数理解析研究所講究録
第722巻 1990年 1-16

PC-PRS GCD算法の改良

理化学研究所　鈴木正幸 (Masayuki SUZUKI)
理化学研究所　佐々木建昭 (Tateaki SASAKI)

1 はじめに

我々は2年前、PC-PRS GCDと名付けた多変数多項式用のGCD算法を提案した。PC-PRS (Power-series Coefficient Polynomial Remainder Sequence) とは、主変数以外の変数をべき級数と見て、各変数およびそれら全体の次数和 (これを項次数と呼ぶ) について、あらかじめ定めた次数以上の項を切捨てて計算した剰余列のことである。GCDは与多項式から有理演算だけで計算でき、しかも余多項式に比べて次数やサイズが小さい。したがって、GCDの次数分だけPRSが計算されていれば、PRSを全部の次数について計算することなく、GCDを計算できる。PRSの最後の要素は余多項式に比べて一般に巨大なものとなるので、PC-PRSによりGCDを計算すれば著しい効率化が期待できる。

我々はPC-PRS GCD算法を定式化し、インプリメントしてみた[1]。その結果、従来のPRS算法に比べてPC-PRS GCD算法は著しく高効率であったが、EZ GCD算法と比較するとまだ改善の必要を感じた。本論文はその改善等に関するものである。

PC-PRS GCD算法の計算量は、GCDに含まれる各変数の次数上限と係数部の項次数上限をいかに見積もるかに大きく依存する。GCDの各変数に関する次数上限は与えられた多項式を一変数多項式に写影して簡単に求めることができるが、係数部の項次数上限を[1]の理論に基づいて求めると、

- 項次数の見積りが高くなることが多い。
- 項次数自身の計算に(主係数間の)GCDが必要なため、時間がかかることがある。

という欠点がある。

経験上は、GCDの係数部の項次数を、各従変数の次数上限の中で最大のもの程度に見積もれば十分であることが多い。こうすれば、PC-PRS GCD算法に比べ、項次数上限を大きく下げて計算できることになり、効率が大きく改善される。例えば、表1に、[1]の問題IVに対しどの程度効率が改良されるのかを示した。

[1] の問題 **IV.**

$$G=\left(\sum_{i=1}^{n}x_i+1\right)^m,\quad P_1=G\cdot\left(\sum_{i=1}^{n}\sum_{j\neq i}^{n}\{x_i(x_j-1)\}^m\right),\quad P_2=G\cdot\left(\sum_{i=1}^{n}\sum_{j\neq i}^{n}\left\{x_i(x_j^2+1)\right\}^m\right).$$

(n, m)	PC-PRS		EZGCD	P_1		P_2		G		
	(1)	(2)		terms	deg	terms	deg	terms	deg	E
IV-(3,2)	108	129	165	55	6	100	8	10	2	4
IV-(3,3)	351	591	441	136	9	270	12	20	3	6
IV-(3,4)	970	2193	990	268	12	535	16	35	4	8
IV-(3,5)	2709	7658	1992	505	15	969	20	56	5	10
IV-(4,2)	259	327	447	152	6	280	8	15	2	4
IV-(4,3)	989	2184	1488	430	9	956	12	35	3	6
IV-(4,4)	3057	11956	3878	1132	12	2214	16	70	4	8
IV-(4,5)	8547	51693	8833	2272	15	4688	20	126	5	10
IV-(5,2)	611	726	1036	315	6	635	8	21	2	4
IV-(5,3)	2848	7106	4239	1245	9	2620	12	56	3	6

PC-PRS (1) : GCD の項次数上限として従変数の最大次数を選んだもの (G の項次数と一致する)。

PC-PRS (2): [1] の理論を用いて計算したもの (理論上限値は最右の E 列に示した)。

表 1: 項次数上限の違いによる PC-PRS GCD 算法の効率比較

PC-PRS GCD 算法の項次数上限をこのように変更すれば、効率が大きく改善されることが分かるが、しかし、この見積りは小さ過ぎることがあり、その場合に項次数の上限を増やし PC-PRS GCD 算法で最初から GCD を再計算するのでは最初の計算を無駄にする。そこで、本論文では、PC-PRS GCD 算法で項次数 e まで計算された多項式から、GCD G を構成する方法を検討する。

一つは Hensel 構成 [2] を使う方法である。PC-PRS GCD 算法により GCD G が項次数 e まで求められているということは、従変数を $y,\cdots,z$ とすれば、多項式の法を $S=(y,\cdots,z)$ に取り、$G \pmod{S^{e+1}}$ が求められていることになる。したがって、Hensel 構成を用いて項次数 e までの G より項次数 $e+1$ までの G が構成できることがわかる。

もう一つの方法は、関係式 $P_k = A_k P_1 + B_k P_2$ を用いて、項次数 e まで求められた P_k を項次数 $e+1$ まで持ち上げるものである。この方法は簡単でありながら、従来検討されることがなかった方法であり、本論文で検討する。この方法では、Hensel 構成に必要な"$F = G \cdot H$ のとき $G_0 \equiv G \pmod{q,S}$ と $H_0 \equiv H \pmod{q,S}$ が互いに素"という条件が不要なため、簡単で効率的な算法を作ることができる。

さらに、この算法では A_k と B_k を同時に求めることもできる。これらが効率的に求められれば、不定積分の低減化や代数的数の逆元など、$\gcd(P_1,P_2) = AP_1 + BP_2$ を満たす A と B を計算する必要のある算法の効率化にも役立つと思われる。

2 記法と注意

本論文を通じて用いる記法と前提事項について説明する。$P,G,\cdots$ は、主変数が x、係数が $y,\cdots,z$ の多項式とする。[1] では GCD の係数部の項次数上限を定めるために項次数と各変数の次数の両方を使っていたが、本章では項次数だけを用いて議論する。また、剰余列は精度の低下が起こらないように計算されるものとする。つまり、剰余列の各 P_i の主係数が定数項を持つように前処理してから計算するものとする。この処理により、GCD および構成時に必要となる主係数同士の GCD は定数項を持つ。

以後、以下の記法を用いる。

$[P]_{e_l}^{e_h}$	: 係数の各項の項次数が $[e_l,e_h]$ に含まれる P の部分多項式を取り出す演算。
$\lceil P \rceil^e$	: $[P]_0^e$ の略記
$\lfloor P \rfloor_e$	: $[P]_e^\infty$ の略記
$P^{\overline{e}}$	: 各係数部が項次数 e 以下で、$\lceil P \rceil^e$ と等しい多項式を表す名前。
$P_{\underline{e}}$	: 各係数部が項次数 e 以上で、$\lfloor P \rfloor_e$ と等しい多項式を表す名前。
$P_{\underline{e_l}}^{\overline{e_h}}$	: 各係数部が項次数 e_l と e_h の間にあって、$[P]_{e_l}^{e_h}$ と等しい多項式を表す名前。
$A \overset{e}{\equiv} B$	: A と B の各係数部が項次数 e まで等しいことを表す。

GCD 計算のあらすじ　項次数 $e+1$ 以上の係数項を 0 と置いて計算した PRS(多項式剰余列) を、

$$(P_1^{\overline{e}}, P_2^{\overline{e}}, \cdots, P_k^{\overline{e}} \neq 0, P_{k+1}^{\overline{e}} = 0),$$

とする。$e \geq \mathrm{tdeg}(G)$ であれば、$P_i^{\overline{e}}$, $i = 1, \cdots, k-1$, で精度の低下が起きていない場合、$P_k^{\overline{e}}$ が真の GCD を与える。しかし $e < \mathrm{tdeg}(G)$ の時には $P_k^{\overline{e}}$ は G を与えず、その時に次の二通りの場合がある。

- P_{k+1} が実は 0 ではない場合。
- P_{k+1} は 0 であるが、$P_k^{\overline{e}}$ の係数の項次数上限 e が小さ過ぎる場合。

これらを考慮して、GCD を求める算法のあらましは次のようになる。

Step 1. 剰余列計算により、$P_k^{\overline{e}} \neq 0, P_{k+1}^{\overline{e}} = 0$ となる $P_k^{\overline{e}}$ を PC-PRS GCD 算法により求める。

Step 2. $P_k^{\overline{e}}$ から GCD の候補 $\tilde{P}$ を計算する ($\tilde{P} = \gamma P_k^{\overline{e}}$, $\mathrm{lc}(\tilde{P}) = g$ となるように主係数の調整を行なう)。
$G^{\overline{e}} = \mathrm{pp}(\tilde{P})$ と置き、$G^{\overline{e}} \mid P_1$ かつ $G^{\overline{e}} \mid P_2$ なら $G^{\overline{e}}$ が求める GCD である。

Step 3. $P_{k-1}^{\overline{e+1}}$ と $P_k^{\overline{e+1}}$ を構成し、除算により $P_{k+1}^{\overline{e+1}}$ を構成する。

Step 4. $P_{k+1}^{\overline{e+1}}$ が 0 なら $e \leftarrow e+1$ とし、そうでなければ $e \leftarrow e+1, k \leftarrow k+1$ として、Step 2. へ。

Step 3. の $P_k^{\overline{e+1}}$ の構成には Hensel 構成と、拡張された Euclid の公式 $P_k = A_k P_1 + B_k P_2$ を用いる構成 (以後、拡張 Euclid 構成と呼ぶことにする) の二通りの方法がある。以下、この二つの構成方法とこれらの構成方法を用いた GCD 算法について説明する。

3　Hensel 構成の利用

今 $P_1 = P = G \cdot H$ であり、$G^{\overline{0}}$ がわかっているとする。$G^{\overline{0}}$ と $H^{\overline{0}}$ が互いに素であるなら、良く知られているように $G^{\overline{0}}$ から G が構成できる。PC-PRS GCD 算法では、$P_k^{\overline{e}} \stackrel{e}{\equiv} G^{\overline{e}}$ となる $P_k^{\overline{e}}$ が求められていて、これから G を Hensel 構成する。

3.1　Hensel の補題

まず Hensel の補題の証明を PC-PRS GCD 算法とつなげられるように書き直し、計算上の注意点を述べる。

定理 3.1 (Hensel の補題) $P \in \mathbf{Z}\{y,...,z\}[x]$ とし、$\lceil \mathrm{lc}(P) \rceil^0 \neq 0$ とする。今、ある整数 e に対し、$\lceil P \rceil^e$ の約数 $G^{\overline{e}}$ が得られているとする。すると、

$$P \overset{e}{\equiv} G^{\overline{e}} H^{\overline{e}}, \tag{1}$$

となる $H^{\overline{e}}$ が得られ、$\left\lceil G^{\overline{e}} \right\rceil^0$ と $\left\lceil H^{\overline{e}} \right\rceil^0$ が互いに素であるなら、

$$P \overset{e+1}{\equiv} G^{\overline{e+1}} H^{\overline{e+1}}, \tag{2}$$

となる $G^{\overline{e+1}}$ と $H^{\overline{e+1}}$ が構成できる。

証明 $\deg_x(F) = d$ とする。まず $H^{\overline{e}}$ は、べき級数係数の多項式除算により、$\lceil P \rceil^e$ を $G^{\overline{e}}$ で割ることにより得られる。$\left\lceil G^{\overline{e}} \right\rceil^0$、$\left\lceil H^{\overline{e}} \right\rceil^0$ をそれぞれ $G^{\overline{0}}$、$H^{\overline{0}}$ と表す。G と H の係数部を項次数 e 以下の項と $e+1$ 以上の項に分けて考える。

$$\begin{aligned}
G &= \lceil G \rceil^e + \lfloor G \rfloor_{e+1}, \\
H &= \lceil H \rceil^e + \lfloor H \rfloor_{e+1}, \\
P &= (\lceil G \rceil^e + \lfloor G \rfloor_{e+1}) \cdot (\lceil H \rceil^e + \lfloor H \rfloor_{e+1}) \\
&= \lceil G \rceil^e \cdot \lceil H \rceil^e + \lceil G \rceil^e \cdot \lfloor H \rfloor_{e+1} + \lceil H \rceil^e \cdot \lfloor G \rfloor_{e+1} + \lfloor G \rfloor_{e+1} \cdot \lfloor H \rfloor_{e+1}.
\end{aligned}$$

これを項次数 $e+1$ までで見ると、

$$P \overset{e+1}{\equiv} \lceil \lceil G \rceil^e \cdot \lceil H \rceil^e \rceil^{e+1} + \lceil G \rceil^0 \cdot [H]_{e+1}^{e+1} + \lceil H \rceil^0 \cdot [G]_{e+1}^{e+1},$$

となる。$\lceil G \rceil^0 \overset{0}{\equiv} G^{\overline{0}}$ および $\lceil H \rceil^0 \overset{0}{\equiv} H^{\overline{0}}$、$\lceil G \rceil^e \overset{e}{\equiv} G^{\overline{e}}$、$\lceil H \rceil^e \overset{e}{\equiv} H^{\overline{e}}$ であるから、上式より

$$P - G^{\overline{e}} \cdot H^{\overline{e}} \overset{e+1}{\equiv} G^{\overline{0}} \cdot [H]_{e+1}^{e+1} + H^{\overline{0}} \cdot [G]_{e+1}^{e+1},$$

が得られる。そこで、$P_{\underline{e+1}}^{\overline{e+1}} = \left\lceil (P - G^{\overline{e}} \cdot H^{\overline{e}}) \right\rceil^{e+1}$ と置き、$G_{\underline{e+1}}^{\overline{e+1}} \overset{e+1}{\equiv} [G]_{e+1}^{e+1}$、$H_{\underline{e+1}}^{\overline{e+1}} \overset{e+1}{\equiv} [H]_{e+1}^{e+1}$ と置くと、

$$\begin{aligned}
P_{\underline{e+1}}^{\overline{e+1}} &= \sum_{i=0}^{d} C_i(y,\cdots,z)_{\underline{e+1}}^{\overline{e+1}} x^i \\
&= G^{\overline{0}} \cdot H_{\underline{e+1}}^{\overline{e+1}} + G^{\overline{0}} \cdot G_{\underline{e+1}}^{\overline{e+1}},
\end{aligned}$$

を満たす $G_{\underline{e+1}}^{\overline{e+1}}$ と $H_{\underline{e+1}}^{\overline{e+1}}$ を求めれば、$G^{\overline{e}}$ と $H^{\overline{e}}$ より、$G^{\overline{e+1}}$ と $H^{\overline{e+1}}$ が構成できる。この $G_{\underline{e+1}}^{\overline{e+1}}$ と $H_{\underline{e+1}}^{\overline{e+1}}$ は次のように求められる。今、$G^{\overline{0}}$ と $H^{\overline{0}}$ は互いに素で、主変数 x だけの多項式であるから、

$$A_0^{\overline{0}} G^{\overline{0}} + B_0^{\overline{0}} H^{\overline{0}} = 1,\ \deg(A_0^{\overline{0}}) < \deg(H^{\overline{0}}),\ \deg(B_0^{\overline{0}}) < \deg(G^{\overline{0}}),$$

となる x の多項式 $A_0^{\overline{0}}$ と $B_0^{\overline{0}}$ が存在する。この式の両辺に x^i をかけると、

$$x^i A_0^{\overline{0}} G^{\overline{0}} + x^i B_0^{\overline{0}} H^{\overline{0}} = x^i,$$

となるが、$x^i A_0^{\overline{0}}$ と $H^{\overline{0}}$ に対しては

$$x^i A_0^{\overline{0}} = Q_i^{\overline{0}} H^{\overline{0}} + R_i^{\overline{0}}, \ \deg(R_i^{\overline{0}}) < \deg(H^{\overline{0}}),$$

となる除算が有理数体上で可能であるから、

$$R_i^{\overline{0}} G^{\overline{0}} + \left\{ Q_i^{\overline{0}} G^{\overline{0}} + x^i B_0^{\overline{0}} \right\} H^{\overline{0}} = x^i,$$

を得る。したがって、$A_i^{\overline{0}} = R_i^{\overline{0}}$ 、$B_i^{\overline{0}} = Q_i^{\overline{0}} + x^i B_0^{\overline{0}}$ と置くと、

$$\begin{aligned} P_{\underline{e+1}}^{\overline{e+1}} &= \sum_{i=0}^{d} C_{i\underline{e+1}}^{\overline{e+1}} x^i \\ &= \sum_{i=0}^{d} C_{i\underline{e+1}}^{\overline{e+1}} \left(A_i^{\overline{0}} G^{\overline{0}} + B_i^{\overline{0}} H^{\overline{0}} \right) \\ &= \left(\sum_{i=0}^{d} C_{i\underline{e+1}}^{\overline{e+1}} A_i^{\overline{0}} \right) G^{\overline{0}} + \left(\sum_{i=0}^{d} C_{i\underline{e+1}}^{\overline{e+1}} B_i^{\overline{0}} \right) H^{\overline{0}}. \end{aligned}$$

以上より、$H_{\underline{e+1}}^{\overline{e+1}} = \sum_{i=0}^{d} C_{i\underline{e+1}}^{\overline{e+1}} A_i^{\overline{0}}$, $G_{\underline{e+1}}^{\overline{e+1}} = \sum_{i=0}^{d} C_{i\underline{e+1}}^{\overline{e+1}} B_i^{\overline{0}}$ と置けばよい。これらの次数は

$$\begin{aligned} \deg(H_{\underline{e+1}}^{\overline{e+1}}) &< \deg(H^{\overline{0}}), \\ \deg(G_{\underline{e+1}}^{\overline{e+1}}) &= \deg(G^{\overline{0}}). \end{aligned}$$

□

注 3.1 この補題で項次数の演算を行なっているところは、$\lceil P \rceil^e$ を $G^{\overline{e}}$ で割るところと、$P_{\underline{e+1}}^{\overline{e+1}}$ の計算、$P - G^{\overline{e}} \cdot H^{\overline{e}}$ のところだけである。$\lceil \mathrm{lc}(P) \rceil^0 \neq 0$ の条件があるので、$G^{\overline{0}} \neq 0$、$H^{\overline{0}} \neq 0$ であり、この計算で精度の低下が起こることはない。□

この補題による構成では $\mathrm{lc}(H^{\overline{e+1}}) = \mathrm{lc}(H^{\overline{e}})$ だが、一般に $\mathrm{lc}(G^{\overline{e+1}}) \neq \mathrm{lc}(G^{\overline{e}})$ となる。したがって、一般にこの構成を繰り返して得られる $G^{\overline{\infty}} \equiv \tilde{G}$ は P の因子 G とは単元倍の違いがある: $\tilde{G} = uG, u \in \mathbf{Z}\{y, \cdots, z\}$。すなわち、いわゆる主係数問題が起こる。

3.2 主係数問題の回避法

前節で述べたように、Hensel 法で構成されるのは G と H ではなく、uG と vH、ただし $uv = 1$、であり、主係数の不定性がある。G の主係数があらかじめわかっていれば、その不定性をなくすように構成できる。その方法について簡単に復習し、項次数の記法で書き直す。

問題点は、$d = \deg(P)$ として、$x^d = A_d^{\overline{0}} G^{\overline{0}} + B_d^{\overline{0}} H^{\overline{0}}$ を満足する $A_d^{\overline{0}}$ と $B_d^{\overline{0}}$ がユニークに決められないことである。今、$\tilde{g} = \gcd(\mathrm{lc}(P_1), \mathrm{lc}(P_2))$ とすると、$\mathrm{lc}(G) \mid \tilde{g}$ である。そこで、G の代わりに、$\mathrm{lc}(\tilde{G}) = \tilde{g}$ かつ $G | \tilde{G}$ なる $\tilde{G}$ を求めることにすれば、構成すべき多項式の主係数がわ

かっていることになる。つまり、G, H, P のかわりに

$$\tilde{G}^{\overline{e}} \stackrel{e}{\equiv} \lceil \tilde{g} \rceil^e G^{\overline{e}} / \mathrm{lc}(G^{\overline{e}}),$$
$$\tilde{H}^{\overline{e}} \stackrel{e}{\equiv} \mathrm{lc}(P^{\overline{e}}) H^{\overline{e}} / \mathrm{lc}(H^{\overline{e}}),$$
$$\tilde{P} = \tilde{g} P,$$

なる $\tilde{G}, \tilde{H}$ および $\tilde{P}$ を構成すれば良い。この $\tilde{P}$ に対し Hensel 構成を行ない、$G^{\overline{e}}$ の主係数を $\lceil \tilde{g} \rceil^e$ にそろえれば、$\tilde{G}$ が構成できる。この算法は EZ 構成と呼ばれている。これらの計算が項次数 e までの精度で行なわれるためには $\tilde{g}$ に項次数 e までの精度があることが必要である。

3.3 PC-PRS + Hensel GCD 算法の性能

PC-PRS 算法と Hensel 構成の合成算法を以後 PC-PRS + Hensel GCD 算法と呼ぶことにする。この算法は数式処理システム GAL 上に、[1] で述べた PC-PRS GCD を用いて実現した。項次数上限による打ち切りは、GAL のべき級数計算機構を用いて実行している。三変数以上の多項式の GCD を計算する場合には、項次数を表すために新たな変数を導入している。PRS を求めるための公式としては縮小 PRS のものを用いている。入力多項式の係数が定数項を持つように、変数変換により前処理する。また主変数は最も高い次数を持つ変数を選んでいる。次数上限をできるだけ下げる方が、PRS の長さを短くすることより効率的であると考えられるからである。もし、$\mathrm{lc}(P_1)$ = 定数あるいは $\mathrm{lc}(P_2)$ = 定数の時は次数上限が最悪の時であり、この時に限り、次数上限を P_1 と P_2 の主変数に関して最低次数の項によって決めている。

表 2 と 3 は PC-PRS + Hensel GCD および REDUCE に装備されている二つの GCD 算法の実行速度である。(単位はミリ秒)。P_1, P_2, G の多項式の大きさを表中の P_1, P_2, G に対応する各列に示す。表の一列目の $X-(i_n, i_m)$ における i_n と i_m は問題 X の中の n と m の値を意味する。GAL は Cambridge Lisp 上のもの、REDUCE は SLISP 上のものである。計算機は FACOM M780 である。

ここで用いたテスト問題では、いずれも G_0 と H_0 が互いに素になる。GCD の主変数に関する次数を d_g と表すと、これらの問題では d_g は与多項式 $P_i, i = 1, 2$、の次数の約半分であり、PC-PRS + Hensel GCD 算法の場合の Hensel 構成の回数は係数部の項次数で決まり、問題 I と問題 II でそれぞれ d_g 回、$2d_g$ 回である。

PC-PRS + Hensel GCD 算法で純粋な PRS 計算の時間と Hensel 構成に要する時間を、表の (PRS) と (Hensel) の列に示した。残りの時間は、前・後処理 (変数変換、項次数の見積り、項次数を表す変数の導入など) に費やされている。問題 I については、Hensel 構成に要する時間は 1/3 程度であり、問題 II については約半分を占めるようになる。問題の次数が高くなるにつれて、PRS の計算時間の占める割合は大きくなる。

これらの問題では、ほとんどの場合について GAL 上の PC-PRS + Hensel GCD 算法が EZ GCD 算法を上回る結果を示している。Hensel 構成の回数が少ない問題 I の方が PC-PRS + Hensel

問題 I.

$$G = \sum_{l=1}^{m} \left(\sum_{i=1}^{n} x_i^l + \sum_{i \neq j}^{n} x_i^l x_j^l \right),$$
$$P_1 = G \cdot \left(1 + 2x_1 + \sum_{l=1}^{m} \left(\sum_{i=1}^{n} x_i^l \right) \right),$$
$$P_2 = G \cdot \left(1 + \sum_{l=1}^{m} \sum_{i=1}^{n} x_i^{l+1} \right)$$

(3,m)	PC-PRS+Hensel			EZ	Trial	P_1		P_2		G		
	total	(PRS)	(Hensel)	GCD	Div.	terms	deg	terms	deg	terms	deg	E
I-(3,2)	105	(11)	(33)	132	295	51	6	68	7	13	4	2
I-(3,3)	218	(32)	(78)	286	2908	103	9	126	10	19	6	3
I-(3,4)	415	(90)	(149)	504	21571	173	12	202	8	25	8	4
I-(3,5)	740	(208)	(261)	815	–	261	15	296	16	31	10	5

E はヒューリスティックに決めた G の項次数上限値である。

表 2: PC-PRS + Hensel GCD 算法の実行速度 –(1)

問題 II.

$$G = \sum_{l=1}^{m} \left(\sum_{i=1}^{n} x_i^l + \sum_{i \neq j}^{n} x_i^l x_j^l + \sum_{i \neq j \neq k}^{n} x_i^l x_j^l x_k^l \right),$$
$$P_1 = G \cdot \left(1 + 2x_1 + \sum_{l=1}^{m} \left(\sum_{i=1}^{n} x_i^l \right) \right),$$
$$P_2 = G \cdot \left(1 + \sum_{l=1}^{m} \sum_{i=1}^{n} x_i^{l+1} \right)$$

(3,m)	PC-PRS+Hensel			EZ	Trial	P_1		P_2		G		
	total	(PRS)	(Hensel)	GCD	Div.	terms	deg	terms	deg	terms	deg	E
II-(3,2)	143	(11)	(63)	174	583	60	8	78	9	15	6	2
II-(3,3)	329	(32)	(169)	389	5464	121	12	145	13	22	9	3
II-(3,4)	658	(91)	(353)	704	34402	203	16	233	17	29	12	4
II-(3,5)	1195	(210)	(646)	1182	–	306	20	342	21	36	15	5

表 3: PC-PRS + Hensel GCD 算法の実行速度 –(2)

GCD 算法に有利という結果が出ているが、二つの問題で PRS 計算に要する時間はほぼ同じであるので、これは我々の Hensel 構成プログラムにまだ改良の余地があるということを意味する。次数の増加に対する計算時間の増加の割合は、PC-PRS + Hensel GCD 算法の方が大きい。この理由は二つ考えられる。一つは、項次数を表すために余分な変数を導入していることであり、もう一つは数係数の膨張である。これらについては、今後の改良点にしたい。

4 $A_kP_1 + B_kP_2 = P_k$ を利用した構成 (拡張 Euclid 構成)

Hensel 構成の欠点は $G^{\overline{0}}$ と $H^{\overline{0}}$ が互いに素な多項式でなければならないことである。そのため、Hensel 構成に基づく EZGCD 算法では、与式をまず無平方分解することで、互いに素な $G^{\overline{0}}$ と $H^{\overline{0}}$ が作れるように前処理を行なっているが、PC-PRS GCD 算法と組み合わせるにはこの方法は効率的ではない。これに対し、以下で述べる構成方法は $G^{\overline{0}}$ と $H^{\overline{0}}$ が互いに素という条件を必要とせず、任意の k について $P_k{}^{\overline{e}}$ を $P_k{}^{\overline{e+1}}$ にリフティングできるものである。したがって GCD の構成にも使え、Hensel 構成よりも一般的な方法である。この構成方法を拡張 Euclid 構成と呼ぶことにする。

4.1 拡張 Euclid 構成のための定理

拡張 Euclid 構成のための定理を示す。

定理 4.1 与多項式 P_1 と P_2、$P_i \in \mathbf{Z}[y,\cdots,z][x], i = 1,2$、に対し、

$$\begin{aligned} P_k{}^{\overline{0}} &\overset{0}{\equiv} A_k{}^{\overline{0}}P_1 + B_k{}^{\overline{0}}P_2, \\ \deg A_k{}^{\overline{0}} &< \deg(P_2) - \deg(P_k{}^{\overline{0}}), \\ \deg B_k{}^{\overline{0}} &< \deg(P_1) - \deg(P_k{}^{\overline{0}}), \end{aligned} \tag{3}$$

を満たす $P_k{}^{\overline{0}}, A_k{}^{\overline{0}}, B_k{}^{\overline{0}}$ がわかっているとする。これより、任意の正の整数 e に対し、

$$\begin{aligned} P_k{}^{\overline{e}} &\overset{e}{\equiv} A_k{}^{\overline{e}}P_1 + B_k{}^{\overline{e}}P_2, \\ \deg A_k{}^{\overline{e}} &< \deg(P_2) - \deg(P_k{}^{\overline{e}}), \\ \deg B_k{}^{\overline{e}} &< \deg(P_1) - \deg(P_k{}^{\overline{e}}), \\ P_k &\overset{0}{\equiv} P_k{}^{\overline{e}}, \\ A_k{}^{\overline{0}} &\overset{0}{\equiv} A_k{}^{\overline{e}}, \\ B_k{}^{\overline{0}} &\overset{0}{\equiv} B_k{}^{\overline{e}}, \end{aligned} \tag{4}$$

となる $P_k{}^{\overline{e}}, A_k{}^{\overline{e}}, B_k{}^{\overline{e}}$ を構成できる。

証明 拡張された Euclid の定理より $P_k = A_kP_1 + B_kP_2$ を満たす A_k と B_k が存在するので、(4) 式を満たす $P_k{}^{\overline{e}}$ と $A_k{}^{\overline{e}}$, $B_k{}^{\overline{e}}$ が存在するのは明らか。証明は項次数に関する数学的帰納法による。

仮定より、$P_k^{\overline{0}}, A_k^{\overline{0}}, B_k^{\overline{0}}$ が得られている。今、$P_k^{\overline{e}}$ と $A_k^{\overline{e}}, B_k^{\overline{e}}$ が構成できていると仮定する。構成すべき $P_k^{\overline{e+1}}$ と $A_k^{\overline{e+1}}, B_k^{\overline{e+1}}$ の各係数を項次数 e 以下の項と $e+1$ の項に分けて考える。

$$\begin{aligned} A_k^{\overline{e+1}} &= A_k^{\overline{e}} + A_{\underline{e+1}}^{\overline{e+1}}, \\ B_k^{\overline{e+1}} &= B_k^{\overline{e}} + B_{\underline{e+1}}^{\overline{e+1}}, \\ P_k^{\overline{e+1}} &= P_k^{\overline{e}} + P_{\underline{e+1}}^{\overline{e+1}}, \end{aligned}$$

すると、

$$\begin{aligned} P_k^{\overline{e+1}} &\overset{e+1}{\equiv} (A_k^{\overline{e}} + A_{\underline{e+1}}^{\overline{e+1}})P_1 + (B_k^{\overline{e}} + B_{\underline{e+1}}^{\overline{e+1}})P_2 \\ &\equiv P_k^{\overline{e}} + \left[(A_k^{\overline{e}}P_1 + B_k^{\overline{e}}P_2)\right]_{e+1}^{e+1} + A_{\underline{e+1}}^{\overline{e+1}}\lceil P_1\rceil^0 + B_{\underline{e+1}}^{\overline{e+1}}\lceil P_2\rceil^0. \end{aligned}$$

したがって、次式を得る。

$$\begin{aligned} P_{\underline{e+1}}^{\overline{e+1}} \overset{e+1}{\equiv}\ & A_{\underline{e+1}}^{\overline{e+1}}\lceil P_1\rceil^0 + B_{\underline{e+1}}^{\overline{e+1}}\lceil P_2\rceil^0 + \left[(A_k^{\overline{0}}P_1 + B_k^{\overline{0}}P_2)\right]_{e+1}^{e+1}, \\ & \deg(P_{\underline{e+1}}^{\overline{e+1}}) \le \deg(P_k^{\overline{0}}), \\ & \deg(A_{\underline{e+1}}^{\overline{e+1}}) \le \deg(A_k^{\overline{0}}), \\ & \deg(B_{\underline{e+1}}^{\overline{e+1}}) \le \deg(B_k^{\overline{0}}). \end{aligned}$$

この式を満たす $A_{\underline{e+1}}^{\overline{e+1}}, B_{\underline{e+1}}^{\overline{e+1}}$ および $P_{\underline{e+1}}^{\overline{e+1}}$ は、副剰余列算法 [3] によって求めることができる。□

注 4.1 A_k や B_k は一般に大きな多項式となるので、この方法は一見すると効率が悪そうに見える。しかし、我々が欲しいのは tdeg(G) までの項次数の $A_k^{\overline{e}}$ と $B_k^{\overline{e}}$ であり、副剰余列を計算する際に現れる多項式は $A_i^{\overline{0}}, B_i^{\overline{0}}, P_i^{\overline{0}}, A_{\underline{e+1}}^{\overline{e+1}}, B_{\underline{e+1}}^{\overline{e+1}}, P_{\underline{e+1}}^{\overline{e+1}}$、つまり係数部の項次数が 0 と $e+1$ の多項式だけである。したがって、$P_{\underline{e+1}}^{\overline{e+1}}$ を求めるための計算量はそう多くない。□

4.2 拡張 Euclid 構成 GCD 算法の性能

この算法は数式処理システム GAL 上で実現した。項次数上限による打ち切りを必要とする計算には、GAL のべき級数計算機構を用いている。三変数以上の多項式の GCD を計算する場合には、係数部の項次数を表すために新たな変数を導入している。入力多項式の係数が定数項を持つように、変数変換により前処理する。また主変数は最も高い次数を持つ変数を選んでいる。PC-PRS GCD 算法とつなげる意味と、係数の項次数を低くすれば数式膨張が抑えられて効率的であると考えられるからである。また、係数の項次数が低ければ拡張 Euclid 構成の回数も少なくて済む。なお、以下の性能評価は、純粋に拡張 Euclid 構成だけで GCD を求める算法についてである。すなわち、まず $P_k^{\overline{0}}$ と $A_k^{\overline{0}}, B_k^{\overline{0}}$ を計算し、これらから拡張 Euclid 構成で $P_k^{\overline{e}}A_k^{\overline{e}}, B_k^{\overline{e}}$ を構成する方法である。こうすることで、拡張 Euclid 構成の効率と Hensel 構成の効率を比較することができる。

表4から表7は、[1] の問題 I から IV に対し、純粋に拡張 Euclid 構成だけで GCD を求めた場合と、二つの PC-PRS GCD 算法で求めた場合、そして REDUCE に装備されている二つの GCD 算法の実行速度の比較である。(単位はミリ秒)。P_1, P_2, G に関する情報を表中の P_1, P_2, G に対応する各列に示す。表の一列目の $X-(i_n, i_m)$ における i_n と i_m は問題 X の中のパラメータ n と m の値を意味する。拡張 Euclid 構成算法および二つの PC-PRS GCD 算法は Cambridge Lisp 上の GAL システムで実現したものであり、REDUCE は SLISP 上のものである。計算機は FACOM M780 である。

[1] の問題 I この問題に対しては、PC-PRS 算法が最も良い結果を得ている。拡張 Euclid 構成算法は、PC-PRS GCD 算法の約 2 倍程度の時間がかかっている。EZ GCD 算法と拡張 Euclid 構成算法を比べると、変数の次数の低いところでは拡張 Euclid 構成算法が速いが、途中で逆転し、次数が高くなるとその差は大きくなる傾向にある。

注 4.2 PC-PRS (1)、すなわち項次数上限として従変数の最大次数を選ぶ PC-PRS 算法、は今の場合、最適の項次数上限で計算することになり、係数部をリフティングする必要はない。このことは、[1] の問題 II~IV に対しても同じである。□

[1] の問題 II この問題についても、[1] の問題 I と同じ傾向が見られる。

[1] の問題 III この問題には構成法が有利である。EZ GCD 算法と拡張 Euclid 構成算法を比べると、次数の低いところでは拡張 Euclid 構成算法が速いが、3 次の所で逆転し、次数が高くなるとその差はわずかに大きくなる傾向にあるが、[1] の問題 I ほどではない。

注 4.3 PC-PRS (1) と (2) は今の場合、同じ項次数上限で同じ計算をするが、(1) の方が (2) より効率的になる理由は、(2) では項次数上限の計算に多大な時間がかかるからである。□

[1] の問題 IV この問題に対しては、三つの算法がそれぞれ異なる傾向を見せている。まず、PC-PRS GCD 算法は変数の数によらず、低次数のところで良い結果を得ている。拡張 Euclid 構成算法は、変数の個数が多い方が良い結果を得ている。EZ GCD 算法は、変数の個数が少ない方が、また次数が高い方が良い結果を得ている。

以上より拡張 Euclid 構成算法は、低次数の多項式に対して EZ GCD 算法を上まる性能を示すこと、変数の個数が多くなれば EZ GCD 算法より有利になる傾向があること、が分かる。したがって、拡張 Euclid 構成算法だけでも十分実用になるものと思われる。

[1] の問題 I.

$$
\begin{aligned}
P_1 &= (1+y)x^4+(1+y-y^2-yz)x^3+(-1-y+z-2y^2-2yz-z^2)x^2 \\
&+ (-2-y+3z+y^2+2yz-z^2+y^3+y^2z)x+(-1+y+3z-2yz-3z^2+yz^2+z^3), \\
P_2 &= (-1+z)x^4+(-4+y+2z-yz-z^2)x^3+(-2+4y+3z)x^2 \\
&+ (3+y-2z-y^2-yz)x+(2-y-3z-y^2+z^2),
\end{aligned}
$$

上式の P_1 と P_2 に対し次の置き換えをする。

$$
x \to \sum_{i=0}^{m} x^i,\ \ y \to \sum_{i=0}^{m} y^i,\ \ z \to \sum_{i=0}^{m} z^i.
$$

(3,m)	Extend	PC-PRS		EZ	Trial	P_1		P_2		G		
	Euclid	(1)	(2)	GCD	Div.	terms	deg	terms	deg	terms	deg	E
I-(3,1)	31	15	12	46	45	27	5	20	5	4	1	1
I-(3,2)	155	97	78	132	673	112	10	80	10	6	2	2
I-(3,3)	447	279	230	338	2889	277	15	198	15	9	3	3
I-(3,4)	1188	677	571	717	–	544	20	392	20	12	4	4
I-(3,5)	3071	1519	1321	1582	–	940	25	680	25	15	5	5

PC-PRS (1) : GCD の項次数の上限として従変数の最大次数を選んだもの (G の項次数と一致する)。

PC-PRS (2) : [1] の理論を用いて計算したもの (上限値は最右の E 列に示した)。

表 4: 拡張 Euclid 構成算法の実行速度 –(1)

[1] の問題 II.

$$G = \sum_{i=1}^{n} x_i + 1, \quad P_1 = G \cdot \left(\sum_{i=1}^{n} x_i^n - 2 \right), \quad P_2 = G \cdot \left(\sum_{i=1}^{n} x_i^{n-1} + 2 \right)$$

(n)	Extend	PC-PRS		EZ	Trial	P_1		P_2		G		
	Euclid	(1)	(2)	GCD	Div.	terms	deg	terms	deg	terms	deg	E
II-(2)	7	4	3	16	5	9	3	6	2	3	1	1
II-(3)	21	10	11	18	12	16	4	16	3	4	1	1
II-(4)	40	17	22	32	44	25	5	25	4	5	1	1
II-(5)	71	29	39	41	199	36	6	36	5	6	1	1
II-(6)	115	44	62	59	953	49	7	49	6	7	1	1

表 5: 拡張 Euclid 構成算法の実行速度 –(2)

[1] の問題 III.

$$G = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i^j + 1, \quad P_1 = G \cdot \left(\sum_{i=1}^{n} \sum_{j=1}^{n} x_i^j - 2 \right), \quad P_2 = G \cdot \left(\sum_{i=1}^{n} \sum_{j=1}^{n} x_i^{j-1} + 2 \right)$$

(n)	Extend	PC-PRS		EZ	Trial	P_1		P_2		G		
	Euclid	(1)	(2)	GCD	Div.	terms	deg	terms	deg	terms	deg	E
III-(2)	18	12	11	21	7	11	4	10	3	5	2	2
III-(3)	59	56	60	58	43	43	6	40	5	10	3	3
III-(4)	183	269	359	175	860	125	8	119	7	17	4	4
III-(5)	490	1376	2089	442	–	296	10	286	9	26	5	5
III-(6)	1142	7802	12694	1007	–	607	12	592	11	37	6	6

表 6: 拡張 Euclid 構成算法の実行速度 –(3)

[1] の問題 **IV.**

$$G = \left(\sum_{i=1}^{n} x_i + 1\right)^m, \quad P_1 = G \cdot \left(\sum_{i=1}^{n}\sum_{j\neq i}^{n} \{x_i(x_j - 1)\}^m\right), \quad P_2 = G \cdot \left(\sum_{i=1}^{n}\sum_{j\neq i}^{n} \left\{x_i(x_j^2 + 1)\right\}^m\right).$$

(n,m)	Extend	PC-PRS		EZ	Trial	P_1		P_2		G		
	Euclid	(1)	(2)	GCD	Div.	terms	deg	terms	deg	terms	deg	E
IV-(2,2)	31	24	22	40	66	16	6	25	8	6	2	2
IV-(2,3)	74	62	57	83	1023	29	9	50	12	10	3	3
IV-(2,4)	154	134	124	153	10704	51	12	81	16	15	4	4
IV-(2,5)	469	417	404	255	–	72	15	122	20	21	5	5
IV-(3,2)	123	108	129	165	–	55	6	100	8	10	2	4
IV-(3,3)	391	351	591	441	–	136	9	270	12	20	3	6
IV-(3,4)	1040	970	2193	990	–	268	12	535	16	35	4	8
IV-(3,5)	2860	2709	7658	1992	–	505	15	969	20	56	5	10
IV-(4,2)	287	259	327	447	–	152	6	280	8	15	2	4
IV-(4,3)	1060	989	2184	1488	–	430	9	956	12	35	3	6
IV-(4,4)	3217	3057	11956	3878	–	1132	12	2214	16	70	4	8
IV-(4,5)	8854	8547	51693	8833	–	2272	15	4688	20	126	5	10
IV-(5,2)	662	611	726	1036	–	315	6	635	8	21	2	4
IV-(5,3)	3020	2848	7106	4239	–	1245	9	2620	12	56	3	6

表 7: 拡張 Euclid 構成算法の実行速度 –(4)

5 まとめ

本論文では、PC-PRS GCD 算法において、GCD の項次数の上限をヒューリスティックに見積もる算法を示し、上限の推定値が小さ過ぎて GCD が求まらない場合に、低い次数で打ち切られた多項式 (剰余列の最後の要素) の係数をリフティングして、真の GCD を構成する方法を示した。一つは従来の Hensel 構成を使う方法 (PC PRS + Hensel GCD 算法) であり、もう一つは関係式 $A_kP_1 + B_kP_2 = P_k$ を使った新しい構成方法 (拡張 Euclid 構成 GCD 算法) である。

注 5.1 [PC-PRS GCD の 改良算法] 2節で示した算法は、GCD の構成という点からは簡潔な算法であるが、十分効率的ではない。現実問題では、2節の算法の Step. 2 で GCD が求まらないのは、$P_k^{\overline{e}}$ の係数の打ち切り次数 e が小さ過ぎる場合がほとんどで、$P_{k+1}^{\overline{e+1}}$ が実は 0 ではないという場合は確率的にはずっと少ないからである。したがって、$P_k^{\overline{e}}$ から $P_k^{\overline{e+1}}$ をまずリフティングして GCD を構成する方を優先すべきで、それでも GCD が求まらない場合に $P_{k-1}^{\overline{e+1}}$ と $P_k^{\overline{e+1}}$ を構成し、除算により $P_{k+1}^{\overline{e+1}}$ を構成すべきである。$P_k^{\overline{e}}$ から GCD を構成する際には、$G^{\overline{0}}$ と $H^{\overline{0}}$ が互いに素であるなら Hensel 構成を使い、互いに素でない時には拡張 Euclid 構成を使う。拡張 Euclid 構成のためには $P_k^{\overline{e}}$ の他に $A_k^{\overline{e}}$, $B_k^{\overline{e}}$ が必要である。PC-PRS GCD 算法においてこの $A_k^{\overline{e}}$ と $B_k^{\overline{e}}$ を同時に計算するのはオーバーヘッドになるので、除算の商を保存しておき、必要になった時点で $A_k^{\overline{e}}$ と $B_k^{\overline{e}}$ を計算することにするのがよい。なお、$A_kP_1 + B_kP_2 = P_k$ を満たす A_k や B_k は、不定積分や代数的数の取り扱いなどにおいて必要となる。拡張 Euclid 算法は PC-PRS GCD 算法の改良のためばかりでなく、A_k と B_k を高速に求めるという算法にも使えると予測できる。
□

実験の結果、項次数上限をヒューリステックな方法で見積り、PC-PRS GCD 算法で GCD を求めると効率が良いことが分かった。PC-PRS GCD 算法のこの改良版は、密な多項式であれば変数の数が増えるに従い EZ GCD 算法より効率が良くなる傾向を示す。一方、多項式の次数が上がるに従い EZ GCD 算法の方が有利になる。しかし、項次数上限の見積り方法、構成プログラムの効率化、数係数の膨張を抑えるための工夫、専用べき級数演算の作成等、プログラム上改良すべき点が多々ある。これらの改良を行なえば、現在の数式処理システムが扱う多項式の次数程度では、PC-PRS GCD 算法は EZ GCD 算法をしのぐ可能性がある。

EZ GCD 算法は、現在最高速の多変数多項式 GCD 算法であり、1973 年の発表以来、工夫に工夫を重ねて効率化されてきた。その EZ GCD 算法に比べて、まだ充分にチューニングされたとは言えない我々の算法が遜色のない性能を示すことは驚きに値する。本章に示した PC-PRS の二つの改良算法と EZ GCD 算法は、変数の個数と次数の増加に対し、それぞれ異なった特性を示す。したがって、PC-PRS GCD 算法をさらにチューニングし、多くのケースでテストして特性を調べた上で EZ GCD 算法と組み合わせれば、最適化された GCD 算法が得られるであろう。それは今後の課題にしたい。

参考文献

[1] T. Sasaki and M. Suzuki. Three New Algorithms for Multivariate Polynomial GCD. *J. Symbolic Computation*, submitted.

[2] J. Moses and D.Y.Y. Yun. The EZ GCD Algorithm. In *Proceedings of ACM'73*, page 159, 1973.

[3] T. Sasaki and A. Furukawa. Secondary Polynomial Remainder Sequence and an Extension of Subresultant Theory. *J. Inf. Proces.*, 7:175–184, 1984.