東京大学　新領域創成科学研究科
社会文化環境学専攻
2019 年度　修士論文

Lattice3 形式の立体骨組の
最適形状を生成するアルゴリズム

Lattice3, a Generative Algorithm
For Shape-Optimized 3-Dimensional Structural Frame

2019 年 1 月 20 日提出

指導教員　准教授　佐藤淳

藤本　月穂
Fujimoto　Tsukiho

目次

1．序

１－１　背景

地球環境問題に対し、省エネルギー・省資源の対策が今も求め続けられている。

軽量かつ強固であるラチス構造は、19 世紀、材料の削減のために多く使われていた。手間がかかるため人件費は高くつくが、それをまかなえるだけの価値が鉄にあったためである。万国博覧会のクリスタルパレスなど、華々しい新たな建築たちを支えた技術であった。
しかし、時代を経て鉄が安価になるとともにこの工法は廃れていき、今では、ラチス構造は倉庫の建設などの大規模建造物のための簡単な構法のひとつとして残るのみである。製造の手間によるコストが見合わず、積極的には使われていないといえる。

現在、先進国ではデジタルファブリケーションの技術開発が進んでいる。例えばオランダでは、2018 年に MX3D と Autodesk 社によってステンレス 3D プリント製の橋が制作され、挙動実験などが行われており、実用化に近付いている。レーザーカッターの性能の向上や、プレハブのノウハウの蓄積も、重ねて複雑な建設作業をより簡易にしていくと考えられる。

複雑な形態を機械によってより簡単に安価で作れるようになってきた今、材料の削減が再び重要な題となってくる [1]。少ない材料で作ることのできる強固な構造としてラチス構造を再び取り上げることで、そのヒントが得られると考える。本論では、新たに既存のものより複雑なラチス構造を研究対象とする。

図 1.1　MX3D と Autodesk 社によるステンレス 3D プリント製の橋 [5]

１－２　ラチス構造

ラチスとは、柱や梁などの弦材の間を、ジグザグ状に補助材でつなぎ、補強した構造のことを一般に指す。斜めに入っている補助材を斜材、弦材に対し垂直方向のものを束材と呼ぶ。

本論では、図のようにラチス化が入れ子構造で繰り返されるものを想定し、n 回繰り返したものを $Lattice^n$ 形式というように表現する。$Lattice^2$ 形式はエッフェル塔に用いられていることで有名である。本論では、任意の $Lattice^n$ 形式に対応するアルゴリズムを目標として、特にラチス化の操作を 3 回繰り返した $Lattice^3$ 形式のものを取り上げ、研究対象とした。$Lattice^2$ 以上はデジタルファブリケーション技術によって達成される複雑な形状である点で新規性がある。また、ラチス化の繰り返しにより、同程度の概形で強度を保ったまま、材料の減量、および意匠面での透明感の獲得が可能である。さらに、最小の構成材が直線であっても、全体として有機的に見える形状の生成も期待できる。

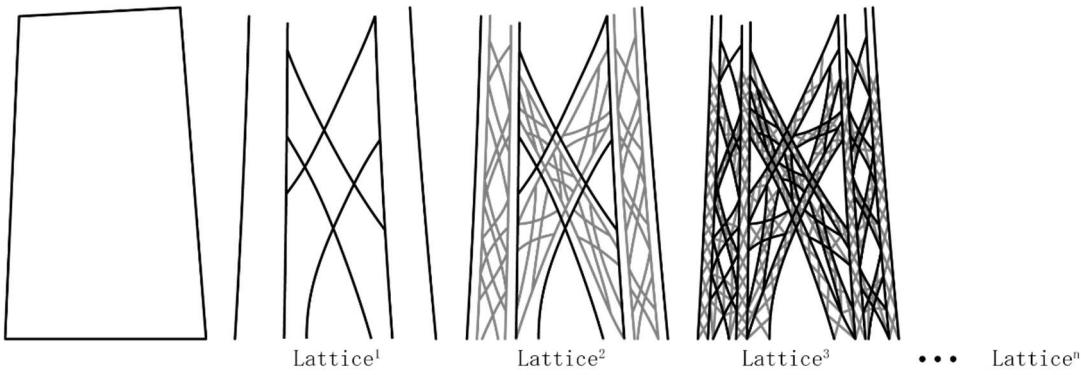


図 1.2　ラチス化の繰り返し

１－３　アルゴリズムの全体構成

本論で提案するアルゴリズムは、

①　Lattice2 モデルを生成

②　Lattice2 モデルを最適化

③　最適化されたモデルをラチス化し、Lattice3 形状を生成

という 3 段階で構成される。

第 2 章では②の最適化について、縮約の妥当性についての検討、本論で用いる最適化法である共役勾配法、開発したアルゴリズムについて順を追って述べる。続いて第 3 章では、モデル生成について述べる。①と③において、生成する形状は異なるが、アルゴリズムの根幹は同じものとみなすことが可能であるため、ここではより複雑な③について詳細に説明する。また、第 4 章ではプロトタイプとしていくつかのモデルに最適化を施し、結果について考察を述べる。

２．構造解析

２－１　共役勾配法

本論では、構造の**最適化**にあたり、**共役勾配法**を用いる。

最適化とは、与えられた制約条件の下で関数の値を最大化あるいは最小化する変数の値を求めることである。**共役勾配法**はこういった最適化法の一つであり、直接関数を解かずとも反復的に極値を導くことができるという特徴がある。
以下、共役勾配法について、金谷健一著『これならわかる最適化数学』を参考として述べる[2]。

２－１－１　一変数の勾配法

はじめに**勾配法**について説明する。

一変数関数$f(x)$の最小値を求めることを考える。
このとき、$f'(x)=0$となる解xを見つけられれば極値として最小値が求まるが、関数が複雑であった場合は解析的に解が求まらないことが多い。このとき、考えている領域で最小の極値があるとわかっている場合に限り、次のように数値的に計算できる。

まず、最小値に近いと考えられる点x_0を初期値として与える。
$f'(x_0)=0$であればそこで関数 f は最小値をとる。$f'(x_0)>0$ならば x 軸上を左に、$f'(x_0)<0$ならば右に進む（図 2.1）。

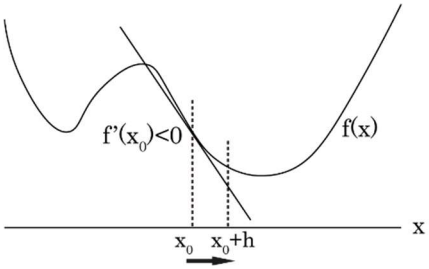


図 2.1　勾配法イメージ

ここで図中のhにあたる値をステップ幅と呼ぶ。
ステップ幅はすなわち進む距離であり、これが大き過ぎると極値を通り越して発散する可能性があり、逆に小さいとなかなか極値に到達せず、最適化に時間がかかる。したがって、ステップ幅は関数値が必ず減少するように、かつなるべく大きな値であることが望ましい。

２－１－２　多変数の勾配法

２変数関数$f(x,y)$の極値を求める勾配法について述べる。
極値に近いと思われる点(x_0,y_0)を初期値として与える。関数値が最も大きく減少する方向は∇fで与えられるため、その方向の直線上で関数が最小になる点まで進む。その点で再び勾配∇fを

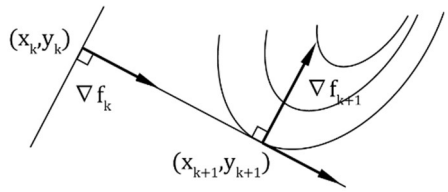


図 2.2　多変数関数の勾配イメージ

計算し、その方向の直線上で関数値が最小となる点まで進む。これを収束するまで繰り返すことで、最終的に極値にたどり着く。

このとき、勾配∇fの方向で関数値が最小となる点を探すことを直線探索と言う。
関数曲面を直線できったときの断面に現れる曲線について、３－１－１で示した勾配法を用いるようなイメージの操作である。

これは n 変数関数のときも同様である。

[多変数の勾配法のアルゴリズム]//

$f(\boldsymbol{x})$について

1. $\boldsymbol{x}$の初期値を与える。
2. 関数$F(t)=f(\boldsymbol{x}+t\nabla f(\boldsymbol{x}))$に対し$t$について直線探索を行う。
3. 返されたtを用いて$\Delta\boldsymbol{x}\leftarrow t\nabla f(\boldsymbol{x}),\quad \boldsymbol{x}\leftarrow\boldsymbol{x}+\Delta\boldsymbol{x}$とする。
4. ステップ２に戻り、これを$\|\Delta\boldsymbol{x}\|<\varepsilon$となるまで繰り返す。
5. $\boldsymbol{x}$を返す。

//

ただし、アルゴリズム中の$\boldsymbol{x}$は$x_1,x_2,x_3,\dots,x_n$を成分とするベクトルを表すとする。また、εは微小な値を表す定数とする。
太字上のアルゴリズム中で、関数$F(t)=f(x+t\nabla f(x))$に対しtについて直線探索を行うには、導関数$F'(t)$が必要である。$\boldsymbol{x}(t)=\boldsymbol{x}_0+t\Delta f_0$とおいて、$F(t)=f(\boldsymbol{x}(t))$を$t$で微分すると次のようになる。

$$\frac{dF}{dt} = \sum_{i=0}^{n} \frac{\partial f}{\partial x_i} \frac{dx_i}{dt} = \sum_{i=0}^{n} \frac{\partial f}{\partial x_i} \frac{\partial f_0}{\partial x_i} = (\nabla f\ , \nabla f_0) \tag{2.1}$$

ただし、$(\nabla f\ , \nabla f_0)$は∇fと∇f_0の内積を表す。ここで、$\partial f_0/\partial x_i$,$\nabla f_0$はそれぞれ$\partial f/\partial x_i$, ∇fの初期値x_0での値である。このことから、Fが極値をとれば$(\nabla f\ , \nabla f_0) = 0$、すなわち初期値$x_0$での直線探索方向$\nabla f_0$と極値をとる点$x$での勾配$\nabla f$とが直交することがわかる。
これを幾何学的に解釈すると、２変数の場合は図のように、直線探索で定まる点ではその点を通る等高線が探索直線に接しているということである。したがって、直線探索の方向は直前の探索方向と直行する。

勾配法のなかでも、このように一回微分のみから勾配を見て極値を探索するものを最急降下法という。この方法は、一階微分しか見ないために計算が簡便であるというメリットがあるが、主に３つのデメリットがある。

① 勾配∇fを式として与える必要があるため、不連続であったり、とがっていたりといった微分のできない関数では不都合が生じる。また、理論的には微分可能であっても複雑すぎる、あるいは変数の数が多すぎる場合計算が煩雑になる。

② 求まる解が局所的な極値（局所解）にとどまり、全体で見た時の最大値、最小値が求まりにくいため、それを求めるには多数の初期値を与える、あるいは求める極値に近い初期値を推定する必要がある。

③ 極値付近の関数曲面の形状によっては収束にかなり時間がかかることがある。そもそも、最急降下法は収束時間の短い方法とは言えない。

ラチス形状に対し、例えば安全率が最小になるよう最適化を行うにあたり、安全率の値を表す関数を微分することは実際には難しく、①のデメリットにあたる。したがって、本論では勾配を求めるにあたって微分は行わず、Δf / h を勾配として用いる。

③のデメリットである収束時間の長さを改善するために、次の共役勾配法が有効である。

２－１－３　逐次最小化法

一階導関数だけでなく二階導関数を用いたニュートン法では、ある点においての関数を放物線で近似（２次近似）し、その放物線の極値を与える解に向かって進むことを反復して行う。いま、２変数関数fについて現在の近似解が$(x^{(K)}, y^{(K)})$であるとき、その２次近似は次のように書ける。

$$f_{II}(x) = f^{(K)} + \left(\nabla f^{(K)}, \boldsymbol{x} - \boldsymbol{x}^{(K)}, \boldsymbol{H}^{(K)}\left(\boldsymbol{x} - \boldsymbol{x}^{(K)}\right)\right) \tag{2.2}$$

ただし$\boldsymbol{x} = \begin{pmatrix} x \\ y \end{pmatrix}$とおき、勾配$\nabla f$とヘッセ行列$\boldsymbol{H}$を次のようにおいた。

$$\nabla f = \begin{pmatrix} \partial f/\partial x \\ \partial f/\partial y \end{pmatrix}, \quad \boldsymbol{H} = \begin{pmatrix} \partial^2 f/\partial x^2 & \partial^2 f/\partial x \partial y \\ \partial^2 f/\partial y \partial x & \partial^2 f/\partial y^2 \end{pmatrix} \tag{2.3}$$

$\boldsymbol{x}, \nabla f, \boldsymbol{H}$の添え字$(K)$は$(x, y)$に$(x^{(K)}, y^{(K)})$を代入することを意味する。
式（3.2）の極値は次式から求まる。

$$\nabla f_{II} = \nabla f^{(K)} + \boldsymbol{H}^{(K)}\left(\boldsymbol{x} - \boldsymbol{x}^{(K)}\right) = 0 \tag{2.4}$$

現在の位置$\boldsymbol{x}^{(K)}$から解$\boldsymbol{x}$に進むには、$\boldsymbol{m}^{(K)} \propto \boldsymbol{x} - \boldsymbol{x}^{(K)}$であるようなベクトル$\boldsymbol{m}^{(K)}$の方向に移動すればよい。式（3.4）より、そのような$\boldsymbol{m}^{(K)}$に関して次の式が成り立つ。

$$\boldsymbol{H}^{(K)}\boldsymbol{m}^{(K)} \propto \nabla f^{(K)} \tag{2.5}$$

このような方向$\boldsymbol{m}^{(K)}$を**共役勾配方向**と呼ぶ。

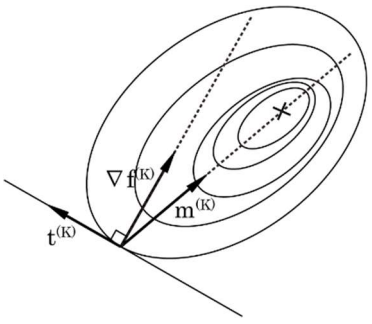


図 2.3　共役勾配方向イメージ

図のように、等高線が正円でない場合、共役勾配方向に進むことでより早く極値解に近付くことができる。

点$\boldsymbol{x}^{(K)}$での等高線の接線ベクトルを$\boldsymbol{t}^{(K)}$とすると、勾配$\nabla f^{(K)}$は接線と直交する。したがって、式（3.5）の両辺と$\boldsymbol{t}^{(K)}$との内積を取ると次式を得る。

$$\left(\boldsymbol{t}^{(K)}, \boldsymbol{H}^{(K)}\boldsymbol{m}^{(K)}\right) = 0 \tag{2.6}$$

共役勾配$\boldsymbol{m}^{(K)}$は勾配$\nabla f^{(K)}$からある程度接線方向$\boldsymbol{t}^{(K)}$にずれているため、ある定数$\alpha^{(K)}$を置いて

$$\boldsymbol{m}^{(K)} = \nabla f^{(K)} + \alpha^{(K)}\boldsymbol{t}^{(K)} \tag{2.7}$$

と書ける。（等高線が正円の場合は$\alpha^{(K)} = 0$となる。）これを式（3.6）に代入すると、$\left(\boldsymbol{t}^{(K)}, \boldsymbol{H}^{(K)}\nabla f^{(K)}\right) + \alpha^{(K)}\left(\boldsymbol{t}^{(K)}, \boldsymbol{H}^{(K)}\boldsymbol{t}^{(K)}\right) = 0$となり、$\alpha^{(K)}$は

$$\alpha^{(K)} = -\frac{\left(\boldsymbol{t}^{(K)}, \boldsymbol{H}^{(K)}\nabla f^{(K)}\right)}{\left(\boldsymbol{t}^{(K)}, \boldsymbol{H}^{(K)}\boldsymbol{t}^{(K)}\right)} \tag{2.8}$$

となる。この$\alpha^{(K)}$を用いて、式（3.7）から定まる共役勾配の方向に直線探索をおこなえば、ヘッセ行列$\boldsymbol{H}^{(K)}$の逆行列を計算することなく、直線探索によってf_{II}の極値を求めることができる。これが共役勾配法の原理である。

ここで、二次近似f_{II}ではなく、もとの関数fについて直線探索を行うことを考える。これには３－１－２で示した勾配法を用いればよい。

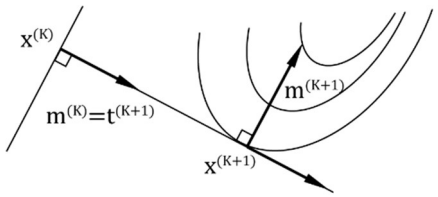


図 2.4　勾配法の反復的用法

このとき、図のように、ある点$\boldsymbol{x}^{(K)}$から伸びた探索直線は次の点$\boldsymbol{x}^{(K+1)}$において等高線に接している。いま各点において探索直線は共役勾配方向に進んでいるため、$\boldsymbol{m}^{(K)} = \boldsymbol{t}^{(K+1)}$が成り立つ。各ステップで同じことが繰り返されるため、ここから次の反復公式を得る。

$$\boldsymbol{m}^{(K)} = \nabla f^{(K)} + \alpha^{(K)}\boldsymbol{m}^{(K-1)} \tag{2.9}$$

$$\alpha^{(K)} = -\frac{\left(\boldsymbol{m}^{(K-1)}, \boldsymbol{H}^{(K)}\nabla f^{(K)}\right)}{\left(\boldsymbol{m}^{(K-1)}, \boldsymbol{H}^{(K)}\boldsymbol{m}^{(K-1)}\right)} \tag{2.10}$$

$$\boldsymbol{x}^{(K+1)} = \boldsymbol{x}^{(K)} + t^{(K)}\boldsymbol{m}^{(K)} \tag{2.11}$$

初期値$\boldsymbol{x}^{(0)}$では$\alpha^{(0)} = 0$として勾配方向$\nabla f^{(0)}$に進み、その後は上記の式に従って進めばよい。ここまでは二変数関数について述べたが、これは多変数関数についても成り立つ。

より複雑な関数について共役勾配法を用いる場合、二階偏微分の計算が煩雑となり、式（3.10）に必要なヘッセ行列を求めることが困難となる。ヘッセ行列を含まない式で代用する方法として、以下のような式が考えられている。

$$\alpha^{(K)} = -\frac{\left(\nabla f^{(K)}, \nabla f^{(K)} - \nabla f^{(K-1)}\right)}{\left(\boldsymbol{m}^{(K-1)}, \nabla f^{(K)} - \nabla f^{(K-1)}\right)} \quad \text{ビール・ソレンソンの式} \tag{2.11}$$

$$\alpha^{(K)} = -\frac{\left(\nabla f^{(K)}, \nabla f^{(K)} - \nabla f^{(K-1)}\right)}{\left|\left|\nabla f^{(K-1)}\right|\right|^2} \quad \text{ポラック・リビエの式} \tag{2.12}$$

$$\alpha^{(K)} = -\frac{\left|\left|\nabla f^{(K)}\right|\right|^2}{\left|\left|\nabla f^{(K-1)}\right|\right|^2} \quad \text{フレッチャー・リーブスの式} \tag{2.13}$$

２－２　実装

２－２－１　目的関数と変数

骨組に対し形態解析を行い、各部材の安全率を短期、長期ともに算出する。目的関数をこれらの安全率の最大値、変数を各節点の xyz 座標とし、安全率の最大値を小さくするアルゴリズムを構築する。

本論の最適化アルゴリズムは、節点の動かし方により、大きく３種類に分けられる。
３－２－２では、各点をそれぞればらばらに動かす Conjugate Gradient、
３－２－３では、端点の移動に対して周囲の点が引っ張られるように追従する CG Pinch、
３－２－４では、部材を種類別に分け、弦材をしなるように動かす CG Curve
について述べる。

なお、プログラム全文は付録に記す。

２－２－２　Conjugate Gradient

全体の流れを図のフローチャートに示す。

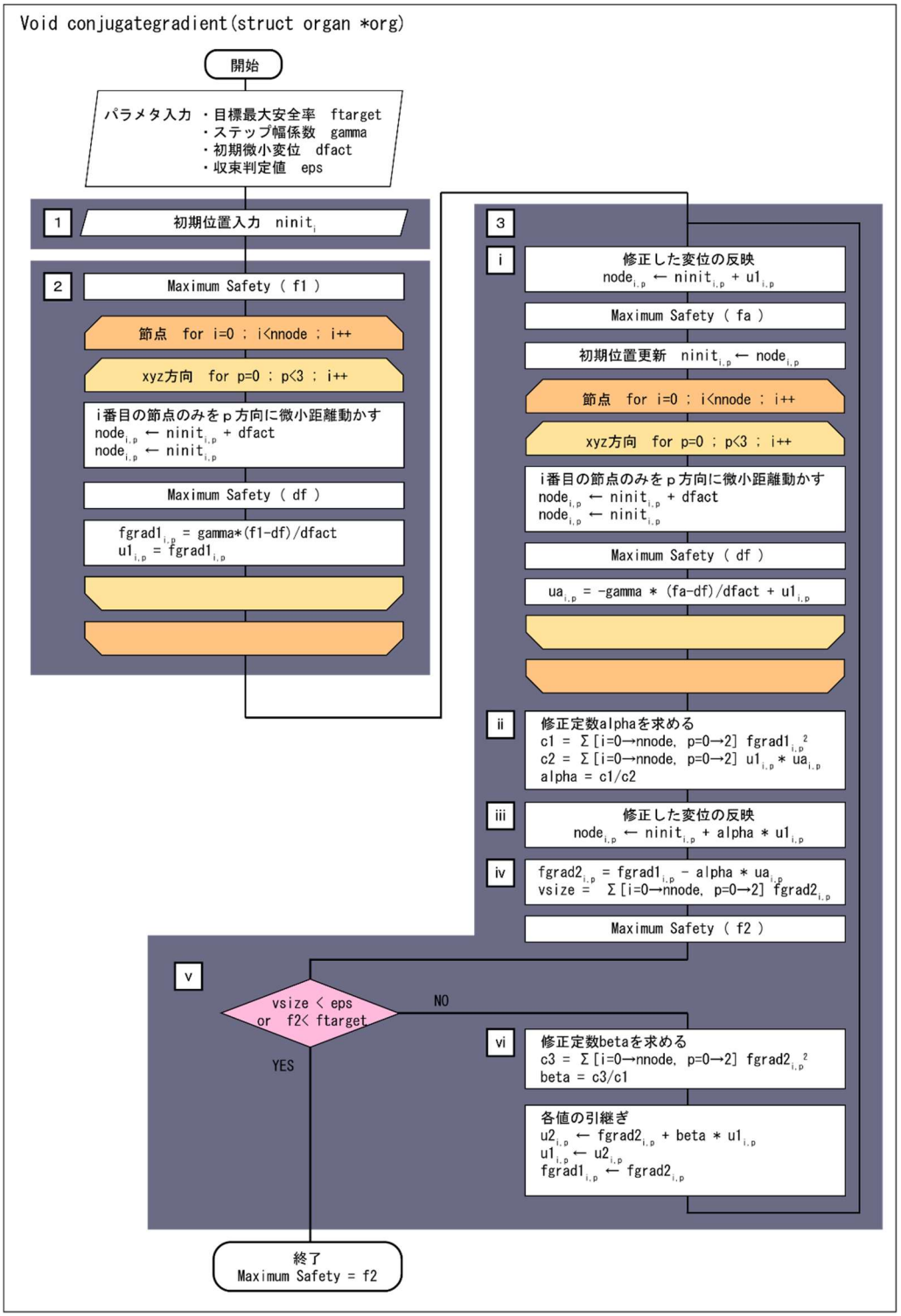


図 2.5　Conjugate Gradient のフローチャート

形態解析アルゴリズムの基本のフロー

アルゴリズムに必要な 4 つのパラメタを表 2.1 のように定める。

表 2.1　アルゴリズムのパラメタ

名称	役割
ftarget	目標とする安全率の値
gamma	ステップ幅係数
dfact	初段階で与える微小変位
eps	収束判定値

本アルゴリズムにおいて、形態解析により求まる目的関数は、その複雑さから微分が困難であるため、変数に微小変位を加えたときの関数の値から直接傾きを求める手法をとる。

共役勾配法を用い、次のようにアルゴリズムを定める [3]。

1　入力

ninit に初期座標、すなわち目的関数に対する初期解を与える。

2　初期解における関数の勾配を求める。

初期座標での安全率の最大値（目的関数の値）を求め、これを fl とする。

次に、ある節点 i の xyz いずれか p の座標について（ひとつの変数について）、

微小変位 dfact を与える。

$$node_{i,p} = ninit_{i,p} + dfact \tag{2.14}$$

安全率の最大値（目的関数の値）を求め、これを df とする。

これに対し、以下のように勾配 fgrad1 とステップ値 u1 を求める。

$$fgrad_{i,p} = gamma * \frac{f1 - df}{dfact} \tag{2.15}$$

$$u1_{i,p} = fgrad1_{i,p} \tag{2.16}$$

この操作をすべての変数に対し行い、関数の勾配を調べる。

3　共役勾配法の反復的操作を行う。

i　ステップ値を変数に加え、勾配を調べる。

ステップ幅の反映。

$$node_{i,p} = ninit_{i,p} + u1_{i,p} \tag{2.17}$$

目的関数の値を求め、これを fa とする。

初期位置を更新する。

$$ninit_{i,p} = node_{i,p} \tag{2.18}$$

2 と同様、ある節点 i の p 座標について微小変位 dfact を与える。

$$node_{i,p} = ninit_{i,p} + dfact \tag{2.19}$$

目的関数の値を求め、df とする。

$$ua_{i,p} = -gamma * \frac{fa - df}{dfact} + u1_{i,p} \tag{2.20}$$

これを各節点に対し行う。

ii 修正定数 α を次式により定める。

$$c1 = \sum fgrad1^2 \tag{2.21}$$

$$c2 = \sum (u1 * ua) \# \tag{2.22}$$

$$\alpha = \frac{c1}{c2} \tag{2.23}$$

iii 修正したステップ幅を変数に加える。

$$node_{i,p} = ninit_{i,p} + \alpha * u1_{i,p} \tag{2.24}$$

iv 次の 3 のループのための値の準備を行う。

修正した勾配方向を次式により定める。

$$fgrad2_{i,p} = fgrad1_{i,p} - \alpha * ua_{i,p} \tag{2.25}$$

また、収束判定のため、vsize を次式により定める。

$$vsize = \sum fgrad2^2 \tag{2.26}$$

目的関数の値をもとめ、f2 とする。

v 収束判定

勾配がゼロ、あるいは目的関数がターゲット値より小さくなった場合、

収束とみなす。すなわち、

$vsize < eps$　または　$f2 < ftarget$

であった場合、収束したとみなし、構造解析を終了する。

vi v で収束しなかった場合、次の修正定数をもとめ、各値を引き継ぐ。

修正定数 β を次式で定める。

$$c3 = \sum fgrad2^2 \tag{2.27}$$

$$\beta = \frac{c3}{c1} \tag{2.28}$$

各値の引継ぎ

$$u2_{i,p} = fgrad2_{i,p} + \beta * u1_{i,p} \tag{2.29}$$

$$u1_{i,p} = u2_{i,p} \tag{2.30}$$

$$fgrad1_{i,p} = fgrad2_{i,p} \tag{2.31}$$

3 の初めに戻り、繰り返す。

v において収束した場合、iv で求めた f2 を構造解析の結果得られる安全率の最大値の改良値として得る。

２－２－３　Conjugate Gradient Pinch

全体の流れを次のフローチャートに示す。

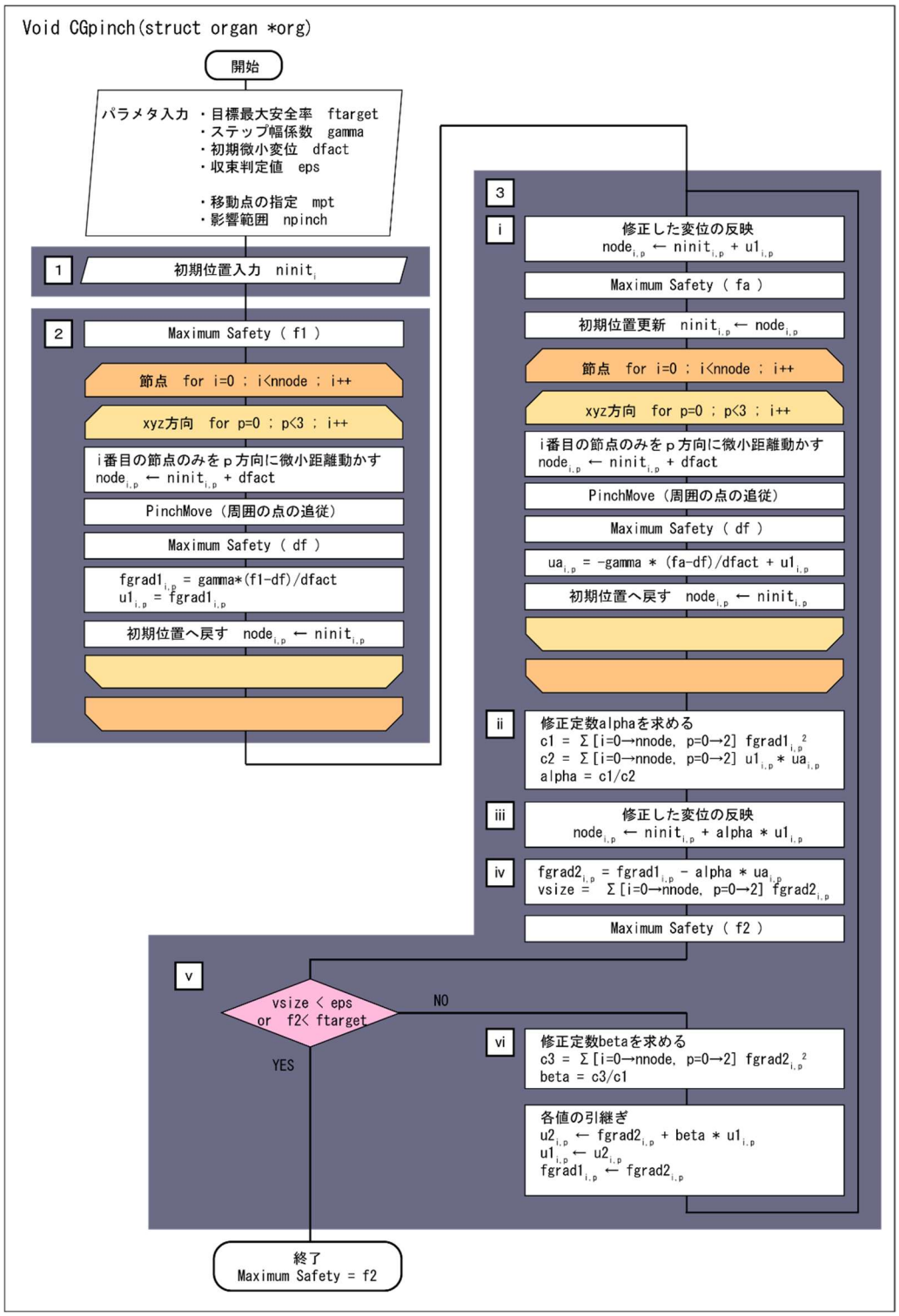


図 2.6　CGpinch のフローチャート

CGpinch では、ConjugateGradient を基本とし、節点の移動に制約条件を加えた。
移動させる節点を指定して mpt とし、その節点が動いたとき、周囲の節点が引っ張られるように追従するアルゴリズムとし、つまむような追従になぞらえ CGpinch と名付けた。

次に示す PinchMove は、周囲の節点の追従を定めるアルゴリズムである。

PinchMove　周囲の節点の追従

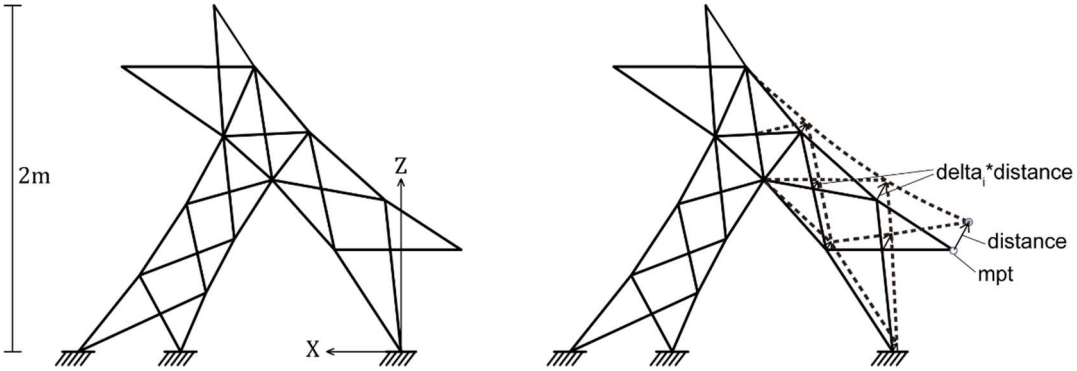


図 2.7　PinchMove のイメージ

図 2.7 にイメージを示す。

ある移動させる節点 i = mpt が長さ distance だけ動いたとき、周囲の点の追従の仕方を次のように定める。

1. 図のように、各節点が i から何番目に近い点かを調べる。
   いくつまで調べるかはパラメタ npinch によって定め、この近さを表す番号を nflag とする。npinch の範囲外の nflag = 0 とする。
   例えば、ある節点 j が i から 2 番目に近い節点だった場合、$nflag_j = 2$ と表す。

2. 周囲のある節点 j に関して、係数δを

$$\delta = \frac{1}{nflag} * d \quad (0 < d < 0.5) \tag{2.32}$$

   とする。ただし d はパラメタとして決まる定数である。

3. 各節点を$\delta * distance$だけ動かす。

これによって、節点 i に近い節点ほど強く追従する pinch（＝つまむ）の操作を行う。

２－２－４　Conjugate Gradient Curve

全体の流れを次のフローチャートに示す。

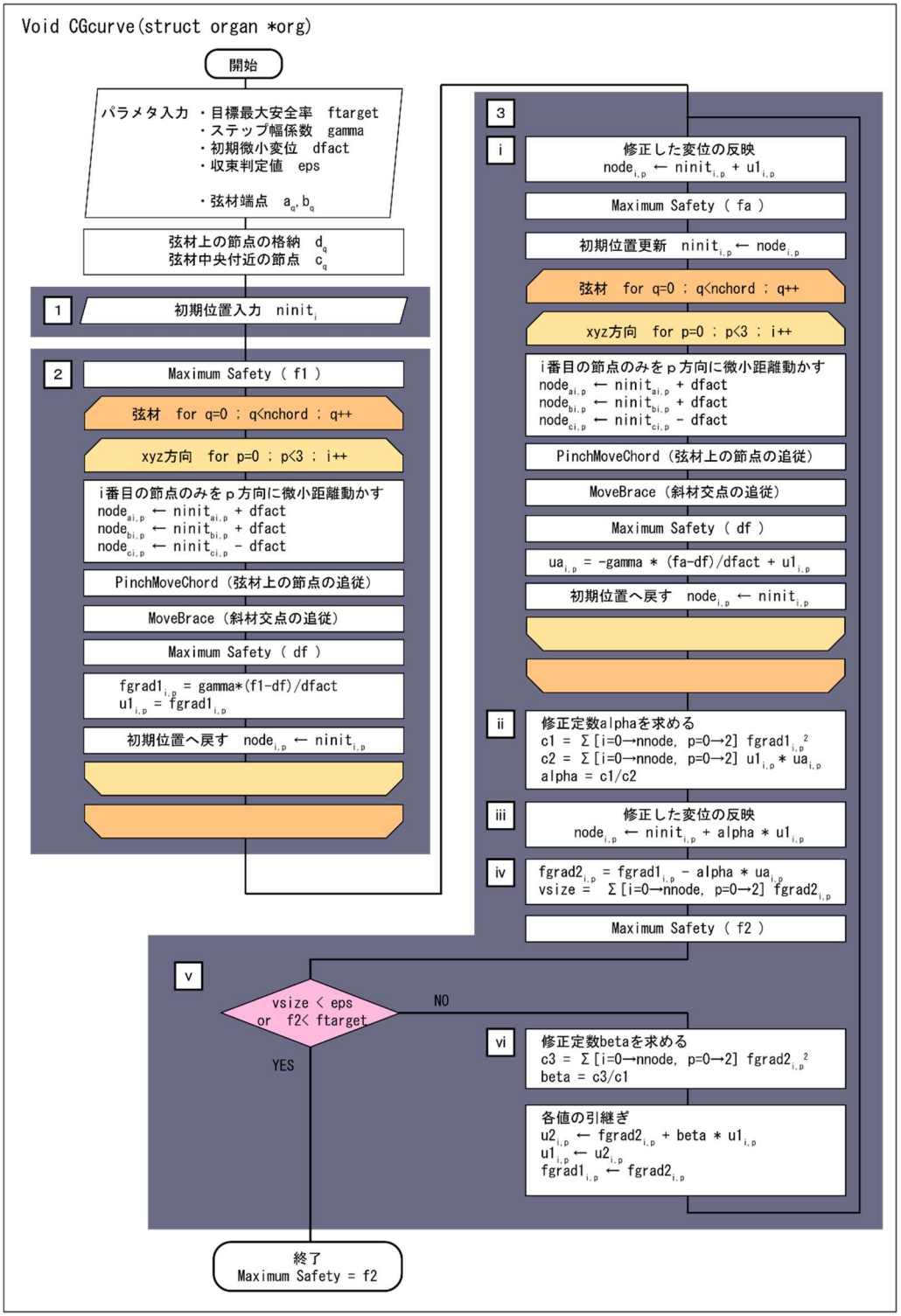


図 2.8　CGcurve のフローチャート

CGcurve もまた、Conjugate Gradient を基本として節点の移動に制約条件を設けたものである。弦材がひとつながりであることをより強く反映し、弦材がしなるように節点を動かすアルゴリズムとした。

入力および節点の移動について、それぞれ詳しく述べる。

弦材の入力

CGcurve では初期入力時点でモデル中の弦材が直線である必要がある。以下の手順で弦材の入力を行う。

1. q 番目の弦材について、端点を点 A,B とし、その節点番号 i をそれぞれ a_q,b_q として格納する。

2. 端点を除く弦上の点を点 D とし、その節点番号 i を $d_{q,r}$ に格納する。
   この際 d の個数をカウントし、ある弦材 q 上の端点を除いた節点の数を $rmax_q$ とする。

3. 中央付近の節点を探し、点 C とする。これを以後弦材の中央近点と表現する。
   $rmax_q/2$ の商が r であるとき（余りについては無視する）、r 番目の節点 D を c_q に変更し、格納する。

PinchMoveChord　弦材上の節点の追従

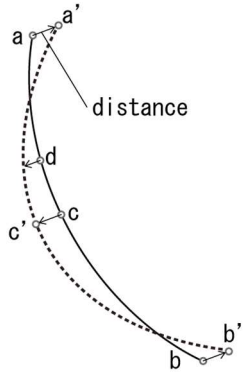


図 2.9　弦材上の節点移動のイメージ

図のように、弦材上の点の節点の追従を考える。このとき、各点の追従移動距離が各点と端点、中央近点との距離に比例するだけでは弦材全体がくの字に曲がっていってしまう。弦材がなめらかな曲線を描くよう節点を追従させるために、以下の条件を満たす係数δを導入する。

① 中央近点 C から近いほど強く追従する
② 端点 A,B に近ければ正、中央近点 C に近ければ負の値をとる

以下の手順によって係数δを求め、節点の追従を行う。

1. 弦材上のある節点 D が端点 A,B のどちらに近いかを判定する。

(ア) AD≦BD のとき

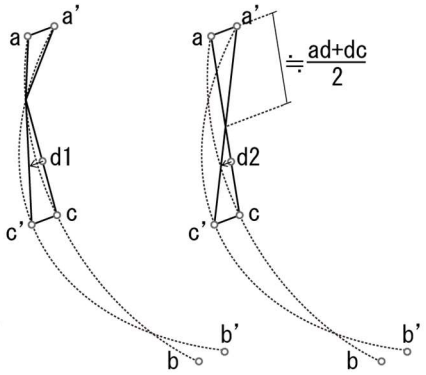


図 2.10　三角形への近似イメージ

図のような三角形に近似することを考え、δを次式によって求める。

$$\delta = \left\{ \frac{\left(\frac{AD + DC}{2} - AD \right)}{\frac{AD + DC}{2}} \right\} * \frac{AD}{AC} \tag{2.33}$$

このとき、先に述べたδの条件について、各因数が以下のように対応する。

① 中央近点 C から近いほど強く追従する
→AD と AC の距離の比 $\frac{AD}{AC}$

② 端点 A,B に近ければ正、中央近点 C に近ければ負の値をとる
→DC＜AD のとき負の値となる値 $\left(\frac{AD+D}{2}-AD\right)$
→割合化するために $\frac{AD+DC}{2}$ で割る

(イ) AD＞BD のとき

同様にして、次式により係数δを求める。

$$\delta=\left\{\frac{\left(\frac{BD+DC}{2}-BD\right)}{\frac{BD+DC}{2}}\right\}*\frac{BD}{BC} \tag{2.34}$$

2. 求めた係数δ を用いて、各点 D を$\delta*distance$分移動する。

MoveBrace　斜材交点の追従

形態解析の都合上、モデル中の Brace は交点で切断したものを入力する。したがって、Brace の直線形状をたもつには、弦材位置の変化に併せて Brace 交点を追従させる必要がある。

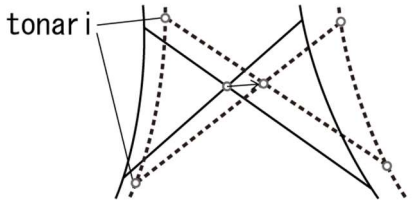


図 2.11　斜材交点の追従

1. 周囲４点を抽出する。プログラム上ではこれらの節点の節点番号を tonari と表記している。このとき、４点は順序関係なくばらばらに抽出される。

2. 斜材端点の組み合わせを割り出す。
   ４点のうち２点の組み合わせ６通りについて、直線距離を求める。
   ６通りのうち、最も長いものが斜材交点を挟んだ点の組み合わせだとわかる。（例えば図において、AB 間の距離が最も長いことから、斜材端部の組み合わせは(A,B),(C,D)とわかる）

3. 各直線の交点を求める。
   これには、計算機科学と地理情報処理で用いられる手法をつかうことで計算量を抑える[2]。
   線分 AB と CD の交点を求める場合、

$$(1-\lambda)\begin{pmatrix}x_a\\y_a\\z_a\end{pmatrix}+\lambda\begin{pmatrix}x_b\\y_b\\z_b\end{pmatrix}=\mu\begin{pmatrix}x_c\\y_c\\z_c\end{pmatrix}+(1-\mu)\begin{pmatrix}x_d\\y_d\\z_d\end{pmatrix} \tag{2.35}$$

$$0<\lambda<1,\ 0<\mu<1$$

   であればよい。それぞれの x,y 座標を用いて

$$\xi=(y_d-y_c)(x_d-x_a)-(x_d-x_c)(y_d-y_a) \tag{2.36}$$

$$\eta=-(y_b-y_a)(x_d-x_a)+(x_b-x_a)(y_d-y_a) \tag{2.37}$$

$$\delta=(y_d-y_c)(x_b-x_a)-(x_d-x_c)(y_b-y_a) \tag{2.38}$$

   とおくと、

$$\lambda=\frac{\xi}{\delta} \tag{2.39}$$

$$\mu=\frac{\eta}{\delta} \tag{2.40}$$

   となる。このとき交点の座標は

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_a \\ y_a \\ z_a \end{pmatrix} + \lambda \begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix} \tag{2.41}$$

で求められる。

この手法では、$\delta = 0$となるとき計算が成り立たなくなってしまうため、その場合はy座標とz座標、あるいはz座標とx座標の組み合わせで再度計算を行う。

３．モデル生成

３－１　既往研究と本生成手法の意義

既往研究では、伊勢坊により Lattice^3 形式の３次元立体骨組モデルを自動生成するアルゴリズムが提案されている。
このアルゴリズムでは、図 3.1 のように、基準線として上部、底部の計８本の直線、および側部のなだらかな４本の曲線を入力し、それらを外形とした Lattice^3 形式の 3D モデルが生成される。この手法は最初に与える曲線の形状に加えて、各部の幅寸法、斜材の密度およびテーパー形状の度合いの３種に対し９つのパラメタを持つ。

既往のアルゴリズムでは、ラチス生成の初段階として、ラチスの層の数に対して各部材の位置関係を見据え、都度描画法を変えていく必要がある。本論では、任意の Lattice^n 形状を元として Lattice$^{(n+1)}$を生成するために、初期形状として概形ではなく中心線を与え、より少ないパラメタでラチス形状を一層増やすアルゴリズムを提案する。これにより、最適化アルゴリズムを用いて求められた形状をモデルに反映することがより簡易になるというメリットも得られる。

また、この研究ではテーパー形状の有用性についても言及されており、本論でもテーパー形状を踏襲する。既往研究で課題として挙げられていた斜材同士の端部の交点については、一部改善を施した。

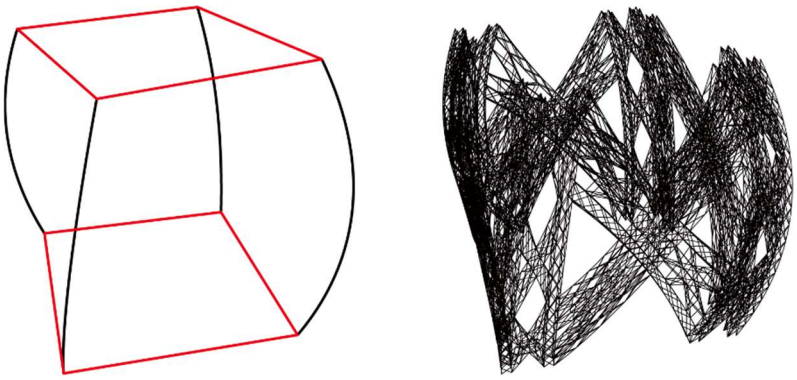


図 3.1　既往研究

３－２　開発環境

モデリングアルゴリズムの開発は 3D CAD ソフト「Rhinoceros」（Robert McNeel & Associates, Version 6 SR21（6.21.19351.9141, 2019/12/17）教育ラボラトリ版）、およびそのプラグイン「Grasshopper」（同社, Version Tuesday, 17 December 2019 09.14　Build 1.0.0007）を使用した。

３－３　２Ｄモデル生成アルゴリズム

図 3.2 にモデル生成のフローチャートを示す。

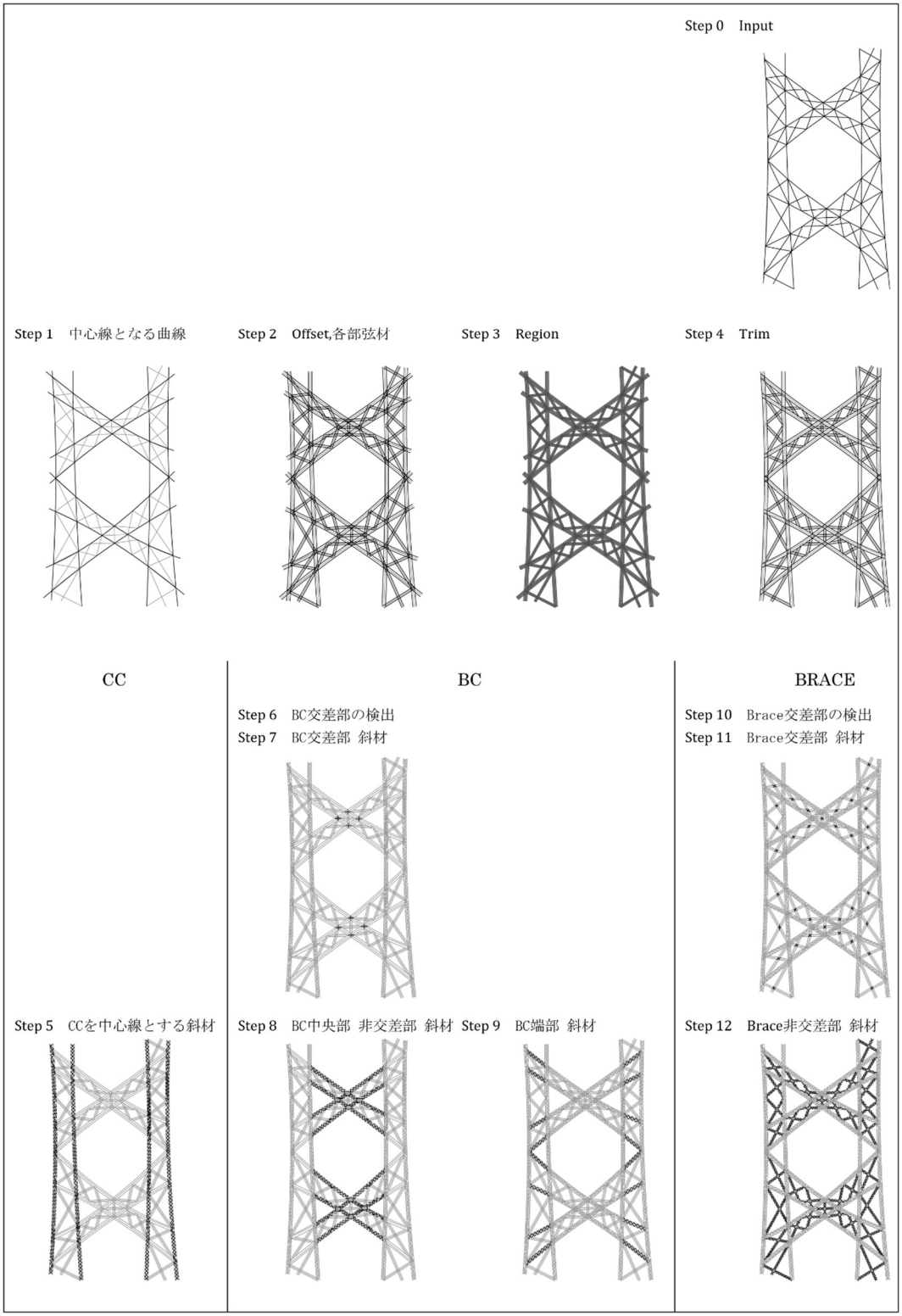


図 3.2　モデル生成フローチャート

以下、アルゴリズムの説明にあたり、Grasshopper で用いられるコンポーネントの名前は囲み線を用いて Component のように示す。

Grasshopper を用いてパラメトリックモデリングを扱う際、データは list に格納される。list は操作を加えることで枝分かれ上に分岐し、データツリーと呼ばれる。Grasshopper ではその枝を切り落とす、ねじり合わせるといったデータ操作が可能である。データ分岐は中括弧を用いて{0;1;2;3}というように左から根本に近い順に示される。図に示す Trim Tree は枝を階層の深さを指定して着ることのできるコンポーネントであり、図形を切り取る Trim とは別のものである（図 3.3）。

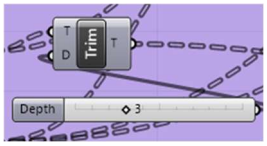


図 3.3　Trim Tree

図 3.4 はモデル生成アルゴリズムの Grasshopper のキャンバスの全体像である。
各ステップを詳細にみていく。

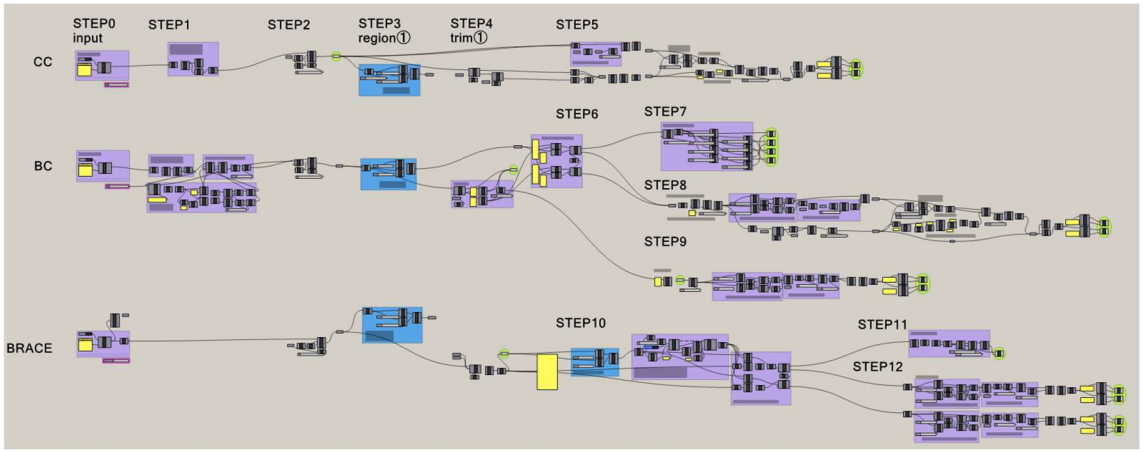


図 3.4 モデル生成アルゴリズムの全体図

Step0 Input

Step0 において各部材は Rhinoceros 上の Layer をもとに、表 3.1 のように 3 種類に分けて入力する。図 3.5 に一例を示す。

表 3.1 部材名称と位置

文中での名称	部材位置
BC	Lattice[1] において斜材、Lattice[2] において弦材
CC	Lattice[1] において弦材、Lattice[2] において弦材
Brace	Lattice[2] 斜材

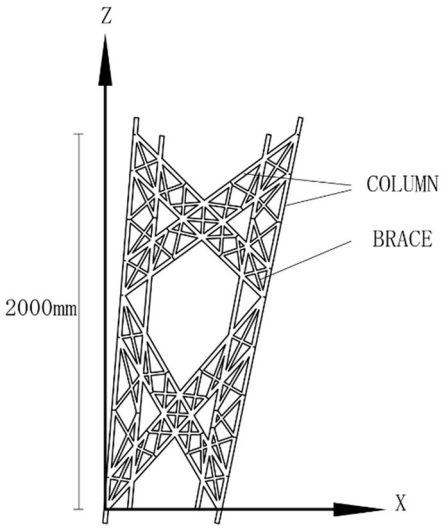


図 3.5 部材配置の一例

Step1　中心線となる曲線の生成

Brace は入力された各線分を中心線として扱う。

Step1-1　CC の中心線

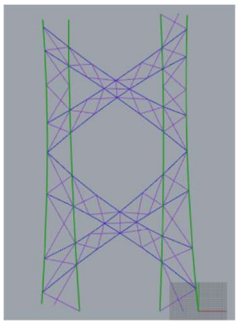


図 3.6　Step1-1

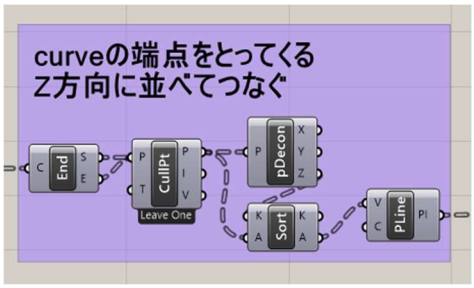


図 3.7　Step1-1　GH

各部材の端点を抽出し、CullPt で重複した点を取り除く。Z 座標に対して昇順に並べ、各点をつなぐ 1 本の曲線を生成。これが CC の中心線となる。ここでは Z 座標をもとにしているが、形状によっては順序を決定する基準を変更する必要がある。

Step1-2　BC の中心線

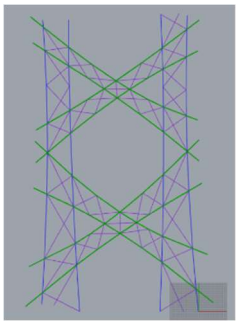


図 3.8　Step1-2

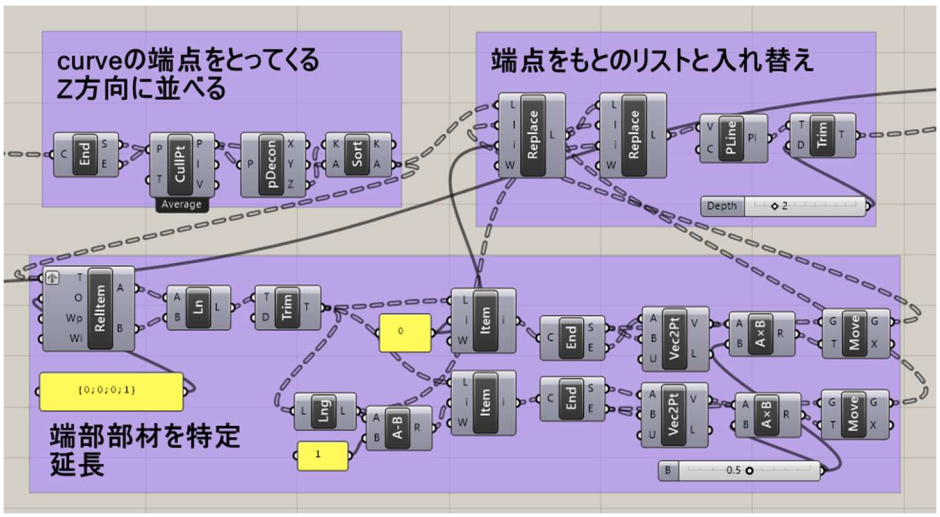


図 3.9　Step1-2 GH

CC と同じく各部材の端点を抽出し Z 座標をもとに並べる（図左上）。後の工程で BC と CC、Brace の交差部を求めるため、BC については入力した中心線の端部を延長する。図の下部では RelItem (Relative Item)により隣り合う点を選択、２点ずつを線分で結び、List Item により線分が格納された list から端部にあたるものだけを選択する。Vec2Pt により線分方向を求め、端点を移動させることで線分を延長する。延長したものを含む新たな list を作成、各点を結んで BC の中心線となる曲線を得る。

Step2　Offset,各部の弦材

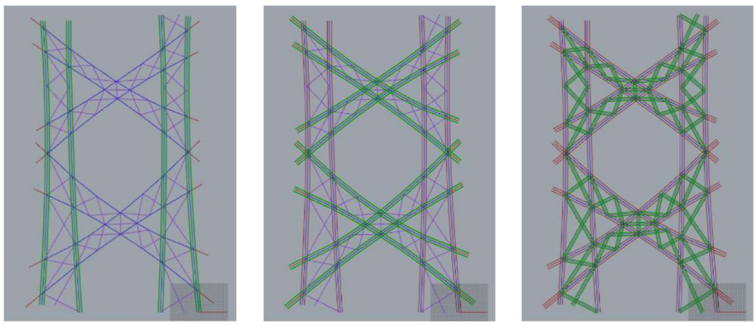


図 3.10　Step2

Step2-1　CC の offset

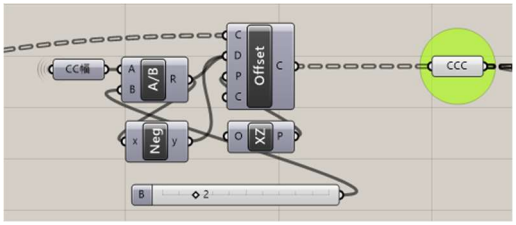


図 3.11　Step2-1 GH

Step2-2　BC の offset

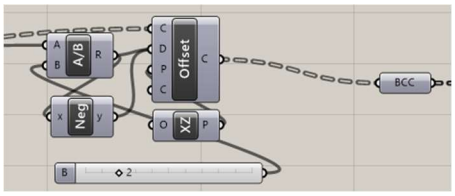


図 3.12　Step2-2 GH

Step2-3　Brace の offset

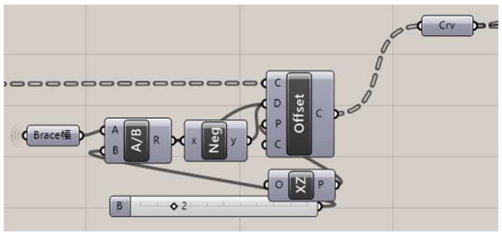


図 3.13　Step2-3　GH

Step1 で求めた曲線を一定値（図中 CC 幅、BC 幅、Brace 幅）分 offset する。CC についてはこの曲線が Lattice3 の弦材となる。他の部分についてはこの曲線をベースとして、交差部の考慮が必要である。

Step3　Region

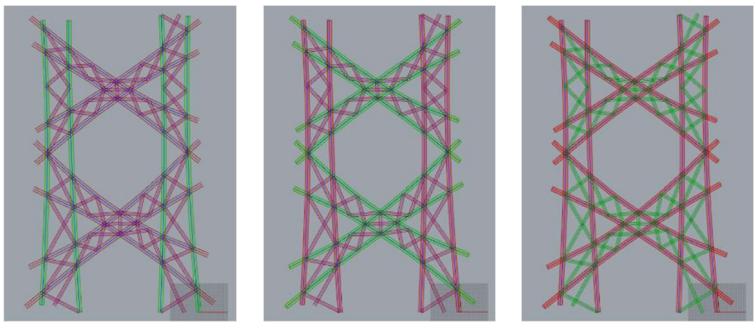


図 3.14　Step3

Step3-1　CC の Region

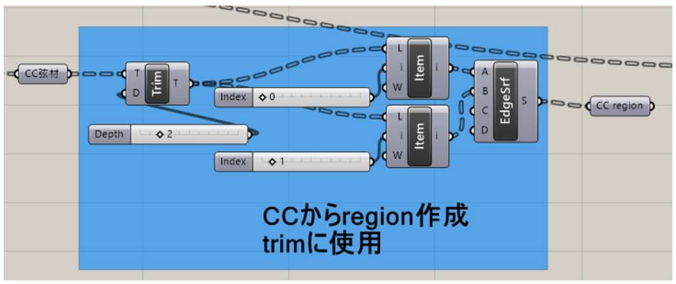


図 3.15　Step3-1　GH

Step3-2　BC の Region

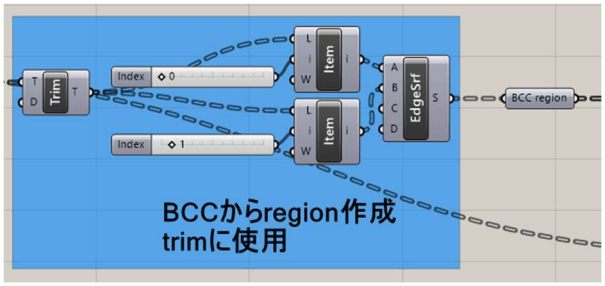


図 3.16　Step3-2　GH

Step3-3　Brace の Region

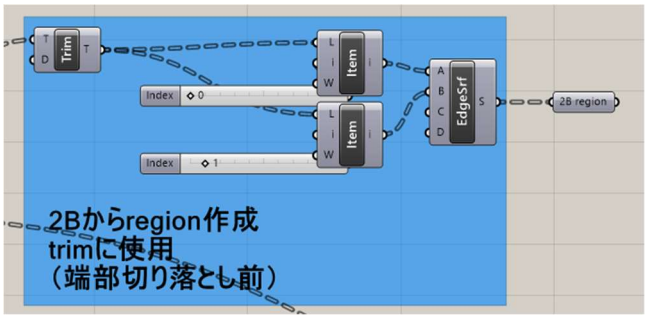


図 3.16　Step3-3　GH

Step2 で offset した曲線から List Item によりついになる 2 本の曲線を選択。EdgeSuf により、閉じた辺を持つ 2 次元の領域（Region）を生成する。Step4 の Trim に用いるための準備段階である。

Step4　Trim

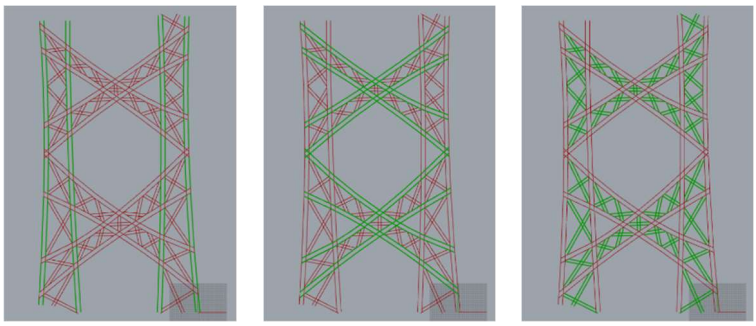


図 3.17　Step4

Step4-1　CC の Trim

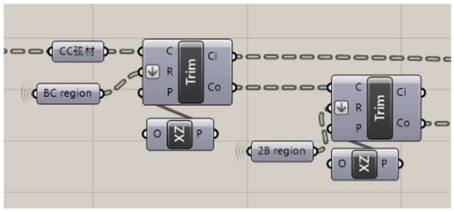


図 3.18　Step4-1　GH

BC、Brace の弦材と CC の弦材との交点を求めるため、ここでは Trim により Step3-2、Step3-3 で生成された BC、Brace の Region によって切り取られる線分を求める。

Step4-2　BC の Trim

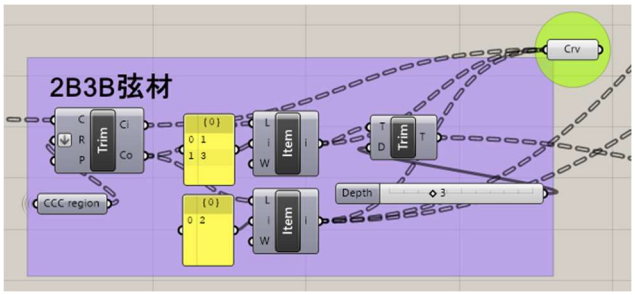


図 3.19　Step4-2　GH

BC は、CC の Region によって切り取られ、残った部分が Lattice3 の弦材となる。

Step4-3　Brace の Trim

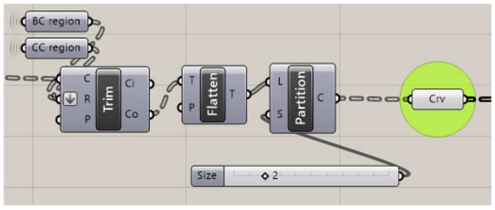


図 3.20　Step4-3　GH

CC および BC の Region により Trim し、Lattice3 の弦材を得る。Flatten および Partition はいずれもデータツリーを扱うコンポーネントであり、生成した弦材を対になるようツリーを整えている。

Step5　CC を中心線とする斜材

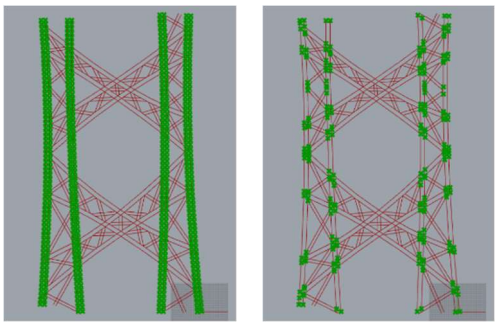


図 3.21　Step5-1

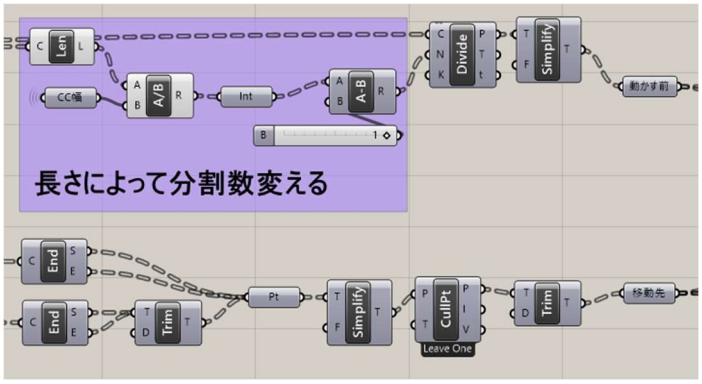


図 3.22　Step5-1　GH

CC 部の Lattice3 の斜材を生成するため、図ではまず CC 部の弦材を分割する点を求める。図上部では単純に弦材を一定数に分割した点を求める。Length により弦材の長さを求め、CC 幅で割った分割数で改めて現在を分割する。これらを「動かす前」の点とする。

下部は BC、Brace 部分と CC 部分との交点を、Step4 で Trim により求めた線分の端点として求める。CullPt によりこれらの点から重複した点を取り除いたものを「移動先」の点とする。

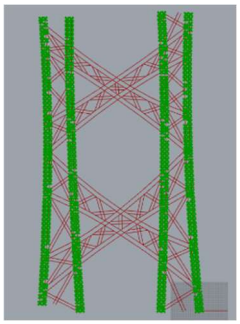


図 3.23　Step5-2

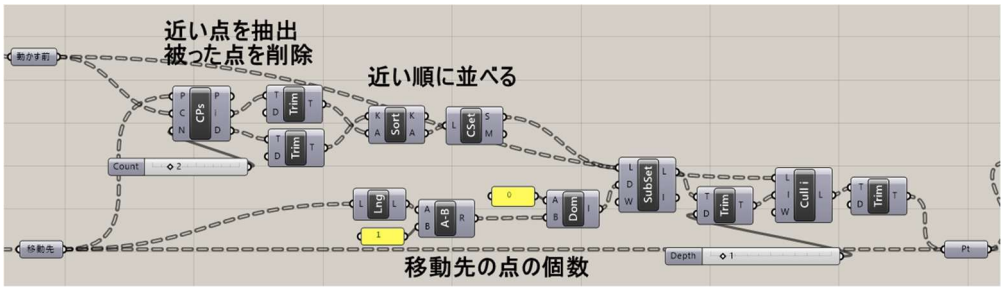


図 3.24　Step5-2　GH

「動かす前」の点の数を保ちながら、「移動先」の点を Lattice3 の斜材の端点として加えることを考える。そのために、「移動先」の点を、それぞれの近くにある「動かす前」の点と入れ替える。

まず CPs (Closest Points)により動かす前の点のなかで移動先の点と距離の近いものを抽出し、その距離の近さの順にこれらを並べる。SubSet により、この list の先頭から、移動先の点の個数分の list を抜き出し、Cull i により「動かす前」の点からこれらを削除する。最後にこの「動かす前」の点と「移動先」の点を合わせ、Lattice3 の斜材の端点とする。

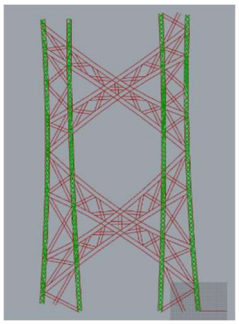


図 3.25　Step5-3

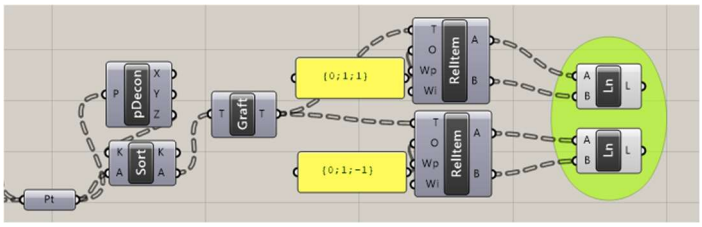


図 3.26　Step5-3　GH

改めて Z 座標を基準として並べなおし、RellItem により隣り合う点同士を線分でつなぐ。これが Lattice3 の斜材となる。

Step6　BC 同士の交差部の検出

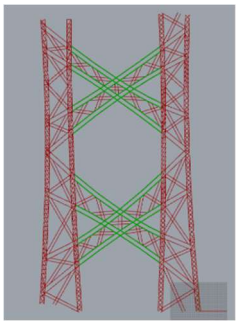


図 3.26　Step6

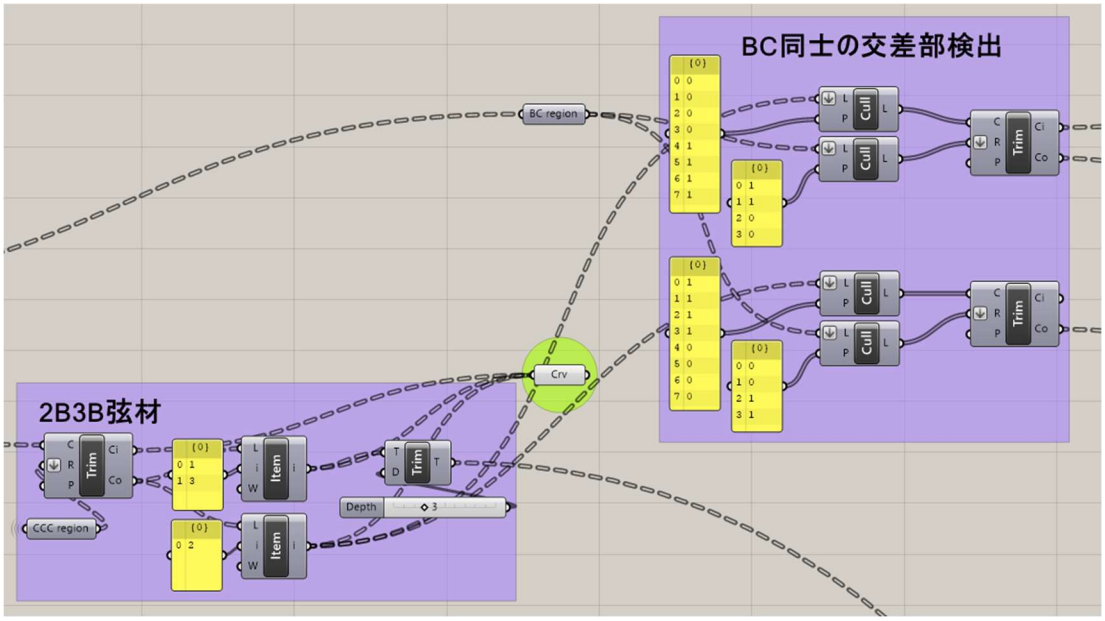


図 3.27　Step6　GH

図の右側が Step6 である。Step4 の Trim により求めた BC 中央部を Cull Pattern を用いて 2 つの組に分ける。片方の組を Step3 の Region を用いて Trim し、BC 同士の交差部の辺にあたる線分を求める。

Step7　BC 同士の交差部　斜材

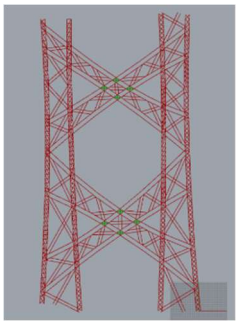


図 3.28　Step7

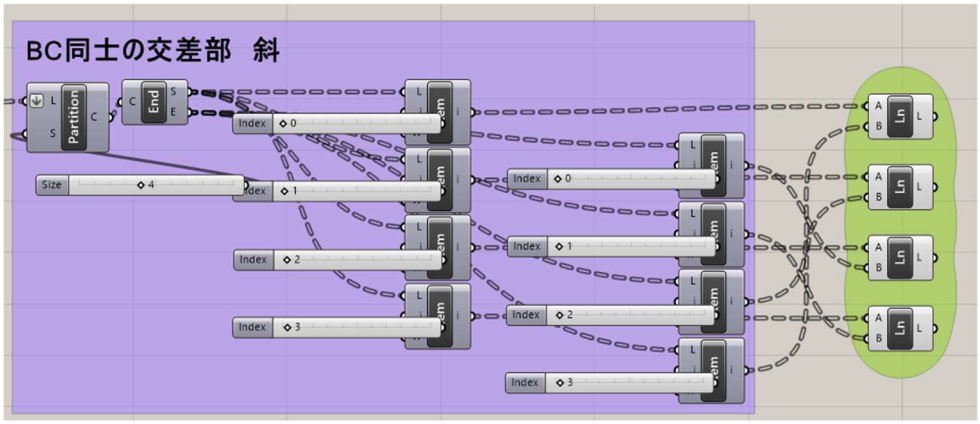


図 3.29　Step7　GH

Step6 で求めた交差部の辺にの端点を交互に結び、交差部にクロスに斜材を入れる。

Step8　BC 中央部　非交差部　斜材

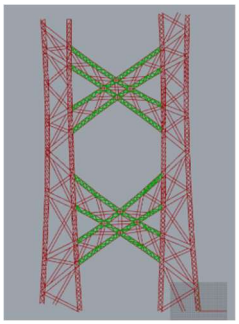


図 3.30　Step8

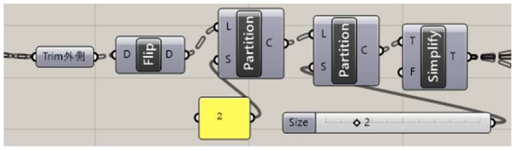


図 3.31　Step8-1　GH

Step6 の Trim で切り取った後の外側部分について、データの整理をする。Flip は list の行と列を逆転させるコンポーネントである。向かい合う 2 つの曲線ごとに list に格納する。

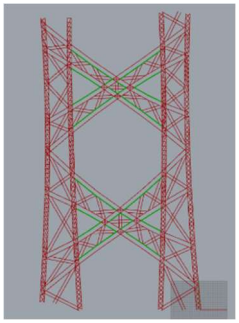


図 3.32　Step8-2

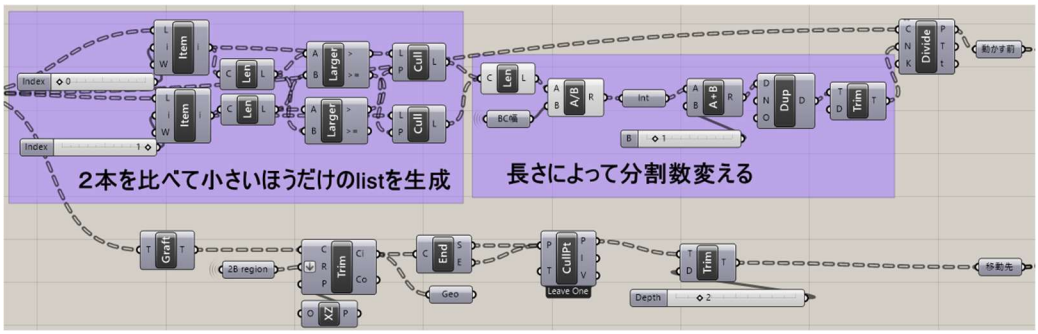


図 3.33　Step8-2　GH

向かい合うふたつの曲線の分割数を揃えるために、2 本のうち短いもののみを抽出する。これを BC 部の幅で割り、分割数の基準とする。

このあとの流れは Step5 とほぼ同様であるため、省略する。

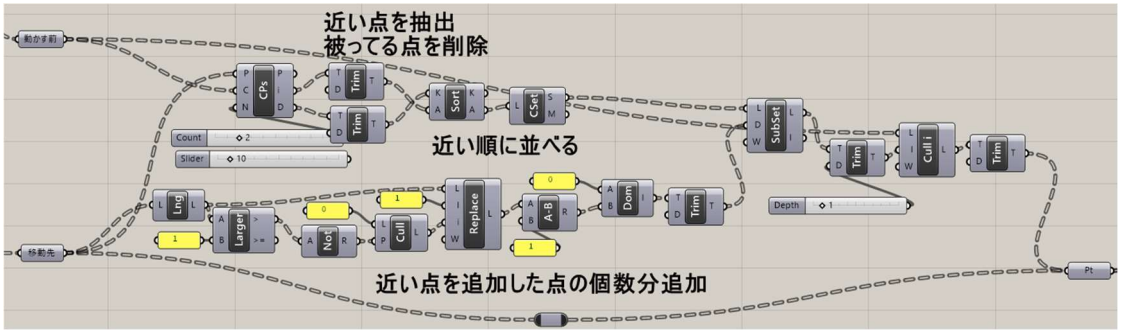


図 3.34　Step8-3　GH

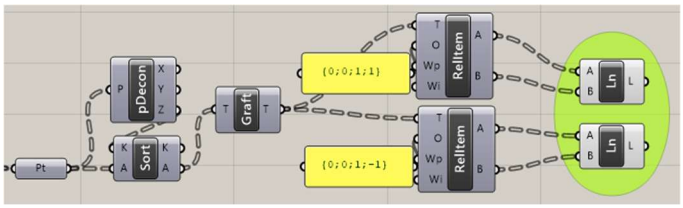


図 3.35　Step8-4　GH

Step9　BC 端部　斜材

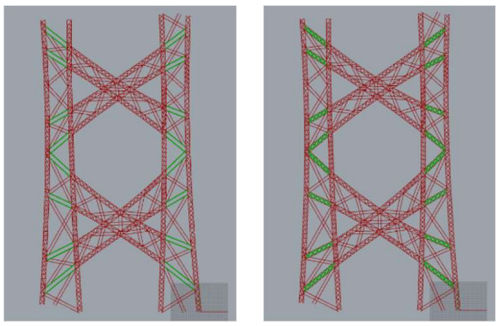


図 3.36　Step9

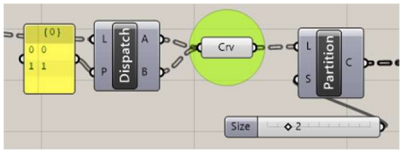


図 3.37　Step9-1　GH

データを整理し、向かい合わせの 2 つの曲線ごとに list に格納する。

以下 Step8 と同様にして BC 端部の斜材を得る。

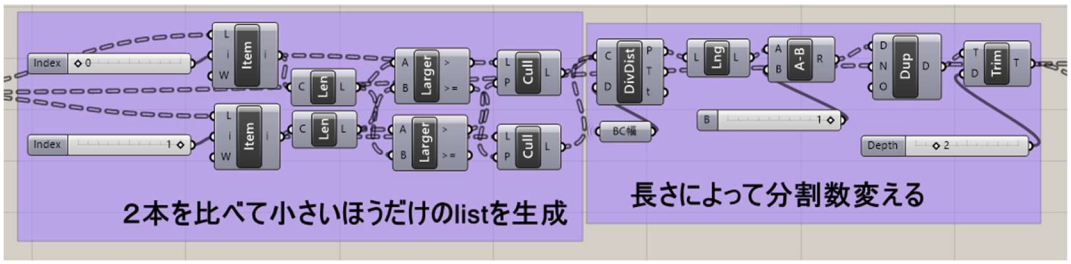


図 3.38　Step9-2　GH

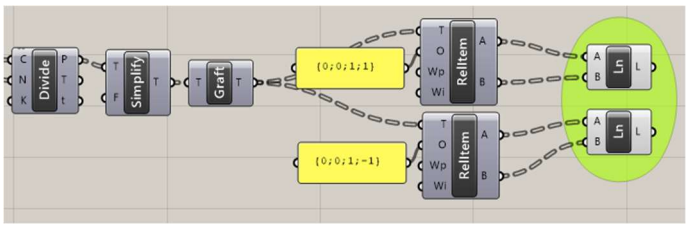


図 3.39　Step9-3　GH

Step10　Brace 同士の交差部の検出

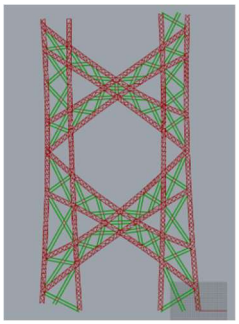


図 3.40　Step10

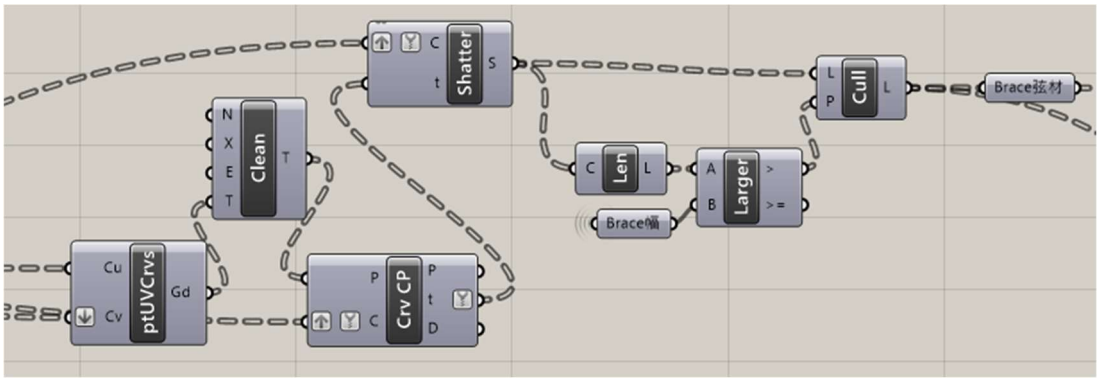


図 3.41　Step10　GH

ptUVcvs を使い、全ての Brace 弦材の交点を求める。この点を用いて弦材を分割し、Brace 幅より小さいものを取り除くことで Brace の非交差部の弦材を得る。

Step11　Brace 同士の交差部　斜材

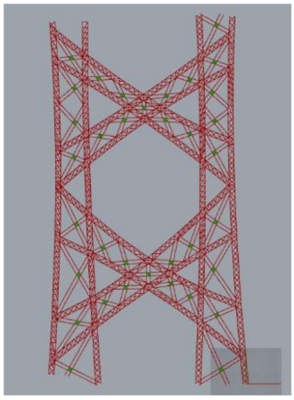


図 3.42　Step11

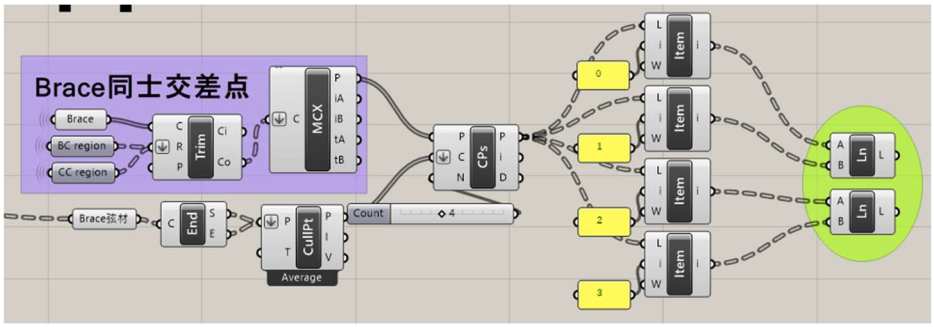


図 3.43　Step11　GH

Step0 で入力した Brace 中心線について、BC および CC の Region で Trim を行い、Brace 同士の交点を得る。同時に Step10 で求めた非交差部の弦材について、端点を取り出し、重複のない list を作成する。Closest Points を用いて、この list の中から 4 つ、Brace 中心線の交点に近い点を選び取ることで、Brace 交差部の斜材の端点が抽出できる。CPs は距離の近い順に点を抽出するため、位置関係から、1 番目と 2 番目、3 番目と 4 番目をそれぞれつなぐことで斜材を得る。

Step12　Brace 非交差部　斜材

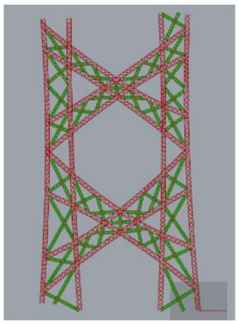


図 3.44　Step12

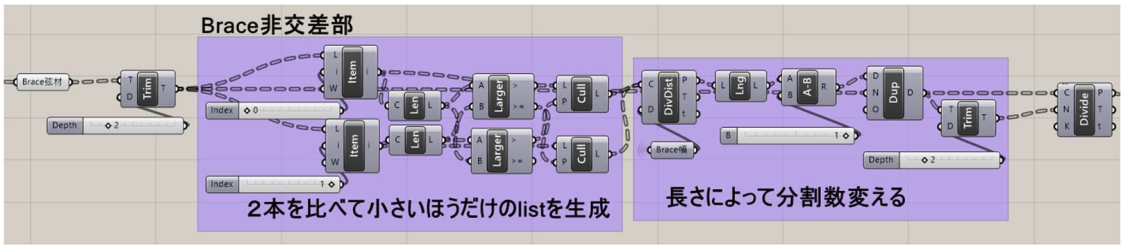


図 3.45　Step12-1　GH

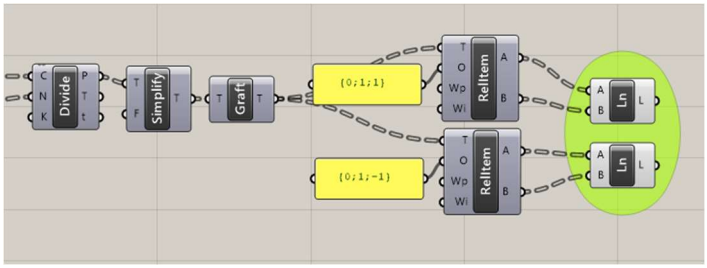


図 3.46　Step12-2　GH

Step9 と同様にして、Brace 非交差部の斜材が生成される。

４．試設計

４－１　モデル

立体モデル、平面モデルの２つをプロトタイプとした。図 4.3, .4.4 にモデル図を示す。
立体モデルは 300mm 角のラチス柱が組み合わさり交差部を持つ立体形状、
平面モデルは Lattice2 形式の柱の一部を取り出した形状とした。

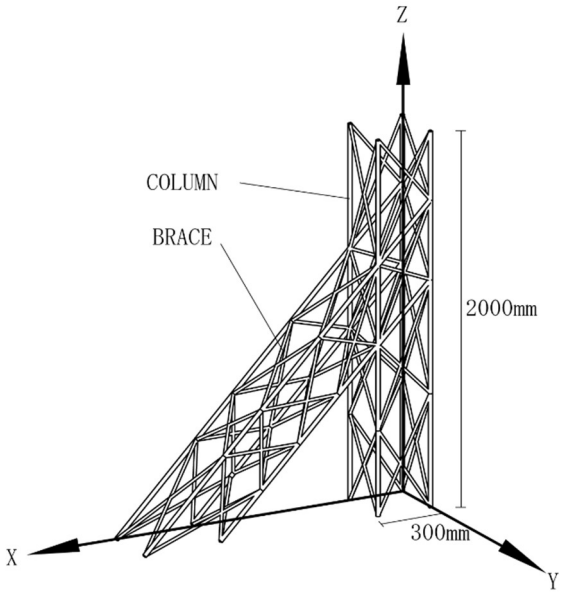


図 4.1　立体モデル図

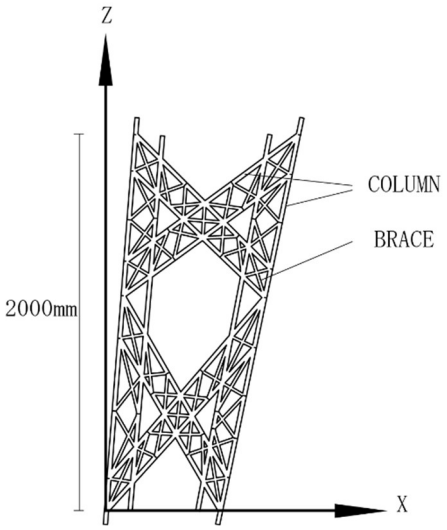


図 4.2　平面モデル図

４－２　形態解析

平面モデル、立体モデルのそれぞれに対し、使用するアルゴリズムおよびパラメタを変えて形態解析を行う。ただし、材料は鋼材 SN400 とした。入力した特性値は表 4.3 に示す。

表 4.1　SN490 の特性値

比重 [g/cm^3]	0.78
ヤング係数 [N/mm^2]	2.05×10^3
ポアソン比	0.333

図 4.3 - 4.7 は、平面モデル 062 の形態解析の様子である。全 7 ステップで形態解析が収束しており、途中形状が変化していく様子が見て取れる。

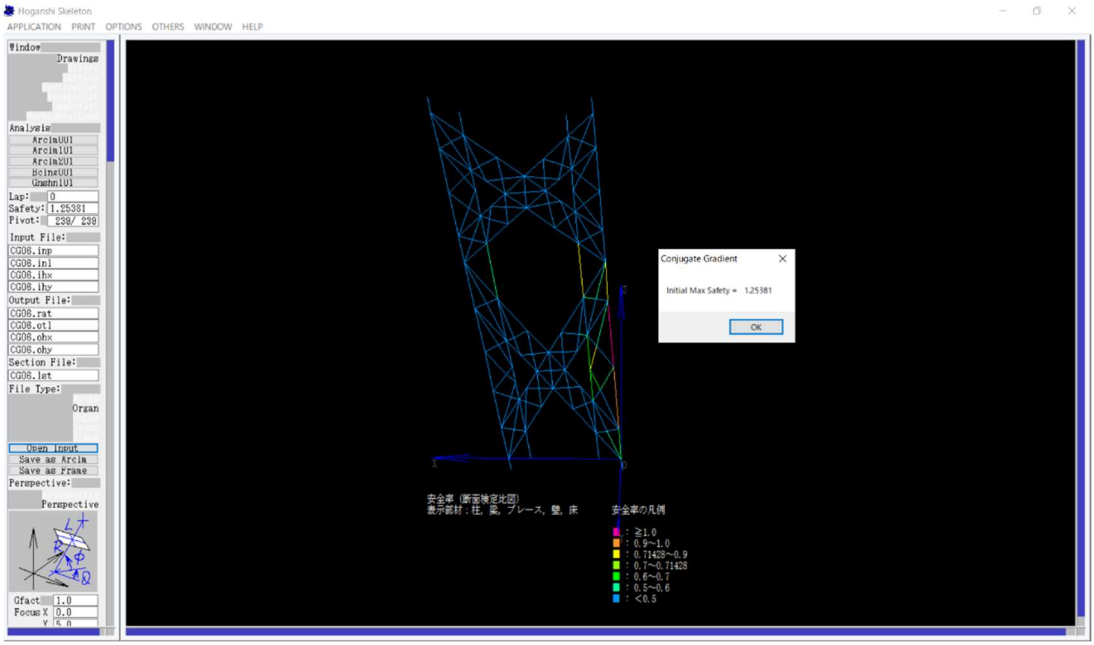


図 4.3　Initial Gradient

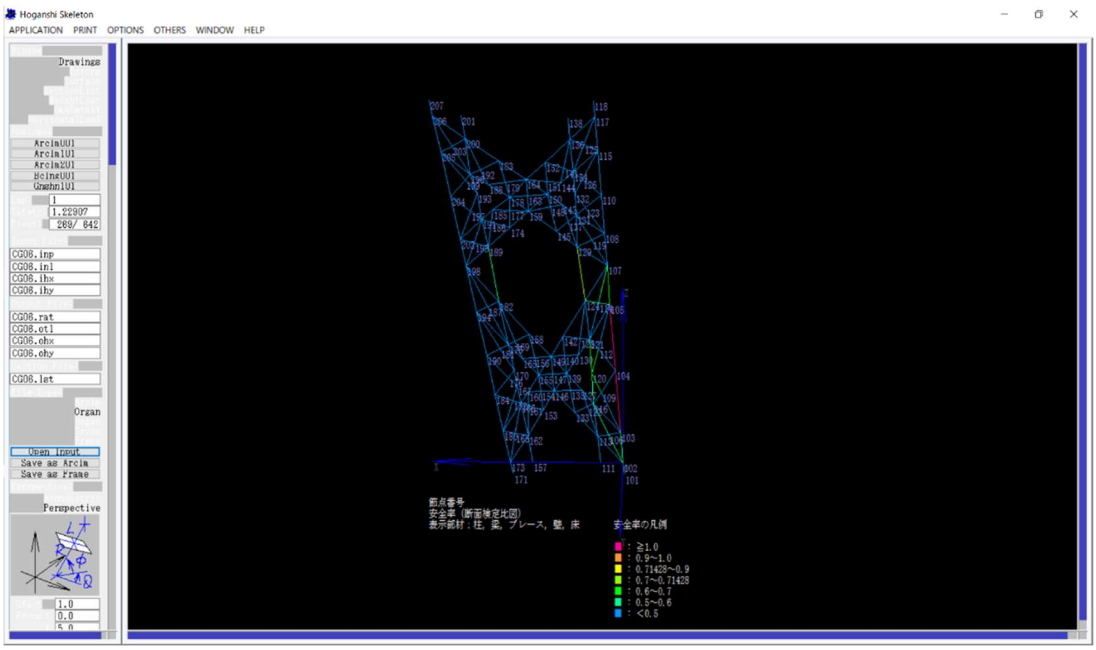


図 4.4　Lap 1

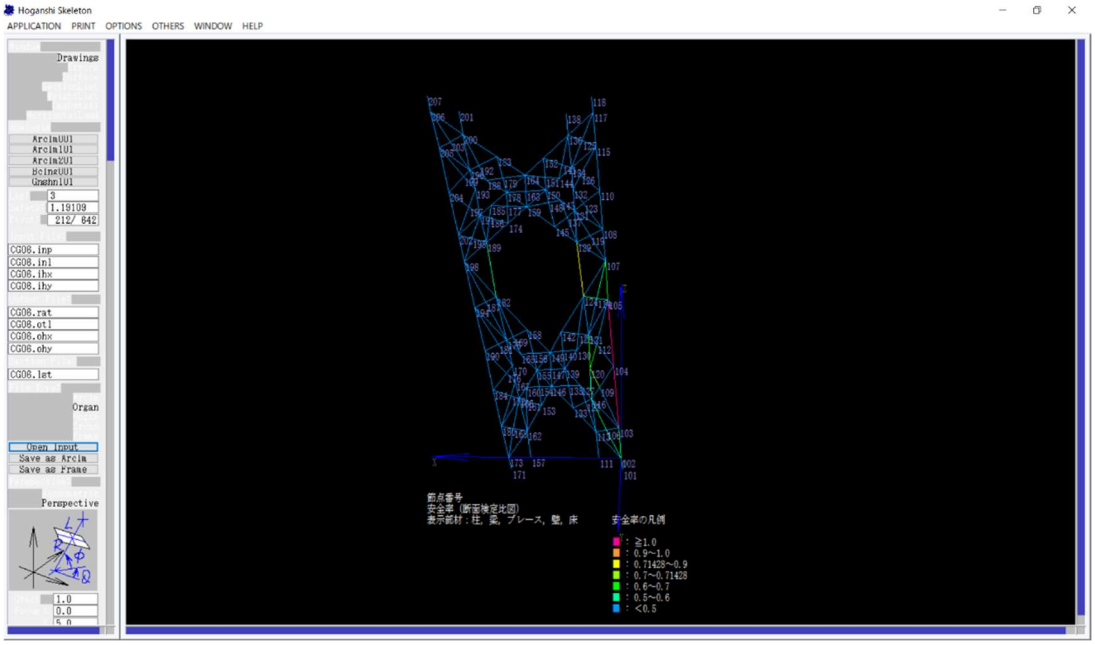


図 4.5　Lap 3

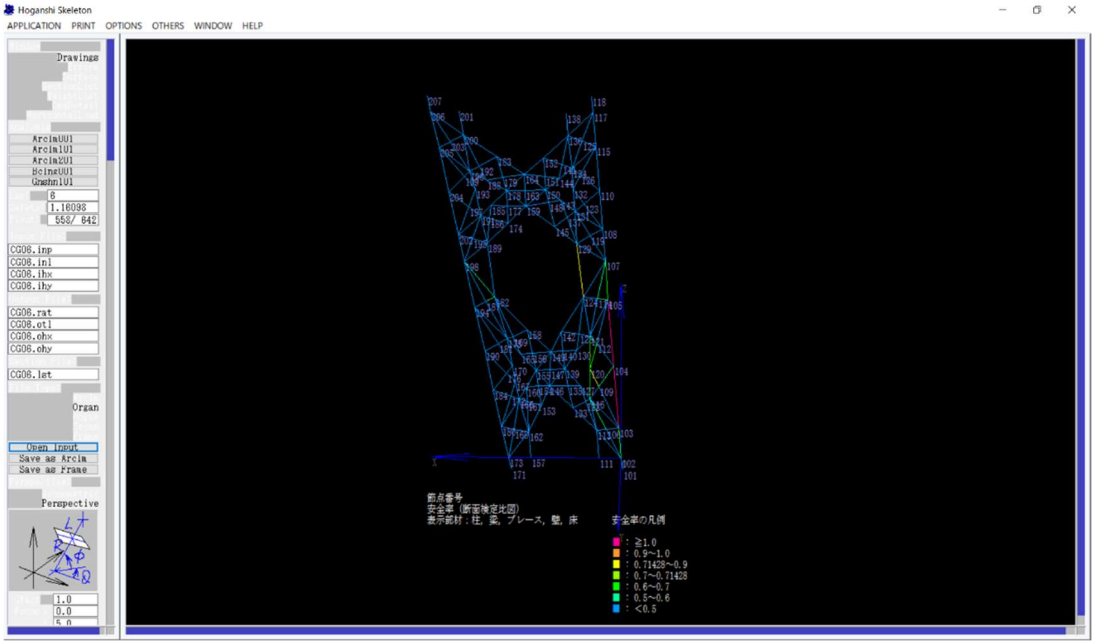


図 4.6　Lap 6

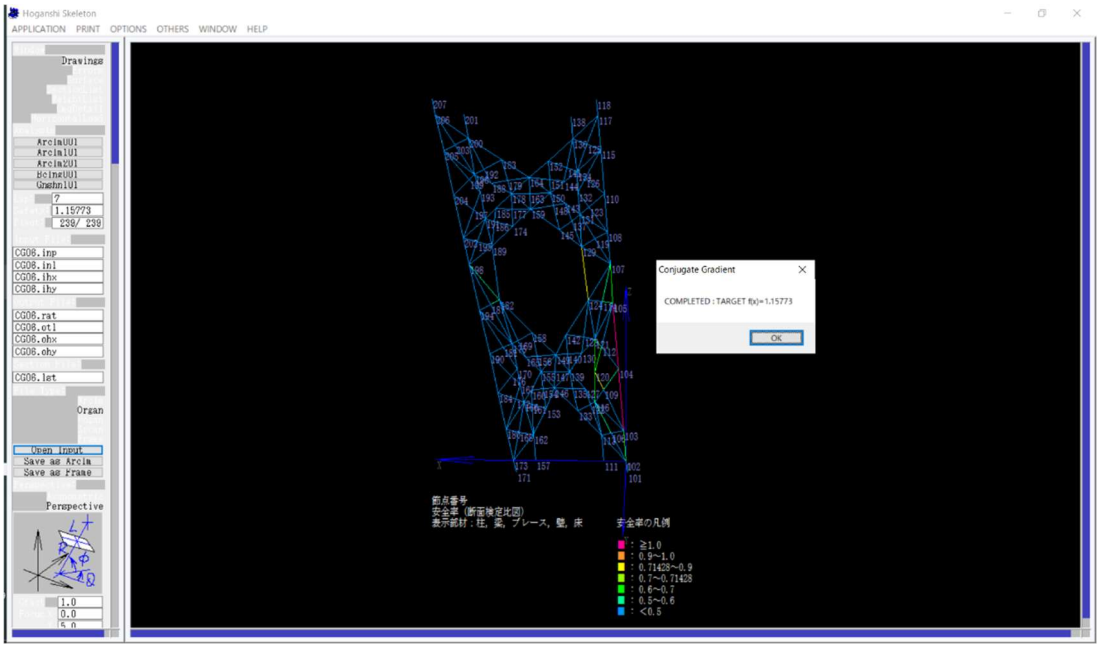


図 4.7　収束

他のモデルに関し、解析結果を表 4.2, 4.3, 4.4, 4.5 に示す。図中、アルゴリズム CGpinch を用いたものに関しては、赤丸で囲まれた節点を移動させる節点として指定した。また、アルゴリズム CGcurve を用いたものに関しては赤で示した部分を一本の弦材として入力している。

表 4.2　立体モデル　CGpinch

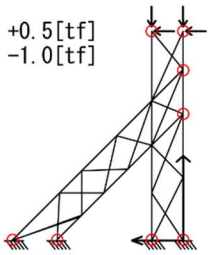
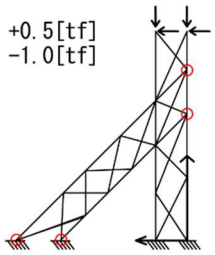
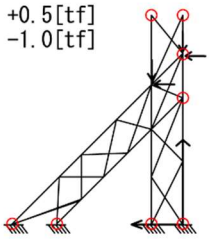
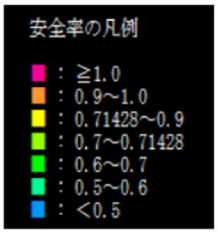
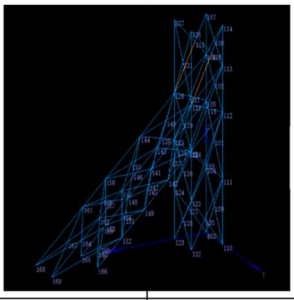
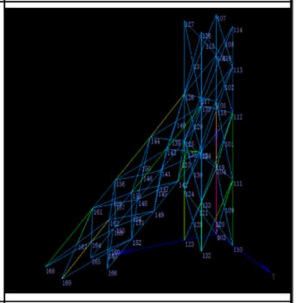
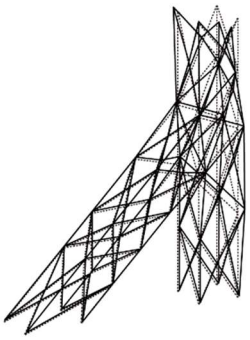
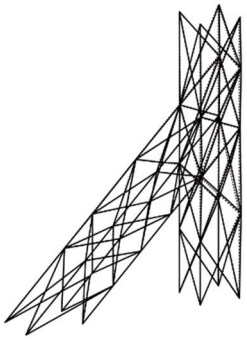
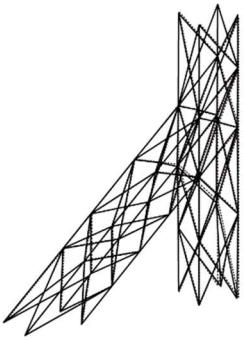
File Name	056	057	059
Algorithm	CGpinch	CGpinch	CGpinch
	X +0.5[tf] Z -1.0[tf] 	X +0.5[tf] Z -1.0[tf] 	X +0.5[tf] Z -1.0[tf] 
初期の安全率 			
断面 Maximum safety rate	0.98991　0.89706	0.98991　0.96409	1.25220　0.99121
Vector Size	0.00016	0.00000	0.00000
Steps	4	4	39
Parameter	ftarget=0.900000 gamma=0.000100 dfact=0.005000 eps=0.000001 npinch=1	ftarget=0.900000 gamma=0.000300 dfact=0.005000 eps=0.000001 npinch=1	ftarget=1.000000 gamma=0.000100 dfact=0.005000 eps=0.000001 npinch=1
COLUMN BRACE	32 * 32 [mm²] 22 * 22 [mm²]	32 * 32 [mm²] 22 * 22 [mm²]	23 * 23 [mm²] 15 * 15 [mm²]
変形図			

表 4.3　立体モデル　CGcurve

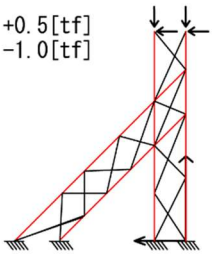
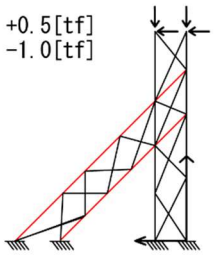
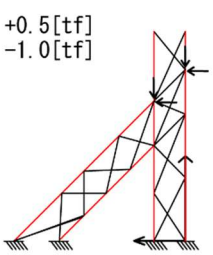
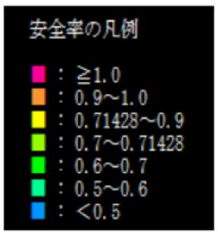
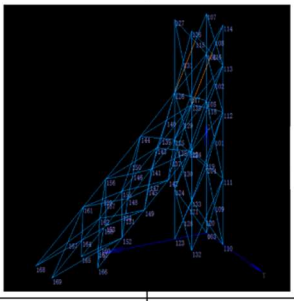
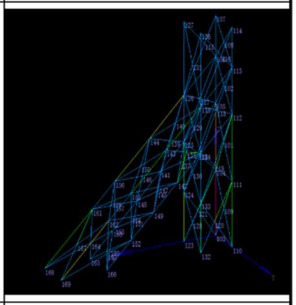
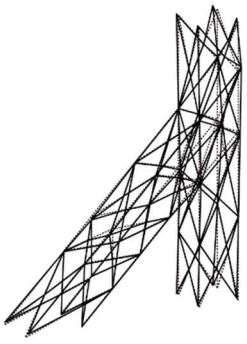
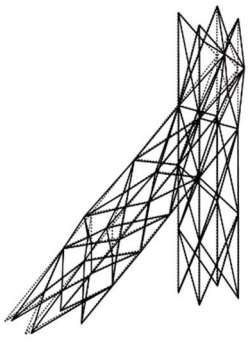
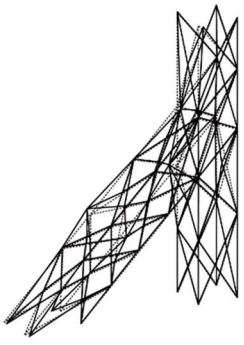
File Name	052	053	054
Algorithm	CGcurve	CGcurve	CGcurve
	X +0.5[tf] Z -1.0[tf] 	X +0.5[tf] Z -1.0[tf] 	X +0.5[tf] Z -1.0[tf] 
初期の安全率 			
Maximum safety rate	0.98991　0.87839	0.98991　0.79711	1.25220　0.97616
Vector Size	0.00057	0.00011	0.00000
Steps	2	3	5
Parameter	ftarget=0.900000 gamma=0.001000 dfact=0.010000 eps=0.000001	ftarget=0.900000 gamma=0.001000 dfact=0.010000 eps=0.000001	ftarget=0.900000 gamma=0.000500 dfact=0.005000 eps=0.000001
断面　COLUMN BRACE	32 * 32 [mm²] 22 * 22 [mm²]	32 * 32 [mm²] 22 * 22 [mm²]	23 * 23 [mm²] 15 * 15 [mm²]
変形図 			

表 4.4　平面モデル　CGpinch

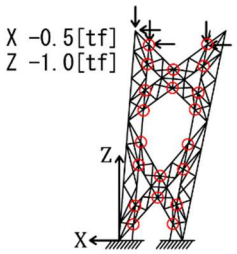
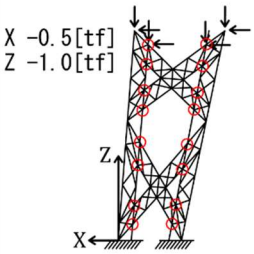
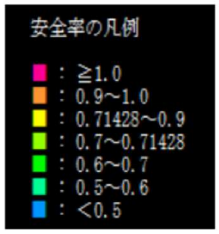
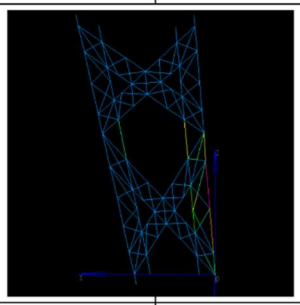
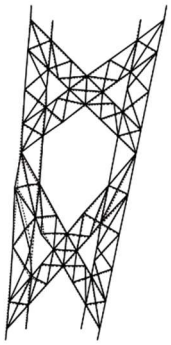
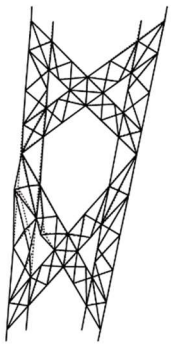
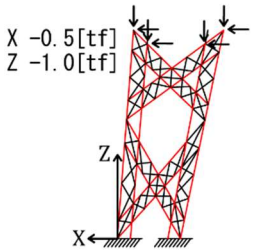
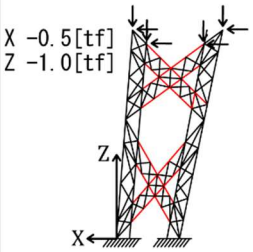
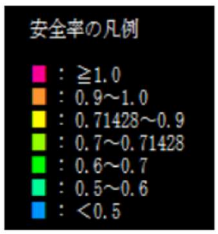
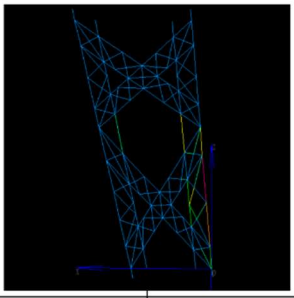
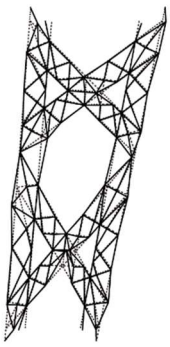
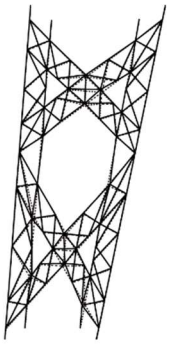
File Name	065	066	
Algorithm	CGpinchxz	CGpinchxz	
			
初期の安全率 			
Maximum safety rate	1.25381　1.04592	1.25381　1.05021	
Vector Size	0.00000	0.00000	
Steps	16	13	
Parameter	ftarget=1.000000 gamma=0.000300 dfact=0.001000 eps=0.000001 npinch=1	ftarget=1.000000 gamma=0.000300 dfact=0.001000 eps=0.000001 npinch=1	
断面　COLUMN BRACE	22 * 22 [mm²] 15 * 15 [mm²]	22 * 22 [mm²] 15 * 15 [mm²]	
変形図			

表 4.5　平面モデル　CGcurve

File Name	061	062	
Algorithm	CGcurvexz	CGcurvexz	
			
初期の安全率 			
Maximum safety rate	1.25381　0.94778	1.25381　1.15773	
Vector Size	0.00000	0.00000	
Steps	73	7	
Parameter	ftarget=0.950000 gamma=0.000100 dfact=0.005000 eps=0.000001	ftarget=1.000000 gamma=0.000100 dfact=0.010000 eps=0.000001	
断面　COLUMN BRACE	22 * 22 [mm²] 15 * 15 [mm²]	22 * 22 [mm²] 15 * 15 [mm²]	
変形図			

４－３　ラチス化

４－３－１　縮約

本論の構造解析は、Lattice^3 中の最小のラチス柱を縮約することで、Lattice^2 形式として処理したと解釈できる。したがって、等価な Lattice^3 形式へとラチス化するにあたり、モデルの各部材について、軸剛性・曲げ剛性の等価なラチス材について検討を行う必要がある。

軸剛性と曲げ剛性について一致するため、次項のような検討を行う。

軸剛性の算出

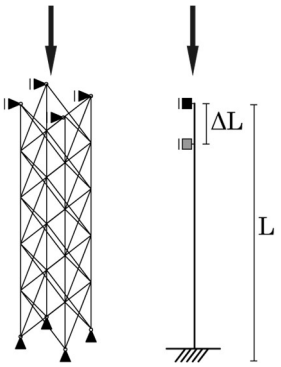


図 4.8　軸剛性の算出

図 4.8 について表 4.6 のように端部節点の拘束条件を設定する。

表 4.6　軸剛性算出における端部節点の拘束条件

		ラチス骨組	単部材
上端部	x,y 方向	ピン	剛
	z 方向	ローラー	ローラー
下端部	x,y,z 方向	ピン	剛

ラチス骨組は上下 4 つずつの節点があることを考慮すると、拘束条件はほぼ同等とみなせる。
ラチス骨組には 4 つの上端節点に対しそれぞれ Z 軸負の向きに N/4 の軸力を加える。これと、一本の部材に N の軸力を加えたものを比較する。

このとき、変形と断面積の関係は式に示される。

$$\frac{N}{A} = E\frac{\Delta L}{L} \tag{4.1}$$

（N ：軸力， A ：断面積， E ：弾性率， L ：全体長さ， ⊿L ：変位）
したがって、ひとつの部材とあるラチスの骨組が同じ荷重に対し同じ変形量であるとき、二者は同等の軸剛性を持つといえる。

曲げ剛性の算出

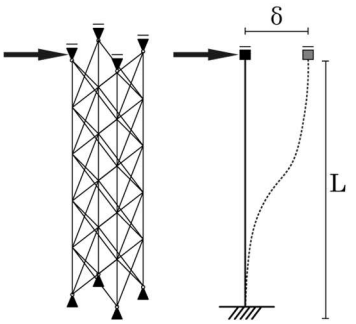


図 4.9　曲げ剛性の算出

図 4.9 について、表 4.2 のように端部節点の拘束条件を設定する。

表 4.7　曲げ剛性算出における端部節点の拘束条件

		ラチス骨組	単部材
上端部	x 方向	ローラー	ローラー
	y,z 方向	ピン	剛
下端部	x,y,z 方向	ピン	剛

軸剛性同様、ラチス骨組は上下 4 つずつの節点があることを考慮すると、拘束条件はほぼ同等とみなせる。ラチス骨組には 4 つの上端節点に対しそれぞれ X 軸の向きに Q/4 のせん断応力を加える。これと、一本の部材に Q のせん断応力を加えたものを比較する。

このとき、変形と断面二次モーメントの関係は式に示される。

$$\delta = \frac{Q \cdot L^3}{12EI} \tag{4.2}$$

（Q ：せん断応力， L ：全体長さ， E=2100 [tf/cm] ：弾性率， I ：断面二次モーメント）
したがって、ひとつの部材とあるラチスの骨組が同じ荷重に対し同じ変形量であるとき、二者は同等の曲げ剛性を持つといえる。

ラチスへの変換

平面モデル 061 について、弦材の 22mm 角材、斜材の 15mm 角材をそれぞれ、弦材の中心間の距離を角材の幅と同等とするラチス柱に変換する。

軸剛性、曲げ剛性の算出にあたり荷重はいずれも 9.8 [N]とし、長さは 400 [mm]とした。結果を表 4.8 に示す。
ただし、このラチス柱は実際の施工においては金属板の切り出し、あるいは 3 D プリンターを使用するものと仮定し、したがって斜材端部と弦材は剛に接しているとした。また、ラチス柱の鉄骨量は（断面積）×（中心線の長さ）×（比重）から求めているため、実際より多少大きい値となっている。

表 4.8A　角材単体の各剛性算出時の変位および鉄骨量

	22mm 角	15mm 角
軸剛性算出時変位 [m]	0.004	0.008
曲げ剛性算出時変位 [m]	1.302	6.047
鉄骨量 [kg]	1.510	0.702

表 4.8B　変換されたラチス柱

元の角材	22mm 角	15mm 角
ラチス弦材幅 [mm]	6.35	4.00
ラチス斜材幅 [mm]	5.00	2.00
軸剛性算出時変位 [m]	0.000	0.029
曲げ剛性算出時変位 [m]	1.170	3.527
鉄骨量 [kg]	1.374	0.340
鉄骨の削減率 [%]	9.01	51.5

このモデルの場合、骨組全体の鉄骨量としては 22.5%の削減になった。

４－３－２　ラチス化アルゴリズムの適用

４－３－１で算出した値を用いて骨組をラチス化する。

図 4.10 に生成結果を示す。図左が Lattice[3] 形式の骨組の線モデル、図右が材の太さを反映したモデルである。

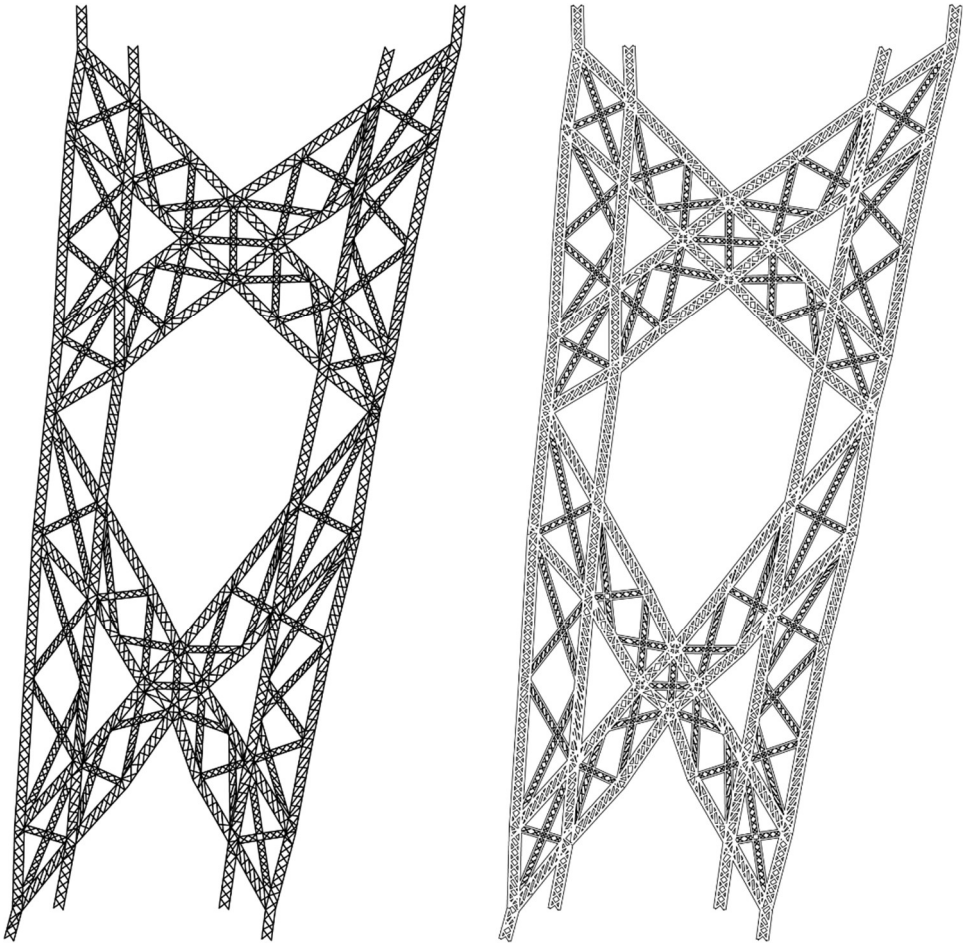


図 4.10　Lattice[3] 形式骨組の線モデル（左）と材の幅を反映したモデル（右）

４－４　考察

まず４－２の形態解析については、特に平面モデルに関して、応力の大きい部分で直線形状からテーパー形状への推移が見られ、テーパー形状の有効性が確認できた。

例として示した以外のパラメタのなかでは解析中に骨組が大きく崩れてしまう事例が多々あった。これは、目的関数が「全ての部材の安全率のなかで最大の値」であるがために、節点操作の条件によっては、形態解析中、パラメタによっては骨組の形状が発散して異常に歪になる事例が多々あった。これは、目的関数が「全ての部材の安全率のなかで最大の値」であるがために、形状操作の条件によっては、クリティカルな部分さえ良くなっていれば、そこへ影響の少ない部分が力学的に良くない形でも、目的関数は良くなるためと考えられる。
荷重と全体形状の関係からクリティカルな部材に対しどの部分の形状を操作するのか、前もっての検討が必要である。あるいは、目的関数に安全率の分布を反映させることで、ひとつの材に集中せず、全体に対して有効な形態変化を行うことができるようになる可能性がある。

骨組のラチス化に関しては、鉄骨量が 22.5%の削減となり、材料の削減に大きく寄与することが確認できた。しかしながら、反映したモデルでは透明感にやや劣るモデルと思われる。必要な剛性に対し、Lattice[1] の段階から全体形状の検討を行うことで、より透明感のある形状も出てくると考える。また、本モデルは高さが 2m であるが、建築大の骨組では各部材幅も大きくなる。スケールの大きな骨組に対しラチス化を行うことで、材の細さが感じられるようになり、透明感が出ると考える。当モデルの形状はレーザーカットなどには適した大きさでもあり、施工法により目指す形状を変えるという意味では良い例になった。

５．まとめ

５－１　本論の成果

本論では、より少ない材料で、強固かつ透明感のある構造体をつくる手法として入れ子状のLattice^{n}形式を取りあげ、中でもLattice^{3}形式に注目した。

共役勾配法をベースとし、部材の安全率の最大値骨組の生成および安全率最小化を行う一連のアルゴリズムを提案し、鉄骨量の削減をプロトタイプとして示した。

５－２　課題

今後の課題は、まず立体骨組に対しても中心線をベースとしたラチス化を行う手法を開発することである。

形態解析に関しては、目的関数に安全率の分布を反映することで、形態の変化の有効性が増す可能性がある。

作成した骨組の形態に対しより意匠面および施工面でも検討の余地がある。

参考文献

[1] 佐藤淳, 佐藤淳構造設計事務所のアイテム, INAX 出版, 2010.

[2] 腰塚武志, 計算幾何学と地理情報処理, 第２ 編, 共立出版, 1993.

[3] 伊勢坊健太, “Lattice3 形式の立体骨組を生成する基本アルゴリズム,” 2019.

[4] 金谷健一, これならわかる最適化数学, 共立出版, 2005, p. 249p.

[5] “"MX3D Bridge".MX3D.com,” MX3D, [オンライン]. Available: https://mx3d.com/projects/mx3d-bridge/. [アクセス日: 04 12 2019].

[6] “連立一次方程式・共役勾配法.PUKIWIKI,” 筑波大学, [オンライン]. Available: http://www.slis.tsukuba.ac.jp/~fujisawa.makoto.fu/cgi-bin/wiki/index.php?%CF%A2%CE%A91%BC%A1%CA%FD%C4%F8%BC%B0%A1%A7%B6%A6%CC%F2%B8%FB%C7%DB%CB%A1. [アクセス日: 06 01 2019].

謝辞

指導教員である佐藤淳先生には、わくわくする未来の見える、刺激的な研究内容を与えていただきました。構造の研究室にいながらも、特にコンピュータ言語に関してはほぼ初学者であった私に、一から丁寧にご指導いただき、おかげさまで論文を書き切るに至りました。学部から約３年間の研究室生活のなかで、佐藤先生は私にとって常に憧れの存在であり、本研究だけでなく、様々なワークショップ、授業、日々のミーティング等を通して、たくさんのことを教えてくださった先生には本当に感謝しています。これからも心の折れそうになったときには佐藤先生の励ましのお言葉を胸に頑張っていきたいと思います。

副指導を引き受けてくださった清家剛先生には、論文の背景といった本論の意義からアウトプットとしての形まで、数々の助言をいただきました。卒業論文から修士卒業に至るまで、続けてみてくださったからこそ、私のつまずく点もわかっていらっしゃったのかなと思います。大変お世話になりました。

佐藤研究室職員の荒木美香さん、古市渉平さん、ケンさん、また転職なさいましたが西村祐哉さんには、日頃から小さな問題の解決を手伝っていただきました。お忙しい中でも梗概やプレゼンの添削など、たくさんのお力添えをいただき感謝しております。

佐藤研究室および八階の同期や後輩のみなさんにも、研究内容から気持ちの持ちようまで、様々な形で支えていただきました。みなさんのおかげで楽しい研究室生活を送ることができました。

この場を借りて、本論文の執筆にあたりお世話になったすべての方々に感謝申し上げます。

藤本　月穂

付録

1-1. Conjugate Gradient

```
void conjugategradient(struct organ *org) /*CONJUGATE*/
/*OPTIMIZE ORGAN WITH CONJUGATE GRADIENT.*/
{
  FILE *fout,*ftxt;
  char non[10],str[256];
  int i,ii,j,jj,k,m,n;
  int nnode,nelem,aelem;
  double df,f1,f2,fa,ftarget,c1,c2,c3,alpha,beta,gamma,vsize,eps;
  double *x1,*x2,*xa,*xx,*dx,dfact; /*COORDINATES*/
  double *u1,*u2,*ua,*fgrad1,*fgrad2; /*GRADIENT VECTOR*/
  double **cmtx; /*MATRIX*/
  struct onode *ninit;

  ftxt=fopen("gradtest.txt","w");

  ftarget=0.8;
  gamma=0.001;
  dfact=0.05;
  eps=0.00001;

  /*POLYGON01.INP : SUCCESSFUL PARAMETERS*/
  /*
  ftarget=0.4;
  gamma=0.001;
  dfact=0.05;
  eps=0.00001;
  */
  /*
  ftarget=0.9;
  gamma=0.002;
  dfact=0.05;
  eps=0.00001;
  */
  /*
  ftarget=0.9;
  gamma=0.004;
  dfact=0.1;
  eps=0.00001;
  */
  /*
  ftarget=0.9;
  gamma=0.03;
  dfact=0.1;
  eps=0.001;
  */

  nnode=org->nnode;
```

```
  nelem=org->nelem;
  m=nnode;

  ninit=(struct onode *)malloc(nnode*sizeof(struct onode));
  fgrad1=mallocdoublevector(nnode); /*FOR Z OF ALL NODES*/
  fgrad2=mallocdoublevector(nnode); /*FOR Z OF ALL NODES*/
  u1=mallocdoublevector(nnode);
  u2=mallocdoublevector(nnode);
  ua=mallocdoublevector(nnode);

  /*INITIAL NODE*/
  for(i=0; i<nnode; i++) *(ninit+i)=*(org->nodes+i);

  /*MAXIMUM SAFETY*/
  arclmautomatic(org);
  fout=fopen((wdraw.childs+1)->otpfile,"r");
  if(fout==NULL) return;
  readsrcanrate(fout,&arc,NULL);
  fclose(fout);
  aelem=arc.nelem;
  f1=0.0;
  for(ii=0;ii<aelem;ii++)
  {
        for(jj=0;jj<4;jj++)
        {
          if(f1<(arc.elems+ii)->srate[jj]) f1=(arc.elems+ii)->srate[jj];
        }
  }

  sprintf(str,"%d",0);
  SetDlgItemText((wmenu.childs+2)->hwnd,ID_LAPS,str);
  SendDlgItemMessage((wmenu.childs+2)->hwnd,ID_LAPS,WM_PAINT,0,0);
  sprintf(str,"%.5f",f1);
  SetDlgItemText((wmenu.childs+2)->hwnd,ID_SAFETY,str);
  SendDlgItemMessage((wmenu.childs+2)->hwnd,ID_SAFETY,WM_PAINT,0,0);

  sprintf(str,"Initial Max Safety = %9.5f",f1);
  fprintf(ftxt,"%s¥n",str);
  MessageBox(NULL,str,"Conjugate Gradient",MB_OK);

  /*INITIAL GRADIENT*/
  fprintf(ftxt,"Initial Gradient¥n");
  for(i=0;i<m;i++)
  {
        for(j=0;j<m;j++)
        {
          if(j==i)
          {
                (org->nodes+j)->d[2]=((ninit+j)->d[2])+dfact; /*UPDATE ONLY Z OF
NODE I*/
```

```
          }
          else
          {
                (org->nodes+j)->d[2]=((ninit+j)->d[2]); /*RESET Z OF OTHER NODES*
/
          }
        }

        /*MAXIMUM SAFETY*/
        arclmautomatic(org);
        fout=fopen((wdraw.childs+1)->otpfile,"r");
        readsrcanrate(fout,&arc,NULL);
        fclose(fout);
        aelem=arc.nelem;
        df=0.0;
        for(ii=0;ii<aelem;ii++)
        {
          for(jj=0;jj<4;jj++)
          {
                if(df<(arc.elems+ii)->srate[jj]) df=(arc.elems+ii)->srate[jj];
          }
        }

        *(fgrad1+i)=gamma*(f1-df)/dfact; /*NEGATIVE GRADIENT*/
        *(u1+i)=*(fgrad1+i);

        fprintf(ftxt,"Node %d Gradient=%9.5f¥n",(org->nodes+i)->code,(f1-df)/dfac
t);
  }
  fprintf(ftxt,"¥n");

  k=0;
  while(1)
  {
        k++;
        sprintf(str,"%d",k);
        SetDlgItemText((wmenu.childs+2)->hwnd,ID_LAPS,str);

        for(i=0;i<m;i++)
        {
          (org->nodes+i)->d[2]=((ninit+i)->d[2])+(*(u1+i)); /*ONLY Z*/
        }

        /*MAXIMUM SAFETY*/
        arclmautomatic(org);
        fout=fopen((wdraw.childs+1)->otpfile,"r");
        readsrcanrate(fout,&arc,NULL);
        fclose(fout);
        aelem=arc.nelem;
        fa=0.0;
```

```
for(ii=0;ii<aelem;ii++)
{
  for(jj=0;jj<4;jj++)
  {
        if(fa<(arc.elems+ii)->srate[jj]) fa=(arc.elems+ii)->srate[jj];
  }
}

fprintf(ftxt,"Step %d Max Safety = %9.5f¥n",k,fa);

/*
sprintf(str,"Max Safety = %.5f",fa);
MessageBox(NULL,str,"Conjugate Gradient",MB_OK);
*/

/*INITIAL NODE*/
for(i=0;i<nnode;i++) *(ninit+i)=*(org->nodes+i);

/*GRADIENT*/
fprintf(ftxt,"Step %d Gradient¥n",k);
for(i=0;i<m;i++)
{
  for(j=0;j<m;j++)
  {
        if(j==i)
        {
          (org->nodes+j)->d[2]=((ninit+j)->d[2])+dfact; /*ONLY Z*/
        }
        else
        {
          (org->nodes+j)->d[2]=((ninit+j)->d[2]); /*ONLY Z*/
        }
  }

  /*MAXIMUM SAFETY*/
  arclmautomatic(org);
  fout=fopen((wdraw.childs+1)->otpfile,"r");
  readsrcanrate(fout,&arc,NULL);
  fclose(fout);
  aelem=arc.nelem;
  df=0.0;
  for(ii=0;ii<aelem;ii++)
  {
        for(jj=0;jj<4;jj++)
        {
          if(df<(arc.elems+ii)->srate[jj]) df=(arc.elems+ii)->srate[jj];
        }
  }

  *(ua+i)=-gamma*(fa-df)/dfact+(*(u1+i)); /*NEGATIVE GRADIENT*/
```

```
          fprintf(ftxt,"Node %d Gradient=%9.5f¥n",(org->nodes+i)->code,(fa-df)/dfa
ct);
        }
        fprintf(ftxt,"¥n");

        /*
        fprintf(stderr,"ITERATION %d¥n",k);
        fprintf(stderr," {u}= %8.3f %8.3f¥n",*(u1+0),*(u1+1));
        fprintf(stderr,"A{u}= %8.3f %8.3f¥n",*(ua+0),*(ua+1));
        */

        c1=0.0;
        c2=0.0;
        for(i=0; i<m; i++)
        {
          c1+=(*(fgrad1+i))*(*(fgrad1+i));
          c2+=(*(u1+i))*(*(ua+i));
        }
        alpha=c1/c2;

        /*
        fprintf(stderr,"Alpha= %8.3f = %8.3f / %8.3f¥n",alpha,c1,c2);
        */

        vsize=0.0;
        for(i=0;i<m;i++)
        {
          (org->nodes+i)->d[2]=((ninit+i)->d[2])+alpha*(*(u1+i)); /*ONLY Z*/

          /*
          fprintf(stderr,"x = x + dx : %8.3f = %8.3f + %8.3f x %8.3f¥n",*(x2+i),*
(x1+i),alpha,*(u1+i));
          */

          *(fgrad2+i)=(*(fgrad1+i))-alpha*(*(ua+i));
          vsize+=(*(fgrad2+i))*(*(fgrad2+i));
        }

        /*MAXIMUM SAFETY*/
        arclmautomatic(org);
        fout=fopen((wdraw.childs+1)->otpfile,"r");
        readsrcanrate(fout,&arc,NULL);
        fclose(fout);
        aelem=arc.nelem;
        f2=0.0;
        for(ii=0;ii<aelem;ii++)
        {
          for(jj=0;jj<4;jj++)
          {
```

```
                if(f2<(arc.elems+ii)->srate[jj]) f2=(arc.elems+ii)->srate[jj];
          }
        }

        sprintf(str,"%d",k);
        SetDlgItemText((wmenu.childs+2)->hwnd,ID_LAPS,str);
        SendDlgItemMessage((wmenu.childs+2)->hwnd,ID_LAPS,WM_PAINT,0,0);
        sprintf(str,"%.5f",f2);
        SetDlgItemText((wmenu.childs+2)->hwnd,ID_SAFETY,str);
        SendDlgItemMessage((wmenu.childs+2)->hwnd,ID_SAFETY,WM_PAINT,0,0);

        sprintf(str,"Step=%d Max Safety=%.5f Vector Size=%.5f",k,f2,vsize);
        fprintf(ftxt,"%s¥n¥n",str);
        /*MessageBox(NULL,str,"Conjugate Gradient",MB_OK);*/

        if(vsize<eps || f2<=ftarget)
        {
          sprintf(str,"COMPLETED : TARGET f(x)=%.5f¥n",f2);
          MessageBox(NULL,str,"Conjugate Gradient",MB_OK);
          return;
        }

        c3=0.0;
        for(i=0; i<m; i++)
        {
          c3+=(*(fgrad2+i))*(*(fgrad2+i));
        }
        beta=c3/c1;

        for(i=0; i<m; i++)
        {
          /*UPDATE INITIAL NODE*/
          *(ninit+i)=*(org->nodes+i);

          /*GRADIENTS*/
          *(u2+i)=(*(fgrad2+i))+beta*(*(u1+i));
          *(u1+i)=*(u2+i);
          *(fgrad1+i)=*(fgrad2+i);
        }
  }

  fclose(ftxt);

  return;
}/*conjugategradient*/
```

1-1. Arclm Automatic

```
double arclmautomatic(struct organ *org) /*CONJUGATE*/
/*OPTIMIZE ORGAN WITH CONJUGATE GRADIENT.*/
{
  char str[256],non[80];
  int i,j,k,n,flag;
  FILE *fout;

  /*TURN OFF MESSAGES*/
  globalmessageflag=0;
  globaldrawflag=0;

  /*SAFETY FLAG*/
  (wdraw.childs+1)->vparam.vflag.ev.srcancolor=1;

  /*INITIAL*/
  arc =arci;
  arcx=arci;
  arcy=arci;
  /*free((wdraw.childs+1)->org.loads);*/

  for(i=0;i<org->nelem;i++)
  {
        for(j=0;j<2;j++)
        {
          for(k=0;k<6;k++) (org->elems+i)->initial[j][k]=0.0;
        }
  } /*INITIAL CMQ UNAVAILABLE.*/

  /*EXTRACT ARCLM*/
  extractarclmfromorgan(org,&arc,&arcx,&arcy);

  (wmenu.childs+2)->vparam.vflag.mv.ftype=F_ARCLM;

  /*SAVE AS ARCLM*/
  saveasarclm((wdraw.childs+1)->inpfilez,&arc);
  saveasarclm((wdraw.childs+1)->inpfilex,&arcx);
  saveasarclm((wdraw.childs+1)->inpfiley,&arcy);

  getviewparam((wmenu.childs+2)->hwnd,
                              &((wdraw.childs+1)->vparam));
  (wdraw.childs+1)->vparam.vflag.ev.deformation=0;

  /*ANALYSIS*/
  arclm001(&arc ,ID_INPUTFILEZ,ID_OUTPUTFILEZ);
  arclm001(&arcx,ID_INPUTFILEX,ID_OUTPUTFILEX);
  arclm001(&arcy,ID_INPUTFILEY,ID_OUTPUTFILEY);
```

```
  /*SRCAN*/
  n=strcspn((wdraw.childs+1)->sctfile,".");
  strncpy(str,(wdraw.childs+1)->sctfile,n);
  str[n]='¥0';

  i=srcan001(str);

  if(i==0) MessageBox(NULL,"Failed.","SRCAN001",MB_OK);

  /*RATE FILE NAME*/
  strcpy((wdraw.childs+1)->otpfile,str);
  strcat((wdraw.childs+1)->otpfile,".rat");
  SetDlgItemText((wmenu.childs+2)->hwnd,ID_OUTPUTFILE,
                                    (wdraw.childs+1)->otpfile);
  fout=fopen((wdraw.childs+1)->otpfile,"r");
  if(fout==NULL) return 0.0;
  readsrcanrate(fout,&arc,NULL);
  fclose(fout);

  /*REDRAW MODEL*/
  clearwindow(*(wdraw.childs+1));
  drawarclmframe((wdraw.childs+1)->hdcC,
                                    (wdraw.childs+1)->vparam,arc,0,ONSCREEN);
  SendMessage((wdraw.childs+1)->hwnd,WM_PAINT,0,0);
  (wmenu.childs+2)->vparam.vflag.mv.ftype=F_ORGAN;
  SendMessage((wmenu.childs+2)->hwnd,WM_INITDIALOG,0,0);

  globalmessageflag=1;
  globaldrawflag=1;

  return 0.0;
}/*arclmautomatic*/
```

2-1. CGpinch

```
void CGpinch(struct organ *org)/*Coded by Fujimoto*/
/*OPTIMIZE ORGAN WITH CONJUGATE GRADIENT.*/
{
  FILE *fout,*ftxt;
  char non[10],str[256];
  int i,ii,j,jj,k,m,n;
  int p,q;
  int nnode,nelem,aelem;
  double df,f1,f2,fa,ftarget,c1,c2,c3,alpha,beta,gamma,vsize,eps;
  double xi2,yi2,zi2,xj2,yj2,zj2;
  //double *x1,*x2,*xa,*xx,*dx; /*COORDINATES*/
  double dfact;
  double **u1,**u2,**ua,**fgrad1,**fgrad2; /*GRADIENT VECTOR*/
  struct onode *ninit; /*node-initial*/

  ftxt=fopen("gradtest.txt","w");

  nnode= org->nnode;
  nelem= org->nelem;
  m=nnode;


  /*パラメタ入力*/

  //for CG05.inp

  ftarget=0.99;
  gamma=0.005;
  dfact=0.01;
  eps=0.000001;
  int npinch=4;//周囲の点への影響度*/

  /*ftarget=0.99;
  gamma=0.0005;
  dfact=0.005;
  eps=0.000001;
  int npinch=4;//周囲の点への影響度*/



  int mpt[]={103,107,110,114,123,127,132,136,105,106,113,113,165,166,168,169};//動
かしたい点
  int nmpt= sizeof mpt / sizeof mpt[0];

  int codes[168];//本当はcodes[m]ってしたい
  for(i=0;i<m;i++)codes[i]=(org->nodes+i)->code;
  /*パラメタ終わり*/
```

```
ninit=(struct onode *)malloc(nnode*sizeof(struct onode));

fgrad1=mallocdoublematrixxyz(nnode,3); /*FOR XYZ OF ALL NODES*/
fgrad2=mallocdoublematrixxyz(nnode,3); /*FOR XYZ OF ALL NODES*/
u1=mallocdoublematrixxyz(nnode,3);
u2=mallocdoublematrixxyz(nnode,3);
ua=mallocdoublematrixxyz(nnode,3);


/*INITIAL NODE*/
for(i=0; i<nnode; i++) *(ninit+i)=*(org->nodes+i);

/*MAXIMUM SAFETY*/
arclmautomatic(org);
/*
n=strcspn((wdraw.childs+1)->sctfile,".");
strncpy(str,(wdraw.childs+1)->sctfile,n);
str[n]='¥0';
strcpy((wdraw.childs+1)->otpfile,str);
strcat((wdraw.childs+1)->otpfile,".rat");
SetDlgItemText((wmenu.childs+2)->hwnd,ID_OUTPUTFILE,
                                    (wdraw.childs+1)->otpfile);
*/
fout=fopen((wdraw.childs+1)->otpfile,"r");
if(fout==NULL) return;
readsrcanrate(fout,&arc,NULL);
fclose(fout);
aelem=arc.nelem;
f1=0.0;
for(ii=0;ii<aelem;ii++)
{
       for(jj=0;jj<4;jj++)
       {
         if(f1<(arc.elems+ii)->srate[jj]) f1=(arc.elems+ii)->srate[jj];
       }
}

sprintf(str,"%d",0);
SetDlgItemText((wmenu.childs+2)->hwnd,ID_LAPS,str);
SendDlgItemMessage((wmenu.childs+2)->hwnd,ID_LAPS,WM_PAINT,0,0);
sprintf(str,"%.5f",f1);
SetDlgItemText((wmenu.childs+2)->hwnd,ID_SAFETY,str);
SendDlgItemMessage((wmenu.childs+2)->hwnd,ID_SAFETY,WM_PAINT,0,0);

sprintf(str,"Initial Max Safety = %9.5f",f1);
fprintf(ftxt,"%s¥n",str);
MessageBox(NULL,str,"Conjugate Gradient",MB_OK);
```

```
  /*INITIAL GRADIENT*/
  fprintf(ftxt,"Initial Gradient¥n");
  for(i=0;i<m;i++)
  {
        for(p=0;p<3;p++)
        {

                for(q=0;q<nmpt;q++)
                {
                  if (codes[i]==mpt[q]&&(p!=2||(ninit+i)->d[p]!=0))
                  {
//sprintf(str,"MovePt=%d p=%d ninit=%.5f¥n",mpt[q],p,(ninit+i)->d[p]);
//MessageBox(NULL,str,"Conjugate Gradient",MB_OK);

                        (org->nodes+i)->d[p]=((ninit+i)->d[p])+dfact;
                        PinchMove(org,ninit,i,p,nnode,nelem,npinch,dfact);/*追従*
/

                        /*for(j=0;j<m;j++)
                        {

                                xi2=(org->nodes+i)->d[0];
                                yi2=(org->nodes+i)->d[1];
                                zi2=(org->nodes+i)->d[2];
                                xj2=(org->nodes+j)->d[0];
                                yj2=(org->nodes+j)->d[1];
                                zj2=(org->nodes+j)->d[2];

                                d2=sqrt((xi2-xj2)*(xi2-xj2)+(yi2-yj2)*(yi2-yj2)+
(zi2-zj2)*(zi2-zj2));
                                delta=d2/d1;

                                if(delta<0.3 && j!=i && (p!=2||(ninit+j)->d[p]!=
0))
                                {
                                        (org->nodes+j)->d[p]=((ninit+j)->d[p])+d
elta*dfact;
                                }

                        }*/


                        /*MAXIMUM SAFETY*/
                        arclmautomatic(org);
                        fout=fopen((wdraw.childs+1)->otpfile,"r");
                        readsrcanrate(fout,&arc,NULL);
                        fclose(fout);
                        aelem=arc.nelem;
```

```
                        df=0.0;
                        for(ii=0;ii<aelem;ii++)
                        {
                          for(jj=0;jj<4;jj++)
                          {
                                if(df<(arc.elems+ii)->srate[jj]) df=(arc.elems+i
i)->srate[jj];
                          }
                        }
                        *(*(fgrad1+i)+p)=gamma*(f1-df)/dfact; /*NEGATIVE GRADIENT
*/
                        *(*(u1+i)+p)=*(*(fgrad1+i)+p);

                        fprintf(ftxt,"Node %d Dim %d Gradient=%9.5f¥n",(org->node
s+i)->code,p,(f1-df)/dfact);

                        for(j=0;j<m;j++) (org->nodes+j)->d[p]=((ninit+j)->d[p]);

//sprintf(str,"Step initial MovePt=%d ¥n p=%d MaxSafety=%.5f¥n",mpt[q],p,df);
//MessageBox(NULL,str,"Conjugate Gradient",MB_OK);
                    }
                  }

        }
  }
  fprintf(ftxt,"¥n");

  /*
  fprintf(stderr,"INITIAL CONDITION¥n");
  fprintf(stderr,"x    = %8.3f %8.3f¥n",*(x1+0),*(x1+1));
  fprintf(stderr,"f(x)= %8.3f¥n",f1);
  fprintf(stderr,"grad f(x)= %8.3f %8.3f¥n",*(fgrad1+0),*(fgrad1+1));
  fprintf(stderr,"{u}       = %8.3f %8.3f¥n",*(u1+0),*(u1+1));
  fprintf(fout,"STEP 0 f %8.3f {x1,x2} %8.3f %8.3f¥n",f1,*(x1+0),*(x1+1));
  gets(non);
  */

  k=0;
  while(1)
  {
        k++;
        sprintf(str,"%d",k);
        SetDlgItemText((wmenu.childs+2)->hwnd,ID_LAPS,str);

        for(i=0;i<m;i++)
        {
                for(p=0;p<3;p++)//XZに限定
                {

                        for(q=0;q<nmpt;q++)
```

```
                        {
                          if (codes[i]==mpt[q]&&(p!=2||(ninit+i)->d[p]!=0))
                          {
                                (org->nodes+i)->d[p]=((ninit+i)->d[p])+(*(*(u1+i)
+p));
                                PinchMove(org,ninit,i,p,nnode,nelem,npinch,(*(*(u
1+i)+p)));/*追従*/

                                /*for(j=0;j<m;j++)
                                {

                                        xi2=(org->nodes+i)->d[0];
                                        yi2=(org->nodes+i)->d[1];
                                        zi2=(org->nodes+i)->d[2];
                                        xj2=(org->nodes+j)->d[0];
                                        yj2=(org->nodes+j)->d[1];
                                        zj2=(org->nodes+j)->d[2];

                                        d2=sqrt((xi2-xj2)*(xi2-xj2)+(yi2-yj2)*(y
i2-yj2)+(zi2-zj2)*(zi2-zj2));
                                        delta=d2/d1;

                                        if(delta<0.3 && j!=i && (p!=2||(ninit+j)
->d[p]!=0))
                                        {
                                                (org->nodes+j)->d[p]=((ninit+j)-
>d[p])+delta*dfact;
                                        }

                                }*/

                          }
                        }

                }
        }

        /*MAXIMUM SAFETY*/
        arclmautomatic(org);
        fout=fopen((wdraw.childs+1)->otpfile,"r");
        readsrcanrate(fout,&arc,NULL);
        fclose(fout);
        aelem=arc.nelem;
        fa=0.0;
        for(ii=0;ii<aelem;ii++)
        {
          for(jj=0;jj<4;jj++)
          {
                if(fa<(arc.elems+ii)->srate[jj]) fa=(arc.elems+ii)->srate[jj];
          }
```

```
        }

//sprintf(str,"Step %d Max Safety(fa) = %9.5f¥n",k,fa);
//MessageBox(NULL,str,"Conjugate Gradient",MB_OK);

        fprintf(ftxt,"Step %d Max Safety = %9.5f¥n",k,fa);

        /*
        sprintf(str,"Max Safety = %.5f",fa);
        MessageBox(NULL,str,"Conjugate Gradient",MB_OK);
        */

        /*INITIAL NODE*/
        for(i=0;i<nnode;i++) *(ninit+i)=*(org->nodes+i);

        /*GRADIENT*/
        fprintf(ftxt,"Step %d Gradient¥n",k);
        for(i=0;i<m;i++)
        {
                for(p=0;p<3;p++)//XYに限定
                {

                        for(q=0;q<nmpt;q++)
                        {
                          if(codes[i]==mpt[q]&&(p!=2||(ninit+i)->d[p]!=0))
                          {
                                (org->nodes+i)->d[p]=((ninit+i)->d[p])+dfact;
                                PinchMove(org,ninit,i,p,nnode,nelem,npinch,dfac
t);/*追従*/

                                /*for(j=0;j<m;j++)
                                {

                                        xi2=(org->nodes+i)->d[0];
                                        yi2=(org->nodes+i)->d[1];
                                        zi2=(org->nodes+i)->d[2];
                                        xj2=(org->nodes+j)->d[0];
                                        yj2=(org->nodes+j)->d[1];
                                        zj2=(org->nodes+j)->d[2];

                                        d2=sqrt((xi2-xj2)*(xi2-xj2)+(yi2-yj2)*(y
i2-yj2)+(zi2-zj2)*(zi2-zj2));
                                        delta=d2/d1;

                                        if(delta<0.3 && j!=i && (p!=2||(ninit+j)
->d[p]!=0))
                                        {
                                                (org->nodes+j)->d[p]=((ninit+j)-
>d[p])+delta*dfact;
                                        }
```

```
                                }*/

                                /*MAXIMUM SAFETY*/
                                arclmautomatic(org);
                                fout=fopen((wdraw.childs+1)->otpfile,"r");
                                readsrcanrate(fout,&arc,NULL);
                                fclose(fout);
                                aelem=arc.nelem;
                                df=0.0;
                                for(ii=0;ii<aelem;ii++)
                                {
                                        for(jj=0;jj<4;jj++)
                                        {
                                          if(df<(arc.elems+ii)->srate[jj]) df=(a
rc.elems+ii)->srate[jj];
                                        }
                                }

                                *(*(ua+i)+p)=-gamma*(fa-df)/dfact+(*(*(u1+i)+p));
 /*NEGATIVE GRADIENT*/


                                fprintf(ftxt,"Node %d Dim %d Gradient=%9.5f¥n",(o
rg->nodes+i)->code,p,(fa-df)/dfact);

                                for(j=0;j<m;j++) (org->nodes+j)->d[p]=((ninit+j)-
>d[p]);
                                //(org->nodes+i)->d[p]=(ninit+i)->d[p];
                            }
                        }

                }
        }
        fprintf(ftxt,"¥n");

        /*
        fprintf(stderr,"ITERATION %d¥n",k);
        fprintf(stderr," {u}= %8.3f %8.3f¥n",*(u1+0),*(u1+1));
        fprintf(stderr,"A{u}= %8.3f %8.3f¥n",*(ua+0),*(ua+1));
        */

        c1=0.0;
        c2=0.0;
        for(i=0; i<m; i++)
        {
                for(p=0;p<3;p++)
                {

                        for(q=0;q<nmpt;q++)
                        {
```

```
                                        if(codes[i]==mpt[q]&&(p!=2||(ninit+i)->d[p]!=0))
                                        {
                                                c1+=(*(*(fgrad1+i)+p))*(*(*(fgrad1+i)+
p));
                                                //c2+=((*(*(u1+i)+p))*(*(*(ua+i)+p)));
                                                c2+=(*(*(u1+i)+p))*(*(*(ua+i)+p));
                                        }
                                }

                        }
                }
                alpha=c1/c2;
//sprintf(str,"Step %d c1=%.5f c2=%.5f alpha=%9.5f¥n",k,c1,c2,alpha);
//MessageBox(NULL,str,"Conjugate Gradient",MB_OK);
                /*
                fprintf(stderr,"Alpha= %8.3f = %8.3f / %8.3f¥n",alpha,c1,c2);
                */

                vsize=0.0;
                for(i=0;i<m;i++)
                {
                        for(p=0;p<3;p++)//XYに限定
                        {

                                for(q=0;q<nmpt;q++)
                                {
                                  if(codes[i]==mpt[q]&&(p!=2||(ninit+i)->d[p]!=0))
                                  {
                                        (org->nodes+i)->d[p]=((ninit+i)->d[p])+alpha*(*(*
(u1+i)+p));
                                        PinchMove(org,ninit,i,p,nnode,nelem,npinch,alpha*
(*(*(u1+i)+p)));/*追従*/

                                        /*for(j=0;j<m;j++)
                                        {

                                                xi2=(org->nodes+i)->d[0];
                                                yi2=(org->nodes+i)->d[1];
                                                zi2=(org->nodes+i)->d[2];
                                                xj2=(org->nodes+j)->d[0];
                                                yj2=(org->nodes+j)->d[1];
                                                zj2=(org->nodes+j)->d[2];

                                                d2=sqrt((xi2-xj2)*(xi2-xj2)+(yi2-yj2)*(y
i2-yj2)+(zi2-zj2)*(zi2-zj2));
                                                delta=d2/d1;

                                                if(delta<0.3 && j!=i && (p!=2||(ninit+j)
->d[p]!=0))
                                                {
```

```
                                                    (org->nodes+j)->d[p]=((ninit+j)-
>d[p])+delta*dfact;
                                                }
                                    }*/


                                    /*
                                    fprintf(stderr,"x = x + dx : %8.3f = %8.3f + %8.3
f x %8.3f¥n",*(x2+i),*(x1+i),alpha,*(u1+i));
                                    */

                                    *(*(fgrad2+i)+p)=(*(*(fgrad1+i)+p))-alpha*(*(*(ua
+i)+p));
                                    vsize+=(*(*(fgrad2+i)+p))*(*(*(fgrad2+i)+p));
                                }
                            }

                }
        }

        /*MAXIMUM SAFETY*/
        arclmautomatic(org);
        fout=fopen((wdraw.childs+1)->otpfile,"r");
        readsrcanrate(fout,&arc,NULL);
        fclose(fout);
        aelem=arc.nelem;
        f2=0.0;
        for(ii=0;ii<aelem;ii++)
        {
          for(jj=0;jj<4;jj++)
          {
                if(f2<(arc.elems+ii)->srate[jj]) f2=(arc.elems+ii)->srate[jj];
          }
        }
//sprintf(str,"Step %d Max Safety(f2) = %9.5f¥n",k,f2);
//MessageBox(NULL,str,"Conjugate Gradient",MB_OK);

        sprintf(str,"%d",k);
        SetDlgItemText((wmenu.childs+2)->hwnd,ID_LAPS,str);
        SendDlgItemMessage((wmenu.childs+2)->hwnd,ID_LAPS,WM_PAINT,0,0);
        sprintf(str,"%.5f",f2);
        SetDlgItemText((wmenu.childs+2)->hwnd,ID_SAFETY,str);
        SendDlgItemMessage((wmenu.childs+2)->hwnd,ID_SAFETY,WM_PAINT,0,0);

        sprintf(str,"Step=%d Max Safety=%.5f Vector Size=%.5f",k,f2,vsize);
        fprintf(ftxt,"%s¥n¥n",str);
        /*MessageBox(NULL,str,"Conjugate Gradient",MB_OK);*/

        /*
        fprintf(stderr,"x    = %8.3f %8.3f¥n",*(x2+0),*(x2+1));
```

```
        fprintf(stderr,"f(x) = %8.3f¥n",f2);
        fprintf(stderr,"grad f(x)   = %8.3f %8.3f¥n",*(fgrad2+0),*(fgrad2+1));
        fprintf(stderr,"VECTOR SIZE = %9.5f¥n",vsize);
        fprintf(fout,"STEP %d f %8.3f {x1,x2} %8.3f %8.3f¥n",k,f2,*(x2+0),*(x2+
1));
        gets(non);
        */

        if(vsize<eps || f2<=ftarget)
        {
          sprintf(str,"COMPLETED : TARGET f(x)=%.5f¥n",f2);
          MessageBox(NULL,str,"Conjugate Gradient",MB_OK);
          return;
        }



        c3=0.0;
        for(i=0; i<m; i++)
        {
                for(p=0;p<3;p++)
                {

                        for(q=0;q<nmpt;q++)
                        {
                          if(codes[i]==mpt[q]&&(p!=2||(ninit+i)->d[p]!=0))
                          {
                                  c3+=(*(*(fgrad2+i)+p))*(*(*(fgrad2+i)+p));
                          }
                        }

                }
        }
        beta=c3/c1;
//sprintf(str,"Step %d Safety=%.5f Beta=%.5f¥n",k,f2,beta);
//MessageBox(NULL,str,"Conjugate Gradient",MB_OK);

        /*繰り返しのため引き継ぎ*/
        for(i=0; i<m; i++)
        {
                /*INITIAL NODE*/
                *(ninit+i)=*(org->nodes+i);

                for(p=0;p<3;p++)
                {

                        for(q=0;q<nmpt;q++)
                        {
                          if(codes[i]==mpt[q]&&(p!=2||(ninit+i)->d[p]!=0))
                          {
```

```
                                        *(*(u2+i)+p)=(*(*(fgrad2+i)+p))+beta*(*(*(u1+i)
+p));
                                        *(*(u1+i)+p)=*(*(u2+i)+p);
                                        *(*(fgrad1+i)+p)=*(*(fgrad2+i)+p);
                                  }
                                }

                        }
                }
          }
  }

  //fclose(ftxt);

  return;
}/*CGpinch*/
```

2-1. PinchMove

```
void PinchMove(struct organ *org,struct onode *ninit,int i,int p,int nnode,int nel
em,int npinch,double distance)/*for CGpinch*/
{
        int j,k;
        double delta;
        long int *nflag;

        /*隣のnodeを辿って探す*/
        nflag = (long int*)malloc(nnode*sizeof(long int));
        for (j = 0; j < nnode; j++) *(nflag + j) = 0;
        *(nflag + i ) = 1;

        for (j = 1; j <= npinch ; j++)/* いくつ探すか*/
        {
                for (k = 0; k < nelem; k++) {

                        if ((org->elems+k)->nnod == 2)
                        {
                                int loff=(*((org->elems+k)->nods+0))->loff;
                                int moff=(*((org->elems+k)->nods+1))->loff;

                                if (*(nflag + loff) == j && *(nflag + moff) == 0)
 {
                                        *(nflag + moff) = j + 1;
                                }
                                if (*(nflag + moff) == j && *(nflag + loff) == 0)
 {
                                        *(nflag + loff) = j + 1;
                                }
                        }
                }
        }

        /*近いnodeのみ移動*/
        for (j = 0; j < nnode; j++)
        {
                if(*(nflag + j) !=0 && j!=i && (p!=2 || (ninit+j)->d[p]!=0))
                {
                        delta =1/(*(nflag + j));
                        (org->nodes+j)->d[p]=((ninit+j)->d[p])+delta*distance;
                }
        }
}
```

3-1. CGcurve

```
void CGcurve(struct organ *org)/*Coded by Fujimoto*/
/*OPTIMIZE ORGAN WITH CONJUGATE GRADIENT.*/
{
  FILE *fout,*ftxt;
  char non[10],str[256];
  int i,ii,j,jj,k,m,n;
  int p,q,r;
  int *rmax;
  double *a0,*b0,*c0;
  int nnode,nelem,aelem;
  double df,f1,f2,fa,ftarget,c1,c2,c3,alpha,beta,gamma,vsize,eps;
 // double xi2,yi2,zi2,xj2,yj2,zj2;
  //double delta,d1,d2;
  //double *x1,*x2,*xa,*xx,*dx; /*COORDINATES*/
  double dfact;
  double **u1,**u2,**ua,**fgrad1,**fgrad2; /*GRADIENT VECTOR*/
  struct onode *ninit; /*node-initial*/

  ftxt=fopen("gradtest.txt","w");

  nnode=org->nnode;
  //nelem=org->nelem;

  //for CG05.inp

  ftarget=0.90;
  gamma=0.0003;
  dfact=0.005;
  eps=0.000001;

  /*
  ftarget=0.95;
  gamma=0.0005;
  dfact=0.005;
  eps=0.000001;

  ftarget=0.90;
  gamma=0.005;
  dfact=0.005;
  eps=0.000001;
  */

  sprintf(str,"ftarget=%f ¥ngamma=%f ¥ndfact=%f ¥neps=%f",ftarget,gamma,dfact,ep
s);
  fprintf(ftxt,"%s¥n",str);
  //fprintf(str,"ftarget=%f ¥ngamma=%f ¥ndfact=%f ¥eps=%f",ftarget,gamma,dfact,ep
s);
```

```
/*//for CG03.inp
ftarget=0.95;
gamma=0.0006;
dfact=0.01;
eps=0.000001;
//d1=3.0;//周囲の点への影響度*/

/*//forCG01.inp
ftarget=0.99;
gamma=0.0001;
dfact=0.001;
eps=0.000001;
*/


//ninit=(struct onode *)malloc(nnode*sizeof(struct onode));
//fgrad1=mallocdoublevector(nnode); /*FOR Z OF ALL NODES*/
//fgrad2=mallocdoublevector(nnode); /*FOR Z OF ALL NODES*/
//u1=mallocdoublevector(nnode);
//u2=mallocdoublevector(nnode);
//ua=mallocdoublevector(nnode);

//int mpt[]={101,102,115,122,119,123};//動かしたい点
//int nmpt= sizeof mpt / sizeof mpt[0];

/*for CG05.inp*/

int a[]={103,110,123,132,125,126,134,136};//動かしたい点のcode
int b[]={107,114,127,136,165,168,166,169};//動かしたい点のcode
//int c[]={107,108};//動かしたい点のcode
int nchord= sizeof a / sizeof a[0];

int *ai,*bi,*ci,**di;
ai=(int *)malloc(nchord*sizeof(int));
bi=(int *)malloc(nchord*sizeof(int));
ci=(int *)malloc(nchord*sizeof(int));
di=(int **)malloc(nchord*sizeof(int *));
for(q=0;q<nchord;q++) *(di+q)=(int *)malloc(20*sizeof(int));//ひとまず20

rmax=(int *)malloc(nchord*sizeof(int));

//int codes[23];//本当はcodes[m]ってしたい
//for(i=0;i<m;i++)codes[i]=(org->nodes+i)->code;

/*Chord両端入力*/
for(q=0;q<nchord;q++)
{
	*(ai+q)=a[q]-101;
	*(bi+q)=b[q]-101;
```

```
  }

  //int c[]={101,102,125,134,141,142,143,144};
  //for(q=0;q<nchord;q++)*(ci+q)=c[q]-101;

  /*Input Points on Chord*/
  for(q=0;q<nchord;q++)
  {

          double xa,ya,za,xb,yb,zb,xab,yab,zab;

          xa=(org->nodes+(*(ai+q)))->d[0];
          ya=(org->nodes+(*(ai+q)))->d[1];
          za=(org->nodes+(*(ai+q)))->d[2];

          xb=(org->nodes+(*(bi+q)))->d[0];
          yb=(org->nodes+(*(bi+q)))->d[1];
          zb=(org->nodes+(*(bi+q)))->d[2];

          xab=xb-xa;
          yab=yb-ya;
          zab=zb-za;

          r=0;
          double X,Y,Z;

          for(i=0;i<nnode;i++)
          {
                 if((i!=*(ai+q)) && (i!=*(bi+q)))
                 {

                         X=3.0;
                         Y=4.0;
                         Z=5.0;//数値は1以上、適当

                         double xi=(org->nodes+i)->d[0];
                         double yi=(org->nodes+i)->d[1];
                         double zi=(org->nodes+i)->d[2];

                         if (xab!=0 && yab!=0 && zab!=0)
                         {
                                 X=(xi-xa)/xab;
                                 Y=(yi-ya)/yab;
                                 Z=(zi-za)/zab;

                                 if(fabs(Y-Z)<0.01 && fabs(Z-X)<0.01 && fabs(X-Y)<
0.01)
                                 {
                                         *(*(di+q)+r)=i;
                                         r=r+1;
```

```
                }
        }

        else if(xab!=0 && yab!=0 && zab==0)
        {
                X=(xi-xa)/xab;
                Y=(yi-ya)/yab;

                if(fabs(X-Y)<0.01 && fabs(zi-za)<0.01)
                {
                        *(*(di+q)+r)=i;
                        r=r+1;
                }

        }
        else if(xab==0 && yab!=0 && zab!=0)
        {
                Y=(yi-ya)/yab;
                Z=(zi-za)/zab;

                if(fabs(Y-Z)<0.01 && fabs(xi-xa)<0.01)
                {
                        *(*(di+q)+r)=i;
                        r=r+1;
                }

        }
        else if(xab!=0 && yab==0 && zab!=0)
        {
                Z=(zi-za)/zab;
                X=(xi-xa)/xab;

                if(fabs(Z-X)<0.01 && fabs(yi-ya)<0.01)
                {
                        *(*(di+q)+r)=i;
                        r=r+1;
                }
        }

        else if(xab!=0 && yab==0 && zab==0)
        {
                X=(xi-xa)/xab;

                if(fabs(yi-ya)<0.01 && fabs(zi-za)<0.01)
                {
                        *(*(di+q)+r)=i;
                        r=r+1;
                }
        }
        else if(xab==0 && yab!=0 && zab==0)
```

```
                        {
                                Y=(yi-ya)/yab;

                                if(fabs(zi-za)<0.01 && fabs(xi-xa)<0.01)
                                {
                                        *(*(di+q)+r)=i;
                                        r=r+1;
                                }
                        }
                        else if(xab==0 && yab==0 && zab!=0)
                        {
                                Z=(zi-za)/zab;

                                if(fabs(xi-xa)<0.01 && fabs(yi-ya)<0.01)
                                {
                                        *(*(di+q)+r)=i;
                                        r=r+1;
                                }
                        }

                        *(rmax+q)=r;
                        int rmid=r/2;
                        if(r!=0) *(ci+q)=*(*(di+q)+rmid);
                }
        }

}

fgrad1=mallocdoublematrixxyz(nchord,3); /*FOR XYZ OF ALL NODES*/
fgrad2=mallocdoublematrixxyz(nchord,3); /*FOR XYZ OF ALL NODES*/
u1=mallocdoublematrixxyz(nchord,3);
u2=mallocdoublematrixxyz(nchord,3);
ua=mallocdoublematrixxyz(nchord,3);


/*INITIAL NODE*/
ninit=(struct onode *)malloc(nnode*sizeof(struct onode));
for(i=0; i<nnode; i++) *(ninit+i)=*(org->nodes+i);

/*MAXIMUM SAFETY*/
arclmautomatic(org);

/*
n=strcspn((wdraw.childs+1)->sctfile,".");
strncpy(str,(wdraw.childs+1)->sctfile,n);
str[n]='¥0';
strcpy((wdraw.childs+1)->otpfile,str);
strcat((wdraw.childs+1)->otpfile,".rat");
SetDlgItemText((wmenu.childs+2)->hwnd,ID_OUTPUTFILE,
                                (wdraw.childs+1)->otpfile);
```

```
  */
  fout=fopen((wdraw.childs+1)->otpfile,"r");
  if(fout==NULL) return;
  readsrcanrate(fout,&arc,NULL);
  fclose(fout);
  aelem=arc.nelem;
  f1=0.0;
  for(ii=0;ii<aelem;ii++)
  {
        for(jj=0;jj<4;jj++)
        {
          if(f1<(arc.elems+ii)->srate[jj]) f1=(arc.elems+ii)->srate[jj];
        }
  }

  sprintf(str,"%d",0);
  SetDlgItemText((wmenu.childs+2)->hwnd,ID_LAPS,str);
  SendDlgItemMessage((wmenu.childs+2)->hwnd,ID_LAPS,WM_PAINT,0,0);
  sprintf(str,"%.5f",f1);
  SetDlgItemText((wmenu.childs+2)->hwnd,ID_SAFETY,str);
  SendDlgItemMessage((wmenu.childs+2)->hwnd,ID_SAFETY,WM_PAINT,0,0);

  sprintf(str,"Initial Max Safety = %9.5f",f1);
  fprintf(ftxt,"%s¥n",str);
  MessageBox(NULL,str,"Conjugate Gradient",MB_OK);


  /*INITIAL GRADIENT*/
  fprintf(ftxt,"Initial Gradient¥n");
  //for(q=0;q<nchord;q++)
  for(q=4;q<nchord;q++)
  {
        for(p=0;p<3;p++)
        {


//                        if(p!=2 || (ninit+(*(ai+q)))->d[p]!=0)(org->nodes+(*(ai+
q)))->d[p]=((ninit+(*(ai+q)))->d[p])+dfact;
//                        if(p!=2 || (ninit+(*(bi+q)))->d[p]!=0)(org->nodes+(*(bi+
q)))->d[p]=((ninit+(*(bi+q)))->d[p])+dfact;
//                        if(p!=2 || (ninit+(*(ci+q)))->d[p]!=0)(org->nodes+(*(ci+
q)))->d[p]=((ninit+(*(ci+q)))->d[p])-dfact;

                        if(p!=2 || ((ninit+(*(ai+q)))->d[p]!=0 && (ninit+(*(ai+
q)))->d[p]!=2.00)) (org->nodes+(*(ai+q)))->d[p]+=dfact;
                        if(p!=2 || ((ninit+(*(bi+q)))->d[p]!=0 && (ninit+(*(bi+
q)))->d[p]!=2.00)) (org->nodes+(*(bi+q)))->d[p]+=dfact;
                        if(p!=2 || ((ninit+(*(ci+q)))->d[p]!=0 && (ninit+(*(ci+
q)))->d[p]!=2.00)) (org->nodes+(*(ci+q)))->d[p]-=dfact;
```

```
                        PinchMoveChord(org,ninit,ai,bi,ci,di,p,q,dfact,rmax);
                        MoveBrace(org,ai,bi,di,rmax,nchord);

                        /*for(j=0;j<m;j++)
                        {

                                xi2=(org->nodes+i)->d[0];
                                yi2=(org->nodes+i)->d[1];
                                zi2=(org->nodes+i)->d[2];
                                xj2=(org->nodes+j)->d[0];
                                yj2=(org->nodes+j)->d[1];
                                zj2=(org->nodes+j)->d[2];

                                d2=sqrt((xi2-xj2)*(xi2-xj2)+(yi2-yj2)*(yi2-yj2)+
(zi2-zj2)*(zi2-zj2));
                                delta=d2/d1;

                                if(delta<0.3 && j!=i && (p!=2||(ninit+j)->d[p]!=
0))
                                {
                                        (org->nodes+j)->d[p]=((ninit+j)->d[p])+d
elta*dfact;
                                }
                        }*/

                        /*MAXIMUM SAFETY*/
                        arclmautomatic(org);
                        fout=fopen((wdraw.childs+1)->otpfile,"r");
                        readsrcanrate(fout,&arc,NULL);
                        fclose(fout);
                        aelem=arc.nelem;
                        df=0.0;
                        for(ii=0;ii<aelem;ii++)
                        {
                          for(jj=0;jj<4;jj++)
                          {
                                if(df<(arc.elems+ii)->srate[jj]) df=(arc.elems+i
i)->srate[jj];
                          }
                        }
                        *(*(fgrad1+q)+p)=gamma*(f1-df)/dfact; /*NEGATIVE GRADIENT
*/
                        *(*(u1+q)+p)=*(*(fgrad1+q)+p);

                        fprintf(ftxt,"Node %d Dim %d Gradient=%9.5f¥n",(org->node
s+i)->code,p,(f1-df)/dfact);

                        for(j=0;j<nnode;j++) (org->nodes+j)->d[p]=((ninit+j)->d
[p]);//全ての点もとに戻す
```

```
//sprintf(str,"Step initial MovePt=%d ¥n p=%d MaxSafety=%.5f¥n",mpt[q],p,df);
//MessageBox(NULL,str,"Conjugate Gradient",MB_OK);


        }
  }
  fprintf(ftxt,"¥n");

  /*
  fprintf(stderr,"INITIAL CONDITION¥n");
  fprintf(stderr,"x    = %8.3f %8.3f¥n",*(x1+0),*(x1+1));
  fprintf(stderr,"f(x)= %8.3f¥n",f1);
  fprintf(stderr,"grad f(x)= %8.3f %8.3f¥n",*(fgrad1+0),*(fgrad1+1));
  fprintf(stderr,"{u}       = %8.3f %8.3f¥n",*(u1+0),*(u1+1));
  fprintf(fout,"STEP 0 f %8.3f {x1,x2} %8.3f %8.3f¥n",f1,*(x1+0),*(x1+1));
  gets(non);
  */

  k=0;
  while(1)
  {
        k++;
        sprintf(str,"%d",k);
        SetDlgItemText((wmenu.childs+2)->hwnd,ID_LAPS,str);

        //for(q=0;q<nchord;q++)
        for(q=4;q<nchord;q++)
        {
                for(p=0;p<3;p++)//XZに限定
                {


//                        if(p!=2 || (ninit+(*(ai+q)))->d[p]!=0 || (ninit+(*(ai+
q)))->d[p]!=2.00) (org->nodes+(*(ai+q)))->d[p]=((ninit+(*(ai+q)))->d[p])+(*(*(u1+
q)+p));
//                        if(p!=2 || (ninit+(*(bi+q)))->d[p]!=0) (org->nodes+(*(bi+
q)))->d[p]=((ninit+(*(bi+q)))->d[p])+(*(*(u1+q)+p));
//                        if(p!=2 || (ninit+(*(ci+q)))->d[p]!=0) (org->nodes+(*(ci+
q)))->d[p]=((ninit+(*(ci+q)))->d[p])-(*(*(u1+q)+p));

                        if(p!=2 || ((ninit+(*(ai+q)))->d[p]!=0 && (ninit+(*(ai+
q)))->d[p]!=2.00)) (org->nodes+(*(ai+q)))->d[p]+=(*(*(u1+q)+p));
                        if(p!=2 || ((ninit+(*(bi+q)))->d[p]!=0 && (ninit+(*(bi+
q)))->d[p]!=2.00)) (org->nodes+(*(bi+q)))->d[p]+=(*(*(u1+q)+p));
                        if(p!=2 || ((ninit+(*(ci+q)))->d[p]!=0 && (ninit+(*(ci+
q)))->d[p]!=2.00)) (org->nodes+(*(ci+q)))->d[p]-=(*(*(u1+q)+p));

                        PinchMoveChord(org,ninit,ai,bi,ci,di,p,q,(*(*(u1+q)+p)),r
max);
                        MoveBrace(org,ai,bi,di,rmax,nchord);
```

```
                }
        }

        /*MAXIMUM SAFETY*/
        arclmautomatic(org);
        fout=fopen((wdraw.childs+1)->otpfile,"r");
        readsrcanrate(fout,&arc,NULL);
        fclose(fout);
        aelem=arc.nelem;
        fa=0.0;
        for(ii=0;ii<aelem;ii++)
        {
          for(jj=0;jj<4;jj++)
          {
                if(fa<(arc.elems+ii)->srate[jj]) fa=(arc.elems+ii)->srate[jj];
          }
        }


        fprintf(ftxt,"Step %d Max Safety = %9.5f¥n",k,fa);

        /*
        sprintf(str,"Max Safety = %.5f",fa);
        MessageBox(NULL,str,"Conjugate Gradient",MB_OK);
        */

        /*INITIAL NODE*/
        for(i=0;i<nnode;i++) *(ninit+i)=*(org->nodes+i);

        /*GRADIENT*/
        fprintf(ftxt,"Step %d Gradient¥n",k);
        //for(q=0;q<nchord;q++)
        for(q=4;q<nchord;q++)
        {
                for(p=0;p<3;p++)//XYに限定
                {
//                              if(p!=2 || (ninit+(*(ai+q)))->d[p]!=0 || (ninit+
(*(ai+q)))->d[p]!=2.00)(org->nodes+(*(ai+q)))->d[p]=((ninit+(*(ai+q)))->d[p])+dfac
t;
//                              if(p!=2 || (ninit+(*(bi+q)))->d[p]!=0 || (ninit+
(*(bi+q)))->d[p]!=2.00)(org->nodes+(*(bi+q)))->d[p]=((ninit+(*(bi+q)))->d[p])+dfac
t;
//                              if(p!=2 || (ninit+(*(ci+q)))->d[p]!=0)(org->nodes
+(*(ci+q)))->d[p]=((ninit+(*(ci+q)))->d[p])-dfact;

                                if(p!=2 || ((ninit+(*(ai+q)))->d[p]!=0 && (ninit+
(*(ai+q)))->d[p]!=2.00)) (org->nodes+(*(ai+q)))->d[p]+=dfact;
```

```
                                    if(p!=2 || ((ninit+(*(bi+q)))->d[p]!=0 && (ninit+
(*(bi+q)))->d[p]!=2.00)) (org->nodes+(*(bi+q)))->d[p]+=dfact;
                                    if(p!=2 || ((ninit+(*(ci+q)))->d[p]!=0 && (ninit+
(*(ci+q)))->d[p]!=2.00)) (org->nodes+(*(ci+q)))->d[p]-=dfact;


                                    PinchMoveChord(org,ninit,ai,bi,ci,di,p,q,dfact,rm
ax);
                                    MoveBrace(org,ai,bi,di,rmax,nchord);


                                    /*MAXIMUM SAFETY*/
                                    arclmautomatic(org);
                                    fout=fopen((wdraw.childs+1)->otpfile,"r");
                                    readsrcanrate(fout,&arc,NULL);
                                    fclose(fout);
                                    aelem=arc.nelem;
                                    df=0.0;
                                    for(ii=0;ii<aelem;ii++)
                                    {
                                             for(jj=0;jj<4;jj++)
                                             {
                                               if(df<(arc.elems+ii)->srate[jj]) df=(a
rc.elems+ii)->srate[jj];
                                             }
                                    }
//sprintf(str,"Step %d i=%d p=%d Max Safety(df)=%9.5f¥n",k,i,p,df);
//MessageBox(NULL,str,"Conjugate Gradient",MB_OK);

                                    *(*(ua+q)+p)=-gamma*(fa-df)/dfact+(*(*(u1+q)+p));
 /*NEGATIVE GRADIENT*/

//sprintf(str,"Step %d i=%d p=%d ua=%9.5f¥n",k,i,p,*(*(ua+i)+p));
//MessageBox(NULL,str,"Conjugate Gradient",MB_OK);

                                    fprintf(ftxt,"Node %d Dim %d Gradient=%9.5f¥n",(o
rg->nodes+i)->code,p,(fa-df)/dfact);

                                    for(j=0;j<nnode;j++) (org->nodes+j)->d[p]=((ninit
+j)->d[p]);
                                    //(org->nodes+i)->d[p]=(ninit+i)->d[p];


                 }
         }
         fprintf(ftxt,"¥n");

         /*
         fprintf(stderr,"ITERATION %d¥n",k);
         fprintf(stderr," {u}= %8.3f %8.3f¥n",*(u1+0),*(u1+1));
```

```
	fprintf(stderr,"A{u}= %8.3f %8.3f¥n",*(ua+0),*(ua+1));
	*/

	c1=0.0;
	c2=0.0;
	//for(q=0;q<nchord;q++)
	for(q=4;q<nchord;q++)
	{
		for(p=0;p<3;p++)
		{

			c1+=(*(*(fgrad1+q)+p))*(*(*(fgrad1+q)+p));
			//c2+=((*(*(u1+i)+p))*(*(*(ua+i)+p)));
			c2+=(*(*(u1+q)+p))*(*(*(ua+q)+p));

		}
	}
	alpha=c1/c2;
	//alpha=c2/c1;

	/*
	fprintf(stderr,"Alpha= %8.3f = %8.3f / %8.3f¥n",alpha,c1,c2);
	*/

	vsize=0.0;
	//for(q=0;q<nchord;q++)
	for(q=4;q<nchord;q++)
	{
		for(p=0;p<3;p++)//XYに限定
		{


//(org->nodes+i)->d[p]=((ninit+i)->d[p])+alpha*(*
(*(u1+i)+p));
//if(p!=2 || (ninit+(*(ai+q)))->d[p]!=0 || (ninit+
(*(ai+q)))->d[p]!=2.00)(org->nodes+(*(ai+q)))->d[p]=((ninit+(*(ai+q)))->d[p])+alph
a*(*(*(u1+q)+p));
//if(p!=2 || (ninit+(*(bi+q)))->d[p]!=0 || (ninit+
(*(bi+q)))->d[p]!=2.00)(org->nodes+(*(bi+q)))->d[p]=((ninit+(*(bi+q)))->d[p])+alph
a*(*(*(u1+q)+p));
//if(p!=2 || (ninit+(*(ci+q)))->d[p]!=0)(org->nodes
+(*(ci+q)))->d[p]=((ninit+(*(ci+q)))->d[p])-alpha*(*(*(u1+q)+p));

if(p!=2 || ((ninit+(*(ai+q)))->d[p]!=0 && (ninit+
(*(ai+q)))->d[p]!=2.00)) (org->nodes+(*(ai+q)))->d[p]+=alpha*(*(*(u1+q)+p));
if(p!=2 || ((ninit+(*(bi+q)))->d[p]!=0 && (ninit+
(*(bi+q)))->d[p]!=2.00)) (org->nodes+(*(bi+q)))->d[p]+=alpha*(*(*(u1+q)+p));
if(p!=2 || ((ninit+(*(ci+q)))->d[p]!=0 && (ninit+
(*(ci+q)))->d[p]!=2.00)) (org->nodes+(*(ci+q)))->d[p]-=alpha*(*(*(u1+q)+p));
```

```
                                        PinchMoveChord(org,ninit,ai,bi,ci,di,p,q,alpha*(*
(*(u1+q)+p)),rmax);
                                        MoveBrace(org,ai,bi,di,rmax,nchord);


                                        *(*(fgrad2+q)+p)=(*(*(fgrad1+q)+p))-alpha*(*(*(ua
+q)+p));
                                        vsize+=(*(*(fgrad2+q)+p))*(*(*(fgrad2+q)+p));



                }
        }

        /*MAXIMUM SAFETY*/
        arclmautomatic(org);
        fout=fopen((wdraw.childs+1)->otpfile,"r");
        readsrcanrate(fout,&arc,NULL);
        fclose(fout);
        aelem=arc.nelem;
        f2=0.0;
        for(ii=0;ii<aelem;ii++)
        {
          for(jj=0;jj<4;jj++)
          {
                if(f2<(arc.elems+ii)->srate[jj]) f2=(arc.elems+ii)->srate[jj];
          }
        }

//sprintf(str,"Step %d Max Safety(f2) = %9.5f¥n",k,f2);
//MessageBox(NULL,str,"Conjugate Gradient",MB_OK);

        sprintf(str,"%d",k);
        SetDlgItemText((wmenu.childs+2)->hwnd,ID_LAPS,str);
        SendDlgItemMessage((wmenu.childs+2)->hwnd,ID_LAPS,WM_PAINT,0,0);
        sprintf(str,"%.5f",f2);
        SetDlgItemText((wmenu.childs+2)->hwnd,ID_SAFETY,str);
        SendDlgItemMessage((wmenu.childs+2)->hwnd,ID_SAFETY,WM_PAINT,0,0);

        sprintf(str,"Step=%d Max Safety=%.5f Vector Size=%.5f",k,f2,vsize);
        fprintf(ftxt,"%s¥n¥n",str);
        /*MessageBox(NULL,str,"Conjugate Gradient",MB_OK);*/

        /*
        fprintf(stderr,"x    = %8.3f %8.3f¥n",*(x2+0),*(x2+1));
        fprintf(stderr,"f(x) = %8.3f¥n",f2);
        fprintf(stderr,"grad f(x)   = %8.3f %8.3f¥n",*(fgrad2+0),*(fgrad2+1));
        fprintf(stderr,"VECTOR SIZE = %9.5f¥n",vsize);
```

```
        fprintf(fout,"STEP %d f %8.3f {x1,x2} %8.3f %8.3f¥n",k,f2,*(x2+0),*(x2+
1));
        gets(non);
        */

        if(vsize<eps || f2<=ftarget)
        {
          sprintf(str,"COMPLETED : TARGET f(x)=%.5f¥n",f2);
          MessageBox(NULL,str,"Conjugate Gradient",MB_OK);
          return;
        }



        c3=0.0;
        //for(q=0;q<nchord;q++)
        for(q=4;q<nchord;q++)
        {
                for(p=0;p<3;p++)
                {

                                c3+=(*(*(fgrad2+q)+p))*(*(*(fgrad2+q)+p));

                }
        }
        beta=c3/c1;

        /*繰り返しのため引き継ぎ*/
        /*INITIAL NODE*/
        for(i=0;i<nnode;i++) *(ninit+i)=*(org->nodes+i);

        //for(q=0;q<nchord;q++)
        for(q=4;q<nchord;q++)
        {
                for(p=0;p<3;p++)
                {

                                *(*(u2+q)+p)=(*(*(fgrad2+q)+p))+beta*(*(*(u1+q)+
p));
                                *(*(u1+q)+p)=*(*(u2+q)+p);
                                *(*(fgrad1+q)+p)=*(*(fgrad2+q)+p);

                }
        }



  }

  fclose(ftxt);
```

```
  return;

}/*CGcurve*/
```

3-2. PinchMoveChord

```
void PinchMoveChord(struct organ *org,struct onode *ninit,int *ai,int *bi,int *ci,
int **di,int p,int q,double distance,int *rmax)
{
        double xa=(org->nodes+(*(ai+q)))->d[0];
        double ya=(org->nodes+(*(ai+q)))->d[1];
        double za=(org->nodes+(*(ai+q)))->d[2];

        double xb=(org->nodes+(*(bi+q)))->d[0];
        double yb=(org->nodes+(*(bi+q)))->d[1];
        double zb=(org->nodes+(*(bi+q)))->d[2];

        double xc=(org->nodes+(*(ci+q)))->d[0];
        double yc=(org->nodes+(*(ci+q)))->d[1];
        double zc=(org->nodes+(*(ci+q)))->d[2];

        int r;
        for(r=0;r<(*(rmax+q));r++)
        {
                double xd=(org->nodes+(*(*(di+q)+r)))->d[0];
                double yd=(org->nodes+(*(*(di+q)+r)))->d[1];
                double zd=(org->nodes+(*(*(di+q)+r)))->d[2];

                double cd=sqrt((xc-xd)*(xc-xd)+(yc-yd)*(yc-yd)+(zc-zd)*(zc-zd));
                if(cd!=0)
                {

                        double ad=sqrt((xa-xd)*(xa-xd)+(ya-yd)*(ya-yd)+(za-zd)*(z
a-zd));
                        double bd=sqrt((xb-xd)*(xb-xd)+(yb-yd)*(yb-yd)+(zb-zd)*(z
b-zd));
                        double ac=sqrt((xa-xc)*(xa-xc)+(ya-yc)*(ya-yc)+(za-zc)*(z
a-zc));
                        double bc=sqrt((xb-xc)*(xb-xc)+(yb-yc)*(yb-yc)+(zb-zc)*(z
b-zc));

                        if(ad<=bd)
                        {
                                double delta=((ad+cd)/2-ad)/(2/(ad+cd))*(ad/ac);
//                              (org->nodes+(*(*(di+q)+r)))->d[p]=((ninit+(*(*(di
+q)+r)))->d[p])+delta*distance;
                                (org->nodes+(*(*(di+q)+r)))->d[p]+=delta*distanc
e;
                        }
                        else
                        {
                                double delta=((bd+cd)/2-bd)/(2/(bd+cd))*(bd/bc);
```

```
//                                  (org->nodes+(*(*(di+q)+r)))->d[p]=((ninit+(*(*(di
+q)+r)))->d[p])+delta*distance;
                                    (org->nodes+(*(*(di+q)+r)))->d[p]+=delta*distanc
e;
                            }
                    }

            }
}
```

3-3. MoveBrace

```
void MoveBrace(struct organ *org,int *ai,int *bi,int **di,int *rmax,int nchord)
{


 FILE *fout,*ftxt;
 char non[10],str[256];

 int i,j,k,q,r;
 //double x1,x2,x3,x4,y1,y2,y3,y4,z1,z2,z3,z4;
 double x[]={0,0,0,0};
 double y[]={0,0,0,0};
 double z[]={0,0,0,0};


 long int nnode = org->nnode;
 long int nelem = org->nelem;

 int *nflag;
 nflag = (int *)malloc(nnode*sizeof(int));
 int *tonari;
 tonari = (int *)malloc(4*sizeof(int));
 double *l;
 l=(double *)malloc(6*sizeof(double));


 double ksi, eta, delta;
 double ramda, mu;



 for(i=0;i<nnode;i++)
 {
        for(k=0;k<nnode;k++) *(nflag +k) = 0;
        *(nflag +i) = 1;

        int search=0;

        /*brace交点以外の点ではないか確認*/
        for(q=0;q<nchord;q++){

                if( *(ai+q)==i || *(bi+q)==i) search++;

                for(r=0;r<(*(rmax+q));r++){
                        if(*(*(di+q)+r)==i)
                        {
                                search++;
                        }
```

```
                }
        }

        if(search==0)/*brace交点だったら*/
        {
         r=0;
         for(j=0;j<nelem;j++)
         {
                int loff=(*((org->elems+j)->nods+0))->loff;
                int moff=(*((org->elems+j)->nods+1))->loff;


                if (*(nflag + loff) == 1 && *(nflag + moff) == 0) {
                        *(tonari + r) = moff;
                        r++;
                }
                if (*(nflag + moff) == 1 && *(nflag + loff) == 0) {
                        *(tonari +r) = loff;
                        r++;
                }
         }


         if(r==4)
         {
                /*周囲４点からなる四角形の対角線の交点*/
                for(k=0;k<4;k++)
                {
                        x[k] = (org->nodes+(*(tonari+k)))->d[0];
                        y[k] = (org->nodes+(*(tonari+k)))->d[1];
                        z[k] = (org->nodes+(*(tonari+k)))->d[2];
                }

                *(l+0)=pow(x[2]-x[1],2.0)+pow(y[2]-y[1],2.0)+pow(z[2]-z[1],2.0);
                *(l+1)=pow(x[0]-x[3],2.0)+pow(y[0]-y[3],2.0)+pow(z[0]-z[3],2.0);
                *(l+2)=pow(x[3]-x[2],2.0)+pow(y[3]-y[2],2.0)+pow(z[3]-z[2],2.0);
                *(l+3)=pow(x[1]-x[0],2.0)+pow(y[1]-y[0],2.0)+pow(z[1]-z[0],2.0);
                *(l+4)=pow(x[1]-x[3],2.0)+pow(y[1]-y[3],2.0)+pow(z[1]-z[3],2.0);
                *(l+5)=pow(x[2]-x[0],2.0)+pow(y[2]-y[0],2.0)+pow(z[2]-z[0],2.0);


                double a=0.0;
                for(k=0;k<6;k++) if(a<=*(l+k)) a=*(l+k);

                for(k=0;k<6;k++)
                {
                if(*(l+k)==a)
                 {

                  int kaisu = 0;
```

```
                while(kaisu<1)
                {
                        switch(k)
                        {
                         case 0:
                         case 1:
                                delta = ( x[2]-x[1] )*( y[0]-y[3] ) -
                                        ( y[2]-y[1] )*( x[0]-x[3] );

                                if(fabs(delta)>0.0)
                                {
                                        ksi = ( y[0]-y[3] )*( x[0]-x[1] ) -
                                                ( x[0]-x[3] )*( y[0]-y[1] );

                                        eta = ( x[2]-x[1] )*( y[0]-y[1] ) -
                                                ( y[2]-y[1] )*( x[0]-x[1] );

                                }
                                else
                                {
                                        delta = ( x[2]-x[1] )*( z[0]-z[3] ) -
                                                ( z[2]-z[1] )*( x[0]-x[3] );

                                        if(fabs(delta)>0.0)
                                        {
                                                ksi = ( z[0]-z[3] )*( x[0]-x[1]
) -
                                                        ( x[0]-x[3] )*( z[0]-z
[1] );

                                                eta = ( x[2]-x[1] )*( z[0]-z[1]
) -
                                                        ( z[2]-z[1] )*( x[0]-x
[1] );
                                        }
                                        else
                                        {
                                                delta = ( y[2]-y[1] )*( z[0]-z
[3] ) -
                                                ( z[2]-z[1] )*( y[0]-y[3] );

                                                ksi = ( z[0]-z[3] )*( y[0]-y[1]
) -
                                                        ( y[0]-y[3] )*( z[0]-z
[1] );

                                                eta = ( y[2]-y[1] )*( z[0]-z[1]
) -
                                                        ( z[2]-z[1] )*( y[0]-y
[1] );
```

```
                                        }

                                }
                                ramda = ksi / delta;
                                mu    = eta / delta;

                                if ( ( ramda >= 0 && ramda <= 1 ) && ( mu >= 0 &&
 mu <= 1 ) )
                                {
                                        (org->nodes+i)->d[0] = x[1] + ramda*( x
[2]-x[1] );
                                        (org->nodes+i)->d[1] = y[1] + ramda*( y
[2]-y[1] );
                                        (org->nodes+i)->d[2] = z[1] + ramda*( z
[2]-z[1] );

                                }

                        break;

                        case 2:
                        case 3:
                                delta = ( x[0]-x[1] )*( y[2]-y[3] ) -
                                        ( y[0]-y[1] )*( x[2]-x[3] );

                                if(fabs(delta)>0.0)
                                {
                                        ksi = ( y[2]-y[3] )*( x[2]-x[1] ) -
                                                ( x[2]-x[3] )*( y[2]-y[1] );

                                        eta = ( x[0]-x[1] )*( y[2]-y[1] ) -
                                                ( y[0]-y[1] )*( x[2]-x[1] );
                                }
                                else
                                {
                                        delta = ( x[0]-x[1] )*( z[2]-z[3] ) -
                                                ( z[0]-z[1] )*( x[2]-x[3] );

                                        if(fabs(delta)>0.0)
                                        {
                                                ksi = ( z[2]-z[3] )*( x[2]-x[1]
 ) -
                                                        ( x[2]-x[3] )*( z[2]-z
[1] );

                                                eta = ( x[0]-x[1] )*( z[2]-z[1]
 ) -
                                                        ( z[0]-z[1] )*( x[2]-x
[1] );
                                        }
```

```
                                        else
                                        {
                                                delta = ( y[0]-y[1] )*( z[2]-z
[3] ) -
                                                        ( z[0]-z[1] )*( y[2]-y
[3] );

                                                ksi = ( z[2]-z[3] )*( y[2]-y[1]
) -
                                                        ( y[2]-y[3] )*( z[2]-z
[1] );

                                                eta = ( y[0]-y[1] )*( z[2]-z[1]
) -
                                                        ( z[0]-z[1] )*( y[2]-y
[1] );
                                        }
                                }

                                ramda = ksi / delta;
                                mu    = eta / delta;

                                if ( ( ramda >= 0 && ramda <= 1 ) && ( mu >= 0 &&
 mu <= 1 ) )
                                {
                                        (org->nodes+i)->d[0] = x[1] + ramda*( x
[0]-x[1] );
                                        (org->nodes+i)->d[1] = y[1] + ramda*( y
[0]-y[1] );
                                        (org->nodes+i)->d[2] = z[1] + ramda*( z
[0]-z[1] );

                                }
                        break;

                        case 4:
                        case 5:
                                delta = ( x[3]-x[1] )*( y[0]-y[2] ) -
                                        ( y[3]-y[1] )*( x[0]-x[2] );

                                if(fabs(delta)>0.0)
                                {
                                        ksi = ( y[0]-y[2] )*( x[0]-x[1] ) -
                                                ( x[0]-x[2] )*( y[0]-y[1] );

                                        eta = ( x[3]-x[1] )*( y[0]-y[1] ) -
                                                ( y[3]-y[1] )*( x[0]-x[1] );

                                }
                                else
```

```
                                {
                                        delta = ( x[3]-x[1] )*( z[0]-z[2] ) -
                                                ( z[3]-z[1] )*( x[0]-x[2] );

                                        if(fabs(delta)>0.0)
                                        {

                                                ksi = ( z[0]-z[2] )*( x[0]-x[1]
) -
                                                        ( x[0]-x[2] )*( z[0]-z
[1] );

                                                eta = ( x[3]-x[1] )*( z[0]-z[1]
) -
                                                        ( z[3]-z[1] )*( x[0]-x
[1] );
                                        }
                                        else
                                        {
                                                delta = ( y[3]-y[1] )*( z[0]-z
[2] ) -
                                                        ( z[3]-z[1] )*( y[0]-y
[2] );

                                                ksi = ( z[0]-z[2] )*( y[0]-y[1]
) -
                                                        ( y[0]-y[2] )*( z[0]-z
[1] );

                                                eta = ( y[3]-y[1] )*( z[0]-z[1]
) -
                                                        ( z[3]-z[1] )*( y[0]-y
[1] );
                                        }
                                }

                                ramda = ksi / delta;
                                mu    = eta / delta;

                                if ( ( ramda >= 0 && ramda <= 1 ) && ( mu >= 0 &&
 mu <= 1 ) )
                                {
                                        (org->nodes+i)->d[0] = x[1] + ramda*( x
[3]-x[1] );
                                        (org->nodes+i)->d[1] = y[1] + ramda*( y
[3]-y[1] );
                                        (org->nodes+i)->d[2] = z[1] + ramda*( z
[3]-z[1] );

                                }
```

```
                        break;

                      }
                      kaisu++;


                  }

                 }
                }
          }
         }
 }
}
```