数理解析研究所講究録
第1814巻 2012年 29-35

安定化理論に基づくISCZ法の有効性について

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 [6] 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。[4, 5] では、その安定化理論の新しい応用として、なるべく正確計算を軽減させながらも最終的に正確係数の正しい出力を得るための手法 ISCZ 法を提案し、Buchberger アルゴリズムに適用して得られた実験結果を報告した。そこでは、有理係数よりも無理係数の方が有効であることはわかったが、もともとすべてを正確演算で実行した結果と比べて、必ずしも優位性を主張することができなかった。本論文では、その原因を追求し、その結果から、ISCZ 法が有効であるための条件について考察する。

2節で、安定化理論と ISCZ 法について復習する。3節で、ISCZ 法を Buchberger アルゴリズムに適用した実験結果を、評価したシンボルの個数を観点に分析し、本手法の有効性について検討する。

2 復習

2.1 安定化理論

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 [3, 6] 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@tokai-u.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呼ぶ。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。本定理の Buchberger アルゴリズムに特化した証明は [2] に、より一般的な場合に対する厳密な証明は [6] にある。

2.2 ISCZ 法

2.2.1 シンボル付き区間

区間と形式的なシンボルを組み合わせた係数（シンボル付き区間）を導入する。区間は、従来と同様の意味の区間である。シンボルは、アルゴリズム実行中に現れる係数の log（記録）を取るのに使われる。例えば、入力係数が 1/3 と 1/9 だったとしよう。これらの精度 3 の（円形）区間はそれぞれ [.333, .0005] と [.111, .0005] である。さて、1/3 に対するシンボルを s、1/9 に対するシンボルを t として、区間と組み合わせると、それぞれ、$[[.333, .0005], s]$ と $[[.111, .0005], t]$ となる。次に、これらの間の演算、例えば、加算を、

$$[[.333, .0005], s] + [[.111, .0005], t] = [[.333, .0005] + [.111, .0005], \dot{+}(s, t)]$$

と定義する。$[.333, .0005] + [.111, .0005]$ に対しては通常の区間演算を使う。シンボル部分 $\dot{+}(s, t)$ は再び形式的なシンボルで、加算を実施したことを記録できれば何でもよい。効率的なシンボル付けについては 2.2.3 節で議論する。

アルゴリズム終了後、最終的なシンボルを正確係数に復元する。先の簡単な例で言えば、もし最終的なシンボルが $\dot{+}(s, t)$ であったとすれば、s に 1/3 を、t に 1/9 を代入し、$\dot{+}$ には加算の意味を与えて、$1/3+1/9 = 4/9$ と復元する。

シンボル付き区間のことを interval-symbol、あるいは単に IS と呼ぶ。

2.2.2 手続き

$\mathcal{A}$ を不連続点 0 の代数的アルゴリズムとする。ISCZ 法の手続きは次の通りである。

R-to-IS 各入力係数 a をペア $[[\tilde{a}, \alpha], Symbol_a]$ に変換する。ここに、$[\tilde{a}, \alpha]$ は a の予め定められた精度の区間、$Symbol_a$ は a を表すシンボル（以下、入力シンボルと呼ぶ）である。

IS 演算 IS 間の演算を次のように実行する：

$$[[A, \alpha], s] + [[B, \beta], t] = [[A, \alpha] + [B, \beta], \dot{+}(s, t)]$$

$$[[A, \alpha], s] - [[B, \beta], t] = [[A, \alpha] - [B, \beta], \dot{-}(s, t)]$$

$$[[A, \alpha], s] \times [[B, \beta], t] = [[A, \alpha] \times [B, \beta], \dot{\times}(s, t)]$$

すなわち、区間部分については区間演算を用い、シンボル部分については加算、減算、乗算の形式的なシンボル $\dot{+}$, $\dot{-}$, $\dot{\times}$ を使って、どういう演算が行なわれたかを記録する。

正しいゼロ書換え 任意のIS $[[E,\epsilon],s]$ に対し、$|E| \leq \epsilon$ ならば、s をそれに対応する実数 $r(s)$ に復元する。もし、$r(s)=0$ ならば、次のステップに進む。そうでなければ、精度を上げて **R-to-IS** に戻る。

IS-to-R 出力のシンボル部分の中の各入力シンボルにそれぞれ対応する入力係数を代入し、演算シンボルに演算の意味を与えて実数値に復元する。

この手法をISCZ法 (IS method with *correct* zero rewriting) と呼ぶ。効率化のために、**正しいゼロ書換え**において、後の再利用のために s の実数値 $r(s)$ を記憶しておくのも一法である。

さて、ある精度 j で $Int(f)_j$ を $Stab(\mathcal{A})$ に入力したときの実行過程は、もしアルゴリズム中のすべてのゼロ書換えが正しいならば、真の出力 f を $\mathcal{A}$ に入力したときの実行過程と完全に一致する。ISCZ法では、各ゼロ書換えにおいて、それが正しいかどうかを確認する。さらに、定理1により、すべてのゼロ書換えが正しくなる精度が存在する。従って、ISCZ法は有限ステップで終了し、その出力の各IS係数のシンボルは正しい正確係数を与える。

これを定理にまとめる。

定理 2 (ISCZ法の停止性と正当性) $\mathcal{A}$ が入力 I で正常終了するとせよ。このとき、$\mathcal{A}$ に対する *ISCZ* 法は、常に有限ステップで終了し、正しい結果、すなわち、$\mathcal{A}(I)$ の出力と同じ結果を与える。

ISCZ法の利点は、ISZ法 [7, 8, 4] と違い、出力の正当性を確認する必要がないことである。任意のIS $[[E,\epsilon],s]$ に対し、$|E| \leq \epsilon$ でない限り、正確計算をスキップすることができ、浮動小数点計算だけで済む。換言すれば、ゼロでない係数についての正確計算を省略することができる。従って、本手法は、$|E| > \epsilon$ の場合が $|E| \leq \epsilon$ の場合よりも多ければ多いほど有効であるということができる。

2.2.3 シンボルリスト

ISCZ法はISのシンボル部分の膨張を招くことがある。ここでは、それを防ぐための一つの工夫について述べる。

基本は、シンボルの代わりに数字を用いることである。すなわち、ISの代わりにペア $[[A,\alpha],M]$ を用いる。ここに、$[A,\alpha]$ は区間、M は入力シンボルまたは整数である。さらに、各ペアが他のペアからどのように構成されたかを示すリストを用意する。これをシンボルリストと呼ぶ。アルゴリズム終了後、このシンボルリストからISを復元する。

1. 2.2.2節で説明した **R-to-IS** 変換を行なう。
2. (初期化) N を0とし、シンボルリスト $\mathcal{L}$ を空リストとする。
3. ペア $[[A,\alpha],L]$ と $[[B,\beta],M]$ の間の演算 $*$ ($+$, $-$, $\times$ etc.) を次のように定義する。
   まず、区間部分を計算し、その結果の区間 $[C,\gamma]=[A,\alpha]*[B,\beta]$ にゼロ書換えを施す。
   もし、$[C,\gamma]$ が $[0,0]$ に書き換えられたら, その結果は **0** で、$\mathcal{L}$ はそのままとする。(**0** は "ゼロIS" で、$\mathbf{0}t=0$ for all $t \in R[x_1,\ldots,x_m]$.)
   そうでなければ、N を $N+1$ とし、新しい係数 $[[C,\gamma],N]$ を作って、$\mathcal{L}$ の最後に $(\dot{*},L,M)$ を付加する。
4. (復元) $[[A,\alpha],N]$ が出力のある係数であるとする。$[[A,\alpha],N]$ を次のようにISに変換する：
   N がシンボルであれば、何もしない。$[[A,\alpha],N]$ がそのISそのものである。
   N が整数であれば、N を $\mathcal{L}$ の N 番目の要素に置き換え、以下再帰的に入力シンボルに行き着くまでこれを行なう。最終的に得られたペア $[[A,\alpha],S]$ がそのISである。

3 有効性の検討

ISCZ 法では、ゼロ書換えの時点で、$|E| > \epsilon$ のときに確かに正確計算を回避することができる。しかし、そのときに評価しなかったシンボルはいずれ評価されるかもしれない。すなわち、実質的に正確計算を省略できない可能性がある。そこで今回は、評価されたシンボルの個数とシンボルの全参照回数を調べた。

前回と同様、ISCZ 法をグレブナ基底を計算する Buchberger アルゴリズムに適用した。例題は、[4] の例題の中から次の 6 個を選んだ。

1. $F = \{f_1, f_2, f_3\}$, ここに $f_1 = \frac{1}{7}x^2 - \frac{326548390854652}{272974017239}x + \frac{1263781236281}{712638126}y^2 + \frac{26872672361827}{7263188218281}z^2$, $f_2 = \frac{3}{8}xy + \frac{12367812638123}{763812368213132}yz - \frac{63812638126}{77263812831}y$, $f_3 = \frac{4}{9}x + \frac{327091270979304}{24122375460421}y + \frac{18467031595309203}{318405459032}z - \frac{356318063693141319}{6436561806418109}$.
2. $F = \{(\sqrt{2}+\sqrt{5})x^3y + \sqrt{3}xy + \sqrt{7}, (\sqrt{3}-\sqrt{2})x^2y^2 - \sqrt{7}xy + 1/\sqrt{11}\}$.
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. $F = \{(\sqrt{2}+\sqrt{3})x^{30} + \sqrt{5} - 1, \sqrt{7}xy + \sqrt{11} + \sqrt{13}\}$.
6. $F = \{\sqrt{2}ex^3y + \sqrt{3}xy + \sqrt{7}/e, \sqrt{3}/\pi \cdot x^2y^2 - \sqrt{7}xy + e/\sqrt{11}\}$.

すべての例 F について、我々はイデアル $\langle F \rangle$ の辞書式順序に関するグレブナ基底を計算した。計算機は、dual core AMD(R) Opteron(R) processor (2.85GHz), 8GB RAM, Linux(R) OS である。

我々は、次の 2 種類の方法を試みた。

1. Maple 12 で実装した ISCZ 法。初期精度は 1 で、精度の増加幅も 1。
2. Maple 12 で実装した ISCZ 法。初期精度は 10 で、精度は倍々で増加。

参考のために、Maple 12 で自前で Buchberger アルゴリズムを実装したもの (R_GB) と、Maple 12 の Groebner パッケージの組込関数 "Basis"(Maple GB) についても cpu 時間を計測した。

方法 1 と方法 2 の実験結果を、表 1 と表 2 に示す。

表 1,2 において、"cpu time" は、初期精度から成功精度に至るまでにかかった cpu 時間（単位は秒）の合計を表す。"SP" は成功精度を、"ZR" はゼロ書換えによって実際にゼロに書き換えられた区間の個数を表す。"# of skipped nonzero coefficients" は、成功に至った精度で、"$|E| > \epsilon$" であるゆえに正確計算がスキップされたゼロでない係数の個数を表す。"len. of symbol lists" はシンボルリストの長さを、"# of evaluated symbols" は評価されたシンボルの個数を表す。"# of references" は、シンボルの参照回数で、

表 1: 方法 1 の実験結果

Ex.	cpu time	SP	ZR	# of skipped nonzero coefficients	len. of symbol lists	# of evaluated symbols	# of references	R_GB	Maple GB
1	0.2	7	12	349	361	361	687 (64)	< 0.1	< 0.1
2	28.7	10	18	304	322	322	622 (75)	14.0	2.1
3	2.4	3	6	81	87	87	161 (17)	0.5	0.2
4	3640.2	12	43	1556	1599	1599	3184 (313)	491.3	40.4
5	351.0	1	59	1978	2037	2037	4007 (467)	1012.6	195.9
6	322.3	8	18	304	322	322	622 (75)	15.0	> 10000

表 2: 方法 2 の実験結果

Ex.	cpu time	SP	ZR	# of skipped nonzero coefficients	len. of symbol lists	# of evaluated symbols	# of references	R_GB	Maple GB
1	0.1	10	12	349	361	361	687 (64)	< 0.1	< 0.1
2	25.5	10	18	304	322	322	622 (75)	14.0	2.1
3	1.2	10	6	81	87	87	161 (17)	0.5	0.2
4	3007.9	20	43	1556	1599	1599	3184 (313)	491.3	40.4
5	132.1	10	59	1978	2037	2037	4007 (467)	1012.6	195.9
6	212.6	10	18	304	322	322	622 (75)	15.0	> 10000

カッコ内はシンボルごとに見たときの最大参照回数を表す。"< 0.1" は計算が 0.1 秒未満に終わって結果が出たことを、"$> n$" は n 秒待っても計算は終わらず、結果が得られなかったことを示す。

実験結果より、ISCZ 法は、有理係数のときよりも無理係数のときの方が有効であることがわかる。特に、超越数を含む場合はそれが顕著である。しかし、例 5 を除き、ISCZ 法は R_GB より遅かった。その理由は、最終のシンボルリストの長さが評価されたシンボルの個数と同じであることから、ゼロ書換えにおいては確かにシンボルの評価を回避できているが、ほとんどすべてのシンボルがいずれは評価されることになるからと考えられる。従って、すべての実行過程を通じて見るならば、Buchberger アルゴリズムに関する限り、ISCZ 法は正確計算を本質的に軽減しているとはいえない。

Buchberger アルゴリズムのように、再帰的な呼び出しが頻発するようなアルゴリズムでは、ISCZ 法はあまり有効でないかもしれない。一方、凸包アルゴリズムのように、平坦な構造をもったアルゴリズムには有効であると考えられる。

4 おわりに

ISCZ 法は、Buchberger アルゴリズムに対しては、ゼロ書換えのときに正確計算を回避できる回数が多いにもかかわらず、それほど有効でないのは、全体を通せば、ほとんどすべてのシンボルが評価されるからであることがわかった。今後は、本手法が有効であるアルゴリズムを特定していく必要がある。前述の通り、例えば凸包アルゴリズムのように、平坦な構造をもったアルゴリズムには有効であろう。実際、凸包アルゴリズムを若干の例で実験したところでは、評価されたシンボルの個数はシンボルリストの長さの 1 割程度にとどまった。さらに、この場合、シンボルリストを導入しないほうがかえって高速になることもわかり、凸包アルゴリズムは ISCZ 法の有効性を主張する上で重要な例になりそうである。詳細については別の機会に譲りたい。

謝辞

本研究は科研費 21500026 の助成を受けたものである。

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

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

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

[4] 白柳 潔, 関川 浩. 安定化理論に基づく log method について. 京都大学数理解析研究所講究録掲載予定

[5] K. Shirayanagi and H. Sekigawa. Reducing Exact Computations to Obtain Exact Results Based on Stabilization Techniques. In *Proc. International Workshop on Symbolic-Numeric Computation 2009 (SNC2009)*, pages 191–197, 2009.

[6] K. Shirayanagi and M. Sweedler. A theory of stabilizing algebraic algorithms. Technical Report 95-28, Mathematical Sciences Institute, Cornell University, 1995. 92 pages. (http://www.ss.u-tokai.ac.jp/~shirayan/msitr95-28.pdf)

[7] K. Shirayanagi and M. Sweedler. Automatic algorithm stabilization. In *ISSAC'96 poster session abstracts*, pages 75–78, 1996.

[8] K. Shirayanagi and M. Sweedler. Remarks on automatic algorithm stabilization. *J. Symbolic Computation*, 26(6):761–766, 1998.