解剖法順序を活かす多重スカイライン法

寒川 光†

FEMによって生成される対称疎行列の三角分解の計算量は，節点の番号付けに依存する．正方形領域に対して自然な順序で番号付けすると $O(n^2)$ であるが，解剖法によると $O(n^{1.5})$ に削減される．しかしこの順序による疎行列は非ゼロ要素が行列全域に散らばり，計算量削減の効果を計算時間の短縮に結び付けるプログラミングが困難になる．本稿では，解剖法順序生成の反復を途中で打ち切ることで生成される順序を用いた対称疎行列の，高速な三角分解の計算法を提案する．この方法は，現在最もポピュラーな三角分解の計算法の1つであるスカイライン法に，この順序による行列の特長を認識する機能を加えた，**多重スカイライン法**である．この方法は容易な並列化が可能である．RISC計算機（RS/6000）と分散メモリ型並列計算機（SP2）での計測結果によって，この方法が有効であることを示す．

Multiple Skyline Method for Nested Dissection Ordering

HIKARU SAMUKAWA†

A computational complexity of triangular factorization of sparse symmetric matrices generated by FEM depends on the numbering schemes. For the problem of square domain, though the natural ordering provides $O(n^2)$, the nested dissection ordering provides $O(n^{1.5})$. However, since non-zero elements of the matrix generated through this scheme are scattered widely, it is difficult to take advantage of the reduced complexity to shorten computational time. We propose a new algorithm of triangular factorization for symmetric sparse matrix generated through termination of nested dissection scheme. The new **multiple skyline method** adds a function to recognize the zero-elements position in triangular factor generated through the nested dissection ordering on existing skyline method programs. The new method provides straightforward parallelization. We demonstrate the effectiveness of this method on RISC computer (RS/6000) and distributed memory parallel computer (SP2).

1. はじめに

有限要素法（FEM）によって生成された対称疎行列は，非ゼロ要素が不規則に分布する．このような行列の三角分解に要する計算量は，行/列の入れ替え（内部節点順序）によって変化する．

1辺のメッシュ数が m の正方形領域で，外周の1辺に1から $m+1$，次にその内側の節点に $m+2$ から $2m+2$ を振るような「自然な順序」での計算量はおよそ m^4 Mflop（Mega FLoating-point OPerations）になる．すなわち行列の次数 $n=(m+1)^2$ に対して $O(n^2)$ である．解剖法による節点順序ではこれを $O(n^{1.5})$ に減らすことができる[1]．

解剖法は（部分）領域を複数の部分領域と，それらが直接には接続しないように入れる境界節点とに分割する操作を，部分領域がそれ以上細かく分割できなくなるまで反復する．各反復で部分領域から分離された境界節点は，部分領域の直後に入れる（最初の反復で生成された境界節点が最後に，2回目の反復で生成された境界節点はその前に置かれる）．なおこの反復を数回で打ち切って生成される節点順序も**解剖法順序**（nested dissection ordering）と呼ぶことにする．

解剖法が開発された70年代初めの計算機は，浮動小数点演算が遅かったため，この計算量の減少が高速化につながると期待された．しかし80年代の科学技術計算用の計算機の主流がベクトル計算機になったため，解剖法による計算量の減少が，そのままの比率で計算時間を短縮することはなかった．これは反復を最後まで行って得られる番号付けで生成された係数行列は，非ゼロ要素が行列の全域に不規則に散らばり，行列要素をアクセスするために増加する処理が，浮動小数点計算の減少を打ち消してしまうからである．この傾向はRISC計算機にもあてはまる．この結果，解剖

† 日本アイ・ビー・エム株式会社東京基礎研究所
Tokyo Research Laboratory, IBM Japan, Ltd.

法順序が計算時間の点で効果を発揮するのは，非常に大規模な問題に限られた．

ベクトル計算機やRISC計算機では，三角分解の計算時間を最小にする解剖法反復回数は，問題の規模と，計算機の性質に依存する．また部分領域を細分化しすぎると，行列生成などの三角分解以外の部分で，計算時間が長くなる傾向がある．さらに解剖法順序を，節点の接続情報（グラフ）だけから生成する一般的なアルゴリズムとして，グラフから得られる行列の固有ベクトルをもとにグラフを分割する操作を反復する方法（グラフ分割法）が知られている[2]．解剖法順序をこのアルゴリズムで求めることは，現在頻繁に解かれる規模の問題で考えると，三角分解そのものよりも多くの計算時間を必要とすることが予想される．

解剖法とFEM解析の時間はこのような関係にある．本稿は（表面的には）三角分解の計算時間を議論するものであるが，順序生成も含めた解析時間，順序生成を除外した時間，三角分解だけの時間の3つの時間を意識している．本稿では順序生成をプリプロセッサで済ませることで，一度生成した節点順序を再利用できる環境を前提とする☆．このような環境で使用される解析パッケージは，部分領域数に大きな幅を想定しなくてはならない．一方解析パッケージに組み込まれるルーチンは，解剖法の反復数の多寡に応じて複数のアルゴリズムを実装するのでは大変なので，単一のアルゴリズムで対応したい．本稿ではこのアルゴリズムとして**多重スカイライン法**を提案する．

2. 解剖法と計算量および計算時間

対象領域を等しい大きさの**2**つの部分領域と，両者が直接接続しないような干渉領域（境界節点）とに分割する操作を r 回反復すると，対象領域は 2^r 個の部分領域と境界に分けられる．この領域分割の方法をRBP（Recursive Bisectional Partitioning）と呼ぶ．

部分領域の節点消去の計算量　対応する行列は縁どり型になる．部分領域数が4の場合を示す．

$$\begin{pmatrix} A_{11} & & & & A_{15} \\ & A_{22} & & & A_{25} \\ & & A_{33} & & A_{35} \\ & & & A_{44} & A_{45} \\ A_{15}^t & A_{25}^t & A_{35}^t & A_{45}^t & A_{55} \end{pmatrix}$$

K 番めの部分領域の節点消去には，対角位置の小行列の三角分解 $A_{KK} \rightarrow C_{KK}^t C_{KK}$ と，前進代入 $C_{K,N+1} = C_{KK}^{-t} A_{K,N+1}$，および階数更新 $A_{N+1,N+1} \leftarrow A_{N+1,N+1} - C_{K,N+1}^t C_{K,N+1}$ による．A_{KK} や $A_{K,N+1}$ は疎行列なので，この部分の計算量は節点順序に依存する．この場合，境界節点に接続する節点を後回しにすると少なくなる☆☆．

境界節点消去の計算量　RBP反復を行うと，部分領域側の計算量が境界節点側に移動する．反復を最後まで行うと，部分領域側の計算量は無視できる範囲になる．境界節点側の計算量は $m \times m$ の正方メッシュの場合には陽なかたちで求められ，$n = (m+1)^2$ に対し $O(n^{1.5})$ である[1]．反復によって計算量が減るのは，反復のたびに部分領域の帯幅が狭くなる効果が大きいので，細長い領域を分割する反復や，小さな部分領域を分割して多くの境界節点を生み出す反復では，減らなかったり，増えたりもする．したがって反復を何回か繰り返して，部分領域側の計算量が少なくなった後では，反復による計算量の減少もわずかになる．

反復回数と計算時間　非ゼロ要素が完全に分散するとカーネルの計算速度は著しく低下するので，反復を最後まで行うことは，計算時間すなわち(計算量)÷(計算速度)の観点からは好ましくない．

そこで反復を途中で打ち切った場合の計算量を調べる．部分領域消去の計算量は，部分領域内部の節点順序で変化する．ここでは境界節点に接続する節点を後回し（他の節点順序はもとの順序）として，プログラムにより数えた（**表1**）☆☆☆．反復回数に対する非ゼロ要素の分布や小行列のサイズの関係は，**図1**（$m = 32$ で6回反復した節点順序で作られるフィルインを含む行列の図）から読み取ることができる．

表1から，最後の数回の反復では計算量の減少はわずかなことが分かる．反復の停止については，解剖法順序が三角分解以外の部分の計算時間を長くする副作用にも着目すべきである．たとえば表の下段に括弧に

表1　反復回数と計算量（Mflop）
Table 1　RBP iterations vs. complexity.

m	0	2	4	6	8	10	12
32	1.2	.67	.59	.48	.46	–	–
64	18.	9.5	7.4	4.9	4.3	4.2	–
128	276.	143.	102.	54.	40.	37.	37.
	(1.0)	(1.0)	(1.1)	(1.3)	(1.6)	(2.2)	(4.0)

☆ 使用者はメッシュ分割の時点では，その節点の接続関係で何回三角分解を実行するかを予想できるので，解剖法順序を用いるかどうか，また用いる場合は部分領域数も決められる．

☆☆ $C_{K,N+1}$ のプロファイルを小さくできる．

☆☆☆ これらの計算量は1節点を1自由度（連立方程式の1未知数）とした場合で，1節点に2から6の自由度が対応する構造解析では，これらの計算量を $2^3 = 8$ 倍から $6^3 = 216$ 倍した計算量が，実際の三角分解で必要になる．

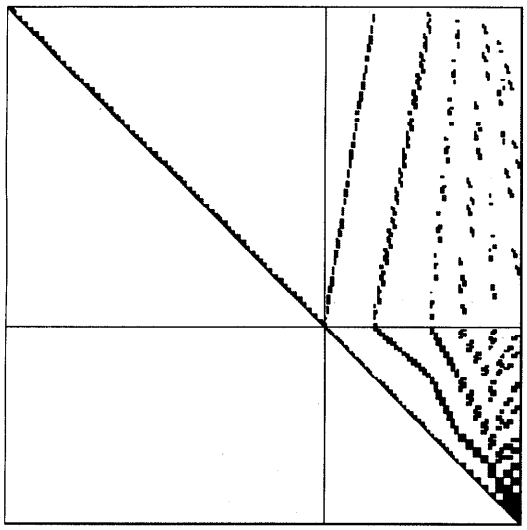

図1 6回のRBP反復による疎行列
Fig. 1 A sparse matrix after six RBP iterations.

入れて示した値は，$m = 128$ の場合の有限要素行列を生成する回数の，反復ゼロに対する比である☆．部分領域が小さくなるとこの種の副作用は増加しやすい．

解剖法順序の生成 順序生成そのものの時間もかなり増える．パラメータスタディなどのために，同じ接続関係で何度も解析するとか，非線形解析で何度も三角分解を行う場合は，この時間は補償されやすい．このような場合以外では，メッシュ分割の過程で部分領域分割が簡単に得られる場合が考えられる．自動車のホイールを例にあげると，メッシュデータは，リムの一部分，スポーク，ハブなどの基本部分を繰り返して作られる．これらの分離が保存されていれば10個程度の領域分割は簡単に得られる．

以上の考察からRBP反復回数は，少ないものはゼロ，多いものでも部分領域にある程度まとまった節点数が残るまでの範囲を対象とすることにする．

3. 多重スカイライン法

多重スカイライン法は解剖法順序で生成される小行列を効率良く扱うために，本章に述べるハイパテーブルと，記憶域での小行列の格納形式を使用するものとする．前者は順序生成プログラムから解析パッケージへのインタフェースを，逐次/並列計算にかかわらず単一とする．

☆ 複数のグループ（後述）をつなぐ要素は複数回生成するという行列生成ルーチンを仮定した．

1	2	5	10	(73)	42	37	34	33
3	4	6	11	(74)	43	38	35	36
7	8	9	12	(75)	44	39	40	41
13	14	15	16	(76)	45	46	47	48
(69)	(70)	(71)	(72)	(81)	(65)	(66)	(67)	(68)
26	27	28	29	(77)	58	59	60	61
21	22	23	30	(78)	62	53	54	55
18	19	24	31	(79)	63	56	50	51
17	20	25	32	(80)	64	57	52	49

図2 正方形領域の分割/番号付け
Fig. 2 Ordering of a square domain.

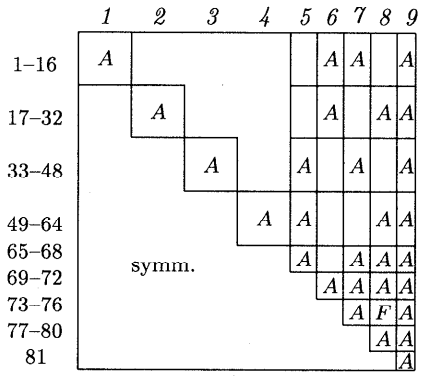

図3 疎行列
Fig. 3 Sparse matrix.

3.1 グループ化とハイパテーブル

各部分領域をそれぞれ1つの**1次グループ**とする．境界節点は，「特定の数」以下の部分領域としか接続しない節点（**2次グループ**）と，それより多くの部分領域の節点に接続する節点（**3次グループ**）に分離する．2次グループは同一の部分領域と接続する節点をまとめて1つのグループとする．3次グループはまとめて最後に1つのグループとする．この「特定の数」を**2次グループ接続数**と呼ぶことにする☆☆．

図2は対象領域を 8×8 の64個の四辺形要素で構成し，4つの部分領域とした例である（節点1，2，3，4に1つの四辺形要素が接続しているが，要素の辺は省略した）．対応する行列を**図3**に示した．A は非ゼロ要素を含む小行列，F は元の行列ではゼロだが，分解後にフィルインを含む小行列を表す．

ここでは2次グループ接続数を2としたので，4つの部分領域と接続する節点81は3次グループになり，

☆☆ 非常に多くのグループに接続する節点を考慮してハイパテーブルを作成することは厄介なので，例外的に接続相手の多い節点を3次グループに回すと，順序生成のプログラムが単純になる．

表2 ハイパテーブル
Table 2 Hyper-table.

グループ	列数	*llen*	*lstblk*
1	16	3	*6,7,9*
2	16	3	*6,8,9*
3	16	3	*5,7,9*
4	16	3	*5,8,9*
5	4	3	*7,8,9*
6	4	3	*7,8,9*
7	4	2	*8,9*
8	4	1	*9*
9	1	0	

最後に回る．以降，グループ番号をイタリック体で示す．部分領域が増えると1節点だけのグループが増えるが，2次グループ接続数はこのようなグループをまとめて1つのグループにするオプションである．部分領域数があまり多くない場合，3次グループを用いることで並列計算での通信量を減らせる．

解析パッケージへ入力する情報は，(1) 節点順序，(2) 各節点が所属するグループ番号，(3) ハイパテーブルの3つとする．

ハイパテーブルは図3を例とすると，**表2** になる．必要な情報は，1，2，3次グループ数，各グループを構成する節点数，接続するグループ数 *llen* とそのリスト *lstblk* である．接続グループは元の行列に対してでなく，フィルインを考慮した三角行列に対して作成する．

図で第*6*グループの列要素に着目すると，行位置で節点33から68までがゼロである．通常のスカイライン法ではプロファイルの内部は密として扱うので，これらのゼロ要素は認識されない．多重スカイライン法では，アクティブになった列が行番号33でゼロに戻ることがハイパテーブルから分かるので，解剖法順序が活かされる．特定の列に着目すると複数のスカイラインがあるので多重スカイライン法と呼ぶ．

3.2 データ構造

ベクトル計算機やRISC計算機ではカーネルループの種類によってMflop/s単位の計算速度が大きく変化する．この現象は，解剖法順序で生成された行列を扱うとき，特に注意を要する．ここでは速いカーネルを使用することと，冗長な演算を排除することの，対立する要件を両立させたい．両立の鍵は，行列の記憶域での形式（データ構造）にある．多重スカイライン法では行位置が1次グループの小行列についてはスカイライン形式を用いる．すなわち主小行列は通常のスカイライン法と同じ，非対角位置の小行列は小行列の底辺とスカイラインに囲まれたプロファイルを扱う☆．

行位置が2，3次グループの小行列については密行列とする．すなわち，対角位置の小行列は三角行列，非対角位置の小行列は長方形行列とする．密行列としても，列位置が2次グループの行列の積 $C_{KI}^t C_{KJ}$ はまばらさをほとんど残さないので，ゼロ演算の混入や記憶容量の面でも無駄は少ない（多重スカイライン法でも通常のスカイライン法と同様に，計算結果を入力行列に上書きする）．

3.3 計算法の例

行列の生成，重ね合せルーチンはハイパテーブルに似た配列に，小行列を格納する配列の先頭インデックスを格納することで（既存の重ね合せルーチンにより）処理できる．この方法により行列要素は，ハイパテーブルとスカイラインインデックスの2段階の間接参照によりアクセスされることになるが，前者は小行列ごとに1回なので，小行列のサイズがある程度の大きさになればオーバヘッドは減る．

1次グループ数を N_1，2次グループ数を $N_2 - N_1$，3次グループ数を $N_3 - N_2$ として，部分領域を消去する計算法を示す．なお $N_3 - N_2$ は1かゼロである[3]．

```
do K = 1, N1
  A_KK → C^t_KK C_KK
  do j = 1, llen(K)
    J = lstblk(j, K)
    C_KJ = C^{-t}_KK A^{(K-1)}_KJ
  enddo
  do i = 1, llen(K)
    I = lstblk(i, K)
    do j = i, llen(K)
      J = lstblk(j, K)
      A^{(K)}_IJ = A^{(K-1)}_IJ − C^t_KI C_KJ
    enddo
  enddo
enddo
```

大文字の K は部分領域の，J，I は境界領域の小行列の添字である．

カーネルは三角分解，前進代入，対称階数 k 更新，階数 k 更新の4種類になる☆☆．更新された行列は2次グループの行列に直接足し込むことができる．

2，3次グループについても *do* $K = N_1 + 1, N_3$ とすることで同様に計算できるが，ここではカーネルは密行列を対象とするので，スカイライン形式のものと

☆ 部分領域の節点で境界節点に接続する節点を後に回す処理を行わずに生成された節点順序に対しても，スカイライン形式なら対処しやすい．

☆☆ 三角分解は通常のスカイライン法のサブルーチンがそのまま，代入は若干修正すれば使用できるが，スカイライン行列を入力とする階数更新ルーチンは一般的でないのであらたに開発することになる．

表3 正方形領域モデルの計算量と計算（CPU）時間
Table 3 Complexity and CPU time (square domain).

部分領域数	計算量（Mflop）	計算時間（秒）
1	2230	13.6
4	1162 (1130+32)	7.6 (7.4+0.2)
16	841 (702+139)	6.1 (5.2+0.9)
64	436 (231+205)	4.0 (2.6+1.4)
256	327 (63+263)	3.5 (1.2+2.3)

（部分領域+境界領域）

で，合計8つのカーネルサブルーチンを用意する．

計算速度の測定 1節点を2自由度として，正方形領域を 128×128 にメッシュ切りしたモデル（$n = ((128+1)^2 - 2) \times 2 = 33,278$）に対し，自然な順序と，RBPを2，4，6，8回適用して計測した（**表3**）．なおこの表の計算量は，表1の計算量の8倍にほぼ一致している．これは多重スカイライン形式にしたことで増加するゼロ演算が少ないことを意味している．

2次グループ接続数は，部分領域数が4と16では2としたので，3次グループが存在するが，部分領域数が64では4として，3次グループなしとした．これは境界線上に節点が49個も現れると，これらの節点をまとめて後回しにすると計算量が増加するからである．

使用した計算機はRS/6000の990型（POWER2アーキテクチャ，クロック周波数71.5 MHz，理論最高性能値286 Mflop/s）である．計算速度は自然な順序では164 Mflop/sであるが，領域数が増えるに従って153，138，109，93 Mflop/sと遅くなる．しかしそれを上回る計算量の減少のために計算時間（CPU時間）は短縮される☆1．なお密行列の積 A^tB ルーチンはRS/6000では A の行数50，列数16程度で十分な性能に達する．ここでの性能の低下は1節点しか存在しないグループ（2×2 の小行列）の影響が大きい．

3.4 並列化の例

スカイライン法には，大規模な問題に対してはブロックスカイライン法と呼ばれる，行列を作業域のサイズに適した等しいサイズの（列数は異なる）ブロックに分割（1次元配列分割）して，外部ファイルに置き，1時点では2つのブロックを対象に計算を進める方法がある．多重スカイライン法をこのような方法に改訂するのは，直接アクセスファイルが可変長レコードをサポートしないので厄介である☆2．

しかし分散メモリ型並列計算機がポピュラーになりつつある現在は，行列を外部ファイルに置くよりも，複数の計算機の主記憶装置に分散するほうが便利になると思われる．

並列化に際しては小行列の行添字ごとにノードに割り付けた☆3．これは可変サイズのブロックによる1次元配列分割になる．行列本体とスカイライン添字はローカルの視野になるが，ハイパテーブルはグローバルの視野としてSPMD（Single Program Multiple Data stream）化した☆4．ノード数（np）は2のべきのみを使用可能とし，メッセージ交換ライブラリはMPIを使用した．

順位とノードの割付け 従属性の検出はハイパテーブルを検索して，並列に消去できる小行列の組合せを調べることで行う．部分領域を順位ゼロ，2次グループの中で他の2次グループからの更新を受けないグループを順位1，順位1のグループからしか更新を受けないグループを順位2，… とする．この結果は1次元配列 $level(N_1+1 : N_3)$ に格納する．

ノードの割付けは，1次グループは $ns = N_1/np$ として，ns 個ずつ配置する．

2，3次グループのノードの割付けは順位を用いて決定した．まず順位1の2次グループは，そのグループに対する更新行列 $F_{IJ}^{(K)}(= C_{KI}^t C_{KJ})$ を作るノードのうち，最も若いものに一致させた．この方法によって隣り合う部分領域が同一所有者の場合，それらに挟まれた境界節点も同じ所有者になるので，この2次グループは通信とは無関係になる．順位2以上の2次グループは，そのグループに対する更新行列の通信量が最も多いノードに一致させた．ただし並列性のないグループは同一ノードに割り付ける．

グループとノードの対応は，1次元配列 $iown(1 : N_3)$ に格納する．この $level$ と $iown$ もハイパテーブル同様，全ノードが同一の内容を持つ．

部分領域の消去の計算では $do\ K = 1, N_1$ のループを次のように変換する．

```
ns = N1/np
  do ks = 1,ns
    do K = ks,N1,ns
      if (iown(K).eq.myid) then
```

ただし $myid$ はそのSPMDプログラムを実行するノード番号である．

☆1 この問題を，キャッシュブロック化やレジスタブロック化等のRISC計算機向きのチューニング[4]を行わないスカイライン法で，自然な順序で解くと34秒（66 Mflop/s）を要した．

☆2 小行列を多数のワークファイルに分散させる方法が求められる．ここで，1つのファイルのレコード長は，そのファイル上の最大の小行列のサイズに合わせる方法が考えられる．

☆3 分散メモリ型並列計算機で，各プロセッサとそれに付随する主記憶装置などをノードと呼ぶ．

☆4 配列を分割して複数のノードに分散する場合，分散された個々の配列はローカルの視野，分割せずに全ノードが同一の内容の配列を持つ場合，その配列をグローバルの視野と呼ぶ[5]．

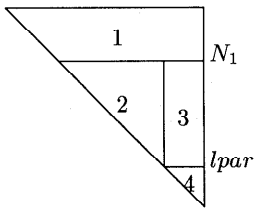

図4 並列化の計算順序
Fig. 4 Order of computations in parallel.

更新される行列 A_{IJ} が他のノードの所有の場合は，$F_{IJ}^{(K)}$ を送信し，対応するノードが受信してから足し込む（非同期通信 *mpi_isend/irecv* を用いる）．

境界領域の消去 逐次処理はKIJ型のループ構成を示したが，これを変更することで通信量を減らしたり，バッファを小さくすることができる．ここでは順位を検索して，並列性がない小行列を特定する（並列性のある最後のグループを $lpar$ とする☆）．

はじめに消去計算は $lpar$ までのグループに対する更新だけを選択的に完了させ，次に同じループ構成で（ただし主小行列の分解と前進代入は行わない），更新行列を $A_{lpar+1:N_3,lpar+1:N_3}$ と同じ順序でバッファに作り，縮約通信 *mpi_reduce* でグループ $lpar+1$ の所有者に送る．最後にこの所有者がこれを分解する．

この計算の分割を，更新計算 $-C_{KI}^t C_{KJ}$ の C_{KI} の位置によって示した（**図4**）．

なお'2'の部分は順位駆動型でループを記述した．これは同一の順位を持つ小行列の集合を1つのブロックと見なして，JIKIJ型のループ構成とし，転置通信 *mpi_alltoallv* により通信量を減らしている．ただしここでの順位は通信の依存性によるものにしておく．

計算速度の測定と考察 逐次計算の計測に用いた 128×128 のモデルと，立方体領域を $20\times20\times20$ にメッシュ切りしたモデル（$n=(20+1)^3\times3=27,783$）に対し，RBP反復を6回適用した64分割データを計測した（**表4**と**表5**）．なお，立方体領域モデルの計算量は14.3Gflopで，部分領域と境界領域はそれぞれ1.7，12.6Gflopである．

使用した計算機はIBMのSP2で，プロセッサにはRS/6000の390H型（990型よりもメモリ帯域が狭く，クロック周波数も66.5 MHzとやや遅い），相互結合網にはクロック周波数40 MHz（最新型の半分）のHPS（High Performance Switch）を搭載している．HPSは各ノードを時間的に等距離に置く[6]．

☆ 解剖法による節点順序では，通常，最初のRBP反復で生成された境界節点が，最後の密行列を形成するので，この部分が $lpar+1$ 以降となる．

表4 正方形領域に対する計算（経過）時間と計算速度
Table 4 Elapsed time and speed (square domain).

np	経過時間（秒）	計算速度（Mflop/s）
1	4.25 (2.6+1.6)	104 (86+133)
2	2.72 (1.9+.64+.17 +.04)	160 (121+265+194+161)
4	1.45 (.94+.35+.12 +.04)	300 (241+485+266+158)
8	1.01 (.54+.33+.11 +.04)	429 (421+519+307+138)
16	.648 (.27+.27+.078+.04)	670 (837+639+423+162)

表5 立方体領域に対する計算（経過）時間と計算速度
Table 5 Elapsed time and speed (cubic domain).

np	経過時間（秒）	計算速度（Mflop/s）
1	102.0 (19.4+82.6)	137 (80+150)
2	54.8 (11.2+28.0+11.8+3.7)	251 (139+291+275+207)
4	35.8 (5.8+16.3+10.0+3.7)	384 (270+500+325+207)
8	24.4 (3.2+11.4+ 6.1+3.7)	564 (493+717+533+207)
16	20.0 (1.7+10.5+ 4.0+3.7)	688 (883+780+811+207)

表の括弧内の数字は，図4に対する4つの部分の処理時間である（逐次処理では2，3，4は分離されない）．

並列化効率は部分領域の消去では良好であるが，ノード数が16以上では低下している．これは境界節点の消去で並列性が減ってゆくことに最大の原因がある．特に立方体の場合，最後の逐次処理部分に3.7秒を要しており，この影響が大きい．

改善策としては，並列性が少なくなったところで行列を再分散して，密行列として全ノードで三角分解する方法が考えられる．また，はじめから境界グループの小行列を2次元配列分割する方法も考えられる．前者については，対称密行列の三角分解ルーチンにメーカ提供のライブラリが利用できれば，再分散だけをプログラミングすればよいことになる．後者の方法は文献7)で用いられている．2次元配列分割は小規模な並列計算機では効果を出しにくい．

他の方法との比較 解剖法順序を三角分解に利用するには，多階層（マルチレベル）部分構造法のアプローチをとり，階層構造（親子関係）を「消去の木」として指定するのが一般的である[8),9)]．消去の木に従う方法は，隣り合う部分領域を2つ1組にして，境界面上の節点を消去する操作を繰り返す．この場合ループ構成はKIJ型に強制され，また各段で行列を並べ替える操作が出てくる．多重スカイライン法は，入力された行列に直接結果を書き込めるので（並列計算で用いる通信バッファ以外には）行列を格納するための追加的な配列は必要ない．このため機械的なループ変換による並列化や，ループ構成の変更も可能という**プログラミングの単純さ**が生まれる．

多階層部分構造法では，消去の木に埋め込まれた従

属性解析の結果を用いるが，木を作成する時点では，ノード数をはじめとする並列計算機の特性は未知の場合もありうる．多重スカイライン法では従属性解析を（ハイパテーブルをもとに）三角分解ルーチンが行う．したがって本稿で述べたように，従属性の制約の緩い1次グループではKIJ型で1対1の非同期通信により計算と通信のオーバラップを，従属性の制約の厳しくなる2次グループではJIKIJ型として転置通信により複数の通信のオーバラップを，並列性のないところでは縮約通信による通信量の削減を狙うという，**柔軟性**を持てる．これらの特長はハイパテーブルと行列の格納形式から生まれている．

分散メモリ型並列計算機は，相互結合網やノードの種類（SMPノード）など，種類が豊富である．これらに対応するためには，柔軟性は重要な性質である．

4. おわりに

並列計算機の能力の向上が著しい現在では，「大規模」と呼ばれる問題に対しては，$O(n^2)$ と $O(n^{1.5})$ の差は急速に広がりつつあり，スカイライン法など現行のアプローチ☆の再検討の時期にあるように思われる．解剖法は $O(n^{1.5})$ を実現できるが，解析全体の処理時間を考えると副作用も少なくない．部分領域数を非常に多くした場合に，副作用を抑えるためのプログラミングは，それほど簡単ではないかもしれない．一方並列化について考えると，少なくとも並列計算機のノード数程度の領域分割は求められる．解析対象と部分領域数を限定すれば，グラフ分割以外の実用的な領域分割の方法が考えられる．ここに解剖法の考え方を採り入れ，多重スカイライン法と組み合わせることができれば，比較的容易に並列計算機の性能を利用できる．

本稿では2次元および3次元構造物の代表として，解剖法の反復回数と計算量の関係がとらえやすい正方形と立方体の例のみを示し，複雑な接続関係を持つ一般的な構造物の計測結果を示せなかった．このような構造物に対して，解剖法順序を高速に生成すること自身，大きな研究課題である．しかしこの課題が解決されなくても，現状で少数の部分構造物に分割して解析することは行われているので，解剖法順序の利点を活用する場面は多いと思われる．なお並列化の計測結果には，小行列 C_{KI} の位置による部分ごとの計算速度を示したので，ある程度は一般的な構造物を解いた場合の性能の推定が可能かと思う．本稿がFEMタイプの疎行列の高速解法に役立てば幸いである．

☆ カットヒルマッキー法による節点順序と，スカイライン法などプロファイル内部を密行列として扱う三角分解ルーチンを組み合わせるアプローチ．

参考文献

1) George, J.A.: Block Eliminations on Finite Element Systems of Equations, *Sparse Matrices and their Applications*, Rose, D.J. and Willoughby, R.A. (Eds.), pp.101–114, Plenum Press (1972).
2) Pothen, A., Simon, H. and Liou, K.: Partitioning Sparse Matrices with Eigenvectors of Graphs, *SIAM J. Matrix Anal. Appl.*, Vol.11, No.3, pp.430–452 (1990).
3) 寒川 光：分散メモリ型並列計算機のための多重スカイライン法，情報処理学会HPC研究会報告，No.63, pp.25–30 (1996).
4) 寒川 光：RISC超高速化プログラミング技法，共立出版 (1995).
5) 寒川 光，郷田 修：HPFと並列処理，計算工学，Vol.1, No.2, pp.31–36 (1996).
6) Stunkel, C.B., Shea, D.G., Abali, B., Atkins, M.G., Bender, C.A., Grice, D.G., Hochschild, P., Joseph, D.J., Nathanson, B.J., Swetz, R.A., Stucke, R.F., Tsao, M. and Varker, P.R.: The SP2 High-performance Switch, *IBM Systems J.*, Vol.34, No.2, pp.185–204 (1995).
7) Gupta, A. and Kumar, V.: A Scalable Parallel Algorithm for Sparse Cholesky Factorization, *Proc. Supercomputing '94*, pp.793–802 (1994).
8) Liu, J.W.H.: The Multifrontal Method for Sparse Matrix Solution: Theory and Practice, *SIAM Review*, Vol.34, No.1, pp.82–109 (1992).
9) 福盛秀雄，河野洋一，西松 研，村岡洋一：階層化サブストラクチャ法の適用による有限要素法の並列化とその実装，情報処理学会HPC研究会報告，No.57, pp.79–84 (1995).

(平成8年11月18日受付)
(平成9年 7月 1日採録)

寒川 光（正会員）

1972年早稲田大学理工学部機械工学科卒業．同年日本ユニバック（株）入社．1984年日本アイ・ビー・エム（株）入社．現在同社東京基礎研究所勤務．数値解析，数値計算法，計算機アーキテクチャに関する仕事に従事．工学博士．計算工学会会員．