《小特集》

# 物理モデルによる自動メッシュ分割

嶋　田　憲　司*

**ABSTRACT**　Automated mesh generation is a significant problem in many computer applications such as engineering analysis, computer graphics and layered manufacturing. Although various algorithms have been proposed thus far, their functionality and coverage are limited, in most cases being application-specific. This paper provides a summary of those algorithms by classifying them and pointing out unsolved problems. In reviewing them, it is clear that none sufficiently address the major problem of node placement, that is to say the locating of nodes such that they maintain a specified distance between nodes and create a mesh with minimal distortion. This paper introduces a new, physically-based approach to this problem. The method was inspired by the pattern of soap bubbles floating in liquid which gives topology and geometry similar to ideal meshes. Each mesh node is modeled as a bubble with a point mass and viscous damping. The force field exerts repelling/attracting forces between bubbles much like intermolecular van del Waals forces. The equilibrium bubble configuration is obtained by solving numerically the governing equation of motion to yield the static force balance. After the bubble locations are decided, their center points are connected by Delaunay triangulation to give a complete mesh topology.

## 1. はじめに

メッシュ分割とは任意形状の連続領域をより単純な形をした要素の集合（メッシュ）に近似することである．2次元領域は，三角形や四角形などの多角形要素の集合に，また3次元領域は，四面体や六面体などの多面体要素の集合に変換される．このようなメッシュ分割は，計算力学をはじめとするさまざまなコンピュータ・アプリケーションに不可欠な作業となっている．

計算力学は，有限要素法や境界要素法，差分法などを用いて固体・流体・熱力学あるいは電磁気学などの問題をコンピュータで数値的に解く方法である．これらの方法は応用力学の支配微分方程式を，メッシュ要素の節点（ノード）のおける離散的な値をもとに定式化して解く方法である．当然，メッシュの大きさや形状，その配置が解の精度と収束性に大きく影響する．メッシュを構成する要素数が多いほど解の精度は向上するが，計算時間（つまりコスト）が多くかかってしまう．したがって，解析上重要な領域部分は細密なメッシュを，それ以外は荒いメッシュを適宜用いることが必要となる[1]．また，要素形状の極端な歪みも解の精度と収束性を悪くする要因となる．このように要素の大きさを連続的に変化させながら歪みの少ないメッシュを作成するのは容易ではない．

メッシュ分割が重要となる他のアプリケーションとして，コンピュータ・グラフィックスがある．近年のグラフィックス・ワークステーションの進歩は著しく，複雑な曲面や3次元形状を陰影つきの絵としてリアルタイムで表示できる．これを可能にしているのが，一秒間に何十万という多面体を描くための専用ハードウエアであり，このハードウエアを生かすためには表示物体の曲面形状を多角形に切り分けて近似する必要がある．計算力学と同様に，この場合も全体の形状を均一の大きさの多角形で近似するのは効率的ではない．形状が急に変化している部分（曲率半径が小さい部分）は細密な多角形メッシュを用い，緩やかに変化している部分は荒い多角形メッシュを用いるのが望ましい．

また，新しい製造技術として最近注目を集めているいわゆるレイヤード・マニファクチュアリングのための形状表現としても，同様の多角形メッシュ近似が用いられている．この製造技術は，3次元製品形状を一

Physically-Based Automatic Mesh Generation. By *Kenji Shimada* (IBM Japan, Ltd., Tokyo Research Laboratory, CIM Technology; and Massachusetts Institute of Technology)
*日本アイ・ビー・エム㈱東京基礎研究所CIMテクノロジー
マサチューセッツ工科大学

定間隔でスライスして，その2次元形状を一層づつ積み重ねてゆくことにより3次元形状を造形する技術である．ここで，形状処理側の問題として重要なのは，いかに2次元スライス形状を効率的かつ安定に算出するかである．CADシステムで用いられる種々の曲面表現形式ごとに，ソフトウエアを開発するのは非効率的なので，標準の形状表現として多角形メッシュが使われている．この際にも，CADシステムで定義された曲面形状と，実際に製造される多角形近似形状との誤差を一定値内に収めるために，曲面の曲率半径に応じて大きさに変化をつけながらメッシュ分割を行うことが必要となる[2]．

このように，メッシュ分割は設計，解析，製造におけるさまざまなコンピュータ・アプリケーションにとって不可欠な基本的作業である．にもかかわらず，これまでは各アプリケーションごとに部分的・断片的に論じられてきただけで，アルゴリズムも対処療法的に専用のものが開発されてきた．そこで，本稿では第2章で問題の定義と要件を包括的に整理し，また従来法を分類したうえでその問題点を指摘する．さらに第3章以降，この問題点を解析するために，幾何学的方法ではなく物理モデルと動力学シミュレーションを用いる新しいメッシュ分割方法を紹介してゆく．

## 2. 自動メッシュ分割問題と従来のアルゴリズム

本章では，まず自動メッシュ分割を行う際の要件を整理し，さらに従来提案されてきたアルゴリズムとその問題点にも言及する．

**図1**にも示したが，本稿でいう自動メッシュ分割は，以下のような入力を出力に変換する問題である．

入力
- 曲線，曲面，空間内の連続領域の形状
- 特にノードを配置したい領域内形状
- ノード間距離分布 $d$

出力　曲線領域に対しては線分列，曲面領域に対しては多角形メッシュ，空間領域に対しては多面体メッシュ

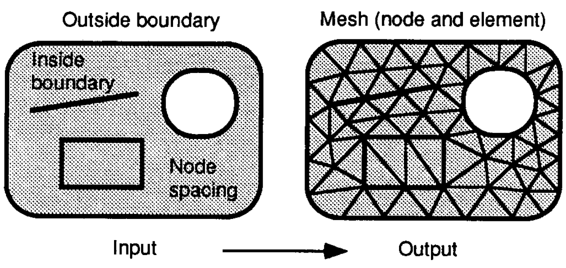



**図1**　メッシュ分割問題の入力と出力

入力項目の2番目の「特にノードを配置したい領域内形状」は，例えば異材料を接合してできた対象形状を解析するために接合面上にノードを配置したい場合などに必要となる．また，3番目の「ノード間距離分布 $d$」は，メッシュを構成する多角形や多面体の大きさ，すなわち一辺の長さを位置の関数として与えたものである．前章でも触れたようにアプリケーションごとに適したものを定義しなくてはならない．

メッシュ分割を行う際の要件は，多くの文献に繰り返し指摘されてきた[3)~7)]．当然，アプリケーションによって微妙に変わってくるが，以下に挙げる項目は全アプリケーションに共通の基本的要件と言ってよい．

- 歪みの少ないメッシュ要素

三角形メッシュの場合には可能なかぎり正三角形に，四面体メッシュの場合には正四面体に近いことが望ましい．計算力学のモデルとして用いるとき，極端に歪んだ要素は解の精度や収束性を悪化させる．

- 連続的に変化するメッシュ要素の大きさ

任意に定義されるノード間距離分布 $d$ にしたがって，メッシュを構成する多角形や多面体の大きさを連続的に制御できること．

- 領域境界と領域内形状上へのノードの配置

与えられた領域の境界上と，これに加えて特に指定された領域内の形状上にノードが正確に置かれていること．

- 曲線，曲面，空間領域のすべてに対応する方法

アプリケーションによっては，上記領域が組み合わされた複合領域をメッシュ分割することが必要となるので，全種の領域に統一的に適応できる方法が望ましい．

以上，一般的なメッシュ分割問題の定義とその要件を述べてきたが，次に従来の方法を概観してみよう．これまで，この問題が最も多く議論されてきたのは有限要素法などの計算力学の分野においてである．特に2次元問題は歴史が長く多様な方法が提案されてきた．3次元問題については近年多くの研究成果が報告されてはいるが，決定版と呼べるものはまだない．

ここでは，従来法を4グループに分けて紹介する．

- 部分領域分割法[8)~10)]

まず，与えられた領域を，凸領域や穴のない領域などのより単純な扱いやすい部分領域に切り分けて，次にそれぞれの部分領域をメッシュ分割する．部分領域のメッシュ分割は，規則正しい格子を写像関数でマッ

ピングしたり，簡単なルールによって内部ノードを加えたりして行う．効果的な部分領域への自動分割方法と部分領域のメッシュ分割方法が課題となる．

• 階層的空間分割法[11)~13)]

2 次元領域を 4 分木で，3 次元領域を 8 分木で階層的に再分割してゆく方法．分割の深さによって要素の大きさを離散的にしか制御できない．滑らかな領域境界を表すために，境界近傍の要素として一部が欠けた正方形や立方体を扱えるように拡張する方法も提案されている．問題点として，分割軸に対する形状の置きかたが分割結果に大きく影響することや，領域内部には歪みの少ない要素が生成されるがより重要な表面に歪みの大きいものが生成されることなどが指摘されている．

• 再帰的二分割法[14),15)]

メッシュ要素として使える形状が得られるまで，領域を再帰的に二分割してゆく方法．部分領域分割法と異なり，要素のレベルまで分割だけで領域を切り分けていく．この際に必要となる形状モデリングのデータ操作などの議論は多く行われているが，ノード間距離の連続的な制御や要素の歪みを少なくすることなどの基本的要求に対してはまだ十分に考慮されていない．

• ノード結合法[16)~19)]

まず領域の境界と内部にノードを配置し，次にこれらを結んでメッシュを構成する方法．いったんノードの集合が与えられると，*Delaunay* の方法を用いてこれを結合して歪みの少ないメッシュを生成できる．この方法は計算幾何学の分野で長く研究されてきたもので，2 次元・3 次元空間（あるいはさらに多次元空間）に散らばった点を接続して正三角形や正四面体に近いメッシュを生成するための効率的かつ安定なアルゴリズムである．問題はいかにして場所ごとに指定されたノード間距離を満たし歪みの少ないメッシュを生成するようなノード配置を得るかである．特に 3 次元の場合は容易ではない．

どの方法もそれぞれ未解決の課題が残されていて，先に挙げた要件を全て満足するほどには完成されていない．最近のひとつの傾向としては，考え方の単純さ，2 次元・3 次元領域の統一的な扱い，*Delaunay* の方法の優秀さなどの理由から，ノード結合法の人気が高まりつつあるようだ．

次章以降，ノード結合法の残された課題であるノードの配置問題を，物理モデルと動力学シミュレーションを用いて解く新しい方法を紹介する．

## 3. 物理モデルによるアプローチ

ノードの配置問題は単純に見えるが，意外に難しい．前章で指摘した要求条件をすべて満たすような幾何学的方法となると簡単には捻出できない．

ところが，私達は自然界のなかに理想的なメッシュ形状を連想させる規則的な繰り返しパターンを多く見ている．平面上であれば，構成要素が六角形のものが特に多い．例えば**図 2** は石鹸水の表面にストローで泡を吹き込むことによってできた六角形パターンである．これ以外にもハチの巣，昆虫の複眼，ベルナール対流など数え上げればきりがない．これらの要因としては表面張力，要素間の引力と斥力，熱・流体力学的な作用などさまざまであるが，結果としてできる形は同じである．この各六角形の中心点をとってこれらを結べば，まさに 2 次元メッシュ分割問題で得たい三角形メッシュができる．

同じように，3 次元の規則正しいパターンもある．身近な例ではビールを注いだときグラスの上部に規則正しく詰まっている細かい泡がある．さらに微視的な例として六方最密充填構造のような金属の分子構造がある．各分子の中心点をとって結べば，3 次元のメッシュ分割問題で得たい正四面体のメッシュができる．

本稿の題目である物理モデルによるメッシュ分割の方法は，このような自然界のメッシュ形状を観察することから着想されたものであり，次のような手順で問題を解く方法である[20)]．

(1) 理想的なメッシュ形状を生成するような物理モデルを設計する．

(2) この物理系の支配方程式を構築する．

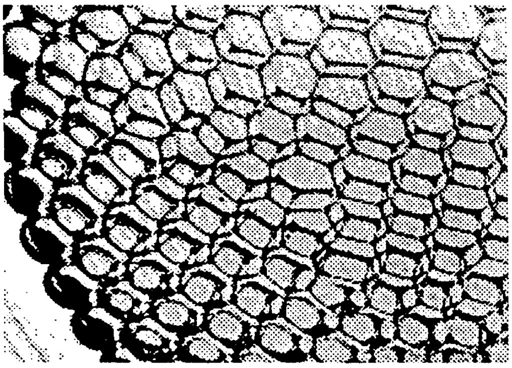


**図 2** 石鹸水の表面に泡を吹き込むことによってできた六角形パターン．各六角形の中心点をとってこれらをつなげば，2 次元のメッシュ分割問題で得たい三角形メッシュができる

(3) この支配方程式を数値的方法で解く.

(4) 解をもとにノード配置を得る.

(5) ノードを *Delaunay* の方法で結んでメッシュを得る.

上記のプロセスにおいて注意すべき点は，いわゆる解析問題，すなわちコンピュータ・シミュレーションによって物理現象を再現する場合と同じだ．まず，「物理モデルのシェイプアップ」，つまり期待する現象を再現するために必要最小限のモデルを設計することである．また，計算量の多さは物理モデルを用いたアプローチに共通の悩みであるので，計算量の少ない数値解法を探さなければならない．また，支配方程式が多変数の複雑な非線形方程式となる場合には解の収束性と安定性にも注意を払う必要がある．

さて，このような考え方に沿って私達が提案している物理モデル，「バブル・システム」による自動メッシュ分割方法の概要をここで紹介したい．バブルは，質量をもった球状粒子であり，その直径は第2章で定義したノード間距離分布によって与えられる．2つのバブル間には両者が接しているときに最も安定な状態となるような力の場を定義する．この安定距離よりも両者が近づくと斥力が働き，遠ざかると引力が働く．また，バブルの速度に比例した粘性抵抗も存在するものとする．

さて，このようなバブルを3個だけ平面上におくと何が起こるだろうか？ 3個の球は，すきまなく互いに接し合う位置に移動する．これが力の平衡位置であり，系のエネルギーが最小になる状態だからである．平面上のすべての場所でノード間距離が一定で，3つのバブルの直径が等しければ，各球の中心点は正三角形を構成する．さらに多くのバブルを加えてゆくと全域にこの3球が接する配置が繰り返されて六角形パターンを生じる．これはいわゆる球の平面上の最密充填配置であり，どのバブルも6個の隣接バブルを持つ状態である．3次元の場合も同様だが，単位となるのは4球が互いに接した配置であり，もしすべての球が均一の大きさであれば，その中心点は正四面体を構成する．これを繰り返すといわゆる六方最密充填となり，各バブルは12個の隣接バブルを持つ．

さて，このようなバブル・システムを用いて3次元立体のメッシュ分割を行うには，**図3**に示したように次元の低いほうから

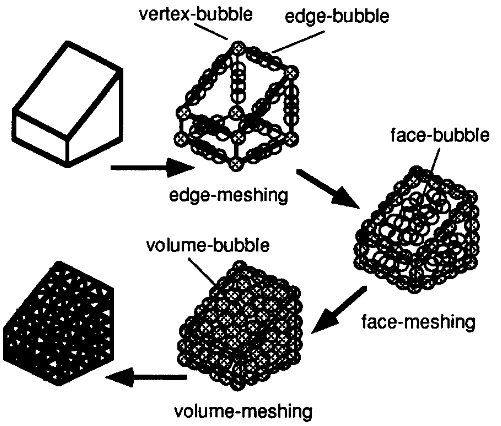
(1) 頂点上に頂点バブル（vertex bubble）を配置する．

(2) 稜線上に稜線バブル（edge bubble）を配置する．

(3) 面上に面バブル（face bubble）を配置する．

(4) 空間内に空間バブル（volume bubble）を配置する．

という4つのステップを順次行えばよい．頂点・稜線・面バブルは，各々頂点上・稜線上・面上にその動きが拘束されたものである．次章で詳しく述べるが，各ステップでは力の平衡状態，すなわち互いに接するようなバブルの配置を動力学シミュレーションによって得る．




| | domain | domain perimeter | element | force-balanced bubble config. |
|---|---|---|---|---|
| edge-meshing | curve | vertices | line segment | |
| face-meshing | surface | curves | triangle | |
| volume-meshing | volume | surfaces | tetrahedron | |

**図3** バブル・システムを用いた3次元立体のメッシュ分割手順．次元の低いほうから順に，頂点，稜線，面，空間バブルを動力学シミュレーションによって互いに接し合うように配置してゆく

## 4. バブル・システムによる自動メッシュ生成

### 4.1 物理モデル

バブルの直径はノード間距離分布 $d(x, y, z)$ によって与えられる．**図4**に示すように2つのバブルが接している状態が安定状態であるから，バブル $i$ とバブル $j$ の位置を各々 $(x_i, y_i, z_i)$ と $(x_j, y_j, z_j)$ とすると，安定距離 $r_0$ は両バブルの半径の和として次のように書

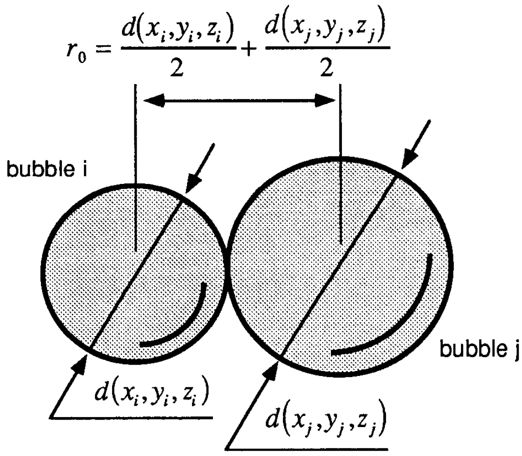



**図 4**　2 つのバブル安定配置．両バブルが接し合う状態が安定配置で，このときの安定距離 $r_0$ は両バブルの半径の和となる

ける．

$$r_0=\frac{d(x_i, y_i, z_i)}{2}+\frac{d(x_j, y_j, z_j)}{2}$$

ここで，**図 5** に示すように両者の距離がこの安定距離よりも近づくと斥力が働き，遠ざかると引力が働くような力の場を，ファン・デル・ワールス力をモデルにして定義する．このバブル間力 $f$ は両者の距離 $r$ の関数であり，

$$f(r_1)=0, f(r_0)=0, f'(0)=0, f'(r_0)=-k_0$$

という条件を満たすような 3 次関数として定義される．ここで，$k_0$ は安定距離におけるバブル間力を線形バネに換算した場合のバネ定数に相当する．この関数は以下の点でファン・デル・ワールス力よりも扱いやすく，動力学シミュレーションの計算量を減らすことにも役立っている．

• 両バブルが極端に接近した場合にも，斥力が無限大に発散しない．

• 一定の距離以上に両バブルが離れるとバブル間力は働かない（力の影響範囲の局所化）．

各バブルに働く力は，他のバブルからの力のベクトル和である．これをすべてのバブルについて得るには，全バブルの組み合わせを逐一調べるのが最も素朴な方法であり，この場合 $O(n^2)$ の計算量が必要となる．しかし，力の影響範囲の局所化や領域の階層的管理などを行うと，一般的なバブル分布の場合には $O(n \log(n))$ の，ほぼ一様なバブル分布の場合には $O(n)$ の計算量となることが報告されている[21)~23)]．

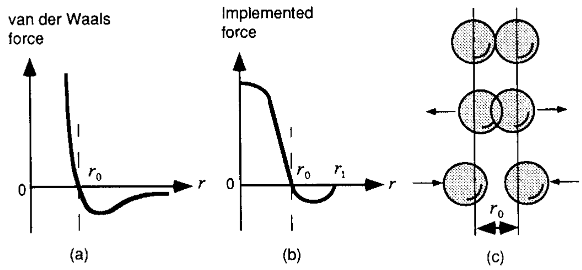



**図 5**　ファン・デル・ワールス力とそれをモデルにして定義されたバブル間力．バブル間距離が安定距離よりも近づくと斥力が働き，遠ざかると引力が働く

### 4.2　支配方程式

次のステップは，前節に述べたようなバブル間力，質点，粘性抵抗を考慮して，バブル・システムの運動方程式を古典的ニュートン力学にもとづいて求めることである．各バブルは慣性モーメントを持たないので，並進運動だけを考慮すればよい．したがって個々のバブルは $x, y, z$ の 3 つの自由度を，n 個のバブル・システムは 3n の自由度を持つことになる．$i$ 番目の自由度に関する並進運動の支配方程式は，

$$m_i\frac{d^2x_i(t)}{dt^2}+c_i\frac{dx_i(t)}{dt}=f_i(t)$$

と書ける．ここで，$m_i$ はバブルの質量を，$x_i$ は自由度 $i$ の座標値を，$c_i$ は粘性抵抗係数を，$f_i$ は隣接バブルが及ぼすバブル間力の合力を表す．上式は動力学の教科書にも載っている典型的な 2 次常微分方程式であるので，速度と加速度の項を差分近似したり，*Runge-Kutta* 法などの数値積分の方法[24)]を用いれば，時間とともにバブルの配置が力の平衡状態に近づいてゆく過程のシミュレーションができる．最終的な力の平衡状態は全バブルが静止している状態であるから，速度と加速度の項は零となって，

$$f_i(t)=0$$

という力の平衡状態となる．動力学シミュレーションを行いつつゆっくりと系全体の運動エネルギーを吸い取ってゆくと，最終的に系がひとつの安定状態に達するわけである．もし分子間力が極く単純な線形バネであれば，動力学シミュレーションを行うまでもなく，直接的に力の平衡状態を算出できる．しかし，バブル・システムのバブル間力が非線形であるため，直接に解を得ることはできない．また，このような非線形な系では，複数の解（いわゆるポテンシャル・エネルギーのローカル・ミニマ）が存在するが，適切な初期値から始めて到達する解はメッシュ分割には十分なもの

ばかりなので心配はいらない．

### 4.3 物理パラメータ

前節で示した支配運動方程式中のバブル質量，粘性抵抗，バブル間力の強さという3つの物理パラメータ値は自由に選択できる．では，系の安定性と応答性を高めるためにはどのような値がよいのか？

安定性と応答性は，質量，粘性抵抗，線形バネで構成された次式で表される2次系では本来相反する要求である．

$$m\frac{d^2x(t)}{dt^2}+c\frac{dx(t)}{dt}+kx(t)=0$$

つまり，質量を軽くしたり，粘性抵抗を小さくすれば系の応答性は向上するが，振動しやすくなり安定性が悪化する．また逆に，大きな慣性や粘性抵抗は安定性を増やすが応答性を悪くする．そこで，制御の分野では，安定性と応答性をある程度満たして，収束時間を最短にするために以下のような標準減衰係数が広く使われている[25]．

$$\zeta=\frac{c}{2\sqrt{mk}}=0.707$$

バブル・システムでもこれに満たすように質量，粘性抵抗，そしてバブル間力の強さを決めてやればよい．ただし，この際に本来非線形なバブル間力を線形バネ換算しなければならない．このためには，**図6**に示すように稜線・面・空間メッシュ分割の各ステップごとに，安定配置における隣接バブルからの力を線形バネに置き換えて計算すればよい．こうして算出した線形バネ係数は次のように与えられる．

- 稜線メッシュ分割

  $k=2k_0$

- 面メッシュ分割

  $k_x=4k_0,\ k_y=2\sqrt{3}\,k_0$

- 空間メッシュ分割

  $k_x=6k_0,\ k_y=(2\sqrt{3}+2)k_0,\ k_z=2\sqrt{6}\,k_0$

### 4.4 個数の適応制御

さて，メッシュ分割を行うべく与えられた稜線・面・空間に対して，いったい何個のバブルを並べればバブルどうしが接しながら領域を過不足なく埋められるだろうか？　数が足りなければ隙間ができて指定されたノード距離関数よりも大きな三角形や四面体ができてしまう．また，多すぎても必要以上に要素数を増やしてしまい効率が悪い．

ひとつの方法は，ノード距離関数を重み付け係数として稜線・面・空間の長さ・面積・体積から，各々の領域を埋めるのに必要十分なバブル個数を算出することである．しかし，この方法だけではうまくゆかない．これらをどこに配置するかという問題が残っているからである．例えば**図7**のように領域がくびれている場合に一方の領域にすべてのバブルを初期配置してしまうと，他方にはバブルが移動できない．もちろん，あらかじめくびれが検出できれば，それぞれの小領域ごとに適切な数を算出することも可能だが，どれぐらいの幅をくびれと見なすかは，その位置でのバブルの大きさも考慮する必要があり簡単には判断できない．

そこで，もうひとつの方法として，バブルの人口密度をチェックして個数を適応制御することが考えられる．各バブルごとに，近隣バブルとの重なりの度合いを算出して，小さすぎる場合は「分裂」させて個数を増やし，大きすぎる場合は「破壊」して個数を減らす．こうすれば初期配置に拘わらず適切な個数のバブルが生成される．

重なり度の算出法はいくつか考えられるが，実際に

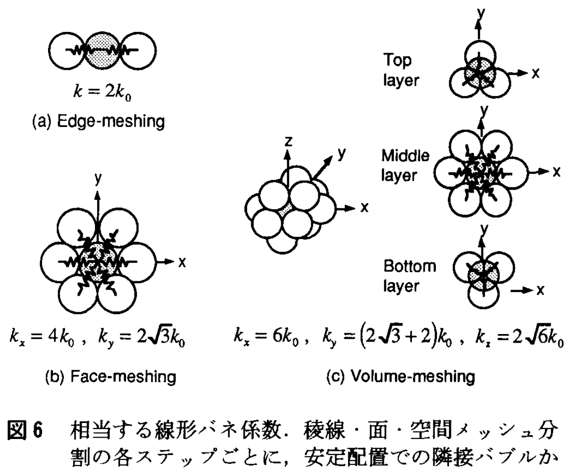



**図6**　相当する線形バネ係数．稜線・面・空間メッシュ分割の各ステップごとに，安定配置での隣接バブルからの力を線形バネに置き換えたものである

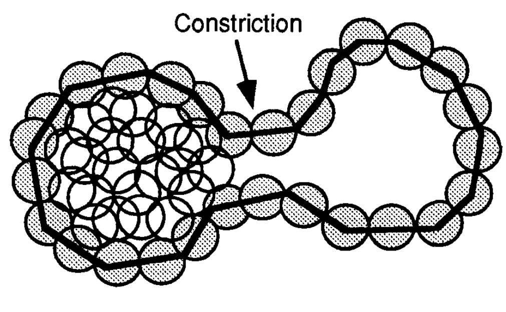



**図7**　くびれた2次元領域への初期バブル配置．一方の領域に多くのバブルを置いてしまうと他方には移動できない．3次元空間でも同じ問題は起こりうる

インプリメントされている方法を，2次元の場合を例にして紹介しよう．バブルを分裂させるかどうかは，**図8**(a)に示すように，着目しているバブルの半径を2倍に膨張させたものを考え，その近隣バブルとの重なりの深さの和をとり，着目しているバブルの半径で割った値で判断する．もし，ノード間距離が一定ならば，6個の近隣バブルと接している理想的状態において，重なり度は600%となる．これが少なすぎる場合にはバブルを2つに分裂させて周辺の人口密度を増やすわけである．

また図8(b)はいつバブルを破壊すればよいかを示している．着目しているバブルと，その近隣バブルとの重なりの深さの和をとり，着目しているバブルの半径で割った値を重なり度とする．この重なり度が大きいことは，この近傍でバブルが過密であることを意味するので，そのバブルを破壊する．

さらに，このようなメカニズムを持つと，解の収束時間を短くすることにも役立つ．初期位置でバブルが過密な部分と過疎な部分があったとしよう．このメカニズムがない場合には過密な部分のバブルが過疎な部分に移動するのを待たねばならない．しかし，過密部のバブルを破壊して過疎度のバブルを分裂させれば，バブルの移動を待つ必要がなくなり，安定解に至るまでの時間を大幅に短縮できるのである．

### 4.5 初期配置

物理モデルによる方法は，幾何学的なアルゴリズムに比べると計算量が多くなりがちなので，可能なかぎりこれを減らすことに留意しなければならない．前節のバブルの分裂と破壊による密度の制御以外で計算量に大きく関与するのは，バブルの初期配置である．

解こうとしているのは，ある安定解に収束するような初期値問題なので，解に近い初期値から始めれば計算時間を短縮できるはずだ．もし，ノード間距離が一定であればこのような初期値を与えるのは比較的簡単である．2次元の場合を例にとると，**図9**に示すような正六角形格子を，与えられた領域に重ねて領域内のあるバブルだけを残せばよい．

しかし，ノード間距離が変化している場合には，このように一定間隔の初期配置では十分ではない．ひとつの方法は，稜線に対しては2分木を，面に対しては4分木（Quadtree）を，空間に対しては8分木（Octree）を用いることである．これらは，領域を階層構造で再帰的に分割していく方法で，CADやコンピュータ・グラフィックスの分野で広く用いられているものだ[26]．

**図10**を使って，2次元の場合について説明しよう．領域とノード間距離が与えられたとき，この領域の中心点にバブルを置いてみる．バブルの直径はノード間距離で指定されたものである．もし，このバブルがこの領域を十分に覆っていれば，そのバブルを残して，これ以上の領域分割をしない．しかし，そうでない場合は，領域を4つに分割してそれぞれの小領域に対して同じ作業を行う．これを適切な初期配置が得られるまで再帰的に繰り返していけばよい．この4分木を使った空間分割において特に工夫されているのは単位セルの形だ．一般には矩形が用いられるが，ここでは内角が60°と120°の平行四辺形を用いている．ノード間距離が一定である場合に図9に示したような正六角形の配置が得られるからである．

## 5. メッシュ分割例とアプリケーション

本章では，第3章と第4章の方法を用いて生成した

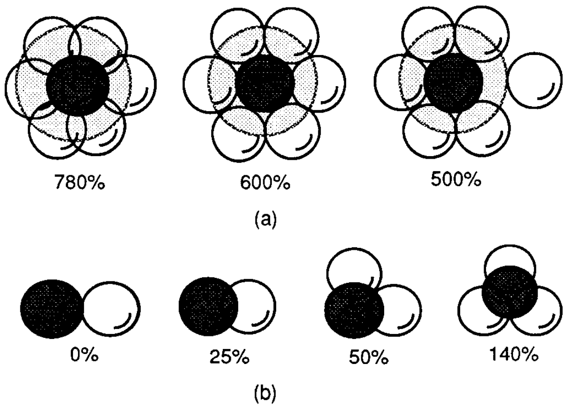



**図8** バブルの重なり度．(a)バブルを分裂させるかどうかを判断するための重なり度．小さすぎる場合にバブルを分裂させ個数を増やす．(b)バブルを破壊するかどうかを判断するための重なり度．大きすぎる場合にバブルを破壊して個数を増やす

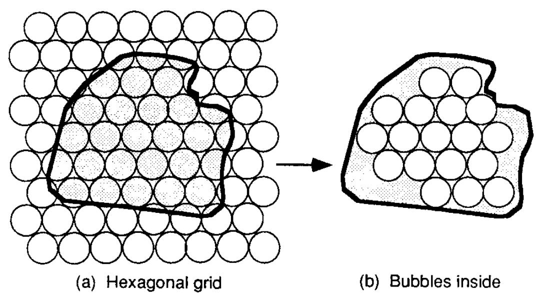



**図9** 正六角形格子を使ったバブルの初期配置方法．格子を与えられた領域に重ねて領域内のあるバブルだけを残す

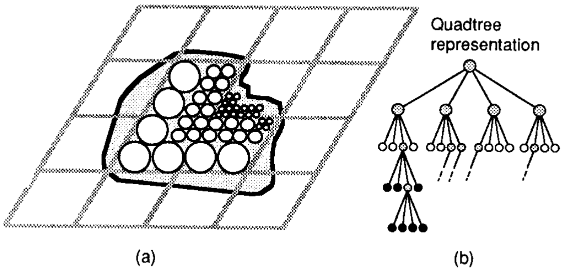



**図10**　4分木（Quadtree）を使ったバブルの初期配置方法．領域中心に置いたバブルが領域を十分に覆っていなければ，領域を4つに再帰的に分割する

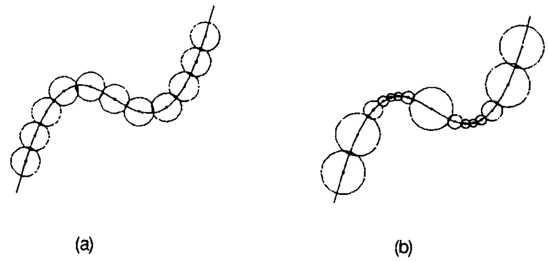



**図11**　平面曲線の線分列近似のためのメッシュ分割．(a)均一のノード距離関数の場合．(b)曲率に応じてもとの曲線と近似線分との差を一定に押さえるようにノード距離関数を定義した場合

稜線・面・空間メッシュ分割例と関連するアプリケーションについて述べる．

**図11**は平面曲線を線分列として近似する稜線メッシュ分割の例である．同図(a)は均一のノード間距離を用いた場合であり，バブルの中心点を線分列で結べば各線分はほぼ同じ長さになる．しかし，このような近似法では曲線の曲率が大きい部分で近似誤差，すなわち元の曲線と近似線分の差が大きくなってしまう．そこで(b)ではノード間距離を曲率に応じて変化するように定義した．曲率が大きい部分はノード間距離を小さく，曲率が小さい部分は大きくしたものである．曲線上のある位置での曲率半径を$\rho$とすると，この近傍での曲線は**図12**に示すような半径$\rho$の円で置き換えられる．ここで，近似誤差を一定値$\varepsilon$に押さえるためのノード間距離分布$d$はピタゴラスの定理から次のように与えられる．

$$d=2\sqrt{2\varepsilon\rho-\varepsilon^2}$$

次に，簡単な平面構造解析の問題を考えよう．中央に円孔をもつ矩形のプレートを両側から引っ張るとする．この形状は横軸と縦軸に線対称であるから，**図13**に示すような四分の一形状だけを考慮すればよい．同図(a)は均一のノード間距離関数によるメッシュ分割例である．しかし，解の精度と収束性を向上させるためには，(b)のように応力が集中する円孔周囲に細密なメッシュを用いるのがよい．実際の解析においては，ノード間距離は過去の経験則や，解の精度を検証する誤差理論から与えられる．

**図14**は，コンピュータ・グラフィックスの表示のために，曲面を三角形で近似した例を示したものである．同図(a)では主曲率をもとに誤差を一定にするようにノード間距離を与えた．(b)は，視点からの距離に反比例するようなノード間距離の例であり，こうすると画面上で表示されるバブルの直径が均一になる．

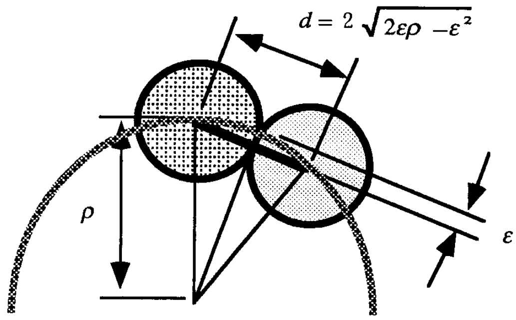



**図12**　曲率にもとづくノード間距離分布．もとの曲線と近似線分との差を一定値内に押さえることができる．曲面を多角形で近似する場合も同様である

つまり，視点から遠い場所には荒いメッシュが生成されて，必要以上に多くの三角形を表示することがない．

最後に，3次元CADモデルを多面体に分割する手順を，**図15**に示す．同図(a)はあるトーラス形状から別のトーラス形状を切り取ってできた3次元領域である．この領域をメッシュ分割するには(b)，(c)，(d)のように，順に頂点バブル，稜線バブル，面バブルを配置してゆく．最後に立体の内側の空間に空間バブルを詰めて，これらの全バブルの中心点を*Delaunay*の方法で結んでやれば四面体メッシュができる．

## 6. おわりに

メッシュ分割問題はさまざまなコンピュータ・アプリケーションに必要となる基本的問題である．本稿では，従来の幾何学的なアルゴリズムではなく，物理モデルと動力学シミュレーションによって，指定されたノード間距離分布に従いながら歪みが少ないメッシュを得る方法を紹介した．第2章で示した基本的な考え方はメッシュ分割以外の問題にも適用できる可能性が

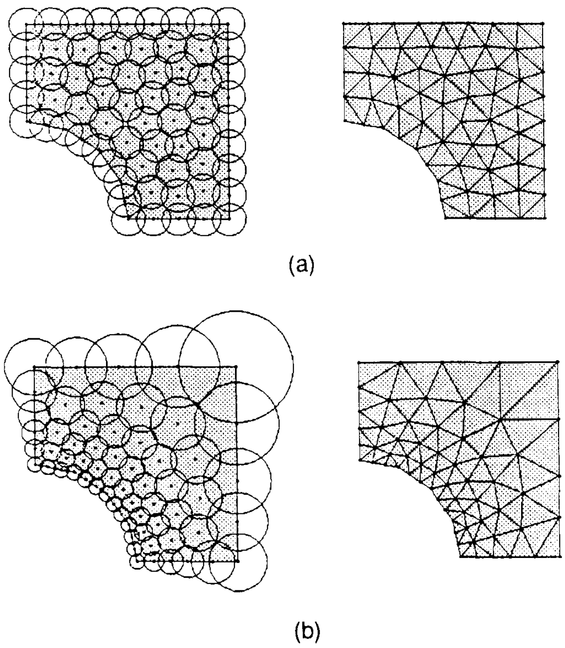



**図13** 構造解析のための2次元領域メッシュ分割例．中央に円孔をもつ矩形のプレートの四分の一の形状．(a)均一のノード間距離によるメッシュ分割例．(b)解の精度と収束性を向上させるためには，応力が集中する円孔周囲に細密なメッシュを用いるのがよい

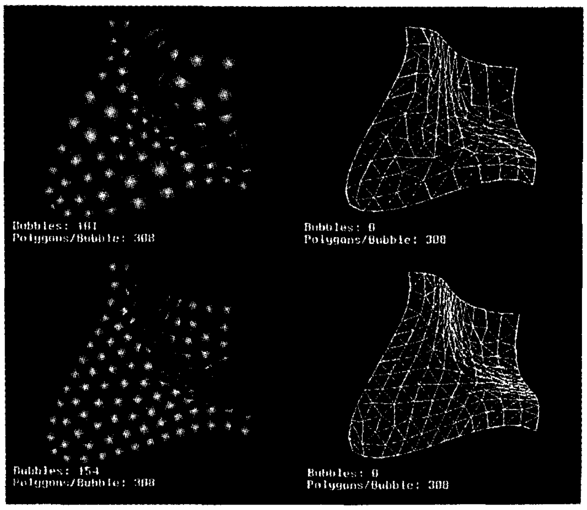


**図14** 曲面の三角形メッシュ近似．(a)曲率に応じて，もとの曲面と近似多面体との差を一定に押さえるようにノード距離関数を定義した場合．(b)画面上で要素の大きさが均一になるように視点からの距離に応じてノード距離関数を定義した場合

あるだろう．これまで幾何学的発想だけで解こうとして行き詰まっていた他の形状処理問題に対して，別の角度からアプローチする手掛かりを本稿が提供できたとしたら幸いである．

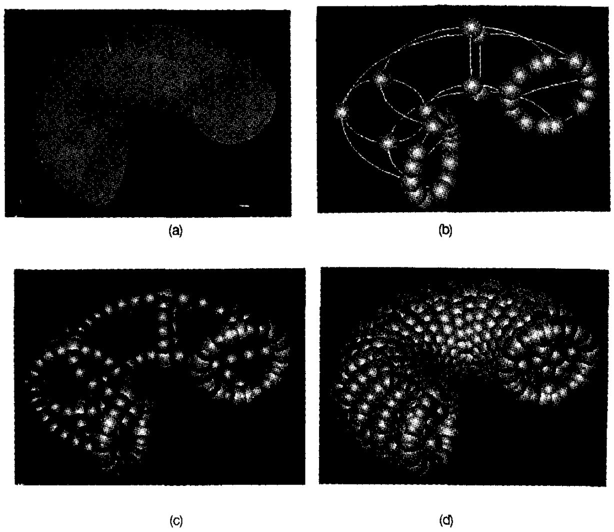



**図15** CADモデルからの3次元メッシュ分割．(a)与えられたCADモデル．(b)頂点上にバブルを配置したところ．(c)稜線上にバブルを配置したところ．(d)面上にバブルを配置したところ．この後は，内部の空間内にバブルを詰めればよい

## 参 考 文 献

1) 矢川，吉村：計算力学のためのエキスパートシステム，応用数理，2-2, 144/169 (1992)

2) X. Sheng and B. E. Hirsch: Triangulation of trimmed surfaces in parametric space, Computer-Aided Design, 24-8, 437/444 (1992)

3) K. Ho-Le: Finite element mesh generation methods: a review and classification, Computer-Aided Design, 20-1, 27/38 (1988)

4) J. C. Cavendish, D. A. Field and W. H. Frey: An approach to automatic three-dimensional finite element mesh generation, International Journal for Numerical Methods in Engineering, 21, 329/347 (1985)

5) D. A. Field and W. D. Smith: Graded tetrahedral finite element meshes, International Journal for Numerical Methods in Engineering, 31, 413/425 (1991)

6) N. Sapidis and R. Perucchio: Advanced techniques for automatic finite element meshing from solid models, Computer-Aided Design, 21-4, 248/253 (1989)

7) M. S. Shephard, K. R. Grice, J. A. Lo and W. J. Schroeder; Trends in automatic three-dimensional mesh generation, Computers & Structures, 30-1/2, 421/429 (1988)

8) E. A. Sadek: A scheme for the automatic generation of triangular finite elements, International Journal for Numerical Methods in Engineering, 15, 1813/1822 (1980)

9) A. Bykat: Automatic generation of triangular grid: I-subdivision of a general polygon into convex subregions. II-triangulation of convex polygons, International Journal for Numerical Methods in Engineering, 10, 1329/1342 (1976)

10) B. Joe and R. B. Simpson: Triangular meshes for regions of complicated shape, International Journal for Numerical Methods in Engineering, 23, 751/778 (1986)

11) M. A. Yerry and M. S. Shephard: A modified quadtree approach to finite element mesh generation, IEEE Computer Graphics and Applications, 3-1, 39/46 (1983)
12) R. Perucchio, M. Saxena and A. Kela: Automatic mesh generation from solid models based on recursive spatial decompositions, International Journal for Numerical Methods in Engineering, 28, 2469/2501 (1989)
13) P. L. Baehmann, S. L. Wittchen, M. S. Shephard, K. R. Grice and M. A. Yarry: Robust, geometrically based, automatic two-dimensional mesh generation, International Journal for Numerical Methods in Engineering, 24, 1043/1078 (1987)
14) B. Wordenweber: Finite element mesh generation, Computer-Aided Design, 16-5, 285/291 (1984)
15) T. C. Woo and T. Thomasma: An algorithm for generating solid elements in objects with holes, Computers and Structures, 18-2, 333/342 (1984)
16) J. C. Cavendish: Automatic triangulation of arbitary planar domains for the finite element method, International Journal for Numerical Methods in Engineering, 8, 679/696 (1974)
17) W. H. Frey: Selective refinement: a new strategy for automatic node placement in graded triangular meshes, International Journal for Numerical Methods in Engineering, 24, 2183/2200 (1987)
18) S. H. Lo: Delaunay triangulation of non-convex planar domains, International Journal for Numerical Methods in Engineering, 28, 2695/2707 (1989)
19) B. Joe: Construction of three-dimensional Delaunay triangulations using local transformations, Computer-Aided Geometric Design, 8, 123/142 (1991)
20) K. Shimada and D. C. Gossard: Computational Methods for Physically-based FE Mesh Generation, Proc. of the IFIP TC5/WG 5.3 Eight International PROLAMAT Conference, 411/420 (1992)
21) J. E. Barnes and P. Hut: A hierarchical O(n log n) force calculation algorithm, Nature, 324-4, 446/449 (1986)
22) L. Greengard and V. Rokhlin: A fast algorithm for particle simulation, Journal of Computational Physics, 73-325 (1987)
23) J. P. Singh, C. Holt, T. Totsuka, A. Gupta and J. L. Hennessy: Load balancing and data locality in hierarchical n-body methods, Stanford University Technical Report, NO. CSL-TR-92-505 (1992)
24) W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling: Numerical recipes in C: the art of scientific computing, Cambridge University Press (1988)
25) K. Ogata: Modern Control Engineering, Prentice-Hall, Inc., 225/247 (1970)
26) M. E. Mortenson: Geometric modeling, John Wiley & Sons, 450/455 (1985)