有限の計算精度のもとでの幾何的アルゴリズムの研究

今井敏行

①

有限の計算精度のもとでの幾何的アルゴリズムの研究

今井 敏行

目次

第 1 章

はじめに

計算幾何学は，幾何的な問題に対するアルゴリズムの開発や効率化と，その限界を研究するアルゴリズム論の一分野である．本論文では，広く一般に，幾何的な問題に対するアルゴリズムを幾何的アルゴリズムとよぶことにする．計算幾何学がアルゴリズム論から，研究対象を幾何的アルゴリズムに限定して意識的に抽出されたのは，1978 年の M. I. Shamos の博士論文からであるといわれる．その後，計算幾何学の研究は急速に進展し，数多くの効率的な幾何的アルゴリズムが開発された [2, 4, 13, 61, 64]．幾何的な問題の解析も進み，平面上の点集合に関する凸包や Voronoi 図の構成などの基本的な問題を始め，多くの幾何的な問題でアルゴリズムの計算量，記憶量の限界が明らかになり [6, 32]，また，そこまで効率化が達成された幾何的アルゴリズムも少なくない [13, 42, 86]．

人の日常は絶えず物体に関わっていて，物体は視覚的に図形として認識されているので，人は，直接的に，幾何的な問題に囲まれているといえる．また，一般に，d 個の属性を持ったデータはどんなデータでも，自然に，d 次元空間の点としてみることができ，情報の処理は d 次元図形の加工という幾何的な問題として捕えることもできる．実際，幾何的な問題は視覚で捕えやすいため，一般的な問題を幾何的な問題に変形してみる場合は多い．このような意味で，人は，間接的にも，幾何的な問題に囲まれている．そのため，幾何的アルゴリズムに対する実用上の需要は多い [36, 38, 78]．

具体的に，後の 5 章で構成法を述べる各種 Voronoi 図を例にする．平面上に有限個の点を与えたとき，それらの点による平面の勢力圏分割を表す図を Voronoi 図という．(正確な定義は 5 章で与える．) 計算幾何学の理論において，当初から，Voronoi 図は中心的な研究対象のひとつであった [5, 36, 58]．初めに与える図形を，線分や多角形など，点以外の図形にも拡張したり，初めに与える図形が分布する空間を 3 次元以上に拡張した各種 Voronoi 図の構成もまた，中心的課題になっている [5, 36, 38, 45, 47]．それと同時に，実用上も，Voronoi 図は画像処理，数値解析から物理学，生態学や都市工学に至るまで，幅広い需要をもつ [23, 38, 57, 63]．そのため，各種 Voronoi 図の構成は理論的な興味だけでなく，実用面からの要求も強く，効率的なアルゴリズムの実装が求められている．

現実的な問題に対応するため，Voronoi 図の構成に限らず，多くの幾何的な問題に対して，効率的な幾何的アルゴリズムの開発とその実装が求められている．しかしながら，理論的には効率的なアルゴリズムが，実装すると実用に堪えないものであることが少なくない．実用程度の入力データ量では高速性を発揮できない場合は論外としても，実用上も効率的と思われる幾何的アルゴリズムでも，素直にプログラムにして実装すると，データ破壊，異常終了，無限ループなどを引き起こすのである．

このような状況を反省し，近年になって，計算幾何学では，この原因と対策に関する研究が積極的に進められるようになった [14, 15, 17, 18, 19, 37, 48, 59, 71]. 幾何的アルゴリズムが，実装すると，データ破壊，異常終了，無限ループなどを引き起こし実用に堪えないものになることをアルゴリズムに**数値的安定性**が欠けているということにする. 後の章で述べるように，数値的安定性を欠く幾何的アルゴリズムが設計される原因は，どのような計算も誤差なく実行できるという無限精度の計算と，都合の悪い入力を例外として排除する非退化入力が，アルゴリズム設計時の前提として採用されていることである．前者の無限精度計算の採用は，現在の実際の計算機には不可避ともいえる数値計算誤差の存在を否定し，後者の非退化入力の採用は現実の数値計算誤差の下では，実用上の入力の需要を満たせない. 実用上の利用が可能な幾何的アルゴリズムを得るためには，計算誤差と退化入力の存在をアルゴリズム構築の初めから意識する必要があろう.

本論文は，有限精度の計算を前提に，退化入力の場合でも正常終了して出力が得られるような数値的に安定な幾何的アルゴリズムの構築法について論じる. 計算精度の有限性と退化入力の可能性を前提とすると，数値的に安定な幾何的アルゴリズムの設計法を 2 種類考えることができる. 第 1 の方法は誤差の問題と退化の問題を分離して扱う方法で，第 2 の方法は誤差と退化を同時に扱う方法である．もう少し詳しく述べると，第 1 の方法は，退化の有無を，計算量をかけても正確に判定して，統一的な対処法を行なうという方法で，**代数的退化対処法**と名付ける. 第 2 の方法は，計算機の浮動小数計算を基本にし，退化の有無は計算誤差のため判定できないとして，計算精度に応じた出力と正常終了を保証するという方法で**位相優先法**とよばれている. 本論文ではこれらの両方について論じる.

本論文は，まず幾何的アルゴリズムの退化に関する定義と一般論を展開する．次に，代数的退化対処法に関して，これまでに提案されてきた手法を示し，複数の視点から退化対処法としての解釈を行ない，代数的退化対処法に分類されるもう一つの方法を新規に提案する. 次に，代数的退化対処法で問題になる計算量の増大に関する考察を行ない，剰余演算を利用した方法を提案する. 次に，位相優先法の解説と，具体例として，新規に開発した各種 Voronoi 図の構成アルゴリズムとその実装を詳述する．最後に，論文全体を総合的にまとめ，代数的退化対処法と位相優先法の得失を論じる.

第 2 章

幾何的アルゴリズムと退化

2.1 アルゴリズム論と計算幾何学

計算幾何学は幾何的な図形を対象としたアルゴリズムの開発や効率化とその限界を研究するアルゴリズム論の一分野である．実際に計算幾何学の理論では，これまでに数多くの効率的アルゴリズムが開発されている．例えば線分の交差発見問題を考える．これは，端点座標の対として与えられた n 個の線分の少なくとも一組の線分対に交差があれば真を，交差がなければ偽を返せという問題ある．ただちに，全ての線分の対に関して交差しているかどうか判定するアルゴリズムを考えることができる．これを素朴なアルゴリズムとよぶことにする．交差線分対の総数は $O(n^2)$ で，一組の線分対の交差判定は定数時間で実行できるから素朴なアルゴリズムの計算量は $O(n^2)$ である．これに対し，J. L. Bentley と T. Ottmann による，平面走査法とよばれる次の有名なアルゴリズム [3, 7, 65] がある (図 2.1)．ここで，線分の端点 p に対し p を端点とする線分を $s(p)$ とする．また y 軸に平行な掃引直線を考え，各端点で掃引直線と交わる線分を y 座標順にソートして保持するために平衡 2 分木 SWEEPLINE を用意する．

アルゴリズム 1 (線分の交差判定)

1. 線分の端点を x 座標でソートし，x 座標の小さいほうから添字を付け $p_1, \cdots, p_{2n}$ とする．

2. SWEEPLINE を空にする．

3. $i \leftarrow 1, \cdots, 2n$ に対し **3.1**, **3.2** を行なう．

3.1. 端点 p_i が線分 $s(p_i)$ の左端点ならば，**3.1.1**, **3.1.2** を行なう．

3.1.1. p_i の x 座標値における各線分の y 座標値をキーとして $s(p_i)$ を SWEEPLINE に挿入する．

3.1.2. SWEEPLINE 上で $s(p_i)$ と隣り合う線分と $s(p_i)$ との 交差判定を行ない，交差したら真を返して終了．

3.2. 端点 p_i が線分 $s(p_i)$ の右端点ならば，**3.2.1**, **3.2.2** を行なう．

3.2.1. $s(p_i)$ を SWEEPLINE から削除する．

3.1.2. SWEEPLINE 上で新規に隣り合う線分対の 交差判定を行ない，交差したら真を返して終了．

4. 偽を返して終了．

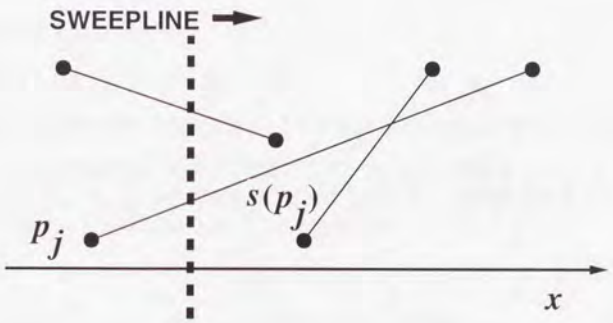


図 2.1: Bentley, Ottmann の線分交差判定

このアルゴリズム 1の計算時間を考える. まず, 1 の端点のソートはヒープソートなどで $O(n \log n)$ で実行できる [1, 65]. 2 と 4 は定数オーダである. 3.1.1 と 3.2.1 は AVL 木などを使えば $O(\log n)$ で実行できる [1, 65]. 3.1.2 と 3.2.2 は定数オーダである. 従って, 全体の計算時間は $O(n \log n + 1 + n(\log n + 1) + 1)$ で, 結局 $O(n \log n)$ となる. これは素朴なアルゴリズムの計算時間 $O(n^2)$ より少ない.

この例でまず注意すべき点はアルゴリズムが高速化された理由である. ソートや平衡 2 分木などの効率のよい基本アルゴリズムをうまく利用するアルゴリズム論の一般論の枠内の手法のほかに, この例では, 掃引直線上で隣り合わない線分対は交わることがないことを利用して線分対の交差判定をする回数を減らしている. このように計算幾何学では, 対象図形のもつ性質を利用することによりアルゴリズム論の一般論の技術だけでない効率化を実現している.

次に注意すべき点はアルゴリズムで用いられる計算である. この例でわかるように, 計算機にアルゴリズムを実装する場合を考えると, 計算は, ソート時のデータの操作や平衡 2 分木の操作のような整数計算を基本におく部分と, 線分対の交差判定や端点座標の比較など浮動小数を基本におく部分に分けられる. この点に関しては幾何的アルゴリズムの構造に関する考察 (2.3節) の中で再度述べることにする.

2.2 アルゴリズムの暴走と退化

このように効率のよい幾何的アルゴリズムが計算幾何学の研究の中から生みだされてきたが, これらの理論上の幾何的アルゴリズムを応用するため実際に計算機にプログラムにし, 現実のデータを入力して実行すると, 役に立たないことが少なくない. 例えば, 明らかに出力が不正確であったり, 無限ループに陥ったり, 入力データを破壊したり, 異常終了をしたりする.

例として, 実数直線 $\mathbf{R}$ 上の 2 個の区間 $I = [x_1, x_2]$, $J = [x_3, x_4]$ の関係を考える. 区間 I, J の端点が一致する場合を除けば, 位置関係は,

1. I は J を含む: $I \supset J$,

2. I は J に含まれる: $I \subset J$,

3. I, J は離れている: $I \cap J = \emptyset$,

4. I, J は互いに他を含まないが交わる: $I \cap J \neq \emptyset$ かつ $I - J \neq \emptyset$ かつ $J - I \neq \emptyset$

の 4 個のタイプに分類される (図 2.2). 次のアルゴリズムは入力変数を x_1, x_2, x_3, x_4 とし, 区間 $I = [x_1, x_2]$, $J = [x_3, x_4]$ の位置関係のタイプを出力する. このようなアルゴリズムは, 多くの幾何的アルゴリズムの部品として使われる基礎的アルゴリズムであろう.

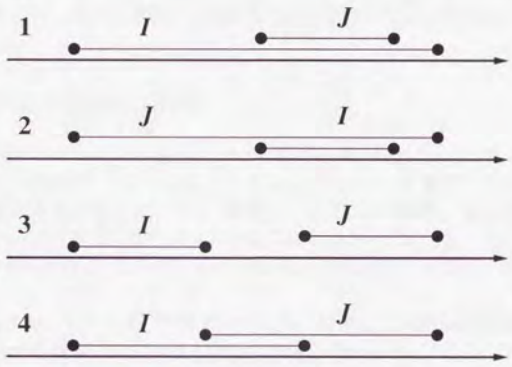


図 2.2: 区間の位置関係

アルゴリズム **2** (**1** 次元区間の位置関係)
1. $x_3 - x_1 > 0$ ならば **1.1** を行ない, そうでないなら **1.2** を行なう.
  1.1. $x_3 - x_2 > 0$ ならば 3 を出力して終了し, そうでないなら **1.1.1** を行なう.
    1.1.1. $x_2 - x_4 > 0$ ならば 1 を出力して終了し, そうでないなら 4 を出力して終了する.
  1.2. $x_1 - x_4 > 0$ ならば 3 を出力して終了し, そうでないなら **1.2.1** を行なう.
    1.2.1. $x_2 - x_4 > 0$ ならば 4 を出力して終了し, そうでないなら 2 を出力して終了する.

このアルゴリズム 1は, 区間 I, J が端点を共有する場合も, 共有しない場合の "極限" として位置関係のタイプを出力するように思える. そこで 3 区間 $I = [0,1], J = [0,2], K = [0,3]$ として, (I, J), (J, K), (K, I) の対をこのアルゴリズムに適用すると, それぞれ出力 2,2,4 を得る. これは $I \subset J$ かつ $J \subset K$ であるが $I \subset K$ ではないと主張していることになり, 論理的に破綻している. このようなアルゴリズムを部品として複雑な幾何的アルゴリズムを構築しても所要の動作を期待できないであろう. したがって, 所要の動作を保証する

アルゴリズムの構築には“一般的”な場合に正常に動作することを保証するだけでは不十分で，全ての“例外的”な場合も考慮にいれる必要がある．このような例外的な場合を退化とよぶ(より正確な意味づけは2.4節で述べる)．退化を個別に対処すると，上記のアルゴリズムのような単純なものでも x_1, x_2, x_3, x_4 のうち2個が1組だけ等しくなる場合が6通り，2個が2組等しくなる場合が3通り，3個が等しくなる場合が4通り，4個が等しくなる場合が1通りある．全部で14通りである．このような退化に個別に適応したアルゴリズムは，一般的には元のアルゴリズムを少し変更することで得られる．そのため，アルゴリズムの理論的な構築の場面では，前提として，入力が一般の位置にあるとされ，退化時の処理についての記述を省略されることが多い．しかしながら，実際には場合の数が多いので，退化を個別に対応し尽くしたアルゴリズムの記述には，退化に対応した部分だけで，元のアルゴリズムの記述を超える労力が要求される．容易に想像できるように，浮動小数の数値計算誤差でも退化時と同様な問題が生じる．本研究では，この退化と計算誤差に起因する問題とその対処法について考える．

2.3 幾何的アルゴリズムの構造

ここで，幾何的アルゴリズムの構造について考える．幾何的アルゴリズムが扱う幾何的対象は，計量データと位相データからなるとしてよいであろう．**計量データ**とは，座標や長さ，角度のような連続値を取り得るデータで，**位相データ**とは接続関係，隣接関係や向きのような離散値を取るデータである．アルゴリズムはこれらのデータを加工していく．まず入力の計量，位相データから始めて，アルゴリズムの実行に必要な計量，位相データを計算していくわけである．

多くの幾何的アルゴリズムで，計量データに施す計算は加減乗除の四則演算と開平演算である．これらの演算結果の複数の値の大小比較を分岐基準として，アルゴリズム中の分岐を行ない，出力の位相データを得る(図2.3)．

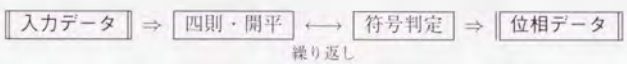


図 2.3: 幾何的アルゴリズムでのデータの加工

容易にわかるように，位相データを得るための演算から除算をなくすことができる．すなわち，$x_1, \cdots, x_n$ を入力変数とし，簡単のため $x = (x_1, \cdots x_n)$ とする．アルゴリズムの分岐手続きを中間変数を消去することで入力変数のみの式に変形し，通分して，2個の式

$$\frac{F_1(x)}{G_1(x)}, \frac{F_2(x)}{G_2(x)} \tag{2.1}$$

の大小の比較に帰着させることができる．これらの差をとり通分して，

$$\frac{F_1(x)}{G_1(x)} - \frac{F_2(x)}{G_2(x)} = \frac{F_1(x)G_2(x) - F_2(x)G_1(x)}{G_1(x)G_2(x)} \tag{2.2}$$

と 0 との大小比較, すなわち, 符号判定に変形すれば, この符号は, 分子と分母の積

$$(F_1(x)G_2(x) - F_2(x)G_1(x))G_1(x)G_2(x) \tag{2.3}$$

の符号と一致する. この積は除算を含まない. これで位相データを求めるためには除算は不要であることがわかった. 次に, 開平演算も不要であることを示す.

定理 1
集合 P を次の 3 条件を満たす最小の集合とする.

1. $\mathbf{R} \subseteq P$,
2. $\forall x_i \in P$,
3. $F(x), G(x) \in P$ ならば $F(x)+G(x), F(x)-G(x), F(x)G(x), \sqrt{F(x)} \in P$.

このとき, 任意の $F(x), G(x) \in P$ に対し, 条件 $F(x) > G(x)$ と同値な条件を, 有限個の $x_1, \cdots, x_n$ の多項式の大小比較を組み合わせて作ることができる.

証明. $H(x) = F(x) - G(x)$ とすると $H(x) \in P$ で, 条件 $F(x) > G(x)$ と条件 $H(x) > 0$ は同等である. 一般に P の元 $H(x)$ は次のような形をしている.

$$H(x) = A_0 + A_1\sqrt{B_1} + \cdots + A_{k-1}\sqrt{B_{k-1}} + A_k\sqrt{B_k}. \tag{2.4}$$

ただし $A_0, \cdots, A_k$ は開平を含まない (すなわち, $x_1, \cdots, x_n$ の多項式). ここで $\sqrt{B_1}, \cdots, \sqrt{B_k}$ の開平のネストの深さの最大値を l とする. また関数 index : $P \to \mathbf{N}_0^2$(ただし $\mathbf{N}_0$ は非負整数全体) を $\mathrm{index}(H(x)) = (l, k)$ で定めることにする. 関数 index による P の像を I とする. $\mathbf{N}_0^2$ 上の辞書式順序 (3.1節, 定義 5参照) を $\leq_{\mathrm{lex}}$ とかくと $(\mathbf{N}_0^2, \leq_{\mathrm{lex}})$ は整列集合になり, これを I に制限しても整列集合になる.

証明は $(l, k) \in I$ に関する (超限) 帰納法で行なう.

$(l, k) = (0, 0)$ のときには, $H(x) = A_0$ には開平が含まれない. したがって命題は成立する.

$(l', k') <_{\mathrm{lex}} (l, k)$ を満たす任意の $(l', k') \in I$ について命題が成立しているとする. $k = 0$ のとき $l = 0$ であるので $k \geq 1$ としてよい.

$$A_0 + A_1\sqrt{B_1} + \cdots + A_{k-1}\sqrt{B_{k-1}} + A_k\sqrt{B_k} > 0 \tag{2.5}$$

$$\Leftrightarrow \quad A_0 + A_1\sqrt{B_1} + \cdots + A_{k-1}\sqrt{B_{k-1}} > -A_k\sqrt{B_k} \tag{2.6}$$

ここで, 左辺を C_{k-1} とおくと, この不等式は次と同等である.

$$\begin{aligned} &(C_{k-1} \geq 0 \text{かつ} -A_k\sqrt{B_k} \geq 0 \text{かつ} C_{k-1}^2 > A_k^2 B_k) \\ \text{または} \quad &(C_{k-1} \geq 0 \text{かつ} -A_k\sqrt{B_k} < 0) \\ \text{または} \quad &(C_{k-1} < 0 \text{かつ} -A_k\sqrt{B_k} < 0 \text{かつ} C_{k-1}^2 < A_k^2 B_k). \end{aligned} \tag{2.7}$$

上の条件のうち, 不等式 $C_{k-1} \geq 0, C_{k-1} < 0$ に関しては, $\mathrm{index}(C_{k-1}) <_{\mathrm{lex}} \mathrm{index}(H(x))$ である. したがって, 帰納法の仮定により, これらの不等式は多項式の大小比較の組合せで表現できる. また, 不等式 $-A_k\sqrt{B_k} \geq 0, -A_k\sqrt{B_k} < 0$ については, それぞれ,

$$(B_k = 0 \text{または} (-A_k \geq 0 \text{かつ} B_k > 0)), (-A_k < 0 \text{かつ} B_k > 0) \tag{2.8}$$

と同等である．このうち，A_k はすでに多項式である．また B_k の開平のネストの深さは $\sqrt{B_k}$ のネストの深さより 1 だけ小さいから，$\mathrm{index}(B_k) <_{\mathrm{lex}} (l,k)$ となる．したがって，帰納法の仮定より，不等式 $B_k \geq 0, B_k > 0$ は多項式の大小比較の組合せで表現できる．

最後に，不等式 $C_{k-1}^2 > A_k^2 B_k, C_{k-1}^2 < A_k^2 B_k$ について扱う．$C_{k-1}^2 - A_k^2 B_k$ は $\sqrt{B_k}$ を含まない．そこで，$C_{k-1}^2 - A_k^2 B_k$ 内に残っている $\sqrt{B_1}, \cdots, \sqrt{B_{k-1}}$ の $k-1$ 個の開平を取り去ることができることを示す．$C_{k-1}^2 > A_k^2 B_k$ について示すが，$C_{k-1}^2 < A_k^2 B_k$ も同様である．

$k = 1$ のときには取り除くべき開平はないから，$k \geq 2$ とする．$\sqrt{B_{k-1}}$ の開平を取り去る操作を示す．$C_{k-1}^2 - A_k^2 B_k$ を計算するとその結果は $\sqrt{B_{k-1}}$ を含む項と $\sqrt{B_{k-1}}$ を含まない項に分けられる．(ただし，$\sqrt{B_{k-1}}$ を含まない項には開平なしの B_{k-1} を含むものもある．) $\sqrt{B_{k-1}}$ を含む項をまとめて $d_{k-1}\sqrt{B_{k-1}}$ とし，$\sqrt{B_{k-1}}$ を含まない項をまとめて D_{k-1} とする:

$$C_{k-1}^2 - A_k^2 B_k = D_{k-1} + d_{k-1}\sqrt{B_{k-1}}. \tag{2.9}$$

このとき d_{k-1} には $\sqrt{B_{k-1}}$ は含まれていない．

$$C_{k-1}^2 > A_k^2 B_k \tag{2.10}$$

$$\Leftrightarrow \quad D_{k-1} + d_{k-1}\sqrt{B_{k-1}} > 0 \tag{2.11}$$

$$\begin{aligned} \Leftrightarrow \quad & (D_{k-1} \geq 0 \text{かつ} - d_{k-1}\sqrt{B_{k-1}} \geq 0 \text{かつ} D_{k-1}^2 > d_{k-1}^2 B_{k-1}) \\ & \text{または} (D_{k-1} \geq 0 \text{かつ} - d_{k-1}\sqrt{B_{k-1}} < 0) \\ & \text{または} (D_{k-1} < 0 \text{かつ} - d_{k-1}\sqrt{B_{k-1}} < 0 \text{かつ} D_{k-1}^2 < d_{k-1}^2 B_{k-1}). \end{aligned} \tag{2.12}$$

ここで，$-d_{k-1}\sqrt{B_{k-1}} \geq 0, -d_{k-1}\sqrt{B_{k-1}} < 0$ はそれぞれ，

$$(-d_{k-1} \leq 0 \text{かつ} B_{k-1} \geq 0), (-d_{k-1} > 0 \text{かつ} B_{k-1} > 0) \tag{2.13}$$

と同等である．$D_{k-1}, d_{k-1}, B_{k-1}$ は $\sqrt{B_{k-1}}$ を含まないので，これで $C_{k-1}^2 > A_k^2 B_k$ から $\sqrt{B_{k-1}}$ の開平を取り去ることができた．

同様の操作を繰り返して，順次 $\sqrt{B_{k-2}}, \cdots, \sqrt{B_1}$ の開平を取り去ることができる．(厳密には k に関する帰納法で証明できる．) 結局，$\sqrt{B_1}, \cdots, \sqrt{B_{k-1}}, \sqrt{B_k}$ の k 個の開平を取り去ることができることになる．これは，はじめの条件 $F(x) > G(x)$ と同等な判定を，上述の大小比較の組で表したとき，大小比較に現われる任意の式において，開平のネストの深さが l より小さくなったことを意味する．すなわち，上述の大小比較の組に現われる，任意の式 E において，

$$\mathrm{index}(E) <_{\mathrm{lex}} (l,k) \tag{2.14}$$

であり，帰納法の仮定により，条件 $E > 0$ と同等な条件を入力変数の多項式の大小比較の組合せで得ることができる．

これにより，すべての場合について証明が尽くされたので (超限) 帰納法により命題は証明された.(証明終)

この命題から，結局，アルゴリズムが，入力データに四則演算と開平を組合せて施した式の値の大小比較で分岐を行ない位相データを求めるならば，加減乗算のみで位相データを求めることができることがわかる．いいかえれば，位相データは入力変数の多項式の大小比較の組合せで求めることができる．

これに対して，計量データは，このように多項式のみでは求めることができない．これは，計量データは連続値をとり，値そのものが意味を持つので，式の値を変えるような式変形ができないからである．

しかしながら，まず位相データを求め，図形の構造を決定したのちに，必要最小限の計算を行なって，計量データを得ることが考えられる．すなわち，アルゴリズムを位相データと計量データの加工に分離して考えることができる．こうすることによって計算誤差の影響を最小限にできる．例えば，5.1.1節で述べる，点 Voronoi 図に関しては，Voronoi 辺や Voronoi 点，Voronoi 領域の接続関係が位相データであり，Voronoi 点の座標値が計量データである．Voronoi 図の双対図形の Delaunay 図は位相データしか持たない．まず位相データを求めるということは，Delaunay 図を求め，得られた図形を双対だと判断して Voronoi 図の位相データに翻訳し，それから計量データである Voronoi 点の座標値を計算する，という手順を踏むことに相当する．

2.4 退化の定義

本節では退化を定式化する．本論文で扱う幾何的アルゴリズムを次の範囲に限定する．

1. 入力変数の個数が限定されている．
2. 入力は整数または浮動小数である．
3. アルゴリズム中の計算のうち，位相データの決定に用いられるものは四則演算と開平に限られる．
4. 条件分岐で大小比較に用いられる式の個数は定数で押さえられる．

第 1 の条件は，入力のサイズが変動しないことを意味する．しかしながら，各入力サイズごとに別のアルゴリズムとして考察することにすれば特に強い制限ではない．第 2 の条件も一般的な幾何的アルゴリズムでは成立する．第 3 の条件は，多くの幾何的アルゴリズムで成り立つ．実際，例えば 2 個のベクトルのなす角度が 2 組あり，それらの大きさの比較をしたければ角の正弦，余弦，正接から比較ができ，これらはベクトルの成分から四則演算と開平を用いて求めることができる．第 4 の条件は，位相データを計算で求めていく過程で，あり得る場合の数に上限があることを意味する．例えば，反復法を用いたアルゴリズムの中には，入力に依存して反復回数がいくらでも大きくなるものがあるが，そのようなものは除外される．しかしながら，計算幾何学がアルゴリズム論と離散数学の影響のもとに発展し，常に計算時間のオーダを問題にしてきたため，実際には第 4 の条件を満たす幾何的アルゴリズムはかなり多い．総合的には本節で扱う幾何的アルゴリズムは，現状の幾何的アルゴリズムの大部分を占める．また，本論文では 3.6節などで，適宜，これらの条件の緩和に関しても考察を加える．

幾何的アルゴリズムの位相データは入力データの多項式の値の大小比較で与えられる．式をすべて左辺に移項すれば，位相データは多項式の値の符号判定で与えられるともいえる．したがって，制限を加えることなく次の条件を追加することができる

- アルゴリズム中で位相データ決定に関する条件分岐は，入力変数の多項式の値の符号によるものにかぎる．

以後, この条件も仮定する. この, 条件分岐で正負の決定のつきかねる時, すなわち, 値がちょうど 0 になる時を退化というわけである. ここで, アルゴリズム中の位相データ決定に用いられる式を判定多項式とよぶことにする.

位相データの決定に用いられる式が開平や除算を含む場合, 前節 (2.3節) で示したような方法で機械的に多項式の組に分解すると, 容易に想像がつくように, 判定多項式の数が元に較べて爆発的に増大することがある. その時には, 計算時間の観点からみてアルゴリズムの実用性は失われる. しかしながら, 多くの場合, そのようなアルゴリズムは, 設計の時点で多項式になるように考慮すれば, 計算時間で実用性を損なわないものが作れることが多い. したがって, この条件はアルゴリズムの実用性を損なわない.

定義 1 (退化)
幾何的アルゴリズムに入力を与えて実行したとき, 判定多項式の値が 0 になるならば, その幾何的アルゴリズムにおいて入力が**退化**しているという.

注意しなくてはならないのは, この定義は, 幾何的問題に関して入力図形が単独で退化していることを定めるものではなく, その幾何的問題を扱うアルゴリズムと入力の組に対して退化していることを定めていることである.

例えば 2.1節の交差線分対の発見問題の効率的なアルゴリズムであるアルゴリズム 1では, 与えられた線分の 2 個の端点が等しい x 座標を持つときは, 入力線分は退化している. なぜならば, アルゴリズムのはじめに線分の端点を x 座標でソートしていて, ソーティングのアルゴリズムの中で, 大小比較のために x 座標の差をとると値が 0 になるからである. それに対して, 素朴なアルゴリズムでは端点の x 座標が一致するものがあってもそれだけでは退化ではない. これは退化であるということが, 入力だけでなくアルゴリズムにも依存することを示している.

しかしながら, 例えば, 交差線分対の発見問題の場合には, ある線分の端点が他の線分の上にのる場合には, どのようなアルゴリズムでも入力は退化になるであろう. このように, どのようなアルゴリズムを採ろうと入力が退化しているとみなされる場合, 問題に固有の入力退化であるとみなしてもよいであろう. このときには多くの場合, 最終結果の位相データが通常と異なる. ここまでで, 入力の退化には, 扱う問題に固有のものとアルゴリズムにより定まるものがあることがわかった. ここでは, 問題に固有の入力退化を**本質的退化**, アルゴリズムにより定まる退化を**形式的退化**とよぶことにする. 本質的退化を厳密に定義するのは難しい. しかし, 形式的退化は本質的退化を含むことは明らかであろう. 以後, 単に退化と言えば形式的退化を指すものとし, 本質的退化に関しては, 6章で若干の考察を加えるが, 本論文では主として形式的退化を扱うことにする.

幾何的アルゴリズムの入力データは位相データと計量データからなる. これらを一列に並べて計量データの入力変数を $x_1, \cdots, x_n$ とし, 位相データの入力変数を $y_1, \cdots, y_m$ とする. アルゴリズムに入力を与えるということは, これらの変数に具体的な値を代入することである. ここで, ある入力に関して, $y_1, \cdots, y_m$ に与える値を固定し, $x_1, \cdots, x_n$ に与える値をわずかに変化させる. その結果はこのアルゴリズムが許容する入力であるべきである. そうでないときには, 入力の位相データと計量データが食い違うことになるが, それは, これらのデータが入力として冗長であることを意味している. そのような時には, 記憶量や入力時間を節約する観点からも, 余分なデータを入力から排除するべきである. ここでは, 入力データのう

ち位相データ部分を固定して，計量データをわずかに変化させても入力として許容されると仮定する．

以後では，退化している入力を考えるときには，位相データ部分を固定し，定数として扱う場合がある．この時には，入力変数を $x_1, \cdots, x_n$ だけからなるとみなし，任意の判定多項式は $x = (x_1, \cdots, x_n)$ の関数として $f(x)$ として表される．この表式に基づく退化の定義は次のようになる．

定義 2 ((形式的) 退化)
幾何的アルゴリズム P の中の判定多項式全体の集合を D_P とする．入力 $a = (a_1, \cdots, a_n) \in \mathbf{Q}^n$ に対し，ある判定多項式 $f(x) \in D_P$ があり，$f(a) = 0$ を満たすとき，入力 a はアルゴリズム P に関して (形式的に) **退化**しているという．

2.5 退化対処法に関わる諸問題

幾何的アルゴリズムで多くの場合，プログラムとして実装して運用し，入力の退化が問題になってから，問題になった場合のみを解消するような方法が採られることが多い．このような方法を**その場しのぎ法**とよぶことにする．

例えば，N 個の三角形が入力として頂点の座標値で与えられ，三角形の包含関係を列挙する問題を考える (図 2.4)．入力の三角形は潰れていないとする．

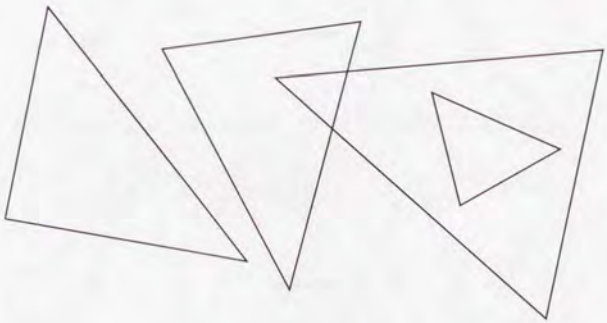


図 2.4: 三角形の包含関係列挙問題

この問題を解くために，三角形の各辺に反時計周りの向きを付け，各辺を延長した有向直線の左右のどちら側に他の三角形の頂点がのるかの判定を基本判定として，この基本判定を組合せたアルゴリズムを考えることができる．このようなアルゴリズムにおいて，平面 $\mathbf{R}^2$ 上の 3 点 $p_i = (x_i, y_i), p_j = (x_j, y_j), p_k = (x_k, y_k)$ に関して，p_i から p_j に向かう有向直線の左側に p_k がある (図 2.5) ことの判定式は

$$f_{ijk}(x) = \begin{vmatrix} 1 & 1 & 1 \\ x_i & x_j & x_k \\ y_i & y_j & y_k \end{vmatrix} > 0 \tag{2.15}$$

である. (本論文では座標系をいわゆる右手系にとる.) これは p_i, p_j, p_k が同一直線上にあるときにのみ 0 になる.

図 2.5: 3 点の位置関係

したがって, $f_{ijk}(x)$ を判定式に使うアルゴリズムでは, 入力で, ある三角形の辺 (またはその延長) 上に他の三角形の頂点がある場合, 入力は退化になる. このとき, N 個の三角形に順番をつけ, さらに, 各三角形が重心方向に微少に縮んでいるとみなし, 縮んだ度合いは, 三角形に付けた順番の前のものほど大きいと仮想してみる (図 2.6).

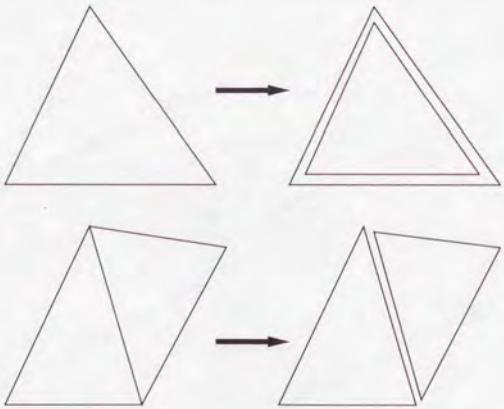


図 2.6: 三角形の微少変形

この仮想の下で有向直線に対する点の左右判定を処理すると, 一般の場合はもちろん, たとえ 2 個の三角形が同じ位置の 3 頂点を占めているような退化時にも, 矛盾のない出力が得られる. これは, その場しのぎ法が有効に見える例である.

一方, N 個の線分が端点の座標値で与えられていて, 交差する線分対を列挙する問題を考える. 線分は 1 点に潰れていないとする (図 2.7).

この場合, 平面 $\mathbf{R}^2$ 上の 4 点 $p_i = (x_i, y_i), p_j = (x_j, y_j), p_k = (x_k, y_k), p_l = (x_l, y_l)$ に関し

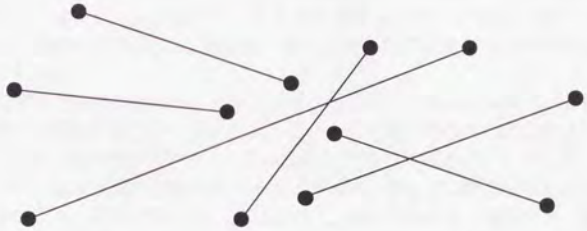


図 2.7: 交差線分対列挙問題

て, 2 線分 p_ip_j, p_kp_l が交差することの判定は, 上で定義した $f_{ijk}(x)$ らを用いて,

$$f_{ijk}(x)f_{jil}(x) < 0 \text{かつ} f_{kli}(x)f_{lkj}(x) < 0 \tag{2.16}$$

とかける. この判定式を使うアルゴリズムでは, 退化は, 入力において, ある線分の延長上に他の線分の端点がのったときに生じる. ここで, 各線分の重心 (中点) 方向に微少に縮んだと仮想してみる (図 2.8).

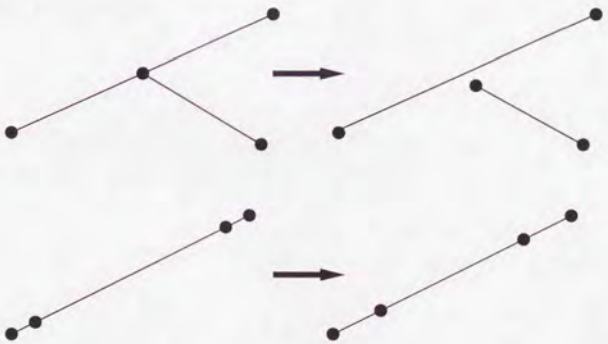


図 2.8: 線分の微少変形

もし, 2 個の線分が同一直線上にある場合には, 結局, このように仮想しても判定式の値は 0 のままであり, 退化状態の対処法にはならない. もちろん, 2 個の線分が同一直線上にない場合の退化には対処できる. これは, その場しのぎ法が対処できない退化を残す例である.

現実には, アルゴリズムを考案する場では, 問題に, どの三角形の頂点も他の三角形の辺の延長上にないものとする, あるいはどの線分の端点も他の線分の延長上にないものとする, というような前提を付けて退化問題を避けることが多い. これをプログラムとして計算機に実装して, 使用しようとして, 初めて退化を考慮することになる. すなわち, まずアルゴリズ

ムに忠実なプログラムを作ることからはじめ，あとは，特定の入力で計算が破綻する度に，その場しのぎ法でプログラムを変更することを繰り返す．このような状態では，プログラムの作成者に過大な負担がかかるうえ，最終的にすべての退化に対処したプログラムが出来上がるとは期待できない．

さらに，計算誤差の問題がある．アルゴリズムのプログラムへの実装では計量データの入力変数は，通常は浮動小数として扱われる．このように扱うかぎり，数値計算の誤差は避けられない．しかも，この問題もアルゴリズム設計の段階では棚上げされていることが常である．しかしながら，値が正確に計算できると仮定すると，判定式の値が0のとき以外はアルゴリズムやプログラムが期待どおり動くのに対して，誤差が存在する場面では，判定式の値が0でないときにも，符号を誤りプログラムが期待した動きをしない可能性がある点において，誤差の問題は，誤差なしの退化対処の問題よりも深刻であるといえる．通常のプログラムの実装では浮動小数を使っているので，誤差と退化の両方の問題を抱えている．

いずれにしろ，現在では社会の多くの局面で計算機が管理や制御の部分の鍵になっている．そこでは，プログラムが正常に動き出力をして正常終了する，ということは必須の条件のはずであるが，実際には，退化を無視したアルゴリズム設計によって正常動作が保証されないプログラムが通用している．よって，アルゴリズムを退化状態でも正常に動作するようにすることは，工学上の問題として，プログラム作成の現場からの強い要請である．その場しのぎ法に代わる，誤差を含む退化の問題への対処法が求められているわけである．

誤差の問題を分離するかどうかで，退化の対処法は大きく二分できるであろう．以下では，誤差の問題を退化の問題と分離する方法，誤差と退化を同時に扱う方法について，それぞれ，3章，5章で論じることにする．

第 3 章

代数的退化対処法

本章では，誤差の問題を分離した退化対処法である代数的退化対処法について扱う．

3.1 退化対処法の代数的形式化

幾何的アルゴリズムにおいて出力図形の構造 (位相データ) を求めるための計算が誤差なく実行できることは，次のように説明できる．幾何的アルゴリズムにおいて，位相データ，計量データに応じて，入力変数は整数または浮動小数の値をとる．したがって，入力データは有限桁である．また，前節でアルゴリズム中の各処理で位相データは，入力変数の多項式の符号判定で決定できることを示した．この場合，多項式中の定数は浮動小数または整数であり，やはり有限桁である．このような多項式に有限桁の数を代入しても結果はやはり有限桁である．これは位相データの決定は，結局，有限桁の数の符号の判定ですむことを示している．しかも判定多項式は有限個であるから，多項式の値の範囲は，大雑把ならば，入力の値の上限と下限から比較的容易に求めることができる．その範囲をカバーできるだけの桁数の計算機構をあらかじめ用意しておけば，アルゴリズム内の位相データの決定は誤差なく実行できる．実際，適当な整数倍をすることによって計算をすべて多倍長の整数計算に帰着させることができる．多倍長計算にかかる計算時間に関しては 3.8節や 4章で考えることにする．

本章では，誤差の問題を退化の問題から分離し，多倍長整数計算に帰着させるなどの方法で，アルゴリズム内の位相データを決定するためのすべての計算を誤差なしでおこなうことを前提とする．このような前提をとると，退化の対処問題は代数的に捕えることができるようになる．本章では，退化対処法を代数的に形式化し，考察していくことにする．

まず，アルゴリズム内に出てくる判定式 $f(x)$ の全体からなる集合を D とする．ここで，$x = (x_1, \cdots, x_n)$ は計量データの入力変数を 1 列に並べたものである．位相データは固定されているとする．

入力変数に与える具体的な値が入力である．入力を $a = (a_1, \cdots, a_n) \in \mathbf{Q}^n$ とする．入力が与えられて，判定式 $f(x) \in D$ は値 $f(a) \in \mathbf{Q}$ を持つようになる．入力 a が (形式的に) 退化しているとは

$$\exists f(x) \in D, f(a) = 0 \tag{3.1}$$

であることである．退化に対処するとは，$f(a) = 0$ のときにも適当な符号を割り当てて，アルゴリズムの実行が破綻しないようにすることである．ここで，アルゴリズムが破綻しないことの定義としては，次の定式化で十分であろう．まず，D で生成される $\mathbf{Q}[x]$ の部分多項式環を P_D とする．

定義 3 ((広義の) 代数的退化対処法)
D を判定多項式全体の集合とする幾何的アルゴリズムにおいて,

$$f(a) > 0 \Rightarrow f(x) >_D 0 \tag{3.2}$$

を満たす P_D を全順序環にする全順序 $\geq_D$ が与えられたとき, この全順序 $\geq_D$ を (広義の) 代数的退化対処法とよぶ.

ここで, "環 R を全順序環にする全順序 $\geq_R$" は次のように定義される.

定義 4 (全順序環)
環 R とその上の全順序 $\geq_R$ が与えられているとする. 任意の $r,s,t \in R$ に対して,

$$r,s >_R 0 \;\Rightarrow\; r+s >_R 0 \text{かつ} rs >_R 0, \tag{3.3}$$

$$r >_R s \;\Rightarrow\; r+t >_R s+t \tag{3.4}$$

が成り立つならば, 全順序 $\geq_R$ を環 R を**全順序環**にする全順序とよぶ.

この退化対処法の定義3は, まず, 任意の判定式から始め, それらの判定式の加減乗算によって作られる任意の多項式に一貫した符号を与えようというものである.

実際には, P_D を確定することは難しいので, P_D を含む全空間 $\mathbf{Q}[x]$ 上の全順序で, 退化対処法の定義の条件を満たすものを見つけることになる. このようにして退化対処法の範囲を狭めたものを (狭義の) **代数的退化対処法**とよぶことにする.

代数的退化対処法の理解のために, 次のような枠組みを用意する. 入力 $a \in \mathbf{Q}^n$ を固定する. 写像 $\varphi : \mathbf{Q}[x] \to \mathbf{Q}$ を

$$\varphi : f(x) \mapsto f(a) \tag{3.5}$$

で定義する. これは環準同型であるから, 環の準同型定理によって,

$$\mathbf{Q}[x]/\mathrm{Ker}\varphi \cong \mathrm{Im}\varphi \; (= \mathbf{Q}) \tag{3.6}$$

である. $\mathbf{Q}[x]/\mathrm{Ker}\varphi$ と $\mathbf{Q}$ を同一視する. $\mathbf{Q}[x]$ と $\mathbf{Q} \times \mathrm{Ker}\varphi$ の間には,

$$\mathbf{Q}[x] \ni f(x) \leftrightarrow (f(a), f(x) - f(a)) \in \mathbf{Q} \times \mathrm{Ker}\varphi \tag{3.7}$$

という一対一対応がある. 退化対処法の定義の条件

$$f(a) > 0 \Rightarrow f(x) >_D 0 \tag{3.8}$$

から, 全順序 $\geq_D$ から自然に導入される $\mathbf{Q} \times \mathrm{Ker}\varphi$ 上の順序は辞書式順序であることが必要である. これは, $\mathrm{Ker}\varphi$ の元は無限小であることを示している. ここで, 無限小とは絶対値が任意の正の有理数より小さいが 0 でない量である. $\mathrm{Ker}\varphi$ はイデアルであり, その基底は $x_i - a_i \; (i = 1, \cdots, n)$ である. ここで, $X_i = x_i - a_i \; (i = 1, \cdots, n)$ と変数変換する. また, $X = (X_1, \cdots, X_n)$ とする. 必要ならば, X_i と $-X_i$ の置換を考えればよいから退化対処法を与える全順序 $\geq_D$ として, 各 X_i はすべて正であるもののみを考えても, 一般性を失わない.

もし $n=1$ ならば, $\mathrm{Ker}\varphi$ は一変数多項式環 $\mathbf{Q}[x_1]$ のイデアルであるから単項イデアルであり, 実際, 単一の元 x_1-a_1 すなわち X_1 から生成される. この場合, イデアルの各元である多項式の正負は, 自動的に, その多項式の X_1 に関する最小次数の項の係数の符号で決まる. (2 個の多項式の大小関係は多項式の差の符号で任意性なく決まる.) しかしながら, $n\geq 2$ のときには, $\mathrm{Ker}\varphi$ は単項イデアルにならない. このため, 符号の決め方に任意性があり, 問題を難しくしている.

$X_i\,(i=1,\cdots,n)$ のベキ全体を PWR と書き, $\geq_D$ を PWR に制限したものも誤解の恐れがないので $\geq_D$ と書くと, 退化対処法の定義からただちに,

- $\geq_D$ は PWR 上の全順序である,
- $W,W',W''\in$ PWR に対し, $W<_D W'\Rightarrow WW''<_D W'W''$,
- $W\in$ PWR に対し, $W\leq_D 1$

が成立する. 逆に, 上の 3 条件を満たす PWR 上の全順序 $\geq'_D$ が得られているとき, $\mathbf{Q}[x]$ や $X_i\,(i=1,\cdots,n)$ の生成するイデアル $\mathrm{Ker}\varphi$ の上に, 全順序 $\geq'_D$ を

$$\begin{aligned} & f(x)>'_D\, g(x) \\ \Leftrightarrow\ & \begin{array}{l} f(x)-g(x)\text{を } X_1,\cdots,X_n \text{の多項式として計算したときの} \\ \text{各項の文字部分のうち,} \geq'_D \text{で最大のものの係数が正} \end{array} \end{aligned} \tag{3.9}$$

と定めて拡張すると, これは退化対処法になることが容易にわかる. 上の 3 条件を満たす PWR 上の順序は代数学の研究対象であり, **項順序** (term order) とよばれている.

非負整数全体を $\mathbf{N}_0$ とかき, $\alpha=(\alpha_1,\alpha_2,\cdots,\alpha_n)\in\mathbf{N}_0^n$ とする. これを多重指数とする: $X^\alpha=X_1^{\alpha_1}X_2^{\alpha_2}\cdots X_n^{\alpha_n}$. このとき, PWR と $\mathbf{N}_0^n$ の間に,

$$\mathbf{N}_0^n\ni\alpha\leftrightarrow X^\alpha\in\mathrm{PWR} \tag{3.10}$$

という自明な一対一対応がつく. この対応によって PWR と $\mathbf{N}_0^n$ を同一視して, PWR で定義された項順序を $\mathbf{N}_0^n$ 上の項順序ともいうことができる. このとき, PWR と $\mathbf{N}_0^n$ の項順序の大小関係が逆転することに注意しておく. $\mathbf{N}_0^n$ 上の項順序の例として辞書式順序と線形結合型順序を挙げることができる. 本論文では, 辞書式順序を既知のものとして用いてきたが, ここで線形結合型順序とともに, $\mathbf{N}_0^n$ 上の辞書式順序も定義しておく.

定義 5 (辞書式順序)
$\mathbf{N}_0^n$ の任意の 2 元 $\alpha=(\alpha_1,\cdots,\alpha_n)$, $\beta=(\beta_1,\cdots,\beta_n)$ に対して,

$$\alpha>_{\mathrm{lex}}\beta\Leftrightarrow\alpha_i\neq\beta_i\text{となる最小の}i\text{に対して}\alpha_i>\beta_i \tag{3.11}$$

と $\geq_{\mathrm{lex}}$ を定めると, $\geq_{\mathrm{lex}}$ は $\mathbf{N}_0^n$ の項順序になる. この全順序 $\geq_{\mathrm{lex}}$ を**辞書式順序**という.

定義 6 (線形結合型順序)
正の実数 $w_1,\cdots,w_n$ を $\mathbf{Q}$ 上線形独立になるようにとる. $\mathbf{N}_0^n$ の任意の 2 元 $\alpha=(\alpha_1,\cdots,\alpha_n)$, $\beta=(\beta_1,\cdots,\beta_n)$ に対して,

$$\alpha>_{\mathrm{lin}}\beta\Leftrightarrow w_1\alpha_1+\cdots+w_n\alpha_n>w_1\beta_1+\cdots+w_n\beta_n \tag{3.12}$$

と $\geq_{\mathrm{lin}}$ を定めると, $\geq_{\mathrm{lin}}$ は $\mathbf{N}_0^n$ の項順序になる. この全順序 $\geq_{\mathrm{lin}}$ をここでは**線形結合型順序**ということにする.

項順序に関しては次のような定理がある [62, 83].

定理 2
$\mathbf{N}_0^n$ の任意の項順序 $\trianglelefteq$ に対し，すべての成分が非負実数である n 次正則行列 T が存在して，任意の $\alpha, \beta \in \mathbf{N}_0^n$ を列ベクトルと見たとき，

$$\alpha \lhd \beta \Leftrightarrow T\alpha <_{\mathrm{lex}} T\beta \tag{3.13}$$

を満たす．

この定理は，項順序は辞書式順序と線形結合型順序の適当な混ぜ合わせに限ることを示している．退化対処法をこのような代数的枠組で定義すると，理論的には任意性の範囲まで指摘できることがわかった．

以下の節では，従来提案されてきた各種の退化対処法を，この代数的枠組で眺め直し，お互いの関係を明らかにする．

3.2 Yap の記号摂動法

本節では C. K. Yap による，記号摂動法 [87] について解説する．

前節の代数的退化対処法は，入力を微小量だけ動かすことを考えると，より直感的に説明できる．入力を a から微小量ベクトル $\varepsilon = (\varepsilon_1, \cdots, \varepsilon_n)$ だけ動かし $a + \varepsilon$ にしてみると，$X_i = x_i - a_i$ $(i = 1, \cdots, n)$ であるから，イデアル $\mathrm{Ker}\varphi$ の任意の元を X_i の多項式 $f(X)$ で表すと，x に $a + \varepsilon$ を代入することは，各 X_i を ε_i に書き換えることに等しい．これは項順序を与えることが入力を微小に動かしたときの各値の変化量の大小関係を与えることに一致していることを示している．入力の微小量の変化という捕え方で，退化対処法を解析的に扱うことができる．

この解析的な扱いを基本にした退化対処法が Yap の記号摂動法である．まずここで用いる記号を定義することにする．文字 $x_1, \cdots, x_n$ のベキ全体の集合を Pwr とする．多重指数 $\alpha = (\alpha_1, \cdots, \alpha_n) \in \mathbf{N}_0^n$ に対して $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$, $X^\alpha = X_1^{\alpha_1} \cdots X_n^{\alpha_n}$ と定める．この集合 Pwr と，前節で定義した $X_1, \cdots, X_n$ のベキ全体 PWR には，多重指数を介した一対一対応

$$\mathrm{Pwr} \ni x^\alpha \leftrightarrow X^\alpha \in \mathrm{PWR} \tag{3.14}$$

がつく．多重指数全体 $\mathbf{N}_0^n$, PWR, Pwr に項順序 $\leq_D$ を導入しておく．$w = x^\alpha \in \mathrm{Pwr}$ に対して，偏微分演算子 ∂_w を

$$\partial_w = \frac{\partial^{\alpha_1 + \cdots + \alpha_n}}{\partial {x_1}^{\alpha_1} \cdots \partial {x_n}^{\alpha_n}} \tag{3.15}$$

と定める．また，各判定多項式の各変数 x_i に関する次数をとり，その最大値を maxdeg_D とかくことにする．これで記号の準備が終わった．

判定多項式を $f(x)$ とすると，Yap の記号摂動法はアルゴリズム的には次のように記述される．

アルゴリズム 3 (Yap の記号摂動法)
1. $R \leftarrow \{x^\alpha \in \mathrm{Pwr} | \forall i\ \alpha_i \leq \mathrm{maxdeg}_D\}$

2. $w \leftarrow R$内で$\leq_D$における最大元
3. $R \leftarrow R - \{w\}$
4. $\partial_w f(a) \neq 0$ならば, その符号を返して終了.
5. $R \neq \emptyset$であるうちは**2**に戻る.
6. 多項式として0を返して終了.

すなわち, $\partial_w f(a) \neq 0$を満たす$w \in \mathrm{Pwr}$のうち, $\leq_D$による最大のものをとり, $\partial_w f(a)$の符号をもって$f(x)$の符号とするわけである. ただし, Pwrの元wの$\leq_D$に関して大きいほうから順に$\partial_w f(a)$求めると, 有限回では$\partial_w f(a) \neq 0$にならない場合がある. 上のアルゴリズムで, maxdeg_Dを用いてRをPwrの部分集合にしたのはRを有限集合にするためである. また, このような方法を採るかぎり, Dまたは$\max \deg_D$が(少なくとも上から押さえる形で)外延的に明らかである必要がある. その意味では, 本論文で扱う幾何的アルゴリズム以外の, 例えば, 上限がない反復を含むアルゴリズムはこの方法では対処できない. しかし, 多くの(一般的な)幾何的アルゴリズムはそのようなことはなく, この記号摂動法が利用できる条件を満たしている. また, 判定式$f(x)$の入力aにおける偏微分係数$\partial_w f(a)$を求める必要があるが, これはあらかじめ求めておくか, 必要なときに数式処理や自動微分によって求めることができる. 数値微分では計算誤差を避けることが難しい.

この記号摂動法の理論的な解析を行なう. $f(x)$を$x_1, \cdots, x_n$を変数とする多項式とすると, $X_i = x_i - a_i$であるから,

$$\frac{\partial f}{\partial x_i}(x) = \frac{\partial f}{\partial X_i}(X + a) \tag{3.16}$$

が成立する. したがって

$$\partial_{x^\alpha} f(x) = \partial_{X^\alpha} f(X + a) \tag{3.17}$$

が成立する. $X_1, \cdots, X_n$の多項式$f(X + a)$をMaclaurin展開して,

$$f(x) = f(X + a) = \sum_{X^\alpha \in \mathrm{PWR}} \frac{1}{(\alpha_1 + \cdots + \alpha_n)!} \partial_{X^\alpha} f(a) X^\alpha \tag{3.18}$$

$$= \sum_{x^\alpha \in \mathrm{Pwr}} \frac{1}{(\alpha_1 + \cdots + \alpha_n)!} \partial_{x^\alpha} f(a) X^\alpha \tag{3.19}$$

を得る. $f(X + a)$は多項式であるから右辺は有限和となる. ここで, xに入力aのかわりに$a + \varepsilon$, $\varepsilon = (\varepsilon_1, \cdots, \varepsilon_n)$を代入すると,

$$f(a + \varepsilon) = \sum_{X^\alpha \in \mathrm{PWR}} \frac{1}{(\alpha_1 + \cdots + \alpha_n)!} \partial_{X^\alpha} f(a) \varepsilon_1^{\alpha_1} \cdots \varepsilon_n^{\alpha_n} \tag{3.20}$$

となる. $\varepsilon_1, \cdots, \varepsilon_n$が正数で十分小さければ, また, 上式の各項の$\varepsilon_1, \cdots, \varepsilon_n$のベキの間に, 項順序$\leq_D$に同期した大小関係を付けることが可能である. また, 各ε_iが正で十分小さいとすると, $f(a + \epsilon)$の符号は展開式の各項で$\varepsilon_1, \cdots, \varepsilon_n$のベキが最大のものの係数の符号によって決定される. すなわち, 上式の右辺で, $\varepsilon_1^{\alpha_1} \cdots \varepsilon_n^{\alpha_n}$が最大のベキであるならば$(1/(\alpha_1 + \cdots + \alpha_n)!)\partial_{x^\alpha} f(a)$の符号が式の符号を決定する. $1/(\alpha_1 + \cdots + \alpha_n)!$は正であるから結局$\partial_{x^\alpha} f(a)$の符号が式の符号を決定する. いいかえると, $\partial_{x^\alpha} f(a) \neq 0$である$x^\alpha$のうち$\leq_D$で最大のものの$\partial_{x^\alpha} f(a)$の符号が$f(a + \varepsilon)$の符号になる. これは前に示した記号摂動法のアルゴリズム3に一致する.

以上によって, 記号摂動法は, 入力 a に具体的に ε という十分小さな摂動を与えたときの符号を求めていることと, 符号を求めるに当たって, 各 ε_i の具体的な値を用いず, 記号として扱っていることが明らかになった.

Yap の記号摂動法は代数的退化対処法を摂動 ε を仮想して解析的に素直にアルゴリズムにしている. そのため, 対象になる幾何的アルゴリズムの制限は少ない. しかしながら, 一般論に近いために, 偏微分をするときの x^α の選び方や偏微分係数の求め方など, 実用的なプログラムの簡潔さや効率性を実現するには問題ごとに個別に困難が残るといえる.

3.3 Edelsbrunner らの SoS

本節では H.Edelsbrunner, E. P. Mücke による SoS (Simulation on Simplicity) とよばれる退化対処法 [14] について考察する. この対処法には, 対象とする判定式をある種の行列式に限るという制限がある. ここでは, 入力変数を d 次元の点 n 個として, $x = (x_{ij})$, $1 \leq i \leq d, 1 \leq j \leq n$ とかくことにする. これに一致するように入力も $a = (a_{ij})$ で表す. SoS では判定式に制限があるにも関わらず, 多くの幾何的アルゴリズムに適用可能である. 実際, n 次元の点 x_j $(1 \leq j \leq n)$ を, $x_j = (x_{1j}, \cdots, x_{dj})^{\mathrm{T}}$ で定めると, 行列

$$A = \begin{bmatrix} x_{j_1} & \cdots & x_{j_d} \end{bmatrix}, B = \begin{bmatrix} 1 & \cdots & 1 \\ x_{j_1} & \cdots & x_{j_{d+1}} \end{bmatrix}, C = \begin{bmatrix} 1 & \cdots & 1 \\ x_{j_1} & \cdots & x_{j_{d+2}} \\ \|x_{j_1}\|^2 & \cdots & \|x_{j_{d+2}}\|^2 \end{bmatrix} \tag{3.21}$$

(ただし, $j_1, \cdots, j_{d+2}$ は 1 以上 n 以下の相異なる整数, $\| \bullet \|$ は Euclid ノルム) の行列式 $|A|$, $|B|$, $|C|$ の入力 a における符号判定ができるだけで

- 与えられた $d-1$ 個のベクトルの張る超平面が分ける 2 個の半空間のうち, 与えられた点はどちらに属すか,
- 与えられた d 個の点を通る超平面が分ける 2 個の半空間のうち, 与えられた点はどちらに属すか,
- 与えられた d 個の点を通る超平面は与えられた点を端点とする線分と交わるか,
- 与えられた $d+1$ 個の点を通る超球は与えられた点を含むか,

などの判定ができる. これらの判定を基本にして, 様々な幾何的アルゴリズムが構成できる.

SoS は汎用性を保ったままアルゴリズム的に記述するのは難しいが, 初めの行列式 $|A|$ に例をとって記述する. まず, 行列 A の列の並び方を

$$A = \begin{bmatrix} x_{j_1} & \cdots & x_{j_d} \end{bmatrix}, \ j_1 < j_2 < \cdots < j_d \tag{3.22}$$

のように限定しても一般性を失わない. この列の並び方での行列式の符号と列の置換の符号から, 列を置換した場合の行列式の符号を求めることができるからである. ここではこの $|A|$ の符号を求めるアルゴリズムを記述する.

その前にいくつかの記号を定義する. 集合 J, S を $J = \{1, \cdots, d\}, S = \{(r, c) \mid r, c \subseteq J, \ |r| = |c|\}$ とする. $(r, c) \in S$ に対して, $r = \{r_1, \cdots, r_k\}, c = \{c_1, \cdots, c_k\}, r_1 < \cdots < r_k, c_1 > \cdots > c_k$ とする. S 上に全順序 $\leq_S$ を

$$(r, c) \leq_S (r', c') \Leftrightarrow (r_1, c_1, \cdots, r_k, c_k, 0) \leq_{\mathrm{lex}} (r'_1, c'_1, \cdots, r'_{k'}, c'_{k'}, 0) \tag{3.23}$$

で与える．ここで，右側の不等式の両辺のベクトルの次元が異なるが，先頭の成分からの比較をすると，両辺のどちらかの成分が 0 になったところで終了するので，辞書式順序で比較可能であることに注意しておく．また，$(r,c) \in S$ に対して，行列 A から第 $r_1,\cdots,r_k$ 行の各行と第 $c_1,\cdots,c_k$ 列の各列を除いた行列を $\tilde{A}(r,c)$ とする．ここで，$(\emptyset,\emptyset) \in S$ は S の最小元であり，$\tilde{A}(\emptyset,\emptyset) = A$ と定める．また，$(J,J) \in S$ であり，$\tilde{A}(J,J) = 1$ と定める．$\tilde{A}(r,c)$ に a を代入したものを，$\tilde{A}(r,c)(a)$ とかく．$\tilde{A}(\emptyset,\emptyset)(a)$ を $A(a)$ とかく．

行列式 $|A|$ に関する SoS をアルゴリズム的に記述すると次のようになる．

アルゴリズム 4 ($|A|$ に関する **SoS**)

1. $S' \leftarrow S$

2. $(r,c) \in S'$ を S' の $\leq_S$ に関する最小元とする．

3. S' から元 (r,c) を取り除く．

4. $|\tilde{A}(r,c)(a)| = 0$ のうちは **2** に戻る．

5. $(-1)^{r_1+\cdots+r_k+c_1+\cdots+c_k}|\tilde{A}(r,c)(a)|$ の符号を返して終了．

$(J,J) \in S, \tilde{A}(J,J) = 1$ であるから，このアルゴリズムは必ず終了する．

次に，アルゴリズムの基礎となる理論的な背景を述べる．まず，摂動 $\varepsilon(i,j),(i,j) \in J \times J$ を入力 a に与える．$J \times J$ に全順序 $\leq_{\mathrm{SoS}}$ を

$$(i,j) \leq_{\mathrm{SoS}} (i',j') \Leftrightarrow (i,-j) \leq_{\mathrm{lex}} (i',-j') \tag{3.24}$$

で定める．摂動 $\varepsilon(i,j),(i,j) \in J \times J$ は大小関係

$$\varepsilon(i,j) < \prod_{(k,l)>_{\mathrm{SoS}}(i,j)} \varepsilon(k,l) \tag{3.25}$$

$$\tag{3.26}$$

が成り立つ正の無限小とする．$\tilde{A}(r,c)(a)$ の各 a_{ij} を $a_{ij}+\varepsilon(i,j)$ に置き換えたものを $\tilde{A}(r,c)(a+\varepsilon)$ とかく．特に，$\tilde{A}(\emptyset,\emptyset)(a+\varepsilon)$ を $A(a+\varepsilon)$ とかく．$|A(a+\varepsilon)|$ は $\varepsilon(i,j)$ の多項式として展開できる：

$$\begin{aligned} |A(a+\varepsilon)| &= |A(a)| \\ &\quad +(-1)^{d+1}|\tilde{A}(\{1\},\{d\})(a)|\varepsilon(1,j_d) \\ &\quad +(-1)^{d+2}|\tilde{A}(\{1\},\{d-1\})(a)|\varepsilon(1,j_{d-1}) \\ &\quad +\cdots. \end{aligned} \tag{3.27}$$

この展開式の各項を摂動 $\varepsilon(i,j)$ の積の大きい順に並べて項の係数が非零になる最初の項の係数の符号を元の行列式 $|A|$ の入力 a における符号に定めることができる．また，$\varepsilon(i,j)$ は単一の正の無限小 e の単項式で与えられているとみることもできる．実際，例えば，

$$\varepsilon(i,j) = e^{2^{id-j}} \tag{3.28}$$

と定めればよいことが容易にわかる．有理数係数の多項式環は 1 変数多項式環 $\mathbf{Q}[e]$ のときには単項イデアル整域であるが，2 変数以上の多項式環になると単項イデアル整域ではないので，展開式の計算などの処理を，数式処理のように記号的に行なうときには，一変数のほ

うが有利である．また，$|A(a+\varepsilon)|$ の展開式で $\varepsilon_{i'_1 j'_1}\varepsilon_{i'_2 j'_2}\cdots\varepsilon_{i'_k j'_k}$ の項の係数は，符号を除いて $|\tilde{A}(\{i'_1,\cdots,i'_k\},\{j'_1,\cdots,j'_k\})(a)|$ と一致する．ここで，$\{i'_1,\cdots,i'_k\}$, $\{j'_1,\cdots,j'_k\}$ を集合として変えずに添字を変更した各項を考えると，$|\tilde{A}(\{i'_1,\cdots,i'_k\},\{j'_1,\cdots,j'_k\})(a)|=0$ のときには，これらの項はすべて $|A|$ の符号を決定しない．$|\tilde{A}(\{i'_1,\cdots,i'_k\},\{j'_1,\cdots,j'_k\})(a)|\neq 0$ のときには，これらの項のうち，積 $\varepsilon_{i'_1 j'_1}\cdots\varepsilon_{i'_k j'_k}$ が最も大きくなるもの以外は $|A|$ の符号を決定するとき参照されない．ここで $\varepsilon_{i'_1 j'_1}\cdots\varepsilon_{i'_k j'_k}$ が最も大きくなるのは

$$i'_1<\cdots<i'_k,\ j'_1>\cdots>j'_k \tag{3.29}$$

のときである．$|A|$ の符号を決定するには，この条件を満たす項を，大きいほうから順に参照していけばよいことになる．この項の係数は符号を含めて

$$(-1)^{i'_1+\cdots+i'_k+j'_1+\cdots+j'_k}|\tilde{A}(\{i'_1,\cdots,i'_k\},\{j'_1,\cdots,j'_k\})(a)| \tag{3.30}$$

である．この条件を満たす項に対応する添字全体のつくる集合が S であり，$\leq_S$ は項の大小関係に対応する S 上の全順序である．

このように，SoS は基本の考え方では Yap の記号摂動法の特別な場合と見ることができる．しかし，対象となる判定式を A のような行列式に限り，項順序を $\leq_S$ に限ることによって符号判定する項の数を減らすことに成功している．この点で SoS は適用できる問題に対してはより効率的であるといえる．$|B|$, $|C|$ に関しても同様な考え方で符号を決定できる．それ以外の判定式に対しても適用可能な場合はあるが，一般に式が複雑になり符号判定に必要な計算が増大する．

3.4 Emiris らの退化解除法

本節では I. Emiris, J. Canny による，退化対処法 [15] について解説する．この対処法は SoS と同様に，対象とする判定式をある種の行列式に限るという制限がある．ここでの記号は基本的に SoS と同様にとる．すなわち，入力変数を d 次元の点 n 個として，$x=(x_{ij}),1\leq i\leq d,1\leq j\leq n$ とかく．入力も $a=(a_{ij})$ で表す．点 x_j $(1\leq j\leq n)$ を，$x_j=(x_{1j},\cdots,x_{dj})^{\mathrm{T}}$ で定める．Emiris らの方法が対象とする行列式は，行列

$$A=\begin{bmatrix} x_{j_1} & \cdots & x_{j_d}\end{bmatrix},B=\begin{bmatrix}1 & \cdots & 1\\ x_{j_1} & \cdots & x_{j_{d+1}}\end{bmatrix} \tag{3.31}$$

(ただし，$j_1,\cdots,j_{d+1}$ は 1 以上 n 以下の相異なる整数) の行列式 $|A|$, $|B|$ である．これらの行列式の入力 a における符号判定により，

- 与えられた $d-1$ 個のベクトルの張る超平面が分ける 2 個の半空間のうち，与えられた点はどちらに属すか，
- 与えられた d 個の点を通る超平面が分ける 2 個の半空間のうち，与えられた点はどちらに属すか，
- 与えられた d 個の点を通る超平面は与えられた点を端点とする線分と交わるか，

などの判定ができる．SoS と同様にこれらの判定を基本にして，様々な幾何的アルゴリズムが構成できる．

Emiris らの方法は正方行列の特性多項式を求めるアルゴリズムに依存する．正方行列の特性多項式を求めるアルゴリズムは W. Keller-Gehrig によって正方行列の積を求める計算時間と同じオーダのもの [44] が開発されている．このアルゴリズムを利用する．記号として V_d を次の Vandermonde 行列とし，V'_d を次の行列とする．

$$V_d = \begin{bmatrix} 1 & \cdots & 1 \\ j_1 & \cdots & j_d \\ \vdots & & \vdots \\ {j_1}^{d-1} & \cdots & {j_d}^{d-1} \end{bmatrix}, V'_d = \begin{bmatrix} j_1 & \cdots & j_d \\ {j_1}^2 & \cdots & {j_d}^2 \\ \vdots & & \vdots \\ {j_1}^d & \cdots & {j_d}^d \end{bmatrix}. \tag{3.32}$$

ここで，

$$\det V_d = \prod_{k>l}(j_k - j_l),\ \det V'_d = \det V_d \prod_{k=1}^{d} j_k \tag{3.33}$$

であることに注意する．したがって，V_d, V'_d は，それぞれ，同一の列がないとき正則である．また，行列 B の第一行をすべて 0 で置き換えたものを B' とする:

$$B' = \begin{bmatrix} 0 & \cdots & 0 \\ x_{j_1} & \cdots & x_{j_{d+1}} \end{bmatrix}. \tag{3.34}$$

行列式 $|A|$ に関する Emiris らの方法をアルゴリズム的に記述すると次のようになる．

アルゴリズム 5 ($|A|$ に関する **Emiris** らの退化対処法)
1. $M \leftarrow -{V'_d}^{-1}A$
2. 特性多項式 $\phi(e) = \det(M - eI_d)$ を求める．
3. $\phi(e)$ の非零の各項のうち e の次数の最小のものの係数を s とする．
4. $(-1)^d s \det V'_d$ の符号を返して終了．

行列式 $|B|$ に関しては次のようになる．

アルゴリズム 6 ($|B|$ に関する **Emiris** らの退化対処法)
1. $M \leftarrow -{V_{d+1}}^{-1}B'$
2. 特性多項式 $\phi(e) = \det(M - eI_{d+1})$ を求める．
3. $\phi(e)$ の非零の各項のうち e の次数の最小のものの係数を s とする．
4. $(-1)^{d+1} s \det V_{d+1}$ の符号を返して終了．

次に，アルゴリズムの基礎となる理論的な背景を述べる．SoS の場合と同じように，摂動 $\varepsilon(i,j), (i,j) \in J \times J$ を入力 a に与える．e を正の無限小として，

$$\varepsilon(i,j) = i^j e,\ (i,j) \in J \times J \tag{3.35}$$

とする．A の入力として $x_{ij} = a_{ij}, x_{ij} = a_{ij} + \varepsilon(i,j)$ を与えたものを，それぞれ，$A(a), A(a+\varepsilon)$ とする．$B(a), B(a+\varepsilon), B'(a)$ も同様の規則で定めておく．行列式 $|A(a+\varepsilon)|$ は e の d 次多項式で，

$$|A(a+\varepsilon)| = |A(a)| + (e\text{の 1 次以上の項}) \tag{3.36}$$

である. 特に d 次の項が $e^d|V'_d|$ になるので e の 1 次以上の非零項は d 次までに少なくとも 1 項は残る. 理論的には SoS と同様に $|A(a+\varepsilon)|$ を展開し, e のベキの小さい項から順に係数が非零になるまで見ていけば, $|A|$ の符号を定めることができる. $|B|$ に関しても同様である. しかしながら, ここでは行列式 $|A|$, $|B|$ を変形して, 同等の結果をより簡易に実現する. $M = -V_d'^{-1}A(a)$ とすると,

$$\begin{aligned} |A(a+\varepsilon)| &= |A(a) + eV'_d| \\ &= |-V'_d||-V_d'^{-1}A(a) - eI_d| \\ &= (-1)^d|V'_d||M - eI_d| \end{aligned} \tag{3.37}$$

であるから, $|V'_d|$ と $|M - eI|$ の符号がわかれば, ただちに $|A(a+\varepsilon)|$ の符号を求めることができる. $|V'_d|$ の符号は既に得られており, $|M - eI_d|$ の符号は e を変数としたときの M の特性多項式 $\phi(e) = |M - eI_d|$ の各非零項のうち, 最小次数のものの係数の符号である. $M = -V_d'^{-1}A(a)$ は既知であり, 行列の特性多項式は行列の積を求めるのと同じオーダの計算時間で求めることができる [44]. $|B|$ のほうも $M = -V_{d+1}^{-1}B'(a)$ とすれば同様に,

$$\begin{aligned} |B(a+\varepsilon)| &= |B'(a) + eV_d| \\ &= \frac{1}{e}|-V_{d+1}||-V_{d+1}{}^{-1}B'(a) - eI_{d+1}| \\ &= (-1)^{d+1}\frac{1}{e}|V_{d+1}||M - eI_{d+1}| \end{aligned} \tag{3.38}$$

から $|B(a+\varepsilon)|$ の符号を求めることができる. これらの理論的背景をもとにアルゴリズム化したものが Emiris らの退化対処法である. ここで用いられた摂動 ε は各変数で同位の無限小を用いている. Yap の方法でも, これと同じ結果を与える全順序 $\leq_S$ を与えることができる. しかし Emiris らの方法は, 入力データ個数を固定することと, 判定式を $|A|, |B|$ の型に限定することによって, 1 個の無限小 e の問題へ変換することに成功している. それによって, 特性多項式を求めるアルゴリズムを利用可能にし, アルゴリズムの簡単化に成功している点に注意したい.

3.5 Fortune の退化対処法

本節では S. Fortune による退化対処法 [17] について考察する. この対処法はこれまでに解説した対処法とは異なり, 多面体のモデリングという特定の問題に即して提案されたもので, Fortune の展開した多面体モデリングの理論のなかでは特に主要な位置を占めているわけではない. 対象とする判定式は行列

$$A = \begin{bmatrix} x_{i_1 1} & x_{i_1 2} & x_{i_1 3} & x_{i_1 4} \\ x_{i_2 1} & x_{i_2 2} & x_{i_2 3} & x_{i_2 4} \\ x_{i_3 1} & x_{i_3 2} & x_{i_3 3} & x_{i_3 4} \\ x_{i_4 1} & x_{i_4 2} & x_{i_4 3} & x_{i_4 4} \end{bmatrix}, \ (i_1 < i_2 < i_3 < i_4) \tag{3.39}$$

の行列式 $|A|$ である. これは X, Y, Z を変数とする 3 次元空間内の 2 次元平面

$$P_i : x_{i1}X + x_{i2}Y + x_{i3}Z + x_{i4} = 0 \tag{3.40}$$

に対して，3平面 $P_{i_1}, P_{i_2}, P_{i_3}$ の交点が平面 P_{i_4} の (Z 軸に関して) 上方にあるかどうかの判定に用いることができる．$k = 1,2,3,4$ に対して，A から第 k 行と第4列を取り除いた行列を $\tilde{A}_k$ とかく．入力を $a = (a_{ij})$ とおく．$A, \tilde{A}_k$ の入力変数に a を代入したものを，それぞれ，$A(a), \tilde{A}_k(a)$ とかく．

行列式 $|A|$ に関する Fortune の退化対処法をアルゴリズム的に記述すると次のように簡潔である．

アルゴリズム 7 ($|A|$ に関する **Fortune** の退化対処法)
1. $|A(a)|, |\tilde{A}_1(a)|, -|\tilde{A}_2(a)|, |\tilde{A}_3(a)|, -|\tilde{A}_4(a)|$ の順で，最初に 0 でないものの符号を返す．

次に，アルゴリズムの基礎となる理論的な背景を述べる．SoS や Emiris らの方法の場合と同じように，摂動 $\varepsilon(i,j)$ を入力 a に与えるのであるが，e を正の無限小として，

$$\varepsilon(i,1) = \varepsilon(i,2) = \varepsilon(i,3) = 0, \varepsilon(i,4) = e^i \tag{3.41}$$

とする．つまり平面の方程式3.40の定数項だけに摂動を与える．すると，A の入力として $x_{ij} = a_{ij} + \varepsilon(i,j)$ を与えたものを，$A(a+\varepsilon)$ とすると，

$$|A(a+\varepsilon)| = |A(a)| + |\tilde{A}_1(a)|e^{i_1} - |\tilde{A}_2(a)|e^{i_2} + |\tilde{A}_3(a)|e^{i_3} - |\tilde{A}_4(a)|e^{i_4} \tag{3.42}$$

が成立する．右辺のすべての項が0にならない限り符号が決定できる．これは，4個の平面 $P_{i_1}, P_{i_2}, P_{i_3}, P_{i_4}$ のうち，1点で交わる3個の平面が少なくとも1組あるという条件を満たせば，符号が決定できることを意味する．実際，Fortune が，このような符号決定法を利用しようとした多面体のモデリングでは，この条件は満たされる．

これらの理論的背景をもとにアルゴリズム化したものが Fortune の退化対処法である．ここで用いられた摂動 ε は，1個の無限小 e を用い，さらに，すべての変数に摂動を加えているわけではない．Yap の方法で，これと同じ結果を与える全順序 $\leq_S$ を与えることはできる．しかし，Fortune の方法は，形式的にすべての退化に対処することを避け，問題によって生じないことがわかっている退化への対処を省くことによって，アルゴリズムの簡単化に成功している点に注意したい．また，この方法は退化を分類する可能性を示唆する．この点に関しては後の3.8節で扱う．

3.6 超準解析による記号摂動法の解釈と記号摂動の限界

本節では記号摂動の代数的な取り扱いとは別に，超準解析 [11, 41] によって従来の方法を解釈する．この観点から記号摂動法の種類の限界を示すことを試みる．

自然数全体の部分集合族 $F = \{\mathbf{N} - A | A \subseteq \mathbf{N}, |A| < \infty\}$ を**非単項フィルタ**とよぶ．非単項フィルタのうち包含関係 $\subseteq$ に関して極大なものを (非単項) **超フィルタ**とよぶ．容易にわかるように，超フィルタは次の5条件を満たす．

- $\emptyset \notin F$
- $A, B \in F$ ならば $A \cap B \in F$
- $A \in F, A \subset B$ ならば $B \in F$

- $|A| < \infty$ ならば $A \notin F$
- $A \notin F$ ならば $\mathbf{N} - A \in F$

超フィルタ F を一つ固定する．ある幾何的アルゴリズム P の入力を $a \in \mathbf{Q}^n$, 判定多項式の全体を D_P とおく．点列 $\{\varepsilon_j \in \mathbf{Q}^n | j \in \mathbf{N}\}$ を，$\varepsilon_j \to 0$ $(j \to \infty)$ を満たし，任意の判定多項式 $f(x) \in D_P$ に対して，$\{j | f(a+\varepsilon_j) = 0\}$ が有限集合になるように取る．このような点列 $\{\varepsilon_j\}$ は a や D_P に依存せずにとることもできる．$\mathbf{Q}_+$ を正の有理数全体の集合とする．制限 $\forall \varepsilon_j \in \mathbf{Q}_+^n$ を加えても，点列 $\{\varepsilon_j\}$ を a や D_P に依存せずにとることができる．

このようにとった $\{\varepsilon_j \in \mathbf{Q}_+^n\}$ に対して，判定多項式 $f(x)$ が正であることを $\{j | f(a+\varepsilon_j) > 0\} \in F$ で定めるとこれはちゃんと定義されていて (well-defined)，判定多項式に符号を定めることができる．このような符号の決定法を**超準解析的な手法**とよぶ [9, 10, 40]．点列 $\{\varepsilon_j\}$ によって点集合を 1 列に並べなくても，いわゆる“フィルタによる収束”[52] によって，同等の議論を進めることができるが，本論文の目的のためには必要がないので，そのような方法を採用しない．

$\mathbf{N}^n$ 上の，ある項順序 $\lhd$ が非負実数を成分とする正則行列 $T = [t_{ij}]$ と辞書式順序 $<_{\text{lex}}$ によって

$$\alpha \lhd \beta \Leftrightarrow T\alpha <_{\text{lex}} T\beta \tag{3.43}$$

と表されたとする (定理 2)．$\alpha, \beta \in \mathbf{N}^n$ は列ベクトルにみなした多重指数である．このとき，$T^{-1} = [t'_{ij}]$ として，各 $i = 1, \cdots, n$ ごとに $\sum_k t'_{ik} k$ に収束する有理数列 $\{d_{ij}\}$:

$$d_{ij} \in \mathbf{Q},\ d_{ij} \to \sum_k t'_{ik} k,\ (j \to \infty) \tag{3.44}$$

をとり，点列 $\{\varepsilon_j\}$ を，

$$\varepsilon_j = (\varepsilon_{1j}, \cdots, \varepsilon_{nj})^{\mathrm{T}}, \tag{3.45}$$

$$\varepsilon_{ij} = 2^{-jd_{ij}} \tag{3.46}$$

と定めると，この点列 $\{\varepsilon_j\}$ から定まる判定多項式の符号は項順序 $\lhd$ で定まる符号と一致する．この意味で超準解析的な手法は項順序による符号決定法を含むことになる．多項式環 $\mathbf{Q}[x]$ を全順序環にする順序が与えられれば，だだちに項順序が得られることから，超準解析的な手法で得られる符号決定法と項順序による符号決定法は同一のものであることがわかる．すなわち，超準解析的な手法は項順序によるものの一つの別種の解釈であるともいえる．

超準解析的な手法で，判定多項式 $f(x)$ が正であることを $\{j | f(a+\varepsilon_j) > 0\} \in F$ で定めることにしたが，$f(a+\varepsilon_j) > 0$ であることをすべての j に関し調べることは直接には無理である．項順序による方法では，$f(x)$ が正であることを示すのに $f(x)$ の各項を見て，命題

$$\exists j' \ (j > j' \Rightarrow f(a+\varepsilon_j) > 0) \tag{3.47}$$

が成り立つことを示している．これは

$$|\{j | f(a+\varepsilon_j) \leq 0\}| < \infty (\text{ゆえに} \{j | f(a+\varepsilon_j) \leq 0\} \notin F) \tag{3.48}$$

を示していることになる．これは符号判定を超フィルタではなく，その前の非単項フィルタの範囲で判定をしていることになる．項順序による方法で，判定多項式の数を有限に制限し

ている理由はここにある．実際，判定多項式の集合が無限集合のときには，あらかじめアルゴリズム中のすべての判定部で退化に対処する判定を個別にしておくことは困難であるし，機械的に項順序に従って判定多項式の項を順に調査すると無限ループに陥り得る．

例えば，$\{\varepsilon_i\}$ を

$$\varepsilon_i = (\varepsilon_{i1}, \cdots, \varepsilon_{in}), \tag{3.49}$$

$$\varepsilon_{ij} = 2^{-2^{ij}} \ (j = 1, \cdots, n) \tag{3.50}$$

と定める．$a = 0 \in \mathbf{Q}^n$ とする．多項式 $f(x)$ が $f(a) = 0$ となることと $f(x)$ が定数項を持たないことは同値である．任意の，零でなく定数項を持たない多項式 $f(x) = \sum f_{jk} x_j^k$ に対して，

$$\begin{aligned} f(a + \varepsilon_i) &= \sum_{j,k} f_{jk} \varepsilon_{ij}^k \\ &= \sum_{j,k} f_{jk} \cdot 2^{-2^{ij}k} \end{aligned} \tag{3.51}$$

である．$f_{jk} \neq 0$ となる j, k のうち，(j, k) の辞書式順序で最小のものを，それぞれ，j', k' とする．i がある程度より大きくなると常に，上式の右辺での各項で $f_{ij'} \cdot 2^{-2^{ij'}k'}$ が右辺の値を支配する．すなわち，

$$\frac{f(a + \varepsilon_i)}{f_{j'k'} \cdot 2^{-2^{ij'}k'}} \to 1, \ (i \to \infty) \tag{3.52}$$

が成立する．これは集合 $\{i \in \mathbf{N} | f(a + \varepsilon_i) \neq 0\}$ が有限集合であることを意味する．したがって，$\{\varepsilon_i\}$ の定め方として式3.50は妥当である．同時に，ある程度 i が大きいと常に $f(a + \varepsilon_i)$ の符号が $f_{j'k'}$ に一致することもわかる．したがって，超準解析的な手法による $f(x)$ の符号は $x_{j'}^{k'}$ の項の係数 $f_{j'k'}$ の符号に一致する．ここで，多項式 $f_k(x), g(x)$ $(k \in \mathbf{N})$ を

$$f_k(x) = x_1^k, \ g(x) = x_2 \tag{3.53}$$

とする．任意の $k \in \mathbf{N}$ に対して，

$$f_k(x) - g(x) = x_1^k - x_2 \tag{3.54}$$

であるから，$(j', k') = (1, k)$ である．$x_{j'}^{k'}$ の係数は1であるから，超準解析的な手法による $f_k(x) - g(x)$ の符号は正である．任意の $k \in \mathbf{N}$ に対して $f_k(x) - g(x)$ が判定多項式である場合，この $\{\varepsilon_i\}$ による超準解析的な手法で符号判定するために，機械的に手続きを定めると，x_2 の項の係数をみる前に，$x_1, x_1^2, x_1^3, \cdots$ の各項の係数をみることになる．もし，アルゴリズムが，個々の判定多項式の x_1 の最大次数などの情報を用いないならば，アルゴリズム全体では判定多項式の x_1 の次数の上限がないので，x_2 の係数をみることなくアルゴリズムは無限ループに陥る．

点列 $\{\varepsilon_j\}$ は入力 a によらずにとることができることで，超準解析的な手法では判定多項式の数が有限である必要はないことがわかる．ここでは，判定多項式の数が有限でない場合，超準解析的な手法を使っても符号判定は困難であることを述べる．判定多項式の全体が無限集合の場合，判定多項式 $f(x)$ によっては，集合

$$\{j | f(a + \varepsilon_j) > 0\}, \{j | f(a + \varepsilon_j) \leq 0\} \tag{3.55}$$

のどちらも無限集合になる可能性がある (判定多項式が有限個の場合, これらの集合のどちらかを有限集合にするように点列 $\{\varepsilon_j\}$ がとれる) から, $f(x)$ が正であることの確認に有限性:

$$|\{j|f(a+\varepsilon_j) \leq 0\}| < \infty \tag{3.56}$$

を用いることができない. したがって, 無限集合が F の要素であるかどうか調べる他の何らかの手続きが必要であろう. ところが, 次の定理が成り立つ [10].

定理 3
集合論の公理系において, 超フィルタの存在は選択公理より弱い公理である.

すなわち, 集合論の公理系では選択公理を用いずに超フィルタの存在を証明することができない. この定理によって, 集合論の下では, 超フィルタ F の要素かどうかを, 手続き的に判定することはもちろん, 手続き的に超フィルタ F を作ることすらできないことがわかる.

計算機内で数学的な処理を行なう場合, 計算機システム自身を, 通常の数学のモデルとして扱う. しかし, そうしている限り, 超準解析的な手法はアルゴリズムとしては記述できない. これは, 超準解析的な手法とともに, 判定多項式の符号判定法の拡張の限界を示しているといえよう.

そうはいえ, 計算機で集合を扱う場合, 要素には自然に順序が付く. これは計算機システムを通常の数学と異なる数学のモデルと見なせる可能性を示している. 集合論の標準モデル以外のモデルの上での計算機システムの扱いは, 今後の課題である.

3.7 Gröbner 基底による退化の解釈と対処法の構成

本節では, Gröbner 基底を利用して退化を扱い, 対処法の構成を行なう.

前節までで示したとおり, 退化対処法は通常, 入力 $a \in \mathbf{Q}^n$ を用いて, n 変数多項式環 $\mathbf{Q}[x]$ のイデアル $I = \{f(x) \in \mathbf{Q}[x] | f(a) = 0\}$ の "つじつまの合った" 符号付けとして定式化される. イデアル論の応用で有用性が知られている Gröbner 基底をもとに退化対処法を以下で新規に考案する.

定義 7 (主項, Gröbner 基底)
一般に, I を (体係数の) 多項式イデアル, $\lhd$ を多重指数に関する項順序とする. I の要素の多項式に対して, $\lhd$ で多重指数が最大の項を**主項**という. I の部分集合 B に対して, I は B で生成され, I の要素の主項全体で生成されるイデアルと B の要素の主項全体で生成されるイデアルが一致するとき, B を I の $\lhd$ に関する **Gröbner 基底**という.

項順序を固定しても Gröbner 基底は一意には定まらないが, Gröbner 基底を固定すると任意の多項式を Gröbner 基底の各要素で整除できるかぎり整除した結果, 整除した順によらず剰余が一意に定まるという性質がある [8, 83]. したがって, 剰余が 0 であることとイデアルに属することが一致する.

ここでは, $f(x), g(x), h(x), \cdots \in \mathbf{Q}[x]$ で生成されるイデアルを $\langle f(x), g(x), h(x), \cdots \rangle$ と書くことにする. また, 項順序と, イデアル $I = \{f(x) \in \mathbf{Q}[x] | f(a) = 0\}$ の Gröbner 基底 $B = \{b_1, \cdots, b_{n'}\}$ を次の条件を満たすように取って固定する.

- 各 b_j は正である. (すなわち, 項順序に関して 0 より大きい.)

- $\{b_j, \cdots, b_{n'}\}$ は $\langle b_j, \cdots, b_{n'}\rangle$ の Gröbner 基底である $(j = 1, \cdots, n')$.
- $b_1, \cdots, b_{n'}$ の主項は項順序によってこの順に並ぶ (b_1 の主項が最大 (多重指数が最小)).
- 主項の多重指数が項順序によって b_j より小さい任意の多項式 $f(x)$, $g(x)$ に関して, 積 $f(x)g(x)$ の主項の多重指数も b_j より小さい $(j = 1, \cdots, n')$.

ここでは簡単のため, 項順序を辞書式順序にとり, $b_i = x_i - a_i$ として話を進める. すなわち, $B = \{x_1 - a_1, \cdots, x_n - a_n\}$ である. また, $\varepsilon_i = x_i - a_i$ と変数変換をする. $\varepsilon = (\varepsilon_1, \cdots, \varepsilon_n)$ とかく. すなわち, 問題を $I = \{f(\varepsilon) \in \mathbf{Q}^n[\varepsilon] | f(0) = 0\}$ の符号判定にして, $B = \{\varepsilon_1, \cdots, \varepsilon_n\}$ にとる (ε_1 が最大). すなわち, $1 \gg \varepsilon_1 \gg \cdots \gg \varepsilon_n \gg 0$ を満たすと思えばよい.

$\mathbf{Q}[\varepsilon_1, \cdots, \varepsilon_i]$ を $\mathbf{Q}[\varepsilon]_i$ と略記する. $\mathbf{Q}[\varepsilon]_0 = \mathbf{Q}$ とする. とくに, $\mathbf{Q}[\varepsilon]_n$ は $\mathbf{Q}[\varepsilon]$ である.

$f \in \mathbf{Q}[\varepsilon]_i$ に対して, その分解 $(f_0, \cdots, f_i)$ を

$$f = f_0 + f_1\varepsilon_1 + \cdots + f_i\varepsilon_i,\ f_j \in \mathbf{Q}[\varepsilon]_j \tag{3.57}$$

で定める. 分解は末尾にいくらでも 0 を付けることができる:

$$(f_0, \cdots, f_i) = (f_0, \cdots, f_i, 0, \cdots, 0). \tag{3.58}$$

しかし, 通常はなるべく短い表記を採り, 計算時に必要に応じて適宜伸ばす.

多項式 $f \in \mathbf{Q}[\varepsilon]_i$ とその分解 $(f_0, \cdots, f_i)$ を同一視する.

容易にわかるように, $(f_0, \cdots, f_{i-1})$ は f を $\{\varepsilon_i, \cdots, \varepsilon_n\}$ で整除したときの剰余である. 構成法から f の分解 $(f_0, \cdots, f_n)$ は末尾の 0 を無視して一意的であることがわかる.

$f, g \in \mathbf{Q}[\varepsilon]_i$ に対して加減算は

$$f \pm g = \begin{cases} f_0 \pm g_0 & (i = 0), \\ f_0 \pm g_0 + (f_1 \pm g_1)\varepsilon_1 + \cdots + (f_i \pm g_i)\varepsilon_i & (i \geq 1) \end{cases} \tag{3.59}$$

を満たす. また乗算は

$$fg = \begin{cases} f_0 g_0 & (i = 0), \\ (f - f_i\varepsilon_i)(g - g_i\varepsilon_i), +(f_i g_i\varepsilon_i + f_i(g - g_i\varepsilon_i) + g_i(f - f_i\varepsilon_i))\varepsilon_i & (i \geq 1) \end{cases} \tag{3.60}$$

を満たす. これらは同時に $f, g \in \mathbf{Q}[\varepsilon]_i$ の i と ε_i の次数に関する帰納的定義にもなっている. 分解の形式では次のようになる.

$$f \pm g = \begin{cases} (f_0 \pm g_0) & (i = 0), \\ (f_0 \pm g_0, f_1 \pm g_1, \cdots, f_i \pm g_i) & (i \geq 1), \end{cases} \tag{3.61}$$

$$\begin{aligned} fg &= \begin{cases} (f_0 g_0) & (i = 0), \\ (h_0, \cdots, h_i) & (i \geq 1), \end{cases} \\ &\text{where } (h_0, \cdots, h_{i-1}) = (f_0, \cdots, f_{i-1})(g_0, \cdots, g_{i-1}), \\ &\quad h_i = (0, \cdots, 0, f_i g_i) + f_i(g_0, \cdots, g_{i-1}) + g_i(f_0, \cdots, f_{i-1})\ (i \geq 1). \end{aligned} \tag{3.62}$$

符号に関しては, $f \in \mathbf{Q}[\varepsilon]_i$ の符号は $i = 0$ のときは, f は数であるので, 正負に応じて, 符号はそれぞれ, $1, -1$ である. また一般には f の符号は, $f = 0$ のときには 0, $f \neq 0$ のとき

には, $f_j \neq 0$ である最小の j に対して f_j の符号, となる. この符号を与える関数を sign とする.

必要になってから sign を計算するのではなく $\mathbf{Q}[\varepsilon]$ に符号を組み込む. f の分解を拡張し, $f_S = \mathrm{sign}(f)$ を分解の先頭に付け加える: $f \mapsto (f_S, f_0, \cdots, f_n)$.

代入 $x_i \leftarrow a_i + \varepsilon_i$ を

$$x_i \leftarrow (\pm 1, \overset{0}{a_i}, \overset{1}{0}, \cdots, 0, \overset{i}{1}, 0, \cdots, \overset{n}{0}) \tag{3.63}$$

と変形する. 計算定義により, さかのぼることなく計算時にすぐに符号を決定できる.

このようにして, 退化に対処されていないアルゴリズム中の各計算を, 多項式の分解表記上の計算に置き換えることによって, 元のアルゴリズムを自動的に退化対処版に変更できる. ただし, 元のアルゴリズムの判定式が多項式である必要がある.

実装に関する点では, 計算機言語によってはオペレータオーバーローディング機能を利用することによって, プログラム本体の改変が不要であるという特徴をもつ. また, 判定式が任意の多項式でよいという汎用性をも持つ. 偏導関数計算が不要なため大掛かりな道具を必要としないという利点も持つ. この実装時の手軽さがこの方法の特長である. その反面で, 既存の各退化対処法が事前に判定式を調べ, 対処しておくという方式 (いわばコンパイラ方式) にであるのに比べ, ここで提案した方式は, アルゴリズムを働かせるときに退化に備えて必要データを計算し蓄えていく方式 (いわばインタプリタ方式) をとっているため, 計算時間や記憶量では優位性はない. ただし, 分解 $f = (f_S, f_0, f_1, \cdots)$ に対して, $f_j = 0$ となるところは記憶場所を節約しておくことができるので, 最悪時の記憶量では同等である.

ここで提案した方法で, 代入 $x_i \leftarrow a_i + \varepsilon_i$ を $x_i \leftarrow (\pm 1, a_i, 0, \cdots, 0, 1, 0, \cdots, 0)$ とするかわりに, x_i を $x_i + \varepsilon_i$ に置き換えて, x_i を文字として残したまま前処理として計算しておくこともちろん考えられる. このときには, 前処理の段階では符号 f_S を与えることができず, 入力 a が与えられて初めて計算することになる. このようにすると, 実質的に数式処理をしていることになり, Yap の方法と同等になる.

3.8 代数的退化対処法の評価と無誤差計算

ここまでの節で, いくつかの退化対処法を紹介し, 提案した. 汎用性という点では Yap の方法や Gröbner 基底を利用した方法が優れている. 一方で, Edelsbrunner らや Emiris らや Fortune は判定式の特殊性を用いて符号判定の簡素化や計算量の効率化をはかっている.

一般に, アルゴリズムの評価には, 計算量, 記憶量, 汎用性, 実装のしやすさなどが用いられる. また, 退化でないときには退化対処法は, 計算時間に影響を与えることがあっても, アルゴリズムの挙動そのものには影響を与えない. これらを考えあわせると, 各対処法の善し悪しは, 対象とする幾何的アルゴリズムにおける, 判定式の複雑さ, 退化の発生頻度, 退化時および非退化時の計算時間の増大の許容度などによって変わると言えるであろう.

Fortune の提案する多面体モデリングでは 3 平面の交点が第 4 の平面の上方にあるかどうかの判定に退化対処法を用いている. その際の入力としては起こらないが, 4 平面のうち, どの 3 平面も 1 点を定めない時には, Fortune の方法は退化対処法としては不十分である. しかし, これは退化対処法に関する興味深い示唆を与えていると思われる. すなわち, 3 平面が 1 点を定めないときに, 第 4 の平面との上下関係を定めることが幾何的に意味を持つかどうかということである. Fortune の方法では, 3 平面が定める 1 点が第 4 の平面上に乗ったときに上方か下方かを決定するだけである. これまで, 本論文では, 形式的に, 判定多項式の

値が0のときを退化と定めて議論をしてきたし，実際，計算幾何学の退化に関する研究ではそれが当然とされてきた．代数的退化対処法は，退化の問題を代数的に形式化したために，もともとは問題にあった幾何的側面を無視して，退化を広範囲に一括してしまっている．この，Fortuneの例は，対処すべき退化と対処しなくてもよい退化など，退化をより精密に分類する可能性を示しているといえる．本論文でも，2.4節の形式的退化，本質的退化や，3.1節の狭義の退化対処法，広義の退化対処法など，退化の階層化に着目してはいるが，この点に関して深く掘り下げることは今後の課題である．

また，これまで，位相データの決定のための判定式の符号判定は誤差なくできるとしてアルゴリズムと理論の説明を行なってきた．それは，判定式の除算と開平を消去することで，計算結果を有限桁に押さえることが可能であるからであった．実際，多くの研究で正確な判定をするための計算は別問題として扱われてきた[14, 15]．例えば，浮動小数を整数化して，整数計算するなどとされてきた．しかし，そうすることによって長い桁の整数計算をしなければならず，非退化時にも計算時間が増大する．十分すぎる桁数をとり実装すると，場合によっては計算時間の増大によって退化対処法を組み込む前に，プログラム自体が事実上の実行不能に陥る．計算時間を押さえるために，判定多項式がどのくらいの値になりえるかを正確に見積ることが考えられるが，一般にこれを正確に見積るのは難しい．

例えば平面上の3点 $x_1 = (x_{11}, x_{12})^{\mathrm{T}}, x_2 = (x_{21}, x_{22})^{\mathrm{T}}, x_3 = (x_{31}, x_{32})^{\mathrm{T}}$ を通る円が第4点 $x_4 = (x_{41}, x_{42})^{\mathrm{T}}$ を含むかどうかを決めることを考える(図3.1)と，

$$\det B = \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \end{vmatrix}, \det C = \begin{vmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ \|x_1\|^2 & \|x_2\|^2 & \|x_3\|^2 & \|x_4\|^2 \end{vmatrix} \tag{3.64}$$

の両方の符号を判定すればよい．ただし，$\| \bullet \|$ はEuclidノルムである．

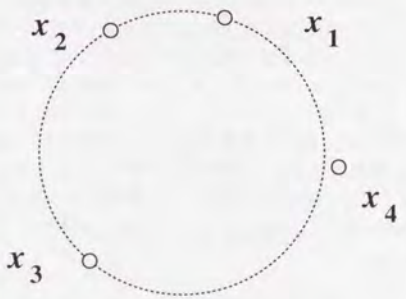


図 3.1: 円に関する点の内外判定

正整数 L を用いて，各 x_{ij} が $|x_{ij}| \leq L$ の範囲の値を取り得るとき，絶対値 $|\det B|, |\det C|$ を見積る．列方向にHadamard不等式を用いると，

$$|\det B| \leq \sqrt{2L^2+1}^3, |\det C| \leq (4L^4 + 2L^2 + 1)^2 \tag{3.65}$$

を得る．行方向に Hadamard 不等式を用いるだけで，

$$|\det B| \le 3\sqrt{3}L^2, |\det C| \le 32L^4 \tag{3.66}$$

となり，より良い見積りが得られる．$|\det C|$ のほうは，行方向に Hadamard 不等式を用いるまえに，行列式の中の 4 行目から 1 行目の L^2 倍を引くと (行列式の値は変わらない)，それだけで見積りが $32L^4$ から $16L^4$ に改善される．さらに $\det C$ を

$$\det C = \begin{vmatrix} 1 & 1 & 1 & 1 \\ x_{11} & x_{21} & x_{31} & x_{41} \\ x_{12} & x_{22} & x_{32} & x_{42} \\ x_{11}^2 & x_{21}^2 & x_{31}^2 & x_{41}^2 \end{vmatrix} + \begin{vmatrix} 1 & 1 & 1 & 1 \\ x_{11} & x_{21} & x_{31} & x_{41} \\ x_{12} & x_{22} & x_{32} & x_{42} \\ x_{12}^2 & x_{22}^2 & x_{32}^2 & x_{42}^2 \end{vmatrix} \tag{3.67}$$

と変形してから，右辺の各項を独立にうまく評価すると，$|\det C| < 10L^4$ 程度まで改善できる．しかしながら，実際には，正確な上限は

$$|\det B| \le 4L^2, |\det C| \le 8L^4 \tag{3.68}$$

である．また，この問題を平面から高次元に拡張すると未解決問題になる．このように計算結果の値の見積りは正確には難しい．

次に，十分に長い桁をとって計算することを考える．入力の数値を単長の整数として，一般に m 倍長の整数どうしの加減算は $O(m)$，乗算は通常の方法で $O(m^2)$，高速乗算法を用いても $O(m \log m \log\log m)$ の計算量がかかる．さらに，実装すると高速乗算法は m が数百程度ないと通常の方法より早くない．基礎的なもので数倍長から十数倍長，若干複雑でも数十倍長程度の整数で足りる幾何的アルゴリズムは少なくないので，乗算にのために退化に対処したアルゴリズムの計算時間が増大することになる．同様の意味で可変多倍長整数を用いるときの実装時のオーバーヘッドも問題になるであろう．実際，そのために，現在通用している汎用数式処理システムで高速乗算法が採用されているものはない [53] という．

判定式の値そのものではなく，符号を求めるのが目的であるから，浮動小数で計算し，誤差を考慮すると符号が決定できない時のみ高精度計算することを Fortune は提案している．実際，現在の多くの計算機では整数よりも浮動小数のほうがより長い桁をサポートし，桁数を考慮するとより高速でもある [17]．したがって，このような方法は，実用上有効である．しかしながら，判定式の値の計算で生じる誤差の見積りも難しいという問題がある．プログラムとして実装されたときにはアルゴリズム中の判定式は計算過程として与えられているので，各過程での最大誤差を独立して見積っていると過大な見積もりとなりがちである．

例えば，$x, y \in \mathbf{R}$ が，それぞれ，$[x_{\min}, x_{\max}]$，$[y_{\min}, y_{\max}]$ の範囲の値をとり得るとすると，積 xy は

$$[\min\{x_{\min}y_{\min}, x_{\min}y_{\max}, x_{\max}y_{\min}, x_{\max}y_{\max}\}, \max\{x_{\min}y_{\min}, x_{\min}y_{\max}, x_{\max}y_{\min}, x_{\max}y_{\max}\}] \tag{3.69}$$

の範囲の値をとる．これを用いて x^2 のとり得る範囲を見積ると，

$$[\min\{x_{\min}x_{\min}, x_{\min}x_{\max}, x_{\max}x_{\min}, x_{\max}x_{\max}\}, \max\{x_{\min}x_{\min}, x_{\min}x_{\max}, x_{\max}x_{\min}, x_{\max}x_{\max}\}] \tag{3.70}$$

となるが，実際には $x_{\min}x_{\max}$ という値は x^2 がとり得ないことがある．具体的に $x_{\min} = -1$，$x_{\max} = 3$ ならば，x^2 の値として $x_{\min}x_{\max} = -3$ という負の値はとり得ない．

このように過大になりがちな誤差の見積りを押さえるには，精密な対処が必要になるがこれは計算時間を増大させる．しかしながら，誤差の見積りを粗くすると，浮動小数計算では符号が決定できない場合が多くなり，高精度計算の割合が増え，やはり計算時間が増大する．計算時間の増大を押さえるには適度な誤差の見積りを設定する必要があるが，これに応える一般的な方法は，現在のところ開発されていない．

固定多倍長整数での計算，符号判定の代用として，本論文では次の節で，数倍長から数十倍長を扱うのに実用的な，剰余を利用した方法を提案する．

第 4 章

剰余演算を利用した多項式の値の符号判定法

4.1 剰余利用のための予備知識

計算機内部では通常, 整数の表現として 2 の補数表現を採用している. 整数を表すためのビット数を n とすると, 2 の補数表現とは, -2^{n-1} から $2^{n-1}-1$ までの整数を n ビットに納める表現法で, 非負の数は最上位のビットを 0 にして, そのまま $n-1$ ビットの整数として 2 進数表記し, 負の数は, その反数 (符号を反転した数) を 2 進数表記したもののすべてのビットを反転し, n ビットの 2 進数として 1 を加えたもので表記するという表現法である (n ビットを超える桁上がりは無視する)(図 4.1). 2 の補数表現では最上位の 1 ビットが 1 ならば負, 0 ならば非負になる. また, 2 の補数表現は -2^{n-1} から $2^{n-1}-1$ までの整数を 2^n で整除した剰余を, 2 進数表記することで n ビットに納める表現法であるとも言える.

0	=	0	0	0	0	·····	0	0	0	0
1	=	0	0	0	0	·····	0	0	0	1
2	=	0	0	0	0	·····	0	0	1	0
⋮										
$2^{n-1}-1$	=	0	1	1	1	·····	1	1	1	1
-2^{n-1}	=	1	0	0	0	·····	0	0	0	0
$-2^{n-1}+1$	=	1	0	0	0	·····	0	0	0	1
⋮										
-3	=	1	1	1	1	·····	1	1	0	1
-2	=	1	1	1	1	·····	1	1	1	0
-1	=	1	1	1	1	·····	1	1	1	1

図 4.1: 2 の補数表現

判定式の値を多倍長整数で求めてから符号を得ることは, 最上位の 1 ビットを知るために最上位ビット以外の全ビットを求めていることになる. 本節で提案する方法は, 剰余計算の結果の剰余を用いて, 判定多項式の値の下位のビットを復元することなく符号を求める方法である.

アルゴリズム中の判定多項式 $f(x)$ は陽に与えられるのではなく, 入力変数 $x=(x_1,\cdots,x_k)$

の加減乗算の手続きの組み合わせとして与えられる．ここでは，加減乗算を基本演算とよぶ．

定理 4
整数 n, n', m, m' と正整数 q に対し，$m \equiv m'$, $n \equiv n' \bmod q$ ならば，次式が成立する：

$$m \pm n \equiv m' \pm n', \ mn \equiv m'n' \bmod q. \tag{4.1}$$

この定理から，$f(x)$ に入力を代入し計算するためには，基本演算ごとに対応する剰余演算におきかえるだけですむことがわかる．これは，剰余演算を利用するためには，汎用の基本剰余演算のアルゴリズムを用意すれば，既存のアルゴリズムの本体を変更する必要がないことを示している．

加減乗算に対応する剰余演算は次のように行なう [34, 82]．ここで利用する計算機は，符号なし単長 (n ビットとする) 整数の基本演算で，次のような情報を得る機能が利用できるとする：

- 加減算に関して，$n+1$ ビット目への繰り上がりがあるかどうかの情報，
- 乗算に関して，結果の下位 n ビットと上位 n ビット．

このような機能は一般的な CPU には備わっている．この意味で本論文で示す方法はハードウェアに依存する．また通常は，$n = 4, 8, 16, 32, 64$ のように，n は 4 の倍数であることが多い．

以下では，$N \in \mathbf{Z}$, $q \in \mathbf{N}$ に対し，N を q で整除した商を $[N/q]$ で表し，剰余を $N\%q$ と表すことにする．したがって，

$$N = [N/q]q + N\%q \tag{4.2}$$

である．本節ではこの式をよく使う．また，$p_0, \cdots, p_m$ を互いに素な 2^n 以下の自然数で，

$$p_i = 2^{n_i} - \alpha_i, \ \alpha_i \geq 0, \ p_i > {\alpha_i}^2 \tag{4.3}$$

を満たすように選ぶ．p_i として 2 のベキ $2^{n'}$ を選んだときには $n_i = n', \alpha_i = 0$ であり，それ以外の時には，$n_i = (p_i$のビット長$) - 1$ となる．また，p_i のうちに偶数があれば，残りはすべて奇数になる．また，$p = \prod p_i$ とする．ここで，X, Y を整数とし，$X_i = X\%p_i, Y_i = Y\%p_i$ $(i = 0, \cdots, m)$ とする．$p_i = 2$ のとき X_i, Y_i は 1 ビットで取り扱いは特別に簡単になり，和と差 $(X \pm Y)\%p_i$ はどちらも X_i と Y_i の排他的論理和 (exclusive or) になり，積 $(XY)\%p_i$ は X_i と Y_i の論理積 (and) になる．$p_i = 2^{n_i} - \alpha_i$ のときは，基本演算を次のように行なう．基本演算の結果 Z が n_i ビットに納まらないときには，下位 n_i ビットを Z_L とし，上位ビットを Z_U とすると $Z = 2^{n_i} Z_\mathrm{U} + Z_\mathrm{L}$ となり，$2^{n_i} \equiv \alpha \bmod p_i$ より $Z \equiv Z_\mathrm{L} + \alpha Z_\mathrm{U} \bmod p_i$ であるから，

$$\text{手続き: } Z_\mathrm{L} \leftarrow Z\text{の下位 } n_i \text{ ビット}; Z_\mathrm{U} \leftarrow Z\text{の上位ビット}; Z \leftarrow Z_\mathrm{L} + \alpha Z_\mathrm{U}; \tag{4.4}$$

を，Z が n_i ビットに納まるまで反復する．もし $Z < p_i$ ならば Z が，さもなければ $Z - p_i$ が基本剰余演算の結果 $Z\%p_i$ である．上記の手続きの反復回数は，一般に $p > \alpha^2$ ならば 3 回以下であることが知られている [34]．また実際には α を掛ける操作は乗算を行なうのではなく α の 2 進数表記を利用してビットシフトを組み合わせる．例えば $\alpha_i = 1$ のときには実際には α_i を掛ける必要はない．また，例えば $\alpha_i = 3$ のときには，3 は 2 進数で $11_{(2)}$ であ

るから, Z_U を 1 ビット左にシフトしたものを $(Z_U << 1)$ とかくと $Z_L + \alpha Z_U$ のかわりに $Z_L + (Z_U << 1) + Z_U$ とすることができ, この部分で乗算を回避できる. したがって, α_i として 2 進数表記が簡単なものを選んだほうが計算は簡単になる.

また, $p_i = 2^{n'}$ で積 $p_i p_j$ が単長 (n ビット) に納まるなら, まず, $Z\%(p_i p_j)$ を計算し, それから $Z\%p_i$, $Z\%p_j$ を求めることで計算時間を短縮することもできる. 具体的には,

$$Z\%p_i = Z\%(p_i p_j) \text{の下位} n' \text{ビット}, \tag{4.5}$$

$$Z\%p_j = (Z\%(p_i p_j))\%p_j \tag{4.6}$$

である. このようにして計算時間を短縮する方式を**複合方式**とよび, $Z\%p_i$, $Z\%p_j$ を直接求める方式を**個別方式**とよぶことにする.

幾何的アルゴリズムで, 入力変数 $x = (x_1, \cdots, x_{n'})$ に入力データ $a \in \mathbf{Z}^{n'}$ を代入し, 基本演算を重ねて判定式 $f(x)$ の値が多倍長整数 $X' = f(a)$ として得られるとする. $f(x)$ は整数係数多項式である. このとき X' の正負によりアルゴリズム中の処理が分岐する. ここで, $-p/2 \leq f(a) < p/2$ であると見積られているとする.

剰余演算を利用する方法では, X' は直接には得られず, 各 p_i $(i = 0, \cdots, m)$ による剰余 $X_i = X'\%p_i$ が得られている. 以下では, 二つの整数がともに偶数あるいは奇数であるとき, その二つの整数の**パリティ** (奇偶性) が一致するということにする.

問題を一般形にすると次のようになる:

問題 1 いくつかの自然数 X_i $(0 \leq X_i < p_i)$ が与えられたとき,

$$X' \equiv X_i \bmod p_i \tag{4.7}$$

を満たす整数 X' $(-p/2 \leq X' < p/2)$ の正負を単長計算で求めよ.

$X = X'\%p$ とする. $X_i = X\%p_i$ が成立する. また, 次の命題が成り立つので, X と $p/2$ の大小関係が求められれば X' の正負は求められる.

$$X' \geq 0 \quad \Leftrightarrow \quad X < p/2, \tag{4.8}$$

$$X' < 0 \quad \Leftrightarrow \quad X \geq p/2. \tag{4.9}$$

以後は X と $p/2$ の大小関係を求めることにする. X を特徴付ける次の定理が役立つ.

定理 5 (中国剰余定理)
p_i $(i = 0, \cdots, m)$ を互いに素な自然数とする. 変数 X の連立合同式:

$$X \equiv X_i \bmod p_i \tag{4.10}$$

の一般解は

$$X \equiv X_0 u_0 q_0 + X_1 u_1 q_1 + \cdots + X_m u_m q_m \bmod p \tag{4.11}$$

である. ただし, p, q_i, u_i は $p = p_0 \cdots p_m$, $q_i = p/p_i$, $u_i q_i \equiv 1 \bmod p_i$ を満たす整数である.

まず, 数倍長程度の議論を個別にした後, 一般の m 倍長の場合について述べる.

4.2 3倍長版 (, 2倍長版)

まず, 3倍長程度の整数の符号判定法について述べる. $m = 4$ とし, $p_0 = 2, p_1 = 2^n - 1, p_2 = 2^{n-1} - 1, p_3 = 2^n - 3$ とおく. $p = p_0p_1p_2p_3 = (2^n - 1)(2^n - 2)(2^n - 3)$ であり, 2^{3n} に近い. すなわち X' のとり得る値は3倍長程度である. 後のために, $\tilde{p} = p_1p_2p_3 (= p/2)$ としておく.

$$X \equiv X_1p_2p_3 - X_2p_1p_3 + X_3p_1p_2 \bmod \tilde{p} \text{ (中国剰余定理)}. \tag{4.12}$$

右辺を Y とおく: $Y = X_1p_2p_3 - X_2p_1p_3 + X_3p_1p_2$. すると

$$X \% \tilde{p} \,(= Y \% \tilde{p}) = Y - [Y/\tilde{p}]\tilde{p}. \tag{4.13}$$

が成立する. このとき $[Y/\tilde{p}]$ は -1 か 0 か 1 を値にとり, その値を求める方法がある. 以下にその方法を示す.

$$[Y/\tilde{p}] = -1 \quad \Leftrightarrow \quad Y < 0, \tag{4.14}$$
$$[Y/\tilde{p}] = 0 \quad \Leftrightarrow \quad 0 \le Y < \tilde{p}, \tag{4.15}$$
$$[Y/\tilde{p}] = 1 \quad \Leftrightarrow \quad \tilde{p} \le Y \tag{4.16}$$

である. Y を p_1p_3 で割って2倍して,

$$[Y/\tilde{p}] = -1 \quad \Leftrightarrow \quad 2Y/p_1p_3 < 0, \tag{4.17}$$
$$[Y/\tilde{p}] = 0 \quad \Leftrightarrow \quad 0 \le 2Y/p_1p_3 < 2p_2, \tag{4.18}$$
$$[Y/\tilde{p}] = 1 \quad \Leftrightarrow \quad 2p_2 \le 2Y/p_1p_3 \tag{4.19}$$

を得る. $2p_2 = p_1 - 1 = p_3 + 1$ を利用して変形すると,

$$2Y/p_1p_3 = X_1 - 2X_2 + X_3 - \frac{X_1p_3 - X_3p_1}{p_1p_3} \tag{4.20}$$

であり, $-1 < \dfrac{X_1p_3 - X_3p_1}{p_1p_3} < 1$ であるので, 整数性を考慮して,

$$[Y/\tilde{p}] = -1 \quad \Leftrightarrow \quad X_1 - 2X_2 + X_3 - \epsilon_1 < 0, \tag{4.21}$$
$$[Y/\tilde{p}] = 0 \quad \Leftrightarrow \quad 0 \le X_1 - 2X_2 + X_3 - \epsilon_1 < 2p_2, \tag{4.22}$$
$$[Y/\tilde{p}] = 1 \quad \Leftrightarrow \quad 2p_2 \le X_1 - 2X_2 + X_3 - \epsilon_1 \tag{4.23}$$

ただし,

$$\epsilon_1 = \begin{cases} 1 & (X_1p_3 - X_3p_1 > 0), \\ 0 & (\text{otherwise}) \end{cases} \tag{4.24}$$

を得る. ϵ_1 が決まれば, 比較方法を工夫すると, $[Y/\tilde{p}]$ の値を単長の範囲の計算と比較で求めることができる. したがって, 残っているのは, $X_1p_3 - X_3p_1 > 0$ かどうかの判定だけである. これも, 整数性と $p_1 = p_3 + 2$ であることを利用すると同値な式

$$X_2 > X_4 + \epsilon_2, \text{ ただし, } \epsilon_2 = \begin{cases} 1 & (X_4 \ge 2^{n-2} - 1) \\ 0 & (\text{otherwise}) \end{cases} \tag{4.25}$$

を得る. これですべてを単長内に収めたことになる. $[Y/\tilde{p}]$ の値を求める方法をまとめると,

$$\epsilon_2 = \begin{cases} 1 & (X_4 \geq 2^{n-2} - 1) \\ 0 & (\text{otherwise}) \end{cases}, \epsilon_1 = \begin{cases} 1 & (X_2 > X_4 + \epsilon_2) \\ 0 & (\text{otherwise}) \end{cases} \tag{4.26}$$

として

$$[Y/\tilde{p}] = \begin{cases} -1 & (2X_3 + \epsilon_1 > X_2 + X_4) \\ 0 & (2X_3 + \epsilon_1 \leq X_2 + X_4 < 2X_3 + 2p_3 + \epsilon_1) \\ 1 & (X_2 + X_4 \geq 2X_3 + 2p_3 + \epsilon_1) \end{cases} \tag{4.27}$$

になる.

すなわち, $[Y/\tilde{p}]$ のパリティを求められる. 次の定理が成り立つ.

定理 6
X' が非負であるための必要十分条件は X と X_0 のパリティが一致することである.

証明. X_0 と X' のパリティは一致する. X' が非負, すなわち $0 \leq X' < \tilde{p}$ のとき, $X' = X$ であり, X' と X のパリティは一致する. また, X' が負, すなわち $-\tilde{p} \leq X' < 0$ のとき, $X' = X - \tilde{p}$ で $\tilde{p}$ は奇数であるから, X' と X のパリティは一致しない. 従って X' が非負のときのみ X と X_0 のパリティが一致する. (証明終)

また, $X' = 0$ であるための必要十分条件は $X_0 = X_1 = X_2 = X_3 = 0$ であるから, 上の定理と合わせて X' の符号判定ができる.

$p_1, p_2, p_3, \tilde{p}$ はすべて奇数で, $[Y/\tilde{p}]$ のパリティも求まるので $X\%\tilde{p}$ $(= X_1p_2p_3 - X_2p_1p_3 + X_3p_1p_2 - [Y/\tilde{p}]\tilde{p})$ のパリティを求めることができ, X' の符号が求められる .

また, 式

$$X \equiv -2X_1p_2 - X_2p_1 \bmod p_1p_2 \text{ (中国剰余定理)} \tag{4.28}$$

から同様にして 2 倍長版を求めることができる. すなわち, まず, 上式の右辺を Y とする: $Y = -2X_1p_2 - X_2p_1$. $[Y/p_1p_2]$ が奇数になるのは $0 < X_1 - X_2 \leq p_2$ のときに限ることは容易に導ける. 次に式

$$X\%(p_1p_2) = -2X_1p_2 + X_2p_1 - [Y/p_1p_2]p_1p_2 \tag{4.29}$$

より, $[Y/p_1p_2]$ のパリティがわかれば $X\%(p_1p_2)$ のパリティがわかり, X_0 のパリティと一致すれば X' は非負, 一致しなければ X' は正になるというものである.

しかし, この方法より, 後でに述べる m 倍長版 (1) の $m = 2$ の場合の方がより単純なので, 本論文では概略を述べるにとどめる.

これらの方法は X_0 $(= X\%p_0)$ と $X = X\%\tilde{p}$ を利用する方法であり, $p_0 = 2$ であるため n ビットを使い切れていないともいえる. このため, p_0 側の桁数を延ばしたのが 4 倍長版, 5 倍長版である.

4.3 4倍長版, 5倍長版

次に4倍長版について述べる.

$p_0 = 2^n, p_1 = 2^n - 1, p_2 = 2^{n-1} - 1, p_3 = 2^n - 3$ とする. また, $\tilde{p} = p_1 p_2 p_3$ とする. p は 2^{4n-1} 弱である.

p_1, p_2, p_3 に関しては3倍長版と同様である:

$$X \equiv X_1 p_2 p_3 - X_2 p_1 p_3 + X_3 p_1 p_2 \bmod \tilde{p} \text{ (中国剰余定理)}. \tag{4.30}$$

右辺を Y とおく: $Y = X_1 p_2 p_3 - X_2 p_1 p_3 + X_3 p_1 p_2$.

$$X\%\tilde{p} \,(= Y\%\tilde{p}) = Y - [Y/\tilde{p}]\tilde{p}. \tag{4.31}$$

このとき $[Y/\tilde{p}]$ は -1 か 0 か 1 で, 求める方法もあることを3倍長版の説明で示した.

次の命題が成り立つので, $[X/\tilde{p}]$ が求められれば X' の符号が求められる .

$$X < p/2 \;\Leftrightarrow\; [X/\tilde{p}] < 2^{n-1}, \tag{4.32}$$
$$X \geq p/2 \;\Leftrightarrow\; [X/\tilde{p}] \geq 2^{n-1}. \tag{4.33}$$

また, $\tilde{p} \equiv 2^{n-1} - 3 \bmod p_0$ より,

$$X = [X/\tilde{p}]\tilde{p} + X\%\tilde{p} \equiv (2^{n-1} - 3)[X/\tilde{p}] + X\%\tilde{p} \bmod p_0. \tag{4.34}$$

この合同式を $[X/\tilde{p}]$ について解く.

$$[X/\tilde{p}] \equiv C(X - X\%\tilde{p}) \bmod p_0 \tag{4.35}$$
$$\equiv C(X_0 - X\%\tilde{p}) \bmod p_0. \tag{4.36}$$
$$\tag{4.37}$$

ただし C は $(2^{n-1} - 3)C \equiv 1 \bmod p_0$ となる自然数であり, p_0 未満にとることができる. n が4の倍数の時, 具体的には, $C = p_0 - 2[(2^{n-1} - 3)/6] - 3$ になる.

$0 \leq [X/\tilde{p}] < p_0$ であるから,

$$[X/\tilde{p}] = (C(X_0 - X\%\tilde{p}))\%p_0 \tag{4.38}$$
$$= (C(X_0 - X_1 p_2 p_3 + X_2 p_1 p_3 - X_3 p_1 p_2 + [Y/\tilde{p}]\tilde{p}))\%p_0 \tag{4.39}$$
$$= (C(X_0 + X_1(p_2 - 2) + 3X_2 + X_3 p_2 + [Y/\tilde{p}](p_2 - 2)))\%p_0. \tag{4.40}$$

$p_0 = 2^n$ であるから, 実質的には桁あふれを無視して計算するだけでよく, p_0 による剰余計算をする必要はない. これで $[X/\tilde{p}]$ が求まり X' の符号が求められる .

この方法は $X_0\,(= X\%p_0)$ と $X\%\tilde{p}$ を利用する方法である. $p_0 = 2^n$ であるため X_0 の方も n ビットを使い切っているが, $X\%\tilde{p}$ の方は $\tilde{p}$ が約 $3n-1$ ビットあり, バランスを考えると p_0 の方のビットを増やせると予想できる. これを増やしたのが5倍長版である.

次に5倍長版について述べる. $p_0 = 2^n, p_1 = 2^n - 3, p_2 = 2^{n-1} - 1, p_3 = 2^{n-2} - 1, p_4 = 2^{n-1} - 3$ とする. 今度は $\tilde{p} = p_2 p_3 p_4$ とする. p は 2^{5n-4} 弱である.

p_2, p_3, p_4 に対して 3 倍長版, 4 倍長版の p_1, p_2, p_3 と同様の議論をする.

$$X \equiv X_2 p_3 p_4 - X_3 p_2 p_4 + X_4 p_2 p_3 \bmod \tilde{p} \text{ (中国剰余定理)}. \tag{4.41}$$

右辺を Y とおく:$Y = X_2 p_3 p_4 - X_3 p_2 p_4 + X_4 p_2 p_3$.

$$X\%\tilde{p}\ (= Y\%\tilde{p}) = Y - [Y/\tilde{p}]\tilde{p}. \tag{4.42}$$

このとき $[Y/\tilde{p}]$ は -1 か 0 か 1 で, 求める方法もある. ここまでは 4 倍長版と同じである.

次の命題が成り立つので, $[[X/\tilde{p}]/p_1]$ が求められれば X' の符号が求められる .

$$\begin{aligned}
X < p/2 &\Leftrightarrow [X/\tilde{p}] < p_0 p_1/2 && (4.43)\\
&\Leftrightarrow [[X/\tilde{p}]/p_1] < p_0/2, && (4.44)\\
X \geq p/2 &\Leftrightarrow [X/\tilde{p}] \geq p_0 p_1/2 && (4.45)\\
&\Leftrightarrow [[X/\tilde{p}]/p_1] \geq p_0/2. && (4.46)\\
& && (4.47)
\end{aligned}$$

また, $X = [X/\tilde{p}]\tilde{p} + X\%\tilde{p}$ であるから,

$$X_0 \equiv [X/\tilde{p}]\tilde{p} + X\%\tilde{p} \bmod p_0,\ X_1 \equiv [X/\tilde{p}]\tilde{p} + X\%\tilde{p} \bmod p_1. \tag{4.48}$$

これらを $[X/\tilde{p}]$ について解く.

$$[X/\tilde{p}] \equiv C_0(X_0 - X\%\tilde{p}) \bmod p_0,\ [X/\tilde{p}] \equiv C_1(X_1 - X\%\tilde{p}) \bmod p_1. \tag{4.49}$$

ただし C_0, C_1 は $C_0\tilde{p} \equiv 1 \bmod p_0$, $C_1\tilde{p} \equiv 1 \bmod p_1$ となる自然数である. n が 4 の倍数のとき, 具体的には,

$$\begin{aligned}
C_0 &= ([2^n/3](2^{n-2}+1))\%2^n, && (4.50)\\
C_1 &= (16([8(2^n-3)/15]+1))\%(2^n-3) && (4.51)
\end{aligned}$$

とすればよい.

$$\begin{aligned}
[X/\tilde{p}]\%p_0 &= (C_0(X_0 - X\%\tilde{p}))\%p_0, && (4.52)\\
[X/\tilde{p}]\%p_1 &= (C_1(X_1 - X\%\tilde{p}))\%p_1. && (4.53)
\end{aligned}$$

ここで,

$$\begin{aligned}
[X/\tilde{p}] &= [[X/\tilde{p}]/p_1]p_1 + [X/\tilde{p}]\%p_1 && (4.54)\\
&\equiv -3[[X/\tilde{p}]/p_1] + [X/\tilde{p}]\%p_1 \bmod p_0 && (4.55)
\end{aligned}$$

であるから, $[[X/\tilde{p}]/p_1]$ について解く.

$$\begin{aligned}
[[X/\tilde{p}]/p_1] &\equiv D([X/\tilde{p}]\%p_1 - [X/\tilde{p}]) \bmod p_0 && (4.56)\\
&\equiv D([X/\tilde{p}]\%p_1 - [X/\tilde{p}]\%p_0) \bmod p_0 && (4.57)
\end{aligned}$$

$$\epsilon_2 \leftarrow \begin{cases} 1 & (X_4 \geq 2^{n-2}-1) \\ 0 & (\text{otherwise}) \end{cases} \tag{4.59}$$

$$\epsilon_1 \leftarrow \begin{cases} 1 & (X_2 > X_4+\epsilon_2) \\ 0 & (\text{otherwise}) \end{cases} \tag{4.60}$$

$$T \leftarrow \begin{cases} -1 & (2X_3+\epsilon_1 > X_2+X_4) \\ 0 & (2X_3+\epsilon_1 \leq X_2+X_4 < 2X_3+2p_3+\epsilon_1) \\ 1 & (X_2+X_4 \geq 2X_3+2p_3+\epsilon_1) \end{cases} \tag{4.61}$$

$$T_0 \leftarrow (X_2p_3p_4 - X_3p_2p_4 + X_4p_2p_3 - T\tilde{p})\%p_0(\text{剰余計算}) \tag{4.62}$$

$$T_1 \leftarrow (X_2p_3p_4 - X_3p_2p_4 + X_4p_2p_3 - T\tilde{p})\%p_1(\text{剰余計算}) \tag{4.63}$$

$$Z_0 \leftarrow (C_0(X_0 - T_0))\%p_0,(\text{剰余計算}) \tag{4.64}$$

$$Z_1 \leftarrow (C_1(X_1 - T_1))\%p_1,(\text{剰余計算}) \tag{4.65}$$

$$Z \leftarrow (D(Z_1 - Z_0))\%p_0.(\text{剰余計算}) \tag{4.66}$$

$Z < p_0/2$ ならば $X' \geq 0$ であり, $Z \geq p_0/2$ ならば $X' < 0$ である.

図 4.2: 剰余を利用した 5 倍長整数の符号判定

ただし D は $3D \equiv 1 \mod p_0$ を満たす整数である. n が 4 の倍数のとき, 具体的には $D = [2^n/3]$ である. $[[X/\tilde{p}]/p_1] < p_0$ であるから結局,

$$[[X/\tilde{p}]/p_1] = (D([X/\tilde{p}]\%p_1 - [X/\tilde{p}]\%p_0))\%p_0. \tag{4.58}$$

これで $[[X/\tilde{p}]/p_1]$ が求まり X' の符号が求められる. 2 倍長版から 5 倍長版までのまとめとして 5 倍長版を手続き的に記述すると次の図 4.2のようになる.

実際には p_0 を法とする剰余計算は桁あふれを無視するだけで実行できる.

$p_0 = 2^n, p_1 = 2^{n-1}-3, p_2 = 2^n-1, p_3 = 2^{n-1}-1, p_4 = 2^n-3$ とすることによって p を 2^{5n-2} 弱にすることができ 2 ビット稼げるが, $[Y/\tilde{p}]$ の値を求めるのが大変になる (不可能ではない). また, 4 倍長版から 5 倍長版にする技術は汎用性がある. これを利用すると 6 倍長版, 7 倍長版も作ることができる. そのときには $X\%\tilde{p}$ の役割は少なくなってくる. 4 倍長版から 5 倍長版にする技術をそのまま使ったのが m 倍長版 (1) である.

4.4 m 倍長版 (1)

$p_1, \cdots, p_{m-1}$ を互いに素な m 個の n ビットの正の奇数にとり, $p_m = 2^n$ にとる. $p = p_1 \cdots p_m$ は 2^{mn} 弱である.

次の命題が成り立つ.

$$X' \geq 0 \Leftrightarrow X < p/2 = p_1 \cdots p_m/2 \tag{4.67}$$

$$\Leftrightarrow [X/p_1] < p_2 \cdots p_m/2 \tag{4.68}$$

$$\Leftrightarrow [[X/p_1]/p_2] < p_3 \cdots p_m/2 \tag{4.69}$$

$$\cdots \tag{4.70}$$

$$\Leftrightarrow \quad [\cdots[[X/p_1]/p_2]\cdots/p_{m-1}] < p_m/2. \tag{4.71}$$

したがって, $[\cdots[[X/p_1]/p_2]\cdots/p_{m-1}]$ の値が求まれば X' の符号が求められる .

$Q_k = [\cdots[[X/p_1]/p_2]\cdots/p_{k-1}]$ とおく.

$$Q_{k+1} = [Q_k/p_k],\ Q_1 = X \tag{4.72}$$

であり, $Q_m = [\cdots[[X/p_1]/p_2]\cdots/p_{m-1}]$ であるから, Q_m が求まれば X' の符号が求められる .

$$\begin{aligned} Q_k &= [Q_k/p_k]p_k + Q_k\%p_k & (4.73)\\ &= Q_{k+1}p_k + Q_k\%p_k & (4.74)\end{aligned}$$

より, 各 p_j $(j = k+1, \cdots, m)$ で剰余をとって,

$$Q_k\%p_j \equiv (Q_{k+1}\%p_j)p_k + Q_k\%p_k \bmod p_j. \tag{4.75}$$

$Q_{k+1}\%p_j$ について解いて,

$$Q_{k+1}\%p_j \equiv q_{kj}(Q_k\%p_j - Q_k\%p_k) \bmod p_j. \tag{4.76}$$

ここで, q_{kj} は $p_k q_{kj} \equiv 1 \bmod p_j$ を満たす自然数で, あらかじめ計算しておく. この式を書き換え,

$$Q_{k+1}\%p_j = (q_{kj}(Q_k\%p_j - Q_k\%p_k))\%p_j \ (j = k+1, \cdots, m) \tag{4.77}$$

を得る. この漸化式により X_i $(= Q_1\%p_i)$ $(i = 1, \cdots, m)$ から Q_m $(= Q_m\%p_m)$ を得ることができ, X' の符号が求められる . この計算手順を図示する. $Q_k\%p_j$ を (k, j) と略記する. この漸化式は $(k, j), (k, k)$ から $(k+1, j)$ が求められることを示している. これを

$$\begin{array}{ll} (k+1, j) & \leftarrow (k, j) \\ & \nwarrow (k, k) \end{array} \tag{4.78}$$

と書くことにする. $(1,1), (1,2), \cdots, (1,m)$ から (m,m) が求まれば X' の符号が求められるわけだが, 実際, 次の図 4.3 ような手順で X' の符号が求められる .

$$\begin{array}{lllll} (m,m) & \leftarrow (m-1,m) & \leftarrow (m-2,m) & \cdots & \leftarrow (1,m) \\ & \nwarrow (m-1,m-1) & \leftarrow (m-2,m-1) & \cdots & \leftarrow (1,m-1) \\ & & \nwarrow (m-2,m-2) & \cdots & \leftarrow (1,m-2) \\ & & & \ddots & \vdots \\ & & & & \nwarrow (1,1) \end{array}$$

図 4.3: m 倍長版 (1) の計算手順

計算時間, 記憶量ともに $O(m^2)$ になっている. ただし, $Q_k\%p_k = Q_k\%p_j$ $(j = k+1, \cdots, m)$ となった時点で $Q_k = Q_k\%p_k$ であることがわかるので, アルゴリズムの実行をここで打ち切ることができる. この方法が商 Q_k $(= [X/p_1 \cdots p_{k-1}])$ を順に求めていく方法なのに対し, 剰余 $X\%(p_1 \cdots p_k)$ を順に求めていく方法が次の m 倍長版 (2) である.

4.5 m 倍長版 (2)

m 倍長版 (1) と同じく, 互いに素な n ビットの正の奇数に $p_1, \cdots, p_{m-1}$ をとり, $p_m = 2^n$ にとる. $p = p_1 \cdots p_m$ は 2^{mn} 弱である. $p_{11} = 1$ とし, $j \geq 2$ のとき, $p_{1j} = p_1 \cdots p_{j-1}$ と定める.

$Y_1 = 0$ とし, $Y_2, \cdots, Y_{m+1}$ を

$$Y_j \equiv X_i \bmod p_i \ (i = 1, \cdots, j-1),\ 0 \leq Y_j < p_{1j} \tag{4.79}$$

を満たす整数とする. すなわち, $Y_j = X\%p_{1j}$ である. 特に, $Y_{m+1} = X$ であるが使わない. 次の命題が成り立つ.

$$X' \geq 0 \ \Leftrightarrow \ X < p/2 = p_{1m}p_m/2 \tag{4.80}$$

$$\Leftrightarrow \ [X/p_{1m}] < p_m/2. \tag{4.81}$$

したがって, $[X/p_{1m}]$ の値が求まれば X' の符号が求められる.

ところが $X = [X/p_{1m}]p_{1m} + Y_m$ であるから, この式の両辺を p_m で剰余をとり,

$$X \equiv [X/p_{1m}]p_{1m} + Y_m \bmod p_m, \tag{4.82}$$

を得る. これを $[X/p_{1m}]$ について解いて,

$$[X/p_{1m}] \equiv c_m(X_m - Y_m) \bmod p_m. \tag{4.83}$$

を得る. ただし, c_m は $c_m p_{1m} \equiv 1 \bmod p_m$ を満たす整数である. $0 \leq [X/p_{1m}] < p_0$ であるから,

$$[X/p_{1m}] = (c_m(X_m - Y_m))\%p_m. \tag{4.84}$$

である. したがって $Y_m\%p_m$ の値が求まれば X' の符号が求められる. 以後はこれを求めることに集中する.

まず, c_j $(j = 1, \cdots, m)$ を $c_j p_{1j} \equiv 1 \bmod p_j$ を満たす整数と定めておく. これは上の c_m の定義と矛盾しない.

$j \geq 2$ のとき, $Y_j \equiv X_i \bmod p_i$ $(i = 1, \cdots, j-1)$ より, $X - Y_j \equiv 0 \bmod p_i$ $(i = 1, \cdots, j-1)$ である. 各 p_i が互いに素であるので, この式は, $X - Y_j$ が p_{1j} の倍数であることを示している:

$$X - Y_j = a p_{1j}. \tag{4.85}$$

p_j で剰余をとってから, a について解くと

$$a \equiv c_j(X_j - Y_j) \tag{4.86}$$

$$\equiv c_j(X_j - Y_j\%p_j) \bmod p_j \tag{4.87}$$

を得る. $A = (c_j(X_j - Y_j\%p_j))\%p_j$ とすると, $0 \leq A < p_j$ なので,

$$Y_{j+1} = Y_j + A p_{1j} \tag{4.88}$$

になる．実際，直接計算によって $0 \le Y_{j+1} < p_{1j+1}$, $Y_j \equiv X \bmod p_i$ $(i = 1, \cdots, j)$ が成り立つことがわかる．さらに，この式は，$j = 1$ のときにも成立する．以下では $j \ge 1$ とする．p_k $(k = j+1, \cdots, m)$ で剰余をとる:

$$Y_{j+1} \% p_k \equiv Y_j \% p_k + ((c_j(X_j - Y_j \% p_j)) \% p_j) p_{1j} \bmod p_k \ (k = j+1, \cdots, m). \tag{4.89}$$

実際の計算では $p_{1j} \% p_k$ をあらかじめ計算しておく．この式は $Y_j \% p_k$ と $Y_j \% p_j$ から $Y_{j+1} \% p_k$ が得られることを表している．$Y_1 = 0$ なので $Y_1 \% p_k = 0$ $(k = 1, \cdots, m)$ である．これにより $Y_m \% p_m$ が求められるので，X' の符号が求められる．

この手順を図示する．$Y_j \% p_k$ を (j, k) と略記する．この漸化式は $(j, j), (j, k)$ から $(j+1, k)$ が求められることを示している．これを

$$\begin{array}{ll} (j+1, k) & \leftarrow (j, k) \\ & \nwarrow (j, j) \end{array}$$

と書くことにする．$(1,1), (1,2), \cdots, (1,m)$ から (m,m) が求まれば X' の符号が求められるわけだが，実際，次の図 4.4 ような手順で X' の符号が求められる．

$$\begin{array}{lllll} (m,m) & \leftarrow (m-1,m) & \leftarrow (m-2,m) & \cdots & \leftarrow (1,m) \\ & \nwarrow (m-1,m-1) & \leftarrow (m-2,m-1) & \cdots & \leftarrow (1,m-1) \\ & & \nwarrow (m-2,m-2) & \cdots & \leftarrow (1,m-2) \\ & & & \ddots & \vdots \\ & & & & \nwarrow (1,1) \end{array}$$

図 4.4: m 倍長版 (2) の計算手順

計算時間，記憶量ともにやはり $O(m^2)$ になっている．ただし，Y_m が小さい $(Y_m < p_{1j})$ 時には，$Y_j \equiv X_k \bmod p_k$ $(k = j, \cdots, m)$ となるので，Y_j を求めた時点でただちに $Y_m = Y_j$ であることがわかり，このアルゴリズムを途中で終了させることができる．m 倍長版 (1), m 倍長版 (2) とも，X の値が小さいときにはアルゴリズムは早く終了する．$-X'$ に同じアルゴリズムを適用すると，今度は $p - X$ の値が小さいときに早く終了する．両方同時に考えると，m 倍長版 (1), m 倍長版 (2) は $|X'|$ が小さいときに早く終わるアルゴリズムである．次の m 倍長版 (3) は逆に $|X'|$ が大きいときに早く終わるという特徴をもつ．

4.6 m 倍長版 (3)

m 倍長版 (1), m 倍長版 (2) とは異なり，互いに素な n ビットの正の奇数に $p_1, \cdots, p_m$ をとり，$p_0 = 2^n$ にとる．m の上限を $m \le 2^{(n-1)/2}$ に限定する．制限は付くが，応用上は十分に役立つ大きさである．$P = p_0 \cdots p_m$ は $2^{(m+1)n}$ 弱である．$p = p_1 \cdots p_m (= P/p_0)$ とする．ここでは X' の範囲を $-P/2 \le X' < P/2 - mp$ とする．mp を引いた理由は後で示す．$(m < 2^{(n-1)/2} < p_0/2$ であるから，$P/2 - mp > (p_0/2 - m)p \ge p$ であり，X' の範囲は m 倍長を余裕をもって確保している．)

次の命題が成り立つ.

$$X \geq 0 \quad \Leftrightarrow \quad X < p_0 p/2 \tag{4.90}$$

$$\Leftrightarrow \quad [X/p] < p_0/2. \tag{4.91}$$

$$\tag{4.92}$$

整数 q_i $(i = 1, \cdots, m)$ を $q_i = p/p_i$ によって定め, 整数 u_i を $u_i q_i \equiv 1 \bmod p_i$ を満たすように定める.

これまでによく出てきた式

$$X = [X/p]p + X\%p \tag{4.93}$$

をもう一度振り返る. この式を p_0 で剰余をとり

$$X_0 \equiv [X/p]p\%p_0 + (X\%p)\%p_0 \bmod p_0 \tag{4.94}$$

を得る. $(X\%p)\%p_0$ が得られれば $[X/p]$ は得られそうである. そこで中国剰余定理 (定理 5) をみる:

$$X \equiv X_1 u_1 q_1 + X_2 u_2 q_2 + \cdots + X_m u_m q_m \bmod p. \tag{4.95}$$

この式は若干変更できる:

$$X\%p \equiv ((X_1 u_1)\%p_1)q_1 + ((X_2 u_2)\%p_2)q_2 + \cdots + ((X_m u_m)\%p_m)q_m \bmod p. \tag{4.96}$$

$Y_i = (X_i u_i)\%p_i$ とすると

$$X\%p \equiv Y_1 q_1 + Y_2 q_2 + \cdots + Y_m q_m \bmod p \tag{4.97}$$

を得る. 右辺を Y とおく. Y の大きさを評価する. $0 \leq Y_i < p_i$, $p_i q_i = p$ であるから

$$Y = Y_1 q_1 + Y_2 q_2 + \cdots + Y_m q_m < p + p + \cdots + p = mp \tag{4.98}$$

を得る. すなわち $X\%p$ と Y は p で整除した剰余は一致し, 大きさは最も異なるときでも mp 未満である. そこで $X\%p$ を Y で近似する. $X \equiv Y \bmod p$ より, $X\%p = Y\%p$ である. これより,

$$X = [X/p]p + X\%p \tag{4.99}$$

$$Y = [Y/p]p + Y\%p \tag{4.100}$$

の両辺を引くと,

$$X - Y = ([X/p] - [Y/p])p \tag{4.101}$$

が成立する. p_0 で剰余をとってから, $[X/p]$ について解くと,

$$[X/p] \equiv c(X_0 - Y) + [Y/p] \bmod p_0 \tag{4.102}$$

を得る. ただし c は $cp \equiv 1 \bmod p_0$ を満たす整数で, あらかじめ計算しておく. 式 4.98から, $0 \leq [Y/p] \leq m - 1$ なので, 次のことがわかる.

$$((c(X_0 - Y))\%p_0 < p_0/2 \quad \Rightarrow \quad [X/p] < p_0/2, \tag{4.103}$$

$$((c(X_0 - Y) + m - 1)\%p_0 \geq p_0/2 \quad \Rightarrow \quad [X/p] \geq p_0/2. \tag{4.104}$$

したがって，この2通りの場合には X' の符号が求められる．($X' < P/2 - mp$ という制限を加えたのはこのためである．)

残りの場合:

$$(c(X_0 - Y))\%p_0 \geq p_0/2 \text{ かつ } (c(X_0 - Y) + m - 1)\%p_0 < p_0/2 \tag{4.105}$$

を考える．このとき，

$$-mp < X' < (m-1)p \tag{4.106}$$

である．m' を，$2^{(n-1)/2}$ を超えない最大の整数とすると，m の定義より $m \leq m'$，$m'm \leq 2^{(n-1)}$ が成り立つ．上の不等式の各辺を m' 倍すると $-m'mp < m'X' < m'(m-1)p$ である．これを評価して

$$-p_0p/2 = -2^{n-1}p < m'X' < 2^{n-1}p - mp = p_0p/2 - mp \tag{4.107}$$

を得る．これは $m'X'$ が再び符号判定問題の値の範囲に入ることを意味する．ここで，$m'X' \equiv m'X_i \bmod p_i$ であるから，$m'X'$ を，このアルゴリズムに適用するには，入力を $X_i \leftarrow (m'X_i)\%p_i$ とするだけでよい．これで符号判定ができないときには，m' 倍してアルゴリズムを適用することを繰り返す．

$$m'^{3m+1} = m'(m'^2)^{1.5m} > m(2^{n-2})^{\frac{n}{n-2}m} = m2^{mn} > mp \tag{4.108}$$

より，$3m+1$ 回以内の繰り返しで必ず符号判定ができ X' の符号が求められる．

計算時間は $O(m^2)$，記憶量は $O(m)$ になっている．ただし，繰り返す回数は $|X'|$ が大きいほど少ないといえる．

4.7 剰余を利用した方法の計算時間

新規に3種類の m 倍長版の剰余からの符号判定アルゴリズムを提案した．アルゴリズムの計算時間は $O(m^2)$，記憶量は $O(m^2)$ と $O(m)$ になっている．

よく考えると，単長入力の下で m 倍長計算を必要とすることは，乗算が $O(m)$ 回あるいは加減算が $O(m^2)$ 回あることを意味する．このための計算時間は，可変多倍長計算をしないかぎり，全体ですでに $O(m^2)$ 以上かかるであろうから，符号判定に $O(m^2)$ より少ない計算時間を要求する必然性は実用上はすくないといえる．しかしながら理論的には剰余 X_i から元の値 X を求めるのに再帰二分化と高速乗算法の組み合わせで $O(m \log m \log\log m)$ が達成されている．したがって，残念ながら本論文で提案した方法はどれも理論的には計算時間で優位ではない．これらの方法を再帰二分化し，計算時間の $O(m \log m)$ にできる可能性もあるが，それは今後の課題である．

本論文で提案した方法は多倍長計算を必要としない分，単純なアルゴリズムになっている．これは実装の容易さと実計算時間の短縮につながる．2倍長版と3倍長版で高速化の予備的な実験を行なった結果を次節で述べる．

4.8 剰余計算の計算幾何学への予備的な適用実験

計算量が理論的に小さいオーダでも，実装時に高速性を発揮できるとは限らない．特に，この，m 倍長整数の計算と符号判定に関しては，幾何的アルゴリズムの特徴で m は数十から数百程度の場合が少なくないので，なおさらである．

具体的な問題で計算時間を比較するため，凸包構成アルゴリズムと単純多角形集合生成アルゴリズムに，多倍長演算と剰余演算を利用した方法を適用した．数値計算部分を2倍長演算，3倍長演算，剰余演算の2倍長版，3倍長版のそれぞれで実装したプログラムを用意した．使用した言語はC，使用した計算機は富士通S-4/ECである．この計算機のCPUはRISC型のSPARCチップであり，本来の単長整数のビット長32ビットを2倍長とみなし，多倍長演算と剰余演算はCの関数として自作した．

剰余計算は p_0, p_2 による剰余を求めるときに個別に求める個別方式と p_0p_2 による剰余から求める複合方式を採用した(4.1節)．

実験対象としたプログラムは表4.1の通りである．

表 4.1: プログラム名の一覧

凸包	多角形	
c2s	bo2s	2倍長演算
c3s	bo3s	3倍長演算
c2m		剰余演算，2倍長版，個別方式
c2n	bo2n	剰余演算，2倍長版，複合方式
c3m		剰余演算，3倍長版，個別方式，場合分けして単長で符号判定
c3n	bo3n	剰余演算，3倍長版，複合方式，場合分けして単長で符号判定
c3np		符号判定に2倍長版を利用したc3nの符号判定の簡略化

凸包構成アルゴリズムは文献[6]による．これは逐次添加型のアルゴリズムで，第1座標順にソートされた点集合を入力とし，K 点の点集合に対する凸包を求めるのに $O(K)$ の計算時間ですむ．入力データを，多倍長演算用に16ビットに区切ることと，剰余演算用に各 p_i での剰余を求めることは，別のプログラムとして用意し，計算時間からは除外した．また，退化時の特別な対処は行なっていない．

実験1では凸包構成にかかる時間を計測した．このアルゴリズムで入力の点集合を点列と見て $v = (v_1, \cdots, v_K)$ とし，点 v_i の座標を (x_{i1}, x_{i2}) とすると，判定多項式は

$$f_{ijk}(x) = \begin{vmatrix} 1 & 1 & 1 \\ x_{i1} & x_{j1} & x_{k1} \\ x_{i2} & x_{j2} & x_{k2} \end{vmatrix} \tag{4.109}$$

である．各座標値が $[-L, L]$ の範囲の整数値を取るとき $-4L^2 \leq f_{ijk}(x) \leq 4L^2$ である．実験のため，$4L^2 < p_0p_1p_2$ となるように $L = 23169$ と選んだ．入力点の総数 K は46339 ($= 2L+1$) とし，第1座標は v_1 から順に -23169 ($= -L$) から始まる連続する整数で，v_K の第1座標は23169 ($= L$) にとった．第2座標は $[-L, L]$ の範囲の整数値を一様乱数によりとった．したがって2倍長計算によってすべての判定が正確に行なえるはずである．実験1ではこのようにして得た5種類の点集合から，はじめの10000点，20000点，30000点，40000点および全点をとり，各プログラムにより凸包を求め，計算時間を測定した．実際には計算時間がほとんど一致したため，点の個数ごとに計算時間の平均をとり，表4.2に示した．

実験2では，4点 $v_1 = (-29491, 16384)$，$v_2 = (-22937, -6553)$，$v_3 = (16384, 29491)$，$v_4 = (29491, -16384)$ の凸包を求め結果を比較した(図4.5)．これは入力が単長でも，計算

表 4.2: 実験 1 の結果 (単位は秒)

点数	c2s	c2m	c2n	c3s	c3m	c3n	c3np
10000	4.4	8.5	6.6	9.7	10.3	8.9	9.6
20000	8.9	17.0	13.2	19.5	20.6	17.9	19.4
30000	13.3	25.7	19.9	29.1	31.0	27.0	29.1
40000	17.6	34.2	26.5	38.7	41.4	35.9	38.7
46339	20.2	39.7	30.8	44.8	47.9	41.7	45.0

結果が 2 倍長を超えるため, 2 倍長計算の結果の正しさが保証されない例である.

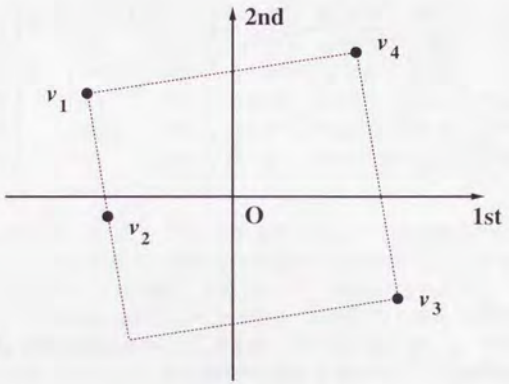


図 4.5: 実験 2 の点配置

表 4.3: 実験 2 の結果 (凸包の時計回り順の頂点順序)

プログラム	c2s	c2m	c2n	c3s	c3m	c3n	c3np
凸包	$v_1v_4v_3v_2$	$v_1v_4v_3v_2$	$v_1v_4v_3v_2$	$v_1v_3v_4v_2$	$v_1v_3v_4v_2$	$v_1v_3v_4v_2$	$v_1v_3v_4v_2$

実験 3 では互いに交わらない単純多角形の集合を生成するプログラムに 2, 3 倍長版複合方式を適用し計算時間を, それぞれ, 2, 3 倍長演算と比較した. アルゴリズムの概要は次のようである: 点列 $v_1, v_2, \cdots, v_K$ を発生させる. 初期辺として, 有向線分 $s_i = \overrightarrow{v_iv_{i+1}}$ $(i = 1, \cdots, K-1), s_K = \overrightarrow{v_Kv_1}$ を取る. 交差する辺対 $s_i = \overrightarrow{v_\alpha v_\beta}, s_j = \overrightarrow{v_\xi v_\eta}$ $(i < j)$ が見つかるかぎり, 辺の置き換え $s_i \leftarrow \overrightarrow{v_\alpha v_\eta}, s_j \leftarrow \overrightarrow{v_\xi v_\beta}$ を行なう. ただし, 置き換えると, 既存の辺と両端点とも一致する辺が生じるときには別の置き換え $s_i \leftarrow \overrightarrow{v_\eta v_\beta}, s_j \leftarrow \overrightarrow{v_\xi v_\alpha}$ を行なう. (この結

果, 既存の辺と両端点とも一致する辺が生じることはない.) 以上のアルゴリズムで, 有限時間内に互いに交わらない単純多角形集合が得られる. 判定多項式は実験1と同じ $f_{ijk}(x)$ である. そのため点列 v_i の座標は -23169 から 23169 までの整数を一様乱数として選んだ. 交差辺対の発見には2.1節で述べたアルゴリズム1型のアルゴリズムを用いた [7, 65]. これは, 線分数 K に対して交差辺対を $O(K \log K)$ で発見する平面走査法のアルゴリズムである. ただし, 平面を走査する掃引直線と交わる辺を保持するために本来なら平衡2分木を使うのであるが, ここでは単純2分木を用いた. 入力点列が一様乱数から得られているので計算時間のオーダには影響はない. 実験1と異なり, 点列生成はプログラム中に組み込み, 判定多項式の計算以外はすべてを単長整数で処理した. 結果を表4.4に示す. 計算時間にはばらつきがあったが, 計算時間の比はほとんど一致した.

表 4.4: 実験3の結果 (計算時間の比 (boXn/boXs) の平均と平均計算時間 (秒))

辺数	2倍長版	3倍長版	bo2s	bo2n	bo3s	bo3n
50	1.00	0.91	0.88	0.88	1.56	1.42
100	1.04	0.90	5.86	6.10	10.32	9.34
200	1.04	0.91	33.52	34.80	59.10	53.52
400	1.04	0.91	169.06	175.90	298.96	270.60
800	1.04	0.90	867.42	906.18	1537.48	1394.86

実験4では, 実験3のプログラムの点列の座標の範囲を単長整数内で広げ, -32766 以上 32766 以下にした場合のプログラムの出力の多角形の総数を調べた. この場合も2倍長計算の結果の正しさは保証されない. 実際, 辺数100で出力結果が2倍長版と3倍長版で異なった. それを図示したものが図4.6の左と中央の図で, それぞれ11個と7個の多角形からなる. このような相異は, 発生点列の初めの37点を取ったときに既に生じる. その時の点配置を図示したものが4.6の右の図である. 図4.6の右の図の2本の線分は, 2倍長版では交わると誤判定され, 3倍長版では交わらないと正しく判定されている.

図 4.6: 実験4での出力例

実験1の結果この凸包構成アルゴリズムに関しては2倍長では多倍長演算を用意したほ

うが剰余演算を利用するより速いが，3倍長で剰余演算を利用する方法が追い付き，剰余演算の仕方によっては逆転することがわかる．また，さらに桁数が延びた場合には，剰余演算が計算時間で有利になることもわかる．実験3の単純多角形集合生成アルゴリズムでも同様の結果が得られた．また，実験2, 4では，実際に入力が単長でも2倍長では判定を誤る例が存在し，その例に対し3倍長では正確な判定ができたことがわかる．また，特に実験4では，乱数により生成した“人工的でない”データでも比較的に小規模なうちに2倍長で判定を誤ることがわかった．これはデータを整数として扱い，退化判定を正確に行なうときには，問題が小規模でも桁の不足による誤判定に注意する必要が新たに生じることを示している．

これらの予備的な実験により，剰余を利用した計算と符号判定は，幾何的アルゴリズムを整数計算で実装するときに，計算時間が増大するのを押さえることがわかる．また，最終的に判定式の値の範囲を無視して桁数を設定し，整数計算すると，退化に近いとき，すなわち値が0に近いときには，うまくアルゴリズムが働いても，浮動小数では誤判定するはずのないくらい大きな値で判定を誤ることがわかる．実際には，アルゴリズムの実装場面では，前に紹介したFortuneの提案のように，浮動小数計算では判定が不確実なところに多倍長整数を用いることが有効であるが，整数に切り替える桁はある程度多めに取る必要があり，剰余演算の利用によって，複雑な幾何的アルゴリズムも退化対処法の組み込みが現実的になり得るといえよう．

第 5 章

位相優先法

これまで, 入力が退化しているとは, 判定多項式の値が 0 になることと定義し, 議論してきた. 代数的退化対処法では, 判定多項式の値の符号が正か負か 0 かどうかを正確に求めることを前提としてきた. 一般に, 誤差をともなう従来のプログラムに較べて, 代数的退化対処法を適用したプログラムでは, 退化対処の前の正確な計算のために計算時間が増大することを指摘した. このような計算時間の増大は好ましくない. 特に, 実用上の観点からすれば, 従来は実用になっていた幾何的アルゴリズムが, 使える計算速度を獲得できなくなる場合もある. 従来程度の計算速度が確保できないという意味で, 代数的退化対処法は実用上の難点があると言える.

計算速度を優先し, 厳密な計算を期待しない場合, 判定式の値が 0 であることを常に正しく判定することはできない. これでは, 退化かどうかの判定も正確にはできないことになる. また, 正確には退化でない場合でも, 退化に近い入力に対しては, 計算誤差のためにアルゴリズムは退化の時と同様の挙動を示すであろう. したがって, この場合, 代数的退化対処法の枠組みでは対処できないことになる.

幾何的アルゴリズムは, 入力データから, 対象とする図形の位相データと計量データを順次計算していく. 退化時には判定多項式の値が 0 になるため, 正か負かの判断で不合理な分岐をし, アルゴリズムは矛盾した構造をもつ図形を扱うようになって破綻する. 厳密な計算を期待しない場合には, さらに判定多項式の値の計算結果が異なり, 符号を誤って判定することも, 退化による不合理な判定の他に, アルゴリズムの破綻原因に加わる.

図形の構造は位相データであり, 離散値 (整数値) として計算され保持される. 一般に離散値だけの範囲ですべての計算が行なえるならば, 誤差が問題になることはないのであるが, 連続値 (浮動小数値) の計算の結果の符号が, 誤判定の可能性があるにも関わらず, 図形の構造の決定に用いられるために, アルゴリズムは矛盾した構造を扱うようになるわけである.

そこで, アルゴリズム中で扱う図形が満たさなければならない構造上の性質を挙げておき, 対応する位相データの無矛盾性の保持を最優先する. すなわち, 連続値である計量データの計算の結果が, 位相データに矛盾を導く判定になる場合には, 積極的に誤判定とみなして, 離散値である位相データの無矛盾性を優先するという幾何的アルゴリズムの設計法が K. Sugihara, M. Iri によって提案され, 点 Voronoi 図構成アルゴリズム [76] が開発された. この幾何的アルゴリズムの設計法が位相優先法である.

概念的であるが位相優先法を定義すると次のようになる. 幾何的アルゴリズム中で扱う図形が満たすべき構造上の性質のうち, アルゴリズム設計者が選んだもの全体を集合 T とする.

定義 8 (位相優先法)
幾何的アルゴリズムにおいて，計量データの計算結果が，位相データを T に反するものにする判定の場合には，位相データを T と整合させることを優先し，計量データの計算結果を誤判定とみなす幾何的アルゴリズムの設計法を**位相優先法**という．

その後，位相優先法は，2 次元点 Voronoi 図構成 [74, 60]，3 次元点 Voronoi 図構成 [33]，線分 Voronoi 図構成 [28]，多角形 Voronoi 図構成 [31]，3 次元凸包構成 [51, 79]，凸多面体の交わり演算 [80] などのアルゴリズムに適用されている．

位相優先法は，厳密な計算が保証されない環境で，

- 無限ループや異常終了を起こすことない，
- 着目した図形の構造を保持した出力が得られる，
- 計算精度に応じた出力が得られる

という特長を持つ退化への対処法である．

位相優先法では，幾何的アルゴリズムを実装する場合，保証すべき図形の構造上の性質を決めたあと，計量データの数値計算誤差に煩わされることなく，ただちに暴走しない実装プログラムを構築することができる．そして得られたプログラムに対して数値計算上の工夫を組み込み改良していくことができる．位相優先法はこのような既存技法に対する柔軟性も持っているといえる．特に，従来のプログラム開発では数値計算上の工夫をすると暴走する場合が変わるプログラムに対して暴走防止の対症的な手当てと数値計算の工夫を同時に考慮しなければならないことを考えると，位相優先法はプログラム開発の手間も軽減するといえよう．

また，多くの場合，幾何的アルゴリズムをどのような用途に利用するかで，出力に求められる内容が異なる．位相優先法では，用途に応じた構造のみを保証することで，過剰に厳密な数値計算による速度の低下を避けつつ，退化への対処ができる．

代数的退化対処法と比較すると，位相優先法は，より概念的であり，具体的に見るには，個々のアルゴリズムに触れる必要があろう．

具体的に個々の幾何的アルゴリズムを設計する場合，扱うアルゴリズムごとに保証すべき図形の構造上の性質を考える必要がある．アルゴリズムによっては，保証すべき図形の構造上の性質を挙げるのが難しいこともある．事実，挙げた構造上の性質だけ満たす図形は，一般に，本来の出力が満たす性質の一部が成り立っているに過ぎない．そのため，得られた出力の構造をもつ出力が本来はありえないという場合もある．使用目的によっては，これは位相優先法の問題点と言える．

本論文では，次節以降で，位相優先法によって新規に開発した線分 Voronoi 図構成アルゴリズムと多角形 Voronoi 図構成アルゴリズムについて詳しく述べる．これらのアルゴリズムは試作プログラムとして実装され，それぞれ，SEGVOR，PLVOR という名で公開されている．また，これらのアルゴリズムで利用している点 Voronoi 図構成アルゴリズムとその周辺についても，必要な範囲で述べることにする．

5.1 Voronoi 図の諸定義と性質

本節では，以下で各種 Voronoi 図構成アルゴリズムの説明をするのに必要な諸定義と性質を述べることにする．

5.1.1 Voronoi図の定義

$g_1, \cdots, g_n$ を，その閉包が互いに共有点を持たない平面 $\mathbf{R}^2$ 上の図形（点，線分など）とし，$G = \{g_1, \cdots, g_n\}$ とおく．平面 $\mathbf{R}^2$ 上の点 v と G の元 g_i の距離 d を，$d(v, g_i) = \inf\{\|u - v\| \mid u \in g_i\}$ で定める．ただし $\|\bullet\|$ は Euclid ノルムを表す．

曲線 $B(g_i, g_j)$ と領域 $R(g_i, g_j)$ を次のように定める：

$$B(g_i, g_j) = \{v \in \mathbf{R}^2 \mid d(v, g_i) = d(v, g_j)\}, \tag{5.1}$$

$$R(g_i, g_j) = \{v \in \mathbf{R}^2 \mid d(v, g_i) < d(v, g_j)\}. \tag{5.2}$$

$B(g_i, g_j)$ を g_i, g_j の **2等分線**とよぶ．$R(g_i) = \bigcap_{j \neq i} R(g_i, g_j)$ とおく．$R(g_1), \cdots, R(g_n)$ が定める平面の分割を G に対する **Voronoi図**といい，$V(G)$ で表す．G の元を $V(G)$ の**生成元**といい，$R(g_i)$ を生成元 g_i の **Voronoi領域**という．$R(g_i)$ の境界を $\partial R(g_i)$ で表す．2個の Voronoi 領域の共通境界である曲線分を **Voronoi辺**とよぶ．また，3個以上の Voronoi 領域に囲まれた境界上の点を **Voronoi点**とよぶ．Voronoi 点と Voronoi 辺はグラフ的な構造をもつ．記述を簡潔にするため，グラフの用語を用いて，Voronoi 領域，Voronoi 辺，Voronoi 点を，単に，**領域**，**辺**，**節点**ともよぶことにする．

辺 b の両側が g_i, g_j の領域で，g_i, g_j, g_k の領域が，節点 v を囲むとする．このとき次の性質が成り立つ．

性質 1 $b \subset B(g_i, g_j)$.

性質 2 $v \in B(g_i, g_j) \cap B(g_j, g_k) \cap B(g_k, g_i)$.

特に，点による勢力圏分割を**点 Voronoi 図**という（図 5.1）．本来は点 Voronoi 図を Voronoi 図とよび，本論文で用いる Voronoi 図はより拡張された意味での Voronoi 図を指すが，用語を簡潔にするために点 Voronoi 図，Voronoi 図の用語を用いる．

本節の点 Voronoi 図構成アルゴリズムが扱う問題を次のように設定する：

問題 2 単位正方形 $[0, 1]^2$ 内に与えられた n 個の点 Voronoi 図をこの正方形内にかけ．

適当な平行移動と拡大縮小を考えれば，この問題設定で一般性を失わない．

Voronoi 図の構造とグラフとの違いは，前者には無限にのびる辺が存在することである．本論文のプログラムでは，正方形 $[-1, 2]^2$ を内部に含む三角形の 3 頂点を生成元に付加することで，無限にのびる辺を 3 個に限定している．このような 3 点を生成元に付加しても，単位正方形 $[0, 1]^2$ の内部では Voronoi 図に変化は起きない．

点 Voronoi 図では 2 等分線 $B(g_i, g_j)$ は直線，$R(g_i, g_j)$ は半平面になる．したがって Voronoi 辺は一般に直線分，Voronoi 領域は凸多角形になる．

生成元集合 G に対して，点 Voronoi 図 $V(G)$ の領域が隣接するような生成元の点の間すべてを線分で結ぶと，G の凸包の三角形分割が得られる（図 5.1）．この三角形分割を G に関する **Delaunay図**といい，$D(G)$ とかく．平面に描かれたグラフとしてみると，Delaunay 図 $D(G)$ と点 Voronoi 図 $V(G)$ は双対となっている．

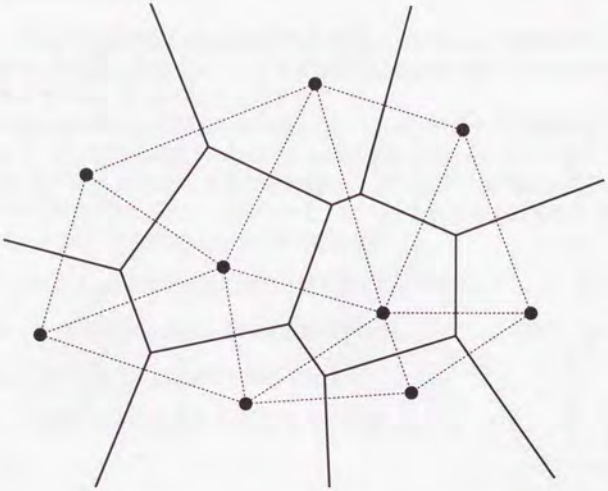


図 5.1: 点 Voronoi 図 (実線) と Delaunay 図 (破線)

5.1.2 逐次添加型アルゴリズム

本論文で説明する各種 Voronoi 図の構成アルゴリズムは逐次添加型に分類される.

本節では, 生成元の添字を付け直し, p_1, p_2, p_3 を前節で付加した 3 点とする. その上で, 生成元集合を改めて $G = \{p_1, p_2, p_3, g_1, \cdots, g_n\}$ とし, この 3 点と生成元の始めの i 個からなる集合を $G_i = \{p_1, p_2, p_3, g_1, \cdots, g_i\}$ とする.

一般に逐次添加型の Voronoi 図構成アルゴリズムは次のようにかける.

アルゴリズム 8 (**Voronoi** 図の逐次添加型構成)
1. $V(G_0)$ を構成する.
2. $i = 1, \cdots, n$ に対して, **2.1** を実行する.
2.1. 点 g_i を添加し $V(G_{i-1})$ を $V(G_i)$ に変更する.

$V(G_0)$ は固定された 3 点の点 Voronoi 図であるから, アルゴリズムの 1 はあらかじめ計算することができる. 本論文で示す各種 Voronoi 図構成アルゴリズムの本体は, 生成元 g_i の添加に伴う Voronoi 図の変更である. グラフ的には, これは $V(G_{i-1})$ のグラフに, 新しい領域 $R(g_i)$ の周となる辺と節点を付け加え, この領域内の古い辺と節点を消去することにより $V(G_i)$ のグラフを得ることである.

5.2 点 Voronoi 図の構成

本節では, 位相優先法の最初の適用例である (2 次元) 点 Voronoi 図構成アルゴリズムに関して説明する. 特に後に線分 Voronoi 図と多角形 Voronoi 図の構成アルゴリズムの説明をするのに必要な部分を中心に述べることにする.

入力の点集合 G に同一の位置を占める点がないことは, 実装する計算機によって判定されているとする. そうでなければアルゴリズム自体が意味を持たないからである.

前節で述べたとおり, アルゴリズム本体は生成元 g_i の添加による Voronoi 図 $V(G_{i-1})$ から $V(G_i)$ への変更の部分である. Voronoi 図を $V(G_{i-1})$ から $V(G_i)$ に変更する場合に, 変化する部分の図形の構造には次のような性質がある (図 5.2).

性質 1. 新しい節点は, 辺の消去される部分と残る部分の境である.

性質 2. 辺のうち, 消去される部分の全体は木構造をなす.

性質 3. 生じる辺は消去される木を囲む閉路をなす.

性質 4. ひとつの領域の周から消去される辺と節点は連結である.

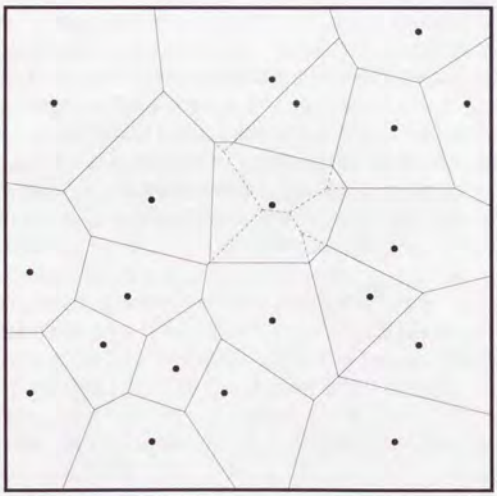


図 5.2: 点 Voronoi 図の更新

これらの性質を満たすように Voronoi 図を更新したものが K. Sugihara, M. Iri による点 Voronoi 図構成アルゴリズム [74] である. 木構造を保持するためのデータ構造 TREE を用意する.

アルゴリズム 9 (生成元 g_i の添加にともなう点 Voronoi 図の更新)
1. TREE を空にする.
2. 生成元 g_i が属する $V(G_{i-1})$ の領域 $R(g_j)$ を求める.
3. 領域の周 $\partial R(G_{i-1})$ 上の節点のうち, g_i の添加で消去されるもの 1 個を見つけ, TREE に加える.
4. TREE の節点 v で, v に接続し TREE に属さない辺 e があるかぎり **4.1-4.5** を行なう.
4.1. 辺 e を TREE に加える.
4.2. v' を辺 e の端点のうち, 節点 v でない方の節点とする.
4.3. v' は消去されるか判定する.
4.4. v' を消去すると性質 2, 性質 4 に反しないか判定する.
4.5. v' は消去されると判定され, 消去すると性質 2, 性質 4 に反しない時に限り, v' を TREE に加える.
5. TREE の辺のうち, 片方の端点が TREE に属さないものの間に新しく節点を付加する.
6. 新しく付加した節点を順に辺で結び, TREE に属する辺と節点を消去し $V(G_i)$ における新しい領域 $R(g_i)$ を構成する.

このアルゴリズム 9のうち 4.3 の部分は, v' を囲む 3 領域を $R(g_j), R(g_k), R(g_l)$ とするとき, 点 g_i は 3 点 g_j, g_k, g_l の作る円の内部にあるかどうかを調べることにより判定できる. これは 3.8節で述べたように行列式の符号判定で求められ, この部分は浮動小数の数値計算になる. 位相優先法は, 木構造を消去するというアルゴリズムの全体像と, 4.5 で浮動小数計算結果を限定的に利用する部分に現われている. 位相データのもつ性質を保持することがアルゴリズムの中心にあることで, 浮動小数の数値計算部分の役割が限定されているのである. アルゴリズム 9の 5 で新しい節点を付加するが, このアルゴリズムではこの節点の座標値を求める必要がない. この座標値は Voronoi 図を描画するときに各節点がどの 3 個の領域に囲まれている Voronoi 点であるかの位相データから, 直接に浮動小数の数値計算で求めることができる. これは Voronoi 図が位相的な構造と Voronoi 点の座標値という計量データを持つのに, その双対の Delaunay 図が接続関係の位相データしか持たないことからも示される. 2.3節で, 多くの幾何的アルゴリズムで位相データの計算と計量データの計算が分離できることを述べたが, 点 Voronoi 図の構成は, このことを示す例になっている. アルゴリズムの細かい部分では, 3 では消去される木構造の最初の 1 点を求めているが, そのような点としては, 判定式の値の絶対値が 0 から遠いものを選ぶことによって, 浮動小数の計算誤差による誤判定をなるべく避けている. また, アルゴリズム 9の 4 では Voronoi 領域の周上の節点をすべて消去されるかどうか判定をして, 一括して消去される部分の決定を行い, 計量データの数値計算結果で絶対値の大きいものの優先度を高めることもできる. このように, アルゴリズムの構築後に数値計算上の工夫を実施できるのも, 位相優先法の特徴である.

消去される部分が木構造であるという性質は, 多くの一般化 Voronoi 図で成立する. 実際, 次の定理が成立する [24].

定理 7
平面上の一般化された Voronoi 図が, 次の性質をみたすとする.

(a) すべての生成元が単連結な Voronoi 領域をもつ.

(b) すべての Voronoi 領域の周は連続曲線である.

このとき，逐次添加法で新たに生じるVoronoi辺が閉じれば，消去される辺と点は木構造をもつ．

これによって，アルゴリズム9は，抽象的な一般化Voronoi図のアルゴリズムから，木構造以外の位相データや計量データの扱いを工夫することによって点Voronoi図を求めるものに具体化・精密化したものとみなせる．

次節以降で述べる線分Voronoi図や多角形Voronoi図との対比として，本節で述べた点Voronoi図の構成アルゴリズムでは，判定式の絶対値，すなわちどれだけ0から遠いかを直接計算結果の信頼度に利用できる理由が，アルゴリズムの位相データを決定する判定式が，アルゴリズム9の4.3で用いるもの1個だけであるためであることを指摘しておく．

このアルゴリズムが，計算誤差による誤判定があっても正常終了することは明らかであるが，従来のVoronoi図構成アルゴリズムでは，無限ループに陥ったり，データ破壊を起こす場合がある．どのように無限ループに陥ったり，データ破壊を起こすかについては，5.3.2節の，本論文で開発した線分Voronoi図の構成アルゴリズムを説明するところで述べることにする．

このアルゴリズム9を用いた点Voronoi図の構成アルゴリズムでは，生成元 p_i $(i = 1, \cdots, n)$ の添加ごとに $O(i)$ の Voronoi 点に関して消去されるかどうかを調べるため，全体の計算時間は最悪時で $O(n^2)$ である．これは分割統治型や平面走査型のアルゴリズムの理論的最適値 $O(n \log n)$ には及ばないが，逐次添加型の一般的な計算時間と一致する．さらに，生成元の添加順序に T. Ohya, M. Iri による四分木を利用した方法 [55] を用いているため，生成元の分布が一様なときは平均 $O(n)$ という高速な計算時間を確保している．このアルゴリズムは Sugihara, Iri によってプログラムとして実装され，VORONOI2 という名で公開されている．

また，3次元の凸包構成アルゴリズム [51] や多面体の集合演算アルゴリズム [80]，分割統治型の点Voronoi図構成アルゴリズム [60] や生成元の分布範囲を3次元に拡張した，3次元点Voronoi図の構成アルゴリズム [33] にも位相優先法が適用されているが，本論文では詳細に立ち入らない．

5.3 線分 Voronoi 図の構成

本節では，位相優先法を適用し，新規に開発した線分Voronoi図の構成アルゴリズムに関して述べる．

線分Voronoi図の定義は，5.1.1節における一般のVoronoi図の定義に準じる．また，本論文で扱う線分Voronoi図では，次の生成元条件を仮定する．

生成元条件: 生成元の直線分は，互いに共有点を持たない．

入力の直線分が長さをもち，互いに交差しないことは，アルゴリズムを実装する計算機で判定されているとする．そうでないとすると，アルゴリズム自体が意味を持たないからである．

5.3.1 線分 Voronoi 図の分割

線分Voronoi図では，辺は，直線分と放物線分が端点で接してできる複雑な曲線であり，計算機内でVoronoi図を扱うときには，より簡単な構造が求められる．そこで，生成元を，両

端点とその間の部分に3分割する．g_i が直線分で，その両端点が p_j, p_k であるとき，$e_i = g_i - \{p_j, p_k\}$ を**開線分**とよぶ．このとき，

$$B(p_j, e_i) = \{x \in \mathbf{R}^2 \mid (x - p_j) \cdot (p_k - p_j) = 0\},$$

$$R(p_j, e_i) = \{x \in \mathbf{R}^2 \mid (x - p_j) \cdot (p_k - p_j) < 0\},$$

$$R(e_i, p_j) = \{x \in \mathbf{R}^2 \mid (x - p_j) \cdot (p_k - p_j) > 0\}$$

とすることによって，端点と開線分を生成元とするVoronoi図を定めることができる．ただし，$\cdot$ を2次元の内積とする．このVoronoi図を**分割線分 Voronoi 図**とよぶことにする (図5.3)．

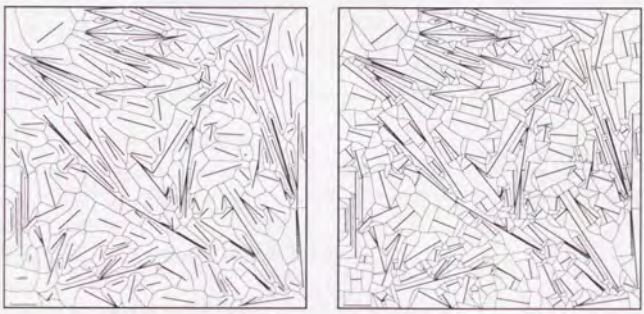


図 5.3: 線分 Voronoi 図と分割線分 Voronoi 図

生成元の分割により，辺はすべて，直線分あるいは放物線分になり，計算機内での処理が容易になる．線分 Voronoi 図は分割線分 Voronoi 図から容易に得られる．

同様にして，開線分と端点の他に，いくつかの点を生成元に加えても，Voronoi 図を定義できる．この Voronoi 図を**点線分 Voronoi 図**とよぶ．

本節で扱う問題を次のように設定する:

問題 3 単位正方形 $[0,1]^2$ 内に与えられた n 本の直線分の分割線分 Voronoi 図をこの正方形内にかけ．

適当な平行移動と拡大縮小を考えれば，この問題設定で一般性を失わない．

本節のアルゴリズムでは，点 Voronoi 図の場合と同様に正方形 $[-1,2]^2$ を内部に含む三角形の3頂点を生成元に加えることで，無限にのびる辺を3個に限定している．このような3点を生成元に加えても，単位正方形 $[0,1]^2$ の内部では Voronoi 図に変化は起きない．

5.3.2 アルゴリズムと数値的不安定性

本節の点線分 Voronoi 図の構成アルゴリズムも逐次添加型である．全端点を先に添加し，続いて開線分を添加する．その基本構造は次の通りである．ただし，与えられた n 個の線分を $g_1, \cdots, g_n$ とし，g_i から端点を除いた開線分を e_i，g_i の端点を p_{2i+2}, p_{2i+3} と名付ける．さらに，前章で付加した 3 頂点を p_1, p_2, p_3 と名付ける．$G_0 = \{p_1, \cdots, p_{2n+3}\}$ とおき，$G_i = G_0 \cup \{e_1, \cdots, e_i\}$ とする．

アルゴリズム 10 (逐次添加型点線分 **Voronoi** 図の構成)

1. 点 Voronoi 図 $V(G_0)$ を作る．

2. $i = 1, \cdots, n$ に対して **2.1** を行なう．

2.1. 開線分 e_i を添加し，$V(G_{i-1})$ を $V(G_i)$ に変更する．

点 Voronoi 図 $V(G_0)$ の構成には 5.2節のアルゴリズム [74] を利用するので，本論文で示すアルゴリズムの本体は，生成元の添加のうちでも，特に，開線分 e_i の添加に伴う Voronoi 図の変更である．

アルゴリズム 10の 2.1 を細かく見ると，次のようになる．

アルゴリズム 11 (開線分 e_i の添加 (図 **5.4**))

1. 端点領域の周 $\partial R(p_{2i+2})$ と $B(p_{2i+2}, e_i)$ との交点を見付け，それを v_∞, v_0 に格納する．ただし，v_∞, v_0 の選び方は，$B(p_{2i+2}, e_i)$ に v_∞ から v_0 への向きを付けたとき，領域 $R(p_{2i+2}, e_i)$ が左になるようにする．

2. v_0 で $B(p_{2i+2}, e_i)$ と交わる辺を b_0 に格納する．

3. $g \leftarrow p_{2i+2}$, $j \leftarrow 0$.

4. $v_j \neq v_\infty$ のうちは **4.1**-**4.3** を行なう．

4.1. b_j の両側の領域の生成元のうち，g でないものを，改めて g とする．

4.2. $\partial R(g)$ の辺を b_j から反時計回りに探索し，$B(g, e_i)$ と，はじめて交わる辺と交点を，b_{j+1}, v_{j+1} に格納する．

4.3. $j \leftarrow j + 1$.

5. $v_0, \cdots, v_\infty$ を順に新しい辺で結び，最後に v_∞, v_0 を辺で結んで，$V(G_i)$ の新しい領域 $R(e_i)$ を確定する．

6. 領域 $R(e_i)$ 内の辺と節点を全て消去する．

アルゴリズム 11の途中で，$\partial R(e_i)$ が $V(G_{i-1})$ の節点を通過する時，生成元の集合が退化していることになる．アルゴリズム 11で，交点 v_∞ が $V(G_{i-1})$ の節点と一致する場合は，退化の一例である．この場合は，数値誤差を伴う計算環境でこのアルゴリズムを実行すると，1 で，v_0, v_∞ のどちらかが見付からなかったり，本来 $v_\infty = v_m$ で **4** を終了するのに，v_∞ と v_m を通る辺としてそれぞれ別の辺が指定されたりして，無限ループに陥る可能性がある．また，そのために，データ構造が破壊され，開線分 e_i の添加前に，$R(p_{2i+2})$ が消滅し，アルゴリズム 11の実行が破綻することもある．これは，本質的退化 (2.4節) の場合もあり，開線分の添加順序を変えるなどの対症療法では，必ずしも回避できない問題である．

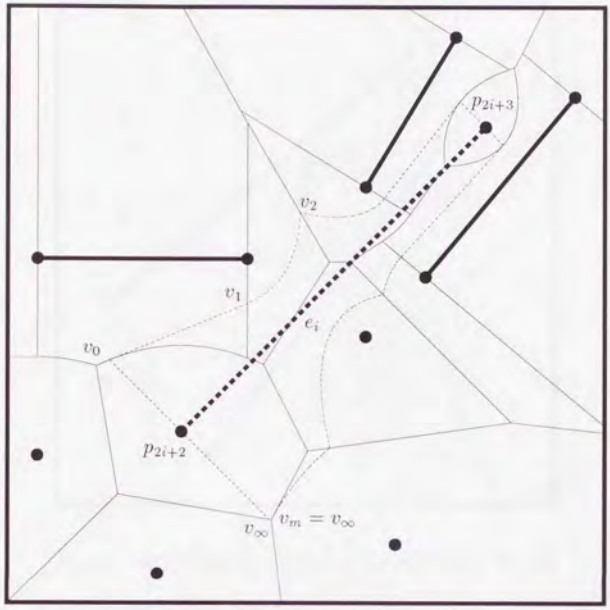


図 5.4: 線分 Voronoi 図構成のための開線分の添加

5.3.3 位相優先法の適用

開線分 e_i の添加によって, Voronoi 図を $V(G_{i-1})$ から $V(G_i)$ に変更する場合に, 変化する部分の位相的構造には次のような性質がある.

性質 3. 新しい節点は, 辺の消去される部分と残る部分の境である.

性質 4. 辺のうち, 消去される部分の全体は木構造をなす.

性質 5. 生じる辺は消去される木を囲む閉路をなす.

性質 6. 消去される木は両端点の領域 $R(p_{2i+2}), R(p_{2i+3})$ をつなぐ 1 個の道を含む.

これらは性質 6 を除いて, アルゴリズム 10の 1 で $V(G_0)$ の構成にも用いられている [74]. 文献 [74] ではさらに, 性質 4 の代わりに,

性質 7. 1 個の領域の周から消去される辺と節点は連結である

を用いているが, これは開線分の添加時には成立しない (図 5.5).

上記の性質を利用して, 開線分 e_i の添加のアルゴリズムに位相優先法を次のように適用する.

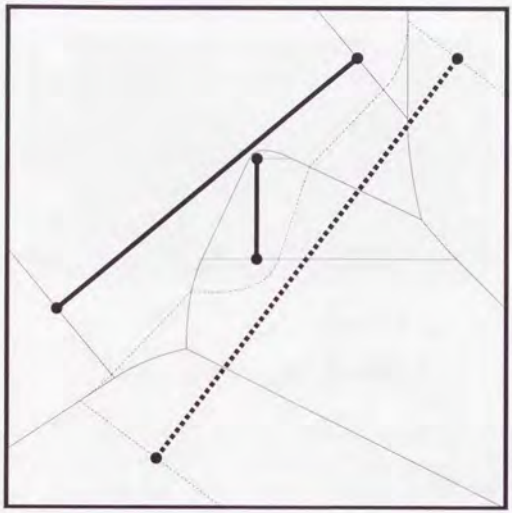


図 5.5: 1 個の領域から 2 か所以上の部分が消去される例

アルゴリズム 12 (位相優先法による開線分の添加)
1. 領域 $V(p_{2i+2})$ の周上の節点から, 道の出発点を決定し, 登録する.
2. 出発点から, 領域 $V(p_{2i+3})$ の周に至るまで, 消去される辺や節点を探索して道を見付けていき, 道上の辺と節点を, 順次, 辺スタック B と節点スタック V に積む.
3. スタック B, V から辺や節点を取り出し, それに接続する辺や節点のうち, 未探索のものを探索し, 消去される節点や辺があれば, スタック V, B に積むことを, スタック B, V が空になるまで行ない, 消去される木を確定する.
4. 消去される木を囲むように節点と辺を付け加え, 消去される木を消去する.

アルゴリズム 11では $R(e_i)$ の境界をまず見付けて, そのあとで内部を取り除いた. 一方, このアルゴリズム 12では, 取り除くべき木をまず見付け, そのあとでそれを囲む境界を生成する. この違いが位相優先法を実現するひとつの鍵となっている. この更新アルゴリズム 12により, 点線分 Voronoi 図はすべての節点が次数 3 をもつ平面グラフとして更新される. 退化状態での次数が 4 以上の節点は, 次数が 3 の複数の節点が, 長さが 0 の辺で隣接しているとして扱う. 次にこのアルゴリズムにおける, 出発点の決定法, 道の探索法, 木の確定法について順に述べる.

ここで, アルゴリズムを記述するのに必要な用語をいくつか定めておく.

帯状の領域 $R(e_i, p_{2i+2}) \cap R(e_i, p_{2i+3})$ を**開線分 e_i で定まる帯状領域** (図 5.6) とよぶ. 新たに生じる領域 $R(e_i)$ はこの帯状領域の内部にある.

生成元 g_j と点 $v \in \mathbf{R}^2$ に対して, 点 $u \in \mathrm{cl}g_j$ が $\|v-u\| = d(v, g_j)$ を満たすとき u を点 v の生成元 g_j への**射影点**という. ただし cl• は閉包を表す. 点線分 Voronoi 図では, 射影点は

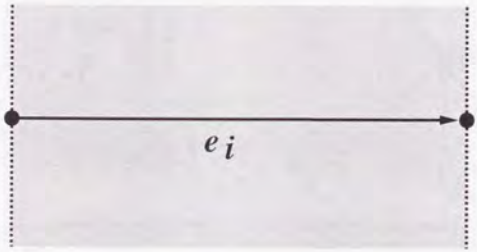


図 5.6: 開線分で定まる帯状領域

(v, g_j) の組に対し，一通りに定まるので，$\mathrm{pr}_v(g_j)$ と書くことにする (図 5.7).

開線分 e_i の端点 p_{2i+2}, p_{2i+3} の座標を (x_{2i+2}, y_{2i+2}), (x_{2i+3}, y_{2i+3}) とする.

また，開線分 e_i を p_{2i+2} から p_{2i+3} に向かう向きを付ける．この帯状領域内の生成元は**添加開線分 e_i に対して右側**および**左側**に分類される.

任意の節点 v に対して，v を囲む領域の生成元のひとつを g とする．このとき，節点 v に関する生成元 g の**左度** $L_v(g)$ を，射影点 $\mathrm{pr}_v(g) = (x, y)$ を用いて

$$L_v(g) = \begin{vmatrix} x & x_{2i+2} & x_{2i+3} \\ y & y_{2i+2} & y_{2i+3} \\ 1 & 1 & 1 \end{vmatrix}$$

で定める (図 5.7). 座標系を右手系にとれば，$L_v(g)$ が正，負のとき射影点 $\mathrm{pr}_v(g)$ が，開線分 e_i を延長した有向直線のそれぞれ左側，右側にある．$L_v(g)$ が正，負のとき，生成元 g は**節点 v から見て** (開線分 e_i の) それぞれ**左**，**右にある**ということにする.

v と $\mathrm{pr}_v(g)$ がともに e_i で定まる帯状領域内にあれば，g が e_i に対して左右のどちら側に分類されるかと，g は v から見て e_i の左右のどちらにあるかは一致する.

Voronoi 図 $V(G_{i-1})$ で両端点の領域 $R(p_{2i+2})$, $R(p_{2i+3})$ が，ある辺を境界に隣接していれば，道としてこの辺をとればよい．隣接していない場合，道は，開線分 e_i に対して左右に分類される生成元の領域の間を通る．以下，この場合を考える.

道上の節点を出発点から道に沿って $v_0, v_1, \cdots$ とし，同様に，道上の辺を道に沿って $b_0, b_1, \cdots$ とする． b_j に v_j を始点とする向きを付けたとき，b_j の左右の領域の生成元を，それぞれ，$g_{\mathrm{L}}(v_j, b_j)$, $g_{\mathrm{R}}(v_j, b_j)$ とする．また，領域 $R(p_{2i+2})$ の周上の節点 v に対し，v に接続する 3 辺のうち，周上にない (唯一の) ものを $b(v)$ と書くことにする．道の出発点 v_0, 出発辺 b_0 は次の特徴により決まる.

特徴 1. v_0 は領域 $R(p_{2i+2})$ の周上の節点である，

特徴 2. v_0 は e_i で定まる帯状領域内にある，

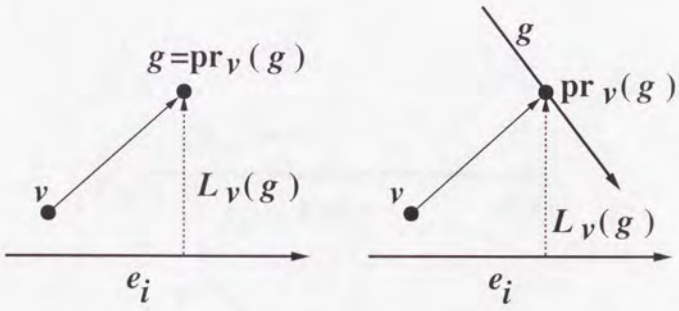


図 5.7: 射影点と左度

特徴 3. v_0 から見て $g_{\mathrm{L}}(v_0, b_0)$, $g_{\mathrm{R}}(v_0, b_0)$ は 開線分 e_i に対しても, それぞれ左, 右にある

誤差が発生する時には, 数値計算に基づく判定を完全には信用できない. そこで, 出発点 v_0 を次のように決定する.

アルゴリズム 13 (出発点の決定 (図 **5.8**))
1. 領域 $R(p_{2i+2})$ の周上の節点で, 直線 $B(p_{2i+2}, e_i)$ から端点 p_{2i+3} の側に最も離れているものを v に格納する.
2. $g_{\mathrm{L}}(v, b(v)), g_{\mathrm{R}}(v, b(v))$ は (v から見て e_i に対して), それぞれ左, 右にあるか, あるいは, 以下の探索が領域 $R(p_{2i+2})$ の周を一周するまで調べる.
$g_{\mathrm{L}}(v, b(v))$ が左ならば
時計回りに領域 $R(p_{2i+2})$ の周をみて v に隣接する点を, 改めて v に格納する.
(そうでなく)$g_{\mathrm{R}}(v, b(v))$ が右ならば
反時計回りに領域 $R(p_{2i+2})$ の周をみて v に隣接する点を, 改めて v に格納する.
3. $v, b(v)$ を, それぞれ v_0, b_0 として終了する

探索をアルゴリズム 13の 1 のような節点から始めた理由は, 出発点の特徴 3 は, 帯状領域内でしか成立しないことと, 帯状領域内にある領域 $R(p_{2i+2})$ の周上の節点は, 最少の場合, 1 個しかないからである.

出発点 v_0, 出発辺 b_0 から出発して, 領域 $R(p_{2i+3})$ に達する消去される道の探索法を考える.

節点 v に辺 b が接続しているとする. 節点 v に接続する辺を反時計回りに並べて $b, b_{\mathrm{R}}(b, v)$, $b_{\mathrm{L}}(b, v)$ とする. 辺 $b_{\mathrm{R}}(b, v), b_{\mathrm{L}}(b, v)$ は b, v から一意に定まる. また, $g_{\mathrm{N}}(b, v) = g_{\mathrm{R}}(v, b_{\mathrm{L}}(b, v))$ $(= g_{\mathrm{L}}(v, b_{\mathrm{R}}(b, v)))$ とする (図 5.9).

見付けるべき道は次の特徴により決定される.

特徴 4 道は両端点の領域 $R(p_{2i+2}), R(p_{2i+3})$ を結び,

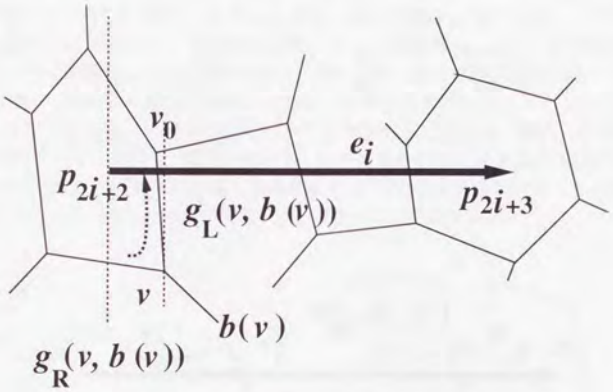


図 5.8: 出発点の決定

特徴 5 道とこの 2 領域を除くと開線分 e_i によって定まる帯状領域は 2 個の連結成分に分かれるが, このとき一方の成分は開線分 e_i の右側にある生成元のみを含み, 他方は左側の生成元のみを含む.

これにより, v_0, b_0 から始めて, 左右の生成元の領域の境を, $v_1, b_1, v_2, \cdots$ と探索していけば, 領域 $R(p_{2i+3})$ に達したとき, 道を見付けたことになる. 節点 v_{j+1} は辺 b_j から自動的に見付かり, 辺 b_j は, 生成元 $g_N(b_{j-1}, v_j)$ が v_j から見て左か右かに応じて, それぞれ, $b_R(b_{j-1}, v_j)$ か $b_L(b_{j-1}, v_j)$ を採ればよい. しかし計算誤差が存在するから, 領域 $R(p_{2i+3})$ に探索が達する前に, 無限にのびる辺に達したり, 領域 $R(p_{2i+2})$ に戻ったり, 既に探索された節点と辺に阻まれて, 閉路を作ってしまったりする場合がある. このような状況を回避するため, これらの状

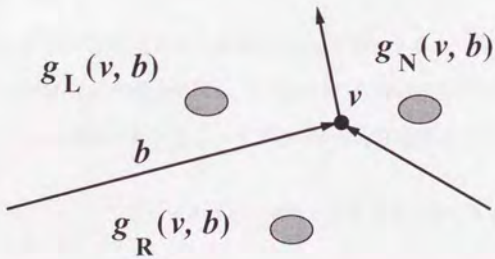


図 5.9: 道の探索に用いる記号

況になったら, 探索中に行なった判定のうち最も判定が微妙なもの (すなわち, 左度 $L_v(g_N(b,v))$ の絶対値が最小のもの) を選び, そこから別方向へ探索を進める (図 5.10). すべての節点と辺が探索の対象になり得るので, 数値誤差があっても, 領域 $R(p_{2i+3})$ に探索が達することが保証される. データ構造として, 探索が節点 v から辺 b へ進んだ時には b から v を指すポインタを持ち, 逆に, 辺 b から節点 v へ進んだ時には v から b を指すポインタを持てば, 探索が領域 $R(p_{2i+3})$ に達した後にポインタをたどって道が容易に得られる. 節点, 辺が探索されたことを, フラグを立てて保持し, 探索の終了後, フラグを効率よく戻すため, 探索された節点, 辺を積むスタック Q_v, Q_b も用意する. 具体的にアルゴリズムを書き下すことは容易なので省略する. 以上により, 道の上のすべての節点, 辺を得ることができる.

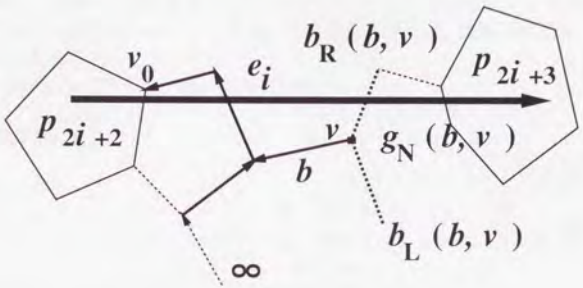


図 5.10: 道の探索

消去される木の確定法を述べる. 道が探索されれば, この道から順次消去される節点, 辺を探索していき, 消去される木を構成することができる. 退化と計算誤差がなければ, 辺ごとに, 新しい節点が生じるか否かを判定するだけで十分であるが, 計算誤差のため, 生じるはずの節点を見落とし, 残るはずの辺が次々と消去されると判定される可能性がある. それでも位相優先法では暴走することはないが, 一般の数値対策と両立させることができるので, 次のような基準を採用した.

基準 1. 辺に節点が生じると判定されたら必ずその判定を採用する.

基準 2. 辺に節点が生じないと判定されても, 次の場合には, 節点が生じるものとして扱う

判定する辺を b とし, b の両端の節点を u, v とする. ただし, v は消去されると判定済みとする.

基準 2.1. 辺 b が消去されると, 消去される辺で閉路ができて木にならない, または, 探索が無限にのびる辺に達する.

基準 2.2. 節点 v が, e_i で定まる帯状領域の外にある.

基準 2.3. 節点 u が, 辺 b の両側の生成元よりも e_i から遠い.

基準 2.4. 節点 u が消去されると，端点領域 $R(p_{2i+2})$，$R(p_{2i+3})$ から 2 個以上の部分が消去される．

これらのうち，基準 2.1 と 2.4 が位相優先法によるもので，残りが一般の数値対策によるものである．

5.3.4 浮動小数の数値計算

位相優先法により，計算誤差に関わらず，処理が途中で破綻しないという意味の数値的安定性が得られる．しかし，これは数値計算はいいかげんでもよいことを意味するわけでは決してない．実用的には，適切な数値計算法を選択することが，アルゴリズムの性能を上げるために重要である．本論文で報告するシステム SEGVOR は試作段階にあり，数値計算部分に，改良の余地があると思われるが，システムの全体像を示すため，数値計算部分について全容を述べることにする．

数値計算部分では次の基本計算を各所で用いる．

(3, 3) 型行列式の計算

行列式

$$\begin{vmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{vmatrix} \tag{5.3}$$

の値を求めるのに，プログラムでは，この行列式を展開した各項

$$\begin{array}{lll} x_{11}x_{22}x_{33}, & x_{12}x_{23}x_{31}, & x_{13}x_{32}x_{21}, \\ -x_{13}x_{22}x_{31}, & -x_{12}x_{21}x_{33}, & -x_{11}x_{23}x_{32} \end{array}$$

を正のものと負のものに分けて和をとり，最後に両者の和をとった．

2 次方程式の求解

2 次方程式 $ax^2+2b'x+c=0$ の第 1 根 $(-b'+\sqrt{b'^2-ac})/a$，第 2 根 $(-b'-\sqrt{b'^2-ac})/a$ を次のように求める：

$d' \leftarrow b^2-ac$ として，
$a \neq 0$ であって，

$d'<0$ のとき，解なし，
$d' \geq 0, b'>0$ のとき，
　第 1 根 $x \leftarrow c/(-b'-\sqrt{d'})$，
　第 2 根 $x \leftarrow (-b'-\sqrt{d'})/a$，
$d' \geq 0, b'<0$ のとき，
　第 1 根 $x \leftarrow (-b'+\sqrt{d'})/a$，
　第 2 根 $x \leftarrow c/(-b'+\sqrt{d'})$，
$d' \geq 0, b'=0$ のとき，

第 1 根 $x \leftarrow +\sqrt{d'}/a$,
第 2 根 $x \leftarrow -\sqrt{d'}/a$,

$a = 0$ であって,

$b' \neq 0$ のとき, 第 1 根, 第 2 根とも $x \leftarrow -c/(b' \cdot 2)$,
$b' = 0, c = 0$ のとき, 不定解であるが解なしとする,
$b' = 0, c \neq 0$ のとき, 解なし.

ここでは ac が b'^2 に較べて非常に小さいときに $|b'| - \sqrt{b'^2 - ac}$ に生じる桁落ちに対する最小限の配慮の他には特別な工夫は行なっていない.

三角形の面積と 3 点の向き

同次座標で考える. 3 点 $(X_1, Y_1)/Z_1, (X_2, Y_2)/Z_2, (X_3, Y_3)/Z_3$ で, 行列式

$$\begin{vmatrix} X_1/Z_1 & X_2/Z_2 & X_3/Z_3 \\ Y_1/Z_1 & Y_2/Z_2 & Y_3/Z_3 \\ 1 & 1 & 1 \end{vmatrix}$$

が正の時, この 3 点は正の向きに並んでいるといい, 負の時は負の向きに並んでいるという. この行列式が 0 のときは 3 点は同一直線上にある. また, この行列式の絶対値はこの三角形の面積の 2 倍を表し, 符号の正, 負はこの 3 点が三角形の周上に, それぞれ, 反時計回り, 時計回りに並ぶことを表す.

ここで $Z_1, Z_2, Z_3 \geq 0$ にしておけば, この行列式の符号は行列式

$$\begin{vmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ Z_1 & Z_2 & Z_3 \end{vmatrix}$$

の符号に等しい. この理由により, プログラム中では同次座標を用いるときは, Z 座標が非負になるように各座標の符号を調整している.

2 直線の交点を求めること

同次座標で考えることにし, 第 1 の直線上の 2 点を $(X_1, Y_1)/Z_1, (X_2, Y_2)/Z_2$ とし, 第 2 の直線上の 2 点を $(X_3, Y_3)/Z_3, (X_4, Y_4)/Z_4$ とする. この 2 直線の交点を $(X, Y)/Z$ とすれば, $(X/Z, Y/Z)$ が 2 個の直線上の点であることより,

$$\begin{vmatrix} X & X_1 & X_2 \\ Y & Y_1 & Y_2 \\ Z & Z_1 & Z_2 \end{vmatrix} = \begin{vmatrix} X & X_3 & X_4 \\ Y & Y_3 & Y_4 \\ Z & Z_3 & Z_4 \end{vmatrix} = 0 \tag{5.4}$$

となる. これより

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} /\!/ \left(\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} \times \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} \right) \times \left(\begin{bmatrix} X_3 \\ Y_3 \\ Z_3 \end{bmatrix} \times \begin{bmatrix} X_4 \\ Y_4 \\ Z_4 \end{bmatrix} \right) \tag{5.5}$$

であることがわかる．ここで $/\!/$ は平行を，$\times$ は3次元の外積を表す．この右辺を計算すると

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} /\!/ \begin{vmatrix} X_2 & X_3 & X_4 \\ Y_2 & Y_3 & Y_4 \\ Z_2 & Z_3 & Z_4 \end{vmatrix} \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} - \begin{vmatrix} X_3 & X_4 & X_1 \\ Y_3 & Y_4 & Y_1 \\ Z_3 & Z_4 & Z_1 \end{vmatrix} \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} \tag{5.6}$$

を得る．同次座標であることから左辺に実数倍の任意性があり，右辺と左辺が等しいとしてよい．そこで

$$d_1 \leftarrow \begin{vmatrix} X_2 & X_3 & X_4 \\ Y_2 & Y_3 & Y_4 \\ Z_2 & Z_3 & Z_4 \end{vmatrix}, \tag{5.7}$$

$$d_2 \leftarrow \begin{vmatrix} X_3 & X_4 & X_1 \\ Y_3 & Y_4 & Y_1 \\ Z_3 & Z_4 & Z_1 \end{vmatrix}, \tag{5.8}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \leftarrow d_1 \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} - d_2 \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} \tag{5.9}$$

とし，$Z \geq 0$ となるように，必要があれば符号を一斉に変え，X, Y, Z を求めることにした．この記法により無限遠点も含めて統一的に扱える．

Voronoi 点の求め方

これまでで示した基本的な計算などを用いて，本節では，開線分の添加によって着目している Voronoi 辺上に新たな Voronoi 点が生じるかどうか判定するとともに，生じる時には，その Voronoi 点の座標値を求めるために用いた具体的な計算法について述べる．

新たに生じる Voronoi 点は2個の新たに生じる Voronoi 辺と1個の既にある Voronoi 辺の交点である．ここでは，はじめにそれぞれの辺を延長した放物線や直線の交点として Voronoi 点の候補を求め，それが既にある Voronoi 辺の上の点であれば，新たに生じる Voronoi 点とみなす方法をとることにする．

プログラムで消去される木の Voronoi 辺と Voronoi 点を登録するとき，Voronoi 点 v が登録済みで，v に接続する Voronoi 辺 b が未登録であり，b の上に新たな Voronoi 点が生じるかどうか判定する場合を考える．もし新たな Voronoi 点 u が生じるならば，u は生成元 $g_{\mathrm{R}}(v,b), g_{\mathrm{L}}(v,b), e_i$ の Voronoi 領域によって反時計回りに囲まれる．以下では簡単のため，u は $g_{\mathrm{R}}(v,b), g_{\mathrm{L}}(v,b), e_i$ のよって反時計回りに囲まれるということにする．このことから u の座標値をこの3個の生成元の位置を表す入力から直接求めることができ，誤差が累積するのを防ぐことができる．また，組み合せ構造から u を囲む生成元の順序が特定でき，放物線と直線の交点のような複数の交点がある場合でも必要な点を区別できる．ここでは必要な点以外の交点の座標値は求めないことにした．これによって，放物線と，その軸に近い直線の交点のように不要な交点の座標値がオーバーフローを起こしかねない場合に，これを回避することができる．

Voronoi 点を，それを囲む生成元の種類によって次の6個の型に分類する：

(点, 点, 点) 型: 3個の点，
(点, 端 - 線) 型: 1個の開線分，2個の点，そのうちただ1個がその開線分の端点，

(点, 点, 線) 型: 1 個の開線分, その端点でない 2 個の点,
(端 - 線, 線) 型: 2 個の開線分, そのただ一方の端点である 1 個の点,
(点, 線, 線) 型: 2 個の開線分, その端点でない 1 個の点,
(線, 線, 線) 型: 3 個の開線分.

求める Voronoi 点の候補がどの型に入るかは組み合せ構造から求まる. 開線分 e_i の添加の時には, 3 個の生成元のうち 1 個は開線分 e_i になるので, (点, 点, 点) 型の Voronoi 点は生じない. そのため Voronoi 図の更新プログラム中には (点, 点, 点) 型の Voronoi 点を求める手続きはない.

以下では の Voronoi 点の残りの 5 個の型別に座標値を求める手続きについて述べる. 共通の記号として, 求める Voronoi 点を u, その座標を (x, y), u を反時計回りに囲む生成元を $g_i, g_j g_k$ として, 開線分より点を先行して書くことにする. このようにすると (点, 点, 点) 型と (線, 線, 線) 型の Voronoi 点の場合を除き, g_i, g_j, g_k の並び方は一意に定まる. また, 局所的な変数として点の名前 p_l とその座標値 (x_l, y_l), $(l = 1, \cdots, 6)$ を用いることにする.

(点, 端 - 線) 型の Voronoi 点の場合

g_i, g_j は点であり, これをそれぞれ p_1, p_2 とする. g_k は開線分で p_1 または p_2 は 1 個の端点である. 反対側の端点を p_3 とする (図 5.11).

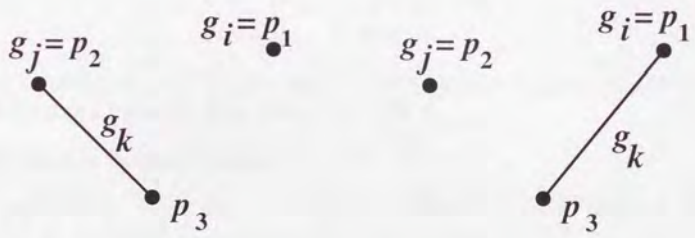


図 5.11: (点, 端 - 線) 型の Voronoi 点を求めるための記号割当

g_k が p_2 に接続する場合

線分 up_2 と線分 p_2p_3 が垂直であることから

$$\begin{bmatrix} x - x_2 & y - y_2 \end{bmatrix} \begin{bmatrix} x_3 - x_2 \\ y_3 - y_2 \end{bmatrix} = 0. \tag{5.10}$$

また, u は線分 p_1p_2 の垂直二等分線上にあることから

$$\begin{bmatrix} x - (x_1 + x_2)/2 & y - (y_1 + y_2)/2 \end{bmatrix} \begin{bmatrix} x_1 - x_2 \\ y_1 - y_2 \end{bmatrix} = 0. \tag{5.11}$$

これら 2 個の式より次の手続きを得る.

$$d_1 \leftarrow \begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{vmatrix}, \tag{5.12}$$

$$d_2 \leftarrow (x_1 - x_2)^2 + (y_1 - y_2)^2, \tag{5.13}$$

$$x \leftarrow (-(y_1 - y_2)d_2/2)/d_1 + x_2, \tag{5.14}$$

$$y \leftarrow ((x_1 - x_2)d_2/2)/d_1 + y_2. \tag{5.15}$$

g_k が p_1 に接続する場合

上の場合の p_1, p_2 を入れ替えればよい:

$$d_1 \leftarrow \begin{vmatrix} x_2 & x_1 & x_3 \\ y_2 & y_1 & y_3 \\ 1 & 1 & 1 \end{vmatrix}, \tag{5.16}$$

$$d_2 \leftarrow (x_2 - x_1)^2 + (y_2 - y_1)^2, \tag{5.17}$$

$$x \leftarrow (-(y_2 - y_1)d_2/2)/d_1 + x_1, \tag{5.18}$$

$$y \leftarrow ((x_2 - x_1)d_2/2)/d_1 + y_1. \tag{5.19}$$

いずれの場合も $d_1 \leq 0$ の時は点 u が定まらないか, あるいは生成元 p_1, p_2, p_3 がこの順に反時計回りに点 u を囲まないため, 交点なしとして扱う.

(点, 点, 線) 型の Voronoi 点の場合

g_i, g_j が点であり, これを p_1, p_2 とする. また, g_k は開線分でその両端に接続する生成元を p_3, p_4 とする. p_3, p_4 の選び方には p_3, p_4 の入れ替えるだけの任意性があるが, 3 点 p_1, p_3, p_4 と p_2, p_3, p_4 がともに正の向きになるように選ぶ (図 5.12). 具体的には p_{M} を線分 p_1p_2 の中点とし, その座標 $(x_{\mathrm{M}}, y_{\mathrm{M}})$ を

$$x_{\mathrm{M}} \leftarrow (x_1 + x_2)/2 \tag{5.20}$$

$$y_{\mathrm{M}} \leftarrow (y_1 + y_2)/2 \tag{5.21}$$

で定めて行列式

$$\begin{vmatrix} x_{\mathrm{M}} & x_3 & x_4 \\ y_{\mathrm{M}} & y_3 & y_4 \\ 1 & 1 & 1 \end{vmatrix}$$

が負の時に限り p_3, p_4 を入れ替えることにした.

次に, 計算を簡単にするため, 点 $p_1, p_2, p_3, p_4, p_{\mathrm{M}}$ に変換

$$(x, y) \mapsto (x', y')$$

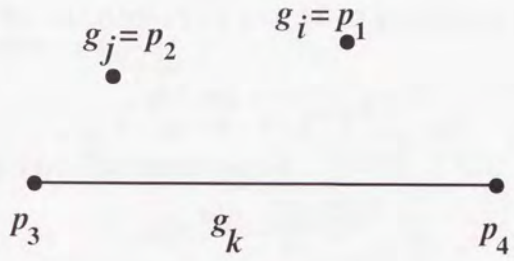


図 5.12: (点, 点, 線) 型の Voronoi 点を求めるための記号割当

を, 手続き

$$d \leftarrow \sqrt{(x_4 - x_3)^2 + (y_4 - y_3)^2} \tag{5.22}$$

$$c \leftarrow (x_4 - x_3)/d \tag{5.23}$$

$$s \leftarrow (y_4 - y_3)/d \tag{5.24}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} \leftarrow \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} x - x_{\mathrm{M}} \\ y - y_{\mathrm{M}} \end{bmatrix} \tag{5.25}$$

によって施す. この変換は平行移動と回転を組み合せた合同変換で, 計算誤差のない限り距離と角度を保存する. 線分 up_{M} と線分 p_1p_2 が垂直であるから

$$\begin{bmatrix} x' - x'_{\mathrm{M}} & y' - y'_{\mathrm{M}} \end{bmatrix} \begin{bmatrix} x'_2 - x'_1 \\ y'_2 - y'_1 \end{bmatrix} = 0 \tag{5.26}$$

が成立し, 点 u は点 p_1 と直線 p_3p_4 から等距離にあるから

$$\begin{vmatrix} x' - x'_{\mathrm{M}} & (x'_4 - x'_3)/d \\ y' - y'_{\mathrm{M}} & (y'_4 - y'_3)/d \end{vmatrix}^2 = (x' - x'_2)^2 + (y' - y'_2)^2 \tag{5.27}$$

が成立する. ここで

$$y'_3 = y'_4 \leq 0,\ x'_4 - x'_3 = d,\ x'_1 = -x'_2,\ y'_1 = -y'_2,\ x'_{\mathrm{M}} = y'_{\mathrm{M}} = 0$$

であることが容易にわかるので $x'_1, y'_1, y'_4, x'_{\mathrm{M}}, y'_{\mathrm{M}}, d$ を消去して整理すると

$$x'_2 x' + y'_2 y' = 0, \tag{5.28}$$

$$\begin{vmatrix} x' & 1 \\ y' - y'_3 & 0 \end{vmatrix}^2 = x'^2 + x'^2_2 + y'^2 + y'^2_2 \tag{5.29}$$

を得る. y' を消去して x の 2 次式にすると次のようになる:

$$A \leftarrow y'_2, \tag{5.30}$$

$$B' \leftarrow -y'_3 x'_2, \tag{5.31}$$

$$C \leftarrow (x'^2_2 + y'^2_2 - y'^2_3) y'_3, \tag{5.32}$$

$$Ax'^2 + 2B'x' + C = 0. \tag{5.33}$$

この方程式の2根から得られる解のうち不要なものを (x'', y'') とすると g_i, g_j, g_k がこの順に u を反時計回りに囲むことから

$$\begin{vmatrix} x'_2 - x'_1 & x' - x'' \\ y'_2 - y'_1 & y' - y'' \end{vmatrix} > 0 \tag{5.34}$$

を満たす．このことから x' は2次方程式の第2根

$$x' = \frac{-B' - \sqrt{B'^2 - AC}}{A}$$

であることがわかる．x' は，実際は前述の2次方程式の求解の手続きにより求める．

求めた x' を放物線の式 (5.29) に代入し y' を求める．具体的には

$$y' \leftarrow (y'_3 - (x'^2 + {x'_2}^2 + {y_2}^2)/y'_3)/2 \tag{5.35}$$

とする．経験的には y' を先に求めたり，x' を直線の式 (5.28) に代入するより誤差の面で総合的に有利であった．Voronoi 点 u の座標値 (x, y) は逆変換

$$\begin{bmatrix} x \\ y \end{bmatrix} \leftarrow \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} x_{\mathrm{M}} \\ y_{\mathrm{M}} \end{bmatrix} \tag{5.36}$$

によって求めた．また，2次方程式に解がないあるいは $y'_3 = 0$ の時には Voronoi 点は生じないものとして扱った．

(端 - 線, 線) 型の Voronoi 点の場合

g_i のみが点であり，これを p_1 とする．また，g_j, g_k は開線分であり，どちらか一方のみが p_1 に接続する．

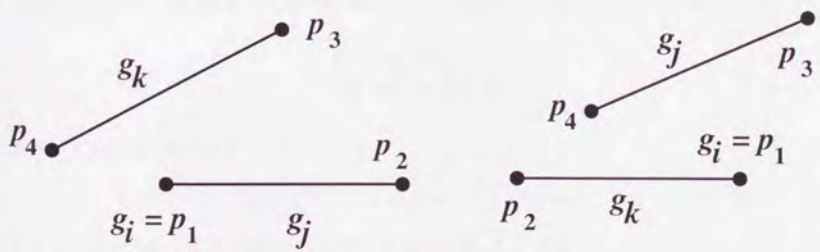


図 5.13: (端 - 線, 線) 型の Voronoi 点を求めるための記号割当

g_j が p_1 に接続する場合

開線分 g_j の端点のうち p_1 でない側で接続する点を p_2 とする. 開線分 g_k の両端に接続する点を p_3, p_4 とする (図 5.13).

$$\begin{vmatrix} x_1 & x_3 & x_4 \\ y_1 & y_3 & y_4 \\ 1 & 1 & 1 \end{vmatrix}$$

が負の時に限り p_3, p_4 を入れ替えることにし, p_3, p_4 の選び方を一意にした. 次に (点, 点, 線) 型の時と同じように, 計算を簡単にするため, 点 p_1, p_2, p_3, p_4 に合同変換

$$(x, y) \mapsto (x', y')$$

を, 手続き

$$d \leftarrow \sqrt{(x_4 - x_3)^2 + (y_4 - y_3)^2}, \tag{5.37}$$

$$c \leftarrow (x_4 - x_3)/d, \tag{5.38}$$

$$s \leftarrow (y_4 - y_3)/d, \tag{5.39}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} \leftarrow \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} x - x_1 \\ y - y_1 \end{bmatrix} \tag{5.40}$$

によって施す.

線分 up_1 と線分 p_1p_2 が垂直であることから

$$\begin{bmatrix} x' - x'_1 & y' - y'_1 \end{bmatrix} \begin{bmatrix} x'_2 - x'_1 \\ y'_2 - y'_1 \end{bmatrix} = 0, \tag{5.41}$$

点 u は点 p_1 と直線 p_3p_4 から等距離にあるから

$$\begin{vmatrix} x' - x'_3 & (x'_4 - x'_3)/d \\ y' - y'_3 & (y'_4 - y'_3)/d \end{vmatrix}^2 = (x' - x'_1)^2 + (y' - y'_1)^2 \tag{5.42}$$

が成立する. u を g_i, g_j, g_k がこの順で反時計回りに囲むから (x', y') は上の方程式の解のうち

$$\begin{vmatrix} x' & x'_1 & x'_2 \\ y' & y'_1 & y'_2 \\ 1 & 1 & 1 \end{vmatrix} \geq 0 \tag{5.43}$$

を満たすものをとればよい. また, ここで

$$y'_3 = y'_4 \leq 0,\ x'_4 - x'_3 = d,\ x'_1 = y'_1 = 0$$

であることが容易にわかるので, 方程式から $x'_1, y'_1, x'_4, y'_4, d$ を消去して整理すると

$$x'_2 x' + y'_2 y' = 0, \tag{5.44}$$

$$\begin{vmatrix} x' - x'_3 & 1 \\ y' - y'_3 & 0 \end{vmatrix}^2 = x'^2 + y'^2 \tag{5.45}$$

を得る. y' を消去して x の 2 次式にすると次のようになる:

$$A \leftarrow y'_2, \tag{5.46}$$
$$B' \leftarrow -y'_3 x'_2, \tag{5.47}$$
$$C \leftarrow -y'_2 y'_3 y'_3, \tag{5.48}$$
$$\tag{5.49}$$

$$Ax'^2 + 2B'x' + C = 0. \tag{5.50}$$

この方程式の 2 根から得られる解のうち

$$\begin{vmatrix} x' & x'_1 & x'_2 \\ y' & y'_1 & y'_2 \\ 1 & 1 & 1 \end{vmatrix} \geq 0 \tag{5.51}$$

を満たすのは第 2 根

$$x' = \frac{-B' - \sqrt{B'^2 - AC}}{A}$$

であることがわかる. x' は, 実際は前述の 2 次方程式の求解の手続きにより求める.

求めた x' を放物線の式 (5.45) に代入し y' を求める. 具体的には

$$y' \leftarrow (y'_3 - x'^2/y'_3)/2 \tag{5.52}$$

とする. 経験的には y' を先に求めたり, x' を直線の式 (5.44) に代入するより誤差の面で総合的に有利であった. Voronoi 点 u の座標値 (x, y) は逆変換

$$\begin{bmatrix} x \\ y \end{bmatrix} \leftarrow \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \tag{5.53}$$

によって求めた. また, 2 次方程式に解がないあるいは $y'_3 = 0$ の時には Voronoi 点は生じないものとして扱った.

g_k が p_1 に接続する場合

開線分 g_k の端点のうち p_1 でない側で接続する点を p_2 とし, 開線分 g_j の両端に接続する点を p_3, p_4 として, あとは上述の g_j が p_1 に接続する場合と同様に扱う (図 5.13). ただし, u を g_i, g_k, g_j がこの順で反時計回りに囲むことから, 2 次方程式の第 1 根

$$x' = \frac{-B' + \sqrt{B'^2 - AC}}{A}$$

が求めるものになることのみが異なる.

(点, 線, 線) 型の Voronoi 点の場合

g_i のみが点であり, これを p_1 とする. また, g_j, g_k は開線分であり, どちらの端点も p_1 と一致しない. 開線分 g_k の両端に接続する点を p_2, p_3 とし, g_j の両端に接続する点を p_4, p_5 とする.

$$\begin{vmatrix} x_1 & x_4 & x_5 \\ y_1 & y_4 & y_5 \\ 1 & 1 & 1 \end{vmatrix}$$

が負の時に限り p_4, p_5 を入れ替え,

$$\begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{vmatrix}$$

が負の時に限り p_2, p_3 を入れ替えることにし, p_2, p_3, p_4, p_5 の選び方を一意にした (図 5.14).

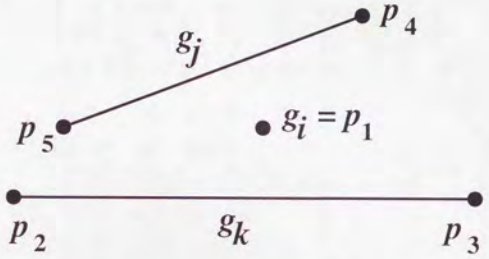


図 5.14: (点, 線, 線) 型の Voronoi 点を求めるための記号割当

次に (点, 点, 線) 型, (点, 端 - 線) 型の時と同じように計算を簡単にするため, 点 p_1, p_2, p_3, p_4, p_5 に合同変換

$$(x, y) \mapsto (x', y')$$

を, 手続き

$$d_j \leftarrow \sqrt{(x_5 - x_4)^2 + (y_5 - y_4)^2} \tag{5.54}$$

$$c \leftarrow (x_5 - x_4)/d_j \tag{5.55}$$

$$s \leftarrow (y_5 - y_4)/d_j \tag{5.56}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} \leftarrow \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} x - x_1 \\ y - y_1 \end{bmatrix} \tag{5.57}$$

によって施す.

点 u は点 p_1 と直線 p_4p_5 から等距離にあるから

$$\frac{1}{d_j} \begin{vmatrix} x' & x'_4 & x'_5 \\ y' & y'_4 & y'_5 \\ 1 & 1 & 1 \end{vmatrix} = (x' - x'_1)^2 + (y' - y'_1)^2 \tag{5.58}$$

が成立し，点 u は 2 直線 p_4p_5, p_2p_3 から等距離にあることと，u, p_2, p_3 および u, p_4, p_5 が正の向きに並ぶよう p_2, p_3, p_4, p_5 を選んだことから

$$d_k \leftarrow \sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2}, \tag{5.59}$$

$$\frac{1}{d_j}\begin{vmatrix} x' & x'_4 & x'_5 \\ y' & y'_4 & y'_5 \\ 1 & 1 & 1 \end{vmatrix} = \frac{1}{d_k}\begin{vmatrix} x' & x'_2 & x'_3 \\ y' & y'_2 & y'_3 \\ 1 & 1 & 1 \end{vmatrix} \tag{5.60}$$

が成立する．また，ここで

$$y'_4 = y'_5 \leq 0,\ x'_5 - x'_4 = d_j,\ x'_1 = y'_1 = 0$$

であることが容易にわかるので方程式から $x'_1, y'_1, x'_4, x'_5, y'_5, d_j$ を消去して整理すると

$$\begin{vmatrix} x' & -1 \\ y' - y'_4 & 0 \end{vmatrix}^2 = x'^2 + y'^2, \tag{5.61}$$

$$\begin{vmatrix} x' & -1 \\ y' - y'_4 & 0 \end{vmatrix} = \begin{vmatrix} x' - y'_3 & (x_2 - x_3)/d_k \\ y' - y'_3 & (y_2 - y_3)/d_k \end{vmatrix} \tag{5.62}$$

を得る．y' を消去して x の 2 次式にすると次のようになる:

$$x'_{\mathrm{M}} \leftarrow (x'_2 + x'_3)/2, \tag{5.63}$$

$$y'_{\mathrm{M}} \leftarrow (y'_2 + y'_3)/2, \tag{5.64}$$

$$S \leftarrow x'_{\mathrm{M}}(y'_2 - y'_3) + (y'_4 - y'_{\mathrm{M}})(x'_2 - x'_3), \tag{5.65}$$

$$A \leftarrow d_k - (x'_2 - x'_3), \tag{5.66}$$

$$B' \leftarrow y'_4(y'_2 - y'_3), \tag{5.67}$$

$$C \leftarrow y'_4(y'_4 A - S \cdot 2), \tag{5.68}$$

$$Ax'^2 + 2B'x' + C = 0. \tag{5.69}$$

p_2, p_3, p_4, p_5 の選び方と施した変換の形から，求める x' は，この方程式の 2 根のうち大きい方である．$A \geq 0$ となるように，必要ならば A, B', C の符号を一斉に反転すれば，x' は 2 次方程式 $Ax'^2 + 2B'x' + C = 0$ の第 1 根

$$x' = \frac{-B' + \sqrt{B'^2 - AC}}{A}$$

であることがわかる．x' は，実際は前述の 2 次方程式の求解の手続きにより求める．

求めた x' を放物線の式 (5.61) に代入し y' を求める．具体的には

$$y' \leftarrow (y'_4 - x'^2/y'_4)/2 \tag{5.70}$$

とする．経験的には y' を先に求めたり，x' を直線の式 (5.62) に代入するより誤差の面で有利であった．Voronoi 点 u の座標値 (x, y) は逆変換

$$\begin{bmatrix} x \\ y \end{bmatrix} \leftarrow \begin{bmatrix} c & -s \\ s & c \end{bmatrix}\begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \tag{5.71}$$

によって求めた．また，2 次方程式に解がない時には Voronoi 点は生じないものとして扱い，$y'_4 = 0$ の時には p_1 と g_j が近接しているとみなし $y' = 0$ とした．

(線, 線, 線) 型の Voronoi 点の場合

g_i, g_j, g_k は開線分である．開線分 g_j, g_k を必要ならば延長してそれらの交点 q_i を同次座標で求め，$(X_i, Y_i)/Z_i$ とする．同様にして g_k, g_i から交点 q_j を，g_i, g_j から交点 q_k を同次座標で求め，それぞれ $(X_j, Y_j)/Z_j$，$(X_k, Y_k)/Z_k$ とする (図 5.15)．具体的には既に述べた 2 直線の交点を求める手続きを用いる．したがって，Z_i, Z_j, Z_k はともに非負である．

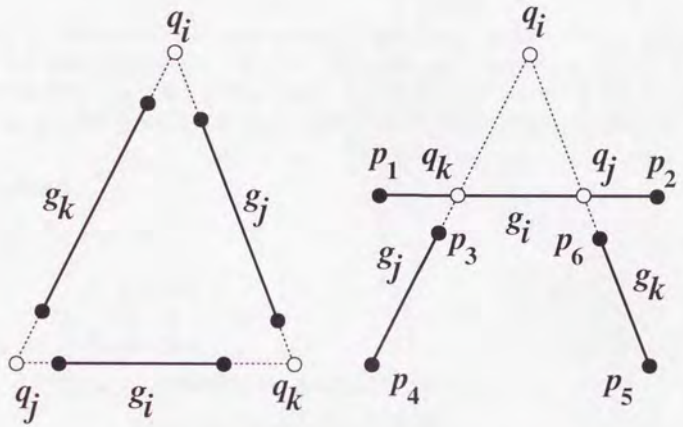


図 5.15: (線, 線, 線) 型の Voronoi 点を求めるための記号割当

u は三角形 $q_iq_jq_k$ の内心または傍心である．内心と傍心の区別はこの 3 頂点の向きで定まり，q_i, q_j, q_k が正の向きの時にのみ内心になる．具体的には

$$t_{ijk} \leftarrow \begin{vmatrix} X_i & X_j & X_k \\ Y_i & Y_j & Y_k \\ Z_i & Z_j & Z_k \end{vmatrix} \tag{5.72}$$

として，t_{ijk} が正の時は内心，非正の時は傍心として扱った．

内心や傍心を求めるための計算は次のようにした．まず D_i, D_j, D_k を

$$D_i \leftarrow \sqrt{(X_jZ_k - X_kZ_j)^2 + (Y_jZ_k - Y_kZ_j)^2}, \tag{5.73}$$

$$D_j \leftarrow \sqrt{(X_kZ_i - X_iZ_k)^2 + (Y_kZ_i - Y_iZ_k)^2}, \tag{5.74}$$

$$D_k \leftarrow \sqrt{(X_jZ_k - X_jZ_i)^2 + (Y_iZ_j - Y_jZ_i)^2} \tag{5.75}$$

と定める．$D_i/Z_jZ_k, D_j/Z_kZ_i, D_k/Z_iZ_j$ は，それぞれ三角形 $q_iq_jq_k$ の 3 辺 q_iq_j, q_jq_k, q_kq_i の

長さになる．内心または傍心を同次座標で考えて $(X,Y)/Z$ とすると，

$$\frac{Z_jZ_k}{\pm D_i}\begin{vmatrix} X/Z & X_j/Z_j & X_k/Z_k \\ Y/Z & Y_j/Z_j & Y_k/Z_k \\ 1 & 1 & 1 \end{vmatrix} = \frac{Z_kZ_i}{\pm D_j}\begin{vmatrix} X/Z & X_k/Z_k & X_i/Z_i \\ Y/Z & Y_k/Z_k & Y_i/Z_i \\ 1 & 1 & 1 \end{vmatrix} = \frac{Z_iZ_j}{\pm D_k}\begin{vmatrix} X/Z & X_i/Z_i & X_j/Z_j \\ Y/Z & Y_i/Z_i & Y_j/Z_j \\ 1 & 1 & 1 \end{vmatrix} \tag{5.76}$$

が成立する．複号をすべて正にとったときは $(X/Z,Y/Z)$ は内心を表し，1 個だけ負をとると対応する点に関する傍心になる．すなわち，等式の第 1 辺だけが負の符号をとると点 q_i に関する (傍接円が辺 q_jq_k に接する) 傍心であり，第 2 辺，第 3 辺だけが負の符号をとると，それぞれ点 q_j, q_k に関する傍心となる．複号の残りのとり方はこれらの場合に帰着できる．

$t_{ijk} > 0$ の場合

式 (5.76) を変形して，

$$D_jD_k\begin{vmatrix} X & X_j & X_k \\ Y & Y_j & Y_k \\ Z & Z_j & Z_k \end{vmatrix} = D_kD_i\begin{vmatrix} X & X_k & X_i \\ Y & Y_k & Y_i \\ Z & Z_k & Z_i \end{vmatrix} = D_iD_j\begin{vmatrix} X & X_i & X_j \\ Y & Y_i & Y_j \\ Z & Z_i & Z_j \end{vmatrix} \tag{5.77}$$

となる．この式は q_i, q_j, q_k が無限遠点の場合にも成立する．これから，

$$\begin{vmatrix} X & D_iX_i + D_jX_j & D_kX_k \\ Y & D_iY_i + D_jY_j & D_kY_k \\ Z & D_iZ_i + D_jZ_j & D_kZ_k \end{vmatrix} = 0, \tag{5.78}$$

$$\begin{vmatrix} X & D_iX_i & D_jX_j + D_kX_k \\ Y & D_iY_i & D_jY_j + D_kY_k \\ Z & D_iZ_i & D_jZ_j + D_kZ_k \end{vmatrix} = 0 \tag{5.79}$$

を得る．これは，

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \mathbin{/\!/} \left(\left(\left(D_i\begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} + D_j\begin{bmatrix} X_j \\ Y_j \\ Z_j \end{bmatrix}\right) \times D_k\begin{bmatrix} X_k \\ Y_k \\ Z_k \end{bmatrix}\right) \times \left(D_i\begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} \times \left(D_j\begin{bmatrix} X_j \\ Y_j \\ Z_j \end{bmatrix} + D_k\begin{bmatrix} X_k \\ Y_k \\ Z_k \end{bmatrix}\right)\right)\right) \tag{5.80}$$

を意味する．ここで $/\!/$ は平行を，$\times$ は 3 次元の外積を表す．この右辺を計算し整理すると

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} /\!/ -D_iD_jD_k\begin{vmatrix} X_i & X_j & X_k \\ Y_i & Y_j & Y_k \\ Z_i & Z_j & Z_k \end{vmatrix}\left(D_i\begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} + D_j\begin{bmatrix} X_j \\ Y_j \\ Z_j \end{bmatrix} + D_k\begin{bmatrix} X_k \\ Y_k \\ Z_k \end{bmatrix}\right). \tag{5.81}$$

同次座標であることから左辺に実数倍の任意性があり，プログラムでは

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \leftarrow D_i \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} + D_j \begin{bmatrix} X_j \\ Y_j \\ Z_j \end{bmatrix} + D_k \begin{bmatrix} X_k \\ Y_k \\ Z_k \end{bmatrix}, \tag{5.82}$$

$$x \leftarrow X/Z, \tag{5.83}$$

$$y \leftarrow Y/Z \tag{5.84}$$

として Voronoi 点 u の座標 (x, y) を求めた．$Z = 0$ の時には Voronoi 点は生じないものとして扱った．

$t_{ijk} \leq 0$ の場合

求める Voronoi 点 u は傍心であるが，3 個の傍心のうちどれであるかを判定するために次のような手続きをとった．開線分 g_i の両端点 p_1, p_2 および，g_j の両端点 p_3, p_4 および，g_k の両端点 p_5, p_6 を

$$\begin{vmatrix} X_i & x_1 & x_2 \\ Y_i & y_1 & y_2 \\ Z_i & 1 & 1 \end{vmatrix} \geq 0, \quad \begin{vmatrix} X_j & x_3 & x_4 \\ Y_j & y_3 & y_4 \\ Z_j & 1 & 1 \end{vmatrix} \geq 0, \quad \begin{vmatrix} X_k & x_5 & x_6 \\ Y_k & y_5 & y_6 \\ Z_k & 1 & 1 \end{vmatrix} \geq 0 \tag{5.85}$$

を満たすように定める (図 5.15)．開線分 g_i, g_j, g_k に，それぞれ p_1 から p_2，p_3 から p_4，p_5 から p_6 に向かう向きを付け，それぞれの相対的な位置を考える．例えば，g_i, g_j に関して

$$L_1 \leftarrow \begin{vmatrix} x_1 & x_3 & x_4 \\ y_1 & y_3 & y_4 \\ 1 & 1 & 1 \end{vmatrix}, \; L_2 \leftarrow \begin{vmatrix} x_2 & x_3 & x_4 \\ y_2 & y_3 & y_4 \\ 1 & 1 & 1 \end{vmatrix} \tag{5.86}$$

とし，g_i の g_j に対する相対的な位置を，

$L_1 \geq 0, L_2 \geq 0$ ならば

$L_1 = 0, L_2 = 0$ のとき，g_i は g_j と同一直線上にある，
それ以外のとき，g_i は g_j の左にある，

$L_1 < 0, L_2 < 0$ ならば，g_i は g_j の右にある，それ以外ならば，g_i は g_j の両側にある，

ということにする．他の組み合せに関しても，同様に定める．

この相対的な位置関係によって，求める傍心を次のように決定する：

g_j は g_i の右になく，g_k は g_i の右になく，
g_i は g_j の左になく，g_i は g_k の左にないとき → q_i に関する傍心，

そうでなく，g_i は g_j の右になく，g_k は g_j の右になく，
g_j は g_i の左になく，g_j は g_k の左にないとき → q_j に関する傍心，

そうでなく，g_i は g_k の右になく，g_j は g_k の右になく，
g_k は g_i の左になく，g_k は g_j の左にないとき → q_k に関する傍心，

そうでないとき → Voronoi 点は存在しない．

q_i, q_j, q_k に関する傍心は，内心を求める手続きのそれぞれ D_i, D_j, D_k に負の符号を付けることで得られる:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \leftarrow \pm D_i \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} + \pm D_j \begin{bmatrix} X_j \\ Y_j \\ Z_j \end{bmatrix} + \pm D_k \begin{bmatrix} X_k \\ Y_k \\ Z_k \end{bmatrix}, \tag{5.87}$$

$$x \leftarrow X/Z, \tag{5.88}$$

$$y \leftarrow Y/Z. \tag{5.89}$$

だだし，複号はちょうど1個だけ負をとる．また，$Z = 0$ の時には Voronoi 点は生じないとして扱った．

5.3.5 計算量と記憶量

アルゴリズムの計算時間，記憶量について考える．生成元の総数を n とする．

理論的には，線分 Voronoi 図の構成には，$O(n)$ の記憶量と $O(n \log n)$ の計算量が必要である．これらを達成するアルゴリズムに，再帰二分法 [86] があるが，プログラムが複雑になるため，本論文では採用しなかった．

線分 Voronoi 図，点線分 Voronoi 図では，点 Voronoi 図と同様に，節点，辺の数はともに $O(n)$ である [38]．

プログラム中で用いたデータ構造では，Voronoi 図として，辺，節点，領域ごとに，接続関係を示す定数個のポインタを持ち，また，全節点の座標値を保持する．これに必要な記憶量は $O(n)$ である．他には，辺や節点の探索用のスタック，フラグ，リスト，左度の格納場所などが $O(n)$，局所変数は定数個で，アルゴリズム全体では $O(n)$ の記憶量である．

アルゴリズムの理論的な計算量を各部分について考える．まず，点 Voronoi 図の構成に $O(n^2)$ かかる．次に，開線分1個の添加ごとに，道の出発点を決定するために辺の総数に比例する $O(n)$ の計算量，道を探索する時には，行詰まった場合に，探索をやり直す節点を探すのに $O(n)$ かかると見積って，結局，$O(n^2)$ の計算量がかかる．道から木構造を得るのには，やはり，辺の総数である $O(n)$ の計算量がかかる．したがって，アルゴリズム全体の計算量は $O(n^2 + n(n + n^2 + n))$ すなわち $O(n^3)$ と見積ることができる．ただし，これは数値計算の精度が非常に悪く，ほとんど無意味な振る舞いをすると仮定した最悪の場合の計算量である．

道を探索する時，既探索節点をヒープに積んで，探索に行詰まる場合に，再探索節点を効率よく求めることができる．この場合，アルゴリズム全体の計算量を $O(n^2 \log n)$ に減らすことができる．しかし，これでは，探索が順調に進んでも，アルゴリズム全体の計算量は $O(n^2 \log n)$ のままになる．本論文では実用性を重視し，このような手段を採らないことにした．

計算量を各部分でみると，アルゴリズム全体の計算量を決めているのは道を探索する部分である．この部分の計算量は，探索に行詰まることがなければ，開線分1個の添加ごとに $O(n)$ ですむ．この時には，アルゴリズム全体の計算量は $O(n^2)$ になる．

現実的には，道を探索する，あるいは木構造を得るために，全辺を探索することはほとんどないであろう．多くの実際的な場合に，計算量 $O(n^2)$ の見積りは過大である．実際，点 Voronoi 図の構成では，生成元が一様に分布するならば，平均計算量は $O(n)$ になる [55]．本論文で提案するアルゴリズムの実際の計算時間について調べるため，実装プログラム SEGVOR によって計算機実験をした．使用した計算機は富士通 S-4/EC である．

線分は両端点の順序対として与え，第 1 の端点を前端点，第 2 の端点を後端点とよんで区別する．実験の全体を通じ，端点の初期の座標は，$[0,1]$ の一様乱数によって与えた．入力線分の取り方は次の 3 種類である．

取り方 1. 線分が第 i 番目まで得られている時，第 $i+1$ 番目の線分を取り，第 1 番目から第 i 番目までの線分との交差判定を順に行ない，交差しなくなるまで，第 $i+1$ 番目の線分を取り直すことを，第 n 番目の線分が得られるまで繰り返す．

取り方 2. 線分が第 i 番目まで得られている時，第 $i+1$ 番目の線分をまず仮に取り，第 1 番目から第 i 番目までの線分との交差判定を順に行ない，交差する度に，第 $i+1$ 番目の線分の後端点を，交点と前端点を結んだ線分の $1:9$ の内分点に置き換えることを，第 n 番目の線分が得られるまで繰り返す．

取り方 3. まず n 個の線分を仮に取り，第 i 番目と第 j 番目の線分 $(i<j)$ の交差判定を行ない，交差したら，それぞれの線分の後端点を入れ換えることを，交差がなくなるまで繰り返す．第 i 番目と第 j 番目の線分の交差判定は (i,j) の辞書式順序で小さいほうから行なう．

理論的には，これらの取り方で，入力線分が有限時間で得られる．線分数が 1024 本の場合の取り方 1, 2, 3 による Voronoi 図の例を図 5.16に示す．

図 5.16: 取り方 1(左), 取り方 2(中), 取り方 3(右) による Voronoi 図の例

次の 3 種類の実験を行なった．

実験 1. 取り方 1, 2, 3 によって得られた線分を生成元とする Voronoi 図を生成元数が 256, 512, 1024, 2048, 4096 の時，それぞれに対して 10 回ずつ求め，計算時間を計測した．

実験 2. 取り方 1, 2, 3 によって得られた入力データを線分の長さの長いものから短いものへ並べ換え，実験 1 と同様に時間を計測した．

実験 3. 取り方 1, 2, 3 によって得られた入力データを線分の長さの短いものから長いものへ並べ換え，実験 1 と同様に時間を計測した．

表 5.1: 実験の結果 (単位は秒)

実験	線分数	取り方 1	取り方 2	取り方 3
実験 1	256	54.7	54.4	54.5
	512	111.5	106.6	107.7
	1024	229.8	226.0	223.9
	2048	488.9	487.4	481.2
	4096	1119.5	1183.6	1076.0
実験 2	256	53.9	54.9	53.0
	512	108.9	105.8	105.9
	1024	230.8	220.1	214.7
	2048	498.0	478.8	468.5
	4096	1115.3	1084.2	1065.1
実験 3	256	54.2	54.2	52.8
	512	109.2	105.9	107.6
	1024	230.0	219.1	215.7
	2048	494.8	484.1	467.3
	4096	1113.8	1087.6	1069.8

実験の結果を表 5.1に示す.

生成元の取り方ごとに, 10 回の線分の生成で, 線分の長さに関する分布は大きな差はなかった.

次に図 5.17に取り方 1, 2, 3 における線分数と各線分数で Voronoi 図を 10 回生成するのに要した合計時間の関係を両対数グラフで示す. 各グラフにおいて破線は計算時間の合計が線分数に比例することと, 2 乗に比例することを表す傾きの直線である. グラフから, どの取り方でも線形は越えるものの, 線形に近い計算時間を持つことがわかる.

アルゴリズムから, 計算時間は消去された辺の総数程度であることがわかる. 辺の総数は, 生成元の添加につれて増加する. そこで, より多くの辺を消去すると思われる長い開線分を生成元が増えないうちに処理したほうが計算時間が短くなると予想できる.

入力線分データごとに実験 2, 実験 3 での計算時間を実験 1 での計算時間で割った値を, それぞれ白い菱形と黒い菱形で示したグラフが図 5.18である. 菱形が縦軸の 1.0 より下にあれば, 線分の並べ換えにより, 計算時間が短縮されたことになるが, 予想した計算時間の変化は, 明確には確認できなかった. 理由としては点 Voronoi 図を構成した時点で, 全体の 2/3 の生成元が, 添加を終えているため, 開線分を添加する時には, 既に辺の総数が十分に多くなっていることなどが考えられる.

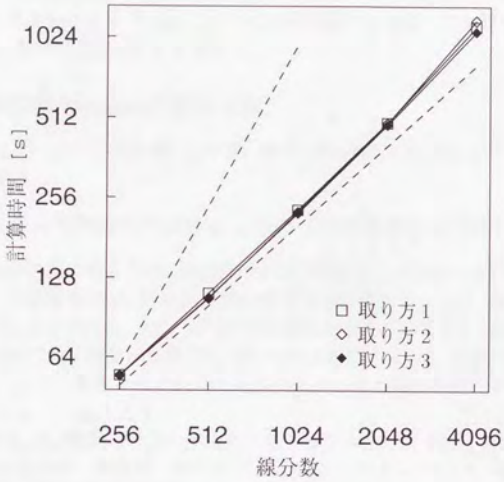


図 5.17: 線分数と計算時間

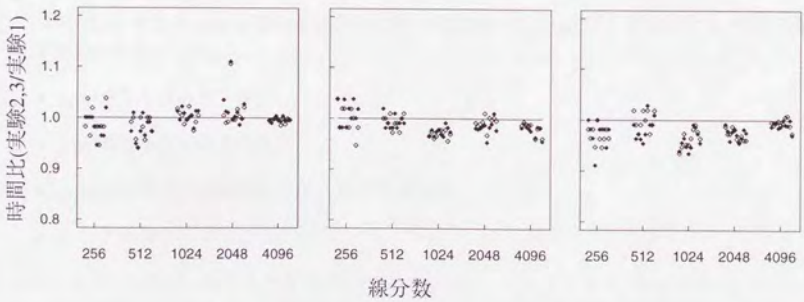


図 5.18: 添加順序による計算時間の比較 (左から取り方 1, 2, 3)

5.4 多角形 Voronoi 図の構成

本節では，前節の線分 Voronoi 図の構成アルゴリズムを拡張して開発した，点，線分，折線，多角形を生成元とする Voronoi 図の位相優先法による構成アルゴリズムに関して述べる．本論文では多角形は辺集合として考える．

5.4.1 多角形 Voronoi 図の分割

g_i $(i = 1, \cdots, N)$ を平面 $\mathbf{R}^2$ 上の点，線分，折れ線または多角形で，次の生成元条件を満たすものとする:

生成元条件: 各 g_i は単純で，任意の g_i, g_j $(i \neq j)$ は共有点を持たない．

ここで，図形が単純であるとは，図形の辺が自己交差をもたないことである．グラフ理論の言葉を使って，生成元条件は，生成元が線分を辺とする平面グラフで，各節点に接続する辺が 2 個以下である，ともいえる．また，入力の点が別の位置を占めること，線分，折れ線や多角形の辺は長さをもつこと，図形が単純であることはアルゴリズムを実装する計算機によって判定されているとする．そうでなければアルゴリズム自体が意味を持たないからである．生成元集合を $G = \{g_1, \cdots, g_N\}$ とする．

本論文では，点，線分，折れ線，多角形に関する Voronoi 図 $V(G)$ を簡潔に**多角形 Voronoi 図**ということにする．多角形 Voronoi 図の定義は 5.1.1節における一般の Voronoi 図の定義に準じる．また，本節で扱う問題を次のように設定する．

問題 4 単位正方形 $[0,1]^2$ 内に与えられた生成元に対し，多角形 Voronoi 図をこの正方形内にかけ．

線分 Voronoi 図のときと同じように，本節のアルゴリズムでは，正方形 $[-1,2]^2$ を内部に含む三角形の 3 頂点を生成元集合に加えることにする．このような 3 点によって，単位正方形 $[0,1]^2$ の内部で Voronoi 図に影響を与えることなしに，無限にのびる辺を 3 個に限定できる．

ここで，多角形 Voronoi 図の構成のために，分割線分 Voronoi 図を再定義する．生成元集合 G の各生成元 g_i から，

- g_i が点ならば自分自身，
- g_i が線分ならば両端の 2 点，
- g_i が折れ線ならば両端の 2 点と角の頂点全部，
- g_i が多角形ならば角の頂点全部

を集めてできる集合 (点だけからなる) を $G_0 = \{p_1, \cdots, p_m\}$ とする．各 g_i から G_0 の元になっている点を取り除くと，g_i が点の場合は空となり，それ以外の場合は線分から両端の点を除いたものがいくつか得られる．この，両端点を除いた線分を**開線分**とよぶ．各 g_i から G_0 の元を取り除いて得られる開線分の全体を $\{e_1, \cdots, e_n\}$ とする．また，各 k $(0 < k \leq n)$ に対して $G_k = G_0 \cup \{e_1, \cdots, e_k\}$ とかくことにする．$p, q \in G_n$ に対して，多角形 Voronoi 図と

同様に, $R(p,q)$ を以下で定める:
p,q が接続する点と開線分のとき ($\{p,q\}=\{p_i,e_k\}$ とし, e_k の両端点を p_i,p_j とする),

$$B(p_i,e_k) = B(e_k,p_i)=\{x\in\mathbf{R}^2 \mid (x-p_i)\cdot(p_j-p_i)=0\}, \tag{5.90}$$
$$R(p_i,e_k) = \{x\in\mathbf{R}^2 \mid (x-p_i)\cdot(p_j-p_i)<0\}, \tag{5.91}$$
$$R(e_k,p_i) = \{x\in\mathbf{R}^2 \mid (x-p_i)\cdot(p_j-p_i)>0\}. \tag{5.92}$$

ただし, $\cdot$ は2次元の内積を表す. p,q がそれ以外のときは, 一般の Voronoi 図での定義式 (5.2) で定める. $R(q)=\bigcap_{p\neq q}R(q,p)$ とおく. $R(q)$ $(q\in G_k)$ が定める平面の分割を G_k に対する**分割線分 Voronoi 図**とよび, G_k に対する分割線分 Voronoi 図も $V(G_k)$ とかく. 分割線分 Voronoi 図 $V(G_k)$ に対して G_k を**生成元集合**とよぶ. この記法は他の各種の Voronoi 図と同一であるが, 各種の Voronoi 図は生成元集合で区別することにし記号を共用にする. また, 生成元集合の他に, **生成元**, **領域**, **辺**, **節点**などの用語も共用にする.

多角形 Voronoi 図 $V(G)$ を分割線分 Voronoi 図 $V(G_n)$ から得るには, 辺のうち, 両側の生成元の端点や開線分が同一の多角形に属するものを消去するだけですむ (図 5.19). そこで以後では分割線分 Voronoi 図の構成を扱う. また, 特に分割線分 Voronoi 図 $V(G_n)$ において, 開線分とその端点の領域の間の辺を**仮想辺**とよぶことにする.

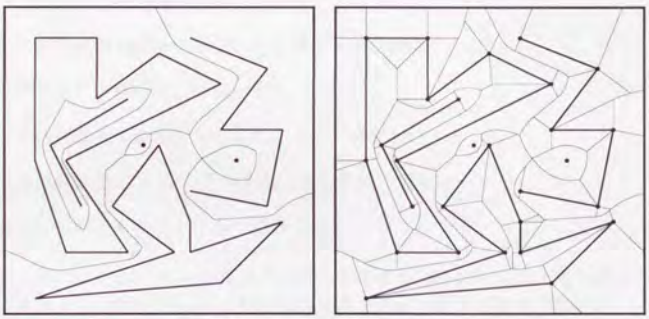


図 5.19: 多角形 Voronoi 図とその分割線分 Voronoi 図.

5.4.2 アルゴリズム

本節で示す多角形 Voronoi 図の構成アルゴリズムは, 5.3節で述べた線分 Voronoi 図の構成アルゴリズム [28] を拡張したものである. 実装システム PLVOR も, SEGVOR を基礎に多角形 Voronoi 図が扱えるように書き改めたものである. 以下, 拡張した部分を中心にそのアルゴリズムと確保した組合せ構造について説明する.

計算誤差を念頭におくので, アルゴリズムを実装する計算機として, 組合せ計算 (整数計算) が正確にでき, 実数計算に誤差はあるが同一の計算には同一の結果を返す計算機を仮定する. また入力の生成元集合はこの計算機で交差しないと判定され, すべての開線分はこの計算機で 0 でない長さが計算されるとする. これらの仮定は計算機の現実に近い.

本節の多角形 Voronoi 図の構成アルゴリズムのもとになった線分 Voronoi 図のアルゴリズムを再び示すと次のようである．開線分を生成元集合に 1 個ずつ加えていく逐次添加型のアルゴリズムである．

アルゴリズム 14 (線分 **Voronoi** 図の構成)
1. 点の Voronoi 図 $V(G_0)$ を作る．
2. $k = 1, \cdots, n$ の順に **2.1** を行なう．
　2.1. 開線分を e_k を添加し $V(G_{k-1})$ を $V(G_k)$ に描き換える．

これで最終的に分割線分 Voronoi 図 $V(G_n)$ を得るというものである．前節で述べたように，この $V(G_n)$ から所要の Voronoi 図は不要な辺を消去するだけで求められる．アルゴリズム 14の 1 の点の Voronoi 図 $V(G_0)$ の構成は位相優先法の既存のアルゴリズム [74] を用いる．したがって，今回拡張した部分は 2.1 の開線分を添加し Voronoi 図を描き替えていく部分である．

線分 Voronoi 図の構成アルゴリズムにおけるこの部分の骨格を以下で説明していく．開線分 e_k を添加するとする．あらかじめ各開線分に任意に向きを付け，その始点，終点の端点が求められるデータ構造を採用しておく．始点側端点を**前端点**，終点側端点を**後端点**とよぶことにする．開線分 e_k の前後の端点を p_i, p_j とする．このときに Voronoi 図の変化する部分は次のような組合せ的な構造をもつ (図 5.20):

- 新しく生じる辺や節点は消去される部分を囲む閉路である，
- その消去される部分は木構造をもつ，
- その木は両端点の領域をつなぐただ一つの道をもつ，
- その道は開線分 e_k の左右にある生成元の領域の間を通る，
- 仮想辺はどの辺も少なくとも一部分は残る．

本節のアルゴリズムではこれらの構造を逆順に利用する．線分 Voronoi 図の構成アルゴリズムでは，開線分 e_k を添加し Voronoi 図を描き替える部分は次のとおりであった:

アルゴリズム 15 (開線分 e_k の添加)
1. 前端点領域 $R(p_i)$ の周上で道の出発点を決める．
2. 探索点を出発点におく．
3. 探索点が後端点領域 $R(p_j)$ に達するまで **3.1** を行ない，道を発見する．
　3.1. 探索点に接続する未探索辺のうち開線分 e_k に関して左右の生成元の領域の間の辺を探索し現探索点でない側の節点をあらためて探索点にする．
4. 道上の節点から消去される辺を探索していき，消去される木構造を決定する．
5. 消去される木構造を囲むように新しい節点や辺を付加し木構造を消去して，添加開線分 e_k の領域を決定する．

ただし，アルゴリズムの 3 の，道を見つけていく課程で領域 $R(p_j)$ に道が達する前に仮想辺や無限にのびる辺を選んだり，閉路を作ったり，出発点の領域 $R(p_i)$ に戻ってきたときに

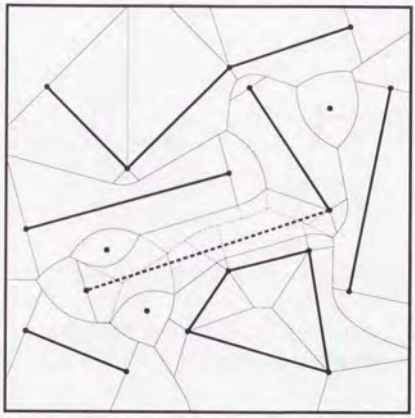


図 5.20: 開線分の添加.

は, 左右の判定のもっとも怪しそうな所から探索をやり直す. また, 道を見つけた後, 4 で木構造を決定するために探索している辺も, 仮想辺や無限にのびる辺に達したり, 閉路を作りそうになったときには強制的にその辺の途中に新しく節点が生じるものとし, 消去される部分が木構造をもつことを確保する. このようにすることで次のような組合せ的な性質が保証された分割線分 Voronoi 図をどのような計算誤差の下でも得ることができる:

- すべての点, 開線分が単連結な領域をもつ,
- 開線分とその端点の領域は必ず仮想辺で隣接する.

この線分 Voronoi 図のアルゴリズムを多角形 Voronoi 図のアルゴリズムに拡張する. このとき, Voronoi 点として, まわりの生成元の種類による型で, 新しいものが導入される:

(端 - 点 - 端) 型: 1 個の点とそれに接続する 2 個の開線分.

この型の節点は生成元の点と同一座標値をもつので, 座標値の計算に関して特に問題は起こらない.

まず, 開線分 e_k の添加前に, 前端点 p_i で e_k に隣接する開線分 e_l が添加されているとする. その後, e_k が添加されるとき, アルゴリズム 15の 1 で道の出発点を決めるところでは, 前端点領域 $R(p_i)$ の周上で仮想辺に接続する節点のうち片方は絶対に出発点にならない (図 5.21). 終点領域 $R(p_j)$ でも同様に, e_k の添加前に p_j に接続する他の開線分が添加されるならば, 絶対に道の終点にならない節点があることがいえる. 多角形 Voronoi 図の構成アルゴリズムではこれらの節点を道の探索で採用しないことにした. もう一つ問題となるのは, 線分 Voronoi 図とは違い多角形 Voronoi 図の構成では, 計算誤差によって仮想辺を経由しないと, アルゴリズム 15の 3 で道の探索が行き詰まる場合があることである (図 5.22). これは, 開線分添加の過程で 1 個の多角形のすべての辺が添加され, 他の未添加の開線分の端点がこの多角形の内部と外部に分離されてしまうからである. このため, 入力の開線分に前処理を

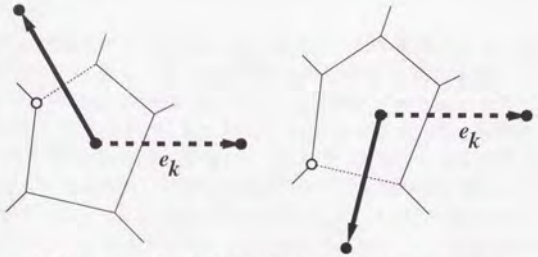


図 5.21: 出発点にならない節点 (白丸)(点線は仮想辺)

して, 多角形を閉じる辺を添加順序で最後になるよう並べかえるようにした. このようにして, どのような計算誤差でも正常終了する多角形 Voronoi 図の構成アルゴリズムを得ることができる.

図 5.22: 多角形で分離される端点 (破線は未添加の辺)

浮動小数点の数値計算に関する工夫や変更点は 5.4.3節に譲る.

他には, 例えば入出力の書式を変更した. これは, SEGVOR のように, 線分 Voronoi 図の場合には, 初めの 2 点が第 1 線分, 次の 2 点が第 2 線分, ··· のように解釈して, 頂点座標の列で入力を簡単に済ませることができるが, PLVOR のように, 多角形 Voronoi 図の場合, 入力として点の座標値の他に開線分と端点の接続関係も必要になり, 入力はより複雑になるからである. 出力も同様である. そのため, 実装の結果, 同一の生成元集合に対して, PLVOR での入出力のサイズは SEGVOR の約 2 倍になった. このようなアルゴリズム内部に大きく影響しない変更点に関しては詳細を省く.

計算誤差の存在を認めるということは, 入力退化を発見できないことを意味する. したがって, 本節のアルゴリズムでは入力退化のための特別な処理を用意していない. しかしながら, 本節のアルゴリズムは, 退化した入力に対しても非退化入力と同一の処理をして正常終了する.

5.4.3 浮動小数の数値計算

本章の冒頭で述べたように, 位相優先法はアルゴリズムを意識的に数値計算部分と組合せ構造構築の部分にわけるので, 絶対に正常終了し, 組合せ的な構造を保証するアルゴリズムを得たのちに, 数値計算部分に計算上の工夫を加えることができる. 線分 Voronoi 図における数値計算上の工夫は前節にあるので, 本節では, 線分 Voronoi 図から多角形 Voronoi 図

へと構成アルゴリズムを拡張したときにSEGVORに変更を加え，PLVORに採用した数値計算部分について述べる．

線分Voronoi図の構成アルゴリズムでは，探索中の道の先端の節点vから生成元の点や開線分が添加開線分e_kに対し左右どちらにあるかを判定し，次に探索する辺を決定している．この左右判定では生成元が開線分e_lのときには節点vからe_lに下ろした垂線の足をpとし，生成元が点のときにはその点をpとして，pがe_kから左右どちらにどのくらい離れているか計算してその値を左右判定の度合いとしている(図5.23)．特に前者の場合，すなわちpが垂線の足であるときには多くの計算が必要で，誤差の影響を強く受けると考えられる．そこでe_kとe_lが隣接するときには垂線の足を求めず，この2個の開線分の端点をp_i, p_j, pとして，pのe_kに対する左右判定を行ない，左右判定の度合いとしてはより信頼性が高いと判断し，一定の値を与えることにした．

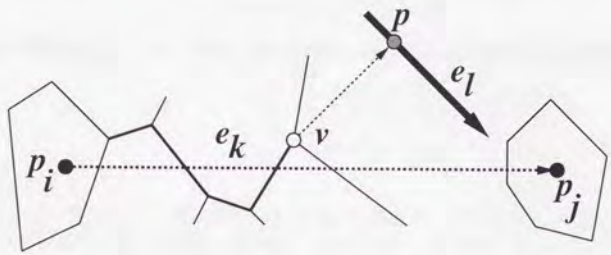


図 5.23: 生成元の開線分に対する左右判定．

また，(線，線，線) 型のVoronoi点の座標値$(X, Y)/Z$(同次座標)の計算で，Voronoi点のまわりの生成元のうち，2個の開線分が1個の端点を共有し，それらの作る角がπであるとき，線分Voronoi図のときに採用した計算法では厳密に計算すると$Z = 0$となり座標値を確定できない．このため，座標値を求めるVoronoi点のまわりの生成元のうち，2個の開線分が1個の端点を共有した場合，以下のように別計算をすることにした．

今，座標値を計算したいVoronoi点を生成元e_i, e_j, e_kがこの順に反時計周りに囲むとする．局所的に変数の添字の付け替えをして，隣接している2個の開線分をe_i, e_jとし，e_iの端点をp_1, p_2, e_jの端点をp_2, p_4とする．すなわち，p_2が2個の開線分e_i, e_jの端点として共有されているとする．これでe_i, e_jに対して，p_1, p_2, p_4は一意に定まる．残りの開線分e_kの端点をp_5, p_6とする．p_5, p_6はp_2, p_5, p_6が正の向きに並ぶようにとる．これでe_kに対して，p_5, p_6は一意に定まる(図5.24)．各p_iの座標値を(x_i, y_i)とする．また，求めるVoronoi点をv，その座標値を(x, y)とする．開線分e_i, e_j, e_kを延長してできる三角形を考えると，e_iとe_jが隣接しているので，点vは，この三角形の内心か三角形の頂点p_2に関する傍心の2通りの場合しかない．

開線分e_i, e_j, e_kの長さd_i, d_j, d_kを

$$d_i \leftarrow \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}, \tag{5.93}$$

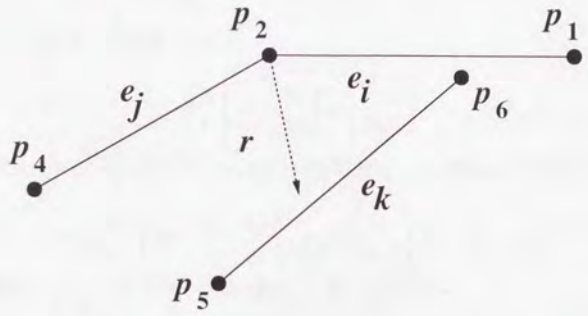


図 5.24: 隣接開線分に関する (線, 線, 線) 型の Voronoi 点を求めるための記号割当

$$d_j \leftarrow \sqrt{(x_4 - x_2)^2 + (y_4 - y_2)^2}, \tag{5.94}$$

$$d_k \leftarrow \sqrt{(x_6 - x_5)^2 + (y_6 - y_5)^2} \tag{5.95}$$

で求めることにする.

開線分 e_i, e_j の作る角の 2 等分線の向きに沿った任意のベクトルを $r = (x_r, y_r)$ とする (図 5.24). 具体的な値は後で定める. Voronoi 点 v はこの 2 等分線の上にあるから, 実数 t を用いて,

$$v = p_2 + tr \tag{5.96}$$

とかける. この式を, 点 v が開線分 e_i, e_j, e_k から等距離にあることを示す式

$$\frac{1}{d_i}\begin{vmatrix} x_1 & x_2 & x \\ y_1 & y_2 & y \\ 1 & 1 & 1 \end{vmatrix} = \frac{1}{d_j}\begin{vmatrix} x_2 & x_3 & x \\ y_2 & y_3 & y \\ 1 & 1 & 1 \end{vmatrix} = \frac{1}{d_k}\begin{vmatrix} x_4 & x_5 & x \\ y_4 & y_5 & y \\ 1 & 1 & 1 \end{vmatrix} \tag{5.97}$$

に代入して t を求めることを考える. この式は v が内心, 傍心のどちらの場合にも成立する. このままでは条件過剰なので, e_i と e_j の対称性を考慮して, 左側 2 辺の和と最右辺の 2 倍による等式

$$\frac{1}{d_i}\begin{vmatrix} x_1 & x_2 & x \\ y_1 & y_2 & y \\ 1 & 1 & 1 \end{vmatrix} + \frac{1}{d_j}\begin{vmatrix} x_2 & x_3 & x \\ y_2 & y_3 & y \\ 1 & 1 & 1 \end{vmatrix} = \frac{2}{d_k}\begin{vmatrix} x_4 & x_5 & x \\ y_4 & y_5 & y \\ 1 & 1 & 1 \end{vmatrix} \tag{5.98}$$

を利用することにする. この式に $v = p_2 + tr$ を代入して t について整頓した結果,

$$t' = \frac{d_k \begin{vmatrix} d_j x_1 - d_i x_4 & x_2 & x_r \\ d_j y_1 - d_i y_4 & y_2 & y_r \\ d_j - d_i & 1 & 0 \end{vmatrix}}{\begin{vmatrix} x_5 & x_6 & x_2 \\ y_5 & y_6 & y_2 \\ 1 & 1 & 1 \end{vmatrix} \cdot 2} - d_i d_j \tag{5.99}$$

$$t \leftarrow \frac{1}{t'} \tag{5.100}$$

を得る. これを用いて v の座標 (x, y) を

$$\begin{bmatrix} x \\ y \end{bmatrix} \leftarrow \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} + t \begin{bmatrix} x_r \\ y_r \end{bmatrix} \tag{5.101}$$

で求める. 次にベクトル r の計算について述べる. 開線分 e_i, e_j に沿った向きのベクトル r_i, r_j を

$$r_i = \begin{bmatrix} x_{r_i} \\ y_{r_i} \end{bmatrix} \leftarrow \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix},\ r_j = \begin{bmatrix} x_{r_j} \\ y_{r_j} \end{bmatrix} \leftarrow \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix} \tag{5.102}$$

と定める. 開線分 e_i, e_j の作る角の 2 等分線は, 一般に, ベクトル

$$r = d_j r_i + d_i r_j \tag{5.103}$$

に沿った向きをもつ. しかしながら, このように r を定めると e_i, e_j の作る角が π に近い場合, r の成分に激しい桁落ちが発生する (図 5.25). 桁落ちを食い止めるため, このようには r をとらず, p_1, p_2, p_4 が正の向きに並ぶ場合には

$$r_i' \leftarrow \begin{bmatrix} x_{r_i} \\ y_{r_i} \end{bmatrix} + \begin{bmatrix} y_{r_i} \\ -x_{r_i} \end{bmatrix},\ r_j' \leftarrow \begin{bmatrix} x_{r_j} \\ y_{r_j} \end{bmatrix} + \begin{bmatrix} -y_{r_j} \\ x_{r_j} \end{bmatrix} \tag{5.104}$$

とし, p_1, p_2, p_4 が正の向きに並ばない場合には

$$r_i' \leftarrow \begin{bmatrix} x_{r_i} \\ y_{r_i} \end{bmatrix} + \begin{bmatrix} -y_{r_i} \\ x_{r_i} \end{bmatrix},\ r_j' \leftarrow \begin{bmatrix} x_{r_j} \\ y_{r_j} \end{bmatrix} + \begin{bmatrix} y_{r_j} \\ -x_{r_j} \end{bmatrix} \tag{5.105}$$

として, r を

$$r \leftarrow d_j r_i' + d_i r_j' \tag{5.106}$$

と定めることにする.

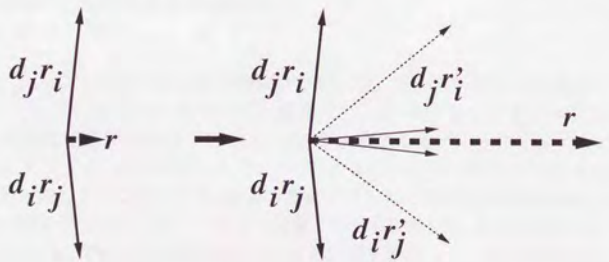


図 5.25: 2 等分線に沿ったベクトルの求め方

このようにしても，ベクトル r は e_i と e_j の2等分線に沿った向きをもち，さらに r'_i, r'_j の成す角は $\pi/2$ を超えないので，実際上桁落ちの影響を食い止めることができる (図5.25)．ここまでの計算で r が0になるときには，開線分 e_i, e_j, e_k が点 p_2 のまわりに密集していると解釈し，Voronoi 点 p は p_2 と同一の場所を占めるとした．その他の分母が0になる場合にはVoronoi 点が生じないとして扱った．

(線, 線, 線) 型以外の Voronoi 点の座標値の計算など，他の浮動小数の数値計算部分は，線分 Voronoi 図での計算法を変更せず利用している．

5.4.4 計算量と記憶量

多角形 Voronoi 図の構成アルゴリズムの計算量について考える．多角形 Voronoi 図に対応する分割線分 Voronoi 図では線分 Voronoi 図と同じく，節点，辺の総数は領域の総数の線形オーダである．線分 Voronoi 図の構成アルゴリズムにかかる時間は，生成元数を n とすると，点 Voronoi 図の構成に $O(n^2)$，すべての開線分の添加に $O(n^3)$ かかり，探索のやり直しが起こらない程度の計算精度があれば開線分の添加も $O(n^2)$ ですむ [28]．記憶量は $O(n)$ である．多角形 Voronoi 図ではアルゴリズムの拡張は計算量と記憶量のオーダにかかわらない部分に施されている．したがって分割線分 Voronoi 図の生成元の点と開線分の総数を N とすると，アルゴリズムの計算量は最悪で $O(N^3)$，ある程度の計算精度があれば $O(N^2)$ である．また記憶量は $O(N)$ である．計算量は最適の $O(N \log N)$[5] に及ばないが，従来の逐次添加型の構成アルゴリズムの計算量と同等である．記憶量の $O(N)$ は最適である．

アルゴリズムを実装したシステム PLVOR によって，実際の計算量を測定する実験を行なった．入力は多角形だけからなるもので，基本的に次のアルゴリズムで得た．

アルゴリズム 16 (入力多角形集合の構成)
1. 端点座標を $[0,1]$ 内の一様乱数で発生させ，端点 $p_1, \cdots, p_n$ をとる．
2. 初期開線分 $e_1, \cdots, e_n$ を，$e_i = (p_i, p_{i+1})$ $(i = 1, \cdots, n-1)$; $e_n = (p_n, p_1)$ にとる．
3. 交差開線分対 (e_i, e_j) が見つかるうちは，**3.1** を実行する．
3.1. e_i, e_j の後端点を交換しても二角形が生じないとき，
e_i, e_j の後端点を交換する．
そうでないとき，
e_i の前端点と e_j の後端点を交換する．
4. 全線分を出力して終了．

ただし，ここで二角形とは2個の開線分であって，向きを無視して両端点が一致するものをいう．アルゴリズム 16の **3** での交差線分対 (e_i, e_j) の探索には2通りの方式をとった．第1の方式は交差開線分対の添字対 (i, j) の辞書式順序で最小のものを探すもので，**総当り方式**とよぶことにする．第2の方式はアルゴリズム 1 [7, 65] に準じた平面走査法によるもので，**平面走査方式**とよぶことにする．両者の方式で得られた開線分の総数 1024 での分割線分 Voronoi 図の例を図5.26に示す．一見して開線分の分布に差があることがわかる．各方式，各線分数ごとに10個の異なる生成元集合での計算時間を測定した．結果は各方式，各線分数とも計算時間のばらつきが小さかったので，計算時間の平均をとり結果を表5.2に示す．両方式で若干の差はあるが，計算量は開線分数の2乗のオーダよりむしろ線形に近いことがわかる．

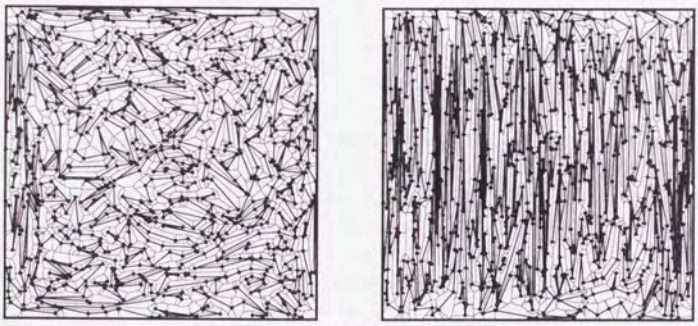


図 5.26: 総当り方式 (左) と平面走査方式 (右)(開線分数 1024).

表 5.2: 計算機実験の結果 (単位は秒).

開線分数	256	512	1024	2048	4096
総当り方式	2.94	5.90	11.87	24.50	51.78
平面走査方式	3.11	6.31	12.73	26.64	57.19

第 6 章

まとめ

本論文では, 計算幾何学で開発されてきた, 効率的なはずのアルゴリズムが多くの場合, 実用にならないという問題点を指摘し, その原因である計算誤差と退化に関する研究を行なった.

計算機の計算精度は有限である. これはふたつの意味をもつ. すなわち, 第 1 の意味は, 理論的な面から, 計算機に有限の記憶量しか与えられていない以上, 計算できる対象は有限個に限られ, アルゴリズム設計者が想定しているすべての場合を扱うには無理があるということである. 第 2 の意味は, 実用面から, 任意の実数データを浮動小数形式で受け入れ, 計算誤差付きで処理しなければ, 計算速度と既存システムとの整合性の点から, 実用性を欠くということである.

しかしながら, 従来, アルゴリズムの理論展開や設計時には無限精度の計算が仮定されることが多い. そうして, 実数値の入力が無限精度で加工できることになると, 次は, 判定式が 0 になる場合を "滅多にないこと" として無視できるようになる. このため, 実際には無視できない計算精度の有限性により, 計算誤差と退化の問題は実用時まで放置されることになる.

本論文では, 計算誤差と退化の問題を取り上げ, 幾何的アルゴリズムの設計の段階から, これらの問題を考慮することにより, アルゴリズムの設計から, 実装, 実用まで, 計算誤差と退化に "その場しのぎ" でない対処法を総合的に研究し, 提案した.

まず 2章で, 計算幾何学での既存の研究をふまえて, 幾何的アルゴリズムで扱うデータを位相データと計量データに分け, 退化を計量データを変数とする判定式の零点として定義した. また, 判定式から除算と開平を消去し判定式を多項式に限定できることも示した. この退化の定義により, 入力を定数ビット以下に制限することで, 無誤差計算が有限精度でもできることを示した. これで, 退化の問題を多項式の零点に関する問題として捕えた.

次に 3章で, 代数的に退化を扱う枠組みにおける退化対処法を, 代数的退化対処法と名付けた. 次に, Yap, Edelsbrunner ら, Emiris ら, Fortune による, 既存の退化対処法を紹介し, 前節の枠組みのなかで, 項順序による多項式環の順序環化, 記号的無限小摂動による零点の解消という手法として, 統一的な解釈を行なった. 退化対処法の簡潔さと適用範囲は裏腹の関係にあり, Yap の方法が最も適用範囲が広く, Edelsbrunner ら, Emiris ら, Fortune と続くが, 簡潔さでは, Fortune が最も優れ, Emiris ら, Edelsbrunner ら, Yap と続く. 適用する場面に応じて使うべき方法は別れるであろう. 続いて, 超準解析の手法を用いて, 無限小摂動の解釈を行ない, 整数論と集合論を基礎におく数学構造から幾何的アルゴリズムを構成する限り, 代数的退化対処法は本質的に記号的無限小摂動法以外にないことを示し, 代数的退化対処法のある種の限界を示した. とはいえ, 同じ枠組みの退化対処法で, 様々な方法が考案できることは, 注目に値する. 今後も研究の発展の余地があると思われる. 続いて, Gröbner 基底

を利用して，新しい退化対処法を考案した．既存の方法が，前処理として実質的に判定多項式の微分式を計算し，幾何的アルゴリズムを書き換える，いわばコンパイラ方式なのに対しこの方法は，アルゴリズムの実行中に判定多項式の微分式に相当するものを計算していく，いわばインタプリタ方式をとる．この方式はコンパイラ方式と異なり，非退化時にも余分な計算が必要な点が明らかに欠点であるが，適用範囲が広く簡潔なため実装が楽で，既に実装されている幾何的プログラムを気楽に退化対処版にしたい場合には有利である．

次に4章で，剰余演算を利用した無誤差計算について取り上げた．代数的退化対処法は，高精度の無誤差計算を前提にする．無誤差計算に関しては，理論的には計算桁数を十分に多くとれば達成できる．しかし，実用の観点からは，長すぎる桁数の計算は計算時間の増大を引き起こし，退化対処法を組み込む前に，アルゴリズムが計算時間的に非実用的な実装になってしまう．本章では，必要十分な桁数を求めることの困難を指摘した．さらに，計算の結果，必要なものが値ではなく符号であることに着目して，剰余演算を用いて長い桁数の数の加減乗算の高速化し，計算して得られた剰余から，値を復元することなく符号を復元する方法を提案した．さらに，予備的な計算機実験を行ない，実際の効果を確認した．

次に5章で，もう一つの退化対処法である位相優先法の解説と，位相優先法に基づく幾何的アルゴリズムの開発を行なった．位相優先法は計算誤差の存在を前提にする方法であり，退化の有無を厳密に判定することなく自然に退化を回避する．問題によって保持すべき位相構造が異なるので，概論的な方法論だけではすぐに利用するというわけにはいかず，実際に役に立つアルゴリズムを構築するためには，具体的な事例ごとの工夫が必要である．本章では，位相優先法の具体例として，新規に開発した線分 Voronoi 図と多角形 Voronoi 図の構成アルゴリズムをその実装プログラムを中心に解説した．そのために必要な (点)Voronoi 図の構成アルゴリズムに関しても概説した．また，実装した線分 Voronoi 図，多角形 Voronoi 図の構成プログラム SEGVOR, PLVOR に関しては，各種の計算機実験を行ない，数値的安定性を実証し，また，実用に足りる計算速度を確保していることを示した．

総合的に代数的退化対処法と位相優先法を比較する．幾何的アルゴリズムを実際に用いるときには，個別の場合に応じて求められるものが異なるであろう．理想は，簡潔なプログラムで無限精度計算をしたものと同一の計量データと矛盾のない位相データが，出力として高速に求められることである．しかし，精度を上げることは計算するビット数を増すことであるから，高速化とは相入れない．従って，計算精度と計算時間のどちらをとるかという問題になる．代数的退化対処法は，判定多項式の符号判定を正確に行なうことを前提とする．このために高精度計算が不可欠である．前の章で示したように剰余演算を用いれば m 倍長の整数の加減乗算と符号判定は簡略化と高速化が計れるが，浮動小数での計算に匹敵するものではない．これは，代数的退化対処法による幾何的アルゴリズムは，本質的に計算時間が余計にかかることを意味する．しかしながら代数的退化対処法は，無限小摂動を与えることに相当する方法なので，出力の位相データは必ず，ある非退化入力によって実現できるものになる．一方，位相優先法は，一般に，位相データの持つべき性質のすべてを保証するわけではない．アルゴリズム設計者が選んだ性質だけが保証される．その意味では，どのような非退化入力データに対する出力にもなり得ないような位相データが得られる場合もある．しかしながら，計算誤差を前提とすることは，判定式の符号決定のために浮動小数計算どころか，どのような誤差をもつ計算をしてもアルゴリズムの実行が破綻しないことを意味するので，計算時間の点では絶対に有利である．

より実用的側面を考慮すると，代数的退化対処法では Fortune の指摘のように，判定多項

式の値が0に近いときに初めて，浮動小数計算から多倍長整数計算に切り替えるのが計算時間の点では有効であろう．ただし，実装は複雑になりがちで，解くべき幾何的問題が複雑になればなるほど，浮動小数計算から多倍長整数計算に切り替える境の見積りが急激に難しくなるので，見積りはあまくなる．そのため，切り替える場合が増え，実装の複雑さと計算時間は共に急激に増大するという問題が残る．位相優先法では計算精度を上げていけば，判定多項式の値の符号の誤判定は少なくなるであろう．出力図形の構造にある程度の精度を求めるなら，計算時間と精度の間でバランスをとることができ，代数的退化対処法よりも柔軟性がある．5章でも述べたように，アルゴリズムを設計してから浮動小数計算部分を変更していくという柔軟性や，位相優先法が保証する図形の性質の選択に任意性もある．これは，位相優先法の利点であるが，同時に，最適な幾何的アルゴリズムの決定が難しいことも意味する．このように，代数的退化対処法と位相優先法のどちらも，実用上の改良を加えることができるが，基本的性格からくる困難を除くには至らない．この基本的性格を退化対処への接近法として要約すると，代数的退化対処法は計算誤差を排除する点で，いわば理論側からの接近法であり，一方，位相優先法は位相的な性質の保証しながら従来の方法を利用する点で，いわば実際側からの接近法ともいえる．

現実の要請は多様であり，代数的退化対処法と位相優先法のそれぞれに得失があり，優劣がつくものではない．それぞれの特性を考慮すると，出力図形の構造的無矛盾性を完全に求めるなら計算時間を無視して代数的退化対処法を，必要な性質だけ確保し従来程度の計算時間を要求するならば位相優先法を選択すべきであろう．どちらを選択した場合にも，実用的にはただ適用するだけでなく，精度と計算時間に関して改善することが望ましいのはいうまでもない．しかし，退化や計算誤差に考慮のない幾何的アルゴリズムのそのままの実装プログラムや，その場しのぎ法に莫大な労力をかけながら対処法として不十分な実装プログラムが通用している現状と比較して，代数的退化対処法と位相優先法のどちらも，プログラマの退化に対処するための労力をなくし，プログラム本体の開発に労力を集中させることができる点と，理論的に効率がよい幾何的アルゴリズムを実際にも利用できるという点で，格段に優れているといえる．

本論文の研究で，線分Voronoi図，多角形Voronoi図を構成するプログラム SEGVOR, PLVOR が具体的なソフトウェアとして得られたことは，代数的退化対処法と位相優先法の理論展開とは別に，それ自体が成果である．SEGVOR, PLVOR は試作段階で，数値計算の方法やデータの持ち方などに検討する余地が残るが，それでも，これまでに，広く知られたソフトウェアで，計算時間と数値的安定性において，SEGVOR, PLVOR 程度の実用性を持つものはない．したがって，SEGVOR, PLVOR が幾何的アルゴリズムの応用分野で活用されていくものと思われる．

最後に，今後の課題についてまとめることにする．代数的退化対処法では，効率的で簡素な方法を開発することが課題である．剰余演算を利用した符号判定では m 倍長整数の符号判定に必要な計算量の理論的解明とその計算量で符号判定を行なうアルゴリズムの開発が課題である．代数的退化対処法は，文字通り代数的な理論と手法であるが，位相優先法は，図形の持つ位相的性質に着目する点で幾何的であるといえる．2章で述べたように，計算幾何学が問題の幾何的側面に注目してアルゴリズム論の中で特別にアルゴリズムの効率化を果たすことを考えると，位相優先法には，幾何的問題特有の退化対処法として研究の余地があろう．前の章で述べたように，判定式の値が0になるときのすべてが，問題に関して対処すべき退化であるかも検討する余地がある．さらに，個別の応用上の需要が高い幾何的問題に対して，代

数的退化対処法や位相優先法による数値的に安定な幾何的アルゴリズムとその実装プログラムの開発，整備も重要な課題である．このように，また，この他にも論文中で随時述べたように，有限精度計算を前提とする幾何的アルゴリズムに関する研究は，着手されはじめたばかりであり，多くの課題が残されている．この分野で，実際に役立つ計算幾何学の構築にむけて，今後の研究の発展が不可欠であろう．

謝辞

本研究を進めるにあたり，多くの方々のお世話になった．特に，杉原厚吉教授には研究の全領域・全期間にわたって多くの指導をいただいた．伊理正夫教授 (現在，名誉教授，中央大学教授) にも学生時代から論文をまとめるまでにわたり，多くの有用な助言をいただいた．速水謙助教授にも論文をまとめるにあたり多くの貴重なご意見をいただいた．久保田光一助手 (現在，中央大学助教授)，富岡豊助手 (現在，松下電器産業 (株))，金子敬一助手には，計算機の使い方など研究の実務から研究の細部にわたる討論まで，大変にお世話になった．研究室の大学院の学生の皆様にも，多くの討論とご意見をいただき大変に参考になった．このほかにも研究の各局面で多くの方々から様々な助言をいただいた．これらのすべての方々に感謝いたします．

参考文献

[1] Aho, A. V., Hopcroft, J. E. and Ullman, J. D.: *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[2] 浅野孝夫, 今井浩, 伊理正夫: 計算幾何学, *bit*, Vol. 16, No. 12–Vol. 17, No. 3, 共立出版, 1984–1985.

[3] 浅野孝夫, 今井浩: 計算とアルゴリズム — 計算機の科学 —, オーム社, 1986.

[4] 浅野哲夫: 計算幾何学, 朝倉書店, 1990.

[5] Aurenhammer, F.: Voronoi Diagrams – A Survey of a Fundamental Geometric Data Structure, *ACM Comput. Surv.*, Vol. 23, pp. 345–405, 1991.

[6] Avis, D., 今井浩, 松永信介: 計算幾何学・離散幾何学, 朝倉書店, 1994.

[7] Bentley, J. L. and Ottmann, T.: Algorithms for Reporting and Counting Geometric Intersections, *IEEE Trans. Comput.*, C-28, 643–647, 1979.

[8] Cox, D., Little, J. and O'Sha, D.: *Ideals, Varieties and Algorithms*, Springer-Verlag, 1992.

[9] Cutland, N. (ed.): *Nonstandard Analysis and its Applications*, LMSST10, Cambridge University Press, 1988.

[10] Davis, M.: *Applied Nonstandard Analysis*, John Wiley & Sons, 1992.

[11] Ebbinghaus, H. -D., et al. (eds.): *Zahlen*, Springer-Verlag, 1988.

[12] Edelsbrunner, H.: Edge-Skeletons in Arrangements with Applications, *Algorithmica*, 1:93–109, Springer-Verlag, 1986.

[13] Edelsbrunner, H.: *Algorithms in Computational Geometry*, Springer-Verlag, 1987.

[14] Edelsbrunner, H. and Mücke, E. P.: Simulation of Simplicity: A Technique to Cope with Degenerate Cases in Geometric Algorithms, *Proc. 4th ACM Ann. Symp. on Computational Geometry*, pp. 118–133, 1988.

[15] Emiris, I. and Canny, J.: An Efficient Approach to Removing Geometric Degeneracies, *Proc. 8th Ann. Symp. on Computational Geometry*, pp. 74–82, 1992.

[16] Fortune, S.: Stable Maintenance of Point-Set Triangulations in two Dimensions, *Proc. 30th IEEE Ann. Symp. on Foundations of Computer Science*, pp. 494–499, 1989.

[17] Fortune, S.: Polyhedral Modelling with Exact Arithmetic, *Proc. 3rd IEEE Symp. Solid Modelling and Applications*, 1995.

[18] Guibas, L., Salesin, D. and Stolfi, J.: Epsilon Geometry—Building Robust Algorithms from Imprecise Calculations, *Proc. 5th ACM Ann. Symp. on Computational Geometry*, pp. 208–217, 1989.

[19] Greene, D. H. and Yao, F.: Finite-Resolution Computational Geometry, *Proc. 27th IEEE Ann. Symp. on Foundations of Computer Science*, pp. 143–152, 1986.

[20] 畑口剛之, 今井敏行: 記号摂動法と剰余演算による符号判定法の応用, 日本応用数理学会平成 7 年度年会講演予稿集, pp. 204–205, 1995.

[21] Hataguchi, T. and Sugihara, K.: Exact Algorithm for Minkowski Operators, *Proc. 2nd Asian Conf. on Comput. Vision*, Vol. 3, pp. 151–155, 1995.

[22] 畑口剛之: Minkowski 演算の高速かつ安定なアルゴリズムの研究, 東京大学大学院工学系研究科計数工学専攻修士論文, 1996.

[23] Held, M.: *On the Computational Geometry of Pocket Machining*, LNCS500, Springer-Verlag, 1991.

[24] 今井敏行, 線分 Voronoi 図の構成算法における計算誤差対策, 東京大学大学院工学系研究科計数工学専攻修士論文, 1989.

[25] 今井敏行, 杉原厚吉: 組合せ構造を優先した線分ボロノイ図の構成法, 情報処理学会研究報告, 89-AL-11-2, 1989.

[26] 今井敏行: 計算誤差に強い線分 Voronoi 図の構成法, 日本応用数理学会平成 4 年度年会研究発表予稿集, pp. 243–244, 1992.

[27] 今井敏行: 計算誤差に強い多角形 Voronoi 図の構成法, 日本応用数理学会平成 6 年度年会講演予稿集, pp. 40–41, 1994.

[28] 今井敏行, 杉原厚吉: 誤差による破綻の心配のない線分 Voronoi 図構成算法, 情報処理学会論文誌, Vol. 35, No. 10, pp. 1966–1976, 1994.

[29] 今井敏行: 剰余演算を利用した多項式の符号判定と計算幾何学への応用, 情報処理学会研究報告, 94-AL-39, pp. 17–24, 1994.

[30] 今井敏行: 多項式の符号判定のための剰余演算の利用法と計算幾何学への応用, 日本応用数理学会論文誌, Vol. 5, No. 2, pp. 11–18, 1995.

[31] 今井敏行: 組合せ構造を優先した多角形 Voronoi 図の構成法, 情報処理学会研究報告, 95-AL-45, pp.17–24, 1995.

[32] 今井浩, 今井桂子: 計算幾何学, 情報数学講座, 共立出版, 1994.

[33] Inagaki, H., Sugihara, K. and Sugie, N.: Numerically Robust Incremental Algorithm for Constructing Three-Dimensional Voronoi Diagrams, *Proc. 4th Canadian Conf. on Computational Geometry*, pp. 334–339, 1992.

[34] 伊理正夫, 野崎昭弘, 野下浩平 (編著): 計算の効率化とその限界, 数学セミナー増刊, 日本評論社, 1980.

[35] 伊理正夫: 数値計算の常識, 共立出版, 1985.

[36] 伊理正夫 (監修), 腰塚武志 (編集): 計算幾何学と地理情報処理, bit 別冊, 共立出版 1986.

[37] 伊理正夫, 杉原厚吉: 計算誤差を考慮した幾何的アルゴリズム, 情報処理学会研究会研究報告, 88-AL-1-1, 1988.

[38] 伊理正夫 (監修), 腰塚武志 (編集): 計算幾何学と地理情報処理 第 2 版, 共立出版, 1993.

[39] 石畑清: アルゴリズムとデータ構造, 岩波講座ソフトウェア工学, 岩波書店, 1989.

[40] 釜江哲朗: ノンスタンダード・アナリシス, 数学セミナー, 1986 年 6 月号, pp. 20–30, 日本評論社, 1986.

[41] 釜江哲朗: 超準的手法にもとづく確率解析入門, 朝倉書店, 1990.

[42] Karasick, M., Lieber, D. and Nackman, L. R.: Efficient Delaunay Triangulation Using Rational Arithmetic, *ACM Trans. Graphics*, Vol. 10, pp. 71–91, 1991.

[43] Keisler, H. J.(斎藤正彦 (訳)): 無限小解析の基礎, 東京図書, 1979.

[44] Keller-Gehrig, W.: Fast Algorithms for the Characteristic Polynomial, *Theoretical Computer Science*, Vol. 36, pp. 309–317, 1985.

[45] Klein, R.: *Concrete and Abstract Voronoi Diagrams*, LNCS400, Springer-Verlag, 1989.

[46] Knuth, D. E.: *The Art of Computer Programming*, Vol. 2, *Seminumerical Algorithms*, 2nd ed., Addison-Wesley, 1981.

[47] 小久保岩生: 一般化 Voronoi 線図の構成算法の研究 — 特に線分に対する Voronoi 線図について, 東京大学大学院工学系研究科計数工学専門課程修士論文, 1985.

[48] Milenkovic, V.: Verifiable Implementations of Geometric Algorithms Using Finite Precision Arithmetic, *Artif. Intell.*, Vol. 37, pp. 377–401, 1988.

[49] 皆川剛, 杉原厚吉: 位相優先分割統治法による 3 次元凸包の構成, 日本オペレーションズ・リサーチ学会 1995 年度春季研究発表会アブストラクト集, pp. 18–19, 1995.

[50] 皆川剛: 位相優先逐次添加型 2 次元 Voronoi 図構成算法, 日本応用数理学会平成 7 年度年会講演予稿集, pp. 210–211, 1995.

[51] 皆川剛: 3次元凸包構成問題に対する分割統治算法の数値的安定化, 東京大学大学院工学系研究科計数工学専攻修士論文, 1996.

[52] 森田紀一: 位相空間論, 岩波全書, 岩波書店, 1981.

[53] 中村憲: 最近の計算機代数の理論と応用, 数学, Vol. 48, No. 1, pp. 12–21, 1996.

[54] 大屋隆生: Voronoi線図の効率的構成法に関する研究, 東京大学大学院工学系研究科計数工学専門課程修士論文, 1982.

[55] Ohya, T., Iri, M. and Murota, K.: a Fast Voronoi-Diagram Algorithm with Quaternary Tree Bucketing, *Information Processing Letters 18*, pp. 227–231, North-Holland, 1984.

[56] Ohya, T., Iri, M. and Murota, K.: Improvements of the Incremental Method for the Voronoi Diagram with Comparison of Various Algorithms, *J. the Operations Research Society of Japan*, Vol. 27, No. 4, pp.306–337, 1984.

[57] Okabe, A. and Suzuki, A.:Stability of Spatial Competition for a Large Number of Firms on a Bounded Two-Dimensional Space, *Environment and Planning A*, Vol. 19, pp. 1067–1082, 1987.

[58] Okabe, A., Boots, B. and Sugihara, K.: *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, John Wiley & Sons, 1992.

[59] Ottmann, T., Theimt, G. and Ullrich, C.: Numerical Stability of Geometric Algorithms. *Proc. 3rd ACM Conf. on Computational Geometry*, pp. 119–125, 1987.

[60] Oishi, Y.: Topology-Oriented Divide-and-Conquer Algorithm for Voronoi Diagrams, *Graphical Models and Image Processing*, Vol. 57, No. 4, pp. 303–314, 1995.

[61] Preparata, F. P. and Shamos, M. I.: *Computational Geometry*, Springer-Verlag, 1985.

[62] Robbiano, L.: Term Orderings on the Polynominal Ring, *Proc. EUROCAL'85*, LNCS204, Springer-Verlag, pp. 513–517, 1985.

[63] 坂元宗和, 高木幹雄: 加重ボロノイ分割, 電子情報通信学会技術研究報告, PRU87-49, pp. 21–30, 1987.

[64] 佐々木建昭, 今井浩, 浅野孝夫, 杉原厚吉: 計算代数と計算幾何, 岩波講座応用数学, 岩波書店, 1993.

[65] Sedgewik, R.: *Algorithms*, 2nd ed., Addison-Wesley, 1988.

[66] 白柳潔, Sweedler, M.: 代数的アルゴリズムの安定化理論, 日本応用数理学会平成7年度年会講演予稿集, pp. 198–199, 1995.

[67] 白柳潔: 近似アルゴリズムにおける安定化手法, コミュニケーション科学研究所の研究活動 1994 年度, p. 36, 日本電信電話 (株) コミュニケーション科学研究所, 1995.

[68] Srinvasan, V. and Nackman, L. R.: Voronoi Diagram for Multiply-Connected Polygonal Domains I: Algorithm, *IBM J. Res. Dev.*, Vol. 31, No. 3, pp. 361–372, 1987.

[69] Srinvasan, V. and Nackman, L. R.: Voronoi Diagram for Multiply-Connected Polygonal Domains II: Implementation and Application, *IBM J. Res. Dev.*, Vol. 31, No. 3, pp. 373–381, 1987.

[70] Sugihara, K. and Iri, M.: Two Design Principles of Geometric Algorithms in Finite-Precision Arithmetic, *Applied Mathematical Letters*, Vol. 2, No. 2, pp. 203–206, 1986.

[71] 杉原厚吉, 伊理正夫: 計算誤差による暴走の心配のないソリッドモデラの提案, 情報処理学会論文誌, Vol. 28, No. 9, pp. 962–974, 1987.

[72] 杉原厚吉, 伊理正夫: 組合せ構造の優先によるボロノイ図構成算法の数値的安定化, 1988 年度応用数学合同研究集会報告集, pp. 183–192, 1988.

[73] 杉原厚吉: パターン認識の道具としてのボロノイ図構成算法の整備, 電子情報通信学会技術研究報告, PRU88-119, pp. 1–8, 1989.

[74] Sugihara, K. and Iri, M.: Construction of the Voronoi Diagram for Over 10^5 Generators in Single-Precision Arithmetic. *Proc. First Canadian Conf. on Computational Geometry*, p. 42, 1989.

[75] Sugihara, K., Ooishi, Y. and Imai, T.: Topology-Oriented Approach to Robustness and its Applications to Several Voronoi-Diagram Algorithms, *Proc. the Second Canadian Conf. in Computational Geometry*, pp. 36–39, 1990.

[76] Sugihara, K. and Iri, M.: Construction of the Voronoi Diagram for "One Million" Generators in Single-Precision Arithmetic. *Proc. IEEE*, Vol. 80, pp. 1471–1484, 1992.

[77] Sugihara, K.: A Simple Method for Avoiding Numerical Errors and Degeneracy in Voronoi Diagram Computation, *IEICE Trans. Fundamentals*, Vol. E75-A, pp. 468–477, 1992.

[78] 杉原厚吉: 計算幾何工学, アドバンストエレクトロニクスシリーズ, 培風館, 1994.

[79] Sugihara, K.: Robust Gift Wrapping for the Three-Dimensional Convex Hull, *J. Computer and System Sciences*, Vol. 39, pp. 391–407, 1994.

[80] Sugihara, K.: A Robust and Consistent Algorithm for Intersecting convex Polyhedra, *Computer Graphics Forum*, Vol. 13, No. 3, 1994.

[81] 高橋大介, 金田康正: 並列処理指向のスーパーコンピュータ利用について — 多倍長桁数の平方根計算を中心として —, センターニュース, Vol. 27, No. 6, pp. 49–56, 1995.

[82] 高橋秀俊, 石橋善弘: 電子計算機による exact な計算の新方法 (modulo p 演算の応用), 情報処理, Vol. 1, pp. 28–42, 1960.

[83] 高山信毅: 環と加群のソフトウェア, 数学, Vol. 45, No. 3, pp. 256–263, 1993.

[84] 徳山豪: はみだし幾何学, 岩波科学ライブラリー 18, 岩波書店, 1994.

[85] 和田秀男, 高速乗算法と素数判定法: 上智大学数学講究録 No. 15, 上智大学数学教室, 1983.

[86] Yap, C. K.: An $O(n \log n)$ Algorithm for the Voronoi Diagram of a Set of Simple Curve Segments, *Discrete Comput. Geom.*, Vol. 2, pp. 365–393, 1987.

[87] Yap, C. K.: A Geometric Consistency Theorem for a Symbolic Perturbation Scheme, *Proc. 4th ACM Ann. Symp. on Computational Geometry*, pp. 118–133, 1988.

[88] 吉田清範: 代数的な量の符号判定に必要な計算精度, 電子情報通信学会論文誌, Vol. J69-A, pp. 543–547, 1986.

付録

Manual of SEGVOR:

A FORTRAN Program for Constructing the Voronoi Diagram of Line Segments

Toshiyuki Imai
Department of Mathematical Engineering and Information Physics
Faculty of Engineering
University of Tokyo
Address: 7-3-1 Hongo Bunkyo-ku Tokyo Japan 113
e-mail: imai@simplex.t.u-tokyo.ac.jp

1994-07-10, Version1.03

This manual shows how to use SEGVOR, a FORTRAN program for constructing the Voronoi diagram of line segments. The manual consists of two sections. The first section gives some basic concepts of the Voronoi diagram of line segments to be used in section 2. If a user is familiar with these concepts, this part may be skipped. The second section shows the procedure of running the SEGVOR program.

1 The Voronoi diagram of line segments

1.1 The Voronoi Diagram Generated by a Generator Set L

We consider a set $L = \{L_1, \ldots, L_n\}$ of finite line segments placed on a plane (see the heavy continuous line segments in Figure 1), and assume that these line segments do not cross each other. Note that a degenerated line, i.e. a point is not allowed. Let $d(p, q)$ be the Euclidean distance between two points p and q, and

$$d(p, L_i) = \min_{q_i} \{d(p, q_i) \mid q_i \in L_i\}, \tag{1}$$

i.e., the distance from a point p to the nearest point on L_i. With this distance, we define

$$V(L_i) = \mathrm{int}\{p \mid d(p, L_i) < d(p, q_j), q_j \in L_j, j \neq i, j = 1, \ldots, n\}, \tag{2}$$

and let

$$\mathcal{V}(L) = \{V(L_1), \ldots, V(L_n)\}, \tag{3}$$

where $\mathrm{int}D$ means the interior of the set D. Then the *Voronoi diagram of line segments* is defined by the set $\mathcal{V}(L)$ (the light continuous lines in Figure 1). Since the Voronoi diagram of line segments is specified by the set L, we say that $\mathcal{V}(L)$ is the Voronoi diagram generated by L. We refer to L as a *generator set* and L_i as a *generator*.

The Voronoi diagram consists of subregions, $V(L_1), \ldots, V(L_n)$. We call these subregions *Voronoi regions*. Two adjacent Voronoi regions share the common line segment (or a half line). We call this line segment a *Voronoi edge*, and call a point at which three or more Voronoi edges meet a *Voronoi vertex*. Note that for any point p on a Voronoi edge between two Voronoi regions $V(L_i)$ and $V(L_j)$, an equation

$$d(p, L_i) = d(p, L_j)$$

holds.

1.2 The Voronoi Diagram of Line Segments Generated by the Decomposed Generator Set L^{d}

The construction of the Voronoi diagram of line segments $\mathcal{V}(L)$ generated by L is two-fold. First, we construct the Voronoi diagram of line segments $\mathcal{V}(L^{\mathrm{d}})$ generated by a modified generator set, called the 'decomposed generator set', L^{d}. The resulting line Voronoi diagram $\mathcal{V}(L^{\mathrm{d}})$ has the property that Voronoi edges of $\mathcal{V}(L)$ are contained in those of $\mathcal{V}(L^{\mathrm{d}})$. Thus, second, we delete unnecessary Voronoi edges in $\mathcal{V}(L^{\mathrm{d}})$. As a result, we obtain $\mathcal{V}(L)$.

To construct $\mathcal{V}(L)$, we consider $2n+4$ points, $p_1, \ldots, p_{2n+4}$. The points p_1, p_2, p_3 are the vertices of a sufficiently large triangle containing all generators in L; If all generators are in a unit square $[0,1]^2$, the triangle should contain a square $[-1,2]^2$. The points p_{2i+2}, p_{2i+3} are the end points of a generator L_i, $i = 1, \ldots, n$. The last point p_{2n+4} is an imaginary point far from the large triangle $\triangle p_1 p_2 p_3$. Let L_i° be the line segment obtained by excluding the end points p_{2i+2}, p_{2i+3} from L_i. Then the line segment L_i is written as $L_i = \{p_{2i+2}, p_{2i+3}\} \cup L_i^{\circ}$. Let

$$L^{\mathrm{d}} = \{p_1, p_2, \ldots, p_{2n+2}, p_{2n+3}, L_1^{\circ}, \ldots, L_n^{\circ}\}. \tag{4}$$

We call the set L^{d} the *decomposed generator set* of L.

We now wish to generate the Voronoi diagram by this decomposed generator set, but to do so, we have to make one more modification on equation (1). Since L_i° does not include its end points, we re-define equation (1) as

$$d(p, L_j^{\circ}) = \inf_{q_j} \{d(p, q_j) \,|\, q_j \in L_j^{\circ}\}. \tag{5}$$

With this distance, we define

$$V(L_i^o) = \mathrm{int}\{p \mid d(p, L_i^o) < d(p, q), q \in g, g \neq L_i, g \in L^d\} \tag{6}$$

for $i = 1, \ldots n$, and

$$V(p_k) = \mathrm{int}\{p \mid d(p, p_k) < d(p, q), q \in g, g \neq p_k, g \in L^d\} \tag{7}$$

for $k = 1, \ldots, 2n + 3$. Let

$$\mathcal{V}(L^d) = \{V(p_4), \ldots, V(p_{2n+3}), V(L_1^o), \ldots, V(L_n^o)\}. \tag{8}$$

We call the set $\mathcal{V}(L^d)$ the Voronoi diagram of line segments generated by the decomposed generator set L^d of L. Note that for any point p on a Voronoi edge between two Voronoi regions $V(g)$ and $V(g')$ $(g, g' \in L^d)$, an equation

$$d(p, g) = d(p, g')$$

also holds.

The line Voronoi diagram generated by the decomposed generator set L^d of L in Figure 1 is shown by the light continuous lines and the broken lines in Figure 2.

Since generators in the original set L are included in generators in the decomposed generator set L^d, the Voronoi edges of $\mathcal{V}(L)$ are included in those of $\mathcal{V}(L^d)$. Thus the final procedure is to delete the unnecessary edges (indicated by the broken lines in Figure 2) from the Voronoi edges in $\mathcal{V}(L^d)$ (A more detailed procedure is given by Okabe, Boots and Sugihara (1992, Section 3.5)). Without loss of generality, we can consider that all terminal points p_i, $i = 4, \ldots, 2n + 3$ are in a unit square $[0, 1]^2$. Under such consideration, these procedures gives us the same diagram as the Voronoi diagram of line segments in the unit square.

2 Procedure of Running SEGVOR

2.1 Input Data File

2.1.1 Name of the Input Data File

The length of the name of an input data file must be less than or equal to 9 characters. A name having more than 9 characters causes a failure. There is no other restriction, but the recommended name is ??f??.dat where ? is determined by a user.

2.1.2 Format of the Input Data File

The file consists of the number, n, of line segments and the coordinates (x_{2i+2}, y_{2i+2}), (x_{2i+3}, y_{2i+3}) of the end points p_{2i+2}, p_{2i+3}, $i = 1, \ldots, n$. All points p_j must be in a unit square $[0, 1]^2$

The input data file should be in the following order.

n : i10 format
x_4 : f10.6 format
y_4 : f10.6 format
$\vdots$
x_{2n+3} : f10.6 format
y_{2n+3} : f10.6 format

An example is shown in Appendix.

2.2 Running SEGVOR

The session with your computer goes as follows (a user is supposed to type bold face letters;[R] means a return key).

```
% segvor [R]
Type of the distribution:
1,2,3 or 4
4 [R]
??f??.dat?
aafaa.dat [R] (type your input data file name)
interrupt num.  of seg.  <= 2
2 [R] (type the number appeared in the above line)
0-non stop
1-stop for each N segments
0 [R]
003-z i,itrs,nofs 1 2 2
003-z i,itrs,nofs 2 2 2
f??.gdt
faa.gdt [R] (type your output data file name)
--501*writing-1--
--005*finished--
1-change generator
2-end
2 [R]
%
```

Then you will get the output file faa.gdt, which we call a *gdt file*.

3 Output File

3.1 Name of variables (see Figure 3)

The names of variables used in the output file are as follows.

ie: the ie−th Voronoi edge
iv: the iv−th Voronoi vertex
ksv(ie): start vertex of Voronoi edge ie
kev(ie): end vertex of Voronoi edge ie
klf(ie): Voronoi region that is left to Voronoi edge ie
krf(ie): Voronoi region that is right to Voronoi edge ie
iemax: maximum index of Voronoi edge
ivmax: maximum index of Voronoi vertices
newpe: index of the top edge in unused edge stack (used only for computational purposes)
nofg: number of terminal points of input line segments (i.e. $2n$)
icol(ie): flag of Voronoi edge ie (used only for computational purpose)
ietype(ie): type of Voronoi edge ie (see Figure 4; -1 = not used)
vorox(iv): x coordinate of Voronoi vertex iv
voroy(iv): y coordinate of Voronoi vertex iv
genex(ig): x coordinate of generator ig
geney(ig): y coordinate of generator ig

3.2 Format of the gdt file

The output data are shown in the following order.

iemax: i10 format
ivmax: i10 format
newpe: i10 format
nofg: i10 format
icol(ie): i2 format
kev(ie): i10 format
ksv(ie): i10 format
ietype(ie): i2 format
krf(ie): i10 format
klf(ie): i10 format
} repeat for $ie = 1$ to iemax.

vorox(iv): e14.6e4 format
voroy(iv): e14.6e4 format
} repeat for $iv = 1$ to ivmax.

genex(ig): e14.6e4 format
geney(ig): e14.6e4 format
} repeat for $ig = 1$ to nofg+4.

An example is shown in Appendix.

3.3 Transformation to a Postscript format file

The program SVPS transforms a gdt format file obtained from the above procedure transforms into a PS (Postscript) format file. A user can see the result through a PS printer or a PS viewer.

Two parameters should be determined before running the program SVPS. One parameter, widcm, determines the size of the output. If you fix widcm = 15.0, the output figure is drawn in a 15cm square. The default value is 15cm. Second, n in the subroutine named prbl determines the number of line segments that approximate a parabolic curve. The default value is 5.

Having fixed these parameter values, a user is supposed to run SVPS.f with the output file named SVPS. Then the session goes as follows.

%svps[R]
`f??.gdt`
faa.gdt [R](type your gdt file name)
`-- 601*ready`
`-- 601*finish`
`xmin`
0[R] (the minimum x-coordinate value in terms of the scale of your input data)
`ymin`
0[R] (the minimum y-coordinate value in terms of the scale of your input data)
`width`
1[R] (the width of your frame in terms of the scale of your input data)
`f??.ps`
faa.ps[R] (type your postscript file name)
`writing`
`Type of generators`
`original -0` (the Voronoi diagram generated by the decomposed generator set L^{d})
`separated -1` (the Voronoi diagram generated by L)
0[R]
`hardcopy -1 x`
`again -2`
`another -3`
`nothing -0`
0[R]
%

Send the resulting PS file to a PS printer. Then you will get a hard copy of your Voronoi diagram of line segments.

Acknowledgments

The author expresses his thanks to Kokichi Sugihara for many suggestions and the permission to use his numerically stable program **VORONOI2** for constructing a Voronoi diagram of points as a subroutine of SEGVOR and to Atsuyuki Okabe for his many valuable comments on earlier drafts.

Notice

The author distributes SEGVOR system only to those users who agree the following.

1. The user uses SEGVOR system for nonprofit purposes only.
2. The author does not bear responsibility for any trouble that user meets in the use of SEGVOR system.
3. When the user publishes any results obtained by SEGVOR system, he explicitly states in the paper that he used SEGVOR system. Also, he send a reprint of the paper to the authors.
4. The user reports to the authors any trouble he meets in the use of SEGVOR system. (The author will remove bugs, if any, in their earliest convenience.)

The address of the author is:

Toshiyuki Imai
Department of Mathematical Engineering and Information Physics
Faculty of Engineering, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113, Japan

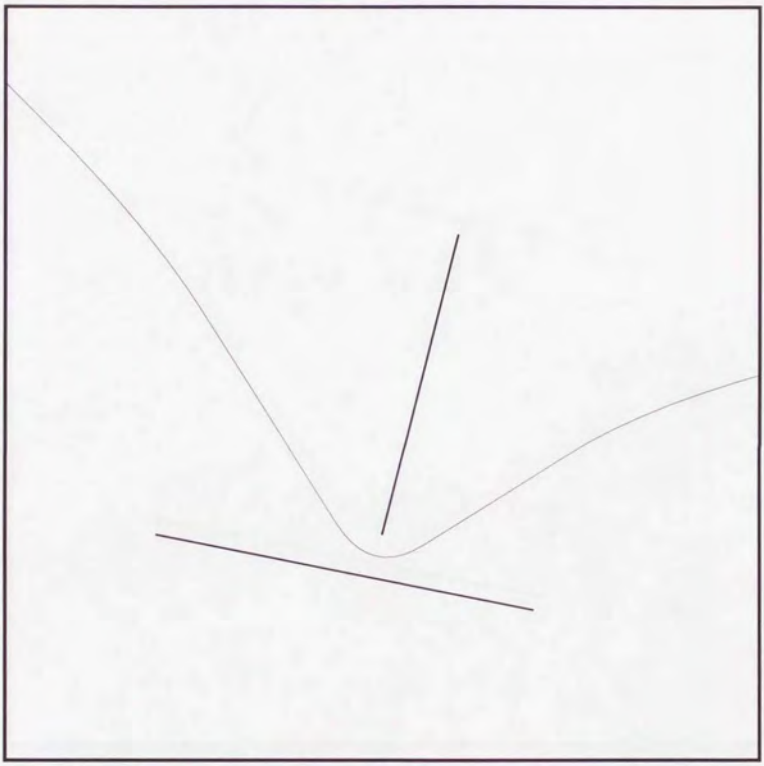


Figure 1: the Voronoi diagram of two line segments

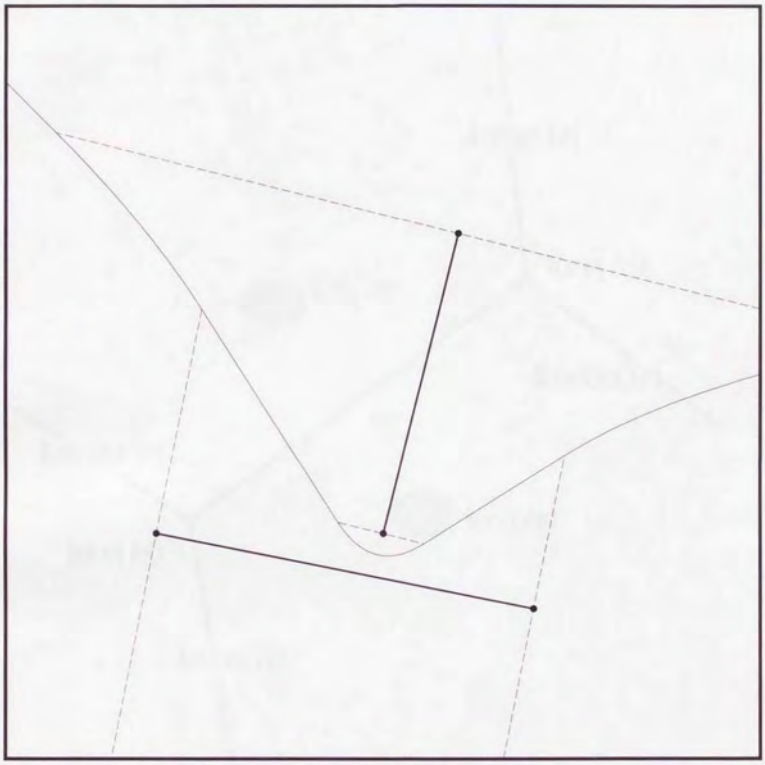


Figure 2: the decomposed Voronoi diagram of two line segments and their terminal points

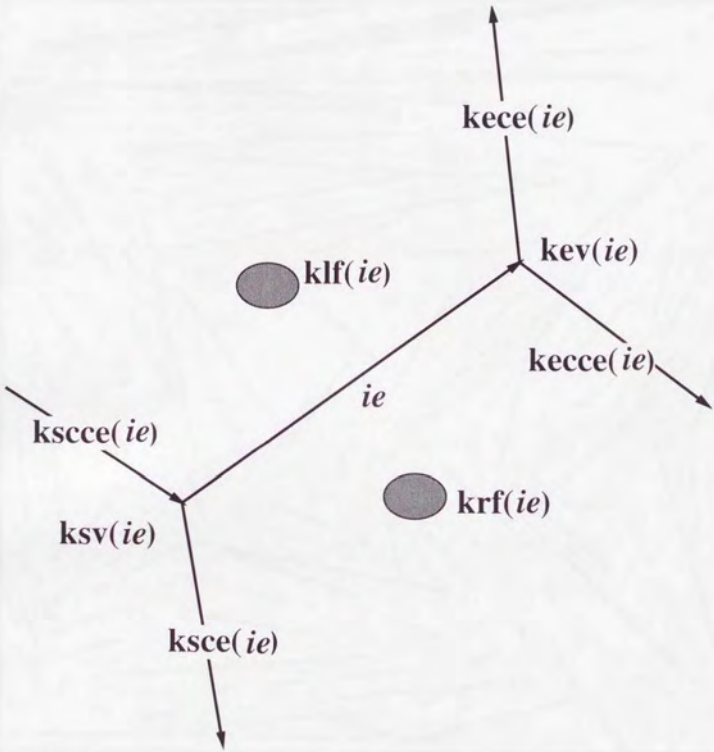


Figure 3: some variables used in a gdt file

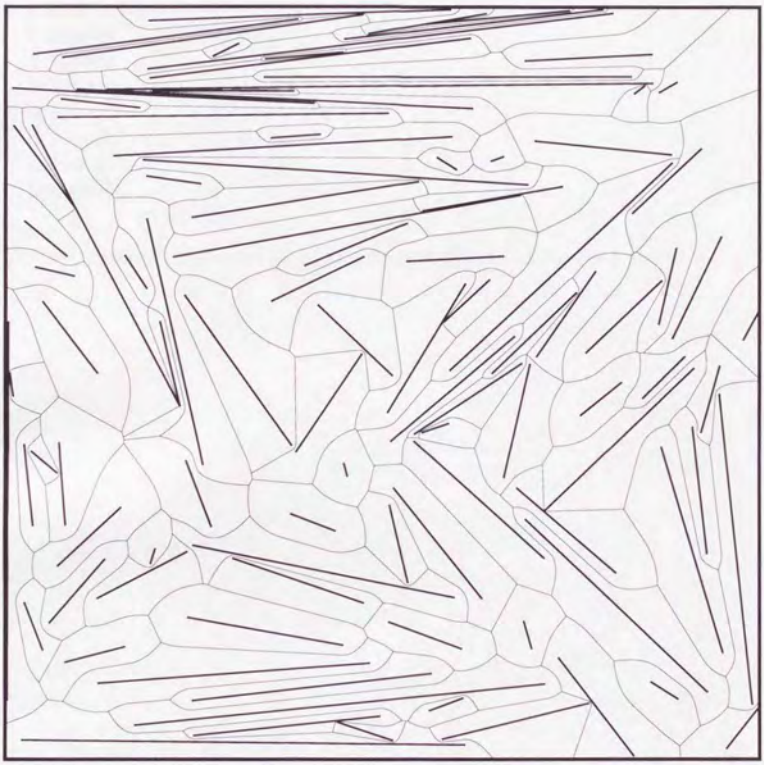


Figure 4: the Voronoi diagram of 100 line segments obtained by SEGVOR and SVPS

Appendix

An example of an input file

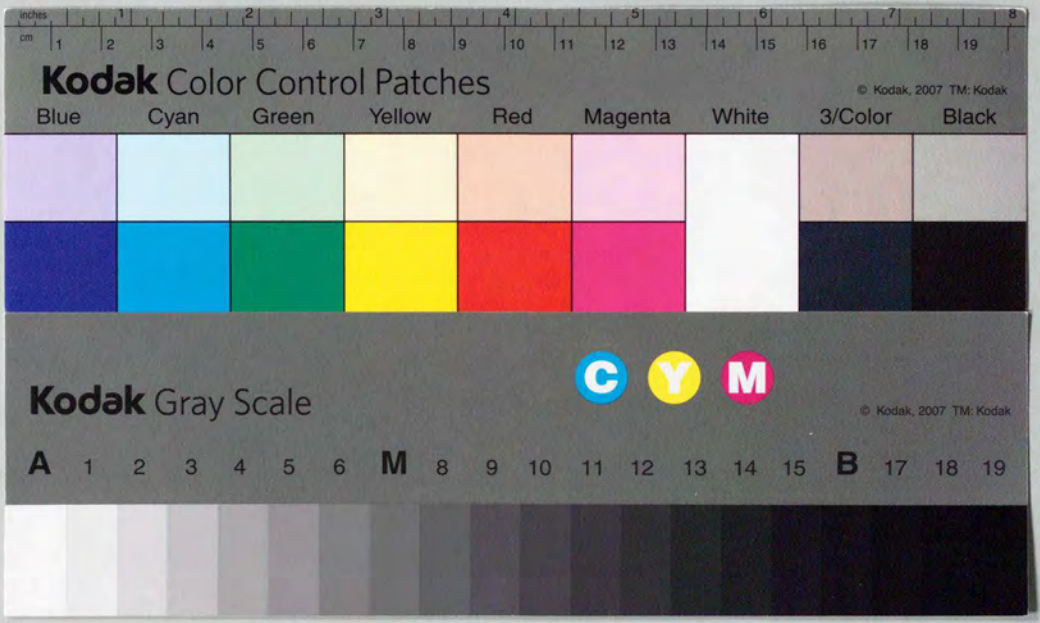
```
        2
0.200000
0.300000
0.700000
0.200000
0.500000
0.300000
0.600000
0.700000
```

Figure 5: the input file of the Voronoi diagram of figures 1 and 2

An example of an output file (gdt file)

28	0		5	21		0.375948E+0000
21		4	9	3		0.543513E+0000
17		6	0		7	-.264491E+0001
4	1		20		9	0.354492E+0001
0		2	8	0		0.554487E+0000
1		4	1		13	0.286378E+0000
6	0		4		9	0.500000E+0000
1		20	7	3		-.614230E+0001
1		12	0		6	-.263347E+0000
2	2		-1		9	-.461674E+0001
0		7	23	0		0.117778E+0001
2		10	1		9	0.555555E+0000
10	0		6		13	0.440071E+0000
1		11	7	2		0.314982E+0000
2		4	0		6	0.447626E+0001
3	3		14		10	0.121525E+0001
0		2	12	0		0.603311E+0001
3		9	1		13	0.369454E+0001
15	0		5		19	0.266667E+0000
1		5	7	4		0.633333E+0000
3		9	0		9	0.754167E+0000
1	4		15		10	0.470833E+0000
0		9	14	0		0.699563E+0000
2		10	1		19	0.462609E+0000
1	0		3		20	0.259929E+0000
1		10	7	3		0.599643E+0000
8		11	0		4	0.666666E-0001
2	1		8		10	0.833333E+0000
0		2	15	0		0.000000E+0000
3		5	1		-1	0.000000E+0000
2	0		1		29	0.500000E+0000
1		14	7	1		0.650000E+0001
8		10	0		0	-.469600E+0001
3	1		-1		0	-.250000E+0001
0		3	28	-.126796E+0002		0.569600E+0001
1		5	3	0.732037E+0001		-.250000E+0001
3	0		7	0.000000E+0000		0.200000E+0000
1		12	9	-.200000E+0002		0.300000E+0000
8		5	0	0.126796E+0002		0.700000E+0000
1	3		4	0.732037E+0001		0.200000E+0000
0		5	19	-.641734E+0000		0.500000E+0000
6		10	2	-.390867E+0001		0.300000E+0000
8	0		4	0.740865E+0000		0.600000E+0000
1		5	9	0.404324E+0000		0.700000E+0000
1		11	0	-.496871E+0001		0.000000E+0000
4	2		-1	0.365736E+0001		0.000000E+0000

Figure 6: the output file (gdt file) of the Voronoi diagram of figure 1 and 2 by SEGVOR (5 columns)


inches
cm
Kodak Color Control Patches
© Kodak, 2007 TM: Kodak
Blue
Cyan
Green
Yellow
Red
Magenta
White
3/Color
Black
Kodak Gray Scale
C
Y
M
© Kodak, 2007 TM: Kodak
A 1 2 3 4 5 6 M 8 9 10 11 12 13 14 15 B 17 18 19