平成11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12年2月4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

G96P107-4 深谷光統

目 次

図 目 次

表 目 次

第 1 章

序論

1.1 背景

現代のコンピューター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ことが多かった。すなわち実数を浮動小数点数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、数値開放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 つの浮動小数点数を両端とする「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事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浮動小数点数演算の変わりに用いると、真の解を含む区間を得られるものの、その区間幅があまりに大きすぎ有効性に乏しい結果しか得られないことが多々あった。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ことが可能になった。

本研究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区間の中の最大値、最小値となりうる値と区間の端の値とを大小比較して区間となる解を求める。最大値、最小値になりうる値は区間の中の関数の極値であるためまず関数を微分して方程式を求める。本論文ではこの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章では浮動小数点数について説明を行う。

第３章では区間解析について説明を行う。

第４章では区間演算について説明を行う。

第５章では機械区間演算について説明を行う。

第６章では従来の多項式の評価について説明を行う。

第７章では新しい多項式の評価について説明を行う。

第８章では数値の実行例を示し検証する。

第９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が標準として使用されている。この浮動小数点は現在パソコン、ワークステーション、並列計算機などさまざまなコンピューターで使用されている。よってここではこの浮動小数点システム上での数値計算の仕組みを説明する。[3]

2.2 IEEE 標準 754 にもとづくシステム

2 進規格化浮動小数点は IEEE 標準 754 で定義されたものであり以下のようにして表現されている。ここで d_i は 0 または 1 である。

$$a = \pm(\frac{1}{2} + \frac{d_2}{2^2} + \frac{d_3}{2^3} + \frac{d_4}{2^4} + \ldots + \frac{d_N}{2^N}) \times 2^e$$

e は $e_{min} \leq e \leq e_{max}$ となる整数であり e_{max} が正の整数 e_{min} を負の整数とする。

$$m = \frac{1}{2} + \frac{d_2}{2^2} + \frac{d_3}{2^3} + \frac{d_4}{2^4} + \ldots + \frac{d_N}{2^N}$$

を符号付き仮数といい、e を指数という

$$a = \pm m \times 2^e$$

通常、単精度、倍精度、拡張倍精度の浮動少数システムがあるが、それらについて説明する。

単精度

$$(N, e_{min}, e_{max}) = (23 + 1, -126, 127)$$

倍精度 (64bit)

$$(N, e_{min}, e_{max}) = (52 + 1, -1022, 1023)$$

拡張倍精度 (80bit)

$$(N, e_{min}, e_{max}) = (64, -16382, 16383)$$

これにおいて仮数部に 1 を足したのは IEEE754 において単精度、倍精度において仮数部の最初の桁は 1 となっているからである。

2 進浮動小数システムの絶対値の最大値を x_{max}、最小値を x_{min} とおくと、

$$x_{max} = (\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^2} + \ldots + \frac{1}{2^N}) \times e_{max}$$

$$x_{min} = \frac{1}{2} 2^{e_{min}}$$

となる。

2.2.1 特別な値

このとき x の絶対値 $|x|$ が x_{max} より大きいときオーバーフローが生じたといい x_{min} より小さいときアンダーフローが生じたという。また

$$0 = +(\frac{0}{2} + \frac{0}{2^2} + \ldots + \frac{0}{2^N}) 2^{e_{min}}$$

である。

IEEE754 で用意された他の特別な数として不当な演算の結果として得られる NaN(Not a Number)、オーバーフローの結果や 0 で割った結果の不当な結果として得られる $\pm\infty$、アンダーフローか $\pm\infty$ での割り算の結果として得られる ± 0 などがある。

2.2.2 ulp

関数 ulp(t)(unit in the last place) は、実数 t を挟む 2 つの浮動小数点数の差を表す。

浮動小数点数システム R の要素の数は有限個で、区間 $[-x_{max}, x_{max}]$ の上に原点対称に分布する。

2.2.3 桁落ち

絶対値がほぼ等しく、かつ、符号が同じ数値を引き算すると、その結果の有効桁数が極度に少なくなる現象を桁落ち (cancelling in digits/cancellation of significant digits) と呼ぶ。

例えば

$$345.678 - 345.666 = 0.012$$

の計算では、6 桁の有効桁数が引き算することによって、わずか 2 桁に減少してしまう。一般に桁落ちは数値計算のいろいろなところで発生する。しかしそれに気づかず計算を続けたため、計算結果が予想していたものより大きく異なることがよくある。桁落ちには十分注意してプログラムを書かなければならない。

2.3 丸め（IEEE754）

IEEE754 では少なくとも次の 3 つの丸めの方向のモードが指定できるように要請している。$c \in R$ とする。

2.3.1 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。これを $\triangle : R \Rightarrow R$ と表す。

2.3.2 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。これを $\bigtriangledown : R \Rightarrow R$ と表す。

2.3.3 0 捨 1 入による丸め

c に最も近い浮動小数点数に丸める。これを $\square : R \Rightarrow R$ と表す。

2.3.4 丸めの演算のみたす条件

丸めの演算を映像として $\bigcirc : R \to R$ と書く。

すなわち

$$\bigcirc \in \{\triangle, \bigtriangledown, \square\}$$

このとき IEEE754 では丸めの演算は次の条件を満たす。

$$\bigcirc x = x \text{（任意の } x \in R \text{ について）}$$

$$x \leq y \Rightarrow \bigcirc x \leq \bigcirc y \text{（任意の } x, y \in R \text{ について）}$$

$x \in R$ のとき、-x や $|x|$ は符号を変えることにより正確に計算できる。

2.3.5　丸めの演算の性質

IEEE754 上では次のことが成立する。

$$\square(-x) = -\square x \text{（任意の } x \in R \text{ について）}$$

$$\triangle(-x) = -\triangle x \text{（任意の } x \in R \text{ について）}$$

$$\triangledown(-x) = -\triangledown x \text{（任意の } x \in R \text{ について）}$$

2.3.6　浮動小数点演算の定義

IEEE754 では浮動小数点の演算は丸めとの関係により次のように定義されている。

$$\cdot \in \{+, -, \times, /\}, \bigcirc \in \{\triangle, \triangledown, \square\} \text{ のとき}$$

$$x \odot y = \bigcirc(x \cdot y) \text{（任意の } x \in R \text{ について）}$$

これは左辺は右辺の数学的に正しい四則演算の結果 $x \cdot y$ を指定された丸めを行って得られた数 $\bigcirc(x \cdot y)$ に一致するように計算されることを意味している。

2.4　むすび

ここで浮動小数点数について概説した。これは精度保証を考える時のすべての基本である。これらを踏まえ以降区間解析、機械区間解析などについて解説する。

第 3 章

区間解析

3.1　はじめに

精度保証の数値計算は実数値で与えられる真の解のを２つの値ではさみそれを浮動小数点演算によりよって計算するものである。ここでは区間解析について説明する。区間解析は浮動小数点システムにおいてどうしても出てきてしまう誤差をどのようにして補うかを考えて考え出されてものであり区間を数の延長として考えるのである。ここで四則演算を定義したものを区間演算という。この演算を浮動小数システム上において行うものを機械区間演算という。ここではそれらについて説明する。[3]

3.2　区間について

3.2.1　区間の定義

区間解析において実数 R 上で考えるとすると当然四則演算はできる。

区間とは２つの実数に挟まれた下のように表すことのできる閉区間のことである。

$$[\underline{x}, \bar{x}] = \{x \in R | \underline{x} \leq x \leq \bar{x}\}$$

ただし $\underline{x} \leq \bar{x} \in R$ で、$\underline{x}$ を下端、$\bar{x}$ を上端という。

ここでは区間を $[x]$ と表すことにする。すなわち

$$[x] = [\underline{x}, \bar{x}]$$

である。

3.2.2　点区間の定義

点区間とは

$$\underline{x} = \bar{x}$$

となるものである。これは一つの実数である。そのためこれを x と書く

3.2.3　直径, 半径, 中心の定義

直径 $d([x])$

$$d([x]) = \underline{x} - \bar{x}$$

半径 $r([x])$

$$r([x]) = \frac{\underline{x} - \bar{x}}{2}$$

中心 $m([x])$

$$m([x]) = \frac{\underline{x} + \bar{x}}{2}$$

3.2.4　最大、最小絶対値の定義

最小絶対値 $\langle [x] \rangle$

$$\langle [x] \rangle = min|x||x \in [x]$$

最大絶対値 $|[x]|$

$$|[x]| = max|x||x \in [x] = max\{|\underline{x}|, |\bar{x}|\}$$

3.3　むすび

次の章ではこれを使って区間演算について解説する。

第 4 章

区間演算について

4.1 はじめに

本章では、精度保証付き数値計算の基礎となる区間演算について説明する。[3]

4.2 演算の定義

区間が二つ与えられたとき、その２つの区間の四則演算を定義する。

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

ここで$\circ \in \{+, -, \times, /\}$とする。この定義では区間演算は実数演算を無限回しないと実行できないような雰囲気があるが実は下のことで済む。このようにすれば有限回の実数演算で区間演算を実行することができる。

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

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

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

$$[x]/[y] = [x] \times [\frac{1}{\bar{y}}, \frac{1}{\underline{y}}](0 \notin [y])$$

また掛け算、割り算については場合分けをすることにより、より少ない手順で演算することができる。

4.2.1 区間演算の場合分け

$[\underline{x}, \bar{x}] \times [\underline{y}, \bar{y}]$

	$[\underline{x}, \bar{x}] > 0$	$0 \in [\underline{x}, \bar{x}]$	$[\underline{x}, \bar{x}] < 0$
$[\underline{y}, \bar{y}] > 0$	$[\underline{x}\underline{y}, \bar{x}\bar{y}]$	$[\underline{x}\bar{y}, \bar{x}\bar{y}]$	$[\underline{x}\bar{y}, \bar{x}\underline{y}]$
$0 \in [\underline{y}, \bar{y}]$	$[\bar{x}\underline{y}, \bar{x}\bar{y}]$	$[min\{\underline{x}\bar{y}, \bar{x}\underline{y}\}, max\{\underline{x}\underline{y}, \bar{x}\bar{y}\}]$	$[\underline{x}\bar{y}, \underline{x}\underline{y}]$
$[\underline{y}, \bar{y}] < 0$	$[\bar{x}\underline{y}, \underline{x}\bar{y}]$	$[\bar{x}\underline{y}, \underline{x}\underline{y}]$	$[\bar{x}\bar{y}, \underline{x}\underline{y}]$

$[\underline{x}, \bar{x}] \div [\underline{y}, \bar{y}]$

	$[\underline{x}, \bar{x}] > 0$	$0 \in [\underline{x}, \bar{x}]$	$[\underline{x}, \bar{x}] < 0$
$[\underline{y}, \bar{y}] > 0$	$[\underline{x}/\bar{y}, \bar{x}/\underline{y}]$	$[\underline{x}/\underline{y}, \bar{x}/\underline{y}]$	$[\underline{x}/\underline{y}, \bar{x}/\bar{y}]$
$[\underline{y}, \bar{y}] < 0$	$[\bar{x}/\bar{y}, \underline{x}/\underline{y}]$	$[\underline{x}/\bar{y}, \bar{x}/\bar{y}]$	$[\bar{x}/\underline{y}, \underline{x}/\bar{y}]$

4.3 むすび

ここでは、区間演算について解説した。次章ではこれをもとに機械区間演算について解説する。

第 5 章

機械区間演算について

5.1 はじめに

前の節で解説した区間演算の問題点として浮動小数点数システムにおいて数値計算をするとき厳密な区間演算は実行できない。なぜなら区間演算において両端の数はじっすであるとし、厳密な実数計算にもとづいて区間演算が定義されていたからである。これを解決する方法として数学的な正解が含まれる区間が演算の結果えられることを念頭に、区間演算を与えられた浮動小数点システム R 上に展開する方法をここで説明する。 [3]

5.2 機械区間とは

機械区間とは下で定義されている IR の要素である。

$$IR = \{[x] \in IR | \underline{x}, \bar{x} \in R\}$$

機械区間は上端と下端の数が浮動小数点で与えられる区間の定義である。

5.3 機械区間の丸めの定義

区間 $[a, b] \in IR$ を区間 $[c, d] \in IR$ に丸める区間の丸めを以下のように定義する。

$$[x] \in \diamond[x]$$

また次をみたすように定義する。

$$\diamond[x] = [x] \text{（すべての } [x] \in IR \text{について）}$$

$$[x] \subset [y] \longrightarrow \diamond[x] \subset \diamond[y] \text{（すべての } [x][y] \in IR \text{について）}$$

$$\diamond(-[x]) = -\diamond[x] \text{（すべての } [x] \in IR \text{について）}$$

この条件をみたすため IEEE754 上の浮動小数点システムにおいては以下のように定義する。

$$\diamond[x] = [\bigtriangledown\underline{x}, \bigtriangleup\bar{x}]$$

5.4　四則演算の定義（機械区間演算）

機械演算の四則演算を次のように定義する。

$$\diamond([x] \circ [y])$$

ただし $\circ \in \{+, -, \times, /\}$ である。

5.5　むすび

本章では、機械区間演算について解説した。計算機上で区間演算を導入しようとするとこの演算が必要になる。これにより精度保証を行うことができるようになる。

第 6 章

従来の方法

6.1 はじめに

区間演算で計算をするとき計算を繰り返すとだんだんと区間が広がっていってしまう。いままで区間演算で多項式を計算をするときいくつかの方法が考えられてきた。ここでは今までのアルゴリズムを説明する。

6.2 普通に計算

$$a_0 + a_1x + a_2x^2 + a_3x^3 \ldots + a_nx^n$$

これは多項式を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。

6.3 ホーナー法

ここではホーナー法について説明する。[1] 次の式を下のように表現することができる。$f(x) = 2x^3 - 3x^2 + 4x - 5 = x(2x^2 - 3x + 4) - 5 = x\{x(2x - 3) + 4\} - 5$ これは $f(x)$ を x で割っていって得られる。この形の式をホーナーの式といいこの形に直すことをホーナー法という。この形にすると掛け算と割り算はわずか 3 回である。一般に $a_n, a_{n-1}, \ldots, a_1, a_0$ を係数とする n 次の多項式 $f(x)$ をホーナー法で直すと、次のようになる。

$$\begin{aligned} f(x) &= a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n \\ &= x(a_0x^{n-1} + a_1x^{n-2} + \ldots + a_{n-1}) + a_n \ldots \\ &= x\{x \ldots \{x(a_0x + a_1) + a_2\} + \ldots + a_{n-1}\} + a_n \end{aligned}$$

この式の掛け算の回数は $n - 1$ 回となり、普通に計算をするのと比較したとき n の数が大きくなればなるほど計算量が少なくなる。

またホーナー法は次のように順番に求めることができる。

$$p_0 = a_0$$

$$p_1 = p_0 x + a_1 = a_0 x + a_1$$

$$p_2 = p_1 x + a_2 = x(a_0 x + a_1)$$

$$p_3 = p_2 x + a_3 = x\{x(a_0 x + a_1) + a_2\} + a_3$$

この手順の結果 $p_n = f(x)$ になる。この関係は、

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

と再帰式で表わされる。このようにホーナー法は計算量が少ないため区間演算をしたとき小さい区間で多項式を解くことができまた再帰的なためコンピューターにとって普通に計算をするよりも都合がいいことがある。

6.4 区間の中心に関するべき

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

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

$$\begin{aligned} f(x) &= a_0 x^n + a_1 x^{n-1} + \ldots + a_{n-1} x + a_n \\ &= b_0 + b_1 (x - c) + b_2 (x - c)^2 + \ldots + b_n (x - c)^n \end{aligned}$$

6.5 微分をしたもので割って次数を下げる

$$f(x) = f'(x) p(x) + q(x)$$

このようにして $f(x)$ を $f'(x)$ で割って計算をすれば次数を一つ下げることができるため高次の計算では普通に計算をするのに比べて区間が広がりにくくなる。

6.6　むすび

本章では多項式の区間を計算する従来の方法について概説を行った。次章では 2 分法を取り入れた方法について説明する。

第 7 章

新しい方法

7.1　はじめに

ここでは前章で説明した従来の方法に比べて精度において優れているアルゴリズムを説明する。

7.2　アルゴリズム

$f(x)$ の区間 I の中の最大値は区間 I の端もしくは $f(x)$ の極大値である。また最小値は区間 I の端もしくは極小値である。また極致とは $f'(x)=0$ を満たす。よって $f'(x)=0$ 0 を満たす $f(x)$ と区間の端の $f(x)$ の最大値、最小値を調べれば $f(I)$ を求めることができる。

$f'(x)$=*0* の解の求め方。

$f'(x)=0$ を求めるには $f''(x)=0$ の解を求めてから 2 分法を使えば良い。また $f''(x)=0$ を求めるためには $f^{(3)}(x)=0$ を求めてから 2 分法を使えば良い。これを繰り返せば最終的には $f^{(n-1)}(x)=0$ を求めることが出来ればすべての解を求めることができる。そしてここで $f^{(n-1)}$ は一次式のため求めることが可能である。

7.3　２分法

区間 $[a,b]$ で連続な関数 $y=f(x)$ が $\mathrm{f}(a)f(b)<0$ を満たせば、この区間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 $f(x)=0$ を満たす実根が存在することがわかったとする。このときこの区間に含まれる解の推定値 c_0 として区間の中点 $c_0=\frac{a_0+b_0}{2}$ を選ぶ。$f(c_0)=0$ なら c_0 が根であり $f(c_0)=0$ なら $f(c_0)\neq 0$ 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。根を含むほうを改めて (a_1,b_1) として再び上の手順を繰り返す。これを繰り返せば区間の幅は毎回 $\frac{1}{2}$ になるから n 回の 2 分割の後では、根の推定値は $\frac{b_0-a_0}{2^n}$ の区間の幅に追いつめられて n 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あげていく方法を 2 分法 (bisection method) という。[2]

7.4　むすび

本章では多項式の区間を評価する新しい方法の理論を解説した。次の章ではしかしこのプログラムは精度では優れているが時間がかかってしまいそうである。次の章では時間を計って検証してみる。

第 8 章

数値実行例

表 8.1: 実行時間

次数	中心	微分	普通	ホーナ	２分
30	5	6	0	0	0
60	5	6	0	0	17
90	7	9	3	0	71
120	11	17	5	0	165
150	16	22	5	0	324
180	28	33	6	0	560
210	44	49	6	0	857
240	60	71	6	0	1323
270	88	105	6	0	1917
300	121	137	6	0	2664

8.1　はじめに

本章では、前々章で解説した普通に計算をする方法と区間の中心を求めて多項式を入れ替えて行う方法とホーナー法、微分して一次次数を下げて計算する方法と前の章で解説した新しい方法の結果について検証する。

8.2　実行例

ここで従来の４つのプログラムと２分法を用いた方法のプログラムの実行時間について表にした。区間の範囲、多項式の係数はランダムな値を入れた。

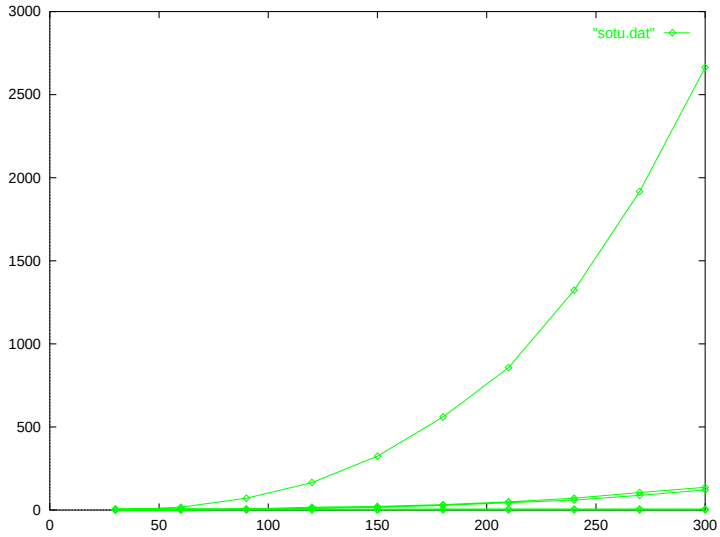

図 8.1: 実行結果

8.3 むすび

一番時間がかからないのがホーナー法である。そして微分して次数を下げて計算する方法は精度は普通に計算をするのに比べて良くはなるが微分をしたりするため少しだけ時間がかかってしまうため次数が低すぎると効率的でない。また区間の中心に関するべきは並び替えるのにやや時間がかかるため微分して次数を下げるより時間がかかってしまう。この結果から 2 分法の方法は他のアルゴリズムに比べ時間がかかってしまう。

第 9 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分法等を用いたアルゴリズムについて説明した。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となっており、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開発がなされてきたが、 その１つに区間演算である。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である。そこで様々な区間幅の増加を縮小する方法が考えてこられた。ここで説明した方法は従来の方法に比べ区間幅の増加を抑えるという点で精度は上がるが計算時間がかなりかかってしまう。今後の課題としてより短い時間で計算する方法を考えたい。

謝辞

この研究を進めるとき、終始丁寧なご指導および御激励をいただき、またさまざま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ご指導、ご鞭撻のほどを賜り、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、日常生活においていろいろとお世話になりました、柏木研究室助手の相馬隆郎さん、博士課程2年宮田孝富さん、博士課程1年の濱田吉信さん、修士課程1年の岩折朱希嗣さん、高崎大輔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下さいました柏木研究室4年の金谷卓充氏、川上修氏、小泉健氏、桜井幹夫氏、白井健氏、州濱陽一氏、長友泰崇氏、波多野伸哉氏、村竹範彦氏、山田浩之氏、吉田直史氏、渡部啓氏に深く感謝致します。

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] 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.ISBN4-254-12577-1C3341.

[2] 数値計算. 片桐重延. 室丘和彦. 志賀清一. 東京電機大学出版局.1955.ISBN4-501-52370-0 C 3355.

[3] 数値計算. 大石進一. 裳華房.1999.ISBN 4-7853-1514-8

[4] 非線形解析入門. 大石進一. コロナ社.1997.ISBN 4-339-02600-X.