数理解析研究所講究録
第1815巻 2012年 133-142

安定化理論に基づくISCZ法の凸包構成への応用

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

関川 浩†
東海大学理学部
HIROSHI SEKIGAWA
SCHOOL OF SCIENCE, TOKAI UNIVERSITY

1 はじめに

安定化理論 [9] は、近似計算で実行すると不安定と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。[6, 7] では、その安定化理論の新しい応用として、なるべく正確計算を軽減させながらも最終的に正確係数の正しい出力を得るための手法ISCZ法を提案し、Buchbergerアルゴリズムに適用して得られた実験結果を報告した。そこでは、有理係数よりも無理係数の方が有効であることはわかったが、もともとすべてを正確演算で実行した結果と比べて、必ずしも優位性を主張することができなかった。[8] では、その原因を追求し、Buchbergerアルゴリズムのように、再帰的な呼び出しが頻発するようなアルゴリズムでは、ISCZ法はあまり有効でないことを結論づけた。さらに、凸包アルゴリズムのように、平坦な構造をもったアルゴリズムには有効であることを予想した。本論文では、実際にISCZ法を凸包アルゴリズムに適用し、無理数を含む場合は格段に有効になることを示す。

まず、2節で、安定化理論とISCZ法について復習する。3節で、ISCZ法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た計算機実験について述べる。

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 [5, 9] を参照されたい。

- データは、すべて多項式環 $R[x_1, \ldots, x_m]$ 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、$R[x_1, \ldots, x_m]$ 内の加減乗である。
- データ上の述語は、不連続点をもつとすればそれは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 $C \geq 0$ 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。

*kiyoshi.shirayanagi@is.sci.toho-u.ac.jp
†sekigawa@tokai-u.jp

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なう。

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 $[A,\alpha]$ なる形で、$A \in R$, α は非負の実数。$[A,\alpha]$ は集合 $\{x \in R \mid |x-A| \leq \alpha\}$ を意味する。

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

$$[A,\alpha] * [B,\beta] = [A * B, \gamma_*].$$

ここに、γ_* は次を満たす。

$$|x-A| \leq \alpha, |y-B| \leq \beta \Rightarrow |x * y - A * B| \leq \gamma_*.$$

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 $[C,\gamma]$ に対し、

$|C| \leq \gamma$ ならば $[C,\gamma]$ を [0,0] に書き換えよ。

$|C| > \gamma$ ならばそのままとせよ。

区間演算の具体的な定義については、文献 [1] を参照されたい。

今、入力 $f \in R[x_1,\ldots,x_m]$ を

$$f = \sum_{i_1,\ldots,i_m} a_{i_1\ldots i_m} x_1^{i_1} \cdots x_m^{i_m}$$

と表したとき、f に対する近似列 $\{Int(f)_j\}_j$ を

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

で定義する。ここに、すべての $i_1,\ldots,i_m$ について、

$$|a_{i_1\ldots i_m} - (a_{i_1\ldots i_m})_j| \leq (\alpha_{i_1\ldots i_m})_j \text{ for } \forall j,$$

$$(\alpha_{i_1\ldots i_m})_j \to 0 \text{ as } j \to \infty.$$

このとき、単に

$$Int(f)_j \to f$$

と書く。

さて、$\mathcal{A}$ を安定化したアルゴリズムを $Stab(\mathcal{A})$ 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) $\mathcal{A}$ は不連続点 0 の代数的アルゴリズムで、入力 $f \in R[x_1,\ldots,x_m]$ 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、$Stab(\mathcal{A})$ は $Int(f)_j$ に対し正常終了し、

$$Stab(\mathcal{A})(Int(f)_j) \to \mathcal{A}(f).$$

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。主題のグレブナ基底の場合も、入力は多項式の集合である。本定理の Buchberger アルゴリズムに特化した証明は [4] に、より一般的な場合に対する厳密な証明は [9] にある。

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 1/9 だったとしよう。これらの精度 3 の（円形）区間はそれぞれ $[.333, .0005]$ と $[.111, .0005]$ である。さて、1/3 に対するシンボルを s、1/9 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、$[[.333, .0005], s]$ と $[[.111, .0005], t]$ となる。次に、これらの間の演算、例えば、加算を、

$$[[.333, .0005], s] + [[.111, .0005], t] = [[.333, .0005] + [.111, .0005], \dot{+}(s, t)]$$

と定義する。$[.333, .0005] + [.111, .0005]$ に対しては通常の区間演算を使う。シンボル部分 $\dot{+}(s, t)$ 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 $\dot{+}(s, t)$ であったとすれば、s に 1/3 を、t に 1/9 を代入し、$\dot{+}$ には加算の意味を与えて、$1/3 + 1/9 = 4/9$ 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

従来の ISZ 法 [10, 11, 6] は、述語を評価する直前で、任意の IS $[[E, \epsilon], s]$ に対し、$|E| \leq \epsilon$ ならば、s が何であれ、$[[E, \epsilon], s]$ を 0 に置き換える手法であった。それに対して、ISCZ 法は、$|E| \leq \epsilon$ ならばその都度、s を実数値に復元して、実際にそれが 0 であるかどうかを確かめるものである。詳細は以下の通りである。

$\mathcal{A}$ を不連続点 0 の代数的アルゴリズムとする。

[ISCZ 法]

R-to-IS 各入力係数 a をペア $[[\tilde{a}, \alpha], Symbol_a]$ に変換する。ここに、$[\tilde{a}, \alpha]$ は a の指定された精度の区間、$Symbol_a$ は a 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 間の演算を次のように実行する：

$$[[A, \alpha], s] + [[B, \beta], t] = [[A, \alpha] + [B, \beta], \dot{+}(s, t)]$$

$$[[A, \alpha], s] - [[B, \beta], t] = [[A, \alpha] - [B, \beta], \dot{-}(s, t)]$$

$$[[A, \alpha], s] \times [[B, \beta], t] = [[A, \alpha] \times [B, \beta], \dot{\times}(s, t)]$$

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、どういう演算が行なわれたかを記録する。

正しいゼロ書換え 述語を評価する直前で、任意の IS $[[E, \epsilon], s]$ に対し、$|E| \leq \epsilon$ ならば、s をそれに対応する実数値 $r(s)$ 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 $Stab(\mathcal{A})$ 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 $\mathcal{A}$ 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) $\mathcal{A}$ が入力 I で正常終了するとせよ。このとき、$\mathcal{A}$ 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、$\mathcal{A}(I)$ の出力と同じ結果を与える。

ISCZ法の利点は、ISZ法と違い、**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 $[[E,\epsilon],s]$ に対し、$|E| \leq \epsilon$ でない限り、正確計算をスキップすることができ、浮動小数点計算だけで済む。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、$|E| > \epsilon$ の場合が $|E| \leq \epsilon$ の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。

3 凸包アルゴリズムへの応用

[8] では、ISCZ法がBuchbergerのアルゴリズムに対してあまり有効でないのは、簡約など再帰的な呼び出しが頻繁に行なわれるため、シンボル係数のシンボルの膨張が激しいからであることを究明した。シンボルの膨張を防ぐために導入されたシンボルリストも功を奏さなかった。(シンボルリストの詳細については、[8] を参照されたい)。そこで、本論文では、再帰的な呼び出しのない、構造が平坦な凸包アルゴリズムに適用する。凸包アルゴリズムは複数存在するが、ここでは簡明なGrahamのアルゴリズム [2, 3] を選定した。そのアルゴリズムを記述する。

Grahamのアルゴリズム

Input: 有限集合 $S = \{P_1, P_2, \ldots, P_m\} \subset \mathbb{R}^2$

Output: S の凸包の頂点集合

1. [基準点] S の点を一つ選び、O とする。
2. [ソート] S の点を、O からの偏角が小さい順にソートする。ただし、偏角が同じ点が複数ある場合は、O から最も遠い点のみをとり、その他は捨てる。その結果を $Q_1, Q_2, \ldots, Q_n$ とし、$Q_0 = O$ とおく。
3. [走査] 内角 $Q_iQ_{i-1}Q_{i-2}$ が 180° 未満かどうか調べながら凸包を構成する。厳密に書くと次のようになる：

```
j := 0
for i from 0 to n do
    j := j + 1
    R_j := Q_i
    while j ≥ 4 and ∠R_j R_{j-1} R_{j-2} ≥ 180° do
        R_{j-1} := R_j
        j := j - 1
return [R_1, R_2, ..., R_j]
```

ここで、一般に3点 $P_1=(x_1,y_1), P_2=(x_2,y_2), P_3=(x_3,y_3)$ に対し、

$$\angle P_3P_2P_1 \geq 180^\circ \Leftrightarrow \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \leq 0$$

が成立するので、Grahamのアルゴリズムは不連続点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。

我々は、このアルゴリズムを適用するための入力例を4つの群に分けた。第一群は1000個の整数点、第二群は1000個の無理数点、第三群は5000個の整数点、第四群は5000個の無理数点からなる。各群に対し、4つの例を設けた。

第一群は、次の4例である。

例 1 $0 \leq X, Y \leq 100$ を満たすランダムに生成された整数の組 (X,Y) 1000点。

例 2 次を満たすランダムに生成された整数の組 (X,Y) 1000点。

$$-100 \leq X, Y \leq 100, \qquad X^2+Y^2 \leq 100^2.$$

例 3 次を満たすランダムに生成された整数の組 (X,Y) 1000点。

$$0 \leq X, Y \leq 100, \qquad X^2+Y^2 \leq 100^2, \qquad Y^2 \leq 3X^2.$$

すなわち、中心が原点、半径100、角度60°の扇形の内部か境界線にある1000点である。

例 4 原点と、次を満たすランダムに生成された整数の組 (X,Y) 999点。

$$1 \leq X, Y \leq 3000, \qquad X^2+Y^2 \leq 3000^2, \qquad \frac{9}{10} \leq \frac{Y}{X} \leq 1.$$

すなわち、中心が原点、半径3000の長細い扇形の内部か境界線にある1000点である。

第二群は、次の4例である。

例 5 $0 \leq X, Y \leq 100$ を満たすランダムに生成された整数の組 (X,Y) 1000点に対し、$(\sqrt{X}, \sqrt{Y})$ を作ったもの。

例 6 次を満たすランダムに生成された整数の組 (X,Y) 1000点に対し、$(sign(X)\sqrt{|X|}, sign(Y)\sqrt{|Y|})$ を作ったもの。ただし、$sign(X)$ は X の符号である。

$$-100 \leq X, Y \leq 100, \qquad X+Y \leq 100.$$

例 7 次を満たすランダムに生成された整数の組 (X,Y) 1000点に対し、$(\sqrt{X}, \sqrt{Y})$ を作ったもの。

$$0 \leq X, Y \leq 100, \qquad X+Y \leq 100, \qquad Y \leq 3X.$$

すなわち、中心が原点、半径10、角度60°の扇形の内部か境界線にある1000点である。

例 8 原点と、次を満たすランダムに生成された整数の組 (X,Y) 999点に対し、$(\sqrt{X}, \sqrt{Y})$ を作ったもの。

$$1 \leq X, Y \leq 3000, \qquad X+Y \leq 3000, \qquad \frac{9}{10} \leq \frac{Y}{X} \leq 1.$$

すなわち、中心が原点、半径 $10\sqrt{30}$ の長細い扇形の内部か境界線にある1000点である。

第三群は、次の 4 例である。

例 9 $0 \le X, Y \le 200$ を満たすランダムに生成された整数の組 (X, Y) 5000 点。

例 10 次を満たすランダムに生成された整数の組 (X, Y) 5000 点。

$$-200 \le X, Y \le 200, \qquad X^2 + Y^2 \le 200^2.$$

例 11 次を満たすランダムに生成された整数の組 (X, Y) 5000 点。

$$0 \le X, Y \le 400, \qquad X^2 + Y^2 \le 400^2, \qquad Y^2 \le 3X^2.$$

すなわち、中心が原点、半径 400、角度 60 ° の扇形の内部か境界線にある 5000 点である。

例 12 原点と、次を満たすランダムに生成された整数の組 (X, Y) 4999 点。

$$1 \le X, Y \le 6000, \qquad X^2 + Y^2 \le 6000^2, \qquad \frac{9}{10} \le \frac{Y}{X} \le 1.$$

すなわち、中心が原点、半径 6000 の長細い扇形の内部か境界線にある 5000 点である。

第四群は、次の 4 例である。

例 13 $0 \le X, Y \le 200$ を満たすランダムに生成された整数の組 (X, Y) 5000 点に対し、$(\sqrt{X}, \sqrt{Y})$ を作ったもの。

例 14 次を満たすランダムに生成された整数の組 (X, Y) 5000 点に対し、$(sign(X)\sqrt{|X|}, sign(Y)\sqrt{|Y|})$ を作ったもの。ただし、$sign(X)$ は X の符号である。

$$-200 \le X, Y \le 200, \qquad X + Y \le 200.$$

例 15 次を満たすランダムに生成された整数の組 (X, Y) 5000 点に対し、$(\sqrt{X}, \sqrt{Y})$ を作ったもの。

$$0 \le X, Y \le 400, \qquad X + Y \le 400, \qquad Y \le 3X.$$

例 16 原点と、次を満たすランダムに生成された整数の組 (X, Y) 4999 点に対し、$(\sqrt{X}, \sqrt{Y})$ を作ったもの。

$$1 \le X, Y \le 6000, \qquad X + Y \le 6000, \qquad \frac{9}{10} \le \frac{Y}{X} \le 1.$$

すなわち、中心が原点、半径 $20\sqrt{15}$ の長細い扇形の内部か境界線にある 5000 点である。

例 4 と例 12 の入力を図 1 に示す。
計算機は、dual core AMD(R) Opteron(R) processor (2.85GHz), 8GB RAM, Linux(R) OS である。
我々は、次の 2 種類の方法を試みた。

1. Maple 12 で実装した ISCZ 法。初期精度は 1 で、精度の増加幅も 1。
2. Maple 12 で実装した ISCZ 法。初期精度は 10 で、精度は倍々で増加。

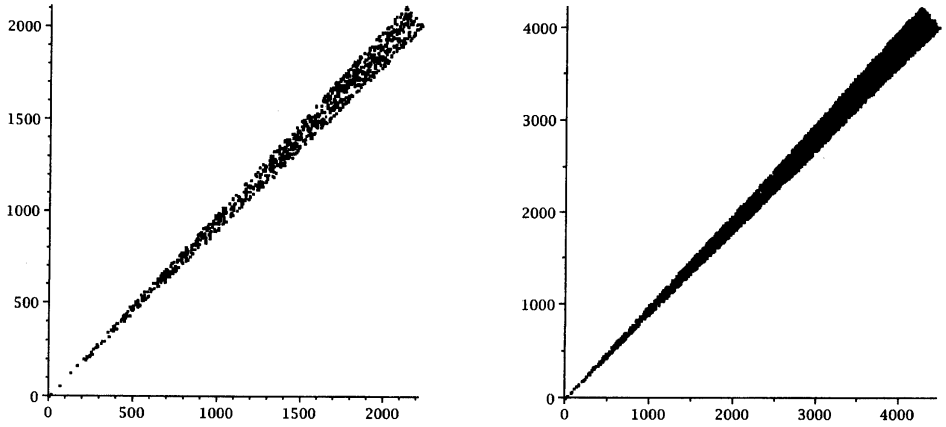

図 1: 例 4 (左) と例 12 (右).

表 1: 方法 1 – 1000 点 (整数)

例	時間 (秒)	成功精度 (桁)	ゼロ書き換え (個)	スキップ数 (個)	R_CH (秒)
1	0.8	6	312	10604	0.5
2	0.8	5	113	11256	0.5
3	0.9	6	196	11051	0.5
4	0.9	7	31	11464	0.6

表 2: 方法 1 – 1000 点 (無理数)

例	時間 (秒)	成功精度 (桁)	ゼロ書き換え (個)	スキップ数 (個)	R_CH (秒)
5	0.9	8	136	11106	8.5
6	1.1	8	64	11394	7.1
7	1.1	8	100	11329	8.6
8	1.1	9	39	11423	6.5

表 3: 方法 1 – 5000 点 (整数)

例	時間 (秒)	成功精度 (桁)	ゼロ書き換え (個)	スキップ数 (個)	R_CH (秒)
9	8.2	6	1693	64920	5.8
10	9.3	6	774	67406	6.6
11	9.1	7	1059	66612	6.3
12	9.8	8	174	68978	7.2

表 4: 方法 1 – 5000 点 (無理数)

例	時間 (秒)	成功精度 (桁)	ゼロ書き換え (個)	スキップ数 (個)	R_CH (秒)
13	10.6	9	660	67602	72.1
14	11.4	10	386	68439	59.5
15	11.1	9	560	68200	72.7
16	11.7	11	222	68695	54.2

表 5: 方法 2 – 1000 点 (整数)

例	時間 (秒)	成功精度 (桁)	ゼロ書き換え (個)	スキップ数 (個)	R_CH (秒)
1	0.8	10	312	10604	0.5
2	1.0	10	113	11256	0.5
3	0.9	10	196	11051	0.5
4	1.0	10	31	11464	0.6

表 6: 方法 2 – 1000 点 (無理数)

例	時間 (秒)	成功精度 (桁)	ゼロ書き換え (個)	スキップ数 (個)	R_CH (秒)
5	0.9	10	136	11106	8.5
6	1.0	10	64	11394	7.1
7	1.1	10	100	11329	8.6
8	1.1	10	39	11423	6.5

表 7: 方法 2 – 5000 点 (整数)

例	時間 (秒)	成功精度 (桁)	ゼロ書き換え (個)	スキップ数 (個)	R_CH (秒)
9	8.6	10	1693	64920	5.8
10	9.9	10	774	67406	6.6
11	9.6	10	1059	66612	6.3
12	10.2	10	174	68978	7.2

表 8: 方法 2 – 5000 点 (無理数)

例	時間 (秒)	成功精度 (桁)	ゼロ書き換え (個)	スキップ数 (個)	R_CH (秒)
13	10.3	10	660	67602	72.1
14	11.3	10	386	68439	59.5
15	10.7	10	560	68200	72.7
16	13.6	20	222	68695	54.2

方法 1 と方法 2 による実験結果をそれぞれ、表 1–4 と表 5–8 に示す。各表において、時間は、初期精度から成功精度に至るまでにかかった cpu 時間（単位は秒）の合計を表す。成功精度とは、ISCZ 法が終了した（成功に至った）ときの入力の浮動小数点精度である。ゼロ書き換えは、成功精度において、ゼロ書換えによって実際にゼロに書き換えられた区間の個数を表す。スキップ数は、成功精度において、"$|E| > \epsilon$" であるゆえに正確計算がスキップされたゼロでない係数の個数を表す。R_CH は、比較のために Maple 12 で自前で Graham のアルゴリズムを実装したもので、全過程を通して正確演算で計算したときの cpu 時間である。

これらの表により、次のことが結論づけられる。

> ISCZ 法は、Graham のアルゴリズムに対し、入力が有理数点の場合は、正確計算の直接法よりも 1.5 倍から 2 倍程度遅くなるが、**無理数点の場合は、5 倍から 8 倍程度高速**となる。従って、正確計算回避の効果が十分に現れているといえる。

さらに、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。

4 おわりに

Graham のアルゴリズムに対する ISCZ 法は、入力が無理数点の場合に大変有効であることが実証された。今後は、他のアルゴリズムにも適用し、ISCZ 法が有効となるアルゴリズムを特定していく予定である。

謝辞

本研究は科研費 21500026 の助成を受けたものである。

[1] G. Alefeld and J. Herzberger. *Introduction to Interval Computations, Computer Science and Applied Mathematics.* Academic Press, 1983.

[2] R. L. Graham. *An efficient algorithm for determining the convex hull of a finite planar set.* Information Processing Letters, 1(4):132–133, 1972.

[3] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[4] K. Shirayanagi. Floating point Gröbner bases. *Mathematics and Computers in Simulation*, 42(4-6):509–528, 1996.

[5] 白柳 潔. アルゴリズムの安定化理論. 数式処理, 5(2):2–21, 1997.

[6] 白柳 潔, 関川 浩. 安定化理論に基づく log method について. 京都大学数理解析研究所講究録掲載予定

[7] K. Shirayanagi and H. Sekigawa. Reducing Exact Computations to Obtain Exact Results Based on Stabilization Techniques. In *Proc. International Workshop on Symbolic-Numeric Computation 2009 (SNC2009)*, pages 191–197, 2009.

[8] 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. 京都大学数理解析研究所講究録掲載予定

[9] K. Shirayanagi and M. Sweedler. A theory of stabilizing algebraic algorithms. Technical Report 95-28, Mathematical Sciences Institute, Cornell University, 1995. 92 pages. (http://www.lab.toho-u.ac.jp/sci/is/shirayanagi/msitr95-28.pdf)

[10] K. Shirayanagi and M. Sweedler. Automatic algorithm stabilization. In *ISSAC'96 poster session abstracts*, pages 75–78, 1996.

[11] K. Shirayanagi and M. Sweedler. Remarks on automatic algorithm stabilization. *J. Symbolic Computation*, 26(6):761–766, 1998.