数理解析研究所講究録
第746巻 1991年 29-36

三次元圧縮性流体解析に現れるブロック五重対角行列群について

張 紹 良†

1 はじめに

近年におけるスーパーコンピュータのめざましい発達によって、流体運動を記述するオイラー、ナビエ・ストークス等の方程式系の数値計算を非常に高速に行うことが可能になってきた。しかし、スーパーコンピュータのベクトル演算機能を十分に生かすためには、今まで以上に数値計算技法にもとづくプログラミング技術が必要になってくる。

三次元領域において、ナビエ・ストークス方程式を空間微分に4精度の AF 法 (Approximate Factorization method) を用いて数値解析を行うとき、計算空間の3方向においてブロック五重対角行列群が現れる。ベクトル計算機を使う場合、これらの五重対角行列群を一つの方向に LU 分解し、他の方向には並列化し、それらを同時に解くことによって計算の効率化を図ることができる。さらに、筆者はこのようなブロック五重対角行列群に対して、両端より LU 分解を同時に行う方法を提案し、従来の LU 分解法と比較を行う。

2 基礎方程式と数値計算手法

三次元圧縮性ナビエ・ストークス方程式は、$Thin-layer$ 近似のもとで一般座標系を使い、次の(1)式、(2)式のように表される。

$$\partial_t \hat{Q} + \partial_\xi \hat{E} + \partial_\eta \hat{F} + \partial_\zeta \hat{G} = R_e^{-1} \partial_\zeta \hat{S} \tag{1}$$

$$\hat{Q} = J^{-1}(\rho, \rho u, \rho v, \rho w, e)^T \tag{2}$$

ただし、$\hat{E}$、$\hat{F}$、$\hat{G}$ は対応する ξ、η、ζ 方向のフラックス、$\partial_\zeta \hat{S}$ は ζ-微分だけの粘性項を表す。Jは座標変換のヤコビアン行列である。

また、圧力は理想気体のエネルギ関係式より、

$$p = (\gamma - 1)(e - \rho \frac{u^2 + v^2 + w^2}{2}) \tag{3}$$

となる。

これらの方程式を $Implicit$ 解法で解く場合、よく用いられている計算手法として AF法がある。

デルタ形式

$$\Delta \hat{Q}^n = \hat{Q}^{n+1} - \hat{Q}^n \tag{4}$$

を用いると、AF法による差分式は以下の通りとなる。

$$[I + h\delta_\xi \hat{A}^n][I + h\delta_\eta \hat{B}^n][I + h\delta_\zeta \hat{C}^n - hR_e^{-1}\delta_\zeta J^{-1}\hat{M}^n]\Delta \hat{Q}^n = -h[\delta_\xi \hat{E}^n + \delta_\eta \hat{F}^n + \delta_\zeta \hat{G} - R_e^{-1}\delta_\zeta \hat{S}^n] \tag{5}$$

†計算流体力学研究所　〒152 目黒区原町1-22-3　電話 03-3711-0454

ここで、$h = \Delta t$ であり、$\hat{A} = \frac{\partial \hat{E}}{\partial \hat{Q}}$、$\hat{B} = \frac{\partial \hat{F}}{\partial \hat{Q}}$、$\hat{C} = \frac{\partial \hat{G}}{\partial \hat{Q}}$、$\hat{M} = \frac{\partial \hat{S}}{\partial \hat{Q}}$ で、流速ヤコビアン行列である。

3 通常の LU 分解法

空間微分に4次精度の AF 法を用いたとき、その数値計算手法は次のブロック五重対角行列を係数行列にする連立1次方程式を解くことに帰着される。

$$\hat{A}_{i,j} = \begin{bmatrix} C_1^{i,j} & D_1^{i,j} & E_1^{i,j} & & & & & \\ B_2^{i,j} & C_2^{i,j} & D_2^{i,j} & E_2^{i,j} & & & & \\ A_3^{i,j} & B_3^{i,j} & C_3^{i,j} & D_3^{i,j} & E_3^{i,j} & & & \\ & \cdots & \cdots & \cdots & \cdots & \cdots & & \\ & & \cdots & \cdots & \cdots & \cdots & \cdots & \\ & & & & A_{l-1}^{i,j} & B_{l-1}^{i,j} & C_{l-1}^{i,j} & D_{l-1}^{i,j} \\ & & & & & A_l^{i,j} & B_l^{i,j} & C_l^{i,j} \end{bmatrix} \tag{6}$$

ただし、$A_k^{i,j}$, $B_k^{i,j}$, $C_k^{i,j}$, $D_k^{i,j}$, $E_k^{i,j}$ $(i = 1 \sim m;\ j = 1 \sim n;\ k = 1 \sim l)$ はそれぞれ 5×5 小行列である。また、l, m, n は計算面での分割数とする。

ベクトル計算機を用いる場合、通常の LU 分解法では、上の (6) 式の $m \times n$ 個の $\hat{A}_{i,j}$ 行列に対して、以下のように同時に $m \times n$ 個を LU 分解

$$\begin{bmatrix} C_1^{i,j} & & & & & & \\ B_2^{i,j} & \bar{C}_2^{i,j} & & & & & \\ A_3^{i,j} & \bar{B}_3^{i,j} & \bar{C}_3^{i,j} & & & & \\ & \cdots & \cdots & \cdots & & & \\ & & \cdots & \cdots & \cdots & & \\ & & & A_{l-1}^{i,j} & \bar{B}_{l-1}^{i,j} & \bar{C}_{l-1}^{i,j} & \\ & & & & A_l^{i} & \bar{B}_l^{i} & \bar{C}_l^{i,j} \end{bmatrix} \begin{bmatrix} I & \bar{D}_1^{i,j} & \bar{E}_1^{i,j} & & & & \\ & I & \bar{D}_2^{i,j} & \bar{E}_2^{i,j} & & & \\ & & I & \bar{D}_3^{i,j} & \bar{E}_3^{i,j} & & \\ & & & \cdots & \cdots & \cdots & \\ & & & & \cdots & \cdots & \cdots \\ & & & & & I & \bar{D}_{l-1}^{i,j} \\ & & & & & & I \end{bmatrix} \tag{7}$$

することができる。I は単位行列を表す。すなわち、このようなブロック五重対角行列の解を求めるのに、上下ブロック三重対角行列の積の形に分解し、前進消去・後退代入を行って解を求める。

4 $Twisted\ LU$ 分解法

筆者は、ベクトル化効率をさらに上げるために、この LU 分解法の計算を行列の対角の両端より同時に行うことを考えた。この場合、前進消去・後退代入は共に対角の中央で折り返して行われる（l が偶数か奇数かで、中央の部分の処理が少し違う）。つまり、行列 $\hat{A}_l^{i,j}$ に対して $Twisted\ LU$（交差型 LU）分解法 [1] を行う。文献 [1] では、$Twisted\ LU$ 分解法は、CG 法等のような反復解法の前処理の並列化のため使用され、各小三重行列（大きさは分割数に依存する）を対象にして行うものである。このアイデアを筆者は直接法（LU 分解法）に応用した。計算の概略を下の図1 (a), (b) に示す。ここで ⟶ 印は計算手順を表す。

$$
\left|\begin{array}{cccccccccc}
C_1^{i,j} & & & & & & & & & \\
B_2^{i,j} & \bar{C}_2^{i,j} & & & & & & & & \\
A_2^{i,j} & \bar{B}_2^{i,j} & \bar{C}_2^{i,j} & & & & & & & \\
 & \cdots & \cdots & \cdots & & & & & & \\
 & & A_{\frac{l}{2}+1}^{i,j} & \bar{B}_{\frac{l}{2}+1}^{i,j} & \bar{C}_{\frac{l}{2}+1}^{i,j} & \bar{D}_{\frac{l}{2}+1}^{i,j} & E_{\frac{l}{2}+1}^{i,j} & & & \\
 & & & & & \cdots & \cdots & \cdots & & \\
 & & & & & & \bar{C}_{l-1}^{i,j} & \bar{D}_{l-1}^{i,j} & E_{l-1}^{i,j} & \\
 & & & & & & & \bar{C}_l^{i,j} & D_l^{i,j} & \\
 & & & & & & & & C_l^{i,j} &
\end{array}\right|
$$

図1 (a)　前進消去時の計算手順の概略

$$
\left|\begin{array}{cccccccc}
I & \bar{D}_1^{i,j} & \bar{E}_1^{i,j} & & & & & \\
 & I & \bar{D}_2^{i,j} & \bar{E}_2^{i,j} & & & & \\
 & & \cdots & \cdots & \cdots & & & \\
 & 0 & 0 & I & 0 & 0 & & \\
 & & \cdots & \cdots & \cdots & & & \\
 & & & \bar{A}_{l-1}^{i,j} & \bar{B}_{l-1}^{i,j} & I & & \\
 & & & & \bar{A}_l^{i,j} & \bar{B}_l^{i,j} & I &
\end{array}\right|
$$

図1 (b)　後退代入時の計算手順の概略

通常の LU分解法に対して、(7) 式の中の $\bar{B}_k^{i,j}$, $\bar{C}_k^{i,j}$, $\bar{D}_k^{i,j}$, $\bar{E}_k^{i,j}$ の計算の流れは次のようになる。

$$
\begin{array}{ccccccccccccc}
 & & & & A_3^{i,j} & & \cdots & & A_{l-2}^{i,j} & & A_{l-1}^{i,j} & & A_l^{i,j} \\
 & & & & \downarrow & & \cdots & & \downarrow & & \downarrow & & \downarrow \\
 & & B_2^{i,j} & & \bar{B}_3^{i,j} & & \cdots & & \bar{B}_{l-2}^{i,j} & & \bar{B}_{l-1}^{i,j} & & \bar{B}_l^{i,j} \\
 & & \downarrow & \nearrow & \downarrow & \nearrow & \cdots & \nearrow & \downarrow & \nearrow & \downarrow & \nearrow & \downarrow \\
C_1^{i,j} & \rightarrow & \bar{C}_2^{i,j} & \rightarrow & \bar{C}_3^{i,j} & \rightarrow & \cdots & \rightarrow & \bar{C}_{l-2}^{i,j} & \rightarrow & \bar{C}_{l-1}^{i,j} & \rightarrow & \bar{C}_l^{i,j} \\
\downarrow & \nearrow & \downarrow & \nearrow & \downarrow & \nearrow & \cdots & \nearrow & \downarrow & \nearrow & \downarrow & \nearrow & \\
\bar{D}_1^{i,j} & & \bar{D}_2^{i,j} & & \bar{D}_3^{i,j} & & \cdots & & \bar{D}_{l-2}^{i,j} & & \bar{D}_{l-1}^{i,j} & & \\
\downarrow & \nearrow & \downarrow & \nearrow & \downarrow & \nearrow & \cdots & \nearrow & \downarrow & \nearrow & & & \\
\bar{E}_1^{i,j} & & \bar{E}_2^{i,j} & & \bar{E}_3^{i,j} & & \cdots & & \bar{E}_{l-2}^{i,j} & & & &
\end{array}
$$

図2　通常の LU 分解法の計算の流れ

図2 に示したように、添字 k に対して、 $\bar{B}_k^{i,j}$, $\bar{C}_k^{i,j}$, $\bar{D}_k^{i,j}$, $\bar{E}_k^{i,j}$ の計算は直前の二つの計算結果 $\bar{B}_{k-1}^{i,j}$, $\bar{C}_{k-1}^{i,j}$, $\bar{D}_{k-1}^{i,j}$, $\bar{E}_{k-1}^{i,j}$ および $\bar{B}_{k-2}^{i,j}$, $\bar{C}_{k-2}^{i,j}$, $\bar{D}_{k-2}^{i,j}$, $\bar{E}_{k-2}^{i,j}$ に依存する。

一方、本論文の中の *Twisted LU* 分解法の中に現れる（図1 (a), (b) 参照）$\bar{A}_k^{i,j}$, $\bar{B}_k^{i,j}$, $\bar{C}_k^{i,j}$, $\bar{D}_k^{i,j}$, $\bar{E}_k^{i,j}$ の計算の流れは図3のようになる。

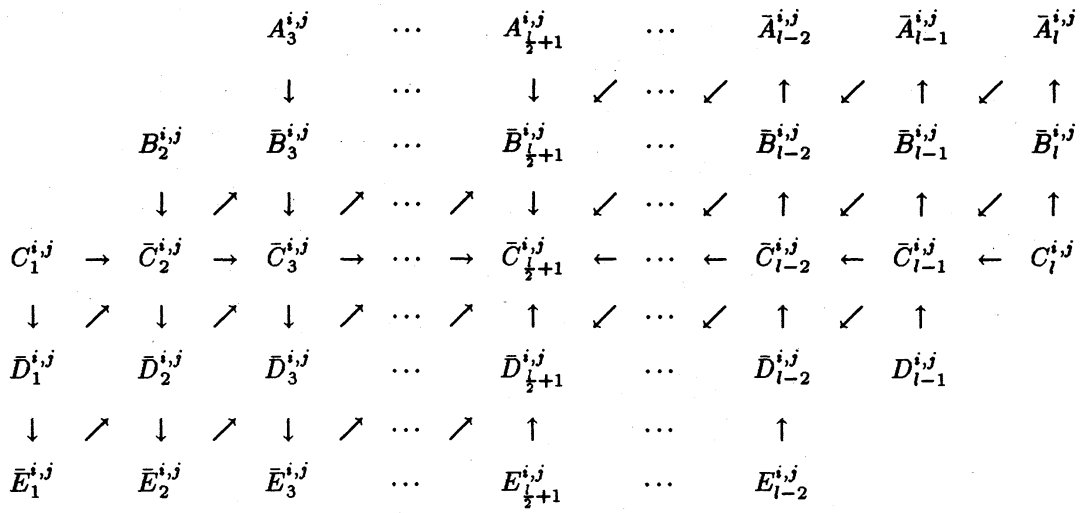

図3　*Twisted LU* 分解法の計算の流れ

添字 k に対して $\frac{l}{2}-1$ までの $\bar{B}_k^{i,j}$, $\bar{C}_k^{i,j}$, $\bar{D}_k^{i,j}$, $\bar{E}_k^{i,j}$ の計算は、直前の二つの計算結果 $\bar{B}_{k-1}^{i,j}$, $\bar{C}_{k-1}^{i,j}$, $\bar{D}_{k-1}^{i,j}$, $\bar{E}_{k-1}^{i,j}$ および $\bar{B}_{k-2}^{i,j}$, $\bar{C}_{k-2}^{i,j}$, $\bar{D}_{k-2}^{i,j}$, $\bar{E}_{k-2}^{i,j}$ に依存し、$\frac{l}{2}+2$ からの $\bar{A}_k^{i,j}$, $\bar{B}_k^{i,j}$, $\bar{C}_k^{i,j}$, $\bar{D}_k^{i,j}$ の計算は、後ろの二つの計算結果 $\bar{A}_{k+1}^{i,j}$, $\bar{B}_{k+1}^{i,j}$, $\bar{C}_{k+1}^{i,j}$, $\bar{D}_{k+1}^{i,j}$ および $\bar{A}_{k+2}^{i,j}$, $\bar{B}_{k+2}^{i,j}$, $\bar{C}_{k+2}^{i,j}$, $\bar{D}_{k+2}^{i,j}$ に依存する。また、$\bar{C}_k^{i,j}$ と $\bar{C}_{l-k+1}^{i,j}$ の計算は同じ形式で表せる（□印は 5×5 小行列、$\square^{-1}$ はその逆行列を意味する）。

$$\bar{C}_k^{i,j} = \square - \square\square - \square\square = \bar{C}_{l-k+1}^{i,j} \tag{8}$$

同様に $\bar{D}_k^{i,j}$ と $\bar{B}_{l-k+1}^{i,j}$ も同じ形式 (9) で、

$$\bar{D}_k^{i,j} = \square^{-1}(\square - \square\square), \quad \bar{B}_{l-k+1}^{i,j} = \square^{-1}(\square - \square\square) \tag{9}$$

$\bar{B}_k^{i,j}$ と $\bar{D}_{l-k+1}^{i,j}$ も (10) 式で、

$$\bar{B}_k^{i,j} = \square - \square\square, \quad \bar{D}_{l-k+1}^{i,j} = \square - \square\square \tag{10}$$

$\bar{E}_k^{i,j}$ と $\bar{A}_{l-k+1}^{i,j}$ も (11) 式で、

$$\bar{E}_k^{i,j} = \square^{-1}\square, \quad \bar{A}_{l-k+1}^{i,j} = \square^{-1}\square \tag{11}$$

それぞれ表せるので、両端より同時に$\bar{C}_k^{i,j}$ と $\bar{C}_{l-k+1}^{i,j}$、$\bar{D}_k^{i,j}$ と $\bar{B}_{l-k+1}^{i,j}$、$\bar{B}_k^{i,j}$ と $\bar{D}_{l-k+1}^{i,j}$、及び$\bar{E}_k^{i,j}$

と $\bar{A}^{i,j}_{l-k+1}$、を計算することが可能である。

通常の *LU* 分解法と本論文の *Twisted LU* 分解法の違いを、前進消去部分を例にとり、実際のプログラミングに近い形で図4, 5 に説明する（ $f^{i,j}_k$ は連立1次方程式の右辺項を表す)。

$$
\begin{aligned}
&DO \quad 100 \quad k = 1, \quad l \\
&DO \quad 100 \quad i = 1, \quad m \\
&DO \quad 100 \quad j = 1, \quad n \\
&\quad \bar{B}^{i,j}_k = B^{i,j}_k - A^{i,j}_k \times \bar{D}^{i,j}_{k-2} \\
&\quad \bar{C}^{i,j}_k = C^{i,j}_k - A^{i,j}_k \times \bar{E}^{i,j}_{k-2} - \bar{B}^{i,j}_k \times \bar{D}^{i,j}_{k-1} \\
&\quad \bar{D}^{i,j}_k = (\bar{C}^{i,j}_k)^{-1} \times (D^{i,j}_k - \bar{B}^{i,j}_k \times \bar{E}^{i,j}_{k-1}) \\
&\quad \bar{E}^{i,j}_k = (\bar{C}^{i,j}_k)^{-1} \times E^{i,j}_k \\
&\quad \bar{f}^{i,j}_k = (\bar{C}^{i,j}_k)^{-1} \times (f^{i,j}_k - A^{i,j}_k \times \bar{f}^{i,j}_{k-2} - \bar{B}^{i,j}_k \times \bar{f}^{i,j}_{k-1}) \\
100 \quad &CONTINUE
\end{aligned}
$$

図4　通常の *LU* 分解法

$$
\begin{aligned}
&DO \quad 100 \quad k = 1, \quad \mathbf{\frac{1}{2} \times l} \\
&DO \quad 100 \quad i = 1, \quad m \\
&DO \quad 100 \quad j = 1, \quad \mathbf{2 \times n} \\
&\quad \bar{B}^{i,j}_k = B^{i,j}_k - A^{i,j}_k \times \bar{D}^{i,j}_{k-2} \\
&\quad \bar{C}^{i,j}_k = C^{i,j}_k - A^{i,j}_k \times \bar{E}^{i,j}_{k-2} - \bar{B}^{i,j}_k \times \bar{D}^{i,j}_{k-1} \\
&\quad \bar{D}^{i,j}_k = (\bar{C}^{i,j}_k)^{-1} \times (D^{i,j}_k - \bar{B}^{i,j}_k \times \bar{E}^{i,j}_{k-1}) \\
&\quad \bar{E}^{i,j}_k = (\bar{C}^{i,j}_k)^{-1} \times E^{i,j}_k \\
&\quad \bar{f}^{i,j}_k = (\bar{C}^{i,j}_k)^{-1} \times (f^{i,j}_k - A^{i,j}_k \times \bar{f}^{i,j}_{k-2} - \bar{B}^{i,j}_k \times \bar{f}^{i,j}_{k-1}) \\
100 \quad &CONTINUE
\end{aligned}
$$

図5　*Twisted LU* 分解法

図5 に示したように、内側 *DO* ループ長が長くなるため、より速い計算速度を得られると期待できる。後退代入部分も同様に速く計算できる。

5 数値実験

以下の数値実験はすべて富士通 $VP-200$ (最大性能 570 $Mflops$) での結果である。

5.1 テスト問題 1

計算面での各方向の分割数 $l=61, m=39, n=40$ のテスト問題に対して、数値実験を行った。その結果を下の表 1 に示し、図 6 にそのときの 3 重ループの構成図を示す。

```
      DO  200  k = 1,  l
      DO  200  i = 1,  m
      DO  200  j = 1,  n
      ..................
200   CONTINUE
```

図 6　3 重ループの構成図

	CPU 時間	$Mflops$
$Twisted\ LU$分解	619 $msec$	330
通常の LU分解	1033 $msec$	194

表 1　3 重ループの両解法の CPU 時間と $Mflops$

この問題では CPU 時間は約 60 % に減少した。

5.2 テスト問題 2

外側のループ (kのループ) はそのままで、内側の 2 重ループ (i と j のループ) に対してベクトル長を長くするために 1 重のループ (p のループ) に手直しをした。同じ分割数 (p のカウントは $m \times n$ 個) のテスト問題に対して、数値実験の結果を下の表 2 に示し、図 7 にそのときの 2 重ループの構成図を示す。

```
      DO  200  k = 1,  l
      DO  200  p = 1,  m × n
      ......................
200   CONTINUE
```

図 7　2 重ループの構成図

	CPU時間	$Mflops$
$Twisted\ LU$分解	524 $msec$	390
通常の LU分解	590 $msec$	340

表2　2重ループの両解法 CPU 時間と $Mflops$

3重ループ (k, i, j) の内側の 2 重ループ (i, j) を 1 重ループ (p のループ) にしたとき、二つの方法ともそれぞれ 85 %、57 % に計算効率が上がった。

5.3　テスト問題 3

ここでは、2次元問題を取り上げる。すなわち、分割数を $l = 481$, $m = 1$ に固定し、n をいろいろ変えた。プログラムはテスト問題 2 で使用したものと同じである。図 8 にそのときの 2 重ループの構成図を示す。

```
      DO   200   k = 1,   l
      DO   200   j = 1,   n
      .....................
200   CONTINUE
```

図8　2 重ループの構成図

	$l = 481, n = 39$	$l = 481, n = 139$	$l = 481, n = 201$
$Twisted\ LU$分解	207 $msec$ (198)	619 $msec$ (237)	868 $msec$ (244)
通常の LU分解	303 $msec$ (133)	781 $msec$ (185)	1048 $msec$ (199)

表3　l を固定したときの両解法の CPU 時間 ($Mflops$)

$Twisted\ LU$ 分解法と通常の LU 分解法の効率の差は行列の大きさにより異なることが表 3 の結果から分かる。サイズが大きければ大きいほど二つの方法の効率はともに上がる。しかし、ベクトル長を段々大きくしても効率は通常飽和してくるので、通常の LU 分解法のベクトル長より長い $Twisted\ LU$ 分解法の利点は、段々と目立たなくなる。また $Twisted\ LU$ 分解法は、作業領域が通常の LU分解法より多くいるので、メモリの大きさが問題となるケースがある。

6 まとめ

両端より同時に分解していく、この *Twisted LU* 分解法は、通常の *LU* 分解法よりベクトル計算機向きアルゴリズムである。また、*Twisted LU* 分解と *LU* 分解法は直接法であるので、テスト問題に限らず、流体解析等で現れる同規模のブロック五重対角行列に対しても同じ効率化が期待できる。

7 謝辞

本研究を行う際、様々な助言をして下さった計算流体力学研究所　藤野清次主任研究員に感謝致します。

8 参考文献

[1] H. A. Van der Vorst: Large tridiagonal and block tridiagonal linear systems on vector and parallel computers, Parallel Comput. 5(1987)45-54