数理解析研究所講究録 1395 巻 2004 年 91-96

PC-PRS算法による多変数近似GCD計算

讃岐 勝

MASARU SANUKI

筑波大学 数理物質科学研究科

DRADUATE SCHOOL OF PURE AND APPLIED SCIENCES,UNIVERSITY OF TSUKUBA *

1 はじめに

Zhi・野田 [ZN99] により、多変数多項式近似 GCD を求める方法としていくつかの実験がされた。PRS(多項式剰余列) 算法は精度が安定しているが係数のサイズが指数的に増大したり、GCD= 1 の場合には最後まで計算しなければならず計算時間がかかってしまう。Modular 算法は GCD= 1 には計算が速いが、GCD≠ 1 の場合には PRS 算法同様に係数が指数的に増大してしまう。EZ-GCD(EZ は Extended Zassenhaus の略) は Hensel 法を用いて求める方法であるが、Hensel 法の問題点である「主係数問題」などがあったり近似 GCD の場合には精度が悪いことが知られている。また補間法は多数の項の場合に弱く、どれも決定的な方法とはいえない。

ここでは精度の安定している PRS 算法に注目することにして、PRS を効率よく計算する方法の 1 つとしてとして、佐々木・鈴木 [SS92] による PC-PRS 算法を紹介して、近似演算に対しても計算できるように定義した近似 PC-PRS 算法をデータを見ながら検証していく。

本論では次の記号を用いる。

K	:	数体
$x, y, \cdots, z$	:	変数、x を主変数とする
$\mathrm{lc}(F)$	:	多項式 F の主係数
$\mathrm{coef}(F)$	:	多項式 F の係数
$\deg_u(F)$	:	変数 u に対する F の次数
$\mathrm{tdeg}(F)$	:	多項式 F の全次数
$\mathrm{cont}(F)$	:	多項式 F の係因子
$\mathrm{pp}(F)$	:	多項式 F の原始的部分　$\mathrm{pp}(F)=F/\mathrm{cont}(F)$
$\gcd(P_1, P_2)$	:	多項式 P_1, P_2の最大公約因子

2 PC-PRS算法

PC-PRS 算法とは、Power-series Coefficient Polynomial Remainder Sequence の略であり、べき級数係数 PRS 算法とも呼ばれる。

与えられた多変数多項式 P_1, P_2 について、PRS 算法を用いて $\gcd(P_1, P_2) = G$ を求めるときに、擬除算を用いての Euclid の互除法によって、$(P_1, P_2, P_3, \cdots, P_k, P_{k+1} = 0)$ と次々と計算していき、$\gcd(P_1, P_2) \neq 1$

*sanuki@math.tsukuba.ac.jp

の場合には、$P_k = cG$ where $c \in K[x, y, \cdots, z]$ なる関係を満たすので、$G = \mathrm{quo}(P_k, c)$ と計算される。ここで、$\mathrm{quo}(P_k, c)$ は P_k を c で割ったときの商を表す。しかし、商を求めるだけなら除数、被除数の全ての情報が必要なわけではない。

例として、除数の次数が100、除数の次数が90の場合には商の次数というのは10となることが分かり、この場合、商を計算するだけなら除数、被除数の高次項(低次項)の10次分の情報だけあれば十分である。

以上に着目したのがPC-PRS算法であり、これによりPRS算法を効率よく計算することが可能であることが知られている。以下はPC-PRS算法のアルゴリズムになる。詳しくは[SS92],[Sa]を見てもらいたい。

アルゴリズム 1 (PC-PRS(P_1, P_2))

Input: Polynomial $P_1, P_2 \in K[y, \cdots, z][x]$
(全次数 $e \leq E$(上限))
Output: $H = \gcd(P_1, P_2)$

$g = \gcd(\mathrm{lc}(P_1), \mathrm{lc}(P_2))$;
$e := 1$;
loop: Calculate PRS $(\tilde{P}_3, \cdots, \tilde{P}_k, \tilde{P}_{k+1} \equiv 0)$;
ただし、$\tilde{P}_i$ は $\mathrm{tdeg}(\mathrm{coef}P_i) > e$ をカットする

If $\deg \tilde{P}_k = 0$ then return *1*
Else $\tilde{P} = g\tilde{P}_k/\mathrm{lc}(\tilde{P}_k)$; ← べき級数除算
$H = \mathrm{pp}(\tilde{P})$;
If $H|P_1$ and $H|P_2$ then return H
Else if $e < E$ then << $e := e + 1$; goto loop>>
Else return 1

但し、PRS $(\tilde{P}_3, \cdots, \tilde{P}_k, \tilde{P}_{k+1} \equiv 0)$ の計算の中で、多項式の乗除算にはべき級数乗除算を使っている。
また、上限 E は次の補題より決めることが出来る。

補題 1 (上限 E の決め方)
$E_i = \mathrm{tdeg}(\mathrm{coef}(P_i)) \quad i = 1, 2, \quad e_i = \mathrm{tdeg}(\mathrm{lc}(P_i)) \quad i = 1, 2$
$e'' = \mathrm{tdeg}(g)$ where $g = \gcd(\mathrm{lc}(P_1), \mathrm{lc}(P_2))$
とするとき、
$E = \min\{E_i - e_i + e'' \mid i = 1, 2\}$ ∎

注意 1 (e をシフトさせる理由)
この場合 e はシフトさせていく必要が無いのだが、近似PC-PRS算法との対称性を見るためにこのように作った。実際は E を用いることで一回でGCDを求めることが出来る。

例 1 (項の数について)
$$F = (1 + x^2 + \sum_{i=1}^{n} y_i{}^i)(2 + x^2 + \sum_{i=1}^{n} y_i)$$
$$G = (1 + x^2 + \sum_{i=1}^{n} y_i)(2 + x - y_1 y_2 y_3)$$
について各 n についてPRS算法で計算したときの項の数を比較する。ここで、各 n について$\gcd(F, G) = 1$であることに注意して頂きたい。

	# term				
	PRS	*PC-PRS*	P_1	P_2	$e = E$
$n = 3$	*324*	*35*	*22*	*14*	*4*
$n = 4$	*1122*	*70*	*33*	*17*	*4*
$n = 5$	*3029*	*126*	*46*	*20*	*4*
$n = 6$	*6828*	*210*	*61*	*23*	*4*
$n = 7$	*13638*	*330*	*78*	*26*	*4*
$n = 8$	*24985*	*495*	*97*	*29*	*4*
$n = 9$	*42872*	*715*	*118*	*32*	*4*

PRS を計算するときにはお互い Coliins の算法 [Co67] を使っています。また表の各数値は PRS 算法で $\gcd(F, G) = 1$ と判定された時の P_k の項の数を表している。 ∎

3 近似PC-PRS算法

ここでは PC-PRS 算法を近似演算に対しても適用できるようにした近似 PC-PRS 算法のアルゴリズムを紹介して計算例を示す。基本となるものは、PC-PRS 算法と近似 PRS を計算する方法である近似 PRS 算法 [ONS91] の 2 つである。簡単にここで使う記号の定義をしてから、近似 PC-PRS 算法のアルゴリズムを示していく。

定義 2 (mmc)
多項式 F の数係数のうち絶対値の最大の数を、ここでは $\mathrm{mmc}(F)$ と表すことにする。

定義 3 (normalization of polynomial(正規化))
多項式 F の正規化とは、$F \to \eta F$ とすることである。ただし、$\mathrm{mmc}(\eta F) = 1$ となるようにする。$\mathrm{Normalize}(F)$ で表すことにする。

定義 4 (prem'(正規化した擬剰余))
多項式 F, G *s.t.* $\deg F \geq \deg G$ について、

$$\mathrm{prem}'(F, G) = \frac{\mathrm{lc}(G)^{\deg F - \deg G + 1} F - QG}{\max\{\mathrm{mmc}(\mathrm{lc}(G)^{\deg F - \deg G + 1}), \mathrm{mmc}(Q)\}}$$

のように定義する。

このとき、近似 PC-PRS 算法のアルゴリズムを次のようにした。

アルゴリズム 2 (**Approx-PC-GCD**$(P_1, P_2; \varepsilon)$)

Input:	Normalized Polynomials P_1, P_2 with $\deg P_1 \geq \deg P_2$
	a small positive number ε_0
Output:	$D = \gcd(P_1, P_2, \varepsilon)$
STEP1:	$k := 2, \gamma := 1; \tilde{P}_3 := \mathrm{prem}'(\tilde{P}_1, \tilde{P}_2)$
STEP2:	if $\mathrm{mmc}(\tilde{P}_{k+1}) < \varepsilon_0$ then goto STEP3;
	if $\deg \tilde{P}_{k+1} = 0$ then goto STEP5;
	$d_k := \deg P_{k-1} - \deg P_k$;
	$k := k+1$; $\gamma := \mathrm{lc}(\tilde{P}_{k-1})^{d_{k-1}} \gamma^{1-d_{k-1}}$;
	$\beta := \mathrm{Normalize}(\mathrm{lc}(\tilde{P}_{k-1})\gamma^{d_{k-1}})$;
	$\tilde{P}_{k+1} := \mathrm{prem}'(\tilde{P}_{k-1}, \tilde{P}_k)/\beta$;goto STEP2;
STEP3:	$\varepsilon := \mathrm{mmc}(\tilde{P}_{k+1})$; $\tilde{P}_k := \mathrm{Normalize}(\tilde{P}_k)$;
	$\tilde{P}_k := \mathrm{Roundoff}(\tilde{P}_k, 2\varepsilon)$;
	$G := \mathrm{Approx} - \gcd(\mathrm{lc}(P_1), \mathrm{lc}(P_2); \varepsilon)$
	$\tilde{P} := G\tilde{P}_k/\mathrm{lc}(\tilde{P}_k)$; ← べき級数除算
	$\tilde{P} := \mathrm{Roundoff}(\tilde{P}, 2\varepsilon)$;
STEP4:	$D' := \mathrm{pp}(\tilde{P}; \varepsilon)$;
	$C_1 := \mathrm{cont}(P_1; \varepsilon)$; $C_2 := \mathrm{cont}(P_2; \varepsilon)$;
	$D := (D' \times \gcd(C_1, C_2; \varepsilon); \varepsilon)$;
	If $D \mid P_1$ and $D \mid P_2$ then return (D, ε);
STEP5:	If $e < E$ then << $e := e+1$; goto Step1 >>;
	Else return $(1, \varepsilon_0)$;

STEP1,STEP2 は近似 PRS を計算しているにすぎずアルゴリズムは PC-PRS 算法とほぼ同じである。また、STEP2 での多項式の乗除算には PC-PRS 算法の時と同様にべき級数乗除算を使っている。しかし、各 $\tilde{P}_i$ について PC-PRS 算法の方では係数の全次数についてにしか制限を掛けなかった。しかし近似 PC-PRS 算法の方では 3 変数以上の場合にそのままでは上手くいかなかった場合があったので (例 2)、次の補題により各従変数についても制限を掛けることにした。

補題 5 (各従変数 u に対する上限 E_u の決め方)

$E_i = \deg_u(\mathrm{coef}(P_i)) \quad i = 1, 2, \quad e_i = \deg_u(\mathrm{lc}(P_i)) \quad i = 1, 2$

$e'' = \deg_u(g) \quad \text{where} \quad g = \gcd(\mathrm{lc}(P_1), \mathrm{lc}(P_2))$

とするとき、

$E_u = \min\{E_i - e_i + e'' \mid i = 1, 2\}$ ∎

この補題により $\tilde{P}_i$ の各従変数 u について次数が E_u より大きいものについては、カットすることにする。

注意 2 (e をシフトさせる理由)

E のままで計算すると、誤差が悪さをして値を出してくれない場合がある (例 2)。なので、効率は少し悪いが $e = 1$ からシフトをさせていくことにする。

例 2 (近似 PC-PRS 算法での 計算例)

$F = (x^2 + y^2 + 0.3z^3 - 1)^2(xy - 0.25) - 10^{-5}xyz$

$G = (x^2 + y^2 + 0.3z^3 - 1)(x - y)^3 - 10^{-5}(x + 1 - z)$

について、精度 $\varepsilon = 10^{-4}$ での近似 GCD を近似 PC-PRS 算法により求める。

$e=1$ のとき　$x^2 - 1.000010$ と計算されるが、割り切れないので $e=2$ にする
$e=2$ のとき　$x^2 + 0.999930y^2 - 1.000010$ と計算されるが、割り切れないので $e=3$ にする
$e=3$ のとき　$x^2 + 0.999930y^2 + 0.2999969z^3 - 1.000010$ と計算され、これは F, G ともに割り切ることが出来るので、これを答えとして返す。

この場合、$e=3$ 以外で計算しようとするときちんとした近似 GCD を求めることはできない。補題 *1* から $E=6$ と計算されるので $e=6$ で設定して計算すると、

$$0.99999x^2 - 0.00027996079995592y^4 + 0.9999200004y^2 + 0.29999399994z^3 - 1$$

というような値が出てきて、これは F, G の近似 GCD とはならない。

また、この近似 GCD 計算において補題 *5* を適用しなかった場合、近似 GCD を計算することは出来ないことを付け加えておく。 ∎

4　実験

実験にあたり、誤差をつけた多項式どうしの近似 GCD を考えるのではなく、$x^2 - y - 1, x^3 - y - 1$ のような根として $x - \sqrt{y+1}, x - \sqrt[3]{y+1}$ のようなべき級数が入る多項式のを考えることにした。精度をつけた場合に、$x - \sqrt{y+1}, x - \sqrt[3]{y+1}$ などはあるところから無視できるので、多項式として考えることが出来るのだが、このようなものについても近似 PC-PRS 算法で計算できるかを検証する。

ここでは単に近似 GCD 計算をするだけではなく、近似無平方分解を用いて各因子をそれぞれ求めれるかを行った (実際にはこの実験の計算では、近似 GCD の計算は 1 回しか行っていないので計算が出来ていれば、近似無平方分解も上手くいっている)。そして出来れば、多変数関数の場合の近接根というものをこれにより考えたかったが、今回はそこまで至らず計算だけに留まった。

実験 1

$F(x,y) = (x^2 - y - 1)(x - 1.09 - 0.514y + 0.178y^2)(x - 1.003 - 0.5038y + 0.127y^2 - 0.068y^3)$
を精度 $\varepsilon = 10^{-2}$ で近似無平方分解をする。右辺の左 2 項は $x - \sqrt{y+1}$ の Taylor 展開したものに誤差を付けたものにしてある。
$\sqrt{y+1} = 1 + 0.5y - 0.125y^2 + 0.0625y^3 + \cdots$

結果

$$\begin{aligned} Q_1 \quad &= x^2 + 0.065644586911383xy^3 + 0.05311446193811xy^2 - 0.013539727391765xy \\ &\quad -0.089809691675558x + 0.0043092117907661y^6 - 0.0047807841345645y^5 \\ &\quad +0.025383933149841y^4 + 0.088229382809088y^3 + 0.057711363787833y^2 \\ &\quad -1.0584931786955y - 1.0898287567993 \\ Q_2 \quad &= x - 0.065644586911383y^3 + 0.12594276903094y^2 - 0.50213013630412y - 1.0015951541622 \end{aligned}$$

ただし、Q_iはi重因子の集りである。この場合$e=3$で計算が終わる。また与えられた多項式は、$F(x,y)=Q_1(x,y)Q_2(x,y)^2+O(0.01(x,y))$と分解することが確かめることが出来る。また、$e=4,5,6$のときの値については、$Q_2$の方にそれぞれ$y$の4,5,6次までの項が付いた値が出てきて、$x-\sqrt{y+1}$をTaylor展開したものに誤差をつけた値がでてきて正しい値ではあるが、この場合本来出てきて欲しくない値である。

実験2

$G(x,y)=(x^3-y-1)(x+0.04y^4-0.06y^3+0.111y^2-0.333y-1)$
について 精度$\varepsilon=10^{-4}$で近似無平方分解する。
右辺の第2式目は$x-\sqrt[3]{y+1}$のTaylor展開したものに誤差をつけた形にしてある。

結果

$$\begin{aligned} Q_1 = \; & x^2 -0.041153xy^4+0.061728xy^3-0.111111xy^2+0.333333xy+x+0.001693y^8-0.005079y^7 \\ & +0.012954y^6-0.041152y^5-0.028809y^4+0.049382y^3-0.111111y^2+0.666666y+1 \\ Q_2 = \; & x+0.040576y^4-0.060864y^3+0.111055y^2-0.333166y-1 \end{aligned}$$

のように分かれることが確認できる。実験1と同様に値はきちんと出すことができた。

5 まとめ

PC-PRS算法を近似計算にも適用できるよう新しく定義した近似PC-PRS算法を用いることで近似GCD計算、近似無平方分解が出来ることは確かめることができた。但し、計算精度・計算効率の問題などをここでは触れることが出来なかったが、近似PRS算法よりもよくなることは期待される。一方で、従変数の次数制限のところについてはシフトをさせているために無駄な計算をしているところもあり、まだまだ改善の余地はある。これらを明確にすることは、近似PC-PRS算法を広める上でやらなければならない仕事である。

参 考 文 献

[Co67] George E. Collins, *Subresultants and Reduced Polynomial Remainder Sequence*, J. of ACM,1967

[ONS91] M.Ochi,M.Noda and T,Sasaki, *Approximate Greatest Common Divisor of Multivariate Polynomials and Its Application to Ill-Conditioned Systems of Algebraic Equation*, Jour. Infor. Proc.,1991,292-300

[SN89] T.Sasaki and M.Noda, *Approximate Square-free Decomposition and Root-finding of Ill-conditoned Algebraic Equation*, Jour. Infor. Proc.,1989

[Sa] 佐々木、今井、他, 計算代数と計算幾何, 岩波講座・応用数学

[SS92] T.Sasaki and M.Suzuki, *Three New Algrithms for Multivariate Polynomial GCD*, Jour.Symb.Comp.,1992

[ZN99] LiHong Zhi and M.Noda, *Approximate GCD of Multivariate Polynomials*, 京都数理解析研究所講義録1138