数理解析研究所講究録
第1652巻 2009年 26-32

安定化理論に基づくInterval Trace Liftingについて

白柳 潔*
東海大学理学部
KIYOSHI SHIRAYANAGI
SCHOOL OF SCIENCE, TOKAI UNIVERSITY

関川 浩†
日本電信電話株式会社　NTT コミュニケーション科学基礎研究所
HIROSHI SEKIGAWA
NTT COMMUNICATION SCIENCE LABORATORIES, NIPPON TELEGRAPH AND TELEPHONE CORPORATION

1　はじめに

安定化理論 [12] は、近似計算で実行すると不安定となるアルゴリズムに対し、それを変形して近似計算で実行しても誤差の影響を抑制し、安定な出力が得られるようにするための理論である。本論文では、その安定化理論を、グレブナ基底計算の高速化技巧の一つである Trace Lifting[13] に応用する。Trace Lifting とは、グレブナ基底を計算するブッフバーガーのアルゴリズムにおいて、ある素数 p について、mod p で S-polynomial の正規形を計算し、それが 0 になったら真の 0 とみなす方法である。本論文で提案する方法は、その mod p の代わりに浮動小数点近似を行うものである。その正しさは、安定化理論によって保証される。従来の Trace Lifting は有理数体上のみにしか使えないが、今回提案の Trace Lifting は実数体上でも使え、より広い範囲に応用できる可能性を開くことができた。

2　安定化理論

2.1　理論の復習

次のアルゴリズムを対象に安定化理論の復習を簡単に行う。詳細は [8, 7] を参照されたい。

- データは、すべて多項式環 $R[x_1, \ldots, x_m]$ の元からなる。R は実数体の部分体である。
- データ間の演算は、$R[x_1, \ldots, x_m]$ 内の加減乗または剰余計算である。
- データ上の述語は、不連続点をもつとすればそれは 0 のみである。

*shirayan@ss.u-tokai.ac.jp
†sekigawa@theory.brl.ntt.co.jp

述語の不連続点が 0 という意味は、If "$C = 0$" then ... else ... のように、値が 0 か否かによって分岐が別れることである。上記クラスのアルゴリズムを、不連続点 0 の代数的アルゴリズムとよぶ。ほとんどの数式処理のアルゴリズムはこのクラスに入るか、このクラスのアルゴリズムに変換可能である。

さて、安定化の 3 つのポイントは、

- アルゴリズムの構造は変えない。
- データ領域において、ふつうの係数を区間係数に変える。
- 述語の評価の直前で、区間係数のゼロ書換えを行なう。

である。すなわち、安定化されたアルゴリズムは次のようになる。

区間領域 データ領域は区間係数多項式の集合。区間係数は $[A, \alpha]$ なる形で、$A \in R$, α は非負の実数。$[A, \alpha]$ は集合 $\{x \in R \mid |x - A| \leq \alpha\}$ を意味する。

区間演算 二項演算 $* \in \{+, -, \times, /\}$ に対し、

$$[A, \alpha] * [B, \beta] = [A * B, \gamma_*].$$

ここに、γ_* は次を満たす。

$$|x - A| \leq \alpha, |y - B| \leq \beta \Rightarrow |x * y - A * B| \leq \gamma_*.$$

ゼロ書換え 不連続点 0 をもつ述語を評価する直前で、各区間係数 $[C, \gamma]$ に対し、

$|C| \leq \gamma$ ならば $[C, \gamma]$ を [0,0] に書き換えよ。

$|C| > \gamma$ ならばそのままとせよ。

区間演算の具体的な定義については、文献 [1] を参照されたい。

今、入力 $f \in R[x_1, \ldots, x_m]$ を

$$f = \sum_{i_1, \ldots, i_m} a_{i_1 \ldots i_m} x_1^{i_1} \cdots x_m^{i_m}$$

と表したとき、f に対する近似列 $\{Int(f)_j\}_j$ を

$$Int(f)_j = \sum_{i_1, \ldots, i_m} [(a_{i_1 \ldots i_m})_j, (\alpha_{i_1 \ldots i_m})_j] \, x_1^{i_1} \cdots x_m^{i_m}$$

で定義する。ここに、すべての $i_1, \ldots, i_m$ について、

$$|a_{i_1 \ldots i_m} - (a_{i_1 \ldots i_m})_j| \leq (\alpha_{i_1 \ldots i_m})_j \text{ for } \forall j$$

$$(\alpha_{i_1 \ldots i_m})_j \to 0 \text{ as } j \to \infty$$

このとき、単に

$$Int(f)_j \to f$$

と書く。

さて、$\mathcal{A}$ を安定化したアルゴリズムを $Stab(\mathcal{A})$ と書くと、次が安定化理論の基本定理である。

定理 1 (安定化理論の基本定理) $\mathcal{A}$ は不連続点 0 の代数的アルゴリズムで、入力 $f \in R[x_1, \ldots, x_m]$ に対し正常終了するとせよ。このとき、f に対する任意の近似列 $\{Int(f)_j\}_j$ に対し、ある n が存在して、$j \geq n$ ならば、$Stab(\mathcal{A})$ は $Int(f)_j$ に対し正常終了し、

$$Stab(\mathcal{A})(Int(f)_j) \to \mathcal{A}(f).$$

簡明を期すため、入力は一つだけの多項式にしているが、入力はもちろん、多項式の有限集合でもよい。主題のグレブナ基底の場合も、入力は多項式の集合である。

2.2 応用

安定化理論の応用にはおよそ 2 通りの方向性があると考えられる。1 つは、安定化に至る精度を予め評価しておいて、その精度（あるいはそれ以上）で実行するもの。もう 1 つは、初期精度を適当に定めてそれから始め、出力の妥当性が得られるまで精度を上げていくものである。昨年提案したグレブナ基底変換法 [9, 10, 11]、今回提案する Interval Trace Lifting はいずれも後者の方向性に属する。本節では、前者の研究について一言言及しておく。

安定化に至る精度を見積もる問題を精度問題と呼び、これは安定化理論の未解決問題の一つであった。ところが最近、Khungurn との共同研究で、多項式の最大公約式 (GCD) を計算するアルゴリズムに関して進展があった [3, 4]。安定化されたアルゴリズムを浮動小数点計算で実行したときの実行過程が、元のアルゴリズムを正確演算で実行したときの実行過程と一致する最小の入力精度を Minimum Converging Precision (MCP) と呼ぶ。まず、GCD 計算のアルゴリズムの場合、MCP を一般に計算する関数は存在しないことを証明した。更に、ユークリッドの互除法について、特定な係数領域 (整数環 $\mathbb{Z}$, 有理数体 $\mathbb{Q}$, 環 $\mathbb{Z}[\xi]$ (ξ はある代数的整数) など) の場合に、その MCP の上界を与えた。シルベスター行列を QR 分解して GCD を求めるアルゴリズムに対しては、MCP については未解決であるものの、正しい次数を得る最小精度や正しい台を得る最小精度についてはその上界を与えた。

3 Interval Trace Lifting

$R = \mathbb{Q}$ のとき、グレブナ基底を計算するブッフバーガーのアルゴリズムに対して Traverso が提唱した Trace Lifting[13] は以下の通りである。

> S-polynomial の正規形を計算する部分において、ある素数 p について、mod p で計算した結果が 0 になったら、真の 0 とみなして $\mathbb{Q}$ 上での正規形計算を省略する。0 にならなかったら、改めて $\mathbb{Q}$ 上で正規形を計算し、0 でないことを確かめてグレブナ基底の元の候補とする。

それを文献 [5] の記法に従って、アルゴリズムの形式に書くと、

Trace Lifting
$Spoly(\phi_p(g_i), \phi_p(g_j)) \longrightarrow^*_{\phi_p(G)} \bar{r}$
If $\bar{r} \neq 0$ then $Spoly(g_i, g_j) \longrightarrow^*_G r$
  If $r \neq 0$ then $G := G \cup \{r\}$

となる。ここに、$Spoly$ は S-polynomial を示す。 また、

$$\mathbb{Q}_{<p>} = \{a/b \mid a, b \in \mathbb{Z},\ p \not| b\}$$

とおくと、ϕ_p は $\mathbb{Q}_{<p>}$ から p 元体への写像で、$a/b \in \mathbb{Q}_{<p>}$ のとき、$\phi_p(a/b) = (a \bmod p)/(b \bmod p)$ によって定義される。多項式 f に対し、$\phi_p(f)$ は自然に定義されるもので、f の各係数が $\mathbb{Q}_{<p>}$ に入っていれば、それらに ϕ_p を施した多項式である。多項式集合 A に対し、$\longrightarrow_A^*$ は、mod A で正規形まで簡約することを示す。G は最終的にグレブナ基底となる集合の局所変数である。

本論文で提案する方法は、「mod p の係数」の代わりに「精度 μ の区間係数」を使う。すなわち、ゼロ書換えで 0 となった区間係数の正規形を 0 とみなす方法である。この Trace Lifting を Interval Trace Lifting と呼ぶ。アルゴリズムの形式で書くと、

Interval Trace Lifting
$Spoly([g_i]_\mu, [g_j]_\mu) \longrightarrow_{[G]_\mu}^* [r]_\mu$
If $Zero\text{-}Rewrite([r]_\mu) \neq 0$ then $Spoly(g_i, g_j) \longrightarrow_G^* r$
  If $r \neq 0$ then $G := G \cup \{r\}$

となる。ここに、多項式 f に対し、$[f]_\mu$ は、f の各係数を精度 μ の区間係数に変換した多項式を表す。多項式の集合 F に対しては、$[F]_\mu = \{[f]_\mu \mid f \in F\}$ と定義される。$Zero\text{-}Rewrite([r]_\mu)$ は、区間係数多項式 $[r]_\mu$ の各区間にゼロ書換えを実行したものである。

ここで、Interval Trace Lifting を利用して、グレブナ基底を計算する全体のアルゴリズムを記述する。$R = \mathbb{R}$ とする。

Interval Trace Lifting によるグレブナ基底計算アルゴリズム
Input: $F \subset R[x_1, \ldots, x_m]$ (有限部分集合)、項順序 $<$
Output: $< F >$ の $<$ に関するグレブナ基底
$\mu := M$ (浮動小数点精度の初期値)
again
loop
$G_\mu \leftarrow$ 精度 μ での Interval Trace Lifting によって得られたグレブナ基底候補
  if G_μ が $< F >$ の $<$ に関する真のグレブナ基底、i.e.,
    (Every S-polynomial of G_μ)$\longrightarrow_{G_\mu}^* 0$ (グレブナ基底性)
    かつ
    (Every element of F)$\longrightarrow_{G_\mu}^* 0$ ($F \subset < G_\mu >$)
  then return G_μ else μ を上げて goto again
  endif
endloop

Interval Trace Lifting において、安定化理論により、精度が十分大きければ、真に 0 でない正規形を誤って 0 とみなすことはなくなる。従って、上記の Interval Trace Lifting によるグレブナ基底計算アルゴリズムは必ず停止し、正しいグレブナ基底を返す。候補 G_μ に対し、$< F >=< G_\mu >$ を確かめなければならないが、上記のアルゴリズムにおいて $F \subset < G_\mu >$ をチェックするだけでよいのは、G_μ の構成法から必ず $< F > \supset G_\mu$ がいえているからである。

4 計算機実験

Interval Trace Lifting によるグレブナ基底計算アルゴリズムを実装し、実際に計算機で実験したのでその一部を報告する。使用システムは Maple 10、計算機環境は Dell Dimension DC051 (Intel(R) Pentium 4

CPU: 3.00GHz, RAM: 2.99GHz, 0.99GB) である。ここでは、例題を 5 つ挙げる。いずれも、与えられた多項式集合 F に対し、イデアル $< F >$ の plex のグレブナ基底を求めるという問題である。ここに、plex は lexicographic order である。ここでは、グレブナ基底といえば、簡約化されたグレブナ基底 (reduced Gröbner basis) を指す。

1. $F = \{f_1, f_2, f_3\}$, ここに $f_1 = \frac{1}{7}x^2 - \frac{3265483908546 52}{272974017239}x + \frac{1263781236281}{712638126}y^2 + \frac{26872672361827}{7263188218281}z^2$, $f_2 = \frac{3}{8}xy + \frac{12367812638123}{76381236821318 2}yz - \frac{63812638126}{7726381283 1}y$, $f_3 = \frac{4}{9}x + \frac{327091270979304}{241223785460421}y + \frac{184670315953092 03}{3184054590 32}z - \frac{3563180636931413 19}{643656180641810 9}$.
2. $F = \{(\sqrt{2}+\sqrt{5})x^3y + \sqrt{3}xy + \sqrt{7}, (\sqrt{3}-\sqrt{2})x^2y^2 - \sqrt{7}xy + \frac{1}{\sqrt{11}}\}$.
3. $F = \{ex + \sqrt{2}y + \sqrt{3}z, exy + \sqrt{5}yz + \sqrt{3}zx, xyz - e\}$, ここに e は Napier's number (2.71828...).
4. $F = \{\sqrt{2}ex^2 + xy^2 - z + 1/4, \sqrt{3}x + y^2z + 1/2, \sqrt{5}ex^2z - 1/2\,x - y^2\}$.
5. $F = \{(\sqrt{2}+\sqrt{3})x^{30} + \sqrt{5} - 1, \sqrt{7}xy + \sqrt{11} + \sqrt{13}\}$.

例 1,2 はそれぞれ文献 [6] にある例かその変形であり、例 3 はグレブナ基底計算ベンチマークの一つ cyclic3 の係数を変えたもの、例 4 はブッフバーガーの教科書 [2](p.214) にある例に係数変形を施したものである。最後の例 5 は、Interval Trace Lifting の効果が発揮できるように人工的に作った例である。

まず、実験に使ったコマンド群を記す。

R_Basis: 自前でブッフバーガーアルゴリズムを実装したもの

itl_Basis: R_Basis の中に Interval Trace Lifting を適用したもの (すなわち、今回提案の、Interval Trace Lifting によるグレブナ基底計算アルゴリズムを実装したもの)

Maple_Basis: Maple 10 の Groebner package の組込関数 (Basis)

Interval Trace Lifting の効果をみるために、あえて R_Basis を作り、その中で Interval Trace Lifting を取り込んだ itl_Basis と R_Basis を比較した。同じ条件下で比較するためである。Maple_Basis は、参考までに R_Basis と比較するために導入した。

Maple における簡単な実行例を示す。

```
> R_Basis([x^2-y-1,x^2+x*y^2-5],plex(x,y));
```

$[-4 + y - y^4 - 5y^3 - 4y^2 + 16x, 16y^2 - 128y - 16y^5 - 16y^4 + 256], 0.047$

2 番目の出力は cpu time で、0.047 秒であったことを示す。

```
> itl_Basis([x^2-y-1,x^2+x*y^2-5],plex(x,y),1,1);
```

(入力の 3 番目の引数は初期精度桁、4 番目は精度上げ幅を示す。この例では、精度 1 桁から始めて、1 桁ずつ上げていくことを表す。)

$[-4 + y - y^4 - 5y^3 - 4y^2 + 16x, 16y^2 - 128y - 16y^5 - 16y^4 + 256], 2, 1, 0.72$

(2 番目の出力は、精度を上げていって初めて成功、つまり、真のグレブナ基底に到達した精度桁 (これを Minimum precision, 略して MP と書く)、3 番目の出力は Interval Trace Lifting によって切り捨てられた回数、つまり、0 とみなされて計算を省略した回数 (cut 数) を表す。4 番目は cpu time である。)

```
> Maple_Basis([x^2-y-1,x^2+x*y^2-5],plex(x,y));
```

$[y^5 - y^2 - 16 + 8y + y^4, -4 + y - y^4 - 5y^3 - 4y^2 + 16x], 0.657$

(2 番目の出力は cpu time である。)

さて、計算機実験の結果を表にまとめると次のようになる。

例	itl_Basis (cpu at MP)	(cut)	(MP)	R_Basis (cpu)	Maple_Basis (cpu)
1	0.24	0	2	0.13	0.02
2	0.48	0	6	0.30	4.53
3	0.13	1	2	0.03	184.63
4	0.58	1	2	0.17	0.28
5	37.06	29	3	42.95	17.70

表において、cpu at MP は、精度が MP のときにかかった cpu time を表す。cpu time の単位はすべて秒である。

考察：今回の実験では、Interval Trace Lifting が著しく有効であることを示す例はなかった。これは、ゼロ書換えによって切り捨てられた回数 (cut 数) が、ほとんどの場合、少数であったことによると考えられる。cut 数が多ければ多いほど、それだけ $\mathbb{R}$ 上の正確演算を省略することができるので、Interval Trace Lifting の効果が期待できる。実際、例 5 では cut 数が多く、R_Basis よりも高速であった。なお、この例の最初の多項式は 30 次式で cut が 29 回であったが、実験によって、次数を n にすると cut 数は $n-1$ になるという予想が得られた。真のグレブナ基底であるかどうかをチェックする部分は、全体と比べてそれほど時間はかからなかった。ほとんどの場合、1 割未満である。これは、候補が真のグレブナ基底か、それに近い確率が高いためと思われる。

5 おわりに

Interval Trace Lifting によるグレブナ基底計算アルゴリズムを提案した。従来の Trace Lifting は有理数体上のみにしか使えないが、Interval Trace Lifting は実数体上でも使え、より広い範囲に応用できる可能性を開くことができた。今後は、更に計算機実験をして、cut 数が多い実例など、Interval Trace Lifting が有効に働く実例を見つけていきたい。

参 考 文 献

[1] Alefeld, G. and Herzberger, J.: *Introduction to Interval Computations, Computer Science and Applied Mathematics, Academic Press* (1983).

[2] Buchberger, B.: Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory, *Chapter 6 in Multidimensional Systems Theory (N. K. Bose ed.), D. Reidel Publishing Company* (1985), 184-232.

[3] Khungurn, P.: Sharayanagi-Sweedler Algebraic Algorithm Stabilization and Polynomial GCD Algorithms, *Master Eng. Thesis at MIT* (2007).

[4] Khungurn, P., Sekigawa, H. and Shirayanagi, K.: Minimum Converging Precision of the QR-Factorization Algorithm for Real Polynomial GCD, *Proc. ISSAC2007* (2007), 227–234.

[5] 野呂、横山: グレブナー基底の計算 基礎篇 計算代数入門, 東京大学出版会 (2003).

[6] Shirayanagi, K.: Floating Point Gröbner Bases, *Mathematics and Computers in Simulation*, **42** 4-6 (1996), 509–528.

[7] 白柳: アルゴリズムの安定化理論, 数式処理, **5** 2 (1997), 2–21.

[8] 白柳: 不安定なアルゴリズムを安定化する, 情報処理 **39** 2 (1998), 111–115.

[9] 白柳, 関川: 安定化理論における台収束の応用について, 数理解析研究所講究録, **1568** (2007), 20–26.

[10] 白柳, 関川: 安定化理論に基づくグレブナ基底変換法, 数式処理, **14** 2 (2007), 13–17.

[11] Shirayanagi, K. and Sekigawa, H.: A new Gröbner basis conversion method based on the stabilization techniques, *International Conference on Applications of Computer Algebra (ACA 2007)*, presented on July 21, 2007.

[12] Shirayanagi, K. and Sweedler, M.: A Theory of Stabilizing Algebraic Algorithms, *Technical Report 95-28, Mathematical Sciences Institute, Cornell University* (1995), 92 pages.

[13] Traverso, C.: Gröbner trace algorithms, *Proc. ISSAC 1988* (1988), 125–138.