# 安定化理論に基づくISCZ法の3次元凸包構成への適用

# Using the ISCZ method based on stabilization theory to construct three-dimensional convex hulls

東邦大学大学院 理学研究科　　奥田 和樹 *1
KAZUKI OKUDA
GRADUATE SCHOOL OF SCIENCE, TOHO UNIVERSITY.

東邦大学 理学部　　白柳 潔 *2
KIYOSHI SHIRAYANAGI
FACULTY OF SCIENCE, TOHO UNIVERSITY

**Abstract**

The ISCZ method (Interval-Symbol method with Correct Zero rewriting) was proposed based on Shirayanagi-Sweedler stabilization theory, to reduce the amount of exact computations as much as possible to obtain the exact results. This method is the floating-point interval method using zero rewriting and symbols. Zero rewriting rewrites an interval coefficient into the zero interval if the interval contains zero. Symbols are used to keep track of the execution path of the original algorithm with exact computations, so that the associated real coefficients can be found by evaluating the symbols. The key point is that at each stage of zero rewriting, one checks to see if the zero rewriting is really correct by exploiting the associated symbol. In this paper, we show the efficiency of the ISCZ method by applying it to three-dimensional convex hull construction.

## 1　はじめに

安定化理論 [1] は、近似計算で実行すると不安定になるアルゴリズムに対し、それを変形して近似計算で実行しても誤差の影響を抑制し、安定な出力が得られるようにするための理論である。文献 [4][5] ではその安定化手法、また発展形である ISCZ 法 ([3]) を、凸包アルゴリズムの一つである、Graham のアルゴリズムに適用し、2 次元の凸包構成での計算機実験を行った。本論では、ISCZ 法を 3 次元凸包アルゴリズムの一つであるギフト包装法に適用し、3 次元の凸包構成での計算機実験を行い、ISCZ 法の有効性を示す。

まず、2 節では、安定化理論と ISCZ 法について復習する。3 節では、3 次元凸包アルゴリズムの一つであるギフト包装法を紹介する。3.1 節では計算履歴を保存するシンボルリストに対しての新しいアイデアを述べる。3.2 節では、ISCZ 法を 3 次元凸包構成のためのギフト包装法アルゴリズムに適用した計算機実験について述べる。

*1　E-mail: 6520002o@st.toho-u.jp
*2　E-mail: kiyoshi.shirayanagi@is.sci.toho-u.ac.jp

# 2 復習

## 2.1 安定化理論

次のアルゴリズムを対象に安定化理論の復習を簡単に行う。詳細は [1] を参照されたい。

- データは、すべての多項式環 $R[x_1, \cdots, x_m]$ の元からなる。$R$ は実数体の部分体である。
- データ間の演算は、$R[x_1, \cdots, x_m]$ 内の加減乗である。
- データ上の述語は、不連続点を持つとすればそれは 0 のみである。

述語の不連続点が 0 という意味は、If "C = 0" then ... else... のように、値が 0 か否かによって分岐が別れることである。従って、C = 0 の代わりに　C > 0 や C ≤ 0 などでもよい。上記クラスのアルゴリズムを、不連続点 0 の代数的アルゴリズムと呼ぶ。ほとんどの数式処理のアルゴリズムはこのクラスに入るか、このクラスのアルゴリズムに変換可能である。

さて、安定化の 3 つのポイントは、

- アルゴリズムの構造は変えない。
- データ領域において、ふつうの係数を区間係数に変える。
- 述語の評価の直前で、区間係数のゼロ書換えを行う。

である。すなわち、安定化されたアルゴリズムは次のようになる。

**区間領域：** データ領域は区間係数多項式の集合。区間係数は $[A, B]$ なる形で、$A, B \in \mathbb{R}, [A, B]$ は集合 $\{r \in \mathbb{R} | A \leq r \leq B\}$ を意味する。

**区間演算：** 二項演算 $\diamond \in \{+, -, \times, \div\}$ に対し、

$$[A, B] \diamond [C, D] = [min(A \diamond C, A \diamond D, B \diamond C, B \diamond D), max(A \diamond C, A \diamond D, B \diamond C, B \diamond D)]$$

**ゼロ書換え：** 不連続点 0 をもつ述語を評価する直前で、各区間係数 $[A, B]$ に対し、

$A \leq 0 \leq B$ ならば $[A, B]$ を $[0, 0]$ に書き換えよ。そうでないならばそのままとせよ。

今、入力 $f \in R[x_1, \cdots, x_m]$ を

$$f = \sum_{i_1, \cdots, i_m} r_{i_1 \cdots im} x_1^{i_1} \cdots x_m^{i_m}$$

と表したとき、$f$ に対する近似値 $\{Int(f)_j\}_j$ を

$$Int(f)_j = \sum_{i_1, \cdots, i_m} [(a_{i_1 \cdots i_m})_j, (b_{i_1 \cdots i_m})_j] x_1^{i_1} \cdots x_m^{i_m}$$

で定義する。ここに、すべての $i_1, \cdots, i_m$ について、

$$(a_{i_1 \cdots i_m})_j \leq r_{i_1 \cdots i_m} \leq (b_{i_1 \cdots i_m})_j \ for \ \forall j$$

$$(b_{i_1 \cdots i_m})_j - (a_{i_1 \cdots i_m})_j \to 0 \ as \ j \to \infty$$

このとき、単に

$$Inf(f)_j \to f$$

と書く。

さて、$A$ を安定化したアルゴリズムを $Stab(A)$ と書くと、次が安定化理論の基本定理である。

**定理 1 (安定化理論の基本定理)**

$A$ は不連続点 $0$ の代数的アルゴリズムで、入力 $f \in R[x_1, \cdots, x_m]$ に対し正常終了するとせよ。このとき、$f$ に対する任意の近似列 $\{Int(f)_j\}_j$ に対し、ある $n$ が存在して、$j \geq n$ ならば、$Stab(A)$ は $Int(f)_j$ に対し正常終了し、

$$Stab(A)(Int(f)_j) \to A(f).$$

簡明を期すため、入力は一つだけの多項式にしているが、入力はもちろん、多項式の有限集合でもよい。

## 2.2 ISCZ 法

### 2.2.1 シンボル付き区間

区間と形式的なシンボルを組み合わせた係数（シンボル付き区間）を導入する。区間は、従来と同様の意味の区間である。シンボルは、アルゴリズム実行中に現れる係数の log (記録) を取るのに使われる。例えば、入力係数が $1/3$ と $1/6$ とする。これらの精度 3 の区間はそれぞれ $[0.333, 0.334]$ と $[0.166, 0.167]$ である。さて、$1/3$ に対するシンボルを $s$、$1/6$ に対するシンボルを $t$ として、区間と組み合わせると、それぞれ、$[[0.333, 0.334], s]$ と $[[0.166, 0.167], t]$ となる。次に、これらの間の演算、例えば、加算を、

$$[[0.333, 0.334], s] + [[0.166, 0.167], t] = [[0.333, 0.334] + [0.166, 0.167], s \dot{+} t]$$

と定義する。$[0.333, 0.334] + [0.166, 0.167]$ に対しては通常の区間演算を使う。シンボル部分 $s \dot{+} t$ は再び形式的なシンボルで、加算を実施したことを記録できれば何でもよい。

アルゴリズムの実行中あるいは実行後に、必要に応じて、シンボルを正確係数（実数値）に復元することができる。先の簡単な例で言えば、もしシンボルが $s \dot{+} t$ であったとすれば、$s$ に $1/3$ を、$t$ に $1/6$ を代入し、$\dot{+}$には加算の意味を与えて、$1/3 + 1/6 = 1/2$ と復元する。

シンボル付き区間のことを interval-symbol、あるいは単に IS と呼ぶ。

### 2.2.2 手続き

A を不連続点 0 の代数的アルゴリズムとする。ISCZ 法の手続きは次の通りである。

**R-to-IS：** 各入力係数 $r$ を $[[A, B], Symbol_r]$ に変換する。ここに、$[A, B]$ は $r$ の予め定められた精度の区間、$Symbol_r$ は $r$ を表すシンボル（以下、入力シンボルと呼ぶ）である。

**IS 演算：** IS 演算を次のように実行する：

$$[[A, B], s] + [[C, D], t] = [[A, B] + [C, D], s \dot{+} t]$$

$$[[A, B], s] - [[C, D], t] = [[A, B] - [C, D], s \dot{-} t]$$

$$[[A, B], s] \times [[C, D], t] = [[A, B] \times [C, D], s \dot{\times} t]$$

すなわち、区間部分については区間演算を用い、シンボル部分については加算、減算、乗算の形式的なシンボル $\dot{+}, \dot{-}, \dot{\times}$ を使って、どういう演算が行われたかを記録する。

**正しいゼロ書換え：** 任意の IS $[[A, B], s]$ に対し、$A \leq 0 \leq B$ ならば、$s$ をそれに対応する実数 $r(s)$ に復元する。もし、$r(s) = 0$ ならば、次のステップに進む。そうでなければ、精度を上げて **R-to-IS** に戻る。

**IS-to-R：** 出力のシンボル部分の中の各入力シンボルにそれぞれ対応する入力係数を代入し、演算シンボルに演算の意味を与えて実数値に復元する。

この手法を ISCZ 法（IS method with correct zero rewriting）と呼ぶ。

さて、ある精度 $j$ で $Int(f)_j$ を $Stab(A)$ に入力したときの実行過程は、もしアルゴリズム中のすべてのゼロ書換えが正しいならば、真の出力 $f$ を $A$ に入力したときの実行過程と完全に一致する。ISCZ 法では、各ゼロ書換えにおいて、それが正しいかどうかを確認する。さらに、定理 1 により、すべてのゼロ書換えが正しくなる精度が存在する。従って、ISCZ 法は有限ステップで終了し、その出力の各 IS 係数のシンボルは正しい正確係数を与える。

これを定理にまとめる。

**定理 2 (ISCZ 法の停止性と正当性)**

$A$ が入力 $I$ で正常終了するとせよ。このとき、$A$ に対する *ISCZ* 法は、常に有限ステップで終了し、正しい結果、すなわち、$A(I)$ の出力と同じ結果を与える。

ISCZ 法の利点は、**出力の正当性を確認する必要がない**ことである。任意の IS $[[A, B], s]$ に対し、$A \leq 0 \leq B$ でない限り、正確計算をスキップすることができ、浮動小数計算だけで済む。換言すれば、**ゼロでない係数についての正確計算を省略することができる。**従って、本手法は、$A \leq 0 \leq B$ でない場合が $A \leq 0 \leq B$ である場合よりも多ければ多いほど有効であるということができる。

## 2.3 シンボルリスト

ISCZ 法では、複雑な計算を行った際に IS のシンボル部分が膨張してしまうことがある。それを防ぐため、IS 演算後のシンボルに自然数を用いる。さらに、各自然数がどのように構成されたかを示すリストを用意する。これをシンボルリストと呼ぶ ([6])。IS-to-R の際に、このシンボルリストを用いて実数値を復元する。

1. R-to-IS を行う。
2. (初期化) $K = 0$ とし、シンボルリスト Symbol_List を空リスト（要素なしのリスト）とする。
3. 2 つの IS $[[a_1, a_2], s]$ , $[[b_1, b_2], t]$ に対する演算 $\diamond \in \{\dot{+}, \dot{-}, \dot{\times}, \dot{\div}\}$ を次のように定義する。
   まず、区間部分を計算し、その結果の区間係数を $[c_1, c_2] = [a_1, a_2] \diamond [b_1, b_2]$ とする。
   $K \leftarrow K + 1$ とし、新しい IS $[[c_1, c_2], K]$ を作り、Symbol_List の末尾に $(\diamond, s, t)$ を付加する。
4. (復元) $[[d_1, d_2], K]$ が出力のある係数とする。$K$ を Symbol_List の $K$ 番目の要素に置き換え、以下再帰的に入力シンボルに行きつくまでこの処理を行い、各シンボルに対応する実数の代入と、対応する演算を施し、実数値 $r(K)$ を復元する。

例えば、

$$\begin{aligned}\frac{1}{3} \times \frac{1}{6} + \frac{4}{9} - \frac{1}{2} &= [[0.332, 0.334], s] \times [[0.165, 0.167], t] + [[0.443, 0.445], u] - [[0.449, 0.501], v] \\ &= [[0.054780, 0.055778], 1] + [[0.443, 0.445], u] - [[0.449, 0.501], v] \\ &= [[0.497780, 0.500778], 2] - [[0.449, 0.501], v] \\ &= [[-0.003220, 0.051778], 3]\end{aligned}$$

の計算過程をシンボルリストに保存する場合、

| | $K$ | IS | Symbol_List |
|---|---|---|---|
| 初期値 | 0 | | [] |
| 1 回目 | 1 | [[0.054780,0.055778],1] | $[(\dot{\times}, s, t)]$ |
| 2 回目 | 2 | [[0.497780,0.500778],2] | $[(\dot{\times}, s, t), (\dot{+}, 1, u)]$ |
| 3 回目 | 3 | [[-0.003220,0.051778],3] | $[(\dot{\times}, s, t), (\dot{+}, 1, u), (\dot{-}, 2, v)]$ |

となる。

ここで、表の 3 回目の IS $[[-0.003220, 0.051778], 3]$ に注目すると、$-0.003220 \leq 0 \leq 0.051778$ となっている。従って、**正しいゼロ書換え**より、実数値 $r(3)$ に復元する。具体的には、

1. Symbol_List の 3 番目の要素を参照する。今回は、$(-, 2, v)$ である。
2. Symbol_List を使用して計算履歴を辿る。今回は、

$$
\begin{aligned}
&(\dot{-}, 2, v) && \text{//Symbol\_List の 2 番目の要素を参照する。}\\
&= (\dot{-}, (\dot{+}, 1, u), v) && \text{//Symbol\_List の 1 番目の要素を参照する。}\\
&= (\dot{-}, (\dot{+}, (\dot{\times}, s, t), u), v)
\end{aligned}
$$

3. 各シンボルに対応する実数の代入と、対応する演算を施し、実数値を復元する。今回は、$s = \frac{1}{3}$, $t = \frac{1}{6}, u = \frac{4}{9}, v = \frac{1}{2}$ に対応するので、

$$(\dot{-}, (\dot{+}, (\dot{\times}, s, t), u), v) = \{\{(\frac{1}{3} \times \frac{1}{6}) + \frac{4}{9}\} - \frac{1}{2}\} = 0$$

以上より、$r(3) = 0$ であるので、IS の区間係数を $[0, 0]$ 書き換える。

$$[[-0.003220, 0.051778], 3] \rightarrow [[0, 0], 3]$$

## 3　凸包アルゴリズムへの適用

前述の ISCZ 法を凸包アルゴリズムの一つであるギフト包装法 (またの名を Jarvis の行進法) に適用し、3 次元の凸包構成の計算機実験を行う。次が 3 次元凸包構成でのギフト包装法アルゴリズムである。

**アルゴリズム 1 (ギフト包装法)**

1. $n$ 個の点のうち、同一線上にはない 3 点を通る平面で、「他のすべての点がこの平面の上にある、またはこの平面の一方側に位置する」ものを見つける。この 3 点 $(p_1, p_2, p_3)$ が作る平面は凸包を構成する面になる。
2. 平面の辺 $\overline{p_2p_3}$ を軸として平面を回転させたとき初めてぶつかった点 $p_4$ を見つける。面 $p_2p_3p_4$ は凸包の面になる。
3. 辺 $\overline{p_3p_4}$ を軸として手順 2. と同じことを繰り返せば凸包が求められる。

ここで、点 $p_i$ の座標を $(x_i, y_i, z_i)$ とすると、

$$G(p_i,p_j,p_k,p_l)=\begin{vmatrix}1 & x_i & y_i & z_i\\ 1 & x_j & y_j & z_j\\ 1 & x_k & y_k & z_k\\ 1 & x_l & y_l & z_l\end{vmatrix} > 0 \Leftrightarrow \begin{array}{c}\text{点 } p_l \text{ を原点とすると}\\ 3 \text{ 点 } p_i,p_j,p_k \text{ は外側から}\\ \text{見ると右手系の向き}\end{array}$$

が成立するので、ギフト包装法は不連続点 0 の代数的アルゴリズムである。従って、安定化理論の適用条件を満たす。

### 3.1 シンボルリストの最適化

ISCZ 法をギフト包装法アルゴリズムに適用し、計算機実験を行った。次の表 1 は、入力を $0 \leq X, Y, Z \leq 1000$ を満たすランダムに生成された整数の組 $(X, Y, Z)$ に対して実験を行った際の計算時間を、安定化手法と ISCZ 法で比べたものである。

| 入力点数 | 100 | 500 | 1000 | 2000 |
|---|---|---|---|---|
| ISCZ 法 | 11.95 | 948.2 | 15619 | 82932 |
| 安定化手法 | 0.719 | 21.39 | 85.17 | 291.4 |

表 1: 3 次元凸包-整数入力（秒）

ISCZ 法は安定化手法と比べ、大幅に計算時間を要した。それはシンボルリストの膨張が原因であったので、プログラムの改善を行った。

従来の方法では、アルゴリズムの開始時にシンボルリストの初期化を行う。簡易的に表すと

```
do{
  Symbol_ List:=[]; K := 0;   // シンボルリストの初期化
  A : 基準面
  for i from 1 to 点の数 −3 do {
    //全ての点を探索し、次の凸包に含まれる面を求める
  }
  A := 求めた凸包の面
}while 凸包が完成
```

しかし、このプログラムではシンボルリストに余計な計算履歴が for 文の繰り返し処理によって、点の数に比例して大きく蓄積されていた。具体的には入力点数 100 点ではシンボルリストの長さは最大で 1673、500 点で 8473、1000 点で 16973、2000 点で 33973 となっていた。そこで、余計な計算履歴を残さないために、アルゴリズムの開始時にシンボルリストを初期化するのではなく、各点を探索する時点でシンボルリストを初期化することで、シンボルリストに余計な計算履歴が残らないようにプログラムを以下のように改善した。

```
do{
   A : 基準面
   for i from 1 to 点の数 −3 do {
     Symbol_ List:=[]; K := 0;   // シンボルリストの初期化
     //全ての点を探索し、次の凸包に含まれる面を求める
   }
   A := 求めた凸包の面
}while 凸包が完成
```

上記のようにプログラムを改善したことで繰り返し処理の計算履歴を全て蓄積することなく、各処理の計算履歴だけが残るようになる。これにより、シンボルリストの長さは入力点数に関係なく、最大で 41 を保った。また、プログラム改善前と改善後で計算時間を比較した。

| 入力点数 | 100 | 500 | 1000 | 2000 |
|---|---|---|---|---|
| 改善前 | 11.95 | 948.2 | 15619 | 82932 |
| 改善後 | 1.881 | 46.82 | 183.5 | 740.4 |

表 2: 3 次元凸包-整数入力（秒）

結果、シンボルリストの長さを最低限に抑えるための新しいアイデアを導入したことで、大幅な計算時間短縮に成功した。

### 3.2　計算機実験

3 次元凸包構成アルゴリズムのギフト包装法で ISCZ 法の効果を確認するために、何も工夫しない素朴な近似値計算と ISCZ 法の比較実験を行い、凸包が正しく出力されるかを確かめる。また、近似値計算、ISCZ 法、(近似を一切使わない厳密な) 正確計算による計算時間を比較する。

計算機は、OS: Windows 10 Enterprise LTSC、CPU: Intel(R) Core(TM) i5-8250U、CPU @ 1.60GHz 1.80GHz、実装メモリ (RAM): 8.00GB であり、数式処理ソフトは Maple 2020 を使用した。

例 1　入力：$0 \leq X, Y, Z \leq 1000$ を満たすランダムに生成された整数の組 $(X, Y, Z)$2000 点

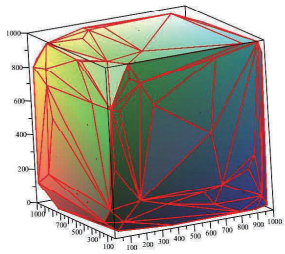

図 1: ISCZ 法

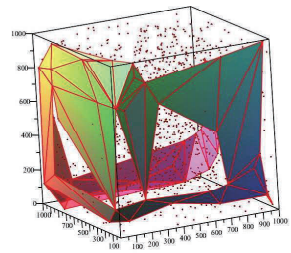

図 2: 近似値計算

ISCZ 法では精度桁 8 以上で凸包を確認できた (図 1)。対して、近似値計算では精度桁を 100 まで上げたが凸包を確認することができなかった (図 2)。

例 2　入力：次を満たすランダムに生成された整数の組 $(X, Y, Z)$2000 点

$$-1000 \leq X, Y \leq 1000\ ,\ 0 \leq Z \leq 2000\ ,\ 2000^2(X^2+Y^2) \leq 1000^2(Z-2000)^2$$

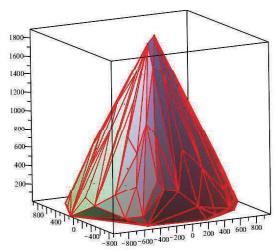

図 3: ISCZ 法

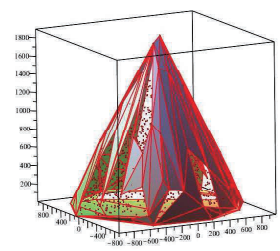

図 4: 近似値計算

ISCZ 法では精度桁 9 以上で凸包を確認できた (図 3)。対して、近似値計算では精度桁を 100 まで上げたが凸包を確認することができなかった (図 4)。

例 3　入力：次を満たすランダムに生成された整数の組 $(X, Y, Z)$2000 点に対し、$(\sqrt{X}, \sqrt{Y}, \sqrt{Z})$ を作ったもの

$$0 \leq X, Y, Z \leq 200, (X-100)^2+(Y-100)^2+(Z-100)^2 \leq 100$$

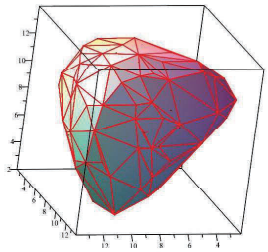

図 5: ISCZ 法

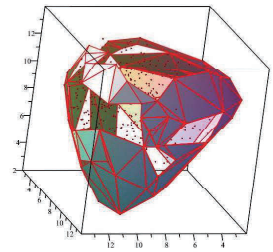

図 6: 近似値計算

ISCZ 法では精度桁 9 以上で凸包を確認できた (図 5)。対して、近似値計算では精度桁を 100 まで上げたが凸包を確認することができなかった (図 6)。
近似値計算、ISCZ 法、正確計算での計算時間は、表 3 と表 4 の通りである。

| 入力点数 | 100 | 500 | 1000 | 5000 |
|---|---|---|---|---|
| 近似値 | 0.172 | 4.219 | 17.34 | 498.4 |
| ISCZ 法 | 1.891 | 45.39 | 181.2 | 4657 |
| 正確計算 | 6.812 | 160.0 | 596.9 | 31226 |

表 3: 3 次元凸包-整数入力（秒）

| 入力点数 | 100 | 500 | 1000 | 5000 |
|---|---|---|---|---|
| 近似値 | 0.421 | 5.110 | 20.65 | 448.2 |
| ISCZ 法 | 1.782 | 47.29 | 186.3 | 4854 |
| 正確計算 | 735.1 | 367843 | ＼ | ＼ |

表 4: 3 次元凸包-無理数入力（秒）

### 3.3 考察

近似値計算では精度桁を大きくしても正しい凸包を確認できなかった。対して、ISCZ 法では比較的低い精度桁で正しい凸包を確認できた。従って、3 次元凸包構成のギフト包装法アルゴリズムに対する ISCZ 法の有効性を確認できた。また、ISCZ 法は、正確計算より高速に結果を出力できた。特に入力が無理数点の場合は、正確演算では膨大な時間を要したため、ISCZ 法の有効性はより顕著である。

## 4 まとめ

- シンボルリストを最低限に抑えるための新しいアイデアを導入したことで、大幅な計算時間の短縮が確認できた。
- 3 次元凸包構成のギフト包装法アルゴリズムに対する ISCZ 法の有効性を確認できた。また、凸包構成の場合、得られた結果が真に正しい凸包であるかを、正確計算で求めることなしに検証することは一般に難しいが、ISCZ 法を使えば、**後の検証の必要もなく結果が真に正しいことを保証**してくれる。
- 今後の課題は、3 次元凸包構成において、Lazy Evaluation with Expression-dags([7]) と ISCZ 法の比較検討を行うことである。

## Acknowledgements

This work was supported by the Research Institute for Mathematical Sciences, an International Joint Usage/Research Center located in Kyoto University. Also, it was supported in part by JSPS KAKENHI Grant Number JP18K11172.

## 参 考 文 献

[1] Kiyoshi Shirayanagi, Moss Sweedler: A Theory of Stabilizing Algebraic Algorithms, Technical Report 95-28, Mathematical Sciences Institute, Cornell University, 1995. 92 pages.

[2] Kiyoshi Shirayanagi, Moss Sweedler: Remarks on Automatic Algorithm Stabilization, Journal of Symbolic Computation, Vol.26, No. 6, 1998, pp.761-766.

[3] Kiyoshi Shirayanagi, Hiroshi Sekigawa: Reducing Exact Computations to Obtain Exact Results Based on Stabilization Techniques. In *Proc. International Workshop on Symbolic-Numeric Computation 2009 (SNC2009)*, 2009, pp. 191-197.

[4] 関川 浩, 白柳 潔: 凸包アルゴリズムの安定化, 日本数式処理学会第 4 回大会資料, 1995, pp. 1-6.

[5] 白柳 潔, 関川 浩: 安定化理論に基づく ISCZ 法の凸包構成への応用, RIMS 講究録 第 1815 巻 2012, pp. 133-142.

[6] 白柳 潔, 関川 浩: 安定化理論に基づく ISCZ 法の有効性について, RIMS 講究録 第 1814 巻 2012, pp. 29-35.

[7] Stefan Schirra, Martin Wilhelm: On Interval Methods with Zero Rewriting and Exact Geometric Computation LNCS 10693, 2017, pp. 211-226.