ハイパフォーマンス
コンピューティング 63－5
（1996．10．4）

分散メモリ型並列計算機のための多重スカイライン法

寒川 光

日本アイ・ビー・エム（株），東京基礎研究所

有限要素法 (FEM) によって生成された疎な対称行列を三角分解する問題を考える．FEM 構造解析では三角分解に帯行列法やスカイライン法を用い，番号付けにはカットヒルマッキー法やウェーブフロント法などのスキームを用いることで，帯幅を小さくするアプローチが一般に使用されている．しかし疎行列の三角分解の計算量は，節点の番号付けと三角分解の計算法の組合せによって大きく変化するので，このような単純なアプローチでは，無駄な計算を多く実行することになりかねない．本稿は，計算量の増加をきょくりょく抑え，カーネルの計算速度も良好で，分散メモリ型の並列計算機向きの並列化も容易な，**多重スカイライン法**を提案するものである．はじめに領域分割順序による番号付けが，自然な順序によるものよりも計算量をいちじるしく少なくする可能性を持っていることを示す．次にこの性質を活かした多重スカイライン法を逐次計算アルゴリズムとして述べ，さらに分散メモリ型の並列計算機を前提にこれを SPMD 化（並列化）する．またこれらの方法をプログラムを作成して性能測定し，アルゴリズムが有望であることを確かめた．

Multiple Skyline Method for Distributed Memory Parallel Computers

Hikaru Samukawa

Tokyo Research Laboratory, IBM Japan, Ltd.

We consider a problem of triangular factorization of sparse symmetric matrices generated by Finite Element Method (FEM). In the area of FEM structural analysis, approaches to combine band-matrix solver or skyline solver with reducing bandwidth of the coefficient matrix by Cuthill McKee or wavefront schemes are widly used. However, since a computational complexity in the factorization of sparse matrices sometimes radically changes according to the combination of factorization algorithms and numbering schemes, such a simple approach may increase its complexity. This paper proposes a new approach named **multiple skyline method**, which has capabilities to suppress complexity increase, good performance of the kernel routines, and simplicity in parallelization for distributed memory parallel computers. The first part of this paper describes the possible advantage of the numbering based on the domain-decomposition to the numbering based on the natural ordering by reducing the complexity. In the next part, the multiple skyline method to take this advantage into account is described as serial algorithm, then the SPMDization (parallelization) for the distributed memory computers is described. Some performance examples are attached to demonstrate the effectiveness of this approach.

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

図 1: 正方形領域の分割/番号付け

1 番号付けと計算量

1.1 自然な順序

1 辺の節点数が k の正方形領域に，自然な順序で番号付けして作られる対称行列の三角分解に要する計算量を考える．総節点数（行列の次数）を k^2，帯半幅 (semi bandwidth) を k とすると，計算量は k^4 と見積もられる．1 辺の節点数を $\sqrt{n}k$ に細分化すると，総節点数は nk^2，帯半幅は $\sqrt{n}k$ になるので，計算量は n^2k^4 となる．すなわち計算量は総節点数 n に対して $O(n^2)$ ということができる．

1.2 領域分割と番号付け

ここでは対象領域を複数の部分領域と境界領域に分割する．図 1 は対象領域が 8×8 の 64 個の四辺形要素で対象領域が構成された例である（節点 1,2,3,4 に 1 つの四辺形要素が接続しているが，要素の辺は省略して節点番号のみを示した）．本稿で用いる**領域分割** (domain-decomposition) は，各部分領域が直接接続しない部分領域と，それ以外の境界領域とに分割する．この例では 4 つの部分領域と，節点番号を

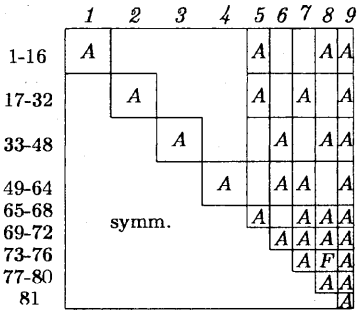

図 2: 疎行列

丸で囲った境界領域とに分割している．

番号付け (numbering または sequence generation) は，「部分領域の内部の節点を先に，2 つの部分領域に接続する境界節点を次に，3 つ以上の部分領域に接続する境界節点を最後に」とグループ化する．対応する行列を図 2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．ここではグループ化を，節点 1 から 16 までを（イタリック体で）*1*, ···，節点 65 から 68 までを *5*, ···，節点 81 を *9* として示した．グループ *1* から *4* までに含まれる節点は，**相互には直接接続していない**（連成がない）．

領域分割順序とフィルイン　この行列のコレスキー分解 $A \rightarrow C^tC$ を考える．行列 A は 81 項の対角項と（上三角部分に）272 項の非ゼロ要素をもつ．A を分解するとフィルインは 312 項になるが，これは自然な順序による場合よりも少ない．これには内部節点と境界節点の相互作用を表す小行列のフィルインを減らすために，**境界節点に接続する内部節点の番号付けを後回し**にしていることが寄与している．

領域分割順序と計算量　帯行列法やスカイライン法では，プロファイル（行列の各列の最初の非ゼロ要素と対角項によって囲まれる部分）の内側は密として扱われる．したがって領域分割順序を与えても，図 2の例でいうと，A_{35} などがゼロ行列になることを認識できない．

しかし C の非ゼロ要素だけに絞り込んでフロント行列を更新するウェーブフロント法では，このゼロ演算を回避するプログラミングが可能である[1]．**アクティブ列の停止機能** (termination of active-columns) はこの好例である [1].

図 1の領域を自然な順序で番号付けした行列のコレスキー分解の浮動小数点演算量は，スカイライン法でもウェーブフロント法でも 7.6 Kflop 値と同じであるが，領域分割順序では，スカイライン法は 20. Kflop 値に増加し，ウェーブフロント法は 5.3 Kflop 値に減少する．この例では 2 分割を再帰的に適用する方法 (**RBP**:Recursive Bisectional Partitioning) を 2 回反復しているが，大規模な問題では反復回数を増やして，部分領域数を増やすことで対応する．

[1]第 i 行を消去した時点でもっとも若いアクティブ列が $i+1$ よりもかなり大きくなった場合，フロント行列を 2 次的なファイルに吐き出すか，元の行列 A に足し込むなどして，アクティブ列なしの状態に（フロントテーブル，フロント行列をクリア）してから，第 $i+1$ 行目の消去に入る．

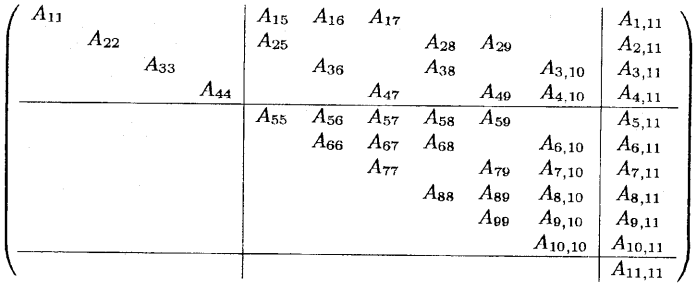

図 3: 2 レベル・スカイライン行列

2 多重スカイライン法

ウェーブフロント法でフロント行列の更新計算を，冗長なゼロ演算を回避するようにプログラミングすると，最内側ループにフロント行列の要素位置を計算する整数演算が入る．このためカーネルの計算速度が遅くなる．そこでスカイライン法の改良を考える．

領域分割によってできた節点を，部分領域内部に属するものと境界領域に属するものとにグループ化し，境界を後回しに番号付けして生成される（1 レベルの階層の）行列は縁どり型と呼ばれる．

多重スカイライン法はグループ化をさらに階層化することで並列性追求の容易性を狙った．ここでは相互には接続していない部分領域を **1 次部分領域** (level 1 sub-domain) と呼ぶことにする．

2.1 2 レベル・スカイライン法

境界節点を，2 つの 1 次部分領域としか接続しない節点（**2 次部分領域**）と，3 つ以上の 1 次部分領域の節点に接続する節点（**3 次部分領域**）に分離する．1 次部分領域が 4 つの場合は，2 次部分領域は最大 ${}_4C_2 = 6$ つにまとめられるが，一般にはこれよりも少なく，図 2 の例では 4 つである．1 次部分領域の番号を *1:4* として，この 6 つの 2 次部分領域がすべて存在する場合の行列の例を図 3に示した．部分領域 *5:10* が 2 次部分領域，*11* が 3 次部分領域である．この行列の特徴は，1 次部分領域の消去の影響が 2 次部分領域では，対角位置の小行列には 2 回ずつ，非対角位置の小行列には 1 回だけ現れることにある[2]．

2 次部分領域の番号付け　境界節点数が多くなると，2 次部分領域の小行列（図 3では $A_{5:10,5:10}$）にフィルインが増加しやすい．したがって 2 次部分領域にも適切な番号付けが要求される．図 4 に正方形領域を 4×4 の 16 個の部分領域に分割した例を示す（左が元の領域，中は 2 次部分領域を太線と丸印で示した）．1 次部分領域を消去したときの非ゼロ要素の分布は，右に示したように，2 次部分領域を節点，1 次部分領域を要素と考えた場合と同様の分布になる（2 次部分領域を◦印で，1 次部分領域を 2 節点要素，3 角形要素，4 辺形要素で表した）．したがってこの部分のフィルインを少なくするには，2 次部分領域にも領域分割順序で番号付けする必要がある．

正方形領域を 32×32 の正方形要素でメッシュ分割化し（節点数 1089），RBP を 6 回反復して 64 個の部分領域に分割すると，境界節点数は 413 になる．この境界節点は 364 節点からなる 112 の 2 次部分領域と，49 節点からなる 1 つの 3 次部分領域に分けられる．

この問題を自然な順序で番号付けすると，三角分解に要する計算量は 1.2Mflop 値，領域分割順序では 0.71Mflop 値である（図 5 にフィルインを含めた行列の例を示した）．ここでは 2 次部分領域を，**1 次部分領域を作成した領域分割順序を逆順に適用**して番号付けしている．しかし 2 次グループを図 3のように規則的に並べると 2.6Mflop 値と，自然な順序よりも計算量が増える．これは 1 次部分領域を消去したあとの 413×413 の行列に多くのフィルインが現れるためである[3]．

[2]非対角位置の $\sum_{K=1}^{I-1} C_{KI}^t C_{KJ}$ において，重ね合わせの加算が実際に発生するためには，2 つ以上の K について小行列 C_{KI} と C_{KJ} がともに存在しなければならないが，境界節点は，複数の部分領域から異なる 2 つを選んだので，C_{KI} と C_{KJ} がともに存在するのは 1 回だけである．

[3]$C_{KI}^t C_{KJ}$ の計算量は I が 1 次部分領域の範囲では同じだが，2 次部分領域に入ると，フィルインがあらたなフィルインを生み出して，計算量を増加させる．

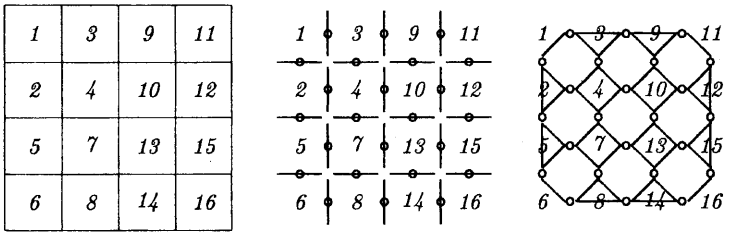

図 4: 2 次部分領域とフィルイン

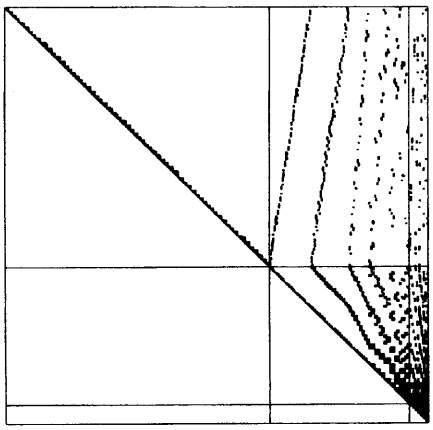

図 5: 6 回の RBP 反復による疎行列

表 1: ハイパーテーブル

部分領域	列数	*llen*	*lstblk*
1	16	3	*5,8,9*
2	16	3	*5,7,9*
3	16	3	*6,8,9*
4	16	3	*6,7,9*
5	4	3	*7,8,9*
6	4	3	*7,8,9*
7	4	2	*8,9*
8	4	1	*9*
9	1	0	

多重スカイライン形式 2 レベルに階層化された行列を格納する形式として，図3の A_{11},A_{15} のように，行位置が 1 次部分領域の小行列については，スカイラインの形状を考慮する[4]．行位置が 2,3 次部分領域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（これらの行列では初期にゼロの要素についてもゼロを入れておく）．

次にこれらの小行列の接続関係をハイパーテーブルとして与える．ここでは図 2を例とする．小行列の位置を，表 1 のようなハイパーテーブルで記述する．必要な情報は，まず 1 次部分領域数（この例では 4），2 次部分領域数（この例では 4）で，これらは三角分解ルーチンにには引数として渡されるものとする．次に各部分領域を構成する節点数，接続する部分領域数 *llen* とそのリスト *lstblk* である．接続部分領域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ので，この例では小行列 A_{78} は接続部分領域に含めておく．

行列要素は非ゼロ要素とフィルインについて，A_{IJ} の I 順に，フィルイン要素をゼロを挿入して 1 次元配列に格納する．この行列の要素はハイパーテーブルとスカイラインインデックスを介してアクセスする．

計算法 1 次部分領域数を N_1，2 次部分領域数を $N_2 - N_1$ として，1 次部分領域を消去する計算法を示す．

```
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 j = 1,llen(K)
      J = lstblk(j,K)
        do i = j,llen(K)
          I = lstblk(i,K)
          if(I.eq.J) then
            A^{(K)}_II = A^{(K-1)}_II − C^t_KI C_KI
```

[4] 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．

表 2: 計算量 (Mflop 値) と計算 (CPU) 時間 (秒)

領域数	計算量 (1+2 次)	計算時間
1	2230	17.3
4	1162(1130+32)	9.4(9.2+0.2)
64	600 (231+369)	5.4(3.0+2.4)

```
        else
          A_IJ^(K) = A_IJ^(K-1) - C_KI^t C_KJ
        endif
      enddo
    enddo
  enddo
```

大文字の K は1次部分領域の，J, I は2次または3次部分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[5]．更新された行列は2次部分領域の行列に直接足し込むことができる．

2次部分領域についても $do\ K = N_1+1, N_2+1$ とすることで同様に計算できる．4種類の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で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，正方領域を 128×128 にメッシュ切りしたモデル ($n = ((128+1)^2-2)\times2 = 33,278$) に対し，自然な順序と，RBP を2回および6回適用して，4および64部分領域として計算した (表2)．使用した計算機は IBM RISC System/6000*[6] (RS/6000) の 590H 型 (POWER2* アーキテクチャ，クロック周波数 72MHz，理論最高性能値 288Mflop/s) である．計算速度は自然な順序では 128Mflop/s であるが，領域数が増えるとやや遅くなる[7]．しかしそれをはるかに上回る計算量の減少のために計算時間 (CPU 時間) は大きく短縮される．

なお測定では，8種類のカーネルのすべてをチューニングしていない (計算速度にはチューニングの余地が残されている)．

2.2 並列化

並列化に際しては小行列の行ごとにプロセッサ割付けを行う．したがって行列本体とスカイラインインデックスはローカルの視野 (自分の所有分だけ) になるが，ハイパーテーブルはグローバルの視野 (全プロセッサがテーブル全体を持つ) として SPMD (Single Program Multiple Data) 化した．なお，プロセッサ数は2のべきのみが使用可能とした．

プロセッサ割付け　N_1 よりも少ないプロセッサ数の並列計算機を使用する場合は，隣合う1次部分領域とそれらに挟まれた2次部分領域を同じプロセッサに割り付ける．RBP を4反復して作られた行列を4プロセッサで解く場合は，1次部分領域は $1:4, 5:8, \ldots$ のように割り付ける．この割付けによって2次部分領域の消去における通信は，RBP を2反復して作られた境界だけが関係し，3,4回目の反復で作られた境界は無関係になる．部分領域とプロセッサの対応は，1次元配列 $iown(1:N_2+1)$ に格納する．

ループ変換　1次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 np はプロセッサ数，myid はその SPMD プログラムを実行するプロセッサ番号である．更新される行列 $A_{IJ}^{(K)}$ が他のプロセッサ所有の場合は，これを送信 (非同期) し，対応するプロセッサが受信して足し込む．足し込みは (通信用のバッファ容量に余裕があれば)，1次部分領域は連性していないので，いつ行ってもよい．

2次部分領域の小行列の消去計算には依存性があるので，あらかじめ非ゼロ行列の位置から，並列に消去できる小行列の組み合わせを特定しておく．図6に示した行列では *9:12* は並列に消去でき，これらが消去されれば *13* と *15* が消去できる．この計算の従属性に基づく**階数** (rank) を1次元配列 $irank(1:N_2+1)$ に格納しておけば，$do\ K = N_1+1, N_2+1$ のループを次のように変換することで SPMD 化できる．

```
do kr = 1,nrank
  do K = N1+1,N2
    if(irank(K).eq.kr.and.iown(K).eq.myid) then
```

ただしこの階数はすべての2次部分領域を異なるプロセッサが所有した場合で，階数更新を計算するプロセッサと足し込みを行うプロセッサが同じ場合は通信は発生しないので，従属性は無視できる (図6の例

[5] 三角分解と代入は通常のスカイライン法のサブルーチンが使用できるが，スカイライン行列を入力として密行列を出力する階数更新ルーチンはあらたに開発することになる．

[6] *印は米国 IBM Corporation の商標である．

[7] キャッシュブロック化 [2] 等を行わないスカイライン法では 34 から 40 秒を要した．

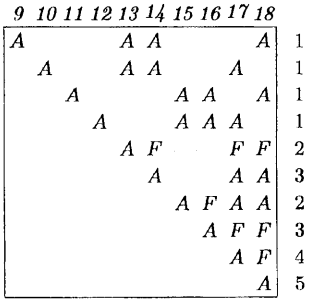

図 6: 計算の従属性に基づく階数

で *13* と *14* が同じ，および *15* と *16* が同じプロセッサの所有ならば，階数 2 と 3 は分ける必要がなくなる）．6 回の RBP 反復による図 5の行列では，計算の従属性による階数は 21 に達するが，これを 16 プロセッサで解く場合の通信依存性に変更すると 6 に減る．足し込みは階数が上がるときに完了させる．

3 次部分領域に対する足し込み用の作業域だけは，全プロセッサが所有して，最後に縮約通信してから，この小行列を所有するプロセッサが三角分解する．

計算速度の測定　128 × 128 のモデルに対し，RBP を 6 回適用した 64 部分領域のデータついて計測した (表 3)．使用した計算機は IBM の SP2* で，プロセッサには RS/6000 の 390H 型（590H 型よりもメモリ帯域が狭く，クロック周波数も 66MHz とやや遅い），相互結合網にはクロック周波数 40MHz（最新型の半分）の HPS (High Performance Switch) を使用している．

計算性能は 1 次部分領域の消去では良好であるが，2 次部分領域では不十分である．これはプロセッサ数の増加にともなう通信量の増加と負荷のアンバランスの影響である．どちらも行列の規模が大きくなるに従って緩和され，またプログラミング上のチューニングの余地も残しているので，多重スカイライン法は有望と考えられる．

表 3: 計算（経過）時間（秒）と通信量（K 倍語）

np	経過時間 (1+2 次)	比	通信量
1	6.0 (3.4+2.6)		
2	3.6 (2.1+1.6)	.60(.60+.61)	28+954
4	2.1 (1.1+1.0)	.34(.31+.39)	57+1815
8	1.5 (.47+1.0)	.25(.14+.39)	130+2453
16	1.2 (.27+.92)	.20(.079+.35)	239+2925

3　まとめ

FEM による数値解析を分散メモリ型の並列計算機で実行する場合は，行列生成のためにも領域分割は欠かすことができない．領域分割に引き続いて，三角分解の計算量が少なくなるような番号付けができると，非常に大規模な問題を高速に解くことができる．

本稿では領域分割に RBP を用い，逐次計算機および並列計算機で処理しやすい多重スカイライン法について述べた．また簡単なプログラムを作成して，この方法の有効性を確かめた．大規模な問題に対する計算時間は，カーネルルーチンの計算速度や並列計算機の通信性能によるよりも，番号付けの最適化によって計算量を減らすことに，より強く依存する．一方番号付けは三角分解の計算法を意識しなければ最適化できない．

本稿では正方形領域に対して RBP を適用したが，複雑な形状の領域に対する RBP による番号付けは (1 節点が 6 自由度としても)，三角分解の計算時間よりも多くの計算時間を要することが予想される．これにはグラフ理論に基づく，固有ベクトルによる領域分割の方法などが知られている [3]．しかし三角分解に対しては，境界節点数を少なくすることも重要なので，領域の形状が複雑になると，この方法だけでは不十分であろう．これらの点を考慮すると，番号付けはメッシュ生成の一部として処理しておき，これを何回も三角分解で利用できるようにすることが望ましい．このためには，生成すべき番号付けの持つべき条件が，三角分解の計算法に照らして明確に定義されなければならない．多重スカイライン法はこのようなアプローチのための 1 つの候補である．

参考文献

[1] C. W. McCormick : Sparse Matrix Operations in NASTRAN, *Theory and Practice in Finite Element Structural Analysis, proc.*, pp.611–631, 東大出版会, 1973.

[2] 寒川 光：『RISC 超高速化プログラミング技法』, 共立出版 ,1995.

[3] A. Pothen, H. Simon and K. Liou : Partitioning sparse matrices with eigenvectors of graphs, *SIAM J. Matrix Anal. Appl.*, Vol.11, No.3, pp.430–452, 1990.