平成 12 年度

卒業論文

区間演算による多項式の値域の評価

Range Evaluation of the Polynomial using Interval Arithmetic

平成 13 年 2 月 5 日

指導教授： 柏木 雅英 助教授

早稲田大学理工学部情報学科

G94P040-7 黒崎 俊行

目 次

第 1 章　序論 **2**

1.1　背景 3

1.2　本論文の構成 4

第 2 章　浮動小数点数 **5**

2.1　はじめに 6

2.2　IEEE754 で用いられる浮動小数点数 6

2.3　丸め 6

2.4　10 進数から 2 進数への変換 7

第 3 章　区間演算 **8**

3.1　はじめに 9

3.2　区間 9

3.3　区間演算 9

3.4　区間演算の問題点 11

第 4 章　多項式の評価 **12**

4.1　はじめに 13

4.2　普通に計算 13

4.3　ホーナー法 13

4.3.1　普通に計算との比較 14

4.4　区間の中心に関するべき 14

4.5 微分したもので割って次数を下げる 15

第 5 章 新しい方法 **17**

5.1 新しい次数を下げる方法 18

5.1.1 前の方法との比較 18

5.2 2 分法の区間版 19

5.2.1 2 分法 19

5.2.2 2 分法を用いた多項式の評価 19

5.3 2 分法への区間演算の適用 20

5.4 アルゴリズム 20

5.4.1 アルゴリズム 1 20

5.4.2 アルゴリズム 2 21

5.5 問題点 21

第 6 章 数値実行例 **23**

6.1 数値実行例 1 24

6.2 数値実行例 2 26

6.3 数値実行例 3 26

6.4 むすび 27

第 7 章 まとめ **28**

第 8 章 謝辞 **30**

第1章　序論

1.1 背景

現代のコンピュータで連続数学を計算機で扱う場合、従来数値計算は近似解を得ることのみに使われる事が多かった。すなわち実数を浮動小数点で近似して計算するのだが、計算機で扱う浮動小数点数は有限個であるためこれでは連続数学の問題を扱うことができない。初期値や実行結果などの数値には丸め誤差が含まれてしまい、得られた計算結果に何の保証もされていない。

そこで近年、計算機の能力の大幅な向上や、数値解法アルゴリズムの開発により、数値計算を近似計算のみにとどめておくのではなく、数値計算によって問題に対する解の存在と誤差限界を数学的に保証するといった研究が行われるようになった。この種の数値計算法は、精度保証付き数値計算と呼ばれており、今後の科学計算法のあるべき一つの方向として考えられるようになっている。

そこで考えられたものは 2 つの浮動小数点を両端とする「区間」の中に連続数学の解をつつみこみ、そして区間同士の演算を定義することにより行う区間演算というアイデアが生まれた。

このアイデアを生かすために、まず区間を数の拡張として考える。幅の狭い区間に求めようとする実数が入ることが示されれば実用上十分であるからである。区間を数として考えると、その四則演算が定義されなければならない。そこで定義されたものが区間演算である。

この区間演算の問題点として、区間演算を浮動小数点演算の代わりに用いると、真の解を含む区間をえられるものの、その区間幅があまりに大きすぎて有効性に乏しい結果しか得られない事が多々あった。しかし、Kulisch,Krawczyk などの研究の成果により真の解を含んだ区間の区間幅を縮小させる事が可能になった。

本論文ではこの問題点を多項式の計算において解決する方法として従来の方法と新たな方法について検証する。

1.2　本論文の構成

本論文の構成は以下の通りである。

第 2 章では浮動小数点について説明を行う。

第 3 章では区間演算について説明を行う。

第 4 章では従来の多項式の評価について説明を行う。

第 5 章では新しい多項式の評価について説明を行う。

第 6 章では数値実行例を示し検証する。

第 7 章では結論として本論文の総括を行う。

第2章　浮動小数点数

2.1 はじめに

現在の数値計算では、実数を表現するために浮動小数点数が標準として使用されている。この浮動小数点数は現在パソコン、ワークステーション、並列計算機など様々なコンピュータで使用されている。よってここではこの浮動小数点数システム上での数値計算の仕組みを説明する。

2.2 IEEE754で用いられる浮動小数点数

数値計算で用いられる数値は、浮動小数点形式で表される。

例えば倍精度浮動小数点 (C 言語の double) は 64bit で表され、その内訳は

符号 s(1bit)	指数 e(11bit)	仮数部 m(52bit)

となっている。

これは、

$$x = \pm 1.m \times 2^{e-1023}$$

の様に実数と対応している。この様に倍精度浮動小数点を用いた場合、実数の内で計算機で正確に表現できる数は有限で高々2^{64} 個にすぎない。よって、例えば浮動小数点を 2 つ加算した結果は、浮動小数点で表せるとは限らず、それに近い浮動小数点で近似する事になる。これによって発生する誤差を、丸め誤差という。通常最も近い浮動小数点に丸める事になっているが、IEEE754 に従う CPU では丸めの向きを変更する事ができる。

2.3 丸め

IEEE754 では、少なくとも次の 3 つの丸めのモードが指定できるように要請されている。ある実数 $c \in R$ があって、

- (1)：上向きの丸め c より大きい浮動小数点数の中で最も小さい数に丸める。
- (2)：下向きの丸め c より小さい浮動小数点数の中で最も大きい数に丸める。
- (3)：最近点への丸め c に最も近い浮動小数点数に丸める。
- (4)：切り捨て 絶対値が c 以下の浮動小数点数の中で c にもっとも近い物に丸める。

2.4 10 進数から 2 進数への変換

10 進整数は 2 進整数に変換できるが、10 進数で有限桁で書ける小数が必ずしも 2 進数で書けるとは限らない。例えば、

$$0.1 = 0.000110011001100.....$$

となり、計算機の中での 2 進数としては正確に 0.1 を表す事ができない。

第3章　区間演算

3.1　はじめに

浮動小数点数システム上での精度保証付き数値計算の原理をまず簡単に説明する。いま、連続数学の問題で解が実数であたえられるものを考える。浮動小数点数を用いた数値計算では、真の解そのものを計算することは一般に不可能である。そこで、真の値の下限を与える浮動小数点数と上限を与える浮動小数点数を計算する。もし、上限と加減の数値が小数点以下１５桁まで一致すれば、真の解の小数点以下１５桁まで計算できた事になる。すなわち精度保証付き数値計算の基本的なアイデアは、浮動小数点数を両端とする区間の中に連続数学の解を包み込もうというものである。

3.2　区間

区間とは

$$[x] = [\underline{x}, \overline{x}] = \{x \in R \mid \underline{x} \leq x \leq \overline{x}\}$$

と表される閉区間であるとする。

区間演算では、区間同士の加減乗除の演算を「演算結果として有り得る集合を包含するように」定義する事により行われる。2 つの区間 [x],[y] が与えられたとき 2 つの区間の四則演算を次の式で定義する。

$$[x] \circ [y] = \{x \circ y \in R \mid x \in [x], y \in [y]\}$$

ただし,$\circ \in \{+, -, \times, /\}$ これを区間演算という。

3.3　区間演算

区間演算は次の式で実現されている。

$[x] + [y] = [\underline{x} + \underline{y}, \overline{x} + \overline{y}]$

$[x] - [y] = [\underline{x} - \overline{y}, \overline{x} - \underline{y}]$

$[x] \times [y] = [\min\{\underline{x}\underline{y}, \overline{x}\overline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}\}, \max\{\underline{x}\underline{y}, \overline{x}\overline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}\}]$

$[x]/[y] = [x] \times [\frac{1}{\overline{y}}, \frac{1}{\underline{y}}]$

しかし、丸め誤差を伴う浮動小数点数演算だけしか利用できないという仮定の下で数値計算の厳密な誤差評価を行う場合、次の様な問題に直面することになる。

「ある計算の誤差を求める計算式が仮にあったとしても、
その式を計算するときの誤差は誰が計算するのか？」

区間演算においても、両端の数を浮動小数点数演算で計算せざるを得ない以上、この問題が生じる。区間演算では、この問題を解決するのに区間の両端を計算する際の丸めの向きを「外向き」にしておくことによって丸めの誤差の影響分を区間内に納めるといった方法がとられている。

具体的には、区間の下限を計算するときには「下向き」、上限を計算するときには「上向き」の丸めを行う。こうする事によって、厳密計算に比べて区間幅は大きくなるものの、区間内に真値を包含しながら計算するという区間演算の性質は損なわれない。

例　簡単な例として $(1/3) \times 3$ を 10 進数で 3 桁の有効数字を持つシステムで計算する事を考える。普通に計算すると

$$\begin{aligned} 1/3 &= 0.333 \\ 0.333 \times 3 &= 0.999 \end{aligned}$$

となる。区間演算の場合は、1 と 3 を $[1,1]$、$[3,3]$ の様に幅 0 の区間としておき、計算を行う。

$$\begin{aligned} [1,1]/[3,3] &= [0.333, 0.334] \\ [0.333, 0.334] \times [3,3] &= [0.999, 1.01] \end{aligned}$$

今の例では幅 0 の区間から出発して区間演算を行うと丸め誤差の把握が可能である。一方はじめから幅を持った区間から出発して区間演算を行えば、それは関数の値域を評価することになる。

このように、区間演算には、

- 丸め誤差の評価

- 関数の値域の評価

という 2 つの役割がある。前者は精度保証付き数値計算の基礎となるものだが、方程式の解の精度保証などにおいては、後者が重要な役割をはたす。

3.4 区間演算の問題点

区間演算の問題点の一つとして、計算された区間は真の値を含むが、区間幅が極端に広くなってしまうことが多い。これは、関数の値域を評価する場合など、区間幅が広いものを扱かった時に顕著化する。

多項式の値を評価するときも、いろいろな方法があり次の章からはこれについて述べる。

第4章　多項式の評価

4.1　はじめに

区間演算で計算をするとき計算を繰り返すと徐々に区間が広がってしまう。また方法により計算量も違うことから、今まで区間演算で多項式を計算するときいくつかの方法が考えられてきた。

4.2　普通に計算

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + ... + a_n x^n$$

これは多項式の値を左から素直に計算する方法である。この方法だと、高次の計算をすると段々区間が広がっていってしまう。また、かけ算の回数は $2n$ 回、足し算の回数は n 回となっている。

4.3　ホーナー法

一般に $a_n, a_{n-1}, ..., a_1, a_0$ を係数とする n 次多項式 $f(x)$ をホーナー法で直すと次の様になる。

$$\begin{aligned} f(x) &= a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0 \\ &= x(a_n x^{n-1} + a_{n-1} x^{n-2} + ... + a_1) + a_0 \\ &= x(x...(x(a_n x + a_{n-1}) + a_{n-2}) + ... + a_1) + a_0 \end{aligned}$$

この式の掛け算の回数は $n-1$、足し算の回数は n 回となり、普通に計算をするのと比較したとき n が大きくなればなるほど普通に計算する方法に比べて計算量が少なくなる。
又、ホーナー法は次の様に順番に求める事が出来る。

$$p_0 = a_0$$

$$
\begin{aligned}
p_1 &= p_0 x + a_1 = a_0 x + a_1 \\
p_2 &= p_1 x + a_2 = x(a_0 x + a_1) + a_2 \\
p_3 &= p_2 x + a_3 = x(x(a_0 x + a_1) + a_2) + a_3
\end{aligned}
$$

この手順の結果 $p_n = f(x)$ になる。この関係は、

$$
p_0 = a_0, p_k = p_{k-1} x + a_k (k = 1, 2, ..., n)
$$

と表される。

4.3.1　普通に計算との比較

一般に区間演算では、区間 a,b,c があれば、$ab + ac \supseteq a(b + c)$ が成り立つ。この関係から、ホーナー法は普通に計算するより区間演算をしたときに小さい区間幅多項式を評価出来る。

4.4　区間の中心に関するべき

この方法は区間の中心 c を求めて $x - c$ の多項式に入れ替える方法である。

$$
x = [a, b], c = \frac{a + b}{2}
$$

$$
\begin{aligned}
f(x) &= a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0 \\
&= b_n (x - c)^n + b_{n-1} (x - c)^{n-1} + ... + b_1 (x - c) + b_0
\end{aligned}
$$

掛け算をするときに区間の中心にすることにより、区間幅の増大を防いでいる。

a_n から b_n への並び替えは組立除法により簡単に求められる。

関数 $f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + ... + a_nx^n$ の係数を $a[0], a[1], ..., a[n]$ としてこれを、上の b_n に並び替える具体的なアルゴリズムは以下の様になる。

```
for(int i=1;i<=n;i++){
    for(int j=n;j>=i;j--){
        a[j-1]+=a[j]*c;
    }
}
```

これは、以下の様に a[i] を更新している。

$$
\begin{aligned}
& a_nx^n + a_{n-1}x^{n-1} + ... + a_1 + a_0 \\
&= (x-c)(a_{n-1}x^{n-1} + ... + a_1) + a_0 \\
&= (x-c)((x-c)(a_{n-2}x^{n-2} + ... + a_1)) + a_0 \\
&\quad ... \\
&= (x-c)((x-c)((x-c)(...((x-c)((x-c)+a_1))..)) + a_0 \\
&= (x-c)((x-c)((x-c)(...((x-c)((x-c)(x-c))))) + a_0
\end{aligned}
$$

$a[0], a[1], ..., a[n]$ はこのようにどんどん更新されていく。最終的に終わった段階の各 $a[i]$ が求める b_n の値となっている。

4.5 微分したもので割って次数を下げる

$$f(x) = f'(x)p(x) + q(x) \tag{4.1}$$

$f(x)$ の値域は、その区間の両端の値もしくは極値の値となる。上の式で極値 $f'(x) = 0$ の地点では $f(x) = q(x)$ となる。これを繰り返していけば、極値での次数をさげていくことができ、その値は両端の区間に含まれることがわかる。よって、$q(x)$ の両端の値と、式 4.1

で $f(x)$ を $q(x)$ で置き換えた、余りの両端での値を次々計算していけば、求める区間はその区間の和になる。

第5章　新しい方法

4.5 節の微分したもので割って次数を下げる方法と似た方法で、より精度の高い解を求められる方法と、 2 分法の区間版について解説する。

5.1 新しい次数を下げる方法

多項式

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + ... + a_n x^n \tag{5.1}$$

において n を $f_n(x)$ の次数とする。

$$\begin{aligned} f_n(x) &= p(x) \\ f_{n-1} &= f_n(x) - \frac{1}{n} x f_n'(x) \end{aligned}$$

$f(x)$ から $f(x)$ の微分に $\frac{1}{n}x$ を掛けて引くと、極値の値を 1 次下げて求める事が出来るようになる。これを上の「 微分したもので割って次数を下げる 」方法と同じ原理で両端の値を求めていくと、$f(x)$ の極値での値を包含する区間が求められる。

5.1.1 前の方法との比較

この方法は、「 微分したもので割って次数を下げる 」方法では、次数が 2 次づつ下がっていくのに比べ、次数のへりが 1 次づつになっている。

また、この方法は「 微分したもので割って次数を下げる 」方法に比べ、最高時の係数が非常に小さい場合に優れた結果を残す。これは、最高時の係数が非常に小さい場合、$f(x) = f'(x)p(x) + q(x)$ の $q(x)$ の係数が非常に大きくなってしまうことから起こると予想される。

5.2　2 分法の区間版

まず区間演算抜きでの 2 分法とそれを使った多項式の評価の方法について説明する。

5.2.1　2 分法

区間 $[a, b]$ で連続な関数 $y = f(x)$ が $f(a)f(b) < 0$ を満たせば、この区間 $[a, b]$ 内に少なくとも一つ $f(x) = 0$ を満たす実根が存在する。今、何らかの方法で、区間 $[a_0, b_0]$ に $f(x) = 0$ を満たす実根の存在がわかったとする。このときこの区間に含まれる解の推定値として区間の中点 $c_0 = \dfrac{a_0 + b_0}{2}$ を選ぶ。このとき、$f(c_0) = 0$ なら c_0 が根であり、$f(c_0) \neq 0$ なら 2 つの区間 $[a_0, c_0]$,$[c_0, b_0]$ のうちどちらかに根を含む。根を含む方を改めて $[a_1, b_1]$ として再び上の手順を繰り返す。これを繰り返せば毎回区間の幅は $\dfrac{1}{2}$ になるから、n 回の 2 分割の後では、根の推定値は $\dfrac{b_0 - a_0}{2^n}$ の区間幅の中に収まる。n の値を適当に選ぶことにより実用上十分な精度が得られる。このようにして近似根の精度を上げていく方法を 2 分法 $(\mathrm{bisection\ method})$ という。

5.2.2　2 分法を用いた多項式の評価

$f(x)$ の区間 I の値域はその両端の値もしくは、その極値での値となる。

極値の x 座標、つまり $f'(x) = 0$ の解の求め方は以下のようになる。

$f'(x) = 0$ をの x を求めるには、$f''(x) = 0$ の解を求めてから 2 分法を用いればよい。また $f''(x) = 0$ の解を求めるには、$f^{(3)} = 0$ の解を求めてから 2 分法を使えばよい。これを繰り返せば最終的には $f^{(n-1)} = 0$ を求める事が出来れば全ての解を求める事ができる。$f^{(n-1)}(x)$ は一次式のため簡単に求める事が出来る。

$f'(x)$ の根が求まれば、$f(x)$ の極値がわかり、その値と与えられた区間 I の両端の値から最大最小を選べば、多項式 $f(x)$ の評価が出来る。

5.3　2 分法への区間演算の適用

まず、1 次式 $f^{(n-1)}(x) = 0$ の解を求める際に、この解は区間として求められる。解の区間 $[\underline{x}, \overline{x}]$、求める区間 I を $[a, b]$ とすれば、まず区間 (1)、$[a, \underline{x}]$ で 2 分法を使いこの区間内に解があるかを調べる。次に、区間 (2)、$[\underline{x}, \overline{x}]$ に解があるかを調べる。この区間では、普通に区間演算を行い、この区間での値域の範囲を求め解の存在を判定する。もし、解があることがわかれば、この区間全体を極値の候補として考えることにする。そして最後に、区間 (3)、$[\overline{x}, b]$ で 2 分法を使い解があるかを調べ求める。

5.4　アルゴリズム

プログラムする際の具体的なアルゴリズムは以下の 2 個となる。

5.4.1　アルゴリズム 1

多項式の全ての解を含む区間の集合を求める。

与式 $f(x) = 0$ 区間 $I = [\underline{I}, \overline{I}]$

(1) $f'(x) = 0$ の全ての解を含む区間の集合 $(I_1, \cdots, I_n)$ を得る。

(2) 区間集合

$$[\underline{I}, \underline{I_1}]$$

$$[\overline{I_1}, \underline{I_2}]$$

$$[\overline{I_2}, \underline{I_3}]$$

$$\vdots$$

$$[\overline{I_{n-1}}, \underline{I_n}]$$

$$[\overline{I_n}, \overline{I}]$$

において、$f(x)$ は単調増加または単調減少。よって、これらの区間 $[a,b]$ について

(a) $f(a)>0$ かつ $f(b)>0$ なら $[a,b]$ に解なし。

(b) $f(a)<0$ かつ $f(b)<0$ なら $[a,b]$ に解なし。

(c) (a),(b) のどちらでもない場合、[a,b] は解の候補、2 分法をつかってなるべく小さくしておく。

(3) $I_1,\cdots,I_n$ においては単調性はいえない。よって、$f(I_1),\cdots,f(I_n)$ を区間演算で評価し、$f(I_k)\ni 0$ ならば、I_k は解の候補。

5.4.2 アルゴリズム 2

多項式の値の評価。

与式 $f(x)=0$ 区間 $I=[\underline{I},\overline{I}]$

(1) $f'(x)=0$ の全ての解を含む区間 $I_1,\cdots,I_n$ をアルゴリズム 1 で求める。

(2) $I_1,\cdots,I_n$ において、$f(I_1),\cdots,f(I_n),f(\underline{I}),f(\overline{I})$ を評価する。

5.5 問題点

アルゴリズム 1 の (2),(c) において、2 分法を使ってなるべく範囲をせばめるところで、$f(a)$,$f(b)$ も区間になっているために、あまり範囲が狭まらないうちに、2 分法が終わってしまう可能性がある。

例えば、$f(x)$ があり a と b の間で 2 分法をおこなうとして、最初の中点 c で $f(c)$ の区間が 0 を含むと、その時点で 2 分法が終わってしまい、[a,b] が解の候補となってしまう。

これについては、区間演算を使った 2 分法としてより精度の高い解を求める際考慮する必要があるが、今のところ解決策は見つかっていない。だが 2 分法の回数を増やし、解の候補となる区間をなるべく狭めておく事により、次の段階の解を見つける際、上の様な事

はおきにくいように思われる。

第6章　数値実行例

6.1　数値実行例 1

普通の方法、ホーナー法、区間の中心に関するべき、微分した物で割って次数を下げる、微分した物を引いて次数を下げる、２分法、のそれぞれの方法の実行結果を調べる。
以下の表は関数 $f(x) = x^3 + 6x^2 + 11x + 6$ について、区間を設定し、その実行結果を調べたものである。

区間	普通の方法	ホーナー法	区間の中心	微分で割る
[-5,-1]	[-168,144]	[-44,76]	[-24,24]	[-24,2]
[-3.5,-1]	[-69.375,67.5]	[-23.75,28.75]	[-3.90625,4.375]	[-1.875,1]
[-3,-1]	[-48,48]	[-18,18]	[-2,2]	[-0.666667,0.6666667]
[-2.5,-1]	[-31.125,31.5]	[-12.75,9.75]	[-1.6875,1.21875]	[-0.666667,0.375]
[-2,-1]	[-18,18]	[-8,5]	[-1,0.25]	[-0.666667,1.42109×10^{-104}]
[-1.5,-1]	[-7.875,7.5]	[-3.75,2.5]	[-0.65625,0]	[-0.666667,0]

微分を引く	２分法
[-24,19.333]	[-24,0.384925]
[-6.83333,4.83333]	[-1.875,0.384193]
[-5,2.33333]	[-0.384906,0.384906]
[-3.16667,2.33333]	[-0.384906,0.375]
[-1.33333,2.33333]	[-0.384906,0]
[-0.5,2.33333]	[-0.384906,0]

次に実行速度について調べたのが次図となる。
多項式の係数には適当な値を入れ、次数は２０次で１０００回繰り返し実行した結果で、使用コンピュータは、celeron 550 memory 192MB、単位は秒となっている。

普通の方法	ホーナー法	区間の中心	微分で割る	微分を引く	2分法
0.173	0.041	0.49	1.781	1.774	76.547

普通に計算以外のすべての方法について区間計算が必要な場合はすべてホーナー法で行っている。また2分法の回数は５０回繰り返している。

6.2　数値実行例 2

次に、$f_{n-1} = f_n(x) - \frac{1}{n} x f_n'(x)$ と微分した物で割って次数を下げる方法についての詳しい比較と数値実行例を観察する。

最高次の係数がものすごく小さい場合の例を試してみる。

$f(x) = 10^{-9}x^5 - x^4 + x^3 - x^2 + x + 1$ この多項式を、微分した物で割る方法、$f'(x)$ に $\frac{1}{n}x$ をかけたものを引いて次数を下げる方法、2 分法の 3 つで評価した物を調べた。

区間	微分で割る	微分を引く	2 分法
[-1.01,2.01]	$[-9.35284 \times 10^8, 4.0806 \times 10^8]$	[-9.23191,1.80603]	[-9.23191,1.32645]

このように、微分した物で割る方法は、最高次の係数がものすごく小さい場合に精度がきわめて悪くなる。その点 $f'(x)$ に $\frac{1}{n}x$ をかけたものを引いて次数を下げる方法はそういう事もなく優れた結果を残す。

6.3　数値実行例 3

最後に 2 分法について、解の候補が無駄に多くなってしまう現象についての数値例を挙げたいと思う。

多項式 $f(x) = x^4 - 14x^3 + 63x^2 - 106x + 56$、区間 [1.42,6.01] で、2 分法の回数 N を変化させて解の候補をみると以下の様な結果が得られた。

N=2 [1.42,1.55][1.94,2.46][3.03375,3.6075][4.755,5.3825][5.85312,6.01]

N=3 [1.42,1.485][1.94,2.2][2.83875,3.15813][4.755,5306875][5.89234,6.01]

N=4 [1.42,1.46062][2.99844,3.15812][5.94137,6.01]

N=30 [1.4303,1.4303][3.10753,3.10753][5.96217,5.96217]

この例では、N=2,3 のとき解の候補が無駄に多く出てきている。この様にだいたいの多項式では、十分に 2 分法の回数を増やしてやる事により、無駄な解の候補を含めずに多項式の評価を行える。

また、今回は見つけられなかったが、上の様な状況が N を大きくしても起きる可能性はある。

6.4 むすび

微分した物で割って次数を下げる方法、$f'(x)$ に $\frac{1}{n}x$ をかけたものを引いて次数を下げる方法共に優れた結果を残していることがわかる。これら 2 つの方法は時間もかからずに精度の高い結果を残すことができる。この 2 つの方法を使う場合はその方程式の性質をよく見どちらを使うか検討したほうがよい。

また、区間演算の 2 分法に関してはまだ不完全なところもあるがほぼすべてにおいて他の方法より優れた結果を残す事ができた。実行時間についてもプログラムの作り方次第でまだまだ改善する事ができると思う。

第7章　まとめ

本論文では多項式の区間評価において計算を繰り返しても区間が広がらない 2 分法のアルゴリズムと $f'(x)$ に $\frac{1}{n}x$ をかけたものを引いて次数を下げる方法について検証した。現代ではコンピュータを使って数値計算をするとき、浮動小数点演算によるものが主流になっており、その数値には当然丸めの誤差が含まれる。そのため、計算値の精度保証について近年様々なところで研究がなされてきたが、その一つに区間演算がある。この区間演算の弱点として計算を繰り返す間に区間幅が極端に大きくなってしまうことがある。そこで区間幅の増加を縮小する様々な方法が考えられてきた。ここで説明した 2 つの方法はともに従来の方法より優れた面がある。が、 2 分法は計算時間と、最悪の場合ホーナー法とあまり変わらぬ結果がでてしまう可能性がある。また $f'(x)$ に $\frac{1}{n}x$ をかけたものを引いて次数を下げる方法では、そこそこ良い結果がでるが、 2 分法に比べると精度は甘いという点が挙げられる。実行時間を考えず、精度を上げたい場合は、本論文の 2 分法の方法を使うのが好ましい。

第8章　謝辞

この研究を進めるとき、終始丁寧な御指導および御激励をいただき、また様々な面でもいろいろと御助言をいただきまた面倒を見てくださった柏木雅英助教授に深く感謝いたします。

また、卒業論文中間発表報告の際など機会のあるごとに御指導、御鞭撻のほどを賜り、研究に方向性をくださった大石進一教授に深く感謝いたします。

また、助手の宮田孝富さん、博士課程 2 年の濱田吉信さん、修士課程 2 年の岩折朱希嗣さん、高崎大輔さん。修士課程 1 年の金谷卓充さん、河原淳さん、白井健一さん、長友泰崇さん、波多野伸哉さん、深谷光統さん、吉田直史さんに深く感謝いたします。

また、意見の交換や協力などをしてくださいました柏木研究室 4 年の石田寛氏、伊志嶺寛之氏、菊池智行氏、塩田孝一氏、新岡幸治氏、西中川英氏、古川周一氏、宝迫 敦氏、宮島信也氏に深く感謝いたします。

最後に、研究だけでなく日常の生活の中でお世話になりました柏木研究室、大石研究室の皆様に深く感謝いたします。

参考文献

[1] 応用解析セミナー, 数値計算, 大石　進一, 裳華房,1999

[2] コンピュータによる数値計算, 水上孝一, 朝倉邦造, 朝倉書店,1985

[3] 数値計算, 片岡重延, 室岡和彦, 志賀清一, 東京電気大学出版局,1955

[4] 現代非線形科学シリーズ６, 精度保証付き数値計算, 大石　進一, コロナ社,2000

[5] 卒業論文, 多項式の区間評価, 深谷 光統,2000