数理解析研究所講究録
第1652巻 2009年 11-18

Comprehensive Gröbner system における Nabeshima algorithmの改良とその検証

篠原　直行
NAOYUKI SHINOHARA*
CREST JST / 立教大学
CREST JST / RIKKYO UNIV.

Abstract

CGS の計算を行う Nabeshima algorithm は, 場合分けの回数が少なくなるように Suzuki-Satou algorithm を改良したもので, それは elimination polynomial を使うことで可能になる. 本論文では, さらに場合分けの回数が少なくなると思われる elimination polynomial の選び方を提案し, その結果について述べる.

1　はじめに

まず, 本論文で使われる記号を紹介する.

K : 体. L : K の代数的閉包. $A=\{a_1,...,a_m\}$: パラメータ. $X=\{x_1,...,x_n\}$: 変数.
$<_A$: $\{\prod a_i^{e_i} : x_i \in A, e_i \in \mathbf{N}\cup 0\}$ における項順序. $<_X$: $\{\prod x_i^{e_i} : x_i \in X, e_i \in \mathbf{N}\cup 0\}$ における項順序.
$<_{A,X}$: $<_A$ と $<_X$ による $A << X$ なるブロック項順序. $\sigma : K[A] \to L$: 環準同型写像.
$lpp(f)$: f の $<_{A,X}$ による頭項. $lc(f)$: f の $<_{A,X}$ による頭係数. $lm(f)$: f の $<_{A,X}$ による頭単項式.
$lpp_A(f)$: f の $<_X$ による頭項. $lc_A(f)$: f の $<_X$ による頭係数. $lm_A(f)$: f の $<_X$ による頭単項式.
$\mathbf{V}(S)=\{a\in L^m : \sigma_a(f)=0,\ \forall f\in S\subset K[A]\}$.
$r\notin X\cup A$: 変数. $<_{A,r}$: $\{r^e\prod a_i^{e_i} : x_i\in A, e, e_i\in \mathbf{N}\cup 0\}$ における項順序.
$<_{A,r,X}$: $<_{A,r}$ と $<_X$ による $A, r << X$ なるブロック項順序.

パラメータ付のイデアルに対する Gröbner basis を考えるとき, 場合分けが必要であることは次の例からわかる.
例1）パラメータ $A=\{a_1,a_2\}$, 変数 $X=\{x\}$, $F=\{a_1x, a_2x^2\}$ としたときの Gröbner basis は, $a_1\neq 0$ ならば $\{x\}$, $a_1=0$ かつ $a_2\neq 0$ のときは $\{x^2\}$ となる.
このように場合分けされた Gröbner basis の集合を Comprehensive Gröbner System (CGS) といい以下で定義される.

定義 1.1 (Comprehensive Gröbner System (CGS)) Eq_i, $NotEq_i \subset K[A]$ とする. $S=\{\{Eq_1, NotEq_1, GB_1\},...,\{Eq_k, NotEq_k, GB_k\}\}$ が $F\subset K[A,X]$ に対する *Comprehensive Gröbner System* であるとは, S が以下の条件を満たす場合をいう.
1. $\cup_{i=1}^{k}(\mathbf{V}(Eq_i)\backslash\mathbf{V}(NotEq_i))=L^m$.

*shnhr@tvs.rikkyo.ne.jp

2. 各 $a_i \in \mathbf{V}(Eq_i)\backslash\mathbf{V}(NotEq_i)$ に対して, $\sigma_{a_i}(GB_i)$ は $\langle\sigma_{a_i}(F)\rangle$ の $L[X]$ における *Gröbner basis*.
さらに, 各 $\{Eq_i, NotEq_i, GB_i\}\}$ を S の断片とよぶ.

($eq \in Eq$ は "$= 0$" を意味し, $noteq \in NotEq$ は "$\neq 0$" を意味している. パラメータを含んだイデアルに対して, パラメータ空間を有限個の断片に分割して, それぞれの断片に対する Gröbner basis を Comprehensive Gröbner basis と呼び, それら全ての集合が Comprehensive Gröbner system S である.)

例 1) の Comprehensive Gröbner system は

$$S = \{\{[\,], [a_1], [x]\}, \{[a_1], [a_2], [x^2]\}, \{[a_1, a_2], [\,], [0]\}\}$$

となる.

2 Suzuki-Satou algorithm

Nabeshima algorithm は Suzuki-Satou algorithm を改良したものである. この節では Suzuki-Satou algorithm について概要を説明する.

定義 2.1 (stable) イデアル $I \subset K[A, X]$ が *ring homomorphism* σ と項順序 $<_A, <_X$ に対して *"stable"* であるとは, I に対して $\sigma(lm_A(I)) = lm(\sigma(I))$ が成り立つことをいう.

次の定理は Suzuki-Satou algorithm の主定理である.

定理 2.2 (Kalkbrener(1997)[1]) I は $K[A][X]$ のイデアル, $G = \{g_1, ..., g_s, g_{s+1}, ..., g_t\}$ は $<_{A,X}$ による I の *Gröbner basis* とする. さらに g_i は, $1 \leq i \leq s$ については $\sigma(lc_A(g_i)) \neq 0$, $s+1 \leq i \leq t$ については $\sigma(lc_A(g_i)) = 0$ となるように順序付けされているものとする. このとき次の三つの条件は同値である.
(*i*) I は σ, $<_A, <_X$ に対して *stable* である.
(*ii*) $\{\sigma(g_1), ..., \sigma(g_s)\}$ は $<_X$ による $\sigma(I)$ の *Gröbner basis* である.
(*iii*) すべての g_i $(s+1 \leq i \leq t)$ に対して, $\sigma(g_i)$ は $\{\sigma(g_1), ..., \sigma(g_s)\}$ を法として 0 に簡約される.

この定理により Suzuki-Satou algorithm による CGS の計算は, 最初に $<_{A,X}$ による Gröbner basis GB を計算し, GB の $<_X$ による頭係数が 0 か否かで場合分けを行う.

Algorithm 2.3 (Suzuki-Satou algorithm)

[Input] $F \subset K[A, X]$: a finite subset of polynomials. $<_A$, $<_X$: term orders.
[Ourput] CGS for F w.r.t. $<_{A,X}$.
S-S($F, <_A, <_X$){
1 GB =RedGröbnerBasis($F, <_{A,X}$);
2 $t = square\_free(lcm(lc_A(g_1), ..., lc_A(g_k)))$, where $g_i \notin K[A]$ and $lc_A(g_i) \notin K$;
1 $Eq = GB \cap K[A]$;
3 $CGS = \{Eq, [t], GB\backslash Eq\}$;
4 for (a factor t_i of t){
5 $CGS = CGS \cup$ S-S($GB \cup \{t_i\}, <_{A,X}$);
6 }
7 return(CGS);
8}

Suzuki-Satou algorithm は, 最初に, パラメータ A を変数とし $A << X$ なるブロックオーダー $<_{A,X}$ で, $F \subset K[A,X]$ に対して Gröbner Basis $GB \subset K[A,X]$ を計算する. このあと, この GB を $K[A]$ 係数の多項式集合とみなし, さらに, $g \in GB \backslash K[A]$ なる全ての g の頭係数 $lc_A(g) \in K[A]$ を "$\neq 0$" と仮定して一つの断片を生成する. その断片は, $Eq = GB \cap K[A]$ として, $\{Eq, [t], GB \backslash Eq\}$ と書くことができ, Eq は "$= 0$", $[t]$ は "$\neq 0$" を意味する. そして $GB \backslash Eq$ は, $\alpha \in V(Eq) \backslash V([t])$ に対する, $\sigma_\alpha(F)$ の $L[X]$ 上の Gröbner basis $\sigma_\alpha(GB \backslash Eq)$ を意味する.

次に, $t = 0$ となる場合を考えることになるが, それは GB に t の素因子 t_i を加えたもの, つまり $GB \cup \{t_i\}$ に対して同様のことを繰り返していくことで断片を得ることができ, 最終的に CGS を計算できるわけである.

例 2) $A = \{a_1, a_2, a_3\}$, $X = \{x\}$, $F = \{a_1 x, a_2 x^2, a_3 x^3\}$ とする. このとき Suzuki-Satou algorithm による F の CGS は

$$\begin{aligned} CGS = \quad & \{\{[\,], [a_1 a_2 a_3], [a_1 x, a_2 x^2, a_3 x^3]\}, \{[a_1], [a_2 a_3], [a_2 x^2, a_3 x^3]\}, \{[a_2], [a_1 a_3], [a_1 x, a_3 x^3]\}, \\ & \{[a_3], [a_1 a_2], [a_1 x, a_2 x^2]\}, \{[a_1, a_2], [a_3], [a_3 x^3]\}, \{[a_1, a_3], [a_2], [a_2 x^2]\}, \{[a_2, a_3], [a_1], [a_1 x]\}, \\ & \{\{[a_1, a_2, a_3], [\,], [0]\}\} \end{aligned}$$

となる.

3 Nabeshima algorithm

この節では, Nabeshima algorithm の概要について説明する. 断片の個数は CGS の計算の仕方, (項順序など) によって変わる. Nabeshima algorithm は断片の個数が少なくなるように Suzuki-Satou algorithm を改良したものであり, それは下で述べる "elimination polynomial" の性質 (1) によるものである.

定義 3.1 (elimination polynomial) GB は $K[A,X]$ 上の *reduced Gröbner basis* とする. このとき, $f \in GB \backslash K[A]$ に対して

$$lpp_A(f) \mid lpp_A(g),\ g \neq f \tag{1}$$

なる $g \in GB \backslash K[A]$ が存在するとき, f を *"elimination polynomial"* とよぶ.

この性質は, $L[X]$ において Gröbner basis を簡約することを考えた場合に, 断片の数を減らせることにつながる.

まずは Nabeshima algorithm の主定理とアルゴリズムを紹介する.

定理 3.2 ([2]) F は $K[A][X]$ の有限集合で, $G = \{g, g_1, ..., g_s\}$ は $<_{A,X}$ による F の *Gröbner basis* とする. $r = 1/lc_A(g)$, $g' = lpp_A(g) + r(g - lm_A(g))$ とし, $F' = \{g'\} \cup (G \backslash \{g\}) \subset K[A, r, X]$ とする. さらに, G' を $<_{A,r,X}$ による F' の *Gröbner basis*, G'' を G' の r に $1/lc_A(g)$ を代入して分母を払ったもの, $\{h_1, ..., h_t\} := \{lc_A(f) \in K[A] : f \in G''\}$ とする. このとき, $h = lcm(h_1, ..., h_t)$ と $\forall \alpha \in L^m \backslash (\mathbf{V}(lc_A(g)) \cup \mathbf{V}(h))$ に対して, $\sigma_\alpha(G'')$ は $<_X$ による $\langle \sigma_\alpha(F) \rangle$ の *Gröbner basis* である.

Algorithm 3.3 (Nabeshima algorithm)

[Input] $F \subset K[A,X]$: a finite subset of polynomials. $<_A$, $<_X$: term orders. $U \in \mathbf{N}$.
[Ourput] CGS for F w.r.t. $<_{A,X}$.
Nabe$(F, U, <_A, <_X)$\{

```
1    CGS =N-Main(F, NotEq, N, U, <_A, <_X);
2    return(CGS);
3}
```

Algorithm 3.4 (N-Main algorithm)

[Input] $F \subset K[A, r, X]$: a finite subset of polynomials. $NotEq \in K[A]$. $N, U \in \mathbb{N} : N < U$.
$<_{A,r}$, $<_X$: term orders.
[Ourput] CGS for F w.r.t. $<_{A,X}$ on $\mathbb{V}(GB \cap K[A]) \backslash \mathbb{V}(NotEq)$, where GB is the Gröbner basis in a segment of the CGS.

```
N-Main(F, NotEq, N, U, <_A, <_X){
1    G_0 := GBasis(F, <_{A,r,X});
2    G := Transform(G_0, NotEq);
7    E := {q : an elimination polynomial of G};
8    if(E ≠ ∅ and N ≤ U){        // Nabeshima step
9        Select q ∈ E s. t. lc_A(q) is the lowest element in lc_A(E);
10       q* := ht_A(q) + r · (q − hm_A(q));
11       G* := (G\{q}) ∪ {q*};
12       {t_1, ..., t_k} := {t_i ∈ factor(lc_A(q))};
13       t := lcm(NotEq, ∏ t_i);
14       if(V(G* ∩ K[A])\V({t}) ≠ ∅){
15           CGS_1 :=N-Main(G*, t, N + 1, U, <_A, <_X);        // t ≠ 0.
16       };
17       CGS_2 :=N-Main(G∪{t_1}, NotEq, 0, U, <_A, <_X)∪...∪N-Main(G∪{t_k}, NotEq, 0, U, <_A, <_X);
19       CGS := CGS_1 ∪ CGS_2;
20   }else{      // Suzuki-Sato step
21       S := {h_1, ..., h_l} = {h_i ∈ factor(lc_A(g)) : lc_A(g) ∉ K, g ∈ G, V(h_i) ⊄ V({NotEq}.};
22       h := lcm(NotEq, ∏ h_i);
23       CGS := {G ∩ K[A], h, G};
24       if(S ≠ ∅){
25           for(h_i ∈ S){
26               CGS := CGS ∪ N-Main(G ∪ {h_i}, NotEq, 0, U);
27           }
28       }else{
29           CGS := {(G ∩ K[A], NotEq, G)};
30       }
31   }
32   return(CGS);
33}
```

Nabeshima algorithm のキーポイントは以下に述べる二点である. 一つは, 単純に, $\{lc_A(g) \in K[A] : g \in GB\backslash K[A]\}$ の個数が少ないほど断片の個数が少なくなると考えることができること. もう一つは, Suzuki-Satou algorithm も Nabeshima algorithm も, まず最初に A も変数とみなして $K[A, X]$ 上での reduced

Gröbner basis GB を計算し, そのあとで A をパラメータとみなす, つまり $<_{A,X}$ ではなく $<_X$ で場合分け（断片の生成）をするということ.

この性質により, すべての $f,g \in GB$ に対して $lpp(f) \nmid lpp(g)$ となる. しかし, $<_X$ で考えた場合, $f,g \in GB \backslash K[A]$ かつ $lpp_A(f) \nmid lpp_A(g)$ となるときが, つまり elimination polynomial が存在するときもある. （**例2** でみると, $lpp(a_1x) = a_1x \nmid lpp(a_2x^2) = a_2x^2$ だが $lpp_A(a_1x) = x \mid lpp_A(a_2x^2) = x^2$ が成り立つということ.) CGS は $L[x]$ 上での Gröbner basis について考えるのであった. いま f を GB の eliminationa polynomial とし $lpp_A(f) \mid lpp_A(g), f \neq g, g \in GB$ とする. このとき, $\sigma_\alpha(lc_A(f)) \neq 0$ なる $\alpha \in L^m$ に対して, $\sigma_\alpha(lpp(f))$ は $\sigma_\alpha(lpp(g))$ を割り切るため, $L[x]$ 上での簡約操作を $\sigma_\alpha(GB)$ にしたときに $\sigma_\alpha(g)$ は消える. これは $lc_A(f) \neq 0$ のときは $lc_A(g)$ の場合分けが不要であることを意味する.

そこで, Nabeshima algorithm では新しい変数 $r \notin A \cup X$ を用意し, ellimination polynomial f を

$$q = lpp_A(f) + r(f - lm_A(f))$$

と置き換えた集合 $GB' = q \cup (GB \backslash \{f\})$ に対して 再度 $<_{A,r,X}$ による reduced Gröbner basis GBr を計算することによって, $lpp_A(f) \mid lpp_A(g)$ なる $g \in GB$ の消去を行う. (r は $lc_A(f)^{-1}$ を意味し, q が $lc_A(f)^{-1}f$ を意味していることに注意.) 実際の計算では GBr の r に $lc_A(f)^{-1}$ を代入し分母を払う作業を行うことで CGS の計算をするわけである. (この操作はアルゴリズム中の $Transform$ が行う.)

再び **例2**, $A = \{a_1, a_2, a_3\}, X = \{x\}, F = \{a_1x, a_2x^2, a_3x^3\}$ について考える. まず GB =RedGröbnerBasis $(F, <_{A,X})$ を計算するが, この場合, GB は F 自身である. したがって, Suzuki-Satou algorithm では, a_1, a_2, a_3 が 0 でないとして断片を生成し, ここから $a_1 = 0, a_2 = 0, a_3 = 0$ の場合をそれぞれ考えることになり, 最終的には 8 個の断片が生まれる. （重複するものを消去しなければ 16 個.)

ところが Nabeshima algorithm で CGS を計算すれば断片の個数は 4 個に減少する. **例2** の F では elimination polynomial が二個存在し, それは a_1x と a_2x^2 である. ここで, $<_X$ で頭項の低い方, a_1x について注目する. CGS では $L[X]$ 上での Gröbner basis を求めたいのであった. 従って, $a_1 \neq 0$ の場合は, 可逆元 a_1^{-1} が存在するとしてよく, a_1x に a_1^{-1} を掛けた x と a_1x を入れ替えた集合 $F' = \{x, a_2x^2, a_3x^3\}$ に対して GB' =RedGröbnerBasis$(F', <_{A,X})$ を計算すればよいわけである. その結果 $GB' = \{x\}$ を得ることができ, 断片 $\{[\,], [a_1], [x]\}$ が生成される. この断片がもつ重要な意味は, $a_1 \neq 0$ のときは a_2, a_3 の場合分けが不要であるということである . (つまり Suzuki-Satou algorithm による CGS の断片 $\{[\,], [a_1a_2a_3], [a_1x, a_2x^2, a_3x^3]\}$, $\{[a_2], [a_1a_3], [a_1x, a_3x^3]\}$, $\{[a_3], [a_1a_2], [a_1x, a_2x^2]\}$, $\{[a_2, a_3], [a_1], [a_1x]\}$ は一つにまとめることができる.)

次に, $a_1 = 0$ なる場合を計算することになるが, これは Suzuki-Satou algorithm の場合と同様ステップを行う. つまり, $GB \cup \{a_1\} = \{a_1, a_1x, a_2x^2, a_3x^3\}$ に対して Nabeshima algorithm の手順を行うことになり, 最終的には

$$CGS = \{\{[\,], [a_1], [x]\}, \{[a_1], [a_2], [x^2]\}, \{[a_1, a_2], [a_3], [x^3]\}, \{[a_1, a_2, a_3], [\,], [0]\}\}$$

となるため断片の個数は 4 個となる.

4 elimination polynomial と断片の個数

Nabeshima algorithm において elimination polynonmial の選び方は断片の個数に大きく影響する. この節ではより断片の個数が少なくなると考えられる elimination polynonmial の選び方を提案する.

4.1 Nabeshima algorithm での elimination polynonmial の選び方

前節でみたように, 例2 では elimination polynomial が二つ存在し, それらは a_1x と a_2x^2 であった. ここでは a_2x^2 を選んだ場合の計算についてみてみる. $r = a_2^{-1}$ として, 多項式を入れ替えたもの $F1'$ は $\{a_1x, x^2, a_3x^3\}$ となり, その $<_{A,r,X}$ による Gröbner basis $GB1$ は $\{a_1x, x^2\}$ であり, これは r を含まないので $Transform$ の操作後に得られる集合もそれ自身である. $GB1$ では elimination polynomial a_1x が存在するため, 再度同様のステップを繰り返すことになる. すなわち, 新たに $r = a_1^{-1}$ として $GB1$ から多項式の入れ替えで $F2' = \{x, x^2\}$ が得られ, その $<_{A,r,X}$ による Gröbner basis $GB2$ は $\{x\}$ となる. ここで, (最初に $a_2 \neq 0$ とした後で $a_1 \neq 0$ としたことに注意して,) 断片 $\{[\,], [a_1a_2], [x]\}$ が得られることがわかる. よって, 最初に a_2x^2 を選んだときの断片は $a_1 \neq 0$ かつ $a_2 \neq 0$ なるときであるが, 前節でみたように $a_1 \neq 0$ の場合は a_2 の場合分けが不要であったことから, この節での計算は断片の個数が多くなる可能性が高いことになる. 実際に, 先に a_2x^2 を選んだ場合の CGS は下のように五つの断片を持ち, 前節の場合よりも一つ断片が多い.

$$CGS = \{\{[\,], [a_1a_2], [x]\}, \{[a_1], [a_2], [x^2]\}, \{[a_2], [a_1], [x]\}, \{[a_1, a_2], [a_3], [x^3]\}, \{[a_1, a_2, a_3], [\,], [0]\}\}$$

Nabeshima algorithm では elimination polynonmial が複数存在する場合, その中から $<_X$ で頭項が最も低いものを選ぶ (同順位のものが複数ある場合は任意のものを選ぶ).

その理由は, $<_{A,r,X}$ において, elimination polynonmial の置き換えで現れたモニックな多項式の頭項より低い頭項をもつ多項式が簡約の際に影響を受けないためである. (この節の例2の計算では, x^2 で a_1x が簡約されなかった.) 経験的にこの選び方は効率が良いといえる.

ただ, 逆に $<_{A,r,X}$ において, elimination polynonmial q の置き換えで現れたモニックな多項式 q^* の頭項より高い頭項をもつ多項式 h が簡約の際に影響を受けない例もある. 例えば, $X = \{x, y\}$, $A = \{a, b\}$, $<_X$ が $x > y$ なる辞書式順序, $q = ay$, $h = bx$ である場合, h は $q^* = y$ によって簡約されない.

この事実より, もともとの選び方より断片の個数が少なくなりやすいと考えられる elimination polynomial の選び方を以下で述べる.

4.2 優先されるべき elimination polynomial

まず次の例を Nabeshima algorithm で計算した場合を考えてみる.

例3 $A = \{a, b, c, d, e, f, g\}$, $X = \{x, y\}$, $<_X$ は $x > y$ なる辞書式順序,

$$F = \{ax, bx^2, cx^2 + dy + e, fy^2, gy^3\}$$

とすると, その $<_{A,X}$ による F の reduced Gröbner basis GB は次のようになる.

$$GB = \{geb, feb, gea, fea, -dby - eb, day + ea, fy^2, gy^3, ax, bx^2, cx^2 + dy + e\}.$$

(ただし, この計算結果は $GB \cap K[A]$ の根基計算して GB に加え, その集合に対して再度 reduced Gröbner basis の計算を行い, それを GB におきなおしたものである.) 従って, elimination polynomial は $-dby - eb, day + ea, fy^2, ax, bx^2, cx^2 + dy + e$ で, その中から Nabeshima algorithm で選ばれうるものは, $x > y$ であることから.

$$-dby - eb, day + ea$$

の二つである. $-dby - eb$ を選べば db が 0 かそうでないか, $day + ea$ を選べば da が 0 かそうでないか, という断片の分岐から CGS の計算が始まる. 一見どちらを選んでも大差はないように見えるが, その計算

結果は, $-dby-eb$ を選んだ場合は断片の個数が 28 個, $day+ea$ を選んだ場合は断片の個数が 19 個 となる. この差には理由があり $L[x]$ での $\sigma_\alpha(F)$ のイデアルを見ることでわかる.

$db \neq 0$ の場合を考えてみると, つまり $\sigma_\alpha(db) \neq 0$ より,

$$\begin{aligned}\sigma_\alpha(\langle F\rangle) &= \sigma_\alpha(\langle ax, x^2, cx^2+dy+e, fy^2, gy^3\rangle) \qquad (b\neq 0)\\ &= \sigma_\alpha(\langle ax, x^2, dy+e, fy^2, gy^3\rangle)\\ &= \sigma_\alpha(\langle ax, x^2, y+ed^{-1}\rangle) \qquad (d\neq 0) \end{aligned} \tag{2}$$

となり, a の場合分けが必要なことが分かる. 一方, $da \neq 0$ の場合を同様にして考えてみると,

$$\begin{aligned}\sigma_\alpha(\langle F\rangle) &= \sigma_\alpha(\langle x, bx^2, cx^2+dy+e, fy^2, gy^3\rangle) \qquad (a\neq 0)\\ &= \sigma_\alpha(\langle x, dy+e, fy^2, gy^3\rangle)\\ &= \sigma_\alpha(\langle x, y+ed^{-1}\rangle) \qquad (d\neq 0) \end{aligned} \tag{3}$$

となり, b の場合分けが不要であることが分かる. これらのことから a の分岐を先にすれば, b の分岐を先に行った場合よりも断片の個数が減る可能性が高いことが分かる. (実際に前者の方が断片の個数が少ないのであった.)

elimination polynomial の選択で, 下で定義される "優先されるべき elimination polynomial" を選ぶことで, より断片がの個数が少なくなると考えられる. (実際に**例3** の場合の後者の計算結果はこの方法で得られたものである.)

定義 4.1 (優先されるべき elimination polynomial) $E \subset K[A,X]$ を $<_{A,X}$ による *Gröbner basis GB* の *elimination polynomial* 全ての集合とする. さらに

$$PE := \{q \in E : lpp_A(q') \nmid lpp_A(q)\ \forall q' \in E \text{ s.t. } lpp_A(q') \neq lpp_A(q)\} \tag{4}$$

とする. このとき PE の中で $<_X$ に対して最高位の頭項をもつ多項式を *"優先されるべき elimination polynomial"* と呼ぶ.

例3 の場合, "優先されるべき elimination polynomial" は ax である.

この "優先されるべき elimination polynomial" によって分岐を決定することで断片の個数は, もともとの elimination polynomial 選び方の場合より, 少なくなると考えられる.

その理由二つあり, 一つは先にも述べた, elimination polynonmial の置き換えで現れたモニックな多項式 q^* の頭項より, 高い頭項をもつ多項式 h ("優先されるべき elimination polynomial" など) が q^* による簡約の際に影響を受けない場合があることである. (**例3** の場合, (2) でみたように ax, x^2 は $y+ed^{-1}$ によって簡約されない.) これは, h のような簡約されない多項式の頭係数によるさらなる分岐が必要となること, つまり, その comprehensive Gröbner system の計算において, このような多項式の頭係数による分岐は必須となることを意味している. このことから, 逆に計算過程で消えにくい多項式, つまり (4) を満たす多項式の頭係数から分岐を始める方が効率が良いと考えられる.

この理由はもともとの選び方 (最も低い頭項をもつものを選ぶ方法) でもいえることで新しい方法と差はないが, 次の理由でで新しい方法が優れていると考えられる. それは, Gröbner basis GB を計算した際に, GB の中で頭項の低い多項式 ℓ の頭係数は, それより高い頭項をもつ多項式 h の頭係数の影響を受けている場合があることと (この逆はないことに注意) , またそのことによって, ℓ の頭係数をみても分岐の優先順位が不明瞭となることであり, これは S-polynomial の計算の性質によるものである. **例3** で説明すると,

もともとの方法で選ばれる elimination polynomial は $-dby-eb$ と $-dby-eb$ である. $day+ea$ は bx^2 と cx^2+dy+e の, $day+ea$ は ax と cx^2+dy+e の S-polynomial である. 従って $-dby-eb$ と $day+ea$ の頭係数 db, da はそれぞれ bx^2 と ax 頭係数の影響を受けている. b より a の分岐を先にした方が良いのは $lpp_A(ax) \mid lpp_A(bx^2)$ の関係からはわかるが, $-dby-eb$ と $day+ea$ からはわからないわけである.

5 結果と考察

$A=\{a\}$, $X=\{x,y,z\}$, $<_X$ は $x>y>z$ なる辞書式順序とする. 新しい elimination polynomial の選び方ともともとの選び方に対して, あとの各条件でそれぞれ１０例ずつの $\{f_1,f_2\} \subset K[A,X]$ に対して, $<_{A,X}$ による comprehensive Gröbner system を計算した.

	$\deg_x(f_1)$	$\deg_y(f_1)$	$\deg_z(f_1)$	$\deg_a(f_1)$	old_average	new_average
(I)	2	2	0	2	5.6	5.6
(II)	2	2	0	3	6.4	6.4
(III)	3	4	0	1	6.9	6.9
(IV)	2	2	2	1	6.9	6.6
(V)	3	2	2	1	9.6	9.5
(VI)	3	3	2	1	11	10.8

(f_1 と f_2 は同じ条件である.)

この表から, 変数の個数が増えると徐々に新しい elimination polynomial の選び方による効果があらわれるとみられる.

参 考 文 献

[1] Kalkbrener, M. On the Stability of Grobner Bases Under Specializations. *Journal of Symbolic Computation* 24:pp51-58,1997.

[2] Nabeshima, K. A speed-up algorithm for computing Comprehensive Gröbner systems. *Proceedings of the International Symposium on Symbolic and Algebraic Computation* (ISSAC 2007), ACM-Press. pp.299-306

[3] Suzuki, A. and Sato, Y. A simple algorithm to compute Comprehensive Gröbner Bases using Gröbner bases. In Dumas, J-G.,editor, *International Symposium on Symbolic and Algebraic Computation*, pp326-331. ACM Press, 2006.