福井大学
工学部 研究報告
第44巻 第1号
1996年3月

非線形方程式の単根の存在区間アルゴリズム

中道 洋平* 長谷川 武光** 佐藤 義雄**

An Algorithm for Enclosing a Simple Zero of Nonlinear Equations

Youhei NAKAMICHI, Takemitsu HASEGAWA, and Yoshio SATOU

(Received Feb. 29, 1996)

We present a modification of an efficient algorithm due to Alefeld et al. for enclosing a simple zero of a nonlinear equation $f(x) = 0$ in the interval $[\ a\ ,\ b\ ]$. Alefeld et al. combine both the bisection method and Newton's method to enclose a simple zero in a stable manner. We modify the methods due to Alefeld et al. to improve their algorithm. Numerical experiments show that our method compares well with the methods of Alefeld et al..

1 はじめに

非線形方程式 $f(x) = 0$ の一根を求める方法として、二分法やニュートン法などがある。二分法は根を取り囲んだ区間に対して確実に収束するが、収束が遅い。ニュートン法は収束が速いが、初期値によっては収束しない場合があるので確実とはいえない。そこで、この2つの方法の中間の性質をもつ方法が Alefeld [3] [2] [1] によって発表された。

Alefeld の方法は連続関数 $f(x)$ に対し、$f(x) = 0$ の1つの単根 x_* を取り囲む初期区間 $[\ a\ ,\ b\ ]$, $(f(a)f(b) < 0)$ を与える。この初期区間を $[\ a_1\ ,\ b_1\ ]$ とかき、次のような単根 x_* を取り囲み収束する区間数列 $\{[\ a_n\ ,\ b_n\ ]\}_{n=1}^{\infty}$ を求める。

$$x_* \in [\ a_{n+1}\ ,\ b_{n+1}\ ] \subseteq [\ a_n\ ,\ b_n\ ] \subseteq \cdots \subseteq [\ a_1\ ,\ b_1\ ] = [\ a\ ,\ b\ ] \tag{1.1}$$

$$\lim_{n\to\infty}(b_n - a_n) = 0 \tag{1.2}$$

区間 $[\ a_n\ ,\ b_n\ ]$ に対し、[3] では点 a_n, b_n とその区間の外の1点を使って2次の補間式を作り、その根から区間 $[\ a_n\ ,\ b_n\ ]$ 上の点を決める。[2] ではその2次の補間式でニュートン法を行い、2次方程式の近似解から区間 $[\ a_n\ ,\ b_n\ ]$ 上の点を決める。[1] では点 a_n, b_n とその区間の外の2点を使って3次の補間式を作り、その根から区間 $[\ a_n\ ,\ b_n\ ]$ 上の点を決める。求めた区間 $[\ a_n\ ,\ b_n\ ]$ 上の点 c_n より $f(a_n)f(c_n) < 0$ が成立すれば、次の区間は $[\ a_n\ ,\ c_n\ ]$ となり、成立しなければ $[\ c_n\ ,\ b_n\ ]$ となる。これらの方法では、根を区間で取り囲むことと補間式を使うことで速く確実に収束する。

本研究では、これらの Alefeld の方法を実際に計算機上で作成し、更に効率的に単根 x_* を取り囲むよう区間上の点の取り方を変更した。実際、Alefeld の方法とそれを改良した本方法を [2] と同じ 145 通りの問題で、関数 $f(x)$ の呼び出し回数と計算時間を比較した。

本論文は、まず2章で Alefeld のアルゴリズムを説明し、3章でその改良した部分を説明する。4章では数値実験の結果をまとめ、最後に5章で結論を述べ、考察する。

* 工学研究科情報工学専攻 ** 工学部情報工学科

2 Alefeld の方法

本章では Alefeld のアルゴリズムについて述べる。ここでは、漸近的に最も収束特性がよく、本研究で参考にした [1] の方法のついて示す。

2.1 サブルーチンの説明

はじめに Alefeld のアルゴリズムで使うサブルーチンを定義する。常に $f(a)f(b) < 0$ が成り立と仮定する。関数 $f(x)$ が区間 [a , b] 上で連続ならば、$f(x) = 0$ の根が区間 [a , b] 上に存在することが保証される。区間 [a , b] 上の点 c に対し、サブルーチン *bracket* を次に示す。

Subroutine *bracket*$(a,\ b,\ c,\ \bar{a},\ \bar{b},\ d)$

if $f(c) = 0$, then c を出力, 終了;
if $f(a)f(c) < 0$, then $\bar{a} = a,\ \bar{b} = c,\ d = b$, else $\bar{a} = c,\ \bar{b} = b,\ d = a$.

このサブルーチンによって、$f(\bar{a})f(\bar{b}) < 0$ となる新しい区間 [$\bar{a}$, $\bar{b}$] $\subset$ [a , b] を構成する。さらに $d < \bar{a}$ ならば $f(\bar{a})f(d) > 0$ のような、そうでなければ $f(\bar{b})f(d) > 0$ となる点 $d \notin$ [$\bar{a}$, $\bar{b}$] を得る。

次にサブルーチン *Newton-Quadratic* の示す。これは $a,\ b,\ d$ と k を入力とし r を出力とする。

Subroutine *Newton-Quadratic*$(a,\ b,\ d,\ r,\ k)$

set $A = f[a,\ b,\ d],\ B = f[a,\ b]$;
if $A = 0$, then $r_k = a - B^{-1}f(a)$;
if $Af(a) > 0$, then $r_0 = a$, else $r_0 = b$;
For $i = 1, 2, \cdots, k$ do:

$$r_i = r_{i-1} - \frac{P(r_{i-1})}{P'(r_{i-1})} = r_{i-1} - \frac{P(r_{i-1})}{B + A(2r_{i-1} - a - b)} \tag{2.1}$$

$r = r_k$.

点 d は $d < a$ ならば $f(a)f(d) > 0$ のような、そうでなければ $f(b)f(d) > 0$ となる点 $d \notin$ [a , b] とする。また k は正整数である。このサブルーチンは次の二次方程式

$$P(x) = f(a) + f[a,\ b](x - a) + f[a,\ b,\ c](x - a)(x - b) \tag{2.2}$$

の 1 つの根 z の近似値 r_k を返す。ここで $f[a,\ b]$, $f[a,\ b,\ c]$ は、差分商

$$f[a,\ b] = \frac{f(b) - f(a)}{b - a}\ ,\quad f[a,\ b,\ c] = \frac{f[b,\ d] - f[a,\ b]}{d - a}$$

を表す。$P(a) = f(a),\ P(b) = f(b)$ となることから、$P(a)P(b) < 0$ となることがわかる。このように二次関数の根ではなく近似解を求めて区間 [a , b] 上の点を決める方法は [2] で採用され、[3] よりよい収束特性が示された。

[1] では、区間 [a , b] 上の点を 4 点で 3 次の補間式を作り、その根を求める方法が取り入れられた。ここで区間 I に対し関数 $f(x)$ が連続で、区間 I に関数 $f(x)$ の根が存在し、点 $a,\ b,\ c,\ d$ が区間 I に含まれるとする。点 $(a, f(a))$, $(b, f(b))$, $(c, f(c))$, $(d, f(d))$ の逆補間多項式 $IP(y)$ は、$f(a)$, $f(b)$, $f(c)$, $f(d)$ がそれぞれが異なった数値ならば、次のように表される。

$$\begin{aligned} IP(y) \ = \ a \ &+ \ (y - f(a))\ f^{-1}[f(a),\ f(b)] \\ &+ \ (y - f(a))(y - f(b))\ f^{-1}[f(a),\ f(b),\ f(c)] \\ &+ \ (y - f(a))(y - f(b))(y - f(c))\ f^{-1}[f(a),\ f(b),\ f(c),\ f(d)] \end{aligned} \tag{2.3}$$

ここで、

$$
\begin{aligned}
f^{-1}[f(a),\ f(b)] &= \frac{b-a}{f(b)-f(a)} \\
f^{-1}[f(a),\ f(b),\ f(c)] &= \frac{f^{-1}[f(b),\ f(c)]-f^{-1}[f(a),\ f(b)]}{f(c)-f(a)} \\
f^{-1}[f(a),\ f(b),\ f(c),\ f(d)] &= \frac{f^{-1}[f(b),\ f(c),\ f(d)]-f^{-1}[f(a),\ f(b),\ f(c)]}{f(d)-f(a)}
\end{aligned}
$$

である。式 (2.3) で示した多項式 $IP(y)$ は、関数 $f(x)$ が逆関数を持たなくても $f(a)$, $f(b)$, $f(c)$, $f(d)$ が異なった値ならば常に成り立つ。そのとき $\bar{x} = IP(0)$ を計算することで、$\bar{x}$ は $f(x) = 0$ の近似解とすることができる。$\bar{x}$ は区間 I の外に存在する場合があるが、漸近的には $f(a)$, $f(b)$, $f(c)$, $f(d)$ は異なる値となり、$\bar{x}$ は区間 I に含まれる。最後にこの $\bar{x} = IP(0)$ の計算をするサブルーチン *ipzero* を示す。これは a, b, c, d を入力とし $\bar{x}$ を出力とする。

Subroutine *ipzero*$(a,\ b,\ c,\ d,\ \bar{x})$

$$
\begin{aligned}
\text{set}\quad & Q_{11} = (c-d)\frac{f(c)}{f(d)-f(c)},\quad Q_{21} = (b-c)\frac{f(b)}{f(c)-f(b)},\quad Q_{31} = (a-b)\frac{f(a)}{f(b)-f(a)}, \\
& D_{21} = (b-c)\frac{f(c)}{f(c)-f(b)},\quad D_{31} = (a-b)\frac{f(b)}{f(b)-f(a)}, \\
& Q_{22} = (D_{21}-Q_{11})\frac{f(b)}{f(d)-f(b)},\quad Q_{32} = (D_{31}-Q_{21})\frac{f(a)}{f(c)-f(a)}, \\
& D_{32} = (D_{32}-Q_{21})\frac{f(c)}{f(c)-f(a)},\quad Q_{33} = (D_{32}-Q_{22})\frac{f(a)}{f(d)-f(a)}, \\
& \bar{x} = a + (Q_{31}+Q_{32}+Q_{33}).
\end{aligned}
$$

2.2 アルゴリズム

この節では連続関数 $f(x)$ の $f(a)f(b) < 0$ となる区間 [a , b] 上の単根を取り囲む方法に対し、本研究で参考にした Alefeld の [1] のアルゴリズムについて述べる。このアルゴリズムは [2] の改良で、前節のサブルーチン *bracket, Newton-Quadratic, ipzero* を使用する。関数 $f(x)$ は 1 ループで最大でも 4 回、漸近的には 3 回だけ計算される。アルゴリズム中の μ は $\mu < 1$ となる正のパラメータで、ここでは [1] と同様に $\mu = 0.5$ とする。

Alefeld のアルゴリズム:

1.1 set $a_1 = a,\ b_1 = b,\ c_1 = a_1 - f[a_1,\ b_1]^{-1}f(a_1)$;

1.2 call *bracket*$(a_1,\ b_1,\ c_1,\ a_2,\ b_2,\ d_2)$;

For $n = 2, 3, \cdots$, do:

1.3 if $n = 2$ or $\prod_{i \neq j}(f_i - f_j) = 0$　ここで$f_1 = f(a_n), f_2 = f(b_n), f_3 = f(c_n), f_4 = f(d_n)$,
　　then call *Newton-Quadratic*$(a_n,\ b_n,\ d_n,\ c_n,\ 2)$,
　　else
　　call *ipzero*$(a_n,\ b_n,\ d_n,\ e_n,\ c_n)$,
　　if $(c_n - a_n)(c_n - b_n) \geq 0$　then call *Newton-Quadratic*$(a_n,\ b_n,\ d_n,\ c_n,\ 2)$,
　　endif;

1.4 set $\tilde{e} = d_n$,　call *bracket*$(a_n,\ b_n,\ c_n,\ \tilde{a}_n,\ \tilde{b}_n,\ \tilde{d}_n)$;

1.5 if $\prod_{i\neq j}(f_i - f_j) = 0$ ここで$f_1 = f(\tilde{a}_n), f_2 = f(\tilde{b}_n), f_3 = f(\tilde{c}_n), f_4 = f(\tilde{d}_n)$,
then call *Newton-Quadratic*($\tilde{a}_n$, $\tilde{b}_n$, $\tilde{d}_n$, $\tilde{c}_n$, 3),
else
call *ipzero*($\tilde{a}_n$, $\tilde{b}_n$, $\tilde{d}_n$, $\tilde{e}_n$, $\tilde{c}_n$),
if $(\tilde{c}_n - \tilde{a}_n)(\tilde{c}_n - \tilde{b}_n) \geq 0$ then call *Newton-Quadratic*($\tilde{a}_n$, $\tilde{b}_n$, $\tilde{d}_n$, $\tilde{c}_n$, 3),
endif;

1.6 call *bracket*($\tilde{a}_n$, $\tilde{b}_n$, $\tilde{c}_n$, $\bar{a}_n$, $\bar{b}_n$, $\bar{d}_n$);

1.7 if $|f(\bar{a}_n)| < |f(\bar{b}_n)|$, then set $u_n = \bar{a}_n$, else set $u_n = \bar{b}_n$;

1.8 set $\bar{c}_n = u_n - 2f[\bar{a}_n, \bar{b}_n]^{-1} f(u_n)$;

1.9 if $|\bar{c}_n - u_n| > 0.5(\bar{b}_n - \bar{a}_n)$, then $\hat{c}_n = 0.5(\bar{b}_n + \bar{a}_n)$, else $\hat{c}_n = \bar{c}_n$;

1.10 call *bracket*($\bar{a}_n$, $\bar{b}_n$, $\hat{c}_n$, $\hat{a}_n$, $\hat{b}_n$, $\hat{d}_n$);

1.11 if $(\hat{b}_n - \hat{a}_n < \mu(b_n - a_n))$,
then $a_{n+1} = \hat{a}_n$, $b_{n+1} = \hat{b}_n$, $d_{n+1} = \hat{d}_n$, $e_{n+1} = \bar{d}_n$,
else
$e_{n+1} = \hat{d}_n$,
call *bracket*($\hat{a}_n, \hat{b}_n$, $0.5(\hat{a}_n + \hat{b}_n)$, a_{n+1}, b_{n+1}, d_{n+1}),
endif.

2.3 収束特性

前節のアルゴリズムに対し、区間数列 $\{[\ a_n\ ,\ b_n\ ]\}_{n=1}^{\infty}$ は

$$b_{n+1} - a_{n+1} \leq L(b_n - a_n)^4 (b_{n-1} - a_{n-1})^3, \quad (L : \text{定数},\ n = 2, 3, \cdots) \tag{2.4}$$

という収束特性を示す。漸近的に関数 $f(x)$ は、1 ループで 3 回呼び出されるので収束の次数は $1.668\cdots$ となることが証明される。(文献 [1])

3 Alefeld の方法の改良

この章では [1] を参考に、より効率的に関数 $f(x)$ を取り囲むアルゴリズムを考える。

3.1 改良点

Alefeld のアルゴリズム [3],[2],[1] ではすべて、前に述べたアルゴリズムの 1.7–1.9 の処理が含まれている。この処理では区間 $[\ \bar{a}_n\ ,\ \bar{b}_n\ ]$ に対し、次の式

$$\bar{c}_n = u_n - 2f[\bar{a}_n, \bar{b}_n]^{-1} f(u_n), \quad \text{ここで } |f(u_n)| = \min\{|f(\bar{a}_n)|, |f(\bar{b}_n)|\} \tag{3.1}$$

によって点 $\bar{c}_n$ を求める。式 (3.1) によって求めた点 $\bar{c}_n$ は、常に区間 $[\ \bar{a}_n\ ,\ \bar{b}_n\ ]$ に含まれ、$|f(\bar{a}_n)| = |f(\bar{b}_n)|$ の時、この区間の始点または終点と一致する。差分商と式 (3.1) より次の式が導かれる。

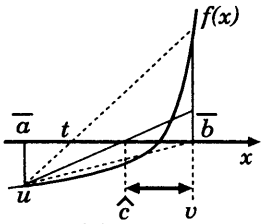
$$\begin{aligned} f(\bar{c}_n) &= f[\bar{c}_n, u_n](\bar{c_n} - u_n) + f(u_n) \\ &= (1 - 2f[\bar{a}_n, \bar{b}_n]^{-1} f[\bar{c}_n, u_n]) f(u_n) \end{aligned} \tag{3.2}$$

関数 $f(x)$ は連続微分可能で、点 $\bar{a}_n$, $\bar{b}_n$, u_n, $\bar{c}_n$ は小区間内に存在し、そこに関数 $f(x)$ の単根 x_* が含んでいると仮定する。すなわち $f(x^*) = 0$, $f'(x^*) \neq 0$ と仮定する。そのとき、$f[\bar{a}_n, \bar{b}_n] \approx f'(x_*) \approx f[\bar{c}_n, u_n]$ となることから式 (3.2) は $f(\bar{c}_n) \approx -f(u_n)$ と仮定できる。これは区間 $[\ \bar{a}_n\ ,\ \bar{b}_n\ ]$ が十分小さいとき、式 (3.1) によって効率的に根を区間 $[\ \bar{c}_n\ ,\ u_n\ ]$ (または $[\ u_n\ ,\ \bar{c}_n\ ]$) で取り囲むことを表す。

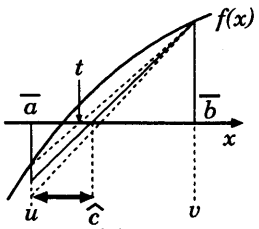
しかし式(3.1)は、ある程度区間$[\bar{a}_n, \bar{b}_n]$が根x_*に収束しないと効果がない。また、$4|f(u_n)| < |f(\bar{a}_n)| + |f(\bar{b}_n)|$が成立しないと区間幅は区間$[\bar{a}_n, \bar{b}_n]$の半分にしか小さくならない。そこで次の式

$$r_n = \frac{f[u_n, \hat{c}_n]}{f[\bar{a}_n, \bar{b}_n]}, \quad (n \geq 2) \tag{3.3}$$

で表されるr_nを定義する。区間$[\bar{a}_n, \bar{b}_n]$に対し、連続微分可能で$f'(x) \neq 0$とすると、$f[\bar{a}, \bar{b}] \approx f'(x_*) \approx f[\bar{c}, u]$となることから式(3.3)の$r_n$は1に収束する。そこで区間$[\bar{a}_n, \bar{b}_n]$の点$\hat{c}_n$を$r_{n-1}$が1に近いときは区間$[\hat{c}_n, u_n]$(または$[u_n, \hat{c}_n]$)を小さくなるように取る。また$r_{n-1}$が1より少し大きい、または小さいときは区間を大きめに取るようにした。具体的には以下のように変更した。



図A



図B

$0 \leq r_{n-1} < 1$のとき、区間$[\bar{a}, \bar{b}]$上の点$\hat{c}_n$を次のように決める。

$$\hat{c}_n = u_n - f(u_n)\frac{v_n - u_n}{(\alpha + r_{n-1})f(v_n) - f(u_n)}, \tag{3.4}$$

ここで、v_nは$|f(v_n)| = \max\{|f(a_n)|, |f(b_n)|\}$となる$v_n \in \{a_n, b_n\}$である。図Aに対し、$r_{n-1}$が1に近いときは点$\hat{c}_n$は点$t$に近付き、$r_{n-1}$が0に近いときは点$\bar{b}$に近付く。$r_{n-1} = 0$のとき点$\hat{c}_n$は点$\bar{b}_n$と重なるため、$r_{n-1}$に$\alpha$だけ加えることにした。本方法では経験的に$\alpha = 0.01$とした。これによって、図Aのような関数$f(x)$の根が点$v_n$の近くにある場合でも効率よく根を区間で取り囲めると考えられる。

次に$1 \leq r_{n-1} < 2$のとき、区間$[\bar{a}, \bar{b}]$上の点$\hat{c}_n$を次のように決める。

$$\hat{c}_n = u_n - f(u_n)\frac{v_n - u_n}{f(v_n) - r_{n-1}f(u_n)}, \tag{3.5}$$

図Bに対し、r_{n-1}が1に近いときは点$\hat{c}_n$は点tに近付き、r_{n-1}が2に近いときは点$\bar{b}$に近付く。これによってAlefeldのアルゴリズムの1.8の処理よりも、より小さく根を区間で取り囲めると考えられる。r_{n-1}が$0 \leq r_{n-1} < 2$でない場合は、$\hat{c}_n$を区間$[\bar{a}_n, \bar{b}_n]$の中点とする。

前に述べたAlefeldのアルゴリズムの1.7–1.9をこのように変更することにより、より小さい区間で単根を取り囲めると考えられる。本方法では、Alefeldの方法よりも高い収束特性を示すことはできないが、少なくとも同じ次数で収束することは明らかである。

3.2 本アルゴリズム

前節で述べたように、[1]を変更したアルゴリズムは以下のようになる。本方法も[1]と同様に、サブルーチン *bracket, Newton-Quadratic, ipzero* を使用し、関数$f(x)$は1ループで最大でも4回、漸近的には3回だけ計算される。アルゴリズム中のμは[1]と同様に$\mu = 0.5$とし、αは経験的に$\alpha = 0.01$とした。

本アルゴリズム:

2.1–2.2: 1.1–1.2 と同じ

For $n = 2, 3, \cdots$, do:

2.3–2.6: 1.3–1.6 と同じ

2.7 if $|f(\overline{a}_n)| < |f(\overline{b}_n)|$, then set $u_n = \overline{a}_n$, $v_n = \overline{b}_n$, else set $u_n = \overline{b}_n$, $v_n = \overline{a}_n$;

2.8 if $n \neq 2$ and $0.0 \leq r < 1.0$, then

$$\text{set} \qquad \hat{c}_n = u_n - f(u_n)\frac{v_n - u_n}{(\alpha + r_{n-1})f(v_n) - f(u_n)},$$

else if $n \neq 2$ and $1.0 \leq r < 2.0$, then

$$\text{set} \qquad \hat{c}_n = u_n - f(u_n)\frac{v_n - u_n}{f(v_n) - r_{n-1}f(u_n)},$$

else set $\hat{c}_n = 0.5(\overline{b}_n + \overline{a}_n)$;

2.9 set

$$r_n = \frac{f[u_n,\ \hat{c}_n]}{f[\overline{a}_n,\ \overline{b}_n]};$$

2.10–2.11: 1.10–1.11 と同じ

4 数値実験

本章では、実際に計算機上で作成した Alefeld の方法 [3] [2] [1] とそれを改良した本方法の数値実験の結果を比較・検討する。

4.1 停止条件

はじめに、プログラムの停止条件について述べる。現在の区間を [a , b] とすると、次の条件を満たしたとき、プログラムを停止する。

$$b - a \leq 2 \cdot tole(a,\ b) \tag{4.1}$$

ここで $tole$ は

$$tole(a,\ b) = 2 \cdot |u| \cdot \underline{macheps} + \underline{tol}$$

を表し、u は、$|f(u)| = \min\{|f(a)|, |f(b)|\}$ となる $u \in \{a, b\}$ である。また $\underline{macheps} = \left(\frac{1}{2}\right)^{-52} = 2.2204460492504 \times 10^{-16}$ で $\underline{tol}$ は、ユーザーが与えた負でない数をさす。

4.2 数値実験

数値実験として、[2] と同じ 145 通りの問題でテストを行った。この問題を異なる $\underline{tol}$ ($\underline{tol} = 10^{-2}$, 10^{-5}, 10^{-7}, 10^{-10}, 10^{-15}, 0) の条件で解いた。

はじめに、最も収束特性が高い [1] の方法と本方法とで、145 通りの問題の中で関数 $f(x)$ の呼び出し回数を比較し、収束がよくなった問題数、悪くなった問題数を示す。

$\underline{tol}$	10^{-2}	10^{-5}	10^{-7}	10^{-10}	10^{-15}	0
収束がよくなった問題数	86	89	94	93	88	88
変化なし	34	30	31	33	34	33
収束が悪くなった問題数	25	26	20	19	23	24

表 4.1: Alefeld の方法と本方法の比較

次に 145 通りの問題を解くために、関数 $f(x)$ が呼び出されたすべての回数を比較する。比較として、本方法と [1] の 2 つの方法 ('95(1)(2))、[2] の 2 つの方法 ('93(1)(2))、[3] の 3 つの方法 ('92(1)(2)(3)) と従来の二分法を用いた。

tol	二分法	'92(1)	'92(2)	'92(3)	'93(1)	'93(2)	'95(1)	'95(2)	本方法
10^{-2}	2203	2137	1996	1964	1975	1958	1941	1943	1537
10^{-5}	3633	2786	2644	2577	2480	2391	2435	2361	1946
10^{-7}	4622	3054	2797	2730	2648	2593	2593	2540	2147
10^{-10}	6051	3208	2906	2939	2783	2693	2739	2670	2271
10^{-15}	8286	3364	2993	3142	2869	2751	2815	2741	2357
0	12206	3406	3045	3240	2906	2768	2847	2762	2399

表 4.2: 関数 $f(x)$ を呼んだ回数

次に 145 通りの問題をすべて解くためにかかった計算時間 (単位は秒) の比較をする。使用した計算機は SONY NWS-5000 である。

tol	二分法	'92(1)	'92(2)	'92(3)	'93(1)	'93(2)	'95(1)	'95(2)	本方法
10^{-2}	1.98	2.15	2.52.	2.38	2.43	2.55	2.43	2.62	2.18
10^{-5}	3.63	2.95	3.43	2.38	3.15	3.15	3.15	3.22	2.83
10^{-7}	4.80	3.25	3.67	3.20	3.40	3.47	3.37	3.47	3.10
10^{-10}	6.48	3.47	3.85	3.42	3.57	3.60	3.60	3.65	3.33
10^{-15}	9.05	3.63	4.00	3.70	3.68	3.70	3.73	3.80	3.48
0	12.95	3.67	4.07	4.12	3.75	3.73	3.75	3.83	3.58

表 4.3: すべての問題を解く計算時間

以上の結果より本方法では、関数 $f(x)$ の呼び出し回数を最も少なく問題を解くことができた。特に、図 A のようなタイプの関数で速く収束するようになった。[1] の方法は 3 次補間式の根を計算時間が多くかかってしまうが、関数の計算が減少したために計算時間も少なくすることができた。

4.3 多重根に対しての実験

ここでは、多重根の実験を行った。初期区間 [-1 , 10] に対し、次の式

$$x^n = 0, \quad n = 3,\ 5,\ 7,\ 9,\ 19,\ 25$$

の根を求めた。問題を解くための、関数が呼び出された回数と計算時間は以下のとおりである。

tol	二分法	'92(1)	'92(2)	'92(3)	'93(1)	'93(2)	'95(1)	'95(2)	本方法
10^{-2}	72	146	182	98	106	130	124	137	115
10^{-5}	132	305	304	217	275	266	291	341	244
10^{-7}	168	399	439	299	371	379	387	449	401
10^{-10}	228	562	537	406	525	488	546	666	582
10^{-15}	322	804	747	562	757	728	779	977	878
0	966	2457	2583	1666	2423	2640	2448	3107	3398

表 4.4: 関数 $f(x)$ を呼んだ回数

tol	二分法	'92(1)	'92(2)	'92(3)	'93(1)	'93(2)	'95(1)	'95(2)	本方法
10^{-2}	0.13	0.28	0.43	0.22	0.22	0.30	0.28	0.33	0.30
10^{-5}	0.27	0.63	0.73	0.50	0.60	0.67	0.73	0.92	0.62
10^{-7}	0.33	0.83	1.08	0.78	0.82	0.90	0.97	1.20	1.07
10^{-10}	0.47	1.20	1.30	0.93	1.20	1.20	1.38	1.82	1.60
10^{-15}	0.65	1.73	1.80	1.37	1.75	1.83	2.02	2.88	2.35
0	2.12	6.40	7.33	4.55	6.08	7.33	6.93	9.30	10.30

表 4.5: すべての問題を解く計算時間

多重根に対しては、本方法も Alefeld の方法と同様に収束特性をよくすることができなかった。特に要求する区間幅が狭くなるほど、収束が悪くなっている。多重根では r_n の定義は意味がないことがわかる。

5 まとめと考察

非線形方程式の $f(x) = 0$ の根を求める方法に対し、Alefeld の方法の改良を行った。Alefeld の方法では、区間上の点を補間式を使って求めることで速く確実に根に収束する。本研究ではその次の処理の点の決め方を更に速く収束するように、区間の弦の傾きの比に依存して算法の場合分けを提案した。この操作により、Alefeld の方法では区間幅がある程度小さくならないと効果がなかった処理も、区間幅が大きくても収束特性がよくなると考えられる。

実際、[2] と同じ問題で実験を行った結果、本方法は漸近的に Alefeld の最も収束特性がよい [1] の方法より関数 $f(x)$ の呼び出し回数が減少した。区間幅を最も小さくすまでの関数 $f(x)$ の呼び出し回数は、[1] に比べて 145 通りの問題に対し 88 の問題で減少し、33 の問題では変化はなく、24 の問題では逆に増加した。全体的には関数 $f(x)$ の呼び出し回数は [1] の約 87% と減少した。

計算時間については、本方法で参考とした [1] の方法は、条件文が増え、3 次の補間式の根を求めることで 1 回のループの時間が増加したが、本方法は関数 $f(x)$ の呼び出し回数が減少した問題に対しては、その分問題を解く時間は短くなった。実験した 145 通りの問題をすべて解く時間は区間幅が最も小さくするのまでに [1] の 94% の時間で解くことができ、[3],[2] の方法よりも速く問題を解くことができた。

また多重根に対して実験をした結果、Alefeld の方法と同様本方法も収束が遅く、二分法よりも劣る結果となった。

今後の課題として、より多くの問題で数値実験を行い、収束が遅くなった関数 $f(x)$ について詳しく調べる必要がある。そしてさらに収束特性を上げるように改良し、[1] の方法の収束の次数 1.668$\cdots$ よりも高い次数の収束特性を示すことが考えられる。

参考文献

[1] G. Alefeld, F. A. Potra, and Yixun Shi, Enclosing Zeros of Continuous Functions, *ACM Trans. Math. Softw. 21* (1995), 327-344.

[2] G. Alefeld, F. A. Potra, and Yixun Shi, On enclosing simple roots of nonlinear equations, *Math.Comput. 61* (1993), 733-744.

[3] G. Alefeld and F. A. Potra, Some efficient methods for enclosing simple zeroes of nonlinear equations, *BIT 32* (1992), 334-344.

[4] 中道洋平, 非線形方程式の単根の存在区間アルゴリズム, 福井大学工学部情報工学科平成 6 年度卒業研究 (1994).

[5] 中道洋平, 長谷川武光, 佐藤義雄, 非線形方程式の単根の存在区間アルゴリズム, 日本応用数理学会平成 7 年度年会講演予稿集 (1995), 76-77.

[6] 中道洋平, 長谷川武光, 佐藤義雄, 非線形方程式の単根の存在区間アルゴリズム, 電気関係学会北陸支部連合大会講演論文集 (1995), 441.