数理解析研究所講究録
第1759巻 2011年 15-25

近似GCD算法GPGCDの複数入力多項式への拡張

GPGCD, an Iterative Method for Calculating Approximate GCD, for Multiple Univariate Polynomials

照井 章*
AKIRA TERUI
筑波大学大学院 数理物質科学研究科
GRADUATE SCHOOL 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算法の入力多項式を3個以上の実係数1変数多項式に対応させた拡張を提案する.

Abstract

We present an extension of our GPGCD method, an iterative method for calculating approximate greatest common divisor (GCD) of univariate polynomials, to multiple polynomial inputs. 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 extend our method to accept more than two polynomials with the real coefficients as an input.

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, 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.

本稿では, 近似代数計算 [22] の算法として, 近似最大公約子 (GCD) の問題を取り上げる. 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.

*terui@math.tsukuba.ac.jp

近似 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, 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 ([3], [6]), Sylvester の終結式行列の QR 分解に基づく算法 ([4], [19], [21]), Padé 近似に基づく算法 [9], 最適化法に基づく算法 (下記参照) などがある. さらに, 種々の悪条件問題に対する解法 ([4], [12], [24]) も議論されている.

本稿では, これらの中で, 近似 GCD の問題を制約つき最小化問題に帰着させ, 反復解法で解く最適化法に着目する. 最適化法を用いた既存の算法では, Levenberg-Marquard 法 [2] や Gauss-Newton 法 [20], 構造化行列を用いた最小二乗法の一種である STLN 法 (Structured Total Least Norm) ([7], [8]) 等が用いられており, 特に, 最近提案された STLN 法に基づく算法は, 近似 GCD を得るのに要する係数の摂動を小さく抑える点で注目を集めている.

我々は, これまでの研究で, GPGCD と呼ばれる反復算法を提案した [16]. 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, 田邊 ([15], [25, 第 4 章]) による修正 Newton 法を用いて局所的最小解を探索するものである. 実験結果では, STLN 法に基づく算法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. 我々がこれまでに提案した算法は, 入力として, 2 つの実係数多項式 [16], および 2 つの複素係数多項式 ([17], [23]) の近似 GCD を計算するものであった. 本稿では, GPGCD 算法を 3 つ以上の実係数多項式に対応させた拡張を提案する.

2 近似 GCD 問題と制約つき最小化問題への帰着

まず, n 個の実係数 1 変数多項式 $P_1(x), \ldots, P_n(x)$ を考える. 次数はそれぞれ $d_1, \ldots, d_n$, 多項式は次式で与えられるものとする.

$$P_i(x) = p_{d_i}^{(i)} x^{d_i} + \cdots p_1^{(i)} x + p_0^{(i)}.$$

ここに, $\min\{d_1, \ldots, d_n\} > 0$ とし, $i \neq j$ に対し, P_i と P_j は一般に互いに素であるとする. 与えられた次数 d (ただし $\min\{d_1, \ldots, d_n\} > d > 0$) に対し, $P_1(x), \ldots, P_n(x)$ 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.

$$\tilde{P}_i(x) = P_i(x) + \Delta P_i(x) = H(x) \cdot \bar{P}_i(x).$$

ここに, $\Delta P_i(x)$ は実係数多項式で, 次数がそれぞれ d_i を超えないような多項式, $H(x)$ は d 次の実係数多項式で, $i \neq j$ に対し, $\bar{P}_i(x)$ と $\bar{P}_j(x)$ は互いに素とする. このような状況で, H を $P_1(x), \ldots, P_n(x)$ 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 $\|\Delta P_1(x)\|_2^2 + \cdots + \|\Delta P_n(x)\|_2^2$ をなるべく小さく保ちつつ, $P_1(x), \ldots, P_n(x)$ の d 次の近似 GCD H を探索する問題を解く**.

実係数 1 変数多項式 $P(x) = p_n x^n + \cdots + p_0 x^0$ に対し, $C_k(P)$ を次式で定義される $(n+k, k+1)$ 行列とする.

$$C_k(P) = \underbrace{\begin{pmatrix} p_n & & \\ \vdots & \ddots & \\ p_0 & & p_n \\ & \ddots & \vdots \\ & & p_0 \end{pmatrix}}_{k+1}.$$

また, ベクトル

$$\boldsymbol{p} = (p_n, \ldots, p_0) \tag{1}$$

を多項式 $P(x)$ の係数ベクトルとする.

本稿では, n 個の多項式 $P_1,\ldots,P_n$ に対する Sylvester 行列の一般化として, 一般化 Sylvester 行列

$$N(P_1,\ldots,P_n)=\begin{pmatrix} C_{d_1-1}(P_2) & C_{d_2-1}(P_1) & \mathbf{0} & \cdots & \mathbf{0} \\ C_{d_1-1}(P_3) & \mathbf{0} & C_{d_3-1}(P_1) & \cdots & \mathbf{0} \\ \vdots & \vdots & & \ddots & \vdots \\ C_{d_1-1}(P_n) & \mathbf{0} & \cdots & \mathbf{0} & C_{d_n-1}(P_1) \end{pmatrix} \tag{2}$$

を用いる (Rupprecht [11, Sect. 3] を参照). さらに, $P_1,\ldots,P_n$ に対する k 次 (ただし $\min\{d_1,\ldots,d_n\} > k \geq 0$) の部分終結式行列として

$$N_k(P_1,\ldots,P_n)=\begin{pmatrix} C_{d_1-1-k}(P_2) & C_{d_2-1-k}(P_1) & \mathbf{0} & \cdots & \mathbf{0} \\ C_{d_1-1-k}(P_3) & \mathbf{0} & C_{d_3-1-k}(P_1) & \cdots & \mathbf{0} \\ \vdots & \vdots & & \ddots & \vdots \\ C_{d_1-1-k}(P_n) & \mathbf{0} & \cdots & \mathbf{0} & C_{d_n-1-k}(P_1) \end{pmatrix} \tag{3}$$

を用いる. ここに, $N_k(P_1,\ldots,P_n)$ の行数 r_k, 列数 c_k は, それぞれ

$$r_k = d_1+d_2+\cdots+d_n-(n-1)k+(n-2)d_1, \tag{4}$$

$$c_k = d_1+d_2+\cdots+d_n-n\cdot k \tag{5}$$

である.

n 個の多項式 $P_1,\ldots,P_n$ の GCD の算法は, 以下の事実に基づく.

命題 1 (Rupprecht [11, Proposition 3.1])
行列 $N_k(P_1,\ldots,P_n)$ が full rank ならば, かつそのときに限り, $\deg(\gcd(P_1,\ldots,P_n)) \leq k$ が成り立つ. (1 変数多項式 P に対し, $\deg(P)$ は P の次数を表す.) ∎

ゆえに, 与えられた次数 d に対し, 行列 $N_{d-1}(\tilde{P}_1,\ldots,\tilde{P}_n)$ がランク落ちを起こすならば, 1 変数多項式 $U_1(x),\ldots,U_n(x)$ (次数はそれぞれ高々 $d_1-d,\ldots,d_n-d$ をみたす) が存在して, $i=2,\ldots,n$ に対し

$$U_1\tilde{P}_i+U_i\tilde{P}_1=0 \tag{6}$$

が成り立つ. このとき, もし $i\neq j$ に対して U_i と U_j が互いに素ならば, $H=\frac{\tilde{P}_1}{U_1}=-\frac{\tilde{P}_2}{U_2}=\cdots=-\frac{\tilde{P}_n}{U_n}$ が求める GCD となる. よって, 本稿で考える問題は, 与えられた多項式 $P_1,\ldots,P_n$ と次数 d に対し, 方程式 (6) をみたすような $\Delta P_1,\ldots,\Delta P_n, U_1,\ldots,U_n$ で, $\|\Delta P_1(x)\|_2^2+\cdots+\|\Delta P_n(x)\|_2^2$ 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.

以下では, $\tilde{P}_i(x)$, $U_i(x)$ を

$$\tilde{P}_i(x)=\tilde{p}_{d_i}^{(i)}x^{d_i}+\cdots+\tilde{p}_1^{(i)}x+\tilde{p}_0^{(i)},\quad U_i(x)=u_{d_i-d}^{(i)}x^{d_i-d}+\cdots+u_1^{(i)}x+u_0^{(i)} \tag{7}$$

で表す. 目的関数の導出を行うと, $\|\Delta P_1(x)\|_2^2+\cdots+\|\Delta P_n(x)\|_2^2$ は

$$\|\Delta P_1(x)\|_2^2+\cdots+\|\Delta P_n(x)\|_2^2=\sum_{i=1}^{n}\left\{\sum_{j=0}^{d_i}\left(\tilde{p}_j^{(i)}-p_j^{(i)}\right)^2\right\} \tag{8}$$

となる.

一方, 制約条件の導出について, 式 (6) は

$$N_{d-1}(\tilde{P}_1,\ldots,\tilde{P}_n)\cdot {}^t(\boldsymbol{u}_1,\ldots,\boldsymbol{u}_n)=\boldsymbol{0} \tag{9}$$

となる. ここに, $\boldsymbol{u}_i$ は, 式 (1) にて定義される $U_i(x)$ の係数ベクトルである. さらに, $U_i(x)$ の係数に対し, 新たな制約

$$\|U_1\|_2^2+\cdots+\|U_n\|_2^2=1 \tag{10}$$

を設ける. これを方程式 (9) の上に配置することにより,

$$\begin{pmatrix} \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_n & -1 \\ N_{d-1}(\tilde{P}_1,\ldots,\tilde{P}_n) & & & \boldsymbol{0} \end{pmatrix}\cdot {}^t(\boldsymbol{u}_1,\ldots,\boldsymbol{u}_n,1)=\boldsymbol{0}, \tag{11}$$

を得る. ここで, 連立方程式 (11) は, (7) の多項式の各係数を変数にもつ連立方程式で,

$$\bar{d}=d_1+\cdots+d_n-(n-1)(d-1)+(n-2)d_1+1 \tag{12}$$

個の方程式をもつことに注意する. 第 j 行の方程式を $g_j=0$ とおく.

ここで, これまでの多項式の係数を表す変数

$$(\tilde{p}_{d_1}^{(1)},\ldots,\tilde{p}_0^{(1)},\ldots,\tilde{p}_{d_n}^{(n)},\ldots,\tilde{p}_0^{(n)},u_{d_1-d}^{(1)},\ldots,u_0^{(1)},\ldots,u_{d_n-d}^{(n)},\ldots,u_0^{(n)}), \tag{13}$$

を, それぞれ $\boldsymbol{x}=(x_1,\ldots,x_{2(d_1+\cdots+d_n)+(2-d)n})$ に置き換える. すると, 式 (8) および方程式 (11) は, それぞれ

$$f(\boldsymbol{x})=(x_1-p_{d_1}^{(1)})^2+\cdots+(x_{d_1}-p_0^{(1)})^2+\cdots$$
$$\cdots+(x_{d_1+\cdots+d_{n-1}+n}-p_{d_n}^{(n)})^2+\cdots+(x_{d_1+\cdots+d_{n-1}+d_n+n}-p_0^{(n)})^2, \tag{14}$$

$$\boldsymbol{g}(\boldsymbol{x})={}^t(g_1(\boldsymbol{x}),\ldots,g_{\bar{d}}(\boldsymbol{x}))=\boldsymbol{0} \tag{15}$$

と表される (式 (15) の $\bar{d}$ の定義は式 (12) の通り). 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 (15) ($\boldsymbol{g}(\boldsymbol{x})=\boldsymbol{0}$) の下で, 式 (14) の $f(\boldsymbol{x})$ を最小化せよ. ∎

3 近似GCDの算法

本稿では, 与えられた近似 GCD の問題を制約つき最適化問題 (問題 1) に帰着させたものを, 非線形最適化法の一つである勾配射影法 ([10]: 頭文字が本稿の算法名 GPGCD の発祥) もしくは修正 Newton 法 ([15], [25, 第 4 章]) によって解く (詳細は本著者の論文 [16] を参照). 我々のこれまでの実験 ([16, Section 5.1], [17, Sect. 4]) では, 例題に対し, 両算法とも同様の収束性を示す一方, 修正 Newton 法の方がより効率的である結果が得られている. よって, 以後は修正 Newton 法を最適化問題の解法として取り入れる.

実際に修正 Newton 法を用いて近似 GCD の問題を解くにあたり, 以下の問題を考慮する必要があるので, 下記の各節において議論する.

1. ヤコビ行列 $J_{\boldsymbol{g}}(\boldsymbol{x})$ の構成, ならびに, 反復計算の過程において, ヤコビ行列 $J_{\boldsymbol{g}}(\boldsymbol{x})$ が full rank である (ランクが $J_{\boldsymbol{g}}(\boldsymbol{x})$ の行数に等しい) こと (第 3.1 節).
2. 反復計算の初期値の設定 (第 3.2 節).
3. 最小化問題を最短ベクトル問題に置き換えること (第 3.3 節).
4. GCD となる多項式と, 摂動項の実際の計算 (第 3.4 節).

3.1 ヤコビ行列の表現とランク

制約式 (15) の定義により, ヤコビ行列 $J_g(\boldsymbol{x})$ は次式の通り求まる.

$$J_g(\boldsymbol{x}) = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ C_{d_1}(U_2) & C_{d_2}(U_1) & \mathbf{0} & \cdots & \mathbf{0} \\ C_{d_1}(U_3) & \mathbf{0} & C_{d_3}(U_1) & & \mathbf{0} \\ \vdots & \vdots & & \ddots & \vdots \\ C_{d_1}(U_n) & \mathbf{0} & \cdots & \mathbf{0} & C_{d_n}(U_1) \end{pmatrix} \begin{pmatrix} 2\cdot{}^t\boldsymbol{u}_1 & 2\cdot{}^t\boldsymbol{u}_2 & 2\cdot{}^t\boldsymbol{u}_3 & \cdots & 2\cdot{}^t\boldsymbol{u}_n \\ C_{d_1-d}(P_2) & C_{d_2-d}(P_1) & \mathbf{0} & \cdots & \mathbf{0} \\ C_{d_1-d}(P_3) & \mathbf{0} & C_{d_3-d}(P_1) & & \mathbf{0} \\ \vdots & \vdots & & \ddots & \vdots \\ C_{d_1-d}(P_n) & \mathbf{0} & \cdots & \mathbf{0} & C_{d_n-d}(P_1) \end{pmatrix}. \quad (16)$$

(ここに, $\boldsymbol{x}$ の変数名は式 (13) で置き換える前のものを用いていることに注意.)

修正 Newton 法の反復計算の過程において, ヤコビ行列 $J_g(\boldsymbol{x})$ が full rank であることが保証される必要がある (そうでないと, 探索方向を決定できない). この問題については, 以下の命題が成り立つ.

命題 2

$i = 1, \ldots, n$ に対し, $\deg d < \min\{d_1, \ldots, d_n\} - 1$ かつ $\deg U_i \geq 1$ とする. $\boldsymbol{x}^* \in V_g$ を, 方程式 (15) をみたす許容点とする. このとき, $\boldsymbol{x}^*$ に対応する多項式の GCD が d を超えないならば, ヤコビ行列 $J_g(\boldsymbol{x}^*)$ のランクは $J_g(\boldsymbol{x}^*)$ の行数に等しい.

証明 Terui [18] を参照. ∎

命題 2 より, 最適化の探索方向が, 探索先の点に対応する GCD の次数を d より大きくするものでない限り, $J_g(\boldsymbol{x})$ が full rank であることが保たれ, 近似 GCD の探索方向を適切に保ちつつ反復計算を続けられることがわかる.

3.2 反復計算の初期値の設定

反復計算の開始時において, 初期値 $\boldsymbol{x}_0$ は一般化 Sylvester 行列の特異値分解 (SVD) [5] に基づいて与える. 式 (3) の行列 $N_{d-1}(P_1, \ldots, P_n)$ に対し, 特異値分解を以下の通り計算する.

$$\begin{gathered} N_{d-1} = U\,\Sigma\,{}^tV, \\ U = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{c_{d-1}}), \quad \Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_{c_{d-1}}), \quad V = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{c_{d-1}}). \end{gathered} \quad (17)$$

ここに, $\boldsymbol{w}_j \in \mathbb{R}^{r_{d-1}}$, $\boldsymbol{v}_j \in \mathbb{R}^{c_{d-1}}$ であり, r_k, c_k はそれぞれ式 (4), (5) による. $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_{c_{d-1}})$ は対角行列で, 第 j 対角要素が σ_j である. また, U と V は直交行列である. 特異値分解の性質 [5, Theorem 3.3] により, 最小特異値 $\sigma_{c_{d-1}}$ は, $\mathbb{R}^{c_{d-1}}$ の単位球面

$$\mathrm{S}^{c_{d-1}-1} = \{\boldsymbol{x} \in \mathbb{R}^{c_{d-1}} \mid \|\boldsymbol{x}\|_2 = 1\}$$

を $N_{d-1}(P_1, \ldots, P_n)$ で写した集合

$$N_{d-1} \cdot \mathrm{S}^{c_{d-1}-1} = \{N_{d-1}\boldsymbol{x} \mid \boldsymbol{x} \in \mathbb{R}^{c_{d-1}}, \|\boldsymbol{x}\|_2 = 1\}$$

上の点の, 原点からの距離の最小値を与える. 式 (17) より

$$N_{d-1} \cdot \boldsymbol{v}_{c_{d-1}} = \sigma_{c_{d-1}} \boldsymbol{w}_{c_{d-1}}$$

が成り立つので,

$$\boldsymbol{v}_{c_{d-1}} = {}^t(\bar{u}^{(1)}_{d_1-d}, \ldots, \bar{u}^{(1)}_0, \ldots, \bar{u}^{(n)}_{d_n-d}, \ldots, \bar{u}^{(n)}_0)$$

に対し, 多項式 $\bar{U}_i(x)$ の係数を

$$\bar{U}_i(x) = \bar{u}^{(i)}_{d_i-d} x^{d_i-d} + \cdots + \bar{u}^{(i)}_0 x^0, \quad (i = 1, \ldots, n)$$

と定めると, $\bar{U}_1(x), \ldots, \bar{U}_n(x)$ は, 式 (7) において $U_i(x) = \bar{U}_i(x)$ とおくことにより, $\|U_1\|_2^2 + \cdots + \|U_n\|_2^2 = 1$ を満たしつつ, $U_1 P_i + U_i P_1$ の最小ノルムを与える.

ゆえに, 初期値として, $P_1, \ldots, P_n, \bar{U}_1, \ldots, \bar{U}_n$ の係数からなるベクトル

$$\boldsymbol{x}_0 = (p^{(1)}_{d_1}, \ldots, p^{(1)}_0, \ldots, p^{(n)}_{d_n}, \ldots, p^{(n)}_0, \bar{u}^{(1)}_{d_1-d}, \ldots, \bar{u}^{(1)}_0, \ldots, \bar{u}^{(n)}_{d_n-d}, \ldots, \bar{u}^{(n)}_0) \tag{18}$$

を与え, 反復計算を行う.

3.3 最小化問題の最近ベクトル問題への置き換え

我々が扱う最小化問題の目的関数 f に対し, 式 (14) より

$$\nabla f(\boldsymbol{x}) = 2 \cdot {}^t(x_1 - p^{(1)}_{d_1}, \ldots, x_{d_1} - p^{(1)}_0, \ldots, x_{d_1+\cdots+d_{n-1}+n} - p^{(n)}_{d_n}, \ldots, x_{d_1+\cdots+d_{n-1}+d_n+n} - p^{(n)}_0, 0, \ldots, 0)$$

が成り立つ. しかし, この最小化問題は, $(x_1, \ldots, x_{d_1+\cdots+d_{n-1}+d_n+n})$-座標 ($P_i(x)$ の係数に対応する) に関し, 初期値 $\boldsymbol{x}_0$ からの距離が最小であるような点 $\boldsymbol{x} \in V_g$ を探索する問題ととらえることができる. ゆえに, 著者のこれまでの算法 ([16], [17]) と同様, 最小化問題の目的関数を $\bar{f}(\boldsymbol{x}) = \frac{1}{2} f(\boldsymbol{x})$ とおき, 制約条件 $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0}$ の下で $\bar{f}(\boldsymbol{x})$ の最小値を求める最小化問題を解く.

3.4 GCD, および摂動を加えた多項式の計算

反復計算が適切に収束した後, $\tilde{P}_i(x), U_i(x)$ $(i = 1, \ldots, n)$ の係数が得られ, それらは式 (6) をみたし, かつ $i \neq j$ に対し, U_i と U_j は互いに素となる. このとき, $\tilde{P}_i(x)$ の GCD である $H(x)$ の係数を計算する必要がある. H は $\tilde{P}_i$ を U_i で除した商として求まるが, 素朴な多項式除算は, 係数に誤差を増大させる恐れがある. そこで, H の係数を最小二乗法 [20] で求め, それから H の係数を用いて $\tilde{P}_i$ の係数を修正する.

$\tilde{P}_i, U_i$ は式 (7) の通り表され, H は $H(x) = h_d x^d + \cdots + h_0 x^0$ で表されるものとする. このとき, 方程式 $HU_i = \tilde{P}_i$ を H について解くにあたり, 連立 1 次方程式

$$C_d(U_i)\, {}^t(h_d \ldots, h_0) = {}^t(\tilde{p}^{(i)}_{d_i}, \cdots, \tilde{p}^{(i)}_0) \tag{19}$$

の最小二乗解を求め, この最小二乗解から求まる GCD (実係数多項式) の候補を $H_i(x)$ とおく. その後, $i = 2, \ldots, n$ に対し, 残差

$$r_i = \sum_{j=1}^{n} \|P_j - H_i U_j\|_2^2$$

を計算し, 最も小さな残差 r_i を与える i に対し, $H_i(x)$ を求める近似GCD $H(x)$ とする. 最後に, $i = 1, \ldots, n$ に対し, 定めた $H(x)$ を用いて, $\tilde{P}_i(x) = H(x) \cdot U_i(x)$ によって $\tilde{P}_i(x)$ を修正する.

3.5 算法

以上をまとめると, 実際に近似 GCD を計算する算法は以下の通りとなる.

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

- 入力:
  - $P_1(x), \ldots, P_n(x) \in \mathbb{R}[x]$, ただし $\min\{\deg(P_1), \ldots, \deg(P_n)\} > 1$ をみたす (命題 2 を参照),
  - $d \in \mathbb{N}$: 近似 GCD の次数, ただし $d < \min\{\deg(P_1), \ldots, \deg(P_n)\} - 1$ をみたす (命題 2 を参照),
  - $\varepsilon > 0$: 反復計算の収束判定のためのしきい値,
  - $u \in \mathbb{N}$: 反復計算の反復回数の上限値.
- 出力: $\tilde{P}_1(x), \ldots, \tilde{P}_n(x), H(x) \in \mathbb{R}[x]$, ただし $\tilde{P}_1, \ldots, \tilde{P}_n$ はそれぞれ $P_1, \ldots, P_n$, の係数に摂動を加えた多項式で, 次数 d の多項式 H を GCD にもつ.

Step 1 [**初期値の設定**] 第 3.2 節の議論に基づき, 初期値 $\boldsymbol{x}_0$ を式 (18) の通り定める.

Step 2 [**反復計算**] 第 3.3 節の議論に基づき, 制約条件 $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0}$ の下で $\tilde{f}(\boldsymbol{x}) = \frac{1}{2} f(\boldsymbol{x})$ の最小値を計算する. ここに, $f(\boldsymbol{x})$, $\boldsymbol{g}(\boldsymbol{x})$ は, それぞれ式 (14), (15) で定義されたものである. 最小値の計算は, 勾配射影法 (修正 Newton 法) の反復計算によって行う. 第 k 回目の反復値 $\boldsymbol{x}_k$ に対し, 探索方向 $\boldsymbol{d}_k$ が $\|\boldsymbol{d}_k\|_2 < \varepsilon$ (ε は与えられたしきい値) をみたすか, または 反復回数が与えられた上限 u に達するまで反復計算を繰り返す.

Step 3 [$\tilde{P}_1, \ldots, \tilde{P}_n$ **および** H **の計算**] 第 3.4 節の議論に基づき, GCD $H(x)$ を計算し, $\tilde{P}_1(x), \ldots, \tilde{P}_n(x)$ の係数を修正した後, $\tilde{P}_1(x), \ldots, \tilde{P}_n(x)$ および $H(x)$ を出力する. もし Step 2 が u 回の反復で収束しなかった場合は, その旨をユーザに報告する. ∎

4 実験

今回提案したアルゴリズム 1 を, 数式処理システム Maple に実装し, 最小二乗法の一つである STLN (structured total least norm) 法を基にした算法 [7] との比較を行った. 計算の比較は, 係数をランダムに生成した多項式の組に対し, それらの近似 GCD を計算することで行った. 実験環境は Intel Core2 Duo Mobile Processor T7400 (Apple MacBook "Mid-2007") at 2.16 GHz, RAM 2GB, MacOS X 10.5 である.

実験に用いる入力多項式は, GCD をもつ多項式をランダムに生成し, それらに摂動項を加える形で行った. まず, モニックな m 次多項式 $P_0(x)$ を, d 次の GCD をもつように生成した. GCD および $m-d$ 次の余因子は, それぞれモニックで, かつ, 係数は $[-10, 10]$ の範囲で乱数で発生させた浮動小数とする. これらに対し, 摂動項として, $m-1$ 次の多項式 $P_{\mathrm{N}}(x)$ を生成した. 係数の与え方は $P_0(x)$ に準ずる. そして, 多項式 $P(x)$ を

$$P(x) = P_0(x) + \frac{e_P}{\|P_{\mathrm{N}}(x)\|_2} P_{\mathrm{N}}(x)$$

によって定め, P_0 に対する摂動項の 2-ノルムの大きさが e_P に等しくなるようにする. 本稿では $e_P = 0.1$ とした.

本稿の実験では, 比較対象である STLN 法を基にした算法 [7] の実装として, その著者らによる実装を用いた (謝辞を参照). STLN 法を基にした算法では, $\mathbb{R}[x]$ 上で複数個の多項式の近似 GCD を計算するプロシージャ `R_con_mulpoly` を用いた. 実験は, Maple 13 上で `Digits=15` の設定, すなわちハードウェアの

(倍精度) 浮動小数演算を用いた. 入力多項式の次数を 10 次, 20 次, 40 次の 3 通りに変化させたた上で, 各次数において, 入力多項式の個数を 3 個から 10 個まで, 1 個ずつ変化させた. そして, 各次数および多項式の個数に対し, 10 組の多項式をランダムに生成してテストを行い, 各算法が規定内 ("規定" については下記を参照) に収束した実験例に対し, 以下に述べる測定値について, 平均値を算出した. 各算法において, 探索の終了を判定するための条件として, アルゴリズム 1 では, 探索方向の方向ベクトルのうち, $\tilde{P}_i(x)$, $U_i(x)$ (式 (6) を参照) の係数に対応する成分の 2-ノルムが $\varepsilon = 1.0 \times 10^{-8}$ よりも小さくなることとし, `R_con_mulpoly` では, 探索終了のしきい値を $e = 1.0 \times 10^{-8}$ とした.

実験結果は, 入力多項式の次数が 10 次, 20 次, 40 次の順に, それぞれ表 1, 2, 3 の通りにまとめている. 実験で与えた多項式の近似 GCD の次数は, それぞれもとの多項式の次数の半分, すなわち, 入力多項式の次数が 10 次, 20 次, 40 次に対し, 近似 GCD の次数はそれぞれ 5 次, 10 次, 20 次である. n は入力多項式の個数を表す. 見出しが "STLN" の列は STLN 法に基づく算法の結果を表し, 見出しが "GPGCD" の列は GPGCD 法の結果を表す. "#Fail" は, 各算法において, "規定内に" 収束しなかった実験例の個数を表す. ここに, その規定は, STLN 法においては, 反復回数が 50 回 (これは, その実装において定義されているしきい値である), GPGCD 法においては, 反復回数が 100 回を表す. "Error" は近似 GCD を得るために与えられた多項式に加えた摂動 $\|\tilde{F} - F\|_2^2 + \|\tilde{G} - G\|_2^2$ の平均値を表す ("$ae{-}b$" は $a \times 10^{-b}$ を表す). "#Iterations" は反復回数の平均値を表す. "Time" は計算時間 (秒) の平均値を表す.

実験結果を見ると, ほとんどの実験において, STLN 法, GPGCD 法の両方とも, 同程度の大きさの摂動で近似 GCD を計算している. 反復回数の平均値では, STLN 法においてはほぼ一定の反復回数で近似 GCD を計算している一方, GPGCD 法においては, 計算例によって反復回数に若干のばらつきがあり, 特に, 一部の計算例 (表 2 の $n = 8, 10$ や表 3 の $n = 7$ の場合) に対しては, 反復回数が大幅に増加していることがうかがえる. 計算効率に関しては, GPGCD 法において, 上述の反復回数が大幅に増加した計算例では, 計算時間が STLN 法のそれに迫る程度であるが, それ以外の大半の計算例では, GPGCD 法の方が STLN 法がより効率よく (最大 4 倍程度) 近似 GCD を計算していることがわかる. なお, GPGCD 法では, 規定内の反復回数で計算が終了しなかった例題 (表 2 の $n = 9, 10$, 表 3 の $n = 6, 9$) があった一方, STLN 法ではそのような例題はなかったことに注意する.

5 まとめ

本稿では, 我々の先行研究の結果 ([16], [17]) を拡張する形で, 複数個の実係数多項式に対する GPGCD 法を示した.

実験結果によると, 本稿で提案した算法は, これまでに提案した算法 ([16], [17]) と同様, 多くの問題に対して, STLN 法に基づく算法と同程度の摂動で近似 GCD を計算する一方, STLN 法に基づく算法よりも数倍程度は効率がよいことを示した. これにより, GPGCD 法は, 入力多項式の次数が小〜中程度ならば, 多くの問題に対し, 実用的な算法であると思われる. 一方で, いくつかの例題に対して GPGCD 法が十分小さな反復回数で収束しない現象が見られたのに対し, STLN 法に基づく算法はすべての実験例に対してほぼ一定の反復回数で収束したことから, 現時点では, STLN 法に基づく算法の方が, 一般的により安定していると思われる.

今後の研究方針としては, 上述に述べた収束性の改善や, 理論的な収束性の解明が必要と思われる. また, 複素係数に対する算法への拡張も可能である. それから, 本稿で用いた一般化 Sylvester 行列は, P_1 を特別扱いするものであるが, P_1 の選択による算法の動作の変化も興味深い問題であろう. さらに, 実数 $a_2, \ldots, a_n$ をランダムに与え, $\gcd(P_1, P_2, \ldots, P_n)$ を $\gcd(P_1, a_2P_2 + \cdots + a_nP_n)$ に写して考えることも可能である. このような変換を行うことで, 一般化 Sylvester 行列の次数を下げることが可能になり, 近似 GCD を計算す

るまた一つのアプローチになる可能性がある. 以上を含む観点から, 今後の研究を進めていきたいと考えている.

謝　辞

We thank Professor Erich Kaltofen for making their implementation for computing approximate GCD available on the Internet and providing experimental results. We also thank the anonymous reviewers of companion paper [18] for their valuable suggestions.

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] 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.

[3] 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.

[4] 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.

[5] James W. Demmel. *Applied numerical linear algebra.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[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] 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.

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

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

[11] D. Rupprecht. An algorithm for computing certified approximate GCD of n univariate polynomials. *J. Pure and Applied Algebra*, Vol. 139, pp. 255–284, 1999.

[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. 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.

[17] 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.

[18] A. Terui. GPGCD, an iterative method for calculating approximate GCD, for multiple univariate polynomials. In V.P. Gerdt, W. Koepf, E.W. Mayr, and E.H. Vorozhtsov, editors, *Computer Algebra in Scientific Computing (Proc. CASC 2010)*, Vol. 6244 of *Lecture Notes in Computer Science*, pp. 238–249. Springer, 2010.

[19] 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.

[20] Zhonggang Zeng. The approximate GCD of inexact polynomials, Part I: a univariate algorithm (extended abstract). preprint. 8 pages.

[21] 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.

[22] 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.

[23] 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. 京都大学数理解析研究所, 2010. 印刷中.

[24] 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.

[25] 藤田宏, 今野浩, 田邉國士. 最適化法. 岩波講座 応用数学 [方法 7]. 岩波書店, 1994.

n	#Fail		Error		#Iterations		Time (sec.)	
	STLN	GPGCD	STLN	GPGCD	STLN	GPGCD	STLN	GPGCD
3	0	0	$0.231e-2$	$0.238e-2$	5.5	11.2	1.17	0.45
4	0	0	$0.265e-2$	$0.266e-2$	4.5	12.5	1.85	0.91
5	0	0	$5.27e-2$	$5.22e-2$	4.7	13.5	3.10	1.53
6	0	0	$0.273e-2$	$0.284e-2$	4.5	14.1	4.36	2.37
7	0	0	$0.294e-2$	$0.292e-2$	4.5	14.5	5.95	3.32
8	0	0	$0.297e-2$	$0.306e-2$	4.2	15.8	7.51	4.65
9	0	0	$0.296e-2$	$0.308e-2$	4.4	14.4	10.12	5.57
10	0	0	$5.48e-2$	$5.62e-2$	4.4	17.9	12.49	8.59

表 1: 入力多項式の次数が 10 次の実験結果. 近似 GCD の次数は 5 次, n は入力多項式の個数. 詳細は第 4 節を参照.

n	#Fail		Error		#Iterations		Time (sec.)	
	STLN	GPGCD	STLN	GPGCD	STLN	GPGCD	STLN	GPGCD
3	0	0	$5.17e-2$	$5.40e-2$	4.5	12.0	3.35	1.52
4	0	0	$5.70e-2$	$5.70e-2$	4.3	13.7	6.22	3.26
5	0	0	$5.89e-2$	$5.85e-2$	4.4	12.7	10.37	4.97
6	0	0	$6.07e-2$	$5.88e-2$	4.2	18.1	15.04	10.23
7	0	0	$6.20e-2$	$6.06e-2$	4.2	19.0	20.83	14.85
8	0	0	$6.26e-2$	$6.21e-2$	4.1	22.2	27.56	22.82
9	0	1	$6.35e-2$	$6.18e-2$	3.9	14.1	34.16	19.05
10	0	1	$6.31e-2$	$6.19e-2$	4.0	25.6	44.62	43.16

表 2: 入力多項式の次数が 20 次の実験結果. 近似 GCD の次数は 10 次, n は入力多項式の個数. 詳細は第 4 節を参照.

n	#Fail		Error		#Iterations		Time (sec.)	
	STLN	GPGCD	STLN	GPGCD	STLN	GPGCD	STLN	GPGCD
3	0	0	$5.32e-2$	$5.39e-2$	4.9	12.8	13.6	5.83
4	0	0	$5.78e-2$	$5.78e-2$	4.3	11.2	24.9	10.1
5	0	0	$6.01e-2$	$5.97e-2$	4.3	12.1	41.5	17.9
6	0	1	$6.11e-2$	$6.07e-2$	4.2	9.78	62.3	21.9
7	0	0	$6.25e-2$	$6.15e-2$	4.1	23.7	87.0	72.8
8	0	0	$6.31e-2$	$6.21e-2$	4.1	9.6	118.3	40.5
9	0	1	$6.38e-2$	$6.20e-2$	4.2	7.89	158.0	43.7
10	0	0	$6.41e-2$	$6.25e-2$	4.1	8.9	200.9	60.2

表 3: 入力多項式の次数が 40 次の実験結果. 近似 GCD の次数は 20 次, n は入力多項式の個数. 詳細は第 4 節を参照.