

スカイライン法のための節点番号付け法の提案

NEW REORDERING ALGORITHM FOR SKYLINE METHOD

谷口 健男*・白石 成人**・伊藤 恭平***

By Takeo TANIGUCHI, Naruhito SHIRAISHI and Kyohei ITO

Node reordering algorithm for decreasing the matrix profile is newly proposed, and its efficiency is surveyed through a number of test examples by comparing with the results by well-known RCM and GPS algorithms. Proposed algorithm consists of three steps; Firstly any graph obtained from the objective matrix is dissected into gatherings of simple subgraphs, secondly all nodes in each subgraph are ordered independently so as to decrease the fill-ins, and finally all nodes are rearranged in order to decrease residual zero entries in the profile area. The results show that the proposed method not only requires less computation time comparing to the GPS algorithm but also can give better reorderings, especially for systems with rather complex topological structures.

1. 序 論

近年、有限要素法や差分法といった数値解析で扱われる系は、ますます巨大化・複雑化し、また解の精度向上のため非常に多くの離散点を有するようになり、その結果導き出される大次元連立 1 次方程式をいかに効率よく解くかが重要な問題となってきている。いま、問題を線形な荷重-変位関係に限ると、解くべき方程式は次式で与えられる。

$$P=K \cdot x \dots\dots\dots(1)$$

ここで、 P は既知の外力ベクトル、 x は求めるべき節点変位ベクトル、 K は剛性行列である。一般に、マトリックス構造解析で現われる行列 K は、① 非常に多くの零要素を含む疎行列であり、② 正定値・対称性を有する場合が多い¹⁾。特に①の性質は、容量・演算時間の大幅な節約を可能にし、これを利用して、① 帯行列法、② スカイライン法、③ 疎マトリックス法等の計算法が開発されている。帯行列法は、図-1(a) に示す半帯幅(HBW)の領域に対し演算を行う方法で、プログラムが

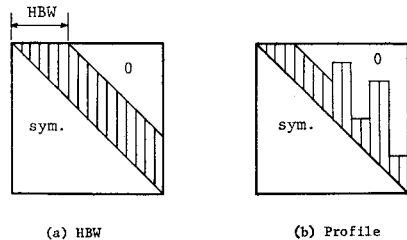


図-1 半帯幅とプロフィール

簡単なわりに能率がよく、小、中規模の問題に関しては最もよく利用されている。スカイライン法は、図-1(b) に示すプロフィール領域に対し演算を行う方法である。プロフィール領域とは、行列内の非零要素と消去過程において非零化する可能性のある要素であり、容量的にみると、帯行列法より有効で、近年よく利用されるようになってきた。これに対し、疎マトリックス法は、行列内の非零要素と消去過程において非零化した要素に対し演算を行う方法で、計算手順は複雑であるが、容量的には最も優れた方法である。よって、内部記憶容量を越えるような大次元問題の解法の 1 つであるウェーブフロント法では、この方法がよく用いられる。これら諸手法を用いたときの容量および演算時間を算定するには評価式が必要であり、以下においてそれを与える。

* 正会員 工博 岡山大学助教授 工学部 (〒700 岡山市津島中 3-1-1)
 ** 正会員 工博 京都大学教授 工学部 (〒606 京都市左京区吉田本町)
 *** 学生会員 岡山大学大学院 (〒700 岡山市津島中 3-1-1)

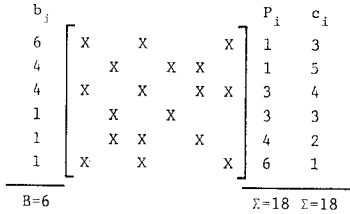


図-2 半帯幅とプロファイル (例)

式 (1) の係数行列 K を正定値・対称・疎であるとする。行列 K の i 行の最終非零要素の列番号を j とすると、半帯幅 B は、次式で表わされる。

$$B = \max_{i \leq N} (j - i + 1) = \max_{i \leq N} b_i \quad \dots\dots\dots (2)$$

また、行列 K の i 列の最初非零要素の行番号を j とすると、プロファイル P は、次式で表わされる²⁾。

$$P = \sum_{i=1}^N (i - j + 1) = \sum_{i=1}^N P_i \quad \dots\dots\dots (3)$$

ここで、 i 行の active column 数を C_i で表わす。active column 数 C_i とは、第 i 行の要素のうち $a_{ij} \neq 0$ ($j > i$) である要素数と、 $a_{ij} = 0$ であっても $a_{kj} \neq 0$ ($k < i, j > i$) である要素数の和を意味する。行列 K は対称であるので、式 (3) は、次のようになる (図-2 参照)。

$$P = \sum_{i=1}^N C_i \quad \dots\dots\dots (4)$$

いま、平均フロント幅 (W_{avg}) と 2 乗平均フロント幅 (W_{rms}) を次のように定義する²⁾。

$$W_{avg} = \frac{1}{N} \sum_{i=1}^N C_i \quad \dots\dots\dots (5)$$

$$W_{rms} = \sqrt{\frac{1}{N} \left(\sum_{i=1}^N C_i^2 \right)} \quad \dots\dots\dots (6)$$

直接法 (消去法) は、前進消去と後退代入の 2 つの部分に分けられ、それぞれに要する演算時間を T_1, T_2 とすると、次のように表わされる²⁾。

$$T_1 = \frac{1}{2} N \cdot W_{rms}^2 (T_0) \quad \dots\dots\dots (7)$$

$$T_2 = \frac{1}{2} N \cdot W_{avg} (T_0) \quad \dots\dots\dots (8)$$

ここで、 N は行列 K の次数、 T_0 は機種による定数である。 T_2 は T_1 に比べて無視できる程度であるので、演算時間の評価においては一般に、 T_1 だけが考慮される。式 (6) を代入すると、 T_1 は次のように表わされる。

$$T_1 = \frac{1}{2} \sum_{i=1}^N C_i^2 (T_0) \quad \dots\dots\dots (9)$$

帯行列法においては、 $C_i = B$ として計算するため、 T_1 は次式で表わされる。

$$T_1 = \frac{1}{2} N \cdot B^2 (T_0) \quad \dots\dots\dots (10)$$

表-1 容量と演算時間

	Band Matrix	Skyline	Wavefront
Memory	NB	$P = \sum_{i=1}^N C_i$	$W = \max_{i \leq 1} \tilde{C}_i$
Execution-time	$T = \frac{1}{2} NB^2 (T_0)$	$T = \frac{1}{2} \sum_{i=1}^N C_i^2 (T_0)$	$T = \frac{1}{2} \sum_{i=1}^N \tilde{C}_i^2 (T_0)$

表-2 Reordering のための評価式

	Band Matrix	Skyline	Wavefront
Memory	B	$\sum_{i=1}^N C_i$	$\max_{i \leq N} \tilde{C}_i$
Execution-time	B	$\sum_{i=1}^N C_i^2$	$\sum_{i=1}^N \tilde{C}_i^2$

次にウェーブフロント法について考えてみる。行列 K の i 行において、消去前・後を通じて零のものを $|Z_i|$ で表わすと、ウェーブフロント法における active column \tilde{C}_i は、

$$\tilde{C}_i = C_i - |Z_i| \quad (i=1, 2, \dots, N) \quad \dots\dots\dots (11)$$

となり、最大フロント幅 W_{max} 、 \tilde{W}_{avg} 、 \tilde{W}_{rms} を、次のように定義する。

$$W_{max} = \max_{i \leq N} \tilde{C}_i \quad \dots\dots\dots (12)$$

$$\tilde{W}_{avg} = \frac{1}{N} \sum_{i=1}^N \tilde{C}_i \quad \dots\dots\dots (13)$$

$$\tilde{W}_{rms} = \sqrt{\frac{1}{N} \left(\sum_{i=1}^N \tilde{C}_i^2 \right)} \quad \dots\dots\dots (14)$$

これより、 T_1 は次式で表わされる。

$$T_1 = \frac{1}{2} \sum_{i=1}^N \tilde{C}_i^2 (T_0) \quad \dots\dots\dots (15)$$

以上の結果をまとめると、表-1 のようになり、したがって、帯行列法、スカイライン法、ウェーブフロント法を有効に利用するために、考慮しなければならない要因は、表-2 のようになる。これら演算時間および容量は節点の番号付けに依存することは明らかであり、したがって、それぞれに対して、節点番号付け法 (Reordering) が必要となる。

本論文では、上述した諸手法のうち、特にスカイライン法を用いたときに必要とされる容量の減少を目的とした節点番号付け法の開発を主目的とする。さらに理論的考察より、その手法が、ウェーブフロント法のデータ前処理としても有効であることを示すとともに、数値実験により、その妥当性の確認を行う。さらに、本手法を利用することにより、任意の系が与えられたとき、それに対して最適な数値計算法の選択も可能となることを示す。すなわち、本手法は帯行列法を適用してもよい系に対しては、帯幅を小さくするような番号付けも可能となる。なお、本手法は対象系が 2 次元、3 次元のいずれに対しても有効である。

2. スカイライン法と、その利用に関する問題点

スカイライン法を用いるときの入力データ量を支配するのは、前節で述べたように、式(4)で表わされるプロフィールである。このプロフィール領域内の非零要素の数は、行・列を入れかえても不変である。よってプロフィールの最小化は、プロフィール領域内の零要素の最小化となり、零要素は L_0 で表わすと、次のようになる。

$$\min P \Rightarrow \min L_0 \dots\dots\dots(16)$$

ここで、前進消去過程において、消去前零であったものが、消去後、非零化するものをフィルインとよび、その個数を $|F|$ で表わし、消去前・後を通じて零のものをゼロとよび、その個数を $|Z|$ で表わすと、 L_0 は次のようになる。

$$L_0 = |F| + |Z| \dots\dots\dots(17)$$

式(16)より、スカイライン法における入力データ量の最小化、すなわちプロフィール最小化問題は、次式で表わされる。

$$\min P \Rightarrow \min (|F| + |Z|) \dots\dots\dots(18)$$

一方、行列 K の第 i 行の消去演算により、 (j, l) 要素は、次式のように修正される。

$$\tilde{k}_{jl} = k_{jl} - k_{ji} \cdot k_{il} / k_{ii} \dots\dots\dots(19)$$

上式中“ \sim ”は i 行の消去演算後の (j, l) 要素の値を示す。上で述べたフィルインとは、

$$k_{jl} = 0 \text{ かつ } \tilde{k}_{jl} \neq 0 \dots\dots\dots(20)$$

を満たす (j, l) 要素をいう。

以下では、D.J. Rose による節点消去の概念を用いる³⁾。すなわち、上三角行列内に m 個の非対角非零要素を有する対称行列 $K(n \times m)$ は、 m 個の線と n 個の節点を有するグラフに置き換えられ、第 i 行の消去は、第 i 点を消去し、 i 点に隣接していた節点どうしを互いに線で結ぶことに等価となる。このように、互いに結びつけられた節点集合を $FRONT_i$ とよぶことにする。いま、図-3(a)に示されるグラフを領域 I, II, III の順序で節点消去を行ったとする。これを、行列で示したものが、図-3(b)であり、斜線領域が Z である。したがって、 $|Z|$ は次式で評価される。

$$|Z| \propto (\text{IIの節点数}) \times (\text{IIIと隣接するI内の節点数}) \dots\dots\dots(21)$$

式(21)より、 $|Z|$ を小さくするには、領域 II もしくは、(IIIと隣接するI内の節点数)を小さくすればよい。さらに、もし次に消去すべき点 v_i ($i=1, 2, \dots, \lambda$) として、常に、

$$v_i \in FRONT_{i-1} \dots\dots\dots(22)$$

より選べば、 $|Z|=0$ となることは明らかである。一方、 v_i の消去によるフィルイン数 $|F_i|$ は、次式で評価される⁴⁾。

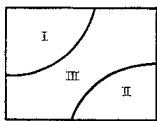
$$|F_i| \propto |FRONT_i|^2 \dots\dots\dots(23)$$

したがって、フィルインを小さくするためには、 $FRONT$ に含まれる節点数を常に小さくしておくことが、必要である。

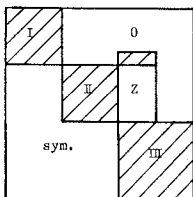
以上の、 $\min |F|$, $\min |Z|$ に関する考察より、 $\min (|F| + |Z|)$ を満たすような消去順序を探すことは非常に困難であると考えられる。なぜならば、 $\min |Z|$ を目的とするならば、式(21)より $|Z|=0$ が最小であり、このためには、図-3(a)のような消去をしてはならない。しかしながら、このような消去を行うことにより、 $\min P$ が得られる例を容易にみつけることができる(図-4参照)。一方、 $\min |F|$ を目的とするならば、式(23)より、 $FRONT$ の大きさを小さくさえすればよいが、これを厳密に行えば、たとえば A. George の提案した Nested Dissection となり⁵⁾、この場合、非常に大きな $|Z|$ が発生する。このように、これら両者を、共存させるには非常に大きな困難さが予想される。

一方、たとえば図-4に示されるような場合を考えれば、グラフの突出領域を、図-4(a)に示した領域 I または II ととらえることにより、少なくとも、プロフィールの減少が可能である。これにより、式(18)の代わりに、次式を扱えばよいことになる。

$$\min P \Rightarrow \min |F| + \min |Z| \dots\dots\dots(24)$$

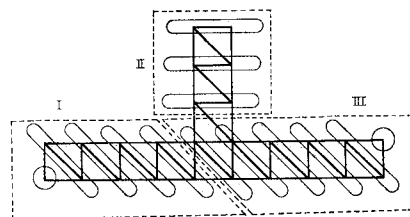


(a) Graph

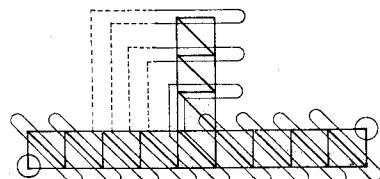


(b) Matrix

図-3 z の発生



(a) Profile Minimization



(b) Bandwidth Minimization

図-4 突出領域を有するグラフ

たとえば、図-4 の I, II, III といった各部分領域に関して、 $\min |F|$ を目的とした消去順序を考え、それらの組合せに関しては、 $\min |Z|$ を目的とする。これが、本論文で提案しようとするプロフィール減少のための基本的な考えである。一方、従来よりよく用いられてきた、たとえば、RCM, GPS 法においては^{5)~7)}、主たる目的は、いわゆる半帯幅の減少であり、したがってグラフのいわゆる幅を小さくせしめることにあり、上述した考えとは全く異なるものである (図-4(b))。

ここまでの、 $P = \sum_{i=1}^N C_i$ の最小化を扱ってきたが、これより近似的に、演算時間の最小化、つまり $\sum_{i=1}^N C_i^2$ の最小化も行え得ると考えられる。次にスカイラインとウェーブフロント両手法との関係を探ってみる。 C_i と \tilde{C}_i の関係は、式 (11) で与えられ、これより $\min \tilde{C}_i$ を行うには、フィルイン最小化を行えばよいということであり、式 (24) において、 $\min |Z|$ を無視すればよいということである。上述した $\min P$ の方法では、逆にこれを入れているにすぎないと考えられ、したがって、 $\min P$ の手法がウェーブフロント法の節点番号付けとしても、ある程度有効であると考えられる。

次節においては、この新たな考え方に従ったアルゴリズムの提案を行い、さらに、さまざまな数値実験により、RCM, GPS 法による結果との比較を行い、本手法の有効さを確認する。

3. プロファイル減少アルゴリズムの提案

前節での考察に基づき、新たなプロフィール減少アルゴリズムの提案を行うが、その考え方に基づくプロフィール減少法は、すでに谷口らにより、アルゴリズム化されている。したがって、まずその方法の概略を示す⁸⁾ (図-5 参照)。グラフの長手軸の1端 S を探し出し、 G のすべての点を S よりの等距離節点集合 (level structure)。

$$\mathcal{L} = \{L_0 = S, L_1, L_2, \dots, L_i, \dots, L_n\}$$

$$\left. \begin{aligned} \text{adj. } L_0 \in L_1, \text{ adj. } L_n \in L_{n-1} \\ \text{adj. } L_i \subset L_{i-1} \cup L_{i+1} \end{aligned} \right\} \dots\dots\dots (25)$$

に分類する⁹⁾。なお上式中、adj. は隣接を意味する。いま、 L_i がグラフの外周辺に位置する節点 (これを境界節点とよぶ) を3個以上含むと仮定すれば、図-5 に示

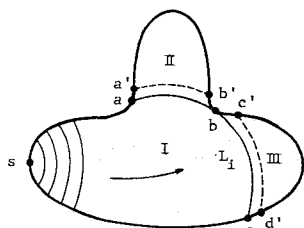


図-5 突出領域の判定

されるように、 L_i によってグラフが領域 I, II, III に3分割され、I → II → III の順に番号付けを行う。この方法は、突起をもつ構造、有孔板等に関しては、従

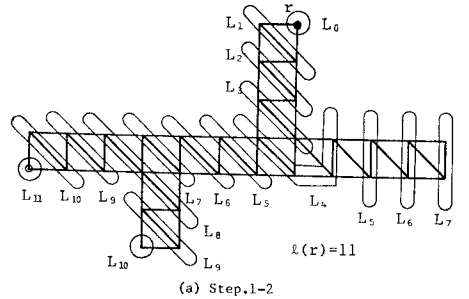
来の方法 (RCM 法, GPS 法) と比べてよい結果を得ている。しかし、次のような問題点がある。

- (1) 入力データとして、境界節点を認識させなければならぬ。
- (2) 2次元のモデルにしか適用できない。

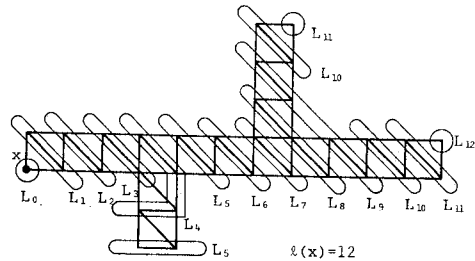
本研究では、これら2点の欠点を克服し、入力データとしては、RCM, GPS 法と同様、節点数、節点の最大次数、節点の隣接関係とし、3次元モデルにも適用できる汎用性のあるアルゴリズムを提案する。

上記欠点のうち、入力データとしての境界節点は、上述した手法の概略よりも明らかなように、グラフの突起の存在の有無の判定に用いられている。すなわち、1つのレベル L_i に3個以上の境界節点を含めば、図-5 に示されるように、突出部が存在することが明らかである。本論文で提案するアルゴリズムでは、これを次のように改良する。

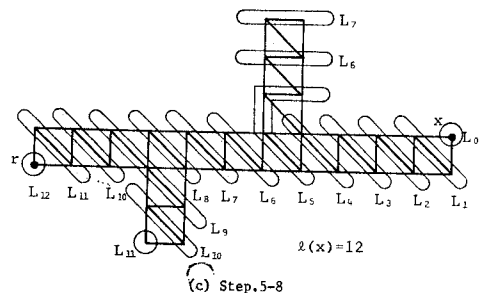
図-5 のグラフの長手軸の一端 S より level structure を作成すると、突起部においては、そのレベルが



(a) Step.1-2



(b) Step.3-6



(c) Step.5-8

図-6 Algorithm 1

a'—b', c'—d' のように分離グラフとなる. このように分離グラフが発生するのは, 突起部あるいは穴まわりだけである. さらに, この考え方は, 3次元グラフに対しても有効であることは明らかであり, 上記 (2) の欠点も克服できることになる.

新たな方法は大きく分けて, 次の3つの部分から成り立っている.

- (1) 出発点の決定.
- (2) グラフを凸な部分グラフの集合に置換.
- (3) 番号付け.

(Algorithm 1) (図—6 参照)

A. George の修正による Pseudo-Peripheral Node⁵⁾を用いて, グラフの長手軸の1端を求める.

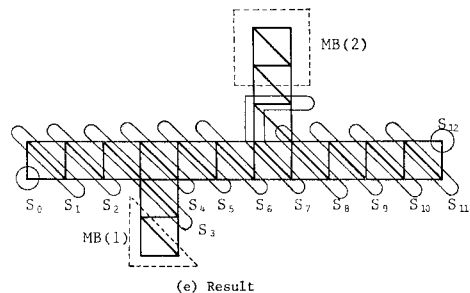
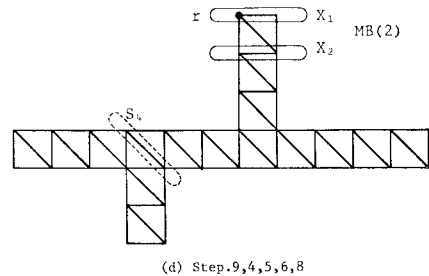
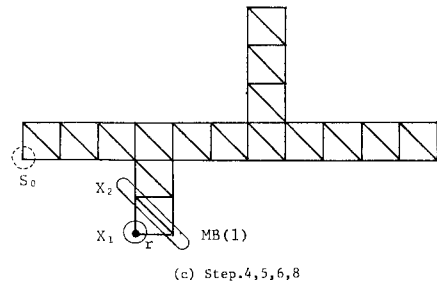
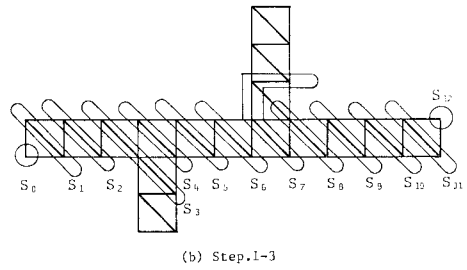
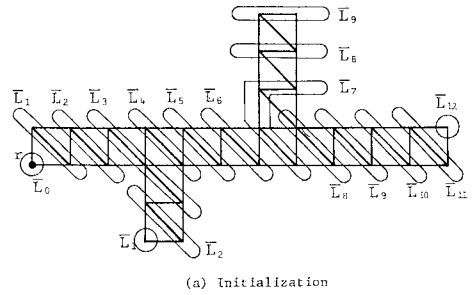
- Step. 1 最小次数の点 r を探す.
- Step. 2 r より level structure を作成する.
($l(r)$: レベルセット数)
 $\mathcal{L}(r) = \{L_0(r), L_1(r), \dots, L_{l(r)}(r)\}$
- Step. 3 最終レベルセット, $L_{l(r)}(r)$ 内のすべての連結グラフ C_I ($I=1, 2, \dots, l$) を探す. (l : 連結グラフ数)
- Step. 4 $I=1$
- Step. 5 C_I の最小次数の点 x より level structure を作成する. ($l(x)$: レベルセット数)
 $\mathcal{L}(x) = \{L_0(x), L_1(x), \dots, L_{l(x)}(x)\}$
- Step. 6 $l(x) > l(r)$ ならば, $r=x$ とおき, go to Step. 5
- Step. 7 $I < l$ ならば, $I=I+1$ とおき, go to Step. 3
- Step. 8 r が出発点である.

以上で, x を始点として作成した level structure $\mathcal{L}(x) = \{L_1, L_2, \dots, L_k\}$ と, 出発点 r が決定できた.

(Algorithm 2) (図—7 参照)

$\mathcal{L}(x)$ を用いて, 終点 r より逆に, “隣接する節点集合列”を作成することにより, グラフを凸な部分(main graph) と, 突起部分(sub graph) の2つに分割する. $\mathcal{L}(x) = \{L_0, L_1, \dots, L_k\}$ を $\bar{\mathcal{L}} = \{\bar{L}_k, \bar{L}_{k-1}, \dots, \bar{L}_0\}$ と置き換える (図—7(a)).

- Step. 1 $I=0$, 出発点 r を含む \bar{L}_I を S_I とする.
- Step. 2 S_I の1点に隣接する \bar{L}_{I+1} 内の1点を探し, その点を含む \bar{L}_{I+1} の連結グラフを探し出し, それらを S_{I+1} とする. この操作を繰り返すことにより, level structure S を得る.
 $S = \{S_0, S_1, S_2, \dots, S_k\}$
なお, レベル数 k は, $\bar{\mathcal{L}}$ のレベル数に等しい.
- Step. 3 グラフのすべての点が S に含まれる場合.
go to Algorithm 3



図—7 Algorithm 2

- 含まれない場合, それらを $V(G)$ とする.
- Step. 4 $J=0, 1, \dots$ の順に, \bar{L}_J に含まれるが, S_J に含まれない点 (すなわち $V(G)$ の点) を探し, あれば, その点を r とする.
- Step. 5 $K=0$. \bar{L}_J に含まれる節点のうち, r を含む連結グラフを探し, それを X_0 とする.
- Step. 2 と同様に X_k と \bar{L} を用いて, level structure を作成する.
- $$X = \{X_0, X_1, \dots, X_\alpha\}$$
- Step. 6 [穴の存在の判定]
- もし, $X_0 (\ni r)$ が \bar{L}_{J-1} と隣接していなければ, go to Step. 8
- Step. 7 もし, $\sum_{i=0}^{\alpha} |X_i| > \sum_{l=J}^{j+\alpha-1} |S_l|$ であれば, $\{X\}_\alpha$ と $\{S\}_j^{j+\alpha-1}$ の内容を入れ替える.
- Step. 8 [突起の判定]
- $$\sum_{i=0}^{\alpha} |X_i| = |X|$$
- とおく.
- もし, $\alpha < |X|^P$ であれば, X の各レベル X_i を S のレベル S_{j+1} に加える.
- $\alpha \geq |X|^P$ であれば, X を MB に保存する.
- Step. 9 $V(G) = \emptyset$ であれば, go to Algorithm 3
- $V(G) \neq \emptyset$ であれば, go to Step. 4

ここで, 問題となるのが, 穴の存在の判定および突起の判定である. 突起の判定に関しては, どの程度の突起を分岐とするかという点であり, これは, 小さな突起を分岐とすると, 逆に L_0 が増大するからである. sub graph において, α : レベルセット数, $|X|$: 節点数, P : パラメーターとして, 次式を満たす場合, 分岐とはしないで, sub graph を main graph に組み込む. なお, P は数値実験より求める.

$$\alpha < |X|^P \dots\dots\dots (26)$$

また, (図-8) のような有孔板の場合, sub graph と, それと同じ \bar{L}_J に属する main graph の節点数を比較して, sub graph の方が大である場合は, 入れかえる. (Algorithm 3) (図-9 参照)

- Step. 1 S のすべての要素を, その順に C に入れる.
- Step. 2 X の最終レベルに隣接する S の節点で, C

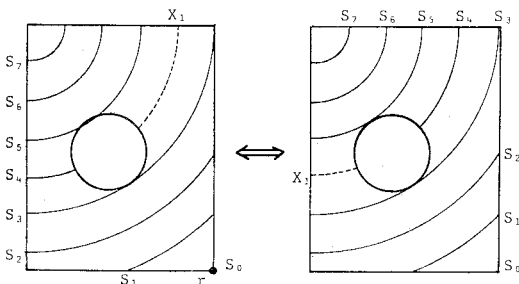
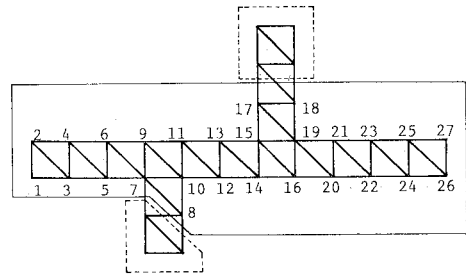
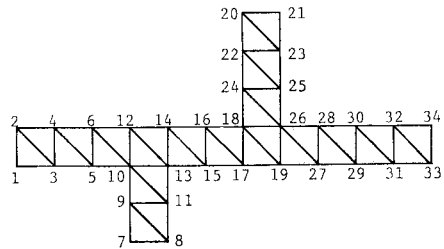


図-8 穴の処理



(a) Step.1



(b) Step.2,3

図-9 Algorithm 3

において最小番号をもつ節点を探し出す. これを $C(\beta)$ とする.

- Step. 3 X をその順序のまま, C 内の $\beta-1$ と β との間に挿入する.

すべての sub graph に対して, Step. 2, 3 を繰り返す.

以上のアルゴリズムのウェーブフロント法のための消去順序付けへの適用法を示す. ウェーブフロント法に必要なのは, 要素番号順であり, また入力データは, 要素-節点関係である. よって, まず要素-節点関係より, 節点の隣接関係を作成し, 本手法を用いて, 節点番号順を決め, 次に, 節点-要素関係を用いて, 要素番号順を決める.

表-4 NEW, GPS, MRCM 法による Reordering の結果比較

Case. No.	No. of Nodes	No. of Elements	Skyline					
			W_{avg}			W_{rms}		
			MRCM	GPS	NEW	MRCM	GPS	NEW
1	42	64	7.9	6.0*	6.2	8.5	6.3*	6.7
2	47	62	5.8	5.7	5.6*	6.1	6.0	5.8*
3	59	129	6.2	5.4	4.4*	6.7	5.8	4.6*
4	99	163	9.8	8.6*	8.9	10.2	8.8*	9.3
5	136	217	9.2	8.3*	8.4	9.9	8.8*	8.9
6	148	246	12.9	11.4*	12.2	13.7	11.9*	13.1
7	193	319	14.9	12.6*	12.8	15.7	13.0*	13.5
8	232	397	15.1	13.7*	13.7*	15.7	14.1	14.0*
9	374	647	16.9	16.4	15.6*	17.7	17.1	16.4*
10	414	722	20.1	19.6	18.8*	20.8	20.5	19.6*
11	157	468	14.2	12.3*	14.1	15.6	12.9*	15.0
12	211	636	16.7	15.6	14.4*	18.1	16.2	15.5*
13	312	942	19.2	17.9	15.2*	20.5	19.0	16.4*

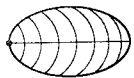
* Greatest Reduction

4. 数値実験および考察

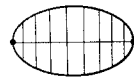
前節のアルゴリズムに基づき、実際にプログラムを作成し、数値実験を行った結果を示す。比較の対象としては、今日よく利用されている RCM 法を大幅に修正した方法(これを MRCM とよぶ)と GPS 法を用いた²⁾。これらの手法の概略を示す。RCM 法は、式(27)を満たす点より出発して、level structure を作成し、level structure 内においては、最小次数の点より逆順に番号付けを行い、最小の帯幅を与えるものを採用する(図-10(a))。

$$D_{min} \leq D \leq D_{min} + \frac{1}{2} D_{max} \dots\dots\dots(27)$$

D: 次数, D_{min}: 最小次数, D_{max}: 最大次数



(a) RCM



(b) GPS

図-10 RCM 法と GPS 法

しかしながら、この方法では、式(27)を満たす点の回数だけ番号付けを行うため、多くの演算時間を必要とする。そのため、本論文では、(Algorithm 1)を用いて出発点を探し出す手順を RCM 法に追加することにより、演算時間の短縮を図った方法(MRCM 法)を開発し、これを用いた³⁾。GPS 法は、長手軸の両端から作成した2つの、level structure を用いて、level structure

が長手軸に直交するように修正を行った方法である。GPS 法は RCM 法(式(27)を用いた場合)と同等の結果を得ることが知られており、現在、最も優れた方法とされている²⁾(図-10(b))。

2次元、3次元に対するパラメーター P と、上述した W_{avg} の関係は、表-3 のようになり、この結果、2次元の場合 P=1/2, 3次元の場合 P=1/4 が最適と考えられ、以後この値を用いる。数値実験結果の評価は、スカイライン法は W_{avg}, W_{rms}, 帯行列法は B, ウェーブフロント法は W_{max}, \tilde{W}_{rms} で行う。ここで示したアルゴリズム(NEW), MRCM, GPS との比較結果は表-4 のとおりであり、Case. 1~10 は2次元モデル、Case. 11~13 は3次元モデルである。また、CPU time は Reordering に要する演算時間、No. of Subgraphs は分岐の数、Shape はグラフの形状を表わし、全体的に凹凸のない構造(Convex)、突起をもつ構造(Concave)、穴を有する構造(Hole)であり、次のような結果を得

表-3 パラメーター P による W_{avg} の比較

Dim.	N	P			
		2	1/2	1/3	0
2-Dim.	59	5.6	4.4	4.4	4.4
	374	16.3	15.6	15.6	15.5
	414	19.2	18.8	19.0	19.0
3-Dim.	211	16.7	14.4	14.4	14.4
		312	16.3	15.6	15.7

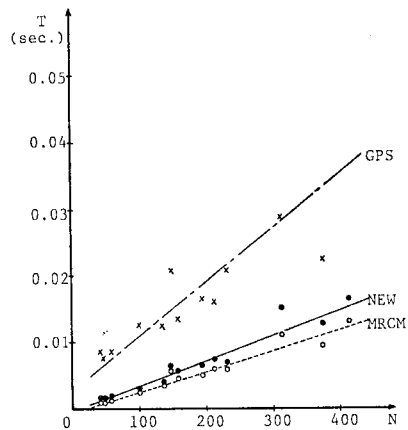
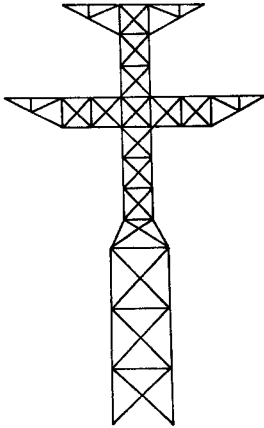


図-11 Reordering による演算時間の比較

表-4 (続) NEW, GPS, MRCM 法による Reordering の結果比較

MRCM	Wavefront						Band Matrix			CPU time			No. of Subgraphs	Shape
	W _{max}			\tilde{W}_{rms}			B			($\times 10^{-4}$ s)				
	GPS	NEW	MRCM	MRCM	GPS	NEW	MRCM	GPS	NEW	MRCM	GPS	NEW		
12	9*	10	11.1	8.2*	8.9	14	8*	13	10.2	85.3	12.7	0	Convex	
8	8	7*	7.3	7.2	6.8*	9	7*	15	11.1	76.4	17.3	1	Hole	
10	9	8*	9.7	8.3	6.7*	15	8*	19	13.9	85.3	20.3	2	Concave	
14	11*	13	13.5	11.5*	12.4	17	11*	15	24.5	126.1	30.0	0	Convex	
16	14*	15	12.3	11.4*	11.4*	25	18*	26	35.0	123.3	40.9	0	Convex	
18	15*	20	17.6	15.6*	17.3	33	16*	21	58.4	205.3	68.1	0	Convex	
24	18*	21	20.1	16.9*	17.8	36	19*	75	49.8	164.4	68.9	1	Hole	
21	19*	19*	20.3	18.7*	18.7*	32	19*	23	58.4	206.4	76.9	0	Convex	
25	22*	24	23.6	22.8	21.9*	36	23*	152	94.1	222.8	127.0	2	Hole	
29*	29*	29*	27.2	27.3	26.2*	44	29*	40	131.1	463.3	183.0	1	Hole	
26	19*	28	26.9	22.2*	26.2	45	18*	37	45.6	135.5	55.0	2	Concave	
31	23*	27	31.1	28.1	27.0*	51	22*	60	60.3	158.7	77.2	2	Concave	
35	30	27*	35.1	32.9	28.3*	56	35*	237	112.7	286.7	201.6	4	Concave	



Case. 3 59 Nodes

図-12 適用例 (その1)

る。

(1) Reordering に要する時間を、節点数を横軸に、演算時間を縦軸にしてプロットすると、図-11 のようになり、本手法は今日最も高速であると考えられる MRCM 法の 1.2 倍程度である。

さらに本手法によれば必要な演算時間は節点数にほぼ比例して増加し、GPS 法のようなばらつきはない。

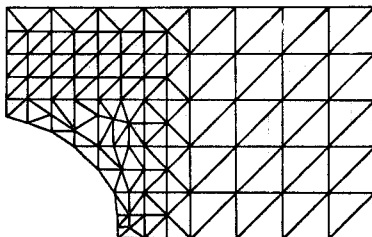
(2) スカイライン法のための Reordering 法としては、本手法は、図-12 に示すような分岐を有する系に対して、容量、演算時間とも最良の結果を与えている。

(3) ウェーブフロント法のための Reordering 法としては、本手法は、容量に関して GPS 法と同程度によく、演算時間は分岐を有する系に対してよい結果を与えている。

(4) 帯行列法のための Reordering 法としては、明らかに GPS 法が最良の結果を与えている。しかし、図-13 に示すような分岐を有しない系に対して、本手法は、MRCM 法と同等またはそれ以上のよい結果を与えている。

以上の結果は、GPS 法と本手法の両者の基本アルゴリズムの差異を明確に示しており、単純な系に対しては GPS 法が、複雑な系に対しては、本手法が適しているといえる。また、単純な系 (分岐を有しない系) に対して、本手法は MRCM 法と同様のアルゴリズムとなるため、帯幅をも減少させ得ることが可能である。

次に、Reordering を行った後、実際に、帯行列法およびスカイライン法を用いて解いた場合の演算時間の比



Case. 4 99 Nodes

図-13 適用例 (その2)

表-5 Band Matrix 法と Skyline 法の演算時間の比較

Case. No.	No. of Nodes	GPS				NEW			
		B	CP time ($\times 10^{-4}$ s)			W_{rms}	CP time ($\times 10^{-4}$ s)		
			Reordering	Band Solver	sum.		Reordering	Skyline	sum.
3	59	8	85.3	53.4	138.7	4.6	20.3	19.2	39.5
4	99	11	126.1	153.3	279.4	9.3	30.0	89.2	119.2

較を 表-5 に示す。解析モデルとしては、図-12, 13 に示す典型的な 2 ケースを用いて、GPS 法を用いて帯行列法に適用した場合、本手法を用いてスカイライン法に適用した場合を示した。これより、明らかにスカイライン法の方がよい結果を与えている。容量的にも演算時間的にもスカイライン法は帯行列法と比べて有効な手法であるといえる。なお、これらの計算には、岡山大学総合情報処理センター ACOS 1000 を用いた。

5. 結 論

本研究により得られた成果は下記のとおりである。

(1) プロファイルの減少は文献 8), 9) に指摘されるように、対象となるグラフを凸な部分グラフの集合に分割することにより行えることが具体的に示された。

(2) 新たなプロファイル減少法が提案され、それが十分実用に供し得るものであることが、計算例より確認された。

(3) この手法を用いることにより、系のグラフが分岐を有するかどうかの判定が可能となり、したがってもし対象が分岐のない系と判定された場合、その Reordering 結果は帯行列法のための Reordering としても十分有効であることが判明した。

(4) さらに、本手法はウェーブフロント法のための Reordering 法としても活用できることが示された。

参 考 文 献

- 1) 鷺津久一郎, ほか: 有限要素法ハンドブック I, 基礎編, 培風館, pp. 58~93, 1982.
- 2) Everstine, G.C.: A comparison of three resequencing algorithms for the reduction of matrix profile and wavefront, International Journal for Numerical Methods in Engineering, 14, pp. 837~853, 1979.
- 3) Rose, D.J.: A graph-theoretic study of the numerical solution of sparse positive-definite systems of linear equations, Graph Theory and Computing, R.C. Read ed., Academic Press, pp. 183~217, 1972.
- 4) Shiraishi, N. and Taniguchi, T.: Design of Algorithms for Reducing the Bandwidth and Profile of Sparse Matrix, Proc. of 2nd International Congress of Numerical Methods for Engineering, (Edited by E. Absi et al.) DUNOD, pp. 187~196, 1980.
- 5) George, A.: Solution of linear systems: Direct methods for finite element problems, Sparse Matrix Techniques, V.A. Barker ed., Lecture Note in Mathematics 572, Springer Verlag, pp. 52~101, 1976.
- 6) Cuthill, E. and McKee, J.: Reducing the bandwidth

- of sparse symmetric matrices, Proc. of ACM National Conference, pp. 157~172, 1969.
- 7) Gibbs, N.E., W.G. Poole and P.K. Stockmeyer : An algorithm for reducing the bandwidth and profile of a sparse matrix, SIAM J. Numer. Anal., 13, pp. 236~250, 1976.
- 8) Taniguchi, T., Shiraishi, N. and Ito, K. : Profile Minimization Algorithm for Sparse Matrix, Advances in Engineering Software. (印刷中)
- 9) 白石成人・谷口健男 : マトリックス構造解析に対するグラフ理論による一考察, 土木学会論文報告集, 第294号, pp. 15~24, 1980.
- (1983.8.22・受付)
-