論　文

並列処理システムによる線形計画計算と実対称行列の三重対角化計算*

金　田　悠紀夫**

Abstract

A parallel processor system which is organized with several minicomputers is expected to be practical use in the near future.

The parallel computing of Linear Programming problems and tridiagonalization of the symmetric matrices are considered to be typical applications of it. So, we analized the parallel computing methods of LP algorithm (the revised simplex method together with the product form of the inverse of current basis) and Dr. Murata's tridiagonalization algorithm of the symmetric banded matrices.

1. まえがき

高速演算機能を有した小形コンピュータや大容量高速の LSI メモリの出現にともない，これら小形コンピュータ多数と大容量メモリユニットを組合せた並列処理システムがいくつか提案されてきている[5]．

代表的なシステム構成として **Fig. 1** に示すような大容量主記憶を多数の演算プロセッサが共有しているものがある．

アプリケーションとしては様々なものが提案されているが，有力なものに大規模システムの解析に出てくる大形行列計算がある．代表的なものとして大形連立一次方程式や偏微分方程式の数値計算，大形線形計画計算，実対称行列（特に帯行列）の固有値計算をあげることができる．

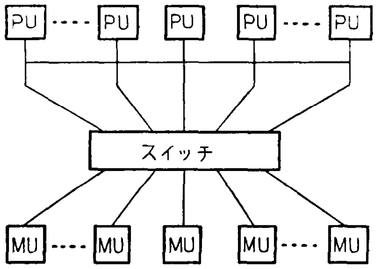


Fig. 1 Example of a parallel computing system.

* Parallel computing techniques of Linear Programming and symmetric matrix tridiagonalization by Yukio KANEDA (Faculty of Engineering, Kobe University)

** 神戸大学工学部システム工学科

前二者の並列計算については文献 10) ですでに述べた．ここでは特にそのアルゴリズムが複雑な後の二者について，計算時間を最も要する線形計画計算の繰り返し計算部 (BTRAN, PRICE, FTRAN, CHUZR の各計算) と実対称行列の固有値計算における三重対角化計算の部分を取り上げアルゴリズムの解析と若干のシミュレーションを行うことにより並列計算の可能性を調べ，いずれの場合もその並列計算アルゴリズムを適切に選ぶと十分幅の広い並列計算を効率よく実行できることを示した．

2. 線形計画計算の並列計算法

2.1 線形計画計算のアルゴリズム

線形計画法は数理計画法の主柱をなすもので，解法のアルゴリズムとしては，シンプレックス法を基本とし逆行列の積形式による表現法を用いた場合が多い．

線形制約および目的関数を

$$\boldsymbol{Ax}=\boldsymbol{b} \qquad (1)$$

$$x_0=\boldsymbol{C}^T\boldsymbol{x} \qquad (2)$$

とすると，積行列法による改訂シンプレックス計算は以下の繰り返し計算を進めることになる．

1. BTRAN 計算

$$\boldsymbol{\pi}^T=\boldsymbol{c}_B{}^T B^{-1}=\boldsymbol{c}_B{}^T E_t\cdots E_2E_1 \qquad (3)$$

2. PRICE 計算

$$d_j=\boldsymbol{\pi}^T\boldsymbol{a}_j-c_j\ (j\notin B) \qquad (4)$$

3. FTRAN 計算

PRICE 計算で選んだ列 $\boldsymbol{a}_s$ に対して,

$$\boldsymbol{\alpha}_s=B^{-1}\boldsymbol{a}_s=E_t\cdots E_2E_1\boldsymbol{a}_s \qquad (5)$$

4. CHUZR 計算

$\boldsymbol{\alpha}_s$ 中の第 l 要素を α_{ls} とすると,

$$\frac{\beta_r}{\alpha_{rs}}=\min_{\alpha_{ls}>0}\frac{\beta_l}{\alpha_{ls}} \qquad (6)$$

なる r と s を求める. ($\boldsymbol{\beta}=B^{-1}\boldsymbol{b}$)

5. WRETA 計算

E_{t+1} を求め, 新 $\boldsymbol{\beta}=E_{t+1}\boldsymbol{\beta}$ とし $\boldsymbol{c}_B$ を更進する.

ただしここで,

A : 線形制約式の係数行列で m 行 n 列
B : 基底行列
$\boldsymbol{\pi}$: 評価ベクトル
$\boldsymbol{c}$: 利益係数ベクトル
$\boldsymbol{c}_B$: 基底利益ベクトル
E_k : 基底変換行列
d_j : リデューストコスト
$\boldsymbol{a}_j$: A の第 j 列
$\boldsymbol{\beta}$: 基底解ベクトル

とする.

2.2 並列計算法

2.1 の 1～4 までの各計算に含まれている並列計算の可能性とその計算量について検討する。

(1) BTRAN 計算

$\boldsymbol{x}^T=[x_1,\cdots,x_m]$ なるベクトル $\boldsymbol{x}^T$ とピボット位置 r の基底変換行列 E との積を求めると,

$$\boldsymbol{x}^T E=\left[x_1,\cdots,x_{r-1},\sum_{i=1}^m x_i\eta_i,x_{r+1},\cdots,x_m\right] \qquad (7)$$

となり, 第 r 要素のみ変更を受ける.

$\sum_{i=1}^m x_i\eta_i$ は演算ユニット数に制限がないとすると, Hellerman 等[5),8)]のアルゴリズムにより, **Fig. 2** のように i レベル (i は $i\geq\log_2 m>i-1$ なる整数) に分けられ各レベルに属する計算は並行して実行できる.

いま, PU の台数を p 台に限ると,

$$\sum_{i=1}^m x_i\eta_i=\sum_{i=1}^{m_1}x_i\eta_i+\sum_{i=m_1+1}^{m_2}x_i\eta_i+\cdots,$$
$$\sum_{i=m_{p-1}}^{m}x_i\eta_i \qquad (8)$$

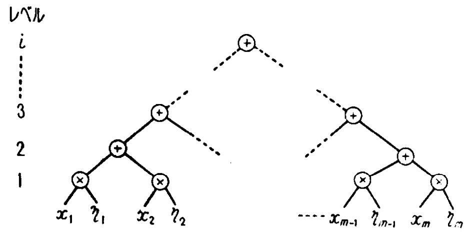


Fig. 2 Execution tree of $\sum_{i=1}^m x_i\eta_i$.

($m_1,\cdots,m_{p-1}$ は $0<m_1<m_2<\cdots<m_{p-1}<m$ なる整数)

のように p グループに分割して並列計算を行い最後に合計計算を行う方式も考えられる.

しかし η ベクトル中の非零要素数が極めて少ない場合には単一 PU で全計算を行った方がよい場合が多い.

次に積計算 $\boldsymbol{c}_B{}^T E_t E_{t-1},\cdots,E_2E_1$ レベルでの並列計算について検討する.

$$((\cdots((\boldsymbol{c}_B{}^T E_t)E_{t-1})\cdots E_2)E_1 \qquad (9)$$

のように括弧で区切って内側から計算を進めていく.

E_j のピボット列を $[\eta_{1,rj},\cdots,\eta_{m,rj}]^T$ とすると, (9)式は初期設定 $\boldsymbol{x}=\boldsymbol{c}_B{}^T$ を行った後

$$x_{rt}=\sum_{i=1}^m x_i\eta_{i,rt} \qquad (10\text{-}1)$$
$$\vdots$$
$$x_{rj}=\sum_{i=1}^m x_i\eta_{i,rj} \qquad (10\text{-}(t-j+1))$$
$$\vdots$$
$$x_{r1}=\sum_{i=1}^m x_i\eta_{i,r1} \qquad (10\text{-}t)$$

の諸式を順に計算していくことになる.

計算式(10-1)から(10-t)の計算項を **Fig. 3** のようにならべかえて考える. x_{rj} の計算式に含まれる変数 x_1～x_m の内で x_{rt}～x_{rj-1} に含まれないものからなる項はもとの位置におき, 含まれているものは x_m の右側で対応する x_{ri} の位置にその項を移動させもとの位置は 0 としている.

Fig. 3 の x_{rt} より左部に属する各項の係数, 変数は (9)式の計算に入る前から決っているので, x_{rt}～x_{r1} に対する計算式で左部に属する各項の計算は並列に実

$$\begin{array}{l}
\qquad\quad x_1 \qquad\quad x_2 \quad \cdots \quad x_m \qquad\quad x_{rt} \quad \cdots \quad x_r \quad \cdots \quad jx_{r1-1}\\
x_{rt}=x_1\eta_{1,rt}+x_2\eta_{2,rt}+\cdots+x_m\eta_{m,rt}\\
\vdots\\
x_{rj}=x_1\eta_{1,rj}+x_2\eta_{2,rj}+\cdots+x_m\eta_{m,rj}+x_{rt}\eta_{rt,rj}+\cdots+x_{rj-1}\eta_{rj-1,rj}\\
\vdots\\
x_{r1}=x_1\eta_{1,r1}+x_2\eta_{2,r1}+\cdots+x_m\eta_{m,r1}+\cdots\cdots\cdots\cdots+x_{r1-1}\eta_{r1-1,r1}
\end{array}$$

Fig. 3 Equations of BTRAN computation.

行でき，各式の並列計算は(8)式で示したものと同一方式で行える．右側に属する部分の計算は列単位で並列に実行でき，x_{rj} の値が定まると，列 x_{rj} に属する項に対する計算が $x_{rj-1}\sim x_{r1}$ に対して並列に行うことができる．

実際の計算においては，B^{-1} の再逆転計算によって求められたスパース性の高い $E_{\tau-1}\sim E_1$ に対する積計算と再逆転後生成された非零要素の比較的多い $E_t\sim E_\tau$ に対する計算を分けて考える．$\boldsymbol{c}_B{}^T E_t, \cdots, E_\tau$ までの積計算は E_t から順に E_τ まで逐次行っていくとし，E_i ごとに(8)式で示した並列計算を行う．$\boldsymbol{x}$ を密ベクトルとすると E 当りの計算量は η ベクトル中の非零要素数 $n\eta$ で決まり $n\eta$ 回の積計算と $n\eta-1$ 回の和計算となる．

$E_{\tau-1}, \cdots, E_1$ の積計算に関しては Fig. 3 のようにならべかえx_mの右辺の行要素が全て0の x_i をレベル1のグループとし，続いてレベル1の全 x_i に対応する列要素を0とし，この操作によって右辺の行要素が全て0となったものをレベル2とする．この操作を引続き行い，全 x_i のレベル分けを行う．各レベルに属する x_i の計算は互に独立で並行して実行できる．

計算量を決めるのは，レベル数 $\boldsymbol{l}$，レベル当りの平均変換行列数 $\overline{ne}$，平均の非零 η 要素数 $\overline{n\eta}$ 等である．

(2) PRICE 計算

d_j の計算は一部または全部の非基底列に対して

$$d_j=\boldsymbol{\pi}^T\boldsymbol{a}_j-c_j \tag{11}$$

を計算することで，$\boldsymbol{\pi}^T=[\pi_1, \cdots, \pi_m]$，$\boldsymbol{a}_j{}^T=[a_{1j}, \cdots, a_{mj}]$ とおくと

$$d_j=\sum_{i=1}^{m}\pi_i a_{ij}-c_j \tag{12}$$

となる。

d_j の計算自体は(8)式と同一の手法を用いてp台のPUで並列計算することができる．しかし一般には $\boldsymbol{a}_j$ 中の非零要素は極めて希薄であるのが普通で計算量も d_j 当りでは少ないので，プロセッサ間通信のオーバヘッドを考えると単一パスとして一台のPUで計算する方が得策であると考えられる．

全体の計算量としては $\boldsymbol{\pi}^T$ を密ベクトルと仮定すると $(n-m)$ 個の d_j 計算を行うことになり，各 d_j 計算は $\boldsymbol{a}_j$ 中の非零要素数を na_j とすると na_j 回の積計算と (na_j-1) 回の和計算を行うことになる．

$(n-m)$ は極めて大きな値となるのが普通であり各 d_j は独立に並行して計算できるので，各PUにバランスよく割当ることにより並列プロセッサが極めて有効に働く計算であるといえる．

今p台のPUによるq列を対象としたマルチプライシングアルゴリズムを採用するとすると，$(n-m)$ 個の d_j 群から q 個の d_j を選ぶ操作になるが，各PUが $(n-m)/p$ 個の要素から小さい順に q 個の要素を一組選び最後に一台のPUがp組に含まれる全要素から小さい順にq個選び出すサーチ操作によって実現することができる．

(3) FTRAN 計算

列 $\boldsymbol{a}_s$ に対する積計算

$$\boldsymbol{\alpha}_s=B^{-1}\boldsymbol{a}_s=E_t, \cdots, E_2E_1\boldsymbol{a}_s \tag{3}$$

の並列計算について考えてみる．

ピボット位置rの変換行列 E と，列ベクトル$\boldsymbol{x}$との積を示すと，

$$E\boldsymbol{x}=\begin{bmatrix}1 & & \eta_1 & & \\ & \ddots & \vdots & & \\ & & 1 & & \\ & & \eta_r & & \\ & & & 1 & \\ & & \vdots & & \ddots \\ & & \eta_m & & & 1\end{bmatrix}\begin{bmatrix}x_1\\ \vdots\\ x_{r-1}\\ x_r\\ x_{r+1}\\ \vdots\\ x_m\end{bmatrix}=\begin{bmatrix}x_1\\ \vdots\\ x_{r-1}\\ 0\\ x_{r+1}\\ \vdots\\ x_m\end{bmatrix}+x_r\begin{bmatrix}\eta_1\\ \vdots\\ \eta_{r-1}\\ \eta_r\\ \eta_{r+1}\\ \vdots\\ \eta_m\end{bmatrix} \tag{14}$$

となる．

ベクトル$\boldsymbol{x}$にベクトル $\boldsymbol{\eta}$ と x_r の積を加える計算で各 $x_i+x_r\eta_i$ $(i=1\sim m)$ は独立に計算可能で，η ベクトル中の非零要素数が多い場合はp台のPUにバランスよくηベクトルの要素を割当て並行計算を行うことができる．しかし三角化法による再逆転で生成されるηベクトルは極めて疎となっている場合が多く，この場合はPU間通信のオーバヘッド等を考えると単一パスとして実行した方がよい場合が多くなる．

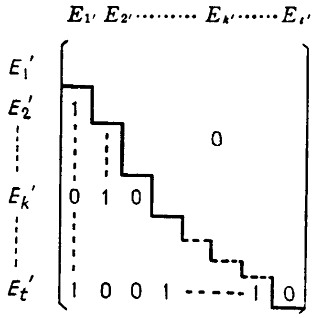
$E_i\cdots E_2E_1\boldsymbol{a}_q$ レベルでの並列計算を考える。$E_t(\cdots(E_2(E_1\boldsymbol{a}_q))\cdots)$ のように括弧で区切って内側から計算を進めるというアルゴリズムをとるとし，$\boldsymbol{x}_0=\boldsymbol{a}_q$，$\boldsymbol{x}_i=E_i\boldsymbol{x}_{i-1}$ $(i=1\sim t)$ とすると計算は順次 $\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_t$ を求めていくことになる．

E_i に対応する η ベクトルを $\boldsymbol{\eta}_{ri}$ とすると $\boldsymbol{x}_i$ は $\boldsymbol{x}_{i-1}$ と $\boldsymbol{\eta}_{ri}$ によって定まることになり，η ベクトルがスパース性を持たなければ，このレベルでの並列計算は困難であるが，スパース性を考慮すると並列計算の可能性がでてくる．

かりに $\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_t$ 中の非零要素の位置が事前に判明しているとすると，$\boldsymbol{x}_{i-1}$ の r_i 要素が0となっている場合，$E_i\boldsymbol{x}_{i-1}$ の計算は不要となる．この不要部分を取除いた計算式を $E_{t'}, \cdots, E_{1'}\boldsymbol{x}_0$ とおきこのレベ

ルでの並列計算法を考える.

変換行列 $E_{1'}, E_{2'}, E_{t'}$ を下図のように並べる.



(14)式において x_r を E の入力と呼び $\eta_i \neq 0$ なる i に対応する x_i を E の出力と呼ぶと, 列 $E_{k'}$ において, $E_{k'}$ の出力で $E_{k'+1} \sim E_{t'}$ までの入力となっているものを選び対応する $E_{k'}$ 列要素を 1 とし他の要素を 0 とする. この操作を $k=1'$ から $t'-1$ まで順次行っていくことにより行列を作る.

行 $E_{1'}, E_{2'}, \cdots, E_{t'}$ において行要素が全て 0 の行は互いに独立で対応する変換行列に対する(14)式で示した計算は並行して行うことができる. また $E_{k'}$ に対する計算が終了すると $E_{k'}$ に対応する列要素は全て 0 となるから新たに行要素が全て 0 となる変換行列があらわれた場合それに対する(14)式の計算は直ちに開始でき他の計算と並行して行える.

$x_0, x_1, \cdots, x_t$ 中の非零要素の位置は一般には不明なのでこの場合は全ての要素が非零として上述の操作を行うことになる.

なおマルチプライシングアルゴリズムを採用した場合には PRICE 計算で選ばれた複数列に対して FTRAN 計算が行われる. 各列に対する FTRAN 計算は互に独立であるから, 並列に計算できる.

BTRAN 計算において場合分けしたのと同様に, E_i が再逆転計算によって生成されたものか以後の繰り返し計算によって生成されたものか分けて考える.

前者の η ベクトルは非零要素は極めて少なく BTRAN 計算と同様の手順を上述の行列にほどこしてレベル分けし, 各レベルに属する E_i に対する計算は並行して行える. 計算量を決めるのはレベル数 l, レベル内の平均変換行列数 $\overline{ne}$, 平均非零 η ベクトル要素数 $\overline{n\eta}$ それに(14)式の計算で $x_r \neq 0$ となる確率 h できまる. レベルごとの平均計算量は $\overline{n\eta} \cdot \overline{ne} \cdot h$ 回の積計算と和計算となる. p 台の PU が分担して並列計算することになる.

後者の計算は FTRAN 計算の後半に出てくるもので列ベクトル x 中の非零要素数が極めて多くなっている場合が多いこと, η ベクトル中の非零要素数も比較的多いことが前者と異なる.

計算量は x を密ベクトルと見なすと変換行列あたり $n\eta$ 回の和計算と積計算を行うことになる. $n\eta$ が大きい場合は p 台の PU が分担して(14)式の計算を並列に実行することも現実的と考えられる.

(4) CHUZR 計算

CHUZR 計算は

$$\min \frac{\beta_l}{\alpha_{ls}} \tag{15}$$

となる l と s をサーチすることになる.

p 台 ($p \ll m$) の PU で並列サーチを行う場合には各 PU にバランスして要素を割当て, 各 PU は並行して割当られた要素から最小のものをサーチし最後に各 PU がサーチした最小の中からさらに最小のものを一台の PU がサーチするという方式が考えられる.

この CHUZR 計算は m の数が大きいと極めて大きな並列パスを作ることになり, 各 PU は効率よく働くことになる.

2.3 実例データをもとにした計算量の推定

実際の大形線形計画計算において, 前述した各計算の計算量を決めるパラメータがどのような値をとるかは大変興味がある. 文献3) に示されている例題を用いてこれらパラメータの推定を行う. 例題は 3 つあり各問題の特性および計算途中で行われた再逆転計算で生成された η ベクトル中の非零要素数, 50 回の繰り返し計算によって生成された η ベクトル中の非零要素数が示されている (**Table 1**).

いま以下に述べる仮定を行うことにする.

(i) システムの構造を示す初期データとして与えられる (不等号) 制約式中の非零要素は各行, 各列にわたってランダムに分布している.

(ii) 基底行列 B に含まれる列は, A の中からランダムに選んだ m 列から構成されているとする.

(iii) 再逆転によって生成された m 個の変換行列中

Table 1 Structure of the LP problems.

問題		A	B	C
行数		822	2,978	3,496
構造変数の数		1,571	6,333	9,154
非零要素数		11,127	47,505	74,907
η ファイル中の非零要素数	再逆転直後	4,861	25,118	37,795
	50回繰返し後	26,443	83,021	115,729

$[m^2/n]$ 列は非零 η 要素は 1 であり，残りの列は平均 {全変換行列中の非零η要素数$-[m^2/n]$}/{$[m-[m^2/n]]$} 個でランダムに配置されているとする.

(iv) 再逆転後の繰り返し計算によって生成される変換行列中の非零要素数の平均は 50 回の繰り返し計算によって生成された新変換行列群の η ベクトルに含まれる非零要素数の 50 分の 1 に等しく各非零要素は各行にわたってランダムに配置されているとする.

各計算の計算量を示すパラメータである $m, n\eta$ の平均値 $\overline{n\eta}$ $(i=t\sim\tau, (\tau-1)\sim1)$ と n_{aj} の平均 $\bar{n}_{aj}$ は **Table 2** のようになる。

BTRAN, FTRAN 計算における $l, \overline{ne}, h$ 等は直接推定はできない．そこで以下に述べる手順でシミュレーションを行いその値を推定した.

サイズ m の一次元アレイと乱数を用いて平均 $\overline{n\eta}$ $(i=(\tau-1)\sim1)$ 個の非零要素がランダムに配置されたベクトル $[m(1-m/n)]$ 個と一個の非零要素のみを配置したベクトル $m-[m(1-m/n)]$ 個をランダムな順に発生さしてゆき，これをηベクトルと考え，**2.2** で示したレベル分けの操作をηベクトルに対し繰り返して行いレベル数 l の値を求めた．同時に $\boldsymbol{a}_j$ としてサイズ m で平均 $\bar{n}_{aj}$ 個の非零要素をランダムに配置したアレイを作成し，このベクトルを $\boldsymbol{x}$ として(14)式で示した繰り返し操作のシミュレートを行い $x_r \neq 0$ となる確率 $h(i=(\tau-1)\sim1)$ を求めた．引き続き平均 $\overline{n\eta}(i=t\sim\tau)$ 個の非零要素を含むηベクトルを順次発生させ(14)式で示した操作のシミュレーションを続行し，確率 $h(i=t\sim\tau)$ を求めた.

以上のシミュレーションを何回か繰り返し得られた値を平均したものが **Table 3** である.

Table 2 Estimated values of $m, \overline{n\eta(i)}$ and $\overline{n_{aj}}$

問　題	A	B	C
m	822	2,978	3,496
$\overline{n\eta}$ $(i=t\sim\tau)$	432	1,158	1,159
$\overline{n\eta}$ $(i=(\tau-1)\sim1)$	8.5	12.0	14.5
$\overline{n_{aj}}$	5.0	5.4	6.2

Table 3 The results of simple simulation

問　題	A	B	C
l	32	44	60
$\overline{ne}$	25.7	67.7	58.3
h $(i=t\sim\tau)$	0.14	0.16	0.37
h $(i=(\tau-1)\sim1)$	0.64	0.68	0.80

3. 実対称行列の並列三重対角化計算

大次元の実対称行列の固有値解析の有力な手法にハウスホルダ法によって三重対角化を行ってから固有値を求めるものがある．この三重対角化計算は多量の計算時間を要する代表的な計算であるが，ここではその並列計算法について検討する.

3.1 実対称行列（非帯行列）の並列三重対角化計算

3.1.1 三重対角化計算法

n 元の実対称行列を A とし第 j 列目まで三重対角化が進んだ行列を $A(j)$ とすると三重対角化計算は鏡像変換 $Q(i)$ を用いて,

$$A(j)=Q(j)A(j-1)Q(j) \tag{16}$$

の計算を $j=1$ から $n-2$ まで 1 ずつ増しながら行うことになる．$(A(0)=A$ と考える)

ただし $Q(j)$ は $A(j-1)$ の第 l 行 m 列要素を $a_{l,m}(j-1)$ とし $i=j+1$ とすると

$$Q(j)=\boldsymbol{I}-\boldsymbol{U}(j)\boldsymbol{U}^T(j)/h$$

$$\boldsymbol{U}^T(j)=(0,\cdots,0,a_{i,j}{}^{(j-1)}\pm s, a_{i+1,j}{}^{(j-1)},\cdots, a_{n,j}{}^{(j-1)})$$

$$s^2=\sum_{k=i}^{n}\{a_{k,j}{}^{(j-1)}\}^2$$

$$h=s^2+|a_{i,j}{}^{(j-1)}|\cdot s \tag{17}$$

実際の計算では変数ベクトル $\boldsymbol{p},\boldsymbol{q}$ と変数 k を導入することにより,

$$\boldsymbol{p}=A(j-1)\boldsymbol{u}(j)/h$$

$$k=\boldsymbol{u}^T(j)\ \boldsymbol{p}/2h$$

$$\boldsymbol{q}=\boldsymbol{p}-k\boldsymbol{u}(j)$$

$$A(j)=A(j-1)-\boldsymbol{u}(j)\boldsymbol{q}^T-\boldsymbol{q}\boldsymbol{u}^T(j) \tag{18}$$

なる計算を進めていくことになる.

3.1.2 並列計算法

三重対角化計算は各 j について上の諸式より $s^2, s, h, \boldsymbol{u}(j), \boldsymbol{p}, k, \boldsymbol{q}, A(j)$ の順に計算を進めていくことになるが，各計算は以後の計算に波及効果を持つのでこのレベルでの並列計算は不能である．したがって各計算内での並列計算の可能性を検討することになる.

s^2 の計算は **2.2** で示した $\sum_{n=i}^{m} x_i\eta_j$ と同一手法で並列計算を行うことが可能であるが，p 台の PU$(p\ll n)$ による並列計算となると(8)式と同様な形式で並列計算を行うことが現実的である.

$s, h, \boldsymbol{u}(j)$ の計算は逐時的に実行するとする.

$\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{u}(j)$ の第 i 要素を $p_i, q_i, u_i(j)$ とする.

$\boldsymbol{p}$ の計算は $i=j+1$ から n までの各 p_i について,

$$p_i=\left\{\sum_{k=j}^{n} a_{i,k}(j-1)\cdot u_k(j)\right\}\Big/ h \tag{19}$$

の計算を行うことになる．各 p_i は互いに独立で $(n-j)$ 個の要素の並列計算が可能となる．また各 p_i の計算も(8)式と同様に並列計算が可能である．

k の計算は

$$k=\left(\sum_{i=j}^{n} u_i(j)\cdot p_i\right)\Big/ 2h \tag{20}$$

で(8)式と同様の方式で並列計算可能である．

$\boldsymbol{q}$ の計算では各 q_i は

$$q_i=p_i-ku_i(j) \tag{21}$$

となり，$i=j+1$ から n までの各 i について並列に計算できる．

$A(j)$ の計算は $l=(j+1)\sim n, m=(j+1)\sim n$ の全 l, m に関して

$$a_{l,m}(j)=a_{l,m}(j-1)-u_l(j)q_m-q_lu_m(j) \tag{22}$$

を計算することになり，全 $a_{l,m}{}^{(j)}$ は互に独立であるから全要素を並列に計算できる．

特に計算量の多い $\boldsymbol{p}, A(j)$ の計算は極めて並列性が高いので，各 PU に分割して効率よく計算できる．

3.2 実対称帯行列の並列三重対角化計算

3.2.1 三重対角化計算法

帯行列に対してハウスホルダ法をそのまま適用すると計算途中で帯行列の性質が失われ，計算時間及びメモリ容量ともに急激に増大する．村田氏等は特殊な鏡像変換を繰り返していくことにより，この帯幅 ($2m+1$ とおく) の増大を抑えている．文献11) で用いている記号を用いると鏡像変換

$Q_1{}^1, Q_{1,1}{}^k, Q_{2,1}{}^k, \cdots, Q^k{}_{[(n-3)/m],1}$

………

$Q_l{}^1, Q_{1,l}{}^k, \cdots, Q^k{}_{[(n-l-2)/m],l}$

………

$Q^1{}_{n-2}$

を用いて順次三重対角化の操作を進めていくことになり，その計算は Q^1AQ^1 形の変換と引き続く一連の Q^kAQ^k 形の変換を行うという操作を繰り返し行うことにより進められる．

$A(j-1)$ に対する Q^1AQ^1 形の変換は $Q_j{}^1A(j-1)Q_j{}^1$ となり，(17) 式における n を $\min(j+m+1, n)$ とおくと(17), (18)式と同一計算式となる．

Q^kAQ^k 形の変換は $Q_{i,j}{}^kA_{i-1}(j-1)Q_{i,j}{}^k$ なる変換を $i=1\sim[(n-j-2)/m]$ まで順次行うことになる。

ここで $A_{i-1}(j-1)$ は $A(j-1)$ に対して $Q_j{}^1, Q_{1,j}{}^k, \cdots, Q_{i-1,j}{}^k$ までの変換を行ったものである．

便宜上 $A_{i-1}(j)=\{a_{fg}\}$, $Q_{i,j}{}^k=\{q_{fg}\}$, $i\cdot m+(j+1)=r$, $\min(r+m-1, n)=r_1$, $\min(r+2m-1, n)=r_2$ とおく．$Q_{i,j}{}^kA_{i-1}(j)Q_{i,j}{}^k$ は Q_1, A_1, B_1, B_2 を

$$Q_1=\begin{bmatrix} q_{r,r} & \cdots & q_{r,r1} \\ \vdots & & \\ q_{r1,r} & \cdots & q_{n,r1}\end{bmatrix} \qquad A_1=\begin{bmatrix} a_{r,r} & \cdots & a_{r,r1} \\ \vdots & & \vdots \\ a_{r1,r} & \cdots & a_{r1,r1}\end{bmatrix}$$

$$B_1=\begin{bmatrix} a_{r,r-m-1} & \cdots & a_{r,r-1} \\ & & \\ a_{r1,r-m-1} & \cdots & a_{r1,r-1}\end{bmatrix} \quad B_2=\begin{bmatrix} a_{r1+1,r} & \cdots & a_{r1+1,r1} \\ \vdots & & \vdots \\ a_{r2,r} & \cdots\cdots & a_{r2,r1}\end{bmatrix} \tag{23}$$

とおくと3つの部分 $Q_1B_1, Q_1A_1Q_1, B_2Q_1$ に分けることができる．($r_1=n$ のときは B_2 は存在しない)

$i=r, j=r-m, n=r_1$ とすると $Q_1A_1Q_1$ は(17), (18)式の計算により求まり，Q_1B_1, B_2Q_1 の計算は，

$$B_2Q_1=B_2(\boldsymbol{I}-\boldsymbol{U}\boldsymbol{U}^T/h)=B_2-(B_2\boldsymbol{U}/h)\boldsymbol{U}^T$$

$$Q_1B_1=(\boldsymbol{I}-\boldsymbol{U}\boldsymbol{U}^T/h)B_1=B_1-\boldsymbol{U}(\boldsymbol{U}^TB_1/h) \tag{24}$$

となる．

3.2.2 並列計算法

$Q_j{}^1A(j-1)Q_j{}^1$ および $Q_1A_1Q_1$ の計算は **3.1.2** で述べたと同一方式で並列計算できる．**3.1.2** と異なるのは(19)式から(22)式までの計算は $n-(j+1)$ までの幅を持っていたが，本計算では幅がほぼ m となり $n\gg m$ とすると，並列の幅は小さいことになる．

$Q_1A_1Q_1, B_2Q_1, Q_1B_1$ は互に独立であるため全て並行して計算できる．

$\boldsymbol{v1}=B_2\boldsymbol{U}/h, \boldsymbol{v2}=\boldsymbol{U}^TB_1/h, \boldsymbol{v3}=B_2Q_1, \boldsymbol{v4}=Q_1B_1$ とおくと，$\boldsymbol{v1}, \boldsymbol{v2}$ の各要素 $v1_i, v2_j$，引続く $\boldsymbol{v3}, \boldsymbol{v4}$ の各要素 $v3_i, v4_j$ の計算は i, j に関して互に独立で全要素を並行して計算できる．

とくに $v1_i, v2_j$ は

$$v1_i=\sum_{j=r_{1+1}}^{r_2} a_{ij}u_j$$

$$v2_j=\sum_{i=r}^{r_1} u_ia_{ij} \tag{25}$$

であり各 $v1_i, v2_l$ に対しても **2.2** で示したように並列計算が可能である．

次に **3.2.1** で示した変換シーケンスを以下のように二次元にならべかえる．

$Q_1{}^1Q_{11}{}^kQ_{21}{}^kQ_{31}{}^k\cdots\cdots\cdots Q^k{}_{[(n-3)/m],1}$

$Q_2{}^1Q_{12}{}^k\cdots\cdots$

………

$Q_l{}^1Q_{1l}{}^k\cdots\cdots Q^k{}_{[(n-l-2)/m],l}$

………

$Q^1{}_{(n-2)}$

このようにならべかえてみると任意の Q の変換計算が行えるためには，Q の存在する位置を i 行 j 列と

すると 1～i 行，1～$j-1$ 列内に存在する全 Q による変換計算が終了していることが必要十分条件である．

この条件を許す範囲内で変換計算の並列化が可能で例えば，同一列内の全 Q の変換を並列に計算するというアルゴリズムをとれば，三重対角化計算は第一列から順に $Q^1{}_{(n-2)}$ の列まで1列ごとに計算を進めていくことになる．このレベルでの並列計算の幅を考えると，1個から最大 $[([(n-3)/m]+1)/3]+1$ 個まで Q についての変換が並列計算できる．n を一定とすると m が大きいほどこのレベルでの並列性が下がるが，この場合は各 Q ごとの変換において大きな並列性を持つことができるので並列性の低下にはつながらない．

むしろ特徴は m が小さいとき，Q 内で十分とれない並列性が，このレベルでの並列性でカバーできることで，村田氏等のアルゴリズムは十分高い並列性を持っていることが判る．

4. 結　論

2. および **3.** において大形線形計画計算，大形実対称行列（特に帯行列）の三重対角化計算の並列計算法を示した．これら計算に用いたアルゴリズムはいずれも現在実用されているもので効率がよいとの評価を受けているものである．

アルゴリズムはいずれもいくつかのレベルに分かれた並列性を有しており，これら並列性を有効に生かせば極めて効率よく並列計算ができることが判明した．

謝　辞　本研究を進めるに当って日頃から御鞭達いただいている電子技術総合研究所電子計算機部長黒川一夫博士，人間機械システム研究室渡辺定久室長に深謝します．

参 考 文 献

1) W. Orchard-Hays: Advanced Linear-Programming Computing Techniques, McGraw-Hill Book Company.
2) J. Buchet: How to Take Into Account the Low Density of Matrices to Design a Mathematical Programming Package. Relevant Effects on Optimization and Inversion Algorithms, Large Spase Sets of Linear Equations, Academic Press, New York, (1971)
3) J. A. Tomlin: Modifing Triangular Factors of The Basis in The Simplex Method, Sparse Matrices and their Applications, Plenum Press, New York-London, (1972)
4) W. W. White: A status report on Computing Algorithms for Mathematical Programming, Computing Surveys, Vol. 5, No. 3 (September 1973)
5) 金田悠紀夫: 並列処理システムの動向，電子技術総合研究所調査報告，第 174 号（昭和 48 年 2 月）
6) C. G. Bell, A. Newell: CMMP: The CMP multiprocessor computer requirements and overview of the initial design, CMU-CS-72-112
7) 三輪 修，乾 範男，内田啓一郎: FACOM 230-75 アレイプロセッサ，昭和 49 年度情報処理学会第 15 回大会
8) H. Hellerman: Parallel Processing of Algebraic expressions. IEEE, Trans. on EC, Vol. 15 (Feburary 1966)
9) 村田健郎: 最近の大型計算機の進歩とベンチマークの問題，京大数理解析研究所研究集会「数値解析とコンピュータ」(1974 年 10 月)
10) 金田悠紀夫: 並列処理システムによる連立一次方程式と楕円形偏微分方程式の数値計算法，情報処理，Vol. 16, No. 2（昭和 50 年 2 月）
11) 村田健郎，堀越清視: 対称帯行列を三重対角化するための新アルゴリズム，情報処理，Vol. 16, No. 2（昭和 50 年 2 月）

（昭和 50 年 5 月 13 日受付）
（昭和 51 年 12 月 4 日再受付）