剰余列とGCDによる辞書式グレブナー基底計算に対する種々の技法

Various Techniques for Computing LEX-order Gröbner Basis by Remainder Sequences and GCDs

筑波大学・数理物質系 名誉教授　　佐々木 建昭[*1]
Tateaki Sasaki
Professor emeritus, University of Tsukuba

Abstract

For a system of m+1 polynomials in main variables $(\boldsymbol{x}) = (x_1, \ldots, x_{m\geq 2})$ and sub-variables $(\boldsymbol{u}) = (u_1, \ldots, u_{n\geq 1})$, we have two famous methods of eliminating $\boldsymbol{x}$ and obtain polynomials in $\boldsymbol{u}$ usually: the resultant method and the Gröbner basis method w.r.t. the term-order $x_1, \ldots, x_m \succ u_1, \ldots, u_n$. The latter method gives us the lowest order element of the ideal but the method is mostly very slow. The former method is quite fast but the resultant is a multiple of the lowest order element, and the multiplier which we call the extraneous factor, is often quite large. In recent several years, the author studied to generate polynomials which are small multiples of corresponding elements of the Gröbner basis. As for the system of two polynomials $\{G, H\} \subset \mathbb{Q}[x, \boldsymbol{u}]$, where G and H are relatively prime, he found that the lowest order element of elimination ideal $\langle G, H\rangle \cap \mathbb{Q}[\boldsymbol{u}]$ can be computed by the PRS (Polynomial Remainder Sequence) and the GCD ($\Rightarrow$ Theorem 1 [6]). As for systems of $m \geq 2$, the situation is complicated, so he made the situation simple by defining *healthy system* and confining himself to treat only healthy systems; most of actual systems are healthy. Let $\mathcal{F} := \{F_1, \ldots, F_{m+1}\}$ be a healthy system and $\mathrm{GB}(\mathcal{F})$ be the Gröbner basis of $\mathcal{F}$ w.r.t. the lexicographic order. Then, we have $\mathrm{GB}(\mathcal{F}) \cap \mathbb{Q}[\boldsymbol{u}] = \{\widehat{S}\}$ ($\Rightarrow$ Theorem 2 [7]). On the basis of Theorems 1 and 2 mostly, he proposed a method to enhance Buchberger's algorithm in [5]. The proposed method is rather complicated and contains many problems, computational as well as theoretical. In this paper, after reviewing the new method as well as Theorems 1 and 2 in Sect. 2, we explain computational techniques in details which are necessary for making the new method efficient. We will show by experiments that, among various computational problems, computation of cofactors by the extended Euclid's method is the most serious problem, and we will propose a sophisticated algorithm for solving this problem.

1　まえがき

1965 年、当時大学院生だった Buchberger により考案されたグレブナー基底算法 [1] は、現代の計算代数における最も重要な算法であることに異論はないだろう。当初は多変数多項式環とそのイデアルでの計算が主な対象だったが、現在は微分環や微分イデアルなどにも拡張され、応用分野は代数幾何や幾何学、統計学などに広がり、とどまるところを知らないほどである。一方、その算法はといえば、不必要な S 多項式の生成を避ける工夫や数係数の巨大化に対処するモジュラー法、さらに計算量が少ない項順序の発見などの副次的な進歩はあったが、算法の骨格は半世紀以上も不変である。それほど算法は単純明快で、変えようがないとも言えよう。

[*1] 〒 305-8571 茨城県つくば市天王台 1-1-1　E-mail: sasaki@math.tsukuba.ac.jp

本稿では、項順序を辞書式 (LEX と略記) に限定したグレブナー基底を扱う。この場合、グレブナー基底計算は変数を高位のものから順に消去するものとなる。変数消去の観点から言えば、グレブナー基底より遥かに昔から多くの消去算法が考案されたが、そのほとんどが多変数多項式を主変数表示する。即ち、x を主変数、$(\boldsymbol{u}) = (u_1, \ldots, u_n)$ を従変数の組とするとき、数体 $\mathbb{K}$ 上の多変数多項式 $F(x, \boldsymbol{u}) \in \mathbb{K}[x, \boldsymbol{u}]$ を $F = f_d(\boldsymbol{u})x^d + f_{d-1}(\boldsymbol{u})x^{d-1} + \cdots + f_0(\boldsymbol{u})$ と表す。そして、$G = g_e(\boldsymbol{u})x^e + \cdots + g_0(\boldsymbol{u})$, $d \geq e$, と表される多項式 G との演算 $\mathrm{ltmElim}(F, G) \stackrel{\mathrm{def}}{=} g_e F - f_d\, x^{d-e} G$ により、F の主項 (最高次項) $f_d\, x^d$ が消去される。これを F の G による**主項消去**という。一方、グレブナー基底算法では、扱う多項式環の単項式全体を適当な順序 $\succ$ (本稿では LEX 順序) により (数係数を除き) 一意的に順序づけ、多変数多項式を単項式の和で表す。即ち、上記多項式 F を $F = c_1 M_1 + c_2 M_2 + \cdots + c_l M_l$ と表す。ここで、各 $c_i \in \mathbb{K}$、$M_i = x^{d_i}\, u_1^{e_{i_1}} \cdots u_n^{e_{i_n}}$ であり、$M_1 \succ M_2 \succ \cdots \succ M_l$ である。そして、F と $G := c'_1 M'_1 + c'_2 M'_2 + \cdots + c'_{l'} M'_{l'}$ とから **S 多項式**を次の算式で生成する：$\mathrm{Spol}(F, G) \stackrel{\mathrm{def}}{=} \frac{L}{c_1 M_1} F - \frac{L}{c'_1 M'_1} G,\ L = \mathrm{lcm}(M_1, M'_1)$ (lcm は最小公倍子)。$\mathrm{Spol}(F, G)$ は、F と G にそれぞれ最小単項式を掛けて両者の先頭単項をキャンセルさせる操作であり、先頭単項消去、略して**単項消去**という。M_1 が M'_1 と等しいか倍数のときは、$\mathrm{Spol}(F, G)$ は F の先頭単項を G により消去する演算となる。これを F の G による **M 簡約**といい、$F \xrightarrow{G}$ と表示する。また、集合 $\mathcal{G} = \{G_1, \ldots, G_k\}$ の要素全体により F を可能な限り M 簡約することを $F \xrightarrow{\mathcal{G}}\!\!\!\!\!\rightarrow$ と表示する。下記では $\mathcal{G}$ のグレブナー基底を $\mathrm{GB}(\mathcal{G})$ と表し (基底要素は通常、$\mathcal{G}$ の元 $G_1, \ldots, G_k$ とは全く異なる)、その基底のどの要素も他の要素で M 簡約不可能なとき $\mathrm{GB}(\mathcal{G})$ を**簡約基底 (reduced basis)** という。

二つの初等的消去法を詳しく説明したのは、両者が算法的にも結果的にも本質的に異なっているからである。以下では与えられた多項式系を $\mathcal{F} = \{F_1, \ldots, F_m, F_{m+1}\} \subset \mathbb{K}[\boldsymbol{x}, \boldsymbol{u}]$, とし、$x_1 \succ \cdots \succ x_m$ とする。$\boldsymbol{x}$ を主変数と呼び、$x_1, \ldots, x_m$ を順に消去して $\boldsymbol{u}$ の多項式を得るものとする。

単項消去に基ずく Buchberger 算法は、F と G の先頭項を消去して最も低い順位の S 多項式を生成するのみならず、その S 多項式を他の多項式で M 簡約して生き残った多項式のあらゆる対に対して、S 多項式を生成し続ける。この “最も低順位の多項式” と “あらゆる対に対して” という二つの極限性のため、$\mathcal{F}$ の (LEX-順序での) グレブナー基底 $\mathrm{GB}(\mathcal{F})$ の最小元がイデアル $\langle F_1, \ldots, F_{m+1} \rangle$ の最小元となる。その代償として、計算に多大な時間がかかる (変数の個数に関して 2 重指数的)。主項消去では、F を係数ベクトル $(f_d, f_{d-1}, \ldots, f_0)$ に対応づけることで、消去結果を行列式で表現できる。実際、$F, G \in \mathbb{K}[x, \boldsymbol{u}]$ に対しては、**擬剰余** $\mathrm{Prem}(F, G) \stackrel{\mathrm{def}}{=} \mathrm{remainder}(g_e^{d-e+1} F, G)$ は $\mathbb{K}[x, \boldsymbol{u}]$ に含まれ、$\deg_x(\mathrm{Prem}(F, G)) < e$ であり、F と $x^{d-e} G, \ldots, x^0 G$ の係数ベクトルを行とする行列式で表される。F と G が互いに素なとき、F と G から x をすべて消去すれば $\boldsymbol{u}$ の多項式が計算できる。これを x に関する**終結式**といい $\mathrm{res}_x(F, G)$ と表す。終結式を F と G の係数ベクトルで表すのが有名な Sylvester 行列式である。F と G を出発多項式として、擬剰余演算で**多項式剰余列** $\mathrm{PRS}(F, G) \stackrel{\mathrm{def}}{=} (P_1 = F, P_2 = G, \ldots, P_{i+1} = \mathrm{Prem}(P_{i-1}, P_i)/\beta_i, \ldots)$ を計算することも当然研究された。ここで、$\beta_i = 1$ とすれば P_{i+1} は一般に P_j $(j \leq i-1)$ の主係数のべき乗を含むので、それら余計な因子を β_i で除去するのである。多項式剰余列でも、剰余 P_{i+1} は F と G の係数ベクトルを行とする行列式で表現され、その理論は部分終結式理論と呼ばれている。

主項消去に基づく変数消去は、上述のような美事な理論化にも拘らず、致命的な欠陥を抱えている：ごく簡単な場合を除きイデアルの最小元が計算できることは稀で、終結式は大抵**余計因子** (**extraneous factor**; $\mathcal{F}$ の零点に対応しない零点をもつ因子) を含み、終結式の大部分が余計因子であることも頻繁に発生する。実際、Sylvester 行列式の零点は F と G の共通零点以外に $f_d(\boldsymbol{u})$ と $g_e(\boldsymbol{u})$ の共通零点を含む。F と G が x に関して疎な場合、Sylvester 行列式の因子は多重化することが多い。また、擬除算では F に g_e を不必要に多く掛けることが多いので、必要最小限だけ掛ける **spsPrem** に置き換えても、やはり余計因子が生じる。しからばと、上記 ltmElim 演算において、乗数を最小化した $\mathbf{critElim}(F.G) \stackrel{\mathrm{def}}{=} (g_e/\gamma)\, F - (f_d/\gamma) x^{d-e} G$, $\gamma = \gcd(f_d, g_e)$、も試してみたが、やはり余計因子が発生する。

計算量解析が示すように、主項消去は単項消去よりも圧倒的に高速である。そこで、多くの研究者が主項

消去により、例えばイデアルの最小元を高速に計算しようと努力したはずである。しかし、筆者は成功したという話を聞いたことがない。1990 年代には**疎終結式**の研究が世界的に流行したが、今では見る影もない。“主項消去と単項消去を融合した変数消去法” は『見果てぬ夢』というのが現況だが、3 年前、互いに素な多項式 $F, G \in \mathbb{K}[x, \boldsymbol{u}]$ に対しては夢でないことが示された [6]。鍵は終結式の**余因子** (別名は**ベズー係数**) である。終結式 $R \stackrel{\text{def}}{=} \mathrm{res}_x(F, G) \in \mathbb{K}[\boldsymbol{u}]$ に対して $R = AF + BG$ を満たす $A, B \in \mathbb{K}[x, \boldsymbol{u}]$ が存在する ; A, B が R の余因子である。条件 $\deg_x(A) < e$, $\deg_x(B) < d$ を満たす A, B は一意に定まる。論文 [6] では、A, B が次数条件を満たすなら、$R/\gcd(\mathrm{cont}_x(A), \mathrm{cont}_x(B))$ はイデアル $\langle F, G\rangle$ の最小元の定数倍である が証明されたのである ($\mathrm{cont}_x(A)$ は多項式 A の主変数 x に関する係数たちの GCD)。

最も簡単な場合とは言え、一点でも突破できれば、後は押せ押せである。3 個以上の多項式からなる系 $\mathcal{F}$ に対しては、多項式系 $\{F, G\}$ に対する上記のような関係式は得られなかったが、多くの系に成立するだろう 3 条件を満たす**健康な系** (詳細は 2 章で説明する) が鍵である [7]。健康な系では、主変数 $\boldsymbol{x}$ は全て消去され、従変数は一つも消去されず (これらが系が健康であるための三条件の二つ)、グレブナー基底 $\mathrm{GB}(\mathcal{F})$ は $\mathrm{GB}(\mathcal{F}) \cap \mathbb{K}[\boldsymbol{u}] = \{\widehat{S}(\boldsymbol{u})\}$ を満たす ことが証明できる。この関係式はイデアル $\langle\mathcal{F}\rangle$ の最小元 $\widehat{S}$ の算法を与えるものではないが、主変数消去を異なる複数のルートで計算すれば、消去結果は全て $\widehat{S}$ の倍数なので、それらの GCD は $\widehat{S}$ のより小さな倍数であるはずだ。その GCD が $\widehat{S}$ の大きな倍数か小さな倍数かは未知だが、幾つかの例題によれば非常に小さな倍数だった (換言すれば余計因子が小さい)。さらに、余因子を利用して余計因子を (不完全だが) 除去する方法も考案された。そして、簡単だが toy ではない例でテストしたところ、イデアルの最小元を十分高速に計算できた。

これらに気をよくして、本稿のタイトルの前半部 “剰余列と GCD による辞書式グレブナー基底計算” に挑戦したのが [5] である。第一段落で述べたように、イデアルの最小元を計算するのに Buchberger 算法は不可欠だろう。とすれば、主項消去演算でできることは、グレブナー基底の (特に順位の低い) 元そのもの、あるいはその小さい倍数を主項消去と GCD 演算で計算し、初与の系に追加することで、Buchberger 算法を効率化することくらいだろう。例題による実験の結果、その方策が有望だとの感触を得たが、グレブナー基底の第二最小元の計算で、計算途中の数係数と多項式が予想を越えて膨張し、この中間式膨張の抑止が不可欠との認識に至った。第 2 章で、[6, 7] で得られた定理と証明を簡潔に説明したあと、[5] の方法を簡単に復習し、そのあとの章で中間式膨張の解決策を記述する。

2　これまでの研究の簡単な復習

2.1　主要 2 定理とその証明の概略

論文 [6] では、互いに素な $F, G \in \mathbb{K}[x, \boldsymbol{u}]$ のなすイデアル $\langle F, G\rangle$ の最小元 $\widehat{S} \in \mathbb{K}[\boldsymbol{u}]$ が扱われた。F と G の x に関する終結式を $R = \mathrm{res}_x(F, G)$ とすれば、終結式論より R は $\widehat{S}$ の倍数である。R の余因子 A と B は次数条件 $\deg_x(A) < \deg_x(G)$, $\deg_x(B) < \deg_x(F)$ を満たすとする。

定理 1 (T.S and D.Inaba [6])
Let $F, G \in \mathbb{K}[x, \boldsymbol{u}]$ be relatively prime, R be $\mathrm{res}_x(F, G)$, and A, B be cofactors of R, satisfying the degree conditions. Let $\widehat{S} \in \mathbb{K}[\boldsymbol{u}]$ be the lowest-order element of $\langle F, G\rangle$ (the leading coefficient is normalized to 1). Then, $\widehat{S}$ is a constant multiple of $R/\gcd(\mathrm{cont}_x(A), \mathrm{cont}_x(B))$.

証明の概要　$\mathrm{GB}(F, G)$ は LEX 順序の簡約グレブナー基底とする。終結式論より、$\mathrm{res}_x(G, H)$ は $\widehat{S}$ の倍数である。LEX 順序の Buchberger 算法を $\langle F, G\rangle$ に適用して得られた $\widehat{S}$ の余因子を $\widetilde{A}, \widetilde{B}$ とする (Buchberger 算法も余因子を計算できるのだ [2])：$\widehat{S} = \widetilde{A}F + \widetilde{B}G$。$\widetilde{A}, \widetilde{B}$ は一般に次数条件を満たさず、高次になることが普通である。そこで $\widetilde{A}' \stackrel{\text{def}}{=} \mathrm{rem}(\widetilde{A}, G)$, $\widetilde{B}' \stackrel{\text{def}}{=} \mathrm{rem}(\widetilde{B}, F)$ (rem は剰余演算) を考える。商部分は打ち消し

あって $\widehat{S} = \widetilde{A}'F + \widetilde{B}'G$ を得るので、$\widetilde{A}', \widetilde{B}' \in \mathbb{K}[x, \boldsymbol{u}]$ ならば、$\widehat{S}$ の最小性より $\gcd(\widetilde{A}', \widetilde{B}') = 1$ なので、次数条件を満たす余因子の一意性より (A, B) は $(\widetilde{A}', \widetilde{B}')$ の倍数となり、定理が得られる。

問題は、$\widetilde{A}', \widetilde{B}'$ が多項式になるか否かである。F と G の主係数 f_d, g_e に対し $\gamma \stackrel{\text{def}}{=} \gcd(f_d, g_e)$ とする。$\gamma = 1$ ならば $\widetilde{A}', \widetilde{B}' \in \mathbb{K}[x, \boldsymbol{u}]$ となることは容易に示せる。γ が定数でない場合、多項式剰余列を計算してみれば、i 番目の剰余 P_i に対する余因子を A_i, B_i とするとき、i が増加するとともに γ の因子が A_i と B_i の主係数に移動していくことがわかる。実は同じことがグレブナー基底計算でも起きる。そして、$\widetilde{A}', \widetilde{B}'$ の次数が e, d をそれぞれ超えたとき、γ がすべて $\widetilde{A}', \widetilde{B}'$ の高次項に移っているのである。証明はやや技巧的なので、詳細は原論文を参照されたい。 □

論文 [7] では健康な系の定義と、健康な系に成立する定理 2 が重要である。我々は高校時代にナイーブに 2 多項式系では変数が 1 個消去でき、3 多項式系では 2 個消去できると思ってきた。それを少し拡張して、系 $\mathcal{F}$ が、条件 A) m 個の主変数は全て消去でき、条件 B) 従変数はどれも消去できず、さらに条件 C) $\mathcal{F}$ のグレブナー基底は $\mathcal{G}_1 \cap \mathcal{G}_2 = \emptyset$ を満たす非零なグレブナー基底 $\mathcal{G}_1$ と $\mathcal{G}_2$ の集合和ではない、を全て満たすとき **$\mathcal{F}$ は健康 (healthy) であると**定義する。条件 C) に反する系も存在する (たとえば、$\mathcal{G}_1$ と $\mathcal{G}_2$ に共通に含まれる主変数がない場合) が、そんな系は $\mathcal{G}_1$ と $\mathcal{G}_2$ を別に扱うべきである。(**注釈** 条件 C) に対応する [7] の Class-C は less specific で、[5] のそれは $\mathcal{G}_1$ と $\mathcal{G}_2$ がグレブナー基底との制限が欠けている)。

定理 2 (T.S and D.Inaba [7])
Let $\mathrm{GB}(\mathcal{F})$ *be the reduced LEX-order Gröbner basis of* $\mathcal{F}$*. If* $\mathcal{F}$ *is healthy then* $\mathrm{GB}(\mathcal{F}) \cap \mathbb{K}[\boldsymbol{u}] = \{\widehat{S}(\boldsymbol{u})\}$.

証明の概要 $\mathrm{GB}(\mathcal{F}) \cap \mathbb{K}[\boldsymbol{u}] = \{S_1(\boldsymbol{u}), \ldots, S_l(\boldsymbol{u})\}$ とする。条件 A,B より右辺は零集合ではなく、条件 C より右辺が二つのグレブナー基底の和集合になることもないので、条件 B より各 S_i は従変数を全て含む。$l = 1$ なら定理は自明なので $l \geq 2$ とし、$S_1 \succ \cdots \succ S_l$ と仮定する。証明には主項消去演算を利用する。

$G = \gcd(S_{l-1}, S_l)$、$R = \mathrm{res}_{u_1}(S_{l-1}/G,\, S_l/G)$ とすれば、$R \in \mathbb{K}[u_2, \ldots, u_n]$、$RG \in G \times \langle S_{l-1}/G,\, S_l/G \rangle \subset \mathrm{GB}(\mathcal{F}) \cap \mathbb{K}[\boldsymbol{u}]$ である。$\deg_{u_1}(G) = 0$ なら $RG \in \mathbb{K}[u_2, \ldots, u_n]$ となり、$0 < \deg_{u_1}(G) < \deg_{u_1}(S_l)$ なら $\deg_{u_1}(RG) < \deg_{u_1}(S_l)$ となり、いずれも S_l の最小性に反する。$\deg_{u_1}(G) = \deg_{u_1}(S_l)$ なら $S_{l-1} = C_{l-1}G$, $S_l = c_l G$, $c_l \in \mathbb{K}[u_2, .., u_n]$, $\gcd(C_{l-1}, c_l) = 1$, と表せる。$c_l \in \mathbb{K}$ ならば $S_{l-1} \xrightarrow{S_l} 0$ となり、$\mathrm{GB}(\mathcal{F})$ の簡約性に反する。$c_l \notin \mathbb{K}$ の場合は、$\deg_{u_1}(C_{l-1}) = 0$ なら $\gcd(C_{l-1}, c_l) = 1$ ゆえ S_{l-1} の主項が消去でき、$\deg_{u_1}(C_{l-1}) > 0$ なら $g = \gcd(\mathrm{lc}_{u_1}(C_{l-1}), c_l)$ を考えれば、上記と同じ論法により C_{l-1} の主項が消去できるので、いずれも $\mathrm{GB}(\mathcal{F})$ の簡約性に反する。以上より、$l = 1$ でなければならない。 □

2.2 剰余列と GCD による Buchberger 算法効率化の復習

算法の核となるアイデアは二つある。第一は、変数消去を従来法である与系 $\mathcal{F}$ の “三角化” から “四角化” に変更することであり、第二は、“主係数イデアル” の最小元も計算することである。第一のアイデアは [7] で、第二のアイデアは [5] で提案された。

従来、$\mathcal{F}$ の主変数消去は、まず F_1 を消去子として $G_i := \mathrm{lastPRS}_{x_1}(F_1, F_i)$, $2 \leq i \leq m{+}1$, を計算する：$\mathrm{lastPRS}_x$ は引数多項式の変数 x に関する疎剰余列の最終元を意味し、$G_i \in \mathbb{K}[x_2, .., \boldsymbol{u}]$ である。次には、G_2 を消去子として $3 \leq j \leq m{+}1$ に対し $H_j := \mathrm{lastPRS}_{x_2}(G_2, G_j)$ を計算する。以下、同様に $x_3, .., x_m$ を消去すれば、主変数を一つ消去する度に生成される多項式が一つ減少し、同じ主変数を含む多項式たちを横方向に、変数消去が進むにつれ下方向に並べれば逆三角形になる。四角化では、変数消去が進行しても生成される多項式の個数が減少しないように、たとえば x_1 の消去は $G_i = \mathrm{lastPRS}_{x_1}(F_i, F_{i+1})$, $1 \leq i \leq m{+}1$, ただし $F_{m+2} = F_1$、と行う。この場合、同じ主変数を含む多項式たちを横に並べれば四角形になるので、この変数消去法を**四角化**と命名し、得られる $m{+}1$ 個の剰余列を **rectangular PRSs**(略して **rectPRSs**)

と呼ぶ。なお、たとえば $\gcd(G_1,G_2)\notin\mathbb{K}$ であれば x_2 を消去後の多項式は m 個に減るが、そんな場合は別扱いが必要である。

1 章の最終段落で、“新算法ではグレブナー基底の要素に近い多項式を与系 $\mathcal{F}$ に追加する” と書いたが、追加多項式は rectPRSs の途中元の組から生成する (rectPRSs の最終元はイデアルの最小元の計算に使う)。rectPRSs の途中元をグレブナー基底と比べると、基底の各元の主変数とその次数に関しては rectPRSs に対応する途中元が存在するが、その途中元の主項では第二主変数が不要であることがわかる。したがって、与系に追加する多項式は、途中元の主係数の組に対し変数消去を行って生成すればよい (下記段落を参照)。この “主係数の組” の成すイデアルが**主係数イデアル**であり、主係数イデアルの主変数 (それは $\mathcal{F}$ の従変数のこともある) の消去には、$\mathcal{F}$ の主変数消去算法を再帰的に使用する。

対象とする途中元の組を $\{R_1,\ldots,R_\ell\}\subset\mathbb{K}[y,z,u,..]$, $\ell\geq 2$, とし、$R_j=y^dC_j+D_j$, $d\geq 1$, $C_j\in\mathbb{K}[z,u,..]$ とする (変数の用法をこの段落でのみ変更した)。ここで、$\{C_1,\ldots,C_\ell\}\subset\mathbb{K}[z,u,..]$ が主係数の組で z が消去すべき変数である。$c_j:=\mathrm{lastPRS}_z(C_j,C_{j+1})$ とし ($C_{\ell+1}=C_1$)、c_j の余因子を a_j,b_j とする：$c_j=a_jC_j+b_jC_{j+1}$。各元 R_j はイデアル $\langle\mathcal{F}\rangle$ の要素だが、c_j はそうではない。そこで y^dc_j を主項とする $\langle\mathcal{F}\rangle$ の多項式を作る必要がある。それは簡単で $W_j\stackrel{\mathrm{def}}{=}a_jR_j+b_jR_{j+1}$ とすれば良い。W_j は**主係数 c_j から全身を復元した多項式**という意味で $\mathrm{LCtoWhole}(c_j)$、略して $\mathrm{LCtoW}(c_j)$ と表す。

2.3 与系 $\mathcal{F}$ の具体例における $\mathrm{GB}(\mathcal{F})$ の第二最小元の計算

前節の四角化は、三角化に比べ変数消去が非経済的だが、GCD だけで最小元 $\widehat{S}$(の小倍数) を計算できる卓抜なアイデアと自負するし、主係数イデアルの最小元 (の小倍数) を基に $\mathrm{GB}(\mathcal{F})$ の要素に近い多項式を生成するアイデアも卓抜だと思う。これらで当初の目的はほぼ達成されると思った。だが、具体例で試してみると全く予想しなかったことが起きた。以下、そのことを具体例で簡単に示す。

例 1 $\mathcal{F}_1:=\{F_1,F_2,F_3\}$: x,y が主変数で $x\succ y$、かつ u,w が従変数で、各 F_i は下記である。

$$\left\{\begin{array}{rcl} F_1 &=& x^4{\cdot}(y{+}u)+x^2{\cdot}(y{-}2w)+(2u{+}w),\\ F_2 &=& x^4{\cdot}(\,y\,u\,)\;+x^2{\cdot}(y{+}2w)+(3u{-}w).\\ F_3 &=& x^4{\cdot}(y{-}u)+x^2{\cdot}(2y{+}u)+(u{-}2w),\end{array}\right.\tag{1}$$

$\mathrm{GB}(\mathcal{F}_1)$ の最小元は $G_{10}=33\,u^7+23\,u^6w-126\,u^6+(14\text{ monomials})+144\,uw^4-256\,uw^3-32\,w^4$ で、系 $\{F_1,F_2,F_3\}$ から四角化で得られる 3 個の終結式を $\widetilde{S}_1,\widetilde{S}_2,\widetilde{S}_3\in\mathbb{Z}[u,w]$ (項数はそれぞれ 112, 98, 98) とすると、$\gcd(\widetilde{S}_1,\widetilde{S}_2,\widetilde{S}_3)=u^2\widehat{S}$ となる。$\widetilde{S}_1,\widetilde{S}_2,\widetilde{S}_3$ は異なる三通りの三角化で得られる終結式で、終結式が大きな余計因子を含み得ることを如実に示している。第二最小元は項数が 60, 係数もほどほどの多項式で、$G_9=y\times(48000\,w^8-419640\,w^7-\cdots-1041048\,w^2)+(6500\,u^6w^5-430980\,u^6w^4-\cdots-5430496\,w^3)$ である。しかし、以下にみるように、新算法による G_9 の計算は大きな中間式膨張を起こす。 //

$\mathcal{F}_1$ の rectPRSs のうち、$\mathcal{F}_1$ から主変数 x,y を全て消去する直前の剰余を $\widetilde{R}_1,\widetilde{R}_2,\widetilde{R}_3$ とする。G_9 はこれら剰余の係数イデアルから計算される。各剰余は $\widetilde{R}_j=y\,C_j+D_j$, $C_j,D_j\in\mathbb{Z}[u,w]$ の形の多項式だが、u に関する次数は $\deg_u(\widetilde{R}_1)=14$, $\deg_u(\widetilde{R}_2)=\deg_u(\widetilde{R}_3)=12$ である。そのため、u に関する剰余列計算が非常に重くなる。そこで、各剰余を G_{10} で M 簡約した：$\widetilde{R}_j\xrightarrow{G_{10}}\!\!\!\!\!\rightarrow\ \widetilde{R}'_j$。その結果、たとえば $\widetilde{R}'_1$ は

$$\widetilde{R}'_1\ =\left\{\begin{array}{l} y\times(-\,349136896959\,u^6w^8-3368123780598\,u^6w^7+\cdots+249988316347584\,w^4)\\ \quad+\,(-\,915846376989\,u^6w^8+1397183224758\,u^6w^7+\cdots-417398434490880\,w^4),\end{array}\right.\tag{2}$$

と低次になるが多項式自体は大きい (y^1-項も y^0-項も項数は 60)。実験では、Mathematica による $\mathrm{GB}(\mathcal{F}_1)$ の計算が 21 秒、新算法による G_{10} の計算が 102 ミリ秒 (いずれも筆者の古いパソコンを使用) なのに、$\mathrm{PRS}_u(C'_1,C'_2)$, $\mathrm{PRS}_u(C'_2,C'_3)$, $\mathrm{PRS}_u(C'_3,C'_1)$ $(C'_j:=\mathrm{lc}_y(\widetilde{R}'_j))$ の計算 (終結式+余因子) だけでそれぞれ

1104 ミリ秒、789 ミリ秒、1127 ミリ秒を要した。これでは新算法が有効とは到底言えない。なお、剰余列計算後の G_9 に至る計算 (次章の先頭段落を参照) は、剰余列計算の約 1/10 ほどの時間で終了した。

2.4 疎な多変数多項式剰余列の算法

以上のことから、算法効率化の鍵は多項式剰余列 (余因子も含む) 計算の効率化である。応用面で現れる多変数多項式の大部分は疎であり、本稿で扱う多項式も疎であるとする。まず、**疎多項式剰余列 (spsPRS)** の算法を明確にしておく。互いに素な多項式 $F, G \in \mathbb{K}[x, \boldsymbol{u}]$, $\deg_x(F) \geq \deg_x(G)$, から出発する剰余列と余因子列は、初期多項式の組 $(P_1, A_1, B_1) := (F, 1, 0)$, $(P_2, A_2, B_2) := (G, 0, 1)$ から、$i = 2, 3, \ldots, k-1$ に対し $(P_{i+1}, A_{i+1}, B_{i+1}) := (\ \mathrm{rem}(\alpha_i P_{i-1}, P_i)/\beta_i,\ \mathrm{rem}(\alpha_i A_{i-1}, A_i)/\beta_i,\ \mathrm{rem}(\alpha_i B_{i-1}, B_i)/\beta_i\ )$ なる算式で逐次的に計算される。ここで、α_i は P_i の主係数 $\mathrm{lc}_x(P_i)$ のべき乗で、剰余 $\mathrm{rem}(\alpha_i P_{i-1}, P_i)$ が $\boldsymbol{u}$ に関しても多項式になるように決めるが、そうすると剰余は一般に $\mathrm{lc}_x(P_{j<i})$ のべき乗を因子に持つので、それを β_i で除去するのである。部分終結式理論では α_i は擬除算で決まり β_i も理論的に決定できる。x に関して疎な多項式に対しても部分終結式理論は使えるが、中間式がかなり膨張する。疎な多項式に対しては、次に示す **疎擬剰余 (spsPrem)** で主項消去を行うと、$\alpha_i = \mathrm{lc}_x(P_i)^{\mu_i}$ で μ_i が最小に決まる。

```
Proc  spsPrem((P_{i-1}, A_{i-1}, B_{i-1}), (P_i, A_i, B_i)) ==
(1)   c_j := lc(P_j); d_j := deg(P_j)  (j ∈ {i-1, i});
(2)   while d := d_{i-1} - d_i ≥ 0 do
(3)       (P_{i-1}, A_{i-1}, B_{i-1}) := c_i (P_{i-1}, A_{i-1}, B_{i-1})
                                       - c_{i-1} x^d (P_i, A_i, B_i);
(4)       c_{i-1} := lc(P_{i-1}); d_{i-1} := deg(P_{i-1}); enddo;
(5)   return (P_{i+1}, A_{i+1}, B_{i+1}) := (P_{i-1}, A_{i-1}, B_{i-1}).
```

なお、上記の μ_i の値は F と G の各項の現れ方に依存する。そのため、β_i を定める公式は発見されてないが、$\beta_i = \prod_{j=2}^{i-1} \mathrm{lc}_x(P_j)^{\nu_j}$ の形であることは分っている [4]。よって、$P'_{i+1} := \mathrm{rem}(\alpha_i P_{i-1}, P_i)$ とするとき、下記の Hearn の **試し除算法 (trial division method)**[3] により、β_i を効率よく除去できる。

$$\textbf{for } j = i-1 \Rightarrow i-2 \Rightarrow \cdots \Rightarrow 2 \textbf{ do while } \mathrm{lc}_x(P_j) \text{ divides } P'_{i+1} \textbf{ then } P'_{i+1} := P'_{i+1}/\mathrm{lc}_x(P_j).$$

spsPRS で余因子も生成する場合には、最終元 P_k は $\gcd(\mathrm{cont}_x(A_k), \mathrm{cont}_x(B_k))$ で割って規格化する。

3 $\mathrm{GB}(\mathcal{F})$ の第二最小元 $\widehat{R}$ の計算の効率化

例 1 で示したように、$\mathrm{GB}(\mathcal{F})$ の第二最小元 $\widehat{R}$ は主変数を全て消去する直前の rectPRSs の剰余 $\widetilde{R}_1, \ldots, \widetilde{R}_\ell$ (これらは上記のように $\widehat{S}$ で M 簡約されるだろう) から計算される。ここで、$\ell \geq 2$ と仮定し、各 $\widetilde{R}_j$ は $\widetilde{R}_j = x_m^e C_j + D_j$ $(C_j, D_j \in \mathbb{K}[\boldsymbol{u}])$ と表せる (e は大抵 1 である) が、主係数系 $\{C_1, \ldots, C_\ell\}$ は健康とは限らない。2.2 節と同様、$c_j := \mathrm{lastPRS}_u(C_j, C_{j+1}) = a_j C_j + b_j C_{j+1}$ $(C_{\ell+1} = C_1)$, $W_j := \mathrm{LCtoW}(c_j) = a_j \widetilde{R}_j + b_j \widetilde{R}_{j+1}$ とし、$\widehat{R}$ から $\widehat{S}$ の倍数の不定性を除くため、$W_j \xrightarrow{\widehat{S}} W'_j$ とする。

問題は、各 W'_j に含まれる余計因子をどう除去するかである。例 1 においては、たとえば W_1 は数係数が約 100 桁で項数が 1016 の多項式で、G_9 に比べて項数が 16 倍以上も大きく、W'_1 も項数が 557 の多項式である。このように余計因子が大部分を占める多項式に対しては、G_{10} に使用した方法とは別の除去方法を探求すべきと思った。まず考えたことは、各 W'_j から取り除ける因子を全て除去したらどうなるか である。即ち、$\widetilde{W}_j \stackrel{\mathrm{def}}{=} W'_j/\mathrm{cont}_{x_m}(W'_j)$ を計算してみようと。$\mathrm{cont}_{x_m}(W'_j)$ は第二最小元 $\widehat{R}$ の因子 $\tilde{r}$ を含むかも知れず、その場合、$\tilde{r}$ を $\widehat{R}$ の **不足因子** とよぶ。例 1 の $\mathcal{F}_1$ に対して $\widetilde{W}_j$ を実際に計算して驚いた: $\widetilde{W}_1 = \widetilde{W}_2 = \widetilde{W}_3 = G_9$

を得たのである。こうなった理由は何か？ こんな事は一般に成立するのか？ の究明は今後の課題として、まずは最終的に $\widetilde{W}_j$ を如何に高速に計算するかを考えた。これ以降の文章が本稿の主要部である。

3.1 数係数の膨張対策：素数を法とするモジュラー算法

グレブナー基底計算が通常、猛烈な数係数膨張を起こすのは周知の事実であり、抑止対策として素数 p を法とする**モジュラー算法**が有効なこともよく知られている。本稿で考えるのは多項式剰余列と余因子計算における係数膨張であり、通常、これは大したことはないと考えられている。だが、例 1 の大きくはない系 $\mathcal{F}_1$ でも、G_9 計算における剰余列の最終元とその余因子の数係数が約 100 桁というのは尋常ではない。そこでモジュラー算法を使うのだが、法は剰余列と余因子だけをみて選んではならず、最終目的の $\widetilde{W}_j$ をみて選ぶべきである。例えば例 1 の剰余列計算では中間式の最大係数は数百桁で、最終的に 100 桁ほどの数係数を計算するためにモジュラー算法を使用しても、全くと言ってよいほど意味はない。しかし、目的とする $\widetilde{W}_j$ の数係数は高々10 桁なので、法はそれより少し大きければ十分なはずだ。このことは、モジュラー算法の有効性と共にプログラミングの複雑さも示唆する。

実際の法は、まず半語長あるいは 1 語長ほどの素数 p を選び、$p \Rightarrow p^2 \Rightarrow \cdots \Rightarrow p^{\lambda+1}$ と、必要最低限な大きさまで上げていく。各 $j \in \{1,\ldots,\ell\}$, $l = 1,2,\ldots$ に対し、$c_j^{(l)} := \mathrm{lastPRS}_{u_1}(C_j, C_{j+1}) \pmod{p^{l+1}}$ とし、$\overline{c}^{(l)} := \gcd(c_1^{(l)},\ldots,c_\ell^{(l)}) \pmod{p^{l+1}}$ とする。このとき、$q := p^{\lambda+1}$ は下記の条件で決定する。

$$q := p^{\lambda+1} \iff \begin{cases} \overline{c}^{(l)} \not\equiv \overline{c}^{(l+1)} \pmod{p^{l+2}} & \forall l < \lambda, \\ \overline{c}^{(\lambda)} \equiv \overline{c}^{(\lambda+1)} \pmod{p^{\lambda+2}}. \end{cases} \tag{3}$$

p を法とするモジュラー算法では、多項式の GCD や係数部の GCD(すなわち content) が定数倍の不定性を持つ。したがって、結果の GCD は、**数値主係数** (先頭単項の係数) を 1 に規格化する。最終的には数値主係数も決定する必要があるが、それには二つの近い素数 p' と p'' を用いて計算すればよい (もっと簡単な方法があるかも知れないが、筆者は思いつかない)。以下では $q' := (p')^{\lambda+1}$, $q'' := (p'')^{\lambda+1}$ とする。

$\overline{c} = \gamma_1 M_1 + \gamma_2 M_2 + \cdots + \gamma_s M_s$ とする：ここで、各 $\gamma_i \in \mathbb{Z}$, M_i は従変数 $u_2,\ldots,u_n$ の単項式である。$\overline{c}$ を q' と q'' を法として計算した結果をそれぞれ $\overline{c}', \overline{c}''$ として、次式で表す。

$$\begin{cases} \overline{c}' \equiv M_1 + \gamma_2' M_2 + \cdots + \gamma_s' M_s \pmod{q'}, \\ \overline{c}'' \equiv M_1 + \gamma_2'' M_2 + \cdots + \gamma_s'' M_s \pmod{q''}. \end{cases} \tag{4}$$

$\overline{c}'$ と $\overline{c}''$ の各係数 γ_i' と γ_i'' を、それぞれ q' と q'' を法として整数から有理数に変換する (**整数有理数変換**: integer-to-rational conversion、あるいは rational reconstruction と呼ばれる) と、次式を得る。

$$\begin{cases} \gamma_i' \equiv \gamma_i/\gamma_1 = \widetilde{\gamma}_i/\widetilde{\gamma}_{1:i} \pmod{q'}, \qquad \gamma_i'' \equiv \gamma_i/\gamma_1 = \widetilde{\gamma}_i/\widetilde{\gamma}_{1:i} \pmod{q''}. \\ \text{上式で、} \widetilde{\gamma}_i/\widetilde{\gamma}_{1:i} \text{が実際に計算される有理数で、} \gcd(\widetilde{\gamma}_i, \widetilde{\gamma}_{1:i}) = 1 \text{を満たす。} \end{cases} \tag{5}$$

このとき、$\overline{c}$ の数値主係数 γ_1 は $\gamma_1 = \mathrm{lcm}(\widetilde{\gamma}_{1:2},\ldots,\widetilde{\gamma}_{1:s})$ (lcm は最小公倍数) である (下記例 2 を参照)。

例 2 例 1 の系 $\mathcal{F}_1$ において規格化定数 γ_1 を決定してみよう。
$q' = 1073738843$ および $q'' = 1073738863$ のとき、$\overline{c}'$ と $\overline{c}''$ は次式のようになる (最低次項は w^{32})。なお、各 c_j は $c_j = \mathrm{lastPRS}_{u_1}(C_j, C_{j+1})$ と計算され、余因子で規格化されてない (余因子は未計算である)。

$$\begin{aligned} \overline{c}' &= w^{38} - 56371298\,w^{37} + 138243860\,w^{36} - 521121094\,w^{35} - 96457750\,w^{34} + \cdots, \\ \overline{c}'' &= w^{38} + 319437303\,w^{37} + 111400391\,w^{36} + 280603914\,w^{35} + 93236114\,w^{34} + \cdots. \end{aligned} \tag{6}$$

整数有理数変換は拡張互除法で実行できる。$(r_1, r_2) := (q', \gamma_i')$ および (q'', γ_i'') $(2 \le i \le s)$ から出発して、j-剰余 $r_j := \mathrm{rem}(r_{j-2}, r_{j-1})$ $(3 \le j \le k)$ とその余因子 a_j, b_j $(a_j r_1 + b_j r_2 = r_j)$ を計算する。すると、法

が大きければ、式 (5) より、どれかの j において、q' に対する (c_j, b_j) と q'' に対するそれが一致するので、その組を $(\widetilde{\gamma}_i, \widetilde{\gamma}_{1:i})$ とすればよい。そんな組が存在しなければ、法が小さすぎるのである。

第 2 係数 (w^{37} 項の係数) に対しては $\widetilde{\gamma}_2/\widetilde{\gamma}_{1:2} = -3497/400$ が得られた。同様に第 3、第 4、第 7 係数に対しては、それぞれ $-12829/800, 59176/375, -43377/2000$ が得られた。だが、第 5、第 6 係数に対しては得られなかった。実は第 5 係数は $1263623/6000$ に変換されるはずだが、$1263623 \times 6000 > q', q''$ であり、明らかに法が小さすぎる。幸運にも今の場合には、これらの法でも第 5 係数と第 6 係数を復元できる。そのトリックは以下のとおりである。復元できた係数の分母因子より、γ_1 は $\mathrm{lcm}(400, 800, 375, 2000) = 12000$ の倍数である。そこで、12000 を $\overline{c}'$ と $\overline{c}''$ に掛け、それぞれ q', q'' で法をとると両者は一致するのである。このことは γ_1 は 12000 でよいことを意味する。 //

素数を法とするモジュラー法の有効性を例 1 の $\mathcal{F}_1$ でチェックする。下記表 I の C_1, C_2, C_3 は、本章先頭で述べた $\widetilde{R}_1, \ldots, \widetilde{R}_\ell$ の主係数 $C_1 \ldots, C_\ell$ を、例 1 の対応する主係数 (u, w の多項式) で置き換えたものである。それらの剰余列と余因子を整数上で計算した計算時間を **spsPRS** 欄に、全く同じものをモジュラー法で計算した結果を **ModspsPRS** 欄に、余因子を計算せず剰余列だけをモジュラー法で計算した結果を (P_k**-only**) 欄に載せた。計算は、古いパソコン (Intel(R)-U2300 (1.20Ghz) 装備、OS は Linux 3.4.100) 上の国産数式処理システム GAL で実行した。

(C_{j_1}, C_{j_2})	**spsPRS**: P_k, A_k, B_k	**ModspsPRS**	(P_k**-only**)
(C_1, C_2)	1104 msec	587 msec	139 msec
(C_2, C_3)	789 msec	307 msec	82 msec
(C_3, C_1)	1127 msec	546 msec	112 msec

Table I : Efficiency comparison of **spsPRS** and **ModspsPRS**
The modulus used is $q = p = 1073738843$

上記の結果をみると、モジュラー法の効果はほんの少ししかないと言える。モジュラー法の実装に関しては、数値 $\gamma \in \mathbb{Z}_p$ の逆元を Lisp で書いた拡張互除法で実行するなど、高速化技法を全く取り込んでいない。いくらなんでも遅すぎるので、将来、高速化を検討すべきであろう。

3.2 中間数式の膨張対策：余因子の高次項だけを別計算

2.3 節の例 1 の下で、G_9 の計算では計算途中に、途中式の項数が G_9 の項数より遥かに増大する**数式膨張**が起きることを述べた。一方、前節までの計算によると、計算時間の大部分は剰余列法による余因子計算が占める。以下でみるように余因子は全項を計算する必要はなく、一部の上位次数項を計算すればよいので、この事実を利用して数式膨張を抑えることを目指す。

まず、多項式集合 $\{\widetilde{R}_1, \ldots, \widetilde{R}_\ell\} \subset \mathbb{Z}[x_m, \boldsymbol{u}]$ から $\widetilde{W}_1, \ldots, \widetilde{W}_\ell$ に至る計算過程を復習する。$\widetilde{R}_j = x_m^e C_j + D_j$ ($C_j, D_j \in \mathbb{Z}[\boldsymbol{u}]$) から出発し、$c_j := \mathrm{lastPRS}_{u_1}(C_j, C_{j+1})$ $(c_j \in \mathbb{Z}[u_2, .., u_n]) \Longrightarrow W_j := \mathrm{LCtoW}(c_j) = x_m^e c_j + [a_j D_j + b_j D_{j+1}] \Longrightarrow W_j \overset{\widehat{S}}{\twoheadrightarrow} W'_j = x_m^e c_j + [a_j D_j + b_j D_{j+1}]' \Longrightarrow \widetilde{W}_j = W'_j/\mathrm{cont}_{x_m}(W'_j)$, $\mathrm{cont}_{x_m}(W'_j) = \gcd(c_j, \mathrm{cont}_{u_1}([a_j D_j + b_j D_{j+1}]'))$ に到る。以上より分ることは、実際に扱う演算は、$\boldsymbol{u}$ の多項式の加減乗算、u_1 を主変数とする GCD 計算 (content 計算を含む) および割り切り除算のみである。GCD 計算も割り切り除算も、引数多項式の (u_1 の) 係数部の (適当な次数範囲の) 高次項あるいは低次項のみで実行できるが、$W_j \overset{\widehat{S}}{\longrightarrow\!\!\!\!\!\twoheadrightarrow} W'_j$ 計算では W_j の高次項が必要なので、以下では c_j の余因子 a_j, b_j の $(u_2, \ldots, u_n)$ に関する高次項を c_j とは別に計算することを考える。そのため、算式 $P'_{i+1} := \alpha_i P_{i-1} - Q_i P_i \Longrightarrow P_{i+1} := P'_{i+1}/\beta_i$ でまず c_j だけを計算し、その際、三つ組 (α_i, Q_i, β_i) を `alQbe-list` に蓄えておく。`alQbe-list` さえあれば、余因子は計算できることに注意されたい。

以下では実際の数式を示しつつ計算法を説明する。$\widetilde{R}_1$ は次式であり、どの項にも u が含まれる。

$$\widetilde{R}_1 \;=\; 1872\,yu^{14} - 834\,yu^{13}w - 11826\,yu^{13} + (110\text{ terms}) + 240\,u^2u^2w^8 + 1012\,u^2w^7 - 16\,uw^8.$$

$\widetilde{R}_1$ を $\widehat{S}$ で M 簡約した $\widetilde{R}'_1$ は 2.3 節の式 (2) で、変数 w だけの項を持つ。そのため、$C_1 = \mathrm{lc}_u(\widetilde{R}'_1)$、(素数 1073738843 を法として計算した) $c_1 = \mathrm{lastPRS}_u(C_1, C_2)$ と $\overline{c} = \gcd(c_1, c_2, c_3)$ は下記となる。

$$\begin{aligned} C_1 &= -\,349136896959\,u^6w^8 - 3368123780598\,u^6w^7 - 17823010042701\,u6w^6 + \cdots \\ &\quad + 125208661728\,w^{11} - 1386437797440\,w^{10} + \cdots + 249988316347584\,w^4, \qquad (7)\\ c_1 &= -\,493256166\,w^{79} - 473881285\,w^{78} + \cdots + 428336652\,w^{12} + +515855854\,w^{11}, \qquad (8)\\ \overline{c} &= w^{11} \times (w^6 - 56371298\,w^5 + \cdots - 382429906\,w - 247496825). \qquad (9) \end{aligned}$$

注釈：上記 (8) と (9) では、余因子 (a_j, b_j)(現時点では未計算) が計算されているとして、余因子で規格化した。規格化には余因子の最低次項のみ計算すればよく、それは下記と同様に容易である。

c_1 は項数が 69 だが $\gcd(c_1, c_2, c_3)$ の次数は 17 なので、余因子 a_j と b_j は w に関して上位次数項を 10 個ほど計算すればよいように思われる。その解析は次節で行なうとして、まず余因子 a_j と b_j の w に関する上位項または下位項を指定された範囲内で計算する方法の有効性を確認する。著者は GAL 上にプログラムを書き、Table I に与えた $(C_1, C_2), (C_2, C_3), (C_3, C_1)$ に対し、w の次数範囲を 15 として余因子 a_j と b_j の上位項を計算した。それら三つの計算時間はそれぞれ 31msec, 26msec, 28msec だった。結果は 良すぎると思われるかも知れないが、剰余列計算も余因子計算も計算途中で大きな数式の積を計算していることを考慮すれば、妥当だろう。なお、論文 [5] を執筆出版後、パソコンを 64bits 仕様版に更新したが、既に実行した例題を再計算する手間を省くため、上記の計算も旧 32bits 仕様のパソコンで実行した。

3.3 U 余因子による余計因子の除去

2.1 節において、互いに素な多項式 $F, G \in \mathbb{K}[x, \boldsymbol{u}]$ から x を消去して得られる終結式 R に対し、$R = AG + BG$ と表す余因子 (A, B) を定義した。系 $\{F_1, \ldots, F_{m+1}\}$ から計算される rectPRSs の任意の要素 R も剰余列で計算されるので、$R = A_1F_1 + \cdots + A_{m+1}F_{m+1}$ と表わされ、(A, B) と全く同様な算法で計算できる。$(A_1, \ldots, A_{m+1})$ を R の**満余因子** (**full cofactors**) と呼ぶ。満余因子は R より遥かに大きい多項式の組である場合が大部分である。実際、たとえば $R = \widehat{S}(\boldsymbol{u})$ では、上記等式の右辺を計算すると、$\boldsymbol{x}$ を含む項が多数現れるが、それらは全て互いにキャンセルする。そこで、各 A_j において、左辺の R に現れる変数より高位の変数すべてに 0 を代入した多項式を $\widetilde{A}_j$ とし、$(\widetilde{A}_1, \ldots, \widetilde{A}_{m+1})$ を R の **U 余因子** (**U-cofactors**) と命名する。U 余因子は [7] で導入されたが、以下に見るように意外な有用性を発揮する。下記定理 3 は [7] の命題 2 を少し一般化したもので、証明はその命題 2 と全く同様である。

定理 3

Let R be an element of rectPRSs of $\mathcal{F}$, such that R is a multiple of $G \in \mathrm{GB}(\mathcal{F})$. Let $(\widetilde{A}_1, \ldots, \widetilde{A}_{m+1})$ be the U-cofactors of R. If $\widetilde{a} := \gcd(\widetilde{A}_1, \ldots, \widetilde{A}_{m+1}) \notin \mathbb{K}$, then $\widetilde{a}$ is an extraneous factor of R.

満余因子の計算法を簡単に説明する。$\{R_1, \ldots, R_{m+1}\}$ を同じ主変数と次数を持つ rectPRSs の要素の組とし、それらの対 (R_{j_1}, R_{j_2}) から主項同士を消去して $r_j = a_jR_{j_1} + b_jR_{j_2}$ を計算したとする。このとき、r_j の満余因子は $a_j \times [R_{j_1}$の満余因子$] + b_j \times [R_{j_2}$の満余因子$]$ である。なお、M 簡約 $\widetilde{R}_j \xrightarrow{\widehat{S}\!\!\!\!\rightarrow} \widetilde{R}'_j$ に対しては、簡約前の満余因子を可能な限り $\widehat{S}$ で簡約してやればよい。

さて、$\mathrm{GB}(\mathcal{F}_1)$ の第二最小元 G_9 の計算に戻ろう。計算は式 (2) の三つの剰余 $(\widetilde{R}'_1, \ldots, \widetilde{R}'_3)$ から出発する。$c_j := \mathrm{lastPRS}_u(\mathrm{lc}_y(\widetilde{R}'_{j_1}), \mathrm{lc}_y(\widetilde{R}'_{j_2}))$、$c_j$ の余因子を (a_j, b_j) とすれば、$W_j := a_j\widetilde{R}'_{j_1} + b_j\widetilde{R}'_{j_2}$ である。各 $\widetilde{R}'_j$ の満余因子を $(A_{1,j}, \ldots, A_{3,j})$ とすれば、W_j に対する満余因子は $(a_jA_{1,j_1} + b_jA_{1,j_2},\ \ldots,\ a_jA_{3,j_1} + b_jA_{3,j_2})$ となる。次に U 余因子だが、定理 3 は rectPRSs の剰余 R に対するものであり、G_9 に対する U 余因子は $\widetilde{R}'_{j_1}$ および $\widetilde{R}'_{j_2}$ の主項 (= y を含む項) も U 余因子に含める必要がある。具体的に言えば、満余因子から、

$y\times$[w の多項式] + [w だけの多項式] を選び出して U 余因子とする。式 (2) の $\widetilde{R}'_1$ に $u=0$ を代入すると、y^1-項も y^0-項も w^4 の倍数であり、$\widetilde{R}'_2$ も $\widetilde{R}'_3$ もそうである。したがって、定理 3 (の拡張) により、式 (9) の $\overline{c}$ は余計因子 w^4 を含む。すなわち、式 (9) の因子である G_9 の主係数は w の 13 次以下の多項式である。

商の次数が不明確のまま割り切り除算で商を定めるには、どれかの次数以下で商の係数が 0 になることを利用するしかない。そのための ‘ゆとり次数” を 2 として、前節では w の次数範囲を 15 としたのである。なお、本稿では述べないが、式 (9) の $\overline{c}$ の因子のうち w^1 は G_9 の因子であることも分る；[7] の命題 1 参照。

3.4　お断りと今後の方針

前節では定理 3 のみならず、その拡張や G_9 に対する余因子など、細かい議論が欠けている。これは頁数の制約があるのと、新しい理論や定理は英語論文 (準備中) に記述したいが為である。

当面の課題は、定理 2 を与系 $\mathcal{F}$ が不健康な場合へ拡張し、LCtoW 多項式の算法を整理し効率化することである。実際、系 $\mathcal{F}_1$ においてすら、主係数イデアルには不健康な系が現れる。実は、これらの解明は既にある程度進んでいる；実際、定理 3 や $\mathcal{F}_1$ の第二最小元 G_9 に関する U 余因子の定義と算法などはその一例である。長期的課題としては、不健康な系も扱うソフトウェアの開発と大規模系を扱う技術の考案である。大規模系では現在見えていない多くの課題が待ち受けているはずで、[5] で言及した “分割征服消去法” は有用なはずだと確信している。是非とも自分の手で実装したい。

謝　辞

この研究は日本学術振興会の科研費 (課題番号 18K03389) および部分的に数理解析研究所・国際共同研究センター (京都大学内) の助成を受けている。

参 考 文 献

[1] B. Buchberger: An algorithm for finding a basis for the residue class ring of a zero-dimensional polynomial ideal (in German), Ph.D Thesis. Univ. of Innsbruck. Math. Inst. (1965).

[2] B. Buchberger: Gröbner bases: an algorithmic methods in polynomial ideal theory. *Multidimensional Systems Theory*, Chap. 6. Reidel Publishing (1985).

[3] A.C. Hearn: Non modular computation of polynomial GCDs using trial division. In: Proceedings EUROSAM ’79 (Springer LNCS **72**), 227-239 (1979).

[4] T. Sasaki: A theory and an algorithm for computing sparse multivariate polynomial remainder sequence. In: Computer Algebra in Scientific Computing, Springer LNCS **11077**, 345-360 (2018).

[5] T. Sasaki: An attempt to enhance Buchberger’s algorithm by using remainder sequences and GCD operation. In: SYNASC 2019, IEEE Conference Publishing Services, 27-34 (2020).

[6] T. Sasaki and D. Inaba: Simple relation between the lowest-order element of ideal $\langle G, H\rangle$ and the last element of polynomial remainder sequence. In: SYNASC 2017, IEEE Conference Publishing Services, 55-62 (2018).

[7] T. Sasaki and D. Inaba: Computing the lowest order element of the elimination ideal of multivariate polynomial system by using remainder sequence. In: SYNASC 2018, IEEE Conference Publishing Services, 37-44 (2019).