スカイライン法のベクトルプロセッサへの適応性†

小国 力†† 小割健一††† 坂本隆俊††††

有限要素法により一般形状物を離散化して得た剛性行列は対称疎行列となる．この対称疎行列を係数とする連立一次方程式を解くには，LDL^T 分解法に基づく直接解法と，前処理つき反復解法が使われている．LDL^T 分解法としては対称帯行列用改訂コレスキー法と，縁どり法に基づくスカイライン法 (Envelope 法ともいう) が有利である[4]．LDL^T 分解法を使う場合には前処理としてのオーダリング法の採用が必須である．本論文ではスカイライン法を中心としてベクトルプロセッサへの適応性を評価し，その有効性を明らかにした．

1. はじめに

ベクトルプロセッサがスカラプロセッサに対してもつ利点は次の3点である[1]：

（1） ベクトル長が長いときの計算が速い

（2） 並列演算器を同時使用すると計算が速い

（3） 大量の拡張記憶装置が使える

である．縁どり法の流れをくむ対称疎行列用のスカイライン法はこの三つの特長を活かすのに相応しい方法であり，しかも，スカラプロセッサ上でもここ10年以上使用されてきている有利な方法である．スカイライン法の欠点は所要記憶容量が大きいことであるが，スカイライン法に適したオーダリング法である Reverse Cuthill-Mckee 法 (RCM 法)[2]を併用することにより所要記憶容量を小さくすることができる．

同一の対称疎行列をもつ連立一次方程式を解くには，スカイライン法のほかにも，直接解法として対称疎行列用ガウス法と対称帯行列用改訂コレスキー法があり，反復解法としては不完全コレスキー分解つき共役勾配法がある．対称疎行列用ガウス法は，オーダリング法として最小次数順序法 (Minimum Ordering Method) を用いて fill-in を少なくした上で LU 分解する．この方法は不規則疎行列（乱疎行列ともいう）向きではあるが，RCM 法経由のスカイライン法にくらべスーパコンピュータへの親和性は非常に劣る．また，RCM 法経由の帯行列用改訂コレスキー法の場合には，スカイライン法と同様な特徴をもつが，複雑な形状を有限要素法で離散化するときには，帯幅が大きくなるので，記憶容量を少なくする点ではスカイライン法の方が有効である．ただ，ベクトルプロセッサの場合は DO ループ長の増加は演算時間上あまり問題とならないので，帯行列用改訂コレスキー法の方がやや有利になる．また，長方形や直方体を離散化して得られる規則的疎行列の場合には ICCG 法が有利だが，不規則的疎行列の場合はスカイライン法の方がはるかによい．

2. スカイライン法と改訂コレスキー分解

有限要素法の剛性行列は対称疎行列で，しかも対角優位であることが多い．このために枢軸選択は不要で，行列の三角分解としては LDL^T 分解または U^{T}DU 分解に基づく改訂コレスキー法を用いる．対称疎行列向きの技法としては，縁どり法を改良したスカイライン法がよい[2]．スカイライン法では各行の最初の非ゼロ成分から対角要素までを，途中のゼロ要素を含めて，行方向に1次元配列の形で記憶装置に格納する．**図1**にスカイライン法の行列要素データの格納例を示す．図1の＊印は非ゼロ要素で，太線で囲んだ範囲を格納する．本論文で扱う行列は対称行列なので，上三角部分を考えても下三角部分で考えても同じであるが，ここでは下三角部分を用いて議論を進める．

スカイライン法における改訂コレスキー分解 LDL^T は次式で計算される：

$$\begin{cases} l_{ij}=a_{ij}-\sum_{k=s}^{j-1}\tilde{l}_{jk}l_{ik} & (1) \\ d_i=a_{ii}-\sum_{k=s}^{i-1}\tilde{l}_{ik}l_{ik} & (2) \\ \tilde{l}_{ij}=l_{ij}/d_j & (3) \end{cases}$$

ここで，s_i と s_j をそれぞれ第 i 行と第 j 行の最初の非ゼロ要素の列番号（つまり，$s_j=\min(k\,;\,a_{jk}\neq 0)$）としたとき，$s=\max(s_i, s_j)$ にとる．$\beta_j=j-s_j+1$ の和 $p=\sum\beta_j$ をプロフィル (profile) または envelope と

† An Adaptability of Skyline Method for Vector Processors by TSUTOMU OGUNI (Software Works, Hitachi, Ltd.), KEN-ICHI KOWARI (Hitachi Computer Consultant Co., Ltd.) and TAKATOSHI SAKAMOTO (Nippon Business Consultant Co., 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)
†††† (株)日本ビジネスコンサルタント

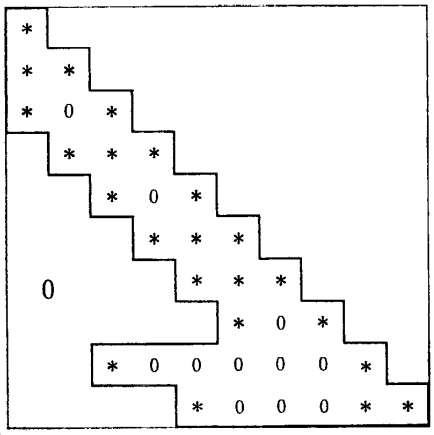

図 1 スカイライン法のデータ
Fig. 1 Data for skyline method.

表 1 スカイライン法に対するベクトルプロセッサの効果
Table 1 Effect of skyline method on vector processor.

(単位：秒)

	元数	非ゼロ要素数	プロフィル	平均帯幅	平均ベクトル長	VP	CPU 時間	
							分解	代入
1	2082	82646	320088	153	76	○	1.5	0.023
						×	12.0	0.24
2	2127	47077	269487	126	63	○	1.1	0.021
						×	6.5	0.21
3	2769	54408	485866	175	87	○	2.4	0.033
						×	20.9	0.36

VP: Vector processor

いっている．また，最大の β_i，つまり $\beta=\max\beta_i$ を最大帯幅，p を行列の元数 n で割った比 p/n を平均帯幅とよぶ．

スカイライン法での計算は第1行から始めて，第 n 行で終わる．このとき，第 i ステップにおける第 i 行の分解に必要な値は，計算済みの第1～第 $(i-1)$ 行までの分解結果であり，第 i 行の最初の非ゼロ要素である (i, s_i) 要素の列番号 s_i より若い番号をもつ列は計算には必要でない．このため，$i-s_i$ が小さい，つまり β_i が小さいと内積計算の量は少なくて済む．スカイライン法に対するベクトルプロセッサの効果を**表1**に示す．表1の平均ベクトル長は $(p-n)/n$ を意味している．

3. スカイライン法とオーダリング法の関係

スカイライン法は各行の先頭の非ゼロ要素から対角要素までを行列データとして格納する．途中のゼロ要素は三角分解計算により非ゼロ要素に変わる（このような要素を fill-in という）ことがあるので，非ゼロ要素として扱う．最終的に fill-in にならずにゼロ要素でいる割合は少なく，表1のデータ1で 0.6%，データ2で 17%にすぎない．先頭の非ゼロ要素から対角要素までの長さの和 p が小さい方が改訂コレスキー法の内積演算量が少なくなるし，記憶領域も少なくて済む．

剛性行列の行と列は，構造物をメッシュ分割したときの節点に対応している．行と列を同時に入れかえて先頭の非ゼロ要素から対角要素までの長さを短くすることができればよい．これは節点番号のつけ直し，つまりオーダリング (ordering) の問題になる．オーダリング法には三角分解のアルゴリズムに対応して種々な方法が知られているが，スカイライン法に対しては RCM 法がよく知られている．RCM 法では帯幅の縮小とプロフィルの縮小を狙っている．

RCM 法を適用した場合の効果を**表2**に示す．いずれも平均帯幅が減り，RCM 法を使えば著しい効果があることが分かる．とくに，スカラプロセッサの場合に顕著である．RCM 法がベクトルプロセッサにどう適応するかを調べると，表2で分かるとおり，かなり顕著な効果が出ている．

表 2 スカイライン法における RCM 法の効果
Table 2 Effect of RCM technique for skyline method.

(単位：秒)

	プロフィル	最大帯幅	平均帯幅	平均ベクトル長	VP	CPU 時間		
						RCM法	三角分解	合計
1	212166	166	102	50	○	0.06	0.88	0.94
					×	0.22	4.38	4.60
	320088	1134	153	76	○	—	1.55	1.55
					×	—	12.0	12.0
2	236563	230	111	55	○	0.05	0.99	1.04
					×	0.13	5.18	5.31
	269487	211	126	63	○	—	1.15	1.15
					×	—	6.53	6.53
3	224747	209	81	40	○	0.08	0.92	1.00
					×	0.16	4.40	4.56
	485866	2258	175	87	○	—	2.45	2.45
					×	—	20.9	20.9

上段：RCM 法適用，下段：RCM 法適用せず

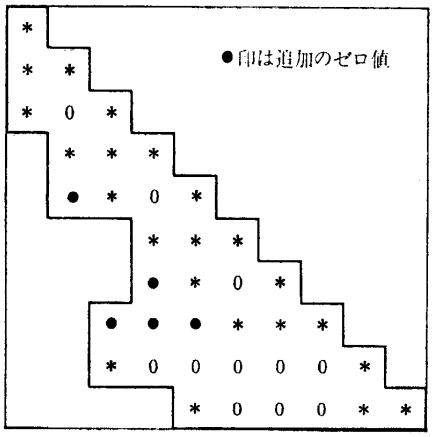

図 2 ペアドスカイライン法のデータ
Fig. 2 Data for paired skyline method.

4. 並列演算器の利用

現在のスーパコンピュータは高速計算を実行するためパイプライン方式だけでなく，複数個の演算器を同時に使用できる並列方式をとり入れている．並列演算器として4個までの乗加算器をスカイライン法で同時に使用することを考える[1]．

前に述べたように，スカイライン法では行の最初の非ゼロ要素から対角要素までを格納する．しかし，スカイライン法では一般的に各行の長さ β_i に規則性がないので，このままで2行を同時に分解することは，ベクトル長が短くなるなどベクトルプロセッサ向きでなくなる．そこで，2行を対 (pair) にして，最初の非ゼロ要素の列番号の小さな方の行に揃えて，他方の行の頭にゼロ値を追加する．図1の行列に対してこの処理をすると，**図2**の行列を得る．第1行は要素数が1なので，ペアから外してよい．

図2で新たに追加されたゼロ要素の位置は●印で示してある．この方式に従って行列要素を格納して三角分解する方法をペアドスカイライン法 (paired skyline method) とよぶことにする．このペアドスカイライン法はペアにした2行を同時に分解するときにベクトル長が短くならないのでベクトルプロセッサ向きとなる．

ペアドスカイライン法はスカイライン法より行列の格納領域が増加する．**表3**から分かるように，実験の範囲内では増加率は RCM 法を経由すると数%程度であり，RCM 法を経由しないと最大 22%であった．最近のように主記憶装置が 1M バイト単位で実装され，しかも外部記憶として拡張記憶装置を使用できる場合には，RCM 法を使うかぎりこの増加率はあまり問題にならない．

表 3 スカイライン法とペアドスカイライン法のプロフィル比較
Table 3 Comparison of profiles between skyline method and paired skyline method.

	RCM法	プロフィル		
		スカイライン法	ペアドスカイライン法	増加率 (%)
1	有	212166	215113	1.4
	無	320088	328476	2.6
2	有	236563	253513	7.2
	無	269487	278580	3.4
3	有	224747	228428	1.6
	無	485866	591029	21.6

ペアドスカイライン法での LDL^T 分解により得た行列 L の (i, j) 要素，$(i, j+1)$ 要素，$(i+1, j)$ 要素，および $(i+1, j+1)$ 要素を同時に求めるための主要な計算は次式で表される．

$$l_{ij}=a_{ij}-\sum_{k=s}^{j-1}\tilde{l}_{jk}l_{ik} \quad (4)$$

$$l_{i\,j+1}=a_{i\,j+1}-\sum_{k=s}^{j-1}\tilde{l}_{j+1\,k}l_{ik}-\tilde{l}_{j+1\,j}l_{ij} \quad (5)$$

$$l_{i+1\,j}=a_{i+1\,j}-\sum_{k=s}^{i-1}\tilde{l}_{jk}l_{i+1\,k} \quad (6)$$

$$l_{i+1\,j+1}=a_{i+1\,j+1}-\sum_{k=s}^{j-1}\tilde{l}_{j+1\,k}l_{i+1\,k}-\tilde{l}_{i+1\,j}l_{i+1\,j} \quad (7)$$

ここで，$s=\max(s_i, s_j)$ で，それぞれ s_i は i 行と $i+1$ 行，s_j は j 行と $j+1$ 行の最初の格納要素の列番号である．つまり，主要 DO ループの処理は，式(4)～(7)の中の四つの内積計算をすることになる．このため，ペアドスカイライン法は，スカイライン法にくらべて DO ループの回数が減る．式(4)と(6)を同時にするやり方を2行同時ペアドスカイライン法といい，式(4)～(6)を同時にするやり方を2行2列同時ペアドスカイライン法とよぶ．

S-810 のスカラプロセッサおよびベクトルプロセッサで各種のスカイライン法を実行したときの結果を**表4**に示す．ベクトルプロセッサ上ではペアドスカイライン法は複数の演算器を同時に使用するので，スカイライン法により計算時間が短縮されているのが分かる．ペアドスカイライン法はスカイライン法より演算量が増加するが，プロフィルの増加率が数%のときは，DO ループの回数を減らしているので，スカラプロセッサでも有効になる．しかも，スカラプロセッサではベクトルプロセッサの場合と異なり，2行同時の

表 4　2行同時分解の効果
Table 4　Effect of simultaneous decomposition with two pivotal rows.

(単位：秒)

アルゴリズム	データ	1				2				3			
	RCM 法	有		無		有		無		有		無	
	ベクトルモード	CPU 時間		CPU 時間		CPU 時間		CPU 時間		CPU 時間		CPU 時間	
		時間	比	時間	比	時間	比	時間	比	時間	比	時間	比
スカイライン法	○	0.94	4.9	1.55	7.7	1.04	5.1	1.15	5.6	1.00	4.6	2.45	8.5
	×	4.60		12.0		5.31		6.53		4.56		20.9	
ペアドスカイライン法 2行同時	○	0.58	6.6	0.92	10.9	0.65	7.9	0.67	8.7	0.63	6.0	1.86	14.5
	×	3.80		10.0		5.11		5.84		3.76		26.9	
ペアドスカイライン法 2行2列同時	○	0.45	8.8	0.65	16.2	0.51	10.1	0.51	11.5	0.49	8.0	1.29	21.7
	×	3.96		10.5		5.14		5.88		3.94		28.0	

ペアドスカイライン法が最も速くなっている.

5. ブロックスカイライン法

大規模な連立一次方程式においては係数行列が主記憶装置の容量をはるかに越えるので，ディスクを外部記憶装置として使う場合を考える.この場合,ディスクとの入出力回数を減らして，入出力に伴うオペレーティングシステムの負荷や，ディスクのアクセス待ち時間を減らすことが必要である[1]. スカイライン法で第 k 行を分解するとき，参照すべき領域は第1行から第 $(k-1)$ 行である. そこで，**図3**に示すように，もとの行列をいくつかの行ブロックに分け，最低限二つのブロックが主記憶装置に入るようにブロックの大きさを決める.

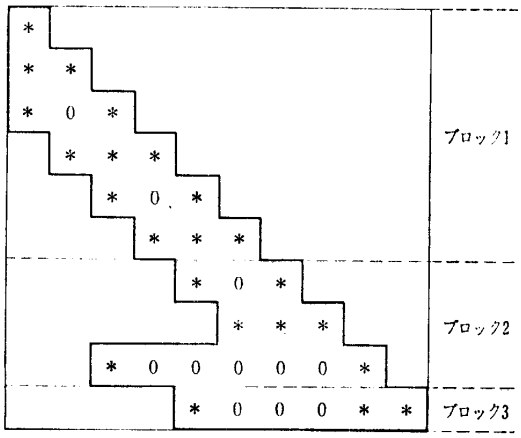

図 3　ブロックスカイライン法
Fig. 3　Block skyline method.

ブロックスカイライン法はブロックガウス法[1]と同じように，第 k ブロックの処理を2段階に分けてやる. つまり，第 k ブロックを第1～第 $(k-1)$ ブロッ

表 5　スカイライン法とブロックスカイライン法の比較（三角分解）
Table 5　A comparison with CPU time of factorization with skyline method and block skyline method.

アルゴリズム	スカイライン法	ブロックスカイライン法		ペアドスカイライン法	ブロックペアドスカイライン法	
補助記憶装置	——	ディスク		——	ディスク	
ブロック数	——	4		——	4	
時　間　(秒)	CPU 時間	CPU 時間	経過時間	CPU 時間	CPU 時間	経過時間
データ 1	1.55 12.0	1.88 12.1	6.54 16.8	0.65 10.5	0.94 11.5	5.6 16.4
データ 2	1.15 6.53	1.43 6.75	5.29 10.6	0.51 5.88	0.76 6.70	4.8 10.9
データ 3	2.45 20.9	2.94 21.2	9.72 28.2	1.29 28.0	1.78 31.0	10.1 39.5

上段はベクトルモード，下段はスカラモード

表 6 拡張記憶装置の利用による効果
Table 6 Effect with usage of extended storage equipment.

(単位：秒)

補助記憶装置		ディスク				拡張記憶			
データ	ブロック数	CPU 時間		経過時間		CPU 時間		経過時間	
		ベクトル	スカラ	ベクトル	スカラ	ベクトル	スカラ	ベクトル	スカラ
1	1	0.82	11.5	3.4	14.2	0.67	11.4	0.68	11.5
	2	0.85	11.5	4.1	15.0	0.66	11.3	0.67	11.4
	4	0.94	11.5	5.6	16.4	0.67	11.3	0.67	11.4
2	1	0.66	6.6	2.8	8.8	0.53	6.4	0.53	6.5
	2	0.70	6.6	3.6	9.5	0.53	6.4	0.53	6.5
	4	0.76	6.7	4.8	10.9	0.54	6.4	0.53	6.5
3	1	1.59	31.1	6.2	36.1	1.32	30.8	1.33	31.2
	2	1.65	31.0	7.5	37.2	1.31	30.6	1.32	30.9
	4	1.78	31.0	10.1	39.5	1.32	30.5	1.33	30.8

クによる更新計算と，第 k ブロック内での更新計算と分解である．更新計算では内積計算が行われる．ブロックスカイライン法についても，2行同時計算をすればベクトルプロセッサとの親和性を増すことができる．スカイライン法とブロックスカイライン法を同一データで実験した結果を**表 5** に示す．RCM 法を経由していないためデータ 3 ではペアドスカイライン法が不利となっている．なお，表 5 ではブロック数を 4 にしたときの結果を与えてある．

表 5 から，ディスクを使用すると，主記憶装置にスカイライン行列が入りきる場合にくらべ，経過時間が 4〜9 倍になっていることが分かる．しかし，ブロックスカイライン法を採用しない場合，経過時間はこの数の数十倍になるはずである．とくに，ブロックスカイライン法をスカラプロセッサに適用した場合には CPU 時間と経過時間の差は相対的に少なく，問題はない．

6. 拡張記憶装置の利用

ブロックスカイライン法ではブロック化した行列データを外部記憶装置に貯え，必要なデータをブロック単位で読んだり書いたりする．このとき外部記憶装置として拡張記憶装置を利用すると入出力に要する時間を短縮できる．拡張記憶は半導体で構成され，主記憶装置の外側におかれ，一時データセットとして磁気ディスク装置の代替として用いられる．その容量は 1 ギガバイトまでである．

ブロックスカイライン法で，外部記憶装置として磁気ディスクと拡張記憶装置を使い，同一データを用いて実験した結果を**表 6** に示す．ここで，データのアクセス法としてはダイレクトアクセス法を用い，RCM 法は経由していない．

ブロック数が大きくなっても拡張記憶装置の場合は経過時間に変動はほとんどないが，ディスクの場合はブロック数が増えて入出力回数が大きくなると，ベクトルモードのときの経過時間が急速に増えることが分かる．したがって，ベクトルプロセッサでは拡張記憶装置を備えることが大変望ましい．なお，S-810 の場合には，拡張記憶装置を FORTRAN 言語により用いるとき，ほぼ磁気ディスクに似た使い方でよい．

7. 帯行列用改訂コレスキー法との比較

一般の形状からなる領域を有限要素によって離散化して得た対称疎行列を帯行列として処理すると，プロ

表 7 対称疎行列用の各法の比較（不規則行列）
Table 7 Comparison with each method for symmetric sparse matrices (irregular matrix).

アルゴリズム	ベクトルモード	自由度	
		2127	2082
スカイライン法	○	0.53	0.46
	×	5.3	4.1
帯行列用改訂コレスキー法	○	0.49	0.32
	×	15.0	7.9
ICCG 法	○	2.1	4.0
	×	24.7	66.0
対称疎行列用ガウス法	○	2.8	4.1
	×	19.8	42.4

グラムは簡単になるが，記憶所要量と演算量が増えてスカラプロセッサでは不利になる．この点を解消したのがスカイライン法であることはすでに述べた．しかし，ベクトルプロセッサでは演算量の増加はあまり問題とはならないであろう．このため，RCM 法によって前処理した行列に対して，帯行列用改訂コレスキー法とスカイライン法によって比較を実施し，その結果を**表7**に示す．どちらの方法も演算器を並列に稼動させるようなアルゴリズムを採用してある．

スカラプロセッサの場合には，いずれもスカイライン法が有利だが，自由度が 2082 の場合には，表2より帯幅が 166 と小さいので帯行列用改訂コレスキー法の方がベクトルプロセッサで優位となっている．ただし，RCM 法を経由しない場合は主記憶装置が足りず，スカイライン法より早目にブロックを用いた方法に切りかえる必要が出てくる．

8. ICCG 法との比較

反復解法の ICCG 法 (Incomplete Conjugate Gradient Method) は拡散方程式を離散化して得られる1行当りの非ゼロ要素数が非常に少ない疎行列については高速に処理することが知られている．しかし，表7に取り上げた，構造解析のような一般の不規則対称疎行列についてもそれがいえるのかどうか問題である．ICCG 法は反復解法なので，計算の途中の fill-in を考慮する必要がないので，入力データと同じ作業領域があればよい．ただし，反復解法の欠点として，この作業領域が主記憶装置内に収まらないといけないという制約がある．

不規則対称疎行列に対する ICCG 法による結果を表7に，拡散方程式を有限要素法により離散化して得られた不規則対称疎行列に対する結果を**表8**に示してある．表7のデータでは1行当りの非ゼロ要素数は 22～42 であるが，表8のデータではわずかに5にすぎない．ここで，スカイライン法については RCM 法による処理込みの CPU 時間になっている．表7より一般の不規則対称疎行列の場合には，ICCG 法はそのままでは使用に耐えないことが分かる．

拡散方程式を離散化して得た対称疎行列の場合について表8よりまとめてみよう．ICCG 法はリストベクトルを使う関係上，自由度が小さいときスカラプロセッサ上では不利になる[3]が，1行当りの非ゼロ要素が少ないので，自由度が大きくなるとスカイライン法よりよい．ベクトルプロセッサ上では当然のことながら全域にわたって有利となる．スカイライン法の加速率が3倍に満たないにもかかわらず，ICCG 法では 20 倍にもなっている．

表8 より疎な不規則対称疎行列に対する比較
Table 8 Comparison with each method for symmetric irregular sparse matrices with much sparcity.
(単位：秒)

アルゴリズム	ベクトルモード	自由度 n				
		666	1193	1852	2688	3647
ICCG 法	○	0.019	0.032	0.048	0.069	0.095
	×	0.156	0.349	0.659	1.19	1.91
	比	8.2	10.9	13.7	17.2	20.1
スカイライン法	○	0.076	0.165	0.301	0.503	0.774
	×	0.117	0.297	0.616	1.15	1.96
	比	1.5	1.8	2.0	2.2	2.5

9. 対称疎行列用ガウス法との比較

LU 分解すべき行列 A の非ゼロ要素のうち，計算の途中で fill-in とならない要素を格納しないアルゴリズムを内積形式の改訂コレスキー法または対称疎行列用ガウス法という．この方法は行列が完全に不規則行列のとき有利であるが，構造解析で扱う一般の対称疎行列に対してどうなるかを検討してみた．

対称疎行列用ガウス法ではゼロ要素は主記憶装置内に格納しないで，各行の非ゼロデータだけを列番号つきで格納しておくため，リストベクトルを使うことになる．所要の記憶容量はスカイライン法にくらべ減るが，リストベクトルを使用するため，処理時間は逆に数倍かかることになり，そのままでは不利である．スカイライン法の場合にはプロフィルを少なくするために RCM 法を適用したが，対称疎行列ガウス法では最小次数順序法により fill-in の最小化をはかることができる．

表7に同一データに対してスカイライン法と対称疎行列用ガウス法の結果を与えてある．表7によると，スカイライン法にくらべかなり劣ることが分かる．また，**表9**に示すように，LDL^T 分解後に与えられる非

表9 プロフィルと非ゼロ要素数
Table 9 Profile and number of non-zero elements.

	スカイライン法	対称疎行列用ガウス法	
自由度	プロフィル	非ゼロ要素数	比 (%)
2082	320,088	318,642	99.5
2127	269,487	227,901	84.5

ゼロ要素数はスカイライン法のプロフィルに近い．しかし，RCM 法によるオーダリングのあとに作られるプロフィルにくらべ，最小次数順序法によるオーダリングのあと LDL^T 分解してできる非ゼロ要素数は 30 % 程度少なくなるのが普通である．

10. むすび

本論文では，大規模な対称疎行列を係数とする連立一次方程式をスーパコンピュータ上で解く場合に，スカイライン法がどう適応するかを示した．

偏微分方程式を離散化する場合，離散化すべき対象の形状と離散化方法によって，扱う行列の形や性格が異なってくる．このため，その行列を処理するアルゴリズムも当然変わってくる．本論文では，どのような行列に対してスカイライン法，帯行列用改訂コレスキー法，および ICCG 法がいちばん優位となるかを明らかにすることができた．

参考文献

1) 小国，後，西方，長堀：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性解析，情報処理学会論文誌，Vol. 25, No. 5, pp. 804-812 (1984).
2) 村田，小国，唐木：スーパーコンピュータ（科学技術計算への適用），丸善，東京 (1985).
3) 小国，後，長堀，加藤：スーパコンピュータにおけるリストベクトルの利用技術，情報処理学会論文誌，Vol. 27, No. 1, pp. 11-19 (1986).
4) George, A. et al.: *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs (1981).

(昭和 60 年 5 月 1 日受付)
(昭和 60 年 9 月 19 日採録)