

大域的収束性を持つ代数方程式の新しい解法

山梨大学工学部 鈴木 智博(Tomohiro SUZUKI) *

山梨大学教育人間科学部 鈴木 俊夫(Toshio SUZUKI) †

1 はじめに

解析関数の対数微分の周回積分が積分領域内の零点の個数を与えることは良く知られている。積分経路を円周とし、その等分点を分点とした数値積分でこの周回積分を近似したときに得られる理論的な評価から我々は新しい反復法を提案し、数値積分誤差法 (Numerical Integration Error Method)[5, 8] と呼んでいる。これまでに他の代数方程式の計算方法との比較 [7] や、重根、近接根に対するアプローチ [6] を報告した。

今回は大域的収束性を持つアルゴリズムについて報告する。数値実験は「平野の改良 Newton 法」との比較を中心に行なう。

2 準備

次の複素係数代数方程式の根を求める問題を考える。

$$f(z) = z^n + a_1 z^{n-1} + \cdots + a_n \quad (a_n \neq 0) \quad (1)$$

$$= \prod_{k=1}^s (z - z_k)^{n_k} \quad (2)$$

ここで $f: C \rightarrow C$, $z, a_k \in C$, $n_k \in Z$ は根 $z_k \in C$ の重複度である。

Γ を正の向きを持つ閉曲線とする。 Γ 内に存在する (1) の根の数を $N \in Z$ としたとき、

$$N = \frac{1}{2\pi i} \oint_{\Gamma} \frac{f'(z)}{f(z)} dz \quad (3)$$

である。いま、 Γ を中心 $\lambda \in C$ 、半径 $\tau \in R$ の円周とする。このとき円周を m 等分した分点は $x_j = \lambda + \tau e^{i\theta j}$ ($j = 0, \dots, m-1, \theta = 2\pi/m$) である。この分点において (3) の

*tomo@cs.yamanashi.ac.jp

†suzuki@edu.yamanashi.ac.jp

Riemann 和を取ると

$$N \simeq \frac{1}{2\pi i} \sum_{j=0}^{m-1} \frac{f'(x_j)}{f(x_j)} (x_{j+1} - x_j) = \frac{\tau}{m} \sum_{j=0}^{m-1} \frac{f'(\lambda + \tau e^{i\theta j})}{f(\lambda + \tau e^{i\theta j})} e^{i\theta j} \quad (4)$$

となる。これを $T(\lambda, \tau, m)$ と定義する。

$$T(\lambda, \tau, m) = \frac{\tau}{m} \sum_{j=0}^{m-1} \frac{f'(\lambda + \tau e^{i\theta j})}{f(\lambda + \tau e^{i\theta j})} e^{i\theta j} \quad (5)$$

(5) を用いる反復法を総称して数値積分誤差法 (NIEM) と呼ぶ。

3 NIEM の反復の修正量

$\nu_k = z_k - \lambda$ ($k = 1, \dots, s$) $\in C$ とする。このとき以下の定理が成り立つ [5]。

定理 1 すべての k ($= 1, \dots, s$) に対して $\tau^m \neq \nu_k^m$ ならば T を次のように表わすことができる。

$$T(\lambda, \tau, m) = \sum_{k=1}^s \frac{n_k}{1 - (\nu_k/\tau)^m} \quad (6)$$

以降、すべての根は λ からの距離によって番号付けされているとする。つまり、

$$|\nu_1| \leq |\nu_2| \leq \dots \leq |\nu_s| \quad (7)$$

である。

ここで、

$$\Delta = \sum_{k=2}^s \frac{n_k}{n_1} \left(\frac{\nu_1}{\nu_k} \right)^m \frac{1 - (\tau/\nu_1)^m}{1 - (\tau/\nu_k)^m} \quad (8)$$

とすると、

$$T = \frac{n_1}{1 - (\nu_1/\tau)^m} (1 + \Delta)$$

と表わせる。 $|\Delta|$ が小さければ上式より $\lambda + \tau \{(T - n_1)/T\}^{\frac{1}{m}}$ のうちのひとつ (m 乗根は m 個の分枝を持つ) が z_1 のよい近似となることが期待できる。NIEM では

$$\tau \left(\frac{T - n_1}{T} \right)^{\frac{1}{m}} \quad (9)$$

を反復の修正量として用いる。実際にこの近似の定量的な評価として次の定理が成り立つ [5]。

定理 2 $\tau < |\nu_1| < |\nu_2|$ ならば、 $|\nu_1/\nu_2|^m \frac{2n}{1-|\tau/\nu_2|^m} < 1/2$ を満す m に対して以下が成り立つ。

$$\min \left| \lambda + \tau \left(\frac{T - n_1}{T} \right)^{\frac{1}{m}} - z_1 \right| \leq \frac{16n}{m} \frac{|\nu_1|}{1 - |\tau/\nu_1|^m} \left| \frac{\nu_1}{\nu_2} \right|^m \quad (10)$$

この定理から m を十分大きく取れば任意精度で z_1 の近似解を得ることができることが分かる (ただし、計算機の有効桁数などを考慮しない場合)。

また反復を ((9) などにより) 繰り返し、ある段階で数値解が根に十分近くなれば m が小さな値でも z_1 のよい近似解が得られることが分かる。実際にすでに提案した我々のアルゴリズム [5] では、反復の最終段階においては $m = 1$ としている。このとき NIEM は本質的に Newton 法と等価である ((5)、(9) を見よ)。

以下では次の仮定の下で議論する。

$$\tau < |\nu_1| \quad (11)$$

4 大域収束アルゴリズム

本文では以下の定理に基づく大域的収束性を持つ代数方程式の解法のアルゴリズムを提案する。

定理 3 $\alpha, q_1, q_2 \in \mathbb{R}$ とする。 $0 < \alpha < 1, 0 < q_1 \leq 1 < q_2$ に対して、 $q_1 < |(T - n_1)/T| < q_2$ という条件で、

$$\min \left| \lambda + \tau \left(\frac{T - n_1}{T} \right)^{\frac{1}{m}} - z_1 \right| < \alpha |z_1 - \lambda| \quad (12)$$

であるような m が存在する。

反復が一回終了した段階で常に (12) が成り立ち、反復の各ステップが有限回の計算で終了するならば、アルゴリズムは大域的収束性を持つことが保証される。

ここで、

$$\begin{aligned} R(n, \lambda) &= \min(n|f(\lambda)/f'(\lambda)|, |f(\lambda)|^{\frac{1}{n}}) \\ C(T, q_j) &= \begin{cases} \left(\Re(T) - \frac{1}{1-q_j^2} \right)^2 + \Im(T)^2 - \left| \frac{q_j}{1-q_j^2} \right|^2 & q_j \neq 1 \\ 0.5 - \Re(T) & q_j = 1 \end{cases} \end{aligned}$$

、 $U(\lambda, \tau)$ を中心 λ 、半径 τ の円形領域とする。

アルゴリズム 4 (Globally Convergent NIEM の第 k 段 (基本形))

(step0) $r = R(n, \lambda^{(k)})$ 、 $T = T(\lambda^{(k)}, r, 3)$ を計算し、 $p = [|T|] + 1$ 、 $\tau_{\max} = r$ 、 $\tau_{\min} = 0$ 、 $\tau = r/n$ 、 $m = 5$ とする。

(step1) $U(\lambda^{(k)}, \tau)$ に根が存在するならば、 $\tau_{\max} = \tau$ 、 $\tau = (\tau_{\max} + \tau_{\min})/2$ とし、これを $U(\lambda^{(k)}, \tau)$ に根が存在しなくなるまで繰り返す。

(step2) $T = T(\lambda^{(k)}, \tau, m)$ を計算し、 $C(T, q_1) > 0$ ならば $\tau_{\max} = \tau$ 、 $\tau = (\tau_{\max} + 5\tau_{\min})/6$ とし、 $C(T, q_2) > 0$ ならば $\tau_{\min} = \tau$ 、 $\tau = (\tau_{\max} + \tau_{\min})/2$ として T の計算を繰り返す。

(step3) $((T - n_1)/T)^{\frac{1}{m}}$ の pm 個の分枝を求め $x_j^{n_1}$ ($j = 1, \dots, m, n_1 = 1, \dots, p$) とする。

(step4) $x_j^{n_1}$ の中から $|f(\lambda^{(k)} + \tau x_j^{n_1})|$ が最小となるものを x とし、 $|f(\lambda^{(k)} + \tau x)| < \alpha|f(\lambda^{(k)})|$ ならば $\lambda^{(k+1)} = \lambda^{(k)} + \tau x$ とし第 k 段を終了し、そうでなければ $\tau = \sqrt{\tau}$ 、 $m = 2m$ として (step1) に戻る。

注意 5 (step1) の条件は *Schur-Cohn* のアルゴリズム [3] で確認できる。 $U(\lambda, R(n, \lambda))$ は少なくとも一つの根をその内部に含む [1]。

次に、このアルゴリズムの改良を考える。以下の命題が成り立つ。

命題 6 $1 < q'_2 < q_2$ かつ、 $n_1 > 1$ に対して $n_1(q'_2 + 1) > q_2 + 1$ ならば、

$$U\left(\frac{1}{1-q_2'^2}, \left|\frac{q'_2}{1-q_2'^2}\right|\right) \supset U\left(\frac{n_1}{1-q_2^2}, \left|\frac{n_1 q_2}{1-q_2^2}\right|\right)$$

である。

上の命題より、実際の計算においては $n_1 > 1$ に対しても $n_1 = 1$ (= onstant) として考えて差し支えないことが分かる。しかし、初期値が根から離れている場合には反復の初期の段階で n_1 が大きな値の方が修正量が大きくなるため有効である。これは、根から離れた位置からは複数の根が重根のように見えるからである。

また、重根に対しては $n_1 = 1$ では十分な精度が得られない。 n_1 を重根の重複度と等しく選らばいわゆる「1/重複度」の精度の低下は起こらない。

$(T - n_1)/T = (\nu_1/\tau)^m Y^m$ とおく。このとき、

$$\min \left| \lambda + \tau \left(\frac{T - n_1}{T} \right)^{\frac{1}{m}} - z_1 \right| = |\nu_1| \min_{l=0}^m \left| 1 - Y e^{il \frac{2\pi}{m}} \right| \quad (13)$$

と表わせる。 $q = |(T - n_1)/T|$ とすると、上式の右辺の最小値は、

$$s(m, q) = \sqrt{(q^{\frac{1}{m}} - 1)^2 + q^{\frac{1}{m}} (\pi/m)^2} \quad (14)$$

で抑えられる。したがって、 $s < 1$ のとき (step4) の $|f(\lambda^{(k)} + \tau x)| < \alpha|f(\lambda^{(k)})|$ が成立しなくても $|\tau x - \nu_1| < |\nu_1|$ であり、 $\lambda^{(k)} + \tau x$ は z_1 に近づいている。

以上を考慮し修正したアルゴリズムは以下となる。

アルゴリズム 7 (Globally Convergent NIEM の第 k 段 (実用形))

(step0) $r = R(n, \lambda^{(k)})$ 、 $\tau_{\max} = r$ 、 $\tau_{\min} = 0$ 、 $\tau = r/n$ 、 $m = 5$ とする。

(step1) $T = T(\lambda^{(k)}, \tau, m)$ を計算し、 $C(T, q_1) > 0$ ならば $\tau_{\max} = \tau$ 、 $\tau = (\tau_{\max} + 5\tau_{\min})/6$ とし、 $C(T, q_2) > 0$ ならば $\tau_{\min} = \tau$ 、 $\tau = (\tau_{\max} + \tau_{\min})/2$ として T の計算を繰り返す。

(step2) $((T-1)/T)^{\frac{1}{m}}$ の m 個の分枝を求め x_j ($j = 1, \dots, m$) とする。

(step3) x_j の中から $|f(\lambda^{(k)} + \tau x_j)|$ が最小となるものを x とし、 $|f(\lambda^{(k)} + \tau x)| < \alpha |f(\lambda^{(k)})|$ または $s(m, q) < 1$ ならば $\lambda^{(k+1)} = \lambda^{(k)} + \tau x$ とし第 k 段を終了し、そうでなければ $\tau = \sqrt{\tau}$ 、 $m = 2m$ として (step1) に戻る。

5 平野法

非線形方程式の解法である Newton 法は大域的収束性を持たないという欠点がある。平野の改良 Newton 法 (平野法)[2] は代数方程式に対して局所的収束速度を損なうことなく大域的収束性が保証された方法である。平野法については [4] に詳しい解説がある。

平野法のアルゴリズムを説明する。以下では、 c_l ($l = 0, \dots, n$) を第 k 段での近似解 $z^{(k)}$ の周りでの f の Taylor 展開の係数とし、減速のためのパラメータを $0 < \beta < 1$ 、 $\lambda > 1$ とする。

アルゴリズム 8 (平野法 の第 k 段 (基本形))

(step1) c_l ($l = 0, \dots, n$) を組み立て除法で計算する。 $\mu = 1$ とする。

(step2) $\zeta_l = (-\mu c_n / c_{n-l})^{\frac{1}{l}}$ ($l = 1, \dots, n$) とする。

(step3) $|\zeta_l|$ が最小である l ($1 \leq l \leq n$) を m とする。

(step4) $|f(z^{(k)} + \zeta_m)| \leq (1 - (1 - \beta)\mu)|c_n|$ ならば $z^{(k+1)} = z^{(k)} + \zeta_m$ として第 k 段を終了し、そうでなければ $\mu = \mu/\lambda$ として (step2) に戻る。

注意 9 (step2) の l 乗根の分枝は $|z^{(k)} + \zeta_l|$ が最も小さくなるようなものを得らば。

平野法は ζ_m を決定するために多くの計算量が必要であるが、通常 m は小さな値であることがほとんどであるため次のような実用的変種が考えられている [4]。

アルゴリズム 10 (平野法 の第 k 段 (実用的変種))

(step0) c_n 、 c_{n-1} を組み立て除法で計算する。 $\zeta_1 = -c_n / c_{n-1}$ とする。

$|f(z^{(k)} + \zeta_1)| \leq \beta |c_n|$ ならば $z^{(k+1)} = z^{(k)} + \zeta_1$ として第 k 段を終了し、そうでなければ $\mu = 1/\lambda$ 、 $K = 2$ として (step1) へ進む。

(step1) $K \leq n$ ならば組み立て除法を続けて c_{n-K} を計算し $\zeta_K = (-\mu c_n / c_{n-K})^{\frac{1}{K}}$ とする。

(step2) $\zeta_l = \zeta_l / \lambda^{\frac{1}{l}}$ ($l = 1, \dots, K-1$) とする。 $K > n$ ならば $K = n$ とする。

(step3) $|\zeta_l|$ が最小である l ($1 \leq l \leq K$) を m とする。

(step4) $|f(z^{(k)} + \zeta_m)| \leq (1 - (1 - \beta)\mu)|c_n|$ ならば $z^{(k+1)} = z^{(k)} + \zeta_m$ として第 k 段を終了し、そうでなければ $\mu = \mu/\lambda$ 、 $K = K + 1$ として (step1) に戻る。

平野法の実用的変種ではほとんどの場合 (step0) で第 k 段が終了する。これは Newton 法そのものである。

上のアルゴリズムで l 乗根の分枝の選び方や ζ_l の選び方などは $z^{(k)}$ に近い数値解を得るという方針に基づいているが、これによって初期値が根から離れている場合には反復回数が極端に増加するという性質がある (数値実験を見よ)。

6 数値実験

幾つかの例に対して行なった数値実験の結果を以下に示す。実験に用いた計算機の CPU は Intel Pentium III 800MHz である。実験 1、2 は倍精度演算、実験 3 は仮数部の桁数が 400 桁の多倍長演算で実行した。

また、NIEM のパラメータは倍精度では $q_1 = 0.8$ 、 $q_2 = 1000$ 、 $\alpha = 0.9$ 、多倍長では $q_1 = 1$ 、 $q_2 = 10000$ 、 $\alpha = 0.9$ とし、平野法のパラメータは $\beta = 3/4$ 、 $\lambda = 2$ としている。

6.1 実験 1 (Newton 法では収束しない例)

$f(z) = z^3 - 3z + 3$ に対して初期値 $\lambda^{(0)} = 2.5$ とした場合 [4] の結果を表 1、表 2 に示す。

表 8: GCNIEM

k	$\lambda^{(k)}$	$ f(\lambda^{(k)}) $	m	τ
0	2.5	1.1125e+01		
1	$1.0715 + 1.0379i$	2.5334	5	7.0635e-01
2	$1.0520 + 5.6544e-01i$	8.9623e-04	5	3.5034e-01
3	$1.0519 + 5.6524e-01i$	4.0885e-16	5	2.4718e-04

減速 Newton 法でこの例を実行すると $z = 1$ に収束してしまい数値解が得られないが、NIEM、平野法ともに数値解が得られている。

表 9: 平野法 (実用的変種)

k	$z^{(k)}$	$ f(z^{(k)}) $	m	μ
0	2.5	1.1125e+01		
1	1.7937	3.3896	1	1
2	1.2841	1.2650	1	1
3	9.5911e-01	1.0049	1	0.5
4	9.5911e-01 – 4.1789e-01	5.3155e-01	2	0.5
5	1.0849 – 5.8686e-01	1.4746e-01	1	1
6	1.0534 – 5.6517e-01	5.3645e-03	1	1
7	1.0519 – 5.6523e-01	7.8423e-06	1	1
8	1.0519 – 5.6524e-01	1.6773e-11	1	1
9	1.0519 – 5.6524e-01	4.8115e-16	1	1

NIEM では一回の反復で f 、 f' の計算を m 回実行する必要がある。上の例で、関数値およびその微分値の計算量のみ考慮すると平野法は NIEM の約 2/3 の計算量で数値解が得られている。

6.2 実験 2 (平野法 (実用的変種) の収束が遅い例)

$f(z) = z^{20} - 1$ に対して初期値を 0 とした場合の結果を表 3、表 4 に示す。

表 10: GCNIEM

k	$\lambda^{(k)}$	$ f(\lambda^{(k)}) $	m	τ
0	0	1.0000		
1	7.4506e-01 – 5.4132e-01	8.0741e-01	10	8.7321e-01
2	8.0900e-01 – 5.8777e-01	3.5068e-05	5	4.9468e-02
3	8.0902e-01 – 5.8779e-01	6.8074e-16	5	1.7540e-05

この例で平野法は、 f の導関数の特殊性から原点近傍では大きな修正量が得られず反復回数が増加する。最終的には数値解が一度単位円の外に出た後、根に 2 次収束する。

また、収束が遅いもう一つの原因は $(-\mu c_n / c_{n-i})^{\frac{1}{i}} (i < j)$ の修正量を考慮しないことが挙げられる。平野法 (基本形) では 1 回の反復で、 $m = 20$ 、 $\mu = 1$ により根 1 が得られる。

NIEM では第一段で一度 $m = 2m$ となるが、3 回の反復で数値解が得られる。

表 11: 平野法 (実用的変種)

k	$z^{(k)}$	$ f(z^{(k)}) $	m	μ
0	0	1.0000		
1	$-5.1763\text{e-}01 + 6.3390\text{e-}17$	1.0000	20	$1.9073\text{e-}06$
2	$-5.6940\text{e-}01 - 5.7051\text{e-}17$	$9.9999\text{e-}01$	1	$3.8147\text{e-}06$
3	$-6.0325\text{e-}01 + 7.3993\text{e-}18$	$9.9996\text{e-}01$	1	$1.5259\text{e-}05$
4	$-6.4844\text{e-}01 - 3.1336\text{e-}18$	$9.9983\text{e-}01$	1	$6.1035\text{e-}05$
5	$-6.9425\text{e-}01 + 1.0733\text{e-}18$	$9.9932\text{e-}01$	1	$2.4414\text{e-}04$

6.3 実験 3 (多倍長演算による高次方程式の例)

根を $\pm 2 \pm 2i$ の範囲の乱数とした高次方程式の場合の結果を以下に示す。

方程式の次数は 100 次 から 1000 次とし、反復回数 (k) と計算時間 (time [sec]) を示した。表中の T.O. は 3000 [sec] で数値解が得られなかったことを示す。また、初期値は 0、 $0.1 + 0.1i$ 、 $1 + i$ 、 $3 + 3i$ の 4 種類とし、終了条件は $|f| < 10^{-20}$ とした。

表 12: GCNIEM

n	100		200		500		700		1000	
	k	time	k	time	k	time	k	time	k	time
0	3	6.0	3	9.2	3	28.0	4	45.8	4	70.6
$0.1+0.1i$	3	7.5	3	10.9	4	35.5	4	41.5	4	57.5
$1+i$	4	7.7	5	20.5	3	21.4	3	40.0	6	119.8
$3+3i$	4	7.6	5	19.3	8	77.3	6	81.2	9	156.5

結果より、平野法は初期値が原点の場合には少ない回数で数値解が得られることが分かる。逆に、初期値が原点から離れるに従って反復回数が増加する。これは先に述べたような原因による。NIEM の初期値による反復回数への影響は平野法ほど大きくない。

7 まとめ

NIEM の修正量を用いて近似解列が確実に根に近づくための条件を定理の形で明確にした。また、この条件を満たしながら反復を続けることにより大域的収束性を持つアルゴリズムを提案した。提案したアルゴリズムは平野法 (基本形) と比較した場合には少ない計算

表 13: 平野法 (実用的変種)

n	100		200		500		700		1000	
	k	time	k	time	k	time	k	time	k	time
0	7	1.0	10	2.7	9	6.6	9	9.1	10	14.5
$0.1+0.1i$	9	1.3	10	2.9	11	9.0	14	16.4	11	15.4
$1+i$	24	3.5	39	12.1	33	26.4	39	37.6	48	66.0
$3+3i$	103	14.8	107	29.9	441	T.O.	316	T.O.	222	T.O.

量で数値解を得ることができる。しかし、高階の微分値による修正量が必要でない場合には平野法 (実用的変種) は Newton 法と同じであるので、これと比較すると圧倒的に計算量は多い。

提案したアルゴリズムを平野法と比較するための数値実験を行なった。これより、提案したアルゴリズムは平野法と同様に大域的収束性を持つこと、初期値に対する反復回数の依存性が少ないことが分かる。

パラメータの決定方法に対する理論的、定量的な指針を設定することや、高次の方程式に適用した場合の並列計算の可能性、一般の解析関数への適用可能性などを検討することが今後の課題である。

参 考 文 献

- [1] Henrici, P., *Applied and computational complex analysis Vol. 1*, Wiley Interscience, New York (1974).
- [2] 平野菅保, 代数方程式の解法および誤差, 第 8 回プログラミングシンポジウム報告集, 情報処理学会, (1967).
- [3] Stewart III, G. W., *On Lehmer's Method Finding the Zeros of a Polynomial*, Math. Comp. 23, (1969), pp. 829-835.
- [4] 杉原正顕, 室田一雄, 数値計算法の数理, 岩波書店, 東京 (1994).
- [5] 鈴木智博, 鈴木俊夫, 武藤秀夫, 数値積分誤差を用いた新しい多項式の零点の解法, 日本応用数学会論文誌, Vol. 9, No. 2, (1999), pp. 65-76.
- [6] 鈴木智博, 鈴木俊夫, 数値積分誤差法による重根、近接根の計算精度, 日本応用数学会論文誌, Vol. 11, No. 1, (2001), (掲載予定).

- [7] 鈴木智博, 鈴木俊夫, 数値積分誤差法を用いた減次操作による多項式の全根計算, 第 28 回数値解析シンポジウム講演予稿集, (1999), pp. 61 – 64.
- [8] Suzuki, T. , Suzuki, T. , Numerical Integration Error Method - A New Method for Polynomial Root-finding -, The 3rd World Congress of Nonlinear Analysts (WCNA2000) Proceedings, (印刷中).