ハイパフォーマンス
コンピューティング 91－2
（２００２．８．２１）

# ある積型反復法の多項式列の錯綜と収束性に及ぼす影響

藤野清次†

一般積型反復法では，収束の加速を目的として，ランチョス原理に基づく元の多項式の他に，新たに２つの多項式が使われている．それらの多項式もランチョス原理と同等の原理から産まれたものであるが，未だその性質がわからない部分も多い．多項式の増加という操作は，積型多項式の選択の自由度を増した反面，その順序に一貫性を持たせる作業が非常に複雑になった．その結果，本稿で指摘するように，算法の一部で積型多項式の順序に錯綜が生じた．本稿では，多項式の順序の錯綜が GPBiCG 法系統の反復法の収束に及ぼす影響について言及する．

# Complexity on the order of polynomial sequence in product-type iterative methods and its effect to convergence

SEIJI FUJINO†

Two polynomilas are newly used for the purpose of the acceleration of the convergence in generalized product-type iterative methods. The operation of the increase in the polynomial incresed degree of freedom of the selection. The work which gave the consistency to the order became very complicated. As a result, certain complication appeared in the order of polynomila. In this article, the effect by the complication to convergence will be made clear.

## 1. はじめに

非対称で疎な係数行列 $A$ を持つ連立１次方程式

$$A\boldsymbol{x} = \boldsymbol{b} \qquad (1)$$

を一般積型 (GP)BiCG 法系統の反復法で解く．ただし，$A$ は大きさ $n \times n$ の正方行列，$\boldsymbol{x}, \boldsymbol{b}$ は大きさ $n$ の解および右辺ベクトルとする．ここでは，反復法として，BiCGStab2 法[1] と GPBiCG 法[2] そして各々の変形版を取り上げる．近年，工学の分野において大規模な問題を解く需要がが増すにつれて，効率がよくかつ収束性が安定な反復法が強く求められている．BiCGStab2 法と GPBiCG 法は，２つの多項式を掛け合わした積型多項式から構成され，さらにそれらは３項（２項）漸化式の形で表される．これらの解法は多くの問題でその有効性が示されてきたが，稀に収束の停滞や発散などが起こる場合があることも知られている．そこで，本稿では，それらの反復法を構成する積型多項式の定義式とその算法中に現れる順番に着目し，その収束不安定性を解き明かす新たな知見が得られたので，数値実験による検証結果と合わせて報告する．

† 九州大学 情報基盤センター
Computing and Communications Center, Kyushu University

## 2. GPBiCG 法と変形版 GPBiCG 法の算法

ランチョス原理に基づく反復法においては多項式 $R_k(\lambda)$, $P_k(\lambda)$ が使われそれらは次の交代漸化式を満たす．

$$\begin{aligned} &R_0(\lambda) = 1,\ P_0(\lambda) = 1, \\ &R_k(\lambda) = R_{k-1}(\lambda) - \alpha_{k-1}\lambda P_{k-1}(\lambda), \\ &P_k(\lambda) = R_k(\lambda) + \beta_{k-1}P_{k-1}(\lambda). \\ &\qquad k = 1, 2, \cdots \end{aligned}$$

さらに，GPBiCG 法では多項式列 $H_k(\lambda)$, $G_k(\lambda)$ が加わり，それらは以下の交代漸化式を構成する．

$$\begin{aligned} &H_0(\lambda) = 1,\ G_0(\lambda) = \zeta_0, \\ &H_k(\lambda) = H_{k-1}(\lambda) - \lambda G_{k-1}(\lambda), \\ &G_k(\lambda) = \zeta_k H_k(\lambda) + \eta_k G_{k-1}(\lambda). \\ &\qquad k = 1, 2, \cdots \end{aligned}$$

以下に GPBiCG 法とその変形版の算法を記す．２つの算法はパラメータ $\zeta_k, \eta_k$ の値の定め方だけが異なる．$\zeta_k$, $\eta_k$ は残差ノルムの局所的最小化から求める．その求め方には，(i) 正規方程式を経由して求めるガウスの消去法，行列 $A$ を QR 分解しその後 (ii) 修正 Gram-Schmidt 法で求める方法や (iii)Householder 法で求める方法等いろいろな方法があるが，本稿ではそのうちで最も簡単な方法 (i) を採用した．また，以下では変形版 GPBiCG 法を**アルゴリズム 1** と呼ぶ．

**[Algorithm 1 と GPBiCG 法]**

Let $\boldsymbol{x}_0$ be an initial guess, and
put $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$. $\boldsymbol{r}_0^*$ is an arbitrary
vector such that $(\boldsymbol{r}_0^*, \boldsymbol{r}_0) \neq 0$,
Set $\boldsymbol{t}_{-1} = \boldsymbol{w}_{-1} = \boldsymbol{0}$, $\beta_{-1} = 0$.
For $k = 0, 1, \ldots,$ do :

begin

$$\boldsymbol{p}_k = \boldsymbol{r}_k + \beta_{k-1}(\boldsymbol{p}_{k-1} - \boldsymbol{u}_{k-1}),$$
$$\alpha_k = \frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_k)}{(\boldsymbol{r}_0^*, A\boldsymbol{p}_k)},$$
$$\boldsymbol{y}_k = \boldsymbol{t}_{k-1} - \boldsymbol{r}_k - \alpha_k \boldsymbol{w}_{k-1} + \alpha_k A\boldsymbol{p}_k,$$
$$\boldsymbol{t}_k = \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k,$$

**[Algorithm 1]** :

if $k$ is even, $\eta_k = 0$, $\zeta_k = \dfrac{(A\boldsymbol{t}_k, \boldsymbol{t}_k)}{(A\boldsymbol{t}_k, A\boldsymbol{t}_k)}$,
if $k$ is odd, compute $\zeta_k$ and $\eta_k$,

**[GPBiCG]** :

compute $\zeta_k$ and $\eta_k$,

$$\boldsymbol{u}_k = \zeta_k A\boldsymbol{p}_k + \eta_k(\boldsymbol{t}_{k-1} - \boldsymbol{r}_k + \beta_{k-1}\boldsymbol{u}_{k-1}),$$
$$\boldsymbol{z}_k = \zeta_k \boldsymbol{r}_k + \eta_k \boldsymbol{z}_{k-1} - \alpha_k \boldsymbol{u}_k,$$
$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k + \boldsymbol{z}_k,$$
$$\boldsymbol{r}_{k+1} = \boldsymbol{t}_k - \eta_k \boldsymbol{y}_k - \zeta_k A\boldsymbol{t}_k,$$
$$\beta_k = \frac{\alpha_k}{\zeta_k} \cdot \frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_{k+1})}{(\boldsymbol{r}_0^*, \boldsymbol{r}_k)}$$
$$\boldsymbol{w}_k = A\boldsymbol{t}_k + \beta_k A\boldsymbol{p}_k,$$

end.

表 1 に GPBiCG 法の算法において現れるベクトルの定義式および反復の $k$ ステップにおける 4 つの多項式列の各ステップ回数を示す.

表 1　GPBiCG 法とアルゴリズム 1 において現れるベクトルの定義式と 4 つの多項式列のステップ回数

| | 定義式 | $H(\lambda)$ | $G(\lambda)$ | $R(\lambda)$ | $P(\lambda)$ |
|---|---|---|---|---|---|
| $\boldsymbol{p}_k$ | $H_k(\lambda)P_k(\lambda)\boldsymbol{r}_0$ | $k$ | | | $k$ |
| $\boldsymbol{y}_k$ | $\lambda G_{k-1}(\lambda)R_{k+1}(\lambda)\boldsymbol{r}_0$ | | $k-1$ | $k+1$ | |
| $\boldsymbol{t}_k$ | $H_k(\lambda)R_{k+1}(\lambda)\boldsymbol{r}_0$ | $k$ | | $k+1$ | |
| $\boldsymbol{u}_k$ | $\lambda G_k(\lambda)P_k(\lambda)\boldsymbol{r}_0$ | | $k$ | | $k$ |
| $\boldsymbol{z}_k$ | $G_k(\lambda)R_{k+1}(\lambda)\boldsymbol{r}_0$ | | $k$ | $k+1$ | |
| $\boldsymbol{r}_{k+1}$ | $H_{k+1}(\lambda)R_{k+1}(\lambda)\boldsymbol{r}_0$ | $k+1$ | | $k+1$ | |
| $\boldsymbol{w}_k$ | $\lambda \underline{H_k(\lambda)} P_{k+1}(\lambda)\boldsymbol{r}_0$ | $\underline{k}$ | | | $k+1$ |

GPBiCG 法と変形版のアルゴリズム 1 および表 1 に記載した多項式のステップ回数から以下のことがわかる.

多項式 $H_{k+1}(\lambda)$ から成る残差ベクトル $\boldsymbol{r}_{k+1}(:= H_{k+1}(\lambda)R_{k+1}(\lambda)\boldsymbol{r}_0)$ を元にパラメータ

$$\beta_k = \frac{\alpha_k}{\zeta_k} \frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_{k+1})}{(\boldsymbol{r}_0^*, \boldsymbol{r}_k)} \tag{2}$$

を求め, それを使って $H_k(\lambda)$ から成る中間ベクトル $\boldsymbol{w}_k(:= \lambda H_k(\lambda)P_k(\lambda)\boldsymbol{r}_0)$ を

$$\boldsymbol{w}_k = A\boldsymbol{t}_k + \beta_k A\boldsymbol{p}_k \tag{3}$$

の式で求めている. ここで, 表に示す多項式 $H(\lambda)$ のステップ回数が $k$ であることに気がつく (表中で下線をつけたもの). 多項式 $H(\lambda)$ の出現の順番からすれば, ここでは $H_k(\lambda)$ ではなく, $H_{k+1}(\lambda)$ からなる中間ベクトルを求める方が, 新しく導入した多項式の役割 (=収束の加速) を考えたときより自然であろう. 表 1 からわかるように, 他の多項式ではステップ回数は単調に増加している. この多項式 $H(\lambda)$ だけが, ベクトル $\boldsymbol{w}_k$ のところでステップ回数が $k+1$ から $k$ に小さくなっている. つまり, 反復過程の 1 つ古いものが入ったベクトルを求めている. この事実は, この錯綜現象が GPBiCG 法の収束に悪い影響を与える可能性があること, を示唆している.

具体的には, 次の反復過程のステップにおいて, $\boldsymbol{w}_{k-1}$ を使って更新される中間ベクトル $\boldsymbol{y}_k$ の値に少し" ズレ" が生じ, これがパラメータ $\zeta_k, \eta_k$ の値に影響を与える. GPBiCG 法では, この 2 つのパラメータを使うため, その影響をまともに受けることになる.

一方, アルゴリズム 1 では, 収束過程の偶数ステップではパラメータの 1 つ $\eta_k = 0$ とする. そのため, 変形版 BiCGSTAB2 法の偶数ステップでは,

$$\boldsymbol{r}_{k+1} = \boldsymbol{t}_k - \eta_k \boldsymbol{y}_k - \zeta_k A\boldsymbol{t}_k = \boldsymbol{t}_k - \zeta_k A\boldsymbol{t}_k \tag{4}$$

となる. また, パラメータの $\zeta_k$ の計算でも $\boldsymbol{y}_k$ が使われていないので, $\boldsymbol{y}_k$ の" ズレ" は偶数ステップにおいては影響しない. 奇数ステップのみ GPBiCG 法と同様に影響を受けるが, 全体の影響の度合は半減する.

また, 漸化式の並べ替えを行なおうとしても, パラメータ $\beta_k$ の計算がベクトル $\boldsymbol{w}_k$ の式に含まれるので, 残差ベクトル $\boldsymbol{r}_{k+1}$ の前に移動できない. さらに, $\boldsymbol{w}_{k-1} = A\boldsymbol{t}_{k-1} + \beta_{k-1}A\boldsymbol{p}_{k-1}$ として, 算法のループの先頭に移動しても, 次の $(k+1)$ ステップ目の多項式 $H(\lambda)$ において順序の錯綜が同様に起こり状況は同じである.

## 3. Gutknecht による BiCGSTAB2 法

ここでは, 以下のような多項式列 $Q_k(\lambda)$ を使った漸化式を導入する. 反復過程のステップが偶数か奇数かによって漸化式は異なる.

$$Q_0 := 1,$$
$$Q_{2k+1} := (1 - \eta_{2k}\lambda)Q_{2k},$$
$$Q_{2k+2} := (1 + \zeta_{2k+1} - \eta_{2k+1}\lambda)Q_{2k+1} - \zeta_{2k+1}Q_{2k}$$
$$k = 0, 1, 2, \cdots$$

Gutknecht による BiCGSTAB2 法および BiCGMin 法の算法を記す. また, 以下では前者の算法を**アルゴリズム 2** と呼ぶ.

**[Algorithm 2 と BiCGMin 法]**

Let $\boldsymbol{x}_0$ be an initial guess, and
put $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$. $\boldsymbol{r}_0^*$ is an arbitrary
vector such that $(\boldsymbol{r}_0^*, \boldsymbol{r}_0) \neq 0$,

$$\alpha_0 = (\boldsymbol{r}_0^*, \boldsymbol{r}_0)/(\boldsymbol{r}_0^*, A\boldsymbol{p}_0), \boldsymbol{t}_1 = \boldsymbol{r}_0 - \alpha_0 A\boldsymbol{p}_0,$$
$$\eta_0 = (\boldsymbol{t}_1, A\boldsymbol{t}_1)/(A\boldsymbol{t}_1, A\boldsymbol{t}_1), \quad \boldsymbol{r}_1 = \boldsymbol{t}_1 - \eta_0 A\boldsymbol{t}_1,$$
$$\boldsymbol{x}_1 = \boldsymbol{x}_0 + \alpha_0 \boldsymbol{p}_0 + \eta_0 \boldsymbol{t}_1,$$
$$\beta_0 = \frac{\alpha_0(\boldsymbol{r}_0^*, \boldsymbol{r}_1)}{\eta_0(\boldsymbol{r}_0^*, \boldsymbol{r}_0)}, \quad \boldsymbol{w}_1 = \boldsymbol{t}_1 + \beta_0 \boldsymbol{p}_0,$$
$$\boldsymbol{p}_1 = \boldsymbol{r}_1 + \beta_0(\boldsymbol{p}_0 - \eta_0 A\boldsymbol{p}_0),$$

begin $k = 1, 2, \ldots,$ do :

$$\alpha_k = \frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_k)}{(\boldsymbol{r}_0^*, A\boldsymbol{p}_k)},$$
$$\boldsymbol{s}_{k+1} = \boldsymbol{t}_k - \alpha_k A\boldsymbol{w}_k,$$
$$\boldsymbol{t}_{k+1} = \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k,$$

**[Algorithm 2]** :

if $k$ is even, $\zeta_k = 0, \eta_k = \dfrac{(\boldsymbol{t}_{k+1}, A\boldsymbol{t}_{k+1})}{(A\boldsymbol{t}_{k+1}, A\boldsymbol{t}_{k+1})}$,

if $k$ is odd, compute $\zeta_k$ and $\eta_k$,

**[BiCGMin]** :

compute $\zeta_k$ and $\eta_k$,

$$\boldsymbol{r}_{k+1} = -\zeta_k \boldsymbol{s}_{k+1} + (1+\zeta_k)\boldsymbol{t}_{k+1} - \eta_k A\boldsymbol{t}_{k+1},$$
$$\boldsymbol{x}_{k+1} = -\zeta_k(\boldsymbol{x}_{k-1} + \alpha_{k-1}\boldsymbol{p}_{k-1} + \alpha_k \boldsymbol{w}_k) + (1+\zeta_k)(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) + \eta_k \boldsymbol{t}_{k+1},$$
$$\beta_k = \frac{\alpha_k(\boldsymbol{r}_0^*, \boldsymbol{r}_{k+1})}{\eta_k(\boldsymbol{r}_0^*, \boldsymbol{r}_k)},$$
$$\boldsymbol{w}_{k+1} = \boldsymbol{t}_{k+1} + \beta_k \boldsymbol{p}_k,$$
$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} - \beta_k(\zeta_k \boldsymbol{w}_k + (1+\zeta_k)\boldsymbol{p}_k - \eta_k A\boldsymbol{p}_k),$$

end.

表 2 に**アルゴリズム 2** と BiCGMin 法において現れるベクトルの定義式と反復の $k$ ステップにおける多項式列 $Q_k(\lambda)$ のステップ回数を表す．表 1 と同様に，ここでも，ベクトル $\boldsymbol{w}_{k+1}$ の部分 (表中で下線をつけたところ) でステップ回数の逆転が起こっている．直前に更新されたベクトル $\boldsymbol{r}_{k+1}$ ではすでに $Q_{k+1}(\lambda)$ が使用されているにも拘らず，多項式 $Q(\lambda)$ のステップ回数が $k+1$ ではなく $k$ に減っている．したがって，**アルゴリズム 2** においても，多項式 $Q(\lambda)$ のステップ回数の逆転現象が収束に影響を及ぼす可能性がある．しかしながら，**アルゴリズム 1** のときと同様に，偶数ステップではベクトル $\boldsymbol{w}_{k+1}$ と $\boldsymbol{s}_{k+1}$ に影響があるが，いずれもパラメータの値が $\zeta_k = 0$ であるので結果的に影響を免れることができる．全体として，奇数ステップのとき**アルゴリズム 2** の収束性全体が影響を受けると言える．

**表 2** アルゴリズム 2 と BiCGMin 法の算法において現われるベクトルの定義式と多項式列 $Q_k(\lambda)$ のステップ数

| ベクトル | 定 義 式 | $Q(\lambda)$ |
|---|---|---|
| $\boldsymbol{s}_{k+1}$ | $Q_{k-1}(\lambda)R_{k+1}(\lambda)\boldsymbol{r}_0$ | $k-1$ |
| $\boldsymbol{t}_{k+1}$ | $Q_k(\lambda)R_{k+1}(\lambda)\boldsymbol{r}_0$ | $k$ |
| $\boldsymbol{r}_{k+1}$ | $Q_{k+1}(\lambda)R_{k+1}(\lambda)\boldsymbol{r}_0$ | $k+1$ |
| $\boldsymbol{w}_{k+1}$ | $\underline{Q_k(\lambda)P_{k+1}(\lambda)\boldsymbol{r}_0}$ | $\underline{k}$ |
| $\boldsymbol{p}_{k+1}$ | $\underline{Q_{k+1}(\lambda)P_{k+1}(\lambda)\boldsymbol{r}_0}$ | $k+1$ |

## 4. 数 値 実 験

数値実験は，九州大学情報基盤センターの富士通スーパコンピュータ VPP5000 上で行った．演算はすべて倍精度で実行した．プログラミング言語は Fortran90 を使用した．コンパイルは frt コマンドのみとし，特別の最適化オプションは使わなかった．各反復法の収束判定条件は各反復での相対残差ノルムが $||\boldsymbol{r}_k||/||\boldsymbol{r}_0|| \leq 10^{-12}$ を満たすときとし，最大反復回数は，問題の難易度に応じて問題 1 と 4 では 500 回，問題 2 では 3000 回，問題 3 では 5000 回とした．右辺項は厳密解がすべて 1 になるように定めた．係数行列の次元数はすべての問題で $n$=16384 とした．反復の初期値 $\boldsymbol{x}_0$ はすべて 0 とした．

### 4.1 テスト問題

以下に示す 4 種類のテスト問題 (すべて実行列) を使って，積型多項式の順序の錯綜が反復法の収束性や安定性に及ぼす影響を調べた．

- **[問題 1]** 以下に示す Toeplitz 行列においてパラメータ $\gamma$ を 1.5 から 0.001 ずつ変化させた．

$$A := \begin{bmatrix} 2 & 1 & & & & \\ 0 & 2 & 1 & & & \\ \gamma & 0 & 2 & 1 & & \\ & \gamma & 0 & 2 & 1 & \\ & & \gamma & 0 & 2 & \ddots \\ & & & \ddots & \ddots & \ddots \end{bmatrix}$$

- **[問題 2 と問題 3]** 正方形領域 $[0, 1]^2$ で定義され全周 Dirichlet 境界条件を課した以下の偏微分方程式を 2 次精度 5 点差分で離散化した境界値問題を扱った．(i) 前処理がない場合と (ii)ILU(0) 分解による前処理がある場合の 2 ケースを調べた．$D$ は実数パラメータを表し，右辺項 $G(x, y)$ は厳密解が $u(x, y)$=1 + $xy$ となるように定めた．

**[問題 2]**

$$-u_{xx} - u_{yy} + Du_x = G(x, y) \qquad (5)$$

**[問題 3]**

$$-u_{xx} - u_{yy} - 43\pi^2 u + D\{(y - 1/2)u_x + (x - 1/3)(x - 2/3)u_y\} = G(x, y)$$

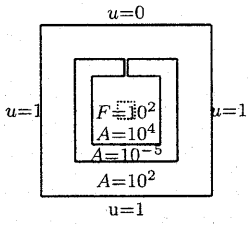
- **[問題 4]** 正方形領域 $[0, 1]^2$ で定義された以下の偏微分方程式

$$-(A(x, y)u_x)_x - (A(x, y)u_y)_y + \beta e^{2(x^2+y^2)}u_x = F(x, y)$$
$$u|_{x=0} = 1, u|_{x=1} = 1, u|_{y=0} = 1, u|_{y=1} = 0,$$

の Dirichlet 型境界条件つき境界値問題を考える．$A(x, y)$ と $F(x, y)$ の値を図に示す．

### 4.2 数値実験結果

問題 1 において，初期残差 $\boldsymbol{r}_0^* = \boldsymbol{r}_0$ のときのアルゴ




リズム 1 と GPBiCG 法の反復回数の変化を図 1 に, 同じくアルゴリズム 2 と BiCGMin 法の反復回数の変化を図 2 に示す. さらに, 初期シャドウ残差 $\boldsymbol{r}_0^*$=乱数のときの 4 つの反復法の反復回数の変化を図 3 に示す. 初期値に用いた乱数は, 区間 [0,1] の乗算合同法による一様乱数を使用した. ただし, 最大反復回数まで反復を繰り返しても収束しなかったものは, 見やすくするために, 図 1 と図 2 において, 最大反復回数 500 回のところに各々縦座標を少しずらして表示した.

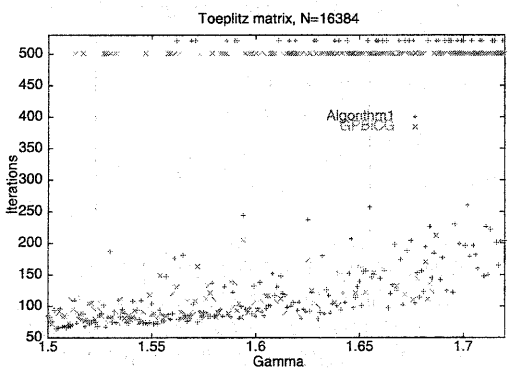



**図 1** 問題 1 において, 初期値 $\boldsymbol{r}_0^* = \boldsymbol{r}_0$ とおき, パラメータ $\gamma$ を変化させたときのアルゴリズム 1, GPBiCG 法の反復回数の変化

表 3 に, 問題 1 において反復 1 回当たりの計算時間 (単位 : ミリ秒) とアルゴリズム 1 の計算時間を 1 にしたときの各反復法の比率を示す. 表 4 に, 最大反復回数までの反復で収束しないときが初めて現れたときのパラメータ $\gamma$ の値を示す. 初期シャドウ残差 $\boldsymbol{r}_0^*$ に乱数を代入すると, 図 3 に示すように, いずれの反復解法も収束が安定し, パラメータ $\gamma$ の値を大きくしても収束することがわかる. また, 問題 1 の $\gamma$ の値が, アルゴリズム 1 や同 2 に比べて, GPBiCG 法のときの 1.967, 同じく BiCGMin 法のときの 1.965 と小さな値までの範囲しか収束せず, $\gamma$ の値の変動に対して収束の頑強性が弱いことがわかる.

表 5 に, 問題 1 において各反復法の収束までの平均反復回数および未収束の場合の数を示す. ただし, 反復回数の平均値は未収束の場合を除いて平均をとったものである.

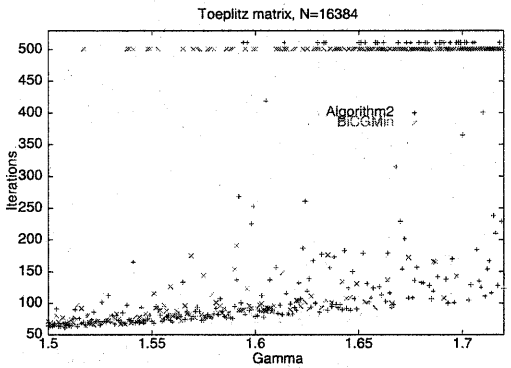



**図 2** 問題 1 において, 初期値 $\boldsymbol{r}_0^* = \boldsymbol{r}_0$ とおき, パラメータ $\gamma$ を変化させたときのアルゴリズム 2, BiCGMin 法の反復回数の変化

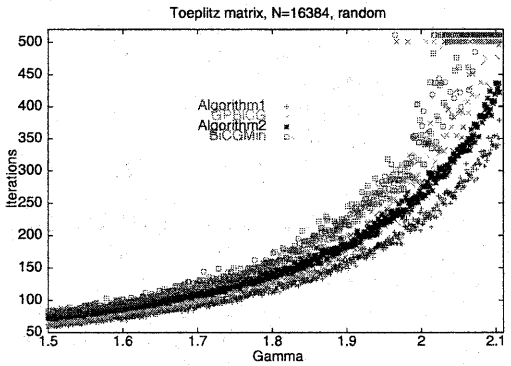



**図 3** 問題 1 において, 初期値 $\boldsymbol{r}_0^*$=乱数とおき, パラメータ $\gamma$ を変化させたときのアルゴリズム 1, GPBiCG 法, アルゴリズム 2, BiCGMin 法の反復回数の変化

**表 3** 問題 1 において反復 1 回当たりの計算時間とアルゴリズム 1 の計算時間を 1 としたときの比率

| | 1 反復時間と比 |
|---|---|
| アルゴリズム 1 | 0.245 (1.00) |
| GPBiCG 法 | 0.287 (1.17) |
| アルゴリズム 2 | 0.255 (1.04) |
| BiCGMin 法 | 0.274 (1.12) |

表 5 に示すように, 問題 1 において, $\boldsymbol{r}_0^* = \boldsymbol{r}_0$ の場合, GPBiCG 法では $\gamma$ の値を 221 回変えた中でおよそ半数の 112 回 (50.7%), 同じく BiCGMin 法でも 105 回 (47.5%) も収束に失敗した. 一方, アルゴリズム 1 や同 2 では, その半分以下の 38 回 (17.2%) および 31 回 (14.0%) しか未収束の場合はなかった. さらに, $\boldsymbol{r}_0^*$=乱数の場合には, 4 つの反復法とも区間 [1.5, 1.72] の間のすべての $\gamma$ の値のとき収束した. さらに,

$\gamma$ の値を 2.105 まで大きくすると, GPBiCG 法で 47 回, BiCGMin 法で 66 回の場合が未収束であった. したがって, 初期シャドウ残差を $\boldsymbol{r}_0^*$=乱数にすると, 収束性が向上し, 多項式の錯綜の影響は一見なくなったかのように見えるが, その影響は潜伏していることがわかる. すなわち, $\gamma$ の値が小さいと行列の性質がよく収束性に影響は出ないが, $\gamma$ の値を大きくすると行列の性質が悪くなり, 潜伏していた錯綜の影響が再び現れているものと思われる.

**表 4** 最大反復回数までの反復で収束しないときが初めて現れたときのパラメータ $\gamma$ の値

| | $\boldsymbol{r}_0^*=\boldsymbol{r}_0$ | $\boldsymbol{r}_0^*$=乱数 |
|---|---|---|
| アルゴリズム 1 | 1.562 | 2.106 |
| GPBiCG 法 | 1.513 | 1.967 |
| アルゴリズム 2 | 1.594 | 2.106 |
| BiCGMin 法 | 1.517 | 1.965 |

**表 5** 問題 1 において各反復法 (前処理なし) の収束までの平均反復回数 (未収束の場合は除く) および未収束の場合の数

| | $\boldsymbol{r}_0^*=\boldsymbol{r}_0$ | | $\boldsymbol{r}_0^*$=乱数 | |
|---|---|---|---|---|
| | $\gamma$: 1.5~1.72 | | $\gamma$: 1.5~1.72 (上段)<br>$\gamma$: 1.5~2.105(下段) | |
| | 平均回数 | 未収束/全試度 | 平均回数 | 未収束/全試度 |
| アルゴリズム 1 | 115.4 | 38/221 | 76.4 | 0/221 |
| | | | 144.3 | 0/606 |
| GPBiCG 法 | 103.9 | 112/221 | 83.2 | 0/221 |
| | | | 158.0 | 47/606 |
| アルゴリズム 2 | 110.1 | 31/221 | 91.1 | 0/221 |
| | | | 171.6 | 0/606 |
| BiCGMin | 92.4 | 105/221 | 100.2 | 0/221 |
| | | | 177.6 | 66/606 |

**表 6** 問題 2 と問題 3 において各反復法の収束までの平均反復回数 (未収束の場合は除く) および未収束の場合の数 ($Dh$: 0.5~64.0) (上段: $\boldsymbol{r}_0^*=\boldsymbol{r}_0$ のとき, 下段: $\boldsymbol{r}_0^*$=乱数のとき)

| 問題 2 | 前処理なし | | ILU(0) 分解つき | |
|---|---|---|---|---|
| | 平均回数 | 未収束/全試度 | 平均回数 | 未収束/全試度 |
| アルゴリズム 1 | 339.8 | 0/640 | 18.08 | 0/640 |
| | 325.1 | 0/640 | 18.63 | 0/640 |
| GPBiCG 法 | 798.0 | 149/640 | 18.43 | 0/640 |
| | 757.5 | 163/640 | 18.68 | 0/640 |
| 問題 3 | 前処理なし | | ILU(0) 分解つき | |
| | 平均回数 | 未収束/全試度 | 平均回数 | 未収束/全試度 |
| アルゴリズム 1 | 1960.3 | 0/640 | 105.5 | 0/640 |
| | 1922.1 | 0/640 | 103.0 | 0/640 |
| GPBiCG 法 | 2275.6 | 7/640 | 114.0 | 0/640 |
| | 2162.7 | 12/640 | 111.5 | 0/640 |

問題 2 において初期値 $\boldsymbol{r}_0^* = \boldsymbol{r}_0$ のとき, アルゴリズム 1 と GPBiCG 法において前処理がない場合を図 4 に, 前処理としてフィルインを考慮しない ILU(0) 分解を施したときの結果を図 5 に示す. いずれもパラメータ $\gamma$ を変化させたときの反復回数をプロットしたものである. 表 6 に, 問題 2 と問題 3 において各反復法の収束までの平均反復回数 (未収束の場合は除く) および未収束の場合の数 ($Dh$:0.5 から 64.0 までの範囲で 0.5 刻み 640 通り) を示す. さらに, 各行の上段に $\boldsymbol{r}_0^*=\boldsymbol{r}_0$ のとき, 下段に $\boldsymbol{r}_0^*$=乱数のときの結果を示す. 前処理がないとき, 未収束の場合が GPBiCG 法で 149 回, BiCGMin 法で 163 回も現れたが, 前処理をすると全ての場合が収束した.

**表 7** 問題 4 において ILU(0) 分解つきの反復法の収束までの平均反復回数および未収束の場合の数

| | $\boldsymbol{r}_0^*=\boldsymbol{r}_0$ | | $\boldsymbol{r}_0^*$=乱数 | |
|---|---|---|---|---|
| | $Dh$: 0.5~200.0 | | $Dh$: 0.5~200.0 | |
| | 平均回数 | 未収束/全試度 | 平均回数 | 未収束/全試度 |
| アルゴリズム 1 | 148.7 | 0/400 | 143.0 | 0/400 |
| GPBiCG 法 | 153.3 | 0/400 | 138.7 | 0/400 |

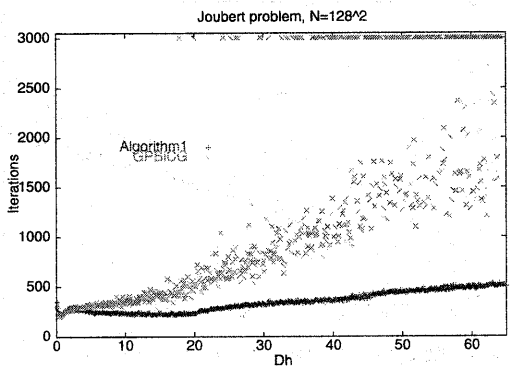



**図 4** 問題 2 において, 初期値 $\boldsymbol{r}_0^* = \boldsymbol{r}_0$ とおきパラメータ $\gamma$ を変化させたときのアルゴリズム 1, GPBiCG 法の反復回数の変化. 前処理なし

問題 3 において初期値 $\boldsymbol{r}_0^* = \boldsymbol{r}_0$ のとき, アルゴリズム 1 と GPBiCG 法において前処理がない場合を図 6 に, 前処理としてフィルインを考慮しない ILU(0) 分解を施したときの結果を図 7 に示す. パラメータ $\gamma$ がおよそ 30 以下の場合, 2 つの解法の収束までの回数はほぼ同じであるが, $\gamma$ が 30 よりも大きくなると, GPBiCG 法の反復回数の方がバラつく.

表 7 に, 問題 4 において各反復法の収束までの平均反復回数および未収束の場合の数を表す. また, 図 8 に, 問題 4 において初期値 $\boldsymbol{r}_0^* = \boldsymbol{r}_0$ と $\boldsymbol{r}_0^*$=乱数とおき, パラメータ $\gamma$ を変化させたときの ILU(0) 分解つきのアルゴリズム 1 の反復回数の変化を, 図 9 に, 同じく ILU(0) 分解つきの GPBiCG 法の反復回数の変化を示す. GPBiCG 法の場合の方が, $\boldsymbol{r}_0^*$=乱数の効果がより鮮明に出ていることがわかる.

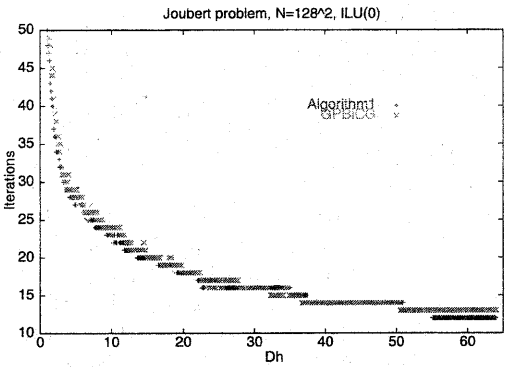



図 5　問題 2 において, 初期値 $\boldsymbol{r}_0^*$=$\boldsymbol{r}_0$ とおきパラメータ $\gamma$ を変化させたときの ILU(0) 分解つきアルゴリズム 1, GPBiCG 法の反復回数の変化

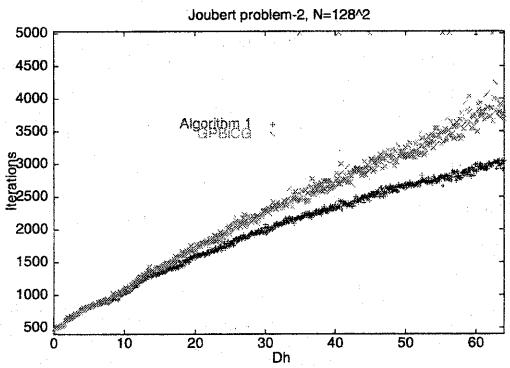



図 6　問題 3 において, 初期値 $\boldsymbol{r}_0^* = \boldsymbol{r}_0$ とおいたときのアルゴリズム 1, GPBiCG 法の反復回数の変化. 前処理なし

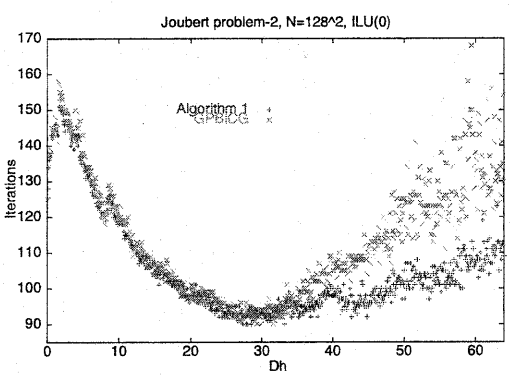



図 7　問題 3 において, 初期値 $\boldsymbol{r}_0^*$=と $\boldsymbol{r}_0$ とおいたときの ILU(0) 分解つきアルゴリズム 1, GPBiCG 法の反復回数の変化

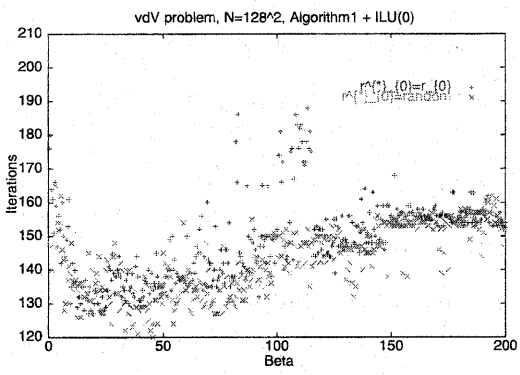



図 8　問題 4 において, 初期値 $\boldsymbol{r}_0^* = \boldsymbol{r}_0$ と $\boldsymbol{r}_0^*$=乱数とおいたときの ILU(0) 分解つきのアルゴリズム 1 の反復回数の変化

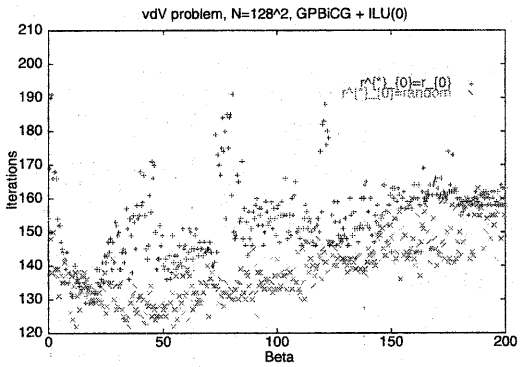



図 9　問題 4 において, 初期値 $\boldsymbol{r}_0^* = \boldsymbol{r}_0$ と $\boldsymbol{r}_0^*$=乱数とおいたときの ILU(0) 分解つきの GPBiCG 法の反復回数の変化

## 5. ま と め

GPBiCG 法には積型多項式列の順序の錯綜という現象が発生していることを指摘し, それが収束性に及ぼす影響の度合を調べた. 行列の前処理や $\boldsymbol{r}_0^*$ に一様乱数の代入により, その影響はかなりの程度低減した. しかし, これは問題点が根本的に解決したことを意味するのではなく, 方程式中のパラメータの値を変化させ行列の性質が悪くなった場合には, 収束の不安定現象が再び現れる可能性があることを示唆している.

## 参 考 文 献


1) Gutknecht, M.H.: Variants of BiCGSTAB for Matrix with Complex Spectrum, *SIAM J. Sci. Comput.*, Vol.14, pp. 1020-1033(1993).

2) Zhang, S.-L.: GPBi-CG: Generalized Product-type Methods Based on Bi-CG for Solving Nonsymmetric Linear Systems, *SIAM J. Sci. Comput.*, Vol.18, pp. 537-551(1997).