安定化理論に基づく計算履歴法の代数体上の因数分解への適用

Using the log method based on stabilization theory to factor polynomials over algebraic number fields

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 log method based on stabilization theory was proposed by Shirayanagi and Sweedler, to reduce the amount of exact computations as much as possible to obtain the exact results. This method is a floating-point interval method with 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. After the algorithm terminates, one evaluates the symbols of the output and get a result with exact coefficients. There are two kinds of the log methods: ISZ method and ISCZ method, depending on the treatment of zero rewriting for interval coefficients with symbols. In this paper, we show the efficiency of the ISZ method by applying it to the factorization of polynomials over algebraic number fields.

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。また、K.Shirayanagi と M.Sweedler は安定化理論の応用として、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計算履歴法を提案した。計算履歴法には ISZ 法と ISCZ 法の 2 種類がある。本論では、ISZ 法を代数体上の因数分解を求めるアルゴリズムに適用し、ISZ 法の有効性を示す。また、従来の ISZ 法に対する新たなアイデアとして、ISZ*法を提案した。ISZ*法の有効性についても示す。

まず 2 節では、安定化理論について復習する。3 節では、計算履歴法および ISZ*法について述べる。4 節では、代数体上の因数分解を求めるアルゴリズムの一つである Kronecker-Trager のアルゴリズムを述べる。4.3 節では、ISZ 法を代数体上の因数分解を求める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　安定化理論の復習

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

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

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。

多くの場合、係数収束より強い「台収束」が重要となる。まず「台」を定義する。多項式

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

の台とは、集合

$$\{x_1^{i_1} \cdots x_m^{i_m} \mid a_{i_1 \ldots i_m} \neq 0\}$$

で、これを $Supp(f)$ と記す。$f_j \to f$ が台収束するとは、係数収束し、かつ、ある有限の n があって、$j \geq n$ ならば、$Supp(f_j) = Supp(f)$ となることである。

ここで、次の方法によって台収束が実現できる。

$Stab(A)$ の実行後に、各区間係数にゼロ書換えを施す。

このアルゴリズムを $Stab(A)_R$ と書く。このとき、次の定理 ([3]) が成立する。

定理 2 (台収束)

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、$Stab(A)_R$ は $Int(f)_j$ に対し正常終了し、

$Stab(A)_R(Int(f)_j)$ は $A(f)$ に台収束する。

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

3 計算履歴法

安定化理論に基づく計算履歴法 ([4, 5]) を述べる。計算履歴法とは、アルゴリズム実行中に現れる係数の log (計算履歴) を取る方法である。計算履歴法は、ゼロ書換えの方法により、ISZ 法と ISCZ 法の 2 種類がある。

3.1 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計算履歴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。

アルゴリズム終了後、最終的なシンボルを正確係数に復元する。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。

3.2 手続き

計算履歴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。

シンボルリスト： 計算履歴法では、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。それを防ぐために、IS の代わりにペア $[[A, B], M]$ を用いる。ここに、$[A, B]$ は区間、M は入力シンボルまたは整数である。さらに、各ペア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])。アルゴリズム終了後、このシンボルリストから IS 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 $[c_1, c_2] = [a_1, a_2] \diamond [b_1, b_2]$ にゼロ書換えを施す。$K \leftarrow K + 1$ とし、新しい係数 (ペア) $[[c_1, c_2], K]$ 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。 $[[d_1, d_2], K]$ を次のように IS に変換する。
   - K がシンボル $\Rightarrow$ 何もしない。$[[d_1, d_2], K]$ がその IS そのものである。
   - K が整数 $\Rightarrow$ 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着くまでこの処理を行う。最終的に得られた係数 $[[a_1, a_2], u]$ がその IS である。

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。

ゼロ書換え： 計算履歴法は、ゼロ書換えの方法により、ISZ 法と ISCZ 法の 2 種類がある。

$$\begin{cases} \text{ISZ 法} & \Rightarrow \text{ 強制的ゼロ書換え} \\ \text{ISCZ 法} & \Rightarrow \text{ 保証されたゼロ書換え} \end{cases}$$

詳細についてはそれぞれ、3.3 と 3.4 で述べる。

計算履歴法について次が成立する。

定理 3 (計算履歴法の停止性と正当性)
A を不連続点 0 の代数的アルゴリズムとし、A が入力 I で正常終了するとせよ。このとき、A に対する計算履歴法は、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。

3.3 ISZ 法 (IS method with zero rewriting)

強制的ゼロ書換え： 任意の IS $[[A,B],s]$ に対し、$A \leq 0 \leq B$ ならば、s が何であれ、$[[A,B],s]$ を $[[0,0],s]$ に書き換えよ。

正当性検証： 実数係数に復元された出力が真に正しい出力かどうかをある手法によって確認する。YES ならば、それを返し、NO ならば、精度を上げて **R-to-IS** に戻る。ここに、ある手法とは “簡単に” 確認できる手法のことで、これが存在すると仮定する。

3.4 ISCZ 法 (IS method with *correct* zero rewriting)

保証されたゼロ書換え： 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。

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 $[[A,B],s]$ 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、ISCZ 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。
以下にシンボルリストを用いた ISCZ 法の具体例を記す。

$\frac{1}{3} \times \frac{1}{6} + \frac{4}{9} - \frac{1}{2}$ を厳密計算すると、$\frac{1}{3} \times \frac{1}{6} + \frac{4}{9} - \frac{1}{2} = 0$ となる。

この計算をシンボルリストを用いた ISCZ 法で計算すると、

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

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、区間係数が $-0.003220 \leq 0 \leq 0.051778$ となっている。従って、**保証されたゼロ書換え**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 $(-, 2, v)$ を参照する。
2. Symbol_List を使用して計算履歴を辿る。今回は、

$$\begin{aligned}
&(\dot{-}, 2, v) && \text{//Symbol_List の 3 番目の要素を参照する。}\\
&= (\dot{-}, (\dot{+}, 1, u), v) && \text{//Symbol_List の 2 番目の要素を参照する。}\\
&= (\dot{-}, (\dot{+}, (\dot{\times}, s, t), u), v) && \text{//Symbol_List の 1 番目の要素を参照する。}
\end{aligned}$$

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。

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

3.5 ISZ*法

従来の ISZ 法に対する新しいアイデアを提案する。基本的な方針は、台収束する精度桁を求めた後に、シンボルを正確係数に復元するというものである。

ISZ*法 (台収束の利用)

ISZ*法では、アルゴリズム終了後、最終的なシンボルを正確係数に復元する前に、出力が台収束しているかどうかを確認する。

- 台収束している　　⇒　シンボルを正確係数に復元。
- 台収束していない　⇒　精度を上げて、**R-to-IS** に戻る。

このように工夫することで、ISZ*法では正確係数の復元なしに、台収束に必要な精度桁を求めることができる。

従来の ISZ 法では、台収束していない場合でもシンボルを正確係数に復元し、正当性検証を行う。すなわち、復元した出力が正しくない場合、**R-to-IS** に戻り、その都度、正確係数の復元を行うことになる。対して、ISZ*法では台収束する精度桁を求めた後、シンボルを正確係数に復元する。すなわち、台収束していない場合、余計にシンボルを正確係数に復元することがない。

4.2 節に代数体上の因数分解での ISZ*法の適用例を述べる。

4　代数体上の因数分解アルゴリズムへの適用

前述の ISZ 法、ISCZ 法、ISZ*法を代数体上の因数分解アルゴリズムに適用する。代数体上の因数分解アルゴリズムには様々なものがある（[9, 10, 11]）が、本研究では Kronecker-Trager のアルゴリズムを選定した（[8, 12]）。以下、それを Trager アルゴリズムという。

4.1　Trager アルゴリズム

記法

$\mathbb{Q}$：有理数体。

α：$\mathbb{Q}$ 上代数的な数。

$m(x)$：α の $\mathbb{Q}$ 上の最小多項式。

α_i：$m(x) = 0$ の相異なる解。

$Norm(f)$：α_i に関するノルム。(f は $\mathbb{Q}(\alpha)$ 上の多項式)

$$Norm(f) \stackrel{\text{def}}{=} \prod_i f(\alpha_i, x) \in \mathbb{Q}[x]$$

ここで、$f(\alpha, x)$ は代数的数 α を便宜上変数として表現した式とする。また、このノルムの定義と終結式の性質により、次の式が成り立つ。

$$Norm(f) = res_t(f(t, x), m(t))$$

必要があれば、$x \leftarrow x - s\alpha\,(s \in \mathbb{Q})$ の変数変換により、$Norm(f)$ の無平方性を確保可能である。このとき、次の定理が成立する。

定理 4

$f \in \mathbb{Q}(\alpha)[x]$ とする。$Norm(f)$ が無平方ならば、$Norm(f)$ の $\mathbb{Q}$ 上の因数分解を $r_1 \cdot r_2 \cdots r_n$ と表すとき、f の $\mathbb{Q}(\alpha)$ 上の因数分解は

$$f(x) = GCD(f, r_1) \cdot GCD(f, r_2) \cdots GCD(f, r_n)$$

で与えられる。

以上が Trager アルゴリズムの原理である。次にアルゴリズムを述べる。

[Trager アルゴリズム]

入力 : 無平方な $f(\alpha, x) \in \mathbb{Q}(\alpha)[x], s \in \mathbb{Q}$

出力 : f の $\mathbb{Q}(\alpha)$ 上の既約因子 $\{f_1, \cdots f_n\}$

$m(t)$: α の $\mathbb{Q}$ 上の最小多項式

$N(x) = res_t(f(t, x - st), m(t))$

if $(deg(GCD(N(x), N(x)')) \neq 0)$ **then**

　$s = s + 1$　**again**

end if

$N(x) = \prod r_i(x)$

for each i **do**

$f_i(x) = GCD(f(x), r_i(x + s\alpha))$ // ユークリッドの互除法

end do

上記アルゴリズムの最後の GCD 計算をユークリッドの互除法を用いて求める。次にユークリッドの互除法のアルゴリズムを述べる。

[ユークリッドの互除法]
入力：多項式 f , $g \in \mathbb{R}[x]$
出力：$GCD(f, g)$

$a = f$, $b = g$
while $b \neq 0$ **do**
　$r =$ **rem** (a, b)　$//a$ を b で割った余り
　$a = b$
　$b = r$
end do
return a

ユークリッドの互除法は不連続点 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。本論の計算機実験では Trager アルゴリズムの最後の GCD 計算に対して、ISZ 法、ISCZ 法、ISZ*法を適用する。

4.2　ISZ*法の適用例

ここでは、Trager アルゴリズムに ISZ*法を適用し、以下の多項式の $\mathbb{Q}(\sqrt{2})$ 上の 因数分解を求める。

$$f(x) = x^4 + 7\sqrt{2}x^3 + (20 + \sqrt{2})x^2 + 46x - 84$$

$s = 0$, $\alpha = \sqrt{2}$ とし、α の $\mathbb{Q}$ 上の最小多項式と $Norm(f)$ を求める。

$\cdot\, m(t) = t^2 - 2$

$\cdot\, Norm(f)$

$= res_t(f(t, x), m(t))$

$= res_t(x^4 + 7tx^3 + (20 + t)x^2 + 46x - 84\,,\, t^2 - 2)$

$= x^8 - 58x^6 + 64x^5 + 230x^4 + 1840x^3 - 1244x^2 - 7728x + 7056$

このとき、$deg(GCD(Norm(f), Norm(f)') = 0$ より、$Norm(f)$ は無平方である。次に、$Norm(f)$ の $\mathbb{Q}$ 上での因数分解を求める。本論の計算機実験では、Maple2020 の無平方分解コマンド sqrfree を用いて求めている。このことに関して、$\mathbb{Q}$ 上の因数分解は、係数の分母の最小公倍数の乗算により、$\mathbb{Z}$ 上の因数分解に帰着される。例えば、$\mathbb{Z}$ 上の因数分解を求める方法として、Zassenhaus アルゴリズム ([7]) がよく知られている。$Norm(f)$ の $\mathbb{Q}$ 上での因数分解を求めると、

$x^8 - 58x^6 + 64x^5 + 230x^4 + 1840x^3 - 1244x^2 - 7728x + 7056$

$= (x^4 - 50x^2 + 120x - 72)(x^4 - 8x^2 - 56x - 98).$

$Norm(f)$ の既約因子をそれぞれ r_1 , r_2 とする。

$$\begin{cases} r_1(x) = x^4 - 50x^2 + 120x - 72 \\ r_2(x) = x^4 - 8x^2 - 56x - 98 \end{cases}$$

$GCD(f(x), r_1(x))$, $GCD(f(x), r_2(x))$ を ISZ*法を用いて求め、モニックに変換する。

$$\begin{cases} f_1 = GCD(f(x), r_1(x)) = x^2 + [[7.0601, 7.0813], 31]x + [[-8.4969, -8.4734], 32] \\ f_2 = GCD(f(x), r_2(x)) = x^2 + [[2.8262, 2.8307], 31]x + [[9.8947, 9.9043], 32] \end{cases}$$

ここで、2 つの因子 f_1 , f_2 が台収束しているかどうかを確認する。（従来の ISZ 法ではこの時点でシンボルを正確係数に復元し、正当性検証を行う。）
因数分解での台収束の確認と正当性検証は次の 4 Step で行う。

[因数分解での台収束の確認]

- **Step1**：全次数の確認
- **Step2**：係数収束の確認

} 台収束の確認

- **Step3**：シンボルの復元
- **Step4**：因数分解を展開し確認

} 正当性検証

$r = GCD(f(x), r_1(x)) \cdot GCD(f(x), r_2(x))$　とする。

- **Step1**：全次数の確認

  $deg(f(x)) = 4$

  $deg(r) = 4$

  従って、全次数の一致が確認できる。

- **Step2**：係数収束の確認

  r を展開し、モニックに変換し、各係数を $f(x)$ の近似値と比較する。

  $r = x^4 + [9.8863, 9.9120]x^3 + [21.351, 21, 476]x^2 + [45.806, 46.187]x + [-84.156, -83.842]$

  $f(x) \fallingdotseq x^4 + 9.8994x^3 + 21.414x^2 + 46x - 84$

  全ての区間係数に $f(x)$ の近似値が含まれているため、係数収束が確認できる。
  よって、台収束を確認したため、正当性検証に進む。

- **Step3**：シンボルの復元

  $GCD(f(x), r_1(x))$ と $GCD(f(x), r_2(x))$ のシンボルを正確係数に復元する。

$$\begin{cases} x^2 + [[7.0601, 7.0813], 31]x + [[-8.4969, -8.4734], 32] & \Rightarrow \quad x^2 + 5\sqrt{2}x - 6\sqrt{2} \\ x^2 + [[2.8262, 2.8307], 31]x + [[9.8947, 9.9043], 32] & \Rightarrow \quad x^2 + 2\sqrt{2}x + 7\sqrt{2} \end{cases}$$

- **Step4**：因数分解を展開し確認

  従って、$f(x)$ の $\mathbb{Q}(\sqrt{2})[x]$ 上での因数分解が得られる。

$$f(x) = (x^2 + 5\sqrt{2}x - 6\sqrt{2})(x^2 + 2\sqrt{2}x + 7\sqrt{2})$$

  この式を展開する。

$$f(x) = x^4 + 7\sqrt{2}x^3 + (20 + \sqrt{2})x^2 + 46x - 84$$

  よって、もとの $f(x)$ に一致し、因数分解が正しいことが確認できる。

4.3 計算機実験

ISZ 法、ISCZ 法、ISZ*法を代数体上の因数分解を求める Trager アルゴリズムに適用する。以下の 3 つの方法で計算機実験を行う。

1. ISZ 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 ISZ 法の比較実験を行い、因数分解が正しく出力されるかを確かめる。
2. ISZ 法、ISCZ 法、(近似を一切使わない) 厳密計算による、計算時間を比較する。
3. ISZ 法、ISZ*法による、計算時間を比較する。

本実験で使用する PC 及び実行環境は次の通りである。

- 使用コンピューター
  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
- 使用ソフト
  数式処理ソフト Maple 2020

例 1

$$f(x) = x^8 + (94 - 84\sqrt{2})x^7 - 7954\sqrt{2}x^6 + (2034 - 4324\sqrt{2})x^5 + (2848\sqrt{2} + 5898)x^4 + (7016\sqrt{2}x - 1848)x^3 - 4255x^2 - 1584\sqrt{2}x + 1430$$

$$\alpha = \sqrt{2}$$

- 近似値計算 (精度桁 25)

  GCD の結果

  ① $-0.00001180808078671529351067974$

  ② $8.748172922516159271658870 \times 10^{-6}$

  精度桁を 10000 まで上げても正しい台は出力されなかった。

- ISZ 法 (精度桁 25)

  GCD の結果

  ① $[[27.3110343093721831509, 27.3110343098612728794], 108]x^4+$
  $[[2567.2372251041256111874, 2567.2372251042252683823], 109]x^3+$
  $[[-463.48362147730069166636, -463.48362147709644447611], 110]x^2+$
  $[[-901.26413221743140409140, -901.26413221741562635171], 111]x+$
  $[[424.85998635408405727899, 424.85998635411470922839], 112]$

  ② $[[67.668319254976914294, 67.668319264790858501], 108]x^4+$
  $[[-8038.5862066034650812969, -8038.5862066025679434089], 109]x^3+$
  $[[-4402.082922667263046773, -4402.082922659653077210], 110]x^2+$
  $[[3451.084282258599229559, 3451.084282263535358505], 111]x+$
  $[[6220.334564630840862668, 6220.334564635619175053], 112]$

  モニックにし、シンボルから正確係数を復元する。

  ① $(x^4 - 84\sqrt{2}x^3 - 46\sqrt{2}x^2 + 51x + 65\sqrt{2})$

  ② $(x^4 + 94x^3 - 12\sqrt{2}x^2 - 33x + 11\sqrt{2})$

  したがって因数分解は、

  $f(x) = (x^4 - 84\sqrt{2}x^3 - 46\sqrt{2}x^2 + 51x + 65\sqrt{2})(x^4 + 94x^3 - 12\sqrt{2}x^2 - 33x + 11\sqrt{2})$

  これは展開すると、もとの $f(x)$ に一致している。

例 2

$f(x) = x^3 - (9\alpha^3 - 7\alpha^2 - 4)x^2 - (63\alpha^5 + 36\alpha^3 + 8\alpha - 5)x - 56\alpha^3 + 35\alpha^2 - 36\alpha + 20$

$\alpha = 2^{\frac{1}{5}} + 3^{\frac{7}{4}}$

- 近似値計算 (精度桁 40)

  GCD の結果

  ① $1.167485506521926416821079833809773952858 \times 10^{61}$

  ② $-4.265869269112584194873949588707547129660 \times 10^{120}$

  精度桁を 10000 まで上げても正しい台は出力されなかった。

- ISZ 法 (精度桁 40)

  GCD の結果

  ① $[[2.864261149601158680086200701827743623965 \times 10^{-15},$
  $2.864261149601158680086200701827743626712 \times 10^{-15}], 329]x+$
  $[[1.290549373627585180097933667477440357403 \times 10^{-12},$
  $1.290549373627585180097933667477440357689 \times 10^{-12}], 330]$

  ② $[[-7.880321329762034552729829592988331295 34 \times 10^{115},$

$$-7.880321329762018503936817357760334384 09\times 10^{115}],1043]x^2+$$
$$[[-1.857493500252287339760742482231340200006\times 10^{119},$$
$$-1.857493500252287339760742482231339608622\times 10^{119}],1044]x+$$
$$[[4.641332445122862606850408048643353865425\times 10^{117},$$
$$4.641332445122863535544214340651877012362\times 10^{117}],1045]$$

モニックにし、シンボルから正確係数を復元する。

① $(x+4+7\cdot 2^{\frac{2}{5}}+42\cdot 2^{\frac{1}{5}}3^{\frac{3}{4}}+189\cdot 3^{\frac{1}{2}})$

② $(x^2-9(2^{\frac{3}{5}}+243\cdot 3^{\frac{1}{4}}+9\cdot 2^{\frac{2}{5}}3^{\frac{3}{4}}+81\cdot 2^{\frac{1}{5}}3^{\frac{1}{2}})x-8\cdot 2^{\frac{1}{5}}-24\cdot 3^{\frac{3}{4}}+5)$

したがって因数分解は、

$$f(x)=(x+4+7\cdot 2^{\frac{2}{5}}+42\cdot 2^{\frac{1}{5}}3^{\frac{3}{4}}+189\cdot 3^{\frac{1}{2}})(x^2-9(2^{\frac{3}{5}}+243\cdot 3^{\frac{1}{4}}+9\cdot 2^{\frac{2}{5}}3^{\frac{3}{4}}+81\cdot 2^{\frac{1}{5}}3^{\frac{1}{2}})x-8\cdot 2^{\frac{1}{5}}-24\cdot 3^{\frac{3}{4}}+5)$$

これは展開すると、もとの $f(x)$ に一致している。

ISZ 法、ISCZ 法、厳密計算での計算時間は、表 1 の通りである。
以下の 5 つの例で比較実験を行う。例 1、例 2 は前述の例と同様である。

例 1 $\alpha=\sqrt{2}$ の 8 次式 (4 次式 $\times$ 4 次式)
例 2 $\alpha=2^{\frac{1}{5}}+3^{\frac{7}{4}}$ の 3 次式 (1 次式 $\times$ 2 次式)
例 3 $\alpha=\sqrt{3}+\sqrt{5}$ の 6 次式 (1 次式 $\times$ 2 次式 $\times$ 3 次式)
例 4 $\alpha=\sqrt{2}$ の 24 次式 (12 次式 $\times$ 12 次式)
例 5 $\alpha=31^{\frac{1}{6}}+19^{\frac{1}{5}}$ の 4 次式 (2 次式 $\times$ 2 次式)

	ISZ 法 (秒)	ISCZ 法 (秒)	厳密計算 (秒)
例 1	0.016	0.062	0.031
例 2	1164	6469	2428
例 3	0.109	0.297	0.156
例 4	30.73	73.04	40.00
例 5	18628	$>$ 1d	54311

表 1: ISZ 法 – ISCZ 法 – 厳密計算

次に、ISZ 法、ISZ*法での計算時間の比較は、表 2 の通りである。
例 1 から例 5 は表 1 と同様である。この実験では、ISZ 法と ISZ*法の計算時間に差を出すために、開始精度桁を必要精度桁よりも、5 低い値から開始する。また、精度桁の上げ幅を 1 としている。表の精度桁は、$\rightarrow$ の左側が開始精度桁であり、右側が正しい出力が確認できた必要精度桁である。

	ISZ 法 (秒)	ISZ*法 (秒)	精度桁
例 1	1.500	0.078	$20 \to 25$
例 2	8043	1178	$35 \to 40$
例 3	2.969	0.266	$55 \to 60$
例 4	1292	34.67	$33 \to 38$
例 5	>1d	18784	$62 \to 67$

表 2: ISZ 法 – ISZ*法

4.4 考察

近似値計算では精度桁を 10000 まで大きくしても正しい台を確認できなかった。対して、ISZ 法では正しい因数分解を出力することが確認できた。従って、代数体上の因数分解での Trager アルゴリズムに対する ISZ 法の有効性を確認できた。

次に計算時間を比較した結果、ISZ 法は ISCZ 法、厳密計算より高速に結果を出力した。特に代数的数が複雑な場合、ISCZ 法、厳密計算でより計算時間を要したため、ISZ 法の有効性はより顕著である。また、ISCZ 法は厳密計算より計算時間を要したため、有効性が見られなかった。

最後に、ISZ 法と ISZ*法の比較では、ISZ*法が ISZ 法より高速に結果を出力した。ISZ 法ではシンボルを正確係数に復元する処理を 5 回行っているのに対し、ISZ*法ではシンボルを正確係数に復元する処理を 1 回に軽減しているためである。

5 まとめ

- 代数体上の因数分解を求める Trager アルゴリズムに対する ISZ 法の有効性を確認できた。また、因数分解の場合、求めた式を展開することで最終出力の正しさを簡単に検証できることが大きな利点である。
- 本論文で新しく提案した ISZ*法は、精度桁が足りない場合、従来の ISZ 法より有効であると確認できた。
- 今後の課題は、他の代数的アルゴリズムに対して ISZ*法の有効性を確認すること。また、台収束の確認方法を理論的に明確にする必要がある。

Acknowledgements

This work was supported by the Research Institute for Mathematical Sciences, an International Joint Usage/Research Center located in Kyoto University.

参 考 文 献

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

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

[3] 白柳 潔, 関川 浩 : 安定化理論における台収束の応用について, RIMS 講究録 第 1568 巻 2007, pp.20-26.

[4] K. Shirayanagi, H. 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.

[5] 白柳 潔, 関川 浩 : 安定化理論に基づく log method について, RIMS 講究録 第 1666 巻 2009, pp.98-105.

[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] H. Zassenhaus : On Hensel factorization . I. J. Number Theory 1, 1969, pp.291-311.

[8] B. M. Trager : Algebraic Factoring and Rational Function Integration, Proc. SYMSAC, 1976, pp.219-226.

[9] P. J. Weinberger, L. P. Rothchild : Factoring Polynomials over Algebraic Number Fields, ACM Trans, Math. Soft 2, 1976, pp.335-350.

[10] E. Kaltofen : Factorization of polynomials. In *Computer algebra*, Springer, Vienna, 1983, pp.95-113.

[11] K. O. Geddes, S. R. Czapor, G. Labahn : Algorithms for Computer Algebra, 1992.

[12] K. Katamachi, K. Shiihara, T. Sasaki : On Kronecker-Trager's Factorization Method, RIMS 講究録 第 848 巻, 1993, pp.6-12.