平成16年度卒業論文

Affine Arithmeticと平均値形式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 and Mean Value Form

平成17年2月2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

1G00P0099-3　波多野 亮

目 次

図 目 次

表 目 次

第1章　序論

1.1 背景

現代の計算機による数値計算では、実数をCPUに組み込まれた浮動小数点数システムによって近似することで行っている。浮動小数点数システムについては、IEEE754Std.に準拠したものがパソコンやワークステーションで一般的に用いられており、この浮動小数点数システムを用いる事で、高性能かつ高速な演算を達成している。この高性能かつ高速な演算を達成するために、精度保証付き数値計算という分野が考え出され、近似された値と真の値の誤差がどのくらいなのか、もしくは近似解の近くに真の解が存在するのかどうか、というように近似された解の正しさを数学的に保証する技術が考えられている。

非線形方程式の解を計算機で求める時も、その技術に基づいた手法である区間演算と縮小写像原理を組み合わせた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を用いる。しかしその非線形方程式が高次元になればなるほど、その実行にかかる時間が飛躍的に長くなることが多く、時間の短縮が必要不可欠である。非線形方程式の全解探索においては、効果的な解の非存在判定、つまり解が存在しない領域を効果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事がわかっている。そこで、その方法として、変数間の相関を考慮した区間演算であるAffine演算を用いて、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ことで、効率的な全解探索を行う事が考案された。しかし、この方法は通常の区間からAffine形式に変換する手間により絶対的に有効な手段とはいえない。

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と平均値形式を応用した解の非存在テストを行い、それによってAffine変換せずに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

まず、本論文の前半で本論文のテーマである Affine Arithmetic と平均値形式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して計算機による数値計算における数値の扱い方とその誤差の問題についてと区間演算による計算結果の精度保証について述べる。

次に、3 章では非線形方程式の全解探索に関す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説明する。

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。

これらをふまえて、本論文の後半にて Affine Arithmetic と平均値形式を利用することで、実際に非線形方程式の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 を利用した従来の方法の原理と問題点と、その問題点を考慮に入れて、平均値形式と Affine 形式を利用する新手法の両方の原理について述べる。

最後に 6 章においてそのアルゴリズムを実装し、実際に適当な非線形方程式を例題としてその全解探索を行い、実行時間や探索領域の分割回数を測定して Affine Arithmetic と平均値形式を利用した非存在領域判定アルゴリズムの有効性を検証する。

第2章　区間演算

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値の表現について述べ、それをふまえて計算機を用いて連続数学の問題を解く場合の数値の扱いと誤差の問題について説明する。その次に、数値の拡張として区間という概念を用いて解を包み込むことで精度を保証する区間演算という考え方について説明する。なおその際、区間演算を浮動小数点数システムに展開した機械区間演算についても簡単に説明する。

2.2 計算機上の数値

計算機では実数を浮動小数点数によって近似することで計算する。浮動小数点数システムについてはIEEE754Std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。またここでは倍精度浮動小数点数のみを扱う。

2.2.1 数値の表現方法

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。IEEE754では正規化数の場合、仮数部の先頭は必ず1に固定されているので、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。つまり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。IEEE754では正規化数以外に、

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

- NaN　$(e = 2047, m \neq 0)$

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のような時、NaN は $\sqrt{-5}$ や ∞/∞ のような不当な演算の結果の時などで得られる。

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 より小さい。

2.2.2　丸め

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。そのため、任意の実数を表現することはできない。こ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 $(\Diamond)$　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 ではこの丸めの方向が指定でき、以下の性質を満たすことが要請されている。$\bigcirc = \{\triangle, \triangledown, \Diamond\}$、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 進数と 2 進数の相互変換

計算機による数値計算にとって、10 進数と 2 進数の相互変換は必要不可欠である。なぜなら人間にとって、10 進数は日常的に用いられており都合がいい (指の本数と同じであるため) が、CPU にとっては 2 進数のほうが都合がいい (電気が通っているかどうかに対応させやすいため) からである。そのため 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。例えば 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。この時 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、2 進数を 10 進数に変換するが、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モード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

前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真の値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という概念について、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。

$$\begin{aligned}
\mathrm{wid}([x]) &= \overline{x} - \underline{x} && (2.8)\\
\mathrm{mid}([x]) &= \frac{\overline{x} + \underline{x}}{2} && (2.9)\\
\mathrm{rid}([x]) &= \frac{\overline{x} - \underline{x}}{2} && (2.10)
\end{aligned}$$

$\mathrm{mid}([x]), \mathrm{rid}([x])$ 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、

$$[x] \cdot [y] = \{x \cdot y \in \mathbf{R} | x \in [x], y \in [y]\} \tag{2.11}$$

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、

$$\begin{aligned}
[x] + [y] &= [\underline{x} + \underline{y}, \overline{x} + \overline{y}] && (2.12)\\
[x] - [y] &= [\underline{x} - \overline{y}, \overline{x} - \underline{y}] && (2.13)\\
[x] \times [y] &= [\min\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}, \max\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}] && (2.14)\\
[x]/[y] &= [x] \times [1/\overline{y}, 1/\underline{y}] && (2.15)
\end{aligned}$$

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。

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

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

次に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。

$$f([x]) = \{f(x) | x \in [x]\} \tag{2.16}$$

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、

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

以上を基にして、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と、このような関数の評価を厳密に行うのは困難な場合があることがわかる。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、

$$f([x]) \subseteq f_{[\,]}([x]) \tag{2.17}$$

が成り立つ。なお f を同値な別の形式の式に置きかえたとき f の区間拡張は一般に異なる。また一般に関数の値域 $f([a,b)$ を $f([a,b]) \subset [c,d]$ となる区間 $[c,d]$ で評価する時、区間 $[c,d]$ のことを $f([a,b])$ の区間包囲という。区間拡張は区間包囲の一種である。

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。

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

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

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

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

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

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

(2.17) 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、得られる区間の幅が約 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。

2.4　機械区間演算

ここで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、求めたい実数解の真の値が幅の狭い区間の中に入ることを目指す。

2.4.1 機械区間演算における区間の定義

区間 $[x]$ を、

$$[\underline{x}, \overline{x}] = \{x \in \mathbf{R} | \underline{x} \leq x \leq \overline{x}\} \tag{2.18}$$

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。つまり、機械区間演算における区間とは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、

$$[x] \subseteq \bigcirc[x] \quad ([x] \in I\mathbf{R}) \tag{2.19}$$

$$\bigcirc[x] = [x] \quad ([x] \in IF) \tag{2.20}$$

$$[x] \subseteq [y] \Rightarrow \bigcirc[x] \subseteq \bigcirc[y] \quad ([x], [y] \in I\mathbf{R}) \tag{2.21}$$

$$\bigcirc(-[x]) = -(\bigcirc[x]) \quad ([x] \in I\mathbf{R}) \tag{2.22}$$

をそれぞれ任意の $[x]$,$[y]$ について満たすようにする。IEEE754 における浮動小数点数システムの時、

$$\bigcirc[x] = [\triangledown\underline{x}, \triangle\overline{x}] \tag{2.23}$$

とすれば上の条件は満たされる。

2.4.2 機械区間演算における四則演算

$I\mathbf{R}$ での四則演算をもとに、機械区間演算における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

$$[x] \bigodot [y] = \bigcirc([x] \cdot [y]) \tag{2.24}$$

で定義する。(2.23) から、具体的には次のようにすればよい。

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

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、丸めモードを下端の計算を下向きで、上端の計算を上向きにして行えばよい。

2.5 おわりに

本章では計算機における数値の取り扱いとそれを基にした区間演算について述べた。計算機を用いた数値計算では、実数を浮動小数点数で近似しなければならない。この時必ず誤差が生じてしまうが、CPU の丸めモード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における誤差を把握することは可能である。この区間演算を応用することでより高精度な数値計算を達成することができる。(区間演算について詳しくは [1]、[2])

第3章　Krawczykの方法を利用した非線形方程式の全解探索

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説明する。まず区間演算と不動点定理を組み合わせたKrawczykの方法について説明する。そしてそれをもとにした、与えられた探索領域における解の存在あるいは非存在を判定する方法と、その方法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問題の基本

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

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時、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、その近似解と真の解の誤差や、真の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、非線形方程式の求解問題は基本的に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である。

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。
この時、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

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

であれば、

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

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、

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

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

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

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与える。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、再帰的にテストを繰り返すアルゴリズムが知られている（図 3.1）。

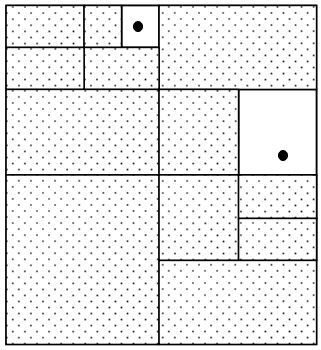

図 3.1: 全解探索のようす

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大体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、解の非存在判定の効率化については課題が残っている。4 章で Affine Arithmetic について説明し、5 章で Affine Arithmetic を利用した全解探索の効率向上の方法とその問題点、その問題を考慮した新手法について説明する。

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、よりシャープな結果を得る事が出来る。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まず説明し、その後に簡単な例題によって通常の区間演算とAffine Arithmeticの違いについて実際に確かめてみる。

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

ここでは、具体的に簡単な例を使うことで Affine Arithmetic と区間演算の違いを実際に見てみることにする。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。

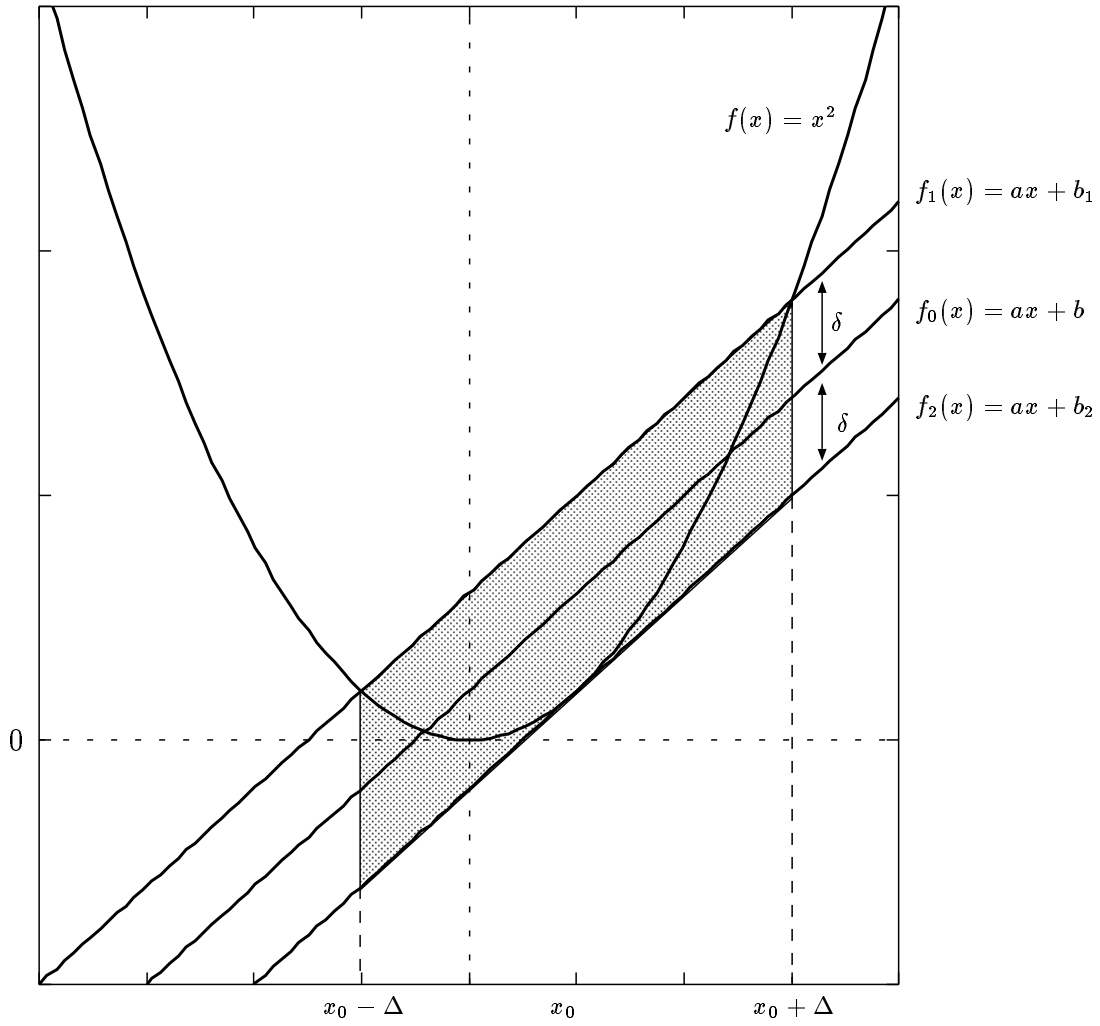

図 4.1: x^2 の包含-式 (4.4) により Δ を求める。このとき、$a = 2x_0, b_1 = -{x_0}^2 + \Delta^2, b_2 = -{x_0}^2$ であり、$b = \frac{1}{2}(b_1 + b_2) = -{x_0}^2 + \frac{1}{2}\Delta^2, \delta = \frac{1}{2}\Delta^2$ が得られる。

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）。

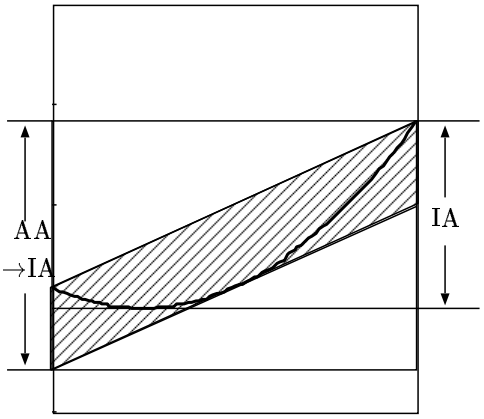

図 4.2: Affine 形式の区間演算への還元

4.4 おわりに

本章では Affine Arithmeticについてと、その有効性を簡単な例を用いた通常の区間演算との比較から説明した。Affine Arithmetic では変数間の相関関係が考慮され演算が行われる。これによって、よりシャープな演算結果が得られる。5 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の原理と問題点とその問題点を考慮にいれて改良した新たな非存在判定テストについて説明する。

第5章　Affine Arithmeticを利用した解の非存在判定法

5.1 はじめに

本章ではまず 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、その原理と問題点を説明する。次にその問題点を考慮に入れて、Affine Arithmetic と平均値形式を利用した非存在判定法について説明する。この方法を用いて、非線形方程式の全解探索の改良を試みるのが本論文の目標である。

5.2 Affine Arithmeticを利用した非存在判定法

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} と書くことにして、

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

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 最小二乗解を用いる方法

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

$$\begin{array}{c} a_{10} + a_{11}\varepsilon_1 + \cdots + a_{1n}\varepsilon_n = 0 \\ a_{20} + a_{21}\varepsilon_1 + \cdots + a_{2n}\varepsilon_n = 0 \\ \vdots \\ a_{m0} + a_{m1}\varepsilon_1 + \cdots + a_{mn}\varepsilon_n = 0 \end{array} \tag{5.3}$$

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 (5.3) は、

$$A\varepsilon = b \tag{5.4}$$

とかける。ダミー変数の条件により、

$$-1 \leq \varepsilon_i \leq 1 \quad (i = 1, 2, \cdots, n) \tag{5.5}$$

で表される超立方体と方程式 (5.3) 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 5.1）。この交わりが空かどうかを判定する。そこで (5.5) 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 5.2）。この方法では (5.3) の解でユークリッドノルム最小のものを得て、そのノルムが $\sqrt{n}$ より大きいならば (5.3) の解空間と (5.5) 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**??**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 (5.4) の最小ノルム解は

$$A^t(AA^t)^{-1}b \tag{5.6}$$

で得ることができる。

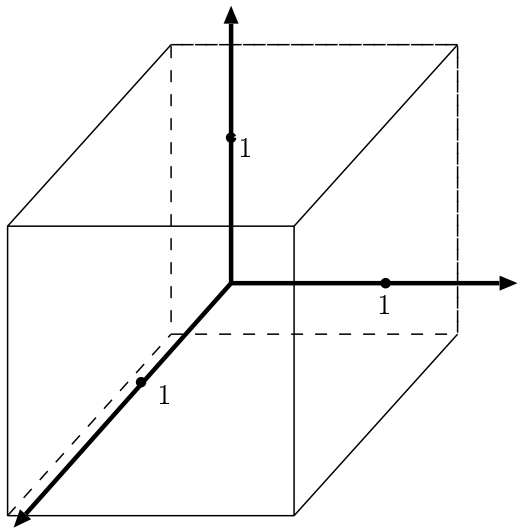

図 5.1: 超立方体 $-1 \leq \varepsilon_i \leq 1$

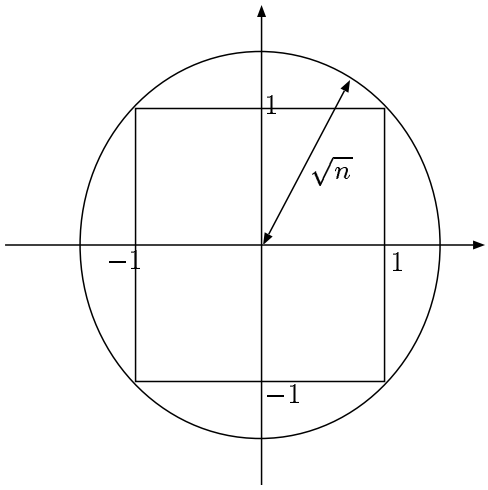

図 5.2: 超立方体と超球

5.3 Affine Arithmeticを利用した非存在判定法の問題点

前述の通り、Affine Arithmeticを利用した非存在判定法を用いると、変数間を考慮した演算を可能にするというAffine Arithmeticの特性上、よりシャープな結果が得られるので、より効率的な非存在領域の除去を達成することができる。これにより非線形方程式の全解探索の実行時間の短縮を可能にしている。

しかし、その一方で与えられた非線形方程式をAffine形式に変換するのに要する手間により、実行時間が増加してしまう。これにより、必ずしも最適な解の非存在判定法とはいえない。

5.4 Affine Arithmeticと平均値形式を利用した非存在判定法

Affine Arithmeticを利用した非存在判定法は、前述の通りAffine形式に変換する手間により、必ずしも最適な非存在判定法とはいえなかった。そこで、それを考慮した新手法を本節にて説明する。

新手法はAffine形式に変換しないことで、前述の問題点を解消した上で、Affine Arithmeticのメリットを利用する、というものである。

具体的には以下のような手順で行う。

1. まず、最初に与えられた非線形方程式を平均値形式(3.7)にする。
2. 次にその平均値形式にされた式の中の区間Iを擬似的にAffine形式に変換する。
3. その式に対して、前述したAffine Arithmeticを利用した非存在判定法(5.2節)を用いることで解の非存在を判定する。

この手法はあくまで擬似的なAffine変換なので、Affine変換による実行時間の増大を抑えつつ、Affine Arithmeticの恩恵を受けることができ、シャープな演算結果が得られるので、効果的な非存在領域の除去、ひいては非線形方程式の全解探索の性能向上が期待できる。

5.5 おわりに

本章では Affine Arithmetic による評価を利用した非線形方程式の解の非存在領域の除去法の原理と問題点とその問題点を考慮した新手法について述べた。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、今回実装する新手法が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と平均値形式を利用した非存在判定を組み入れた新しい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と平均値形式を利用した非存在判定法の有効性を検証する。

6.2 実験にあたって

6.2.1 プログラムについて

プログラミング言語にC++を用いて6.4節の方程式の全解探索を行う。

6.2.2 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 実験で用いる全解探索アルゴリズム

6.3.1 準備

まず準備として、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必要な従来のアルゴリズムについて説明する。

まず従来の区間演算のみを用いた解の非存在判定法である。

アルゴリズム 6.1 (区間演算を用いた非存在判定)

1. $F(X)$ を求める。
2. 少なくとも一つの $i(i = 1, \cdots, m)$ に対して

$$0 \notin F_i(X) \tag{6.1}$$

非存在判定は成功となる。そうでなければ失敗である。

次に、区間演算のみを用いた解の存在判定法である。

アルゴリズム 6.2 (区間演算を利用した存在判定 (クラフチックの方法))

1. $F'(X)$ を求め、これを利用して

$$Y \approx (m(F'(X)))^{-1} \tag{6.2}$$

を求める。
2. E を m × m 単位行列、m × m 区間行列 R を

$$R = E - YF'(X) \tag{6.3}$$

とするとき、R の最大値ノルムを求める。
3. Y, R を用いて

$$K(X) = m(X) - Yf(m(X)) + R(X - m(X)) \tag{6.4}$$

を求める。
4.

$$X \cap K(X) = \emptyset \tag{6.5}$$

の時、解の非存在判定が成功するので判定を終了させる。そうでなけれ ば *5* へ。
5.

$$K(X) \subseteq X \tag{6.6}$$

が成立する時、X 内に解が存在するので、*6* へ進む。そうでなければ失敗となり判定を終了する。
6. R の最大値ノルムが *1* より小さければ、X 内に存在する解は一意である ので、存在判定は成功となる。そうでなければ *7* へ進む。
7. 探索領域を X から $K(X)$ に縮小させて、判定を終了する。

最後に最小二乗解を用いた非存在判定法である。Affine Arithmeticを利用した非存在判定法はいくつか提案されているが、今回の実験では、5章で説明した最小二乗解を用いた方法を採用することにする。

アルゴリズム 6.3 (最小二乗解を用いた非存在判定)

1. AA^T を計算する。
2. AA^T の逆行列 C を求める。
3. Cb を求める。ただし、計算結果を C_{b} とする。
4. $A^T C_b$ を求める。
5. *4* で求めた最小二乗解のユークリッドノルムを二乗したものを方程式の数 を二乗したものと比べて、大きかったら非存在判定は成功となる。そう でなければ非存在判定は失敗となる。

6.3.2 実装する全解探索アルゴリズム

6.3.1 節で説明したアルゴリズムを踏まえて、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説明する。

アルゴリズム 6.4 (Affine Arithmeticと平均値形式を利用した全解探索)

1. 領域のリスト S、T を $S = \{ X^{(0)} \}$、$T = \emptyset$ で初期化する 。
2. リスト S が空なら終了、さもなくば S 内の領域で最大辺をとるような超 直方体領域 X を探す。X を探索領域とし、S から取り出す。(S から X を削除する。)
3. 区間演算を用いた解の非存在判定 (*6.1*) を行う。 成功したら *2* へ戻る。失敗したら *4* へ。
4. 与えられた方程式を平均値形式にし、疑似アフィン変換する。
5. 最小二乗解を用いた解の非存在判定 (*6.3*) を行う。成功 したら、X を捨てて、*2* へ戻る。失敗したら、*6* へ進む。
6. クラフチックの方法 (*6.2*) を用いて、探索領域 X における解の存在判定を行う。
7. *6.2* における *5* の解の存在を保証する十分条件が成立 したら、*8* へ進む。成立しなかったら、X をその最大辺で二等分し、こ れにより作成された二つの領域をリスト S に挿入し、*2* へ戻る。
8. *6.2* の *6* における解の一意性を保証する十分条件が成 立したら存在判定は成功となるので、探索領域 X をリスト T に挿入し 、*2* へ戻る。成立しなかったら *9* へと進む。

9. *6.2* の *7* により X を縮小させる。この縮小させた領域 をその最大辺で二等分し、二等分することで作成された二つの領域をリ スト S に挿入し、*2* へ戻る。

以上のアルゴリズムと擬似アフィン変換せずに従来の区間演算による判定法のみを実装したアルゴリズムを比較して実験を行う。なお本論文では便宜上、前者を方法 A、後者を方法 B と呼ぶことにする。また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探索された領域の数を測定する。

6.4 実験

6.4.1 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。

$$\begin{aligned} g(x_1) + x_1 + x_2 + \cdots + x_n &= 1 \\ g(x_2) + x_1 + x_2 + \cdots + x_n &= 2 \\ &\vdots \\ g(x_n) + x_1 + x_2 + \cdots + x_n &= n \end{aligned} \tag{6.7}$$

ここに、ダイオードの特性を

$$g(x) = 2.5x^3 - 10.5x^2 + 11.8x \tag{6.8}$$

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、平均値形式を擬似 Affine 変換して非存在判定を行う全解探索と擬似 Affine 変換せずに判定を行う方法の性能の比が、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。

n	解の個数
2	1
3	1
4	3
5	5

表 6.1: 各次元における解の個数

6.4.2 実行結果

以下の表が実行結果をまとめたものである。それぞれ実行時間についてと探索領域数である。

n	方法 A	方法 B
2	0.01	0.01
3	0.13	0.43
4	1.261	8.031
5	71.402	1,860.7

表 6.2: 方法 A,B(実行時間、単位:s)

n	方法 A	方法 B
2	158	464
3	1,168	4,606
4	9,966	41,606
5	73,828	326,866

表 6.3: 方法 A,B(探索領域数)

6.4.3 実験の考察

まず実行時間を比較してみると、次元が小さい時はその差はあまりないが次元がだんだん大きくなっていくにつれて、実行時間に大幅な差が出てくるのがわかる。

次に探索領域数を比較してみると、これも同様に、次元が小さい時は差はあまりないが、次元がだんだん大きくなっていくにつれて、探索領域数に大幅な差が出てくるのがわかる。

これは擬似 Affine 変換する方法は最小二乗解を利用した判定法を用いることができるようになるためにできる差であると思われる。

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平均値形式の擬似 Affine 変換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。従来の区間演算による判定法を用いる方法と比べると、実行時間、探索領域数ともに少なくなっており、全解探索の効率化が達成されている。特に次元が増すごとにより効率的である。

この実験から擬似 Affine 変換によってある程度の性能向上が見込めるということがわかった。しかし当然これは最適とはいえないので今後さらに改良することでより良い全解探索を達成することが今後の課題である。また今回用いた最小二乗解を利用する方法以外の方法 (線形計画法を用いる方法など) についても、本論文における新手法に基づいたアルゴリズムを実装し、比較検討することも必要である。

6.6 おわりに

本論文の内容は以上だが、最後に本論文作成にあたってお世話になった方への謝辞と参考文献を記して、本論文の終わりとする。

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島信也氏に深く感謝いたします。

また、本論文作成にあたって適切にご助言をくださった柏木研究室博士課程3年の濱田吉信氏に深く感謝いたします。

また、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年の石塚昭史氏、円田直氏、北村健志氏、中桐康佳氏、横関鉄平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年の金子大樹氏、小糸彰氏、河野大祐氏、冨永雄一氏、大和資治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年の伊志嶺寛之氏、麻生香雄氏、岩木利幸氏、高橋玄氏、寺田洋介氏、山下亮輔氏、木原剛史氏、具志頭大輔氏、玉井純氏、辻祐介氏、磨矢龍二氏、野本侑希氏、花井正弘氏、水口義雄氏、宮田和浩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

[8] 菊地智行,"Affine Arithmetic を利用した非線形方程式の解の非存在領域の除去", 平成 13 年度卒業論文