数理解析研究所講究録 第2104巻 2019年 14-19

近似GCDアルゴリズムの新たな組み合わせ
Towards a new approximate GCD algorithm

長坂耕作

Kosaku Nagasaka*

神戸大学人間発達環境学研究科

Graduate School of Human Development and Environment, Kobe University†

Abstract

We implemented the approximate GCD algorithm [KYZ06] in our LIBSNAP library, and did a couple of performance tests with other algorithms. In this preliminary report, we show the result. Moreover, Giesbrecht et al. [GHL17] proposed an iterative algorithm to compute the nearest rank-deficient matrix polynomial. We also give the outline of our algorithm to compute an approximate GCD by following their framework.

1 はじめに

本発表では，近似 GCD アルゴリズムについて取り上げるため，その定義を 2 つ与えておく。

定義 1 (近似 GCD（次数指定）)
多項式 $f(x), g(x) \in \mathbb{C}[x]$ と次数 $d \in \mathbb{Z}_{\geq 1}$ に対して，次式を満たす多項式 $d(x) \in \mathbb{C}[x]$ で次数 d のものを，$f(x)$ と $g(x)$ の d 次の近似 *GCD* という。

$$\exists \Delta_f(x), \Delta_g(x), f_1(x), g_1(x) \in \mathbb{C}[x],\ \deg(\Delta_f) \leq \deg(f),\ \deg(\Delta_g) \leq \deg(g),$$
$$f(x) + \Delta_f(x) = f_1(x)d(x),\ g(x) + \Delta_g(x) = g_1(x)d(x)$$

◁

定義 2 (近似 GCD（許容度指定）)
多項式 $f(x), g(x) \in \mathbb{C}[x]$ と許容度 $\varepsilon \in \mathbb{R}_{\geq 0}$ に対して，次式を満たす多項式 $d(x) \in \mathbb{C}[x]$ で次数最大のものを，$f(x)$ と $g(x)$ の許容度 ε の近似 *GCD* という。

$$\exists \Delta_f(x), \Delta_g(x), f_1(x), g_1(x) \in \mathbb{C}[x],\ \deg(\Delta_f) \leq \deg(f),\ \deg(\Delta_g) \leq \deg(g),$$
$$f(x) + \Delta_f(x) = f_1(x)d(x),\ g(x) + \Delta_g(x) = g_1(x)d(x),\ \|\Delta_f\|_2 < \varepsilon \|f\|_2,\ \|\Delta_g\|_2 < \varepsilon \|g\|_2$$

◁

発表者は，このような近似 GCD を求める複数のアルゴリズムの性能評価を目的とし，LIBSNAP なるライブラリの形で，QRGCD[CWZ04], ExQRGCD[NM13], UVGCD[Zen11], Fastgcd[BB07], PivQR[Boi07], GPGCD[Ter13], STLN-GCD[KYZ07] などを実装してきている。また，実装においては，それぞれのアルゴリズムの部分部分を組み替えることによる性能向上についても評価してきている。本発表の目的は，STLN-GCD の異なる版 [KYZ06] の実装とその評価を報告することと，行列多項式に対する最近特異行列計算法 [GHL17] の近似 GCD への展開についての検討の途中報告を行うことである。

*This work was supported by JSPS KAKENHI Grant Number 15K00016.

†nagasaka@main.h.kobe-u.ac.jp

2　STLN-GCD の異なる版とその評価

以下，参考文献 [KYZ07] のものを「STLN-GCD」と，参考文献 [KYZ06] のものを「STLN-GCD2」と呼ぶことにする。まず，これら 2 つのアルゴリズムの違いについて簡単に紹介しておく。

STLN-GCD では，近似 GCD を求める問題を，定義 1 の条件式を満たす多項式 $\Delta_f(x)$, $\Delta_g(x)$ に関する次の最適化問題（Structured Total Least Norm）に帰着する（$S_k(a,b)$ は，$a(x), b(x)$ の部分終結式行列）。

$$\min \|(\vec{h}_k\ E_k)\|_2 \ \textbf{subject to}\ \vec{b}_k + \vec{h}_k \in \mathrm{Range}(A_k + E_k),$$
$$(\vec{b}_k\ A_k) := S_k(f,g),\ (\vec{h}_k\ E_k) := S_k(\Delta_f, \Delta_g)$$

具体的な手順としては，初期値を部分終結式行列を用いた最小二乗法により求め，最適化本体はペナルティ法（本実装ではペナルティとして $w = 1.0e9$ を採用）で反復する。最適化問題を解いたあとで，任意の方法で近似 GCD を求める（本実装では，UVGCD の初期値として使われる方法を採用）。

一方，STLN-GCD2 では，次の最適化問題に帰着する。

$$\min \|\Delta c\|_2 \ \textbf{subject to}\ \exists \vec{x},\ A(c+\Delta c)\vec{x} = \overrightarrow{b(c+\Delta c)},$$
$$A(c) := (A_1(c)\ A_2(c)),\ (A_1(c)\ \overrightarrow{b(c)}\ A_2(c)) := S_k(f,g)$$

具体的な手順としては，初期値をラグランジュの乗数法により求め，最適化本体はペナルティ法（本実装ではペナルティとして $w = 1.0e9$ を採用）で反復する。最適化問題を解いたあとで，任意の方法で近似 GCD を求める（本実装では，UVGCD の初期値として使われる方法を採用）。ただし，STLN-GCD と異なり，$\mathbb{C}$ は $\mathbb{C}$ のままでなく，$\mathbb{R}$ に埋め込む形で計算を行う。

2.1　実験と結果

LIBSNAP では，実装に際して，なるべく原論文に忠実に記述し，コード上の最適化は基本的に実施しないが，可能な限り BLAS/LAPACK の関数を使用する方針を採用している。今回の実験は，Intel Xeon E5-2687W v4 と 256GB メモリのハードウェア上で，GNU C Compiler 5.4.0 (optimized with -O3 -march=native)，ATLAS 3.11.39 (as BLAS) + LAPACK 3.7.0 (via LAPACKE)，Ubuntu 16.04.3 LTS (x86_64, 4.4.0-97-generic) を用いて行った。

実験に用いたデータセットは，LIBSNAP での比較用に長らく用いてきたもの（定義は同じであるが，今回，次数の一部に欠落が発見されたため再生成）と，文献 [Boi07] の Boito 8.1.1, 8.3.1, 8.4.1, 8.6.1 の多項式である。まず，前者の多項式の定義を与えておく。

half degree $k = 1, 2, \ldots, 9, 10$ に対して，単位 2 ノルムを持つ次数 $10k$ の多項式を 100 組（次数 $5k$ の GCD を持ち，正規化前は係数 $\in [-99, 99] \subset \mathbb{Z}$）生成し，2 ノルムが 10^{-8} の次数 $10k$ の多項式を摂動として加え，再正規化したもの。

low degree 摂動前に次数 k の GCD を持ち，それ以外は half degree と同じ。

asymmetric 単位 2 ノルムを持つ次数 $2k$ と $18k$ の多項式の 100 組であり，それ以外は low degree と同じ。

次に，後者の多項式の定義を与えておく。

Boito 8.1.1 (Zeng's Test 2)（実験に際しては正規化を実施）

$$f(x) = \prod_{i=1}^{10}(x - \omega_i),\ g(x) = \prod_{i=1}^{10}(x - \omega_i + 10^{-i}),\ \omega_i = (-1)^i(i/2)$$

手法	再改善	次数	時間 (秒)	摂動の平均
uvgcd	refine なし	50.	0.01359	5.7729075e-9
gpgcd	refine なし	50.	0.01047	1.9088913e-8
gpgcd	UVGCD で改善	50.	0.01684	5.7729074e-9
stlngcd	refine なし	50.	0.01796	5.7976402e-9
stlngcd	UVGCD で改善	50.	0.02349	5.7729074e-9
stlngcd2	refine なし	50.	0.02218	5.7729074e-9
stlngcd2	UVGCD で改善	50.	0.02740	5.7729075e-9
uvgcdC	refine なし	50.	0.02974	5.7729075e-9
gpgcdC	refine なし	50.	0.06235	1.9088913e-8
gpgcdC	UVGCD で改善	50.	0.08054	5.7729075e-9
stlngcdC	refine なし	50.	0.07885	7.3214823e-9
stlngcdC	UVGCD で改善	50.	0.09520	5.7729074e-9
stlngcd2C	refine なし	50.	0.1081	5.7729075e-9
stlngcd2C	UVGCD で改善	50.	0.1232	5.7729074e-9

表 1: half degree ($k = 10$), single thread

Boito 8.3.1 (Zeng's Test 3) （実験に際しては正規化を実施）

$$f(x) = d(x)\left(\sum_{j=0}^{3} x^j\right),\ g(x) = d(x)\left(\sum_{j=0}^{4}(-x)^j\right)$$

ここで，$d(x)$ は，n 次で係数が $[-5, 5]$ からのランダムな整数係数多項式。

Boito 8.4.1 (Zeng's Test 1) （実験に際しては正規化を実施）

$$f(x) = f_1(x)d(x),\ g(x) = g_1(x)d(x),\ k = n/2$$

$$\begin{aligned} d(x) &= \textstyle\prod_{j=1}^{k}((x - r_1\alpha_j)^2 + r_1^2\beta_j^2),\ r_1 = 0.5,\ r_2 = 1.5, \\ f_1(x) &= \textstyle\prod_{j=1}^{k}((x - r_2\alpha_j)^2 + r_2^2\beta_j^2),\ \alpha_j = \cos(j\pi/n), \\ g_1(x) &= \textstyle\prod_{j=k+1}^{n}((x - r_1\alpha_j)^2 + r_1^2\beta_j^2),\ \beta_j = \sin(j\pi/n). \end{aligned}$$

Boito 8.6.1 （実験に際しては正規化を実施）

$$f(x) = (x^3 + 3x - 1)(x - 1)^n,\ g(x) = f'(x)$$

表 1 から表 6 は，それぞれの実験結果である。実験結果からは，UVGCD の性能の高さが垣間見え，STLN-GCD2 は STLN-GCD よりもよい結果と見えるが，計算時間まで考慮すると UVGCD は及ばない。

3 検討中の方法

ISSAC 2017 では，行列多項式に対する新しい最近特異行列計算法 [GHL17] が提案された。その方法の本来の目的は，与えられた $A = (A_{ij}) \in \mathbb{R}[t]^{n\times n}$（正則）に対し，次式を満たす $\Delta A = (\Delta A_{ij}) \in \mathbb{R}[t]^{n\times n}$ を求めることである。

$$\min \|\Delta A\|,\ \det(A + \Delta A) = 0,\ \deg(\Delta A_{ij}) \le \deg(A_{ij})$$

手法	再改善	次数	時間（秒）	摂動の平均
uvgcd	refine なし	10.	0.03072	1.9953791e-9
gpgcd	refine なし	10.	0.02084	1.5108537e-8
gpgcd	UVGCD で改善	10.	0.02895	1.9953791e-9
stlngcd	refine なし	10.	0.03118	2.0205575e-9
stlngcd	UVGCD で改善	10.	0.03794	1.9953792e-9
stlngcd2	refine なし	10.	0.04295	1.9953792e-9
stlngcd2	UVGCD で改善	10.	0.04962	1.9953791e-9
uvgcdC	refine なし	10.	0.05640	1.9953791e-9
gpgcdC	refine なし	10.	0.1125	1.5108523e-8
gpgcdC	UVGCD で改善	10.	0.1357	1.9953791e-9
stlngcdC	refine なし	10.	0.1403	2.4758635e-9
stlngcdC	UVGCD で改善	10.	0.1613	1.9953792e-9
stlngcd2C	refine なし	10.	0.1996	1.9953792e-9
stlngcd2C	UVGCD で改善	10.	0.2174	1.9953791e-9

表 2: low degree ($k = 10$), single thread

当該アルゴリズムは，1) $\mathbb{R}[t]^{n\times n}$ の問題を，$\mathbb{R}^{s\times t}$ の問題に帰着，2) $\mathbb{R}^{s\times t}$ への埋込は，多項式の畳み込み行列（Toeplitz）を使用，3) $\mathbb{R}^{s\times t}$ 上の，SLRA (Structured Low Rank Approximation) と解釈（ただし，Frobenius-norm での最小化），4) 等式条件付最小化問題に対する Newton 法を使用，となっている。特に最後の Newton 法部分に関しては，$A \in \mathbb{R}^{s\times t}$ に対し，$(A + \Delta A)\vec{b} = 0$, $\|\vec{b}\| = 1$ を満たす $\Delta A \in \mathbb{R}^{s\times t}$ と $\vec{b} \in \mathbb{R}^t$ を探索するのだが，乗数 $\vec{\lambda}$ と未知ベクトル $\vec{x}$（A の構造へ）からの関数 L に関する次式で反復計算を行う（初期値は，SVD や STLN や Lift-and-Project など）ものとなっている。

$$\nabla^2 L \begin{pmatrix} \Delta_x \\ \Delta_\lambda \end{pmatrix} = -\nabla L$$

近似 GCD の多くの方法では，Sylvester 行列や部分終結式行列の SLRA (Structured Low Rank Approximation) と解釈して，近似 GCD を求めているため，上記の方法は，ステップ 2 以降をそのまま近似 GCD の算法として活用することが可能である。懸念されることは，近似 GCD でよく使われるのは 2-norm であり，Frobenius-norm でないことが挙げられる。しかしながら，少なくとも，次の近似 GCD を本方法で算出可能であることは確認済みである。

$$\varepsilon\text{-GCD}(0.9999x^2 + 1.9999x + 1.0001,\ 1.0001x^2 - 0.9999)$$

4 まとめ

STLN-GCD2 について，原論文では STLN-GCD との優劣に関しての記載がないようだが，今回の実験では，STLN-GCD2 の方が良い結果となっている。特に，初期値が改善され，全般の計算時間に短縮傾向がみられる。ただ，どちらにせよ UVGCD の優秀さが際立っている。

検討中の方法については，原論文によれば二次収束なので速度に期待が持てるため，特に，次数指定型の近似 GCD で性能評価をしたいと考えている。

手法	再改善	次数	時間（秒）	摂動の平均
uvgcd	refine なし	10.	0.01409	2.4682949e-9
gpgcd	refine なし	10.	0.02556	1.495299e-8
gpgcd	UVGCD で改善	10.	0.03374	2.4682949e-9
stlngcd	refine なし	10.	0.07450	7.8231015e-8
stlngcd	UVGCD で改善	10.	0.08287	2.4682949e-9
stlngcd2	refine なし	10.	0.04199	2.4682949e-9
stlngcd2	UVGCD で改善	10.	0.04870	2.4682949e-9
uvgcdC	refine なし	10.	0.03364	2.4682949e-9
gpgcdC	refine なし	10.	0.1421	1.495299e-8
gpgcdC	UVGCD で改善	10.	0.1646	2.4682949e-9
stlngcdC	refine なし	10.	0.3887	1.153819e-6
stlngcdC	UVGCD で改善	10.	0.4171	2.4682949e-9
stlngcd2C	refine なし	10.	0.1914	2.4682949e-9
stlngcd2C	UVGCD で改善	10.	0.2104	2.4682949e-9

表 3: asym degree ($k = 10$), single thread

参 考 文 献

[BB07] Dario A. Bini and Paola Boito. Structured matrix-based methods for polynomial ϵ-gcd: Analysis and comparisons. In *Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation*, ISSAC '07, pages 9–16, New York, NY, USA, 2007. ACM.

[Boi07] P. Boito. *Structured Matrix Based Methods for Approximate GCD*. Ph.D. Thesis. Department of Mathematics, University of Pisa, Italia, 2007.

[CWZ04] Robert M. Corless, Stephen M. Watt, and Lihong Zhi. *QR* factoring to compute the GCD of univariate approximate polynomials. *IEEE Trans. Signal Process.*, 52(12):3394–3402, 2004.

[GHL17] Mark Giesbrecht, Joseph Haraldson, and George Labahn. Computing the nearest rank-deficient matrix polynomial. In *ISSAC'17—Proceedings of the 2017 ACM International Symposium on Symbolic and Algebraic Computation*, pages 181–188. ACM, New York, 2017.

[KYZ06] Erich Kaltofen, Zhengfeng Yang, and Lihong Zhi. Approximate greatest common divisors of several polynomials with linearly constrained coefficients and singular polynomials. In *ISSAC 2006*, pages 169–176. ACM, New York, 2006.

[KYZ07] Erich Kaltofen, Zhengfeng Yang, and Lihong Zhi. Structured low rank approximation of a Sylvester matrix. In *Symbolic-numeric computation*, Trends Math., pages 69–83. Birkhäuser, Basel, 2007.

[NM13] Kosaku Nagasaka and Takaaki Masui. Extended qrgcd algorithm. In *Proceedings of the 15th International Workshop on Computer Algebra in Scientific Computing - Volume 8136*, CASC 2013, pages 257–272, New York, NY, USA, 2013. Springer-Verlag New York, Inc.

[Ter13] Akira Terui. GPGCD: an iterative method for calculating approximate GCD of univariate polynomials. *Theoret. Comput. Sci.*, 479:127–149, 2013.

d	perturb.				real time (msec.)			
	STLN	STLN2	UV	GP	STLN	STLN2	UV	GP
1	8.86e+3*	9.10e+3	3.98e-12	2.06e+3*	32.242	36.597	0.504	12.014
2	9.70e+3*	5.84e+3	2.65e-10	4.42e-1*	20.145	7.568	0.418	8.640
3	5.03e+3*	5.57e+3	1.79e-8	1.25e+0*	16.403	12.264	0.482	8.299
4	9.41e+2*	2.63e+3	9.88e-7	2.05e-1*	23.100	5.386	0.328	8.687
5	4.76e+3*	5.28e-5	5.28e-5	5.54e-1*	13.693	4.166	0.297	7.086
6	2.15e-3*	2.15e-3	2.15e-3	3.65e-1*	12.517	3.581	0.282	6.464
7	8.34e-2*	8.34e-2	8.34e-2	5.24e+0*	14.880	7.492	0.363	5.826
8	2.03e+0	2.03e+0	2.03e+0	2.03e+0*	3.580	0.855	0.272	5.662
9	4.70e+1	4.70e+1	4.70e+1	4.70e+1	0.555	0.562	0.394	1.141
10	7.74e+2	7.74e+2	7.74e+2	7.74e+2	0.864	0.240	0.373	0.285

表 4: Boito 8.1.1（「*」は，収束しなかった可能性あり）

n	relative perturb. for our impl.				real time (msec.)			
	STLN	STLN2	UV	GP	STLN	STLN2	UV	GP
50	4.10e-16*	5.55e-16	1.57e-16	2.68e-15	1.466	2.391	2.550	0.912
100	5.89e-16	6.59e-16	1.72e-16	3.31e-15	5.416	10.665	6.817	2.474
200	5.96e-16	5.84e-16	1.95e-16	7.20e-15	27.461	52.627	30.608	8.596
1000	5.49e-16*	6.71e-16	1.84e-16	2.86e-14	10988.350	5946.857	1038.688	524.620

表 5: Boito 8.3.1（「*」は，未収束あり（収束分のみ未掲載））

[Zen11] Zhonggang Zeng. The numerical greatest common divisor of univariate polynomials. In *Randomization, relaxation, and complexity in polynomial equation solving*, volume 556 of *Contemp. Math.*, pages 187–217. Amer. Math. Soc., Providence, RI, 2011.

n	relative perturb. for our impl.				real time (msec.)			
	STLN	STLN2	UV	GP	STLN	STLN2	UV	GP
12	1.08e-13	1.47e-13	1.69e-14	5.98e-14	70.351	21.658	1.181	7.437
14	3.67e-13	5.87e-13	5.01e-14	3.14e-13	96.434	30.099	1.861	30.529
16	1.61e-12	2.09e-12	2.20e-13	1.16e-12	135.023	36.530	1.603	42.590
18	1.50e-11	2.85e-11	9.53e-13	6.75e-12	169.493	46.015	2.231	48.496
20	5.73e-11	1.27e-10	2.78e-12	2.95e-11	179.018	57.732	2.497	54.944

表 6: Boito 8.4.1（「*」は，収束しなかった可能性あり）