

# スーパーコンピュータにおけるリストベクトルの 利用技術†

小国 力<sup>††</sup> 後保 範<sup>††</sup>  
長堀 文子<sup>††</sup> 加藤 真一<sup>†††</sup>

スーパーコンピュータがもつ特長の一つにリストベクトル機能がある。これまで、リストベクトルは大規模疎行列を係数にもつ連立一次方程式において非ゼロ要素を処理するために用いられてきたが、スーパーコンピュータの場合にはこのほかにもいろいろな応用を考えることができる。特に、帯幅の小さな帯行列処理や高速フーリエ変換などにも威力を発揮することがわかっている。本論文ではスーパーコンピュータにおけるリストベクトルの使用法について、スカラコンピュータの場合と比較しながらマトリックス計算を中心に述べる。

## 1. はじめに

スーパーコンピュータがもつハードウェア上の特長としては<sup>1),2)</sup>,

- (1) 大規模主記憶装置
- (2) リストベクトル ( $A(L/I)$ ) といった使い方を  
するとき、配列  $L$  のことをいう)
- (3) ベクトル演算器とその並行処理

がある。このうち、本論文の主題であるリストベクトルは疎行列の処理および DO ループの多重度を下げてコンピュータの処理性能を向上するのに使用できる。

疎行列の処理としては、行列の  $LU$  分解のときに最もよくリストベクトルを利用できる。特に最近多用されている前処理つき反復解法において目ざましく、ICCG 法や ILUBCG 法の急速な普及の主要因になっている。また、回路解析などの非対称疎行列に対する直接解法においてもそれなりの効果を上げている。

リストベクトルは、新しい使い方として DO ループの多重度を下げるために利用できる。DO ループは FORTRAN オブジェクトを作るときにかなりのソフトウェア的な負荷を生じ、短いループ長のときには影響が大きい。このため、多重 DO ループのときには、リストベクトルを使って DO ループの多重度を少なくするとともに、最内側の DO ループ長を大きくすると性能が上がる。このとき、ハードウェアの欠点であるバンクコンフリクトを避けるようにリストベクトルを

作ることが必要である。

## 2. 疎行列用直接解法

疎行列を係数にもつ連立一次方程式を直接解法により解くときには、係数行列の各列の非ゼロ要素を行番号とともに 1 次元配列に詰めて貯える必要がある。したがって、これらの要素をスーパーコンピュータ上で演算するには間接番地付け方式の配列参照、つまりリストベクトルによる配列参照用のハードウェア命令が必要である。国産のスーパーコンピュータはリストベクトルによる配列参照用のベクトル命令をもっているので、疎行列処理に威力を発揮している。

ガウス法における部分軸選択を例にとると、リストベクトル方式はそうでない標準の方式に比べ、スカラプロセッサ上で 1.6 倍、ベクトルプロセッサ上で 2.0 倍の処理時間を要している。したがって、メモリに余裕があるときや、アルゴリズムを手直してリストベクトルを使わないですむときは、当然のことながらリストベクトルによる配列参照をやめるべきである。たとえば、帯行列に近い場合は、無理してリストベクトルを使うよりは、多少帯幅が広くても帯行列用ガウス法を使った方がよい。さらに、各回の消去計算結果について、ある閾値より絶対値が小さい要素を 0 とみなすようにすると、この判定による負荷のため疎行列用ガウス法はいっそう不利になる。

### 2.1 非対称疎行列用ガウス法

非対称疎行列  $A$  を  $LU$  分解するには、線形計画法でいう積形式によるガウス法を使う。つまり、行列  $A$  と分解結果をともに列方式で貯える。このとき、もとの行列  $A$  はそのままにして、分解後の行列  $L$  と  $U$  を  $L-\eta$  ベクトルと  $U-\eta$  ベクトルとよぶ形で  $A$  とは

† Usage Techniques for List Vectors of Supercomputers by TSUTOMU OGUNI, YASUNORI USHIRO, FUMIKO NAGAHORI (Software Works, Hitachi, Ltd.) and SHINICHI KATO (Hitachi Computer Consultant, Ltd.).

†† (株)日立製作所ソフトウェア工場

††† 日立コンピュータコンサルタント(株)

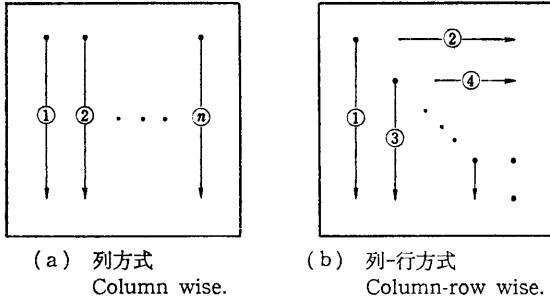


図 1 行列の番地付け  
Fig. 1 Addressing of a matrix.

別に貯えることが必要である。つまり、

$$A = L_1 L_2 \dots L_n U_n \dots U_2 U_1$$

と分解したとき、 $\eta$  ベクトル用の配列には順に  $L_1, U_1, L_2, U_2, \dots, L_n, U_n$  の枢軸列を入れる。

疎行列  $A$  を部分軸選択しながら  $LU$  分解して各  $L_i$  と  $U_i$  を作ると、 $i$  が大きくなるにつれて fill-in が急速に増える。このため、 $LU$  分解中の fill-in をシミュレートして、あらかじめ行列  $A$  の行や列をうまく入れかえて fill-in の発生が少なくなるようにしておくといよい。この場合、軸要素としては対角要素を選ぶ。

さて、図 1 の (b) に示すようなデータ配置をとってみよう。このように配置すると、リストベクトルをうまく使うことができる。つまり、分解結果  $L$  と  $U$  に対し、下三角行列  $L$  ではリストベクトルに行番号をしまい、上三角行列  $U$  ではこのリストベクトルに列番号をしまう。さらに、もう一つのリストベクトルを用いて合計二つのリストベクトルを用いると、まだ分解の済んでいない部分の消去計算の DO ループを 1 多重度だけ下げることができる。この場合、バンクコンフリクトを防ぐために、扱う行列  $A$  はかなり疎でなければならない。

### 2.2 対称疎行列用ガウス法

対称疎行列を  $LDL^T$  分解するときも積形式による対称行列用ガウス法を使う。このときも、もとの行列  $A$  はそのままにして、分解後の行列  $L$  または  $L^T$  を  $\eta$  ベクトルとよぶ形で  $A$  とは別に貯える必要がある。つまり、

$$A = LDU_n \dots U_2 U_1$$

と分解し、対角行列  $D$  と、上三角行列  $U_1, U_2, \dots, U_n$  を  $\eta$  ベクトルとして貯える。

非対称行列の場合と同じように、 $LDL^T$  分解時の fill-in が少なくなるように前処理を実施するが、帯行列用ガウス法やスカイライン法に比較して極端に性能

が悪くなる。

### 3. 疎行列用反復解法

偏微分方程式を差分法や有限要素法で離散化すると対称疎行列か非対称疎行列を係数とする連立一次方程式が得られる。対称疎行列を係数とする反復解法では ICCG 法 (Incomplete Cholesky Conjugate Gradient Method) が古典的な SOR 法や CG 法より収束が格段に良いことが知られてきた。一方非対称疎行列を係数とする反復解法では ILUCR 法 (Incomplete LU decomposition Conjugate Residual Method) および ILUBCG 法 (Incomplete LU Decomposition Bi-Conjugate Gradient Method) が注目されてきている。各解法ともリストベクトルを使用した同じ工夫でスカラ用解法と反復回数当りの収束を同一にしたままでベクトル化できる。ここでは差分法などで離散化して発生する規則的非対称疎行列に対する ILUCR 法と有限要素法などで離散化して発生する不規則的対称疎行列に対する ICCG 法のベクトル化について述べる。

#### 3.1 規則的非対称行列での ILUCR 法のベクトル化

ILUCR 法は方向ベクトル  $P_i$  を元の行列  $A$  に対して、 $k$  個前までの  $P_{i-k}$  と  $A^T A$  直交させる方法であるがここでは  $k=1$  の場合の計算手順を図 2 に示す。CR 法の前処理としての不完全三角分解には種々の方法があるが、ここでは  $LDU$  の形に分解し、下三角行列  $L$  および上三角行列  $U$  は対角要素を除き元の行列  $A$  の下三角行列および上三角行列と同一になる

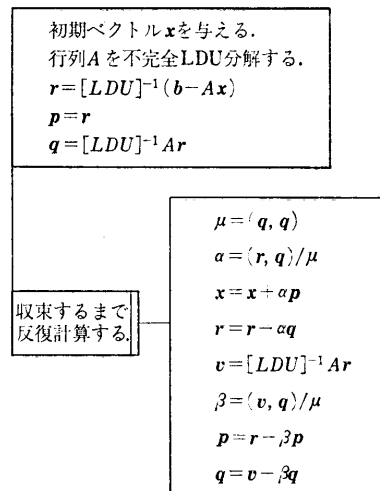


図 2 ILUCR 法の計算手順  
Fig. 2 Flow of ILUCR method.

ものとする。このとき  $L$  と  $U$  の対角要素は対角行列  $D$  の要素の逆数とする。

CR 法そのものはベクトル計算機向き解法であるが不完全三角分解と組み合わせた ILUCR 法はそのままでは次の二つの項目がベクトル計算には不適となる。

- (a) 行列  $A$  の不完全三角分解 ( $LDU$  と表す)。
- (b) 不完全三角分解を使用した前進および後退代入 ( $q=[LDU]^{-1}r$  と表す)。

三次元の移流拡散方程式を差分法で離散化した行列においては上記二つの計算は図3に示すデータの依存関係がある。これは  $i, j, k$  のどの添字を最内側の DO ループにしても、一つ前で計算した値を使用するというデータの依存関係がありベクトル計算に不適となる。ここで  $nx, ny, nz$  はそれぞれ  $x, y, z$  方向の分割数で、 $n$  は行列の次元数で  $n=nx \times ny \times nz$  とする。

データの依存関係を解消する対策として各添字  $i, j, k$  を加えた値、すなわち  $i+j+k$  の値が一定となる組合せのものをまとめて計算する方法がある。 $i+j+k$  の値が一定となるものをまとめて計算するにはリストベクトルが有効である。この方法を採用すると次のことがわかる。

(a)  $q(i, j, k)$  を計算するための  $q(i-1, j, k)$ ,  $q(i, j-1, k)$  および  $q(i, j, k-1)$  はすでに計算済みの値となりデータの依存関係がなくなる。このためベクトル計算に適する。

(b)  $q(i-1, j, k)$ ,  $q(i, j-1, k)$  および  $q(i, j, k-1)$  の値の計算が完了した後  $q(i, j, k)$  を計算するという関係は保持しているため反復回数当りの収束速度は変化しない。

図4に ILUCR 法および ILUBCG 法に共通なベクトル計算向き  $q=[LDU]^{-1}r$  の計算方法を示す。このとき使用するリストを図5に示す。これは三次元問題で  $nx=ny=nz=3$  とした場合の具体例であり、 $n=27, m1=3, m2=9$  となる。本工夫により ICCG 法, ILUCR 法および ILUBCG 法は全面的にベクトル化できる。ベクトル計算機 S-810 の汎用計算機 M-280 H に対する本プログラムの向上効果は次のとおりである。

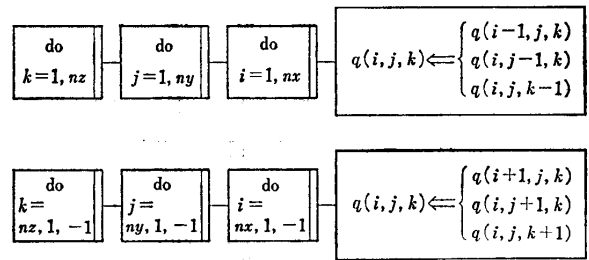


図3 不完全三角分解と  $q=[LDU]^{-1}r$  におけるデータの依存関係

Fig. 3 Data dependency for incomplete  $LDU$  decomposition and  $q=[LDU]^{-1}r$ .

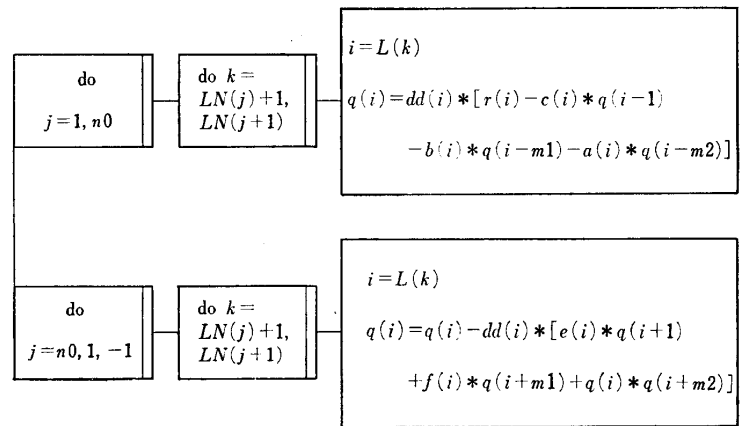


図4 ベクトル計算機向き  $q=[LDU]^{-1}r$  の計算

Fig. 4 Adaptive computation for  $q=[LDU]^{-1}r$  with vector computers.

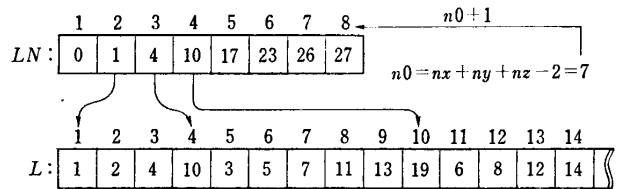
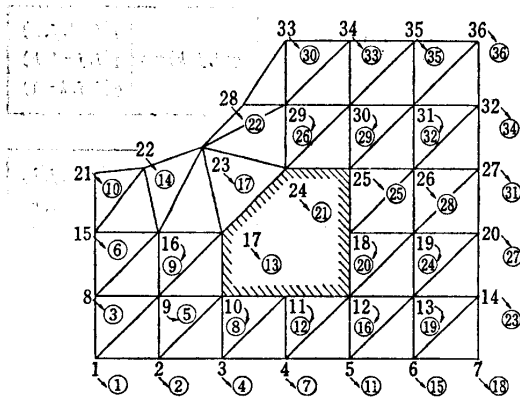


図5  $q=[LDU]^{-1}r$  の計算に使用するリストの例  
Fig. 5 Example of list vector for  $q=[LDU]^{-1}r$ .

- (a) ICCG 法は  $30 \times 30 \times 30$  分割で約 50 倍
- (b) ILUCR 法は  $40 \times 20 \times 20$  分割で約 50 倍
- (c) ILUBCG 法は  $40 \times 20 \times 20$  分割で約 40 倍

3.2 不規則的対称行列での ICCG 法のベクトル化  
汎用計算機向き ICCG 法と収束率が同一なベクトル計算機向き計算方法を示す。この計算では行列  $A$  の不完全コレスキー分解 ( $LDL^T$  で表す) と分解後の行列を使用した  $q=[LDL^T]^{-1}r$  のベクトル化が問題となる。また行列  $A$  は 1 行当りの非ゼロ要素が一般には少ないためベクトル長を長くする工夫がある。これはリストを付けた疎行列の記憶方法に少し工夫をすれば



注) 1, 2, 3...オリジナル番号付け  
①, ②, ③...ベクトル計算機向き番号付け

図 6 有限要素分割と節点の番号付け  
Fig. 6 Finite element mesh and numbering nodes.

次元数  $n$  のベクトル長で計算できるためここでは説明を省略する。

不完全コレスキー分解と  $q=[LDL^T]^{-1}r$  のベクトル化に関する工夫は同一である。図 6 に不規則的疎行列を発生させる有限要素分割と節点の番号付けを示す。ベクトル計算機向きにするには節点の適切な番号付けが必要である。オリジナル番号付けからベクトル計算機向き番号付けへの変換は、プログラムで次の条件のもとに実行する。

- (a) 直接結合する節点の番号の大小関係はオリジナルの番号付けとベクトル計算機向きの番号で一致させる。
- (b) 連続する番号ができるだけ直接結合しないようにベクトル計算機向き番号を付ける。

(a) は ICCG 法の収束率を変化させないための条件であり、(b) はベクトル長をできるだけ長くするための条件である。この条件を満足するように自動作成した疎行列の列番号テーブルを図 7 に示す。そのときの番号付けを図 6 に○印を付けた数字で示す。具体的な変換方法については「ベクトル計算機向き ICCG 法<sup>3)</sup>」を参照されたい。図 7 のブロック⑦を見ると、下三角テーブル  $LL$  は 11 から 17 の番号で構成されていることがわかる。すなわち、ブロック⑦の行番号に対応する 18 番から 22 番の未知数の計算はすでに計算が完了している 11 番から 17 番の値を使用して実行できる。またブロック⑦の上三角行列  $LU$  を見ると、23 から 30 の番号で構成されていることがわかる。これは後退代入計算においても未知数となる 23 番から 30 番の値はすでにブロック⑧および⑨で計算が完了していることがわかる。この関係を利用して、不規則的疎

4	7	8	12	16			
5	8	9	13	17	21		
6	9	10	⑤	14			
11			15	18	19	23	
7	11	12	⑥	16	19	20	24
9	13	14	17	21	22	26	
15			18	23			
11	15	16	⑦	19	23	24	27
16			20	24	25	28	
13	17		21	25	26	29	
17			22	26	30		
15	18	19	23	27			
16	19	20	⑧	24	27	28	31
20	21		25	28	29	32	
17	21	22	⑨	26	29	30	33
19	23	24	27	31			
20	24	25	28	31	32	34	
21	25	26	29	32	33	35	
22	26		30	33			
24	27	28	31	34			

下三角(LL)                      上三角(LU)

図 7 ベクトル計算機向き番号付け非ゼロ要素番号テーブル

Fig. 7 Adaptive nonzero elements numbering table for vector computer.

行列の不完全  $LDL^T$  分解と  $q=[LDL^T]^{-1}r$  の計算がベクトル化できる。ここで工夫した不規則的疎行列の ICCG 法を HITAC M-280 H (スカラで計算) と S-810 モデル 20 のベクトル計算で測定した。その結果反復回数当りの収束は同一であり、S-810 は M-280 H に比較して約 30 倍高速になった。測定は拡散方程式を有限要素法で離散化した次元数約 5,000 の連立一次方程式を使用した。ICCG 法の反復回数は 100 回で、行列の 1 行当りの非ゼロ要素数は平均約 7 である。

#### 4. 小規模密行列の処理

スーパーコンピュータはもともと長い配列に関する演算を高速に進めるために考え出されたものである。一つの配列演算を実施するとき、この演算のためのベクトル命令を実行する前にベクトルプロセッサの立ち上げが必要である。短い DO ループのときにはこの立ち上げに要する時間が相対的に大きくなり、処理性能の向上が抑えられる。このため、短ループ長のときには、DO ループの多重度を減らすことにより処理性能の劣化を防げばよく、そのためにはリストベクトルを使えばよい。このとき、ハードウェアのもつバンクコンフリクトの欠点を避ける必要がある。

最近の大型コンピュータの主記憶装置はある単位 (バンクとよぶ) に分かれていて、その単位に含まれるバイト数または倍長語数だけのメモリパスを設けて

ある。二つのバンク内の同一位置にあるバイトまたは倍長語は同一のメモリパスを通して主記憶装置からプロセッサ上にもってこられる。したがって、同一の主記憶装置番地またはバンクごとの同一相対番地からひんぱんにデータをもってくる、一つのパスに負担がかかりすぎ、処理性能が落ちる。この現象をバンクコンフリクトといっている。

4.1 行列の積

二つの  $n$  次元の行列の行列積を2次元配列のまま計算するときは、3重の DO ループを使い、それぞれの DO ループの長さは  $n$  である。  $n$  が小さくて、10か20のときには DO ループ処理の過負荷が相対的に大きくなって、ベクトルプロセッサを使っても性能が上がらない。そこで、2重の DO ループを1重化して DO ループ長を  $n^2$  にすれば、DO ループの回数も減るし、DO ループ長も大幅に増えて性能も上がる。そのために以下に述べるリストベクトル使用方法がある。

まず、コーディングを図8の(a)から(c)に示そう。図(a)は通常のコーディングである。図(b)は表1に示す2組のリストベクトル  $LI$  と  $LJ$  を使って DO ループの多重度を1減らしている。このとき、DO

```

DO 30 J=1, N
DO 20 I=1, N
S=0.0
DO 10 K=1, N
10 S=S+A(I, K)*B(K, J)
20 C(I, J)=S
30 CONTINUE
    
```

(a)

```

DO 30 L=1, N
DO 20 K=1, N*N
I=LI(K)
J=LJ(K)
20 C(I, J)=C(I, J)+A(I, L)*B(L, J)
30 CONTINUE
    
```

(b)

```

DO 30 L=1, N
NSN=N*(L-1)
NSI=L-1
DO 20 K=1, N*N
20 C(LL(K))=C(LL(K))+A(LI(K)+NSN)*B(LJ(K)+NSI)
30 CONTINUE
    
```

(c)

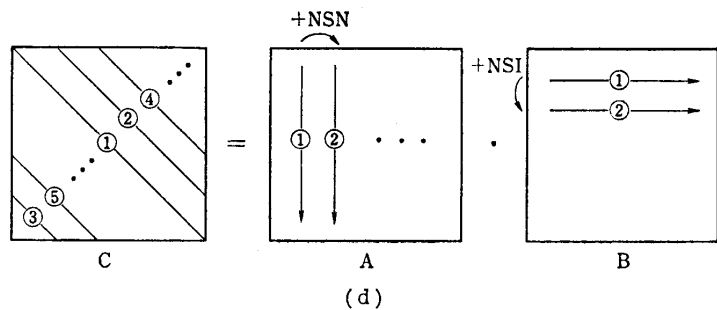


図8 リストベクトル利用の行列積アルゴリズム

Fig. 8 Algorithms of matrix product with list vectors.

ループの順序を入れかえている。図(c)は、リストベクトル  $LI$  および  $LJ$  を使って2次元配列  $A$  と  $B$  を1次元化し、図(d)の丸印の中の番号順に計算することにより性能の向上を図っている。

上述の三つのコーディングによる実測結果を表2に示す。  $n$  が10のとき、リストベクトル使用の著しい

表1 行列積用のリストベクトル  
Table 1 List vectors for matrix product.

$K$	1	2	3	...	$N$	$N+1$	$N+2$	...	$2N-1$	$2N$	$2N+1$	...	$N^2-1$	$N^2$
$LI$	1	2	3	...	$N$	1	2	...	$N-1$	$N$	1	...	$N-1$	$N$
$LJ$	1	2	3	...	$N$	2	3	...	$N$	1	3	...	$N-2$	$N-1$
$LL$	1	$N+2$	$2N+3$	...	$N^2$	$N+1$	$2N+2$	...	$N^2+N-1$	$N$	$2N+1$	...	$N^2-2N-1$	$N^2-N$

表2 行列積の性能結果  
Table 2 Results of performance on matrix products.

		単位 ms							
モード	方式	次数							
		5	10	20	40	60	80	100	120
スカラー	(a)	0.07	0.47	3.3	25.	82.	190.	370.	640.
	(b)	0.14	1.1	8.8	70.	240.	560.	1100.	1900.
	(c)	0.12	0.94	7.4	59.	200.	470.	930.	1700.
ベクトル	(a)	0.07	0.29	1.2	5.6	13.	31.	40.	69.
	(b)	0.03	0.09	0.44	3.0	8.6	26.	39.	73.
	(c)	0.02	0.06	0.35	3.0	8.9	25.	39.	75.

表3 リストベクトル使用のガウス法(密行列の場合)  
Table 3 Gauss method with list vector (on dense matrix).

(単位 ms)

モード	方式		20	30	40	50	60	70	80	100
	スカラー	標準		1.2	3.9	8.5	16.	27.	43.	64.
リスト			2.5	8.1	18.	36.	62.	99.	148.	290.
比			2.0	2.0	2.1	2.2	2.2	2.2	2.2	2.3
ベクトル	標準		0.83	1.8	2.9	4.5	6.4	8.9	15.	18.
	リスト		0.62	1.0	1.8	3.3	5.6	8.5	18.	25.
	比		0.74	0.55	0.64	0.72	0.87	0.95	1.2	1.3

効果がでることに気がつく。逆にスカラーコンピュータになると圧倒的に従来方式がよい。

#### 4.2 密行列のLU分解

小規模密行列のときも2次元配列Aを1次元配列化することによりDOループ長を大きくできる。このためには、消去計算の順序にデータを順序付ける必要がある。つまり、図1の(a)に示すFORTRANの列方式の番地付けから、(b)に示す消去順序への変換用リストベクトルを使えばよい。このとき、行列Aの(i,j)要素を(a)から(b)の順序へ変換する規則はつぎのようになる。

$$i+(j-1) \times N \rightarrow \begin{cases} (j-1)(2N-j)+i & (i \geq j) \\ i(2N-i)-N+j & (i < j) \end{cases}$$

LU分解にあたって列による部分軸選択は当然やってもよいが、nが小さいときの部分軸選択の負担は相対的に大きくなる。

nを変えたとき、通常的方式とリストベクトル方式のLU分解がどうなるかを示したのが表3である。この表からnが30程度のとき最もリストベクトル方式が有利となるが、nが80をこすと、従来の単純な方式のほうがよくなる。

### 5. 帯行列用直接解法

帯行列AをLU分解またはLDL<sup>T</sup>分解するとき、配列Aとしては行数が帯幅で、列数が行列の次数nであるような2次元配列を割り当てる。しかも、帯行列の分解の過程で参照または更新される範囲は一定の領域に限られる。このため、この領域を1次元配列として連続した番号付けを行い、もとの2次元配列の番号付けとの対応をリストベクトルの形で保てば、ベクトルプロセッサの性能を活かすことができる。

#### 5.1 非対称帯行列

上帯幅がp,下帯幅がqのn元帯行列AをLU分

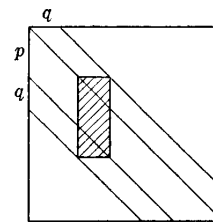
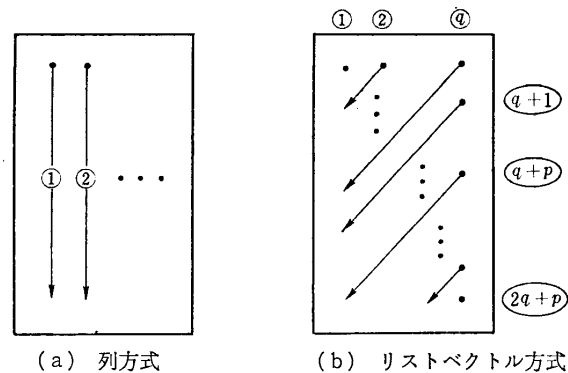


図9 帯行列と作業領域

Fig. 9 Band matrix and its working set.



(a) 列方式

(b) リストベクトル方式

図10 帯行列に対する番地付け

Fig. 10 Addressing for band matrix.

解するとき、部分軸選択をすれば最内側のDOループのループ長mはp+2qまたは2p+qとなる。したがって、この値が小さいときにはすでに述べたようにスーパーコンピュータの性能を十分に発揮できない。このため、リストベクトルを使って2次元配列Aを1次元配列扱いすることを考える。帯行列は図9に示すようにm×nの2次元配列となり、作業領域は斜線で囲む部分である。

LU分解の各段階で参照されるこの作業領域の大きさは各段階で変わらず、ただ対角線に沿って一段ずつ右下にずれていくにすぎない。したがって、この範囲をFORTRAN流の番地付けの代わりに、スーパーコンピ

表 4 帯行列用番地付けの実測値  
Table 4 Results for addressings on band matrices.  
ベクトルモード (単位 ms)

次数	方式	帯幅							
		10	12	14	16	18	20	26	30
200	列方式	4.8	5.4	6.3	6.9	7.5	8.3	—	—
	リスト方式	4.2	4.4	4.8	5.2	5.6	5.8	—	—
	比	0.87	0.81	0.76	0.75	0.74	0.69	—	—
300	列方式	7.3	8.3	9.4	10.4	11.5	12.5	15.8	18.1
	リスト方式	6.3	6.5	7.1	7.7	8.1	8.8	11.7	13.5
	比	0.86	0.78	0.75	0.74	0.70	0.70	0.74	0.74

表 5 対称帯行列用番地付けの結果  
Table 5 Results for addressing on band symmetric matrix.  
帯幅  $m=n/10$  (単位 s)

モード	方式	次数 $n$	帯幅 $m=n/10$							
			100	200	300	400	500	600	800	1000
スカラ	従来方式	$3.3 \times 10^{-3}$	0.020	0.060	0.13	0.25	0.42	0.97	1.9	
	方式 1	$6.0 \times 10^{-3}$	0.043	0.14	0.34	0.66	1.1	2.7	5.2	
	方式 2	$4.5 \times 10^{-3}$	0.034	0.11	0.27	0.52	0.89	2.1	4.1	
ベクトル	従来方式	$3.2 \times 10^{-3}$	0.013	0.029	0.051	0.081	0.12	0.21	0.34	
	方式 1	$2.0 \times 10^{-3}$	0.008	0.022	0.049	0.092	0.16	0.36	0.70	
	方式 2	$1.0 \times 10^{-3}$	0.003	0.008	0.018	0.032	0.054	0.13	0.24	

ユー上で効率のよくなるような番地付けを考える。そして、この対応付けにリストベクトルを用いる。新しい番地付けのやり方にはいくつかあるが、最も効率がよいのは図 10 の方式 (b) である。つまり対象とする長方形の斜め方向に順番を付けていく。このやり方だとリストベクトルの隣り合った要素の値が異なるためにバンクコンフリクトを生ずることがなく、最大限にベクトルプロセッサの性能を活かせる。

図 10 の (a) および (b) のそれぞれ的方式で帯行列を計算させたときの処理時間を表 4 に示す。表 4 から  $n$  が 20 前後のときリストベクトル方式の効果が上がることがわかる。

5.2 対称帯行列

対称帯行列のときは上帯幅と下帯幅が等しいので、半帯幅を扱えばすむ。しかも、普通は部分軸選択をせずに対角要素を軸要素に選ぶので、改訂コレスキー分解の過程で帯幅が変化することはない。いま、この半帯幅を  $m$  とおく。  $m$  が小さいとスーパーコンピュータの性能が十分発揮できないので、リストベクトルを使って  $m$  行  $n$  列の 2次元配列  $A$  を 1次元化することを考える。帯行列の上半分を図 11 に示そう。このとき、作業領域は斜線で囲んだ領域となる。

改訂コレスキー分解の各段階で参照されるこの作業

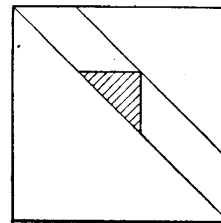
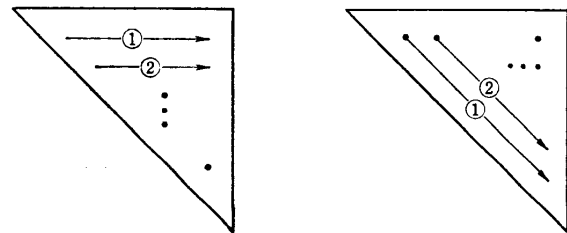


図 11 対称帯行列と作業領域  
Fig. 11 Band symmetric matrix and its working set.



(a) 方式 1 (b) 方式 2

図 12 対称帯行列に対する番地付け  
Fig. 12 Addressings for band symmetric matrix.

領域の大きさは一定で、ただ対角線に沿って一段ずつ右下にずれていくにすぎないので、この範囲の最適な番地付けを考える。このとき最も効率がよいのは図 12 の (b) に示すような斜め方向の順番付けであって、バンクコンフリクトを防ぐことができる。

図 12 の (a) および (b) のそれぞれの方式で帯行列を計算させたときの処理時間を表 5 に示す。

## 6. 高速フーリエ変換

高速フーリエ変換 (FFT) には種々の計算アルゴリズムが提案されている。その一つの方法である、入力データに重ねてデータ変換する方式、いわゆるビット反転方式はベクトル演算に適合しない。そこでデータの入力領域と出力領域を交互に使用する方法が用いられる。ここでは基底が 2 の複素 FFT の計算にリストベクトルを使用して高速化する方法を述べる。

図 13 に 16 点の FFT のデータの流れ図を示す。この場合のデータの番地付けは図 13 において横方向に付けても、縦方向に付けてもよい。また DO ループの構成で  $L$  および  $M$  のどちらを内側の DO ループのループ長としてもよい。いま番地付けは横方向に、内側の DO ループ長を  $L$  とすると 1 段当りの変換は次のようにプログラムされる。ここで  $L, M$  は図 13 と同じ意味で使用している。

```

SUBROUTINE FFT 1 (X, Y, T, L, M)
  COMPLEX*16 X(L, 2, M), Y(L, 2, M), T(L,
  M)
  DO 20 J=1, M
    DO 10 I=1, L
      Y(I, J, 1)=X(I, 1, J)+X(I, 2, J)*T(1, J)
10  Y(I, J, 2)=X(I, 1, J)-X(I, 2, J)*T(1, J)
20  CONTINUE
  RETURN
  END

```

このプログラムで 1024 点の FFT を実行すると最内側の DO ループ長、すなわちベクトル長は 512, 256, 128, 64, 32, 16, 8, 4, 2, 1 とだんだん短くなっていく。単純に対策をするには  $L$  と  $M$  の大きさが逆転したところで、DO 10 と DO 20 のループを入れ換えるとともに番地付けも同時に縦方向番地付けに変える方法がある。しかしそれでも、1,024 点の FFT ではベクトル長 32 の長さで実行される場合が最も多くなる。リストベクトル機能をもつベクトル計算機なら、ベクトル長をさらに長くして高速化することができる。

図 13 において横方向に番地付けしたとすると、ストアされる要素は  $L \times M$  の長さで連続アドレスとなる。問題はロードするほうのデータである。回転因子  $T$  はテーブルとして用意しておけばよいから、ベクトル長を長くするための問題点は変換データのロード

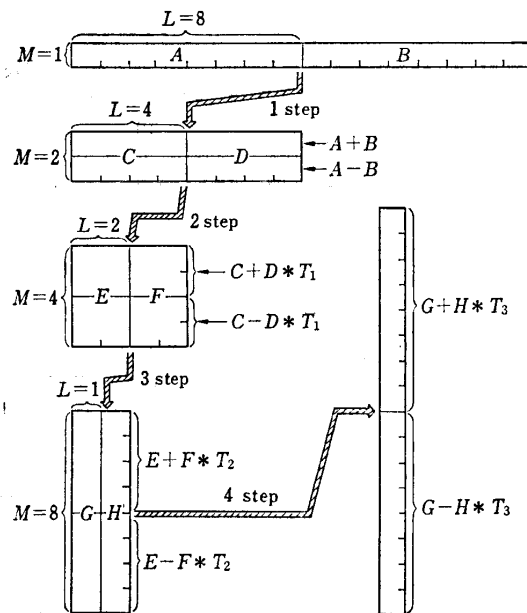


図 13 16 点 FFT のデータの流れ  
Fig. 13 Data flow for 16 points FFT.

にある。図 13 の第 2 段で見ると  $C$  のデータは相対番地で 0, 1, 2, 3, 8, 9, 10, 11 に第 3 段の  $E$  のデータは 0, 1, 4, 5, 8, 9, 12, 13 に入っている。 $D$  は  $C$  と、 $F$  は  $E$  と同じである。このため以上のようなデータをあらかじめ用意しておけば、リストベクトルを使用して取り出すことができる。この場合の 1 段当りの変換は次のようなプログラムとなる。ここで  $N/2$  は変換点数の半分の値であり、 $L$  は図 13 と同じ意味で使用している。

```

SUBROUTINE FFT 2 (X, Y, T, LIST, L,
  N/2)
  COMPLEX*16 X(N/2*2), Y(N/2, 2), T(N/2)
  DIMENSION LIST (N/2)
  DO 10 I=1, N/2
    Y(I, 1)=X(LIST(I))+X(LIST(I)+L)*T(I)
10  Y(I, 2)=X(LIST(I))-X(LIST(I)+L)*T(I)
  RETURN
  END

```

HITAC S-810 モデル 20 において本方式を使用すると前述の工夫した方法に比較して 1,024 点の FFT で 2.3 倍高速化された。またスカラ処理と比較すると約 30 倍高速化される。

## 7. むすび

本論文では、連立一次方程式の求解および高速フーリエ変換においてリストベクトルの使用によりスーパ



コンピュータの性能をどう活かせるかを実験に基づいて示した。特に、連立一次方程式の反復解法において著しい成果が得られることがわかり、今後、数値流体力学や半導体デバイスシミュレーションなどに普及していくものと考えられる。ここで示した成果は S-810 用行列演算副プログラム集 MATRIX/HAP に取り入れる予定である。

リストベクトルの大規模線形計画法プログラムへの適用なども大きなテーマとしてまだ残っており、今後とも継続して研究していきたい。

謝辞 高速フーリエ変換について指導していただいた東京都立大学小野寺嘉孝教授にお礼申し上げます。

### 参 考 文 献

- 1) 小国 力, 後 保範, 西方政春, 長堀文子: 直接解法による連立一次方程式のコンピュータ解法の特長解析, 情報処理学会論文誌, Vol. 25, No. 5, pp. 804-812 (1984).
- 2) 小国 力: スーパーコンピュータの利用技術—行列計算, 物理学会講習会テキスト, 東京 (1984).
- 3) 後 保範: ベクトル計算向き ICCG 法, 京大数理解析研究所講究録, No. 514, pp. 110-134, 京都 (1984).

(昭和 60 年 4 月 15 日受付)

(昭和 60 年 6 月 20 日採録)