ハイパフォーマンス
コンピューティング 49－4
（1993．10．14）

# 差分スキームの再考による並列計算機向き不完全 LU 分解について

*藤野 清次　　**竹内 敏己

*計算流体力学研究所　　**花王 株式会社

本報告では、使用される差分スキームを再考することによって、複数の CPU を持つベクトル計算機向き不完全 LU 分解について考察する。ここで考える差分スキームは、高い計算効率を得るために通常の 5 点差分の点の配置とは異なった配置をしている。その工夫によって水平線上の点を同時に更新できる。またここで取り扱う格子は縦横比が等しい等方性格子とする。本研究では、本分解法の収束特性: 収束までの反復回数、CPU 時間そして収束解の精度などが調べられた。その結果、本分解法は従来の 5 点差分スキームのベクトル化手法: 超平面法 (Hyperplane method) に比べて大変効率が高く、並列計算機向きであることがわかった。

# ILU factorization suited to vector computer by consideration on difference schemes

*Seiji Fujino　　**Toshiki Takeuchi

*Institute of Computational Fluid Dynamics　　** KAO Corporation

In this paper Incomplete LU factorization suited to vector computer is proposed by consideration on difference schemes. The distribution of stencils used in the present difference schemes differs from that of stencils of usual 5 points difference schemes for gaining high efficiency. The gridpoints on the horizontal line can be simultaneously calculated on the vector computer. The convergence property of the present factorization is examined in terms of number of iterations, CPU time and preciseness of converged solutions. It can be shown that the proposed method is more efficient than the usual hyperplane method through some numerical experiments on vector computer.

## 1. はじめに

ここでは、2次元長方形領域で定義された方程式を2次精度の中心差分で離散化した方程式に対して、対称行列用の前処理つき共役勾配法 (Preconditioned Conjugate Gradients Method)[5] あるいは非対称行列用の前処理つき Bi-CG STAB 法 [8] を使ってベクトル計算機上で効率よく解く方法を考える。領域内の格子点の順番付けを辞書式でしたとき、離散化後の方程式は通常の5点差分近似式で表される。前処理として使用される不完全 (Incomplete)Cholesky 分解付き CG 法 (以下 ICCG 法と呼ぶ) あるいは不完全 LU 分解付き Bi-CG STAB 法 (以下 ILU/Bi-CG STAB 法と呼ぶ) などは、超平面法 (Hyperplane Method) を使ってベクトル化されることが多い [7]。この超平面法は、使用法が簡便な反面、メモリ中のデータを等間隔にアクセスすることになるため、汎用のベクトル計算機では計算速度に必然的に限界が生じる。また、アクセス間隔の大きさによっては、Memory Bank Conflict(メモリ衝突) と呼ばれる現象が現れ、ベクトル計算機の持つ高速演算性能が大幅に低下することが知られている [1],[2],[3]。ここで、汎用のベクトル計算機とはメモリが多バンク構成からなるベクトル計算機を意味する。ICCG 法や ILU/Bi-CG STAB 法などが、ベクトル計算機を使って半導体製造のデバイス/プロセスシミュレーションや電磁場解析などにおいて盛んに用いられている現在、この前処理方法の一層の高速化を図ることは十分意義があることだと思われる [6]。

本研究では、メモリへのアクセスが連続かつベクトル長も十分長いベクトル計算機向き不完全 LU 分解について、以下の手順に従って考察する。最初に、第2章で従来の5点差分スキームに対する超平面法によるベクトル化と、本論文で提案する新しい差分スキームならびにそのベクトル化について論じる。第3章では、本分解法における元の係数行列 A の要素と不完全 LU 分解後の行列要素との対応、および Gustafsson 流の修正 [4], [7] を施したときのそれぞれの要素の対応について述べる。第4章では、縦横比が等しい等方性格子を用いたとき、対称/非対称な係数行列に対して、それぞれ MICCG(Modified ICCG) 法および MILU/Bi-CG STAB 法で解いたときの、従来の超平面法と本方法によるベクトル化の比較検討を行なう。収束時の近似解の精度比較、収束までの反復回数と CPU 時間およびパラメータ依存性などの観点から両者の収束特性について調べる。以上の考察ならびにベクトル計算機による数値実験から、本論文で提案するベクトル化の方法が、従来の超平面法によるベクトル化に比べて効率的であることを明らかにする。

## 2. 差分スキームとベクトル化

以下では、格子点の順番付けはすべて辞書式 (自然な順番付け) で行なうものとする。

### 2.1 従来の5点差分スキームと超平面法

Fig.1(a) に通常の5点差分スキームで用いられる5個の格子点を示す。また Fig.1(b) に不完全分解の処理のベクトル化などで用いられる超平面の1例を示す。

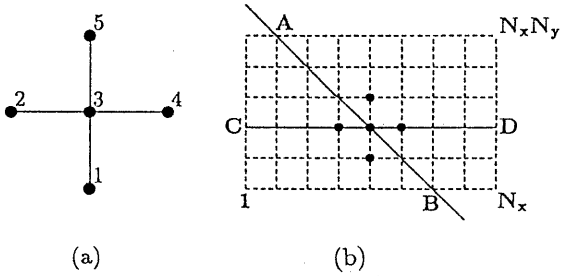



Fig. 1. Usual 5 points difference scheme and its vectorization

ベクトル計算機では、Fig.1(b) 中の斜め線 (点 A と点 B を結んだ線) 上の点を同時に前進/後退

代入の計算を行なうことができる。しかし、水平方向の格子点数をいま$N_x$とすると超平面上の格子点番号の間隔は$N_x-1$になり、メモリへ連続的にアクセスできない。そのため、計算速度はその計算機の持つ最高演算速度に比べてかなり低い。また、平均ベクトル長もオーダ$O(\frac{N_x}{2})$と短くその大きさも一定でないため、この方法でベクトル計算機の高速性を十分に引き出すのは難しい。

そこで、Fig.1(b) から容易に推察できるように、点 C と点 D を結んだ水平線上の格子点を差分スキームで使用しないようにできれば、メモリへ連続的にアクセスできる。そのためには、差分スキーム自体を設計する、すなわち格子点の個数と配置を決定する、必要が生じる。次の節で差分スキームを具体的に導出する。

## 2.2 7(5) 点差分スキームとベクトル化

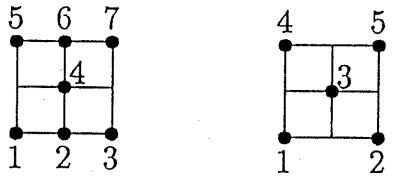
ここでは次のようなタイプの方程式

$$c_1 f + c_2\frac{\partial f}{\partial x} + c_3\frac{\partial f}{\partial y} + c_4\frac{\partial^2 f}{\partial x^2} + c_5\frac{\partial^2 f}{\partial x \partial y} + c_6\frac{\partial^2 f}{\partial y^2} = c_0 \tag{2.1}$$

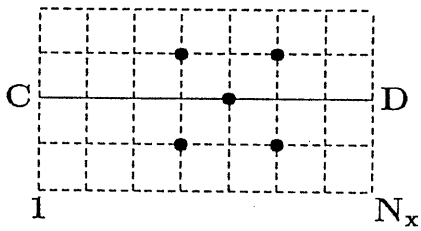
を考える。ここで$c_i$ $(i=0,\cdots,6)$ は定数とする。そして次の三つの条件を満たすように格子点$(i_0, j_0)$ に対する差分スキームを構成する。

- 格子点$(i_0, j_0)$ が位置する水平線 (Fig.1(b) 点 C と点 D を結んだ線) 上の格子点を差分スキームで使わない.
- 格子点$(i_0, j_0)$ との距離が近い格子点を使う (この方が精度を保つ上で好ましい).
- 周囲の格子点の配置も格子点$(i_0, j_0)$ を中心とする対称性をできるだけ保存する.

この条件を満たすものとして、Fig.2(a),(b) に示すような差分スキームが考えられる。




(a)7 points　　(b)5 points




(c)Horizontal line

Fig. 2. Difference schemes with 7 and 5 points and its vectorization

### 2.2.1 対称な 7 点を使う差分スキーム

いま、Fig.2(a) の格子点配置の場合を考える。そのとき、2 変数$x, y$の関数$f$=$f(x,y)$ が定義された格子点$(i_0, j_0)$ とその周囲 6 点の Taylor 展開は次のように表せる。ここで、$h$, $k$はそれぞれ$x, y$方向の格子幅、そして$h_s = h^2 + k^2$とする。

$$\begin{aligned}
f(x+h,y+k) &= f + hf_x + kf_y + \frac{h^2}{2}f_{xx} + \frac{k^2}{2}f_{yy} \\
&\quad + hkf_{xy} + O(h_s) \\
f(x-h,y+k) &= f - hf_x + kf_y + \frac{h^2}{2}f_{xx} + \frac{k^2}{2}f_{yy} \\
&\quad - hkf_{xy} + O(h_s) \\
f(x+h,y-k) &= f + hf_x - kf_y + \frac{h^2}{2}f_{xx} + \frac{k^2}{2}f_{yy} \\
&\quad - hkf_{xy} + O(h_s) \\
f(x-h,y-k) &= f - hf_x - kf_y + \frac{h^2}{2}f_{xx} + \frac{k^2}{2}f_{yy} \\
&\quad + hkf_{xy} + O(h_s)
\end{aligned}$$

$$
\begin{aligned}
f(x,y+k) &= f + kf_y + \frac{k^2}{2}f_{yy} + O(k^2)\\
f(x,y-k) &= f - kf_y + \frac{k^2}{2}f_{yy} + O(k^2)\\
f(x,y) &= f
\end{aligned}
$$

また、$f_x, f_y$は関数$f$を$x$あるいは$y$で偏微分することを意味する。$f_{xx}, f_{yy}, f_{xy}$なども同様である。いま (2.1) 式を離散化した次の差分展開式を考える。係数$a_i$は定数とする。

$$
\begin{aligned}
(2.2)\ & a_1 f(x+h,y+k) + a_2 f(x-h,y+k) +\\
& a_3 f(x+h,y-k) + a_4 f(x-h,y-k) +\\
& a_5 f(x,y+k) + a_6 f(x,y-k) +\\
& a_7 f(x,y) = c_0
\end{aligned}
$$

このとき、(2.1) 式と (2.2) 式との間には次の関係がある。

$$
\begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1\\
1 & -1 & 1 & -1 & 0 & 0 & 0\\
1 & 1 & -1 & -1 & 1 & -1 & 0\\
1 & 1 & 1 & 1 & 0 & 0 & 0\\
1 & -1 & -1 & 1 & 0 & 0 & 0\\
1 & 1 & 1 & 1 & 1 & 1 & 0
\end{pmatrix}
\begin{pmatrix} a_1\\ a_2\\ a_3\\ a_4\\ a_5\\ a_6\\ a_7 \end{pmatrix}
=
\begin{pmatrix} c_1\\ \frac{1}{h}c_2\\ \frac{1}{k}c_3\\ \frac{2}{h^2}c_4\\ \frac{1}{hk}c_5\\ \frac{2}{k^2}c_6 \end{pmatrix}
$$

ところが、(2.2) 式の未知の係数$a_i$は全部で7個あるのに対し、(2.1) 式で与えられた係数$c_i$は6個しかない。離散式の式の数が1つだけ多い。そこで、自由度を一つ (それを$a_6$とする) 持たせた行列表現が次のように導ける。

$$
\begin{pmatrix} a_1\\ a_2\\ a_3\\ a_4\\ a_5\\ a_6\\ a_7 \end{pmatrix}
= \frac{1}{4}
\begin{pmatrix}
0 & 1 & 1 & 2 & 1 & -1\\
0 & -1 & 1 & 2 & -1 & -1\\
0 & 1 & -1 & 0 & -1 & 1\\
0 & -1 & -1 & 0 & 1 & 1\\
0 & 0 & 0 & -4 & 0 & 4\\
0 & 0 & 0 & 0 & 0 & 0\\
4 & 0 & 0 & 0 & 0 & -4
\end{pmatrix}
\times
\begin{pmatrix} c_1\\ \frac{1}{h}c_2\\ \frac{1}{k}c_3\\ \frac{2}{h^2}c_4\\ \frac{1}{hk}c_5\\ \frac{2}{k^2}c_6 \end{pmatrix}
+ \frac{a_6}{2}
\begin{pmatrix} 1\\ 1\\ -1\\ -1\\ -2\\ 2\\ 0 \end{pmatrix}
$$

このとき$a_6 = a_5$とおくと各係数$a_i$は次のように表せる。係数$a_6$を一つ余分に使うことになるが、得られた差分スキームは格子点$(i_0, j_0)$を中心とする対称性が保たれている。

$$
\begin{pmatrix} a_1\\ a_2\\ a_3\\ a_4\\ a_5\\ a_6\\ a_7 \end{pmatrix}
= \frac{1}{4}
\begin{pmatrix}
0 & 1 & 1 & 1 & 1 & 0\\
0 & -1 & 1 & 1 & -1 & 0\\
0 & 1 & -1 & 1 & -1 & 0\\
0 & -1 & -1 & 1 & 1 & 0\\
0 & 0 & 0 & -2 & 0 & 2\\
0 & 0 & 0 & -2 & 0 & 2\\
4 & 0 & 0 & 0 & 0 & -4
\end{pmatrix}
\begin{pmatrix} c_1\\ \frac{1}{h}c_2\\ \frac{1}{k}c_3\\ \frac{2}{h^2}c_4\\ \frac{1}{hk}c_5\\ \frac{2}{k^2}c_6 \end{pmatrix}
$$

[例] 次の形の Helmholtz 方程式の場合、すなわち$c_2 = c_3 = c_5 = 0$のときを考える。

$$c_1 f + c_4 \frac{\partial^2 f}{\partial x^2} + c_6 \frac{\partial^2 f}{\partial y^2} = c_0.$$

このとき各係数$a_i$は次のようになる。

$$a_1 = a_2 = a_3 = a_4 = \frac{1}{2h^2}c_4,\ a_5 = a_6 = \frac{1}{k^2}c_6 - \frac{1}{h^2}c_4,\ a_7 = c_1 - \frac{2}{k^2}c_6$$

### 2.2.2 5点を使う差分スキーム

等方性格子: $h = k$で$c_4 = c_6$のとき、次の5個の係数だけで差分スキームが表せ、7点を使う差分スキームに比べて効率面で有利となる。このとき格子点の配置は Fig.2(b) のようになる。この5点差分スキームの効率評価は第4章で行なう。

$$a_1 = a_2 = a_3 = a_4 = \frac{1}{2h^2}c_4,\ a_7 = -\frac{2}{k^2}c_4$$

## 2.3 水平線法

以下では、従来の超平面法によるベクトル化と識別するために、格子点$(i_0, j_0)$が位置する水平線 (Fig.2(c) 中の点 **C** と点 **D** を結んだ線) 上の格子点を同時に計算できるという意味で、Fig.2(a),(b) に示した差分スキームに対するベクトル化の方法を、水平線法(Horizontal Line Method) と呼ぶことにする。Table 1 に超平面法と水平線法の主な特徴をまとめた。表中で$N_x$は$x$方向の格子点数とする。

Table 1. Characteristics of hyperplane and horizontal line method

| | 超平面法 | 水平線法 | |
|---|---|---|---|
| 格子幅比 | 等方/非等方 | 等方 | 非等方 |
| ベクトル長 | $O(\frac{N_x}{2})$ | $N_x$ | $N_x$ |
| 差分点数 | 5 | 5 | 7 |
| アクセス間隔 | $N_x - 1$ | 連続 | 連続 |

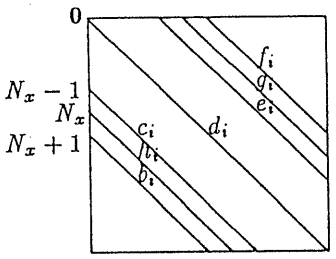



Fig. 3. Non-zero elements of matrix A

超平面法と水平線法、両者のベクトル計算機上の性能については第4章で調べる。

## 3. 行列要素の対応

ここでは、元の係数行列 A の要素と不完全 LDU 分解後の行列要素との対応について記述する。いま $x$ 方向の格子点数を $N_x$とする。元の係数行列 A の対角要素を $d_i$とし, 対角要素の下 $N_x-1, N_x, N_x+1$ だけ離れた要素をそれぞれ $c_i, h_i, b_i$とする。同様に、対角要素の上 $N_x-1, N_x, N_x+1$ だけ離れた要素をそれぞれ $e_i, g_i, f_i$とする。Fig.3 に行列 A の非零要素を示す。不完全 LDU 分解における非零要素は元の行列 A の非零要素と同じ位置のものだけとする。このとき、下 (上) 三角行列 L(U) の非零要素は元の行列 A の対応する要素と等しくなる。また、対角行列 D の要素 $dd_i$は、

$$\begin{aligned} dd_i^{-1} &= d_i - b_i f_{i-N_x-1} d_{i-N_x-1} \\ &\quad -c_i e_{i-N_x+1} d_{i-N_x+1} - h_i g_{i-N_x} d_{i-N_x} \end{aligned}$$

で表される。Gustafsson 流の修正不完全 LDU 分解をしたときの対角項 $dd_i$は、

$$\begin{aligned} dd_i^{-1} &= d_i - b_i f_{i-N_x-1} d_{i-N_x-1} \\ &\quad -c_i e_{i-N_x+1} d_{i-N_x+1} - h_i g_{i-N_x} d_{i-N_x} \\ &\quad -\alpha\{b_i(e_{i-N_x-1} + g_{i-N_x-1}) d_{i-N_x-1} \\ &\quad +c_i(f_{i-N_x+1} + g_{i-N_x+1}) d_{i-N_x+1} \\ &\quad +h_i(e_{i-N_x} + f_{i-N_x}) d_{i-N_x}\} \end{aligned}$$

で表される。ここで$\alpha$はパラメータである。Fig.2 (b) の5点差分スキームの場合は、上式で $h_i$=$g_i$=0 としたものに相当する。

## 4. 等方性格子に対する二つの方法の特性評価

ここで使用した差分スキームは、Fig.1(a) と Fig.2(c) に示した5点差分である。また数値実験はすべてベクトル計算機 VP-200(最高演算速度: 570Mflops) を使用した。

### 4.1 対称行列のとき

解いた方程式はポアソン方程式: $f_{xx}+f_{yy}=-b$ で、厳密解が異なる三つの問題を (M)ICCG 法で解いた。解析領域は $[0,1]\times[0,1]$ とし、演算は倍精度で行なった。(M)ICCG 法の収束判定は相対残差 $L_2$ノルムが $10^{-12}$以下とした。右辺項 $b$ は方程式の解が厳密解となるように定めた。

- 問題 A の厳密解: $f_A(x,y) = e^{-2x^2} + e^{-2y^2}$
- 問題 B の厳密解: $f_B(x,y) = e^{xy}$
- 問題 C の厳密解: $f_C(x,y) = \sin(\pi x)\sin(\pi y)$

Table 2(a),(b) に格子点数: 250×250 のときの、全周 Dirichlet 境界条件および混合型境界条件 (左右の境界を Neumann 条件、上下の境界を Dirichlet 条件にした) に対する、収束解と厳密解との差の絶対値を示す。本差分スキームの打ち切り誤差が $h_s = h^2+k^2$で評価されるため、実際の計算でも従来の超平面法の結果より少し誤差が大きいことが Table 2(a) からわかる。Table 2(b) では、境界の Neumann 条件の指定を1次精度の1階微分値で与えたため、Table 2(a) の結果より誤差が大きくなっている。また、Table 3 に計算速度 (Mflops 単位) と超平面法のときのアクセス間隔を示す。括弧内の数字は超平面法の速度を1としたときの比率を表す。超平面法

ではアクセス間隔が4の倍数(ケース(c))や8の倍数(ケース(d))のとき、メモリ衝突の影響を強く受けて計算速度が低下するのに対し、水平線法ではメモリへのアクセスがすべて連続的に行なわれるのでメモリ影響を受けないことがわかる。またその速度も超平面法のときに比べはるかに速く、最高演算速度の71%まで達した(格子点数500×500のとき)。

Table 2. Estimation of error between converged and exact solutions

(a)Results for Dirichlet boundary conditions

| 問題 | 誤差 | 超平面法 | 水平線法 |
|---|---|---|---|
| A | 最大誤差 | $2.78\times10^{-6}$ | $2.78\times10^{-6}$ |
| | $L_2$ノルム | $4.69\times10^{-9}$ | $4.69\times10^{-9}$ |
| B | 最大誤差 | $5.08\times10^{-8}$ | $2.68\times10^{-6}$ |
| | $L_2$ノルム | $1.01\times10^{-10}$ | $5.86\times10^{-9}$ |
| C | 最大誤差 | $1.33\times10^{-5}$ | $5.31\times10^{-5}$ |
| | $L_2$ノルム | $2.64\times10^{-8}$ | $1.06\times10^{-7}$ |

(b)Results for mixed boundary conditions

| 問題 | 誤差 | 超平面法 | 水平線法 |
|---|---|---|---|
| A | 最大誤差 | $2.52\times10^{-2}$ | $3.71\times10^{-2}$ |
| | $L_2$ノルム | $4.51\times10^{-6}$ | $7.28\times10^{-5}$ |
| B | 最大誤差 | $1.27\times10^{-2}$ | $1.14\times10^{-2}$ |
| | $L_2$ノルム | $2.45\times10^{-5}$ | $2.60\times10^{-5}$ |
| C | 最大誤差 | $1.21\times10^{-3}$ | $1.28\times10^{-3}$ |
| | $L_2$ノルム | $1.48\times10^{-6}$ | $1.40\times10^{-6}$ |

Table 3. Computational speeds for various cases

| | 格子点数 | 超平面法のアクセス間隔 | 超平面法 Mflops | 水平線法 Mflops |
|---|---|---|---|---|
| (a) | $099^2$ | 098 :偶数 | 126 (1.00) | 257 (2.04) |
| (b) | $100^2$ | 099 :奇数 | 138 (1.00) | 258 (1.88) |
| (c) | $101^2$ | 100 :4の倍数 | 103 (1.00) | 264 (2.57) |
| (d) | $249^2$ | 248 :8の倍数 | 128 (1.00) | 373 (2.92) |
| (e) | $250^2$ | 249 :奇数 | 238 (1.00) | 373 (1.57) |
| (f) | $251^2$ | 250 :偶数 | 190 (1.00) | 373 (1.97) |

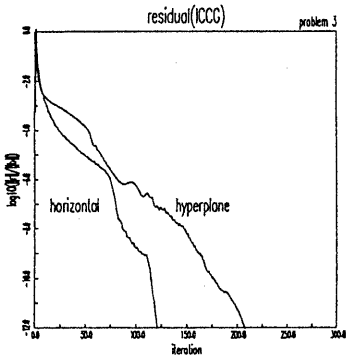
Table 4に収束までの反復回数とCPU時間を示す。括弧内の数字はMICCG法の結果である。計算時間がおよそ半減(44%~54%)したことがわかる。これは、水平線法によるベクトル化により、(1)不完全分解に要する時間が短くなったことと、(2)反復回数自体が減少したこと、この二つの要因によるところが大きい。また、Table 5は、格子点数: 250×250のときの主な処理の反復1回当たりのCPU時間(ミリ秒単位)と不完全分解に要した時間を比較したものである。表中で「行列の積」とは、行列Aとベクトルとの積の計算をさす。水平線法によって、不完全分解に要する時間が37%に激減したことがわかる。Fig.4(a),(b)に問題Cに対する相対残差$L_2$ノルムの履歴を示す。

Table 4. Number of iterations and CPU time

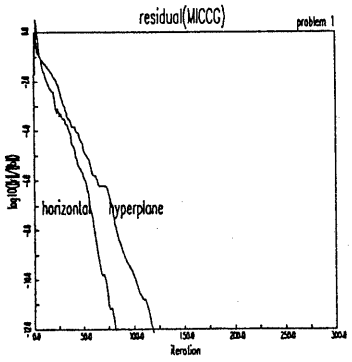
| 問題 | 収束性 | 超平面法 | 水平線法 |
|---|---|---|---|
| A | 反復回数 | 264 (136) | 224 (115) |
| | CPU (秒) | 2.50 (1.29) | 1.33 (.691) |
| B | 反復回数 | 273 (137) | 224 (115) |
| | CPU (秒) | 2.57 (1.29) | 1.33 (.689) |
| C | 反復回数 | 208 (119) | 121 (082) |
| | CPU (秒) | 1.94 (1.11) | .721 (.490) |

Table 5. CPU time per one iteration

| 内 訳 | 超平面法 | 水平線法 |
|---|---|---|
| 行列の積 | 1.34 | 1.33 |
| 不完全分解 | 7.02 (1.00) | 2.57 (0.37) |
| その他 | 2.07 | 2.05 |
| 合 計 | 10.43(1.00) | 5.95 (0.57) |




(a)ICCG法




(b)MICCG 法

Fig. 4. Convergence history for problem C

## 4.2 非対称行列のとき

ここでは、1 階の微分項とパラメータ $B$ を含む方程式 $f_{xx}+f_{yy}+Bf_x=-b$ を (M)ILU/Bi-CG STAB 法で解いた。解析領域は $[0,1]\times[0,1]$ とし、境界条件は全周 Dirichlet 条件そして演算は倍精度で行なった。(M)ILU/Bi-CG STAB 法の収束判定は相対残差 $L_2$ ノルムが $10^{-12}$ 以下とした。厳密解は対称行列のときと同じものとした。パラメータ $B$ は (= 2, 10, 100) の 3 通り変化させ、右辺項 $b$ は方程式の解が厳密解となるように定めた。Table 6 に格子点数: 250×250 のときの収束解と厳密解との差の絶対値を示す。三つの問題の中では、問題 B で従来の 5 点差分スキームに比べて本差分スキームの方が誤差が若干大きいが、他の二つの問題ではむしろ本分解法の方が誤差が小さい。

Table 6. Error estimation of two methods

| 問題 | 誤差 | 超平面法 | 水平線法 |
|---|---|---|---|
| A | 最大誤差 | $2.87\times10^{-4}$ | $2.70\times10^{-5}$ |
| | $L_2$ ノルム | $5.36\times10^{-7}$ | $4.36\times10^{-9}$ |
| B | 最大誤差 | $1.46\times10^{-4}$ | $1.00\times10^{-3}$ |
| | $L_2$ ノルム | $2.99\times10^{-7}$ | $2.26\times10^{-6}$ |
| C | 最大誤差 | $1.95\times10^{-3}$ | $7.59\times10^{-4}$ |
| | $L_2$ ノルム | $3.90\times10^{-6}$ | $1.50\times10^{-6}$ |

Table 7. Number of iterations and CPU time

| 問題 | 収束性 | 超平面法 | 水平線法 |
|---|---|---|---|
| A | 反復回数 | 231 (94) | 163 (74) |
| | CPU | 4.18 (1.70) | 1.83 (.83) |
| B | 反復回数 | 230 (104) | 171 (79) |
| | CPU | 4.16 (1.88) | 1.92 (.89) |
| C | 反復回数 | 206 (97) | 140 (64) |
| | CPU | 3.72 (1.75) | 1.57 (.72) |

Table 7 に格子点数: 250×250 のときの収束までの反復回数と CPU 時間を示す。括弧内の数字は前処理として MILU 分解をしたときの結果である。収束までの CPU 時間はいずれの場合も半分以下 (41%~42%) になった。Table 5 と同様に、ここでも水平線法によるベクトル化によって、不完全分解に要する時間が大幅に (43%に) 減った。

また Table 8(a),(b) は格子点: 250×250 のときパラメータ $B$ を変化させたときの反復回数と CPU 時間 (秒単位) である。1 階微分項に掛かるパラメータ $B$ を大きくしても本分解法の方が効率的であることがわかる。

Table 8. Convergence property for some parameters $B$

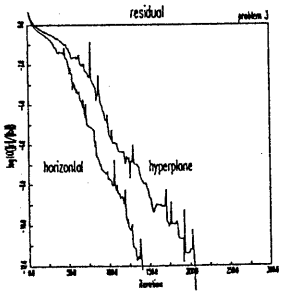
(a)Number of iterations

| | | $B$=2 | $B$=10 | $B$=100 |
|---|---|---|---|---|
| A | 超平面 | 231 (94) | 203 (96) | 112 (48) |
| | 水平線 | 163 (74) | 136 (69) | 83 (41) |
| B | 超平面 | 230 (104) | 243 (89) | 110 (46) |
| | 水平線 | 171 (74) | 152 (68) | 83 (42) |
| C | 超平面 | 206 (97) | 199 (93) | 111 (45) |
| | 水平線 | 140 (64) | 139 (65) | 89 (40) |

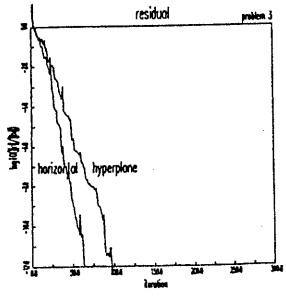
(b)CPU time

| | | $B$=2 | $B$=10 | $B$=100 |
|---|---|---|---|---|
| A | 超平面 | 4.18 (1.70) | 3.70 (1.73) | 2.03 (.880) |
| | 水平線 | 1.83 (.834) | 1.53 (.778) | .935 (.467) |
| B | 超平面 | 4.16 (1.88) | 4.44 (1.60) | 2.00 (.845) |
| | 水平線 | 1.92 (.836) | 1.71 (.766) | .936 (.477) |
| C | 超平面 | 3.72 (1.75) | 3.63 (1.68) | 2.02 (.832) |
| | 水平線 | 1.57 (.722) | 1.56 (.734) | 1.00 (.455) |

Fig.5.(a),(b) は (M)ILU/Bi-CG STAB 法の相対残差 $L_2$ ノルムの履歴を表す。




(a) ILU/Bi-CG STAB 法




(b)MILU/Bi-CG STAB 法

Fig. 5. Convergence history of problem C

## 5. まとめ

提案した5点あるいは7点差分スキームに対する不完全LU分解法は、従来の5点差分スキームに対してベクトル化を超平面法で行なった不完全 LU 分解法に比べて、(1) ベクトル化が容易、(2) 反復1回当たりの CPU 時間が短い、そして (3) 収束までの反復回数も少なくなる、などの優れた特長を有する。そのため、(M)ICCG 法あるいは (M)ILU/Bi-CG STAB 法で偏微分方程式を離散化した方程式を解いたとき、収束までの CPU 時間はかなり少なくて済む。ただ、本差分スキームは、従来の2次精度の5点差分式に比べて打切り誤差が少し大きい公式に基づいて導出されているため、得られた解が要求した精度をもつかどうか確認する必要がある。それを見極めた上で使用すれば、本方法はベクトル計算機上で有用であると思われる。本差分スキームの非等方性格子に対する性能評価並びに3次元問題への拡張は今後の課題である。

## 参考文献


[1] Fujino S., Mori M., Takeuchi T., Performance of hyperplane ordering on vector computers, J. of Computational and Applied mathematics, Vol.38, **23**(1991), 125-136.

[2] 藤野, 長谷川, ベクトル計算機における Fill-in 付き (M)ICCG 法の性能評価, 日本応用数理学会論文誌, Vol.2, **2** (1992), 105-118.

[3] 藤野, 竹内, ベクトル計算機におけるメモリ競合の数理的考察, 日本応用数理学会 論文誌, Vol.3, **2**(1993), 59-72.

[4] Gustafsson I., A class of first order factorization method, BIT, Vol.18(1978), 142-156.

[5] Meijerink J.A., Van der Vorst H.A., An Iterative Solution Method for Linear Systems of which the Coefficient Matrix is a Symmetric M-matrix, Math. of Comp., Vol.31, **137**(1977), 148-162.

[6] Pommerell C., Fichtner W., New developments in Iterative Methods for Device Simulation: *Simulation of Semiconductor Device and Process* Vol.4, edited by W. Fichtner, Zurich (1991), 243-248.

[7] 後 保範, ベクトル計算機向き ICCG 法, 京都大学数理解析研究所 講究録 No.514 (1984), 110-134.

[8] Van der Vorst H.A., Bi-CG STAB: A First and Smoothly Converging Variant of Bi-CG for the Nonsymmetric Linear Systems, SIAM J. Sci. Stat. Comput., Vol.13 (1992), 631-644.


**[APPENDIX]**

Fig. 6 に格子点数:31×31 で Dirichlet 条件のとき、行列 A と本差分スキームの行列 $A_h$ における各固有値 $\lambda_i$ の最小固有値 $\lambda_{min}$ に対する比をプロットしたものを示す。条件数は 364 と 182 である。

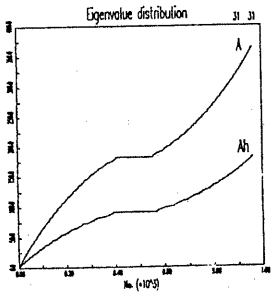



Fig. 6. Plot of ratios: $\lambda_i/\lambda_{min}$