次数指定型/update可能な高速１変数近似GCD計算
Fast Approximate GCD Computation for Univariate Polynomials with Degree Updating

讃岐　勝

Masaru Sanuki *

筑波大学医学医療系臨床医学域

Department of Clinical Medicine, Faculty of Medicine, University of Tsukuba †

Abstract

１変数近似 GCD の計算法について、次数指定型の高速算法を紹介する。次数指定型の算法は適切な次数を設定する必要があるが、本講演では次数が間違っていた場合にも、前の結果を利用して無駄なく近似 GCD の計算ができる方法を紹介する。

Abstract

In this paper, we propose a fast method for computing approximate GCD of univariate polynomials with floating-point numbers. Our method requires to input the degree of approximate GCD, however, it is difficult to determine the degree correctly in general. So that, we propose how to update the degree rapidly.

1　はじめに

１変数多項式の近似 GCD を計算する種々の方法が提案されており，大きく次の２つに分類できる．

- 直接計算型（次数探索型）：Euclid の互除法 [20] とその数値的安定手法 [23]，Sylvester 行列の QR 法による方法である QRGCD 法と [17, 8] その理論的な改良 [16].
- 次数指定型：近似 GCD の次数を入力として与える方法．行列を用いる方法の多くがこちらに分類される．入力する次数（候補）を予め計算する方法と，既知のものとして計算する方法がある．

現在では数値計算を基にした次数指定型の方法が数値的に安定しているため，広く利用されている．

一般的に多項式を利用する数式処理の方法に比べ，行列を利用する数値計算の方法は効率が劣るように言われることが多いが現状においては大差がない．その原因は整数係数の場合に利用される half-GCD 法に対応するような計算量 $O(n^2)$ 未満の方法が浮動小数係数の場合には未だ存在していないからである [21]. ここで，n は入力多項式の次数の最大値である．

行列を利用した計算法は，素朴な QR 法 [8] や特異値分解による方法 [7] からはじまり，行列の高速算法を利用する方法へと発展を遂げてきたが，未だ計算量 $O(n^2)$ 未満の方法はこれまで提案されていない．筆

*本研究は筑波大学研究基盤支援プログラム（タイプ A）「数値数式融合計算による中規模構造化行列の高速 rank 計算法の開発」の助成を受けています

†sanuki@md.tsukuba.ac.jp

者による浮動小数係数の half-GCD 法は失敗に終わっており，既存の方法からの拡張では不可能と思われた．しかし，[22] においてまったく新しい方法で計算量 $O(n^2)$ 未満の方法が提案することに成功し，それは浮動小数係数の half-GCD 法とは異なり数値計算を基にした方法であり数値的な不安定性はない．ただし，算法自身荒削りで効率の面では改善する必要がある．本稿では，算法を細かく見ることで効率化を行う．

本稿では，Bezout-Hankel 行列とそれを利用する GCD と知られる Barnett の方法の改良を利用した算法の効率化を検討する．2 章では，使用する Bezout-Hankel 行列および Barnett の方法の改良について紹介し，GCD に計算における改良点を確認する．3 章では，前章で確認した改良のポイントについて実際に高速計算できることを示す．4 章では，入力の次数が間違っていた場合に次数を修正して計算する方法を述べる．

1.1 記号

本稿では，次を記号を使用する．浮動小数係数の 1 変数多項式の集合を $\mathbb{F}[x]$ で表す．2 つの与多項式 $f(x), g(x) \in \mathbb{F}[x]$ は $\deg(f) = n > \deg(g) = n-1$ を満たすように次の方法で前処理する．

- $f(0) = 0$ のときは $f(x)/x$ を $f(x)$ と置き直す．$g(x)$ についても同様.
- $\deg(f) = \deg(g)$ のときは $f(x) - \mathrm{lc}(f)/\mathrm{lc}(g)g(x)$ を g と見なす.
- $\deg(f) - \deg(g) = d > 1$ のときは，$g(x)$ を $x^{d-1}g(x)$ で置き直す.

2 Bezout-Hankel 行列

まず，本稿で利用する Bezout-Hankel 行列について定義する.

定義 1 (Bezout-Hankel 行列)
多項式 $f(x)$ と $g(x)$ からなる *Bezout-Hankel* 行列 $\mathcal{H}_n(f,g)$ とは $g(x)/f(x)$ の無限遠点上で *Taylor* 展開から得られる $\mathbb{F}\{x^{-1}\}$ 上の形式的べき級数 $g(x)/f(x) = h_1x^{-1} + h_2x^{-2} + \ldots \in \mathbb{F}\{x^{-1}\}$ の係数 h_i から構成される次の *Hankel* 行列をである[1)].

$$\mathcal{H}_n(f,g) = \begin{pmatrix} h_1 & h_2 & \cdots & h_n \\ h_2 & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ h_n & \cdots & \cdots & h_{2n-1} \end{pmatrix} = (\boldsymbol{h}_1\ \boldsymbol{h}_2\ \ldots\ \boldsymbol{h}_n) \in \mathbb{F}^{n\times n}.$$

∎

補題 2 (行列を構成するための計算量)
Bezout-Hankel 行列の要素 $h_1, \ldots, h_{2n-1}$ を構成するための計算量は $O(M(n))$ である．ここで，$M(n)$ は次数 n の多項式同士の積の計算量を表す. ∎

注意 1
$O(M(n))$ と $O(n^2)$ は一般に等しくはない．乗算の場合，n が大きい場合には *FFT* などの高速計算が利用可能であり，$O(n\log n)$ で評価される．本稿では計算量 $O(M(n))$ 以下の算法の構築を目指している.

この Bezout-Hankel 行列を用いた近似 GCD を求める方法として，Pade 近似による方法 [18] と Barnett の定理の改良版がある．オリジナルの方法は companion 行列の方法 [1, 2] で，次で紹介するのは Bezout-Hankel 行列による方法である [10].

[1)] 前処理によって，級数は h_1 から始まる.

命題 3 (Barnett の定理の改良版 [10])
$k = \deg(\gcd(f,g))$ とする．このとき，$\mathcal{H}_n(f,g) \in \mathbb{F}^{n\times n}$ の前 $n-k$ 列は線形独立であり，かつ，後ろ k 列は $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{n-k}$ の線型結合でかける．

$$\sum_{j=1}^{n-k+1} q_{i,j}\boldsymbol{h}_j + q_{i,n-k}\boldsymbol{h}_{n-k} = \boldsymbol{h}_{n-k+i} \quad \text{for } 1 \le i \le k \tag{1}$$

さらに，GCD の係数 c_i は次の関係式でかける．

$$f_m \begin{pmatrix} c_k \\ c_{k-1} \\ \vdots \\ c_0 \end{pmatrix} = c_k \begin{pmatrix} f_m & & & \\ f_{m-1} & f_m & & \\ \vdots & \vdots & \ddots & \\ f_{m-k-1} & f_{m-k} & \cdots & f_m \end{pmatrix} \begin{pmatrix} 1 \\ q_{1,n-k} \\ \vdots \\ q_{k,n-k} \end{pmatrix}. \tag{2}$$

それゆえ，$\gcd(f,g) = 1 + c_{k-1}/c_k x^{k-1} + \cdots + c_0/c_k$ を得ることが可能である． ∎

近似 GCD を計算するためには，式 (1) を満たす $q_{i,j}$ を計算すればよいが，計算を効率化するため次の部分行列 $\mathcal{H}_{n-k}$ を利用する．

$$\mathcal{H}_n = \left(\begin{array}{c|ccc} \mathcal{H}_{n-k} & \tilde{\boldsymbol{h}}_{n-k+1} & \cdots & \tilde{\boldsymbol{h}}_n \\ \hline * & * & * & * \end{array}\right)$$

ここで，$\mathcal{H}_{n-k}$ は正則であり，かつ，Hankel 行列である．このとき，式 (1) を満たす $q_{i,j}$ を求める線形方程式は次でかける．

$$\mathcal{H}_{n-k}\boldsymbol{q}_i = \tilde{\boldsymbol{h}}_i \quad \text{for } 1 \le i \le k \tag{3}$$

ここで，$\boldsymbol{q}_i = (1, q_{i,2}, \ldots, q_{i,n-k})^T$ である．右辺だけが異なる線形方程式（または，1 つにまとめた一般線形方程式）を解く方法の 1 つは逆行列を求めることである．そして，近似 GCD 計算に必要なのは，線形方程式 (3) の解全体ではなく解の一部である．今回の場合は，最後の要素だけあれば近似 GCD が求められる．反復法では，解の一部を求めることは難しいので，本稿では逆行列を用いた方法で解の一部を求めることを考える．

3 近似 GCD の高速計算

Hankel 行列の逆行列を求める方法の 1 つとして，[14, 15] が知られている．この方法は，構造化行列の性質を利用して，高速計算可能な行列による線形方程式の解を利用するものである．

よく知られた方法は，$\mathcal{H}_{n-k}\boldsymbol{x} = \boldsymbol{e}_1$ と $\mathcal{H}_{n-k}\boldsymbol{y} = \boldsymbol{e}_{n-k}$ の線形方程式の解 $\boldsymbol{x}$ を $\boldsymbol{y}$ を用いる方法であり，$\mathcal{H}_{n-k}^{-1}$ は次で表現できる．

$$\begin{aligned} \mathcal{H}_{n-k}^{-1} \;=\; & \frac{1}{y_{n-k}} \Bigg\{ \begin{pmatrix} y_1 & & & \\ y_2 & y_1 & & \\ \vdots & \vdots & \vdots & \\ y_{n-k} & \cdots & \cdots & y_1 \end{pmatrix} \begin{pmatrix} x_1 & x_2 & \cdots & x_{n-k} \\ x_2 & \vdots & & \\ \vdots & & & \\ x_{n-k} & & & \end{pmatrix} \\ & - \begin{pmatrix} 0 & & & \\ x_1 & & & \\ \vdots & \cdots & \cdots & \\ x_{n-k-1} & x_{n-k-2} & \cdots & x_1 \end{pmatrix} \begin{pmatrix} y_2 & y_3 & \cdots & y_{n-k-1} \\ y_3 & \cdots & \cdots & \\ \vdots & \cdots & & \\ 0 & & & \end{pmatrix} \Bigg\}. \end{aligned} \tag{4}$$

表れる行列はすべて構造をもつ行列であり，逆行列そのものを構成するための計算量は $O(n^2 \log n)$ である．$\mathcal{H}_{n-k}^{-1}$ の最終行だけを求めようとすると，構造を有効に利用できなくなってしまうため，計算量は $O(2n^2)$ となってしまうため，特性を利用できない．

ここで，Gohberg-Heinhin inverse formula と呼ばれる次の方法を用いる．2 つの線形方程式 $\mathcal{T}_{n-k}\boldsymbol{x} = \boldsymbol{e}_1$ と $\mathcal{T}_{n-k}\boldsymbol{z} = \boldsymbol{e}_{n-k}$ の解を用いることによって，$\mathcal{H}_{n-k}^{-1}$ は次で表すことができる．

$$\begin{aligned}\mathcal{H}_{n-k}^{-1} &= \frac{1}{x_1}\Bigg\{ \begin{pmatrix} x_{n-k} & x_{n-k-1} & \cdots & x_1 \\ x_{n-k-1} & & & \\ \vdots & & & \\ x_1 & & & \end{pmatrix} \begin{pmatrix} y_{n-k} & y_{n-k-1} & \cdots & y_1 \\ & y_{n-k} & \cdots & \\ & & \ddots & \vdots \\ & & & y_{n-k} \end{pmatrix} \\ &\quad - \begin{pmatrix} y_{n-k-1} & y_{n-k-2} & \cdots & 0 \\ y_{n-k-2} & \cdots & & \\ \vdots & & & \\ 0 & 0 & \ldots & 0 \end{pmatrix} \begin{pmatrix} 0 & x_{n-k} & \cdots & x_2 \\ 0 & & & x_3 \\ \vdots & & & \\ 0 & 0 & 0 & 0 \end{pmatrix} \Bigg\}.\end{aligned}$$

式 (4) 同様，行列の積の和による表現であるが，最終行だけ必要であることを考えると，2 つ目の積の最終行は 0 ベクトルなので，計算する必要がない．さらいに 1 つ目について最終行は

$$\frac{1}{x_1} x_1 (y_{n-k}\ y_{n-k-1}\ \cdots\ y_1) = (y_{n-k}\ y_{n-k-1}\ \cdots\ y_1) \tag{5}$$

で表現されるので，$1/x_1$ を乗ずる必要なく，さらに線形方程式 $\mathcal{H}_{n-k}\boldsymbol{x} = \boldsymbol{e}_1$ を解く必要がない．

4 update

前章までは次数が正しく与えられた場合を考えた．本章では，正しくない次数 k_0 が与えられた場合について検討する．

4.1 $k < k_0$ のとき

$k < k_0$ を仮定する．このとき，$\mathcal{H}_{n-k_0}$ は正則である（Pade 近似などを参照）．

$$\mathcal{H}_{n-(k_0-1)} = \left(\begin{array}{c|c} \mathcal{H}_{n-(k_0)} & \tilde{\boldsymbol{h}}_{n-k_0+1} \\ \hline \tilde{\boldsymbol{h}}_{n-k_0+1}^T & h_{2n-2k_0+1} \end{array} \right)$$

このとき，$\mathcal{H}_{n-k_0+1}$ もまた正則である．この行列の逆行列 $\mathcal{H}_{n-k_0+1}^{-1}$ に関する関係式はすぐに導出できないが最後の列は次でかける．

$$\mathcal{H}_{n-k_0+1}^{-1} = \begin{pmatrix} \mathcal{H}_{n-k_0}^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & 0 \end{pmatrix} + \frac{1}{x_{n-k_0+1}} \boldsymbol{x}^+ \boldsymbol{z}^+$$

ただし，$\mathcal{T}_{n-k_0+1}\boldsymbol{x}^+ = \boldsymbol{e}_1$ and $\mathcal{T}_{n-k_0+1}\boldsymbol{z}^+ = \boldsymbol{e}_{n-k_0+1}$ である．

次はこれまでを整理したものである．

補題 4

1. $\mathcal{H}_{n-k}^{-1}$ の最終行は $\mathcal{T}_{n-k}\boldsymbol{z} = \boldsymbol{e}_{n-k}$ で計算できる．
2. $k < k_0$ のとき，$\mathcal{H}_{n-k_0+1}^{-1}$ の最終行は $\mathcal{T}_{n-k_0+1}\boldsymbol{z} = \boldsymbol{e}_{n-k_0+1}$. で表現される．

4.2 $k > k_0$ のとき

$\mathcal{H}_{k_0}$ は正則でない行列であるため，本方法には向かない．

4.3 計算方法

大きめに次数を評価しながら，近似 GCD 候補を計算する．Bezout-Hankel 行列を元いて計算されるのは，近似 GCD の候補そのものであるので得られた近似 GCD 候補で試し割を行い，割り切れなかった場合には次数を修正して計算を行う．

次数が小さいとわかっている場合には，数回の試行で近似 GCD 計算の候補の計算が終わるが，次数が大きい入力からスタートした場合，互いに素のときには終了までに時間を要してしまう．

参 考 文 献

[1] S. Barnett. *Greatest common divisor of two polynomials.* Linear Algebra Appl., **3**, 1970, 7–9.

[2] S. Barnett. *Greatest common divisor of several polynomials.* Proc. Camb. Phil. Soc., **70**, 1971, 263–268.

[3] S. Belhaj. Block factorization of Hankel matrices and Euclidean al gorithm. *Math. Medel. Nat. Phenom.* **5(7)**, 2010, 48–54.

[4] D. Bini and P. Boito. Structured matrix-based methods for polynomial ϵ-gcd: analysis and comparisons. *Proc. of ISSAC'07*, ACM Press, 2007, 9–16.

[5] B. Beckermann and G. Labahn, *When are two numerical polynomials relatively prime?*, J. Symb. Comput., **26** (1998), 677–689.

[6] B. Beckermann and G. Labahn, *A fast and numerically stable Euclidean-like algorithm for detecting relatively prime numerical polynomials*, J. Symb. Comput., **26** (1998), 691–714.

[7] R. Corless, P. Gianni, B. Trager and S. Watt. The singular value decomposition for polynomial systems. *Proc. of ISSAC'95*, ACM Press, 1995, 195–207.

[8] R. Corless, S. Watt and L. Zhi. *QR* factoring to compute the GCD of univariate approximate polynomials. *IEEE Trans. Signal Proces.*, **52(12)** (2004), 3394–3402.

[9] G. M. Diaz-Toca and L. Gonzalez-Vega. *Barnett's theorems about the greatest common divisor of several univariate polynomials though Bezout-like matrices. J. Symb. Compu.*, **34**, (2002), 59–81.

[10] G. M. Diaz-Toca and L. Gonzalez-Vega. Computing greatest common divisors and squarefree decompositions through matrix methods: The parametric and approximate cases, *Linear Algebra Appl.*, **412(2-3)**, (2006), 222–246.

[11] G. H. Golub and C. F. Van Loan, *Matrix computations*, Johns Hopkins Univ. Press, Baltimore, Maryland, 1989.

[12] U. Helmke and P. A Fuhrmann. *Bezoutians.* Linear Algebra Appl., **122/123/124**, 1989, 1039-–1097.

[13] G. Heinig and K. Rost, *Algebraic method for Toeplitz-like matrices and operators*, Birkhäuser, 1984.

[14] I. S. Iohvidov. *Hankel and Toeplitz matrices and forms: algebraic theory*, Birkhäuser, 1982.

[15] G. Labahn, D. K. Choi and S. Cabay, *The inverse of block Hankel and block Toeplitz matrices*, SIAM J. Comput., 19 (1), 1990, 98-123.

[16] K.Nagasaka and T.Masui, *Extended QRGCD algorithm*, Lecture Notes in Computer Science. Volume 8136, Proc. of CASC2013, Springer, 2013, 257–272.

[17] H. Ohsako, H. Sugiura and T. Torii. A stable extended algorithm for generating polynomial remainder sequence (in Japanese). *Trans. of JSIAM (Japan Society for Indus. Appl. Math.)* **7** (1997), 227–255.

[18] V. Pan. Univariate polynomials: nearly optimal algorithms for factorization and rootfinding. *Proc. of ISSAC'01*, ACM Press, 2001, 253–267.

[19] L. B. Rall. *Convergence of the Newton process to multiple solution*, Numer. Math. **9**, 1966, 23–37.

[20] T. Sasaki and M-T. Noda. *Approximate square-free decomposition and root-finding of ill-conditioned algebraic equations. J. Inform. Proces.*, **12** (1989), 159–168.

[21] M. Sanuki. *Challenge to fast and stable computation of approximate univariate GCD, based on displacement structures*, Proc. of SNC2011, ACM Press, 2011, 178–186.

[22] 讃岐 勝, べき級数演算を利用した 1 変数多項式の高速近似 GCD 計算，数式処理に掲載，日本数式処理学会, 2019 (to appear)

[23] M. Sanuki and T. Sasaki. Computing approximate GCD in ill-conditioned cases. *Proc. of SNC'07*, ACM Press, 2007, 170–179.

[24] A. Terui. An iterative method for calculating approximate GCD of univariate polynomials. *Proc. of ISSAC'09*, ACM Press, 2009, 351–358.

[25] Z. Zeng. The approximate GCD of inexact polynomials part I: a univariate algorithm. *Preprint*, 2004.

[26] L. Zhi. Displacement structure in computing the approximate GCD of univariate polynomials. *Proc. of ASCM2003 (Asian Symposium on Computer Mathematics)*, World Scientific, 2003, 288–298.