsymbol{q}_k^T, \quad k = 1,\ 2,\ \ldots, n \tag{4}$$

ただし，$r_k = 1 + \boldsymbol{q}_k^T A_{k-1}^{-1}\boldsymbol{p}_k \neq 0$ である．ここで，式 (4) に対してSherman-Morrisonの公式 (3) を適用すると

$$A_k^{-1} = A_{k-1}^{-1} - r_k^{-1}A_{k-1}^{-1}\boldsymbol{p}_k\boldsymbol{q}_k^T A_{k-1}^{-1} \tag{5}$$

を導くことができる．$A^{-1} = A_n^{-1}$ として，A_n^{-1} を式 (5) を用いて展開すると

$$A_0^{-1} - A^{-1} = \Psi\Omega^{-1}\Xi^T \tag{6}$$

が得られる．ただし，

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

であり，対角行列 Ω の 対角成分 r_k は

$$r_k = 1 + \boldsymbol{q}_k^T A_{k-1}\boldsymbol{p}_k \tag{7}$$

である．さらに，

$$\boldsymbol{u}_k := \boldsymbol{p}_k - \sum_{i=1}^{k-1}\frac{(\boldsymbol{v}_i^T, A_0^{-1}\boldsymbol{p}_k)}{r_i}\boldsymbol{u}_i \tag{8}$$

$$\boldsymbol{v}_k := \boldsymbol{q}_k - \sum_{i=1}^{k-1}\frac{(\boldsymbol{q}_k^T, A_0^{-1}\boldsymbol{u}_i)}{r_i}\boldsymbol{v}_i \tag{9}$$

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 (6) と式 (7) はそれぞれ

$$A_0^{-1} - A^{-1} = A_0^{-1}U\Omega^{-1}V^T A_0^T \tag{10}$$

$$r_k = 1 + \boldsymbol{q}_k^T A_0^{-1}\boldsymbol{u}_k \tag{11}$$

Sherman-Morrison formula

1: for $k = 1$ to n do
2: $\quad \boldsymbol{p}_k = \boldsymbol{e}_k$, $\boldsymbol{q}_k = (\boldsymbol{a}^k - s\boldsymbol{e}_k)^T$
3: $\quad \boldsymbol{u}_k = \boldsymbol{p}_k$, $\boldsymbol{v}_k = \boldsymbol{q}_k$
4: $\quad$ for $i = 1$ to $k-1$ do
5: $\quad\quad \boldsymbol{u}_k = \boldsymbol{u}_k - \{(v_i)_k \ / \ (sr_i)\}\boldsymbol{u}_i$
6: $\quad\quad \boldsymbol{v}_k = \boldsymbol{v}_k - \{(\boldsymbol{q}_k,\ \boldsymbol{u}_i) \ / \ (sr_i)\}\boldsymbol{v}_i$
7: $\quad$ endfor
8: $\quad$ for $i = 1$ to n do
9: $\quad\quad$ if $|(u_k)_i| <$ tolU dropoff $|(u_k)_i|$
10: $\quad\quad$ if $|(v_k)_i| <$ tolV dropoff $|(v_k)_i|$
11: $\quad$ endfor
12: $\quad r_k = 1 + (v_k)_k/s$
13: endfor

図 1: Sherman-Morrison 法

と書き直すことができる [5]. Bru ら [5] によれば $A_0^{-1} - A^{-1}$ が A の近似逆行列 M^{-1} となるもっとも簡単な A_0, $\boldsymbol{p}_k$, $\boldsymbol{q}_k$ 取り方は

$$A_0 = sI, \quad \boldsymbol{p}_k = \boldsymbol{e}_k, \quad \boldsymbol{q}_k = (\boldsymbol{a}^k - s\boldsymbol{e}_k)^T \tag{12}$$

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. $A_0 = sI$ として, 式 (10) の左辺を M^{-1} とおくと, $M^{-1} = s^{-1}I - A^{-1}$ から M^{-1} は A^{-1} と対角成分のみが異なることになる. さらに, Bru らは $\rho(A) < s$ を満たす取り方の 1 つとして, $s = 1.5\,\|A\|_\infty$ の選択を提案している. 従って, 4 章における数値実験でもこの値を使用する.

A_0, $\boldsymbol{p}_k$, $\boldsymbol{q}_k$ を式 (12) のように取ると, 式 (8), (9), (11) はそれぞれ

$$\boldsymbol{u}_k = \boldsymbol{p}_k - \sum_{i=1}^{k-1} \frac{(v_i)_k}{sr_i}\boldsymbol{u}_i \tag{13}$$

$$\boldsymbol{v}_k = \boldsymbol{q}_k - \sum_{i=1}^{k-1} \frac{(\boldsymbol{q}_k, \boldsymbol{u}_i)}{sr_i}\boldsymbol{v}_i \tag{14}$$

$$r_k = 1 + (\boldsymbol{v}_k)_k/s \tag{15}$$

となり, 式 (10) から前処理行列 M^{-1} は

$$M^{-1} = s^{-1}I - A^{-1} = s^{-2}U\Omega^{-1}V^T \tag{16}$$

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. 従って, Sherman-Morrison 法に基づいて近似逆行列を求めるには, 式 (13) から式 (15) を用いて式 (16) の右辺のよ

うに行列分解を行えばよい．式 (16) は，M^{-1} と A^{-1} が理論上は対角成分のみが異なるということを意味している．ただし，実際には計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 Sherman-Morrison 法による近似逆行列前処理の実装は図 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 法による行列分解で問題となるのは，式 (13) と式 (14) による $\boldsymbol{u}_1 \ldots, \boldsymbol{u}_k$ と $\boldsymbol{v}_1 \ldots, \boldsymbol{v}_k$ の計算が逐次処理となり並列化がそれほど簡単にはできないということである．

3 Sherman-Morrison 法前処理の並列化

3.1 行列分解の問題点

Sherman-Morrison 法による近似逆行列の計算の実装では，式 (13) と式 (14) にによる $\boldsymbol{u}_k$ と $\boldsymbol{v}_k$ の計算に $\boldsymbol{u}_1$, $\boldsymbol{u}_2, \ldots, \boldsymbol{u}_{k-1}$, $\boldsymbol{v}_1$, $\boldsymbol{v}_2, \ldots, \boldsymbol{v}_{k-1}$ 及び r_1, $r_2, \ldots, r_{k-1}$ が必要であり，行列 U と V の並列実装がそれほど簡単ではない．本節では各セルの行列分解において，列ベクトルの計算を通信と交互に行うことで，式 (13) と式 (14) の実装を部分的に並列化して行列 U, V, Ω の計算の高速化を実現する．

3.2 行列の割り当て

Sherman-Morrison 法を用いて近似逆行列前処理を求める場合，行列 U, V, Ω を計算することが必要となる．本稿では，3 つの行列の各列ベクトル $\boldsymbol{u}_k$, $\boldsymbol{v}_k$, r_k を各セルへ均等に分割することを考える．行列の次元数と使用するセル台数がそれぞれ n, d である場合，行列 U の列ベクトルの分割は

$$U = \{\underbrace{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m}_{\text{セル}_0}, \underbrace{\boldsymbol{u}_{m+1}, \ldots, \boldsymbol{u}_{2m}}_{\text{セル}_1}, \ldots, \underbrace{\boldsymbol{u}_{n-m+1}, \ldots, \boldsymbol{u}_n}_{\text{セル}_{d-1}}\}$$

となるように行う．ここで，m は各セルの担当する列ベクトルの個数であり，$m = n/d$ である．本節では列ベクトルが各セルに均等に割り当てられることを想定しているが，特に行列の列数がセル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 $l-1$ 番目までのセルは integer$(n/d) + 1$ 列だけ担当して，l 番目以降のセルは interger(n/d) 列だけ担当することになる．ただし，integer(n/d) は n/d の整数部分を表す．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 のみが各セルに割り当てられる．なお，係数行列 A は行方向に分割され，l 番目のセルが $ml + 1$ 行 n 列から $(l+1)m$ 行 n 列までの非零要素を担当する．

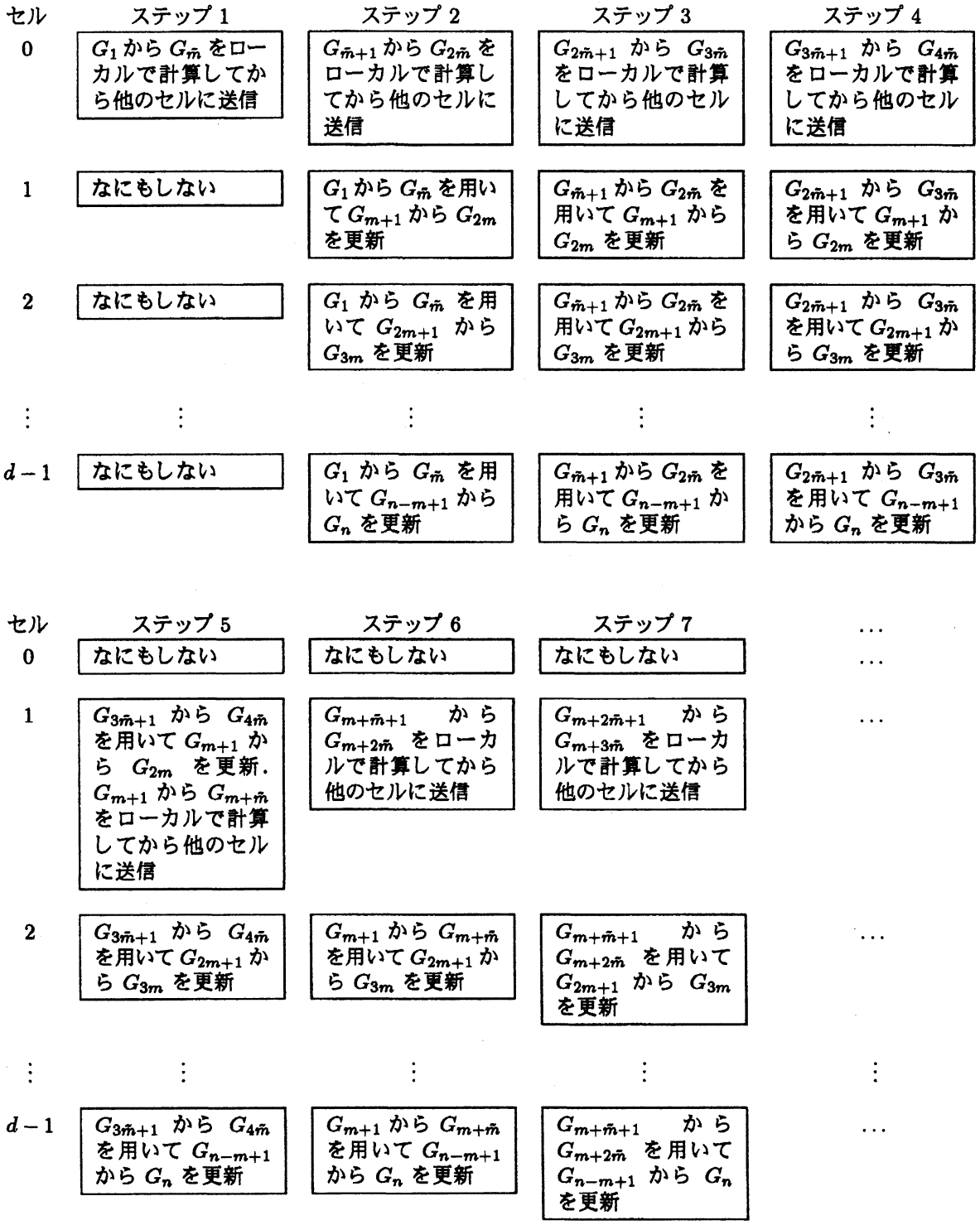


図 2: G_k を並列計算するしくみ ($m/\tilde{m} = 4$ の場合)

3.3 前処理行列の実装

3.2 節で述べたように列ベクトルを各セルに割り当て，式 (13) と式 (14) を用いて $\boldsymbol{u}_k$ と $\boldsymbol{v}_k$ を計算することを考える．式 (13) と 式 (14) はローカルに計算できる部分と通信を行うことで計算できる部分の 2 つに分けることができる．ただし，n は d で割り切れるものとし $m = n/d$ として考えることにする．このような場合，l 番目のセルが担当する 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．今，式 (13) を

$$\boldsymbol{u}_k = \boldsymbol{p}_k - \sum_{i=1}^{lm} \frac{(v_i)_k}{sr_i}\boldsymbol{u}_i - \sum_{i=lm+1}^{k-1} \frac{(v_i)_k}{sr_i}\boldsymbol{u}_i, \quad k = lm+1, \ldots, (l+1)m \tag{17}$$

のように分割すると，通信を行うことで計算できる部分は，式 (17) の右辺第 2 項であり，この計算は別のセル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とは式 (17) の右辺第 3 項であり，これらは他のセルからデータを受信することなしに計算できる．式 (17) の第 2 項で用いられているベクトルを別のセルから受信していないと $\boldsymbol{u}_{lm+1}$ が得られないので，第 2 項の計算が終わらないと第 3 項の計算を開始できないことが問題となる．従って，通常 l 番目のセルは 0 から $l-1$ 番目までのセル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，各セルが担当する列ベクトルの計算と通信を交互に行うことで，式 (13) と 式 (14) の計算を部分的に並列化することは可能である．つまり，あるセルが自分の担当する U, V, Ω の列ベクトルを何個か計算して，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セルに送信することになる．このときデータを送信したセルにはまだ計算すべきベクトルが残っているので，以降はこのセルとこのセルからデータを受信した残りのセルとで並列に動作することができる．例えば，おのおののセルが 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，もしセル 0 が 8 個すべて計算してから通信を行うと通信回数は 1 回で済むが，セル 0 の計算が終了しないと 1 番目以降のセルは動作しないことになる．それに対して，セル 0 が 2 個だけ計算してデータを残りのセルに送信すると，次のステップでは 1 番目以降のセル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，セル 0 は次の 2 個の列ベクトルをローカルに計算できる．従って，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 (14) 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，

$$G_k := \{\boldsymbol{u}_k, \boldsymbol{v}_k, r_k\} \tag{18}$$

を定義し，以降の説明では集合 G_k を用いて説明する．さらに，各セルの担当する G_k の個数つまり U, V, Ω のそれぞれの列ベクトルの個数を m，1 回に通信する G_k の個数を $\tilde{m}$，使用するセル台数を d とおくことにする．このような部分的な並列化をより一般化してまとめると図 2 のように表現できる．$m/\tilde{m}$ が通信回数に相当し，図 2 の例ではこの回数は 4 である．通信回数が多くなれば 1 ステップ目におけるセル 0 の計算が早く終わるので，全セルの動作する 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．従って，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セルの台数効果を向上させることには結び付くとは限らない．1回に通信する G_k の個数 $\tilde{m}$ の最適な値は並列計算システムの通信性能などに依存する．本稿では $\tilde{m}$ を以降ブロック数と呼ぶことにする．1セルの担当する G_k の個数 m が割り切れない場合には余りの個数分だけ最後に通信を行う．例えば，$m = 501$ 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個ずつ他セルに送信し，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

数値実験にはPentium Xeon 3.6GHzを2台搭載したノードを3つ用意し，それらをネットワークで接続したPCクラスタシステムを用いた．各ノードのメモリ搭載量は1GBである．最初に6台のセルを用いてSherman-Morrison法における前処理の計算でブロック数を変化させ適切な $\tilde{m}$ を決定した．次に，セル台数を1，2，4，6台と変更させてSherman-Morrison法による前処理を計算したときの並列処理効率を計測した．さらに6台のセルで，Sherman-Morrison法による前処理とMR法による前処理をGMRES(m)法 [1] に適用してそれぞれの性能を評価した．連立1次方程式の初期近似解は0ベクトル，GMRES(m)法の反復終了条件は

$$\|\boldsymbol{r}_i\|_2 \,/\, \|\boldsymbol{b}\|_2 < 1.0 \times 10^{-12} \tag{19}$$

とした．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．Sherman-Morrison法に関する各種の値として，tolUとtolVには0.1もしくは0.01を設定した．スカラ s には2章で述べたように，Bruら [5] にならって $s = 1.5\,\|A\|_\infty$ とした．また，MR法の非零要素の切り捨てを行う閾値をtolとし，この値にも0.1もしくは0.01とした．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として，この値には1から3を設定した．

[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 [4]．

$$\begin{aligned} -\tfrac{d}{dx}\left[\{\exp(-xy)\}u_x\right] - \tfrac{d}{dy}\left[\{\exp(xy)\}u_y\right] + 10.0(u_x + u_y) - 60.0u &= f(x,y) \\ u(x,\ y)|_{\partial\Theta} &= 1 + xy \end{aligned} \tag{20}$$

上記の偏微分方程式を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．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 $\mathrm{tolU} = \mathrm{tolV} = 0.1$ と $\mathrm{tolU} = \mathrm{tolV} = 0.01$ のいずれの場合も通信回数が128回の時つまり $\tilde{m} = 48$ のときに M^{-1} の計算時間は最小になった．従って，この事実に基づき，残差ノルムの収束評価を行う数値実験ではブロック数にこの値を設定した．次に，Sherman-Morrison法による M^{-1} の計算時間をセル台数を1台から6台まで変化させて計測した結果を図3に示した．ただし，各セル台数と

表 1: 数値例 1: Sherman-Morrison 法のブロック数と計算時間の関係

(a). tolU = tolV = 0.1 の場合

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	6144	3072	1536	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 の場合

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	6144	3072	1536	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

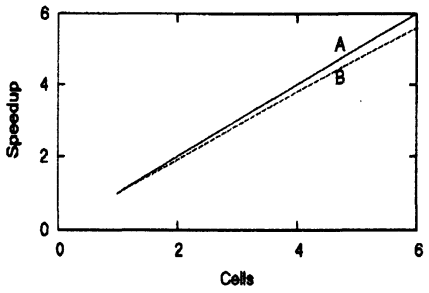


図 3: 数値例 1: Sherman-Morrison 法による前処理計算の台数効果, (tolU = tolV = 0.01), A: 理想, B: 実際

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. セルを 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, 理想の 90%程度の速度向上が得られた. 従って, 本稿で提案した部分的な並列化でも Sherman-Morrison 法による前処理の計算にはセル台数に応じて計算時間を短縮できると考えられる.

次に, MR 法による前処理と Sherman-Morrison 法による前処理をそれぞれ GMRES(m) 法に適用して, 残差ノルムの収束に関する数値実験を行った. 表 2 には GMRES(m) 法の残差ノルムが式 (19) 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 法による前処理を適用した場合, tol = 0.1 かつ imax = 1 の場合のみ GMRES(40) 法が収束した. このときの計算時間は Sherman-Morrison 法を適用したどの場合よりも早く収束している. しかし, それ以外の場合では MR 法の前処理を適用しても GMRES(m) 法は 1 時間以内には収束しなかった. それに対して, Sherman-Morrison 法による前処理を適用した場合, GMRES(20) 法の tolU = tolV = 0.1 の場合を除いて GMRES(m) 法はすべて 10 分以内に収束している. 従って, 後者による前処理の方が前者よりも安定して残差ノルムが収束することになる.

[数値例 2] Florida Sparse Matrix Collection [6] のデータベースにある行列の 1 つであ

表 2: 数値例 1: 連立 1 次方程式の計算時間と反復回数, (time: 計算時間 (秒), iter: 反復回数)

前処理の種類	前処理計算	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)	56.0	–	–	514.0	12125	289.0	5839
SM 法 (tolU = tolV = 0.01)	384.0	452.0	1160	420.0	565	409.0	382
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	6021

–: 1 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 3: 数値例 2: Sherman-Morrison 法のブロック数と計算時間の関係

(a). tolU = tolV = 0.1

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	3927	1964	982	491	246	123	62	31
計算時間 (秒)	1835.0	1465.0	1284.0	1185.0	1175.0	1223.0	1228.0	1234.0

る "af23560" を係数とする連立 1 次方程式を解く. この係数行列の次元数と非零要素数はそれぞれ 23560, 460456 であり, 係数行列の疎構造は数値例 1 で扱ったものに比べて不規則である. 連立 1 次方程式の右辺ベクトル $\boldsymbol{b}$ は厳密解の要素がすべて 1.0 となるように設定されている. 数値例 2 の係数行列では, 必ずしも次元数 n がセル台数 $d = 6$ で割り切れるとは限らない. この場合, 次元数は $n = 23560$ なので, integer$(n/d) = 3926$, mod$(n, d) = 4$ となる. 従って, セル 0 から 3 まで担当するベクトルの次元数は $m = 3927$, セル 4 から 5 までは $m = 3926$ 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. これらのことを踏まえて最適なブロック数 $\tilde{m}$ の計測結果を表 3 に示した. tolU = tolV = 0.1 のとき $\tilde{m} = 246$ のときに計算時間は最小となった. 従って, 残差ノルムの収束評価の実験で Sherman-Morrison 法による前処理を用いるときは, $\tilde{m} = 246$ を用いることにした. なお, tolU = tolV = 0.01 のとき 1 時間以内に前処理行列を計算できなかったので掲載を省略した. 次にこの係数行列について, 数値例 1 同様に, セル台数を変化させて Sherman-Morrison 法の前処理にかかる計算時間を計測し, この結果を図 4 に示す. 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. 従って, 本稿で提案した Sherman-Morrison 法による前処理の並列化は, このように

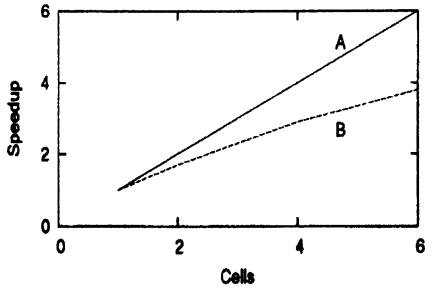


図 4: 数値例 2: Sherman-Morrison 法による前処理計算の台数効果, (tolU = tolV = 0.1), A: 理想, B: 実際

表 4: 数値例 2: 連立 1 次方程式の計算時間と反復回数, (time: 計算時間 (秒), iter: 反復回数)

前処理の種類	前処理計算	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)	1175.0	1344.0	759	1328.0	621	1302.0	549
SM 法 (tolU = tolV = 0.01)	–	–	–	–	–	–	–
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	–	–	–	–	–	–

–: 1 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

非零要素の並びが比較的不規則な問題でも効果を発揮すると考えられる. 表 4 には, 数値例 1 と同様に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. 前処理なしでは GMRES(m) 法は収束しなかった. また, MR 法による前処理を適用しても収束性は改善されなかった. それに対して Sherman-Morrison 法で前処理を計算すると, 閾値 tolU = tolV = 0.1 では残差ノルムは収束しなかったが, これらの値を 0.01 にして非零要素の切り捨てを少なくすると残差ノルムは収束している. この前処理計算には MR 法で反復を 3 回にした場合よりも計算時間がかかっているが, MR 法で前処理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.

表 5: 数値例 3: Sherman-Morrison 法のブロック数と計算時間の関係

(a-1). tolU = tolV = 0.1

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	40000	20000	10000	5000	2500	1250	625	313
計算時間 (秒)	1512.0	1146.0	1055.0	1034.0	1030.0	1027.0	1027.0	1028.0

(a-2). tolU = tolV = 0.01

通信回数	1	2	4	8	16	32	64	128
$\tilde{m}$	40000	20000	10000	5000	2500	1250	625	313
計算時間 (秒)	1513.0	1146.0	1056.0	1034.0	1031.0	1027.0	1026.0	1027.0

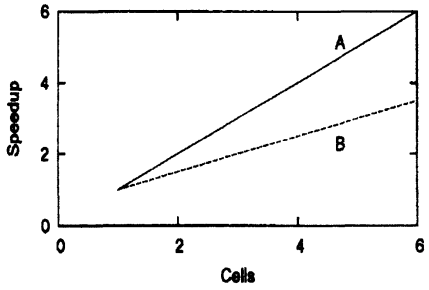


図 5: 数値例 3: Sherman-Morrison 法による前処理計算の台数効果, (tolU = tolV = 0.01), A: 理想, B: 実際

[数値例 3] $n = 240000$ とするとき，次のようなブロック対角行列

$$A = \begin{pmatrix} A_1 & & \\ & \ddots & \\ & & A_{\tilde{n}} \end{pmatrix} \in R^{n \times n} \tag{21}$$

を係数とする連立 1 次方程式を考える [2]. ただし，

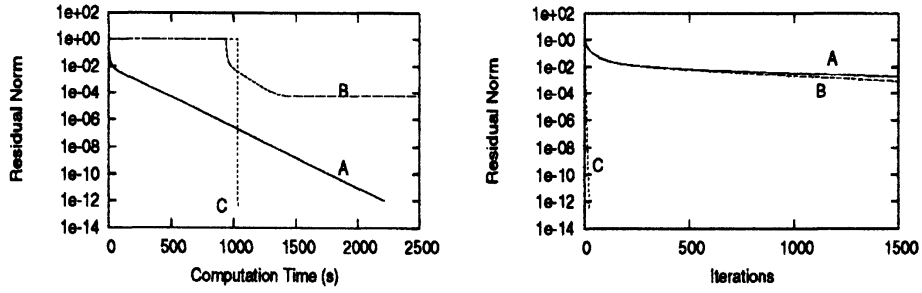
$$A_j = \begin{pmatrix} \lambda_{re,j} & \lambda_{im,j} \\ -\lambda_{im,j} & \lambda_{re,j} \end{pmatrix} \in R^{2\times 2}, \quad j = 0,\ 1, \ldots, \tilde{n} \tag{22}$$

かつ $\tilde{n} = 120000$ である. 厳密解の要素はすべて [0, 1] の範囲の乱数として右辺 $\boldsymbol{b}$ を設定した. $\lambda_{im,j}$ は $[-1,\ 1]$ の範囲の乱数とし，$\lambda_{re,j}$ は $[10^{-3},\ 10^{-2}]$ の範囲の乱数として設定した. 最初に通信回数を変化させて，セル 6 台で Sherman-Morrison 法による前処理計算にかかる時間を計測した結果を表 5 に示した. 通信回数が 64，つまり $\tilde{m} = 625$ のときに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でブロック数はこの値を用いた. 次に，数値例 1 と 2 と同様の手順でセル台数を変化させて，Sherman-Morrison 法による前処理計算に要する時間の計測結果を図 5 に示した. この例では，セル 6 台で理想

表 6: 数値例 3: 連立 1 次方程式の計算時間と反復回数, (time: 計算時間 (秒), iter: 反復回数)

前処理の種類	前処理計算	非定常反復法					
		GMRES(20)		GMRES(30)		GMRES(40)	
	time	time	iter	time	iter	time	iter
なし	0.0	1891.0	24083	2065.0	23455	2216.0	23131
SM 法 (tolU = tolV = 0.1)	1027.0	1034.0	24	1034.0	24	1033.0	24
SM 法 (tolU = tolV = 0.01)	1027.0	1035.0	24	1034.0	24	1034.0	24
MR 法 (tol = 0.1, imax = 1)	940.0	–	–	–	–	–	–
MR 法 (tol = 0.01, imax = 1)	940.0	–	–	–	–	–	–
MR 法 (tol = 0.1, imax = 2)	1882.0	–	–	–	–	–	–
MR 法 (tol = 0.01, imax = 2)	1894.0	–	–	–	–	–	–
MR 法 (tol = 0.1, imax = 3)	2825.0	–	–	–	–	–	–
MR 法 (tol = 0.01, imax = 3)	2853.0	–	–	–	–	–	–

–: 1 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 6: 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)

の 60%程度の速度向上が得られた. さらに, 数値例 1 と 2 同様に 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 6 に示す. 前処理なしの場合残差ノルムは収束するものの, その反復回数は Sherman-Morrison 法を適用したときよりも約 1000 倍余計にかかっている. また, MR 法による前処理を適用すると前処理の適用する時間は Sherman-Morrison 法とほぼ同じであるが, 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 法を適用した場合, 前処理計算だけで 17 分程度かかっている. 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見られており, 前処理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なしの場合よりも少ない. さらにこのような例について, 図 6 に GMRES(40) 法の残差ノルムの収束する様子を示す. 前処理なしで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 法を適用した GMRES(40) 法の残差

ノルムは途中の1000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1000秒付近まで前処理の計算をしており，この後24回の反復回数ですぐに収束したことが原因である．MR法による前処理を適用した場合，前処理の計算が終了した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1500秒付近から停滞しており前処理としての効果が発揮されていない．

5 結論

本稿では，Sherman-Morrison法により近似逆行列を求める手順を並列化する手法について提案した．行列分解を行う過程が逐次処理のため，この近似逆行列の計算の並列化はそれほど簡単ではなかった．しかし，行列分解において列ベクトルを $\tilde{m}$ 個だけ計算する度に通信を行い，このような計算と通信を交互に繰り返すことにより部分的な並列化が可能となり，その速度向上にも一定の効果が認められた．従って，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法による前処理計算を高速化することが可能である．

参考文献

[1] 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] Huckel. T.: Approximate Sparsity Patterns for the Inverse of a Matrix and Preconditioning, *Appl. Numer. Math.*, No. 30, pp. 291–303, (1999).

[4] Simoncini, Y.: On the Convergence of Restarted Krylov Subspace Method, *SIAM J. Matrix. Anal. Appl.*, Vol. 22, No. 2, pp. 430–452, (2000).

[5] Bru, R., Credán, J., Marín, J., Mas, J.: Preconditioning Sparse Nonsymmetric Linear systems with the Sherman-Morrison Formula, *SIAM J. Sci. Comput.*, Vol. 25, No. 2, pp. 701–715, (2003).

[6] University of Florida Sparse Matrix Collection. [Online] http://www.cise.ufl.edu/research/sparse/matrices.