数理解析研究所講究録
第 1733 巻 2011 年 180-191

Eisenstat 版前処理の実装とその改良

九州大学情報基盤研究開発センター　藤野清次 (Seiji Fujino)
Research Institute for Information Technology, Kyushu University
九州大学工学部電気情報工学科　村上啓一 (Keiichi Murakami)
九州大学大学院システム情報科学府情報学専攻　尾上勇介 (Yusuke Onoue)

1　はじめに

連立一次方程式の解を反復法を用いて求めるとき，前処理を用いることで収束性が大幅に改善される．しかし，前処理には反復法における 1 反復当りの計算量を増加するという問題がある．これに対して Eisenstat の提案した前処理つき反復法の実装では，前処理つき反復法における 1 反復あたりの計算量を前処理なしの場合とほぼ同程度まで削減することができる [3]．この前処理つき反復法は非常に優れた手法であるが，理論的な計算量から予想される計算時間と数値実験における実際の計算時間との間に大きな差が現れることがあった．そこで，本論文では，対称行列向けの解法と非対称行列向けの解法における Eisenstat trick の効果をメモリアクセス回数の観点から比較する．そして，理論的な計算時間の見積もりと実験結果との間に違いが現れる理由を明らかにする．

2　対称行列向けの解法における Eisenstat trick を用いた前処理

解くべき連立一次方程式を

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

とする．ただし，$A \in \mathbb{R}^{n\times n}$ を実対称正定値行列，$\boldsymbol{x} \in \mathbb{R}^n$ を解ベクトル，$\boldsymbol{b} \in \mathbb{R}^n$ を右辺ベクトルとし，(1) 式を SSOR 型前処理つき CG 法を用いて解くことを考える．SSOR 型前処理では，係数行列 A を

$$A = L + D + L^T \tag{2}$$

と分離し，前処理行列 M を

$$M = (L^T + D/\omega)D^{-1}(L + D/\omega) \tag{3}$$

とする [1]．ただし，L，D は各々係数行列の下三角行列，対角行列を意味する．この前処理行列をもちいて (1) 式に両側前処理を施すと，前処理後の係数行列 $\tilde{A}$，解ベクトル $\tilde{\boldsymbol{x}}$，右辺項ベクトル $\tilde{\boldsymbol{b}}$ は

$$\begin{aligned}\tilde{A} &= (L^T + D/\omega)^{-1}A(L + D/\omega)^{-1},\\ \tilde{\boldsymbol{x}} &= (L + D/\omega)\boldsymbol{x},\\ \tilde{\boldsymbol{b}} &= (L^T + D/\omega)^{-1}\boldsymbol{b}\end{aligned} \tag{4}$$

と各々表され，前処理後の残差ベクトル $\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x}$ は，

$$\begin{aligned}\tilde{\boldsymbol{r}} &= \tilde{\boldsymbol{b}} - \tilde{A}\tilde{\boldsymbol{x}}\\ &= (L^T + D/\omega)^{-1}\boldsymbol{b} - (L^T + D/\omega)^{-1}A(L + D/\omega)^{-1}(L + D/\omega)\boldsymbol{x}\\ &= (L^T + D/\omega)^{-1}(\boldsymbol{b} - A\boldsymbol{x})\\ &= (L^T + D/\omega)^{-1}\boldsymbol{r}\end{aligned} \tag{5}$$

である．以下前処理後の残差ベクトル $\tilde{\boldsymbol{r}}$ を変換残差と呼ぶ．前処理行列を適切に選べば，係数行列のスペクトル半径が改善と固有値の密集化により反復回数を大幅に減らすことができる，しかし，前処理を適用すると CG 法における反復 1 回当りの計算量は，もともと必要な行列ベクトル積 1 回に加え，前進・後退代入が各 1 回必要となり増加する．

Eisenstat は前処理行列が (3) 式の形のとき，Eisenstat trick と呼ばれる計算量削減技法を提案した [3][6]．まず，前処理後の係数行列を以下のように式変形する．

$$\begin{aligned}\tilde{A} &= (L^T + D/\omega)^{-1}A(L + D/\omega)^{-1}\\ &= (L^T + D/\omega)^{-1}(L^T + D/\omega + L + D/\omega + D - 2D/\omega)(L + D/\omega)^{-1}\\ &= (L + D/\omega)^{-1} + (L^T + D/\omega)^{-1}(I + (1 - 2/\omega)D(L + D/\omega)^{-1})\end{aligned} \tag{6}$$

Eisestat の提案した手法では，上の関係を用いて，行列 $\tilde{A}$ とベクトル $\boldsymbol{p}$ の積を以下の手順で計算する．

1. $\boldsymbol{y} = (L + D/\omega)^{-1}\boldsymbol{p}$
2. $\boldsymbol{z} = \boldsymbol{p} + (1 - 2/\omega)D\boldsymbol{y}$
3. $\boldsymbol{w} = (L^T + D/\omega)^{-1}\boldsymbol{z}$
4. $\tilde{A}\boldsymbol{p} = \boldsymbol{y} + \boldsymbol{w}$

この算法では $\tilde{A}$ と $\boldsymbol{p}$ の積を前進・後退代入各 1 回，対角行列とベクトルの積 1 回，ベクトルの和 2 回で行っている．通常 $\tilde{A}$ と $\boldsymbol{p}$ の積には行列ベクトル積 1 回と前進・後退代入が各々1 回必要であり，行列の非零要素数が対角要素の要素数に比べて極端に少ない場合を除けば，対角行列とベクトルの積やベクトルの和の計算量は行列ベクトル積計算に比べてコストが小さいため，この手法により計算量を削減することができる [5].

以下に Eisenstat trick を適用した SSOR 型前処理つき CG 法の算法を示す．

Eisenstat trick を適用した SSOR 型前処理つき CG 法の算法

1. Let $\boldsymbol{x}_0$ be an initial guess,
2. set $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$,
3. compute $\tilde{\boldsymbol{r}}_{k+1} = (L^T + D/\omega)^{-1}\boldsymbol{r}_0$,
4. $\boldsymbol{p}_0 = \tilde{\boldsymbol{r}}_0$,
5. for $k = 0, 1, \ldots,$
6. compute $(L + D/\omega)^{-1}\boldsymbol{p}_k$,
7. $\boldsymbol{q}_k = \boldsymbol{p}_k + (1 - 2/\omega)D(L + D/\omega)^{-1}\boldsymbol{p}_k$,
8. compute $(L^T + D/\omega)^{-1}\boldsymbol{q}_k$,
9. $\tilde{A}\boldsymbol{p}_k = (L + D/\omega)^{-1}\boldsymbol{p}_k + (L^T + D/\omega)^{-1}\boldsymbol{q}_k$,
10. $\alpha_k = \dfrac{(\tilde{\boldsymbol{r}}_k, \tilde{\boldsymbol{r}}_k)}{(\boldsymbol{p}_k, \tilde{A}\boldsymbol{p}_k)}$,
11. $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k(L + D/\omega)^{-1}\boldsymbol{p}_k$,
12. $\tilde{\boldsymbol{r}}_{k+1} = \tilde{\boldsymbol{r}}_k - \alpha_k\tilde{A}\boldsymbol{p}_k$,
13. compute $\boldsymbol{r}_k = (L^T + D/\omega)\tilde{\boldsymbol{r}}_{k+1}$,
14. if $\|\boldsymbol{r}_{k+1}\|_2/\|\boldsymbol{r}_0\|_2 \le \epsilon$ stop,
15. $\beta_k = \dfrac{(\tilde{\boldsymbol{r}}_{k+1}, \tilde{\boldsymbol{r}}_{k+1})}{(\tilde{\boldsymbol{r}}_k, \tilde{\boldsymbol{r}}_k)}$,
16. $\boldsymbol{p}_{k+1} = \tilde{\boldsymbol{r}}_{k+1} + \beta_k\boldsymbol{p}_k$,
17. end for.

上記の算法中では 6 行目と 8 行目，9 行目，および 13 行目の前進・後退代入計算であり，各々行列ベクトル積 0.5 回分の計算量を要する．これらのうち 13 行目の変換残差から残差の計算は収束判定のために行われており，毎回計算を行う必要がない．以下にその削減の手順を示す．

1. 要求誤差 ϵ_d と，それより少し大きな値 ϵ を用意する．
2. 反復開始から $\tilde{\boldsymbol{r}}_{k+1}\|_2/\|\tilde{\boldsymbol{r}}_0\|_2 \le \epsilon$ となるまで，変換残差から残差を求める計算を行わない．
3. $\tilde{\boldsymbol{r}}_{k+1}\|_2/\|\tilde{\boldsymbol{r}}_0\|_2 \le \epsilon$ となった後も，計算量削減のために収束判定を m 回ごとに行う．
4. 最終的に $\|\boldsymbol{r}_{k+1}\|_2/\|\boldsymbol{r}_0\|_2 \le \epsilon$ を満たしたとき反復計算を終了する．

この手法では，m を大きくするとより大きな計算量削減効果が期待できるが，m を大きくしすぎると相対残差が要求する精度以下となっても少しだけ余分の反復を続けるが実無視できる程度である．上記の計算量削減手法を適用した前処理を Tri(Triangular，三角方程式) 型前処理と呼ぶ．以下に Tri 型前処理つき CG 法の算法を示す．

Tri 前処理つき CG 法の算法

1. Let $\boldsymbol{x}_0$ be an initial guess,
2. set $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$,
3. compute $\tilde{\boldsymbol{r}}_{k+1} = (L^T + D/\omega)^{-1}\boldsymbol{r}_0$,
4. $\boldsymbol{p}_0 = \tilde{\boldsymbol{r}}_0$,
5. for $k = 0, 1, \ldots,$
6. compute $(L + D/\omega)^{-1}\boldsymbol{p}_k$,

$\boldsymbol{q}_k = \boldsymbol{p}_k + (1-2/\omega)D(L+D/\omega)^{-1}\boldsymbol{p}_k$,
compute $(L^T + D/\omega)^{-1}\boldsymbol{q}_k$,
$\tilde{A}\boldsymbol{p}_k = (L+D/\omega)^{-1}\boldsymbol{p}_k + (L^T+D/\omega)^{-1}\boldsymbol{q}_k$,
$\alpha_k = \dfrac{(\tilde{\boldsymbol{r}}_k, \tilde{\boldsymbol{r}}_k)}{(\boldsymbol{p}_k, \tilde{A}\boldsymbol{p}_k)}$,
$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k (L+D/\omega)^{-1}\boldsymbol{p}_k$,
$\tilde{\boldsymbol{r}}_{k+1} = \tilde{\boldsymbol{r}}_k - \alpha_k \tilde{A}\boldsymbol{p}_k$,
if $\|\tilde{\boldsymbol{r}}_{k+1}\|_2 / \|\tilde{\boldsymbol{r}}_0\|_2 \le \epsilon$ then,
  if $\mathrm{mod}(k, m) = 0$ then,
    compute $\boldsymbol{r}_k = (L^T + D/\omega)\tilde{\boldsymbol{r}}_{k+1}$,
    if $\|\boldsymbol{r}_{k+1}\|_2 / \|\boldsymbol{r}_0\|_2 \le \epsilon$ stop,
  end if,
end if,
$\beta_k = \dfrac{(\tilde{\boldsymbol{r}}_{k+1}, \tilde{\boldsymbol{r}}_{k+1})}{(\tilde{\boldsymbol{r}}_k, \tilde{\boldsymbol{r}}_k)}$,
$\boldsymbol{p}_{k+1} = \tilde{\boldsymbol{r}}_{k+1} + \beta_k \boldsymbol{p}_k$,
end for.

3 非対称行列向けの解法における Eisenstat trick を用いた前処理

(1) 式において，係数行列が非対称の場合の解法に Tri 型前処理を適用することを考える．非対称行列向けの解法として，ここでは BiCGStab 法を用いる．BiCGStab 法の算法を以下に示す．

BiCGStab 法の算法

Let $\boldsymbol{x}_0$ be an initial guess, and put $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$,
choose $\boldsymbol{r}_0^*$ such that $(\boldsymbol{r}_0, \boldsymbol{r}_0^*) \neq 0$,
set $\boldsymbol{p}_0 = \boldsymbol{r}_0$,
for $k = 0, 1, \dots$,
  compute $A\boldsymbol{p}_k$,
  $\alpha_k = \dfrac{(\boldsymbol{r}_k, \boldsymbol{r}_0^*)}{(A\boldsymbol{p}_k, \boldsymbol{r}_0^*)}$,
  $\boldsymbol{t}_k = \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k$,
  compute $A\boldsymbol{t}_k$,
  $\zeta_k = \dfrac{(A\boldsymbol{t}_k, \boldsymbol{t}_k)}{(A\boldsymbol{t}_k, A\boldsymbol{t}_k)}$,
  $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k + \zeta_k \boldsymbol{t}_k$,
  $\boldsymbol{r}_{k+1} = \boldsymbol{t}_k - \zeta_k A\boldsymbol{t}_k$,
  if $\|\boldsymbol{r}_{k+1}\|_2 / \|\boldsymbol{r}_0\|_2 \le \epsilon$ stop,
  $\beta_k = \dfrac{\alpha_k}{\zeta_k} \dfrac{(\boldsymbol{r}_{k+1}, \boldsymbol{r}_0^*)}{(\boldsymbol{r}_k, \boldsymbol{r}_0^*)}$,
  $\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k (\boldsymbol{p}_k - \zeta_k A\boldsymbol{p}_k)$,
end for.

係数行列 A を対称行列の場合と同様に

$$A = L + D + U \tag{7}$$

と分離する．ここで，U は係数行列の狭義上三角行列を意味する．前処理行列を，

$$M = (L + D/\omega)D^{-1}(U + D/\omega) \tag{8}$$

と置き，両側前処理後の係数行列を以下のように変形する [2].

$$\tilde{A} = (U + D/\omega)^{-1} A (L + D/\omega)^{-1}$$

$$= (L + D/\omega)^{-1} + (U + D/\omega)^{-1}(I + (1 - 2/\omega)D(L + D/\omega)^{-1}) \tag{9}$$

Tri 型前処理付き BiCGStab 法の算法を以下に示す.

Eisenstat trick を適用した GS 型前処理付き BiCGStab 法の算法

Let $\tilde{\boldsymbol{x}}_0$ be an initial guess, and put $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$,
choose $\boldsymbol{r}_0^*$ such that $(\boldsymbol{r}_0, \boldsymbol{r}_0^*) \neq 0$,
compute $\tilde{\boldsymbol{r}}_0 = (U + I)^{-1}\boldsymbol{r}_0$,
set $\boldsymbol{p}_0 = \tilde{\boldsymbol{r}}_0$,
for $k = 0, 1, \ldots,$
  compute $(L + D)^{-1}\boldsymbol{p}_k$,
  $\boldsymbol{q}_k = \boldsymbol{p}_k + (1 - 2/\omega)D(L + I)^{-1}\boldsymbol{p}_k$,
  compute $(U + D/\omega)^{-1}\boldsymbol{q}_k$,
  $\tilde{A}\boldsymbol{p}_k = (L + D\omega)^{-1}\boldsymbol{p}_k + (U + D/\omega)^{-1}\boldsymbol{q}_k$,
  $\alpha_k = \dfrac{(\tilde{\boldsymbol{r}}_k, \boldsymbol{r}_0^*)}{(\tilde{A}\boldsymbol{p}_k, \boldsymbol{r}_0^*)}$,
  $\boldsymbol{t}_k = \tilde{\boldsymbol{r}}_k - \alpha_k \tilde{A}\boldsymbol{p}_k$,
  compute $(L + D/\omega)^{-1}\boldsymbol{t}_k$,
  $\boldsymbol{u}_k = \boldsymbol{t}_k + (1 - 2/\omega)D(L + D/\omega)^{-1}\boldsymbol{t}_k$,
  compute $(U + D/\omega)^{-1}\boldsymbol{u}_k$,
  $\tilde{A}\boldsymbol{t}_k = (L + D/\omega)^{-1}\boldsymbol{t}_k + (U + D/\omega)^{-1}\boldsymbol{u}_k$,
  $\zeta_k = \dfrac{(\tilde{A}\boldsymbol{t}_k, \boldsymbol{t}_k)}{(\tilde{A}\boldsymbol{t}_k, \tilde{A}\boldsymbol{t}_k)}$,
  $\tilde{\boldsymbol{x}}_{k+1} = \tilde{\boldsymbol{x}}_k + \alpha_k \boldsymbol{p}_k + \zeta_k \boldsymbol{t}_k$,
  $\tilde{\boldsymbol{r}}_{k+1} = \boldsymbol{t}_k - \zeta_k \tilde{A}\boldsymbol{t}_k$,
  if $\|\tilde{\boldsymbol{r}}_{k+1}\|_2 / \|\tilde{\boldsymbol{r}}_0\|_2 \leq \epsilon$ then,
    if $\mathrm{mod}(k, m) = 0$ then,
      compute $\boldsymbol{r}_{k+1} = (U + D/\omega)\tilde{\boldsymbol{r}}_{k+1}$,
      if $\|\boldsymbol{r}_{k+1}\|_2 / \|\boldsymbol{r}_0\|_2 \leq \epsilon_d$ stop,
    end if,
  end if,
  $\beta_k = \dfrac{\alpha_k}{\zeta k} \dfrac{(\tilde{\boldsymbol{r}}_{k+1}, \boldsymbol{r}_0^*)}{(\tilde{\boldsymbol{r}}_{k+1}, \boldsymbol{r}_0^*)}$,
  $\boldsymbol{p}_{k+1} = \tilde{\boldsymbol{r}}_{k+1} + \beta_k(\boldsymbol{p}_k - \zeta_k \tilde{A}\boldsymbol{p}_k)$,
end for.

4 対称・非対称行列における計算量とメモリアクセス回数の比較

表 1 に前処理なしの場合と Eisenstat trick を用いた場合の CG 法, および BiCGStab 法における 1 反復当りの計算量の見積もりを示す. “nnz” は係数行列の非零要素数, “n” は行列の次元数, “$Lnnz$” は狭義下三角部分の非零要素数, “$Unnz$” は狭義上三角部分の非零要素数を各々意味する.

極端に疎な行列を除いて非零要素数 nnz は次元数 n に比べて大きい値であるため, m を十分大きくとれば表 1 より Eisenstat trick を適用した前処理つきの場合の 1 反復あたりの計算量は前処理なしの場合と同程度となることがわかる. したがって 1 反復あたりの計算時間も前処理をつけた場合と前処理なしの場合とが近い値となることが予想される. しかし, 実際には計算量の見積もりと数値実験の結果との間には差があり, これは計算量表 1 の計算量の見積もりだけでは説明することができない.

以下では, Eisenstat trick を用いた前処理つき反復法における理論的な計算時間の見積もりと実際の計算時間との間に差が現れる理由を明らかにするために, 反復法におけるメモリアクセス回数を評価に加えることを考える. まず, CG 法における疎行列ベクトル積 疑似 Fortran プログラムを以下に示す. ただし行列は CRS 形式で下三角行列のみが格納されているものとする.

表 1: 前処理なし CG 法，Eisenstat trick つき CG 法および BiCGStab 法における 1 反復当りの計算量.

行列	対称行列		非対称行列	
解法	CG 法		BiCGStab	
内訳 \ 前処理	-	Tri	-	Tri
Av	$2nnz$	0	$4nnz$	0
Lv	0	0	0	0
$L^{-1}v$	0	$nnz+n$	0	$4(Lnnz+n)$
Uv	0	$(nnz+n)/m$	0	$2(Unnz+n)/m$
$U^{-1}v$	0	$nnz+n$	0	$4(Unnz+n)$
(v,w)	$6n$	$(4+2/m)n$	$10n$	$(8+2/m)n$
$v\pm w$	$3n$	$5n$	$6n$	$10n$
αv	$3n$	$4n$	$6n$	$8n$
合計	$2nnz$ $+12n$	$(2+1/m)nnz$ $+(15+3/m)n$	$4nnz$ $+22n$	$(4+2\frac{Unnz}{m\cdot nnz})nnz$ $+(30+4/m)n$

プログラム 1:　対称行列における疎行列ベクトル積 疑似 Fortran プログラム

```
1.  do i = 1, ncol
2.  tmp = val(rowptr(i)) * p(i)
3.  tmp2 = p(i)
4.    do j = rowptr(i), rowptr(i + 1) - 2
5.      tmp = tmp + val(j) * p(colind(j))
6.      Ap(colind(j)) = Ap(colind(j)) + val(j) * tmp2
7.    enddo
8.    Ap(i) = Ap(i) + tmp2
9.  enddo
```

プログラム 1 において，5 行目で下三角行列とベクトルの積を行方向に，6 行目で上三角行列とベクトルの積を列方向に計算する．セミナーで尾上さんが指摘されたように，行列が対称の場合，対称性を利用して 2 方向の計算を同じループで処理することができ，行列の要素を格納する配列 val へのアクセス回数を削減することが可能である．これに対して，対称行列における Eisenstat trick を用いた疎行列ベクトル積疑似プログラムを以下に示す．

プログラム 2:　対称行列における Eisenstat trick を適用した疎行列ベクトル積 疑似 Fortran プログラム

```
omega2 = 1.0d0 - 2.0d0/omega
do i = 1, ncol
  tmp = p(i)
  do j = rowptr(i), rowptr(i + 1) - 2
    tmp = tmp - val(j) * y(colind(j))
  enddo
  y(i) = tmp * omega * pivot(i)
  z(i) = p(i) + omega2 * y(i)
enddo
w = z
do i = ncol, 1, -1
  w(i) = w(i) * omega * pivot(i)
  tmp = w(i)
  do j = rowptr(i), rowptr(i + 1) - 2
    w(colind(j)) = w(colind(j)) - val(j) * tmp
  enddo
  Ap(i) = y(i) + w(i)
enddo
```

プログラム 2 において 1 行目から 9 行目までが TriCG 法の算法における

$$
\begin{aligned}
&\text{compute } (L + D/\omega)^{-1}\boldsymbol{p}_k,\\
&\boldsymbol{q}_k = \boldsymbol{p}_k + (1 - 2/\omega)D(L + D/\omega)^{-1}\boldsymbol{p}_k,
\end{aligned}
$$

の計算に該当し，10 行目から 18 行目までが

$$
\begin{aligned}
&\text{compute } (L^T + D/\omega)^{-1}\boldsymbol{q}_k,\\
&A\boldsymbol{p}_k = (L + D/\omega)^{-1}\boldsymbol{p}_k + (L^T + D/\omega)^{-1}\boldsymbol{q}_k,
\end{aligned}
$$

の計算に該当する．アルゴリズム中の“ $A\boldsymbol{p}_k$ ”，“ $(L + D/\omega)^{-1}\boldsymbol{p}_k$ ”，“ $(L^T + D/\omega)^{-1}\boldsymbol{q}_k$ ”，“ $(1 - 2/\omega)$ ”はプログラム中で，配列“ Ap ”，“ lp ”，“ lq ”，および変数“ omega2 ”に各々対応する．

プログラム 2 では，1 行目から 9 行目で $\boldsymbol{p}_k + (1 - 2/\omega)(L + D/\omega)^{-1}\boldsymbol{p}_k$ の計算により求まるベクトルを配列 q に格納する．このとき配列 q は 1 番目の要素から行列の次元数である ncol 番目の要素まで順番に求まってゆくが，10 行目以降の $(L^T + D/\omega)^{-1}\boldsymbol{q}_k$ の計算では配列 q を ncol 番目の要素から 1 番目の要素まで逆順に参照するため，プログラム 1 のように 2 方向の計算を同時にするということができない．プログラム 1 の 5 行目，6 行目の計算がプログラム 2 では 5 行目，15 行目にあたり，各々で配列 val の値をメモリからロードするためプログラム 1 に比べてプログラム 2 におけるメモリアクセス回数は増加する．

非対称行列の場合，プログラム 1 のように行列の対称性を利用してメモリアクセス回数を削減することはできない．行列を CRS 形式で格納した場合の，非対称行列における疎行列ベクトル積疑似 Fortran プログラムを以下に示す．

プログラム 3:　非対称行列における疎行列ベクトル積 疑似 Fortran プログラム

```
1. do i = 1, ncol
2. tmp = 0.0d0
3.    do j = rowptr(i), rowptr(i + 1) - 1
4.       tmp = tmp + val(j) * p(colind(j))
5.    enddo
6.    Ap(i) = tmp
7. enddo
```

プログラム 1 中で 5 行目と 6 行目の 2 行で行われていた $A\boldsymbol{p}$ の計算は，非対称行列の場合プログラム 3 において 4 行目で行われる．非対称行列における Eisenstat trick を適用した疎行列ベクトル積疑似 Fortran プログラムを以下に示す．

プログラム 4:　非対称行列における Eisenstat trick を適用した疎行列ベクトル積 疑似 Fortran プログラム

```
omega2 = 1.0d0 - 2.0d0/omega
do i = 1, ncol
   tmp = p(i)
   do j = lrowptr(i), lrowptr(i + 1) - 1
      tmp = tmp - lval(j) * y(lcolind(j))
   enddo
   y(i) = tmp * omega * pivot(i)
enddo
do i = ncol, 1, -1
   tmp = p(i) + omega2 * y(i)
   do j = urowptr(i), urowptr(i + 1) - 1
      tmp = tmp - uval(j) * w(ucolind(j))
   enddo
   w(i) = omega * tmp * pivot(i)
   Ap(i) = y(i) + w(i)
enddo
```

行列が非対称の場合には，対称行列の場合に比べて Eisenstat trick 適用によるメモリアクセス回数の増加は小さい．これは対称行列における疎行列ベクトル積計算でのメモリアクセス回数の削減を非対称行列に適用できないためである．表 2 に前処理なしの場合と Eisenstat trick を用いた場合の反復法における 1 反復当りのメモリアクセス回数の見積もりを示す．

表 2: 前処理なし CG 法と Eisenstat trick つき CG 法および BiCGStab 法における 1 反復当りのメモリアクセス回数．

行列	対称行列		非対称行列	
解法	CG 法		BiCGStab 法	
内訳 \ 前処理	-	Tri	-	Tri
$A\boldsymbol{v}$	$4nnz$ $+n$	0	$6nnz$ $+2n$	0
$L\boldsymbol{v}$ $U\boldsymbol{v}$	0	$(3nnz+n)/2m$	0	$3(Unnz+n)/m$
$L^{-1}\boldsymbol{v}$ $U^{-1}\boldsymbol{v}$	0	$6nnz+8n$	0	$6nnz+16n$
$(\boldsymbol{v},\boldsymbol{w})$ $\boldsymbol{v}\pm\alpha\boldsymbol{w}$	$6n$	$6n$	$17n$	$17n$
合計	$4nnz$ $+10n$	$(6+1.5/m)nnz$ $+(14+0.5/m)n$	$6nnz$ $+19n$	$(6+\frac{3Unnz}{m\cdot nnz})nnz$ $+(33+3/m)n$

5 数値実験

対称行列に対して，前処理なしの CG 法，および加速 IC(0)，SSOR，Tri の 3 種類の前処理つき CG 法の合計 4 種類の解法の収束性を調べた．非対称に対しては，前処理なし BiCGStab 法，および ILU(0)，Tri の 2 種類の前処理つき BiCGStab 法の合計 3 種類の解法の収束性を調べた．

5.1 計算機環境と計算条件

計算機環境

1. CPU：Intel Xeon X5570 (2.93GHz, 8MB L2 Cache. 4cores) ×2，メモリ：24GB，OS：RedHat Enterprize Linux 5.2.
2. プログラム：Fortran90，コンパイラ：Portland Group Fortran 90 compiler ver.10.5.
3. 最適化オプションは“-O3”を使用した．
4. 計算はすべて倍精度浮動小数点演算で行い，時間計測には時間計測関数 getrusage を用いた．

計算条件

1. 収束判定値は相対残差の 2 ノルム：$\|\boldsymbol{r}_{k+1}\|_2/\|\boldsymbol{r}_0\|_2 \leq 10^{-8}$ とした．
2. Tri において ϵ_d による収束判定は相対残差の 2 ノルム：$\|\boldsymbol{r}_{k+1}\|_2/\|\boldsymbol{r}_0\|_2 \leq 10^{-8}$ とし，$\tilde{\boldsymbol{r}}$ と ϵ による判定は変換残差の 2 ノルム：$\|\tilde{\boldsymbol{r}}_{k+1}\|_2/\|\tilde{\boldsymbol{r}}_0\|_2 \leq 10^{-6}$ とした．
3. 右辺項は厳密解が $\hat{\boldsymbol{x}} = (1,\ldots,1)^T$ となるように $\boldsymbol{b} = A\hat{\boldsymbol{x}}$ と定めた．
4. 初期近似解 $\boldsymbol{x}_0$ はすべて 0，最大反復回数は行列の次元数が 10000 以下のとき 10000 回，それより大きいときは次元数と等しい回数とした．
5. 行列は予め対角スケーリングによって対角項をすべて 1 に正規化した．
6. SSOR，Tri のパラメータ ω は 0.5 から 1.5 まで 0.1 刻みで変化させた．
7. パラメータ m は 5，10，15，20 の 4 通りについて実験を行った．

5.2 テスト行列

表 3 に対称行列 17 個の特徴を，表 4 に非対称行列 17 個の特徴を非零要素数の大きいものから順に示す．表 3 中の"平均非零要素数"は非零要素数を次元数で割った値を意味する．行列 ft21a1，ft21a2，ft21a3，ft21b3 は鹿島建設 (株)，tikatan，tikatan_big は九州大学工学部牛島教授，cla-pla1000-1 は鹿児島大学工学部 岡田教授，BEAM，CABLE，s2 は (株) ホクトシステム各々提供された行列，表 4 中の行列 air-cfl5 はマンチェスター大学 F. Costen 講師，行列 waseda は早稲田大学若尾教授より各々提供された行列である．その他の行列はフロリダ大学の疎行列データベース [7] から選出した．

表 3: 対称行列の特徴.

行列	次元数	非零要素数	平均非零要素数
tikatan_big	400,221	5,456,441	13.6
s3dkq4m2	90,449	4,427,725	49.0
s3dkt3m2	90,449	3,686,223	40.8
ct20stif	52,329	2,600,295	49.7
CABLE	59,002	1,986,094	33.7
ft21a2	41,483	1,404,593	33.9
ft21a1	41,483	1,391,371	33.5
ft21a3	41,483	1,391,320	33.5
ft21b3	36,575	1,290,353	35.3
msc23052	23,052	1,142,686	49.6
tikatan	35,301	464,201	13.1
cla-pla1000-1	6,475	306,512	47.3
BEAM	10,626	233,268	22.0
bodyy5	18,589	128,853	6.9
bcsstk15	3,948	117,816	29.8
nasa4704	4,704	104,756	22.3
s2	19,800	67,036	3.4

表 4: 非対称行列の特徴.

行列	次元数	非零要素数	平均非零要素数
waseda	19,060	24,377,548	1278.99
air-cfl5	1,536,000	19,435,428	12.65
xenon2	157,464	3,866,688	24.56
poisson3Db	85,623	2,374,949	27.74
matrix_9	103,430	2,121,550	20.51
language	399,130	1,216,334	3.05
xenon1	48,600	1,181,120	24.30
dc2	116,835	766,396	6.56
dc3	116,835	766,396	6.56
poisson3Da	13,514	352,762	26.10
raefsky2	3,242	293,551	90.55
chipcool0	20,082	281,150	14.00
chipcool1	20,082	281,150	14.00
wang4	26,068	177,196	6.80
Zhao1	33,861	166,453	4.92
comsol	1,500	97,645	65.10
big	13,209	91,465	6.92

5.3 実験結果

表 5 に対称行列に対する 4 種類の前処理つき CG 法の収束性を，表 6 に非対称行列に対する 3 種類の前処理つき BiCGStab 法の収束性を各々示す．表中の " 平均時間 " は合計時間を反復回数で割った値を，" ratio1 " は前処理なし CG 法，および前処理なし BiCGStab 法の合計時間に対する各前処理つきの場合の合計時間の比を，" ratio2 " は前処理なし CG 法，および前処理なし BiCGStab 法の平均時間に対する各前処理つきの場合の平均時間の比を各々意味する．時間単位は " 平均時間 " の欄を除いてすべて秒とする．

対称行列に対する実験に対して，合計時間について表 5 より以下の知見が得られる．

- 17 ケース中 15 ケースで Tri 前処理が最速結果を示した．
- 行列 cla-pla1000-1 に対して前処理なし CG 法が，bodyy5 に対して加速 IC(0) 前処理つき CG 法がそれぞれ最速結果を示した．

また，平均時間を比較すると以下の考察が得られる．

- 行列 bodyy5 を除く 16 ケースで，加速 IC(0) と SSOR 前処理により平均時間が CG 法の 2 倍以上となっているのに対し，Tri 前処理により平均時間が CG 法の 2 倍以上となったのは行列 tikatan_big のみであった．
- 行列は非零要素数の多い順としており，非零要素数の多い行列は，少ない行列に比べて前処理をつけることによる平均時間の増加幅が大きかった．

非対称行列に対する実験に対して，合計時間について表 6 より以下の知見が得られる．

- 17 ケース中 16 ケースで Tri 前処理つき BiCGStab 法が最速結果を示した．
- 行列 comsol に対して前処理なしの BiCGStab 法が最速結果を示した．

平均時間について，

- 前処理なし BiCGStab 法と比較して ILU(0) 前処理をつけた場合は平均時間が 2 倍以上となっているのに対し，Tri 前処理では最大 1.65 倍，最少 1.01 倍であり，前処理による 1 反復当りの計算時間の増加が小さかった．
- 平均非零要素数の少ない行列に対して，Tri 前処理つき BiCGStab 法における ratio2 の値が小さくなる傾向を示した．
- 特に，平均非零要素数の極端に大きな行列 waseda において，Tri 前処理つき BiCGStab 法の 1 反復あたりの計算時間は前処理なしの場合の 1.01 倍であり，前処理なしとほぼ同じ時間となった．

対称行列に対する実験結果と非対称行列に対する実験結果を比較すると，全体的に非対称行列における Tri 前処理による前処理なしの場合に対する平均時間の増加は対称行列の場合に比べて小さくなっていることがわかる．

表 7 に Tri 型前処理つき CG 法の前処理なしに対する平均時間比における理論値と実測値の比較を，表 8 に Tri 型前処理つき BiCGStab 法の前処理なしに対する平均時間比における理論値と実測値の比較を各々示す．表中の " 実測値 " は表 5，表 6 における " ratio2 " 欄の値，" 計算量 "，" メモリアクセス " 欄の " 理論値 " は 3 節の表 1，および表 2 より，前処理なしの場合に対する 1 反復当りの計算量とメモリアクセス回数の見積もりの Tri 前処理を用いた場合の比を各行列ごとに具体的に求めた値，" 差 " は実測値と理論値の差の絶対値を各々示す．

表 7 より，

- 計算量の理論値は実測値との差が 0.5 以上となる行列が行列 BEAM を除く 16 ケース中 9 ケースであったのに対し，メモリアクセス回数の理論値は行列 tikatan_big，bodyy5 を除いて，実測値との差が 0.2 以下と小さい値を示した．
- 行列 tikatan_big において理論値と実測値との差が他の行列に比べて大きな値となった．これは行列の非零要素数が 5,456,441 と非常に大きいためキャッシュミスが増加し，さらに，平均非零要素数が小さいため Eisenstat trick による計算量削減効果が小さかったためと考えられる．

表 8 より，

- 行列 matrix_9 を除く 16 ケースのうち，行列 waseda 以外の 15 ケースでメモリアクセス回数に基づく理論値は計算量に基づく理論値よりも実測値との一致度が良かった．
- 非対称行列の場合は，対称行列でおこるメモリアクセス回数の大幅な増加がないため，計算量に基づく理論値とメモリアクセス回数に基づく理論値との間の差が対称行列の場合よりも小さかった．

以上より，Eisenstat trick を用いた前処理つき反復法の計算時間の指標として，1 反復あたりのメモリアクセス回数から求めた値の方が計算量より求めた値よりも正確であることがわかる．

6 まとめ

Eisenstat trick を用いた前処理つき反復法について，計算量の削減と，メモリアクセス回数の見積もりによる評価の二つを行った．その結果，改良版の SSOR 型前処理は加速 IC(0) や ILU(0) 型の前処理つきよりも計算時間を短縮することができ，前処理として有効であることがわかった．また，Eisenstat trick は対称行列の場合と非対称行列の場合とで 1 反復あたりの計算時間に対する影響が異なり，メモリアクセス回数による計算時間の評価によりこれを説明できることがわかった．

表 5: 対称行列に対する 4 種類の前処理つき CG 法の収束性.

行列	前処理	ω	m	反復回数	前処理時間	反復時間	合計時間	ratio1	平均時間 [ms]	ratio2
tikatan_big	-	-	-	242	-	-	3.88	1.00	16.05	1.00
	加速 IC(0)	1.00	-	102	0.02	4.18	4.19	1.08	40.95	2.55
	SSOR	1.50	-	54	-	-	2.04	0.53	37.83	2.36
	Tri	1.50	5	55	-	-	**1.93**	0.50	35.12	2.19
s3dkq4m2	-	-	-	28,728	-	-	176.12	1.00	6.13	1.00
	加速 IC(0)	1.62	-	11,430	0.19	181.59	181.77	1.03	15.89	2.59
	SSOR	1.00	-	10,411	-	-	154.72	0.88	14.86	2.42
	Tri	1.00	20	10,420	-	-	**108.18**	0.61	10.38	1.69
s3dkt3m2	-	-	-	40,614	-	-	217.58	1.00	5.36	1.00
	加速 IC(0)	1.82	-	14,108	0.21	192.21	192.42	0.88	13.62	2.54
	SSOR	1.00	-	14,712	-	-	186.82	0.86	12.70	2.37
	Tri	1.00	20	14,720	-	-	**126.68**	0.58	8.61	1.61
ct20stif	-	-	-	26,108	-	-	92.21	1.00	3.53	1.00
	加速 IC(0)	2.40	-	8,720	0.23	77.72	77.94	0.85	8.91	2.52
	SSOR	0.70	-	8,440	-	-	69.55	0.75	8.24	2.33
	Tri	0.70	20	8,440	-	-	**48.55**	0.53	5.75	1.63
CABLE	-	-	-	7,035	-	-	33.91	1.00	4.82	1.00
	加速 IC(0)	2.06	-	2,941	0.24	36.83	37.06	1.09	12.52	2.60
	SSOR	1.00	-	1,863	-	-	21.61	0.64	11.60	2.41
	Tri	1.00	5	1,865	-	-	**14.95**	0.44	8.01	1.66
ft21a2	-	-	-	1,334	-	-	4.53	1.00	3.40	1.00
	加速 IC(0)	1.42	-	503	0.07	4.39	4.45	0.98	8.73	2.57
	SSOR	1.30	-	319	-	-	2.55	0.56	7.98	2.35
	Tri	1.20	20	320	-	-	**1.75**	0.39	5.48	1.61
ft21a1	-	-	-	537	-	-	1.81	1.00	3.36	1.00
	加速 IC(0)	1.32	-	346	0.05	3.01	3.07	1.70	8.71	2.59
	SSOR	1.50	-	193	-	-	1.52	0.84	7.89	2.35
	Tri	1.50	15	195	-	-	**1.06**	0.59	5.43	1.61
ft21a3	-	-	-	1,972	-	-	6.67	1.00	3.38	1.00
	加速 IC(0)	1.42	-	1,446	0.06	12.51	12.57	1.89	8.65	2.56
	SSOR	1.30	-	794	-	-	6.33	0.95	7.97	2.36
	Tri	1.30	15	795	-	-	**4.34**	0.65	5.46	1.61
ft21b3	-	-	-	1,712	-	-	5.27	1.00	3.08	1.00
	加速 IC(0)	1.42	-	1,022	0.06	8.06	8.12	1.54	7.89	2.56
	SSOR	1.30	-	524	-	-	3.75	0.71	7.16	2.33
	Tri	1.40	20	520	-	-	**2.57**	0.49	4.94	1.60
msc23052	-	-	-	23,053	-	-	34.27	1.00	1.49	1.00
	加速 IC(0)	2.88	-	11,847	0.11	39.95	40.06	1.17	3.37	2.27
	SSOR	0.70	-	8,016	-	-	23.96	0.70	2.99	2.01
	Tri	0.80	20	8,060	-	-	**16.79**	0.49	2.08	1.40
tikatan	-	-	-	113	-	-	0.13	1.00	1.15	1.00
	加速 IC(0)	1.00	-	76	0.00	0.20	0.20	1.57	2.68	2.33
	SSOR	1.50	-	39	-	-	0.10	0.74	2.46	2.14
	Tri	1.50	5	40	-	-	**0.07**	0.52	1.68	1.46
cla-pla1000-1	-	-	-	249	-	-	**0.16**	1.00	0.64	1.00
	加速 IC(0)	2.46	-	387	0.04	0.59	0.63	3.94	1.52	2.38
	SSOR	0.80	-	298	-	-	0.39	2.45	1.31	2.05
	Tri	0.80	10	310	-	-	0.28	1.79	0.92	1.44
BEAM	-	-	-	max	-	-	-	-	-	-
	加速 IC(0)	2.48	-	7,329	0.03	9.50	9.53	-	1.30	-
	SSOR	1.00	-	6,138	-	-	7.05	-	1.15	-
	Tri	0.90	15	6,060	-	-	**4.89**	-	0.81	-
bodyy5	-	-	-	86	-	-	0.030	1.00	0.35	1.00
	加速 IC(0)	1.00	-	41	0.00	0.03	**0.025**	0.83	0.61	1.75
	SSOR	1.40	-	98	-	-	0.055	1.83	0.56	1.61
	Tri	1.30	20	100	-	-	0.040	1.33	0.40	1.15
bcsstk15	-	-	-	530	-	-	0.083	1.00	0.16	1.00
	加速 IC(0)	1.82	-	251	0.01	0.09	0.099	1.19	0.37	2.38
	SSOR	1.00	-	182	-	-	0.061	0.73	0.34	2.13
	Tri	1.00	5	185	-	-	**0.044**	0.53	0.24	1.52
nasa4704	-	-	-	4,829	-	-	0.76	1.00	0.16	1.00
	加速 IC(0)	2.48	-	2,201	0.01	0.83	0.84	1.11	0.38	2.39
	SSOR	1.00	-	1,622	-	-	0.56	0.73	0.34	2.18
	Tri	1.00	5	1,625	-	-	**0.39**	0.51	0.24	1.52
s2	-	-	-	1,311	-	-	2.01	1.00	1.53	1.00
	加速 IC(0)	1.18	-	604	0.01	2.22	2.23	1.11	3.68	2.40
	SSOR	1.50	-	359	-	-	1.18	0.59	3.28	2.14
	Tri	1.40	20	360	-	-	**0.79**	0.39	2.20	1.44

表 6: 非対称行列に対する 3 種類の前処理つき BiCGStab 法の収束性.

行列	前処理	ω	m	反復回数	前処理時間	反復時間	合計時間	ratio1	平均時間 [ms]	ratio2
waseda	-	-	-	2,055	0.06	171.50	171.56	1.00	83.45	1.00
	ILU(0)	-	-	28	115.96	5.25	121.21	0.71	187.50	2.25
	Tri	0.9	5	55	0.09	4.55	**4.64**	0.03	84.36	1.01
air-cfl5	-	-	-	35	0.07	3.66	3.73	1.00	104.57	1.00
	ILU(0)	-	-	26	1.55	6.06	7.61	2.04	233.08	2.23
	Tri	1.2	15	15	0.13	2.09	**2.22**	0.60	148.13	1.42
xenon2	-	-	-	930	0.01	15.13	15.14	1.00	16.27	1.00
	ILU(0)	-	-	break	-	-	-	-	-	-
	Tri	0.9	5	335	0.02	6.71	**6.73**	0.44	20.10	1.24
poisson3Db	-	-	-	254	0.01	3.33	3.34	1.00	13.11	1.00
	ILU(0)	-	-	79	0.53	2.56	3.09	0.93	32.41	2.47
	Tri	1.4	5	85	0.02	1.55	**1.56**	0.47	18.40	1.40
matrix_9	-	-	-	max	-	-	-	-	-	-
	ILU(0)	-	-	54	0.15	1.07	**1.22**	-	19.81	-
	Tri	0.7	10	450	0.01	4.98	5.00	-	11.10	-
language	-	-	-	25	0.01	0.40	0.41	1.00	16.00	1.00
	ILU(0)	-	-	6	0.13	0.23	0.36	0.88	38.33	2.40
	Tri	1.1	10	10	0.01	0.23	**0.24**	0.59	24.00	1.50
xenon1	-	-	-	766	0.00	3.74	3.74	1.00	4.88	1.00
	ILU(0)	-	-	break	-	-	-	-	-	-
	Tri	1.0	5	265	0.01	1.62	**1.62**	0.43	6.12	1.25
dc2	-	-	-	342	0.00	1.56	1.56	1.00	4.56	1.00
	ILU(0)	-	-	96	0.06	1.04	1.10	0.71	10.83	2.38
	Tri	1.0	5	85	0.01	0.54	**0.55**	0.35	6.47	1.42
dc3	-	-	-	1,762	0.00	8.02	8.02	1.00	4.55	1.00
	ILU(0)	-	-	285	0.06	3.09	3.15	0.39	10.84	2.38
	Tri	1.2	10	310	0.01	1.94	**1.94**	0.24	6.26	1.38
poisson3Da	-	-	-	109	0.00	0.14	0.14	1.00	1.25	1.00
	ILU(0)	-	-	41	0.05	0.14	0.19	1.40	3.39	2.72
	Tri	1.3	5	40	0.00	0.07	**0.07**	0.52	1.78	1.42
raefsky2	-	-	-	318	0.00	0.21	0.21	1.00	0.66	1.00
	ILU(0)	-	-	31	0.05	0.06	0.10	0.49	1.81	2.75
	Tri	1.1	5	55	0.00	0.05	**0.05**	0.22	0.84	1.27
chipcool0	-	-	-	175	0.00	0.19	0.19	1.00	1.09	1.00
	ILU(0)	-	-	45	0.02	0.14	0.16	0.84	3.11	2.87
	Tri	1.2	5	55	0.00	0.09	**0.09**	0.48	1.65	1.52
chipcool1	-	-	-	164	0.00	0.18	0.18	1.00	1.10	1.00
	ILU(0)	-	-	39	0.02	0.12	0.14	0.78	3.08	2.80
	Tri	1.5	15	45	0.00	0.07	**0.07**	0.41	1.64	1.50
wang4	-	-	-	125	0.00	0.09	0.09	1.00	0.74	1.00
	ILU(0)	-	-	45	0.01	0.10	0.10	1.07	2.11	2.84
	Tri	1.2	5	45	0.00	0.05	**0.05**	0.54	1.13	1.52
Zhao1	-	-	-	41	0.00	0.03	0.04	1.00	0.83	1.00
	ILU(0)	-	-	13	0.01	0.03	0.04	1.00	2.15	2.60
	Tri	0.5	5	10	0.00	0.01	**0.01**	0.37	1.30	1.57
comsol	-	-	-	296	0.00	0.07	**0.07**	1.00	0.24	1.00
	ILU(0)	-	-	185	0.02	0.10	0.13	1.83	0.56	2.38
	Tri	0.8	20	460	0.00	0.13	0.13	1.87	0.28	1.20
big	-	-	-	2,650	0.00	1.12	1.12	1.00	0.42	1.00
	ILU(0)	-	-	820	0.01	0.94	0.95	0.85	1.15	2.71
	Tri	1.1	15	765	0.00	0.54	**0.54**	0.48	0.70	1.65

表 7: Tri 型前処理つき CG 法の前処理なしに対する平均時間比における理論値と実測値の比較.

行列	実測値	計算量		メモリアクセス	
		理論値	差	理論値	差
tikatan_big	2.19	1.16	1.03	1.55	0.64
s3dkq4m2	1.69	1.05	0.64	1.51	0.18
s3dkt3m2	1.58	1.06	0.52	1.51	0.07
ct20stif	1.63	1.05	0.58	1.51	0.12
CABLE	1.66	1.06	0.60	1.51	0.15
ft21a2	1.61	1.06	0.55	1.51	0.10
ft21a1	1.61	1.07	0.54	1.52	0.09
ft21a3	1.61	1.07	0.54	1.52	0.09
ft21b3	1.60	1.06	0.54	1.51	0.09
msc23052	1.40	1.05	0.35	1.51	0.11
tikatan	1.46	1.10	0.36	1.50	0.04
cla-pla1000-1	1.44	1.06	0.38	1.52	0.08
BEAM	-	1.08	-	1.51	-
bodyy5	1.15	1.14	0.01	1.49	0.34
bcsstk15	1.52	1.09	0.43	1.53	0.01
nasa4704	1.52	1.08	0.44	1.51	0.01
s2	1.44	1.18	0.26	1.47	0.03

表 8: Tri 型前処理つき BiCGStab 法の前処理なしに対する平均時間比における理論値と実測値の比較.

行列	実測値	計算量		メモリアクセス	
		理論値	差	理論値	差
waseda	1.01	1.03	0.02	1.04	0.02
air-cfl5	1.42	1.12	0.29	1.16	0.25
xenon2	1.24	1.08	0.16	1.10	0.14
poisson3Db	1.40	1.07	0.33	1.09	0.32
matrix_9	-	1.09	-	1.11	-
language	1.50	1.25	0.25	1.39	0.11
xenon1	1.25	1.11	0.14	1.13	0.12
dc2	1.42	1.18	0.24	1.25	0.17
dc3	1.38	1.21	0.17	1.28	0.10
poisson3Da	1.42	1.07	0.35	1.09	0.33
raefsky2	1.27	1.07	0.20	1.07	0.20
chipcool0	1.52	1.15	0.38	1.18	0.34
chipcool1	1.50	1.12	0.38	1.15	0.35
wang4	1.52	1.20	0.32	1.27	0.25
Zhao1	1.57	1.23	0.34	1.32	0.24
comsol	1.20	1.04	0.16	1.05	0.16
big	1.65	1.17	0.48	1.24	0.41

参考文献

[1] Axelsson, O.: Iterative Solution Methods, Cambridge University Press, 1994.

[2] Chan, T. F., van der Vorst, H. A.: Approximate and Incomplete Factorizations, in David E. Keyes, Ahmed Samed and V. Venkatakrishnan(eds), Parallel Numerical Algorithms, 1997.

[3] Eisenstat, S. C.: Efficient implementation of a class of preconditioned conjugate gradient methods, SIAM J. Sci. Stat. Compute., Vol.2, No.1, pp.1-4, 1981.

[4] 野寺 隆: 大型疎行列に対する PCG 法, Seminar on Mathematical Science, No.7, 1983.

[5] Saad, Y.: Iterative Methods for Sparse Linear Systems, 2nd Edition, SIAM, Philadelphia, 2003.

[6] 高橋秀俊, 野寺隆: PCG アルゴリズムの効果的な実現の一考察, 情報処理学会第 24 回全国大会講演集, pp.901-902, 1982.

[7] University of Florida Sparse Matrix Collection: http://www.cise.ufl.edu/research/sparse/matrices/index.html