数理解析研究所講究録
第1930巻 2015年 60-72

近似GCD算法GPGCDの新たな実験結果について
GPGCD, an Iterative Method for Calculating Approximate GCD of Univariate Polynomial: New Test Results

照井 章*
Akira Terui
筑波大学 数理物質系
Faculty of Pure and Applied Sciences, University of Tsukuba

Abstract

我々は，これまでに，1変数多項式に対する近似GCDの反復算法であるGPGCD法を提案している．これは，与えられた多項式および次数に対し，可能な限り小さな摂動を加えることにより，与えられた次数のGCDおよびそのための摂動を求める算法である．GPGCD算法は，与えられた問題を制約つき最小化問題に帰着させ，これを，勾配射影法の一般化の一つである「修正Newton法」と呼ばれる反復算法で解くものである．本稿では，GPGCD算法と，同じく最適化法に基づく近似GCD算法である，STLN (Structured Total Least Norm) に基づく算法と，UVGCD法に対し，新たな例題を多数追加して動作の比較を行った実験結果を報告する．

Abstract

The GPGCD method, proposed by the author, is an iterative method for calculating approximate greatest common divisor (GCD) of univariate polynomials. For a given pair of polynomials and a degree, our algorithm finds a pair of polynomials which has a GCD of the given degree and whose coefficients are perturbed from those in the original inputs, making the perturbations as small as possible, along with the GCD. In our GPGCD method, the problem of approximate GCD is transferred to a constrained minimization problem, then solved with the so-called modified Newton method, which is a generalization of the gradient-projection method, by searching the solution iteratively. In this paper, we report new results of experiments that are carried out for our method and two other approximate GCD algorithms: the method based on the Structured Total Least Norm (STLN) and the UVGCD method.

1 はじめに

多項式や行列を対象とする代数計算(数式処理) において，数式・数値融合算法は，最近，注目を集めている．中でも，近似代数計算 [24], すなわち，与えられた問題自体には代数的関係が存在しないが，その近傍に, 代数的関係をもつものが存在した場合に，そのような代数的関係をもつ系を, もとの系からの摂動をなるべく小さく保ちながら探索するような計算は，従来の計算代数の算法が適用不可能もしくは困難であるような，浮動小数係数多項式などの問題に，計算代数的手法を適用可能なものとして，期待されている．

近似最大公約子 (GCD) は，近似代数計算の中でも最も古くから活発に研究が行われてきた問題の一つである．これは，多項式の組(一般的には互いに素) と，次数 d が与えられたときに，与えられた多項式の

*terui@math.tsukuba.ac.jp

係数に摂動を加え，d 次の GCD をもつような系を探索し，見つかった GCD を，与えられた多項式の近似 GCD と呼ぶものである．

近似 GCD の計算においては，これまでに様々な算法が提案されており，それらの中には，多項式剰余列 (PRS) に基づく算法 ([1], [13], [14]), Sylvester の終結式行列の特異値分解 (SVD) に基づく算法 ([4], [6]), Sylvester の終結式行列の QR 分解に基づく算法 ([5], [20], [23]), Padé 近似に基づく算法 [10], 最適化法に基づく算法 ([3], [8], [9], [22]) などがある．さらに，種々の悪条件問題に対する解法 ([5], [12], [27]) も議論されている．

我々は，これまでの研究で，GPGCD 法と呼ばれる反復算法を提案した ([16], [17], [18], [25]). これは，与えられた GCD の計算問題を制約つき最適化問題に帰着させ，勾配射影法の一般化と位置づけることのできる修正 Newton 法 ([15], [28, 第 4 章]) を用いて局所的最小解を探索するものである．これまでの実験結果では，最小二乗法の一つである STLN (Structured Total Least Norm) 法に基づく算法 ([8], [9]) と同程度の摂動で近似 GCD を探索でき，かつ大幅な効率化が図られることを示した．本稿では，GPGCD 法の一連の算法のうち，実係数もしくは複素係数の 2 つの入力多項式に対する算法 ([16], [17]) に対し，STLN 法に基づく算法の他に UVGCD 法 [21] も比較対象に加え，さらに多くの例題を加えてこれらの近似 GCD 算法の比較実験を行った結果を，論文 [18] の内容をもとに報告する．

2 近似 GCD 算法 GPGCD の概要

本章では，近似 GCD 算法 GPGCD の概要として，実係数多項式に対する算法の概要を，筆者による別稿 [26] の内容をもとに述べる．(複素係数多項式に対する算法については別稿 [25]，全体の詳細については論文 [18] を参照.)

2.1 近似 GCD 問題と制約つき最小化問題への帰着

$F(x), G(x)$ を互いに素な実係数 1 変数多項式の組とし，次式で与えられるものとする．

$$F(x) = f_m x^m + f_{m-1} x^{m-1} + \cdots + f_0, \quad G(x) = g_n x^n + g_{n-1} x^{n-1} + \cdots + g_0. \tag{1}$$

($m \geq n > 0$, $f_m \neq 0$, $g_n \neq 0$ とする.) 与えられた次数 d $(n \geq d > 0)$ に対し，$F(x)$ と $G(x)$ の係数に摂動を加えることにより，次式のような $\tilde{F}(x)$ と $\tilde{G}(x)$ を計算することを考える．

$$\tilde{F}(x) = F(x) + \Delta F(x) = H(x) \cdot \bar{F}(x), \quad \tilde{G}(x) = G(x) + \Delta G(x) = H(x) \cdot \bar{G}(x). \tag{2}$$

ここに，$\Delta F(x), \Delta G(x)$ は，次数がそれぞれ $F(x), G(x)$ の次数を超えないような多項式，$H(x)$ は d 次の多項式で，$\bar{F}(x)$ と $\bar{G}(x)$ は互いに素とする．式 (2) をみたす $\tilde{F}, \tilde{G}, \bar{F}, \bar{G}, H$ が計算されたとき，H を F と G の**近似 GCD** と呼ぶ．本稿では，**与えられた次数 d に対し，摂動のノルム $\|\Delta F(x)\|_2^2 + \|\Delta G(x)\|_2^2$ をなるべく小さく保ちつつ，F と G の d 次の近似 GCD H を探索する問題を解く**．

$\tilde{F}(x), \tilde{G}(x)$ を，それぞれ

$$\tilde{F}(x) = \tilde{f}_m x^m + \cdots + \tilde{f}_0 x^0, \quad \tilde{G}(x) = \tilde{g}_n x^n + \cdots + \tilde{g}_0 x^0 \tag{3}$$

と表す．$\tilde{F}$ と $\tilde{G}$ が d 次の GCD をもつとき，部分終結式の理論により，$\tilde{F}$ と $\tilde{G}$ の $d-1$ 次の部分終結式は

0になる．ゆえに，$\tilde{F}$と$\tilde{G}$の$d-1$次の部分終結式行列

$$N_{d-1}(\tilde{F},\tilde{G}) = \begin{pmatrix} \tilde{f}_m & & & \tilde{g}_n & & \\ \vdots & \ddots & & \vdots & \ddots & \\ \tilde{f}_0 & & \tilde{f}_m & \tilde{g}_0 & & \tilde{g}_n \\ & \ddots & \vdots & & \ddots & \vdots \\ & & \tilde{f}_0 & & & \tilde{g}_0 \end{pmatrix} \tag{4}$$

$$\underbrace{\qquad}_{n-d+1} \qquad \underbrace{\qquad}_{m-d+1}$$

（式(4)のように，$\tilde{F}$の係数を$n-d+1$列，$\tilde{G}$の係数を$m-d+1$列並べた行列）はランクがfull rankから1落ちるため，互いに素な多項式$A(x)$と$B(x)$が存在して

$$A\tilde{F} + B\tilde{G} = 0 \tag{5}$$

（ただし$\deg(A) = n-d$, $\deg(B) = m-d$）をみたす．ゆえに，本稿で考える問題は，与えられた$F(x)$, $G(x)$, dに対し，方程式(5)をみたすような$\Delta F(x)$, $\Delta G(x)$, $A(x)$, $B(x)$で，$\|\Delta F\|_2^2 + \|\Delta G\|_2^2$がなるべく小さくなるものを探索する問題に帰着される．

$\|\Delta F\|_2^2 + \|\Delta G\|_2^2$は

$$\|\Delta F\|_2^2 + \|\Delta G\|_2^2 = (\tilde{f}_m - f_m)^2 + \cdots + (\tilde{f}_0 - f_0)^2 + (\tilde{g}_n - g_n)^2 + \cdots + (\tilde{g}_0 - g_0)^2 \tag{6}$$

と表される．一方，方程式(5)は，$A(x)$, $B(x)$を，それぞれ$A(x) = a_{n-d}x^{n-d} + \cdots + a_0x^0$, $B(x) = b_{m-d}x^{m-d} + \cdots + b_0x^0$と表すことにより，

$$N_{d-1}(\tilde{F},\tilde{G}) \cdot {}^t(a_{n-d}, \ldots, a_0, b_{m-d}, \ldots, b_0) = \mathbf{0} \tag{7}$$

と表される．ゆえに，方程式(7)は，$\tilde{f}_m, \ldots, \tilde{f}_0, \tilde{g}_n, \ldots, \tilde{g}_0, a_{n-d}, \ldots, a_0, b_{m-d}, \ldots, b_0$を変数とする$m+n-d+1$個の連立方程式

$$g_1 = \tilde{f}_m a_{n-d} + \tilde{g}_n b_{m-d} = 0, \ldots, g_{m+n-d+1} = \tilde{f}_0 a_0 + \tilde{g}_0 b_0 = 0 \tag{8}$$

と表される（式(7)における第j行の方程式をg_jとおいた）．さらに，$A(x)$と$B(x)$に対し，$\|A(x)\|_2^2 + \|B(x)\|_2^2 = 1$なる制約を加える．これを

$$g_0 = a_{n-d}^2 + \cdots + a_0^2 + b_{m-d}^2 + \cdots + b_0^2 - 1 = 0 \tag{9}$$

とし，方程式(8)に加える．

ここで，これまでの多項式の係数を表す変数を，それぞれ$\mathbf{x} = (x_1, \ldots, x_{2(m+n-d+2)})$に置き換える．すると，式(6)および方程式(8)（方程式(9)を含む）は，それぞれ

$$f(\mathbf{x}) = (x_1 - f_m)^2 + \cdots + (x_{m+1} - f_0)^2 + (x_{m+2} - g_n)^2 + \cdots + (x_{m+n+2} - g_0)^2, \tag{10}$$

$$\mathbf{g}(\mathbf{x}) = {}^t(g_0(\mathbf{x}), g_1(\mathbf{x}), \ldots, g_{m+n-d+1}(\mathbf{x})) = \mathbf{0} \tag{11}$$

と表される．

以上により，本稿で考える近似GCDの問題は，以下の制約つき最小化問題に帰着される．

問題 1

方程式(11) ($\mathbf{g}(\mathbf{x}) = \mathbf{0}$)の下で，式(10)の$f(\mathbf{x})$を最小化せよ．∎

2.2 勾配射影法と修正 Newton 法

本節では，$\mathbf{x} \in \mathbb{R}^n$ に対し，制約条件 $\mathbf{g}(\mathbf{x}) = \mathbf{0}$（ ただし $\mathbf{g}(\mathbf{x}) = {}^t(g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_m(\mathbf{x}))$, $m \leq n$ とする）のもとで，目的関数 $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ を最小化する問題を考える．許容領域を $V_{\mathbf{g}} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{g}(\mathbf{x}) = \mathbf{0}\}$ とするとき，$\mathbf{x} \in V_{\mathbf{g}}$ に対し，ヤコビ行列 $J_{\mathbf{g}}(\mathbf{x}) = \left(\dfrac{\partial g_i}{\partial x_j}\right)$ が full rank, すなわち $\mathrm{rank}(J_{\mathbf{g}}(\mathbf{x})) = m$ が成り立つならば，$V_{\mathbf{g}}$ は $\mathbb{R}^n$ の $n-m$ 次元微分多様体となる．このとき，$V_{\mathbf{g}}$ 上で f を最小化する局所解の候補として，“1 次の必要条件” [28, 第 1 章] をみたす点を探索する．

勾配射影法 [11] は，$V_{\mathbf{g}}$ 上の点 $\mathbf{x}_k$ に対し，「 射影」と「 引き戻し」と呼ばれる計算のステップを繰り返すことにより，$V_{\mathbf{g}}$ 上で f を最小化する局所解を探索する．一方，修正 Newton 法 ([15], [28, 第 4 章]) は，勾配射影法の拡張の一つと位置づけられ，Newton 法で用いられる Lagrange 関数の Hesse 行列を修正することにより，さまざまな解法を導出するものである．勾配射影法と同値な解法では，$V_{\mathbf{g}}$ 上の点 $\mathbf{x}_k$ に対し，探索方向 $\mathbf{d}_k$ は，$\mathbf{x}_k$ が許容領域 $V_{\mathbf{g}}$ 内にあるならば f の最急降下方向 $-\nabla f(\mathbf{x}_k)$ になり，$\mathbf{x}_k$ が許容領域から外れると，これに $\mathbf{x}_k$ を許容領域 $V_{\mathbf{g}}$ に引き戻す方向が加わる．この意味で，修正 Newton 法は，1 回の反復計算で，勾配射影法における「 射影」と「 引き戻し」の 2 つのステップを同時に実行するという 特徴をもつ．

2.3 近似 GCD の算法

実際の近似 GCD の算法を構築するにあたっては，主に以下の問題を考慮する必要がある．

1. 反復計算の過程において，ヤコビ行列 $J_{\mathbf{g}}(\mathbf{x})$ が full rank である（ ランクが $J_{\mathbf{g}}(\mathbf{x})$ の行数に等しい）こと．
2. 反復計算の初期値の設定．
3. GCD となる多項式の実際の計算．

以上に関する議論の詳細は論文 [18] に譲る．実際の近似 GCD の算法は以下の通り である．

アルゴリズム 1 (GPGCD: 修正 Newton 法（勾配射影法）に基づく近似 GCD の算法)

入力 $F(x), G(x) \in \mathbb{R}[x]$（ ただし $\deg(F) \geq \deg(G) > 0$），$d \in \mathbb{N}$ (ただし $d \leq \deg(G)$): 近似 GCD の次数，$\varepsilon > 0$: 残差のしきい値，$u \in \mathbb{N}$: 反復回数のしきい値．

出力 $\tilde{F}(x), \tilde{G}(x), H(x) \in \mathbb{R}[x]$. $\tilde{F}$, $\tilde{G}$ は，それぞれ F, G の係数に摂動を与えたもので，d 次の GCD H をもつ．

Step 1 **[初期値の設定]** 反復計算の初期値 $\mathbf{x}_0$ を与える（ 詳細は論文 [18] を参照）．

Step 2 **[反復計算]** 修正 Newton 法により，式 (10) および方程式 (11) に対し，制約条件 $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ の下で $\bar{f}(\mathbf{x}) = \frac{1}{2} f(\mathbf{x})$ の最小解を求める（ $\bar{f}(\mathbf{x}) = \frac{1}{2} f(\mathbf{x})$ とおくことについての詳細は論文 [18] を参照）． 探索方向 $\mathbf{d}_k$ が $\|\mathbf{d}_k\|_2 < \varepsilon$ をみたすか，反復回数が u を超えた段階で，反復計算を終了する．

Step 3 **[$\tilde{F}$, $\tilde{G}$, H の計算]** 実際の GCD となる $H(x)$ と，H を GCD としてもつ $\tilde{F}(x)$, $\tilde{G}(x)$ の計算を行い，$\tilde{F}$, $\tilde{G}$, H を返す．もし Step 2 が u 回の反復で終了しなかった場合は，その旨をユーザに報告する．

3 実験

我々は，GPGCD 法(アルゴリズム 1) を数式処理システム Maple 上に実装した [1)]．本稿では，以下の実験例に対し，GPGCD 法を，最小二乗法の一つである STLN (Structured Total Least Norm) 法に基づく算法 [9](以下 "STLN 法" と記す) および UVGCD 法 [21] と比較した実験結果について述べる．

1. (第 3.1 節) 所定の条件のもとで係数を無作為に与えた多数の実験対象の多項式.
2. (第 3.2 節) 悪条件な多項式，高次多項式，1 個または多数の重複零点をもつような多項式等 ([2], [22]).

実験 1 においては，実係数多項式，複素係数多項式とも実験の対象にしたが， 実験 2 においては，実係数多項式のみを対象にした点に注意する．

本稿の実験に用いた計算機環境は以下の通りである．Intel Core2 Duo Mobile Processor T7400 (in Apple MacBook "Mid-2007" model) 2.16 GHz, RAM 2GB, Mac OS X 10.6. ソフトウェア環境は数式処理システム Maple 15 を用い，システム変数 `Digits=15` とし，ハードウェアの浮動小数演算を用いた．

3.1 実験 1: 無作為に生成した多数の多項式に対する近似 GCD の計算

本実験では，アルゴリズム 1 を，STLN 法および UVGCD 法と比較した．実装は，各著者がそれぞれ自身で Maple に行ったものを用い，実係数多項式，複素係数多項式の両方に対する計算を行った．我々によるアルゴリズム 1 の実装では，修正 Newton 法に基づく最適化を用いた．STLN 法においては，複数個(一般に 3 個以上) の入力多項式に対して近似 GCD を求めるプロシージャ `R_con_mulpoly`(実係数多項式用) および `C_con_mulpoly`(複素係数多項式用) を用いた(以下同様)．UVGCD 法においては，実多項式および複素多項式の入力に対して近似 GCD を求めるプロシージャ `uvgcd` を用いた(以下同様)．

本実験における入力多項式は以下の要領で作成した． まず，モニックな m 次多項式 $F_0(x)$ と n 次多項式 $G_0(x)$ を，次数 d 次の GCD をもつように生成した．このとき，GCD および各多項式の余因子の係数 c は $c \in [-10, 10]$ をみたす浮動小数で無作為に与えた．これらに加え，"ノイズ" として，$m-1$ 次の多項式 $F_{\mathrm{N}}(x)$ と $n-1$ 次多項式 $G_{\mathrm{N}}(x)$ を生成した．係数は，$F_0(x)$, $G_0(x)$ の場合と同様の要領で与えた．そして，実験対象の多項式 $F(x)$, $G(x)$ を次式で与えた．

$$F(x) = F_0(x) + \frac{e_F}{\|F_{\mathrm{N}}(x)\|_2} F_{\mathrm{N}}(x), \quad G(x) = G_0(x) + \frac{e_G}{\|G_{\mathrm{N}}(x)\|_2} G_{\mathrm{N}}(x).$$

ここで，"ノイズ" の多項式 $F_{\mathrm{N}}(x)$ および $G_{\mathrm{N}}(x)$ は，それぞれの 2-ノルムが e_F および e_G に等しくなるように規格化する．本稿の実験では $e_F = e_G = 0.1$ とした．

本実験においては，入力多項式に対し，もう一つ，以下の条件を課した: 入力多項式 $F(x)$ と $G(x)$ が入力次数 d を上回る次数の GCD を持たないよう，d 次部分終結式行列 $N_d(F, G)$(式 (4) 参照) の最小特異値が 1 以上になる組のみを採用した [2)].

各実験例においては，入力多項式を 100 個ずつ，上記の要領で作成した．アルゴリズム 1 においては $u = 200$, $\varepsilon = 1.0 \times 10^{-8}$ とおいた．`R_con_mulpoly` および `C_con_mulpoly` においては，許容度を $e = 1.0 \times 10^{-8}$ と

[1)] 筆者による実装を "Project Hosting on Google Code" [19] にて公開している．

[2)] 我々の最初の実験結果 [17, Section 5.2] では，実験に用いたいくつかの入力多項式において，GPGCD 法による近似 GCD が十分小さな摂動の範囲内で求まらない場合があった．後になって，これらの場合の入力多項式を詳しく調べたところ，これらの入力多項式では，偶然，近似 GCD の次数が d を上回っていたことがわかった．そこで，本稿における実験では，上記の条件を新たに加えて実験例を生成した上で実験を行ったところ，最初の実験の際に起きたような現象は見られなかった．

Ex.	m,n	d	Perturbation (12)			Time (sec.)			#Iterations	
			STLN	UVGCD	GPGCD	STLN	UVGCD	GPGCD	STLN	GPGCD
1	10,10	5	$5.64e-2$	$1.79e-1$	$5.64e-2$	0.38	0.64	0.04	4.46	4.50
2	20,20	10	$6.22e-2$	$1.85e-1$	$6.22e-2$	1.16	0.86	0.06	4.40	4.40
3	30,30	15	$6.65e-2$	$1.87e-1$	$6.65e-2$	2.43	1.34	0.10	4.37	4.46
4	40,40	20	$6.48e-2$	$1.96e-1$	$6.48e-2$	4.05	2.09	0.13	4.11	4.15
5	50,50	25	$6.91e-2$	$1.91e-1$	$6.91e-2$	6.30	3.34	0.19	4.03	4.16
6	60,60	30	$6.75e-2$	$1.94e-1$	$6.75e-2$	9.09	4.37	0.26	4.00	4.18
7	70,70	35	$6.89e-2$	$2.08e-1$	$6.89e-2$	12.47	5.71	0.35	3.96	4.13
8	80,80	40	$6.78e-2$	$1.91e-1$	$6.78e-2$	16.95	7.95	0.44	3.16	4.11
9	90,90	45	$6.92e-2$	$1.95e-1$	$6.92e-2$	22.09	10.20	0.57	3.96	4.10
10	100,100	50	$6.98e-2$	$1.95e-1$	$6.98e-2$	27.48	13.02	0.69	3.88	4.09

表 1: 実係数多項式に対する近似 GCD の計算実験結果．詳細は第 3.1 節を参照．

Ex.	m,n	d	Perturbation (12)			Time (sec.)			#Iterations	
			STLN	UVGCD	GPGCD	STLN	UVGCD	GPGCD	STLN	GPGCD
1	10,10	5	$5.92e-2$	$1.54e-1$	$5.92e-2$	1.58	0.64	0.11	4.50	4.46
2	20,20	10	$6.40e-2$	$1.41e-1$	$6.40e-2$	5.34	1.31	0.20	4.30	4.30
3	30,30	15	$6.63e-2$	$1.40e-1$	$6.63e-2$	11.63	2.12	0.35	4.21	4.24
4	40,40	20	$6.61e-2$	$1.34e-1$	$6.61e-2$	21.57	3.51	0.55	4.15	4.13
5	50,50	25	$6.86e-2$	$1.48e-1$	$6.86e-2$	34.23	5.03	0.83	4.06	4.10
6	60,60	30	$6.86e-2$	$1.51e-1$	$6.86e-2$	50.40	7.39	1.16	4.02	4.05
7	70,70	35	$6.94e-2$	$1.41e-1$	$6.94e-2$	69.54	10.31	1.56	3.93	4.05
8	80,80	40	$6.85e-2$	$1.44e-1$	$6.85e-2$	93.77	14.01	2.07	3.91	4.07
9	90,90	45	$6.84e-2$	$1.52e-1$	$6.84e-2$	122.97	18.30	2.65	3.90	4.04
10	100,100	50	$6.94e-2$	$1.65e-1$	$6.94e-2$	157.02	23.72	3.37	3.86	4.04

表 2: 複素係数多項式に対する近似 GCD の計算実験結果．詳細は第 3.1 節を参照．

おいた．uvgcd においては，許容度の初期値を $\delta = 1.0 \times 10^{-2}$ とおき，我々が与えた次数の近似 GCD が得られるまで，許容度の値を変化させた．

実験結果を表 1 および表 2 に示す．表 1 が実係数多項式に対する実験結果で，表 2 が複素係数多項式に対する実験結果である．m, n はそれぞれ多項式 F および G の次数を表し．d はそれらの近似 GCD の次数を表す．列 "STLN" は STLN 法の実験結果，列 "UVGCD" は UVGCD 法の実験結果，"GPGCD" は GPGCD 法(アルゴリズム 1) の実験結果を表す．列 "Perturbation" は，摂動を与えて近似 GCD の存在を検出した多項式の組 $(\tilde{F},\tilde{G})$ の，与えられた入力多項式の組 (F,G) からの摂動量を

$$\sqrt{\|\tilde{F}-F\|_2^2+\|\tilde{G}-G\|_2^2} \tag{12}$$

で表したものである．ここに，"aeb"(a と b は実数値) は $a \times 10^b$ を表す．列 "#Iterations" は近似 GCD の計算に要した反復回数を表す．列 "Time" は計算時間を秒単位で表す．(ここで，UVGCD の計算時間は，許容度 δ の値をを変化させた，いわゆる "試行錯誤" の時間は含まず，最終的に d 次の近似 GCD の計算に成功した時の計算時間を表すことに注意する．)

実験結果より，以下のことがわかる．近似 GCD を検出した時の多項式の摂動の大きさは，GPGCD 法の

場合はSTLN法の場合とほぼ同程度で，これらはUVGCDの場合の1/10程度の大きさである．計算時間（速さ）については，GPGCD法の場合，各計算例において，STLN法の10倍から30倍程度，UVGCD法の6倍から10倍程度の速さで，GPGCD法が他の算法に比べて極めて速い．

注意 1
本実験において，GPGCD法の実装は，入力する多項式の個数が2個の場合に限られるのに対し，STLN法の実装は，3個以上の多項式に対しても近似GCDを計算できるような実装であるので，計算結果の比較には注意が必要である．なお，Kaltofen [7] は，彼らが作成した別の実装として，入力を2個の実係数多項式に限る実装 [8] を用いた実験結果を筆者に報告した．それによると，入力多項式の次数がともに100次で，近似GCDの次数が50次の場合，実験結果（実行環境: ThinkPad 1.8 GHz, RAM 1GB）は，反復回数が2回で計算時間は約9秒とのことである．この結果は，GPGCD法の効率を見極める上で参考になるであろう．

3.2 実験 2: 悪条件な多項式等に対する近似GCDの計算

本実験では，悪条件多項式を含む実験例 ([2], [22]) に対し，アルゴリズム1とSTLN法およびUVGCD法との比較を以下の通り行った．

本実験において，GPGCD法とSTLN法では近似GCDの次数を与えるのに対し，UVGCD法においては，与えられる許容度δに対し，同算法内で近似GCDの次数の見積もりを行う点に注意する．また，本節のいくつかの実験においては，実験結果として，入力多項式に加えられた摂動 (12) に代わり，計算される近似GCDの，初期値として与えたGCDからの相対誤差 (14) を計測している点に注意する．これは，人為的な摂動を加えていないような多項式において，計算された近似GCDの，与えられたGCDに対する“近さ”を比較するためである．

本節を通して，実験結果を示す表の列 “STLN”, “UVGCD”, “GPGCD”, “Perturbation”, “Time” は，それぞれ前節のそれらと同じものを表す．

例 1
悪条件多項式の例 [22, Test 1]. n を正の偶数，$k = n/2$ とおき，$p_n = u_n v_n$ および $q_n = u_n w_n$ を次式で定める．

$$
\begin{aligned}
u_n &= \prod_{j=1}^{k}[(x - r_1\alpha_j)^2 + r_1^2\beta_j^2], \quad v_n = \prod_{j=1}^{k}[(x - r_2\alpha_j)^2 + r_2^2\beta_j^2], \\
w_n &= \prod_{j=k+1}^{n}[(x - r_1\alpha_j)^2 + r_1^2\beta_j^2], \quad \alpha_j = \cos\frac{j\pi}{n}, \quad \beta_j = \sin\frac{j\pi}{n},
\end{aligned}
\tag{13}
$$

ここに $r_1 = 0.5, r_2 = 1.5$ である．p_n と q_n の零点は，それぞれ半径 r_1, r_2 の円周上に位置する．実験は $n = 6, \ldots, 20$ の範囲で2刻みに行った．

実験結果を表3に示す．“Relative error of GCD” は

$$
\frac{\|\bar{u}_n(x) - u_n(x)\|_2}{\|u_n(x)\|_2} \tag{14}
$$

によって求めた値である．ここに，u_n は式 (13) によってあらかじめ与えられたGCD，$\bar{u}_n$ は計算された近似GCDを表す．表中，(*1) は，STLN法において，実装内に定義された反復回数のしきい値50回の範囲内で計算結果が収束しなかったことを示し，(*2) は，GPGCD法において，反復回数100回の範囲内で計算結果が収束しなかったことを示す．実験結果から，STLN法とGPGCD法においては，n の増加に伴

n	Relative error of GCD (14)		
	STLN	UVGCD	GPGCD
6	$1.04e-14$	$4.60e-15$	$3.68e-15$
8	$3.98e-13$	$7.90e-13$	$4.30e-13$
10	$1.08e-10$	$7.89e-12$	$1.08e-10$
12	$2.87e-10$	$2.95e-11$	$2.94e-10$
14	$3.10e-9$	$3.65e-10$	$3.14e-9$
16	$6.22e-9$ (*1)	$3.83e-10$	$8.00e-9$
18	$1.38e-6$ (*1)	$9.68e-9$	$1.36e-6$
20	$6.95e-6$ (*1)	$1.21e-8$	$7.11e-6$ (*2)

表 3: 多項式 (13) に対する実験結果. 詳細は例 1 を参照.

い，近似 GCD の精度が低下(相対誤差が増加) していることがわかる．一方，UVGCD 法においては，大きな n の値に対し，他の算法と比較して近似 GCD の精度がより高いことがわかる．

例 2

悪条件多項式の例 [22, Test 2]. $p(x)$, $q(x)$ を次式で定める．

$$p(x)=\prod_{j=1}^{10}(x-x_j),\quad q(x)=\prod_{j=1}^{10}(x-x_j+10^{-j}),\quad x_j=(-1)^j(j/2). \tag{15}$$

ここで，q の零点は，j の増加に伴い，p の最も近い零点との差が $1/10, 1/10^2, \ldots$ と減少している．実験では，近似 GCD の次数 d を 1 から 10 まで 1 ずつ増加させながら近似 GCD の計算を行った．

実験結果を表 4 および表 5 に示す．本実験においては，p と q が厳密には互いに素であることから，入力多項式に対して加えられた摂動 (12) を計測した．なお，UVGCD 法に対する実験結果は表 5 に示し，STLN 法および GPGCD 法に対する実験結果を表す表 4 と区別している．これは，STLN 法および GPGCD 法においては近似 GCD の次数 d を与えているのに対し，UVGCD 法においては，目標とする次数の近似 GCD が求まるような許容度 δ を与えているためである．表 4 中，(*1) は，STLN 法において，実装内に定義された反復回数のしきい値 50 回の範囲内で計算結果が収束しなかったことを示す．

実験結果を見ると，$d \geq 6$ に対しては，すべての算法において，入力多項式にほぼ同程度の摂動を加えることにより，近似 GCD を検出していることがわかる．しかし，より小さな d の値に対しては，UVGCD 法における摂動の大きさが他の算法におけるそれらに比べて相当小さく，STLN 法がそれに続いていることがわかる．

例 3

高次多項式の例 [22, Test 3]. p_n, q_n を次式で定める．

$$p_n=u_n v,\quad q_n=u_n w,\quad v(x)=\sum_{j=0}^{3}x^j,\quad w(x)=\sum_{j=0}^{3}(-x)^j. \tag{16}$$

ここに，$u_n(x)$ は p_n と q_n の n 次の GCD となる多項式で，各項の係数は $[-5,5]$ の範囲の整数を無作為に与えたものである．また，$v(x)$ と $w(x)$ はともに係数を固定した多項式である．

実験結果を表 6 に示す．本実験では，計算された近似 GCD の相対誤差 (14) を計測している．本実験では，算法間における計算時間の差が他の実験に比べて著しく大きかったので，計算時間も示した．計算され

d	Perturbation (12)	
	STLN	GPGCD
1	$5.17e{-1}$ (*1)	$3.21e3$
2	$6.95e{-4}$ (*1)	$3.06e0$
3	$1.97e{-5}$	$1.26e0$
4	$2.89e{-6}$	$2.25e{-1}$
5	$5.28e{-5}$	$4.75e{-1}$
6	$2.15e{-3}$	$2.16e{-3}$
7	$8.34e{-2}$	$8.34e{-2}$
8	$2.04e0$	$2.04e0$
9	$4.70e1$	$4.70e1$
10	$7.73e2$	$7.73e2$

表 4: 多項式 (15) (STLN 法および GPGCD 法) に対する実験結果. 詳細は例 2 を参照.

た近似 GCD の精度を見ると, UVGCD の場合が精度が最も高く, ついで STLN 法, GPGCD 法の順に続く. 一方, 計算時間を見ると, GPGCD 法が他の算法に比べて効率的であることがわかる.

例 4

大きな多重度の零点をもつ多項式の例 [2, Example 4.5]. 正整数 k に対し, $u_k(x), v_k(x)$ を次式で定める.

$$u_k(x) = (x^3 + 3x - 1)(x-1)^k, \quad v_k(x) = u'(x). \tag{17}$$

$u_k(x)$ と $v_k(x)$ の GCD は $w_k(x) = (x-1)^{k-1}$ である点に注意する.

実験結果を表 7 に示す. 表中, (*1) および (*2) は例 1 と同様, (*1) は STLN 法において実装内に定義された反復回数のしきい値 50 回の範囲内で計算結果が収束しなかったことを示し, (*2) は GPGCD 法において反復回数 100 回の範囲内で計算結果が収束しなかったことを示す.

GPGCD 法および STLN 法においては, $k = 35$ および 45 の場合, 計算される近似 GCD の精度が低下している. 一方で, UVGCD 法においては, それらの大きな k の値に対しても, 計算される近似 GCD の精度が高いことがわかる.

例 5

複数個の大きな多重度の零点をもつ多項式の例 [22, Test 6]. 非負整数 $m_1, \ldots, m_4$ に対し, $p_{[m_1,m_2,m_3,m_4]}(x)$, $q_{[m_1,m_2,m_3,m_4]}(x)$ を次式で定める.

$$\begin{aligned} p_{[m_1,m_2,m_3,m_4]}(x) &= (x-1)^{m_1}(x-2)^{m_2}(x-3)^{m_3}(x-4)^{m_4}, \\ q_{[m_1,m_2,m_3,m_4]}(x) &= \frac{d}{dx} p_{[m_1,m_2,m_3,m_4]}(x), \end{aligned} \tag{18}$$

$p_{[m_1,m_2,m_3,m_4]}(x)$ と $q_{[m_1,m_2,m_3,m_4]}(x)$ の GCD は $(x-1)^{m'_1}(x-2)^{m'_2}(x-3)^{m'_3}(x-4)^{m'_4}$ ($m'_j = \max\{m_j - 1, 0\}$, $j = 1, \ldots, 4$) である点に注意する.

実験結果を表 8 に示す. 表中, (*1) および (*2) は例 1 および例 4 と同様, (*1) は STLN 法において実装内に定義された反復回数のしきい値 50 回の範囲内で計算結果が収束しなかったことを示し, (*2) は GPGCD 法において反復回数 100 回の範囲内で計算結果が収束しなかったことを示す. さらに (*3) は, GPGCD 法において, 反復公式に用いられる連立 1 次方程式 [18, Eq. 30] の解の大きさが過大になり, 計算が異常終了したことを示す.

UVGCD		
δ	d	Perturbation (12)
$1.0e-11$	1	$8.02e-10$
$1.0e-10$	2	$3.27e-8$
$1.0e-9$	3	$6.03e-7$
$1.0e-8$	4	$1.99e-5$
$1.0e-7$	5	$3.45e-4$
$1.0e-6$	5	$3.45e-4$
$1.0e-5$	6	$9.61e-3$
$1.0e-4$	7	$1.79e-1$
$1.0e-3$	8	$3.18e0$
$1.0e-2$	8	$3.18e0$
$1.0e-1$	9	$5.00e1$
$1.0e0$	10	$8.40e2$

表 5: 多項式 (15) (UVGCD 法) に対する実験結果. 詳細は例 2 を参照.

n	Relative error of GCD (14)			Time (sec.)		
	STLN	UVGCD	GPGCD	STLN	UVGCD	GPGCD
50	$1.60e-15$	$1.04e-16$	$2.63e-15$	1.77	0.22	0.04
100	$1.16e-15$	$1.59e-16$	$4.41e-15$	8.17	0.31	0.06
200	$1.14e-15$	$1.06e-16$	$1.23e-14$	45.09	0.83	0.12
500	$1.35e-15$	$1.37e-16$	$1.84e-14$	552.09	3.39	0.64
1000	$1.42e-15$	$1.69e-16$	$5.30e-14$	4318.38	18.66	3.27

表 6: 多項式 (16) に対する実験結果. 詳細は例 3 を参照.

実験結果を見ると, GPGCD 法および STLN 法の場合は, 入力多項式の次数が大きくなるのに伴い, 近似 GCD の精度が低下していることがわかる. これに対し, UVGCD 法の場合は, 算法が(収束の意味で) 極めて安定しており, 計算される近似 GCD の精度もより高いことがわかる.

4 まとめ

本稿では, 我々が提案している近似 GCD 算法 GPGCD 法のうち, 実係数もしくは複素係数の 2 つの入力多項式に対する算法について, 近似 GCD 算法の STLN 法および UVGCD 法との比較実験を行った結果を報告した.

本稿の実験例に対しては, GPGCD 法の, STLN 法および UVGCD 法に対する長所や短所として, 以下の事実が明らかになった. まず, 入力多項式がすでに厳密な GCD, もしくは微小な摂動によって求まるような近似 GCD をもつ場合には, UVGCD 法が, 近似 GCD を最も高精度で, かつ比較的速い収束性(効率) により求めることが示された. 一方で, 入力多項式がもつ "ノイズ" が比較的大きな場合, すなわち, 近似 GCD をもつために必要な摂動が比較的大きな場合には, GPGCD 法と STLN 法が, UVGCD 法に比べてより小さな摂動で近似 GCD を計算することが示された. さらに, これらの場合における効率を比較すると, GPGCD 法が他の算法に比べて大幅に効率的に(最大で STLN 法の約 30 倍, UVGCD 法の約 10 倍

k	Relative error of GCD (14)		
	STLN	UVGCD	GPGCD
15	$2.35e-13$	$3.08e-15$	$1.86e-12$
25	$1.64e-11$	$1.13e-14$	$6.67e-11$
35	$3.79e-10$ (*1)	$8.02e-15$	$3.58e-9$ (*2)
45	$4.23e-8$ (*1)	$1.13e-14$	$1.78e-7$ (*2)

表 7: 多項式 (17) に対する実験結果. 詳細は例 4 を参照.

$[m_1, m_2, m_3, m_4]$	Relative error of GCD (14)		
	STLN	UVGCD	GPGCD
$[2,1,1,0]$	$1.11e-13$	$9.42e-16$	$2.83e-13$
$[3,2,1,0]$	$7.33e-13$	$3.31e-15$	$8.23e-12$
$[4,3,2,1]$	$2.35e-9$	$2.95e-13$	$2.68e-9$
$[5,3,2,1]$	$1.89e-8$	$3.38e-12$	$5.56e-9$
$[9,6,4,2]$	$4.72e-8$ (*1)	$5.31e-11$	$6.05e-8$ (*2)
$[20,14,10,5]$	$5.06e-1$ (*1)	$3.13e-10$	$9.98e-1$ (*2)
$[80,60,40,20]$	$1.0e0$ (*1)	$1.08e-3$	$1.0e0$ (*2)
$[100,60,40,20]$	$1.0e0$ (*1)	$2.16e-4$	N/A (*3)

表 8: 多項式 (18) に対する実験結果. 詳細は例 5 を参照.

で) 近似 GCD を計算することが示された.

近似 GCD 算法には, 今回比較対象にした算法以外にも複数の算法があり, それらのいくつかは, 今回比較を行った算法が用いている最適化法以外のアプローチに基づくものもある. 今後は, こうした算法も比較対象にして, GPGCD 法の精度, 安定性, 効率等の長所や短所を明らかにし, 今後の算法の改善に役立てたいと考えている.

謝　辞

We thank Erich Kaltofen and Zhonggang Zeng for making their implementations for approximate GCD available on the Internet, and Erich Kaltofen for providing experimental results in 注意 (Remark) 1.

参 考 文 献

[1] B. Beckermann and G. Labahn. A fast and numerically stable Euclidean-like algorithm for detecting relatively prime numerical polynomials. *J. Symbolic Comput.*, Vol. 26, No. 6, pp. 691–714, 1998. Symbolic numeric algebra for polynomials.

[2] Dario A. Bini and Paola Boito. A fast algorithm for approximate polynomial gcd based on structured matrix computations. In Dario Andrea Bini, Volker Mehrmann, Vadim Olshevsky, Eugene E. Tyrtyshnikov, and Marc Barel, editors, *Numerical Methods for Structured Matrices and Applications*, Vol. 199 of *Operator Theory: Advances and Applications*, pp. 155–173. Birkhäuser, Basel, 2010.

[3] Paulina Chin, Robert M. Corless, and George F. Corliss. Optimization strategies for the approximate GCD problem. In *Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation (Rostock)*, pp. 228–235 (electronic), New York, 1998. ACM.

[4] R. M. Corless, P. M. Gianni, B. M. Trager, and S. M. Watt. The singular value decomposition for polynomial systems. In *Proceedings of the 1995 International Symposium on Symbolic and Algebraic Computation*, pp. 195–207. ACM, 1995.

[5] Robert M. Corless, Stephen M. Watt, and Lihong Zhi. *QR* factoring to compute the GCD of univariate approximate polynomials. *IEEE Trans. Signal Process.*, Vol. 52, No. 12, pp. 3394–3402, 2004.

[6] I. Z. Emiris, A. Galligo, and H. Lombardi. Certified approximate univariate GCDs. *J. Pure Appl. Algebra*, Vol. 117/118, pp. 229–251, 1997. Algorithms for algebra (Eindhoven, 1996).

[7] E. Kaltofen. Private communication, 2009.

[8] E. Kaltofen, Z. Yang, and L. Zhi. Structured low rank approximation of a Sylvester matrix. In D. Wang and L. Zhi, editors, *Symbolic-Numeric Computation*, Trends in Mathematics, pp. 69–83. Birkhäuser, 2007.

[9] Erich Kaltofen, Zhengfeng Yang, and Lihong Zhi. Approximate greatest common divisors of several polynomials with linearly constrained coefficients and singular polynomials. In *Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation*, pp. 169–176, New York, NY, USA, 2006. ACM.

[10] Victor Y. Pan. Computation of approximate polynomial GCDs and an extension. *Inform. and Comput.*, Vol. 167, No. 2, pp. 71–85, 2001.

[11] J. B. Rosen. The gradient projection method for nonlinear programming. II. Nonlinear constraints. *J. Soc. Indust. Appl. Math.*, Vol. 9, pp. 514–532, 1961.

[12] Masaru Sanuki and Tateaki Sasaki. Computing approximate gcds in ill-conditioned cases. In *SNC '07: Proceedings of the 2007 international workshop on Symbolic-numeric computation*, pp. 170–179, New York, NY, USA, 2007. ACM.

[13] T. Sasaki and M-T. Noda. Approximate square-free decomposition and root-finding of ill-conditioned algebraic equations. *J. Inform. Process.*, Vol. 12, No. 2, pp. 159–168, 1989.

[14] A. Schönhage. Quasi-gcd computations. *J. Complexity*, Vol. 1, No. 1, pp. 118–137, 1985.

[15] K. Tanabe. A geometric method in nonlinear programming. *J. Optim. Theory Appl.*, Vol. 30, No. 2, pp. 181–210, 1980.

[16] A. Terui. GPGCD, an iterative method for calculating approximate gcd of univariate polynomials, with the complex coefficients. In *Proceedings of the Joint Conference of ASCM 2009 and MACIS 2009*, Vol. 22 of *COE Lecture Note*, pp. 212–221. Faculty of Mathematics, Kyushu University, December 2009.

[17] A. Terui. An iterative method for calculating approximate GCD of univariate polynomials. In *Proceedings of the 2009 International Symposium on Symbolic and Algebraic Computation*, pp. 351–358, New York, NY, USA, 2009. ACM Press.

[18] A. Terui. GPGCD: An iterative method for calculating approximate GCD of univariate polynomials. *Theoretical Computer Science*, Vol. 479, pp. 127–149, 2013. Special Issue on Symbolic Numeric Computation SNC 2011. arXiv:1207.0630.

[19] Akira Terui. GPGCD: an approximate polynomial GCD library (version 0.2), May 2010. Project Hosting on Google Code (`http://code.google.com/p/gpgcd/`).

[20] Christopher J. Zarowski, Xiaoyan Ma, and Frederick W. Fairman. QR-factorization method for computing the greatest common divisor of polynomials with inexact coefficients. *IEEE Trans. Signal Process.*, Vol. 48, No. 11, pp. 3042–3051, 2000.

[21] Zhonggang Zeng. ApaTools: a software toolbox for approximate polynomial algebra. In *Software for algebraic geometry*, Vol. 148 of *IMA Vol. Math. Appl.*, pp. 149–167. Springer, New York, 2008.

[22] Zhonggang Zeng. The numerical greatest common divisor of univariate polynomials. In Leonid Gurvits, Philippe Pébay, J. Maurice Rojas, and David Thompson, editors, *Randomization, Relaxation, and Complexity in Polynomial Equation Solving*, Vol. 556 of *Contemporary Mathematics*, pp. 187–217. AMS, 2011.

[23] L. Zhi. Displacement structure in computing approximate GCD of univariate polynomials. In *Computer mathematics: Proc. Six Asian Symposium on Computer Mathematics (ASCM 2003)*, Vol. 10 of *Lecture Notes Ser. Comput.*, pp. 288–298. World Sci. Publ., River Edge, NJ, 2003.

[24] 佐々木建昭, 加古富士雄.「近似代数」とは？(特集: 数式処理とその周辺) . 数理科学, Vol. 36, No. 11, pp. 8–20, November 1998.

[25] 照井章. 近似 GCD 算法 GPGCD の複素係数多項式への拡張. In *Computer Algebra — Design of Algorithms, Implementations and Applications*, 数理解析研究所講究録, No. 1814, pp. 97–107. 京都大学数理解析研究所, October 2012.

[26] 照井章. 制約つき最適化に基づく 1 変数多項式の近似 GCD の反復算法. 数式処理, Vol. 16, No. 2, pp. 103–106, December 2009.

[27] 大迫尚行, 杉浦洋, 鳥居達生. 多項式剰余列の安定な拡張算法. 日本応用数理学会論文誌, Vol. 7, No. 3, pp. 227–255, September 1997.

[28] 藤田宏, 今野浩, 田邊國士. 最適化法. 岩波講座 応用数学 [方法 7]. 岩波書店, 1994.