数理解析研究所講究録
第1638巻 2009年 95-111

一般化 Bi-CGSTAB(s, L) (= 一般化 IDR(s, L))

Generalized Bi-CGSTAB(s, L) (= Generalized IDR(s, L))

谷尾 真明，杉原 正顯
Masaaki Tanio, Masaaki Sugihara

東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo

1 はじめに

係数行列が大規模非対称な連立一次方程式の数値解法として，近年 IDR(s) 法 [7] が提案され，わが国で特に注目を集めている．この解法は，IDR 原理と呼ばれる新しい原理から導かれ，理論的観点からの特長「係数行列のサイズを $N \times N$ するとき，$N + N/s$ 回の行列ベクトル積演算で真の解を与える」をもち，また，実際の問題を解く場面においても，一般に，既存の手法より収束性が良いことが実証されている．しかし，同時に，算法に使われる安定化多項式の次数が 1 次に限定されているため，歪対称に近い係数行列を持つ連立一次方程式に対しては収束性が悪化することも指摘されている．これに対して，我々は，IDR 原理を一般化し，安定化多項式の次数が L 次で，$N + N/s$ 回の行列ベクトル積演算で真の解を与えるアルゴリズム（一般化 IDR(s, L) 法）を提案し，実際に，歪対称に近い係数行列を持つ連立一次方程式に対しても収束性が悪化しないことを示した [10].

一方，IDR(s) 法が提案されて，すぐに，IDR(s) 法は Bi-CG 法と関連付けて解釈出来ることが示された [5]. 論文 [5] では，block Krylov subspace[1] という概念を用いて，高次元 shadow residual を持つ Bi-CG 法が定義され，さらに，IDR(1) と Bi-CGSTAB の関係に対する深い洞察をもとに，この高次元 shadow residual を持つ Bi-CG 法に対して 1 次の安定化多項式を付加したアルゴリズム (論文中では Bi-CGSTAB(s) 法と呼ばれている) が提案されている．そして，この Bi-CGSTAB(s) 法が IDR(s) 法と等価であることが証明されている．

本稿では，一般化 IDR(s, L) 法についても Bi-CG 法と関連付けて解釈出来ることを示す．ただし，論文 [5] で提案された高次元 shadow residual を持つ Bi-CG 法を用いることでは目標は達成されず，新しい高次元 shadow residual を持つ Bi-CG 法 (GBi-CG(s) 法と命名する) を必要とする．一旦，これが出来てしまえば，L 次の安定化多項式を付加する部分は，Bi-CG 法から Bi-CGSTAB(L) 法（Bi-CG 法に L 次の安定化多項式を付加したアルゴリズム）を導いたのと同様に—より複雑ではあるが，アイデアは同じ—安定化多項式を付加することができ，一般化 IDR(s, L) 法と同じアルゴリズムが導かれる．なお，このような導出の観点から，アルゴリズムを呼ぶとき，GBi-CGSTAB(s, L) 法と呼ぶことにする．このアルゴリズムは，一般化 IDR(s, L) 法と全く同じではあるが，導出原理が

全く違うので,「名は導出原理を表す」という意味で別名を付けておく.

本論文の構成は以下の通りである. 第 2 章において, 既存研究である Bi-CG 法, Bi-CGSTAB(L) 法の説明を行う. 第 3 章において, 高次元 shadow residual を持つ Bi-CG 法を定義し, 論文 [5] で提案された高次元 shadow residual を持つ Bi-CG 法, 今回新たに提案する GBi-CG(s) 法について説明を行う. 第 4 章においては, 3 章で定義した新しい高次元 shadow residual を持つ Bi-CG 法に対して, L 次の安定化多項式を付加したアルゴリズム (GBi-CGSTAB(s, L) 法 (= 一般化 IDR(s, L) 法)) を導出する. 第 5 章では, 数値実験結果を与える. 第 6 章で結論と今後の課題を述べる.

■記法と定義

以下論文を簡潔に記すために必要な, 記法と定義を記す. なお以下の定義に関しては, [5] の論文の記法を参考にした.

記法 1. $\widetilde{R} \in \mathbb{C}^{N\times s}$, $\boldsymbol{v} \in \mathbb{C}^N$ として, もし $\boldsymbol{v}$ が $\widetilde{R}$ の列ベクトルすべてと直交している時, "$\boldsymbol{v} \perp \widetilde{R}$" と記し, $\boldsymbol{v}$ は $\widetilde{R}$ と直交しているという.

記法 2. C を $N \times s$ 行列とした時, $C\boldsymbol{e}_j$ ($j = 1, 2, ..., s$) は C の j 番目の列ベクトルを表す.

記法 3. $\boldsymbol{u} = \boldsymbol{v} - C\boldsymbol{\beta}$ といった, ベクトルの更新がこの論文においては, 重要な役目を果たしている. ここで $\boldsymbol{u}, \boldsymbol{v}$ は N 次元ベクトル, C は, $N \times s$ 行列であり, $\boldsymbol{\beta}$ は s 次元の係数ベクトルである. $\boldsymbol{v}, C$ が与えられている状態で, 更新後のベクトル $\boldsymbol{u}$ が, $N \times s$ の行列 $\widetilde{R}$ と直交するように係数ベクトル $\boldsymbol{\beta}$ を定めた上で, ベクトル $\boldsymbol{u}$ を更新するとき, 以下のように記述する:

$$\boldsymbol{u} = \boldsymbol{v} - C\boldsymbol{\beta} \text{ such that } \boldsymbol{u} \perp \widetilde{R}.$$

同様にして, 更新後のベクトルに, A を作用させたベクトル $A\boldsymbol{u}$ が $\widetilde{R}$ と直交するように係数ベクトル $\boldsymbol{\beta}$ を定めた上で, ベクトル $\boldsymbol{u}$ を更新するときは, 以下のように記述する:

$$\boldsymbol{u} = \boldsymbol{v} - C\boldsymbol{\beta} \text{ such that } A\boldsymbol{u} \perp \widetilde{R}.$$

定義 1. B を $N \times N$ の行列, $\widetilde{R}$ を $N \times s$ の行列とした時, 空間 $\mathcal{K}_k(B, \widetilde{R})$ を以下のように定義する:

$$\mathcal{K}_k(B, \widetilde{R}) \equiv \left\{ \sum_{j=0}^{k-1} B^j \widetilde{R} \boldsymbol{\gamma}_j \mid \boldsymbol{\gamma}_j \in \mathbb{C}^s \right\}.$$

$s = 1$ の時は, Krylov 部分空間そのものである. このように定義される Krylov 部分空間を block Krylov subspaces と言う [1].

2 Bi-CG 法と Bi-CGSTAB(L) 法

2.1 Bi-CG 法

Bi-CG 法は, $\boldsymbol{r}_k \perp \mathcal{K}_k(A^*, \tilde{\boldsymbol{r}}_0)$, $\boldsymbol{x}_k \in \mathcal{K}_k(A, \boldsymbol{r}_0)$ となる解ベクトル $\boldsymbol{x}_k$ と残差ベクトル $\boldsymbol{r}_k$ を, 2 組の 2 項漸化式によって求める方法である. 具体的には, 解ベクトルと残差ベクトルは以下のような漸化

式によって更新される:

$$\begin{cases} r_{k+1} = r_k - \alpha_k A u_k \ \text{ such that } \ r_{k+1} \perp \tilde{r}_k, \\ x_{k+1} = x_k + \alpha_k u_k, \\ \tilde{r}_{k+1} = \tilde{r}_k - \alpha_k A^* \tilde{u}_k, \\ u_{k+1} = r_{k+1} - \beta_{k+1} u_k \ \text{ such that } \ A u_{k+1} \perp \tilde{r}_k, \\ \tilde{u}_{k+1} = \tilde{r}_{k+1} - \beta_{k+1} \tilde{u}_k. \end{cases} \tag{1}$$

この漸化式によって生成される残差ベクトル r_k, 補助ベクトル u_k は以下の式を満たす:

$$r_k, A u_k \perp \mathcal{K}_k(A^*, \tilde{r}_0). \tag{2}$$

この関係式, 特に r_k に関するものから, $k = N$ とすると $\mathcal{K}_N(A^*, \tilde{r}_0) = \mathbb{C}^N$ となり $r_N = \mathbf{0}$, つまり $k \leq N$ において解が求められることが導かれる. ただ, 上の式において, $\tilde{r}_k$ の更新には任意性が存在する. $\tilde{q}_k(t)$ を最高次数の係数が 0 でない k 次の多項式として $\tilde{s}_k := \tilde{q}_k(A^*)\tilde{r}_0$ と定義した時, $\tilde{r}_k$ を $\tilde{s}_k$ で置き換えたとしても以下の式で生成される残差ベクトルは, 式 (1) で生成される残差ベクトルと一致することが知られている (但し, 丸め誤差は考えない):

$$\begin{cases} r_{k+1} = r_k - \alpha_k A u_k \ \text{ such that } \ r_{k+1} \perp \tilde{s}_k, \\ u_{k+1} = r_{k+1} - \beta_{k+1} u_k \ \text{ such that } \ A u_{k+1} \perp \tilde{s}_k. \end{cases} \tag{3}$$

この時, 式 (1) の係数 α_k, β_{k+1} は以下によって計算できる:

$$\begin{cases} \alpha_k &= \dfrac{(r_k, \tilde{s}_k)}{(A u_k, \tilde{s}_k)}, \\ \beta_{k+1} &= \dfrac{(A r_{k+1}, \tilde{s}_k)}{(A u_k, \tilde{s}_k)} = \dfrac{\tau_k}{\tau_{k+1}} \dfrac{(r_{k+1}, \tilde{s}_{k+1})}{(A u_k, \tilde{s}_k)}. \end{cases} \tag{4}$$

ただし, τ_k は多項式 $\tilde{q}_k(t)$ の最高次数の係数である.

現在よく用いられている Bi-CG 法を拡張した算法は, 上記の $\tilde{r}_k$ の任意性および, ベクトルの内積に関する 以下の関係式を利用して導かれる:

$$(r_k, \tilde{s}_k) = (r_k, \tilde{q}_k(A^*)\tilde{r}_0) = (\tilde{q}_k(A) r_k, \tilde{r}_0), \ (A u_k, \tilde{s}_k) = (A \tilde{q}_k(A) u_k, \tilde{r}_0). \tag{5}$$

実際, 以下の方針で Bi-CG 法を拡張することが可能である. (i) r_k, $u_k, x_k, \tilde{s}_k$ の更新の代わりに, $\tilde{q}_k(A) r_k$, $\tilde{q}_k(A) u_k, x'_k$ の更新を行う. 初期の shadow residual $\tilde{r}_0$ は更新しない. x'_k は, 新たな残差 $\tilde{q}_k(A) r_k$ に対応する近似解とする. (ii) (i) において, 多項式 $\tilde{q}_k(A)$ は, 新たな残差の収束性を改善するように設定を行う. (iii) (i) の $\tilde{q}_k(A) r_k$, $\tilde{q}_k(A) u_k$ の更新の中で, 多項式を除いた部分である r_k, u_k の更新に関しては, 式 (4), (5) より $\tilde{q}_k(A) r_k$, $\tilde{q}_k(A) u_k$, $\tilde{r}_0$ を使って, Bi-CG 法の係数 α_k, β_{k+1} を算出する.

この方針により様々な解法が発見された [6, 8, 4, 11]. 次に, それらの解法の中で, 本論文と最も関係のある Bi-CGSTAB(L) 法について解説を行う.

2.2 Bi-CGSTAB(L) 法

Bi-CGSTAB(L) 法は, Bi-CG 法の残差ベクトルに高次の安定化多項式を付加した残差ベクトルを算出するアルゴリズムである. 高次の安定化多項式は, 残差のノルムを局所的に最小化するように

設定される. 以下, その概要を説明する. $k = mL$ とする. まず, $p_i(t)$ $(i = 1, 2, ...)$ を $p_i(0) = 1$ を満たす L 次の多項式と定義する. この時, k 次の多項式 Q_k を以下のように定義する:

$$Q_k(t) = p_m(t) \dots p_2(t) p_1(t).$$

さらに, 前の章で定義した Bi-CG における残差ベクトル $\boldsymbol{r}_k$, 補助ベクトル $\boldsymbol{u}_k$ をそれぞれ $\boldsymbol{r}_k^{\mathrm{B}}$, $\boldsymbol{u}_k^{\mathrm{B}}$ と置き直した上で, 以下の残差ベクトルと補助ベクトルを定義する:

$$\boldsymbol{r}_k \equiv Q_k(A)\boldsymbol{r}_k^{\mathrm{B}}, \quad \boldsymbol{u}_{k-1} \equiv Q_k(A)\boldsymbol{u}_{k-1}^{\mathrm{B}}.$$

また, 解ベクトル $\boldsymbol{x}_k$ は, 上記で定義した残差ベクトル $\boldsymbol{r}_k$ に対応する近似解とする. Bi-CGSTAB(L) 法は, 初期の残差ベクトルと解ベクトルと補助ベクトルの組 $[\boldsymbol{r}_0, \boldsymbol{x}_0, \boldsymbol{u}_{-1}]$ からスタートして, $[\boldsymbol{r}_L, \boldsymbol{x}_L, \boldsymbol{u}_{L-1}]$, $[\boldsymbol{r}_{2L}, \boldsymbol{x}_{2L}, \boldsymbol{u}_{2L-1}]$,... と, L ごとに更新していく仕組みである. 1 反復のアルゴリズムは, 前半は Bi-CG part, 後半は MR part と分かれている (図 1 参照).

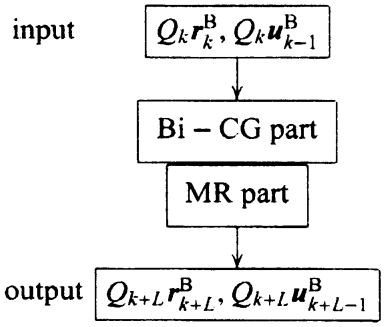

図 1　Bi-CGSTAB(L) の 1 反復

Bi-CG part では, Bi-CG 法における係数 β_k, α_k, β_{k+1}, α_{k+1},... , β_{k+L-1}, α_{k+L-1} の計算と, ベクトルの更新を行っていくことにより, 入力 $Q_k\boldsymbol{r}_k^{\mathrm{B}}, Q_k\boldsymbol{u}_{k-1}^{\mathrm{B}}$ に対して, $Q_k\boldsymbol{r}_{k+L}^{\mathrm{B}}, \dots, A^L Q_k\boldsymbol{r}_{k+L}^{\mathrm{B}}$ と $Q_k\boldsymbol{u}_{k+L-1}^{\mathrm{B}}, \dots, A^L Q_k\boldsymbol{u}_{k+L-1}^{\mathrm{B}}$ を出力する.

MR part は, Bi-CG part で得られたベクトルをもとに, $\boldsymbol{r}_{k+L}$と$\boldsymbol{u}_{k+L-1}$ を構成する. その際に, $p_{m+1}(t) = 1 - \gamma_1^{(m+1)}t - \cdots - \gamma_L^{(m+1)}t^L$ の係数 $\gamma_1^{(m+1)}, \dots, \gamma_L^{(m+1)}$ を決める自由度が存在するが, $\boldsymbol{r}_{k+L}$ のノルムが最小になるように設定する. すなわち

$$\text{find } \gamma_1^{(m+1)}, \dots, \gamma_L^{(m+1)} \quad \text{such that } \min \|Q_k\boldsymbol{r}_{k+L}^{\mathrm{B}} - \sum_{i=1}^{L} \gamma_i^{(m+1)} A^i Q_k\boldsymbol{r}_{k+L}^{\mathrm{B}}\|_2$$

により定める. $p_{m+1}(t)$ が決まると, $Q_{k+L}(t) = p_{m+1}(t)Q_k(t)$ より,

$$\begin{cases} Q_{k+L}\boldsymbol{r}_{k+L}^{\mathrm{B}} & = Q_k\boldsymbol{r}_{k+L}^{\mathrm{B}} - \sum_{i=1}^{L} \gamma_i^{(m+1)} A^i Q_k\boldsymbol{r}_{k+L}^{\mathrm{B}}, \\ Q_{k+L}\boldsymbol{u}_{k+L-1}^{\mathrm{B}} & = Q_k\boldsymbol{u}_{k+L-1}^{\mathrm{B}} - \sum_{i=1}^{L} \gamma_i^{(m+1)} A^i Q_k\boldsymbol{u}_{k+L-1}^{\mathrm{B}} \end{cases}$$

と, Bi-CG part で算出したベクトルを使い, 次の残差ベクトルと補助ベクトル $\boldsymbol{r}_{k+L}, \boldsymbol{u}_{k+L-1}$ を算出する. アルゴリズムの詳細に関しては [4] を参考にされたい.

3 高次元 shadow residual を持つ Bi-CG 法

まず，$\widetilde{R}_0$ を $N \times s$ のフルランク行列とする．Bi-CG 法において，k 反復後の残差 $\boldsymbol{r}_k$ に課される条件は，

$$\boldsymbol{r}_k \perp \mathcal{K}_k(A^*, \widetilde{r}_0)$$

であったが，高次元 shadow residual を持つ Bi-CG 法においては，残差に条件：

$$\boldsymbol{r}_k \perp \mathcal{K}_k(A^*, \widetilde{R}_0) \tag{6}$$

を課す．以下，この条件を満たすアルゴリズムを２つ導入する．

3.1 Bi-CG(s) 法

2008 年，Sleijpen らは，[5] において高次元 shadow residual を持つ Bi-CG 法 (以下 Bi-CG(s) 法[*1] と呼ぶ) を導入した．Bi-CG(s) 法では，shadow residual の高次元化に伴って，Bi-CG 法のアルゴリズム (式 (1)) の補助ベクトルを高次元化する ($\boldsymbol{u}_k$ を $N \times s$ 行列 U_k とする) ことにより，条件 (6) を満たす残差を算出する (アルゴリズム 1)．

アルゴリズム 1　Bi-CG(s) アルゴリズム

set $\boldsymbol{x}_0$ and calculate $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$
set $N \times s$ matrix $U_0 = [\boldsymbol{r}_0, A\boldsymbol{r}_0, ..., A^{s-1}\boldsymbol{r}_0]$
$k = 0$
while $\|\boldsymbol{r}_k\| > \varepsilon$
  $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - AU_k\alpha_k$ such that $\boldsymbol{r}_{k+1} \perp \widetilde{R}_k$
  $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + U_k\alpha_k$
  $\boldsymbol{v} = \boldsymbol{r}_{k+1}$
  for $j = 1, ..., s$
    $U_{k+1}\boldsymbol{e}_j = \boldsymbol{v} - U_k\boldsymbol{\beta}_{k+1}^{(j)}$ such that $AU_{k+1}\boldsymbol{e}_j \perp \widetilde{R}_k$
    $\boldsymbol{v} = AU_{k+1}\boldsymbol{e}_j$
  end for
  $k = k + 1$, update $\widetilde{R}_k$
end while

ここで，高次元 shadow residual $\widetilde{R}_k$ ($k = 0, 1, \cdots, K-1$) は，列ベクトル $\widetilde{R}_0, ..., \widetilde{R}_{K-1}$ が張る空間が，$\mathcal{K}_K(A^*, \widetilde{R}_0)$ と一致するようにとる．具体的には，最も単純には $\widetilde{R}_k = (A^*)^k\widetilde{R}_0$，他には，$\Phi_k(t)$ を k 次多項式（最高次係数は非零とする）として $\widetilde{R}_k = \Phi_k(A^*)\widetilde{R}_0$ などとする．

[*1] [5] では，対応するアルゴリズムは Bi-CG 法の拡張とだけ書かれていて，名前はついていない．この論文においては便宜上 Bi-CG(s) 法と呼ぶ．

このアルゴリズムにより生成される残差ベクトルは, 以下の性質を持つことが分かっている.

命題 1 ([5]). Bi-CG(s) 法は, k 反復までアルゴリズムが破綻せず, $0 \le i < k$ において, $\alpha_i(s)$ (s 次元ベクトル α_i の第 s 成分)$\neq 0$ が満たされるとする. この時以下が成り立つ.
a) $r_k \in \mathcal{K}_{ks+1}(A, r_0) - \mathcal{K}_{ks}(A, r_0)$.
b) $r_k, AU_k e_j \perp \mathcal{K}_k(A^*, \widetilde{R}_0)$.

命題 1 の仮定は, generic という条件下で, 真の解を与えるまで満たされると期待される. 仮定が満たされる時, Bi-CG(s) 法が真の解を与えるまでに必要な行列ベクトル積の演算回数を見積る. まず, b) より $r_{\lceil N/s \rceil} = \mathbf{0}$ が分かる. また, 1 反復あたり行列ベクトル積は $2s$ 回必要である. よって, 高々 $2s \times N/s = 2N$ 回の行列ベクトル積演算で Bi-CG(s) 法は真の解を与える.

3.2 GBi-CG(s) 法

Bi-CG(s) 法において, $k+1$ 反復目の補助ベクトル $U_{k+1}e_j$ の更新で用いる s 本の補助ベクトル U_k は, 計算過程で新しく算出されるベクトルをすぐ用いることが可能である. そのように $U_{k+1}e_j$ を更新するようにしたアルゴリズムが以下である (以下 GBi-CG(s) 法と呼ぶ). このアルゴリズムは, Bi-CG(s) 法と比べて, 新しいベクトルを用いている分だけ, 数値的な安定性が増していると期待される.

アルゴリズム 2　GBi-CG(s) アルゴリズム

set x_0 and calculate $r_0 = b - Ax_0$
set $N \times s$ matrix $U_0 = [r_0, Ar_0, ..., A^{s-1}r_0]$
$k = 0$
while $\|r_k\| > \varepsilon$
  $r_{k+1} = r_k - AU_k\alpha_k$ such that $r_{k+1} \perp \widetilde{R}_k$
  $x_{k+1} = x_k + U_k\alpha_k$
  $U_{k+1}e_1 = r_{k+1} - U_k\beta_k^{(1)}$ such that $AU_{k+1}e_1 \perp \widetilde{R}_k$
  $v = AU_{k+1}e_1$
  for $j = 2, ..., s$
    set $U_k^{(j)} = [r_{k+1}, AU_{k+1}e_1, ..., AU_{k+1}e_{j-2}, U_ke_j, ..., U_ke_s]$
    $U_{k+1}e_j = v - U_k^{(j)}\beta_{k+1}^{(j)}$ such that $AU_{k+1}e_j \perp \widetilde{R}_k$
    $v = AU_{k+1}e_j$
  end For
  $k = k + 1$, update $\widetilde{R}_k$
end while

この GBi-CG(s) 法に対して, Bi-CG(s) 法に対する命題 1 に対応して以下が成り立つ.

命題 2. GBi-CG(s) 法のアルゴリズムにおいて, $\sigma_k := \widetilde{R}_k^* A U_k$, $\sigma_k^{(j)} := \widetilde{R}_k^* A U_k^{(j)}$ と定義する. また, U_k' を $N \times s$ の行列で, $U_k' := [r_k, AU_k e_1, ..., AU_k e_{s-1}]$ と定義した時, $\sigma_k' := \widetilde{R}_k^* U_k'$ と定義する. この時, $i < k$ となるすべての i に対して $\sigma_i, \sigma_i', \sigma_i^{(j)} (j = 2, ..., s)$ が正則であるならば, 以下が成り立つ:
a) k 反復までアルゴリズムは破綻しない.
b) $r_k \in \mathcal{K}_{ks+1}(A, r_0) - \mathcal{K}_{ks}(A, r_0)$.
c) $AU_k e_j \ (j = 1, ..., s) \in \mathcal{K}_{ks+j+1}(A, r_0) - \mathcal{K}_{ks+j}(A, r_0)$.
d) r_k, $AU_k e_j \ (j = 1, ..., s) \perp \mathcal{K}_k(A^*, \widetilde{R}_0)$.

命題 2 の仮定は, generic という条件下で, 真の解が求まるまで満たすと期待される. また, 仮定が満たされる時, GBi-CG(s) 法が真の解を与えるまでに必要な行列ベクトル積の演算回数は, Bi-CG(s) 法と全く同様の議論により, 高々 $2s \times N/s = 2N$ 回となることが分かる.

4 GBi-CGSTAB(s, L) 法

この章では, 前章で定義した GBi-CG(s) 法に対して, L 次の安定化多項式を付加したアルゴリズム, GBi-CGSTAB(s, L) 法を導く. ここで, GBi-CGSTAB(s, L) 法に L 次安定化多項式を付加する手順は, Bi-CG 法から, 高次の安定化多項式を付加した Bi-CGSTAB(L) 法を導いたのと同じである.

4.1 概要

この章において, 前章の GBi-CG(s) のアルゴリズムにおける $U_k, U_k^{(j)}, r_k$ をそれぞれ $U_k^{GB}, U_k^{(j),GB}, r_k^{GB}$ と記す. さらに, 以下 $k = mL$ として, $p_i(t)$ $(i = 1, 2, ...)$ を L 次の多項式で $p_i(0) = 1$ を満たすものとする. この時, k 次多項式 $Q_k(t)$ を $Q_k(t) = p_m(t) \cdots p_2(t) p_1(t)$ と定義する. さらに, Bi-CGSTAB(L) と同様に, 以下のようなベクトル, および行列を定義する:

$$r_k \equiv Q_k(A) r_k^{GB}, \quad U_{k-1} \equiv Q_k(A) U_{k-1}^{GB}.$$

また, 解ベクトル x_k は, 上記で定義した残差ベクトル r_k に対応する近似解とする. $k = mL$ とした時, GBi-CGSTAB(s, L) 法は, 初期の残差ベクトル, 解ベクトル, 補助ベクトル s 本の組 $[r_0, x_0, U_{-1}]$ からスタートして, $[r_L, x_L, U_{L-1}]$, $[r_{2L}, x_{2L}, U_{2L-1}]$ と L ごとに残差ベクトル, 解ベクトル, 補助ベクトルを算出するアルゴリズムになっていて, 前半部分の GBi-CG(s) part と後半部分の MR part により構成される (図 2).

GBi-CG(s) part は, i-th step iteration を $i = 0, ..., L-1$ まで順に行うことで入力 $Q_k r_k^{GB}, Q_k U_{k-1}^{GB}$ に対して, $Q_k r_{k+L}^{GB}, ..., A^L Q_k r_{k+L}^{GB}$ と $Q_k U_{k+L-1}^{GB}, ..., A^L Q_k U_{k+L-1}^{GB}$ を出力する (図 3).

MR part は, GBi-CG(s) part で得られたベクトルを元に, r_{k+L} と U_{k+L-1} を構成する (図 4). その際に, $p_{m+1}(t) = 1 - \gamma_1^{(m+1)} t - \cdots - \gamma_L^{(m+1)} t^L$ の係数 $\gamma_1^{(m+1)}, ..., \gamma_L^{(m+1)}$ を決める自由度が存在するが, それ

は, r_{k+L} のノルムが最小になるように設定する. すなわち

$$\text{find } \gamma_1^{(m+1)}, ..., \gamma_L^{(m+1)} \quad \text{such that } \min \|Q_k r_{k+L}^{GB} - \sum_{i=1}^{L} \gamma_i^{(m+1)} A^i Q_k r_{k+L}^{GB}\|_2$$

により定める. $p_{m+1}(t)$ が決まると, $Q_{k+L}(t) = p_{m+1}(t)Q_k(t)$ より,

$$\begin{cases} Q_{k+L} r_{k+L}^{GB} & = Q_k r_{k+L}^{GB} - \sum_{i=1}^{L} \gamma_i^{(m+1)} A^i Q_k r_{k+L}^{GB}, \\ Q_{k+L} U_{k+L-1}^{GB} & = Q_k U_{k+L-1}^{GB} - \sum_{i=1}^{L} \gamma_i^{(m+1)} A^i Q_k U_{k+L-1}^{GB} \end{cases}$$

と, 今まで算出したベクトルを使い, r_{k+L}, U_{k+L-1} が算出できる.

4.2 アルゴリズムの詳細

複雑な GBi-CG(s) part について, 詳細に述べる.

GBi-CG(s) part における i-th step iteration は, input : $A^j Q_k r_{k+i}^{GB}, A^j Q_k U_{k+i-1}^{GB}$ $(j = 0, ..., i)$ に対して, output : $A^j Q_k r_{k+i+1}^{GB}, A^j Q_k U_{k+i}^{GB}$ $(j = 0, ..., i+1)$ を算出する (図 5).

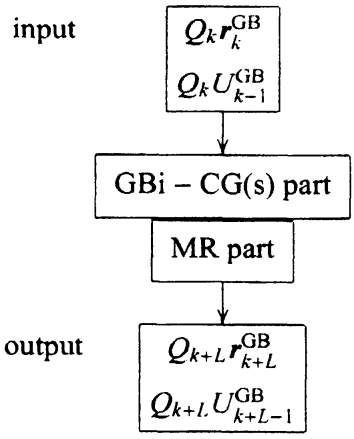

図 2 GBi-CGSTAB(s, L) の 1 反復

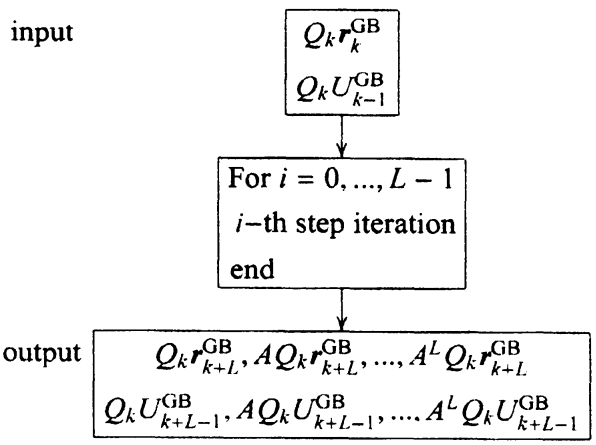

図 3 GBi-CG(s) part の概要

ここで, i-th step iteration の更新の仕組みについて説明を行う. i-th step iteration では, GBi-CG(s) アルゴリズムの係数ベクトルを計算した後, GBi-CG(s) のアルゴリズムに沿って, ベクトルを更新をすることを繰り返す. その際に, 係数ベクトル $\beta_{k+i}^{(1)}, ..., \beta_{k+i}^{(s)}, \alpha_{k+i}$ の算出法が問題になるが, それに関しては後に補足説明を行うことにして, ここでは係数ベクトルは与えられるとして概要を説明する.

まずアルゴリズム 2 における係数 $\beta_{k+i}^{(1)}$ を計算する. 次に $A^j Q_k U_{k+i-1}^{GB} e_1$ $(j = 0, ..., i)$ に対して, 演算:$A^j Q_k U_{k+i}^{GB} e_1 = A^j Q_k r_{k+i}^{GB} - A^j Q_k U_{k+i-1}^{GB} \beta_{k+i}^{(1)}$ によって, それぞれ, $A^j Q_k U_{k+i-1}^{GB} e_1 \rightarrow A^j Q_k U_{k+i}^{GB} e_1$ $(j = 0, ..., i)$ と更新する. さらに, 更新後のベクトル, $A^i Q_k U_{k+i}^{GB}$ に A を作用させたベクトル, $A^{i+1} Q_k U_{k+i}^{GB}$ を保存する. この時点で, $Q_k U_{k+i-1}^{(2),GB}, ..., A^i Q_k U_{k+i-1}^{(2),GB}$ が利用可能な状態であることに注意する.

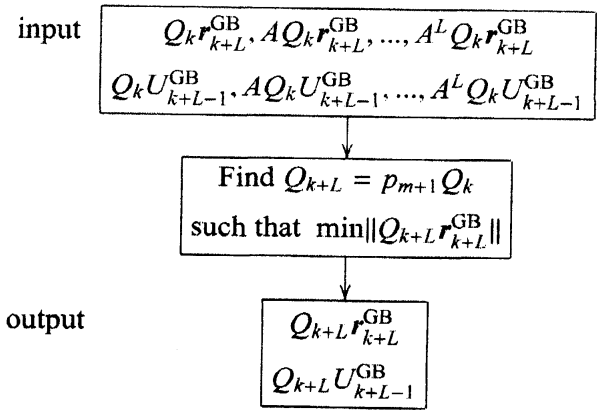

図 4 GBi-CGSTAB(s, L) における MR(minimum residual) part の概要

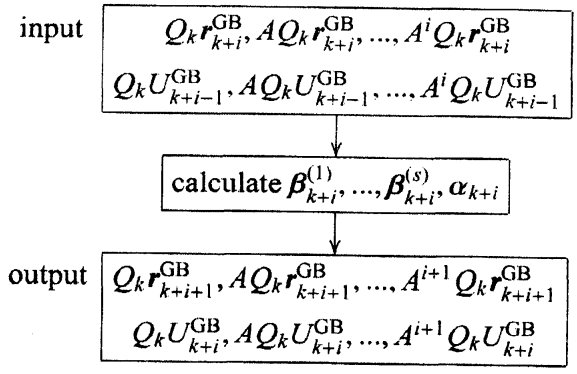

図 5 GBi-CG(s) part における i-th step iteration の概要

続いて, アルゴリズム 2 における, 係数 $\beta^{(2)}_{k+i}$ を計算する. そして現在利用可能なベクトルを使用して, $A^j Q_k U^{GB}_{k+i-1} e_2$ $(j = 0, ..., i)$ に対して, 演算: $A^j Q_k U^{GB}_{k+i} e_2 = A^{j+1} Q_k U^{GB}_{k+i+1} e_1 - A^j Q_k U^{(2),GB}_{k+i-1} \beta^{(2)}_{k+i}$ を行うことによって, それぞれ, $A^j Q_k U^{GB}_{k+i-1} e_2 \rightarrow A^j Q_k U^{GB}_{k+i} e_2$ $(j = 0, ..., i)$ と更新する. さらに更新後のベクトル $A^i Q_k U^{GB}_{k+i} e_2$ に A を作用させたベクトル, $A^{i+1} Q_k U^{GB}_{k+i} e_2$ を保存する. この時点で, $Q_k U^{(3),GB}_{k+i-1}, ..., A^i Q_k U^{(3),GB}_{k+i-1}$ が利用可能な状態となる.

以下同様に行い, $A^j Q_k U^{GB}_{k+i} e_1$, $A^j Q_k U^{GB}_{k+i} e_2$,..., $A^j Q_k U^{GB}_{k+i} e_s$ と順に更新を行っていくことにより, 最終的に, $A^j Q_k U^{GB}_{k+i}$ $(j = 0, ..., i+1)$ を得る. U に関する更新が終わったら, r の更新を行うために, アルゴリズム 2 の係数ベクトル α_{k+i} の計算を行う. 次に $j = 0, ..., i$ において, 演算: $A^j Q_k r^{GB}_{k+i+1} = A^j Q_k r^{GB}_{k+i} - A^{j+1} Q_k U^{GB}_{k+i} \alpha_{k+i}$ を行うことにより, $A^j Q_k r^{GB}_{k+i} \rightarrow A^j Q_k r^{GB}_{k+i+1} (j = 0, ..., i)$ と更新する. さらに, 更新後のベクトル $A^i Q_k r^{GB}_{k+i+1}$ に A を作用させたベクトル $A^{i+1} Q_k r^{GB}_{k+i+1}$ を新たに保存することにより, output を得る. i-th step において, 注目すべきなのは, 行列ベクトル積の演算を $s+1$ 回行うことで, output を得ることができる点である.

この一連の流れにより, 残差ベクトル r_k から, 残差ベクトル r_{k+L} を新たに生成できることは容易にわかり, また GBi-CG(s)part において, 必要な行列ベクトル積の演算は $(s+1)L$ 回である.

ここで, 係数ベクトル $\alpha_k, \beta^{(t)}_k$ の計算法に関して補足を行う. アルゴリズムでは, $s \times s$ 行列 M と s 次元ベクトル m を用いることで係数の計算を行う. α_k の決定法に関しては比較的容易であることから, ここでは余白の関係上 $\beta^{(1)}_k$ と $\beta^{(t)}_k$ $(t = 2, ..., s)$ の決定法に関してのみ説明を行う.

$\beta^{(1)}_k$ の決定法

[$i > 0$ の時] $\beta^{(1)}_{k+1}$ を算出する段階において, 利用可能なベクトルは $A^j Q_k r^{GB}_{k+i}$ $(j = 0, ..., i)$ と $A^j Q_k U^{GB}_{k+i-1} e_t$ $(j = 0, ..., i)$ $(t = 1, ..., s)$ である. その中でも $A^i Q_k U^{GB}_{k+i-1}$ に着目する. $A^i Q_k r^{GB}_{k+i} - A^i Q_k U^{GB}_{k+i-1} \beta \perp \widetilde{R}_0$ となる, β は, アルゴリズム 2 において, $\widetilde{R}_{k+i-1} = (A^*)^{i-1} Q_k (A^*) \widetilde{R}_0$ と設定した際の, $A r^{GB}_{k+i} - A U^{(1)}_{k+i-1} \beta^{(1)}_{k+i} \perp \widetilde{R}_{k+i-1}$ と一致する. よって, 係数ベクトル $\beta^{(1)}_{k+i}$ は, $\beta^{(1)}_{k+i} =$

$(\widetilde{R}_0^* A^i Q_k U_{k+i-1}^{\mathrm{GB}})^{-1}(\widetilde{R}_0^* A^i Q_k \boldsymbol{r}_{k+i})$ を計算することにより求まる. 後述のアルゴリズムにおいては, $\boldsymbol{m} = \widetilde{R}_0^* A^i Q_k \boldsymbol{r}_{k+i}^{\mathrm{GB}}$, $M\boldsymbol{e}_t = \widetilde{R}_0^* A^i Q_k U_{k+i-1}^{\mathrm{GB}} \boldsymbol{e}_t$ $(t = 1, ..., s)$ と対応しており, $\boldsymbol{\beta}_{k+i}^{(1)} = M^{-1}\boldsymbol{m}$ により計算を行っている.

[$i = 0$ の時] $i = 0$ の場合は, 1 つ前の反復において生成されたベクトル $A^L Q_{k-L} \boldsymbol{r}_k^{\mathrm{GB}}$ と $A^L Q_{k-L} U_{k-1}^{\mathrm{GB}}$ に着目する. この時, $A^L Q_{k-L} \boldsymbol{r}_k^{\mathrm{GB}} - A^L Q_{k-L} U_{k-1}^{\mathrm{GB}} \boldsymbol{\beta} \perp \widetilde{R}_0$ を満たす $\boldsymbol{\beta}$ は, $A\boldsymbol{r}_k^{\mathrm{GB}} - AU_{k-1}^{\mathrm{GB}} \boldsymbol{\beta}_k^{(1)} \perp (A^*)^{L-1} Q_{k-L}(A^*)\widetilde{R}_0$ を満たす $\boldsymbol{\beta}_k$ と一致しており, この式は, アルゴリズム 2 において, $\widetilde{R}_{k-1} = (A^*)^{L-1} Q_{k-L}(A^*)\widetilde{R}_0$ と設定した場合における更新:$U_{k-1}^{\mathrm{GB}} \boldsymbol{e}_1 \to U_k^{\mathrm{GB}} \boldsymbol{e}_1$ が満たす関係式そのものである. 従って $\boldsymbol{\beta}_k^{(1)} = (\widetilde{R}_0^* A^L Q_{k-L} U_{k-1}^{\mathrm{GB}})^{-1}(\widetilde{R}_0^* A^L Q_{k-L} \boldsymbol{r}_k^{\mathrm{GB}})$ により, $\boldsymbol{\beta}_k^{(1)}$ を得ることができる. 後述のアルゴリズムにおいては, 命題 2 の d) より $\boldsymbol{m} = -\gamma_L^{(m)} \widetilde{R}_0^* A^L Q_{k-L} \boldsymbol{r}_k^{\mathrm{GB}} = \widetilde{R}_0^* Q_k \boldsymbol{r}_k^{\mathrm{GB}}$, $M = -\gamma_L^{(m)} \widetilde{R}_0^* A^L Q_{k-L} U_{k-1}^{\mathrm{GB}}$ と対応しており, 演算 $\boldsymbol{\beta}_k^{(1)} = M^{-1}\boldsymbol{m}$ によって, 係数ベクトル $\boldsymbol{\beta}_k^{(1)}$ を得る.

$\boldsymbol{\beta}_k^{(t)}$ の決定 $(t = 2, ..., s)$

[$i > 0$ の時] $\boldsymbol{\beta}_{k+1}^{(t)}$ を算出する段階において, 利用可能なベクトルは $A^j Q_k \boldsymbol{r}_{k+i}^{\mathrm{GB}}$ $(j = 0, ..., i)$ と $A^j Q_k U_{k+i}^{\mathrm{GB}} \boldsymbol{e}_\nu$ $(j = 0, ..., i+1)$ $(\nu = 1, ..., t-1)$ と $A^j Q_k U_{k+i-1}^{\mathrm{GB}} \boldsymbol{e}_\nu$ $(j = 0, ..., i)$ $(\nu = t, ..., s)$ である. よって, $A^j Q_k U_{k+i-1}^{(t),\mathrm{GB}}$ $(j = 0, ..., i)$ もまた利用可能であり, その中でも $A^i Q_k U_{k+i-1}^{(t),\mathrm{GB}}$ に着目する. $A^{i+1} Q_k U_{k+i}^{\mathrm{GB}} \boldsymbol{e}_{t-1} - A^i Q_k U_{k+i-1}^{(t),\mathrm{GB}} \boldsymbol{\beta} \perp \widetilde{R}_0$ となる $\boldsymbol{\beta}$ は, アルゴリズム 2 において, $\widetilde{R}_{k+i-1} = (A^*)^{i-1} Q_k(A^*)\widetilde{R}_0$ と設定した際の, $A^2 U_{k+i}^{\mathrm{GB}} \boldsymbol{e}_{t-1} - AU_{k+i-1}^{(t),\mathrm{GB}} \boldsymbol{\beta}_{k+i}^{(t)} \perp \widetilde{R}_{k+i-1}$ と一致する. よって, 係数ベクトル $\boldsymbol{\beta}_{k+i}^{(t)}$ は, $\boldsymbol{\beta}_{k+i}^{(t)} = (\widetilde{R}_0^* A^i Q_k U_{k+i-1}^{(t),\mathrm{GB}})^{-1}(\widetilde{R}_0^* A^{i+1} Q_k U_{k+i}^{\mathrm{GB}} \boldsymbol{e}_{t-1})$ を計算することにより求める. 後述のアルゴリズムにおいては, $\boldsymbol{m} = \widetilde{R}_0^* A^i Q_k \boldsymbol{r}_{k+i}^{\mathrm{GB}}$, $M\boldsymbol{e}_\nu = \widetilde{R}_0^* A^{i+1} Q_k U_{k+i}^{\mathrm{GB}} \boldsymbol{e}_\nu$ $(\nu = 1, ..., t-1)$, $M\boldsymbol{e}_\nu = \widetilde{R}_0^* A^i Q_k U_{k+i-1}^{\mathrm{GB}} \boldsymbol{e}_\nu$ $(\nu = t, ..., s)$ と対応しており, $\widetilde{R}_0^* A^i Q_k U_{k+i}^{(t),\mathrm{GB}} = [\boldsymbol{m}, M\boldsymbol{e}_1, ..., M\boldsymbol{e}_{t-2}, M\boldsymbol{e}_t, ..., M\boldsymbol{e}_s]$ であることから, 演算 $\boldsymbol{\beta}_{k+i}^{(t)} = [\boldsymbol{m}, M\boldsymbol{e}_1, ..., M\boldsymbol{e}_{t-2}, M\boldsymbol{e}_t, ..., M\boldsymbol{e}_s]^{-1} M\boldsymbol{e}_{t-1}$ によって, 係数ベクトル $\boldsymbol{\beta}_{k+i}^{(t)}$ を得る.

[$i = 0$ の時] $i = 0$ の場合は, 1 つ前の反復において生成されたベクトル $A^L Q_{k-L} \boldsymbol{r}_k^{\mathrm{GB}}$, $A^L Q_{k-L} U_{k-1}^{\mathrm{GB}} \boldsymbol{e}_\nu$ $(\nu = t, ..., s)$ と現反復において生成された $AQ_k U_k^{\mathrm{GB}} \boldsymbol{e}_\nu$ $(\nu = 1, ..., t-1)$ に着目する. この時, $A^{L+1} Q_{k-L} U_k^{\mathrm{GB}} \boldsymbol{e}_{t-1} - A^L Q_{k-L} U_{k-1}^{(t),\mathrm{GB}} \boldsymbol{\beta} \perp \widetilde{R}_0$ を満たす $\boldsymbol{\beta}$ は, $A^2 U_k^{\mathrm{GB}} \boldsymbol{e}_{t-1} - AU_{k-1}^{(t),\mathrm{GB}} \boldsymbol{\beta}_k^{(t)} \perp (A^*)^{L-1} Q_{k-L}(A^*)\widetilde{R}_0$ を満たす $\boldsymbol{\beta}_k$ と一致しており, この式は, アルゴリズム 2 において, $\widetilde{R}_{k-1} = (A^*)^{L-1} Q_{k-L}(A^*)\widetilde{R}_0$ と設定した場合における, 更新:$U_{k-1}^{\mathrm{GB}} \boldsymbol{e}_t \to U_k^{\mathrm{GB}} \boldsymbol{e}_t$ が満たす関係式そのものである. 従って $\boldsymbol{\beta}_k^{(t)} = (\widetilde{R}_0^* A^L Q_{k-L} U_{k-1}^{(t),\mathrm{GB}})^{-1}(\widetilde{R}_0^* A^{L+1} Q_{k-L} \boldsymbol{r}_k^{\mathrm{GB}})$ であるが, この時, $\widetilde{R}_0^* A^L Q_{k-L} U_{k-1}^{(t),\mathrm{GB}}$ の $1, ..., t-1$ 列目の演算, $\widetilde{R}_0^* A^{L+1} Q_{k-L} U_k^{\mathrm{GB}} \boldsymbol{e}_\nu$ $(\nu = 1, ..., t-1)$ は, 陽には分からない. しかし, 最高次の係数に着目すると, 命題 2 の d) より, $-\gamma_L^{(m)}(\widetilde{R}_0^* A^{L+1} Q_{k-L} U_k^{\mathrm{GB}} \boldsymbol{e}_\nu) = \widetilde{R}_0^* AQ_k U_k^{\mathrm{GB}} \boldsymbol{e}_\nu$ $(\nu = 1, ..., t-1)$ が言える. 後述のアルゴリズムにおいては, $M\boldsymbol{e}_\nu = \widetilde{R}_0^* AQ_k U_k^{\mathrm{GB}} \boldsymbol{e}_\nu = -\gamma_L^{(m)} \widetilde{R}_0^* A^{L+1} Q_{k-L} U_k^{\mathrm{GB}} \boldsymbol{e}_\nu$ $(\nu = 1, ..., t-1)$, $M\boldsymbol{e}_\nu = -\gamma_L^{(m)} \widetilde{R}_0^* A^L Q_{k-L} U_{k-1}^{\mathrm{GB}} \boldsymbol{e}_\nu$ $(\nu = t, ..., s)$, $\boldsymbol{m} = -\gamma_L^{(m)} \widetilde{R}_0^* A^L Q_{k-L} \boldsymbol{r}_k^{\mathrm{GB}}$ と対応しており, $-\gamma_L^{(m)} \widetilde{R}_0^* A^L Q_{k-L} U_k^{(t),\mathrm{GB}} = [\boldsymbol{m}, M\boldsymbol{e}_1, ..., M\boldsymbol{e}_{t-2}, M\boldsymbol{e}_t, ..., M\boldsymbol{e}_s]$ であることから, 演算 $\boldsymbol{\beta}_k^{(t)} = [\boldsymbol{m}, M\boldsymbol{e}_1, ..., M\boldsymbol{e}_{t-2}, M\boldsymbol{e}_t, ..., M\boldsymbol{e}_s]^{-1} M\boldsymbol{e}_{t-1}$ によって, 係数ベクトル $\boldsymbol{\beta}_k^{(t)}$ を得る.

これらをまとめると, GBi-CGSTAB(s, L) アルゴリズムが得られる:

アルゴリズム 3　GBi-CGSTAB(s, L) アルゴリズム

1. Select an $\boldsymbol{x}_0$, $N \times s$ matrices $\widetilde{R}_0$

2. Set $U_0 = [\boldsymbol{r}_0, A\boldsymbol{r}_0, ..., A^{s-1}\boldsymbol{r}_0]$, Compute $U_1 = AU_0$
3. $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$
4. $M = \widetilde{R}_0^* U_1$, $\boldsymbol{m} = \widetilde{R}_0^* \boldsymbol{r}_0$
5. Solve $M\boldsymbol{\gamma} = \boldsymbol{m}$ for $\boldsymbol{\gamma}$
6. $\boldsymbol{r}_0 \leftarrow \boldsymbol{r}_0 - U_1\boldsymbol{\gamma}$, $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}_0 + U_0\boldsymbol{\gamma}$
7. $\boldsymbol{r}_1 = A\boldsymbol{r}_0$, iter = 0, $\omega = -1$
8. repeat until $\|\boldsymbol{r}\| < \varepsilon$ (tolerance)
9. $M \leftarrow -\omega M$
10. For $i = 0, 1, ..., L-1$
11. If (iter = 0)$\cap$($i = 0$) $i = 1$
12. $\boldsymbol{m} \leftarrow \widetilde{R}_0^* \boldsymbol{r}_i$
13. For $j = 1, ..., s$
14. if($j = 1$)
15. Solve $M\boldsymbol{\gamma} = \boldsymbol{m}$ for $\boldsymbol{\gamma}$
16. $U_k\boldsymbol{e}_j \leftarrow \boldsymbol{r}_k - \sum_{q=1}^{s} U_k\boldsymbol{e}_q\gamma(q)$ $(k = 0, ..., i)$
17. else
18. Solve $[\boldsymbol{m}, M\boldsymbol{e}_1, ..., M\boldsymbol{e}_{j-2}, M\boldsymbol{e}_j, ..., M\boldsymbol{e}_s]\boldsymbol{\gamma} = M\boldsymbol{e}_{j-1}$ for $\boldsymbol{\gamma}$
19. $U_k\boldsymbol{e}_j \leftarrow U_{k+1}\boldsymbol{e}_{j-1} - \boldsymbol{r}_k\gamma(1) - \sum_{q=1}^{j-2} U_{k+1}\boldsymbol{e}_q\gamma(q+1) - \sum_{q=j}^{s} U_k\boldsymbol{e}_q\gamma(q)$ $(k = 0, ..., i)$
20. end
21. Compute $U_{i+1}\boldsymbol{e}_j = AU_i\boldsymbol{e}_j$
22. $M\boldsymbol{e}_j \leftarrow \widetilde{R}_0^* U_{i+1}\boldsymbol{e}_j$
23. end
24. Solve $M\boldsymbol{\gamma} = \boldsymbol{m}$ for $\boldsymbol{\gamma}$
25. $\boldsymbol{r}_k \leftarrow \boldsymbol{r}_k - U_{k+1}\boldsymbol{\gamma}$ $(k = 0, ..., i)$
26. $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}_0 + U_0\boldsymbol{\gamma}$
27. $\boldsymbol{r}_{i+1} = A\boldsymbol{r}_i$
28. end
29. For $j = 1, 2, ..., L$
30. $\tau_{ij} = \frac{1}{\sigma_i}(\boldsymbol{r}_i, \boldsymbol{r}_j)$, $\boldsymbol{r}_j = \boldsymbol{r}_j - \tau_{ij}\boldsymbol{r}_i$ $(i = 1, 2, ..., j-1)$
31. $\sigma_j = (\boldsymbol{r}_j, \boldsymbol{r}_j)$, $\gamma'_j = \frac{1}{\sigma_j} = (\boldsymbol{r}_j, \boldsymbol{r}_0)$
32. end
33. $\gamma_L = \gamma'_L$, $\omega = \gamma_L$
34. $\gamma_j = \gamma'_j - \sum_{i=j+1}^{L} \tau_{ji}\gamma_i$ $(j = L-1, ..., 1)$

35. $\gamma''_j = \gamma_{j+1} + \sum_{i=j+1}^{L-1} \tau_{ji}\gamma_{i+1}$ $(j = 1, ..., L-1)$
36. $x_0 = x_0 + \gamma_1 r_0$, $r_0 = r_0 - \gamma'_L r_L$, $U_0 = U_0 - \gamma_L U_L$
37. $U_0 = U_0 - \gamma_j U_j$ $(j = 1, ..., L-1)$
38. $x_0 = x_0 + \gamma''_j r_j$, $r_0 = r_0 - \gamma'_j r_j$ $(j = 1, ..., L-1)$
39. $r \leftarrow r_0$, iter $\leftarrow$iter $+(s+1)L$
40. end repeat

このアルゴリズムは，[10] に与えた一般化 IDR(s, L) 法のアルゴリズムに他ならない[*2]．しかし，ここに与えた導出法はかなり素直なものであり，アルゴリズムを解析する点からも有用であると期待される．

最後に，初期ベクトルの与え方に関して補足しておく．ここでは，アルゴリズムの記述の簡単化のため，初期補助ベクトルを $U_0 = [r_0, Ar_0..., A^{s-1}r_0]$ としたが，この方法では A を何度も作用させるため，丸め誤差の影響で列ベクトルの一次独立性が損なわれる恐れがある．したがって，$r_0, Ar_0..., A^{s-1}r_0$ を直交化して用いることが薦められる (後の数値実験でもそのようにした)．

4.3 計算量と必要メモリ

まず，GBi-CGSTAB(s, L) 法は，そのコアとなる GBi-CG(s) 法のアルゴリズム部分が破綻しなければ，決して破綻しないことが示せる．

つぎに，アルゴリズムが破綻しないという条件下で，GBi-CGSTAB(s, L) 法が真の解を与えるまでに必要な行列ベクトル積の演算回数を見積る．GBi-CG(s) 法の残差ベクトルに関して，$r^{GB}_{\lceil N/s \rceil} = \mathbf{0}$ となることから，GBi-CGSTAB(s, L) 法の残差ベクトルに関して $r_{\lceil N/s \rceil} = 0$ がいえる．また，GBi-CGSTAB(s, L) 法は，初期の残差ベクトル，解ベクトル，補助ベクトル s 本の組 $[r_0, x_0, U_{-1}]$ からスタートして，$[r_L, x_L, U_{L-1}]$, $[r_{2L}, x_{2L}, U_{2L-1}]$, $\cdots$ と L ごとに残差ベクトル，解ベクトル，補助ベクトルを算出するアルゴリズムになっており，かつ，この 1 ステップを進めるのに $(s+1)L$ 回の行列ベクトル積を要する．よって，高々 $(N/s) \times (1/L) \times (s+1)L = N + N/s$ の行列ベクトル積の演算で真の解が得られることが分かる．この結果は，GIDR(s, L) 法の文脈においても示されていたことであるが，結果の導出に違いがある．なお，この結果は，GBi-CGSTAB(s, L) 法 (=GIDR(s, L) 法) が IDR(s) 法, Bi-CGSTAB(s) 法と同じ特長「$N + N/s$ 回の行列ベクトル積演算で真の解を与える」を持つことを意味するものである．

GBi-CGSTAB(s, L) 法に必要な計算量とメモリの詳細な評価を行うと表 2 のようになる．表では，[7] に従って行列ベクトル積 (表では MVS) を 1 にスケーリングをした数値を載せている．AXPY は，ベクトルの単純な和とスケーリングを，それぞれ 0.5 と換算して算出された計算量である．DOT はベクトルの内積の回数を表わす．さらに表右におけるメモリとは，アルゴリズムを動かす上

[*2] 正確にいうと完全には一致しておらず，アルゴリズム 3 の 14〜20 行において [10] とは異なる．ただし，この差異によって，算出される補助ベクトルは定数倍されるだけであり，本質的な違いではない．

で保存する必要がある N 次元ベクトルの本数を示している (メモリの換算において, 行列 A や, preconditioner は除いて考えており, 連立一次方程式の右ベクトルや解ベクトルは含んでいる).

表 2　行列ベクトル積の回数でスケーリングされた計算量と必要なメモリ

METHOD	MVS	AXPY	DOT	MEMORY
Bi-CGSTAB(L)	1	$\frac{3}{4}(L+3)$	$\frac{1}{4}(L+7)$	$2L+5$
IDR(s)	1	$2s+\frac{3}{2}+\frac{1}{s+1}$	$s+\frac{2}{s+1}$	$3s+5$
GBi-CGSTAB(s,L)	1	$\frac{s(L+1)}{2}+2+\frac{L-1}{2s+2}$	$s+\frac{L+3}{2s+2}$	$sL+L+2s+3$
GBi-CGSTAB($1,L$)	1	$\frac{3}{4}(L+3)$	$\frac{1}{4}(L+7)$	$2L+5$
GBi-CGSTAB($s,1$)	1	$s+2$	$s+\frac{2}{s+1}$	$3s+4$

なお, 表において, IDR(s) 法と GBi-CGSTAB($s,1$) 法 (=GIDR($s,1$) 法) の AXPY の計算量が違っていることに注意されたい. 両者は, 丸め誤差がなければ同じ残差ベクトル列を生成するアルゴリズムであるが, アルゴリズムの作り方に差があり, そのため AXPY にその差が現れている. (この差により, GBi-CGSTAB($s,1$) 法の方が IDR(s) 法より計算時間が少なくて済むと期待される.)

5　数値実験

ここでは, GIDR(s,L) 法 (=GBi-CGSTAB(s,L) 法) に関する発表 [10] において, 口頭で簡単に紹介した数値実験結果の詳細を記す. 数値実験においては Matlab 7.5 を使用した. また, 数値実験で比較した算法は, Bi-CGSTAB(L), IDR(s) 法, GBi-CGSTAB(s,L) (=GIDR(s,L)) 法である. Bi-CGSTAB(L) 法, IDR(s) 法に関しては, それぞれ [4], [7] をもとに書いたプログラムを用いた. 初期 shadow residual の設定に関しては, $s=1$ の場合は $\widetilde{R}_0=r_0$ と設定し, $s>1$ の場合は $\widetilde{R}_0$ の第一列 $=r_0$, 他の列は乱数を用いて定めた後に列ベクトルに関して直交化を行ったものを使用した. また, s が共通の場合において, IDR(s) 法, GBi-CGSTAB(s,L) 法は共通の初期 shadow residual $\widetilde{R}_0$ を持つように設定した. 初期解ベクトルは, $x_0=\mathbf{0}$ と設定した. また, 収束判定条件は $\|r_n\|/\|b\|<10^{-8}$ とした (ただし, r_n は, $b-Ax_n$ ではなく, アルゴリズムにおいて計算される残差である).

■数値例 1 (3 次元対流問題)　1 つ目の例は, [4, 7] において使用された例で, 連立一次方程式の係数行列が歪対称行列に近いことから, 1 次の安定化多項式が収束性が悪いことが知られているものである.

そのテスト問題は, ディリクレ境界条件を持つ, 領域 $[0,1]\times[0,1]\times[0,1]$ 上の偏微分方程式

$$u_{xx}+u_{yy}+u_{zz}+1000u_x=F$$

を有限差分法により離散化した際に出てくる連立一次方程式である. 関数 F は, 解 u が $u(x,y,z)=\exp(x,y,z)\sin(\pi x)\sin(\pi y)\sin(\pi z)$ となるように設定される. また, 差分法は中心差分法を使い, グ

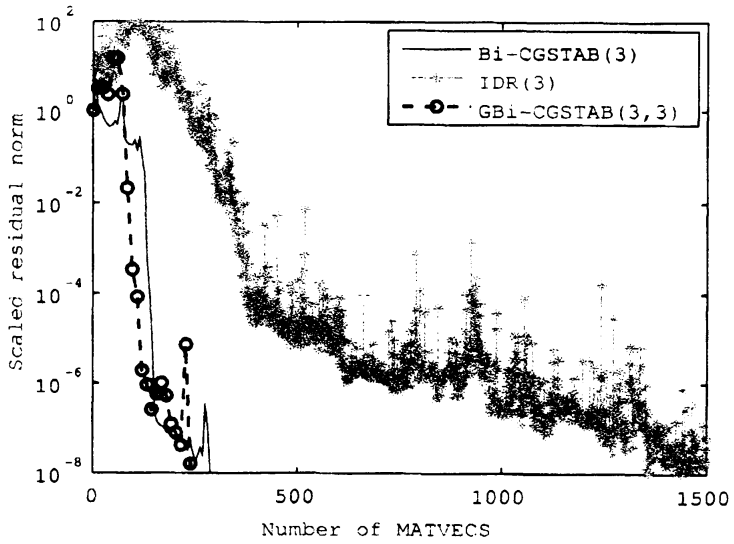

図 6　$s = L = 3$ と設定した 3 つの手法の比較 (数値例 1)

表 3　Bi-CGSTAB(L), IDR(s) における収束までの行列ベクトル積の演算回数 (数値例 1)

METHOD	MATVECS
Bi-CGSTAB(1)	2112
Bi-CGSTAB(2)	252
Bi-CGSTAB(3)	294
Bi-CGSTAB(4)	208
IDR(1)	2044
IDR(2)	1947
IDR(3)	1660
IDR(4)	1150

表 4　GBi-CGSTAB(s, L) における収束までの行列ベクトル積の演算回数 (数値例 1)

s \ L	1	2	3	4
1	2070	240	252	224
2	1983	234	270	252
3	1396	232	252	240
4	1155	240	255	240

リッド点として 3 方向とも 52 点を取る．係数行列のサイズは 125000×125000 である．

数値実験は，Bi-CGSTAB(L) 法, IDR(s) 法, GBi-CGSTAB(s, L) 法それぞれに対して，s, L を $s = 1, 2, 3, 4$, $L = 1, 2, 3, 4$ と変えて行った．なお，前処理は行わなかった．数値実験結果を，図 6 (残差ノルムの履歴), 表 3, 4 (行列ベクトル積の回数) に示す．

これらの結果から，GBi-CGSTAB(s, L) 法は，IDR(s) 法が苦手とする歪対称に近い係数行列を持つ連立一次方程式に対しても，Bi-CGSTAB(L) 法と同様のよい収束性をもつことが分かる．

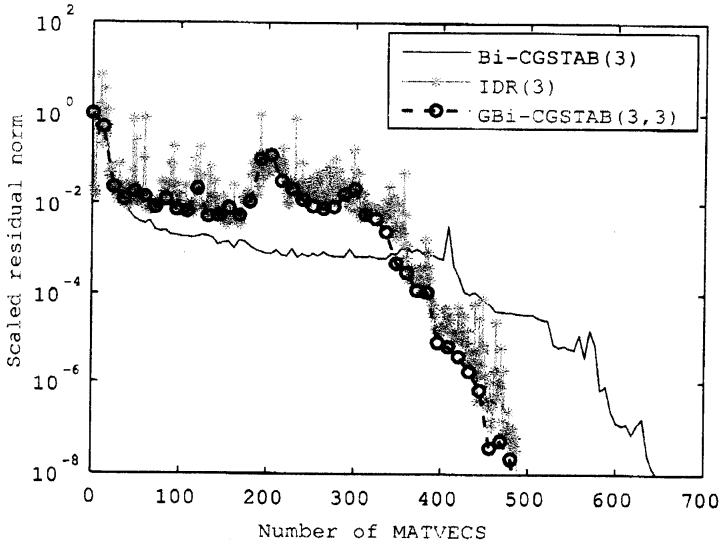

図 7　$s = L = 3$ と設定した 3 つの手法の比較 (数値例 2)

表 5　Bi-CGSTAB(L), IDR(s) における収束までの行列ベクトル積の演算回数 (数値例 2)

METHOD	MATVECS
Bi-CGSTAB(1)	748
Bi-CGSTAB(2)	652
Bi-CGSTAB(3)	648
Bi-CGSTAB(4)	640
IDR(1)	718
IDR(2)	528
IDR(3)	488
IDR(4)	450

表 6　GBi-CGSTAB(s, L) における収束までの行列ベクトル積の演算回数 (数値例 2)

s \ L	1	2	3	4
1	782	656	648	640
2	519	516	522	516
3	488	496	492	480
4	445	440	450	460

■数値例 2 (raefsky2[2])　2 つ目の数値例としては，[2] に与えられているテスト行列の一つ，raefsky2 を用いた．右ベクトル$\boldsymbol{b}$ は 解ベクトルの成分がすべて 1 になるように設定した．あとは数値例 1 と同様の条件で数値実験を行った．結果を図 7 (残差ノルムの履歴), 表 5, 6 (行列ベクトル積の回数) に示す．これらの結果から，数値例 2 においては，IDR(s) 法, GBi-CGSTAB(s, L) 法は同程度の収束性を持ち，Bi-CGSTAB(L) 法のみ収束性が少し劣るということが分かる．

上記数値例 1,2 から，GBi-CGSTAB(s, L) 法は，問題ごとに，IDR(s) 法，Bi-CGSTAB(L) 法のうち良い方と同じ収束性を持つということが分かる．上記以外にも多くの数値実験を行ったが，同様の結果が得られた．したがって，"GBi-CGSTAB(s, L) 法は IDR(s) 法と Bi-CGSTAB(L) 法の良いとこ取りの方法である" と言える．

6 結論と今後の課題

高次元 shadow residual を持つ Bi-CG 法である GBi-CG(s) 法を新たに提案して，GBi-CG(s) 法に L 次の安定化多項式を付加したアルゴリズム GBi-CGSTAB(s, L) 法を導出した．そして，GBi-CGSTAB(s, L) 法が GIDR(s, L) 法と一致することを確認した．また数値実験によって，GBi-CGSTAB(s, L) (=GIDR(s, L)) 法の有用性を示した．

本稿のまとめとして，Bi-CG 法をもとにした算法の関係を表 7 に示す．なお，表にある ML(s)BiCG 法に関しては本文では言及しなかったが，[9] で提案された高次元 shadow residual を持つ Bi-CG 法であり，1 次の安定化多項式を付加したアルゴリズム ML(s)BiCGSTAB 法（非常に複雑なアルゴリズムであるが丸め誤差に強い）も提案されている．この表から明らかなように，高次元 shadow residual を持つ Bi-CG 法のうち，ML(s)BiCG 法，Bi-CG(s) 法に関しては，高次の安定化多項式を付加したアルゴリズムがまだ提案されていない．これらのアルゴリズムを開発することは，今後の大きな課題といえる．

表 7　Bi-CG 法をもとにした手法の関係図．列のパラメータ L は安定化多項式の次数，行のパラメータ s は shadow residual の次元を表わす．太文字で表した算法が我々が開発したもの

s \ L	0	1	L
1	Bi-CG	Bi-CGSTAB	Bi-CGSTAB(L)
s		IDR(s)	**GIDR(s, L)**
	ML(s)BiCG	ML(s)BiCGSTAB	?
	Bi-CG(s)	Bi-CGSTAB(s)	?
	GBi-CG(s)	**GBi-CGSTAB(s, 1)**	**GBi-CGSTAB(s, L)**

参考文献

[1] J. I. Aliaga, D. L. Boley, R. W. Freund and V. Hernandez, A Lanczos-type method for multiple starting vectors, Math. Comput., 69 (2000), pp. 1577–1602.

[2] University of Florida sparse matrix collection, http://www.cise.ufl.edu/research/sparse/matrices/index.html.

[3] R. Fletcher, Conjugate gradient methods for indefinite systems, in Proceedings of the Dundee Biennal Conference on Numerical Analysis 1974, G. A. Watson, ed., Springer-Verlag, New York,

1975, pp. 73–89.

[4] G. L. G. Sleijpen and D. R. Fokkema, BICGSTAB(L) for equations involving unsymmetric matrices with complex spectrum, ETNA, 1 (1993), pp. 11–32.

[5] G. L. G. Sleijpen, P. Sonneveld and M. B. van Gijzen, Bi-CGSTAB as an Induced Dimension Reduction Method, Appl. Math. Anal., REPORT 08-07 (2008), Delft Univer. Tech.

[6] P. Sonneveld, CGS, A Fast Lanczos-Type Solver for Nonsymmetric Linear System, SIAM J. Sci. and Stat. Comput., 10 (1989), pp. 36–52.

[7] P. Sonneveld and M. B. van Gijzen, IDR(s): A Family of Simple and Fast Algorithms for Solving Large Nonsymmetric Systems of Linear Equations, SIAM J. Sci. Comput., 31 (2008), pp. 1035–1062.

[8] H. A. van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 631–644.

[9] M. Yeung and T. F. Chan, ML(k)BiCGSTAB: A BiCGSTAB Variant based on multiple Lanczos starting vectors, SIAM J. Sci. Comput., 29 (1999), pp. 1263–1290.

[10] 谷尾真明, 杉原正顯, GIDR(s, L) : 一般化 IDR(s), 日本応用数理学会 2008 年度年会 講演予稿集, pp. 411–412, 2008.

[11] 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, Vol. 5 (1995), pp. 343–360.