近似グレブナ基底の精度保証計算

鈴木正幸*
岩手大学工学部

1 はじめに

本稿では，近似係数代数方程式の根を精度保証付きで求めることが目的である．そのために，計算精度を保証する数のシステムを作成し，その上で精度保証付でグレブナ基底を計算し，そのグレブナ基底から，固有値法 (例えば [1]) による求根を精度保証付きで行った．

白柳は [2] において，グレブナ基底計算の安定化を行い，[3] において一般的な多項式算法の安定化を提案している．本稿に非常に深く関連する白柳のグレブナ基底の安定化についてまとめておく．

- 区間数を係数とする多項式に対し，通常のブッフバーガ算法を用いて基底計算するが，
- 0 を含む区間数を 0 とみなす規則を追加する．

つまり，算法の構造を全く変えず，係数と 0 簡約規則だけを追加し，数値的に安定な計算ができ，入力の精度を上げていくことができれば，真の基底へ収束していくことが保証される．

近接根を持つ場合には，中間基底の頭項の精度が悪くなる．精度の悪い基底を用いて簡約や S 多項式を行った場合，最終的に求まる根の精度は著しく悪いものになってしまう．白柳の安定化手法を用いて，十分な精度で根を求めるためには，かなり高い精度を必要とする．

Stetter[5] は，重根や近接根を持つ方程式に対し，従来のグレブナ基底ではなく，双対基底より得られる根の重度に関する情報を用いて，安定に根の計算できる表現 (拡張グレブナ基底) を提案している．精度の悪い基底からでも，近似的な根の位置を計算する手法である．

根の重復度や近接度は，精度の低下として通常の基底計算にあらわれるはずである．Stetter の論文では精度に関する議論はないが，基底計算で精度を監視することで，数値的に安定に根の計算ができる近似基底が求められると考えられる．本稿では，こうした観点に基づき，数値的に悪条件な場合の実験を行い，基底，簡約，根などの精度の関係を調べ報告する．

2 精度を保証する数のシステム

近似係数を扱う場合，数のデータ型によらず，正確な表現なのかどうかが重要である．数を正確に表現された数 (正確数) と，正確には表現されていない数 (不正確数) に区別し，不正確数に対し，次のことを要請する:

- 計算誤差の混入を防ぐこと，
- 演算結果の精度保証を行なうこと，
- 計算精度の制御ができること，

*suzuki@cis.iwate-u.ac.jp

2.1 有効数

これらの要求に対しこれまで区間数が用いられてきたが，精度に関する操作が明確で容易になるように，(z, n, e, m) の表現を用い，これを本稿では**有効数** と呼ぶことにする．その意味は，

- (z, n, e, m) の値域 は，区間数 $((m-1) \times 2^e, (m+1) \times 2^e)$ と同じとする．
- z は桁落ち桁数，n は有効桁数 $(\log_2(m))$ とする．

区間数だけでは，桁落ちや有効桁の情報が扱いにくいため，有効数のシステムを作成している．

有効数のシステムで，有効精度がなくなってしまう (無効数) 場合に次の二通りがある．

- **無効ゼロ:** $(z, 0, e, 0) = (-2^e, 2^e)$
- **無効数:** $(z, 1, e, 1) = (0, 2 \times 2^e)$

白柳は，無効ゼロのみ 0 になると言う立場であるが，筆者は，無効数であっても計算の精度を上げると，z の値が大きくなり漸近的に 0 に近づく場合があるので，無効数も 0 を含むと見なす立場である．以後有効数を用いた計算は，

- 入力係数の桁落ちなしで計算を開始.
- 無効ゼロのみを 0 とみなす，または，
- 無効桁数を 0 とみなす.

こととなる．

2.2 計算履歴

計算途中に現れる数が，入力多項式の係数の四則演算で求まる場合に，入力係数からの演算履歴を持つことができる．履歴を持つ区間数とその応用については [4] がある．

本システムでは，履歴の表現として，積和標準形を採用した．標準系を計算するための時間は膨大になるが，

- 計算による桁落ちを最小にできること
- 精度に関与する係数の特定がしやすいこと

ことからこの表現を採用した．計算履歴を持つ有効数では，

- 記号計算で 0 と分かるものを 0 とできる，
- 精度の悪い項の消去による精度の回復ができる，
- 精度クリィカルな係数，計算が特定できる，
- 精度に関する構成ができる．

3 根の精度保証計算

3.1 計算方法

根の計算手順は,

1. 通常のブッフバーガ算法によるグレブナ基底を計算し,
2. 固有値法
   2.1 グレブナ基底を用いて，normal set, 剰余環での乗算行列 を求める.
   2.2 この行列の固有値を，固有多項式を DKA 法で解いて求める.

この算法を有効数上にインプリメントし，次の例の計算を精度を変えて行った．これは，x に近接根を持つ楕円 P_1, P_2 の交点を求めるものである.

$$\begin{cases} P_1 = & 1.027748y^2 - 0.467871xy + 2.972252x^2 \\ & +0.662026y + 0.0785252x - 3.888889 \\ \\ P_2 = & 3.958378y^2 + .701807xy + 1.041622x^2 \\ & +.0785252y + .662026x - 3.888889 \end{cases}$$

このグレブナ基底を辞書式順序 $(y > x)$ で求める場合に，悪条件となる．この例題を Maple で計算してみると、計算が終了しなかったり、定数が出力される状態になり正しく計算が行えない.

3.2 十分な精度での計算 (出発精度 64 ビット)

まず，十分な精度で求めた基底を示す．入力係数の精度を 64 ビットで与えた場合は，かなりの桁落ちがあるものの，基底は次のように求まる.

$$\begin{aligned} g_1 = & \ [[33, e-41] \ -2.2756529e-3] \ y \\ & +[[53, e-40] \ -4878.7517] \ x^3 + [[52, e-39] \ -4464.4549] \ x^2 \\ & +[[52, e-39] \ 5594.3765] \ x + [[52, e-39] \ 4690.2769] \end{aligned}$$

$$\begin{aligned} g_2 = & \ [[27, e-22] \ 28.569566] \ x^4 \\ & +[[23, e-21] \ -3.8467693] \ x^3 + [[26, e-20] \ -60.203667] \ x^2 \\ & +[[23, e-20] \ 6.9233999] \ x + [[26, e-21] \ 28.831612] \end{aligned}$$

ここで，例えば $[[33, e-41] \ -2.2756529e-3]$ が有効数表現で，有効桁が 33 ビット， $\pm 2^{-41}$ の誤差を持ち，中心の浮動少数表現が $-2.2756529e-3$ となる有効数を表している.

上の基底 g_1 と g_2 から固有値法で得られる x の根 は,

$$[[21, e-20] \ 1.0497255], \ [[21, e-20] \ 1.0497264], \ [[40, e-39] \ -1.2044155], \ [[40, e-40] \ -.76039096]$$

となる.

3.3 精度クリティカルな場合 (出発精度 32ビット)

同じ例題を，入力係数の精度を低くして行うと，中間基底の頭項の数係数の精度がほとんどなくなる場合が出現する．その場合の計算結果を以下に示す.

$$\begin{aligned} g_1 = \quad & [[1, e-9] \ -.00195312]\ y \\ & +[[21, e-8] \ -4878.7460]\ x^3 + [[20, e-7] \ -4464.4531]\ x^2 \\ & +[[20, e-7]\ 5594.3671]\ x + [[20, e-7]\ 4690.2734] \end{aligned}$$

根を求める立場から，この基底の取り扱いに次の 2 通りが考えられる：

1. y の係数は 0 ではないことは確なので, このまま基底計算を行う.

   しかしこのあとの基底の精度は著しく低下し，根に関する情報はほとんど得られず，精度を上げていくことでしか根を求めることはできない.

2. y の係数を 0 と見る．(正確には，計算精度を，y の係数を 0 と見なせる値に取り直して，計算を続ける．)

上記 2 の場合，精度の悪い頭項を無視すると次の基底が得られる：

$$\begin{aligned} g_1 = \quad & [[34, e-32]\ 3.9583779]\ y^2 \\ & +[[29, e-32]\ .65818061]\ y + [[33, e-32]\ 3.9583779]\ x^2 \\ & +[[32, e-32]\ .56123591]\ x + [[34, e-32] \ -6.9971346] \\ g_2 = \quad & [[30, e-28] \ -2.5732910]\ yx \\ & +[[29, e-27]\ 2.7012532]\ y + [[29, e-25]\ 10.694771]\ x^2 \\ & +[[27, e-28] \ -.36956347]\ x + [[29, e-25] \ -11.396894] \\ g_3 = \quad & [[21, e-8] \ -4878.7460]\ x^3 + [[20, e-7] \ -4464.4531]\ x^2 \\ & +[[20, e-7]\ 5594.3671]\ x + [[20, e-7]\ 4690.2734] \end{aligned}$$

この場合の x の根は,

$$1.0496427,\ 1.0497339,\ -1.2044155,\ -.76039096$$

と求まり，近似的な根になっていることがわかる.

3.4 履歴を持つ有効数で求めた基底：(出発精度 32ビット)

g_1 の頭項係数の精度の悪さから g_2 が無精度になってしまった 3.3 の計算は，履歴付で行うことにより，g_2 をある程度の精度で求めることができる.

$$\begin{aligned} g_1 = \quad & [[3, e-11] \ -.00195]\ y \\ & +[[24, e-11] \ -4878.75098]\ x^3 + [[24, e-11] \ -4464.45459]\ x^2 \\ & +[[22, e-9]\ 5594.375]\ x + [[24, e-11]\ 4690.27686] \end{aligned}$$

$$\begin{aligned} g_2 = \quad & [[21, e-7] \ -12554.438]\ x^4 \\ & +[[20, e-6] \ -11488.328]\ x^3 + [[19, e-5]\ 14395.90625]\ x^2 \\ & +[[21, e-7]\ 12069.445]\ x + [[19, e-5]\ 12210.] \end{aligned}$$

ここで，g_1 の頭項係数で大きな桁落ちが起きているが，その内訳は以下のような 20 個の係数積の和になっている．(見易さのために g.c.d. をとり，g_1 の頭項係数値とは異なっているが相対的な桁落ちの程度を見て欲しい)

MSB	有効桁	打切次	値	係数積
7	28	-21	-80.7955	$-a_{02}a_{10}^2b_{20}^3$
6	29	-23	52.7994	$-a_{11}^2a_{00}b_{20}^3$
6	27	-21	41.12638	$2a_{20}a_{11}a_{00}b_{20}^2b_{11}$
4	27	-23	-13.70878	$a_{20}a_{11}^2b_{20}^2b_{00}$
4	27	-23	-10.6779	$-2a_{20}^2a_{11}b_{20}b_{11}b_{00}$
4	27	-23	8.00850	$-a_{20}^2a_{00}b_{20}b_{11}^2$
3	26	-23	7.35159	$a_{20}a_{10}^2b_{20}^2b_{02}$
3	23	-20	-4.97646	$2a_{20}a_{02}a_{10}b_{20}^2b_{10}$
⋮				
-1	22	-23	.452808	$-2a_{20}^2a_{10}b_{20}b_{02}b_{10}$
-3	23	-26	.1016952	$a_{20}^2a_{11}b_{20}b_{10}b_{01}$
-3	23	-26	-.0766291	$-a_{20}^2a_{02}b_{20}b_{10}^2$
-4	23	-27	-.0464585	$-a_{20}a_{11}a_{01}b_{20}^2b_{10}$
-4	24	-28	.0396060	$-a_{20}^3b_{11}b_{10}b_{01}$
-5	23	-28	-.0180936	$a_{20}^2a_{01}b_{20}b_{11}b_{10}$
-7	23	-30	.0069724	$a_{20}^3b_{02}b_{10}^2$
-12	4	-16	-.0001373	総和

算法はまだ実現できていないが，この表からどの入力係数のどの有効桁までが桁落ちに関与しているかを判定することができると考えられる．

4 まとめ

近似係数で与えられた代数方程式の根を，精度保証付きで求めるためのシステム，算法の実装を行った．このシステムを用い，近接根を持つ方程式で実験を行った．

近似的な根を求めるという立場であれば, 真の基底に，台収束，係数収束していない近似基底であっても，根の情報を含む表現があることを示した．これは Stetter の表現と同じである．

与えられた問題から，精度クリティカルな計算を見つけることが重要である．本システムでは，これを結果の有効桁を失うこと，あるいは，指定された有効桁以下になった場合に検出する．

有効桁がなくなる場合には，

- 結果が 0 である，
- 入力係数の精度が足りず 0 か否か判定できない，

場合が存在する．この場合，より精度の良い計算をするために，

- 入力係数の精度を上げる，
- 結果を 0 と見なせるように，入力係数の精度を下げる，

ことが考えられる．

入力係数が正確にわかっている場合はよいが，実際にはわかっていないことが多く，その場合には計算が精度クリティカルか，その場合にはその桁落ちの程度，0と見なせる精度，その計算に関与する係数の特定が必要となる．履歴を持つ数は精度クリティカルな係数の特定のためにも使えると考えている．係数後との計算結果への寄与がわかり，打ち切った精度から問題へのフィードバック，精度による構成などが可能となるだろう．ただし，時間計算量，空間計算量ともに非常に大きく，使い方には工夫が必要である．

参 考 文 献

[1] Corless, R. M.: Gröbner Bases and Matrix Eigenproblems. ACM SIGSAM Bull. 30, 4 (1996), 26–32

[2] Shirayanagi, K.: An Algorithm to Compute Floating Point Gröbner Bases, *Mathematical Computation with Maple V: Ideas and Applications*, Birkhäuser, 1993, 96–106

[3] Shirayanagi, K., Sweedler, M.: A Theory of Stabilizing Algebraic Algorithms, *Tech.Rept.95–28*, Cornell Univ., 1–92, 1995

[4] Shirayanagi, K., Sweedler, M.: Automatic Algorithm Approximation

[5] Stetter, H. J.: Stabilization of Polynomial Systems Solving with Groebner Bases, P̂roc. ISSAC'97, 1997, 117–124