境界適合ボクセル要素の開発

正員 鈴木克幸* 正員 大坪英臣*
閔勝載* 上杉忠輝**

The Development of Boundary Fit Voxel Element

by Katsuyuki Suzuki, *Member* Hideomi Ohtsubo, *Member*
Seungjae Min, Tadateru Uesugi

Summary

In the analysis of 3 dimensional solid structure, the mesh generation is always the most time consuming and sometimes makes the analysis impossible. Here, the authors have developed the boundary fitted voxel element to reduce the cost of mesh generation in 3D solid analysis. The shape functions that satisfy the continuity between elements are proposed, and integration method to compute the stiffness matrix, mass matrix and load vector of complicated shape is proposed. The element are implemented to MSC/NASTRAN as user defined element, and the unified numbering method of element and nodes for voxel elements are proposed. Also, the postprocessing technique, which does not use the finite element mesh, is proposed, and implemented using VRML. The analysis examples of static and eigenfrequency problems are shown.

1. 緒 言

製品開発において有限要素法が使われるようになって久しい。近年では、そのプリ・ポスト処理までを考慮に入れたCAE の研究が盛んに行われて、構造解析・流体解析などの力学的シミュレーション技術の中心的な役割を担うまでになっているが、現状では解析と設計はそれぞれ独立した業務として成り立っており、統合的な設計業務が行われているとは言い難い。その最大の原因として、解析モデルの生成、なかでも有限要素メッシュの生成に莫大な労力を要することが挙げられる。これに対して、近年メッシュレス解析法[1]、ボクセル解析法[2]などの解析モデル生成を容易に行うことができる方法が盛んに研究されており、将来性が有望視されている[3]。筆者らは以前、多重ボクセル情報を用いた解析法を提案し、ソリッド構造の解析に有効であることを示した[4]。

一方において、これらの方法は境界条件の取り扱いや、要素剛性の計算法、ソルバー技術などに対して従来の有限要素法とは大きく異なるパラダイムに基づいた方法であり、これまで 40 年にわたって有限要素法に対して研究されてきた様々な知識、実務におけるノウハウ、プログラムの蓄積が使えなくなってしまうという問題がある。

また、ボクセル解析法は境界の取り扱いが特殊であり、それを従来の有限要素法の枠組みで用いようとする時、大きな問題となる。特に、境界上に節点がないためその変位や応力のポストプロセスに特殊な処理を要する。また、ボクセル解析法は 3 次元ソリッドに的を絞った方法であるが、解析にビームやシェルなどの要素を同時に使いたい場合、ボクセルで表現されたソリッド構造と従来の有限要素で表現されたビームやシェルの結合部分での適合性に問題が発生する。

そこで、この論文ではボクセル解析と有限要素解析の融和を目指し、ボクセル形状をしているが物体の境界上に節点を配置した「境界適合ボクセル要素」の開発を試みる。そして、この要素を汎用有限要素解析コード MSC/ NASTRAN に要素としてインプリメントし、ソリッド・シェル複合構造に適用することによってその有効性を実証する。

* 東京大学工学系研究科
** 三井造船

原稿受理 平成 10 年 7 月 10 日
秋季講演会において講演 平成 10 年 11 月 12，13 日

2．境界適合ボクセル要素の開発

2.1 境界適合ボクセル

鈴木ら[4]は、多重ボクセルを境界上の要素に対して定義することにより自由度を増やさずにより正確な解析を行う手法を提案した。これは、従来のボクセル解析が Fig. 1(a)のように、単純に形状表現ボクセルを解析に用いるものだったのに対して、Fig. 1(b)のように解析に用いる節点数をほとんど増やさず、形状表現のみをより詳細なボクセルを用うものである。形状ボクセルに対しては解析自由度は存在しないが、変位拘束や荷重の境界条件は形状ボクセルに対して与え、等価な解析自由度の値に変換を行う。しかし、前述のようにこれは通常の有限要素解析と大きく異なる考え方になるため、様々な問題が生じる。このボクセル解析を通常の有限要素法と組み合わせて用いるには、Fig. 1(c)のように要素境界上に節点があることが望ましい。このように、境界上に節点を配置し、境界にあった形で表現したボクセルを境界適合ボクセルと呼ぶことにする。見方を変えれば、これは境界上のボクセルをちょうど境界にあった形をした「要素」と捕らえることによって、3次元ソリッドに対する非常に簡単な「要素生成技術」であると考えることができる。そして、これにより専用ソルバーを必要としてきたボクセル解析法において、従来の有限要素ソルバーによる解析が可能になる。

計算に必要な記憶容量という意味では、従来の汎用有限要素ソルバーは任意の要素形状をはじめから仮定しており、ボクセル解析の大きなメリットの一つであるすべての要素の要素剛性を記憶する必要がないというメリットを享受することができない。しかし、この境界適合ボクセル要素による形状表現は通常のボクセルによる形状表現より格段によく、たとえば Fig. 2 に示すような形状であればボクセルによる表現では 100×100 程度の分割を要していたものが、境界適合ボクセルを用いれば 10×10 程度の分割ですむ。すなわち、これを用いれば従来 3 次元ソリッドで行われてきた（あるいはそれ以下の）要素数で解析を行うことができ、非常に簡単にかつ確実に要素分割が行えるというメリットだけ享受

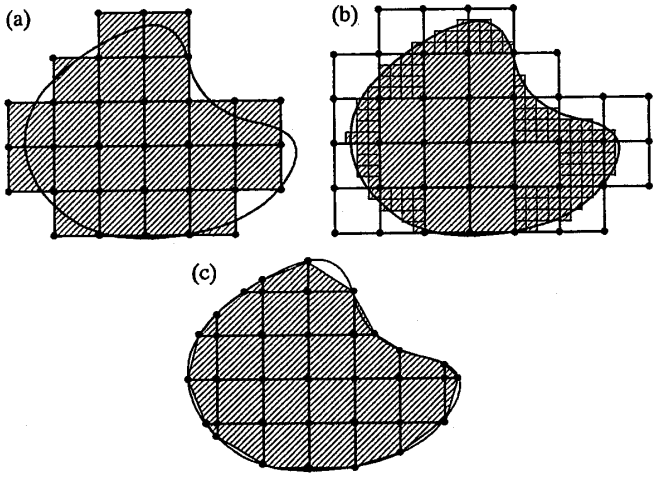

Fig. 1 Comparison of Geometric Representation using Geometric Voxel and Boundary Fit Voxel

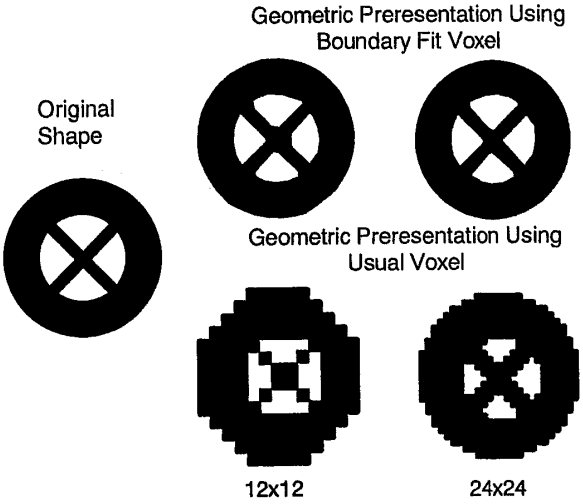

Fig. 2 Geometric Representation

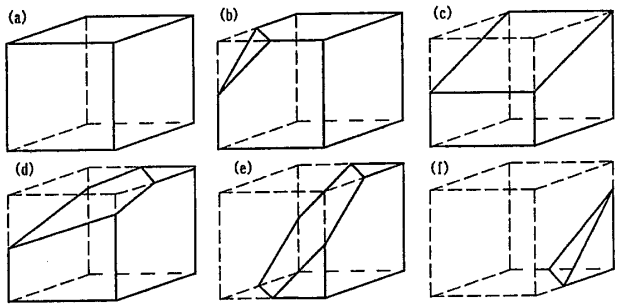

Fig. 3 Examples of Boundary Fit Voxel Element

することができる。

3次元の要素としては、例えば Fig. 3 に示すような要素が考えられる。他にも、切断面がちょうどボクセルの頂点に一致する場合などで場合分けすると、30 種類の形状に分類することができる。(a)や(c)の形状の要素は 6 面体であり、また(f)の要素は 4 面体であり、通常の汎用有限要素プログラムに必ず含まれているが、他の(b)、(d)、(e)などの形状に対する要素はない。そこで、これらの要素に対する形状関数を定義する必要がある。

2.2 境界適合ボクセル要素の形状関数

一般に、有限要素法における形状関数として用いるためには、以下の 3 つの条件を満足しなければならない[4]。

1) 要素内のある節点で 1、他の節点で 0 の値をとる。
2) 要素内および要素境界上で関数が連続である。
3) 線形の関数を厳密に表現できる。

これらの条件を満足させるように形状関数を定義する方法は、大きく分けて直交座標系の座標値の多項式をそのまま用いる Lagrange 要素と、ある中間パラメーターによって直交座標の座標値を表す関数と変位を表す関数を仲介させる Isoparametric 要素がある。一般に、3 次元ソリッドにおいては、4 面体形状に対してはラグランジュ要素が、6 面体に対してはアイソパラメトリック要素が用いられるが、この形状適合ボクセル要素に対しては基本的にグローバル

場 xyz 座標に直行するボクセルを用いているので、Lagrange 要素が有利である。

例えば、Fig. 3 (b), (d), (e)の 3 つのタイプの要素はすべて節点が全部 10 個あり、

$$N_i = a_{i1} + a_{i2}x + a_{i3}y + a_{i4}z + a_{i5}yz + a_{i6}zx + a_{i7}xy + a_{i8}x^2 + a_{i9}y^2 + a_{i10}z^2$$

という xyz に関する 2 次の完全多項式を用いれば、上記の条件の 1)と 3)を容易に満足することができる。ところが、Fig. 4 のように要素タイプ(b)の面 2-3-10-9-6 と、要素タイプ(e)の面 1-5-10-8-6 が互いに接する場合を考えると、この面での形状関数の多項式は、y=一定の条件を入れて以下のようになる。

$$N_i = a_{i1} + a_{i2}x + a_{i4}z + a_{i6}zx + a_{i8}x^2 + a_{i10}z^2$$

すなわち、この接する面に存在する節点は 5 つであるので、それぞれの節点での値が 1、または 0 となる条件のみだけでは 6 つの係数をすべて決めることはできず、その面内に属さない節点の情報が必要になる。ところが、このもう 1 つの係数を決定するべき節点の情報は隣り合う要素間で共有できないため、互いに異なる値を取る可能性があり、要素間の形状関数の連続性を満足できない。

そこで、新しい考え方として、6 面体のかどを切り落とした場合もその切り落としたかどにある自由度が存在するものとして扱う方法を考える。要素の節点結合情報を表現する際には、20 個全ての節点の対応を書き下すとし、たとえば Fig. 5 に示すようにかどの点（8 番の節点に対応）が切り落とされた場合にも、その節点は残し、節点 1～8 および新たに生成された節点 15、16、20 を用いて 11 の節点を持った要素として解析を行うことができる。

一般に、立方体の頂点のみの 8 つの節点よりなる形状関数は 8 節点アイソパラメトリック要素に用いる場合と同様にξ,η,ζ座標系を用いて簡単に定義することができる。この 8

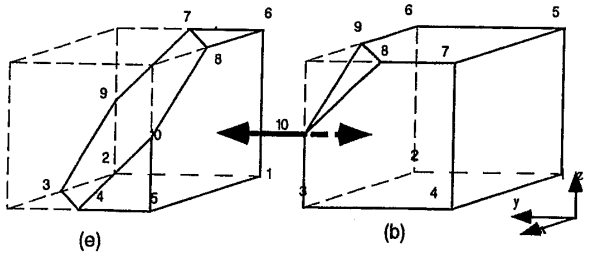

Fig. 4 Discontinuity of Shape Functions

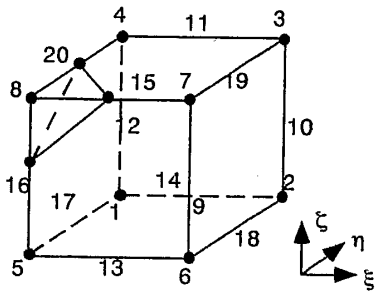

Fig. 5 Element with Exterior Node

つの節点に加え、Fig 5 の節点 15 が加わった場合には、

$$N_{15}(\xi,\eta,\zeta) = \frac{1}{4}\frac{1-\xi^2}{1-\xi_{15}^2}(1-\eta)(1+\zeta)$$

という形状関数を定義し、さらに、節点 7、8 に対応する形状関数を以下のように定義する。

$$N_7' = N_7(\xi,\eta,\zeta) - N_7(\xi_{15},\eta_{15},\zeta_{15})N_{15}(\xi,\eta,\zeta)$$

$$N_8' = N_8(\xi,\eta,\zeta) - N_8(\xi_{15},\eta_{15},\zeta_{15})N_{15}(\xi,\eta,\zeta)$$

また、このようにいちいちある節点が存在するかしないかで形状関数を変えるかわりに、その辺上に節点がなければ $N_{15}(\xi,\eta,\zeta) = 0$ とすることによってプログラムのアルゴリズムは非常に簡単になる。

ただし、辺上の節点がどちらかのかどに非常に近い場合には、関数の分母が非常に大きくなり、関数の最大値もそれに従い大きくなる。これは解析を不安定にし、精度を落とすことになるため、辺上の節点がある程度かどに近い場合には節点はかどにあるものとし、辺上には節点を発生させないこととする。このようにして定義した形状関数は、その辺上での節点の位置のみで一意的に決まるので、要素間の関数の連続性は必ず満足される。

2.3 剛性行列、質量行列、荷重ベクトルの計算

前節で定義された形状関数に対して境界適合ボクセルの要素剛性行列、質量行列を求めるには、領域において以下のような積分を行う必要がある。

$$[K_e] = \int_{\Omega_e} [B]^T [D][B] d\Omega$$

一般に、有限要素法においてはこの積分にガウス積分が用いられるが、これは積分を行う領域の形状が 2 次元の場合 3 角形や 4 角形、3 次元の場合 4 面体や 6 面体といった比較的単純な領域でのみ可能であり、ここで積分を行うような複雑な形状の領域に対してはガウス積分を行うことは難しい。そこで、ボクセルをさらに細分化した形状ボクセル[3]をあらかじめ定義しておき、それを用いて積分を行う。境界にあるボクセル要素に対しては、それをさらに細分化（たとえば 8×8×8 に分割）し、それぞれが物体内部に存在するか外部に存在するかを示すフラッグ（内部なら 1、外部なら 0 の値を持つ）を持たせる。これらは、境界のボクセル要素の詳細な形状を表現するので、「形状ボクセル」と呼ぶ。これによるデータ量の増加は 1 要素あたり 512 bit=64 byte ですむ。

この情報を用い、物体内部にある形状ボクセルに対して各形状ボクセルの中央の点で積分するべき関数の値を評価し、以下のように形状ボクセルの体積を重みとしてかけて和をとる。また、動解析に必要な質量マトリクスに関してもまったく同様に行うことができる。

$$[K_e] = \int_{\Omega_e} [B]^T [D][B] d\Omega \sim \sum_{invoxels} [B]^T [D][B] V_e$$

荷重ベクトルの計算は以下のように境界に沿った積分により計算される。

$$\{f\} = \int_{\Gamma_i} [N]^T \{t\} d\Gamma$$

まず、形状ボクセルの表面（2つの形状ボクセルが隣接していない面）の頂点を、Fig. 6のように最も近い物体表面上の点へと投影する。こうして、形状ボクセルの面は物体表面上の四角形に対応づけられるので、その四角形領域で外力$\{t\}$を積分し、その値$\delta\{t\}_i$を形状ボクセルの面での積分値とする。形状ボクセルの頂点と、それを物体表面上に投影した点との対応付けが一意的であるならば、ボクセルの表面と物体表面は共に閉空間であることから、両者に正則な写像関係を定義することが出来ることが判る。もしも外力のタイプが均一な圧力であるならば、単純に、形状ボクセルの面の法線方向に対して圧力を評価すれば良い。こうして要素において、境界の単位面積当たり$\{t\}$の境界積分を受ける場合、等価節点力は上記の式より計算できる。

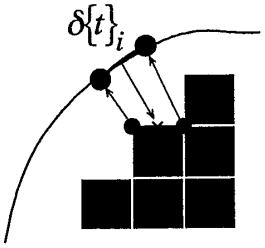

Fig. 6 Integration of Boundary Traction

3．MSC/NASTRANへのインプリメント

最後に、ここまでで述べた形状関数の定義、要素剛性マトリクス、荷重ベクトルの積分法を汎用有限要素法コードにインプリメントする。一般に、有限要素コードでの外部要素のサポート方法によってこのインプリメント方法は2種類に分けることができる。すなわち、Fig. 7 (a)に示すようにユーザーが要素剛性マトリクスの計算方法をサブルーチンとして与え、入力データとしては通常の有限要素型のデータ同様に節点情報とコネクティビティ情報を与える方法と、Fig. 7 (b)に示すように入力データとして要素剛性マトリクスの各成分の情報をそのまま与えてしまう方法である。この場合は、コネクティビティデータは与える必要がない。

サブルーチン型のインプリメンテーションの方がより通常の有限要素解析に近いスマートなやり方であるといえるが、MSC/NASTRANでは、ユーザー要素を定義するのに後者の方法が用いられる。すなわち、GENEL要素として定義して、そのパラメータとして節点情報、要素剛性マトリクスの各成分情報を入力する形になる。また、質量マトリクスに関しては、要素ごとではなく全体の質量マトリクスの成分を直接入力する形になる。

ボクセル情報に基づく解析では、x-y-z座標軸に沿った規則的な格子が用いられるため、ある格子点や要素の位置の情報を得るために、単に要素番号を指定すれば容易にその位置が決定できるというメリットがあった。すなわち、従来の有限要素法に用いられているような要素情報－コネクティビティ情報的なデータ構造は必要ない。一般に有限要素データにおいては、節点と要素はまったく別のものとして扱われ、それぞれ独自のIDが割り振られる。それに対して、ここではFig. 8に示すように空間をセル分割し（ここでは2次元的に書いているが、実際は3次元的な分割になる）、各セルに順に番号を振る。図中の番号はセルの番号を示す。たとえば、節点1には要素9のかどの節点が対応し、節点4は要素11の辺の中間節点であることがわかる。これにより、要素か節点か、また、かどの頂点の節点か辺上の節点かを指定することなく、単にセルの番号を示せばそれが何かを容易に決定することができる。MSC/NASTRANにおいては節点番号、要素番号は必ずしも連続した番号である必要はないため、このセル番号を直接節点番号、要素番号に用いる。

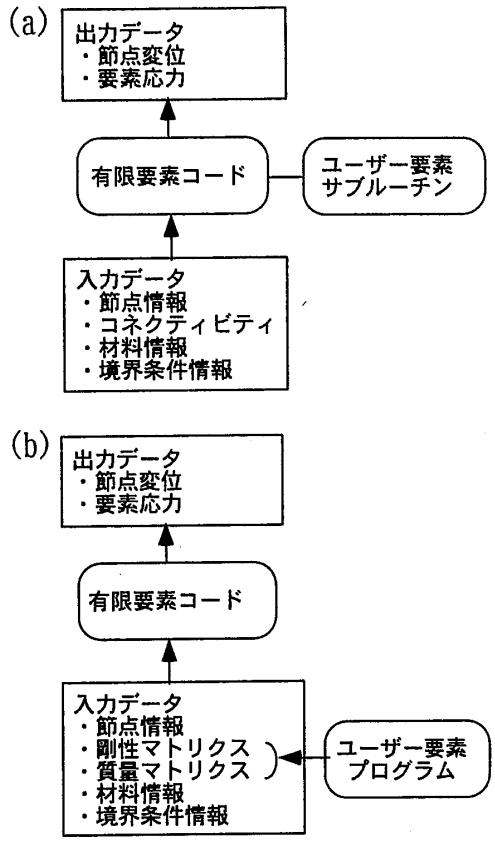

Fig. 7 Implementation to MSC/NASTRAN

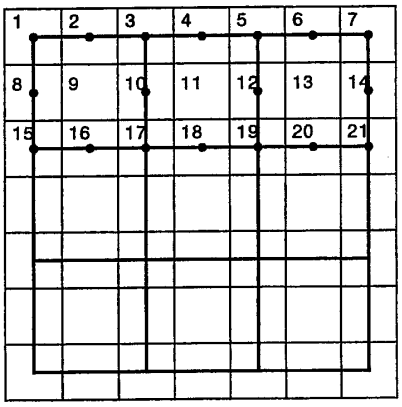

Fig. 8 Cell Division of 3D Space

4．解析の流れ

解析の流れは以下のようになる。形状モデルは、CADで設計したモデルをSTLフォーマットとして出力したものを

用いた。STL はほとんどの 3 次元 CAD がサポートしている上、基本的にポリゴンデータであり表示が容ポリゴンとの交点を計算することによって行う。また、境界にあるボクセルに対しては形状ボクセルを定義され、要素剛性行列、要素質量行列を計算する。変位拘束、荷重などの境界条件は STL データのパッチに与える。境界条件の与えられたパッチに関連するボクセルには、相当する節点への適当な拘束、等価節点荷重として MSC/NASTRAN フォーマットの境界条件が出力される。これらの一連の MSC/NASTRAN の入力データは自動生成され、テキストファイルとして保存される。

これらの結果の可視化及び GUI 構築には VRML フォーマットを用いた。VRML(Virtual Reality Modeling Language)[5]は 3 次元オブジェクトを記述する言語で、インターネット上での利用を想定して設計されている。1996 年に定義された VRML 2.0 ではアニメーションの機能も付け加えられ、動的な解析結果の表示も可能である。

前述のように、このボクセルを用いた解析手法は設計者が自ら解析を行うことを目指した解析システムである。設計者は有限要素メッシュには興味はなく、興味があるのは結果として得られる変位、応力値である。現在ポストプロセッシングにおいて有限要素メッシュを同時に表示するのは、妥当なメッシュで切っていることを明示するためである。境界適合ボクセル要素を用いた解析では、確実に要素分割をすることが可能であり、メッシュの表示は不要である。そこで、メッシュ分割前の形状データをそのまま用いたポストプロセスを考える。すなわち、NASTRAN が出力した節点における変位情報を元に、適当な補間により形状データである STL データの頂点における変位および応力値を計算する。変形の表現は、単純に評価点の変位を拡大した変形後の座標に対して Geometry ノードを作成することによって実現される。また、応力値に関しては評価点において計算された Mises 応力を RGB 値に変換し、Geometry ノードにカラーを定義することによって実現できる。結果の表示には、VRML フォーマットにて出した解析結果情報を VRML ブラウザにて表示する。解析の全体の流れを Fig. 9 に示す。

5. 精度の検証

まず、この境界適合ボクセル要素が十分な精度で解析を行うことができることの実証を、厳密解のある問題との比較で精度を検証することによって行う。Fig. 10 に示すように、一様外圧を受ける厚肉円管の解析を行った。対称性を用い、1/4 部分を切り出して、すべり条件を課すことによって右側に示すような等価な問題とすることができる。これを、前述の境界適合ボクセル要素を 2 次元にしたものを用いて解析を行った。分割としては 20×20 の分割を行い、さらに境界のボクセルに対して 8×8 の形状ボクセルでより詳細に形状を定義した。これにより総節点数として 702、また、材料のある領域の要素数として 262 となった。なお、ヤング率としては 1000 GN/m²、ポアソン比として 0.3 が用いられた。

Fig. 10 に、変位を形状関数に基づいて補間した関数の厳密解に対する誤差の分布をグレースケールで示す。ただし、誤差は厳密解と解析解の差のノルムを厳密解のノルムで割った相対的なものである。図中に、いくつかの点での誤差の値を示す。1.379 %となっている所が領域中最も誤差が大き

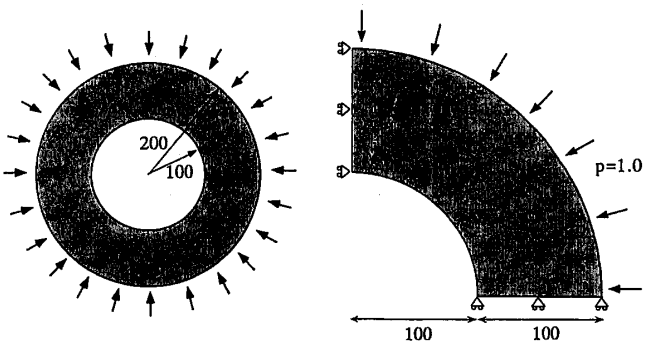

Fig. 10 Cylinder with External Pressure

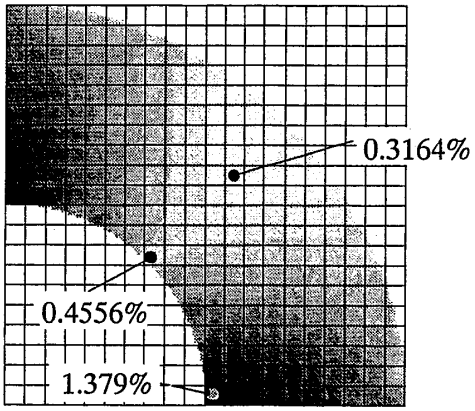

Fig. 11 Error in Displacement

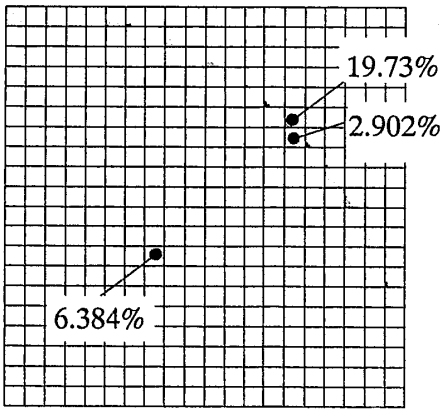

Fig. 12 Error in von Mises Stress

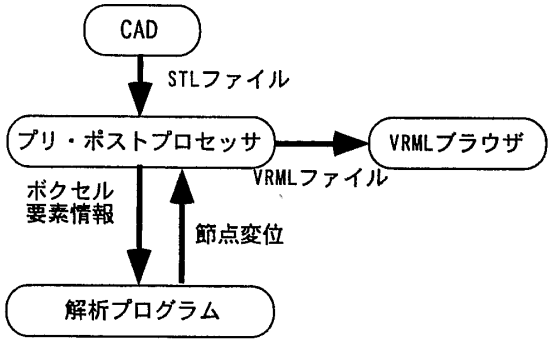

Fig. 9 Flow of Analysis

い要素である。ほとんどのところは誤差が1%以下となっており、全般に変位に関しては境界も含めて非常によい解が得られているといえる。

次に、この問題に対する応力の精度を見てみる。Fig. 11 に von Mises 応力の相対誤差（応力の誤差を応力値で割ったもの）の分布を示す。特に要素の中の充填率が低い要素において、比較的高い誤差が出ているのがわかる。これは、要素の中の物体外にある節点で比較的大きな変位値が出てしまい、（これは、空間の変位値なので間違いではない）その変位値を含めて応力を評価しているため、そのようになってしまうと考えられる。

要素のサイズを小さくして解析を試みたところ、変位は要素サイズに応じて精度が上がっていったが、応力に関しては上記のような充填率の低下に伴うおかしな応力値の発生が見られ、形状関数を用いた応力値の評価は適当でないことがわかった。しかし、変位は正確に求められているので、それを用いて適当な補間をして応力値を求めれば良い応力値はえられると考えられる。

5．3次元ソリッドの解析への適用例

5.1 ギアの静解析

それでは、このボクセル要素を実際の工業製品の解析に適用した例を見てみよう。Fig. 13 に示すギアに対して、20×20×2 のボクセル要素に分割して解析を行った。そして、境界上のボクセルに対して 8×8×8 の形状ボクセルを定義した。これにより、640 の要素、1893 個の節点が定義され、MSC/NASTRAN の入力データを作成した。境界条件としては内周上の変位を完全固定し、ギアの 1 つの歯の線上に分布荷重をかけた。

解析結果の Mises 応力を Fig. 14 に示す。非常に粗いボクセル分割ではあるが、応力解析がきちんと行えるのがわかる。従来の有限要素法では、このような複雑な形状に対してメッシュを切る場合はせいぜい 4 面体要素による解析しか行えず、数千以上の要素を必要としていたのに比べると、はるかに少ない要素数で解析できる。

5.2 タービン翼の固有振動解析

次に、タービン翼の固有振動解析を行った。Fig. 15 に形状モデルを示す。境界条件は基部を完全に固定し、材質はアルミとした。翼がねじれているのは。ピッチ角がついているためあり、変形しているわけではない。
これを、5×5×5 のボクセル要素に分割し、解析を行った。Fig. 16 に 1 番目から 4 番目までの固有振動モードの様子とその固有振動数を示す。1～3 番目は曲げのモードで、4 番目はねじりのモードであることが判る。

5.3 フック（ソリッド・シェル混合構造）の解析

最後に、ソリッド・シェル混合要素の例題として、フックの解析を行った。Fig. 16 に形状モデルを示す。フックの部分はソリッドでモデル化され、その周りの部分は板としてモデル化されている。板の部分を周辺単純支持し、フックに荷重をかけたときの解析を行った。Fig. 17 に、解析モデルのメッシュ図を示す。境界適合ボクセル要素 419 個、シェル要素 1526 個、節点数 2903 個としてモデル化されている。細かいネジ頭の部分等も、形状を簡略化する必要なくモデル化されていることがわかる。

6．結　言

ボクセルの考え方を用いることで従来の有限要素解析のボトルネックとなっていた 3 次元ソリッドの要素生成を非

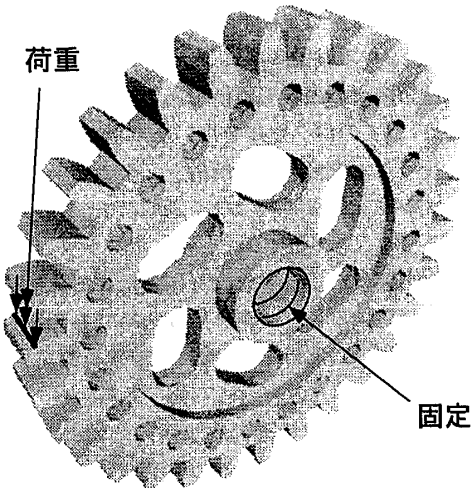

Fig. 13 Analysis Model of Gear

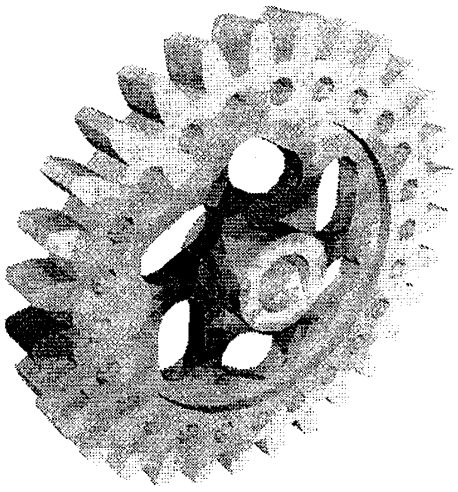

Fig. 14 Analysis Results of Gear

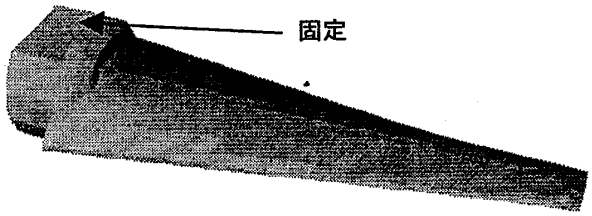

Fig. 15 Analysis Model of Turbine Blade

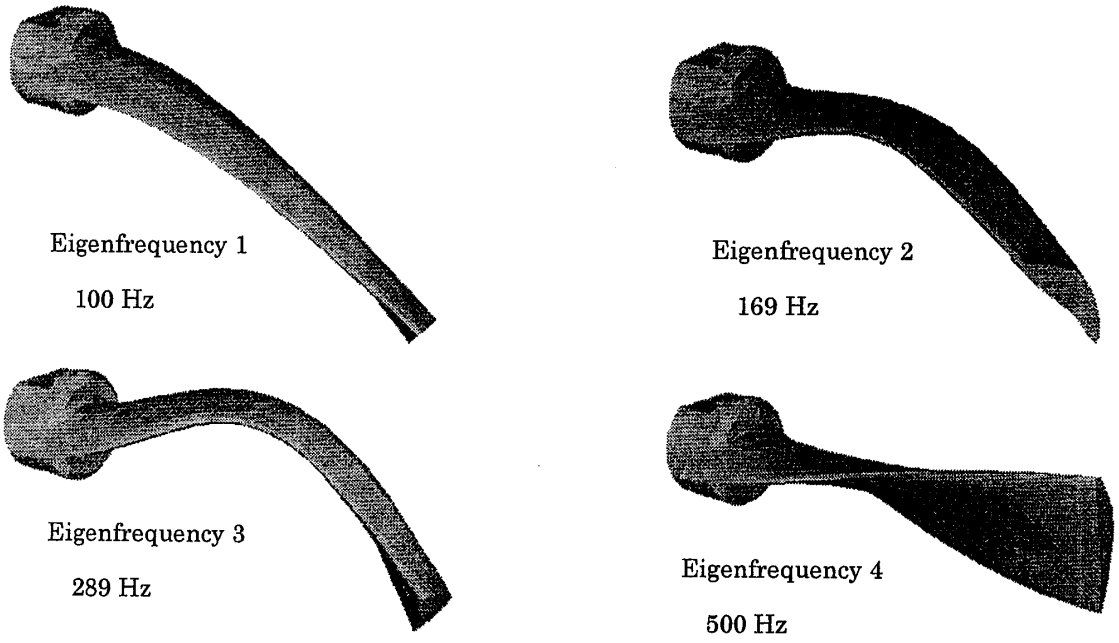

Fig. 16 Eigenfrequencies and modes of Turbine Blade

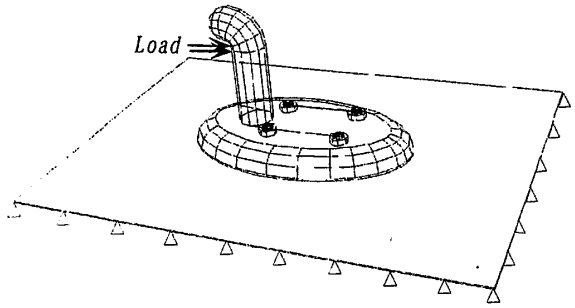

Fig. 17 Model of Hook

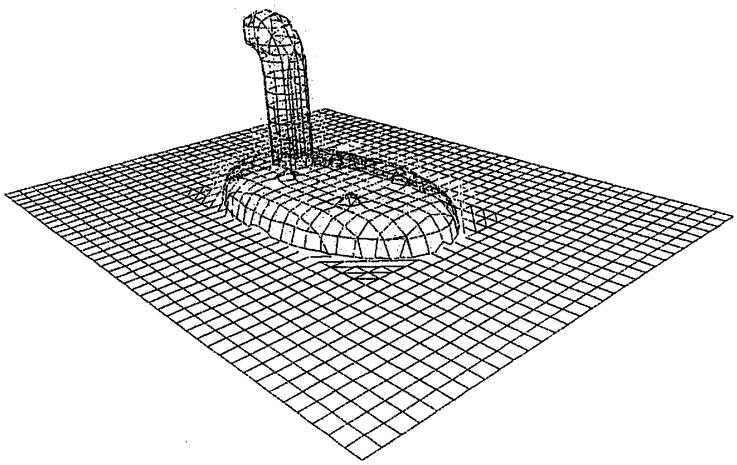

Fig. 18 Finite Element Mesh of Hook

常に簡単に行い、また境界形状もスムーズに再現することができる境界適合ボクセルを提案し、従来の有限要素解析程度の自由度で解析をすることでボクセル解析の計算時間の問題を解決することができることを示した。

形状関数として、要素間の関数の連続性を常に満足することのできる方法を提案した。また、複雑な形状をしたボクセル要素に対してその形状関数を積分して要素剛性行列、要素質量行列を、そして境界荷重に対して、それをこの要素の等価節点荷重を計算する方法を提案した。

さらに、この新しく開発した要素を汎用有限要素解析プログラム MSC/NASTRAN で用いる方法を提案し、インプリメンテーションを行い、ユーザー定義要素として節点、要素を統一的に番号付ける方法を提案した。例題として、静解析、固有振動解析、ソリッド・シェル混合構造の解析の例を示した。他にも、動解析なども MSC/ NASTRAN の機能そのままを用いて行うことも可能である。

謝辞

例題の作成にあたって、中西克嘉君、石橋淳生君（東京大学工学系研究科修士課程）の助力を得た。ここに感謝の意を表する。

参考文献

1) Belytschko, T. et. al. "Meshless Methods: An Overview and Recent Developments" Computer Methods in Applied Mechanics and Engineering, vol. 139, pp. 3-47 (1996)
2) Kikuchi, N. & Diaz, A. 第 13 回 Quint セミナーテキスト、(株)くいんと (1997)
3) 鈴木克幸,他：ボクセル情報を用いたソリッド構造の解析法、日本造船学会論文集 Vol. 182 pp 595-600 (1997)
4) Hughes T. J. R. : The Finite Element Method ; Prentice-Hall International Editions, 132-137 (1987)
5) VRML Architecture Group http://vag.vrml.org/