2008－HPC－116 (14)
2008／8／5

積型反復法の前処理の適切な選択について

Moe Thuthu 尾上勇介 藤野清次†
(九州大学大学院システム情報科学府 †九州大学情報基盤研究開発センター)

本研究では，反復1回あたりの行列・ベクトル積の計算回数を2回に抑えるという観点から，GPBi-CG 法では両側前処理の導出だけが意味があり，一方 GPBiCG_AR 法では，両側，左，右前処理すべてが導出できることを理論的に考察し，かつ後者の前処理つき反復法が有用であることを数値実験で明らかにする．

Flexible preconditioner of product-type of iterative methods

Moe Thuthu Yusuke Onoue Seiji Fujino†
(Graduate School of Information Science and Electrical Engineering, Kyushu University,
†Research Institute for Information Technology, Kyushu University)

In this research, we discuss flexibility of preconditioning of the conventional GPBi-CG and our proposed GP-BiCG_AR methods. Through some numerical experiments, validity of preconditioned GPBiCG_AR method will be demonstrated in view of convergence rate and robustness of convergence for various realistic problems.

1. はじめに

非対称行列を係数行列に持つ線形方程式に対して，BiCG 法系統，GMRES 法系統など様々な反復法が使われている．本稿では，張らによって提案された GPBi-CG 法 [4] の前処理および筆者らに提案された GPBiCG_AR 法 [2] [3] の前処理について議論する．そして，行列・ベクトル積の計算回数が2回である という観点から，GPBi-CG 法では両側前処理の導出だけが意味があり，一方 GPBiCG_AR 法では，両側，左，右前処理すべてが導出できることを明らかにする．

本稿の構成は次のとおりである．第2節で，両側前処理つき GPBi-CG 法について記述する．第3節で，GPBi-CG 法の右前処理の導出が出来ないことを背理法 を使って証明する．第4節で，前処理なし GPBiCG_AR 法の算法を紹介する．そして，両側，右前処理つき GPBiCG_AR 法の算法を記述する．第5節で，数値実験を通して，両側前処理つき GPBi-CG 法の収束の不安定性，一方両側，右前処理つき GPBiCG_AR 法の収束の安定性を明らかにする．第6節で，まとめと今後の課題を述べる．

2. 両側前処理つき GPBi-CG 法

解くべき線形方程式 $A\boldsymbol{x} = \boldsymbol{b}$ に対して，前処理行列 $K(\approx A = K_1K_2)$ の逆行列を左から掛けた方程式を考える．

$$K^{-1}Ax = K^{-1}b. \tag{1}$$

$\tilde{A} = (K_1^{-1}AK_2^{-1})$, $\tilde{\boldsymbol{x}} = (K_2\boldsymbol{x})$, $\tilde{\boldsymbol{b}} = (K_1^{-1}\boldsymbol{b})$ とおくと，次の変換された方程式が得られる．

$$\tilde{A}\tilde{x} = \tilde{b}. \tag{2}$$

次に，近似解 $\boldsymbol{x}_n$ と残差ベクトル $\boldsymbol{r}_n$ に対して次の変換を施す．

$$\tilde{\boldsymbol{x}}_n := K_2\boldsymbol{x}_n,\ \tilde{\boldsymbol{r}}_n := K_1^{-1}\boldsymbol{r}_n. \tag{3}$$

同様に，補助ベクトルに対して，次の変換を行う．

$$\begin{aligned}&\tilde{\boldsymbol{p}}_n := K_2\boldsymbol{p}_n,\ \tilde{\boldsymbol{t}}_n := K_1^{-1}\boldsymbol{t}_n,\\ &\tilde{\boldsymbol{y}}_n := K_1^{-1}\boldsymbol{y}_n,\ \tilde{\boldsymbol{\omega}}_n := K_1^{-1}\boldsymbol{\omega}_n, \tilde{\boldsymbol{u}}_n := K_2\boldsymbol{u}_n,\\ &\tilde{\boldsymbol{z}}_n := K_2\boldsymbol{z}_n,\ \tilde{\boldsymbol{r}}_0^* := K_1^H\boldsymbol{r}_0^*.\end{aligned} \tag{4}$$

K^H はエルミート行列を表す．変換後の式を元の行列の記号に戻し書き下すと，両側前処理つき GPBi-CG 法の算法が以下のように得られる．

$\boldsymbol{x}_0$ is an initial guess, $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$,
set $\boldsymbol{t}_{-1} = \boldsymbol{w}_{-1} = 0$, $\beta_{-1} = 0$,
for $n = 0, 1, \cdots$ until $||\boldsymbol{r}_{n+1}|| \leq \varepsilon\ ||\boldsymbol{r}_0||$ do :
begin

$$\begin{aligned}&\boldsymbol{p}_n = K^{-1}\boldsymbol{r}_n + \beta_{n-1}(\boldsymbol{p}_{n-1} - \boldsymbol{u}_{n-1}),\\ &\alpha_n = \frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)}{(\boldsymbol{r}_0^*, A\boldsymbol{p}_n)},\\ &\boldsymbol{y}_n = \boldsymbol{t}_{n-1} - \boldsymbol{r}_n - \alpha_n\boldsymbol{w}_{n-1} + \alpha_n A\boldsymbol{p}_n,\end{aligned} \tag{5}$$

$$
\begin{aligned}
&\boldsymbol{t}_n = \boldsymbol{r}_n - \alpha_n A\boldsymbol{p}_n,\\
&K^{-1}\boldsymbol{t}_n = K^{-1}\boldsymbol{r}_n - \alpha_n K^{-1}A\boldsymbol{p}_n,\\
&\boldsymbol{a}_n = \boldsymbol{t}_n,\ \ \boldsymbol{b}_n = \boldsymbol{y}_n,\ \ \boldsymbol{c}_n = AK^{-1}\boldsymbol{t}_n,\\
&\zeta_n = \frac{(\boldsymbol{b}_n,\boldsymbol{b}_n)(\boldsymbol{c}_n,\boldsymbol{a}_n) - (\boldsymbol{b}_n,\boldsymbol{a}_n)(\boldsymbol{c}_n,\boldsymbol{b}_n)}{(\boldsymbol{c}_n,\boldsymbol{c}_n)(\boldsymbol{b}_n,\boldsymbol{b}_n) - (\boldsymbol{b}_n,\boldsymbol{c}_n)(\boldsymbol{c}_n,\boldsymbol{b}_n)},\\
&\eta_n = \frac{(\boldsymbol{c}_n,\boldsymbol{c}_n)(\boldsymbol{b}_n,\boldsymbol{a}_n) - (\boldsymbol{b}_n,\boldsymbol{c}_n)(\boldsymbol{c}_n,\boldsymbol{a}_n)}{(\boldsymbol{c}_n,\boldsymbol{c}_n)(\boldsymbol{b}_n,\boldsymbol{b}_n) - (\boldsymbol{b}_n,\boldsymbol{c}_n)(\boldsymbol{c}_n,\boldsymbol{b}_n)},\\
&(\text{if } n=0,\ \text{then } \zeta_n = \frac{(\boldsymbol{c}_n,\boldsymbol{a}_n)}{(\boldsymbol{c}_n,\boldsymbol{c}_n)},\ \eta_n = 0)\\
&\boldsymbol{u}_n = \zeta_n K^{-1}A\boldsymbol{p}_n + \eta_n(K^{-1}\boldsymbol{t}_{n-1} - K^{-1}\boldsymbol{r}_n\\
&\qquad +\beta_{n-1}\boldsymbol{u}_{n-1}), \qquad (6)\\
&\boldsymbol{z}_n = \zeta_n K^{-1}\boldsymbol{r}_n + \eta_n\boldsymbol{z}_{n-1} - \alpha_n\boldsymbol{u}_n, \qquad (7)\\
&\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n\boldsymbol{p}_n + \boldsymbol{z}_n, \qquad (8)\\
&\boldsymbol{r}_{n+1} = \boldsymbol{t}_n - \eta_n\boldsymbol{y}_n - \zeta_n AK^{-1}\boldsymbol{t}_n, \qquad (9)\\
&\beta_n = \frac{\alpha_n}{\zeta_n}\cdot\frac{(\boldsymbol{r}_0^*,\boldsymbol{r}_{n+1})}{(\boldsymbol{r}_0^*,\boldsymbol{r}_n)},\\
&\boldsymbol{w}_n = AK^{-1}\boldsymbol{t}_n + \beta_n A\boldsymbol{p}_n,
\end{aligned}
$$

end

3. GPBi-CG 法の右前処理の導出不可性

背理法を用いて，命題「**GPBi-CG 法の右前処理では，前進・後退代入の計算回数が反復 1 回当たり 3 回になる**」，を以下のように証明する．

まず，右前処理における変換は，

$$
\begin{aligned}
\tilde{A} &:= AM^{-1},\ \tilde{\boldsymbol{x}} := M\boldsymbol{x},\ \tilde{\boldsymbol{b}} := \boldsymbol{b},\\
\tilde{\boldsymbol{r}} &:= \tilde{\boldsymbol{b}} - \tilde{A}\tilde{\boldsymbol{x}} = \boldsymbol{b} - A\boldsymbol{x} = \boldsymbol{r}
\end{aligned}
\qquad (10)
$$

を用いる．算法中の式 (9) において，背理法により，「中間ベクトル $\boldsymbol{t}$ で前進・後退代入の計算が必ず生じる」，ことを以下のように証明する．

まず，式 (9) の中間ベクトル $\boldsymbol{t}$ で前進・後退代入の計算が生じない，と仮定 (**仮定 1** とおく) する．$\tilde{\boldsymbol{t}} := K\boldsymbol{t}$ とおくと，

$$
\tilde{\boldsymbol{t}} = K\boldsymbol{t},\ \ \tilde{A}\tilde{\boldsymbol{t}} = AM^{-1}K\boldsymbol{t} \qquad (11)
$$

となる．このとき，仮定より，

$$
K = I \quad \text{and} \quad K = M \qquad (12)
$$

を同時に満たす行列 K は存在しないため，上記の**仮定 1** が誤りになる．したがって，式 (9) において，中間ベクトル $\boldsymbol{t}$ で前進・後退代入の計算が必ず 1 回生じることになる．

次に，背理法により，「中間ベクトル $\boldsymbol{t}$ 以外の補助ベクトルで前進・後退代入の計算が最低 2 回生じる」，ことを証明する．

まず，ベクトル $\boldsymbol{t}$ 以外の補助ベクトルで前進・後退代入の計算が 1 回以下しか生じない，と仮定 (**仮定 2** とおく) する．式 (6) では，$\tilde{\boldsymbol{r}} = \boldsymbol{r}, \tilde{A}\tilde{\boldsymbol{p}} = AM^{-1}\tilde{\boldsymbol{p}}$ となる．ここで，$\tilde{\boldsymbol{u}} = M\boldsymbol{u}$ とおくと M^{-1} が 2 個，$\tilde{\boldsymbol{u}} = M^{-1}\boldsymbol{u}$ とおくと M が 2 個生じる．したがって，仮定より，中間ベクトル $\tilde{\boldsymbol{u}}$ は，

$$
\tilde{\boldsymbol{u}} := \boldsymbol{u} \qquad (13)
$$

と置換する必要がある．式 (7) でも，$\tilde{\boldsymbol{z}} = M\boldsymbol{z}$ とおくと M^{-1} が 2 個，$\tilde{\boldsymbol{z}} = M^{-1}\boldsymbol{z}$ とおくと M が 2 個生じるので，**仮定 2** より，ベクトル $\tilde{\boldsymbol{z}}$ は，

$$
\tilde{\boldsymbol{z}} := \boldsymbol{z} \qquad (14)
$$

と置換する必要がある．式 (5) でも，$\tilde{\boldsymbol{p}} = M\boldsymbol{p}$ とおくと，M^{-1} が 2 個，$\tilde{\boldsymbol{p}} = M^{-1}\boldsymbol{p}$ とおくと，M が 2 個生じるので，**仮定 2** より，ベクトル $\tilde{\boldsymbol{p}}$ は，

$$
\tilde{\boldsymbol{p}} := \boldsymbol{p} \qquad (15)
$$

と置換する必要がある．式 (8) でも，$\tilde{\boldsymbol{z}} = \boldsymbol{z}$, $\tilde{\boldsymbol{p}} = \boldsymbol{p}$, $\tilde{\boldsymbol{x}} = M\tilde{\boldsymbol{x}}$ より，$M^{-1}\boldsymbol{z}$, $M^{-1}\boldsymbol{p}$ の合計 2 回の前進・後退代入計算が生じる．この結果，**仮定 2** は誤りである．

したがって，中間ベクトル $\boldsymbol{t}$ 以外の補助ベクトルで前進・後退代入の計算が最低 2 回は生じる．以上の議論により，**GPBi-CG 法の右前処理では，前進・後退代入の計算回数が反復 1 回当たり 3 回になる**，ことが証明された．■

同様に，左前処理についても，背理法により，前進・後退代入の計算が反復 1 回当たり 3 回になることが証明 (証明は割愛) される．したがって，GPBi-CG 法において，実用的な観点から，その前処理は両側前処理だけである．

一方，GPBiCG_AR 法については，両側，左，右前処理の導出がすべて可能である．

4. 前処理なし GPBiCG_AR 法

前処理なし GPBiCG_AR 法の算法を以下に示す．

$\boldsymbol{x}_0$ is an initial guess, $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, $\beta_{-1} = 0$,
for $n = 0, 1, \cdots$ until $||\boldsymbol{r}_{n+1}|| \le \varepsilon\ ||\boldsymbol{r}_0||$ do :
begin

$$
\begin{aligned}
&\boldsymbol{p}_n = \boldsymbol{r}_n + \beta_{n-1}(\boldsymbol{p}_{n-1} - \boldsymbol{u}_{n-1}),\\
&A\boldsymbol{p}_n = A\boldsymbol{r}_n + \beta_{n-1}(A\boldsymbol{p}_{n-1} - A\boldsymbol{u}_{n-1}),\\
&\alpha_n = \frac{(\boldsymbol{r}_0^*,\boldsymbol{r}_n)}{(\boldsymbol{r}_0^*,A\boldsymbol{p}_n)},\\
&\boldsymbol{a}_n = \boldsymbol{r}_n,\ \ \boldsymbol{b}_n = A\boldsymbol{z}_{n-1},\ \ \boldsymbol{c}_n = A\boldsymbol{r}_n,\\
&\zeta_n = \frac{(\boldsymbol{b}_n,\boldsymbol{b}_n)(\boldsymbol{c}_n,\boldsymbol{a}_n) - (\boldsymbol{b}_n,\boldsymbol{a}_n)(\boldsymbol{c}_n,\boldsymbol{b}_n)}{(\boldsymbol{c}_n,\boldsymbol{c}_n)(\boldsymbol{b}_n,\boldsymbol{b}_n) - (\boldsymbol{b}_n,\boldsymbol{c}_n)(\boldsymbol{c}_n,\boldsymbol{b}_n)},\\
&\eta_n = \frac{(\boldsymbol{c}_n,\boldsymbol{c}_n)(\boldsymbol{b}_n,\boldsymbol{a}_n) - (\boldsymbol{b}_n,\boldsymbol{c}_n)(\boldsymbol{c}_n,\boldsymbol{a}_n)}{(\boldsymbol{c}_n,\boldsymbol{c}_n)(\boldsymbol{b}_n,\boldsymbol{b}_n) - (\boldsymbol{b}_n,\boldsymbol{c}_n)(\boldsymbol{c}_n,\boldsymbol{b}_n)},\\
&(\text{if } n=0,\ \text{then } \zeta_n = \frac{(\boldsymbol{c}_n,\boldsymbol{a}_n)}{(\boldsymbol{c}_n,\boldsymbol{c}_n)},\ \eta_n = 0)
\end{aligned}
$$

$\boldsymbol{u}_n = \zeta_n A\boldsymbol{p}_n + \eta_n(\boldsymbol{t}_{n-1} - \boldsymbol{r}_n + \beta_{n-1}\boldsymbol{u}_{n-1})$,
compute $A\boldsymbol{u}_n$,
$\boldsymbol{t}_n = \boldsymbol{r}_n - \alpha_n A\boldsymbol{p}_n$,
$\boldsymbol{z}_n = \zeta_n \boldsymbol{r}_n + \eta_n \boldsymbol{z}_{n-1} - \alpha_n \boldsymbol{u}_n$,
$A\boldsymbol{z}_n = \zeta_n A\boldsymbol{r}_n + \eta_n A\boldsymbol{z}_{n-1} - \alpha_n A\boldsymbol{u}_n$,
$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{p}_n + \boldsymbol{z}_n$,
$\boldsymbol{r}_{n+1} = \boldsymbol{t}_n - A\boldsymbol{z}_n$,
compute $A\boldsymbol{r}_{n+1}$,
$\beta_n = \dfrac{\alpha_n}{\zeta_n} \cdot \dfrac{(\boldsymbol{r}_0^*, \boldsymbol{r}_{n+1})}{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)}$,
end

compute $A\boldsymbol{u}_n$ と同 $A\boldsymbol{r}_{n+1}$ は行列・ベクトル積 $A\boldsymbol{u}_n$ と $A\boldsymbol{r}_{n+1}$ を定義通り計算することを意味する.

4.1 GPBiCG_AR 法の両側前処理

GPBiCG_AR 法では, 近似解 $\boldsymbol{x}_n$ と残差ベクトル $\boldsymbol{r}_n$ に対して次の変換を施す.

$$\tilde{\boldsymbol{x}}_n \;:=\; K_2\boldsymbol{x}_n,\ \tilde{\boldsymbol{r}}_n := K_1^{-1}\boldsymbol{r}_n.eqn4b \quad (16)$$

補助ベクトルに対しても次の変換を各々行う.

$$\tilde{\boldsymbol{p}}_n \;:=\; K_2\boldsymbol{p}_n,\ \tilde{\boldsymbol{t}}_n := K_1^{-1}\boldsymbol{t}_n, \quad (17)$$
$$\tilde{\boldsymbol{u}}_n \;:=\; K_2\boldsymbol{u}_n,\ \tilde{\boldsymbol{z}}_n := K_2\boldsymbol{z}_n,\ \tilde{\boldsymbol{r}}_0^* := K_1^H\boldsymbol{r}_0^*. \quad (18)$$

変換された式を元の方程式の記号に戻し書き下すと, 両側前処理つき GPBiCG_AR 法の算法が次のように得られる.

$\boldsymbol{x}_0$ is an initial guess, $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, $\beta_{-1} = 0$,
for $n = 0, 1, \cdots$ until $||\boldsymbol{r}_{n+1}|| \le \varepsilon\ ||\boldsymbol{r}_0||$ do :
begin
$\boldsymbol{p}_n = K^{-1}\boldsymbol{r}_n + \beta_{n-1}(\boldsymbol{p}_{n-1} - \boldsymbol{u}_{n-1})$,
$A\boldsymbol{p}_n = AK^{-1}\boldsymbol{r}_n + \beta_{n-1}(A\boldsymbol{p}_{n-1} - A\boldsymbol{u}_{n-1})$,
$\alpha_n = \dfrac{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)}{(\boldsymbol{r}_0^*, A\boldsymbol{p}_n)}$,
$K^{-1}\boldsymbol{t}_n = K^{-1}\boldsymbol{r}_n - \alpha_n K^{-1}A\boldsymbol{p}_n$,
$\boldsymbol{a}_n = \boldsymbol{r}_n$, $\boldsymbol{b}_n = A\boldsymbol{z}_{n-1}$, $\boldsymbol{c}_n = AK^{-1}\boldsymbol{r}_n$,
$\zeta_n = \dfrac{(\boldsymbol{b}_n,\boldsymbol{b}_n)(\boldsymbol{c}_n,\boldsymbol{a}_n) - (\boldsymbol{b}_n,\boldsymbol{a}_n)(\boldsymbol{c}_n,\boldsymbol{b}_n)}{(\boldsymbol{c}_n,\boldsymbol{c}_n)(\boldsymbol{b}_n,\boldsymbol{b}_n) - (\boldsymbol{b}_n,\boldsymbol{c}_n)(\boldsymbol{c}_n,\boldsymbol{b}_n)}$,
$\eta_n = \dfrac{(\boldsymbol{c}_n,\boldsymbol{c}_n)(\boldsymbol{b}_n,\boldsymbol{a}_n) - (\boldsymbol{b}_n,\boldsymbol{c}_n)(\boldsymbol{c}_n,\boldsymbol{a}_n)}{(\boldsymbol{c}_n,\boldsymbol{c}_n)(\boldsymbol{b}_n,\boldsymbol{b}_n) - (\boldsymbol{b}_n,\boldsymbol{c}_n)(\boldsymbol{c}_n,\boldsymbol{b}_n)}$,
(if $n = 0$, then $\zeta_n = \dfrac{(\boldsymbol{c}_n,\boldsymbol{a}_n)}{(\boldsymbol{c}_n,\boldsymbol{c}_n)}$, $\eta_n = 0$)
$\boldsymbol{u}_n = \zeta_n K^{-1}A\boldsymbol{p}_n + \eta_n(K^{-1}\boldsymbol{t}_{n-1} - K^{-1}\boldsymbol{r}_n + \beta_{n-1}\boldsymbol{u}_{n-1})$,
compute $A\boldsymbol{u}_n$,
$\boldsymbol{t}_n = \boldsymbol{r}_n - \alpha_n A\boldsymbol{p}_n$,
$\boldsymbol{z}_n = \zeta_n K^{-1}\boldsymbol{r}_n + \eta_n \boldsymbol{z}_{n-1} - \alpha_n \boldsymbol{u}_n$,
$A\boldsymbol{z}_n = \zeta_n AK^{-1}\boldsymbol{r}_n + \eta_n A\boldsymbol{z}_{n-1} - \alpha_n A\boldsymbol{u}_n$,
$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{p}_n + \boldsymbol{z}_n$,
$\boldsymbol{r}_{n+1} = \boldsymbol{t}_n - A\boldsymbol{z}_n$,
compute $A\boldsymbol{r}_{n+1}$,
$\beta_n = \dfrac{\alpha_n}{\zeta_n} \cdot \dfrac{(\boldsymbol{r}_0^*, \boldsymbol{r}_{n+1})}{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)}$,
end

4.2 GPBiCG_AR 法の右前処理

GPBiCG_AR 法の右前処理 ($K_1 = I$ とおく) では, 変換された線形方程式は $\tilde{A} = (AK_2^{-1})$, $\tilde{\boldsymbol{x}} = (K_2\boldsymbol{x})$, $\tilde{\boldsymbol{b}} = \boldsymbol{b}$ となる. このとき, 近似解 $\boldsymbol{x}_n$ と残差ベクトル $\boldsymbol{r}_n$ に対して次の変換を施す.

$$\tilde{\boldsymbol{x}}_n \;:=\; K_2\boldsymbol{x}_n,\ \tilde{\boldsymbol{r}}_n := \boldsymbol{r}_n. \quad (19)$$

補助ベクトルに対しても次の変換を各々行う.

$$\tilde{\boldsymbol{p}}_n \;:=\; K_2\boldsymbol{p}_n,\ \tilde{\boldsymbol{t}}_n := \boldsymbol{t}_n, \quad (20)$$
$$\tilde{\boldsymbol{u}}_n \;:=\; \boldsymbol{u}_n,\ \tilde{\boldsymbol{z}}_n := K_2\boldsymbol{z}_n,\ \tilde{\boldsymbol{r}}_0^* := \boldsymbol{r}_0^*. \quad (21)$$

右前処理つき GPBiCG_AR 法の算法が次のように得られる.

$\boldsymbol{x}_0$ is an initial guess, $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, $\beta_{-1} = 0$,
for $n = 0, 1, \cdots$ until $||\boldsymbol{r}_{n+1}|| \le \varepsilon\ ||\boldsymbol{r}_0||$ do :
begin
$\boldsymbol{p}_n = K_1^{-1}\boldsymbol{r}_n + \beta_{n-1}(\boldsymbol{p}_{n-1} - K_1^{-1}\boldsymbol{u}_n)$,
$A\boldsymbol{p}_n = AK_1^{-1}\boldsymbol{r}_n + \beta_{n-1}(A\boldsymbol{p}_{n-1} - AK_1^{-1}\boldsymbol{u}_n)$,
$\alpha_n = \dfrac{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)}{(\boldsymbol{r}_0^*, A\boldsymbol{p}_n)}$,
$\boldsymbol{a}_n = \boldsymbol{r}_n$, $\boldsymbol{b}_n = A\boldsymbol{z}_{n-1}$, $\boldsymbol{c}_n = AK_1^{-1}\boldsymbol{r}_n$,
$\zeta_n = \dfrac{(\boldsymbol{b}_n,\boldsymbol{b}_n)(\boldsymbol{c}_n,\boldsymbol{a}_n) - (\boldsymbol{b}_n,\boldsymbol{a}_n)(\boldsymbol{c}_n,\boldsymbol{b}_n)}{(\boldsymbol{c}_n,\boldsymbol{c}_n)(\boldsymbol{b}_n,\boldsymbol{b}_n) - (\boldsymbol{b}_n,\boldsymbol{c}_n)(\boldsymbol{c}_n,\boldsymbol{b}_n)}$,
$\eta_n = \dfrac{(\boldsymbol{c}_n,\boldsymbol{c}_n)(\boldsymbol{b}_n,\boldsymbol{a}_n) - (\boldsymbol{b}_n,\boldsymbol{c}_n)(\boldsymbol{c}_n,\boldsymbol{a}_n)}{(\boldsymbol{c}_n,\boldsymbol{c}_n)(\boldsymbol{b}_n,\boldsymbol{b}_n) - (\boldsymbol{b}_n,\boldsymbol{c}_n)(\boldsymbol{c}_n,\boldsymbol{b}_n)}$,
(if $n = 0$, then $\zeta_n = \dfrac{(\boldsymbol{c}_n,\boldsymbol{a}_n)}{(\boldsymbol{c}_n,\boldsymbol{c}_n)}$, $\eta_n = 0$)
$\boldsymbol{u}_n = \zeta_n A\boldsymbol{p}_n + \eta_n(\boldsymbol{t}_{n-1} - \boldsymbol{r}_n + \beta_{n-1}\boldsymbol{u}_{n-1})$,
compute $A\boldsymbol{u}_n$,
$\boldsymbol{t}_n = \boldsymbol{r}_n - \alpha_n A\boldsymbol{p}_n$,
$\boldsymbol{z}_n = \zeta_n K_1^{-1}\boldsymbol{r}_n + \eta_n \boldsymbol{z}_{n-1} - \alpha_n K_1^{-1}\boldsymbol{u}_n$,
$A\boldsymbol{z}_n = \zeta_n AK_1^{-1}\boldsymbol{r}_n + \eta_n A\boldsymbol{z}_{n-1} - \alpha_n AK_1^{-1}\boldsymbol{u}_n$,
$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{p}_n + \boldsymbol{z}_n$,
$\boldsymbol{r}_{n+1} = \boldsymbol{t}_n - A\boldsymbol{z}_n$,
compute $A\boldsymbol{r}_{n+1}$,
$\beta_n = \dfrac{\alpha_n}{\zeta_n} \cdot \dfrac{(\boldsymbol{r}_0^*, \boldsymbol{r}_{n+1})}{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)}$,
end

同様に, 左前処理 ($K_2 = I$ とおく) つき GPBiCG_AR 法の算法が導出 (割愛) される. 表 1 に, GPBi-CG 法と GPBiCG_AR 法の前処理における

反復1回あたりの前進・後退代入計算の回数を示す．両側前処理つき GPBi-CG 法と，GPBiCG_AR 法の両側，左，右前処理の回数が2回である．

Table 1: Number of computation of forward and backward substitutions per one iteration.

	前処理		
解法	左	両側	右
GPBi-CG	3	**2**	3
GPBiCG_AR	**2**	**2**	**2**

5. 数値実験

数値実験は，九州大学情報基盤研究開発センターに設置された Hitachi SR11000 モデル J1 で行った．主メモリは 128Gbytes，CPU は POWER5 (クロック 1.9GHz)，OS: AIX5.3 を使用した．プログラムは Fortran90 で実装，計算は倍精度浮動小数点演算，最適化オプションは，-64 -Oss -nolimit -noscope -noparallel を使用した．反復法の初期近似解 $\boldsymbol{x}_0$ はすべて零とし，最大反復数は1万回，収束判定条件は，相対残差の2ノルム $||\boldsymbol{r}_{n+1}||_2/||\boldsymbol{r}_0||_2$ の値が 10^{-12} 以下になったときとした．また，調べた前処理つき反復法は，両側前処理つき GPBi-CG 法，GPBiCG_AR 法では両側前処理つき (図中では Two-prec. と略記)，右前処理つき (同じく Right-prec. と略記)，合計3種類である．方程式の右辺項 $\boldsymbol{b}$ は，厳密解を $\hat{\boldsymbol{x}} = [1, 1, \cdots, 1]^T$ とし $\boldsymbol{b} = A\hat{\boldsymbol{x}}$ から生成した．初期シャドウ残差 $\boldsymbol{r}_0^*$ は初期残差 $\boldsymbol{r}_0$ を代入した．また，前処理は元の係数行列 A の非零要素と同じ場所の fill-in だけを使う ILU(0) 分解を採用した．表2にテスト行列 [1] の特徴を示す．

Table 2: Description of test matrices.

行列	次元数	非零要素数	平均非零要素	解析分野
bcircuit	68,902	375,558	5.45	回路
sme3da	12,504	874,887	69.97	構造
sme3dc	42,930	3,148,656	73.34	構造
bfw398a	398	3,678	9.61	構造
cheby-shev2	2,053	18,447	8.99	構造
big	13,209	91,465	6.92	構造

5.1 実験結果と観察

表3に，真の相対残差 (TRR: True Relative Residual) の最大のものと行列6個に対する平均値 (常用対数の値) を示す．表中の太字は各行列においてその平均値が最もよかったものを表す．また，「max」はまったく収束しなかったことを表す．

Table 3: Worst score and average score of TRR.

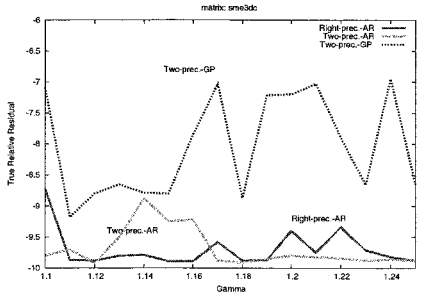
	GPBi-CG		GPBiCG_AR			
	両側		両側		右	
行列	最悪	平均	最悪	平均	最悪	平均
bcircuit	-4.11	-7.97	-10.63	-10.82	-10.30	**-10.85**
sme3da	-6.44	-8.31	-9.32	**-9.53**	-8.00	-9.45
sme3dc	-6.95	-8.04	-8.88	**-9.69**	-8.72	**-9.69**
bfw398a	-11.98	-12.25	-12.12	**-12.33**	-12.01	-12.28
cheby-shev2	-5.25	–	-12.00	-12.06	-12.03	**-12.16**
big	-9.22	-11.62	-12.01	**-12.19**	-12.02	-12.15

表4に，行列 chebyshev2 において，パラメータ γ を変えたときの，収束状況を示す．表中で，「inf」は発散したことを，「break」は収束過程での異常強制終了したことを各々表す．GPBi-CG 法の収束不安定現象が頻発するのに対して，GPBiCG_AR 法の収束安定性が目立つ．

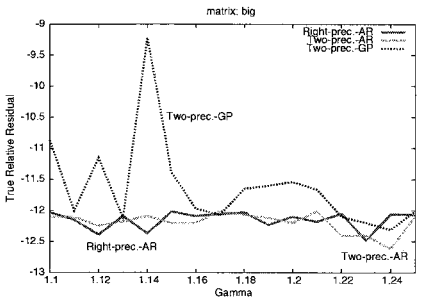
Table 4: Variation of TRR at various parameter γs for matrix chebyshev2.

	GPBi-CG	GPBiCG_AR	
γ	両側	両側	右
1.10	-5.84	-12.00	-12.26
1.11	-5.25	-12.03	-12.17
1.12	-5.70	-12.14	-12.11
1.13	-6.62	-12.14	-12.07
1.14	inf	-12.04	-12.05
1.15	break	-12.15	-12.03
1.16	-5.49	-12.12	-12.11
1.17	break	-12.07	-12.14
1.18	-5.45	-12.09	-12.09
1.19	-8.55	-12.06	-12.10
1.20	break	-12.00	-12.18
1.21	-6.21	-12.10	-12.15
1.22	-7.89	-12.01	-12.12
1.23	-6.22	-12.01	-12.39
1.24	inf	-12.01	-12.33
1.25	-5.48	-12.03	-12.18

図1(a), (b) に，行列 sme3dc，big に対して，γ の値を 1.1 から 1.25 まで 0.01 刻みで変化させたときの両側前処理つき GPBi-CG 法，2種類の前処理つき GPBiCG_AR 法の近似解 $\boldsymbol{x}_{n+1}$ に対する真の相対残差：$||\boldsymbol{b} - A\boldsymbol{x}_{n+1}||_2/||\boldsymbol{b} - A\boldsymbol{x}_0||_2$ の変動の様子を示す．

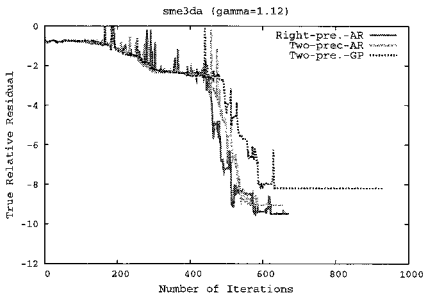


(a)matrix: sme3dc

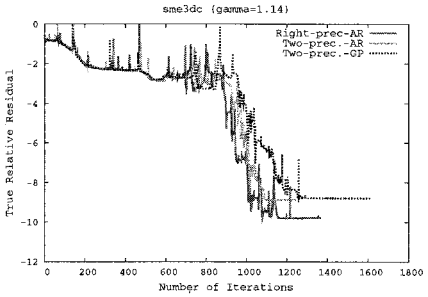


(b)matrix big

図 1: Variation of True Relative Residual when γ of ILU(0) varies from 1.1 to 1.25 for sme3dc, big.



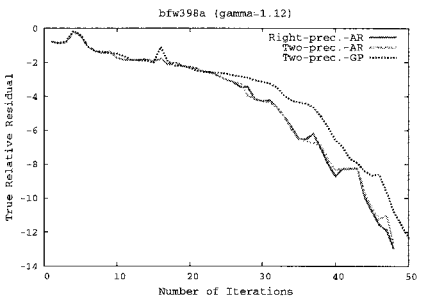
(b)matrix sme3da and $\gamma = 1.12$



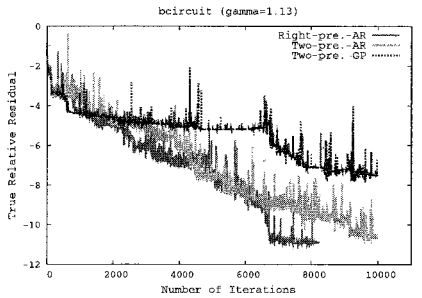
(c)matrix: sme3dc and $\gamma = 1.14$

図 2: True Relative Residual history of preconditioned GPBi-CG and GPBiCG_AR methods for bcircuit, sme3da and sme3dc.

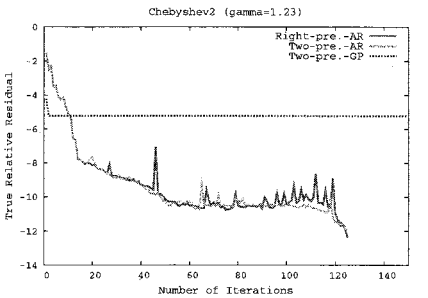
図 2(a), (b), (c) に，行列 bcircuit, sme3da, sme3dc における，両側前処理つき GPBi-CG 法，2 種類の前処理つき GPBiCG_AR 法の真の相対残差: $||\boldsymbol{b} - A\boldsymbol{x}_{n+1}||_2/||\boldsymbol{b} - A\boldsymbol{x}_0||_2$ の履歴を示す．同様に，図 3(a), (b), (c) に，行列 bfw398a, chebyshev2, big における，両側前処理つき GPBi-CG 法，2 種類の前処理つき GPBiCG_AR 法の同じく真の相対残差の履歴を示す．



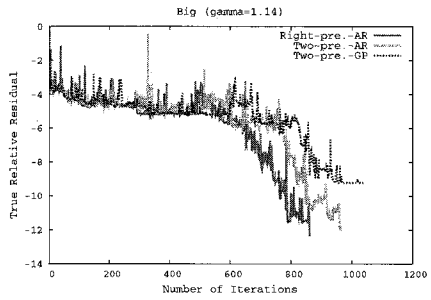
(a)matrix bfw398a and $\gamma = 1.12$



(a)matrix: bcircuit and $\gamma = 1.13$



(b)matrix chebyshev2 and $\gamma = 1.23$



(c)matrix big and $\gamma = 1.14$

図 3: True Relative Residual history of preconditioned GPBi-CG and GPBiCG_AR methods for bfw398a, chebyshev2 and big.

図 2-3 からわかるように，GPBi-CG 法では，パラメータ γ によっては，収束の不安定現象が現れた，一方，GPBiCG_AR 法では同様の現象は現れず，安定した収束性を示した．表 5-6 に，行列 sme3dc, big における，反復回数と計算時間 [sec.] の変動を示す表中の太字の数字は，収束までの計算時間が最も少なかったものを表す．表から，GPBiCG_AR 法の方が，GPBi-CG 法よりも有用であることがわかる．

Table 5: Variation of iterations and CPU times at various parameter γs for matrix sme3dc.

	GPBi-CG		GPBiCG_AR			
	両側		両側		右	
γ	回数	CPU	回数	CPU	回数	CPU
1.10	1900	191.9	1162	**100.0**	1235	126.3
1.11	1489	**150.4**	1221	105.7	1157	**118.4**
1.12	1698	172.8	1192	102.5	1183	120.7
1.13	1619	162.9	1293	110.4	1263	129.5
1.14	1614	164.0	1251	108.4	1368	139.6
1.15	1521	155.2	1288	110.3	1185	120.7
1.16	1717	172.3	1320	114.1	1213	122.5
1.17	1803	182.9	1248	106.9	1291	130.8
1.18	1693	171.9	1220	105.3	1265	128.1
1.19	1849	184.8	1318	111.9	1339	137.1
1.20	1921	194.6	1252	108.2	1549	155.5
1.21	2016	202.5	1342	114.6	1239	127.5
1.22	1940	193.9	1331	114.8	1367	138.2
1.23	1800	181.3	1322	113.9	1430	145.9
1.24	1962	195.5	1441	123.9	1482	148.5
1.25	1927	194.0	1319	112.0	1285	130.6

Table 6: Variation of iterations and CPU times at various parameter γs for matrix big.

	GPBi-CG		GPBiCG_AR			
	両側		両側		右	
γ	回数	CPU	回数	CPU	回数	CPU
1.10	1012	4.16	850	3.01	821	3.38
1.11	956	**3.93**	837	2.95	883	3.57
1.12	1035	4.26	837	2.95	919	3.71
1.13	1011	4.17	919	3.23	901	3.65
1.14	1041	4.27	963	3.39	861	3.50
1.15	963	3.99	860	3.03	898	3.64
1.16	1010	4.16	860	3.04	907	3.68
1.17	1017	4.18	825	2.90	829	3.37
1.18	1017	4.18	808	**2.85**	846	3.44
1.19	986	4.06	871	3.08	896	3.65
1.20	1013	4.17	895	3.17	894	3.62
1.21	1100	4.51	977	3.43	848	3.44
1.22	1059	4.35	988	3.48	895	3.62
1.23	1045	4.30	919	3.23	983	3.99
1.24	1058	4.35	912	3.22	873	3.55
1.25	1074	4.40	963	3.40	811	**3.30**

6. まとめ

従来の GPBi-CG 法では，両側前処理しか構築できないことを理論的に明らかにした．一方，GPBiCG_AR 法では，両側，左，右前処理を構築でき，前処理選択の幅を広げられた．さらに，両側前処理つき GPBi-CG 法では，収束の不安定現象が色々な行列に対してまた様々なパラメータ γ に対して現れることを指摘した．一方，前処理つき GPBiCG_AR 法では，収束の不安定現象がほとんど現れず，効率を重視した前処理を適宜選択することができ，実際上有用であることがわかった．

参考文献

[1] Univ. of Florida Sparse Matrix Collection, http://www.cise.ufl.edu/research/sparse/matrices/

[2] Moe Thuthu, S. Fujino: Stability of GPBiCG_AR method based on minimization of associate residual, The Proc. of ASCM 2007, Singapore, Dec. 15-17, 2007.

[3] Moe Thuthu, 藤野清次, 尾上勇介：複数の前処理導出が可能な GPBiCG_AR 法の収束性評価，日本応用数理学会平成 20 年度研究部会連合発表会，首都大学東京，3 月，2008.

[4] 張紹良，藤野清次：ランチョス・プロセスに基づく積型反復解法，日本応用数理学会論文誌，5(1995)，343-360.