九州大学学術情報リポジトリ
Kyushu University Institutional Repository

基本数値演算 ： 内積の評価とその応用

藤野, 清次
九州大学情報基盤センター

渡部, 善隆
九州大学情報基盤センター

南里, 豪志
九州大学情報基盤センター

https://doi.org/10.15017/4782110

出版情報：九州大学情報基盤センター年報. 2, pp.45-53, 2002-03. 九州大学情報基盤センター
バージョン：
権利関係：

KYUSHU UNIVERSITY

情報基盤センター年報第2号（2002年）
九州大学情報基盤センター

基本数値演算：内積の評価とその応用

Performance Estimation of Inner-Product and its Application

藤野清次　渡部善隆　南里豪志
Seiji Fujino　Yoshitaka Watanabe　Takeshi Nanri

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

要旨　内積計算は，科学技術計算，特に大規模連立1次方程式の解を求める上で重要な演算の一つである．しかしながら，内積計算をうまくやらないと，例えば共役勾配法のような反復法の収束過程において収束の停滞や停止に遭遇することがある．これらは基本演算の丸め誤差の蓄積に起因するところが大きい．そこで，本研究では，補償付きアルゴリズムを使って内積計算に伴う数値誤差を減らしたGP(一般積型) BiCG法の収束性を評価し，これらの問題点の解決や収束性の改善を図る．いくつかの数値実験を通して本方法の改善効果を検証する．

Abstract　Inner product evaluation is one of an underlying factor for solving large scaled non-symmetric system of linear equations. The bad remedy for inner product, however, sometimes experiences stagnation and breakdown in the iterative process such as Conjugate Gradient like method. An accumulation of round-off errors may hinder or preclude convergence. In this paper, we present the compensated GPBiCG method by means of compensated algorithm which substantially mitigates the problem of stagnation and breakdown. Numerical comparisons of the compensated GPBiCG method and the original one are given.

1　はじめに

1.1　積和 (内積) 計算

科学技術計算の分野では次のような計算がよく出てくる。ここで、a_i, b_i はデータで N はその個数とする。

$$S = \sum_{i=1}^{N} a_i b_i \tag{1.1}$$

一般に、上のような計算は**積和計算**と呼ばれる。すなわち、$S = a_1b_1 + a_2b_2 + \cdots + a_Nb_N$ という二つのデータの積の和を求めることを意味する。また、ベクトル a, b に対する内積 (a, b) の定義と同じであるため**内積計算**とも呼ばれる。積和を求めるプログラム (左はFortran90形式、右はC言語形式) を示す。

```
S=0.0                s=0.0;
DO K=1,N             for (i=0; i<N; i++){
S=S+A(I)*B(I)        s+=a[i]*b[i];
END DO                       }
```

2　内積計算の使用事例

次の連立1次方程式を反復解法で解くことを考える。

$$Ax = b. \tag{2.1}$$

ここで、係数行列 A は非対称疎行列で大きさは $n \times n$ とする。b と x は n 次元の右辺ベクトルと解ベクトルする。一般に、このような連立1次方程式は大規模になればなるほど計算時間あるいは必要なメモリー容量の観点から反復解法で解かれることが多い。行列 A が対称正定値行列の場合は、適当な前処理を係数行列に施した後、共役勾配 (Conjugate Gradient) 法を適用するのが最も効率的である。しかしながら、行列が非対称の場合には、反復解法の選択により効率も大きく変わりその選択は重要な鍵である。クリロフ (Krylov) 部分空間法はその中でも有力な反復解法の一つとされる。この部分空間法に属する反復解法には、BiCG法 [1]、CGS法 [14]、BiCGSTAB法 [15]、BiCGSTAB2法 [4]、BiCGSTAB(ℓ)法 [13]、GPBiCG [16] 法などの解

法がある [12]。これらの解法は各々特徴ある性質を有している。例えば、BiCGSTAB 法は、BiCG 法の残差ベクトルとそれを加速する 1 次の MR(Minimal Residual) 多項式との積を構成し残差を 2 ノルムに関して最小化するという、解法の高速性と安定化を実現した独自の解法である。その後、BiCGSTAB 法流の考えを受け継ぐ, 例えば、BiCGSTAB2 法や BiCGSTAB(ℓ) 法など有力な反復解法が続々と現れた。

一方、Lanczos 原理に基づく一般化積型反復法の理論の構築から誕生したのが一般積型の BiCG 法、すなわち GPBiCG 法と呼ばれる解法である [2] [16]。CG 法系統のアルゴリズムでは一般に二つのパラメータ : α_k, β_k が必要となるが, GPBiCG 法ではさらにパラメータが二つ ζ_k, η_k 必要になる。このため、反復第 k ステップにおける残差ノルム $||r_k||_2$ の最小化から二つのパラメータ : ζ_k, η_k を決定するために、2×2 の逆行列の計算を行う必要が生じ、結果として内積演算が増加することになった。総和計算や積和計算では, N 個のデータ a_i の総和計算 : $S = \sum_{i=1}^{N} a_i$ をするとき “情報落ち” という現象が現れ、求まった総和 S の値の精度が著しく低下することがある。そして、総和計算に対して欠落した情報を補償する Kahan の補償アルゴリズムが有効であることがよく知られている [3] [5] [6] [8]。

そこで、本論文では、第 3 章で総和計算に対する Kahan の補償アルゴリズムを、N 個のデータ a_i, b_i に対する積和計算 : $S = \sum_{i=1}^{N} a_i b_i$ に応用し、その有効性を精度および計算時間の両面から検証する。次に、第 4 章でこの補償アルゴリズムを GPBiCG 法のアルゴリズム中に現れる内積計算に応用し、補償付き内積計算を使用した GPBiCG 法の収束性や有用性について検証する。終りに第 5 章で簡単なまとめを行う。

3 補償付き総和計算と積和計算

積和計算 $S = a_1 b_1 + a_2 b_2 + \cdots + a_N b_N = \sum_{i=1}^{N} a_i b_i$ に対する通常の計算法 (以下、この方法を “定義通り” の積和計算と呼ぶことにする) によるプログラムの一例を Fig.1(a)) に、同じく補償付きアルゴリズムを採用したときの積和計算 (以下、“補償付き” の積和計算と呼ぶ) のプログラムの一例を Fig.1(b) に示す。
補償付き積和計算における補償過程の仕組みと Fig.1(b) 中の [2L] 行と [3L] 行との対応関係を Fig.2 に示す。ここで、Y の上位桁を Y_h、下位の桁を Y_ℓ で表すことにする。この図は、$|S| \geq |Y|$ と仮定すると、S+Y の加算において浮動小数演算の結果失われた Y

```
S=0.0
DO I=1,N
S=S+A(I)*B(I)
END DO
```

(a) 通常の内積計算

```
      S=A(1)*B(1)
      C=0.0
      DO I=2,N
[1L]  Y=A(I)*B(I)-C
[2L]  T=S+Y
[3L]  C=(T-S)-Y
[4L]  S=T
      END DO
```

(b) 補償付き内積計算

図 1: Two evaluation methods for the inner-product.

の下位の桁の情報 $-Y_\ell$ が一時的変数 C に入り、その符号を変えたものが次のステップの計算で S に加えられ精度が補償されるという仕組みを示している。

● Y_h は Y の上位の桁, Y_ℓ は Y の下位の桁を表す

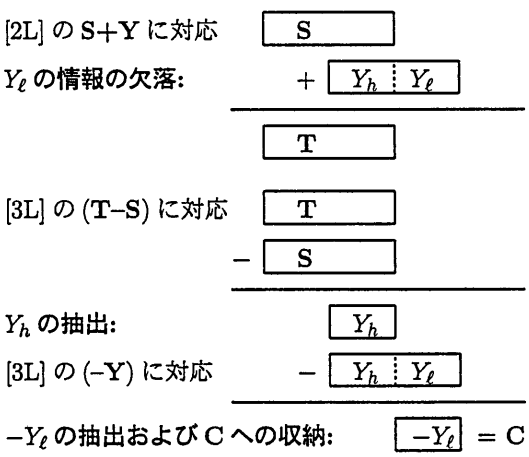

図 2: Mechanism of evaluation for the compensated inner-product.

3.1 補償付き積和計算に対するテスト問題

ここでは、Fig.3 に示す五つの問題 [9] に対して通常の積和計算および補償付き積和計算の絶対誤差と計算時間の比較を行った。ただし、すべての問題で項数 N は 200 万項と一定の値にした.

- 問題 (3.2), 同 (3.3), 同 (3.5) では $x = 0.5$,
- 問題 (3.2) では $a = 0.95$,

- 問題 (3.3) では x の値を区間 $(0, \pi/2)$ において等間隔に 128 分割した点の値を使用.
- 問題 (3.5) では二つの積和計算を別々に計算.

$$\sum_{i=1}^{N} \frac{1}{i} \times \frac{1}{i+1} = \frac{N}{N+1} \tag{3.1}$$

$$\sum_{i=1}^{N} \frac{a^{i-1}}{1+a^{i-1}x} \frac{1}{1+a^{i}x} = \frac{1}{1-a}\left(\frac{1}{1+x} - \frac{a^{N}}{1+a^{N}x}\right) \tag{3.2}$$

$$\sum_{i=1}^{N} \sin ix \sin(i+2)x = \frac{N}{2}\cos 2x - \frac{\cos(N+3)x \sin Nx}{2\sin x} \tag{3.3}$$

$$\sum_{i=1}^{N} \frac{1}{i(i+1)} \times \frac{1}{i+2} = \frac{1}{4} - \frac{1}{2(N+1)(N+2)} \tag{3.4}$$

$$\sum_{i=1}^{N} \cos(2i-1)x \sin 2ix + \sum_{i=1}^{N} \cos(2i+1)x \sin 2ix = \frac{\sin 2(N+1)x \sin 2Nx}{2\sin x} \tag{3.5}$$

図 3: Test problems for evaluation method of the inner-product.

3.2 補償付き積和計算に対する数値実験

数値実験は九州大学情報基盤センターに設置された 2 種類の計算機 Compaq GS320 および Fujitsu GP7000F/900 上で行った。計算機の主な仕様を Table 1 に示す。計算には 1PE(Processing Element) のみを使用した。使用言語は Fortran90 とし、コンパイルは各々f 90 コマンドおよび frt コマンドで行った。最適化オプションは、計算順序の変更による副作用 (誤差の増減) の影響を完全に除去するために今回は使用しなかった。計算はすべて倍精度演算で行った。倍精度演算での計算機イプシロンはおよそ 2.22×10^{-16} である。

Table 2 に、計算機 GS-320 を使用したときの 5 つの問題に対する計算結果を示す。同様に、Table 3 に、計算機 GP7000F/900 を使用したときの同計算結果を示す。時間の単位はすべてミリ秒である。また、Table2 と Table3 の"Computation" の欄の下線を付けた数字は真の値と異なる部分を意味し, そのときの誤差も示す. また, 括弧の中の数字は内積計算を定義通りに実行したときの時間を 1 としたときの各演算方法の時間の比率を表す.

表 1: A specification of Compaq GS320 and Fujitsu GP7000F/900.

	GS320	GP7000F/900
Vendor	Compaq	Fujitsu
CPU	Alpha21264	SPARC64
Clock	731 MHz	300 MHz
PE	32	64
Memory	64 GB	64 GB
OS	Tru64UNIX	Solaris7

- "補償" : 補償付き積和計算の結果。
- "定義" : 定義通りの積和計算の結果。
- "S 補償" : 計算順序の依存性を調べるために、計算順序乱数を発生させ計算順序を入れ換えた (Shaffle) ときの補償付き積和計算の結果。
- "S-D" : 同じくシャッフルさせて順番だけを変えた定義通りの積和計算の結果。
  (c) "S-補償" と (d) "S-D" の場合は、配列の番地の指定で間接アドレッシングを使用するので計算時間が余計に必要になる。また、シャッフルは一様乱数を発生させてそれを元に配列の順番を入れ換えた。
- Table 3 の中の "Lib.(Library の略)" : システムにあらかじめ用意されていた積和計算用の数学関数ライブラリー (サブルーチン名 : DSUM) を使用したときの結果。

Table 2 および Table 3 の結果から以下の 4 つのことがわかる。

1. どの問題でも補償付き積和計算の方が定義通りに積和計算を行うよりも誤差が少ない。
2. 計算時間は、補償付き積和計算の方が GS320 では 21% ～ 28%、GP7000F/900 では約 30% ほど余計にかかる。
3. 問題 (3.1) においてデータをシャッフルした場合でも、補償付き積和計算の方が誤差が少ない。

4. GP7000F/900 上のライブラリーを使用すると、定義通りに積和計算を行うときよりも誤差が 1/2 ～ 1/4 程度に減少する。しかし、補償付き積和計算を行う方が誤差が少ない。

表 2: Computations, errors, CPU time (in milliseconds) and ratios of several evaluation methods for inner-product of two millions data on GS320.

		Computation	Error	Time(ratio)
(3.1)	補償	.9999995000002500	0.0	46. (1.24)
	定義	.9999995000002957	.457E-13	37. (1.00)
	S 補償	.9999995000002501	.111E-15	253. (0.98)
	S-D	.9999995000002670	.171E-13	258. (1.00)
(3.2)	補償	13.33333333333332	0.0	46. (1.28)
	定義	13.33333333333331	.142E-13	36. (1.00)
(3.3)	補償	540302.4574874059	0.0	46. (1.24)
	定義	540302.4574874293	.234E-07	37. (1.00)
(3.4)	補償	.2499999999998750	0.0	46. (1.21)
	定義	.2499999999976820	.219E-11	38. (1.00)
(3.5)	補償	-.1921864745672792	.878E-12	90. (1.22)
	定義	-.1921865002368577	.256E-07	74. (1.00)

表 3: Computations, errors, CPU time (in milliseconds) and ratios of several evaluation methods for inner-product of two millions data on GP7000F/900.

		Computation	Error	Time(ratio)
(3.1)	補償	.9999995000002500	0.0	229. (1.29)
	定義	.9999995000002957	.457E-13	178. (1.00)
	Lib.	.9999995000002745	.245E-13	179. (1.00)
	S 補償	.9999995000002500	0.0	268. (1.21)
	S-D	.9999995000002938	.438E-13	221. (1.00)
(3.2)	補償	13.33333333333332	0.0	234. (1.31)
	定義	13.33333333333331	.142E-13	178. (1.00)
(3.3)	補償	540302.4574874059	0.0	234. (1.31)
	定義	540302.4574874293	.234E-07	178. (1.00)
	Lib.	540302.4574874123	.640E-08	179. (1.00)
(3.4)	補償	.2499999999998750	0.0	234. (1.31)
	定義	.2499999999976820	.219E-11	178. (1.00)
	Lib.	.2499999999991168	.758E-12	179. (1.00)
(3.5)	補償	-.1921864745672792	.878E-12	461. (1.29)
	定義	-.1921865002368577	.256E-07	356. (1.00)
	Lib.	-.1921864799223840	.535E-08	358. (1.00)

Fig.4 に、x の値として区間 $(0, \pi/2)$ を等間隔に 128 分割した点で問題 (3.3) の計算をしたときの 2 通りの方法：定義通り (図中に破線で示す)、補償付き (図中に実線で示す) による積和計算の絶対誤差をプロットしたものを示す。補償付きの場合、グラフの下方にスケールアウトしたときの誤差は厳密に 0 であったことを付記する。さらに、同様の条件で、三つの方法：定義通り (Definition)、補償付き (Compensated)、ライブラリー (DSUM) で積和計算をしたときの各方法の誤差の単純平均値を Table 4 に示す。これらの図や表の結果から、x の値にかかわらず、補償付きで積和計算を行う方が大幅に誤差が少ないことがわかった。

表 4: Average errors of three evaluation methods for problem (3.3) at 128 sample points of x defined in $(0, \pi/2)$.

Computer	Compensated	Definition	DSUM
GP7000	3.4873 E-09	5.1803 E-07	8.5919 E-08
GS320	3.4844 E-09	5.1803 E-07	-

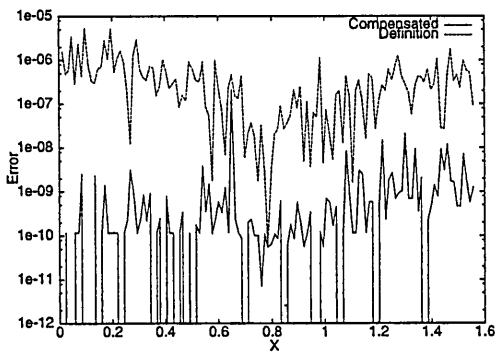

図 4: Plots of errors of two evaluation methods for $S = \sum_{i=1}^{2000000} \sin ix \sin(i+2)x$ at 128 sample points of x defined in $(0, \pi/2)$.

4 補償付き内積計算の GPBiCG 法への応用

ここでは、前章で扱った補償付き積和計算を GPBiCG 法のアルゴリズムの中の内積計算に応用し、補償付き (Compensated) 内積計算を使った GPBiCG 法 (以下では、C-GPBiCG 法と呼ぶことにする) の収束性について調べる。

4.1 反復法 GPBiCG 法について

一般化積型反復解法の GPBiCG 法のアルゴリズムは以下のようにまとめられる [16]。

● **GPBiCG 法のアルゴリズム:**

Let x_0 be an initial guess,
and put $r_0 = b - Ax_0$.
Set $t_{-1} = w_{-1} = 0$, $\beta_{-1} = 0$.
For $k = 0, 1, \ldots,$ do :
begin
$p_k = r_k + \beta_{k-1}(p_{k-1} - u_{k-1})$,
$y_k = t_{k-1} - r_k - \alpha_k w_{k-1} + \alpha_k A p_k$,
$t_k = r_k - \alpha_k A p_k$,
$u_k = \zeta_k A p_k + \eta_k(t_{k-1} - r_k + \beta_{k-1} u_{k-1})$,
$z_k = \zeta_k r_k + \eta_k z_{k-1} - \alpha_k u_k$,
$x_{k+1} = x_k + \alpha_k p_k + z_k$,
$r_{k+1} = t_k - \eta_k y_k - \zeta_k A t_k$,
$w_k = A t_k + \beta_k A p_k$
end.

ここで、収束過程におけるパラメータ α_k と β_k は $\alpha_k = \frac{(r_0^*, r_k)}{(r_0^*, Ap_k)}$, $\beta_k = \frac{\alpha_k}{\zeta_k} \cdot \frac{(r_0^*, r_{k+1})}{(r_0^*, r_k)}$ と与えられる。GPBiCG 法では、さらに二つのパラメータが必要であり、それらは残差ノルム $||r_{k+1}||_2 = ||t_k - \eta_k y_k - \zeta_k A t_k||_2$ の最小化から決定される。すなわち、

$$\begin{aligned} ||r_{k+1}||_2^2 &= ||t_k - \eta_k y_k - \zeta_k A t_k||_2^2 \\ &= (t_k, t_k) + \eta_k^2 (y_k, y_k) + \zeta_k^2 (At_k, At_k) - \\ &\quad 2\{\eta_k (t_k, y_k) + \zeta_k (y_k, At_k) - \eta_k \zeta_k (y_k, At_k)\} \end{aligned} \tag{4.1}$$

と表されるので、ζ_k, η_k に関して偏微分しそれらの値を 0 とおくと、次の二つの式が得られる。

$$(At_k, At_k)\zeta_k + (y_k, At_k)\eta_k = (t_k, At_k), \tag{4.2}$$
$$(y_k, At_k)\zeta_k + (y_k, y_k)\eta_k = (t_k, y_k). \tag{4.3}$$

これら二つの式から、パラメータ ζ_k と η_k は以下のように求められる。

$$\zeta_k = \frac{(y_k, y_k)(t_k, At_k) - (t_k, y_k)(y_k, At_k)}{(At_k, At_k)(y_k, y_k) - (y_k, At_k)(y_k, At_k)} \tag{4.4}$$
$$\eta_k = \frac{(At_k, At_k)(t_k, y_k) - (y_k, At_k)(t_k, At_k)}{(At_k, At_k)(y_k, y_k) - (y_k, At_k)(y_k, At_k)} \tag{4.5}$$

ただし、$k = 0$ のとき、$\zeta_k = \frac{(At_k, t_k)}{(At_k, At_k)}$, $\eta_k = 0$ とする。この二つのパラメータ ζ_k と η_k を決定するために、五つの内積計算: (y_k, y_k), (At_k, t_k), (y_k, t_k), (At_k, At_k), (At_k, y_k) を求める必要がある。このため、GPBiCG 法では内積計算をより正確に行うことが非常に重要である。ここで、補償付き内積演算を適用したのは、二つのパラメータ ζ_k, η_k を求めるために必要な 5 個の内積演算のみとした。すなわち、元の CG 法のパラメータ α_k, β_k の計算で必要な内積演算については定義通りに内積演算を行った。テストに使った二つの問題の出典は文献 [4] と文献 [7] である。

4.2 問題 1

ここでは、次の実数 Toeplitz 行列に対して元の GPBiCG 法と補償付内積演算を使った C-GPBiCG 法の収束性を調べた [4]。行列の次元数 N は (a)16384(=128^2) と (b)262144(=512^2) の 2 ケースである。また、係数行列に前処理は何も施していない。

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

連立 1 次方程式の右辺項は解ベクトルが全て 1 になるように定めた。また、パラメータ γ の値は (a)1.50 から 1.70 まで、(b)1.50 から 1.65 まで変化させ二つの反復法の収束性を調べた。Table 5 にケース (a)N=128^2 のとき、Table 6 にケース (b)N=512^2 のときの、収束までの反復回数と GS320 上での CPU 時間 (単位：秒) を上段に表す。また, 下段の数字は反復 1 回あたりの CPU 時間 (単位：ミリ秒) と GPBiCG 法の時間を 1.0 としたときの C-GPBiCG 法の時間の比率である. 収束判定条件は相対残差 2 ノルム $||r_k||_2/||r_0||_2$ が 10^{-12} 以下のときとした。解ベクトルの初期値はすべて 0、最大反復回数は 500 回とした。表中の Iter.(反復回数) の欄の "∞" 印は最大反復回数までの繰り返しでも収束しなかったことを表す。また, 表中の "GP" は GPBiCG 法を表し, "C-GP" とは補償付き内積演算を使った C-GPBiCG 法を表す. また, "It." は反復回数を "T" は時間を表す.

これらの２つの表から、

1. この問題ではパラメータ γ の値をだんだん大きくするといずれの反復法も解きにくくなり収束まで時間がかかること、

2. 元の GPBiCG 法に比べて補償付きの C-GPBiCG 法の方がより大きな γ の値まで、すなわちより解きにくい問題まで安定に解けること,

3. 補償付きの C-GPBiCG 法は反復１回当たりの計算時間が増えているので, 収束までの計算時間は元の GPBiCG 法に比べて必ずしも減少していないこと

などがわかる。

表 5: The number of iterations, total CPU time for convergence (in seconds), CPU time per one iteration (in mikki-sec.) and ratios of GPBiCG and C-GPBiCG methods in case of (a)N=128^2 on GS320.

	Parameter : γ									
	1.50		1.55		1.60		1.65		1.70	
	It.	T	It.	T	It.	T	It.	T	It.	T
GP	68	1.10	∞		∞		136	2.23	∞	
	16.2	1.0					16.4	1.0		
C-GP	56	1.23	80	1.41	94	1.65	107	1.88	116	2.04
	17.6	1.08	–	–	–	–	17.6	1.07	–	–

Fig.5 に (a)N=128^2 の場合、γ=1.70 のときの GPBiCG 法と C-GPBiCG 法の収束の履歴を示す。同様に、Fig.6 に (b)N=512^2 の場合、γ=1.62 のとき、Fig.7 に γ=1.65 のときの GPBiCG 法と C-GPBiCG 法の収束の履歴を示す。いずれの図も水平方向は反復回数、垂直方向は相対残差２ノルム (常用対数目盛) とする。図から以下のことが観察される.

1. Fig.5 では、元の GPBiCG 法の残差が 10^{-9} までしか残差が減少しないこと, またその後は発散に向かったこと, これに対して、C-GPBiCG 法の残差は要求精度の 10^{-12} まで収束速度を落さず減少し収束したこと、

表 6: The number of iterations, total CPU time for convergence (in sec.), CPU time per one iteration (in milli-sec.) and ratios of GPBiCG and C-GPBiCG methods in case of (b)N=512^2 on GS320.

	Parameter : γ									
	1.50		1.55		1.60		1.62		1.65	
	It.	T	It.	T	It.	T	It.	T	It.	T
GP	69	28.0	75	30.4	94	39.1	165	68.8	∞	
	405	1.0	405	1.0	416	1.0	417	1.0		
C-GP	72	30.5	69	29.2	83	35.1	97	41.2	136	57.6
	423	1.04	423	1.04	423	1.02	424	1.02	–	–

2. Fig.6 では、元の GPBiCG 法では残差が 10^{-10} まで小さくなると停滞を始め収束速度が鈍るのに対して, C-GPBiCG 法は停滞による収束の遅れもほとんどなく収束したこと、

3. Fig.7 でも、Fig.5 と同様に、GPBiCG 法の残差は 10^{-10} までしか減少せずその後は発散したこと。一方、C-GPBiCG 法の残差は 10^{-12} に達し正常に収束したこと、

4. いずれの場合も丸め誤差の影響が非常に大きいこと.

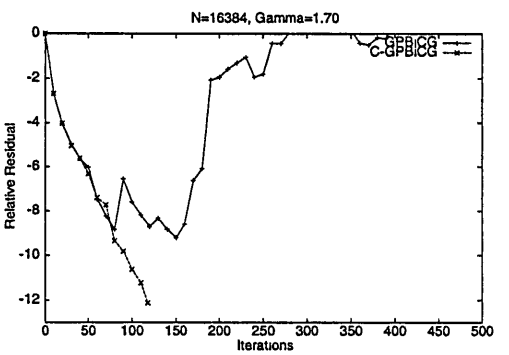

図 5: Convergence history of GPBiCG and C-GPBiCG methods for γ=1.70 in case of (a)N=128^2.

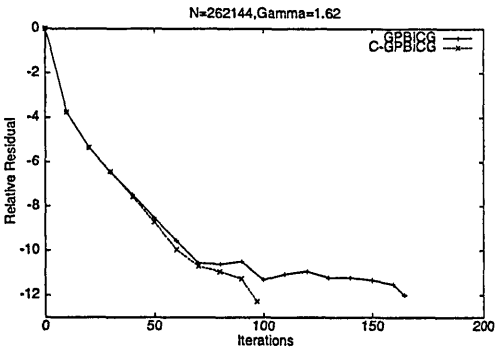

図 6: Convergence history of GPBiCG and C-GPBiCG methods for γ=1.62 in case of (b)N=512^2.

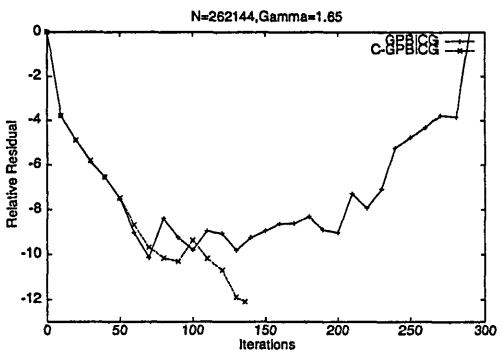

図 7: Convergence history of GPBiCG and C-GPBiCG methods for γ=1.65 in case of (b)N=512^2.

4.3 問題 2

2 次元領域 $[0,1] \times [0,1]$ で全周 Dirichlet 境界条件を課した以下の偏微分方程式 [7] に対して、通常の 2 次精度 5 点差分で離散近似し、得られた連立 1 次方程式を二つの反復解法 (元の GPBiCG 法と補償付き C-GPBiCG 法) で解いた。ここで、D は定数とする。

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

右辺の関数 $G(x,y)$ は解が全て $u(x,y) = 1+xy$ になるように定めた。格子点数は 128^2=16384 とし、格子幅 $h(=\frac{1}{128+1})$ は縦横軸方向ともに一定とした。前処理なしの場合とフィルインを考慮しない ILU(0) 分解を前処理として使用した場合の 2 ケースについて反復法の収束性を調べた。反復計算の初期値は全て 0、収束判定条件は相対残差 2 ノルム $||r_k||_2/||r_0||_2$ が 10^{-12} 以下とした。

Table 7 に前処理なしの場合、Table 8 と Table 9 に ILU(0) 分解の場合の収束までの反復回数と同 CPU 時間 (単位 : 秒) を各行の上段に示す。下段の数字は反復 1 回当たりの CPU 時間 (単位 : ミリ秒) と GPBiCG 法の時間を 1 としたときの比率である. ただし、Table 8 と Table 9 の違いは使用した計算機が前者が GS320 そして後者が GP7000F/900 だけである。また、Table 7 のときの最大反復回数は 5 万回とした。また、表中の "∞" 印は最大反復回数まで反復を繰り返しても収束しなかったことを表す。これらの表から、

1. Table 7 から、Dh の値が大きくなると収束までの反復回数が急激に増大し、収束しないこともあること、一方、Table 8 や Table 9 に示すように前処理 (ILU(0) 分解) を係数行列に行うと、
2. 収束までの反復回数はいずれも大幅に減少する。
3. しかし、$Dh = 2^{-2} \sim 2^{0}$ のときは解きにくく反復回数がまだ多いことがわかる。さらに、
4. 補償付きの C-GPBiCG 法は元の GPBiCG 法に比べて反復回数をかなり割合で減少させる効果があるが、実際の計算時間の面では、GS320 上では時間短縮の効果が少ないが、GP7000F/900 上ではかなりの効果がある

ことがわかった。

Fig. 8 に Dh=2^{-1} のときの前処理なしの GPBiCG 法と前処理なしの C-GPBiCG 法の収束の履歴を示す。

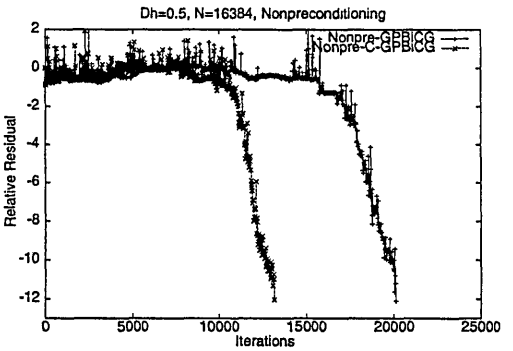

図 8: Convergence history of nonpreconditioning GPBiCG and nonpreconditioning C-GPBiCG methods for Dh=2^{-1}.

表 7: The number of iterations, total CPU time for convergence (in seconds), CPU time per one iteration (in mille-sec.) and ratios of nonpreconditioning GPBiCG and C-GPBiCG methods on GS320.

	Dh									
	0.0		2^{-3}		2^{-2}		2^{-1}		2^{0}	
	It.	T	It.	T	It.	T	It.	T	It.	T
GP	1601	6.53	3903	15.9	7648	31.6	20115	83.3	∞	
	4.076	1.0	4.07	1.0	4.13	1.0	4.14	1.0		
C-GP	1649	9.42	3858	23.5	7694	44.7	13177	75.9	∞	
	5.71	1.40	6.08	1.49	4.13	1.41	5.76	1.39		

表 8: The number of iterations, total CPU time for convergence (in sec.), CPU time per one iteration (in milli-sec.) and ratios of ILU(0)-GPBiCG and ILU(0)-C-GPBiCG methods on GS320.

	Dh											
	2^{-2}		2^{-1}		2^{0}		2^{2}		2^{3}		2^{4}	
	It.	T	It.	T	It.	T	It.	T	It.	T	It.	T
GP	2008	11.3	2407	13.2	2452	13.5	673	3.70	321	1.82	237	1.31
	5.63	1.0	5.46	1.0	5.49	1.0	5.49	1.0	5.67	1.0	5.52	1.0
C-GP	1571	11.2	2564	18.4	2383	17.0	742	5.33	267	1.92	168	1.22
	7.14	1.27	7.18	1.31	7.14	1.30	7.18	1.30	7.17	1.27	7.23	1.31

表 9: The number of iterations, total CPU time for convergence (in sec.), CPU time per one iteration (in milli-sec.) and ratios of ILU(0)-GPBiCG and ILU(0)-C-GPBiCG methods on GP7000F/900.

	Dh											
	2^{-2}		2^{-1}		2^{0}		2^{2}		2^{3}		2^{4}	
	It.	T	It.	T	It.	T	It.	T	It.	T	It.	T
GP	2008	42.7	2407	50.6	2452	51.8	673	14.2	321	6.77	237	5.00
	21.3	1.0	21.0	1.0	21.1	1.0	21.0	1.0	21.1	1.0	21.1	1.0
C-GP	1571	36.8	2564	60.8	2383	55.9	742	17.4	267	6.29	168	3.95
	23.4	1.10	23.7	1.13	23.4	1.11	23.4	1.11	23.5	1.12	23.5	1.11

プロットは反復 20 回おきに行った。Fig. 9 に Dh=2^{-2} のときの ILU(0)-GPBiCG 法、ILU(0)-C-GPBiCG 法の収束の履歴を示す。プロットは反復 5 回おきに行った。いずれの場合も、

1. 収束過程の前半では両反復解法ともに残差が停滞する様相を示すが、
2. 収束過程の後半になると収束性に大きな開きが出て補償付き C-GPBiCG 法の方が元の GPBiCG 法に比べて収束性がよいこと

がわかる. これらの収束性の違いが内積演算の方法の違いだけで現れたことが特徴的である. しかしながら, 補償付き内積計算を行なうと,

1. 反復 1 回当たり GS320 では約 30%, GP7000F では約 10%余分に計算時間を必要とすること,
2. また, 例えば, 表 9 の Dh=2^{-1} のときの結果が示すように, 元の GPBiCG 法の反復回数に比べて C-GPBiCG 法の反復回数が逆に増加したこと,

などの問題点もいくつかあることも判明した. 特に, 2 つ目の問題点は単に内積計算の高精度化を図る補償演算だけでは解決し得ない反復解法自体の本質とも考えられるので, 別の解析手段やさらに慎重な考察や洞察が必要と思われる. これらは今後の課題としたい.

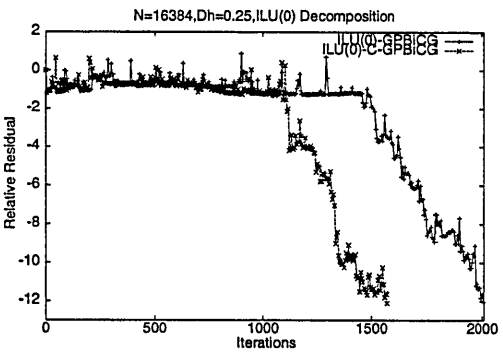

図 9: Convergence history of ILU(0)-GPBiCG and ILU(0)-C-GPBiCG methods for $Dh=2^{-2}$.

5 おわりに

内積計算に対する補償付き演算の精度と時間について検証した。さらに, 補償付き内積演算を組み込んだ C-GPBiCG 法の収束性を調べた。その結果、元の GPBiCG 法では残差の停滞や発散などが見られた場合でも補償付き C-GPBiCG 法は収束しその有効性が示された。

謝辞

本研究を進める過程で, 東京大学大学院工学系研究科 張紹良助教授ならびに理化学研究所情報環境室 阿部邦美博士との度々に亘る有益なる議論が大いに参考になった. 両氏に心より感謝の意を表したい.

参考文献

[1] Fletcher, R., Conjugate Gradient Methods for Indefinite Systems, Lecture Notes in Mathematics 506(1976), 73-89.

[2] 藤野清次, 張紹良, 反復法の数理, 朝倉書店, 東京, 1996.

[3] Goldberg, D., What every computer scientist should know about floating-point arithmetic, *ACM Computing Surveys*, 23(1991), 5-48.

[4] Gutknecht, M.H., Variants of BiCGSTAB for Matrix with Complex Spectrum, *SIAM J. Sci. Comput.*, 14(1993), 1020-1033.

[5] 長谷川武光, 数値計算のつぼ (1) $0.1 \times 10 = 1$?, 名古屋大学大型計算機センター ニュース, 31(2000), 271-280.

[6] Higham, N., Accuracy and Stability of Numerical Algorithms, *SIAM*, Philadelphia, 1996.

[7] Joubert, W., Lanczos methods for the solution of nonsymmetric systems of linear equations, *SIAM J. Matrix Anal. Appl.*, 13(1992), 926-943.

[8] Kahan, W., Further remarks on reducing truncation errors, *Comm. ACM*, 8(1965), p.40.

[9] 森口繁一, 宇田川欽久, 一松信, 岩波 数学公式 II 級数, 岩波書店, 東京, 1988.

[10] 二宮市三, 応用数理の遊歩道 (23) 数値計算にこと寄せて, 応用数理学会誌, 10(2000), 79-83.

[11] Reichel, L., Trefethen, L.N., Eigenvalues and Pseudo-eigenvalues of Toeplitz Matrices, *Lin. Alg. Appl.*, 162-164(1992), 153-185.

[12] Saad, Y., van der Vorst, H.A., Iterative Solution of Linear Systems in the 20-th Century, Report *umi*-99-152, Univ. of Minnesota, Mineapolis, 1999.

[13] Sleijpen, G., Fokkema, D.R., BiCGstab(ℓ) for linear equations involving unsymmetric matrices with complex spectrum, *Electronic Trans. on Numer. Anal.*, 1(1993), 11-32.

[14] Sonneveld, P., CGS: A Fast Lanczos-type Solver for Nonsymmetric Linear Systems, *SIAM J. Sci. Stat. Comput.*, 10(1989), 36-52.

[15] van der Vorst, H.A., 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), 631-644.

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