平成19年度修士論文

アフィンベキ級数演算を利用した常微分方程式の精度保証

Accuracy guarantee of ordinary differential equation using Affine PSA

平成20年2月4日

指導教授　柏木雅英　准教授

早稲田大学大学院理工学研究科情報・ネットワーク専攻

3606U027-7　柏木 啓一郎

目 次

図 目 次

表 目 次

第1章　序論

1.1 背景

非線形現象を伴う回路や各種システムに対する解析は, 数値計算によって行うのが一般的である．ところが, 数値計算で得られた結果には様々な誤差が混入している．近年，精度保証付き数値計算という分野が発展し，得られた近似解と真の解の誤差がどのくらいなのか，あるいは近似解の近くに真の解が存在するのかどうかといった，近似解の正しさを数学的に保証する技術が考えられている．そうした中，アフィン演算とよばれる区間演算が提案され，研究されている [1]．このアフィン演算は計算の際に変数間の相関が考慮されるところに特徴があり，従来の区間演算で起こりがちな区間幅の爆発的な増大を抑える効果が期待できる．そうした特徴からアフィン演算 は様々な分野での応用が見込まれる．

また，振り子などの運動方程式は常微分方程式の初期値問題で表せるが，初期値鋭敏性が強い．つまり，初期に混入した小さな誤差が，最終的に非常に大きな誤差へとなってしまう．

1.2 本論文の目的

ベキ級数演算では，初期に混入した丸め誤差などが Wrapping Effect などにより拡大され続けるために，反復を繰り返すうちに区間幅が極端に増大してしまう．そのために，今まではベキ級数演算に平均値形式を用いた方法が使われてきた．その原因となる区間演算の Wrapping Effect を解決する演算方法がアフィン演算 である．アフィン演算 は変数間の相関を考慮するため，演算結果の区間幅の極端な広がりを抑えるという利点を持つ．

本論文では，ベキ級数演算，及びアフィン演算の特性を利用し，高速なアフィン演算を用いたベキ級数演算を提案し，実際に常微分方程式の初期値問題を解き，その有効性を示す．

1.3 本論文の構成

本論文の序盤においては本論文において必要な知識について述べる．まず，2 章において精度保証付き数値計算の基礎となる計算機による数値計算での数の扱いとその誤差の問題，区間演算による計算結果の精度保証の考え方について述べる．次に，3 章で平均値形式について，その原理と

特徴を述べる．次に，4 章では区間演算のひとつであるアフィン演算について，その原理と特徴を述べる．次に，5 章では，さらに丸め誤差を考慮したアフィン演算について解説をする．

本論文の中盤においては，ベキ級数演算の説明，及び既存の方法についての説明を行う．まず，6 章では，ベキ級数演算について述べる．次に，7 章では，平均値形式を用いたベキ級数演算について述べる．

本論文の終盤においては，本論文における提案方法，及びその有効性についての説明を行う．まず，8 章において，アフィン演算を用いたベキ級数演算について述べる．そして，9 章で，それぞれの方法を数値例をもとに比較し，まとめる．

第2章　区間解析

2.1 はじめに

本章では区間演算の考え方について述べる．まず計算機における数の表現について簡単に述べ，計算機を用いて連続数学の問題を解く場合の数の扱いと誤差の問題，数の拡張として考えられた区間演算により解を包み込むことで精度を保証しようという考え方について述べる．

2.2 計算機上の数

計算機では実数を浮動小数点数で近似して計算する．浮動小数点数システムについては IEEE 標準 754 が広く用いられているので，まずこの IEEE754 による 2 進浮動小数点数システムについて簡単に述べる．ここでは倍精度浮動小数点数のみを扱う．

2.2.1 データ形式

倍精度浮動小数点数は 64bit からなり，

s(符号部,1bit)	e(指数部,11bit)	m(仮数部,52bit)

のように表される．

s は符号を表し，0 なら正，1 なら負である．指数部 e は符号なし整数とみて $0 \leq e \leq 2047$ の値をとるが，$e = 0$ と $e = 2047$ を特別な場合の表現に使用することにし，$1 \leq e \leq 2046$ のとき（正規化数という），上の浮動小数点数は，

$$x = \pm 1.m \times 2^{e-1023} \tag{2.1}$$

という実数を表現している．正規化数の場合先頭は必ず 1 なのでそれはメモリに格納しない．これによって，52bit で 53bit の精度をもつことができる．倍精度浮動小数点数は正規化されている場合 10 進でおよそ 15 桁の精度をもっている．正規化数以外に，

- $0 \quad (e = m = 0)$
- $\pm\infty \quad (e = 2047, m = 0)$
- NaN $\quad (e = 2047, m \neq 0)$

のような特別な数がある．$\pm\infty$ はオーバーフローやゼロでの割り算の結果など，NaN は $\sqrt{-5}$ や ∞/∞ のような不当な演算の結果などで得られる．

また，正規化数よりも絶対値が小さい数は仮数部の先頭を 1 とせず，非正規化数として表現される．このときは $e = 0$ とし，

$$x = \pm 0.m \times 2^{-1022} \tag{2.2}$$

という実数に対応する．この場合，精度は 53bit より少なくなっている．

2.2.2 丸めの切り替え

倍精度浮動小数点数システムで表現できる数は 2^{64} 個以下であり有限個である．したがって，任意の実数を表現することはできない．また，浮動小数点数は四則演算について閉じていない．たとえば，ふたつの浮動小数点数の和が浮動小数点数で表せるとは限らない．これらのような場合にはその結果を本来の結果に近い浮動小数点数で近似する．この操作を丸めといい，この近似によって生じる誤差を丸め誤差という．いま，$c \in \mathbf{R}$ を丸めるとすると，その方法には

上方向への丸め ($\triangle$) c 以上の最小の浮動小数点数で近似する

下方向への丸め ($\triangledown$) c 以下の最大の浮動小数点数で近似する

最近点への丸め ($\diamondsuit$) c に最も近い浮動小数点数で近似する．このような数が 2 つあるときは，仮数部の最後のビットが偶数であるような浮動小数点数に近似する（偶数丸め方式）

0 方向への丸め 絶対値 $|c|$ 以下で c に最も近い浮動小数点数で近似する

の種類がある．IEEE754 にしたがった CPU ではこの丸めの方法を指定することができる．精度保証付き数値計算においてはこのうち上方向への丸め，下方向への丸め，最近点への丸めの利用が重要となっているが，IEEE754 では以下の性質を満たすことが要請されている．$\bigcirc \in \{\triangle, \triangledown, \diamondsuit\}$，任意の $x, y \in \mathbf{R}$ に対し，

$$\bigcirc x = x \tag{2.3}$$

$$x \leq y \Rightarrow \bigcirc x \leq \bigcirc y \tag{2.4}$$

$$\bigcirc(-x) = -(\bigcirc x) \tag{2.5}$$

さらに，

$$\cdot \in \{+, -, \times, /\}$$

に対し，IEEE754 では浮動小数点数演算 $\bigodot$ を，

$$x \bigodot y = \bigcirc(x \cdot y) \quad (x, y \in \mathbf{R}) \tag{2.6}$$

で定義する．式 (2.6) の右辺は実数での演算による数学的に正しい結果（これは実数）を丸めて得られる浮動小数点数であり，これに一致するように左辺が定義されている．

2.2.3 10 進小数と 2 進小数の変換

多く扱われる 10 進数は計算機においては 2 進数に変換される．このとき注意すべきことは

10 進の有限小数がかならずしも 2 進の有限小数で表せるとは限らない

ことである．10 進の 0.1 を 2 進で表現すると

$$0.000\dot{1}100\dot{0}$$

と循環小数になり，これは有限桁では表せない．10 進の 0.1 は浮動小数点数では正確に表現できず丸めにより近似が行われることになる．また，逆に浮動小数点数を 10 進で表示する場合（C 言語の printf など），2 進小数を 10 進小数に変換するが，ここには丸めによる誤差が入りうる．したがって，精度保証付きで計算結果を表示するには丸めの方向を制御できる表示関数を自分で作る必要がある．

2.3 区間演算

上に述べたように浮動小数点数は有限個であり，厳密に実数を扱うことはできない．そこで，浮動小数点数を両端にもつ区間で解を包み込むということが考えられた．この数の拡張としての区間演算について，ここではひとまず計算機のことを忘れ，四則演算で閉じた実数体での区間演算を考える．

2.3.1 区間の定義

区間 $[x]$ は,

$$[\underline{x}, \overline{x}] = \{x \in \mathbf{R} | \underline{x} \leq x \leq \overline{x}\} \tag{2.7}$$

であらわされる閉区間とする．ここで $\underline{x} \leq \overline{x} \in \mathbf{R}$ であり，$\underline{x}$ を $[x]$ の下端，$\overline{x}$ を $[x]$ の上端という．$\underline{x} = \overline{x}$ のとき $[x]$ は点区間となり，これは実数となる．また，このような $[x]$ からなる閉区間の集合を $I\mathbf{R}$ とかく．

区間について以下を定義する．

$$\mathrm{mid}([x]) = \frac{\overline{x} + \underline{x}}{2} \tag{2.8}$$

$$\mathrm{rad}([x]) = \frac{\overline{x} - \underline{x}}{2} \tag{2.9}$$

$\mathrm{mid}([x]), \mathrm{rad}([x])$ をそれぞれ区間 $[x]$ の中心，半径という．

2.3.2 四則演算

数の拡張として区間の四則演算を定義する．以下 $\cdot \in \{+, -, \times, /\}$ とする． 区間 $[x] = [\underline{x}, \overline{x}], [y] = [\underline{y}, \overline{y}]$ に対して,

$$[x] \cdot [y] = \{x \cdot y \in \mathbf{R} | x \in [x], y \in [y]\} \tag{2.10}$$

と表される集合により定義する．ただしこのような集合を求めるには無限回の演算を行う必要はなく,

$$[x] + [y] = [\underline{x} + \underline{y}, \overline{x} + \overline{y}] \tag{2.11}$$

$$[x] - [y] = [\underline{x} - \overline{y}, \overline{x} - \underline{y}] \tag{2.12}$$

$$[x] \times [y] = [\min\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}, \max\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}] \tag{2.13}$$

$$[x]/[y] = [x] \times [1/\overline{y}, 1/\underline{y}] \tag{2.14}$$

とすることで有限回の演算で済む．ただし，除算は $0 \notin [y]$ のときのみ定義される．また，乗算と除算については表 2.1，表 2.2 のように上端と下端による場合分けで演算回数を減らすことが可能である．

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

表 2.1: 区間の乗算 $[x] \times [y]$

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

表 2.2: 区間の除算 $[x]/[y]$

2.3.3 関数の値域評価

今度は初等関数を区間上の関数に拡張することを考える．そのためには次のように定義すればよい．

$$f([x]) = \{f(x) | x \in [x]\} \tag{2.15}$$

ただし，関数 f は $\mathbf{R}$ の部分集合において定義され，その任意の閉区間において連続であるとする．例えば，

$$\begin{aligned} \sqrt{[x]} &= [\sqrt{\underline{x}}, \sqrt{\overline{x}}] \quad (\underline{x} \geq 0) \\ \exp([x]) &= [\exp(\underline{x}), \exp(\overline{x})] \end{aligned}$$

さて，次にこれらの初等関数の合成や四則演算からなる関数を区間上に拡張することを考えるのだが，このような関数の評価を厳密に行うのは困難なことがある．そこで，このような関数の演算規則を区間演算でおきかえて得られる関数を区間拡張とよび，$f_{[\,]}([x])$ とかく．このとき，

$$f([x]) \subseteq f_{[\,]}([x]) \tag{2.16}$$

が成り立つ．また，f を同値な式で書きかえたとき f の区間拡張は一般に異なる．

例　$f(x) = x^2 - 2x$ を区間 $[0.9, 1.1]$ で評価することを考える．このとき $f(x)$ を3通りに表現してその区間拡張を比べてみる．真の評価 $f([0.9, 1.1]) = [-1, -0.99]$ である．

$f(x) = x^2 - 2x$

$$\begin{aligned} f_{[\,]}([0.9, 1.1]) &= [0.9, 1.1]^2 - 2 \times [0.9, 1.1] \\ &= [0.81, 1.21] - [1.8, 2.2] \\ &= [-1.39, -0.59] \end{aligned}$$

$f(x) = x(x - 2)$

$$\begin{aligned} f_{[\,]}([0.9, 1.1]) &= [0.9, 1.1] \times ([0.9, 1.1] - 2) \\ &= [0.9, 1.1] \times [-1.1, -0.9] \\ &= [-1.21, -0.81] \end{aligned}$$

$f(x) = (x - 1)^2 - 1$

$$\begin{aligned} f_{[\,]}([0.9, 1.1]) &= ([0.9, 1.1] - 1)^2 - 1 \\ &= [-0.1, 0.1]^2 - 1 \\ &= [0, 0.01] - 1 \\ &= [-1, -0.99] \end{aligned}$$

(2.16) で等式が成立するとき，厳密に包み込んでいるという．上の例では関数 f の表現を変えることで，得られる区間の幅にして 80 倍もの差が出ている一方，3 番目の区間拡張では厳密な包み込みができていることがわかる．どのように式を表現したら良い区間拡張が得られるかというのは難しい問題である．

なお，関数 f の値域 $f([\underline{x}, \overline{x}])$ を $f([\underline{x}, \overline{x}]) \subseteq [\underline{y}, \overline{y}]$ なる区間 $[\underline{y}, \overline{y}]$ で評価するとき，この $[\underline{y}, \overline{y}]$ を $f([\underline{x}, \overline{x}])$ の区間包囲という．区間拡張は区間包囲のひとつである．

2.4　計算機上の区間演算

今度は先に述べた実数体上の区間演算を有限の浮動小数点数システムに拡張した機械区間演算の実現を考える．以下浮動小数点数の集合を F とおく．この区間演算によって，求めたい実数解が幅の狭い区間の中に入ることがいえたならばそれは十分価値のあることといえる．

2.4.1 区間の定義

区間 $[x]$ を,

$$[\underline{x}, \overline{x}] = \{x \in \mathbf{R} | \underline{x} \leq x \leq \overline{x}\} \tag{2.17}$$

で定義する．ここで $\underline{x} \leq \overline{x} \in F$ であり，$\underline{x}$ を $[x]$ の下端，$\overline{x}$ を $[x]$ の上端という．すなわち，実数上の区間の定義で上端と下端が浮動小数点数であるものとなる．このような $[x]$ からなる閉区間の集合を IF とかく．

丸めのモード

$$\bigcirc \ : \ I\mathbf{R} \to IF$$
$$\bigcirc \in \{\triangle, \triangledown, \diamondsuit\}$$

とする．丸めによって $I\mathbf{R}$ の区間 $[\underline{x}, \overline{x}]$ を IF の区間に移すには,

$$[x] \subseteq \bigcirc[x] \quad ([x] \in I\mathbf{R}) \tag{2.18}$$
$$\bigcirc[x] = [x] \quad ([x] \in IF) \tag{2.19}$$
$$[x] \subseteq [y] \Rightarrow \bigcirc[x] \subseteq \bigcirc[y] \quad ([x], [y] \in I\mathbf{R}) \tag{2.20}$$
$$\bigcirc(-[x]) = -(\bigcirc[x]) \quad ([x] \in I\mathbf{R}) \tag{2.21}$$

をそれぞれ任意の $[x]$,$[y]$ についてみたすようにする．IEEE754 にしたがった浮動小数点数システムであれば,

$$\bigcirc[x] = [\triangledown\underline{x}, \triangle\overline{x}] \tag{2.22}$$

とすれば上の条件は満たされる．

2.4.2 四則演算

$I\mathbf{R}$ での四則演算をもとに，IF 上の四則演算を定義する．

$$\cdot \in \{+, -, \times, /\}$$
$$\bigcirc \in \{\triangle, \triangledown, \diamondsuit\}$$

に対し，IF の演算を

$$[x] \bigodot [y] = \bigcirc([x] \cdot [y]) \tag{2.23}$$

で定義する．(2.22) から，具体的には次のようにすればよい．

$$
\begin{aligned}
[x]+[y] &= [\bigtriangledown(\underline{x}+\underline{y}), \bigtriangleup(\overline{x}+\overline{y})] \\
[x]-[y] &= [\bigtriangledown(\underline{x}-\overline{y}), \bigtriangleup(\overline{x}-\underline{y})] \\
[x]\times[y] &= [\min\{\bigtriangledown(\underline{x}\underline{y}), \bigtriangledown(\underline{x}\overline{y}), \bigtriangledown(\overline{x}\underline{y}), \bigtriangledown(\overline{x}\overline{y})\}, \\
&\quad \max\{\bigtriangleup(\underline{x}\underline{y}), \bigtriangleup(\underline{x}\overline{y}), \bigtriangleup(\overline{x}\underline{y}), \bigtriangleup(\overline{x}\overline{y})\}] \\
[x]/[y] &= [\min\{\bigtriangledown(\underline{x}/\underline{y}), \bigtriangledown(\underline{x}/\overline{y}), \bigtriangledown(\overline{x}/\underline{y}), \bigtriangledown(\overline{x}/\overline{y})\}, \\
&\quad \max\{\bigtriangleup(\underline{x}/\underline{y}), \bigtriangleup(\underline{x}/\overline{y}), \bigtriangleup(\overline{x}/\underline{y}), \bigtriangleup(\overline{x}/\overline{y})\}]
\end{aligned}
$$

また，乗算と除算の場合には実数の場合と同様に，表 2.1，表 2.2 を利用して演算回数を少なくすることができる．そのためには，下端の計算を下向きで，上端の計算を上向きで行えばよい．

2.5 おわりに

本章では計算機での数値の取り扱いと区間演算について述べた．計算機を用いて数値計算を行う場合，実数を浮動小数点数で近似しており，そこには必ず誤差が生じる．一方，CPU の丸めの制御と区間演算の手法を利用することで浮動小数点数の四則演算での誤差を把握することが可能である．区間演算は精度保証付き数値計算における非常に重要な技法のひとつである．(区間解析についてくわしくは [2]，[3])

なお，本論文の数値実験における区間演算は機械区間演算のことをさす．

第3章　平均値形式

3.1 はじめに

区間演算における過大評価の問題を本質的に解決するための方法として変数間の相関性を考慮しつつ計算を行う方法が考えられる．本章で説明する，平均値形式（Mean Value Form）はそのような方法で，代表的なものの一つである．

3.2 平均値形式

次のようなある関数

$$f(x), \quad f : \mathbb{R}^n \to \mathbb{R}^m$$

に対して，$I \in \boldsymbol{IR}^n$ の像 $f(I) = \{f(x) \mid x \in I\}$ を評価することを考える．$x, c \in I$ に対して，平均値の定理

$$f(x) - f(c) = \int_0^1 f'(c + t(x - c))\mathrm{d}t(x - c)$$

が成立する．ここで，

$$\int_0^1 f'(c + t(x - c))\mathrm{d}t \in \mathrm{co}\{f'(x) \mid x \in I\}$$

(co は凸包) が成立することを利用すると，$f(I) = \{f(x) \mid x \in I\}$ は

$$f(c) + F'(I)(I - c)$$

のように評価出来る．ここで F' は f の導関数 f' の区間包囲である．区間行列 $F'(I)$ は導関数 f' に区間 I を代入して区間演算を行うことによって計算することが出来る．区間行列は凸集合であるから，凸包を考える必要は無い．c は 入力区間 I に含まれる任意の点だが，普通は I の中心に取る．

また，変数の数が多いときは実用的ではないが，一変数関数に関してはより高次の Taylor 展開を考える方法も役に立つ．Taylor 展開

$$f(x) = \sum_{k=0}^{n} \frac{1}{k!} f^{(k)}(c)(x - c)^k + R_{n+1}$$

に対して，Bernoulli の剰余項

$$R_{n+1} = \frac{1}{n!}\int_0^1 (1-t)^n f^{(n+1)}(c+t(x-c))\mathrm{d}t(x-c)^{n+1}$$

を考える ($f^{(k)}$ は f の k 階導関数)．ここで，$s = (1-t)^{n+1}$ とおくと，

$$R_{n+1} = \frac{1}{(n+1)!}\int_0^1 f^{(n+1)}(c+(1-\sqrt[n+1]{s})(x-c))\mathrm{d}s(x-c)^{n+1}$$

と変型できるので，平均値形式と同様に考えて，

$$\sum_{k=0}^{n}\frac{1}{k!}f^{(k)}(c)(I-c)^k + \frac{1}{(n+1)!}f^{(n+1)}(I)(I-c)^{n+1}$$

のような区間演算を行うことで $f(I)$ の包含が得られる．平均値形式は $n=0$ の場合に相当する．

3.3 自動微分法

上記の平均値形式を実際に用いるとき，f の導関数 f' に区間を代入して微分 (ヤコビ行列) を区間行列として計算する必要が生じる．f が簡単で導関数が手で計算できるようなものならば問題はないが，長大なプログラムで f が記述されている場合など，それは困難であることが多い．数値的に微分を得る方法として $(f(x+\Delta x)-f(x))/\Delta x$ のような計算に基づく数値微分法がよく用いられるが，この方法では誤差の把握が出来ず，また区間値に対する微分を得ることは原理的に不可能である．

自動微分法は，関数 $f:\mathbb{R}^n \to \mathbb{R}$ に対して，その値の計算方法がプログラムの形で与えられたとき (すなわち与えられた値 x に対して $f(x)$ が数値計算出来るとき)，$f'(x)$ を数値的に計算する方法である．この方法では,「f を数式的に手で微分してそれに x を代入する」という操作をすることなく，それと全く同じ解が得られる．

詳細は，[4] を見て頂くのが早いが，ここで簡単に説明しておく．なお，以下で説明する方法は文献 [4] で言うところの **BU(ボトムアップ) 算法**と呼ばれるものに相当する．

BU 算法では，関数の計算中に現われる全ての変数において，通常の値に加えて入力変数 $x_1,\cdots,x_n$ に関する偏微分値を保持する．すなわち，n 変数関数の計算では全ての数値は

$$(v, \frac{\partial v}{\partial x_1}, \frac{\partial v}{\partial x_2}, \cdots, \frac{\partial v}{\partial x_n})$$

のような $n+1$ 次元ベクトルで表現される.

まず，入力変数 $x_1, \cdots, x_n$ の値 $v_1, \cdots, v_n$ が与えられているとして，入力変数を表す計算機中の変数を

$$\begin{aligned} x_1 &= (v_1, 1, 0, 0, \cdots, 0, 0) \\ x_2 &= (v_2, 0, 1, 0, \cdots, 0, 0) \\ &\vdots \\ x_n &= (v_n, 0, 0, 0, \cdots, 0, 1) \end{aligned}$$

の様に初期化する.

また，変数 p, q が $p = (p_0, p_1, \cdots, p_n)$，$q = (q_0, q_1, \cdots, q_n)$ で表されていたとき，二項演算 $r = g(p, q)$ ($p + q$，$p \times q$ など) や，単項演算 $r = h(p)$ ($\sin(p)$ など) は，次のように計算する:

$$\begin{aligned} r_0 &= g(p_0, q_0) \\ r_i &= \frac{\partial g}{\partial p}(p_0, q_0)p_i + \frac{\partial g}{\partial q}(p_0, q_0)q_i \quad (1 \leq i \leq n) \\ r_0 &= h(p_0) \\ r_i &= \frac{\mathrm{d}h}{\mathrm{d}p}(p_0)p_i \quad (1 \leq i \leq n) \end{aligned}$$

この規則に従って通常の関数の計算順序で計算を進めて行けば，最終結果 (と副産物として中間変数) の入力変数に関する偏微分値が関数値と同時に得られるはずである.

平均値形式等に現われる導関数の区間包囲 $F'(I)$ の計算については，自動微分のための $n+1$ 次元ベクトルの各要素を全て区間とし，初期化時の $v_1, \cdots, v_n$ を区間として計算を行なえばよい.

3.4 おわりに

本章では，区間演算における，区間の増大を抑えるための方法である，平均値形式について説明した.

第4章　アフィン演算

4.1 はじめに

アフィン演算は区間演算の一手法で，1994 年に Stolfi らによって提案された．アフィン演算における計算は変数間の相関を考慮して行われるため，通常の区間演算で起こりやすい区間幅の爆発的な増大を抑える効果がある．本章ではアフィン形式と区間表示との変換，アフィン形式での演算について述べる．

4.2 アフィン演算について

アフィン演算においては，変数 x はアフィン形式

$$x = x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n \tag{4.1}$$

で表される．ここに ε_i は $-1 \leq \varepsilon_i \leq 1$ の値をとることがわかっているダミー変数とする．式 (4.1) の右辺をアフィン演算多項式という．

4.2.1 初期化

入力区間 $[\underline{I}, \overline{I}]$ はつぎのようにアフィン演算形式に初期化する．

$$\frac{\overline{I} + \underline{I}}{2} + \frac{\overline{I} - \underline{I}}{2}\varepsilon \tag{4.2}$$

一般に，入力が区間ベクトル，区間行列などの場合は成分ごとに別のダミー変数を用意することになる．

4.2.2 アフィン演算多項式の区間表示

アフィン形式 (4.1) を区間表示 $[\underline{x}, \overline{x}]$ にもどすには，

$$\underline{x} = x_0 - \Delta, \quad \overline{x} = x_0 + \Delta \tag{4.3}$$

とすればよい．ここで，

$$\Delta = \sum_{i=1}^{n} |x_i| \tag{4.4}$$

4.2.3 線形演算

アフィン多項式

$$\begin{aligned} x &= x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n \\ y &= y_0 + y_1\varepsilon_1 + \cdots + y_n\varepsilon_n \end{aligned}$$

と実数 α に対して,

$$x \pm y = (x_0 \pm y_0) + \sum_{i=1}^{n}(x_i \pm y_i)\varepsilon_i \tag{4.5}$$

$$x \pm \alpha = (x_0 \pm \alpha) + \sum_{i=1}^{n} x_i\varepsilon_i \tag{4.6}$$

$$\alpha x = (\alpha x_0) + \sum_{i=1}^{n}(\alpha x_i)\varepsilon_i \tag{4.7}$$

で線形演算を定義する．複号同順.

4.2.4 非線形演算

非線形演算の像は一般には (4.1) のようなアフィン多項式で表すことはできない．このため，アフィン演算の非線形演算では像を線形な式で近似し，新たなダミー変数を設け，近似との誤差の項を追加して得られる領域をその結果とする.

4.2.4.1 非線形単項演算

$f(x)$ を非線形単項演算とする．$x \in I$ とすると，I における $f(x)$ の像 $f(I)$ は曲線を描くためアフィン形式では表現できない．アフィン演算では，I における $f(x)$ の近似を

$$ax + b \tag{4.8}$$

とし，$f(x)$ と (4.8) との最大誤差 δ を,

$$\delta = \max_{x \in I} |f(x) - (ax + b)| \tag{4.9}$$

で求め，ダミー変数 ε_{n+1} を追加して，新たなアフィン演算多項式

$$(ax+b)+\delta\varepsilon_{n+1} \tag{4.10}$$

を得て，これを $f(I)$ の包含とする．アフィン演算による非線形単項演算の評価のようすを 4.3 節の図 4.1 に示す.

4.2.4.2 非線形二項演算

アフィン演算 多項式 x, y:

$$x = x_0 + x_1\varepsilon + \cdots + x_n\varepsilon_n \tag{4.11}$$

$$y = y_0 + y_1\varepsilon + \cdots + y_n\varepsilon_n \tag{4.12}$$

に対する非線形二項演算 $z = f(x, y)$ について考える. 二項演算の場合も, 式 (3.11), 式 (3.12) より

$$\delta = \max_{-1\le\varepsilon_i\le 1} |f(x,y) - (z_0 + z_1\varepsilon_1 + \cdots + z_n\varepsilon_n)| \tag{4.13}$$

の最小値と, そのときの $z_0, \cdots, z_n$ の値が求まれば最適であるが, これを直接解くことは極めて難しい.

単項演算の場合と同様に,x と y の変域 X, Y を求め,$(x, y) \in X \times Y$ において $f(x, y)$ を最良近似する一次式 $ax + by + c$ を求め,

$$ax + by + c + \delta'\varepsilon_{n+1}(\delta' = \max_{\substack{x\in X\\ y\in Y}} |f(x,y) - (ax+by+c)|) \tag{4.14}$$

を結果とする方法を使うことはできる．しかし, 単項演算の場合と違ってこの方法は必ずしも最良の近似を与えるとは限らない．これは,x と y に相関がある場合, 点 (x, y) は一般に $X \times Y$ で与えられる長方形領域内全ての点をとるとは限らないためである.

このように二項演算の場合最良近似問題は未解決である．しかし, 現状では $f(x, y)$ を領域 $X \times Y$ において $ax + by + c$ で近似する方法をとっている．つまり, 式 (3.18) の z を決定する ($z_0, \cdots, z_{n+1}$ の値を決定する) かわりに,(それより性能の悪い評価であるが) 式 $ax + by + c + \delta'\varepsilon_{n+1}$ における a, b, c, δ' の値を決定する.
例えば [1] で定義された乗算 (Stolfi の乗法) の場合,

$$\begin{aligned}\delta' &= \max_{\substack{x\in X\\ y\in Y}} |x\times y - (ax+by+c)| \\ &= \max_{\substack{x\in X\\ y\in Y}} |(x-b)(y-a) + (-ab-c)| \end{aligned} \tag{4.15}$$

より,a, b, c が

$$a = y_0, b = x_0, c = -ab = -x_0 y_0 \tag{4.16}$$

のとき δ' の値が最小になり,

$$\delta' = \max_{\substack{x \in X \\ y \in Y}} |(x - x_0)(y - y_0)| \tag{4.17}$$

となる．以上より乗算を計算するには,

$$x \times y = y_0 x + x_0 y - x_0 y_0 + \delta' \varepsilon_{n+1} \tag{4.18}$$

を行なえばよい.

この δ' の値は, 一般的には, 式 (4.13) で定義される δ より大きめになってしまい, 必ずしも常に最良の値をとるわけではないが, 必ず真の値を包含し, また計算しやすいので現時点ではアフィン形式どうしの乗算の主流となっている.

アフィン形式どうしの除算についても様々な方法がある．本論文では除算は，逆数を取って乗算する事で実現している.

4.2.4.3 非線形演算の包含

このように，アフィン演算では非線形演算が行われるごとにダミー変数の個数が増える．また，アフィン演算での非線形演算は，δ と，それを与える a, b, c を決定することに帰着し，δ を最小にすることにより得られた $f(I)$ の包含が演算の結果として与えられる.

4.3 計算例

ここで例題を取り上げてアフィン演算と区間演算の違いをみることにする．そのまえに非線形単項演算のひとつである x^2 の包含の求め方 (a, b, δ の決定法) を図 4.1 に示す.

例題 4.1 $f(x) = 2x - x$ を区間 $I = [-2, 3]$ で評価する．$2x - x = x$ であるから，当然真の評価は $f(I) = [-2, 3]$ となる.

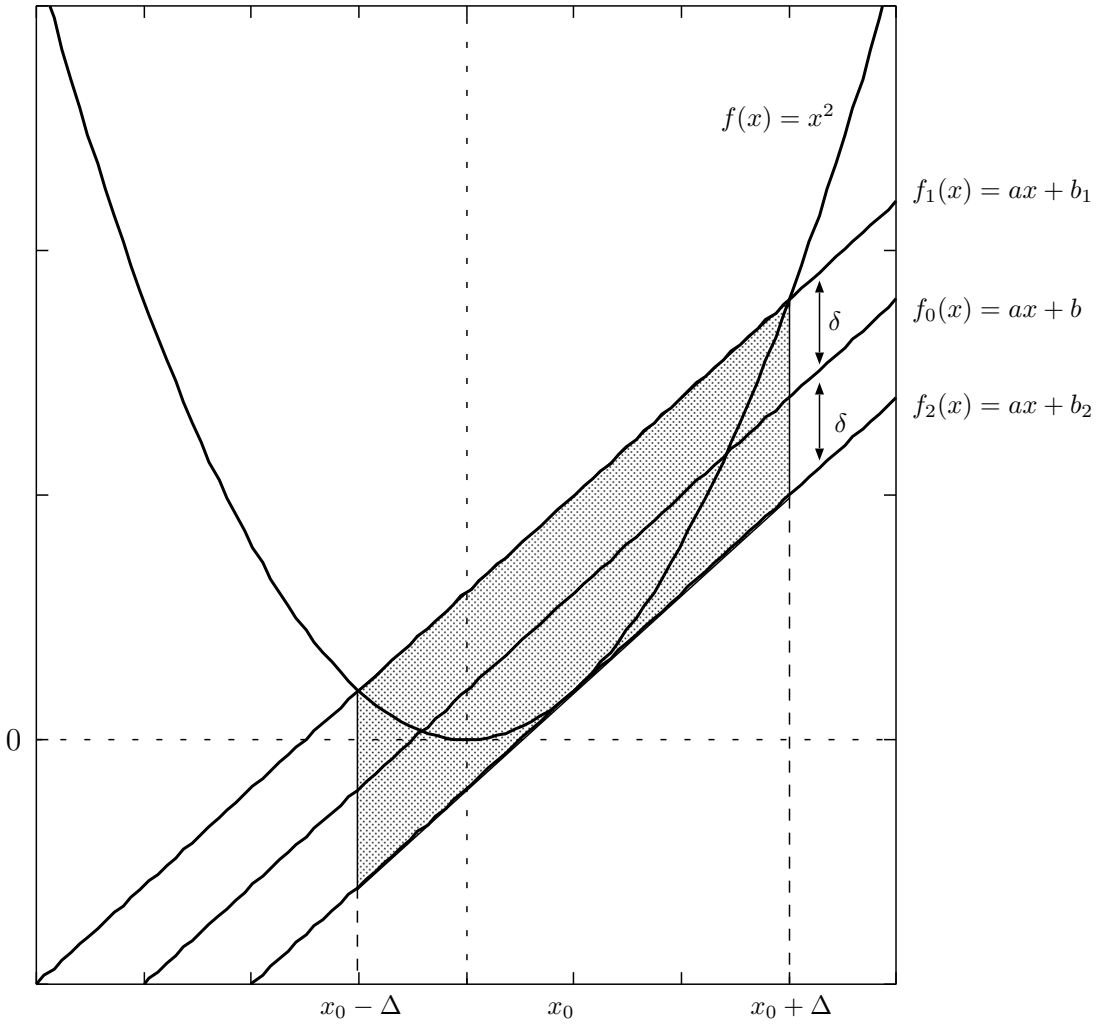


図 4.1: x^2 の包含-式 (4.4) により Δ を求める．このとき，$a = 2x_0, b_1 = -{x_0}^2 + \Delta^2, b_2 = -{x_0}^2$ であり，$b = \frac{1}{2}(b_1 + b_2) = -{x_0}^2 + \frac{1}{2}\Delta^2, \delta = \frac{1}{2}\Delta^2$ が得られる．

区間演算の場合

$$\begin{aligned} 2 \times [-2,3] - [-2,3] &= [-4,6] - [-2,3] \\ &= [-4-3, 6-(-2)] \\ &= [-7,8] \end{aligned}$$

アフィン演算の場合 $[-2,3] = 0.5 + 2.5\varepsilon$ なので，

$$\begin{aligned} 2 \times (0.5 + 2.5\varepsilon) - (0.5 + 2.5\varepsilon) &= (1.0 + 5.0\varepsilon) - (0.5 + 2.5\varepsilon) \\ &= (0.5 + 2.5\varepsilon) \\ &\to [-2,3] \end{aligned}$$

例題 4.2 $f(x) = x^2 - 2x$ を区間 $I = [0.9, 1.1]$ で評価する．真の評価は $f(I) = [-1, -0.99]$ となる．

区間演算の場合

$$\begin{aligned} [0.9,1.1]^2 - 2 \times [0.9,1.1] &= [0.81,1.21] - [1.8,2.2] \\ &= [0.81-2.2, 1.21-1.8] \\ &= [-1.39, -0.59] \end{aligned}$$

平均値形式を用いた場合 $c = \mathrm{mid}(I) = 1.0$ となり，

$$\begin{aligned} f(c) + f'(I)(I-c) &= -1 + (2 \times [0.9,1.1] - 2) \times ([0.9,1.1] - 1.0) \\ &= -1 + [-0.2, 0.2] \times [-0.1, 0.1] \\ &= [-1.02, -0.98] \end{aligned}$$

アフィン演算の場合 $[0.9,1.1] = 1.0 + 0.1\varepsilon_1$，また，$I$ における x^2 の近似 $ax + b$ は $a = 2, b = -0.995$ で得られ，このとき $\delta = 0.005$ となる．アフィン演算形式 $x = 1.0 + 0.1\varepsilon_1$ に対して

$$\begin{aligned} x^2 - 2x &= (2 \times (1.0 + 0.1\varepsilon_1) - 0.995 + 0.005\varepsilon_2) - 2 \times (1.0 + 0.1\varepsilon_1) \\ &= -0.995 + 0.005\varepsilon_2 \\ &\to [-1, -0.99] \end{aligned}$$

例題 4.3 $f(x) = x^2 + 2x$ を区間 $I = [0, 2]$ で評価する．真の評価は $f(I) = [0, 8]$ となる．

区間演算の場合

$$\begin{aligned}[0,2]^2 + 2 \times [0,2] &= [0,4] + [0,4] \\ &= [0,8]\end{aligned}$$

アフィン演算の場合 $[0,2] = 1 + \varepsilon_1$，また，$I$ における x^2 の近似 $ax + b$ は $a = 2, b = -0.5$ で得られ，このとき $\delta = 0.5$ となる．アフィン形式 $x = 1 + \varepsilon_1$ に対して

$$\begin{aligned}x^2 + 2x &= (2 \times (1 + \varepsilon_1) - 0.5 + 0.5\varepsilon_2) + 2 \times (1 + \varepsilon_1) \\ &= 3.5 + 4\varepsilon_1 + 0.5\varepsilon_2 \\ &\rightarrow [-1, 8]\end{aligned}$$

例題 4.1 においては $2x$ と x の自明な相関関係を，例題 4.2 においては x^2 と $2x$ という I において傾きの近い関数の相関関係を考慮しているために通常の区間演算より優れた評価が得られている．

例題 4.3 では得られた評価は通常の区間演算より悪いように見える．しかしアフィン演算による評価では変数間の相関関係が考慮されており，得られた直方体領域のすべての値をとるわけではない．すなわち，$I = [0,2]$ において $f(I) = [-1,8]$ という評価が得られているが，ダミー変数 $\varepsilon_1, \varepsilon_2$ の変動により，例えば直方体（ここでは長方形）領域内の $(0,8)$ や $(2,0)$ といった点はアフィン演算による評価で得られた領域内にはない．

これはアフィン多項式を直方体領域しか表現できない区間表示に変換した（例題では $\rightarrow$ で示されている部分）ことが原因であり，アフィン演算では特に必要ない限り計算結果はアフィン演算形式のままにしておくのがよいとされる（図 4.2）．

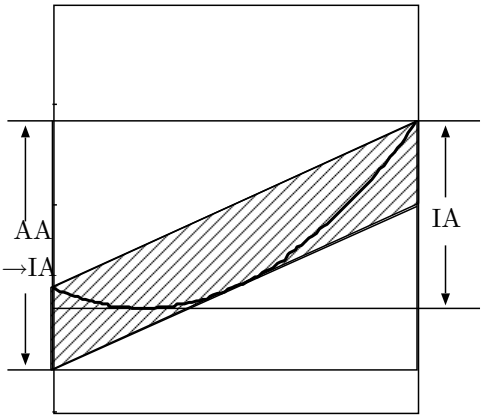


図 4.2: アフィン演算形式の区間演算への還元

4.4 おわりに

本章ではアフィン演算について，その方法と有効性を通常の区間演算との比較から簡単に述べた．何度もいうようにアフィン演算では変数間の相関関係が考慮され演算が行われ，非線形演算は線形な近似に誤差項を付加した領域をもってその結果とする．

第5章　アフィン演算における丸め誤差

5.1 はじめに

現在, 多くのパソコンやワークステーションではIEEE標準754という浮動小数点数システムを持ち, これらの計算機を援用して数値計算を行う場合は計算の各過程における丸めの向きを制御することができる．そのため, 通常の区間演算の場合は, このような計算機に対して丸め誤差まで考慮したプログラムを容易に実装できるという利点がある.

一方，アフィン演算は実数の表現が [下端 , 上端] という通常の区間演算のような形式をとらないため，変量の初期化や計算の各過程でアフィン多項式の係数に混入する計算機の丸め誤差をどのように反映させるかという問題点を抱えている.

5.2 機械区間演算

本節では計算機上で区間演算を展開するための手法について述べる．以下，IEEE標準754に基づく浮動小数点数システムの場合を考える．区間の両端が浮動小数点数で与えられたものを機械区間というが, 計算機上で区間演算を展開するためにはまず実区間を機械区間に置き換える必要がある．ただし，その際も数学的に正しい解を常に含んでいなければならないことに注意する．実空間 $[a,b](a,b \in \mathbf{R})$ は [(a の下向き丸め),(b の上向き丸め)] として機械区間表示される.
例) 実区間 $[1/7, 1/3]$ はC++で

```
volatile double a=1.,b=7.,c=3.;
down(); l=a/b;
up(); u=a/c;
```

のように ($1/7$ の下向き丸め)l と ($1/3$ の上向き丸め)u を計算することができ,
[0.14285714285714284921($< 1/7$),($1/3 <$)0.3333333333333333]
のような機械区間に置き換えられる.

5.3 実数のアフィン形式による表示

a, b を浮動小数点数とする．$[a,b]$ がアフィン形式 $c + r\varepsilon$ に変換される時，c と r は以下のようにして決められる．図1は a, b, c, r の関係を示し

ている.c^* は $(a+b)/2$ の真の値である．なお，この初期化は下の三つの演算法にも適用される．

```
up();
c=(a+b)/2;
r=c-a;
```

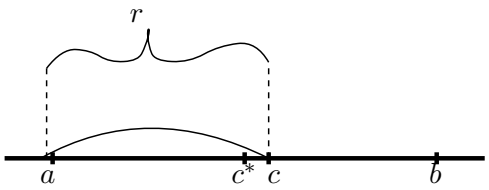


図 5.1: a, b, c, r の関係

以後は

```
up();
c=(a+b)/2;
r=c-a;
```

を

```
c=up((a+b)/2);
r=up(c-a);
```

として記述していく．

5.4　丸め誤差を評価するの方法

本論文ではアフィン演算における丸め誤差の取り扱いに関して，以下の方法を実装した．

この方法では，変数 x, y をそれぞれ

$$x = x_0 + x_1\varepsilon_r, \quad y = y_0 + y_2\varepsilon_r \tag{5.1}$$

と初期化し，$z = x + y$ を

$$z = z_0 + (|x_1| + |y_2| + |z_r|)\varepsilon_r \tag{5.2}$$

で評価する．ただし，

$$\begin{aligned} \texttt{down}(); \quad & \underline{\texttt{z}_0} = \texttt{x}_0 + \texttt{y}_0; \\ \texttt{up}(); \quad & \overline{\texttt{z}_0} = \texttt{x}_0 + \texttt{y}_0; \\ & \texttt{z}_0 = (\underline{\texttt{z}_0} + \overline{\texttt{z}_0})/2; \\ & \texttt{z}_\texttt{r} = \texttt{z}_0 - \underline{\texttt{z}_0}; \end{aligned} \tag{5.3}$$

丸め誤差に関する dummy 変数 ε_r の係数は常に絶対値和をとる点に注意する．(たとえば，$x - y$ の誤差項は $(|x_1| + |y_2| + |z'_r|)\varepsilon_r$)．また，$z^2$ のような非線形演算の場合，$z^2 = w_0 + w_1\varepsilon_1$ のように，追加する dummy 変数 ε_1 の項に丸め誤差を足し込む．z^2 を計算する過程で生じる丸め誤差は，ε_1 が打ち消される場合に限って打ち消すことができるのでこのようにしてよい．

5.4.1　アフィン形式

$$x_i = \texttt{up}(\frac{\underline{x_i} + \overline{x_i}}{2}), R_x = \texttt{up}(\sum_{i=0}^{n} \texttt{up}(\frac{\overline{x_i} - \underline{x_i}}{2})) \tag{5.4}$$

とすると

$$x = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + \cdots + x_n\varepsilon_n + R_x\varepsilon_r \tag{5.5}$$

5.4.2 通常の区間表示への還元

$$[\mathtt{down}(x_0 - \mathtt{up}(\sum_{i=1}^{n} |x_i|) - R_x)), \mathtt{up}(x_0 + \mathtt{up}(\sum_{i=1}^{n} |x_i|) + R_x))] \tag{5.6}$$

5.4.2.1 線形計算

$$\underline{p_i} = \mathtt{down}(x_i + y_i) \tag{5.7}$$
$$\overline{p_i} = \mathtt{up}(x_i + y_i) \tag{5.8}$$

とすると

$$\begin{aligned} x + y &= \mathtt{up}(\frac{\underline{p_0} + \overline{p_0}}{2}) + \mathtt{up}(\frac{\underline{p_1} + \overline{p_1}}{2})\varepsilon_1 + \cdots + \mathtt{up}(\frac{\underline{p_n} + \overline{p_n}}{2})\varepsilon_n \\ &+ \mathtt{up}(\sum_{i=0}^{n} \mathtt{up}(\frac{\overline{p_i} - \underline{p_i}}{2}) + R_x + R_y)\varepsilon_r \end{aligned} \tag{5.9}$$

又,

$$\underline{q_i} = \mathtt{down}(x_i - y_i) \tag{5.10}$$
$$\overline{q_i} = \mathtt{up}(x_i - y_i) \tag{5.11}$$

とすると

$$\begin{aligned} x - y &= \mathtt{up}(\frac{\underline{q_0} + \overline{q_0}}{2}) + \mathtt{up}(\frac{\underline{q_1} + \overline{q_1}}{2})\varepsilon_1 + \cdots + \mathtt{up}(\frac{\underline{q_n} + \overline{q_n}}{2})\varepsilon_n \\ &+ \mathtt{up}(\sum_{i=0}^{n} \mathtt{up}(\frac{\overline{q_i} - \underline{q_i}}{2}) + R_x + R_y)\varepsilon_r \end{aligned} \tag{5.12}$$

次に,

$$\underline{z_i} = \min(\mathtt{down}(x_i\underline{\alpha}), \mathtt{down}(x_i\overline{\alpha})) \tag{5.13}$$
$$\overline{z_i} = \max(\mathtt{up}(x_i\underline{\alpha}), \mathtt{up}(x_i\overline{\alpha})) \tag{5.14}$$

とすると

$$
\begin{aligned}
[\underline{\alpha},\overline{\alpha}]x &= \mathtt{up}(\frac{\underline{z_0}+\overline{z_0}}{2})+\mathtt{up}(\frac{\underline{z_1}+\overline{z_1}}{2})\varepsilon_1+\cdots+\mathtt{up}(\frac{\underline{z_n}+\overline{z_n}}{2})\varepsilon_n \\
&+ \mathtt{up}(\sum_{i=0}^{n}\mathtt{up}(\frac{\overline{z_i}-\underline{z_i}}{2})+\max(|\underline{\alpha}|R_x,|\overline{\alpha}|R_x))\varepsilon_r \qquad (5.15)
\end{aligned}
$$

5.4.3 非線形単項計算

$$
\begin{aligned}
\underline{z1_i} &= \min(\mathtt{down}(x_i\underline{a}),\mathtt{down}(x_i\overline{a})) \qquad (5.16)\\
\overline{z1_i} &= \max(\mathtt{up}(x_i\underline{a}),\mathtt{up}(x_i\overline{a})) \qquad (5.17)\\
A &= \mathtt{up}(\frac{\underline{z1_0}+\overline{z1_0}}{2})+\mathtt{up}(\frac{\underline{b}+\overline{b}}{2}) \qquad (5.18)
\end{aligned}
$$

とすると

$$
\begin{aligned}
[\underline{a},\overline{a}]x+[\underline{b},\overline{b}]+\overline{\delta}\varepsilon_{n+1} &= \mathtt{up}(\frac{\mathtt{down}(A)+\mathtt{up}(A)}{2}) \\
&+ \mathtt{up}(\frac{\underline{z1_1}+\overline{z1_1}}{2})\varepsilon_1 \\
&+ \cdots+\mathtt{up}(\frac{\underline{z1_n}+\overline{z1_n}}{2})\varepsilon_n \\
&+ \mathtt{up}(\sum_{i=0}^{n}\mathtt{up}(\frac{\overline{z1_i}-\underline{z1_i}}{2}) \\
&+ \max(\mathtt{up}(|\underline{a}|R_x),\mathtt{up}(|\overline{a}|R_x)) \\
&+ \mathtt{up}(\frac{\overline{b}-\underline{b}}{2}) \\
&+ \overline{\delta}+\mathtt{up}(\frac{\mathtt{up}(A)-\mathtt{down}(A)}{2}))\varepsilon_{n+1} \qquad (5.19)
\end{aligned}
$$

これを $f(x)$ の包含とする.

5.4.4 非線形二項計算

$$
\underline{z2_i} = \min(\mathtt{down}(y_i\underline{b}),\mathtt{down}(y_i\overline{b})) \qquad (5.20)
$$

$$\overline{z2_i} = \max(\mathtt{up}(y_i\underline{b}), \mathtt{up}(y_i\overline{b})) \tag{5.21}$$

$$P = \mathtt{up}(\frac{\underline{z1_0} + \overline{z1_0}}{2}) + \mathtt{up}(\frac{\underline{z2_0} + \overline{z2_0}}{2}) + \mathtt{up}(\frac{\underline{c} + \overline{c}}{2}) \tag{5.22}$$

$$Q_i = \mathtt{up}(\frac{\underline{z1_i} + \overline{z1_i}}{2}) + \mathtt{up}(\frac{\underline{z2_i} + \overline{z2_i}}{2}) \tag{5.23}$$

とすると

$$\begin{aligned}
[\underline{a}, \overline{a}]x + [\underline{b}, \overline{b}]y + [\underline{c}, \overline{c}] + \overline{\delta}\varepsilon_{n+1} &= \mathtt{up}(\frac{\mathtt{down}(P) + \mathtt{up}(P)}{2}) \\
&+ \mathtt{up}(\frac{\mathtt{down}(Q_1) + \mathtt{up}(Q_1)}{2})\varepsilon_1 \\
&+ \cdots + \mathtt{up}(\frac{\mathtt{down}(Q_n) + \mathtt{up}(Q_n)}{2})\varepsilon_n \\
&+ \mathtt{up}(\mathtt{up}(\frac{\mathtt{up}(P) - \mathtt{down}(P)}{2}) \\
&+ \sum_{i=1}^{n} \mathtt{up}(\frac{\mathtt{up}(Q_i) - \mathtt{down}(Q_i)}{2}) \\
&+ \mathtt{up}(\sum_{i=0}^{n} \mathtt{up}(\frac{\overline{z1_i} - \underline{z1_i}}{2}) + \max(\mathtt{up}(|\underline{a}|R_x), \mathtt{up}(|\overline{a}|R_x)) \\
&+ \mathtt{up}(\sum_{i=0}^{n} \mathtt{up}(\frac{\overline{z2_i} - \underline{z2_i}}{2}) + \max(\mathtt{up}(|\underline{b}|R_y), \mathtt{up}(|\overline{b}|R_y)) \\
&+ \mathtt{up}(\frac{\overline{c} - \underline{c}}{2}) + \overline{\delta})\varepsilon_{n+1}
\end{aligned} \tag{5.24}$$

これを $f(x, y)$ の包含とする.

5.5 おわりに

本章では，丸め誤差を考慮したアフィン演算の方法を提案した.

第6章　ベキ級数演算

6.1 はじめに

本章では，次のような正規形の連立一階常微分方程式に対する精度保証付き数値計算について述べる．

$$\frac{dx(t)}{dt} = f(x(t), t), \quad t \in [a, b], \tag{6.1}$$

ここで， $x(t)$ は n 次元のベクトル値関数である．

このような微分方程式を数値的に解くためには，定義域を $a = t_1 < t_2 < \cdots < t_m = b$ のように分割して離散化し， $x(t_i)$ の近似 x_i を未知数とする有限次元方程式に帰着させることがよく行われる．この過程でいわゆる離散化誤差が発生し，それを厳密に見積もることは極めて難しい．しかし，もし隣同士の値 x_i と x_{i+1} との関係を正確に記述することが出来れば，真の値 (微分方程式の軌道上の値) $x(t_i)$ に一致する解 x_i を持つような有限次元方程式が得られるはずである．すなわち，条件 $x(t_s) = v$ の下で正確な $x(t_e)$ の値をを与えるような関数 $\phi(v, t_s, t_e)$ を計算出来れば，その正確な解をもつ有限次元方程式を，

$$\begin{aligned} x_2 &= \phi(x_1, t_1, t_2) \\ x_3 &= \phi(x_2, t_2, t_3) \\ &\vdots \\ x_m &= \phi(x_{m-1}, t_{m-1}, t_m). \end{aligned} \tag{6.2}$$

のように書くことが出来る．この方程式は $n \times m$ 個の未知数と $n \times (m-1)$ 個の方程式を持つ．従って， n 個の境界条件を付加することによって，解けることが期待出来る．

以上により，例えば Krawczyk の方法 [5, 6] などによって有限次元方程式の精度保証付きの解を計算すれば，離散的な m 個の点における与えられた微分方程式の精度保証付きの解を得ることが出来る．

本章では，上で述べた隣同士の点の正確な関係を表す関数 $\phi(v, t_s, t_e)$ の計算方法を示す．この方法は，Lohner の方法 [7] の拡張と見なせる．本章では，必要な場面では区間演算が使用されることを注意しておく．

6.2 ベキ級数演算 (Power Series Arithmetic)

本節では第 3 節での議論の準備として，ベキ級数演算 (Power Series Arithmetic, 以下 PSA と略す) を定義する．

6.2.1 Type-I Power Series Arithmetic

Type-I PSA では，order-n のベキ級数を扱い，それより高次の項は切り捨てる．$[a_0, a_1, a_2, \cdots, a_n]$ という表記で，ベキ級数

$$a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n \quad (+O(t^{n+1})) \tag{6.3}$$

を表すことにする．

6.2.1.1 加算，減算

加算，減算は，次のように行われる．

$$[a_0, \cdots, a_n] \pm [b_0, \cdots, b_n] = [a_0 \pm b_0, \cdots, a_n \pm b_n] \tag{6.4}$$

6.2.1.2 乗算

乗算は，次のように行われる．

$$[a_0, \cdots, a_n] \times [b_0, \cdots, b_n] = [c_0, \cdots, c_n] \tag{6.5}$$

$$c_k = \textstyle\sum_{i=0}^{k} a_i b_{k-i} \tag{6.6}$$

6.2.1.3 関数

関数の適用は次のように行われる．

$$f([a_0, \cdots, a_n]) = f(a_0) + \sum_{i=1}^{n} \frac{1}{i!} f^{(i)}(a_0)[0, a_1, \cdots, a_n]^i \tag{6.7}$$

上式中に現れる加算及び乗算は，式 (6.4), 式 (6.5) によって行われる．除算は逆数関数と乗算の組み合わせによって行われる $(x/y = x \times (1/y))$．

Type-I PSA は，全く打ち切りの無い演算を行った場合の最初の $(n+1)$ 項を保存するように定義された演算である．このような演算方式は，[6, 8] などでも議論されている．また，Mathematica などのいくつかの数学ソフトウェアは，このような演算機構を備えている．

6.2.2 Type-II Power Series Arithmetic

Type-II PSA も Type-I と同様 order-n のベキ級数を取り扱う．Type-II PSA では，取り扱う級数の定義域 (t の変域) を，$[0, d]$ のように定める必要がある．. $[a_0, \cdots, a_n]$ という表記は同じく

$$a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n, \tag{6.8}$$

を表すが，最後の項の係数 a_n は一般に区間となり，$[a_0, \cdots, a_n]$ は $f(t) \in a_0 + \cdots + a_n t^n$ in all t を満たすような $[0, d]$ 上で定義された連続関数の集合を表す．これは，Kaucher, Miranker[9, 10] によって提案された interval functoid の一種であると考えられる．

6.2.2.1 加算，減算

算及び減算は Type-I PSA と同様となる．

$$[a_0, \cdots, a_n] \pm [b_0, \cdots, b_n] = [a_0 \pm b_0, \cdots, a_n \pm b_n]. \tag{6.9}$$

6.2.2.2 乗算

乗算は，次のように行われる．

$$\begin{aligned} [a_0, \cdots, a_n] \times [b_0, \cdots, b_n] &= [c_0, \cdots, c_{2n}], \\ c_k &= \textstyle\sum_{i=\max(0,k-n)}^{\min(k,n)} a_i b_{k-i}. \end{aligned} \tag{6.10}$$

をと計算し，$[a_0, \cdots, a_n]$ と $[b_0, \cdots, b_n]$ を打ち切りなしで乗算した後，$[c_0, \cdots, c_{2n}]$ の次数を n に減らす．

6.2.2.3 減次

$A = [a_0, \cdots, a_m]$ をベキ級数，n を $n < m$ を満たす自然数とする．このとき A の次数 n への減次は次の $B = [b_0, \cdots, b_n]$ で定義される:

$$b_i = a_i \quad (i \leq 0 \leq n-1) \tag{6.11}$$

$$b_n = a_n + \sum_{i=n+1}^{m} a_i [0, d]^{i-n} \tag{6.12}$$

これは，項 $a_i t^i$ を $a_i t^{i-n} t^n$ と変形し，t^{i-n} を $[0,d]^{i-n}$ で置き換えることによって，高次の剰余を t^n の係数に加えている．よって，乗算の結果 C は丸めなしの乗算によって得られる可能性のある結果を全て含んでいる．

なお，区間演算の性質によって，多項式の評価の結果は計算順序によって変化する．式 (6.12) の計算結果は，最善の区間

$$a_n + \sum_{i=n+1}^{m} a_i t^{i-n} \mid t \in [0,d] \tag{6.13}$$

を含むが，一般にそれより半径の大きい区間になる．b_n の計算には，次のような Horner の方法

$$b_n = a_n + [0,d](a_{n+1} + [0,d](\cdots(a_{m-1} + a_m[0,d])\cdots) \tag{6.14}$$

を用いることを推奨する．

6.2.2.4 関数

関数は次のように適用される．

$$\begin{aligned} f([a_0,\cdots,a_n]) &= f(a_0) + \sum_{i=1}^{n-1} \frac{1}{i!} f^{(i)}(a_0)[0,a_1,\cdots,a_n]^i & (6.15)\\ &\quad + \frac{1}{n!} f^{(n)}\left(\sum_{i=0}^{n} a_i [0,d]^i\right)[0,a_1,\cdots,a_n]^n & (6.16) \end{aligned}$$

上のアルゴリズム中に現れる加算及び乗算は Type-II PSA によって行われる．この方法では Lagrange の剰余項を用いている．乗算と同様に，このベキ級数の多項式も Horner の方法で計算するとよい．除算 x/y は Type-I PSA と同様に $x \times (1/y)$ を計算する．いうまでもなく，ここでの乗算と逆数計算は Type-II PSA で行われる．

6.3 $\phi(v,t_s,t_e)$ の計算方法

$\phi : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^n$ は次の初期値問題の保証付き解を計算することによって計算出来る．

$$\begin{aligned} \frac{dx(t)}{dt} &= f(x(t),t) & (6.17)\\ x(t_s) &= v & (6.18)\\ t &\in [t_s,t_e] & (6.19) \end{aligned}$$

$x(t)$ が正確に計算出来れば， $\phi(x, t_s, t_e)$ は $x(t_e)$ で得られる．

本節では，Picard の反復法に基づき， $x(t)$ の包み込みを得るアルゴリズムを与える．このアルゴリズムは，式 (6.17) を同値な不動点形式

$$x(t) = v + \int_{t_s}^{t} f(x(s), s)ds \tag{6.20}$$

に変換し，定数関数 v を初期値とする反復によって近似解を得，真の解の存在を Schauder の不動点定理によって示すものである．

アルゴリズム 6.1 $t_e > t_s$, $\Delta = t_e - t_s$ とし， *Type-II PSA* の定義域を $[0, \Delta]$ とする．

Step-1 ベキ級数ベクトル X を

$$X = \begin{pmatrix} [v_1] \\ \vdots \\ [v_n] \end{pmatrix} \tag{6.21}$$

のように初期化し， $m = 0$ とする．

Step-2(近似解の生成) *Step-2(1)–Step-2(3)* を適当な回数繰り返す．

Step-2(1) ベキ級数 T を $(t_s + t)$ を m 次で打ち切ったものとする．

$$T = \begin{cases} [t_s] & (m = 0) \\ [t_s, 1] & (m = 1) \\ [t_s, 1, 0, \cdots, 0] & (m > 2) \end{cases} \tag{6.22}$$

Step-2(2) $X = f(X, T)$ を *Type-I PSA* で計算する．$X = v + \int_0^t X dt$ を計算する．これらによって， X の次数は m から $m + 1$ となる．

Step-2(3) $m = m + 1$

Step-3 $T = [t_s, 1, 0, \cdots, 0]$ *(order m)* とする．

Step-4 $Y = f(X, T)$ を *Type-II PSA* で計算し，$Y = v + \int_0^t Y dt$ を計算し，*6.2.2.3*節によって Y の次数を $m + 1$ から m に減次する．

Step-5

$$r = \max_{1 \le i \le n} |Y_i^{(m)} - X_i^{(m)}| \tag{6.23}$$

を計算する．ここで添字 $^{(k)}$ は t^k の係数を表す．

Step-6 $1 \leq i \leq n$ に対して，$X_i^{(m)} = X_i^{(m)} + [-2r, 2r]$ を行う．

Step-7 $Y = f(X, T)$ を *Type-II PSA* で計算し，$Y = v + \int_0^t Y dt$ を計算し，*6.2.2.3*節によって Y の次数を $m+1$ から m に減次する．

Step-8(存在検証) 全ての i について $Y_i^{(m)} \subset X_i^{(m)}$ が成立すれば，Y に式 *(6.17)* の解が存在することが *Schauder* の不動点定理により検証される．

Step-9(解の改良) *Step-9(1)–Step-9(2)* を $Y_i^{(m)}$ が十分小さくなるまで繰り返す．

Step-9(1) $X = f(Y, T)$ を *Type-II PSA* で計算し，$X = v + \int_0^t X dt$ を計算し，*6.2.2.3*節によって X の次数を $m+1$ から m に減次する．

Step-9(2) 全ての i について $Y_i^{(m)} = Y_i^{(m)} \cap X_i^{(m)}$ とする．

Step-10 $\phi(v, t_s, t_e)$ は

$$\phi(v, t_s, t_e) = Y(\Delta) = \begin{pmatrix} \sum_{i=0}^{m} Y_1^{(i)} \Delta^i \\ \vdots \\ \sum_{i=0}^{m} Y_n^{(i)} \Delta^i \end{pmatrix}. \tag{6.24}$$

で得られる．

なお，Step-8 において， $0 \leq k \leq m-1$ で $Y_i^{(k)} = X_i^{(k)}$ が常に成立している．よって最後の項だけで比較を行えばよい．

6.4 おわりに

本章では，連立常微分方程式に対する精度保証付き数値計算について述べた．

第7章　平均値形式を用いたベキ級数演算

7.1 はじめに

本章では，平均値形式を用いて，ベキ級数演算を行う方法について述べる．なお，この方法においては，以前に求めた Jacobi 行列を保存しておくことによって，区間幅の増大を抑えている．

7.2 $\phi_v(v, t_s, t_e)$ の計算方法

ベキ級数演算に平均値形式を用いるためには，$\phi(c, t_s, t_e)$，及び $\phi(v, t_s, t_e)$ に対する正確な Jacobi 行列 $\phi_v(v, t_s, t_e)$ が必要となる．$\phi_v(v, t_s, t_e)$ は次のようにして得ることが出来る．

$$\begin{aligned}
\frac{dx(t)}{dt} &= f(x(t), t) && (7.1)\\
\frac{dy(t)}{dt} &= f_x(x(t), t)y(t) && (7.2)\\
x(t_s) &= v && (7.3)\\
y(t_s) &= I \quad (n \times n \text{ identity matrix}) && (7.4)\\
t &\in [t_s, t_e] && (7.5)
\end{aligned}$$

という連立初期値問題を考える．これを第 6 章の**アルゴリズム 6.1** で解けば，$\phi_v(v, t_s, t_e)$ は $y(t_e)$ で得ることが出来る．

なお，上で述べた方法は自動微分の手法 [8] と深い関連がある．具体的には，初期値 v に関する微分を同時に計算しながら第 6 章の**アルゴリズム 6.1** を行うことによって，上の行列値関数 $y(t)$ の (i, j) 要素は “$x_i(t)$ の v の第 j 要素に関する微分” として得られる．

という連立初期値問題を考える．これを第 6 章の**アルゴリズム 6.1** で解けば，$\phi(c, t_s, t_e)$ は $c(t_e)$ で得ることが出来る．

7.3 x_n の計算方法

x_n は以下のように計算して求める．

$$\begin{aligned}
x_0 &= v && (7.6)\\
c_n &= \begin{cases} mid(x_0) \ \ (n = 0) \\ mid(\phi(c_{n-1}, t_{n-1}, t_n)) \ \ (n \geq 1) \end{cases} && (7.7)
\end{aligned}$$

$$r_n = \begin{cases} x_0 - c_0 \quad (n=0) \\ \phi(c_{n-1}, t_{n-1}, t_n) - mid(\phi(c_{n-1}, t_{n-1}, t_n)) \quad (n \geq 1) \end{cases} \tag{7.8}$$

とすると

$$\begin{aligned} x_1 &= \phi(c_0, t_0, t_1) + \phi_v(x_0, t_0, t_1)(x_0 - c_0) && (7.9) \\ &= c_1 + r_1 + \phi_v(x_0, t_0, t_1) r_0 && (7.10) \\ x_2 &= \phi(c_1, t_1, t_2) + \phi_v(x_1, t_1, t_2)(x_1 - c_1) && (7.11) \\ &= c_2 + r_2 + \phi_v(x_1, t_1, t_2) r_1 + \phi_v(x_1, t_1, t_2)\phi_v(x_0, t_0, t_1) r_0 && (7.12) \\ &\vdots && (7.13) \end{aligned}$$

となり x_n は以下のように求められる.

$$x_n = c_n + \sum_{i=0}^{n-1} \left(\left(\prod_{j=i}^{n-1} \phi_v(x_j, t_j, t_{j+1}) \right) \times r_i \right) \tag{7.14}$$

7.4 おわりに

本章では，平均値形式を用いて，ベキ級数演算を行う方法について述べた.

第8章　アフィン演算を用いたベキ級数演算

8.1 はじめに

本章では，本論文で提案する，アフィン演算を用いて，ベキ級数演算を行う方法について述べる．

8.2 $\phi(v, t_s, t_e)$ の計算方法

以下にアフィン演算を用いたベキ級数演算の方法について述べる．なお，$af()$ はアフィン形式で評価し，$in()$ は区間で評価するものとする．また，$af([])$ はベキ級数のそれぞれの項をアフィン形式で評価することを表すものとする．

アルゴリズム 8.1 $t_e > t_s$, $\Delta = t_e - t_s$ とし，*Type-II PSA* の定義域を $[0, \Delta]$ とする．

Step-1 ベキ級数ベクトル X を

$$X = \begin{pmatrix} af([v_1]) \\ af([v_2]) \\ \vdots \\ af([v_n]) \end{pmatrix} \tag{8.1}$$

のように初期化し， $m = 0$ とする．

Step-2(近似解の生成) *Step-2(1)–Step-2(3)* を適当な回数繰り返す．

Step-2(1) ベキ級数 T を $(t_s + t)$ を m 次で打ち切ったものとする．

$$T = \begin{cases} af([t_s]) & (m = 0) \\ af([t_s, 1]) & (m = 1) \\ af([t_s, 1, 0, \cdots, 0]) & (m > 2) \end{cases} \tag{8.2}$$

Step-2(2) $X = f(X, T)$ を *Type-I PSA* で計算する．$X = v + \int_0^t X dt$ を計算する．これらによって， X の次数は m から $m + 1$ となる．

Step-2(3) $m = m + 1$

Step-3 $T = af([t_s, 1, 0, \cdots, 0])$ *(order m)* とする．

Step-4 $Y = f(X,T)$ を *Type-II PSA* で計算し，$Y = v + \int_0^t Y dt$ を計算し，*6.2.2.3*節によって Y の次数を $m+1$ から m に減次する．

Step-5

$$r = \max_{1 \le i \le n} |in(Y_i^{(m)} - X_i^{(m)})| \tag{8.3}$$

を計算する．ここで添字 $^{(k)}$ は t^k の係数を表す．

Step-6 $1 \le i \le n$ に対して，$X_i^{(m)} = af(in(X_i^{(m)}) + [-2r, 2r])$ を行う．なお，ここから始まる存在検証について，詳しくは *8.3*節で説明する．

Step-7 $Y = f(X,T)$ を *Type-II PSA* で計算し，$Y = v + \int_0^t Y dt$ を計算し，*6.2.2.3*節によって Y の次数を $m+1$ から m に減次する．

Step-8(存在検証) 全ての i について $in(Y_i^{(m)}) \subset X_i^{(m)}$ が成立すれば，Y に式 *(6.17)* の解が存在することが *Schauder* の不動点定理により検証される．

Step-9 $\phi(v, t_s, t_e)$ は

$$\phi(v, t_s, t_e) = Y(\Delta) = \begin{pmatrix} \sum_{i=0}^m Y_1^{(i)} \Delta^i \\ \vdots \\ \sum_{i=0}^m Y_n^{(i)} \Delta^i \end{pmatrix}. \tag{8.4}$$

で得られる．

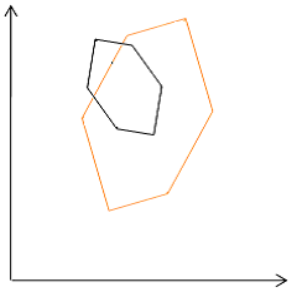
なお，Step-8 において， $0 \le k \le m-1$ で $Y_i^{(k)} = X_i^{(k)}$ が常に成立している．よって最後の項だけで比較を行えばよい．

8.3 解の存在検証について

アフィンベキ級数演算における解の存在検証を行う際には，アフィン変数同士の包含関係，つまり n 次凸多面体同士の包含関係が成立するか否かを判別する必要がある．

例えば $n = 2$ の場合には，アフィン変数 $X = (x_0, x_1)$ と $X' = (x_0', x_1')$ の包含関係 $X' \subset X$ は，図 8.1 のような図形同士の包含関係を判別するということになる．

図 8.1: X' と X の包含関係



しかし，この判定を高速に行うのは容易ではない．そのために本論文では，敢えて x_0，x_1 の相関性をなくし，独立した区間と見なすことによって，包含関係の判別を高速に行えるようにする．具体的には以下のようにする．

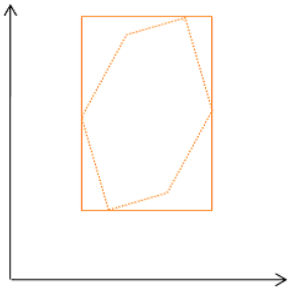
$$x_0 = a_0 + a_1\varepsilon_1 + a_2\varepsilon_2 \tag{8.5}$$
$$x_1 = b_0 + b_1\varepsilon_1 + b_2\varepsilon_2 \tag{8.6}$$

とすれば

$$x_0 = a_0 + (|a_1| + |a_2|)\varepsilon_3 \tag{8.7}$$
$$x_1 = b_0 + (|b_1| + |b_2|)\varepsilon_4 \tag{8.8}$$

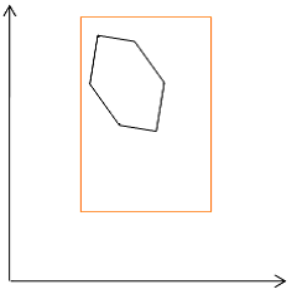
と変換すればよい．これはちょうど，アフィン変数を一度，区間形式で評価し，さらに再度アフィン形式で評価したものと同じとなる．また，この変換は図 8.2 のようになっている．

図 8.2: X の変換



このような変換を行うことで，図 8.3 のような包含関係を判別すれば良くなり，高速に解の存在検証を行うことが出来る．また，この解の存在検証において，X が膨らんでしまっているが，この区間の増大分はベキ級数演算の特性によって，$\Delta \leq 1$，とりわけ $\Delta \fallingdotseq 0$ の場合に非常に小さい増大とすることが出来る．

図 8.3: X の変換



8.4 $\phi(v, t_s, t_e)$ の改良

前述の方法で $\phi(v, t_s, t_e)$ を求めた場合，イプシロンの数が多くなり過ぎて，膨大なメモリ領域と膨大な計算時間が必要となってしまう．よって，この計算方法を実際に用いるためには，イプシロンの数を減ら必要がある．アフィン変数は最良の場合には，n 個の一次独立なベクトルにまとめることができるはずであるが，それを求めるのもまた，膨大な時間がかかってしまう．

そのために，本論文では，各ステップ毎において $\phi(v, t_s, t_e)$ を求めた際に発生したイプシロンを n 次の独立したイプシロンと見立てて，n 個のイプシロンにまとめることにする．具体的には以下のように行う．

$$X_0 = \begin{cases} a_{00} \\ b_{00} \end{cases} \tag{8.9}$$

$$X_1 = \phi(X_0, t_0, t_1) \tag{8.10}$$

$$X_1 = \begin{cases} a_{10} + a_{11}\varepsilon_1 + a_{12}\varepsilon_2 + a_{13}\varepsilon_3 \\ b_{10} + b_{11}\varepsilon_1 + b_{12}\varepsilon_2 + b_{13}\varepsilon_3 \end{cases} \tag{8.11}$$

となった場合に，X_1 を以下のように変換する．

$$X_1 = \begin{cases} a_{10} + (|a_{11}| + |a_{12}| + |a_{13}|)\varepsilon_1 \\ b_{10} + (|b_{11}| + |b_{12}| + |b_{13}|)\varepsilon_2 \end{cases} \tag{8.12}$$

そして次のステップにおいて

$$X_2 = \phi(X_1, t_1, t_2) \tag{8.13}$$

$$X_2 = \begin{cases} a_{20} + a_{21}\varepsilon_1 + a_{22}\varepsilon_2 + a_{23}\varepsilon_3 + a_{24}\varepsilon_4 + a_{25}\varepsilon_5 \\ b_{20} + b_{21}\varepsilon_1 + b_{22}\varepsilon_2 + b_{23}\varepsilon_3 + b_{24}\varepsilon_4 + b_{25}\varepsilon_5 \end{cases} \tag{8.14}$$

となった場合に，X_2 を以下のように変換する．

$$X_2 = \begin{cases} a_{20} + a_{21}\varepsilon_1 + a_{22}\varepsilon_2 + (|a_{23}| + |a_{24}| + |a_{25}|)\varepsilon_3 \\ b_{20} + b_{21}\varepsilon_1 + b_{22}\varepsilon_2 + (|b_{23}| + |b_{24}| + |b_{25}|)\varepsilon_4 \end{cases} \tag{8.15}$$

第9章　数値例による性能検証

9.1 はじめに

本節ではアフィン演算を用いたベキ級数演算の有効性をいくつかの数値例により示す．まず 9.2 節では本数値例で使用する方法について説明する．次に 9.3 節で数値例，及びその実行結果を示す．

9.2 使用する方法

本章では 9.3 章で使用する方法は以下のとおりである．

方法 A
6 章の区間演算によるベキ級数演算を用いた，連立常微分方程式の解法．

方法 B
7 章の平均値形式を用いたベキ級数演算を用いた，連立常微分方程式の解法．

提案方法
8 章のアフィン演算を用いたベキ級数演算を用いた，連立常微分方程式の解法．

なお，検証は全て

- OS: Windows Vista
- CPU: Intel Core2Duo 2.4GHz
- メモリ: 2GB
- 開発環境: VisualC++2005, Boost 1.34

上で行う．

9.3 数値例

本節では数値例により**方法 A，方法 B，提案方法**の性能を比較する．

例題 1
以下のような常微分方程式の初期値問題を考える．

$$\frac{d^2x}{dt^2} = -x \tag{9.1}$$
$$x(t_s) = 0 \tag{9.2}$$

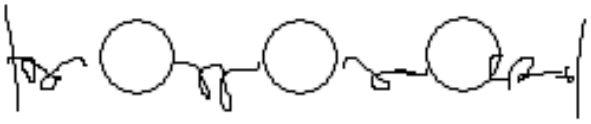
$$\frac{dx}{dt}(t_s) = 1 \tag{9.3}$$
$$t_s = 0 \tag{9.4}$$

これは以下のような一階連立常微分方程式となる．

$$\frac{dx_0}{dt} = x_1 \tag{9.5}$$
$$\frac{dx_1}{dt} = -x_0 \tag{9.6}$$
$$x_0(t_s) = 0 \tag{9.7}$$
$$x_1(t_s) = 1 \tag{9.8}$$
$$t_s = 0 \tag{9.9}$$

なお，この初期値問題において，$x_0(t)$ は $sin(t)$ となる．$\Delta = 0.25$ とし，反復させた際の解の区間幅を 表 9.1 に，実行時間を表 9.2 に示す．区間幅が 1 以上になった場合発散したと見なし，表では – としている．また，x_0 の区間の増大の様子を片対数グラフにしたものを図 9.1 に示す．

例題 2



図のような 3 つの質点を質量を無視できる 4 つのバネで連結している場合について考える．質点の質量を左から m_0，m_1，m_2 とし，バネの弾性定数を左から k_0，k_1，k_2，k_3 とする．各質点の釣り合いの位置からの変位を x_0，x_1，x_2 とすると，それぞれの質点の運動方程式は次のようになる．

$$m_0\frac{d^2x_0}{dt^2} = k_1x_2 - (k_0 + k_1)x_0 \tag{9.10}$$
$$m_1\frac{d^2x_2}{dt^2} = k_1x_0 + k_2x_4 - (k_1 + k_2)x_2 \tag{9.11}$$
$$m_2\frac{d^2x_4}{dt^2} = k_2x_2 - (k_2 + k_3)x_4 \tag{9.12}$$

これから，以下のような一階連立常微分方程式の初期値問題を考える．

$$\frac{dx_0}{dt} = x_1 \tag{9.13}$$
$$m_0\frac{dx_1}{dt} = k_1x_2 - (k_0+k_1)x_0 \tag{9.14}$$
$$\frac{dx_2}{dt} = x_3 \tag{9.15}$$
$$m_1\frac{dx_3}{dt} = k_1x_0 + k_2x_4 - (k_1+k_2)x_2 \tag{9.16}$$
$$\frac{dx_4}{dt} = x_5 \tag{9.17}$$
$$m_2\frac{dx_5}{dt} = k_2x_2 - (k_2+k_3)x_4 \tag{9.18}$$
$$x_0(t_s) = x_2(t_s) = x_4(t_s) = 1 \tag{9.19}$$
$$x_1(t_s) = x_3(t_s) = x_5(t_s) = 0 \tag{9.20}$$
$$t_s = 0 \tag{9.21}$$

$k_0 = k_1 = k_2 = k_3 = 2$，$m_0 = m_1 = m_2 = 1.5$，$\Delta = 0.25$ とし，反復させた際の解の区間幅を 表 9.3，9.4 及び 9.5 に，実行時間を表 9.6 に示す．区間幅が 1 以上になった場合発散したと見なし，表では – としている．また，x_0 の区間の増大の様子を片対数グラフにしたものを図 9.2 に示す．

例題 3

以下のような一階連立常微分方程式を考える．

$$\frac{dx_0}{dt} = x_0x_1 \tag{9.22}$$
$$\frac{dx_1}{dt} = x_2 \tag{9.23}$$
$$\frac{dx_2}{dt} = -x_1 \tag{9.24}$$
$$x_0(t_s) = x_1(t_s) = 1 \tag{9.25}$$
$$x_2(t_s) = 0 \tag{9.26}$$
$$t_s = 0 \tag{9.27}$$

なお，この初期値問題において，$x_1 = e^{sint}$, $x_2 = cost$ となる．$\Delta = 0.25$ とし，反復させた際の解の区間幅を 表 9.7，9.8 に，実行時間を表 9.9 に示す．区間幅が 1 以上になった場合発散したと見なし，表では – として

いる．また，x_0 の区間の増大の様子を片対数グラフにしたものを図 9.3 に示す

表 9.1: x_0 と x_1 の区間幅 (例題 1)

反復数	方法 A		方法 B		提案方法	
	x_0	x_1	x_0	x_1	x_0	x_1
1	2.78e-17	1.11e-16	2.78e-17	1.11e-16	2.78e-17	1.11e-16
50	8.73e-11	8.73e-11	5.61e-15	1.25e-14	4.87e-15	6.33e-15
100	2.34e-05	2.34e-05	1.23e-14	2.54e-14	9.69e-15	1.25e-14
150	-	-	1.86e-14	3.75e-14	1.47e-14	1.88e-14
200	-	-	1.10e-13	1.30e-13	1.99e-14	2.52e-14
250	-	-	1.49e-09	1.49e-09	2.57e-14	3.19e-14
300	-	-	2.67e-05	2.67e-05	3.16e-14	3.79e-14
350	-	-	4.77e-01	4.77e-01	3.67e-14	4.34e-14
400	-	-	-	-	4.27e-14	4.90e-14
450	-	-	-	-	4.90e-14	5.38e-14
500	-	-	-	-	5.56e-14	5.85e-14
550	-	-	-	-	6.21e-14	6.21e-14
600	-	-	-	-	6.81e-14	6.58e-14
650	-	-	-	-	7.52e-14	7.01e-14
700	-	-	-	-	8.25e-14	7.34e-14
750	-	-	-	-	9.00e-14	7.72e-14
800	-	-	-	-	9.74e-14	8.05e-14
850	-	-	-	-	1.06e-13	8.51e-14
900	-	-	-	-	1.14e-13	8.87e-14
950	-	-	-	-	1.21e-13	9.30e-14
1000	-	-	-	-	1.29e-13	9.76e-14

表 9.2: 実行時間 (例題 1)

反復数	方法 A	方法 B	提案方法
1	0.000	0.001	0.001
50	0.005	0.043	0.059
100	0.011	0.081	0.147

図 9.1: 区間幅の比較 (例題 1)

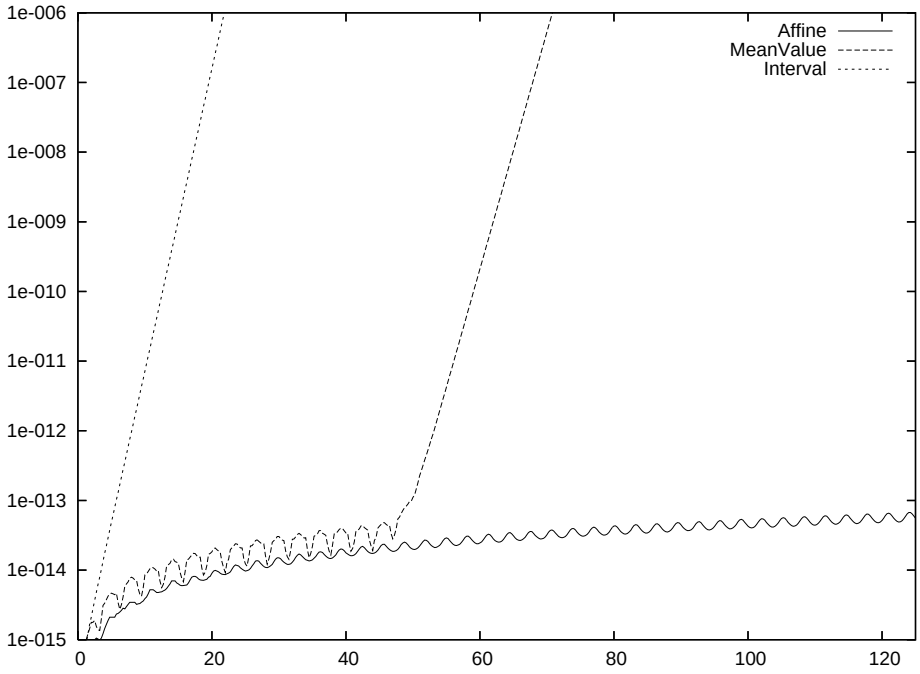


表 9.3: x_0 と x_1 の区間幅 (例題 2)

反復数	方法 A		方法 B		提案方法	
	x_0	x_1	x_0	x_1	x_0	x_1
1	1.11e-16	1.11e-16	1.11e-16	1.11e-16	1.11e-16	5.55e-17
50	3.62e-05	7.72e-05	9.42e-15	1.67e-14	9.78e-15	1.44e-14
100	-	-	1.66e-13	2.58e-13	2.36e-14	3.18e-14
150	-	-	7.37e-05	1.25e-04	2.92e-14	4.36e-14
200	-	-	-	-	4.65e-14	6.37e-14
250	-	-	-	-	4.95e-14	7.50e-14
300	-	-	-	-	6.96e-14	9.62e-14
350	-	-	-	-	6.96e-14	1.04e-13
400	-	-	-	-	9.00e-14	1.25e-13
450	-	-	-	-	8.97e-14	1.32e-13
500	-	-	-	-	1.12e-13	1.56e-13
550	-	-	-	-	1.12e-13	1.62e-13
600	-	-	-	-	1.34e-13	1.88e-13
650	-	-	-	-	1.34e-13	1.94e-13
700	-	-	-	-	1.54e-13	2.17e-13
750	-	-	-	-	1.59e-13	2.27e-13
800	-	-	-	-	1.73e-13	2.49e-13
850	-	-	-	-	1.84e-13	2.58e-13
900	-	-	-	-	1.92e-13	2.79e-13
950	-	-	-	-	2.08e-13	2.91e-13
1000	-	-	-	-	2.09e-13	3.08e-13

表 9.4: x_2 と x_3 の区間幅 (例題 2)

反復数	方法 A		方法 B		提案方法	
	x_2	x_3	x_2	x_3	x_2	x_3
1	1.11e-16	5.20e-18	1.11e-16	5.20e-18	1.11e-16	8.67e-18
50	5.11e-05	1.09e-04	1.11e-14	2.93e-14	1.14e-14	1.93e-14
100	-	-	1.91e-13	2.83e-13	2.30e-14	3.07e-14
150	-	-	8.75e-05	1.38e-04	3.49e-14	5.57e-14
200	-	-	-	-	4.67e-14	6.13e-14
250	-	-	-	-	6.05e-14	9.30e-14
300	-	-	-	-	7.01e-14	9.42e-14
350	-	-	-	-	8.42e-14	1.28e-13
400	-	-	-	-	9.29e-14	1.24e-13
450	-	-	-	-	1.08e-13	1.61e-13
500	-	-	-	-	1.16e-13	1.57e-13
550	-	-	-	-	1.33e-13	1.95e-13
600	-	-	-	-	1.41e-13	1.92e-13
650	-	-	-	-	1.58e-13	2.27e-13
700	-	-	-	-	1.64e-13	2.28e-13
750	-	-	-	-	1.83e-13	2.59e-13
800	-	-	-	-	1.88e-13	2.69e-13
850	-	-	-	-	2.07e-13	2.88e-13
900	-	-	-	-	2.11e-13	3.10e-13
950	-	-	-	-	2.31e-13	3.15e-13
1000	-	-	-	-	2.35e-13	3.50e-13

表 9.5: x_4 と x_5 の区間幅 (例題 2)

反復数	方法 A		方法 B		提案方法	
	x_4	x_5	x_4	x_5	x_4	x_5
1	1.11e-16	1.11e-16	1.11e-16	1.11e-16	1.11e-16	5.55e-17
50	3.62e-05	7.72e-05	9.42e-15	1.67e-14	9.75e-15	1.45e-14
100	-	-	1.24e-13	2.00e-13	2.36e-14	3.18e-14
150	-	-	5.02e-05	9.35e-05	2.93e-14	4.37e-14
200	-	-	-	-	4.63e-14	6.37e-14
250	-	-	-	-	4.97e-14	7.51e-14
300	-	-	-	-	6.96e-14	9.61e-14
350	-	-	-	-	6.98e-14	1.04e-13
400	-	-	-	-	8.98e-14	1.25e-13
450	-	-	-	-	8.99e-14	1.32e-13
500	-	-	-	-	1.12e-13	1.57e-13
550	-	-	-	-	1.12e-13	1.62e-13
600	-	-	-	-	1.34e-13	1.88e-13
650	-	-	-	-	1.35e-13	1.94e-13
700	-	-	-	-	1.54e-13	2.18e-13
750	-	-	-	-	1.59e-13	2.27e-13
800	-	-	-	-	1.73e-13	2.50e-13
850	-	-	-	-	1.84e-13	2.58e-13
900	-	-	-	-	1.92e-13	2.79e-13
950	-	-	-	-	2.08e-13	2.91e-13
1000	-	-	-	-	2.09e-13	3.08e-13

表 9.6: 実行時間 (例題 2)

反復数	方法 A	方法 B	提案方法
1	0.000	0.004	0.016
50	0.021	0.251	1.573

図 9.2: 区間幅の比較 (例題 2)

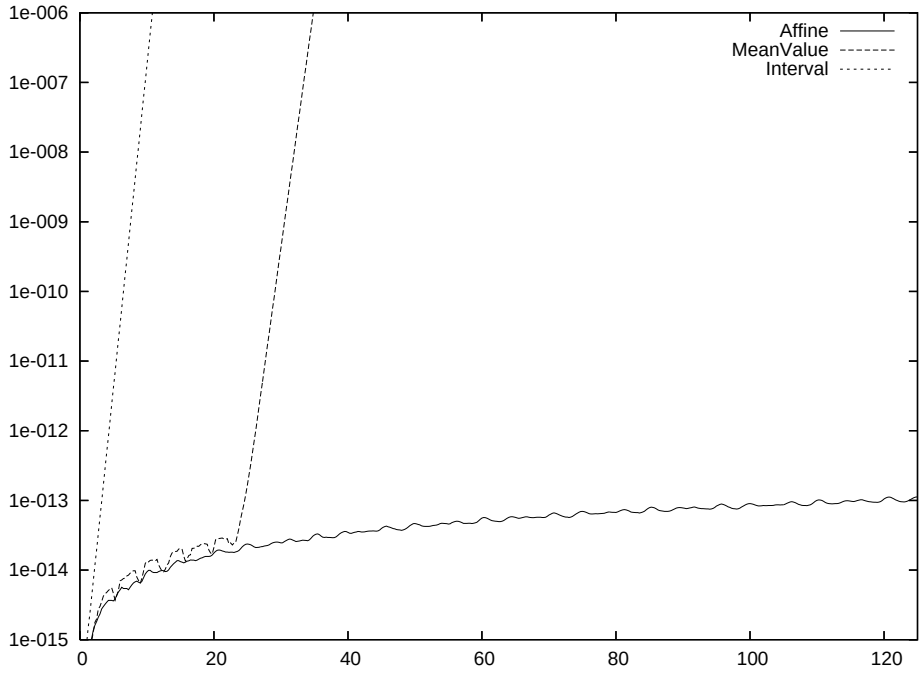


表 9.7: x_0 と x_1 の区間幅 (例題 3)

反復数	方法 A		方法 B		提案方法	
	x_0	x_1	x_0	x_1	x_0	x_1
1	2.22e-16	1.11e-16	2.22e-16	1.11e-16	2.22e-16	1.11e-16
50	3.27e-10	8.73e-11	2.33e-14	1.40e-14	1.83e-14	6.33e-15
100	8.67e-05	2.34e-05	4.56e-14	2.89e-14	3.52e-14	1.37e-14
150	-	-	6.54e-14	4.25e-14	5.07e-14	2.01e-14
200	-	-	6.66e-03	1.72e-13	6.21e-14	2.63e-14
250	-	-	-	-	7.32e-14	3.25e-14
300	-	-	-	-	8.27e-14	3.90e-14
350	-	-	-	-	9.11e-14	4.45e-14
400	-	-	-	-	9.85e-14	5.01e-14
450	-	-	-	-	1.05e-13	5.47e-14
500	-	-	-	-	1.11e-13	5.94e-14
550	-	-	-	-	1.15e-13	6.36e-14
600	-	-	-	-	1.19e-13	6.83e-14
650	-	-	-	-	1.23e-13	7.25e-14
700	-	-	-	-	1.27e-13	7.72e-14
750	-	-	-	-	1.31e-13	8.14e-14
800	-	-	-	-	1.36e-13	8.52e-14
850	-	-	-	-	1.40e-13	8.93e-14
900	-	-	-	-	1.45e-13	9.31e-14
950	-	-	-	-	1.50e-13	9.70e-14
1000	-	-	-	-	1.55e-13	1.02e-13

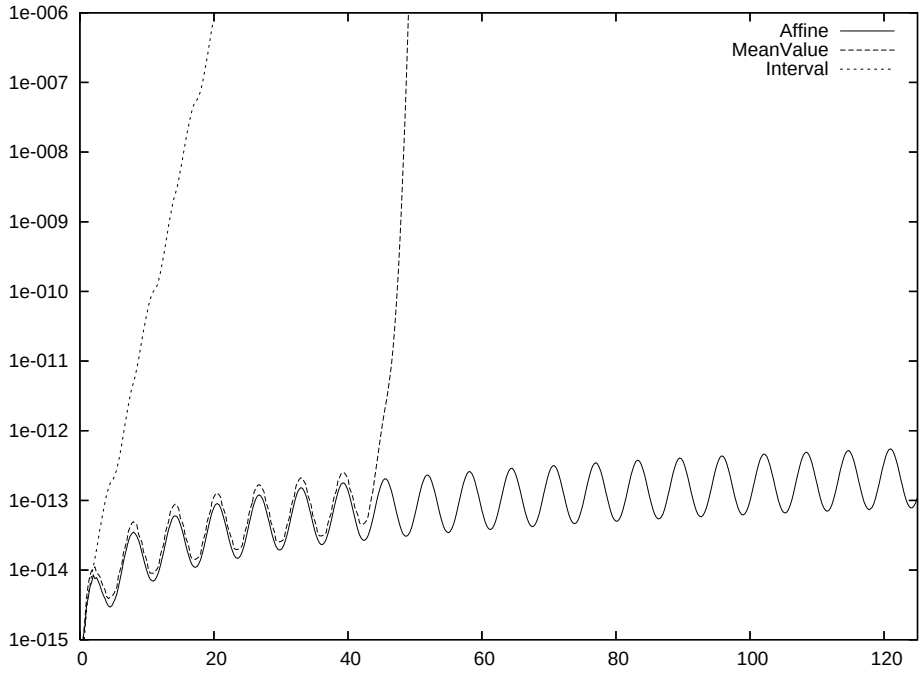
表 9.8: x_2 の区間幅 (例題 3)

反復数	方法 A	方法 B	提案方法
	x_2	x_2	x_2
1	2.78e-17	2.78e-17	2.78e-17
50	8.73e-11	5.91e-15	4.70e-15
100	2.34e-05	1.33e-14	9.96e-15
150	-	2.02e-14	1.51e-14
200	-	1.46e-13	2.00e-14
250	-	-	2.53e-14
300	-	-	3.07e-14
350	-	-	3.65e-14
400	-	-	4.27e-14
450	-	-	4.87e-14
500	-	-	5.50e-14
550	-	-	6.07e-14
600	-	-	6.67e-14
650	-	-	7.43e-14
700	-	-	8.18e-14
750	-	-	8.94e-14
800	-	-	9.74e-14
850	-	-	1.06e-13
900	-	-	1.14e-13
950	-	-	1.22e-13
1000	-	-	1.30e-13

表 9.9: 実行時間 (例題 3)

反復数	方法 A	方法 B	提案方法
1	0.000	0.096	0.037
50	0.017	4.735	2.592
100	0.033	9.361	5.469

図 9.3: 区間幅の比較 (例題 3)



9.4　まとめ

数値例から，全ての例題において，提案方法が最も区間の増大を抑えていることが分かる．特に，既存の方法では到底解くことができないような1000回の反復においても，まだ区間が発散していないことが分かる．また，実行時間においても，問題によっては平均値形式を上回る速度が出ているのが分かる．この数値例から，アフィンベキ級数演算の有効性を示すことができた．

謝辞

本研究, 論文を作成するにあたり, また日常生活あらゆる場面において懇切丁寧な御指導及び御激励を幾度となくいただき, 日常のいろいろな会話からもたくさんの知識とものの考え方を学ばせていただいた柏木雅英准教授に心より深く感謝いたします.

また, 研究会などにおいて様々な知識をご教授くださり, 精度保証付き数値計算への興味をもたせてくれた大石進一教授に深く感謝いたします.

また, 共同研究を行って下さる等，非常に多くの機会で親身になってご助言くださった柏木研究室助手の宮島信也氏に深く感謝いたします.

また, 同学年として様々な面から楽しい時間を過ごさせてくれ, 時には意見交換なども交わして頂いた 内村 創氏，吉住 賢氏に深く感謝いたします.

最後に, 研究含めすべての面でお世話になった柏木研究室の環境と, それを作り維持発展してきたすべての関係者の方々に深く感謝いたします.

参考文献

[1] Marcus Vinícius A.Andrade, João L.D.Comba and Jorge Stolfi,"Affine Arithmetic",INTERVAL'94, St.petersburg(Russia),March 5-10,1994

[2] 大石進一著,"数値計算", 裳華房,1999 年

[3] 大石進一著,"精度保証付き数値計算", コロナ社,2000

[4] 伊理正夫, 久保田光一 : "高速自動微分法 (I), (II)", 応用数理, Vol. 1, No. 1, No. 2, pp.17-35, pp.53–63, 1991.

[5] Masahide Kashiwagi and Shin'ichi Oishi : "Krawczyk-Based Numerical Validation Using Rational Arithmetic", Proc. 1993 International Symposium on Nonlinear theory and its Applications (NOLTA '93), pp.399–402 (Dec 1993).

[6] R. E. Moore : "Methods and Applications of Interval Analysis", SIAM, Philadelphia (1979).

[7] R. J. Lohner : "Enclosing the Solutions of Ordinary Initial and Boundary Value Problems", In E. Kaucher, U. Kulisch and Ch. Ullrich (eds.) : "Computer Arithmetic, Scientific Computation and Programming Languages", B. G. Teubner, Stuttgart, pp.255–286 (1987) .

[8] L. B. Rall : "Automatic Differentiation : Techniques and Applications", Lecture Notes in Computer Science No. 120, Springer, New York (1981).

[9] E. W. Kaucher and W. L. Miranker : "Self-Validating numerics for function space problems", Academic Press, New York (1984).

[10] E. W. Kaucher and W. L. Miranker : "Validating computation in a function space", Reliability in Computing (eds. R. E. Moore), Academic Press, San Diego, pp.403–425 (1988).