平成12年度卒業論文

Affine Arithmeticを利用した非線形方程式の解の非存在領域の除去

Elimination of Non-existence Region of the Solution of Nonlinear Equations using Affine 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

G97P030-4　菊池智行

目 次

第 1 章　序論 **1**

1.1　背景 2

1.2　本論文の目的 2

1.3　本論文の構成 2

第 2 章　区間解析 **4**

2.1　はじめに 5

2.2　計算機上の数 5

2.2.1　データ形式 5

2.2.2　丸めの切り替え 6

2.2.3　10 進小数と 2 進小数の変換 7

2.3　区間演算 7

2.3.1　区間の定義 8

2.3.2　四則演算 8

2.3.3　関数の値域評価 8

2.4　計算機上の区間演算 10

2.4.1　区間の定義 10

2.4.2　四則演算 11

2.5　おわりに 12

第 3 章　非線形方程式の全解探索と Krawczyk の方法 **13**

3.1　はじめに 14

3.2　非線形方程式の求解について 14

3.3　Krawczyk の方法 14

3.3.1　縮小写像原理 15

3.3.2　平均値形式 15

3.3.3　Krawczyk の方法 16

3.4　非線形方程式の全解探索 18

3.5 アルゴリズムの停止性 20
3.6 おわりに 20

第 4 章 Affine Arithmetic 21
4.1 はじめに 22
4.2 Affine Arithmetic について 22
4.2.1 初期化 22
4.2.2 Affine 多項式の区間表示 22
4.2.3 線形演算 23
4.2.4 非線形演算 23
4.3 計算例 24
4.4 おわりに 28

第 5 章 Affine Arithmetic による解の非存在領域の除去 29
5.1 はじめに 30
5.2 Affine Arithmetic を利用した非存在テスト 30
5.2.0 区間表示に還元する方法 30
5.2.1 絶対値の和を用いる方法 31
5.2.2 単体法の Phase-1 を用いる方法 32
5.2.3 一般逆行列を用いる方法 32
5.2.4 ニュートン写像を用いる方法 33
5.3 おわりに 35

第 6 章 数値実験 36
6.1 はじめに 37
6.2 実験に関して 37
6.2.1 全解探索アルゴリズム 37
6.2.2 プログラムについて 40
6.2.3 実験環境 40
6.3 実験 1 40
6.3.1 例題 40
6.3.2 実行結果 41
6.3.3 実験 1 からの考察 44
6.4 実験 2 47
6.4.1 例題 47
6.4.2 実行結果 48

6.4.3 実験 2 からの考察 48
6.5 まとめ 49

謝辞 **50**

参考文献 **52**

図 目 次

3.1 全解探索のようす 19

4.1 x^2 の包含 25
4.2 Affine 形式の区間演算への還元 28

5.1 Affine 形式の絶対値関数 $|x|$ 31
5.2 超立方体 $-1 \leq \varepsilon_i \leq 1$ 33
5.3 超立方体と超球 34

表 目 次

2.1 区間の乗算 $[x] \times [y]$ 9
2.2 区間の除算 $[x]/[y]$ 9

6.1 各次元における解の個数 41
6.2 実行結果 41
6.3 実行結果 42
6.4 実行結果 42
6.5 実行結果 42
6.6 実行結果 43
6.7 実行結果 43
6.8 実行結果 43
6.9 実行結果 44
6.10 実行結果の比較 46
6.11 実行結果の比較 46
6.12 実行結果の比較 46
6.13 実行結果の比較 47
6.14 実行結果 48

第1章　序論

1.1 背景

方程式を解いたり、積分を求めるといった連続数学の問題を計算機を使って解く研究が起こり、このような数値計算の理論は発達し多くの成果をあげてきた。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。

非線形方程式の求解問題はそのような問題のひとつであり、その解法については縮小写像原理と区間演算の手法を組み合わせたKrawczykの方法を利用して、与えられた領域内のすべての解を得るアルゴリズムが知られている。しかし、このようなアルゴリズムは問題が高次元になるとその実行にかかる時間が飛躍的に長くなることが多く、時間の短縮が求められている。非線形方程式の全解探索においては、解が存在しない領域を効率的に見つけて探索領域から除去することが探索全体の時間の短縮に有効であることがわかっている。一方で最近、Affine Arithmeticとよばれる区間演算が提案され、研究されている。このAffine Arithmetic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。

1.2 本論文の目的

非線形方程式の全解探索アルゴリズムにおける、解の非存在領域の判定の効率化をねらいとしてAffine Arithmeticを応用した解の非存在テストを行い、それによって全解探索全体の性能を向上できるかどうかを数値実験により検証し、その考察を与えることが本論文の目的である。

1.3 本論文の構成

本論文の前半においては本論文のテーマであるAffine Arithmeticを利用した全解探索を行うのに必要な事項について述べる。

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章では非線形方程式の全解探索をめぐる事項について述べる。

ある領域内にただ一つ解が存在することを保証するKrawczykの方法とそれを用いた全解探索を実現するアルゴリズムについてとりあげる。

4章では区間演算のひとつであるAffine Arithmeticについて、その原理と特徴を述べ、簡単な例題により通常の区間演算とAffine Arithmeticの比較を行う。

これらをふまえ、本論文の後半においては実際にAffine Arithmeticにより全解探索の改良を試みる。

5章においては非線形方程式の解の非存在領域を除去するアルゴリズムについて、Affine Arithmeticを利用した方法の原理を述べる。

続く6章においてそのアルゴリズムを実装し、実際に非線形方程式を例題としてその全解探索を行い、実行時間などを測定してAffine Arithmeticを利用した非存在領域の除去アルゴリズムの有効性を検証する。

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。$\bigcirc = \{\triangle, \triangledown, \diamondsuit\}$、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、

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

に対し、IEEE754 では浮動小数点数演算 $\odot$ 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

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

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。

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。

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

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

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 $[x]/[y]$

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

$$\begin{aligned} \bigcirc &: I\mathbf{R} \to IF \\ \bigcirc &= \{\triangle, \triangledown, \Diamond\} \end{aligned}$$

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。

$$\begin{aligned} \cdot &= \{+, -, \times, /\} \\ \bigcirc &= \{\triangle, \triangledown, \Diamond\} \end{aligned}$$

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。

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

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 [1]、[2])

なお、当たり前のことであるが、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　非線形方程式の全解探索とKrawczykの方法

3.1 はじめに

本章では非線形方程式の解の探索について述べる。区間演算と不動点定理を組み合わせた Krawczyk の方法などの、与えられた探索領域で解の存在あるいは非存在を判定する方法を取り上げ、それらを用いた非線形方程式の全解探索アルゴリズムについて述べる。

3.2 非線形方程式の求解について

$$f : \mathbf{R}^m \to \mathbf{R}^m \quad f(x) = 0 \quad x \in \mathbf{R}^m \tag{3.1}$$

と表される多次元の非線形方程式の区間 I 内のすべての解を求める問題を考える。このような非線形な方程式の解は一般に有限ステップでの計算で直接的に求めることはできず、ニュートン法のような反復による間接法を用いている。ニュートン法はおおまかには、

$$x_{n+1} = x_n - f'(x_n)^{-1} f(x_n) \tag{3.2}$$

という計算を繰り返して近似解を求めるが、このような解法の精度保証を考えるとき単に計算過程を区間演算化しても効果はない。反復の間に入る誤差は把握できるが、その近似解と真解の誤差や、真解が近似解の近くに本当に存在するかどうかを判定することはできないからである。したがってこのような場合、まずある領域に解が存在することを数学的に保証することが必要となる。

3.3 Krawczyk の方法

Krawczyk の方法は不動点定理を基にし、区間演算の手法を使って導かれる方法で、ある区間について方程式の解をただ一つ含むことを保証する。この方法について述べるために必要な知識を準備として簡単に述べる。

3.3.1 縮小写像原理

X を完備ノルム空間(バナッハ空間)とする。写像 $g : X \to X$ に対し、

$$g(x) = x \tag{3.3}$$

なる $x \in X$ を g の不動点という。$x, y \in X$ の距離 $d(x, y)$ を

$$d(x, y) = \|x - y\| \tag{3.4}$$

で定義する。

不動点の存在を保証する定理を不動点定理という。不動点定理のひとつである縮小写像原理を以下に示す。ここで縮小写像とは、

$$\exists \alpha < 1, \quad \forall x, \forall y \in X \quad d(g(x), g(y)) \leq \alpha d(x, y) \tag{3.5}$$

を満たす写像である。

定理 **3.1** (縮小写像原理)
X を完備ノルム空間、S を空でない X の閉部分集合とし、$g : S \to S$ を縮小写像とする。このとき以下が成立する。

1. g の不動点 x^* が S にただ一つ存在する。
2. $x_0 \in S$、$x_{k+1} = g(x_k)$ で定義される点列 $\{x_k\}$ は x^* に収束し、 その収束速度は
$$d(x_k, x^*) \leq \frac{\alpha^k}{1 - \alpha} d(x_1, x_0) \tag{3.6}$$
で評価される。

非線形方程式の解の探索においては、X を $\mathbf{R}^m$ でおきかえて考えればよい。

3.3.2 平均値形式

平均値形式は区間演算における区間の増大を抑える方法のひとつである。Krawczyk の方法はニュートン写像にこの平均値形式を適用してえられる。$f : \mathbf{R}^m \to \mathbf{R}^m$ として、区間 I で $f(I)$ を評価することを考える。

$$m(I) = f(c) + F'(I)(I - c) \tag{3.7}$$

を f の平均値形式という。ここで $c = \mathrm{mid}(I)$、$F'(I)$ を $f'(I)$ の区間包囲とする ($F'(I)$ の評価には通常の区間演算を用いる)。このとき、(3.7) が $f(I)$ の区間包囲となることが平均値の定理から示される。

例　$f(x) = x^2 - 2x$ を区間 $I = [0.9, 1.1]$ で評価する。真の評価 $f(I) = [-1, -0.99]$ である。

通常の区間演算による場合

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

平均値形式を用いた場合 $c = \mathrm{mid}(I) = 1.0$、$f'(x) = 2x - 2$ により、

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

このように、通常の区間演算で起こっていた区間幅の増大が、平均値形式を用いることでかなり改善されていることがわかる。これは、強い相関をもった関数 x^2 と $2x$(I においては傾きが近い) について、区間演算ではその相関を無視していたのに対し、平均値形式では、

$$F'([0.9, 1.1]) = 2[0.9, 1.1] - 2 = [-0.2, 0.2]$$

のように相関が打ち消されていることによる。このような打ち消しあいが生じるとき、平均値形式はよい評価を与える。

3.3.3　Krawczyk の方法

方程式 (3.1) を不動点問題に帰着させ、縮小写像原理を利用することを考える。すなわち、ある関数 $g(x)$ について、

$$f(x) = 0 \iff g(x) = x \tag{3.8}$$

となるように問題をおきかえるということである。最も単純に考えられるのは $g(x) = x + f(x)$ とすることである。しかし、これでは右辺が縮小性をもつかどうかは f に依存する。縮小性をもたせるためには、近似解

の近辺で傾きを 0 に近づければよい。そこで、まずニュートン法の反復を与える写像を、

$$g(x) = x - L^{-1}f(x) \tag{3.9}$$

で定義する。ただし L は近似解の近くでの f の傾きを近似した正則な行列であるとする。このとき、$f(x) = 0 \iff g(x) = x$ である。これを利用して g の区間包囲を評価する。このとき通常の区間演算により、

$$G(I) = I - L^{-1}F(I) \tag{3.10}$$

としてしまうと $G(I) \subseteq I$ は成立せず、中への写像とならない。なぜなら、式 (3.9) は、傾きを 0 に近づけるために、傾きの近い 2 つの関数の減算を行っており、これが区間幅の増大を引き起こすからである。そこで、この区間幅の増大を抑えるために平均値形式による評価を用いる。(3.9) に平均値形式を適用して ($K(I)$ とおく)

$$\begin{aligned} K(I) &= g(c) + G'(I)(I - c) \\ &= g(c) + (E - L^{-1}F'(I))(I - c) \\ &= c - L^{-1}f(c) + (E - L^{-1}F'(I))(I - c) \end{aligned} \tag{3.11}$$

により、$K(I) \subseteq I$ を判定する。以上から次の Krawczyk の方法が導かれる。

定理 3.2 (Krawczyk の方法)
領域 I 内の方程式 (3.1) の解について、

1. $c = \mathrm{mid}(I)$、$L \simeq f'(c)$ を正則な行列とする。
2. E を単位行列、$M = E - L^{-1}F'(I)$ とし、
   $K(I) = c - L^{-1}f(c) + M(I - c)$
   を求める。
3. $M, K(I)$ について、
   (a) $\|M\| < 1$ かつ $K(I) \subset I \Rightarrow I$ にただ一つ解が存在する。
   (b) $K(I) \cap I = \emptyset \Rightarrow I$ に解はない。

ここで、$m \times n$ 行列のノルムを、ベクトルのノルム $\|\cdot\|$ に対し、

$$\|A\| = \sup_{\|x\|=1} \|Ax\|$$

となるように定義する。$u = (u_1, u_2, \cdots, u_m)^t \in \mathbf{R}^m$ のノルムが最大値ノルム

$$\|u\| = \max_{1 \leq i \leq m} |u_i|$$

であれば、

$$\|A\| = \max_{1 \leq i \leq m} \sum_{1 \leq j \leq n} |A_{ij}|$$

で評価される。区間ベクトル $I = (I_1, I_2, \cdots, I_m)^t$ のノルムを、

$$\|I\| = \max_{1 \leq i \leq m} \max_{x \in I_i} |x| \tag{3.12}$$

で定義し、$m \times n$ 区間行列 $A = (A_{ij})$ のノルムを、

$$\|A\| = \max_{1 \leq i \leq m} \sum_{1 \leq j \leq n} \max_{x \in A_{ij}} |x| \tag{3.13}$$

で定義する。

この Krawczyk の方法において、3b のとき I に解がないことは、$g(x)$ の不動点が存在しないことから証明される。また、3a のときに I にただ一つ解が存在することは、区間写像 $K(I)$ と区間行列 M が

$$\{g(x) | x \in I\} \subseteq K(I) \tag{3.14}$$

$$\{g'(x) | x \in I\} \subseteq M \tag{3.15}$$

をみたすように定義されていることから縮小写像原理によって示され、同時に I 内の任意の点を初期点とするニュートン法の反復によって解に収束することが示される。

3.4 非線形方程式の全解探索

非線形方程式 (3.1) の全解探索アルゴリズムについて考える。Krawczyk の方法 (定理 3.2) はある区間内にただ一つ解が存在するための十分条件をあたえる。逆に、$f(I)$ の区間包囲 $F(I)$ について $0 \notin F(I)$ ならば I 内に解は存在しない。このように、

- I に $f(x) = 0$ の解がないための十分条件を満たす

- I に $f(x) = 0$ の解があるための十分条件を満たす

ことで解の存在あるいは非存在を保証できる。非線形方程式の全解探索ではこれらの方法でテストを行い、判定できなければ区間を分割して、再帰的にテストを行っていくというアルゴリズムが知られている (図 3.1)。

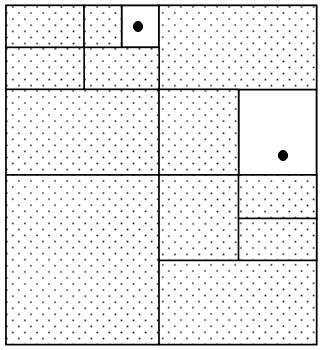

図 3.1: 全解探索のようす

すなわち、解の探索を行う区間 (直方体領域) を T としたとき、方程式 (3.1) の T 内の解を探索するアルゴリズムは以下のようになる。

アルゴリズム **3.3** (非線形方程式の全解探索)

1. 区間のリスト $\mathcal{L}$ を $\mathcal{L} = \{T\}$ で初期化する。
2. $\mathcal{L}$ が空ならば終了。そうでなければ、$\mathcal{L}$ の先頭の要素取り出して、I とし、$\mathcal{L}$ から削除。
3. 区間演算による評価 $F(I)$ を求め、$0 \notin F(I)$ ならば I に解はない。*2* へ。
4. Krawczyk の方法を用いて、解が存在するまたは存在しないと判定できたら 2 へ。
5. I をふたつの区間 I_1,I_2 に分割し、リスト $\mathcal{L}$ の末尾に追加する。2 へ。

全解探索のアルゴリズムはおおむね以上のようであるが、全解探索においては時間、記憶量の面での効率化が重要とされる。とくに、解の非存在の判定の効率化が全解探索全体の効率向上にもつながることがわかっている。

3.5 アルゴリズムの停止性

アルゴリズム 3.3 は解をただ一つ含む区間を、解の存在あるいは非存在のための条件の判定、領域の分割と再帰的な条件判定により得ることができるように思われるが、これは有限ステップで停止するとは限らない。たとえば分割された領域の境界上にちょうど解がのったような場合、解の唯一存在を保証するための十分条件

$$\|M\| < 1 \text{ かつ } K(I) \subset I$$

は満たされず、分割を繰り返しても有限ステップでアルゴリズムが停止することはない。そこで、[5] を参考に、存在判定を行う際に区間を変更する。区間 I の中心がニュートン写像によって修正される量の定数倍と I の半径の大きいほうを新たな区間の半径とし、この新たな区間を解の存在判定の対象とする。以上はすなわち、

$$c = \mathrm{mid}(I) \tag{3.16}$$
$$\delta = \rho\|L^{-1}f(c)\| \tag{3.17}$$
$$I' = I \cup B(c, \delta) \tag{3.18}$$

によって得られた I' に対して Krawczyk の方法を適用するというものである。ただし、ρ は $\rho > 1$ なる定数、$B(c, r)$ は中心 c、半径 r の球とする。

このような方法により、ある条件のもとで全解探索アルゴリズムが有限ステップで停止することが示される。[5]

3.6 おわりに

本章では非線形方程式の精度保証付きの求解法について述べた。上述のように全解探索のアルゴリズムは確立されているが、効率化については課題が残っている。4 章、5 章で Affine Arithmetic を利用した全解探索の効率向上の方法をとりあげる。

第4章　Affine Arithmetic

4.1 はじめに

Affine Arithmetic は区間演算の一手法で、1994 年に Stolfi ら [4] によって提案された。Affine Arithmetic 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 Affine 形式と区間表示との変換、Affine 形式での演算について述べたあと、簡単な例題によって通常の区間演算と Affine Arithmetic の違いについて述べ、5 章の Affine Arithmetic を応用した非存在判定テストのための足がかりとする。

4.2 Affine Arithmetic について

Affine Arithmetic においては、変数 x は Affine 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 Affine 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 Affine 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 Affine 多項式の区間表示

Affine 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

Affine 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 Affine 多項式で表すことはできない。このため、Affine Arithmetic 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。

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 Affine 形式では表現できない。Affine Arithmetic 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 Affine 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。Affine Arithmetic 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。

非線形二項演算

$f(x,y)$ を非線形二項演算、$(x,y) \in I_x \times I_y = I$ とすると、I での $f(x,y)$ の像 $f(I)$ は曲面を描くため Affine 形式では表現できない。Affine Arithmetic では、I における $f(x,y)$ の近似を

$$ax + by + c \tag{4.11}$$

とし、$f(x,y)$ と (4.11) との最大誤差 δ を、

$$\delta = \max_{(x,y)\in I} |f(x,y) - (ax + by + c)| \tag{4.12}$$

で求め、ダミー変数 ε_{n+1} を追加して、新たな Affine 多項式

$$(ax + by + c) + \delta\varepsilon_{n+1} \tag{4.13}$$

を得て、これを $f(I)$ の包含とする。

非線形演算の包含

このように、Affine Arithmetic 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、Affine Arithmetic 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 Affine Arithmetic 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 $(a, b, \delta$ 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。

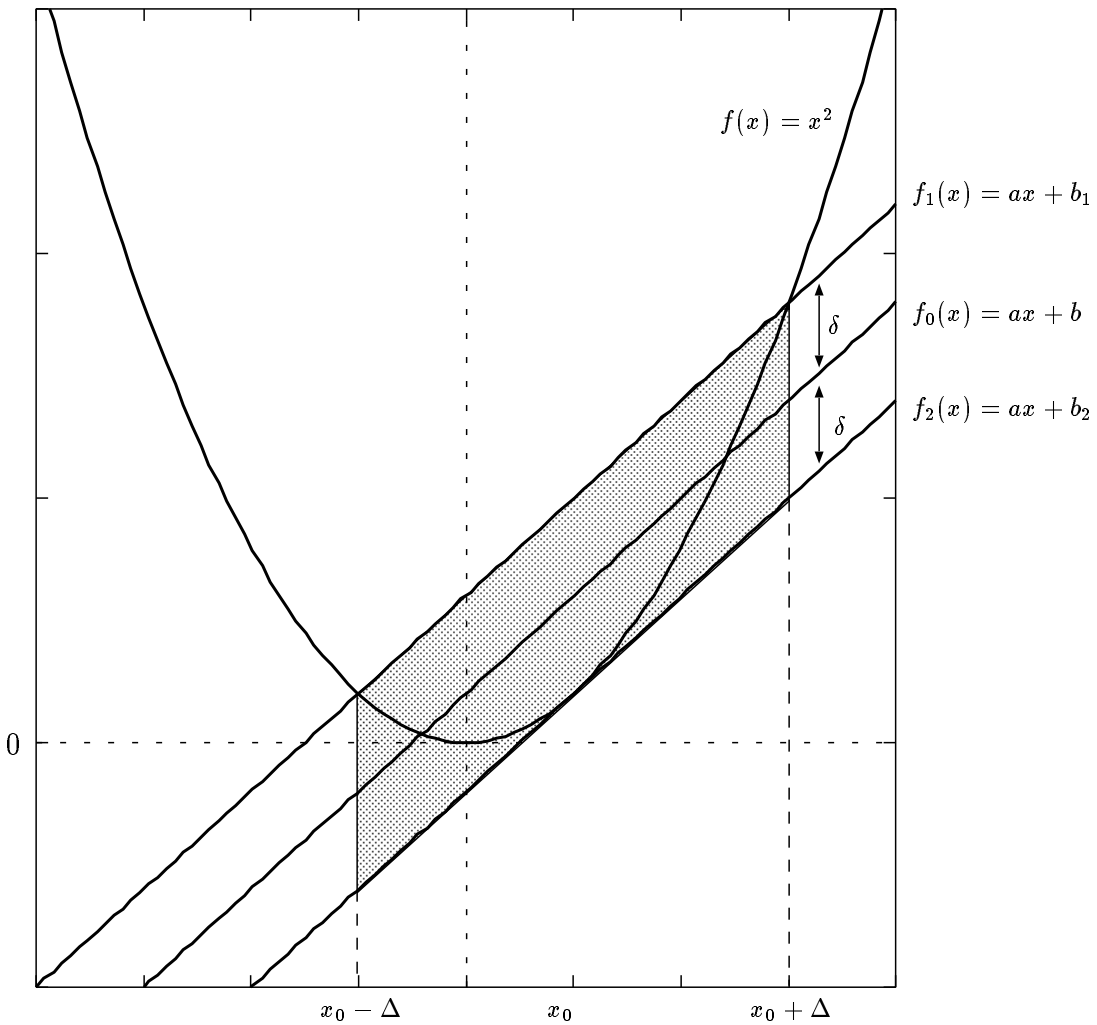

図 4.1: x^2 の包含-式 (4.4) により Δ を求める。このとき、$a = 2x_0, b_1 = -{x_0}^2 + \Delta^2, b_2 = -{x_0}^2$ であり、$b = \frac{1}{2}(b_1 + b_2) = -{x_0}^2 + \frac{1}{2}\Delta^2, \delta = \frac{1}{2}\Delta^2$ が得られる。

区間演算の場合

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

Affine Arithmetic 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、

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

Affine Arithmetic 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。Affine 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

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

Affine Arithmetic 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。Affine 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 Affine Arithmetic 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 Affine Arithmetic による評価で得られた領域内にはない。

これは Affine 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、Affine Arithmetic では特に必要ない限り計算結果は Affine 形式のままにしておくのがよいとされる (図 4.2)。

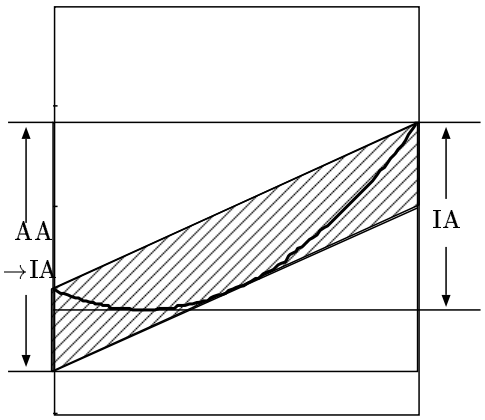

図 4.2: Affine 形式の区間演算への還元

4.4 おわりに

本章では Affine Arithmetic 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 Affine Arithmetic 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。次章ではこのような Affine Arithemtic の特徴を非線形方程式の全解探索に利用する方法についてその原理を述べる。

第5章　Affine Arithmeticによる解の非存在領域の除去

5.1 はじめに

本章では 4 章で述べた Affine Arithmetic を利用して、非線形方程式の解の非存在領域を判定、除去する方法について述べる。「Affine Arithmetic を応用して非線形方程式の全解探索アルゴリズムを改良する」という本論文の目標の上で中心となる章である。

5.2 Affine Arithmeticを利用した非存在テスト

方程式 (3.1) について、探索領域 I において Affine Arithmetic を用いて $f(I)$ の評価を

$$\begin{aligned} f_1 &= a_{10} + a_{11}\varepsilon_1 + \cdots + a_{1n}\varepsilon_n \\ f_2 &= a_{20} + a_{21}\varepsilon_1 + \cdots + a_{2n}\varepsilon_n \\ &\vdots \\ f_m &= a_{m0} + a_{m1}\varepsilon_1 + \cdots + a_{mn}\varepsilon_n \end{aligned} \tag{5.1}$$

のように得ることを考える。ただし $m \leq n$ となる。[6] において、この評価式をそのまま利用あるいは応用することによる解の非存在判定の方法が提案されている。以下にそれらを示す。

5.2.0 区間表示に還元する方法

単純に $f_1, \cdots, f_m$ を式 (4.3)、式 (4.4) にしたがって通常の区間表示に変換する方法である。このとき、f_i を区間表示に戻したものを f_{Ii} とかくことにして、

$$0 \notin f_{Ii} \quad (1 \leq \exists i \leq m) \tag{5.2}$$

ならば、

$$f(x) = 0 \text{ の解が } I \text{ に存在しない}$$

ことがいえる。この方法は、3.4 節の全解探索アルゴリズムで述べた、解が存在しないための十分条件 $0 \notin F(I)$ の判定において、区間包囲 $F(I)$ の評価に Affine 形式の区間表示への還元を用いたものであるが、4.3 節に述べているように、この還元によって Affine 演算のもつ相関性が失われ、評価が甘くなることが考えられる。

5.2.1 絶対値の和を用いる方法

$$R = \sum_{i=1}^{m} |f_i| \tag{5.3}$$

を Affine Arithmetic により評価する。このとき、

$$R > 0 \Rightarrow I \text{ に } f(x) = 0 \text{ の解は存在しない。}$$

がいえる。この方法は、5.2.0 の方法ともとの考え方を同じくしており、別の角度から評価したものとみることができる。なお、Affine 多項式に対する絶対値関数

$$|f| = |a_0 + a_1\varepsilon_1 + \cdots + a_n\varepsilon_n| \tag{5.4}$$

はひとつの単項演算としてその結果を求めることができる(図 5.1)。Affine 形式 R に対する不等式 $R > 0$ の判定は、R の式 (4.3)、式 (4.4) によって得られる通常の区間表示の下端を 0 と比較する。

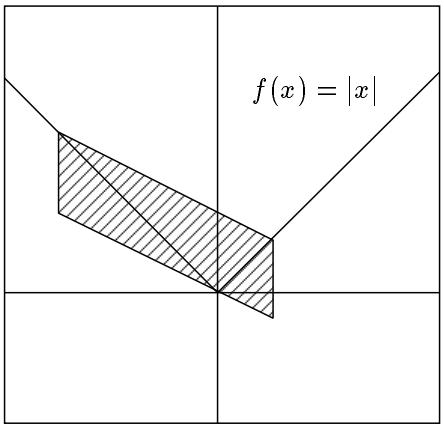

図 5.1: Affine 形式の絶対値関数 $|x|$

5.2.2 単体法の Phase-1 を用いる方法

式 (5.1) から $\varepsilon = (\varepsilon_1 \cdots \varepsilon_n)^t$ についての線形方程式

$$\begin{array}{c} a_{10} + a_{11}\varepsilon_1 + \cdots + a_{1n}\varepsilon_n = 0 \\ a_{20} + a_{21}\varepsilon_1 + \cdots + a_{2n}\varepsilon_n = 0 \\ \vdots \\ a_{m0} + a_{m1}\varepsilon_1 + \cdots + a_{mn}\varepsilon_n = 0 \end{array} \tag{5.5}$$

を解くことを考える。

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \quad b = \begin{pmatrix} -a_{10} \\ \vdots \\ -a_{m0} \end{pmatrix}$$

により、式 (5.5) は、

$$A\varepsilon = b \tag{5.6}$$

とかける。ダミー変数の条件により、

$$-1 \leq \varepsilon_i \leq 1 \quad (i = 1, 2, \cdots, n) \tag{5.7}$$

で表される超立方体と方程式 (5.5) の解空間が交わりをもたない

$$\Rightarrow I \text{ に } f(x) = 0 \text{ の解が存在しない}$$

ことがいえる (図 5.2)。この交わりが空かどうかを判定するには線形計画問題を利用する。すなわち、式 (5.5)、式 (5.7) を制約条件として、単体法の $\mathrm{Phase\text{-}1}$ を動かす (線形計画問題、単体法について $[3]$)。その結果、有限ステップで許容解が存在するか否かが判定される。許容解が存在しないときは (5.5) の解空間と (5.7) の超立方体が交わりをもたないことになる。この方法は 5 つの方法のうちで解の非存在領域を除去する能力が最も高いと考えられるが、最も多くの時間がかかると思われる。また、$\mathrm{Phase\text{-}1}$ はその過程を区間演算化しても精度保証つきの計算ができるわけではなく、計算の誤差の把握が困難であると思われる。

5.2.3 一般逆行列を用いる方法

(5.7) の超立方体の代わりに、それを包み込むような半径 $\sqrt{n}$ の超球を使う (図 5.3)。この方法では (5.5) の解でユークリッド ノルム最小のもの

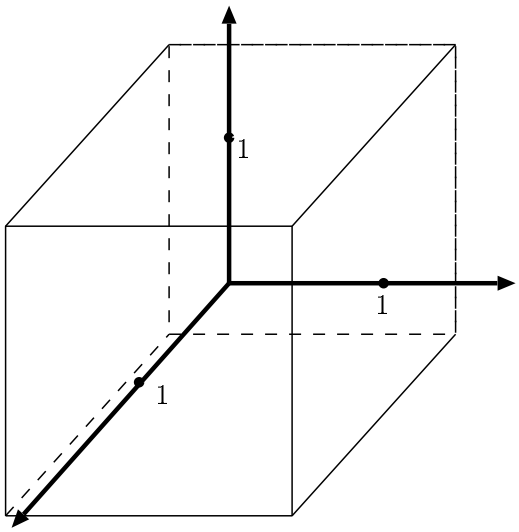

図 5.2: 超立方体 $-1 \leq \varepsilon_i \leq 1$

を得て、そのノルムが $\sqrt{n}$ より大きいならば (5.5) の解空間と (5.7) の超立方体が交わりをもたないことがいえる。この方法は $5.2.2$ に比べると解の非存在領域の除去能力は劣るものの、実行時間は短くできると考えられる。

なお、方程式 (5.6) の最小ノルム解は

$$A^t(AA^t)^{-1}b \tag{5.8}$$

で得ることができる。

5.2.4 ニュートン写像を用いる方法

ニュートン作用素

$$n(x) = x - L^{-1}f(x) \tag{5.9}$$

を利用する。すなわち、

$$n(I) \cap I = \emptyset \tag{5.10}$$

$$\Rightarrow I \text{ に } f(x) = 0 \text{ の解が存在しない}$$

により解の非存在を判定する。ニュートン写像の評価を区間演算で行うと、傾きの近い区間の減算となるため区間幅が増大してしまい、評価を

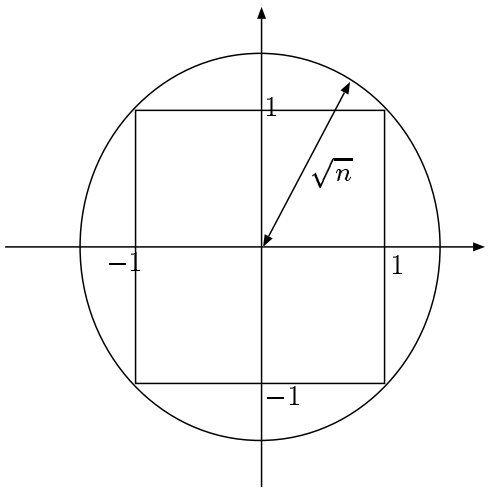

図 5.3: 超立方体と超球

利用することは難しい。ここに Affine Arithmetic を用いるならば、変数間の相関が反映されるため解の非存在判定に利用することができることになる。Affine 形式により、

$$\begin{aligned} f_i(\tfrac{\overline{I_1}+\underline{I_1}}{2} + \tfrac{\overline{I_1}-\underline{I_1}}{2}\varepsilon_1, \cdots, \tfrac{\overline{I_m}+\underline{I_m}}{2} + \tfrac{\overline{I_m}-\underline{I_m}}{2}\varepsilon_m) \\ = g_i(\varepsilon_1, \cdots, \varepsilon_n) \end{aligned} \tag{5.11}$$

と変数変換を行い、g の Jacobi 行列の近似として、

$$L = \begin{pmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mm} \end{pmatrix} \tag{5.12}$$

をとり、

$$\begin{pmatrix} n_1 \\ \vdots \\ n_m \end{pmatrix} = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_m \end{pmatrix} - L^{-1} \begin{pmatrix} g_1 \\ \vdots \\ g_m \end{pmatrix} \tag{5.13}$$

で得られた n_i に対し、n_i を区間で再評価した N_i について、

$$(N_1, \cdots, N_m) \cap ([-1,1], \cdots, [-1,1]) = \emptyset \tag{5.14}$$

が得られるならば、

$g(\varepsilon) = 0$ の解が $([-1,1], \cdots, [-1,1])$ にない

すなわち、

$$f(x) = 0 \text{ の解が } I \text{ に存在しない}$$

ことがいえる。なお、式 (5.13) の右辺は、

$$-L^{-1} \begin{pmatrix} a_{10} + a_{1m+1}\varepsilon_{m+1} + \cdots + a_{1n}\varepsilon_n \\ \vdots \\ a_{m0} + a_{mm+1}\varepsilon_{m+1} + \cdots + a_{mn}\varepsilon_n \end{pmatrix} \tag{5.15}$$

と展開される。

5.3 おわりに

本章では非線形方程式の解の非存在領域の、Affine Arithmetic による評価を利用した除去法についてその原理を述べた。これらの除去法を 3 章で述べた従来のアルゴリズムに組み入れ、全解探索の性能向上に寄与するかどうかを 6 章で検証することにする。

第6章　数値実験

6.1 はじめに

本章では 3 章で述べた全解探索アルゴリズムと、5 章で述べた Affine Arithmetic を利用した非存在判定を組み入れた全解探索アルゴリズムを実装し、非線形方程式の全解探索を行ってその性能を測定し、Affine Arithmetic による非存在テストの有効性を検証する。

6.2 実験に関して

6.2.1 全解探索アルゴリズム

3 章、5 章を踏まえ、本章の実験に用いる探索領域 T 内の方程式 (3.1) の全解探索アルゴリズムについてここでまとめておく。

全解探索アルゴリズムは、アルゴリズム 3.3(p.19) にもとづいており、アルゴリズム 6.1 とアルゴリズム 6.2 は停止性を考慮しているかどうか (3.5 節 (p.20) 参照) という点で異なる。

以下のアルゴリズムにおいて、

- 方法 A は Affine Arithmetic による非存在テストを行わない、従来の方法である。
- 方法 B,C,D は Affine Arithmetic による非存在テストを行う。ただし、
  - 方法 B は 5.2.1 から 5.2.4 の非存在テストのいずれかを行う。
  - 方法 C は 5.2.0 の非存在テストを行う。
  - 方法 D はまず 5.2.0 の非存在テストを行い、失敗したら 5.2.1 から 5.2.4 の非存在テストのいずれかを行う。

というものである。単純に通常の区間に変換する 5.2.0 のテストの有効性を検証するために 3 パターンの非存在テストを行うことにする。

なお、実験で用いる全解探索アルゴリズムの性能の評価の指標として、全解探索の実行時間と、以下の全解探索アルゴリズム中の領域の分割が行われた回数を測定する。

アルゴリズム **6.1** (従来の全解探索への Affine Arithmetic の利用)

1. 区間のリスト $\mathcal{L}$ を $\mathcal{L} = \{T\}$ で初期化する。
2. $\mathcal{L}$ が空ならば終了。そうでなければ、$\mathcal{L}$ の先頭の要素を 取り出して、I とし、$\mathcal{L}$ から削除。
3. 区間演算による評価 $F(I)$ を求め、$0 \notin F(I)$ ならば I に解はない。*2* へ。
4. 方法 **A,B** 次へ。

方法 **C,D** Affine Arithmetic の区間表示への還元による 5.2.0 の非存在テストを行い、解が存在しないと判定できたら *2* へ。

5. 方法 **A,C** 次へ。

方法 **B,D** Affine Arithmetic を利用した非存在テストのいずれかを行う。

テスト **1** 絶対値の和を用いた 5.2.1 のテストを行い、解が存在しないと判定できたら *2* へ。

テスト **2** 単体法の Phase-1 を用いた 5.2.2 のテストを行い、解が存在しないと判定できたら *2* へ。

テスト **3** 一般逆行列を用いた 5.2.3 のテストを行い、解が存在しないと判定できたら *2* へ。

テスト **4** ニュートン写像を用いた 5.2.4 のテストを行い、解が存在しないと判定できたら *2* へ。

6. $c = \mathrm{mid}(I)$ を求め、$L \simeq f'(c)$ を計算する。もし L が非正則ならば *10* へ。
7. E を単位行列、$M = E - L^{-1}F'(I)$ とし、
$K(I) = c - L^{-1}f(c) + M(I - c)$
を求める。
8. $\|M\| < 1$ かつ $K(I) \subset I$ ならば I にただ一つ解が存在する。*2* へ。
9. $K(I) \cap I = \emptyset$ ならば I に解はない。 *2* へ。
10. I をふたつの区間 I_1, I_2 に分割し、リスト $\mathcal{L}$ の末尾に追加する。*2* へ。

アルゴリズム **6.2** (停止性を考慮した全解探索への Affine Arithmetic の利用)

1. 区間のリスト $\mathcal{L}$ を $\mathcal{L} = \{T\}$ で初期化する。
2. $\mathcal{L}$ が空ならば終了。そうでなければ、$\mathcal{L}$ の先頭の要素を 取り出して、I とし、$\mathcal{L}$ から削除。
3. 区間演算による評価 $F(I)$ を求め、$0 \notin F(I)$ ならば I に解はない。*2* へ。
4. 方法 **A,B** 次へ。

   方法 **C,D** Affine Arithmetic の区間表示への還元による 5.2.0 の非存在テストを行い、解が存在しないと判定できたら *2* へ。
5. 方法 **A,C** 次へ。

   方法 **B,D** Affine Arithmetic を利用した非存在テストのいずれかを行う。

   テスト **1** 絶対値の和を用いた 5.2.1 のテストを行い、 解が存在しないと判定できたら *2* へ。

   テスト **2** 単体法の Phase-1 を用いた 5.2.2 のテストを行い、解が存在しないと判定できたら *2* へ。

   テスト **3** 一般逆行列を用いた 5.2.3 のテストを行い、解が存在しないと判定できたら *2* へ。

   テスト **4** ニュートン写像を用いた 5.2.4 のテストを行い、解が存在しないと判定できたら *2* へ。
6. $c = \mathrm{mid}(I)$ を求め、$L \simeq f'(c)$ を計算する。もし L が非正則ならば *11* へ。
7. $I' = I \cup B(c, \rho||L^{-1}f(c)||)$ を求める。ここで、$B(c,r)$ は中心 c、半径 r の球である。
8. E を単位行列、$M = E - L^{-1}F'(I')$ とし、
   $K(I') = c - L^{-1}f(c) + M(I' - c)$
   を求める。
9. $\|M\| < 1$ かつ $K(I') \subset I'$ ならば I' にただ一つ解が存在する。*2* へ。
10. $K(I') \cap I' = \emptyset$ ならば I に解はない。 *2* へ。
11. I をふたつの区間 I_1, I_2 に分割し、リスト $\mathcal{L}$ の末尾に追加する。*2* へ。

6.2.2 プログラムについて

プログラミング言語に C++を用いる。C++のテンプレート機能を利用して区間演算、自動微分、ベクトル及び行列、Affine 演算、単体法、Krawczyk 法のライブラリを実装し、それらを用いて 6.3 節、6.4 節の方程式の全解探索を行う。

また、Affine 多項式の係数についてはそれを区間で評価することで精度の保証を行う。しかしながら単体法の Phase-1 による 5.2.2 の非存在テストにおいてはそのアルゴリズム上この方法が利用できないため (p.32)、通常の倍精度浮動小数点数を用いることとする。したがって、5.2.2 を利用した全解探索には精度保証はなく、その結果は参考として扱う。

6.2.3 実験環境

数値実験を行う計算機環境は以下のとおりである。

CPU PentiumII 350MHz

メモリ 128MB

OS Free BSD 2.2.8

コンパイラ g++ 2.7.2.1

6.3 実験 1

6.3.1 例題

[5] を参考に、以下のエサキダイオードからなる非線形抵抗回路の回路方程式の全解探索を試みる。

$$\begin{aligned} g(x_1) + x_1 + x_2 + \cdots + x_n &= 1 \\ g(x_2) + x_1 + x_2 + \cdots + x_n &= 2 \\ &\vdots \\ g(x_n) + x_1 + x_2 + \cdots + x_n &= n \end{aligned} \tag{6.1}$$

ここに、ダイオードの特性を

$$g(x) = 2.5x^3 - 10.5x^2 + 11.8x \tag{6.2}$$

で与える。次元数 $n = 2, 3, 4, 5, 6$ とし、解の探索範囲を $[-3, 3] \times [-3, 3] \times \cdots \times [-3, 3]$ とする。各次元における解の個数を表 6.1 に示す。

この例題では、従来の方法と非存在テストを組み入れた全解探索の性能の比が、n を大きくするにつれてどう変化するかを調べることができるであろう。

n	解の個数
2	1
3	1
4	3
5	5
6	5

表 6.1: 各次元における解の個数

6.3.2 実行結果

実行結果を示す。以下の結果において、$(\cdot)$ で囲まれた数値は得られた解の個数が正しい場合よりも 1 つ少なかったこと、したがって正しい結果が得られなかったことを示している。

n	方法 A	方法 B			
		テスト 1	テスト 2	テスト 3	テスト 4
2	0.027638	0.029408	0.12586	0.033120	0.029497
3	0.35638	0.31716	1.6957	0.36449	0.26360
4	4.0338	2.8665	18.577	3.1803	2.3361
5	42.931	21.785	162.706	27.682	19.031
6	417.513	142.559	(895.260)	191.176	131.696

表 6.2: 例題 1-アルゴリズム 6.1-方法 A,B(実行時間、単位:s)

n	方法 A	方法 C	方法 D			
			テスト 1	テスト 2	テスト 3	テスト 4
2	0.027638	0.021711	0.023497	0.088675	0.024206	0.021344
3	0.35638	0.18208	0.19833	1.1915	0.21266	0.18626
4	4.0338	1.5564	1.6833	15.141	1.8357	1.4025
5	42.931	11.451	12.145	124.916	14.133	9.0200
6	417.513	73.792	71.605	(797.515)	77.468	53.391

表 6.3: 例題 1-アルゴリズム 6.1-方法 A,C,D(実行時間、単位:s)

n	方法 A	方法 B			
		テスト 1	テスト 2	テスト 3	テスト 4
2	76	37	27	39	35
3	516	204	104	203	152
4	3,589	1,055	413	978	746
5	24,076	5,124	1,443	4,961	3,633
6	160,110	22,921	(4,520)	22,722	16,926

表 6.4: 例題 1-アルゴリズム 6.1-方法 A,B(分割数)

n	方法 A	方法 C	方法 D			
			テスト 1	テスト 2	テスト 3	テスト 4
2	76	30	30	27	30	27
3	516	131	130	104	129	117
4	3,589	643	631	413	622	489
5	24,076	2,981	2,863	1,443	2,830	1,961
6	160,110	12,868	11,924	(4,520)	11,536	7,419

表 6.5: 例題 1-アルゴリズム 6.1-方法 A,C,D(分割数)

n	方法 A	方法 B			
		テスト 1	テスト 2	テスト 3	テスト 4
2	0.10468	0.029199	0.11290	0.033532	0.028996
3	1.9907	0.31463	1.4965	0.39350	0.25814
4	27.844	6.1919	18.903	3.9188	2.3017
5	399.896	83.688	159.463	39.529	19.147
6	4742.411	1372.450	915.420	246.003	132.393

表 6.6: 例題 1-アルゴリズム 6.2-方法 A,B(実行時間、単位:s)

n	方法 A	方法 C	方法 D			
			テスト 1	テスト 2	テスト 3	テスト 4
2	0.10468	0.021105	0.023728	0.088555	0.023914	0.021159
3	1.9907	0.17973	0.19750	1.1998	0.21466	0.18476
4	27.844	1.6698	1.6941	15.249	1.8712	1.3746
5	399.896	15.097	13.971	121.577	17.799	8.8615
6	4742.411	99.888	88.628	792.855	102.413	52.316

表 6.7: 例題 1-アルゴリズム 6.2-方法 A,C,D(実行時間、単位:s)

n	方法 A	方法 B			
		テスト 1	テスト 2	テスト 3	テスト 4
2	487	37	27	39	35
3	4,716	205	104	220	152
4	39,246	2,795	410	1,292	743
5	326,789	24,064	1,420	7,849	3,610
6	2,891,630	297,470	4,513	31,846	16,918

表 6.8: 例題 1-アルゴリズム 6.2-方法 A,B(分割数)

n	方法 A	方法 C	方法 D			
			テスト 1	テスト 2	テスト 3	テスト 4
2	487	30	30	27	30	27
3	4,716	131	130	104	129	117
4	39,246	702	665	410	657	486
5	326,789	3,972	3,366	1,420	3,591	1,938
6	2,891,630	18,458	14,859	4,513	13,984	7,411

表 6.9: 例題 1-アルゴリズム 6.2-方法 A,C,D(分割数)

6.3.3 実験 1 からの考察

まず、アルゴリズム 6.1 とアルゴリズム 6.2 の違いを各表の方法 A の数値から考える。アルゴリズム 6.2 の方が長い時間を要し、分割数も多い。これは、探索アルゴリズムにおける Krawczyk の方法の適用の違いのため、すなわち、停止性のために領域を (小さくすることなく) 修正するアルゴリズム 6.2 では Krawczyk の方法で判定に成功する割合が小さくなるため、結果的に領域の分割回数が増え、実行時間が長くなるものと思われる。

次に Affine Arithmetic を用いた全解探索について調べてみる。ここで、実験結果の数値をわかりやすく見るために、アルゴリズム 6.1、アルゴリズム 6.2 それぞれについて、Affine Arithmetic を利用しない従来の方法 (方法 A としたもの) の数値を 100 とおいたときの Affine Arithmetic を利用した各方法の実験結果を比較したものを表 6.10 から表 6.13 に示す。これらの表から Affine Arithmetic を用いたそれぞれの非存在テスト (以下、非存在テストという言葉は Affine Arithmetic による非存在テストを指すものとする) の有効性について検討する。

まず、表を全体的に見渡してみる。分割数の比が常に 100 より小さいので、少なくとも非存在テストが無駄ではないことを確認する。次に、非存在テストを行わない場合と行った場合の実行時間の比から見ると、次元数 n が大きくなるにつれて非存在テストを行った場合の実行時間の比が小さくなっていることがわかる。このことから問題の規模が大きくなると非存在テストを行うことがより効果的になると考えられる。これは、n が小さいうちは、非存在テストによって解が存在しない領域を除去できても、そのテスト自体にかかるコスト (Affine Arithmetic による評価と非

存在判定) が相対的に大きくなって全体の実行時間の短縮につながらない一方、n が大きくなると、非存在テストそのもののコストが相対的に小さくなり、解の非存在領域の除去の効果が高くなるためと思われる。

次に、Affine Arithmetic を用いたそれぞれの非存在テストについて検討する。実行時間は当然その性能を端的に表すが、分割数はそのテストによる非存在領域の除去能力を表しているとみることができる。だから、分割数が少ないのに時間がかかる場合があれば、それは除去能力は高いがテストを 1 回行うためのコストが高いとみることができる。非存在テストを行った結果時間が多くかかるとなればその非存在テストはもちろん有効とはいえないが、除去能力自体が高ければ問題の規模が非常に大きくなると実行時間が飛躍的に短縮されるかもしれないという見方もできるであろう。

さて、実験結果から Affine Arithmetic を用いた非存在テストについて以下のようなことがいえる。

- 最も単純な、区間表示への還元による 5.2.0 の非存在テスト (方法 C) が意外に効果的である。
- 非存在テスト 1 から 4 すべてについて、5.2.0 の非存在テストを併用した方法 D の方が、併用しない方法 B よりも実行時間、分割数両面でよい性能をもつ。
- ニュートン写像を用いたテスト 4(5.2.4 の非存在テスト) が最も実行時間を短縮する。
- 分割数が最も少ないのは単体法の Phase-1 を用いたテスト 2(5.2.2 の非存在テスト) の場合で、しかも方法 B と方法 D で分割数が変わらない。この非存在テストが Affine Arithmetic による評価を利用した非存在テストにおいてはほぼ最良の除去能力をもっていると考えられる。ただし実行時間があまりにかかりすぎる。単体法による非存在テスト 1 回に時間が多くかかるためと思われる。
- 絶対値の和を用いたテスト 1(5.2.1 の非存在テスト) と一般逆行列を用いたテスト 3(5.2.3 の非存在テスト) も有効であり、またそれぞれを利用した全解探索の実行性能が近いともみえるが、テスト 1 を利用した場合、アルゴリズム 6.2 では、次元が大きくなるほど効果的とはいえないようである。

n	方法 C	方法 B/方法 D			
		テスト 1	テスト 2	テスト 3	テスト 4
2	78.6	106/85.1	456/321	120/87.6	107/77.3
3	51.1	89.0/55.7	476/334	102/59.7	74.0/52.3
4	38.6	71.1/41.7	461/375	78.8/45.5	57.9/34.8
5	26.7	50.7/28.3	379/291	64.5/32.9	44.3/21.0
6	17.7	34.1/17.2	(214)/(191)	45.8/18.6	31.5/12.8

表 6.10: 例題 1-アルゴリズム 6.1(実行時間比較)

n	方法 C	方法 B/方法 D			
		テスト 1	テスト 2	テスト 3	テスト 4
2	39.5	48.7/39.5	35.5/35.5	51.3/39.5	46.1/35.5
3	25.4	39.5/25.2	20.2/20.2	39.3/25.0	29.5/22.7
4	17.9	29.4/17.6	11.5/11.5	27.2/17.3	20.8/13.6
5	12.4	21.3/11.9	5.99/5.99	20.6/11.8	15.1/8.15
6	8.04	14.3/7.45	(2.82)/(2.82)	14.2/7.21	10.6/4.63

表 6.11: 例題 1-アルゴリズム 6.1(分割数比較)

n	方法 C	方法 B/方法 D			
		テスト 1	テスト 2	テスト 3	テスト 4
2	20.2	27.9/22.7	108/84.6	32.0/22.8	27.7/20.2
3	9.03	15.8/9.92	75.2/60.3	19.8/10.8	13.0/9.28
4	6.00	22.2/6.08	67.9/54.8	14.1/6.72	8.27/4.94
5	3.78	20.9/3.49	39.9/30.4	9.88/4.45	4.79/2.22
6	2.11	28.9/1.87	19.3/16.7	5.19/2.16	3.08/1.19

表 6.12: 例題 1-アルゴリズム 6.2(実行時間比較)

n	方法 C	方法 B/方法 D			
		テスト 1	テスト 2	テスト 3	テスト 4
2	6.16	7.60/6.16	5.54/5.54	8.01/6.16	7.19/5.54
3	2.78	4.35/2.76	2.21/2.21	4.66/2.74	3.22/2.48
4	1.79	7.12/1.69	1.04/1.04	3.29/1.67	1.89/1.24
5	1.22	7.36/1.03	0.435/0.435	2.40/1.10	1.10/0.593
6	0.638	10.3/0.514	0.156/0.156	1.10/0.484	0.585/0.256

表 6.13: 例題 1-アルゴリズム 6.2(分割数比較)

6.4 実験 2

6.4.1 例題

同じく [5] を参考に、Duffing 方程式の 1/3 分数調波解の決定方程式の全解探索を試みる。この決定方程式は以下のようになる。

$$
\begin{aligned}
&\left(\frac{9}{\Omega}-1\right)c_2-\frac{3\sigma}{\omega}c_3+\frac{9\varepsilon}{\Omega}\cdot\\
&(0.75{c_2}^3-0.75{c_2}^2c_4+0.75{c_3}^2c_4+0.75c_2{c_3}^2\\
&+1.5c_2{c_4}^2+1.5c_2{c_5}^2-1.5c_2c_3c_5)\\
&=0\\
&\frac{3\sigma}{\omega}c_2+\left(\frac{9}{\Omega}-1\right)c_3+\frac{9\varepsilon}{\Omega}\cdot\\
&(0.75{c_3}^3+0.75{c_2}^2c_3-0.75{c_2}^2c_5+0.75{c_3}^2c_5\\
&+1.5c_3{c_4}^2+1.5c_3{c_5}^2+1.5c_2c_3c_4)\\
&=0\\
&\left(\frac{9}{\Omega}-9\right)c_4-\frac{9\sigma}{\omega}c_5+\frac{9\varepsilon}{\Omega}\cdot\\
&(-0.25{c_2}^3+0.75{c_4}^3+1.5{c_2}^2c_4+1.5{c_3}^2c_4\\
&+0.75c_2{c_3}^2+0.75c_4{c_5}^2)\\
&=0\\
&\left(\frac{9}{\Omega}-9\right)c_5+\frac{9\sigma}{\omega}c_4-\frac{9}{\Omega}+\frac{9\varepsilon}{\Omega}\cdot\\
&(0.25{c_3}^3+0.75{c_5}^3-0.75{c_2}^2c_3+1.5{c_2}^2c_5\\
&+1.5{c_3}^2c_5+0.75{c_4}^2c_5)\\
&=0
\end{aligned}
\tag{6.3}
$$

$\sigma=2^{-5},\varepsilon=1,\omega=4,\Omega=\omega^2$ とし、解の探索範囲を $c_2\in[-3,3],c_3\in$

$[-3,3], c_4 \in [-0.3,0.3], c_5 \in [-0.3,0.3]$ とする。この方程式は 7 つの解をもつが、その中に $c_2 = c_3 = 0$ となる解があり、この探索範囲で全解探索を行うとその解は分割される領域の境界上にあることになる。このような場合アルゴリズム 6.1 による全解探索は有限ステップで停止しない。ここではアルゴリズム 6.2 のみによって全解探索を行う。また、Affine Arithmetic を用いた非存在テストは、実験 1 の結果を踏まえて方法 C、方法 D のみとする。

この例題では、x^2, x^3 という非線形単項演算のほかに $x * y$ という非線形二項演算が用いられる。このような二項演算が入った場合の非存在テストを利用した全解探索の性能をみることができるであろう。

6.4.2 実行結果

実行結果を示す。以下の結果において、$(\cdot)$ で囲まれた数値は得られた解の個数が正しい場合よりも 1 つ少なかったこと、したがって正しい結果が得られなかったことを示している。

	方法 A	方法 C	方法 D			
			テスト 1	テスト 2	テスト 3	テスト 4
時間 (s)	49.407	58.108	59.290	(942.176)	31.789	13.021
分割数	38,281	6,859	6,466	(1,080)	3,187	1,343

表 6.14: 例題 2-方法 A,C,D

6.4.3 実験 2 からの考察

実験結果から、この例題においてもニュートン写像を用いたテスト 4 が最も実行時間を短縮しており、参考扱いではあるが単体法の Phase-1 を用いたテスト 2 は実行時間は論外といえるほどかかるものの分割数を最小にしている。実験 1 で同程度の性能であった絶対値の和を用いたテスト 1 と一般逆行列を用いたテスト 3 を比べると、この実験ではテスト 3 が従来の方法より実行時間を短縮していて効果があると思われるのに対し、テスト 1 では実行時間で従来の方法を上回ることができなかった。

6.5 まとめ

本章では実際にプログラムを実装し、実行することにより非線形方程式の全解探索に Affine Arithmetic を用いた非存在領域の除去アルゴリズムを組み入れることの有効性を検証した。2 つの異なる方程式での実験では、いずれもニュートン写像を用いる 5.2.4 の非存在テストが実行時間の短縮で最も優れた性能を発揮した。その有効性は問題の規模が大きい方が顕著であった。また、単体法の Phase-1 を用いた 5.2.2 の非存在テストは非存在領域を除去する能力が最良であると思われる。ただし単体法の実装には精度保証ができておらず、さらに精度保証ができたとしても実行時間の面では実用的とはとうていいえない。しかしその実行で費やされる領域の分割回数は Affine Arithmetic を利用した全解探索における分割数の、期待される最小値となり、アルゴリズムを改良する上での目安の数値として利用することができるであろう。

また、このような非存在テストを全解探索に組み入れるときは、区間表示への還元による 5.2.0 の非存在テストを先に行うと実行時間の短縮に有効であったことをいっておかなければならない。

謝辞

本研究を行い、論文を作成するにあたり、あらゆる面で適切かつ丁寧なご指導とご助言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いただいた柏木雅英助教授に心より深く感謝いたします。

また、本研究と関わりの深い数値計算の様々な知識を講義などでご教授くださり、自分に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、本研究に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田孝富氏に深く感謝いたします。

また、数値計算が専門でないながらも、知識不十分な自分にたくさんの有用な知識と技術をくださった柏木研究室修士課程 2 年の岩折朱希嗣氏と高崎大輔氏に深く感謝いたします。

また、本研究にご助言くださるとともに日常生活でもお世話になった柏木研究室修士課程 1 年の河原淳氏、金谷卓充氏、白井健一氏、長友泰崇氏、波多野伸哉氏、深谷光統氏、吉田直史氏に深く感謝いたします。

また、ときに意見もいただきつつ、楽しい時間を過ごさせてくれた柏木研究室 4 年の黒崎俊行氏、石田寛氏、塩田孝一氏、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] 大石進一著”数値計算”, 裳華房,1999 年

[2] 大石進一著,”精度保証付き数値計算”, コロナ社,2000

[3] L.W.Swanson 著, 田畑吉雄訳,”線型計画法”, 現代数学社,1983

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

[5] 神沢雄智, 柏木雅英, 大石進一, 中村晴幸,”有限ステップで停止する非線形方程式のすべての解を精度保証付きで求めるアルゴリズム”, 電子情報通信学会論文誌:Vol.J80-A,No.7,pp.1130-1137,1997.7

[6] 柏木雅英,”Affine Arithmetic とその応用”, 日本応用数理学会年会, 平成 11 年度

[7] 川上修,”Affine Arithmetic を利用した全解探索”, 平成 11 年度卒業論文