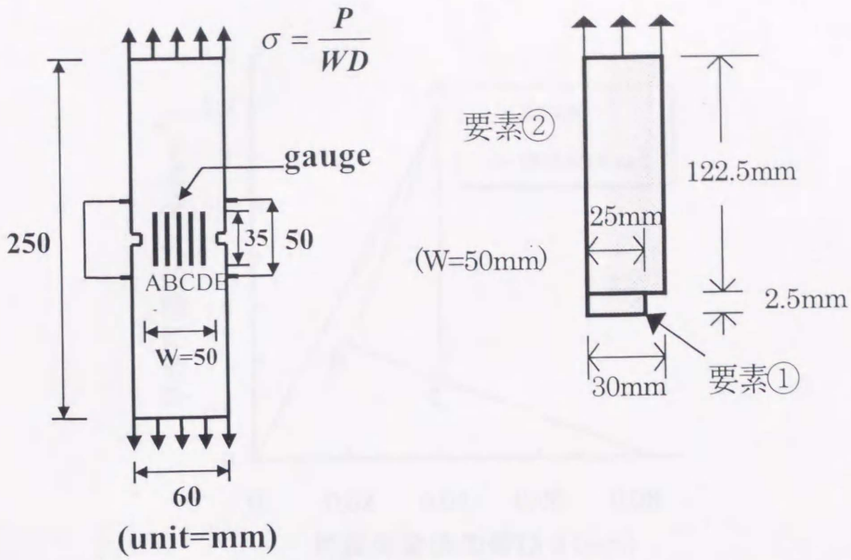
琉球大学学術リポジトリ

コンクリートの引張破壊挙動に関する解析的研究

メタデータ	言語: Japanese 出版者: 富山潤 公開日: 2021-12-15 キーワード (Ja): キーワード (En): 作成者: 富山, 潤, Tomiyama, Jun メールアドレス: 所属:
URL	http://hdl.handle.net/20.500.12000/25354



(a)供試体　　(b)要素分割

図 2-8　直接引張試験体と要素分割図

Rmin 法を改良した計算例は，剛性方程式の未知数が2で，増分の回数もわずか3回で解を求めている。静的解析では，図 2-9 に示す破壊点 A から B，C とスナップバックが現われている。しかし，この方法では，スナップバックが現われると強制変位増分（または荷重増分）を引張から圧縮に作用させるという特別な処理が必要となる。動的解析では，A から直接 C まで急激に応力が低下し，C から後は両解析方法とも一致している。すなわち，本増分解析法は，増分の途中にスナップバックが現われた場合でも，動的釣合式を用いるため，特別な処理をすることなくスナップバック領域を避けて解を得ることが出来る。なお，この解析に用いた時間増分 Δt，強制変位の増分 Δu は，それぞれ $\Delta t = 1.142 \times 10^{-4}$ sec，$\Delta u = 0.0001mm$ である。

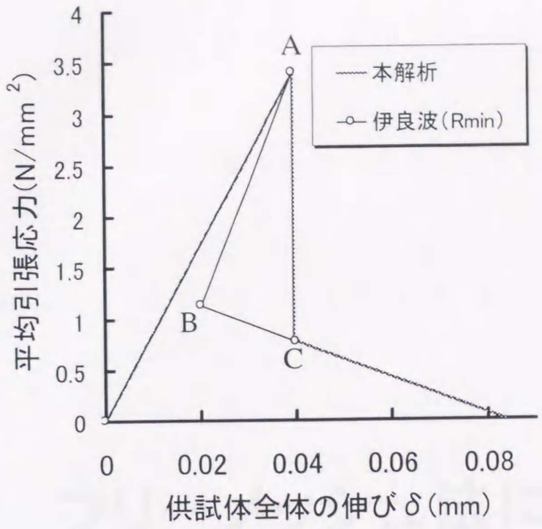


図 2-9 平均引張応力ー伸び

2. 6 まとめ

コンクリートの引張破壊挙動を数値解析するためには，引張軟化特性を考慮する必要がある。いったん軟化が生じると，見かけの剛性が負となるために，スナップバックのような不安定現象が起生する場合がある。このような場合，静的釣合式を用いた増分法では軟化以降の解析が困難となる。スナップバック現象を解析出来る増分法として，弧長増分法や Rmin 法を応用した増分法などがあげられるが，前者は解析アルゴリズムが複雑で，後者は引張軟化の進行や除荷が頻繁に生じると，安定した解が得られないことがある。そこで本章では，コンクリートの引張軟化特性を考慮した一増分法として，動的釣合式である運動方程式を用いた増分法を示し，その有効性を論じた。

有効性を検討するための例として，スナップバックが生じる両側中央切欠きを有するコンクリート棒の直接引張試験をとりあげ，簡単なトラス要素を用いて解析した。

静的釣合式を用いた Rmin 法を応用した増分法と本解析結果を比較した結果，静的釣合式を用いた Rmin 法を応用した増分法[7]では，スナップバックが生じると強制変位増分（または，荷重増分）を引張から圧縮に作用させるという特別な処理が必要となるのに対して，本解析法は，動的釣合式を用いているため，軟化以降にスナップバック現象が生じても，特別な処理をすることなく，その領域を避けて安定した解を得ることが出来る。

第3章　フリーメッシュ法によるコンクリートの引張破壊挙動の解析法

第3章 フリーメッシュ法によるコンクリートの引張破壊挙動の解析法

3．1　はじめに

フリーメッシュ法と有限要素法との違いを図3-1(a)に示す。図3-1(a)に示すように有限要素法では，プレプロセスとして要素生成が必要であり，要素生成と剛性マトリックスの組立および解析を別々に行う。これに対しフリーメッシュ法では，プレプロセスとして入力データである節点情報を作成し，その後の要素生成，剛性マトリックスの組立および解析までをシームレス(継ぎ目なく)に行うことが出来る解析手法である。

さらに本法は，グローバルな要素生成は必要とせず，あるひとつの節点に着目してこれを中心節点とする。そして，その中心節点のまわりにある節点群からローカル要素生成に必要な最適な節点を選択する(図3-1(b)参照)。その節点を衛星節点と呼ぶ。

次に，ローカル要素より要素剛性マトリックスを計算し，全体剛性マトリックスに足し合わせる。その足し合わせ方も独特で，中心節点まわりのすべてのローカル要素について中心節点に寄与する要素剛性マトリックスの行成分を足し合わせるのである。これを全節点に対して行う。そのイメージを図3-2に示す。

上記のプロセスを全ての節点で行い，最後に連立一次方程式を解く。その連立一次方程式を解く手法としては，従来有限要素法で用いられた直接法や反復法を用いることが出来る。

このようにフリーメッシュ法は，中心節点に関する処理を他の節点と独立して行うことが出来ることから，節点ベース有限要素法とも呼称されている。また，節点ベースで解析が可能であるため，並列計算に適している。

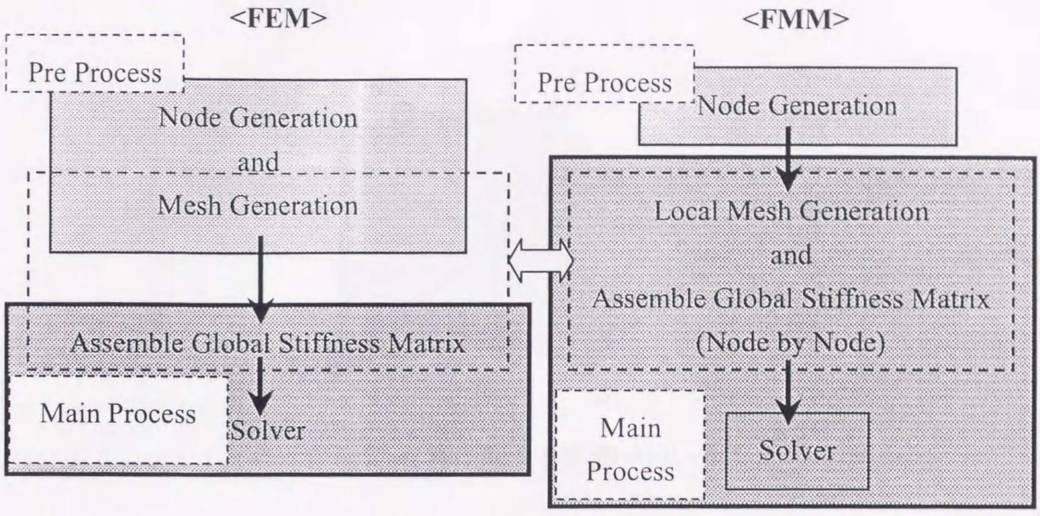
以上より，フリーメッシュ法は，解析データとして節点情報を与えるだけで，その後の解析プロセスをシームレスに行うことが出来，さらに並列形計算に適しているために大規模な解析にも適用可能である。このため，フリーメッシュ法は，従来の有限要素法では困難であると考えられる大規模化・複雑形状化する構造物のモデル作成および計算が比較的容易に行うことが出来る有力な解析ツールであると考えられる。

フリーメッシュ法の適用範囲としては，熱伝導問題[10]，流体問題[27]，弾塑性大変形問題[28]などがあげられる。

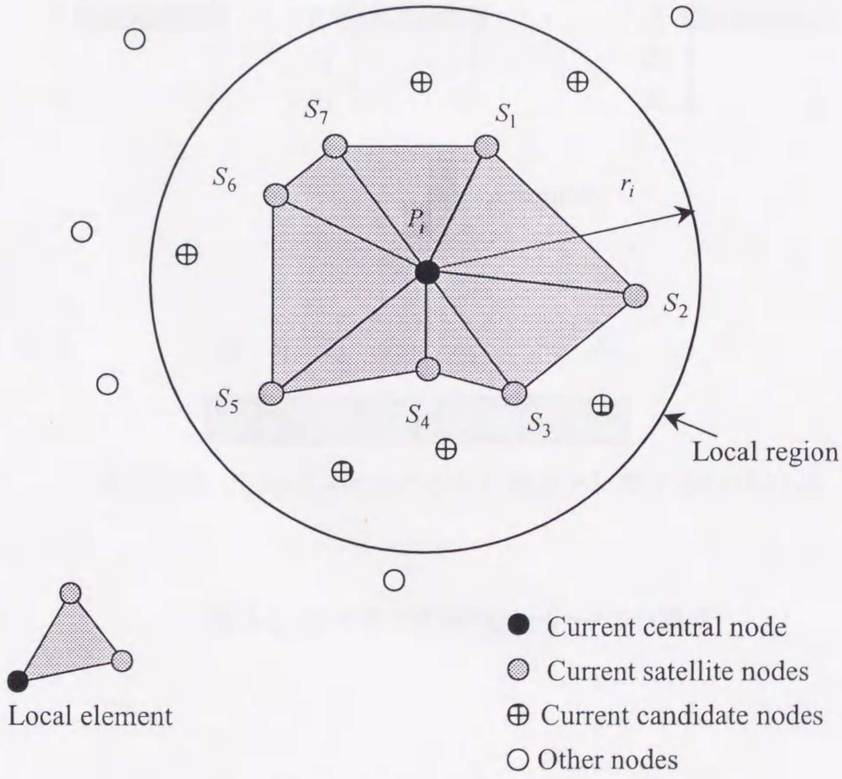
本論文では，このフリーメッシュ法を初めて材料非線形問題であるコンクリート構造物の破壊解析に適用した。増分解析方法として，第2章で示した増分法を用いた。

本章では，はじめにフリーメッシュ法の解析アルゴリズムを述べる。それからフリーメッ

シュ法をコンクリートの破壊挙動の解析に適応する上での問題点と解決法および解析方法を述べる。



(a) 有限要素法とフリーメッシュ法の違い



(b)中心節点とローカル領域，ローカル要素

図 3-1 フリーメッシュ法

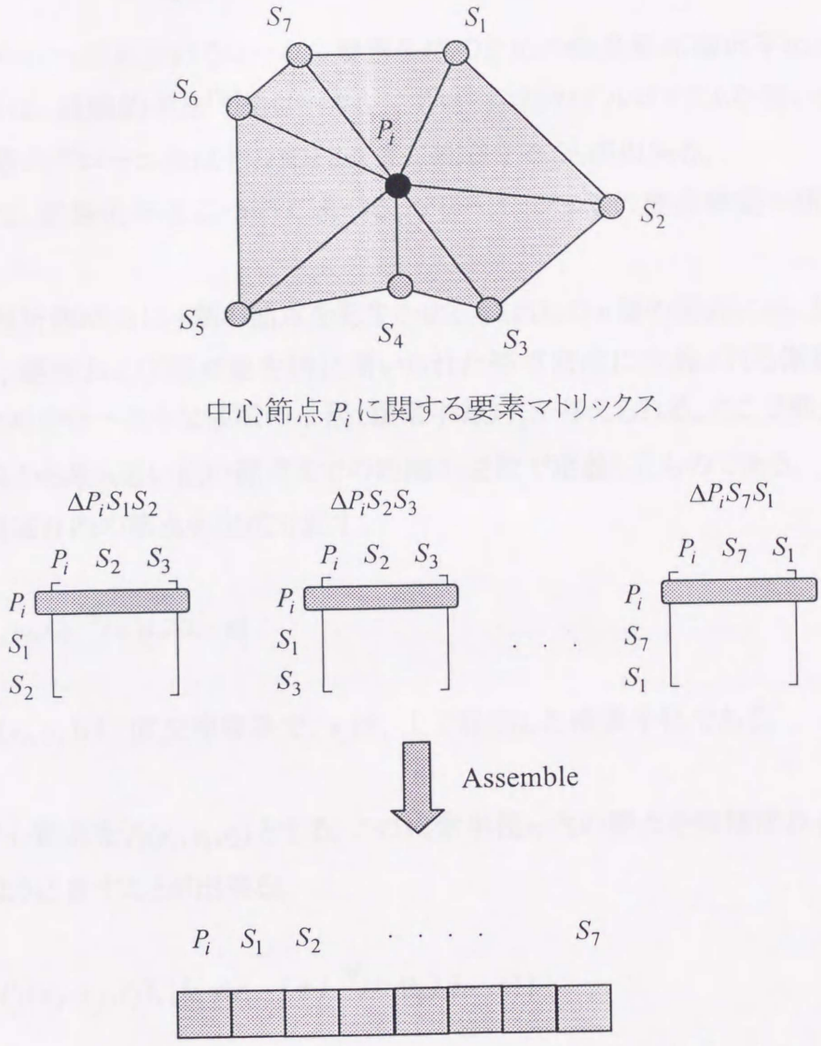


図 3-2 全体系での剛性マトリックスの構成

3．2 要素の生成および構築

3．2．1 衛星節点選択手法

フリーメッシュ法におけるローカル要素生成のための衛星節点選択手法として，2次元であれば，経験的手法[10]やローカルにデローニ法のアルゴリズムを用いるもの[29]がある。後者のデローニ法はそのまま3次元に拡張することが出来る。

ここでは，経験的手法について述べる。フリーメッシュ法の要素構築手順を，図 3-3 に示す。

まず，解析領域Ωにn個の節点を発生させる。これらのn個の節点には，図 3-4 に示すように，座標および節点発生時に用いられた節点密度に定義される衛星節点を検索するためのローカルな領域の半径（検索半径）r_iが与えられる。ここで節点密度とは，その節点から最も近い他の節点までの距離の逆数で定義したものである。

解析領域Ω内の節点を次式で表す。

$$P_i(x_i, y_i, r_i),\ {}^{\forall}i \in \{1,2,3,\cdots n\} \tag{3.1}$$

ここで，(x_i, y_i)は，直交座標系で，r_iは，上で説明した検索半径である。

今，中心節点を$P_i(x_i, y_i, r_i)$とする。この検索半径r_i内の節点を候補節点C_iとすると，次式のように表すことが出来る。

$$C_i = \{P_j(x_j, y_j, r_j), \mid l_{ij} < r_i,\ j \neq i\ \ {}^{\forall}j \in \{1,2,3,\cdots n\}\} \tag{3.2}$$

ここで，l_{ij}は，中心節点と候補節点間の距離である。

候補節点数をn_cで表すと，式(3.2)は

$$C_i = \left\{P_1^{c(i)}, \cdots, P_{n_c}^{c(i)}\right\} \tag{3.3}$$

と表すことが出来る。

なお，式(3.2)を満たす候補節点C_iを探す際に，解析領域内のすべての節点に対して上記のチェックを行うには，$O(n^2)$の計算時間がかかるので，本研究では，この検索の高速化を計るために，空間に関する一種のハッシュ法であるバケット法（Bucket

Method)[30]を用いている。バケット法については，3．2．3(2)節で述べる。

この候補節点C_iから，衛星節点S_iおよびローカル要素L_iを決定するアルゴリズムは，以下のようになる[10][31]。

まず，図 3-5 のように候補節点どうしを結合させ，任意の組み合わせからつくられる次のような集合を考え，

$$\{\overline{C_j^i C_k^i} \mid j<k,\ {}^{\forall}j,k \in \{1,2,3\cdots,n_c\} \tag{3.4}$$

この線分を長さの短い順に以下に示す判定を行う。もし，長さの等しい線分がある場合は，何らかの判定基準を決め（例えば節点番号の小さいものを含む線分をより短いとみなすなど），一意に決定する順位付けを行う。

また，中心節点とそれぞれの線分に対して，2つの領域を定義する。$\Delta P_i P_j P_k$の内部領域および外部領域である（図 3-6，図 3-7 参照）。

（1） $\Delta P_i P_j P_k$の内部領域に他の候補節点があれば，現在処理している線分$\overline{C_j^i C_k^i}$についての判定は行わず，次の線分の判定に移る。

（2） 外部領域内に候補節点が存在しないならば，次の線分に移る。

（3） 外部領域に他の候補節点が存在する場合，それらを$\hat{C}_l^i,(l=1,2,3\cdots,\lambda)$とする。ここで$\lambda$は外部領域内に存在する候補節点数である。このとき，

$$\left\{\overline{C_j^i C_k^i} < \overline{P_i \hat{C}_l^i}\ {}^{\forall}l \in \{1,2,3,\cdots,\lambda\}\right\} \tag{3.5}$$

ならば，すべての$\hat{C}_l^i$を候補節点から除外する。

（4） 以上のことをすべての線分について繰り返し，最終的に残った候補節点が中心節点P_iのまわりの衛星節点となる。

このようにして，最終的に中心節点$P_i(x_i,y_i,r_i)$まわりに衛星節点S_iとして，

$$S_i = \left\{P_1^{S(i)},\cdots,P_{n_s}^{S(i)}\right\} \tag{3.6}$$

が残ることになる。ここで，n_sは中心節点P_iまわりの衛星節点数である$(n_s \le n_c)$。

ローカル要素を作る前に，この衛星節点を時計まわり（あるいは反時計まわり）に並べ

ると，中心節点 P_i に関する衛星節点は，次式のように表せる。

$$S_i' = \left\{ P_1^{S(i)'}, \cdots, P_{n_s}^{S(i)'} \right\} \tag{3.7}$$

最後に，中心節点と隣接する二つの衛星節点をつなぐことによって，ローカル要素が生成される（図 3-8 参照）。 境界以外では，ローカル要素数 n_e と衛星節点数 n_s は等しくなる。

中心節点に関する行ベクトルは，各ローカル要素から計算する。各ローカル要素の剛性マトリックスは，次式で表される。

$$[K]^{e(i)} = \begin{bmatrix} k_i^{e(i)^T} \\ k_j^{e(i)^T} \\ k_k^{e(i)^T} \end{bmatrix}, \ (i = 1,2,3,\cdots,n) \tag{3.8}$$

ここで， $k_i^{e(i)^T}$ は，現在の中心節点に関する $[K]^{e(i)}$ の行ベクトルであり， $k_j^{e(i)^T}$ ， $k_k^{e(i)^T}$ は，現在の衛星節点に関する行ベクトルである。

この中心節点に関係する「節点マトリックス」 $k^{L(i)^T}$ は，次のようになる。

$$k^{L(i)^T} = \sum_1^{n_e} k_i^{e(i)^T} \tag{3.9}$$

これは，第 i 番目の節点に寄与する節点マトリックス，言い換えると全体剛性マトリックスのうちの第 i 番目（現在の中心節点）の行ベクトル成分を表している。ゆえに，全体剛性マトリックスは，

$$[K] = \begin{bmatrix} k^{L(1)^T} \\ k^{L(2)^T} \\ \cdot \\ \cdot \\ \cdot \\ k^{L(n)^T} \end{bmatrix} \tag{3.10}$$

となる。

この全体剛性マトリックス$[K]$の記憶方法として，従来の有限要素法と同様に，スカイライン法，バンドで記憶する方法，非ゼロ成分のみを記憶する方法などが考えられる。このうち，非ゼロ成分のみを記憶する方法は，節点のナンバリングに依存することなく記憶容量を抑えることが出来るうえ，反復解法に適している[32]ため，並列計算も行いやすいという特徴がある。

フリーメッシュ法では，節点ごとに上記の節点マトリックスを作成し全体剛性マトリックスに足し合わせており，全体剛性マトリックスの中心節点に寄与する行ベクトルのなかで非ゼロ成分を持つのは中心節点および衛星節点に対応する行成分だけである（図3-9 参照）。このため，非ゼロ成分とともに，中心節点および衛星節点の節点番号を記憶することで，全体剛性的マトリックスに関する任意の計算を行うことが出来る。なお，本研究では，全体剛性マトリックスの記憶方法として非ゼロ成分のみを記憶する方法を採用している。非ゼロ成分を記憶する際に必要な非ゼロ成分の位置を示すインデックスは衛星節点の並びとなっている（図 3-9 参照）

ここで，フリーメッシュ法の全体剛性マトリックスの組み立ては，同じ三角形要素を3回参照することになり，剛性マトリックス作成時間は，FEM に比べ単純に3倍の時間を要することになる。しかし，節点単位で全ての処理が可能であるため，要素単位で処理するFEMよりも並列分散処理に適している。

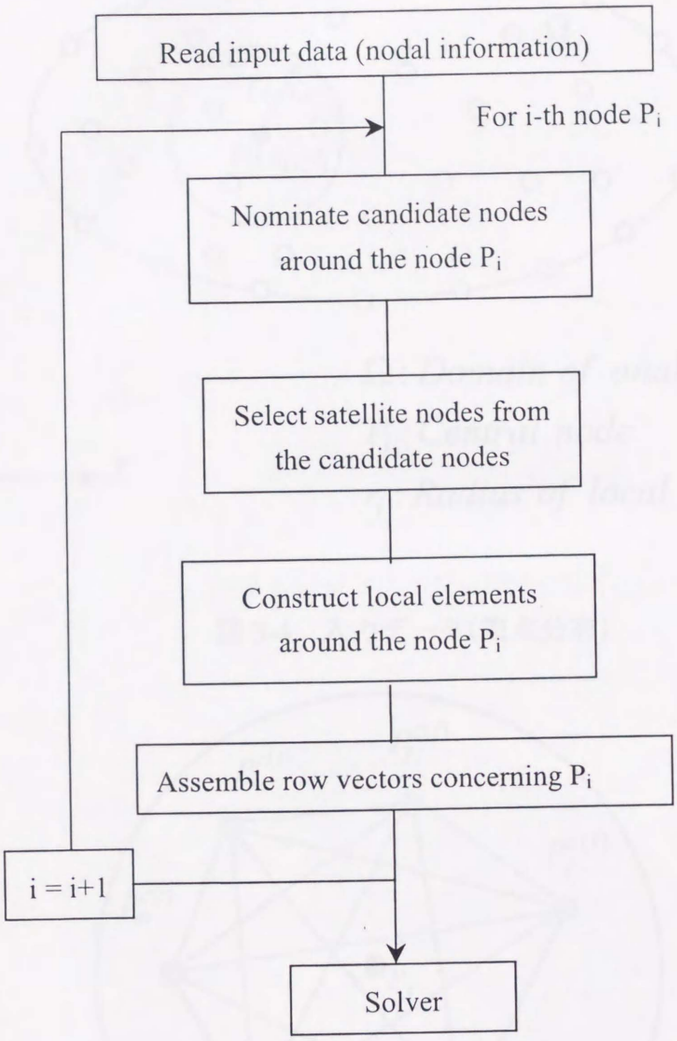


図 3-3 フリーメッシュ法の要素構築手順

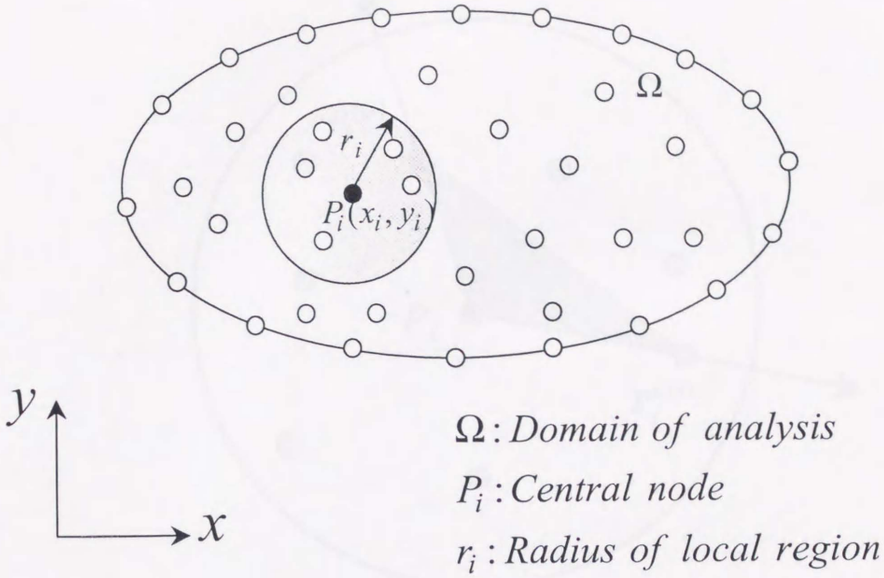


図 3-4　入力データ(節点分布)

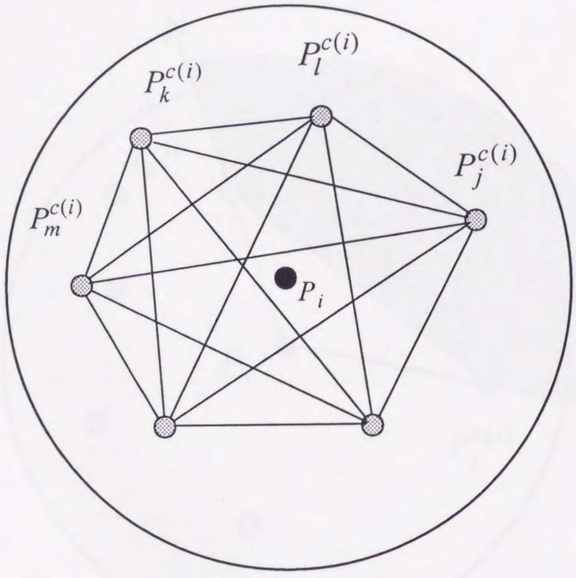


図 3-5　衛星節点どうしのすべての線分の組み合わせ

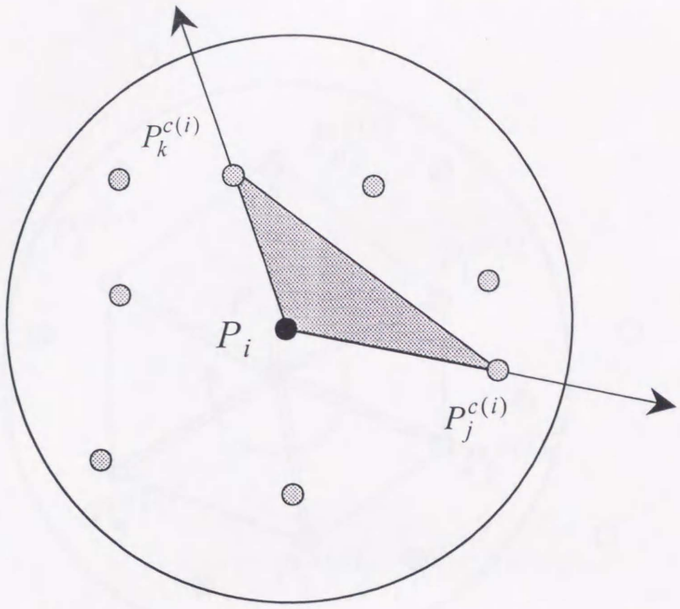


図 3-6　$\Delta P_i P_j^{c(i)} P_k^{c(i)}$ 内部の領域

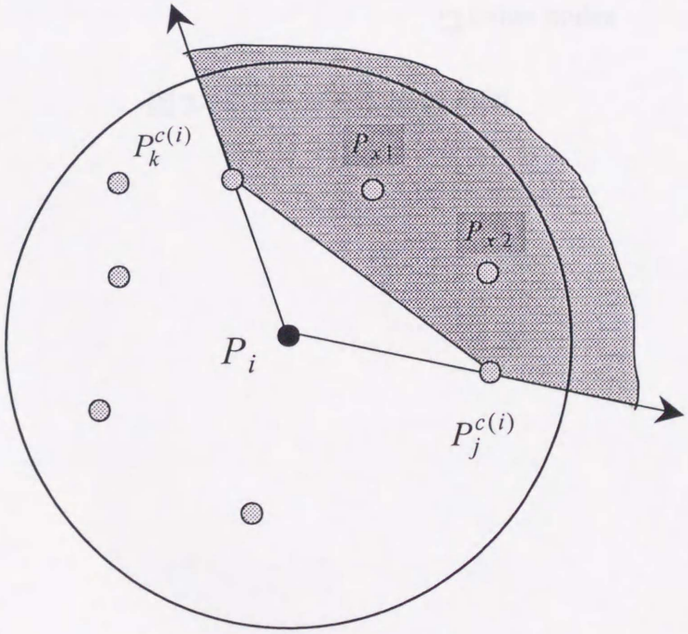


図 3-7　$\Delta P_i P_j^{c(i)} P_k^{c(i)}$ 外部の領域

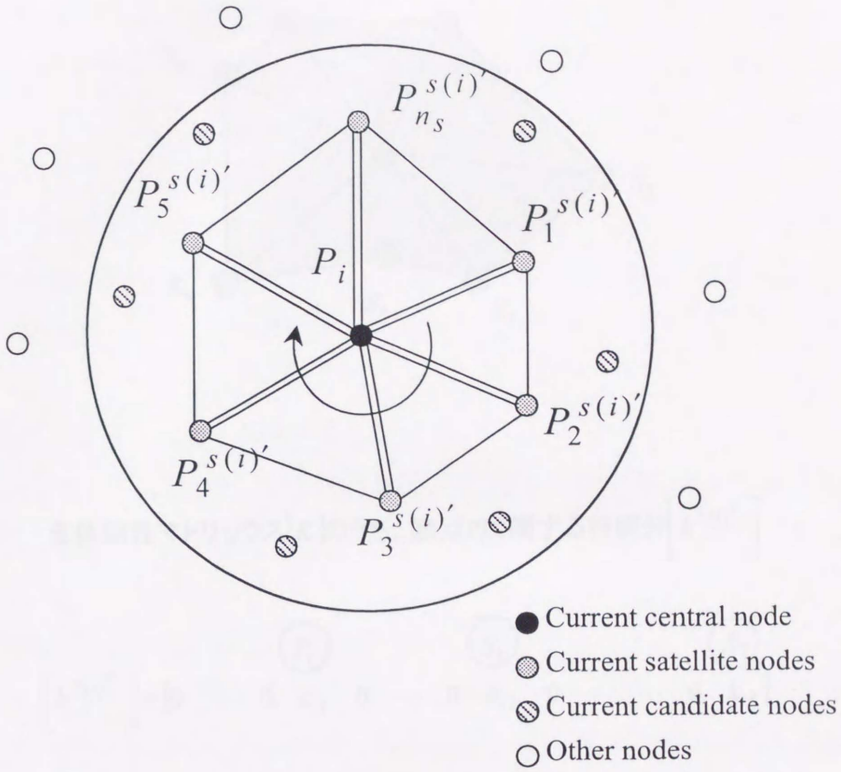
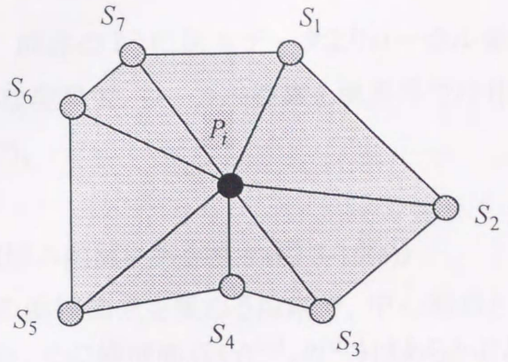


図 3-8　ローカル要素概念図



全体剛性マトリックス$[K]$のうち，節点P_iに関する行成分$\left[k^{L(i)^T}\right]$

$$\left[k^{L(i)^T}\right]=[0 \ \cdots \ 0 \ \overset{P_i}{k_{i1}} \ 0 \ \cdots \ 0 \ \overset{S_1}{k_{i2}} \ 0 \ \cdots \ \cdots \ 0 \ \overset{S_7}{k_{i8}}]$$

$$\Downarrow$$

$$\left[k^{L(i)^{T'}}\right]=[k_{i1} \quad k_{i2} \quad k_{i3} \quad k_{i4} \quad k_{i5} \quad k_{i6} \quad k_{i7} \quad k_{i8}]$$

$$+$$

$$index\left[k^{L(i)^{T'}}\right]=[P_i \quad S_1 \quad S_2 \quad S_3 \quad S_4 \quad S_5 \quad S_6 \quad S_7]$$

図 3-9　非ゼロ成分の記憶

3. 2. 2　複雑境界の内外判定

フリーメッシュ法では，前述のように節点データよりローカル要素を自動的に生成するが，以下に示す複雑な境界で，ローカル要素を境界外では作成しないように，境界に関する内外判定を行う。

(1) 中心節点が解析領域の内部にある場合（図 3-10(a)）

中心節点のまわりの候補節点を集める段階で，中心節点と候補節点を結ぶ線が境界線と交わる場合，その候補節点（$P_1^{c(i)}, P_2^{c(i)}$）はあらかじめ除外する。

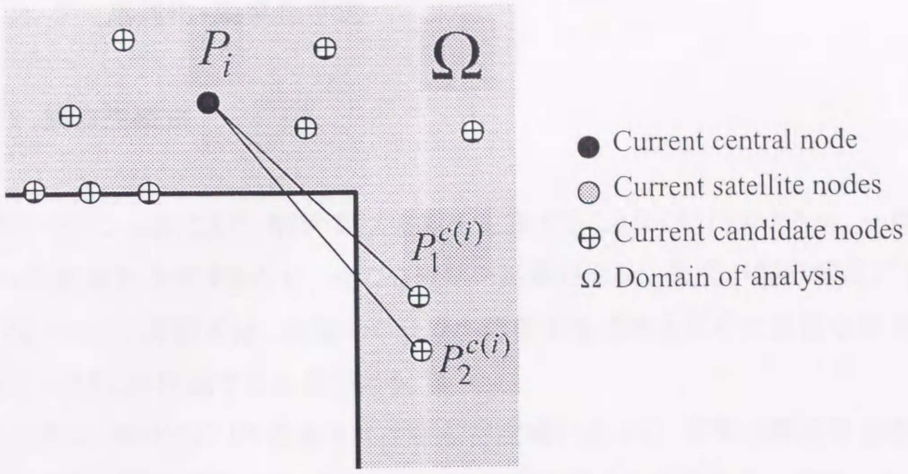
(2) 中心節点が境界上にある場合（図 3-10(b)）

(1)の判定を行った上で，着目中心節点が位置する境界線以外の境界線上に候補節点が残っていれば，これらの候補節点（$P_3^{c(i)}, P_4^{c(i)}, P_5^{c(i)}$）を除外する。

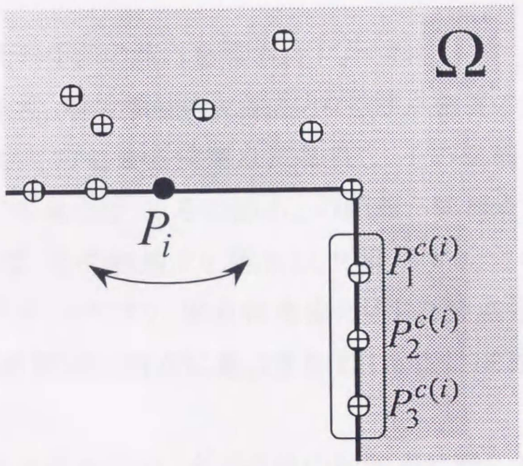
(3) 中心節点が頂点にある場合（図 3-10(c)）

同様に(1)，(2)の処理を行う。ただし，この場合，中心節点は2つの境界線上に位置することになるが，これらを1つの境界線とみなして判定を行う。凸型の頂点には特別な処理を必要としないが，図のような凹型の頂点の場合は，最終的に中心節点と衛星節点からローカルな要素を作る時点で，領域外部の要素となる $\Delta P_i S_1^{c(i)}, S_2^{c(i)}$ をローカルな要素としない。

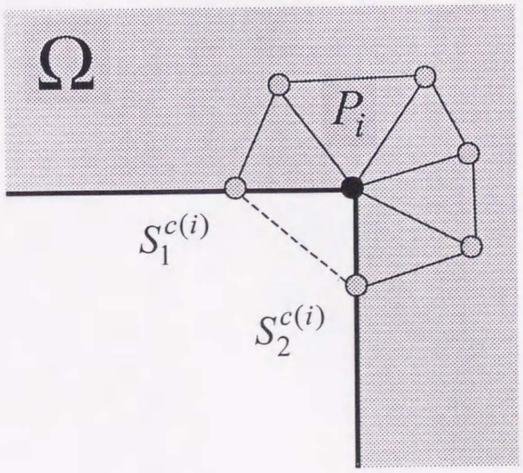
以上のような処理を行うことにより，凹型の境界を含む複雑な解析領域にも対応することが出来る。



(a)中心節点が解析領域内部にある場合



(b)中心節点が解析境界上にある場合



(c)中心節点が頂点にある場合

図 3-10　境界に対して内外判定が必要な場合の処理

3. 2. 3　高速化・効率化手法

(1) 節点生成法

フリーメッシュ法により，解析者は要素を意識することなく解析できるが，一時的にローカルな要素を作成するため，そのローカル要素の形状，密度が解析精度に大きく影響する。つまり，解析者は，良質のローカル要素を生成するために良質な節点分布を解析データとして作成する必要がある。

ここでは，矢川らにより提案された節点密度場によって，良質な節点発生を行うことが出来る手法[30]を紹介する。この手法は3次元の場合でもそのまま適用することが出来る。

その手法の流れを図 3-11 に示す。

まず，解析領域内に基本となる節点密度の5，6倍の密度で候補点をあらかじめ発生させておく。ある候補点に着目し，その候補点から最も近い節点（すでに採用された点）を見つけて，その節点との距離が候補点の位置での節点密度値の逆数より大きければ，その候補点を節点として採用する。この操作を全ての候補点に対して繰り返して行うことにより，節点密度場に応じた節点分布を得ることが出来る。実際の計算では，解析領域の頂点に節点を配置し，次に境界線，境界内部の順に節点を発生していく。

ある候補点で，全ての他の候補点に対して上記の判定を行うと，かなりの時間を要する。この判定を高速化する手法であるバケット法[30]を次の節で概説する。

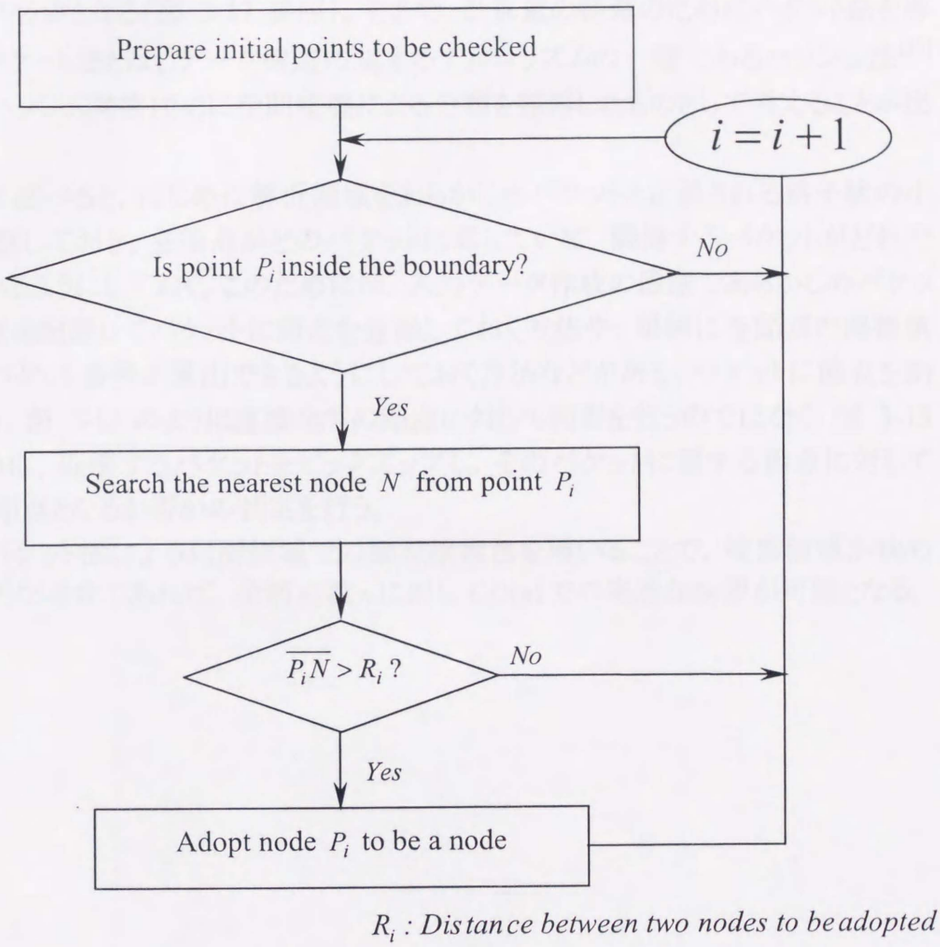


図 3-11 節点生成アルゴリズム(矢川ら)

(2) バケット法

フリーメッシュ法では，各節点ごとに一度，周囲の節点を集める必要がある。これは一時的にローカルな要素を生成するに当たり，必要となる衛星節点を選択するためのプロセスであるが，単純にこれを全ての節点について行うと，総節点数の2乗の計算量（$O(n^2)$）が必要となる（図 3-12 参照）。そこで，計算量の軽減のためにバケット法を導入する。バケット法とは，データ構造に関するアルゴリズムの一種であるハッシュ法[30]においてハッシュ関数H(x)に空間座標による分類を採用したものとして考えることが出来る。

具体的に述べると，はじめに解析領域をあらかじめバケットと定義される格子状の小領域に分割しておき，各節点がどのバケットに属していて，隣接するバケットがどれであるかわかるようにしておく。このためには，入力データ作成の段階であらかじめバケットに各節点を配置してバケットに節点を登録しておく方法や，単純に各節点の座標値のみからバケット番号が算出できるようにしておく方法などがある。バケットに節点を割り振った後，図 3-12 のように直接全ての節点に対して判定を行うのではなく，図 3-13 に示すように，近接するバケットをピックアップし，そのバケットに属する節点に対してのみ候補節点となるか否かの判定を行う。

上記のバケット法による局所領域での節点検索法を用いることで，検索領域が狭められ理想的な場合であれば，全節点数nに対して$O(n)$での高速な検索が可能となる。