区間解析における関数の値域の評価に関する研究

On Range Evaluation of Functions Applying Interval Analysis

2005年3月

早稲田大学大学院理工学研究科
情報科学専攻 数値解析研究

宮島 信也

目 次

図 目 次

表 目 次

第 1 章

序論

1.1 背景

理工学において，積分や方程式の求解などの連続数学の問題を解くことに対する需要は非常に高い．しかしそれらの問題は数式処理等による，いわゆる解析的な手法で解くことが困難であるため，計算機上の離散的かつ有限精度の計算に置き換えて，すなわち数値的に解かれている．例えば昨今非線形現象を伴う回路や各種システムに対する解析がさかんに行われるようになったが，その要因のひとつとして数値シミュレーションを行うための計算機のハードウェアの性能が飛躍的に向上した点が挙げられる．以上のことから，計算機を援用した数値計算は理工学全体を通じていまや必須の存在であると言ってよい．

現在の電子計算機による数値計算は浮動小数点演算を基礎としている．浮動小数点演算では，実数を計算機上に表現する段階で有限桁の浮動小数点数に近似するため，メモリ上に占める領域が一定のまま絶対値の大きい数値を扱える反面，その計算結果には必ず丸め誤差が混入してしまう．そのため，ベクトルの内積計算や算術式の評価，連立一次方程式の直接法による求解などの有限回の代数演算で結果が求まるプロセスにおいてさえも，計算式と数値によってはわずか数ステップの計算で有効数字が完全に失われてしまうことがある．ここでは，計算機の丸め誤差の怖さを端的に示す例として次のような算術式の評価を紹介する:

例題 1.1 2 進 53 桁の仮数部を持つ倍精度浮動小数点演算において
$x = 192119201, y = 35675640$ として，

$$\text{式} : z = \frac{1682xy^4 + 3x^3 + 29xy^2 - 2x^5 + 832}{107751}$$

を評価すると，真の値 $z = 1783$ に対して計算結果は $\tilde{z} = 7.18056\cdots \times 10^{20}$ となる．

また，微分方程式をはじめとする非線形関数方程式を数値的に解く場合，これを有限次元非線形方程式に近似する過程で離散化誤差が混入するし，一般の非線形問題は Newton 法などの反復法によって解かれるが，その場合は未収束による打ち切り誤差なども混入する．以下の災害はこのような数値計算上の誤差が原因であることが分かっている．

- 1991 年 2 月 25 日，サウジアラビアの Dharan 市で発生したパトリオットミサイルの欠陥による 28 人の死亡 (例えば [1][2])
- 1991 年 8 月 23 日，ノルウェーの Stavanger 市近郊の Gandsfjorden で発生した Sleipner A 海上施設の沈没．これにより 10 億ドル近くの損害が生じた (例えば [3])
- 1996 年 6 月 4 日，フランス領ギアナからの初飛行で打ち上げ直後に発生した Ariane 5 ロケットの爆発 (例えば [4])

実は，計算機を用いた数値計算の抱えるこのような問題点は，1945 年にハンガリー出身の数学者である John von Neumann が今日の計算機の基礎を築いた時点で既に指摘されて

いたことである．その指摘を受ける形で，実際に数値計算を行うためのアルゴリズムはその安定性や収束性が常に意識され，誤差の影響をなるべく受けないで済むように設計されてきた．ただし，それらのアルゴリズムは得られた結果が数学的にどの程度まで正しい結果であるかを保証するものではなく，また浮動小数点演算によってそのような保証を与えるのは技術的に難しいことでもあった．

このような問題に対し，近年，丸め誤差を含む浮動小数点演算においても演算結果の数学的な正しさを実用的なレベルで保証できることが明らかにされ [5]，数値計算の精度を保証するという観点から数値計算法全体を見直す動きが世界的規模で始まった．こうして生まれたのが精度保証付き数値計算 (numerical calculation with guaranteed accuracy)，または自己検証的算法 (self-validating numerics) と呼ばれる数値計算法であり，今後の数値計算法のあるべき一つの方向として大いに注目されている [6]-[19]．

精度保証付き数値計算の根幹をなす重要な技術の一つに区間演算が挙げられる．これは計算機上における実数を「ある幅を持った実数の集合 (区間)」として表し，区間同士の演算を一種の集合算として定義するものである．区間演算は歴史的には須永照雄 [20] $(1929-1995)$ 等によって 1950 年代に提唱された．データ型としての区間は具体的には [下端, 上端] の形を取り，例えば円周率を表す無理数 π であれば $[3.14159, 3.14160]$ のような区間で表すことができる．このように，2 つの浮動小数点数のなす区間によって実数を挟み込み，かつその区間幅によって誤差評価をも同時に行う．実数のこのような表現自体は数学基礎論における構成的実数論 [21] に端を発しているのだが，須永の区間演算はそこから区間同士の演算を「演算結果としてあり得る集合を包含するように」定義したものであり，区間演算を用いた数値解析 (区間解析) の先駆けとも言うべきものである．このような須永の功績については文献 [22] に詳述されており，彼の思想は P．S．Dwyer [23]，R．E．Moore[24][25] らによって受け継がれた．ここで区間解析の応用例及び区間解析によりもたらされた成果を以下に示す．

- ケプラー予想の解明 (例えば [26])
- コンピュータグラフィック及びコンピュータ支援設計における空間領域構成法の表示 (例えば [27])
- ロボット制御 (例えば [28][29])
- 重力係数 G の測定 (例えば [30])
- ダブルバブル最小化問題の解決 (例えば [31])

区間演算の大きな欠点として，計算された区間は真の解を確かに含むものの，区間幅の爆発的な増大が起こることが挙げられる．これはとりわけ，はじめから幅を持った区間を入力として関数の値域を評価する場合など，比較的幅の広い区間を扱う際に顕在化する問題であり，そのような例を次に示す:

例題 1.2 関数 $f(x) = x^2 - 2x$ の値域を区間 $[0.9, 1.1]$ で評価することを考える．区間演算の定義 (第 2 章で述べる) に従って計算した結果が $[-1.39, -0.59]$ であるのに対して，真の像 $\{f(x) \mid x \in [0.9, 1.1]\}$ は $[-1, -0.99]$ である．

上記の例では区間演算に従って計算した結果が真の像の実に 80 倍の区間の開きになっていることが分かる．この区間幅の爆発的な増大の原因は二つの関数 x^2 と $2x$ が $[0.9, 1.1]$ の区間では同じ x の関数で互いに相関があり，しかも区間内で傾きが非常に近い値をとりそれの減算をするためであり，区間演算ではこの相関性を無視して独立な値として計算しているため上記のような区間幅の爆発的な増大が起こってしまうのである．Krawczyk 法を用いた非線形方程式の全解探索アルゴリズム [32][33] や区間解析による多変数関数の最大値探索法 [34][35][36] が多大な計算時間を必要とするといった問題，また非線形常微分方程式の初期値問題の長時間積分 [37] は困難であるといった問題の原因の一つにはこの区間幅の爆発的な増大が挙げられる．

以上より区間演算における上記の問題を解決する，すなわち区間演算において区間幅の爆発的な増大を抑制し関数の値域のシャープな包含を得ることは精度保証付き数値計算において重要な課題である．

1.2 本論文の目的

1.1 節の最後に示した課題に基づき，本論文は区間演算において区間幅の爆発的な増大を抑制し関数の値域をシャープに包含する方法について考察を与えるものである．本論文は全体を大きく前半 · 後半の 2 つに分け，それぞれ異なる視点から 1.1 節の最後に示した課題の達成を目指す．

最初に本論文の前半 (第 3 章，第 4 章，第 5 章) における研究目的について説明する．区間演算で起こる区間幅の爆発的な増大を解決するために様々な手法が提案されているが (詳細は第 2 章で述べる)，まずその方法の一つで，区間演算の有力な拡張であるアフィン演算に注目する．アフィン演算は 1994 年にブラジル・Campinas 大学の Jorge Stolfi ら [38] によって初めて提案され，1997 年に Luiz H. de Figueiredo の LNCC (ブラジル国立科学計算研究所) 学位論文 [39] によって体系的にまとめられた区間演算の一種である．区間演算では，区間同士の相関が全く考慮されていないのに対し，アフィン演算ではアフィン形式という特殊な形式を用いることで，区間の上限・下限のみならず変数間相互の相関関係をも表現できる．変数間の相関関係を表現できることで，計算を進めていくうちに減算で起きるべき打ち消しが起きるし，同類項もまとめることもできる．これによりアフィン演算は区間演算における区間幅の爆発的な増大を抑制し，通常の区間演算よりも関数の値域をシャープに包含することが多いという特徴を持つ．この特徴により，アフィン演算は値域を評価する関数が複雑な場合や，値域を評価する関数のそれぞれの項同士において変数間の相関が強い場合等に特にその効果を発揮する．例えば上記の例をアフィン演算で計算すると $[-1.0, -0.99]$ となり，真の像と一致する．

一方で，アフィン演算はその非線形演算に関する取り扱いが極めて難しく，現状でなお多くの課題を残している．そのため，アフィン演算は非線形方程式の全解探索問題における解の非存在領域の除去 [40] や解の存在領域の検証 [41][42] 等に大きな威力を発揮するにもかかわらず，多くの研究者の間に浸透しているとは言い難い状況である．そこで，本論文の前半では区間演算の新しい局面を切り拓く可能性を秘めたアフィン演算を整備 · 補完することによって，アフィン演算の実用化に大きく貢献することを目的とする．

アフィン演算における非線形演算は真の像そのものではなく，線形近似と新しい誤差項の追加により真の像を包含し，その包含を演算結果としているのだが，そこで新しく追加される誤差項の係数は一般にはその後の演算で打ち消しが起きず，区間評価の性能を落とす (区間幅の増大を引き起こす) 原因となる．アフィン演算では主な非線形単項演算 ($\sqrt{x}$, $1/x$ など) は既にその最良の計算方法 (新しく追加される誤差項の係数の絶対値を理論的に最小に抑える計算方法) が発見されている．

一方，Jorge Stolfi，Luiz H. de Figueiredo らにより定義されたアフィン形式同士の乗算 (Stolfi の乗法) は，簡便ではあるけれど変数 x, y 間に相関がある場合に新しく追加される誤差項の見積もりが甘く，その結果真の像 ($f(x, y) = x \times y$) の包含をシャープに行っていないため，区間評価の性能をかなり落としていた．そこで宮田孝富，柏木雅英 [43] によりアフィン形式同士の乗算の改良版 (改良乗算) が 2000 年に提案された．この方法は誤差項同士の積を計算する際に変数間の相関をある程度考慮するような工夫を施すことで，Stolfi の乗法よりも新しく追加される誤差項の係数の絶対値を小さくすることに成功している．しかしながらその新しく追加された誤差項の係数の絶対値は理論的に最小に抑えられたものではなく，改良の余地 (その新しく追加された誤差項の係数の絶対値をさらに小さくすることが理論上可能である) を残している．

そこでアフィン演算の乗算において新しく追加される誤差項の係数の絶対値を理論的に最小に抑える計算方法 (最良乗算) を提案する．最良乗算では非線形二変数関数 (曲面) $f(x, y) = x \times y$ を平面で線形近似する際に，その平面の傾きは Stolfi の乗法や改良乗算と同様に見積もられるものの，その平面の切片と (曲面と平面との) 誤差の見積もり方に対し工夫が施されており，Stolfi の乗法や改良乗算よりも常に新しく追加される誤差項の係数の絶対値を同等以下にすることができる．最良乗算を提案した後にこの最良乗算を実用時間で実行する方法を提案し，さらにいくつかの数値例により最良乗算の有効性を確認する．

アフィン形式同士の除算については文献 [38][39] では定義されておらず，従来は $x/y = x \times (1/y)$ のように，逆数をとる非線形単項演算と Stolfi の乗法の合成として計算していた (文献 [38][39] の方法)．しかしこのように除算を 2 つの非線形演算に分解すると，1 回の演算で誤差項が 2 つ新しく追加されてしまい，さらに Stolfi の乗法では誤差項の見積もりが甘いため，区間評価の性能を落としてしまう．そこで柏木雅英 [44][45] は除算 x/y を分解せずに直接計算する方法を 1999 年に提案した (柏木の方法)．この方法は非線形二変数関数 $f(x, y) = x/y$ に対して直接線形近似を行う方法で，計算時間を優先して，複雑な場合分けが必要な部分を近似することにより，計算を一部簡略化している．この柏木の方法における複雑な場合分けが必要な部分を緻密に計算した方法が白井健一 [46][47](柏木雅英研究室修士) により 2000 年に提案された (白井の方法)．この方法は変数 x, y 間に相関が無い場合であれば最適な評価を与える (新しく追加される誤差項の係数の絶対値を理論的に最小に

抑えることができる) が，変数 $x,\ y$ 間に相関がある場合には最適な評価を与えることはできない．ゆえに変数 x, y 間に相関がある場合に白井の方法よりも新しく追加される誤差項の係数の絶対値を小さくすることができる方法は理論上はあると考えられるが，これまでは知られていなかった．

そこでアフィン演算の除算において 3 つの方法を提案する (以後これら 3 つの方法を提案手法 1，提案手法 2，提案手法 3 と呼ぶ)．提案手法 1，提案手法 2 は文献 [38][39] の方法の様に除算を逆数をとる非線形単項演算と乗算の合成として計算する際に，行われる乗算を先に述べた改良乗算，最良乗算に変更したものである．これらの方法は 1 回の演算で誤差項が 2 つ新しく追加されてしまうものの，変数 $x,\ y$ 間の相関が強い場合には改良乗算，最良乗算が効果を発揮するため白井の方法よりも新しく追加される誤差項の係数の絶対値を小さくすることが期待できる．提案手法 3 は白井の方法を改良したものである．提案手法 3 では非線形二変数関数 (曲面) $f(x, y) = x/y$ を平面で線形近似する際に，その平面の傾きは白井の方法と同様に見積もられるものの，その平面の切片と (曲面と平面との) 誤差の見積もり方に対し工夫が施されており，白井の方法よりも常に新しく追加される誤差項の係数の絶対値を同等以下にすることができる．これら 3 つの方法を提案した後にいくつかの数値例によりアフィン演算における除算の全ての方法を比較・検証し，提案手法の有効性を示す．

次に本論文の後半 (第 6 章) における研究目的について説明する．値域を評価する関数を多項式に限定すると，多項式の性質を生かすことにより，区間演算で起こる区間幅の爆発的な増大を抑制し関数の値域のシャープな包含を得るための効率的な方法が提案できる．多項式の値域をシャープに包含することができると，例えば，多くの部分が多項式により構成されている非線形多変数関数の大域的最適化や多くの部分が多項式により構成されている非線形連立方程式の全解探索アルゴリズムの性能を速度面で向上させることができる．そこで本論文の後半では多項式の値域のシャープな包含を得る方法を提案することにより，多項式の値域の評価が多数回必要となる例題に対して，上記のようなアルゴリズムの性能向上に貢献することを目的とする．多項式の値域の評価において，区間演算で起こる区間幅の爆発的な増大を抑制する方法として，これまでに Horner 法 (例えば [18][48])，平均値形式 (例えば [48]) に基づく方法，Alefeld の方法 [49][50]，アフィン演算を使用する方法 [51] が提案されている．

本論文の後半では多項式の値域の評価方法として，5 つの方法を提案する (以後これら 5 つの方法を提案手法 1，提案手法 2，提案手法 3，提案手法 4，提案手法 5 と呼ぶ)．提案手法 1 は与えられた多項式を区間の中心に関するべきの形式に変形してから値域を評価する方法であり，提案手法 2 は提案手法 1 により区間の中心に関するべきの形式に変形された多項式をさらに変形した後，Horner 法を適用する方法である．提案手法 3 は区間の中心に関するべきの形式を二次式ずつ括弧でくくり，くくられた括弧内の最大値，最小値を繰り返し厳密に見積もっていく方法である．また，提案手法 4 では与えられた多項式をその導関数で割ったときの剰余を利用して値域を評価する方法であり，提案手法 5 では与えられた多項式と，その導関数を用いて作成された別の多項式との差を利用して値域を評価する方法である．これらの提案手法を用いて多項式の値域を評価することにより，Horner 法，平均値形式に基づく方法，Alefeld の方法，アフィン演算を使用する方法よりも多項式の値

域のシャープな包含を得ることが期待できる．これらの手法を提案した後，例によっていくつかの数値例により，多項式の値域を評価する全ての方法を比較·検証し，提案手法の有効性を示す．

1.3 本論文の構成

本論文は次のような構成をとる．

第 2 章では本論文における議論の基礎となり，精度保証付き数値計算の根幹をなす重要な技術の一つである区間演算について解説する．区間演算は計算機上における実数を「ある幅を持った実数の集合 (区間)」として表し，区間同士の演算を一種の集合算として定義するものである．区間解析は歴史的には須永照雄らによって 1950 年代に提唱された．区間演算では [下端 , 上端] の 2 つの浮動小数点数のなす区間によって実数を挟み込み，かつその区間幅によって誤差評価をも同時に行う．区間演算の大きな欠点として，計算された区間は真の解を確かに含むものの，区間幅の爆発的な増大が起こることが挙げられる．この区間幅の爆発的は区間演算では変数間の相関性を無視して独立な値として計算しているために起こる．

本章ではまず，計算機上における実数の表現について述べ，区間演算をその用語とともに定義する．次に区間演算の問題点である区間幅の爆発的な増大について説明する．最後にこれまでに提案されてきた区間幅の爆発的な増大を抑制する方法について述べる．

第 3 章では区間演算の有力な拡張であるアフィン演算の定義等について述べる．アフィン演算は Jorge Stolfi，Luiz H. de Figueiredo ら [38][39] によって 1994 年に提案された区間演算の一種である．区間演算では，区間同士の相関が全く考慮されていないのに対し，アフィン演算ではアフィン形式という特殊な形式を用いることで，区間同士の相関を考慮し，区間演算における区間幅の爆発的な増大を抑制する．一方で，アフィン演算はその非線形演算に関する取り扱いが極めて難しく，現状でなお多くの課題を残している．アフィン演算における非線形演算は真の像そのものではなく，線形近似と新しい誤差項の追加により真の像を包含し，その包含を演算結果としているのだが，そこで新しく追加される誤差項の係数は一般にはその後の演算で打ち消しが起きず，区間評価の性能を落とす原因となる．アフィン演算では主な非線形単項演算は既にその最良の計算方法 (新しく追加される誤差項の係数の絶対値を理論的に最小に抑える計算方法) が発見されている．しかしながら非線形二項演算では最良の計算方法は一般には知られていない．

本章ではアフィン演算の定義とその有効性，また，非線形二項演算における問題点について述べる．また，アフィン演算を計算機に実装するときに問題となる丸め誤差の取り扱いについて，IEEE 標準 754 の規格に基づく浮動小数点数システムを備えた CPU の存在を仮定し，OS が Free BSD，言語が C，C++の場合を例として説明を行う．

第 4 章ではアフィン演算における乗算に関して述べる．第 3 章で述べられた非線形二項

演算の問題は乗算においても例外ではなく，アフィン演算の現状の乗算では最良の計算は実現できていなかった．そこで本章ではアフィン演算を，特に乗算において整備・補完することによって，アフィン演算の実用化に貢献することを目的とする．Jorge Stolfi らにより定義されたアフィン形式同士の乗算 （Stolfi の乗法) [38][39] は，簡便ではあるけれど新しく追加される誤差項の見積もりが甘く，その結果真の像の包含をシャープに行っていないため区間評価の性能をかなり落としていた．そこで宮田孝富，柏木雅英によりアフィン形式同士の乗算の改良版 (改良乗算) [43] が 2000 年に提案された．この方法は誤差項同士の積を計算する際に変数間の相関をある程度考慮するような工夫を施すことで，Stolfi の乗法よりも新しく追加される誤差項の係数の絶対値を小さくすることに成功している．しかし，その新しく追加された誤差項の係数の絶対値は理論的に最小に抑えられたものではなく，さらに小さくすることが可能である．

本章ではまずこれまでの方法として Stolfi の乗法，改良乗算について説明する．次にアフィン演算の乗算において新しく追加される誤差項の係数の絶対値を理論的に最小に抑える計算方法 (最良乗算) を提案し，さらにこの最良乗算を実用時間で実行する方法を提案する．本章の最後では数値例により最良乗算の有効性を確認する．

第 5 章ではアフィン演算における除算に関して述べる．乗算同様，第三章で述べられた非線形二項演算の問題は除算においても例外ではなく，アフィン演算の現状の除算では最良の計算は実現できておらず，改良の余地を残していた．そこで本章ではアフィン演算を，特に除算において整備・補完することによって，アフィン演算の実用化に貢献することを目的とする．アフィン形式同士の除算については原論文では定義されておらず，従来は x/y=x(1/y) のように，逆数をとる非線形単項演算と乗算の合成として計算していた (文献 [38][39] の方法)．しかしこのように除算を 2 つの非線形演算に分解すると，1 回の演算で誤差項が 2 つ新しく追加されてしまい，区間評価の性能を落としてしまう．そこで柏木雅英 [44][45] は除算 x/y を分解せずに直接計算する方法を 1999 年に提案した (柏木の方法)．この方法は非線形二変数関数 f(x,y)=x/y に対して直接線形近似を行う方法で，計算時間を優先して，複雑な場合分けが必要な部分を近似することにより，計算を一部簡略化している．この柏木の方法における複雑な場合分けが必要な部分を緻密に計算した方法が白井健一 (柏木雅英研究室修士)[46][47] により 2000 年に提案された (白井の方法)．この方法は変数 x,y 間に相関がある場合には最適な評価を与えることはできない．ゆえに変数 x，y 間に相関がある場合に白井の方法よりも新しく追加される誤差項の係数の絶対値を小さくすることが可能である．

本章ではまずこれまでの方法として文献 [38][39] の方法，柏木の方法，白井の方法について説明する．次に白井の方法よりも新しく追加される誤差項の係数の絶対値を小さくすることが期待できる 3 つの方法を提案する (提案手法 1〜提案手法 3)．本章の最後においてはいくつかの数値例によりアフィン演算の除算における全ての方法を比較 · 検証し，提案手法の有効性を示す．

第 6 章では多項式の値域の区間評価について述べる．値域を評価する関数を多項式に限

定すると，多項式の性質を生かすことにより，区間演算で起こる区間幅の爆発的な増大を抑制するための効率的な方法を提案できる．本章では値域を評価する関数を多項式に限定し，区間演算で起こる区間幅の爆発的な増大を抑制し関数の値域のシャープな包含を得る方法を提案する．多項式の値域の評価において，区間演算で起こる区間幅の爆発的な増大を抑制する方法として，これまでに Horner 法 (例えば [18][48])，平均値形に基づく方法 (例えば [48]) ，Alefeld の方法 [49][50]，アフィン演算を使用する方法 [51] が提案されている．

本章ではまずこれまでの方法として Horner 法平均値形式に基づく方法，Alefeld の方法，アフィン演算を使用する方法について説明する．次に 5 つの方法を提案する (提案手法 1〜提案手法 5)．本章の最後においていくつかの数値例を行うことにより，多項式の値域を評価する全ての方法を比較 · 検証し，提案手法の有効性を示す．

第 7 章では結論を述べ，本論文を総括する．

第 2 章

区間解析

2.1 はじめに

本章では，本論文の議論の基礎となる区間解析について述べる．区間解析とは区間演算を用いた数値解析のことであるが，区間という考え方が導入された理由については前章において本論文の背景として述べた．また，区間演算の考え方を数値計算に適用して，計算機の浮動小数点演算の演算結果を実用的な速さと性能で精度保証できるようになるまでの歴史的な経緯についても前章で簡潔に述べた．本章で行う説明は，前章の説明を受けて区間解析の現況をより具体的に述べるものである．

まず本章では，計算機における数値の表現について述べ，次に区間解析および精度保証付き数値計算の根幹をなす区間演算をその用語とともに説明する．最後に区間演算の抱える問題点と，現在よく知られているその対処法について概観する．

2.2 計算機における数値の表現

本節では計算機における数値 (特に IEEE 標準 754 に基づいた浮動小数点数) の表現とその表現上で起こる丸め誤差について述べる．理工学において現れるさまざまな問題を解くときにそれが数式処理によって解析的に解ける場合は全体的に見て少なく，多くの場合は計算機上の離散的かつ有限精度の計算に置き換えて解かれていることは前章で述べた．つまり，計算機の性能がどんなに進歩したとしても，メモリが有限である以上すべての実数を計算機で表現できるわけではない．たとえば，円周率などの無理数は循環しない無限小数となるので，そのすべての桁をメモリに入れることは当然できない．このように計算機上に表現できない実数は適意に丸められて (すなわち近似されて) 浮動小数点数として表現される．

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 という規格に基づいた 2 進数の浮動小数点数システムを採用している．そのようなシステムのもとで，一般に実数 x は次のような形で表される (ただし実数 x の絶対値がごく小さい場合を除く)：

$$x = \pm(1 + \frac{x_1}{2} + \frac{x_2}{2^2} + \cdots + \frac{x_N}{2^N}) \times 2^n, \qquad (x_i \in \{0, 1\}) \tag{2.1}$$

式 (2.1) の形で表される浮動小数点数のことを正規化数と言う．計算機には実数 x の符号や x_i $(i = 1, \cdots, N)$ の値，指数 n に関する情報を記憶させるが，式 (2.1) の中の

$$1 + \frac{x_1}{2} + \frac{x_2}{2^2} + \cdots + \frac{x_N}{2^N} \tag{2.2}$$

の部分を記憶させる領域のことを仮数部と言い，2^n の部分を記憶させる領域のことを指数部と言う．とくに倍精度浮動小数点数 (C 言語における `double`) は 8byte = 64bit で表されるが，符号を表すのに 1bit (0 なら正，1 なら負)，x_i を表すのに 52bit (つまり $N = 52$)，指数部を表すのに 11bit がそれぞれ割り当てられている．ただし，指数部を表す 11bit 浮動小数点数 e は符号なし整数で，指数 n と次のような対応がつけられている．

$$n = e - 1023, \quad (1 \leq e \leq 2046) \tag{2.3}$$

なお，本来 11bit の符号なし整数によって 0 から $2^{11}-1=2047$ までの数を表現できるが，後述するように $e=0$ と $e=2047$ は正規化数を表現する用途には用いられない．つまり倍精度浮動小数点数における正規化数の指数 n は -1022 から 1023 までの値を取る．

一方，単精度浮動小数点数 (C 言語における `float`) の場合は全体が 4byte = 32bit で表され，その内訳は符号に 1bit，仮数部に 23bit，指数部に 8bit ($n=e-127,\ 1\leq e\leq 254$) である．サイズの違いこそあれ，基本的な仕組みは倍精度浮動小数点数と同様である．

さて，正規化数で表現できる絶対値最大の倍精度浮動小数点数は，

$$x=\pm(1+\frac{1}{2}+\frac{1}{2^2}+\cdots+\frac{1}{2^{52}})\times 2^{1023}=2^{1024}-2^{971} \tag{2.4}$$

である．また，正規化数で表現できる絶対値最小の倍精度浮動小数点数は，

$$x=\pm(1+\frac{0}{2}+\frac{0}{2^2}+\cdots+\frac{0}{2^{52}})\times 2^{-1022}=2^{-1022} \tag{2.5}$$

であるが，$e=0$ とし (この場合 n との間に式 (2.3) の関係は成立しない)，実数 x と

$$x=\pm(0+\frac{x_1}{2}+\frac{x_2}{2^2}+\cdots+\frac{x_{52}}{2^{52}})\times 2^{-1022},\qquad (x_i\in\{0,1\}) \tag{2.6}$$

のような対応をつけることで，より絶対値の小さい倍精度浮動小数点数の表現を可能にしている (このことは単精度浮動小数点数の場合も同様)．式 (2.6) のように仮数部の最初の桁を 0 に固定した浮動小数点数のことを非正規化数と言う．非正規化数で表現できる絶対値最大の倍精度浮動小数点数は，

$$x=\pm(0+\frac{1}{2}+\frac{1}{2^2}+\cdots+\frac{1}{2^{52}})\times 2^{-1022}=2^{-1022}-2^{-1074} \tag{2.7}$$

である．また，非正規化数で表現できる絶対値最小の倍精度浮動小数点数は，

$$x=\pm(0+\frac{0}{2}+\frac{0}{2^2}+\cdots+\frac{1}{2^{52}})\times 2^{-1022}=2^{-1074} \tag{2.8}$$

である．ここで注意すべきことは，正規化数における仮数部の最初の桁は必ず 1 に固定しており，その 1 はメモリに格納しないので，たとえば倍精度浮動小数点数においては仮数部 52bit で 53bit 分の精度を達成しているのに対し，非正規化数における仮数部の最初の桁は必ず 0 に固定しており，有効数字が 2 進で 53 桁未満 (52 桁とも限らない) となるので，正規化数のときのように 53bit 分の精度は達成されないという点である．

その他，倍精度浮動小数点数で 0 を表現するには仮数部，指数部を表すそれぞれ 52bit，11bit の浮動小数点数をいずれも 0 として，

$$x=\pm(0+\frac{0}{2}+\frac{0}{2^2}+\cdots+\frac{0}{2^{52}})\times 2^{-1022} \tag{2.9}$$

のように表現する．また，$\pm\infty$ や NaN (Not a Number，$\sqrt{-5}$ や ∞/∞ など不当な演算の結果として得られる) などの特殊な数を表現する場合，倍精度浮動小数点数では e の値を 2047 にすることによって対処している (単精度浮動小数点数の場合もこれに準ずる)．正規

化された倍精度浮動小数点数の相対的な精度は，$2^{-52} \simeq 10^{-15.65}$(この値のことをマシンイプシロンと言う) より 10 進でおよそ 15 桁の精度があると考えられる．プロセッサによっては倍精度浮動小数点数よりさらに高精度の拡張倍精度浮動小数点数 (C 言語における `long double`) を備える場合もあるが，プロセッサごとに規格が乱立しており統一されていない．

このように浮動小数点数システムによって実数を計算機上に表現する方法は，絶対値の非常に大きい数値に対しても絶対値の非常に小さい数値に対しても，常に一定のメモリしか割り当てずに済む点ですぐれている．反面，浮動小数点数で表現できない数を浮動小数点数に近似する過程で丸め誤差が生じてしまう欠点がある．この丸め誤差は，無理数など本質的に計算機上で表現できない数を丸めるときだけでなく，既に浮動小数点数として表された数値の間でなされる演算に対して，その演算結果が丸められて生じることもある．たとえば加減算のような線形演算に対してさえも，絶対値の極端に離れた 2 つの浮動小数点数を足すときにはいわゆる情報落ち (loss of trailing digits，積み残しとも言う) が起きるし，逆に非常に近い値をとる 2 つの浮動小数点数の間の引き算ではいわゆる桁落ち (cancelation) が起きて有効数字がいっぺんに失われる．そして，その丸め誤差はときに (わずか数ステップからなる代数演算に対してさえも) 演算結果の有効数字を完全に消失させてしまうほど猛威をふるうことがあるのは，前章で指摘したとおりである．

2.3 計算機における丸めの向きの指定の例

一般に IEEE 標準 754 の規格に基づく浮動小数点数システムを実装している CPU では丸めの向きの指定が可能であるが，本節では実際に OS が FreeBSD で，プログラミング言語が C 言語，あるいは C++言語の場合について，丸めの向きの指定方法を述べる．

```
#define up()    asm volatile ("fldcw __RoundUp")
#define down()  asm volatile ("fldcw __RoundDown")
#define near()  asm volatile ("fldcw __RoundNear")
#define chop()  asm volatile ("fldcw __RoundChop")
```

`up()`, `down()`, `near()`, `chop()` はそれぞれ上向き丸め (切り上げ)，下向き丸め (切り捨て)，最近点への丸め (0 捨 1 入)，絶対値を小さくする向きへの丸めを行うように定義されたマクロである．ただし，プログラム中のどこかに

```
int _RoundUp    = 0x1b3a;
int _RoundDown  = 0x173a;
int _RoundNear  = 0x133a;
int _RoundChop  = 0x1f3a;
```

のようなグローバル変数を用意する必要がある．

2.4 区間演算

本節では区間解析および精度保証付き数値計算の根幹をなす区間演算について述べる．前章でも少し紹介したが，区間演算は浮動小数点演算における丸め誤差を数学的な厳密さをもって把握するための基本的な方法である．区間演算は歴史的には 1950 年代に，当時東京大学の学生であった須永照雄 [20](1929 − 1995) 等により提案された．

区間演算では計算機上における実数を「ある幅を持った実数の集合 (区間)」として表現する．また，区間は区間と実数，あるいは区間同士の間で定義された演算に関して「閉じて (演算結果が区間になって)」いて，かつ「演算結果としてあり得る集合を包含するように」定義されたものであり，一種の集合算と見做すことができる．

データ型としての区間は具体的には [下端, 上端] のように 2 つの浮動小数点数のなす区間によって実数を挟み込む．また，区間はその区間幅によって計算機上では表せない実数や区間同士の演算結果などの誤差を見積もっている．

たとえば，$a, b, c, d \in \boldsymbol{R}$ として，2 つの (閉) 区間 $[a, b]$ と $[c, d]$ の間の四則演算はそれぞれ以下のように定義される：

$$[a, b] + [c, d] = [a + b, c + d] \tag{2.10}$$

$$[a, b] - [c, d] = [a - d, b - c] \tag{2.11}$$

$$[a, b] \times [c, d] = [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}] \tag{2.12}$$

$$[a, b]/[c, d] = [a, b] \times [\frac{1}{d}, \frac{1}{c}] \qquad ([c, d] \ni 0) \tag{2.13}$$

しかし，計算機上において実数 a, b, c, d の値は浮動小数点数に丸められてしまうのだから，「式 (2.10)〜式 (2.13) を計算するときに生じる計算機の丸め誤差はどのようにして計算するのか」という問題が当然起こってくる．実は，2.3 節で述べたように，IEEE 標準 754 に基づく浮動小数点数システムにおいては，計算結果の丸めの向きを自由に指定することができる．そこで，式 (2.10)〜式 (2.13) において区間の下端を「切り捨て」て，上端を「切り上げ」て，それぞれ計算することによって，計算機の丸め誤差まで含めて「演算結果としてあり得る集合を包含するように」定義することができる．

また，このように区間同士の演算方法が定義された演算を組み合わせてつくられる関数に対して，その関数の計算手順を区間演算に置きかえることは，関数の値域を包み込んだ区間 (以後，本論文ではこれを区間包囲を呼ぶ．また必要に応じて区間演算を用いて関数の区間包囲を得ることを「関数の値域を評価する」，あるいは「関数を評価する」と記述する) を得るための一つの手段である．つまり，区間演算には「計算機の丸め誤差を厳密に把握する」役割と「関数の値域を評価する」役割の 2 つがある．方程式の解を精度保証付きで求める場合など，むしろ後者の方が重要な役割を果たす．

2.5 区間演算の問題点とその解決法

本節では区間演算の抱える問題点と，現在よく知られているその対処法について述べる．区間演算の問題点の 1 つとして，確かに計算された区間は真の値 (あるいは像) を含むもの

の，区間幅が極端に広くなってしまうことが少なくない点が挙げられる．これは関数の値域を評価する場合など，入力として比較的幅の広い区間を扱った場合に顕在化する問題であり，そのような例を例題 1.2 で示した．

この現象は，2 つの関数 x^2 と $2x$ が同じ x の関数で互いに相関があるにもかかわらず，区間演算における減算の定義に従えばその相関性は無視されてしまい，両者が独立に動くものとして計算してしまうことに原因がある．

なお，例題 1.2 中の式を $x^2 - 2x = x(x-2)$ と変形して評価すればそのときの結果は $[-1.21, -0.81]$ といくらか改善されるし，$x^2 - 2x = (x-1)^2 - 1$ と変形すれば真の像 $[-1, -0.99]$ が得られる．このように，区間演算においては数学的に同値な式でも計算の順序によって大きく結果が異なる場合がある．一般にどのように式を変形すれば良い (過大評価の少ない) 結果が得られるかはきわめて難しい問題であり，対象とする関数を多項式に限っても最適な方法 (過大評価を最小に抑える方法) はいまのところ知られていない．

また，同様に変数間の相関性を表現できないことに起因する問題として，ラッピング効果 (Wrapping Effect) と呼ばれる現象がある (図 2.1 参照)．n 次元ユークリッド空間 $\boldsymbol{R}^n (n > 1)$ の部分集合を区間で表す場合，ベクトルの各成分が区間値をとる「区間ベクトル」によってこれを表現するが，区間ベクトルによって表すことができるのは $\boldsymbol{R}^n$ 内の超直方体領域でしかない．換言すれば，各成分間に相関があったとしてもそれを区間ベクトルによって表現することはできない．これによって区間幅の増大が起きてしまう．例えば

$$A = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 0.5 & 0.5 \\ -0.5 & 0.5 \end{pmatrix}, \quad I = \begin{pmatrix} [-1,1] \\ [-1,1] \end{pmatrix} \tag{2.14}$$

のとき，$(AB)I$ と $A(BI)$ を区間演算で計算すると，

$$(AB)I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} I = \begin{pmatrix} [-1,1] \\ [-1,1] \end{pmatrix} \tag{2.15}$$

$$A(BI) = A \begin{pmatrix} [-1,1] \\ [-1,1] \end{pmatrix} = \begin{pmatrix} [-2,2] \\ [-2,2] \end{pmatrix} \tag{2.16}$$

のように結果が大きく異なる．これは I に B をかけた結果の真の像は超直方体領域でないのに区間演算の結果としてはそれを包む (wrap する) 超直方体領域になってしまい，その結果にさらに A をかけたときにも同様の過大評価が生じているためである．以後 2.5.1 と 2.5.2 節において，区間評価における過大評価の問題に対して現在よく知られている対処法について述べる．

2.5.1 区間分割

本節では区間評価における過大評価の問題を解決するきわめて自明な方法として，入力区間を分割して別々に評価し，最後に和集合を取る方法 (区間分割) について述べる．この方法では区間演算による入力区間 I における $f(x)$ の評価を $F(I)$ とするとき，I を

$$I = \bigcup_i I_i \tag{2.17}$$

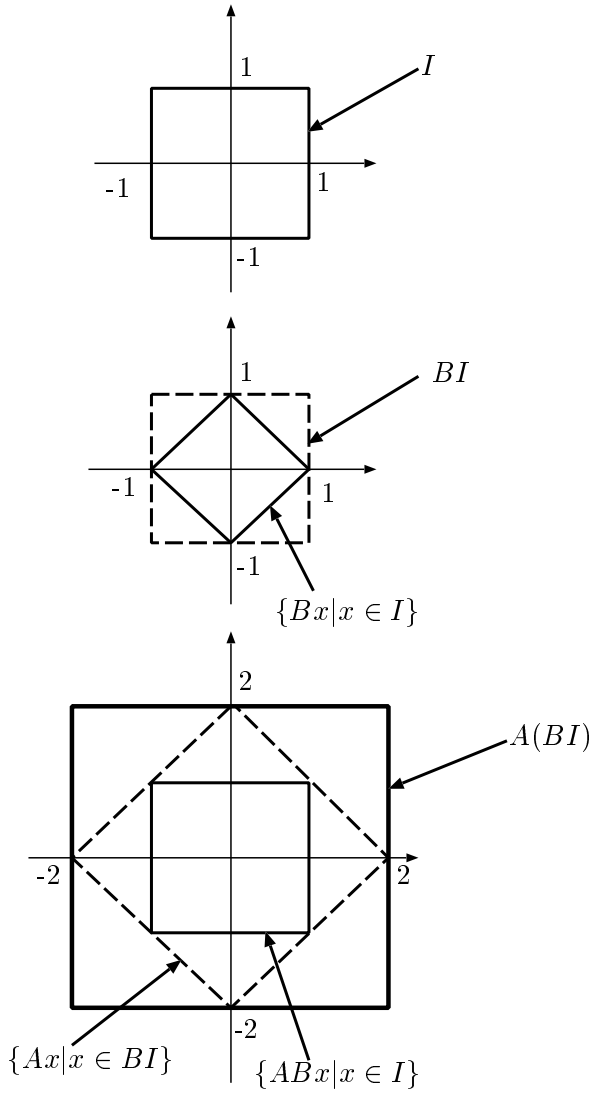

図 2.1: ラッピング効果

のように分解し，分割されたそれぞれの小区間に対して区間演算を行い，最終結果を

$$\bigcup_i F(I_i) \tag{2.18}$$

で表せば良い．この方法は単純で過大評価を抑える効果も高いが，分割数を多くすると当然それだけ多くの時間がかかり，とくに変数の数が多い問題に対しては，充分に効果を発揮させるのにあまりに多くの分割を必要とする欠点がある．

2.5.2 平均値形式

本節では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代表的な方法な方法として平均値形式 (Mean Value Form) について述べる．

平均値形式では，関数 $f(x), f : \boldsymbol{R}^n \to \boldsymbol{R}^m$ に対して $I \subset \boldsymbol{IR}^n$ の像 $f(I) = \{f(x)|x \in I\}$ 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

$$f(x) - f(c) = \int_0^1 f'(c + t(x - c))\,\mathrm{d}t(x - c) \tag{2.19}$$

が成立する．ここで，

$$\int_0^1 f'(c + t(x - c))\,\mathrm{d}t \in \mathrm{co}\{f'(x)|x \in I\} \tag{2.20}$$

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

$$f(c) + F'(I)(I - c) \tag{2.21}$$

のように評価できる．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できる．区間行列は凸集合であるから，凸包を考える必要はない．c は入力区間 I に含まれる任意の点であるが，普通は I の中心に取る．前述の例をこの方法で計算すると，

$$\begin{aligned} f(c) + F'(I)(I - c) &= f(1) + F'([0.9, 1.1])([0.9, 1.1] - 1) \\ &= -1 + [-0.2, 0.2][-0.1, 0.1] \\ &= [-1.02, -0.98] \end{aligned} \tag{2.22}$$

とかなり改善されることがわかる．先に述べたように，区間演算の定義に従ってただ計算したときには x^2 と $2x$ の間の強い相関を無視したことによって区間幅の大幅な増大を招いたが，この方法では

$$F'([0.9, 1.1]) = 2[0.9, 1.1] - 2 = [-0.2, 0.2] \tag{2.23}$$

のように，傾きがきちんと打ち消されていることがわかる．

また，変数の数が多いときに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．すなわち，Taylor 展開

$$f(x) = \sum_{k=0}^{n} \frac{1}{k!} f^{(k)}(c)(x-c)^k + R_{n+1} \tag{2.24}$$

に対して，Bernoulli の剰余項

$$R_{n+1} = \frac{1}{n!} \int_0^1 (1-t)^n f^{(n+1)}(c + t(x-c))\,\mathrm{d}t(x-c)^{n+1} \tag{2.25}$$

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

$$R_{n+1} = \frac{1}{(n+1)!} \int_0^1 f^{(n+1)}(c + (1 - \sqrt[n+1]{s})(x-c))\,\mathrm{d}s(x-c)^{n+1} \tag{2.26}$$

と変形できるので，平均値形式と同様に考えて，

$$\sum_{k=0}^{n} \frac{1}{(k+1)!} f^{(k)}(c)(I-c)^k + \frac{1}{(n+1)!} f^{(n+1)}(I)(I-c)^{n+1} \tag{2.27}$$

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

2.6 むすび

本章では，本論文の議論の基礎となる区間解析について説明した．そのためにまず，計算機上における実数の表現について述べ，次に区間解析で基本となる区間演算をその用語とともに定義した．最後に区間演算で問題となる区間幅の増大の問題と，それに対して現在よく知られている解決法について概観した．

次章ではこの区間幅の増大の問題を解決するための方法の一つで，区間演算の有力な拡張であるアフィン演算の定義とその有効性，また，非線形二項演算における問題点について述べる．

第 3 章

アフィン 演算

3.1 はじめに

本章では区間演算の有力な拡張であるアフィン演算の定義等について述べる．アフィン演算は Jorge Stolfi，Luiz H. de Figueiredo ら [38][39] によって 1997 年に提案された区間演算の一種である．区間演算では，区間同士の相関が全く考慮されていないのに対し，アフィン演算ではアフィン形式という特殊な形式を用いることで，区間同士の相関を考慮し，区間演算における区間幅の爆発的な増大を抑制する．一方で，アフィン演算はその非線形演算に関する取り扱いが極めて難しく，現状でなお多くの課題を残している．アフィン演算における非線形演算は真の像そのものではなく，線形近似と新しい誤差項の追加により真の像を包含し，その包含を演算結果としているのだが，そこで新しく追加される誤差項の係数は一般にはその後の演算で打ち消しが起きず，区間評価の性能を落とす原因となる．アフィン演算では主な非線形単項演算は既にその最良の計算方法 (新しく追加される誤差項の係数の絶対値を理論的に最小に抑える計算方法) が発見されている．しかしながら非線形二項演算では最良の計算方法は一般には知られていない．

本章ではアフィン演算の定義とその有効性，また，非線形二項演算における問題点について述べる．また，アフィン演算を計算機に実装するときに問題となる丸め誤差の取り扱いについて，本章では IEEE 標準 754 の規格に基づく浮動小数点数システムを備えた CPU の存在を仮定し，OS が Free BSD，言語が C，C++の場合を例として説明を行う．なお，第 2.3 節で示した計算機の丸めの向きを制御するためのマクロ `up()`，`down()`，`near()`，`chop()` を以下の説明において適宜用いる．

3.2 アフィン形式

アフィン 演算は，変数間の相関性を考慮することにより，区間演算で区間が爆発的に増大する問題を解決する方法の一つである．

アフィン 演算では，変量 x はアフィン形式

$$x = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + \cdots + x_n\varepsilon_n \tag{3.1}$$

で表される．ここで，x_i は実数であり，ε_i はそれが区間 $[-1, 1]$ に含まれることだけが分かっているような dummy 変数であり，x の変量の部分のそれぞれ独立した構成要素である．

x_0 をこのアフィン形式の central value，係数 x_i を部分偏差 (partial deviations)，ε_i を noise symbols という．noise symbol である ε_i はそれぞれ独立した値をとって変量 x を構成し，係数 x_i がそのふれ幅を決定する．

たとえば，アフィン形式 x, y があり，それが

$$x = 1 + 0.5\varepsilon_1 \tag{3.2}$$

$$y = 1 + 0.5\varepsilon_2 \tag{3.3}$$

とする．これは変数 x, y 間に相関性がない状態で，このとき点 (x, y) がとり得る領域は図 3.1 のようになる．

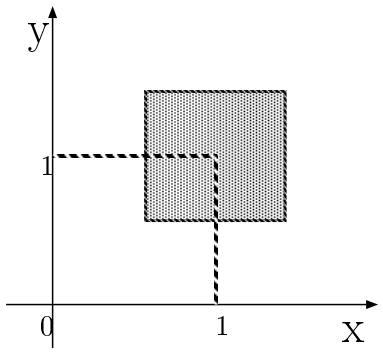

図 3.1: x と y に相関がない場合

これに対し，

$$x = 1 + 0.5\varepsilon_1 \tag{3.4}$$
$$y = 1 + 0.4\varepsilon_1 + 0.1\varepsilon_2 \tag{3.5}$$

では，それぞれがとり得る範囲は $[0.5, 1.5]$ で変わらないが ε_1 の係数を見ると分かるように両者には強い相関があり点 (x, y) のとり得る範囲は図 3.2 のようになる．

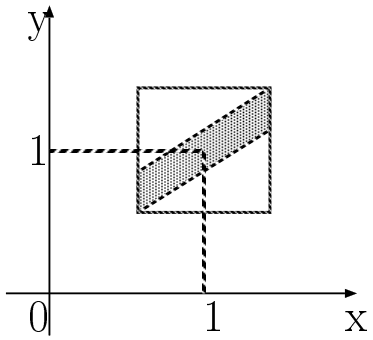

図 3.2: x と y に相関がある場合

また，アフィン演算を計算機上で実行するためには計算過程で生じる丸め誤差を厳密に見積もる必要があり，その結果アフィン形式は次式の様になる．

$$\mathtt{x}[0] + \mathtt{x}[1]\varepsilon_1 + \cdots + \mathtt{x}[n]\varepsilon_n + \mathtt{r}\varepsilon_r \tag{3.6}$$

ここで ε_r は $\varepsilon_1, \cdots, \varepsilon_n$ と同様に区間 $[-1, 1]$ に含まれることだけが分かっているような dummy 変数であり，$\mathtt{r}$ は非負の実数である．$\mathtt{x}[0], \cdots, \mathtt{x}[n]$ はそれぞれ $x_0, \cdots, x_n$ の値に対応してお

り，$x_0, \cdots, x_n$ の値を求める過程で生じた計算機の丸め誤差全ての絶対値和を r によって表す．この $\mathtt{r}\varepsilon_r$ の項だけは加減算による打ち消しが起きない．計算機には x[0], $\cdots$, x[n] 及び r の値を記憶させる．

3.3 初期化及び通常の区間への還元

超直方体領域

$$X = (X_1, \cdots, X_m)^T = ([\underline{X_1}, \overline{X_1}], \cdots, [\underline{X_m}, \overline{X_m}])^T \tag{3.7}$$

において m 変数関数 $f(x_1, \cdots, x_m)$ の値域を評価する場合，最初に与えられる m 個の変数 $x_1, \cdots, x_m$ は一般に互いに無相関であるから，m 個の ε を用いて

$$\begin{aligned} X_1 &= \frac{\overline{X}_1 + \underline{X}_1}{2} + \frac{\overline{X}_1 - \underline{X}_1}{2}\varepsilon_1 \\ X_2 &= \frac{\overline{X}_2 + \underline{X}_2}{2} + \frac{\overline{X}_2 - \underline{X}_2}{2}\varepsilon_2 \\ &\vdots \\ X_m &= \frac{\overline{X}_m + \underline{X}_m}{2} + \frac{\overline{X}_m - \underline{X}_m}{2}\varepsilon_m \end{aligned} \tag{3.8}$$

のようなアフィン形式で初期化しておく．これに従い計算機上で区間 (変域)$[a, b]$(a, b はすでに浮動小数点数であることを仮定) を

$$\frac{a+b}{2} + \frac{b-a}{2}\varepsilon, \quad (-1 \le \varepsilon \le 1) \tag{3.9}$$

ように初期化する際，その計算過程で生じる丸め誤差の影響まで厳密に考慮した初期化を行うには

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

を計算し，$\mathtt{c} + \mathtt{r}\varepsilon, \ (-1 \le \varepsilon \le 1)$ を結果とすればよい．

アフィン形式

$$x = x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n \tag{3.10}$$

は，次のようにして通常の区間に変換できる．

$$[x_0 - \Delta, x_0 + \Delta], \Delta = \sum_{i=1}^{n} |x_i| \tag{3.11}$$

これを計算機の丸め誤差まで含めて厳密に評価するには以下で与えられる区間 [xl, xu] を結果とすれば良い．なお，以下のプログラムコード中の Delta は式 (3.11) 中の Δ に対応している．

```
Delta = fabs(x[1]);
up ();
for(i = 2; i ≤ n; i + +)Delta+ = fabs(x[i]);
xu = x[0] + Delta;
down ();
xl = x[0] − Delta;
```

3.4 アフィン 演算

2 つのアフィン形式 x, y,

$$x = x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n \tag{3.12}$$

$$y = y_0 + y_1\varepsilon_1 + \cdots + y_n\varepsilon_n \tag{3.13}$$

に対して，以下に各種演算の定義を行う．すなわち，

$$\begin{aligned} z &= f(x) \\ &= f(x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n) \end{aligned} \tag{3.14}$$

という $f(x)$ や

$$\begin{aligned} z &= f(x, y) \\ &= f(x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n, y_0 + y_1\varepsilon_1 + \cdots + y_n\varepsilon_n) \end{aligned} \tag{3.15}$$

という $f(x, y)$ を，

$$z = z_0 + z_1\varepsilon_1 + \cdots + z_n\varepsilon_n \tag{3.16}$$

のような，アフィン形式で表す．

以後，アフィン演算の定義を，線形演算と，非線形演算に分けて行う．

3.5 線形演算

演算 f が線形な場合の演算は，以下のように定義できる (αは実定数)．

$$x \pm y = (x_0 \pm y_0) + (x_1 \pm y_1)\varepsilon_1 + \cdots + (x_n \pm y_n)\varepsilon_n \tag{3.17}$$

$$x \pm \alpha = (x_0 \pm \alpha) + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n \tag{3.18}$$

$$\alpha x = (\alpha x_0) + (\alpha x_1)\varepsilon_1 + \cdots + (\alpha x_n)\varepsilon_n \tag{3.19}$$

(複号同順)

式 (3.17) ようなアフィン演算同士の加減算を計算機の丸め誤差まで厳密に考慮しながら行うには以下のようにする：

```
down ();
for (i = 0; i <= N; i++)   z[i] = x[i] ± y[i];
up ();
zr = xr + yr;
for (i = 0; i <= N; i++){
  c[i] = (z[i] + (x[i] ± y[i])) / 2;
  r[i] = c[i] − z[i];
  zr+ = r[i];
}
```

ただし，xr, yr, zr はそれぞれ x , y , $x \pm y$ の丸め誤差項の係数とし，$\mathtt{c[0]} + \mathtt{c[1]}\,\varepsilon_1 + \cdots + \mathtt{c[n]}\,\varepsilon_n + \mathtt{zr}\varepsilon_{\mathrm{r}}$ を演算結果とする．

また，式 (3.18) のようなアフィン演算と定数との加減算を計算機の丸め誤差まで厳密に考慮しながら行うには以下のようにする：

```
down ();
for (i = 0; i <= N; i++)   z[i] = x[i] ± α;
up ();
zr = xr;
for (i = 0; i <= N; i++){
  c[i] = (z[i] + (x[i] ± α)) / 2;
  r[i] = c[i] − z[i];
  zr+ = r[i];
}
```

ただし，xr, zr はそれぞれ x ,$x \pm \alpha$ の丸め誤差項の係数とし，$\mathtt{c[0]} + \mathtt{c[1]}\,\varepsilon_1 + \cdots + \mathtt{c[n]}\,\varepsilon_n + \mathtt{zr}\varepsilon_{\mathrm{r}}$ を演算結果とする．

加えて，式 (3.19) のようなアフィン演算の定数倍に対して計算機の丸め誤差まで厳密に

見積もる場合は以下のようにする：

```
up();
for (i = 0; i <= N; i++) {
  if (x[i] > 0) {
    lx = (-l) * x[i];      ux = u * x[i];
    cl[i] = -lx;           cu[i] = ux;
  } else {
    lx = l * x[i];      ux = (-u) * x[i];
    cl[i] = -ux;        cu[i] = lx;
  }
}
zr = xr + yr;
for (i = 0; i <= N; i++){
  c[i] = (cl[i] + cu[i]) / 2;
  r[i] = c[i] - cl[i];
  zr+ = r[i];
}
```

ただし，l,u は実数 α をそれぞれ下向き，上向き丸めで計算 (近似) したもの (実はこの計算は α が区間の場合にも適用でき，その場合 l,u はそれぞれ区間 α の下端と上端を表す)，cl[i], cu[i] はそれぞれ $\alpha * x[i]$ の下端と上端，xr, yr, zr はそれぞれ x , y , $\alpha * x$ の丸め誤差項の係数 (いずれも非負) であるとし，$c[0] + c[1]\,\varepsilon_1 + \cdots + c[n]\,\varepsilon_n + zr\varepsilon_r$ を演算結果とする．

特に，式 (3.19) において $\alpha = -1$ (negation) の場合は

```
for (i = 0; i <= N; i++)   z[i] = -x[i];
zr = xr;
```

この操作によって新たな丸め誤差は混入しないが，丸め誤差専用項の係数は常に非負を保っていなければならないことに注意する．$z[0] + z[1]\,\varepsilon_1 + \cdots + z[n]\,\varepsilon_n + zr\varepsilon_r$ を演算結果とする．

3.6 非線形単項演算

アフィン形式

$$x = x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n$$

に対する非線形な演算 f について，$z = f(x)$ は一般にアフィン形式で表すことは出来ない．そこで，f を線形演算で近似し，近似誤差を新しいダミー変数 ε_{n+1} を導入することによって表すことを考える．まず，x の変域 X を

$$X = [x_0 - \Delta_x, x_0 + \Delta_x], \quad \Delta_x = \sum_{i=1}^{n} |x_i| \tag{3.20}$$

で求める．次に，この領域 X において f をなるべく良く近似する (誤差を最小にする) ような一次関数 $ax+b$ を求める (図 3.3 参照)．誤差の最大値

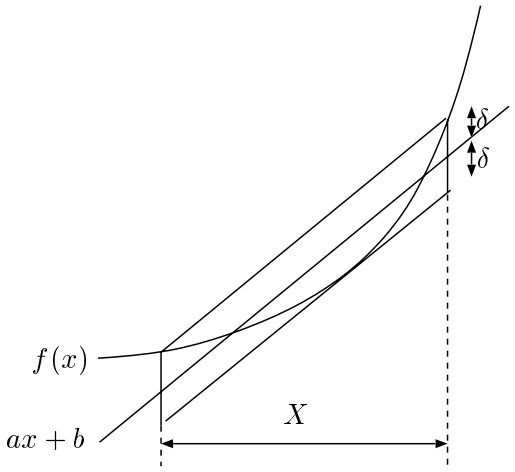

図 3.3: 非線形関数の線形近似

$$\delta = \max_{x \in X} |f(x) - (ax+b)| \tag{3.21}$$

を求め，これをダミー変数 ε_{n+1} の係数とする．すなわち，非線形関数 f は区間 X において

$$f(x) \in ax + b + \delta\varepsilon_{n+1} \tag{3.22}$$

であり，よって単項演算の結果は

$$a(x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n) + b + \delta\varepsilon_{n+1} \tag{3.23}$$

とすればよい．これはダミー変数が一つ増加したアフィン形式であり，領域 X において $f(x)$ の真の像を包含している．

非線形単項演算において計算機の丸め誤差まで厳密に見積もる場合は a, b を通常の区間演算を用いて算出し (このとき a, b は区間となる)，δ をその絶対値ができるだけ大きくなるように適意に丸めの向きを変更して算出する．このような算出の後に得られた式 (3.23) に対して，3.5 節で述べた計算機の丸め誤差まで厳密に見積もる線形演算を適用すれば，その演算結果となるアフィン形式は計算機の丸め誤差まで厳密に見積もりながら領域 X において $f(x)$ の真の像を包含している．

3.7 非線形二項演算

アフィン形式 x, y:

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

に対する非線形二項演算 $z = f(x, y)$ について考える．例えば，$x \times y$， x/y， x^y などである．非線形二項演算も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 $f(x, y)$ をなるべく良く近似する (δ を最小にする) 一次式 $ax + by + c$ を求め，

$$ax + by + c + \delta\varepsilon_{n+1} \tag{3.24}$$

を結果とする．

非線形二項演算において計算機の丸め誤差まで厳密に見積もる場合も非線形単項演算と同様に，a, b, c を通常の区間演算を用いて算出し (このとき a, b, c は区間となる)，δ をその絶対値ができるだけ大きくなるように適意に丸めの向きを変更して算出する．このような算出の後に得られた式 (3.24) に対して，3.5 節で述べた計算機の丸め誤差まで厳密に見積もる線形演算を適用すれば，その演算結果となるアフィン形式は計算機の丸め誤差まで厳密に見積もりながら領域 $X \times Y$ において $f(x, y)$ の真の像を包含している．

3.8 最適な非線形演算について

最適な評価を与える非線形演算を，追加されるダミー変数 ε_{n+1} の係数 δ が最も小さくなる非線形演算と定義する．すなわち非線形単項演算の場合，最適な評価を与えるアフィン形式

$$z_0 + z_1\varepsilon_1 + \cdots + z_n\varepsilon_n + \delta\varepsilon_{n+1} \tag{3.25}$$

は，$(\varepsilon_1, \cdots, \varepsilon_n) \in [-1, 1] \times \cdots \times [-1, 1]$ において

$$\delta = \max_{-1 \le \varepsilon_i \le 1} |f(x) - (z_0 + z_1\varepsilon_1 + \cdots + z_n\varepsilon_n)| \tag{3.26}$$

を最小にするような $z_0, z_1, \cdots, z_n$ により与えられる．ここで，3.6 節で述べた方法で一次関数 $ax + b$ を δ が最小になるように選んだ場合，式 (3.23) は最適な非線形演算となる (式 (3.23)=式 (3.25) となる)．

二項演算の場合も，最適な評価を与えるアフィン形式

$$z_0 + z_1\varepsilon_1 + \cdots + z_n\varepsilon_n + \delta\varepsilon_{n+1} \tag{3.27}$$

は $(\varepsilon_1, \cdots, \varepsilon_n) \in [-1, 1] \times \cdots \times [-1, 1]$ において

$$\delta = \max_{-1 \le \varepsilon_i \le 1} |f(x, y) - (z_0 + z_1\varepsilon_1 + \cdots + z_n\varepsilon_n)| \tag{3.28}$$

を最小にするような $z_0, z_1, \cdots, z_n$ で与えられる．しかし，式 (3.27) を直接求めることは極めて難しい．3.7 節で述べたように，x と y の変域 X， Y を求め，$(x, y) \in X \times Y$ において非線形二項演算 $z = f(x, y)$ をなるべく良く近似する (δ を最小にする) ような一次式 $ax + by + c$ を選び，

$$a(x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n) + b(y_0 + y_1\varepsilon_1 + \cdots + y_n\varepsilon_n) + c + \delta\varepsilon_{n+1} \tag{3.29}$$

を結果としても式 (3.29) は最適な非線形演算となるとは限らない (式 (3.27)=式 (3.29) とは限らない)．これは，x と y に相関性がある場合，点 (x, y) は一般に $X \times Y$ で与えられる長方形領域内の全ての点を取らないためである．一般に点 (x, y) の取り得る領域は，$n = 1$ の場合は線分に，$n \geq 2$ の場合は点 (x_0, y_0) を対称の中心とする点対称な凸 $2n$ 角形になる．以後，このような領域を本論文では領域 D と呼ぶ．例えば

$$x = 5 + 2\varepsilon_1 - \varepsilon_2 - \varepsilon_3 \quad x \in [1, 9] \tag{3.30}$$

$$y = 9 - \varepsilon_1 + 4\varepsilon_2 + 2\varepsilon_3 \quad y \in [2, 16] \tag{3.31}$$

のとき，領域 D は図 3.4 の斜線部となる．そこで，領域 D において非線形二項演算 $z = f(x, y)$

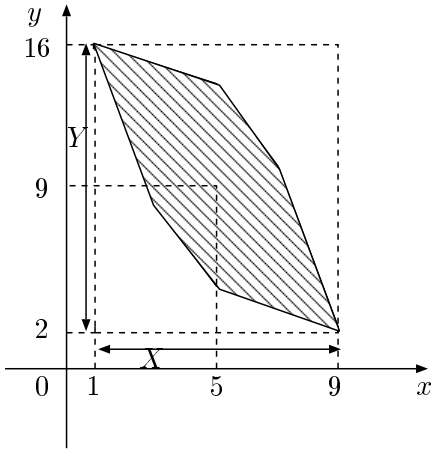

図 3.4: 点 (x, y) の取り得る領域

をなるべく良く近似する (δ を最小にする) ような一次式 $ax + by + c$ を選び，

$$a(x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n) + b(y_0 + y_1\varepsilon_1 + \cdots + y_n\varepsilon_n) + c + \delta\varepsilon_{n+1} \tag{3.32}$$

を結果とすれば式 (3.32) は最適な非線形演算となる (式 (3.27)=式 (3.32) となる) が，この非線形二項演算を実現する (式 (3.32) の a, b, c, δ を求める) 方法は一般には難しい．アフィン形式同士の乗除算も例外ではなく，最適な方法は知られていなかった．

3.9 むすび

本章では区間幅の増大の問題を解決するための方法の一つで，区間演算の有力な拡張であるアフィン演算の定義とその有効性，また，非線形二項演算における問題点について述べた．

次章ではアフィン演算における乗算について述べる．まずこれまでの方法として，Jorge Stolfi，Luiz H．de Figueiredo らにより定義された Stolfi の乗法，宮田孝富，柏木雅英によ

り提案された改良乗算の説明を行う．次に最適な評価を与える方法 (最良乗算) を提案し，さらにこの最良乗算を実用時間で実行する方法を提案する．次章の最後においてはいくつかの数値例により最良乗算の有効性を確認する．

第 4 章

アフィン 演算における乗算

4.1 はじめに

本章ではではアフィン演算における乗算に関して述べる．第 3 章で述べられた非線形二項演算の問題は乗算においても例外ではなく，アフィン演算の現状の乗算では最良の計算は実現できていなかった．そこで本章ではアフィン演算を，特に乗算において整備・補完することによって，アフィン演算の実用化に貢献することを目的としている．Jorge Stolfi らにより定義されたアフィン形式同士の乗算 （Stolfi の乗法) [38][39] は，簡便ではあるけれど新しく追加される誤差項の見積もりが甘く，その結果真の像の包含をシャープに行っていないため区間評価の性能をかなり落としていた．そこで宮田孝富，柏木雅英によりアフィン形式同士の乗算の改良版 (改良乗算) [43] が 2000 年に提案された．この方法は誤差項同士の積を計算する際に変数間の相関をある程度考慮するような工夫を施すことで，Stolfi の乗法よりも新しく追加される誤差項の係数の絶対値を小さくすることに成功している．しかし，その新しく追加された誤差項の係数の絶対値は理論的に最小に抑えられたものではなく，さらに小さくすることが可能である．

本章ではまずこれまでの方法として Stolfi の乗法，改良乗算について説明する．次にアフィン演算の乗算において新しく追加される誤差項の係数の絶対値を理論的に最小に抑える計算方法 (最良乗算) を提案し，さらにこの最良乗算を実用時間で実行する方法を提案する．本章の最後では数値例により最良乗算の有効性を確認する．

4.2 これまでの方法

本節ではアフィン演算における乗算のこれまでの方法について，Stolfi の乗法，改良乗算の説明を行う．

4.2.1 Stolfi の乗法

Stolfi の乗法では x, y の変域を X, Y として,$(x, y) \in X \times Y$ において,

$$x \times y = y_0 x + x_0 y - x_0 y_0 + (\sum_{i=1}^{n} |x_i|)(\sum_{i=1}^{n} |y_i|)\varepsilon_{n+1} \tag{4.1}$$

を結果とする．この方法は変数 x, y 間に相関がない場合に限れば最適な評価を与えるが，変数 x, y 間に相関がある場合には最適な評価を与えることができない．なお，この方法の計算量は $O(n)$(n は ε の個数) である．

4.2.2 改良乗算

4.2.1 節に示した Stolfi の乗法 [38] は 3.7 節の考え方を基礎としているが，本節ではそれとは異なる考え方による乗算の定義を示す．この方法では，アフィン形式 x, y:

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

の積を直接展開したもの

$$\begin{aligned} x \times y &= (x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n)(y_0 + y_1\varepsilon_1 + \cdots + y_n\varepsilon_n) \\ &= x_0y_0 + \sum_{i=1}^{n}(y_0x_i + x_0y_i)\varepsilon_i + \sum_{i=1}^{n}\sum_{j=1}^{n}x_iy_j\varepsilon_i\varepsilon_j \end{aligned} \tag{4.2}$$

をもとに乗算の定義を行う．式 (4.2) 自体はアフィン形式でないが，$-1 \leq \varepsilon_k \leq 1\,(k = 1, \cdots, n)$ のもとで

$$\{\sum_{i=1}^{n}\sum_{j=1}^{n}x_iy_j\varepsilon_i\varepsilon_j \,|\, -1 \leq \varepsilon_k \leq 1\,(k = 1, \cdots, n)\} \subset \{c + \delta\,\varepsilon_{n+1} \,|\, -1 \leq \varepsilon_{n+1} \leq 1\} \tag{4.3}$$

を満たすような c,δ の値を決定すれば，アフィン形式

$$(x_0y_0 + c) + \sum_{i=1}^{n}(y_0x_i + x_0y_i)\,\varepsilon_i + \delta\,\varepsilon_{n+1} \tag{4.4}$$

は乗算 $x \times y$ の真の像を包含しており，この式 (4.4) をアフィン演算における乗算の定義として良い．この考え方によれば，Stolfi の乗法は式 (4.3) の c,δ を

$$c = 0 \tag{4.5}$$

$$\delta = (\sum_{i=1}^{n}|x_i|)(\sum_{i=1}^{n}|y_i|) = \sum_{i=1}^{n}\sum_{j=1}^{n}|x_iy_j| \tag{4.6}$$

で見積もった方法であると見做すことができる (この式 (4.6) が式 (4.3) を満たすのは明らかである)．

しかし，式 (4.6) の評価では同類項である $x_iy_j\varepsilon_i\varepsilon_j$ と $x_jy_i\varepsilon_j\varepsilon_i$ をまとめずに絶対値和を取っている点で改善の余地を残している．また，式 (4.6) の評価では $x_iy_i\varepsilon_i^2$ の部分が常に非負となることを反映できていない．すなわち，$-1 \leq \varepsilon_i \leq 1$ のもとで $x_iy_i\varepsilon_i^2$ を単独で置き換えるなら本来

$$x_iy_i\varepsilon_i^2 = \frac{x_iy_i}{2} + \frac{x_iy_i}{2}\varepsilon_{\mathrm{new}},\,(-1 \leq \varepsilon_{\mathrm{new}} \leq 1) \tag{4.7}$$

とすべきところを，式 (4.6) では

$$x_iy_i\varepsilon_i^2 = x_iy_i\varepsilon_{\mathrm{new}} \tag{4.8}$$

と評価しており，その分だけ最終的な区間評価の性能を落としている (δ を大きくしている)．

これに対し，以上の問題点を考慮したうえで

$$\delta = \sum_{i=1}^{n}|\frac{x_iy_i}{2}| + \sum_{j=1}^{n-1}\sum_{k=j+1}^{n}|x_jy_k + x_ky_j| \tag{4.9}$$

とし，

$$x \times y = y_0x + x_0y + (-x_0y_0 + \sum_{i=1}^{n}\frac{x_iy_i}{2}) + \delta\varepsilon_{n+1} \tag{4.10}$$

を演算結果としたのが，アフィン演算における改良乗算である．この方法は Stolfi の乗法と比べて常に同等以上の評価を与えるが，その評価が最適であるとは限らない．なお，この方法の計算量は $O(n^2)$ である．

4.3 提案手法 (最良乗算)

本節ではアフィン演算の乗算において誤差項の係数の絶対値を理論的に最小に抑える計算方法 (最良乗算) を提案する．また本節ではこの最良乗算を実用時間で実行する手法を提案する．最良乗算の手順は以下のとおりである．

step1 $a = y_0, \quad b = x_0$ により a,b を決定する．

step2 点 (x, y) が取り得る領域を D,

$$d(x, y) = xy - (ax + by) \tag{4.11}$$

とおき，次のような $d_{\max}, \quad d_{\min}$ を求める．

$$d_{\max} = \max_{(x,y)\in D} d(x, y) \tag{4.12}$$

$$d_{\min} = \min_{(x,y)\in D} d(x, y) \tag{4.13}$$

step3

$$c = \frac{d_{\max} + d_{\min}}{2} \tag{4.14}$$

$$\delta = \frac{d_{\max} - d_{\min}}{2} \tag{4.15}$$

より c,δ を求め,

$$x \times y = ax + by + c + \delta\varepsilon_{n+1} \tag{4.16}$$

を結果とする．

□

この方法は Stolfi の乗法，改良乗算に比べて常に同等以上の評価を与え，その評価は最適である．**step1** における a, b の決定の仕方が最適であることを 4.3.1 節で証明し，**step2** の詳細を 4.3.2 節，4.3.3 節で説明する．

4.3.1 線形近似方法 (step1 について)

本節では乗算 $x \times y$ に対する線形近似の傾きを 4.3 節の **step1** のように決定することに関して，その最適性を保証する次の定理の証明を行う．

定理 1 領域 $D \subset \boldsymbol{R}^2$ を点 (x_0, y_0) を対称の中心とする点対称な有界閉集合とする．また,

$$d(a, b, x, y) = xy - (ax + by) \quad (a, b, x, y \in \boldsymbol{R}) \tag{4.17}$$

とし，$d_{\max}(a,b), d_{\min}(a,b),\ \delta(a,b)$ をそれぞれ次のように定める．

$$d_{\max}(a,b) = \max_{(x,y)\in D} d(a,b,x,y) \tag{4.18}$$

$$d_{\min}(a,b) = \min_{(x,y)\in D} d(a,b,x,y) \tag{4.19}$$

$$\delta(a,b) = \frac{d_{\max}(a,b) - d_{\min}(a,b)}{2} \tag{4.20}$$

このとき $\delta(a,b)$ に関して次式が成立する．

$$\min_{a,b\in \boldsymbol{R}} \delta(a,b) = \delta(y_0, x_0) \tag{4.21}$$

□

x, y がアフィン形式のとき，点 (x,y) の取り得る領域 (joint range) は 3.8 節で述べたように点 (x_0, y_0) を対称の中心とする点対称な有界閉集合となるから，定理 1 を適用できる．

この定理より，最良乗算に用いる線形近似の傾きは，Stolfi の乗算や改良乗算におけるそれと同じであることがわかる (ただしこれらの方法はいずれも $d_{\max}$ のかわりにその上界を，$d_{\min}$ のかわりにその下界を与えている．そして，その見積もりの優劣が最終的な性能の優劣に直結している)．

定理 1 は次のように証明される．

証明 1 $d_{\max}(y_0, x_0)$，$d_{\min}(y_0, x_0)$ を与える点 (x,y) をそれぞれ (p_1, q_1)，(p_2, q_2) とおく．

$$x \times y - (y_0 x + x_0 y) = (x - x_0)(y - y_0) - x_0 y_0 \tag{4.22}$$

であるため，点 (x_0, y_0) に関して点 (p_1, q_1) と点対称の位置にある点 (p_3, q_3)

$$(p_3, q_3) \triangleq (2x_0 - p_1, 2y_0 - q_1) \tag{4.23}$$

は $d_{\max}(y_0, x_0)$ を与え，点 (x_0, y_0) に関して点 (p_2, q_2) と点対称の位置にある点 (p_4, q_4)

$$(p_4, q_4) \triangleq (2x_0 - p_2, 2y_0 - q_2) \tag{4.24}$$

は $d_{\min}(y_0, x_0)$ を与える．このとき，$a, b \in \boldsymbol{R}$ のもとで常に

$$d_{\max}(a,b) \geq p_1 q_1 - (ap_1 + bq_1) \tag{4.25}$$

$$d_{\min}(a,b) \leq p_2 q_2 - (ap_2 + bq_2) \tag{4.26}$$

$$d_{\max}(a,b) \geq p_3 q_3 - (ap_3 + bq_3) \tag{4.27}$$

$$d_{\min}(a,b) \leq p_4 q_4 - (ap_4 + bq_4) \tag{4.28}$$

が成立していることに注意して，

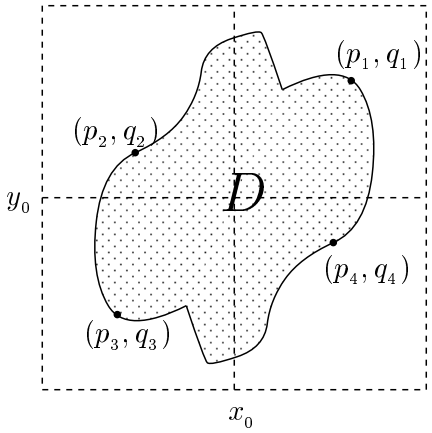

図 4.1: 定理 1 の証明

$$
\begin{aligned}
\delta(a,b) &= \frac{d_{\max}(a,b) - d_{\min}(a,b)}{2} \\
&\geq \frac{1}{2}\left[\frac{p_1q_1 - (ap_1 + bq_1)}{2} + \frac{p_3q_3 - (ap_3 + bq_3)}{2}\right] \\
&\quad -\frac{1}{2}\left[\frac{p_2q_2 - (ap_2 + bq_2)}{2} + \frac{p_4q_4 - (ap_4 + bq_4)}{2}\right] \\
&\quad (\text{式 } (4.25) - \text{式 } (4.28) \text{ より}) \\
&= \frac{p_1q_1 + p_3q_3 - p_2q_2 - p_4q_4}{4} - \frac{(p_1 + p_3) - (p_2 + p_4)}{4}a - \frac{(q_1 + q_3) - (q_2 + q_4)}{4}b \\
&= \frac{p_1q_1 + p_3q_3 - p_2q_2 - p_4q_4}{4} \\
&\quad (\text{式 } (4.23), \text{式 } (4.24) \text{ より}) \qquad (4.29)
\end{aligned}
$$

一方，明らかに

$$d_{\max}(y_0, x_0) = p_1q_1 - (y_0p_1 + x_0q_1) \qquad (4.30)$$

$$d_{\min}(y_0, x_0) = p_2q_2 - (y_0p_2 + x_0q_2) \qquad (4.31)$$

$$d_{\max}(y_0, x_0) = p_3q_3 - (y_0p_3 + x_0q_3) \qquad (4.32)$$

$$d_{\min}(y_0, x_0) = p_4q_4 - (y_0p_4 + x_0q_4) \qquad (4.33)$$

が成立していることに注意して，

$$\delta(y_0, x_0) = \frac{d_{\max}(y_0, x_0) - d_{\min}(y_0, x_0)}{2}$$

$$
\begin{aligned}
&= \frac{1}{2}\left[\frac{p_1q_1-(y_0p_1+x_0q_1)}{2}+\frac{p_3q_3-(y_0p_3+x_0q_3)}{2}\right] \\
&\quad -\frac{1}{2}\left[\frac{p_2q_2-(y_0p_2+x_0q_2)}{2}+\frac{p_4q_4-(y_0p_4+x_0q_4)}{2}\right] \\
&\quad (\text{式}\,(4.30)-\text{式}\,(4.33)\ \text{より}) \\
&= \frac{p_1q_1+p_3q_3-p_2q_2-p_4q_4}{4}-\frac{(p_1+p_3)-(p_2+p_4)}{4}y_0-\frac{(q_1+q_3)-(q_2+q_4)}{4}x_0 \\
&= \frac{p_1q_1+p_3q_3-p_2q_2-p_4q_4}{4} \\
&\quad (\text{式}\,(4.23),\text{式}\,(4.24)\ \text{より}) \qquad (4.34)
\end{aligned}
$$

ゆえに

$$\min_{a,b\in\boldsymbol{R}}\delta(a,b)=\delta(y_0,x_0) \qquad (4.35)$$

であることが示され，ここに題意は満たされた．

□

4.3.2 $d_{\max}, d_{\min}$ の求め方 (step2 について)

本節では **step2** の詳細を説明する．まず最初に次の定理を証明する．

定理 2 $x, y \in \boldsymbol{R}$ において，点 (x, y) の取り得る領域 $D \subset \boldsymbol{R}^2$ が，有界閉集合であるとする．また，

$$d(x,y)=xy-(ax+by),\quad (a,b\in\boldsymbol{R}) \qquad (4.36)$$

とし，$d_{\max}, d_{\min}$ をそれぞれ次のように定める．

$$
\begin{aligned}
d_{\max} &= \max_{(x,y)\in D} d(x,y) && (4.37)\\
d_{\min} &= \min_{(x,y)\in D} d(x,y) && (4.38)
\end{aligned}
$$

このとき，$d_{\max}, d_{\min}$ を与える点 (x, y) はいずれも領域 D の境界 ∂D 上に存在する．

□

x, y がアフィン形式のとき，領域 D は 3.8 節で述べたように有界閉集合となるから，定理 2 を適用できる．

証明 2

$$d(x,y)=xy-(ax+by) \qquad (4.39)$$

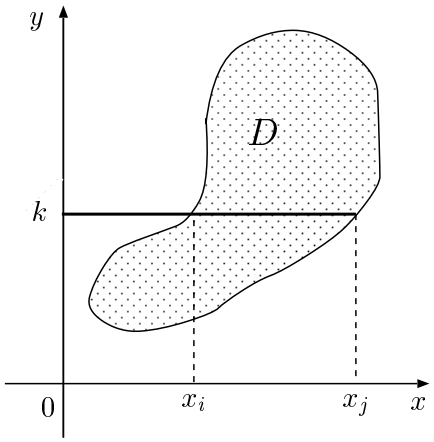

図 4.2: 定理 2 の証明

において，y を固定すると，$d(x,y)$ は x について線形となり，$d(x,y)$ が最大，最小となるのは x の値が x の取り得る領域の上端，あるいは下端のときだけである．例えば図 4.2 において，$y=k \quad (k \in [\underline{y}, \overline{y}])$ と固定したとき，$d(x,y)$ が最大，最小となるのは $x=x_i$ あるいは $x=x_j$ のときである．よって題意は満たされた．

□

この定理より，$d_{\max}, d_{\min}$ を求めるためには領域 D の辺上の点だけ調べればよいことが分かる．そこで領域 D のある辺上での $d(x,y)$ の最大値，最小値求め，この最大値，最小値を $d_{\max}, d_{\min}$ の候補とすることを考える．証明 1 より，点 (x_0, y_0) に関して点対称な二辺は同一の候補を与えるので，ある辺上での $d(x,y)$ の最大値，最小値 ($d_{\max}, d_{\min}$ の候補) を得たら，その辺と点 (x_0, y_0) に関して点対称な辺上に関してはこの操作を行う必要は無い．ゆえに領域 D の辺上での $d(x,y)$ の最大値，最小値を得るという操作を点 (x_0, y_0) に関して点対称でない n 本の辺に対して行うことにより，全部で $2n$ 個の候補を得ることができる．この $2n$ 個の候補内の最大値，最小値が $d_{\max}, d_{\min}$ である．

4.3.3 領域 D の辺の求め方 (最良乗算を実用時間で実行する方法)

領域 D の辺上での $d(x,y)$ の最大値，最小値を得るためには，$2n$ 個の領域 D の辺を得る必要があるので，本節ではその方法について考える．まず凸 $2n$ 角形を形成する領域 D の辺の候補を得ることを考える．領域 D の辺の候補は，

$$x = x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n \tag{4.40}$$

$$y = y_0 + y_1\varepsilon_1 + \cdots + y_n\varepsilon_n \tag{4.41}$$

に対して，式 (4.40)，式 (4.41) 中のどれか 1 個の ε だけをパラメータとし，他の ε を 1 または -1 に固定することによって得られる．ゆえに

$$
\begin{aligned}
&x = x_0 + \alpha_x + x_i\varepsilon_i \\
&y = y_0 + \alpha_y + y_i\varepsilon_i \quad (i = 1, 2, \cdots, n) \qquad (4.42)\\
&\text{ただし} \\
&\alpha_x = \sum_{k=1,k\neq i}^{n} x_k\varepsilon'_k, \quad \alpha_y = \sum_{k=1,k\neq i}^{n} y_k\varepsilon'_k \qquad (4.43)\\
&\varepsilon'_k = 1 \quad \text{or} \quad -1 \qquad (4.44)
\end{aligned}
$$

となる．パラメータとする ε_i をどれにするかで n 通り，さらにパラメータとする ε_i を決定したとき，その ε_i 以外の $n-1$ 個の ε を，1，-1 のどちらに固定するかで 2^{n-1} 通り存在するので，領域 D の辺の候補は全部で $n \times 2^{n-1}$ 個存在する．

そこで式 (4.42) を $d(x, y)$ に代入すると，$d(x, y)$ は ε_i に関する二次式 $d(\varepsilon_i)$ となり，$-1 \leq \varepsilon_i \leq 1$ の範囲における $d(\varepsilon_i)$ の最大値，最小値が領域 D の辺上での $d(x, y)$ の最大値，最小値 ($d_{\max}, d_{\min}$ の候補) である．前述のように，点 (x_0, y_0) に関して点対称な二辺は同一の候補を与えるので，ある辺上での $d(x, y)$ の最大値，最小値を得たら，その辺と点 (x_0, y_0) に関して点対称な辺に関してはこの操作を行う必要は無い．

だが領域 D の辺の候補全てを $d(x, y)$ に代入し，最大値，最小値を得ると，最良乗算の計算時間が $O(2^n)$ となってしまうので，n の増加時に指数関数的に増加してしまう．よって最良乗算が，n の増加時には実用時間で実行することができなくなってしまう．

そこで次に $n \times 2^{n-1}$ 個の候補の中から $2n$ 個の領域 D の辺を得ることを考える．式 (4.40)，式 (4.41) 中の $\varepsilon_i (i = 1, 2, \cdots, n)$ を消去すると

$$
y_i x - x_i y - x_0 y_i + y_0 x_i + \sum_{k=1,k\neq i}^{n} (-y_i x_k + x_i y_k)\varepsilon_k = 0 \qquad (4.45)
$$

となる．この式中の ε_k を 1，-1 のどちらかに固定したものが，式 (4.42) の ε_i を消去したものである．領域 D は凸 $2n$ 角形なので，領域 D の辺は式 (4.45) 中の定数項が最大，または最小となるものである．式 (4.44) を

$$
\varepsilon'_k = \begin{cases} 1 & (-y_i x_k + x_i y_k \geq 0) \\ -1 & (-y_i x_k + x_i y_k < 0) \end{cases} \qquad (4.46)
$$

とすると式 (4.45) 中の定数項が最大となり，式 (4.44) を

$$
\varepsilon'_k = \begin{cases} -1 & (-y_i x_k + x_i y_k \geq 0) \\ 1 & (-y_i x_k + x_i y_k < 0) \end{cases} \qquad (4.47)
$$

とすると式 (4.45) 中の定数項が最小となるので，式 (4.44) を式 (4.46) あるいは式 (4.47) とすれば，$n \times 2^{n-1}$ 個の候補を得る必要が無くなり，$2n$ 個の領域 D の辺の式を直接得ることができる．なお，式 (4.46)，式 (4.47) により得られる二辺は点 (x_0, y_0) に関して点対称であ

るので，どちらか片方の辺を $d(x,y)$ に代入し最大値，最小値を得たら，もう片方の辺に関してはこの操作を行う必要は無い．n 個の辺のみを $d(x,y)$ に代入し，最大値，最小値を得るだけで済むと，最良乗算の計算時間は $O(n^2)$ となるので，n の増加時にも指数関数的に増加することはない．よって最良乗算を n の増加時にも実用時間で実行することができる．

4.4 数値例

本節ではいくつかの数値例により，前章で提案した提案手法 (最良乗算) とこれまでの乗算の方法との比較を行い，最良乗算の有効性を確認する．なお，筆者が数値実験を行った計算機環境は CPU:Pentium IV 3.4GHz，メモリ:4.0GB，コンパイラ: Microsoft VisualC++ Ver. 6.0 であり，全ての計算を倍精度浮動少数点数を用いて行っている．

4.4.1 数値例 1

$p \in [0,2], q \in [1,3]$ において，

$$(p + 0.5q + 0.5) \times (3p - q + 4) \tag{4.48}$$

の値を計算する．変量 p, q は $p = 1 + \varepsilon_1$, $q = 2 + \varepsilon_2$ のようなアフィン形式で表すことができ，

$$\begin{cases} x = p + 0.5q + 0.5 \\ y = 3p - q + 4 \end{cases} \tag{4.49}$$

とおいたところに，p, q を代入することで，次の式を得る．

$$\begin{cases} x = 2.5 + \varepsilon_1 + 0.5\varepsilon_2 \\ y = 5 + 3\varepsilon_1 - \varepsilon_2 \end{cases} \tag{4.50}$$

これに対し，乗算のそれぞれの方法のプログラムを実行する．上式を，これらのプログラムでは

```
x0=2.5
x1=1.
x2=0.5
y0=5.
y1=3.
y2=-1.
```

と入力する．Stolfi の乗法を行うプログラムで得られる出力結果は，

```
z0=12.500000
z1=12.500000
z2=0.000000
z3=6.000000
```

となる．これらの出力結果のうち z_0〜z_3 が x/y を

$$x \times y = z_0 + z_1\varepsilon_1 + z_2\varepsilon_2 + z_3\varepsilon_3 \tag{4.51}$$

のように表わしている．
同様に，改良乗算を行うプログラムで得られる出力結果は，

```
z0=13．750000
z1=12．500000
z2=0．000000
z3=2．250000
```

となり，最良乗算を行うプログラムで得られる出力結果は，

```
z0=1．302083
z1=12．500000
z2=0．000000
z3=1．822917
```

となる．追加される誤差項の係数 z_3 は一般には打ち消しが起きず，ここでの過大評価は区間評価の性能を落とす原因となる．ゆえにこの場合，$|z_3|$ が小さい乗法ほど，シャープな値域の評価を与える乗法である．よってこの例では最良乗算が最もシャープな値域の評価を与える乗法であることが確認できる．

数値例 1 では明確に与えられたアフィン形式 x, y に対して全ての方法で乗算 $x \times y$ を行った．このとき追加される ε の係数 δ の値，実行時間 (μsec) を表 4.1 にまとめておく．なお，Stolfi の乗法，改良乗算，最良乗算を行ったときに与えられる δ を $\delta_s, \delta_i, \delta_b$ とし，実行時間を t_s, t_i, t_b とする．

4.4.2 数値例 1 に関する考察

表 4.1 より，最良乗算が Stolfi の乗法，改良乗算よりも小さな δ を与えることが確認できる．二番目の数値例においては δ の値が 3 つの方法で等しいが，これは入力した変数 x, y 間に相関がないためである．また表 4.1 より最良乗算の計算時間は Stolfi の乗法，改良乗算よりも多少大きいものの，比較に足るものであることも確認できる．

表 4.1: 明確に与えられた x, y に対する Stolfi の乗法，改良乗算，最良乗算の性能比較

与えられたアフィン形式		
δ_s	δ_i	δ_b
t_s	t_i	t_b

$x = -5 + 3\varepsilon_1 - \varepsilon_2, \quad x \in [-9, -1]$ $y = -1.5 - 0.3\varepsilon_1 - 0.2\varepsilon_2, \quad y \in [-2, -1]$		
2.000000	0.850000	0.618750
0.172000	0.188000	0.500000

$x = 3 + \varepsilon_1, \quad x \in [2, 4]$ $y = 4 + 2\varepsilon_2, \quad y \in [2, 6]$		
2.000000	2.000000	2.000000
0.156000	0.187000	0.390000

$x = 5 + 2\varepsilon_1 - 0.6\varepsilon_2 + 0.4\varepsilon_3, \quad x \in [-9, -1]$ $y = 5 - 0.7\varepsilon_1 - 1.1\varepsilon_2 + 0.2\varepsilon_3, \quad y \in [-2, -1]$		
6.000000	3.530000	2.412321
0.250000	0.328000	0.672000

$x = -28 + 10\varepsilon_1 - 3\varepsilon_2 - 4\varepsilon_3 + 5\varepsilon_4, \quad x \in [-50, -6]$ $y = -4.5 - \varepsilon_1 + 0.3\varepsilon_2 - 0.2\varepsilon_3, \quad y \in [-6, -3]$		
33.00000	21.95000	13.61250
0.312000	0.735000	0.969000

$x = -19.5 - 4\varepsilon_1 + 0.5\varepsilon_2 - 2\varepsilon_3 + \varepsilon_4 + 3\varepsilon_5, \quad x \in [-30, -9]$ $y = -16 - 0.3\varepsilon_1 + 1.6\varepsilon_2 + 0.6\varepsilon_3 + 0.1\varepsilon_4 + 1.4\varepsilon_5, \quad y \in [-20, -12]$		
42.00000	32.45000	21.00849
0.390000	0.734000	1.578000

$x = 64 + 7\varepsilon_1 + 8\varepsilon_2 - 15\varepsilon_3 + 13\varepsilon_4 + 2\varepsilon_5 + 12\varepsilon_6, \quad x \in [7, 121]$ $y = 47.5 + 20\varepsilon_1 + \varepsilon_2 - 14\varepsilon_3 + 3\varepsilon_4 + 6\varepsilon_5 + 1.5\varepsilon_6, \quad y \in [2, 93]$		
2593.500	2380.000	1543.234
0.485000	1.172000	1.515000

4.4.3 数値例 2

アフィン形式

$$x = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + \cdots + x_n\varepsilon_n \tag{4.52}$$

$$y = y_0 + y_1\varepsilon_1 + y_2\varepsilon_2 + \cdots + y_n\varepsilon_n \tag{4.53}$$

同士の乗算 $x \times y$ に関して，Stolfi の乗法，改良乗算，最良乗算をそれぞれ行う．このとき $\frac{\delta_i}{\delta_s}, \frac{\delta_b}{\delta_s}$ を 10000 回算出したときの平均を表 4.2 に，$\frac{t_i}{t_s}, \frac{t_b}{t_s}$ を 10000 回算出したときの平均を表 4.3 にそれぞれ示す．なお，係数 $x_0, \cdots, x_n, y_0, \cdots, y_n$ は表 4.2，4.3 中の条件を満たす乱数として与えた．

4.4.4 数値例 2 に関する考察

表 4.2 より Stolfi の乗法により与えられた δ の絶対値に対して，改良乗算によるそれは 7 割程度であるのに対し，最良乗算によるそれは半分程度であることが分かる．この例からも最良乗算の有効性が確認できる．

また最良乗算の計算時間は Stolfi の乗法，改良乗算よりも多少大きいものの，比較に足るものであることが表 4.3 からも確認できる．特に表 4.3 より ε の増加時に $\frac{t_i}{t_s}$ の平均，$\frac{t_b}{t_s}$ の平均 が増加していることが確認できる．これは Stolfi の乗法の計算量が $O(n)$ なのに対し，改良乗算，最良乗算の計算量が $O(n^2)$ であるという事実と一致した結果といえる．

表 4.2: それぞれの方法で乗算 $(x \times y)$ を行ったときの δ の比の平均値の比較

ε の係数の条件	ε の個数	$\frac{\delta_i}{\delta_s}$ の平均	$\frac{\delta_b}{\delta_s}$ の平均
$-1 \leq x_i \leq 1$ $-1 \leq y_i \leq 1$	4	0.728984	0.513756
$-1 \leq x_i \leq 1$ $-1 \leq y_i \leq 1$	10	0.735067	0.456805
$-0.1 \leq x_i \leq 0.1$ $-0.1 \leq y_i \leq 0.1$	4	0.768755	0.526211
$-0.1 \leq x_i \leq 0.1$ $-0.1 \leq y_i \leq 0.1$	10	0.743259	0.465285
$-0.01 \leq x_i \leq 0.01$ $-0.01 \leq y_i \leq 0.01$	4	0.755748	0.482209
$-0.01 \leq x_i \leq 0.01$ $-0.01 \leq y_i \leq 0.01$	10	0.706588	0.434766

$(-10 \leq x_0 \leq 10, \quad -10 \leq y_0 \leq 10)$

表 4.3: それぞれの方法で乗算 $(x \times y)$ を行ったときの実行時間の比の平均値の比較

ε の係数の条件	ε の個数	$\frac{t_i}{t_s}$ の平均	$\frac{t_b}{t_s}$ の平均
$-1 \leq x_i \leq 1$ $-1 \leq y_i \leq 1$	4	1.483871	1.952621
$-1 \leq x_i \leq 1$ $-1 \leq y_i \leq 1$	10	2.974898	3.016410
$-0.1 \leq x_i \leq 0.1$ $-0.1 \leq y_i \leq 0.1$	4	1.548387	1.803020
$-0.1 \leq x_i \leq 0.1$ $-0.1 \leq y_i \leq 0.1$	10	2.825889	3.102564
$-0.01 \leq x_i \leq 0.01$ $-0.01 \leq y_i \leq 0.01$	4	1.535417	1.937500
$-0.01 \leq x_i \leq 0.01$ $-0.01 \leq y_i \leq 0.01$	10	2.778739	2.962802

$(-10 \leq x_0 \leq 10, \quad -10 \leq y_0 \leq 10)$

4.4.5 数値例3

次式で表されるロジスティクス写像の値域の包含を得ることを考える．

$$x_{i+1} = 3.5x_i(1 - x_i) \qquad (i = 0, \cdots, m-1) \tag{4.54}$$

ここで，定義域（入力区間）を $x_0 \in [0.1, 0.101]$ とする．使用した方法は以下のとおり．

方法 A 通常の区間演算を使用して式 (4.54) の値域（x_m の変域）の包含を得る．

方法 B 乗算に Stolfi の乗法を採用したアフィン演算を使用して式 (4.54) の値域の包含を得る．

方法 C 乗算に改良乗算を採用したアフィン演算を使用して式 (4.54) の値域の包含を得る．

方法 D 乗算に最良乗算を採用したアフィン演算を使用して式 (4.54) の値域の包含を得る．

様々な m に対して方法 A から方法 D を適用したときの，式 (4.54) の値域の包含の区間幅を表 4.4 に，実行時間（秒）を表 4.5 に示す．なお，表中の “–” は求める包含の上端，あるいは下端の計算時にオーバーフローが起きたことを意味する．

表 4.4: 方法 A から方法 D が与える包含の区間幅の比較

m	**方法 A**	**方法 B**	**方法 C**	**方法 D**
10	1.40895e+11	8.43160e-3	7.46694e-3	1.44726e-3
20	–	2.66065e-2	1.10738e-2	6.64208e-8
30	–	6.89483e-4	9.28337e-7	1.33020e-10
40	–	2.88862e-6	1.86072e-9	1.75282e-15
50	–	3.81372e-11	2.45189e-14	3.51088e-18

表 4.5: 方法 A から方法 D の実行時間（秒）の比較

m	**方法 A**	**方法 B**	**方法 C**	**方法 D**
10	0.000031	0.0046	0.0062	0.0078
20	–	0.0234	0.0265	0.0281
30	–	0.0516	0.0546	0.0625
40	–	0.0859	0.0968	0.1140
50	–	0.1375	0.1703	0.1906

4.4.6 数値例 3 に関する考察

表 4.4 より，方法 A では区間幅の爆発的な増大が起きているのに対し，方法 B，方法 C，方法 D では値域がシャープに包含されていることが確認できる．さらに全ての方法の中でも方法 D が最もシャープな包含を与えている．よってこの数値例からも最良乗算の性能の高さを確認できる．

表 4.5 より，方法 B，方法 C，方法 D は方法 A に比べて非常に多くの計算時間を必要とすることが確認できる．しかしながら，方法 B，方法 C，方法 D はこの弱点を補って余りあるだけのシャープな包含を与えている．アフィン演算を使用する方法（方法 B，方法 C，方法 D）同士の実行時間に関しては，数値例 2，数値例 3 で与えた結果と同様の傾向が確認できる．

4.5 むすび

本章ではアフィン演算における乗算について述べた．まずこれまでの方法として，Stolfi の乗法，改良乗算の説明を行った．次に最適な評価を与える方法 (最良乗算) を提案し，さらにこの最良乗算を実用時間で行う方法を提案した．最後に，いくつかの数値例により最良乗算の有効性を確認した．

次章ではアフィン演算における除算について述べる．まずこれまでの方法として，文献 [38][39] の方法，柏木の方法，白井の方法について説明する．次に 3 つの方法を提案する (提案手法 1〜提案手法 3)．最後に，いくつかの数値例によりアフィン演算の除算における全ての方法を比較 · 検証し，提案手法の有効性を示す．

第 5 章

アフィン演算における除算

5.1 はじめに

本章ではアフィン演算における除算に関して述べる．乗算同様，第三章で述べられた非線形二項演算の問題は除算においても例外ではなく，アフィン演算の現状の除算では最良の計算は実現できておらず，改良の余地を残していた．そこで本章ではアフィン演算を，特に除算において整備・補完することによって，アフィン演算の実用化に貢献することを目的としている．アフィン形式同士の除算については原論文では定義されておらず，従来はx/y=x(1/y) のように，逆数をとる非線形単項演算と乗算の合成として計算していた (文献[38][39] の方法)．しかしこのように除算を 2 つの非線形演算に分解すると，1 回の演算で誤差項が 2 つ新しく追加されてしまい，区間評価の性能を落としてしまう．そこで柏木雅英[44][45] は除算 x/y を分解せずに直接計算する方法を 1999 年に提案した (柏木の方法)．この方法は非線形二変数関数 f(x,y)=x/y に対して直接線形近似を行う方法で，計算時間を優先して，複雑な場合分けが必要な部分を近似することにより，計算を一部簡略化している．この柏木の方法における複雑な場合分けが必要な部分を緻密に計算した方法が白井健一 (柏木雅英研究室修士)[46][47] により 2000 年に提案された (白井の方法)．この方法は変数 x,y 間に相関がある場合には最適な評価を与えることはできない．ゆえに変数 x，y 間に相関がある場合に白井の方法よりも新しく追加される誤差項の係数の絶対値を小さくすることが可能である．

本章ではまずこれまでの方法として文献 [38][39] の方法，柏木の方法，白井の方法について説明する．次に白井の方法よりも新しく追加される誤差項の係数の絶対値を小さくすることが期待できる 3 つの方法を提案する (提案手法 1〜提案手法 3)．本章の最後においてはいくつかの数値例によりアフィン演算の除算における全ての方法を比較 · 検証し，提案手法の有効性を示す．

5.2 これまでの方法

本節ではアフィン演算における除算について，これまでの方法として文献 [38][39] の方法，柏木の方法，白井の方法の説明を行う．

5.2.1 文献 [38][39] の方法

1.2 節で述べたように，アフィン 演算の除算においては文献 [38][39] で定義すらされておらず，文献 [38][39] では x/y は

$$\frac{x}{y} = x \times \frac{1}{y} \tag{5.1}$$

のように式変形され，逆関数をとる非線形単項演算と Stolfi の乗法の組み合わせにより計算されていた．この方法では 2 つの非線形演算となるため新しい ε を 2 つも追加する必要があり，かつ変数 x, y 間に相関があるときに Stolfi の乗法が δ を甘く見積もるため，区間評価の性能を落としてしまう．なお，この方法の計算量は $O(n)$(n は ε の個数) である．

5.2.2 柏木の方法

本節では除算を 1 回の非線形演算で行う柏木の方法を説明する．本方法は $\frac{x}{y}$ を $ax+by+c$ で近似する方法であるので，

$$\max_{\substack{\underline{x}\le x\le\overline{x}\\ \underline{y}\le y\le\overline{y}}} |\frac{x}{y} - (ax+by+c)| \tag{5.2}$$

をなるべく小さくする a，b，c を決定する問題を考える．ただし x, y の変域を $[\underline{x}, \overline{x}], [\underline{y}, \overline{y}]$ とする．

ここで y を固定すると，$\frac{x}{y} - (ax+by+c)$ は x について線形，すなわち $|\frac{x}{y} - (ax+by+c)|$ を x の関数と見たとき最大となるのは，$x = \underline{x}$ または $x = \overline{x}$ のときである．すなわち式 (5.2) を小さくする問題は，

$$\max\left(\max_{\underline{y}\le y\le\overline{y}} |\frac{\underline{x}}{y} - (a\underline{x}+by+c)|,\ \max_{\underline{y}\le y\le\overline{y}} |\frac{\overline{x}}{y} - (a\overline{x}+by+c)|\right) \tag{5.3}$$

を小さくする問題と等価である．ここで b を固定し，

$$\begin{aligned} p &= \max_{\underline{y}\le y\le\overline{y}} (\frac{\underline{x}}{y} - by) \\ q &= \min_{\underline{y}\le y\le\overline{y}} (\frac{\underline{x}}{y} - by) \\ r &= \max_{\underline{y}\le y\le\overline{y}} (\frac{\overline{x}}{y} - by) \\ s &= \min_{\underline{y}\le y\le\overline{y}} (\frac{\overline{x}}{y} - by) \end{aligned} \tag{5.4}$$

とすると，式 (5.3) は

$$\max\left(\frac{p-q}{2}, \frac{r-s}{2}\right) \tag{5.5}$$

を下回らないことは明らかである．また a，c を

$$a\underline{x} + c = \frac{p+q}{2} \tag{5.6}$$

$$a\overline{x} + c = \frac{r+s}{2} \tag{5.7}$$

を満たすように定めると式 (5.3) の値は式 (5.5) の値に一致する．したがって δ は式 (5.5) で与えられ，a，c は式 (5.6)，式 (5.7) を満たすように決めれば良い．よって式 (5.5) を小さくするような b を求める問題に帰着する．そこで，x をパラメータとする y の関数

$$f_x(y) = \frac{x}{y} - by \tag{5.8}$$

を考えその $\underline{y} \leq y \leq \overline{y}$ における最大最小を考えるとグラフの形から表 5.1 のようになる．$e_x(b) =$($f_x(y)$ の最大値 $-f_x(y)$ の最小値）とすると，p，q は $x = \underline{x}$ のときの最大最小，r，s は $x = \overline{x}$ のときの最大最小であるから，式 (5.3) を小さくする問題は

$$\max(e_{\underline{x}}(b), e_{\overline{x}}(b)) \tag{5.9}$$

を小さくする問題になる．$e_{\underline{x}}(b)$ と $e_{\overline{x}}(b)$ のグラフは図 5.1 のようになるので，この 2 つのグラフの交点で最小値をとる．

グラフの式はわかっているので交点を求めることは可能であるが，場合分けが複雑になり，計算時間も増加する．そこで $e_x(b)$ の近似

$$\tilde{e}_x(b) = (\overline{y} - \underline{y})|b + \frac{x}{\underline{y}\overline{y}}| \tag{5.10}$$

を使って b を近似的に算出する．このとき b は，

$$b = -\frac{1}{\underline{y}\overline{y}} \frac{\underline{x} + \overline{x}}{2} \tag{5.11}$$

で求められる．

b から a, c を計算するには，次の方法が簡単である．式 (5.4) を計算すれば，式 (5.5)，式 (5.6) 式 (5.7) により，a，c，δ は計算できる．p, q, r, s は，$\dfrac{x}{y} - by$ のグラフの形より高々6 点における関数評価で計算できる．

p, q は，点 $\underline{y}$，　点 $\overline{y}$，点 $\sqrt{-\dfrac{\underline{x}}{b}}$ ($b \neq 0$, $-\dfrac{\underline{x}}{b} > 0$ で $\underline{y} < \sqrt{-\dfrac{\underline{x}}{b}} < \overline{y}$ の場合のみ) において関数 $\dfrac{\underline{x}}{y} - by$ を計算したときの最大，最小である．

r, s は，点 $\underline{y}$，　点 $\overline{y}$，点 $\sqrt{-\dfrac{\overline{x}}{b}}$ ($b \neq 0$, $-\dfrac{\overline{x}}{b} > 0$ で $\underline{y} < \sqrt{-\dfrac{\overline{x}}{b}} < \overline{y}$ の場合のみ) において関数 $\dfrac{\overline{x}}{y} - by$ を計算したときの最大，最小である．

なお，この方法の計算量は $O(n)$ である．

5.2.3　白井の方法

この方法は，柏木の方法に基づき，複雑な場合分けをした結果最良の b を求めたものである．

$$\underline{x} > 0 \quad \text{かつ} \quad -\frac{\overline{x}}{\overline{y}^2} \geq -\frac{\underline{x}}{\underline{y}^2} \tag{5.12}$$

のとき，

$$A = \frac{\overline{x}\underline{y} - \underline{x}\overline{y}}{\underline{y}\overline{y}} \tag{5.13}$$

$$B = A(\overline{y} - \underline{y}) + 2(-\underline{x} - \overline{x} + 2\sqrt{\underline{x}\overline{x}}) \tag{5.14}$$

$$A' = \frac{\underline{x}\underline{y} - \overline{x}\overline{y}}{\underline{y}\overline{y}} \tag{5.15}$$

$$B' = A'(\overline{y} - \underline{y}) + 2(-\underline{x} - \overline{x} + 2\sqrt{\underline{x}\overline{x}}) \tag{5.16}$$

表 5.1: $f_x(y) = \dfrac{x}{y} - by$ の最大値と最小値

($x \geq 0$ の時)

b の値	最大値	最小値
$b \geq -\dfrac{x}{\overline{y}^2}$	$\dfrac{x}{\underline{y}} - b\underline{y}$	$\dfrac{x}{\overline{y}} - b\overline{y}$
$b \leq -\dfrac{x}{\underline{y}^2}$	$\dfrac{x}{\overline{y}} - b\overline{y}$	$\dfrac{x}{\underline{y}} - b\underline{y}$
$-\dfrac{x}{\underline{y}^2} < b < -\dfrac{x}{\overline{y}^2}$	$\max\left(\dfrac{x}{\underline{y}} - b\underline{y}, \dfrac{x}{\overline{y}} - b\overline{y}\right)$	$2\sqrt{-bx}$

($x < 0$ の時)

b の値	最大値	最小値
$b \leq -\dfrac{x}{\overline{y}^2}$	$\dfrac{x}{\overline{y}} - b\overline{y}$	$\dfrac{x}{\underline{y}} - b\underline{y}$
$b \geq -\dfrac{x}{\underline{y}^2}$	$\dfrac{x}{\underline{y}} - b\underline{y}$	$\dfrac{x}{\overline{y}} - b\overline{y}$
$-\dfrac{x}{\overline{y}^2} < b < -\dfrac{x}{\underline{y}^2}$	$2\sqrt{-bx}$	$\min\left(\dfrac{x}{\underline{y}} - b\underline{y}, \dfrac{x}{\overline{y}} - b\overline{y}\right)$

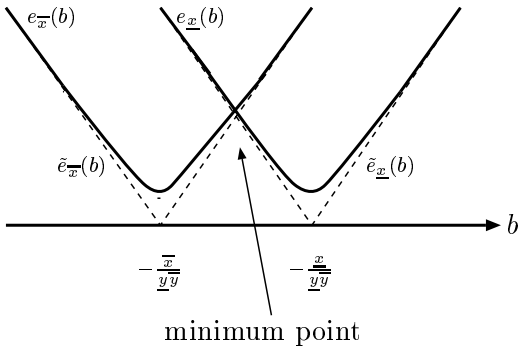

図 5.1: $e_x(b)$ と $e_{\overline{x}}(b)$

として，b の値の候補には

$$b = -\frac{(\sqrt{\overline{x}}-\sqrt{\underline{x}})^2}{4\underline{y}^2} \tag{5.17}$$

$$b = \frac{B \pm \sqrt{B^2-(\overline{y}-\underline{y})^2A^2}}{(\overline{y}-\underline{y})^2} \tag{5.18}$$

$$b = -\frac{(\sqrt{\overline{x}}-\sqrt{\underline{x}})^2}{4\overline{y}^2} \tag{5.19}$$

$$b = \frac{B' \pm \sqrt{B'^2-(\overline{y}-\underline{y})^2A'^2}}{(\overline{y}-\underline{y})^2} \tag{5.20}$$

の 6 つがあり，これらの中で条件

$$-\frac{\overline{x}}{\underline{y}\overline{y}} \le b \le -\frac{\underline{x}}{\underline{y}\overline{y}} \tag{5.21}$$

を満たすもののうち，最小の値をとるものを選べばよい．また，

$$\overline{x} < 0 \quad \text{かつ} \quad -\frac{\underline{x}}{\underline{y}^2} \ge -\frac{\overline{x}}{\overline{y}^2} \tag{5.22}$$

のとき，

$$A = \frac{\underline{x}\overline{y}-\overline{x}\underline{y}}{\underline{y}\overline{y}} \tag{5.23}$$

$$B = A(\overline{y}-\underline{y}) - 2(-\underline{x}-\overline{x}-2\sqrt{\underline{x}\overline{x}}) \tag{5.24}$$

$$A' = \frac{\overline{x}\overline{y}-\underline{x}\underline{y}}{\underline{y}\overline{y}} \tag{5.25}$$

$$B' = A'(\overline{y}-\underline{y}) - 2(-\underline{x}-\overline{x}-2\sqrt{\underline{x}\overline{x}}) \tag{5.26}$$

として，b の値の候補には

$$b = -\frac{(\sqrt{-\overline{x}}-\sqrt{-\underline{x}})^2}{4\underline{y}^2} \tag{5.27}$$

$$b = \frac{-B \pm \sqrt{B^2-(\overline{y}-\underline{y})^2A^2}}{(\overline{y}-\underline{y})^2} \tag{5.28}$$

$$b = -\frac{(\sqrt{-\overline{x}}-\sqrt{-\underline{x}})^2}{4\overline{y}^2} \tag{5.29}$$

$$b = \frac{-B' \pm \sqrt{B'^2-(\overline{y}-\underline{y})^2A'^2}}{(\overline{y}-\underline{y})^2} \tag{5.30}$$

の 6 つがあり，これらの中で条件

$$-\frac{\overline{x}}{\underline{y}\overline{y}} \le b \le -\frac{\underline{x}}{\underline{y}\overline{y}} \tag{5.31}$$

を満たすもののうち，最小の値をとるものを選べばよい．それ以外の場合には，

$$b = -\frac{1}{\underline{y}\overline{y}}\frac{\underline{x}+\overline{x}}{2} \tag{5.32}$$

とする．a と c と δ の選び方は柏木の方法と全く同様である．この方法は変数 x, y 間に相関がない場合には最適な評価を与える．だがこの方法は変数 x, y 間に相関がある場合には最適な評価を与えることができない．

なお，この方法の計算量は $O(n)$ である．

5.3 アフィン演算の除算における提案手法

本節ではアフィン演算の除算において 3 つの方法を提案する．以後，これらの方法を提案手法 1，提案手法 2，提案手法 3 と呼ぶ．

5.3.1 提案手法 1

この方法は，逆数をとる非線形単項演算と 4.2.2 節で述べた改良乗算を用いて，$x \times (1/y)$ を行う方法である．文献 [38][39] の方法と同様，1 回の除算で 2 回の非線形演算を行うため，2 つの ε が追加されてしまうが，変数 x, y 間の相関が強いときに改良乗算が効果を発揮するのでこの方法はこれまでの方法よりも小さな δ を与えることが期待できる．アフィン形式

$$x = x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n \tag{5.33}$$

$$y = y_0 + y_1\varepsilon_1 + \cdots + y_n\varepsilon_n \tag{5.34}$$

に対して，この方法で除算 x/y を行うことを考える．y の変域が $[\underline{y}, \overline{y}]$ であるとき 3.6 節より $1/y$ は

$$\begin{aligned}
\frac{1}{y} &= -\frac{1}{\underline{y}\overline{y}}y_0 + \frac{\underline{y}+\overline{y}+2\sqrt{\underline{y}\overline{y}}}{2\underline{y}\overline{y}} + (-\frac{y_1}{\underline{y}\overline{y}})\varepsilon_1 + \cdots + (-\frac{y_n}{\underline{y}\overline{y}})\varepsilon_n + \frac{\underline{y}+\overline{y}-2\sqrt{\underline{y}\overline{y}}}{2\underline{y}\overline{y}}\varepsilon_{n+1} \\
&\quad (0 < \underline{y} \text{のとき}) \\
\frac{1}{y} &= -\frac{1}{\underline{y}\overline{y}}y_0 + \frac{\underline{y}+\overline{y}-2\sqrt{\underline{y}\overline{y}}}{2\underline{y}\overline{y}} + (-\frac{y_1}{\underline{y}\overline{y}})\varepsilon_1 + \cdots + (-\frac{y_n}{\underline{y}\overline{y}})\varepsilon_n + \frac{-\underline{y}-\overline{y}-2\sqrt{\underline{y}\overline{y}}}{2\underline{y}\overline{y}}\varepsilon_{n+1} \\
&\quad (\overline{y} < 0 \text{ のとき})
\end{aligned} \tag{5.35}$$

となる．これを

$$\frac{1}{y} = z_0 + \sum_{i=1}^{n} z_i\varepsilon_i + z_{n+1}\varepsilon_{n+1} \tag{5.36}$$

とおき，改良乗算を用いると，$x \times (1/y)$ は，

$$\begin{aligned} x \times \frac{1}{y} &= (x_0 z_0 + \sum_{i=1}^{n} \frac{x_i z_i}{2}) + \sum_{i=1}^{n} (x_0 z_i + x_i z_0)\varepsilon_i \\ &\quad + (\sum_{i=1}^{n} |\frac{x_i z_i}{2}| + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} |x_i z_j + x_j z_i| + \sum_{i=0}^{n} |x_i z_{n+1}|)\varepsilon_{n+1} \end{aligned} \tag{5.37}$$

で与えられる．この方法の計算量は $O(n^2)$ である．

5.3.2 提案手法 2

この方法は，逆数をとる非線形単項演算と 4.3 節で述べた最良乗算を用いて，$x \times (1/y)$ を行う方法である．文献 [38][39] の方法と同様，1 回の除算で 2 回の非線形演算を行うため，2 つの ε が追加されてしまうが，変数 x, y 間の相関が強いときに最良乗算が効果を発揮するのでこの方法は提案手法 1 よりも小さな δ を与えることが期待でき，さらにこれまでの方法より小さな δ を与える場合が期待できる．アフィン形式

$$x = x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n \tag{5.38}$$

$$y = y_0 + y_1\varepsilon_1 + \cdots + y_n\varepsilon_n \tag{5.39}$$

に対して，この方法で除算 x/y を行う場合の手順は以下のとおりである．

step1 y の変域が $[\underline{y}, \overline{y}]$ であるとき 3.6 節より $1/y$ は

$$\begin{aligned} \frac{1}{y} &= -\frac{1}{\underline{y}\overline{y}} y_0 + \frac{\underline{y} + \overline{y} + 2\sqrt{\underline{y}\overline{y}}}{2\underline{y}\overline{y}} + (-\frac{y_1}{\underline{y}\overline{y}})\varepsilon_1 + \cdots + (-\frac{y_n}{\underline{y}\overline{y}})\varepsilon_n + \frac{\underline{y} + \overline{y} - 2\sqrt{\underline{y}\overline{y}}}{2\underline{y}\overline{y}}\varepsilon_{n+1} \\ &\quad (0 < \underline{y} \text{のとき}) \\ \frac{1}{y} &= -\frac{1}{\underline{y}\overline{y}} y_0 + \frac{\underline{y} + \overline{y} - 2\sqrt{\underline{y}\overline{y}}}{2\underline{y}\overline{y}} + (-\frac{y_1}{\underline{y}\overline{y}})\varepsilon_1 + \cdots + (-\frac{y_n}{\underline{y}\overline{y}})\varepsilon_n + \frac{-\underline{y} - \overline{y} - 2\sqrt{\underline{y}\overline{y}}}{2\underline{y}\overline{y}}\varepsilon_{n+1} \\ &\quad (\overline{y} < 0 \text{ のとき}) \end{aligned} \tag{5.40}$$

である．これを

$$z = z_0 + z_1\varepsilon_1 + \cdots + z_n\varepsilon_n + z_{n+1}\varepsilon_{n+1} \tag{5.41}$$

とおく．

step2 点 (x, z) が取り得る領域を領域 D，また $d(x, z)$ を

$$d(x, z) = xz - (z_0 x + x_0 z) \tag{5.42}$$

とおき，次のような $d_{\max}$，　$d_{\min}$ を求める．

$$d_{\max} = \max_{(x,z)\in D} d(x, z) \tag{5.43}$$

$$d_{\min} = \min_{(x,z)\in D} d(x, z) \tag{5.44}$$

$d_{\max}$，　$d_{\min}$ を求める具体的な方法については 4.3.2 節，4.3.3 節を参照されたい．

step3

$$c = \frac{d_{\max} + d_{\min}}{2} \tag{5.45}$$

$$\delta_{\mathrm{BM}} = \frac{d_{\max} - d_{\min}}{2} \tag{5.46}$$

より c,δ_{BM} を求め,

$$\frac{x}{y} = x \times z = z_0 x + x_0 z + c + \delta_{\mathrm{BM}} \varepsilon_{n+2} \tag{5.47}$$

を結果とする.

□

この方法の計算量は $O(n^2)$ である.

5.3.3 提案手法 3

この方法では柏木の方法，白井の方法と同様に除算の演算結果を

$$\frac{x}{y} = ax + by + c + \delta\varepsilon_{n+1} \tag{5.48}$$

と記述する．a, b に関しては白井の方法と同様の手順で求める．これにより a, b は長方形領域 $X \times Y$ に対して最適なものとなる．そしてその a, b を用いて領域 D に対して最適な c, δ を求める．この方法の手順を以下に示す.

step1 白井の方法により a,b を求める.

step2 点 (x, y) が取り得る領域を領域 D，また $d(x, y)$ を

$$d(x, y) = \frac{x}{y} - (ax + by) \tag{5.49}$$

とおき，次のような $d_{\max}$, $d_{\min}$ を求める.

$$d_{\max} = \max_{(x,y)\in D} d(x, y) \tag{5.50}$$

$$d_{\min} = \min_{(x,y)\in D} d(x, y) \tag{5.51}$$

step3

$$c = \frac{d_{\max} + d_{\min}}{2} \tag{5.52}$$

$$\delta = \frac{d_{\max} - d_{\min}}{2} \tag{5.53}$$

より c,δ を求め,

$$\frac{x}{y} = ax + by + c + \delta\varepsilon_{n+1} \tag{5.54}$$

を結果とする.

□

この方法における a,b の取り方は常に最適とは限らないが，与えられた a, b に対する c，δ の取り方は常に最適である．この方法は柏木の方法，白井の方法に比べ，常に同等以下の δ を与えることができる．

なお，この方法の計算量は $O(n^2)$ である．

$d_{\max}, d_{\min}$ の求め方，すなわち **step2** の詳細を以下に説明する．まず最初に次の定理を証明する．

定理 3 $d_{\max}, d_{\min}$ を与える点 は常に領域 D の境界 ∂D 上に存在する．

□

証明 3

$$d(x, y) = \frac{x}{y} - (ax + by) \tag{5.55}$$

において，y を固定すると，$d(x, y)$ は x について線形となり，$d(x, y)$ が最大，最小となるのは x の値が x の取り得る領域の上端，あるいは下端のときだけである．ゆえに図 5.2 において，$y = k \quad (k \in [\underline{y},\overline{y}])$ と固定したとき，$d(x, y)$ が最大，最小となるのは $x = x_i$ あるいは $x = x_j$ のときである．(図 5.2 は $0 < \underline{x}$ かつ $0 < \underline{y}$ で，ε が 2 個のときの領域 D の例であるが，他の場合も同様である)．よって題意は満たされた．

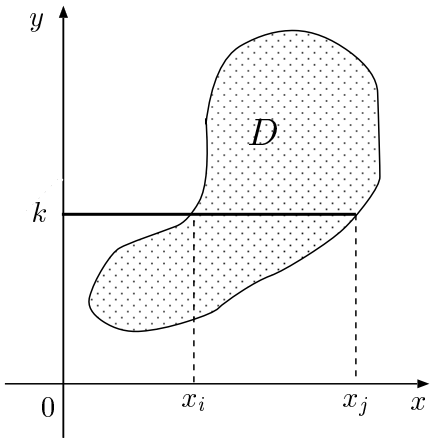

図 5.2: 定理 3 の証明

□

この定理より，$d_{\max}, d_{\min}$ を求めるためには領域 D の辺上の点だけ調べればよいことが分かる．そこで領域 D のある辺上での $d(x, y)$ の最大値，最小値を求め，この最大値，最小値を $d_{\max}, d_{\min}$ の候補とすることを考える．ゆえに領域 D の辺上での $d(x, y)$ の最大値，最小値を得るという操作を点 (x_0, y_0) に関して点対称でない n 本の辺に対して行うことにより，全部で $2n$ 個の候補を得ることができる．この $2n$ 個の候補内の最大値，最小値が $d_{\max}$, $d_{\min}$ である．

領域 D の辺上での $d(x, y)$ の最大値，最小値を得るためには，$2n$ 個の領域 D の辺を得る必要がある．$2n$ 個の領域 D の辺を得る方法については第 4.3.3 節を参照されたい．4.3.3 節に示した方法により得られた式 (4.42) を $d(x, y)$ に代入すると，$d(x, y)$ は ε_i に関する式 $d(\varepsilon_i)$ となり，$-1 \leq \varepsilon_i \leq 1$ の範囲における $d(\varepsilon_i)$ の最大値，最小値が領域 D の辺上での $d(x$, $y)$ の最大値，最小値 ($d_{\max}, d_{\min}$ の候補) となる．

5.4 数値例

本章ではいくつかの数値例により，前章で提案した 3 つの提案手法とこれまでの除算の方法との比較を行い，提案手法の有効性を示す．なお，筆者が数値実験を行った計算機環境は CPU:Pentium IV 3.4GHz，メモリ:4.0GB，コンパイラ: Microsoft VisualC++ Ver. 6.0 であり，全ての計算を倍精度浮動少数点数を用いて行っている．

5.4.1 数値例 1

$p \in [0, 2], q \in [1, 3]$ において，

$$(p + 0.5q + 0.5)/(3p - q + 4) \tag{5.56}$$

の値を計算する．変量 p, q は $p = 1 + \varepsilon_1, q = 2 + \varepsilon_2$ のようなアフィン形式で表すことができ，

$$\begin{cases} x = p + 0.5q + 0.5 \\ y = 3p - q + 4 \end{cases} \tag{5.57}$$

とおいたところに，p, q を代入することで，次の式を得る．

$$\begin{cases} x = 2.5 + \varepsilon_1 + 0.5\varepsilon_2 \\ y = 5 + 3\varepsilon_1 - \varepsilon_2 \end{cases} \tag{5.58}$$

これに対し,3 つの方法のプログラムを実行する．上式を, これらのプログラムでは

```
x0=2.5
x1=1.
x2=0.5
y0=5.
y1=3.
y2=-1.
```

と入力する．文献 [38][39] の方法を行うプログラムで得られる出力結果は,

```
z0=0.833333
z1=-0.500000
z2=0.444444
z3=1.555556
```

となる．出力結果 z_0〜z_3 が x/y を

$$\frac{x}{y} = z_0 + z_1\varepsilon_1 + z_2\varepsilon_2 + z_3\varepsilon_3 \tag{5.59}$$

のように表わしている．
同様に, 柏木の方法を行うプログラムで得られる出力結果は,

```
z0=1.123903
z1=-0.379873
z2=0.504508
z3=1.084796
```

となり，白井の方法を行うプログラムで得られる出力結果は,

```
z0=1.052083
z1=-0.631944
z2=0.534722
z3=1.003472
```

となり, 提案手法 1 を行うプログラムで得られる出力結果は,

```
z0=0.694444
z1=-0.500000
z2=0.444444
z3=1.138889
```

となり，提案手法 2 を行うプログラムで得られる出力結果は,

```
z0=0.582176
z1=-0.500000
z2=0.444444
z3=0.971065
```

となり，提案手法 3 を行うプログラムで得られる出力結果は,

```
z0=0.805509
z1=-0.631944
z2=0.534722
z3=0.694491
```

となる．追加される誤差項の係数 z_3 は一般には打ち消しが起きず，ここでの過大評価は区間評価の性能を落とす原因となる．ゆえにこの場合，$|z_3|$ が小さい除法ほど，シャープな値域の評価を与える除法である．よってこの例では提案手法 3 が最もシャープな値域の評価を与える除法であることが確認できる．

数値例 1 では明確に与えられたアフィン形式 x, y に対して全ての方法で除算 x/y を行った．このとき追加される ε の係数 δ の値，実行時間 (μsec) を表 4.1 にまとめておく．

なお，文献 [38][39] の方法，柏木の方法，白井の方法，提案手法 1，提案手法 2，提案手法 3 を行ったときの δ を $\delta_j, \delta_k, \delta_s, \delta_1, \delta_2, \delta_3$ とし，実行時間を $t_j, t_k, t_s, t_1, t_2, t_3$ とする．ここで文献 [38][39] の方法，提案手法 1，提案手法 2 では 2 回の非線形演算が必要となるため，$\delta_j, \delta_1, \delta_2$ は 2 回の非線形演算により新しく追加された 2 個の ε の係数の絶対値和とする．

5.4.2 数値例 1 に関する考察

表 5.2 より提案手法 1，提案手法 2 は文献 [38][39] の方法よりも小さな δ を与えていることが確認でき，柏木の方法，白井の方法に対しては数値例により δ の大小関係は様々であることが確認できる．

表 5.2 の 2 番目の数値例においては δ が文献 [38][39] の方法，提案手法 1，提案手法 2 の間で等しく，白井の方法と提案手法 3 で等しいが，これは乗算の場合と同様，入力した変数 x, y 間に相関がないためである．また提案手法 2，提案手法 3 は提案手法 1 よりも小さな δ を与えていることも表 5.2 より確認できる．これは提案手法 2，提案手法 3 は提案手法 1 よりもさらに変数 x, y 間の相関を考慮して除算を行う方法だからであると考えられる．

提案手法 1，提案手法 2 は柏木の方法，白井の方法に比べ，変数 x, y 間の相関が考慮されているという長所を持つ反面，2 回の非線形演算を行わなければならないという短所を持っている．ゆえに表 5.2 において提案手法 1 が柏木の方法，白井の方法よりも小さな δ を与えている数値例や提案手法 2 が柏木の方法，白井の方法，提案手法 3 よりも小さな δ を与えている数値例では，この方法の長所の方が短所よりも強く反映されており，上記とは逆の結果を与えている数値例では，この方法の短所の方が長所よりも強く反映されている．特に提案手法 1，提案手法 2 は x の区間幅に対して，y の区間幅が狭いときほど δ が小さくなることが確認できる．その理由は，逆数をとる非線形単項演算で追加される誤差項の絶対値が，x, y の区間幅と比べて十分小さくなるためであると考えられる．

また表 5.2 で示した例では提案手法 1，提案手法 2 は柏木の方法，白井の方法よりも高速であることが確認できる．これらの方法は文献 [38][39] の方法よりも速度は劣るものの比較に足る速度で実行できることが確認できる．提案手法 3 の実行時間は 3 番目の以降の数値例おいて全ての方法の中で最も大きいことが確認できる．これは，提案手法 3 は白井の方法における複雑な b を算出法を行った上でさらに計算量が $O(n^2)$ である c, δ の算出を行っているためであると考えられる．それぞれの方法の速度に対するさらなる考察は 5.4.4 節で与えることにする．

表 5.2: 明確に与えられた x, y に対するそれぞれの除法の性能比較

与えられたアフィン形式					
δ_j	δ_k	δ_s	δ_1	δ_2	δ_3
t_j	t_k	t_s	t_1	t_2	t_3

$x = -5 + 3\varepsilon_1 - \varepsilon_2, \quad x \in [-9, -1]$ $y = -1.5 - 0.3\varepsilon_1 - 0.2\varepsilon_2, \quad y \in [-2, -1]$					
1.386038	1.168861	1.072330	0.811039	0.568019	0.628396
0.704000	1.546500	2.345500	0.548000	1.222000	2.083000

$x = 3 + \varepsilon_1, \quad x \in [2, 4]$ $y = 4 + 2\varepsilon_2, \quad y \in [2, 6]$					
0.345299	0.250000	0.234784	0.345299	0.345299	0.234784
0.720000	1.501000	2.534938	0.548000	0.784000	2.286000

$x = 5 + 2\varepsilon_1 - 0.6\varepsilon_2 + 0.4\varepsilon_3, \quad x \in [2, 8]$ $y = 5 - 0.7\varepsilon_1 - 1.1\varepsilon_2 + 0.2\varepsilon_3, \quad y \in [3, 7]$					
0.444733	0.310345	0.300645	0.327114	0.258521	0.211095
0.798000	1.751240	2.816199	0.924000	1.332000	3.258000

$x = -28 + 10\varepsilon_1 - 3\varepsilon_2 - 4\varepsilon_3 + 5\varepsilon_4, \quad x \in [-50, -6]$ $y = -4.5 - \varepsilon_1 + 0.3\varepsilon_2 - 0.2\varepsilon_3, \quad y \in [-6, -3]$					
2.548220	2.111616	1.954758	1.934331	1.294132	1.541524
1.018000	2.154334	3.078334	1.050000	1.551000	4.432000

$x = -19.5 - 4\varepsilon_1 + 0.5\varepsilon_2 - 2\varepsilon_3 + \varepsilon_4 + 3\varepsilon_5, \quad x \in [-30, -9]$ $y = -16 - 0.3\varepsilon_1 + 1.6\varepsilon_2 + 0.6\varepsilon_3 + 0.1\varepsilon_4 + 1.4\varepsilon_5, \quad y \in [-20, -12]$					
0.238508	0.182368	0.178908	0.198717	0.144897	0.103517
1.174000	2.472275	3.600275	1.614000	1.848000	5.263000

$x = 64 + 7\varepsilon_1 + 8\varepsilon_2 - 15\varepsilon_3 + 13\varepsilon_4 + 2\varepsilon_5 + 12\varepsilon_6, \quad x \in [7, 121]$ $y = 47.5 + 20\varepsilon_1 + \varepsilon_2 - 14\varepsilon_3 + 3\varepsilon_4 + 6\varepsilon_5 + 1.5\varepsilon_6, \quad y \in [2, 93]$					
35.971935	24.141613	22.769912	34.824086	28.043138	17.872021
1.300000	2.665095	4.415979	1.676000	2.099000	6.517000

5.4.3 数値例 2

アフィン形式

$$x = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + \cdots + x_n\varepsilon_n \tag{5.60}$$

$$y = y_0 + y_1\varepsilon_1 + y_2\varepsilon_2 + \cdots + y_n\varepsilon_n \tag{5.61}$$

同士の除算 x/y に関して，それぞれの方法を行う．このとき $\frac{\delta_k}{\delta_j}, \frac{\delta_s}{\delta_j}, \frac{\delta_1}{\delta_j}, \frac{\delta_2}{\delta_j}, \frac{\delta_3}{\delta_j}$ を 10000 回算出したときの平均を表 5.3 に，$\frac{t_k}{t_j}, \frac{t_s}{t_j}, \frac{t_1}{t_j}, \frac{t_2}{t_j}, \frac{t_3}{t_j}$ を 10000 回算出したときの平均を表 5.4 にそれぞれ示す．なお，係数 $x_0, \cdots, x_n, y_0, \cdots, y_n$ は表 5.3，5.4 中の条件を満たす乱数として与えた．

5.4.4 数値例 2 に関する考察

表 5.3 より一般的な傾向として，提案手法 3 が文献 [38][39] の方法，柏木の方法，白井の方法，提案手法 1，提案手法 2 よりも小さな δ を与えていることが確認できる．

また表 5.4 より一般的な傾向として提案手法 1，提案手法 2 は柏木の方法，白井の方法よりも高速であり，文献 [38][39] の方法よりも速度は劣るものの比較に足る速度で実行できることが確認できる．これは，提案手法 1，提案手法 2 の計算量は $O(n^2)$ であり柏木の方法，白井の方法の計算量は $O(n)$ であるものの，柏木の方法，白井の方法における a, b の算出のための計算量の大きさがその差を上回っているためであると考えられる．また白井の方法と同様の手順で a, b を求める提案手法 3 の計算量が他の方法に比べて大きいことが確認できる．これも前述の白井の方法における a, b の算出のための計算量に対する考察と提案手法 3 の計算量は $O(n^2)$ であるという事実に一致した結果といえる．個々の方法の実行時間を見ると，表 5.4 より，$\frac{t_1}{t_j}$ の平均，$\frac{t_2}{t_j}$ の平均，$\frac{t_3}{t_j}$ の平均は ε の増加とともに増加していることが確認できる．これは文献 [38][39] の方法の計算量が $O(n)$ なのに対し，提案手法 1，提案手法 2，提案手法 3 の計算量が $O(n^2)$ であるという事実と一致した結果といえる．

表 5.3: それぞれの方法で除算 (x/y) を行ったときの δ の比の平均値の比較

ε の係数の条件	ε の個数	$\frac{\delta_k}{\delta_j}$ の平均	$\frac{\delta_s}{\delta_j}$ の平均	$\frac{\delta_1}{\delta_j}$ の平均	$\frac{\delta_2}{\delta_j}$ の平均	$\frac{\delta_3}{\delta_j}$ の平均
$-1 \leq x_i \leq 1$ $-1 \leq y_i \leq 1$	4	0.736306	0.724533	0.833454	0.673349	0.565599
$-1 \leq x_i \leq 1$ $-1 \leq y_i \leq 1$	10	0.696362	0.658858	0.854832	0.698948	0.484408
$-0.1 \leq x_i \leq 0.1$ $-0.1 \leq y_i \leq 0.1$	4	0.751361	0.746013	0.912231	0.720222	0.705010
$-0.1 \leq x_i \leq 0.1$ $-0.1 \leq y_i \leq 0.1$	10	0.751469	0.745568	0.813764	0.589040	0.495333
$-0.01 \leq x_i \leq 0.01$ $-0.01 \leq y_i \leq 0.01$	4	0.750268	0.749698	0.715123	0.563804	0.245882
$-0.01 \leq x_i \leq 0.01$ $-0.01 \leq y_i \leq 0.01$	10	0.750542	0.749089	0.833443	0.665291	0.539594

$(-10 \leq x_0 \leq 10, \quad 2 \leq y_0 \leq 10)$

表 5.4: それぞれの方法で除算 (x/y) を行ったときの実行時間の比の平均値の比較

ε の係数の条件	ε の個数	$\frac{t_k}{t_j}$ の平均	$\frac{t_s}{t_j}$ の平均	$\frac{t_1}{t_j}$ の平均	$\frac{t_2}{t_j}$ の平均	$\frac{t_3}{t_j}$ の平均
$-1 \leq x_i \leq 1$ $-1 \leq y_i \leq 1$	4	2.202174	3.040884	1.172043	1.344086	3.333333
$-1 \leq x_i \leq 1$ $-1 \leq y_i \leq 1$	10	2.332525	3.172097	1.919786	2.005348	5.828877
$-0.1 \leq x_i \leq 0.1$ $-0.1 \leq y_i \leq 0.1$	4	2.199569	3.038278	1.172043	1.344086	3.333333
$-0.1 \leq x_i \leq 0.1$ $-0.1 \leq y_i \leq 0.1$	10	2.000101	3.384397	1.768473	1.926108	6.157635
$-0.01 \leq x_i \leq 0.01$ $-0.01 \leq y_i \leq 0.01$	4	2.202309	3.041019	1.172043	1.677419	3.440860
$-0.01 \leq x_i \leq 0.01$ $-0.01 \leq y_i \leq 0.01$	10	2.067748	2.841147	1.768473	1.847291	6.157635

$(-10 \leq x_0 \leq 10, \quad 2 \leq y_0 \leq 10)$

5.4.5 数値例3

次式で表されるような写像の値域の包含を得ることを考える.

$$x_{i+1} = \frac{a_i x_i + b_i}{c_i x_i + d_i}, \quad (i = 0, \cdots, m-1) \tag{5.62}$$

ここで $a_i, b_i, c_i, d_i,\ i = 0, \cdots, m-1$ は $a_i \in [5, 15]$, $b_i \in [5, 15]$, $c_i \in [5, 15]$, $d_i \in [5, 15]$ を満たす乱数で与えた. また mid, rad を $mid \in [5, 15]$, $rad \in [0, 1]$ を満たす乱数で与え, 定義域（入力区間）を $x_0 \in [mid - rad, mid + rad]$ とした. 使用した方法は以下のとおり.

方法 A 通常の区間演算を使用して式 (5.62) の値域（x_m の変域）の包含を得る.

方法 B 除算に文献 [38][39] の方法を採用したアフィン演算を使用して式 (5.62) の値域の包含を得る.

方法 C 除算に柏木の方法を採用したアフィン演算を使用して式 (5.62) の値域の包含を得る.

方法 D 除算に白井の方法を採用したアフィン演算を使用して式 (5.62) の値域の包含を得る.

方法 E 除算に提案手法 1 を採用したアフィン演算を使用して式 (5.62) の値域の包含を得る.

方法 F 除算に提案手法 2 を採用したアフィン演算を使用して式 (5.62) の値域の包含を得る.

方法 G 除算に提案手法 3 を採用したアフィン演算を使用して式 (5.62) の値域の包含を得る.

様々な m に対して方法 A から方法 G を適用したときの, 式 (5.62) の値域の包含の区間幅を表 5.5 に, 実行時間（秒）を表 5.6 に示す.

表 5.5: 方法 A から方法 G が与える包含の区間幅の比較

m	2	4	6	8	10
方法 A	2.26524e-1	6.12670e-1	2.66380e-1	1.85210e-2	6.89950e-2
方法 B	5.42150e-4	2.57075e-4	2.42670e-10	3.72665e-11	3.04008e-13
方法 C	3.91549e-4	2.08071e-4	2.02816e-10	3.57800e-11	2.64724e-13
方法 D	3.76863e-4	2.02907e-4	2.00490e-10	3.57658e-11	2.63627e-13
方法 E	3.62469e-4	1.97748e-4	1.90830e-10	3.52429e-11	2.50610e-13
方法 F	3.41752e-4	1.90316e-4	1.88218e-10	3.52260e-11	2.50138e-13
方法 G	7.58926e-5	9.51283e-5	9.36544e-11	3.14791e-11	1.45664e-13

表 5.6: 方法 A から方法 G の実行時間（秒）の比較

m	2	4	6	8	10
方法 A	1.40e-6	2.96e-6	4.21e-6	5.46e-6	6.87e-6
方法 B	9.69e-5	3.11e-4	6.50e-4	1.11e-3	1.69e-3
方法 C	4.69e-5	1.47e-4	2.88e-4	4.81e-4	7.22e-4
方法 D	6.72e-5	1.81e-4	3.44e-4	5.55e-4	8.16e-4
方法 E	9.53e-5	3.23e-4	7.14e-4	1.26e-3	2.01e-3
方法 F	9.53e-5	3.45e-4	7.88e-4	1.49e-3	2.53e-3
方法 G	6.87e-5	2.24e-4	5.09e-4	9.83e-4	1.73e-3

5.4.6 数値例 3 に関する考察

表 5.5 より，方法 B から方法 D は方法 A に比べシャープな包含を与えていることが分かる．また方法 E，方法 F，方法 G が方法 B，方法 C，方法 D よりもシャープな包含を与えていることから，数値例 3 は本論文で提案した手法がこれまでの方法よりもシャープな包含を与える例であるといえる．数値例 1，数値例 2 では柏木の方法，白井の方法の方が提案手法 1 よりも小さな δ を与えることがあったが，数値例 3 では提案手法 1 の方が柏木の方法，白井の方法よりもシャープな包含を与えている．これは，数値例 3 では数値例 1，数値例 2 よりも変数間の相関の強い除算が行われているためであると考えられる．

表 5.6 より，方法 A 以外の方法（アフィン演算を使用する方法）は方法 A に比べて多くの計算時間を必要とすることが確認できる．しかしながら，方法 A 以外の方法はこの弱点を補って余りあるだけのシャープな包含を与えている．

次にアフィン演算を使用する方法同士の実行時間に関して述べる．表 5.6 より，方法 B，方法 E，方法 F の実行時間が方法 C，方法 D，方法 G の実行時間よりも多いことが確認できる．これは数値例 1，数値例 2 における実行時間に関する結果とは一致しない結果である．数値例 3 がこのような結果を与えた理由を以下に示す．柏木の方法，白井の方法，提案手法 3 では 1 回の除算で 1 個の ε が追加されるのに対し，Stolfi の乗法，提案手法 1，提案手法 2 では 1 回の除算で 2 個の ε が追加される．よって方法 C，方法 D，方法 G では式 (5.62) の各計算ステップで，$i+1$ 個の ε を持つアフィン形式同士の除算が行われるのに対し，方法 B，方法 E，方法 F では式 (5.62) の各計算ステップで，$2i+1$ 個の ε を持つアフィン形式同士の除算が行われる．したがって方法 B，方法 E，方法 F を実行する場合，方法 C，方法 D，方法 G の実行時よりも多くの ε を持ったアフィン形式同士の除算が各計算ステップにおいて行われているため，より多くの実行時間が必要となるのである．

一方で m の増加につれて方法 G の実行時間が方法 B，方法 E，方法 F の実行時間に近づいていくことが確認できる（$m = 10$ のときには方法 G の方が方法 B よりも多くの計算時間を必要としている）．これより，m の増加につれて，追加される ε の個数が 2 個であることよりも計算量が $O(n^2)$ であることの方が計算時間を考える際に大きな要因となっていく

ことが確認できる．

方法 B，方法 E，方法 F 同士の実行時間に関しては，数値例 2，数値例 3 で与えた結果と同様の傾向が確認できる．方法 C，方法 D，方法 G 同士の実行時間に関しても，数値例 2，数値例 3 で与えた結果と同様の傾向が確認できる．

5.5 むすび

本章ではアフィン演算における除算について述べた．まずこれまでの方法として，文献 [38][39] の方法，柏木の方法，白井の方法について説明した．次に 3 つの方法を提案した (提案手法 1〜提案手法 3)．最後に，いくつかの数値例によりアフィン演算における全ての除算の方法を比較，検証し，提案手法の有効性を示した．

次章では区間演算で評価する関数を多項式に限定した場合の評価方法について述べる．まずこれまでの方法として Horner 法，平均値形式に基づく方法，Alefeld の方法，アフィン演算を使用する方法について説明する．次に 5 つの方法を提案する (提案手法 1〜提案手法 5)．次章の最後においていくつかの数値例を行うことにより，多項式関数の値域を評価する全ての方法を比較·検証し，提案手法の有効性を示す．

第 6 章

多項式の値域の区間評価

6.1 はじめに

本章では多項式の値域の区間評価について述べる．値域を評価する関数を多項式関数に限定すると，多項式の性質を生かすことにより，区間演算で起こる区間幅の爆発的な増大を抑制するための効率的な方法を提案できる．本章では値域を評価する関数を多項式関数に限定し，区間演算で起こる区間幅の爆発的な増大を抑制し関数の値域のシャープな包含を得る方法を提案する．多項式関数の値域の評価において，区間演算で起こる区間幅の爆発的な増大を抑制する方法として，これまでに Horner 法 (例えば [18][48])，平均値形に基づく方法 (例えば [48]) ，Alefeld の方法 [49][50]，アフィン演算を使用する方法 [51] が提案されている．

本章ではまずまずこれまでの方法として Horner 法平均値形式に基づく方法，Alefeld の方法，アフィン演算を使用する方法について説明する．次に 5 つの方法を提案する (提案手法 1〜提案手法 5)．本章の最後においていくつかの数値例を行うことにより，多項式関数の値域を評価する全ての方法を比較 · 検証し，提案手法の有効性を示す．

以後，特に注釈が無い限り $X = [\underline{X}, \overline{X}]$ における多項式

$$f(x) = a_n x^n + \cdots + a_1 x + a_0 \tag{6.1}$$

の値域の評価について考える．

6.2 これまでの方法

本節では区間演算を用いた多項式の値域の評価方法について，これまでに提案されてきた方法として，Horner 法，平均値形式に基づく方法，Alefeld の方法，アフィン演算を使用する方法について説明する．なお，それらの説明の前に数値例における比較対象として単純に評価を行う通常法についても触れる．

6.2.1 通常法

通常法では単純に区間演算を用いて次式の計算を行う．

$$F(X) = a_n X^n + a_{n-1} X^{n-1} + a_{n-2} X^{n-2} + \cdots + a_2 X^2 + a_1 X + a_0 \tag{6.2}$$

1.1 節で述べたようにこの方法を用いると区間幅の爆発的な増大が起こる．なお，この方法の計算量は $O(n)$(n は多項式の次数) である．

6.2.2 Horner 法

Horner 法では式 (6.2) に対し，以下のような式変形を行った後区間演算を適用する．

$$F(X)_H = ((\cdots((a_n X + a_{n-1})X + a_{n-2})X + \cdots a_2)X + a_1)X + a_0 \tag{6.3}$$

以後，下付き文字 H はその多項式が Horner 法により評価されたことを示す．

一般に，区間 A, B, C において，次式が成立する．

$$AB + AC \supseteq A(B + C) \tag{6.4}$$

ゆえに式 (6.3) は式 (6.2) よりもシャープな評価を与えることができる．

また通常法の加算回数は n 回，乗算回数は $2n$ 回であるのに対し，Horner 法の加算回数，乗算回数はそれぞれ n 回である．よって Horner 法は通常法よりも高速に実行することができる．この方法の計算量が $O(n)$ であることは明らかである．

6.2.3 平均値形式に基づく方法

平均値形式に基づく方法では，$f(x)$ の導関数の値域を Horner 法により区間評価したもの，すなわち

$$F'(X)_H = (na_nX^{n-1} + \cdots + 2a_2X + a_1)_H \tag{6.5}$$

を求め，$x \in X$ における $f(x)$ の値域を次式により評価する．

$$f(y) + F'(X)_H(X - y) \tag{6.6}$$

ただし y は $y \in X$ を満たす任意の実数とし，6.4 節においては X の中点として数値実験を行っている．この方法の計算量は $O(n)$ である．

6.2.4 Alefeld の方法

Alefeld の方法は式 (6.6) 中の $F'(X)_H$ を

$$J = (\sum_{i=1}^{n}(\sum_{j=i}^{n}a_jy^{j-i})X^{i-1})_H \tag{6.7}$$

に置き換えて評価を行ったものである．すなわち式 (6.7) により得られた J を用いて，$x \in X$ における $f(x)$ の値域を次式により評価する．

$$f(y) + J(X - y) \tag{6.8}$$

Alefeld は文献 [49] の中で

$$J \subseteq F'(X) = na_nX^{n-1} + \cdots + 2a_2X + a_1 \tag{6.9}$$

であることを証明し，

$$J \subseteq F'(X)_H \tag{6.10}$$

であることを言及している．従って，Alefeld の方法は平均値の定理に基づく方法と比べて常に同等あるいはシャープな評価を与える．なお，この方法の計算量は $O(n^2)$ である．

6.2.5 アフィン演算を使用する方法

アフィン演算を使用する方法では次式をアフィン演算を用いて評価する．

$$F(X)_H = ((\cdots((a_nX + a_{n-1})X + a_{n-2})X + \cdots a_2)X + a_1)X + a_0 \tag{6.11}$$

なお，文献 [51] では，この方法における乗算に 4.2.2 節で述べた改良乗算を使用している．このため，6.4 節の数値実験でもこの方法におけるアフィン演算の乗算には改良乗算を使用した．この方法の計算コストは改良乗算を使用しているため $O(n^2)$ である．

6.3 提案手法

本節では多項式の値域の評価方法について，5 つの方法を提案する．

6.3.1 提案手法 1

この方法では入力区間 $X = [\underline{X}, \overline{X}]$ の中心 c

$$c = \frac{\underline{X} + \overline{X}}{2} \tag{6.12}$$

を用いて $Y = X - c$ を導入することにより式 (6.2) を以下のように式変形する．

$$B_nY^n + B_{n-1}Y^{n-1} + \cdots + B_2Y^2 + B_1Y + B_0 \tag{6.13}$$

係数 $B_n, B_{n-1}, \cdots, B_1, B_0$ は n 次多項式 $f_n(x)$ を $x - c$ で割ったときの商を $f_{n-1}(x)$ とおくと次式により求めることができる．

$$\begin{aligned}
B_0 &= f_n(x) - (x-c)f_{n-1}(x) \\
B_1 &= f_{n-1}(x) - (x-c)f_{n-2}(x) \\
&\vdots \\
B_{n-2} &= f_2 - (x-c)f_1 \\
B_{n-1} &= f_1 - (x-c)a_n \\
B_n &= a_n
\end{aligned} \tag{6.14}$$

ここで係数 $B_0, \cdots, B_n$ は数学的には実数であるが，計算機上でこの方法を実行する際には丸め誤差を考慮しなければならないため区間として求める．すなわち係数 $B_0, \cdots, B_n$ は $a_0, \cdots, a_n$ を用いて区間演算により算出される．具体的には以下のプログラムを実行するこ

とにより求めることができる．

```
for(i = 0; i <= n; i++){
  B_i = [a_i, a_i];
}
C = [c, c];
for(i = 1; i <= n; i++){
  for(j = n; j >= i; j--){
    B_{j-1} = B_{j-1} + a_j * C;
  }
}
```

式 (6.13) の変形により次の 2 つの効果が期待できる．

[**効果** 1]

入力区間 X とその中心 c に関して，一般に次式が成立する．

$$\max(|\underline{X}|, |\overline{X}|) \geq |\underline{X} - c| = |\overline{X} - c| \tag{6.15}$$

ゆえに区間 A の区間幅を $w(A)$ とおくと次式が成立する．

$$w(Y^n) \leq w(X^n) \quad (n = 1, 2, \cdots) \tag{6.16}$$

□

[**効果** 1 **の例**]

$X \in [1, 3]$ であるとき

$$X^3 = [1, 3]^3 = [1, 27]$$

となる．よって

$$w(X^3) = 27 - 1 = 26$$

である．これに対し区間 X の中心 c は

$$c = \frac{1 + 3}{2} = 2$$

なので，

$$(X - c)^3 = [1 - 2, 3 - 2]^3 = [-1, 1]^3 = [\mathbf{-1}, \mathbf{1}]$$

となる．よって

$$w((X - c)^3) = 1 - (-1) = 2$$

である．

□

[**効果** 2]

$$\overline{X} - c = r \tag{6.17}$$

とおくと，

$$Y = [-r, r] \tag{6.18}$$

となる．これにより偶数 $m \quad (m = 2, 4, \cdots, \quad m \leq n)$ に対し，

$$Y^m = [0, r^m] \tag{6.19}$$

となり，Y の偶数乗の値域の評価の負の部分を除去することができる．

□

[**効果** 2 **の例**]

$X \in [2, 8]$ であるとき

$$X^2 = [2, 8]^2 = [4, 64] \tag{6.20}$$

となる．よって

$$w(X^2) = 60$$

である．これに対し区間 X の中心 c は

$$c = \frac{2+8}{2} = 5$$

なので，

$$(X - c)^2 = [2 - 5, 8 - 3]^2 = [-3, 3]^2 = [\mathbf{0}, \mathbf{9}] \tag{6.21}$$

となる．よって

$$w((X - c)^2) = 9$$

である．

□

以上の 2 つの効果により式 (6.13) は式 (6.2) よりもシャープな値域を与えることが期待できる．

一方，式 (6.13) に直接 Horner 法を適用すると，Horner 法の各計算ステップにおいて区間の中心が移動してしまうため，前述の 2 つの効果が期待できなくなる．ゆえにこの形式に直接 Horner 法を適用するのは有効とは限らない．なお，この方法の計算量は $O(n^2)$ である．

6.3.2 提案手法 2

式 (6.13) に直接 Horner 法を適用するのは有効とは限らないと 6.3.1 節で述べた．そこでこの方法では

$$Y^2 = [0, r^2] \quad (r \text{ は } X \text{ の半径}) \tag{6.22}$$

を用いて式 (6.13) に Horner 法を適用することを考える．すなわち式 (6.13) を以下のように変形する．

$$\begin{aligned}&((\cdots((B_eY^2 + B_{e-2})Y^2 + B_{e-4})Y^2 + \cdots B_4)Y^2 + B_2)Y^2 \\ &+Y((\cdots((B_oY^2 + B_{o-2})Y^2 + B_{o-4})Y^2 + \cdots B_3)Y^2 + B_1) + B_0\end{aligned} \tag{6.23}$$

ここで e, o は n を上回ることのない最大の偶数，奇数であり，n を用いて以下のように表すことができる．

$$e = n - (n \bmod 2) \tag{6.24}$$

$$o = n - 1 + (n \bmod 2) \tag{6.25}$$

式 (6.13) を式 (6.23) に変形することにより，提案手法 1 の持つ 2 つの効果をほぼ失わずに計算できることが期待できる．さらにこの方法では Horner 法を用いているため，提案手法 1 と比べ同等以上のシャープな評価を与えることが期待できる．また，提案手法 2 の計算量は提案手法 1 のそれよりも小さくなることも期待できる．この方法の計算量は $O(n^2)$ である．

6.3.3 提案手法 3

X の中心 c，半径 r を用いて，変数 t を

$$t = x - c \tag{6.26}$$

とおく．すなわち

$$t \in Y = [-r, r] \tag{6.27}$$

である．$x \in X$ における $f(x)$ の値域は $t \in Y$ における次式の値域により包含することができる．

$$B_nt^n + B_{n-1}t^{n-1} + \cdots + B_2t^2 + B_1t + B_0 \tag{6.28}$$

この方法では式 (6.28) を

$$(\cdots(((\alpha)t^2 + B_lt + B_{l-1})t^2 + B_{l-2}t + B_{l-3})t^2 + \cdots + B_2)t^2 + B_1t + B_0 \tag{6.29}$$

のように変形する．ただし

$$\alpha = \begin{cases} B_n t^2 + B_{n-1} t + B_{n-2} & (n \bmod 2 = 0) \\ B_n t + B_{n-1} & (n \bmod 2 = 1) \end{cases} \tag{6.30}$$

$$l = \begin{cases} n-3 & (n \bmod 2 = 0) \\ n-2 & (n \bmod 2 = 1) \end{cases} \tag{6.31}$$

である．次にそれぞれのくくられた括弧内 (t^2 の係数が比較的幅の広い区間となった二次多項式) に対して，単純に区間演算により評価するのではなく，最大値，最小値を厳密に評価していく．この操作はそれぞれのくくられた括弧内は高々二次多項式であるので可能である．

具体的な手順を以下に示す．

step1 6.3.1 節で示した方法により係数 $B_0, \cdots, B_n$ を求める．

step2

$$\alpha = \begin{cases} B_n t^2 + B_{n-1} t + B_{n-2} & (n \bmod 2 = 0) \\ B_n t + B_{n-1} & (n \bmod 2 = 1) \end{cases} \tag{6.32}$$

とし，式 (6.33) を満たす実数 p，q を求める．

$$p = \min_{-r \le t \le r} \alpha \tag{6.33}$$

$$q = \max_{-r \le t \le r} \alpha \tag{6.34}$$

なお，計算途中に混入される丸め誤差を考慮すべく，p, q の算出には区間演算を使用する．区間として算出された最小値の下端を p とし，区間として算出された最大値の上端を q とする．

step3

$$m = \frac{n + (n \bmod 2)}{2} - 1 \tag{6.35}$$

$$l = \begin{cases} n-3 & (n \bmod 2 = 0) \\ n-2 & (n \bmod 2 = 1) \end{cases} \tag{6.36}$$

とし，$m = l, \cdots, 1$ の間次式に従い p, q を更新する．ここで m は 2 づつ減少していくことに注意する．

$$\begin{aligned} p &= \min_{t \in Y}([p,q]t^2 + B_m t + B_{m-1}) \\ &= \min\{\min_{t \in Y}(pt^2 + B_m t + B_{m-1}), \min_{t \in Y}(qt^2 + B_m t + B_{m-1})\} \qquad (6.37) \\ q &= \max_{t \in Y}([p,q]t^2 + B_m t + B_{m-1}) \\ &= \max\{\max_{t \in Y}(pt^2 + B_m t + B_{m-1}), \max_{t \in Y}(qt^2 + B_m t + B_{m-1})\} \qquad (6.38) \end{aligned}$$

step2 と同様に，計算途中に混入される丸め誤差を考慮すべく，p, q の算出には区間演算を使用する．区間として算出された最小値の下端を p とし，区間として算出された最大値の上端を q とする．

step4 求める値域の評価は $[p, q]$ となる．

□

この方法の計算量は $O(n^2)$ である．

6.3.4 提案手法 4

n 次多項式 $f(x)$ の値域の上限，下限の候補となる値は $f(\underline{X}),\quad f(\overline{X})$，あるいは $x \in X$ における $f(x)$ の極値である．ゆえにこの方法では $f(\underline{X}),\quad f(\overline{X})$ と $x \in X$ における $f(x)$ の極値を考慮することにより多項式の極値を評価することを考える．

n 次多項式 $f(x)$ をその導関数 $f'(x)$ ($n-1$ 次) で割ったときの商を $q(x)$(1 次)，剰余を $s(x)$($n-2$ 次) とおくと次式が成立する．

$$f(x) = f'(x)q(x) + s(x) \tag{6.39}$$

ここで，$f'(x) = 0$ を満たす x を $x' \quad (x' \in X)$ とおき，x' を式 (6.39) に代入すると，

$$f'(x')q(x') = 0 \tag{6.40}$$

より式 (6.39) は以下のようになる．

$$f(x') = s(x') \tag{6.41}$$

これより次式が成立する．

$$f(x') \in [\min_{x \in X} s(x), \max_{x \in X} s(x)] \tag{6.42}$$

この式より $x \in X$ における $f(x)$ の極値 $f(x')$ は $x \in X$ における $s(x)$ の値域の評価で包含できることが分かる．ゆえに $x \in X$ における $f(x)$ の極値 $f(x')$ の変わりに $s(x)$ の値域の評価の上限，下限を $f(x)$ の値域の上限，下限の候補としても，$f(x)$ の $x \in X$ における値域を包含することができる．

$s(x)$ の値域の上限，下限の候補となる値は当然 $s(\underline{X}), s(\overline{X})$ 及び $x \in X$ における $s(x)$ の極値である．$x \in X$ における $s(x)$ の極値は $s(x)$ の算出を再帰的に行う (現在の $s(x)$ を $f(x)$ とおき，再び $s(x)$ を求める) ことによりその包含が可能となる．具体的には以下のとおりである．$f(x)$ の次数が n なのに対し，$s(x)$ の次数は $n-2$ であるため，$s(x)$ の算出を一度行うたびに $s(x)$ の次数は 2 づつ減少していく．ゆえに $s(x)$ の算出を再帰的に行うことにより $s(x)$ の次数を減少させてゆき，$s(x)$ が二次多項式になったらその極値を算出すれば最初に求めた $s(x)$ の極値の包含が可能となる．このとき，各ステップで算出される $s(x)$ に

対し，極値は $s(x)$ が二次多項式になるまで算出しないものの，$s(\underline{X}), s(\overline{X})$ は常に算出し，$f(x)$ の値域の上限，下限の候補としていく．n が奇数の場合は $s(x)$ が二次式にはならないため，$s(x)$ の極値を算出する必要はない．

なお，$s(x)$ が三次多項式以上でも計算途中に混入される丸め誤差を考慮しながらその極値を算出することは可能ではあるが，計算量が膨大になってしまう．ゆえにこの方法では，$s(x)$ が三次多項式以上の場合には極値を算出せず，再び $s(x)$ を算出している．

以上を踏まえて本方法の具体的な手順を以下に示す．

step1 $f(\underline{X}), \quad f(\overline{X})$ を算出し，$f(x)$ の値域の上限，下限の候補とする．

step2 $f(x)$ をその導関数 $f'(x)$ で割ったときの剰余 $s(x)$ を算出する．ここで

$$s(x) = S_{n-2}x^{n-2} + \cdots + S_1 x + S_0 \tag{6.43}$$

としたとき，係数 $S_0, \cdots, S_{n-2}$ は次式により求めることができる．

$$S_i = a_i(1 - \frac{i}{n}) - \frac{(i+1)a_{i+1}a_{n-1}}{n^2 a_n}, \quad i = 0, \cdots, n-2 \tag{6.44}$$

係数 $S_0, \cdots, S_{n-2}$ は数学的には実数であるが，計算機上でこの方法を実行する際には丸め誤差を考慮しなければならないため区間として求める．すなわち係数 $S_0, \cdots, S_{n-2}$ は $a_0, \cdots, a_n$ を用いて区間演算により算出される．

次に $s(\underline{X}), s(\overline{X})$ を算出し，$f(x)$ の値域の上限，下限の候補とする．

step3 $s(x)$ が二次多項式，あるいは一次多項式になったら **step4** に進む．さもなくば $s(x)$ を $f(x)$ とおき **step2** へ戻る．

step4 $s(x)$ が二次多項式の場合，$x \in X$ における $s(x)$ の極値を算出し，$f(x)$ の値域の上限，下限の候補とする．

step5 得られた候補の中での最大値を値域の上限，最小値を値域の下限とする．

□

なお，計算機上の丸め誤差を厳密に見積もりながらこの方法を実行するために，途中で得られる候補は全て区間演算により算出されなければならない．このとき全ての候補は区間となり，値域の上限はこれらの区間の上端の中の最大値とし，値域の下限は区間の下端の中の最小値となる．この方法の計算量は $O(n^2)$ である．

6.3.5 提案手法 5

提案手法 4 と同様にこの方法でも $f(\underline{X}), f(\overline{X})$ と $x \in X$ における $f(x)$ の極値を考慮することにより多項式の値域を評価することを考える．n 次多項式 $f(x)$ とその導関数 $f'(x)$ を用いて，$d(x)$ ($n-1$ 次) を

$$d(x) = f_n(x) - \frac{x}{n}f'(x) \tag{6.45}$$

とする．ここで，$f'(x)=0$ を満たす x を x'　$(x' \in X)$ とおき，x' を式 (6.45) に代入すると，

$$\frac{x'}{n}f'(x')=0 \tag{6.46}$$

より式 (6.45) は以下のようになる．

$$d(x')=f(x') \tag{6.47}$$

これより次式が成立する．

$$f(x') \in [\min_{x\in X} d(x), \max_{x\in X} d(x)] \tag{6.48}$$

この式より $x \in X$ における $f(x)$ の極値 $f(x')$ は $x \in X$ における $d(x)$ の値域の評価で包含できることが分かる．ゆえに $x \in X$ における $f(x)$ の極値 $f(x')$ の変わりに $d(x)$ の値域の評価の上限，下限を $f(x)$ の値域の上限，下限の候補としても，$f(x)$ の $x \in X$ における値域を包含することができる．

$d(x)$ の値域の上限，下限の候補となる値は当然 $d(\underline{X}), d(\overline{X})$ 及び $x \in X$ における $d(x)$ の極値である．$x \in X$ における $d(x)$ の極値は $d(x)$ の算出を再帰的に行う (現在の $d(x)$ を $f(x)$ とおき，再び $d(x)$ を求める) ことによりその包含が可能となる．具体的には以下のとおりである．$f(x)$ の次数が n なのに対し，$d(x)$ の次数は $n-1$ であるため，$d(x)$ の算出を一度行うたびに $d(x)$ の次数は 1 づつ減少していく．ゆえに $d(x)$ の算出を再帰的に行うことにより $d(x)$ の次数を減少させてゆき，$d(x)$ が二次多項式になったらその極値を算出すれば最初に求めた $d(x)$ の極値の包含が可能となる．このとき，各ステップで算出される $d(x)$ に対し，極値は $d(x)$ が二次多項式になるまで算出しないものの，$d(\underline{X}), d(\overline{X})$ は常に算出し，$f(x)$ の値域の上限，下限の候補としていく．

なお，$d(x)$ が三次多項式以上でも計算途中に混入される丸め誤差を考慮しながらその極値を算出することは可能ではあるが，計算量が膨大になってしまう．ゆえにこの方法では，$d(x)$ が三次多項式以上の場合には極値を算出せず，再び $d(x)$ を算出している．

以上を踏まえて本方法の具体的な手順を以下に示す．

step1　$f(\underline{X}),\quad f(\overline{X})$ を算出し，$f(x)$ の値域の上限，下限の候補とする．

step2　$f(x)$ とその導関数 $f'(x)$ を用いて

$$d(x)=f(x)-\frac{x}{n}f'(x) \tag{6.49}$$

を算出する．ここで

$$d(x)=D_{n-1}x^{n-1}+\cdots+D_1x+D_0 \tag{6.50}$$

としたとき，係数 $D_0,\cdots,D_{n-1}$ は次式により求めることができる．

$$D_i=a_i(1-\frac{i}{n}),\quad i=0,\cdots,n-1 \tag{6.51}$$

係数 $D_0, \cdots, D_{n-1}$ は数学的には実数であるが，計算機上でこの方法を実行する際には丸め誤差を考慮しなければならないため区間として求める．すなわち係数 $D_0, \cdots, D_{n-1}$ は $a_0, \cdots, a_n$ を用いて区間演算により算出される．

次に $d(\underline{X}), \quad d(\overline{X})$ を算出し，$f(x)$ の値域の上限，下限の候補とする．

step3 $d(x)$ が二次多項式になったら **step4** に進む．さもなくば $d(x)$ を $f(x)$ とおき **step2** へ戻る．

step4 $x \in X$ における $d(x)$ の極値を算出し，$f(x)$ の値域の上限，下限の候補とする．

step5 得られた候補の中での最大値を値域の上限，最小値を値域の下限とする．

□

提案手法 4 と同様に，計算機上の丸め誤差を厳密に見積もりながらこの方法を実行するために，途中で得られる候補は全て区間演算により算出されなければならない．このとき全ての候補は区間となり，値域の上限はこれらの区間の上端の中の最大値とし，値域の下限は区間の下端の中の最小値となる．この方法の計算量は $O(n^2)$ である．

6.4 数値例

本節では本章で提案した提案手法の有効性をいくつかの数値例により示す．なお，筆者が数値実験を行った計算機環境は CPU:Pentium IV 3.4GHz，メモリ:4.0GB，コンパイラ: Microsoft VisualC++ Ver. 6.0 である．

6.4.1 数値例 1

本節では真の値域の分かっている簡単な問題に対してそれぞれの方法の有効性を検証する．式 (6.52) と式 (6.53) に対して，5 つのこれまでの方法及び 5 つの提案手法を適用したときの値域の評価，その評価の区間幅，実行時間を表 6.1 と表 6.2 に示す．さらにこれらの表では真の値域とその区間幅も示す．

$$0.25x^4 - 2x^3 + 5.5x^2 - 6x + 2, \quad x \in [0.5, 5] \tag{6.52}$$

$$0.6x^5 + 37.5x^4 + 935x^3 + 11625x^2 + 72072x + 38.33, \quad x \in [-15, -10] \tag{6.53}$$

6.4.2 数値例 1 に関する考察

表 6.1，表 6.2 により，これらの例に関しては，全ての提案手法が全てのこれまでの方法と比べて，シャープな値域の評価を与えていることが確認できる．特に提案手法 4 は，これらの例に関しては，真の値域に非常に近い評価を与えていることが分かる．

表 6.1: それぞれの方法を式 (6.52) に適用したときの性能比較

方法	値域の評価	区間幅	実行時間 (μsec)
真の値域	$[-0.25, 15.75]$	16	—
通常法	$[-276.6093750, 292.5000000]$	569.10938	1.890000
Horner 法	$[-124.8750000, 100.1250000]$	225.00000	0.938000
平均値形式に基づく方法	$[-199.32714844, 198.92285156]$	398.25000	3.797000
Alefeld の方法	$[-47.53125000, 47.12695312]$	94.658203	5.984000
アフィン演算を使用する方法	$[-19.911621, 24.451172]$	44.362793	80.87500
提案手法 1	$[-9.483398438, 17.22656250]$	26.709961	3.922000
提案手法 2	$[-8.006835938, 15.75000000]$	23.756836	3.281000
提案手法 3	$[-2.047851563, 17.22656250]$	19.274415	8.000000
提案手法 4	$[-2.250000000, 15.75000000]$	18.000000	3.750000
提案手法 5	$[-14.25000000, 15.75000000]$	30.000000	4.094000

表 6.2: それぞれの方法を式 (6.53) に適用したときの性能比較

方法	値域の評価	区間幅	実行時間 (μsec)
真の値域	$[-178229, -178182]$	47	—
通常法	$[-3154791.670, 2798380.830]$	5953172.500	2.375000
Horner 法	$[-799791.6700, 493958.3300]$	1293750.000	0.985000
平均値形式に基づく方法	$[-493025.42, 136614.58]$	629640.0000	4.468000
Alefeld の方法	$[-240994.1700, -115416.6700]$	125577.5000	8.563000
アフィン演算を使用する方法	$[-216091.82625, -140319.01375]$	75772.81250	121.8280
提案手法 1	$[-178307.2950, -178103.5450]$	203.7500000	5.766000
提案手法 2	$[-178240.2638, -178170.5762]$	69.68750000	4.656000
提案手法 3	$[-178307.2950, -178103.5450]$	203.7500000	11.00000
提案手法 4	$[-178229.1700, -178181.6610]$	47.50900000	5.078000
提案手法 5	$[-179484.6700, -170831.1700]$	8653.500000	6.359000

さらに表 6.1，表 6.2 により提案手法 1 と提案手法 2 は Alefeld の方法よりも高速であり，全ての提案手法がアフィン演算を使用する方法よりも高速であることが確認できる．また全ての提案手法は通常法，Horner 法，平均値形式に基づく方法に比べ大きな計算量を必要とするものの，比較に足るものであることも確認できる．

また表 6.1，6.2 により提案手法 2 は提案手法 1 よりもシャープな評価を与え，かつ高速であることも確認できた．これは提案手法 2 が提案手法 1 の持つ効果をほぼ失うことなく Horner 法を実行しているという事実と一致する結果といえる．提案手法 5 に比べ提案手法 4 の方が高速であることも確認できる．これは多項式の次数が n であるとき提案手法 4 の $s(x)$ が $n-2$ 次であるのに対し，提案手法 5 の $d(x)$ が $n-1$ 次であることから，これらの方法における反復処理 ($s(x), d(x)$ を再帰的に求める処理) の回数が，提案手法 4 の方が少ないためであると考えられる．

6.4.3 数値例 2

$x \in [c-r, c+r]$ における多項式

$$\frac{a_n}{n!}x^n + \frac{a_{n-1}}{(n-1)!}x^{n-1} + \cdots + \frac{a_2}{2!}x^2 + \frac{a_1}{1!}x + a_0 \tag{6.54}$$

の値域の評価をそれぞれの方法で行うことを考える．この問題に対して，通常法，Horner 法，平均値形式に基づく方法，Alefeld の方法，アフィン演算を使用する方法，提案手法 1，提案手法 2，提案手法 3，提案手法 4，提案手法 5 を適用したときに得られる評価の区間幅を W_S，W_H，W_M，W_A，W_a，W_1，W_2，W_3，W_4，W_5 とし，実行時間を T_S，T_H，T_M，T_A，T_a，T_1，T_2，T_3，T_4，T_5 とする．このとき，

$$R_H = \frac{W_H}{W_S}, R_M = \frac{W_M}{W_S}, R_A = \frac{W_A}{W_S}, R_a = \frac{W_a}{W_S}, R_1 = \frac{W_1}{W_S}, R_2 = \frac{W_2}{W_S}, R_3 = \frac{W_3}{W_S}, R_4 = \frac{W_4}{W_S},$$

$$R_5 = \frac{W_5}{W_S}$$

とし，

$$S_H = \frac{T_H}{T_S}, S_M = \frac{T_M}{T_S}, S_A = \frac{T_A}{T_S}, S_a = \frac{T_a}{TS}, S_1 = \frac{T_1}{T_S}, S_2 = \frac{T_2}{T_S}, S_3 = \frac{T_3}{T_S}, S_4 = \frac{T_4}{T_S}, S_5 = \frac{T_5}{T_S}$$

とする．

最初に $n = 20$ とし a_i $(i = 0, \cdots, 20)$ を $-3 \leq a_i \leq 3$ を満たす乱数とし，定義域の中心 c，半径 r を $0 \leq c \leq 20,\ 5 \leq r \leq 15$ を満たす乱数として R_H，R_M，R_A，R_a，R_1，R_2，R_3，R_4，R_5，S_H，S_M，S_A，S_a，S_1，S_2，S_3，S_4，S_5 を 10000 回計算したときの平均値を表 6.3 に示す．

次に $n = 20$ とし係数 a_i $(i = 0, \cdots, 20)$ を $-3 \leq a_i \leq 3$ を満たす乱数とし，定義域の中心 c，半径 r を $0 \leq c \leq 20,\ 1 \leq r \leq 3$ を満たす乱数として R_H，R_M，R_A，R_a，R_1，R_2，R_3，R_4，R_5，S_H，S_M，S_A，S_a，S_1，S_2，S_3，S_4，S_5 を 10000 回計算したときの平均値を表 6.4 に示す．

区間幅の比	区間幅の比の平均値	実行時間の比	実行時間の比の平均値
R_H	0.498938	S_H	0.496000
R_M	6.209390	S_M	1.280000
R_A	0.528144	S_A	8.720000
R_a	0.220861	S_a	157.7980
R_1	0.303932	S_1	5.040000
R_2	0.224772	S_2	2.480000
R_3	0.204929	S_3	6.240000
R_4	0.128108	S_4	4.960000
R_5	0.127160	S_5	5.040000

表 6.3: $n = 20,\ -3 \leq a_i \leq 3\ (i = 0, \cdots, 20),\ 0 \leq c \leq 20,\ 5 \leq r \leq 15$ のときの全ての方法の性能比較

区間幅の比	区間幅の比の平均値	実行時間の比	実行時間の比の平均値
R_H	0.570368	S_H	0.368000
R_M	1.894690	S_M	1.280000
R_A	0.648755	S_A	12.80000
R_a	0.357009	S_a	143.1190
R_1	0.491813	S_1	3.760000
R_2	0.420906	S_2	3.680000
R_3	0.378825	S_3	6.320000
R_4	0.382115	S_4	4.960000
R_5	0.271090	S_5	5.040000

表 6.4: $n = 20,\ -3 \leq a_i \leq 3\ (i = 0, \cdots, 20),\ 0 \leq c \leq 20,\ 1 \leq r \leq 3$ のときの全ての方法の性能比較

最後に $n = 10$ とし係数 a_i $(i = 0, \cdots, 10)$ を $-3 \leq a_i \leq 3$ を満たす乱数とし，定義域の中心 c，半径 r を $0 \leq c \leq 20,\ 5 \leq r \leq 15$ を満たす乱数として R_H，R_M，R_A，R_a，R_1，R_2，R_3，R_4，R_5，S_H，S_M，S_A，S_a，S_1，S_2，S_3，S_4，S_5 を 10000 回計算したときの平均値を表 6.5 に示す.

区間幅の比	区間幅の比の平均値	実行時間の比	実行時間の比の平均値
R_H	0.890332	S_H	0.500000
R_M	6.192280	S_M	1.516130
R_A	0.888572	S_A	5.000000
R_a	0.377766	S_a	67.39130
R_1	0.619190	S_1	2.532260
R_2	0.525418	S_2	2.258060
R_3	0.413061	S_3	2.580650
R_4	0.332993	S_4	2.774190
R_5	0.320235	S_5	3.016130

表 6.5: $n = 10,\ -3 \leq a_i \leq 3\ (i = 0, \cdots, 20),\ 0 \leq c \leq 20,\ 5 \leq r \leq 15$ のときの全ての方法の性能比較

6.4.4 数値例 2 に関する考察

表 6.3 より，一般的な傾向として，全ての提案手法はアフィン演算を使用する方法以外のこれまでの方法よりもシャープな評価を与えていることが確認できる．全ての提案手法は Horner 法，平均値形式に基づく方法に比べより大きな計算量を必要とするものの，それを補って余りあるだけのシャープな評価を与えている．また提案手法同士の間では，提案手法 4，提案手法 5 が他の提案手法に比べてシャープな評価を与えており，提案手法 2 が他の提案手法に比べて高速であることが確認できる．

表 6.4 における平均値形式に基づく方法以外の全ての方法の区間幅の比は表 6.3 におけるそれよりも大きいことが確認できる．これより平均値形式に基づく方法以外の全ての方法は入力区間が広い場合の方が通常法よりもよりシャープな評価を与えることが確認できる．一方平均値形式に基づく方法は入力区間が狭い場合の方が通常法に対する評価のシャープさの比が向上することが確認できる．なお，表 6.4 の計算量に関する傾向は表 6.3 のそれとほぼ同様であることも確認できる．

表 6.5 における平均値形式に基づく方法以外の全ての方法の区間幅の比は表 6.3 におけるそれよりも大きいことが確認できる．これより平均値形式に基づく方法以外の全ての方法は多項式の次数が大きい場合の方が通常法よりもよりシャープな評価を与えることが確認できる．一方平均値形式に基づく方法は多項式の次数が変化しても通常法に対する評価のシャープさは変化しないことも確認できる．また表 6.5 における Horner 法と平均値形式に

基づく方法以外の方法の実行時間の比は表 6.3 におけるそれよりも小さいことから Horner 法と平均値形式に基づく方法以外の方法は次数が小さい方が通常法に対する計算量の比が小さくなることが確認できる．これは通常法，Horner 法，平均値形式に基づく方法の計算量が $O(n)$ なのに対しその他の方法の計算量が $O(n^2)$ であるという事実と一致する結果といえる．

6.5 むすび

本章では区間演算で評価する関数を多項式に限定した場合の評価方法について考察した．まずこれまでの方法として Horner 法，平均値形式に基づく方法，Alefeld の方法，アフィン演算を使用する方法について説明した．次に 5 つの方法を提案した (提案手法 1〜提案手法 5)．本章の最後ではいくつかの数値例を行うことにより，多項式関数の値域を評価する全ての方法を比較 · 検証し提案手法の有効性を示した．

次章では結論を述べ，本論文を総括する．

第 7 章

結論

本論文は区間演算において区間幅の爆発的な増大を抑制し関数の値域をシャープに包含する方法について考察を与えた．

第 1 章では本論文の背景，目的について述べた．

第 2 章では本論文における議論の基礎となり，精度保証付き数値計算の根幹をなす重要な技術の一つである区間演算についての解説とその問題点である区間幅の爆発的な増大について説明した．またこれまでに提案されてきた区間幅の爆発的な増大を抑制する方法について述べた．

第 3 章では Jorge Stolfi，Luiz H. de Figueiredo らによるアフィン演算の定義とその有効性，また，非線形二項演算における問題点について述べた．

第 4 章ではアフィン演算における乗算について述べた．まずこれまでの方法として，Jorge Stolfi，Luiz H．de Figueiredo らにより定義された Stolfi の乗法，宮田孝富，柏木雅英により提案された改良乗算の説明を行った．次に最適な評価を与える方法 (最良乗算) を提案し，さらにこの最良乗算を実用時間で実行する方法を提案した．第 4 章の最後においてはいくつかの数値例により最良乗算の有効性を確認した．第 4 章で提案した最良乗算の応用分野への適用が今後の課題となる．

第 5 章ではアフィン演算における除算について述べた．まずこれまでの方法として，従来の方法，柏木雅英により提案された柏木の方法，白井健一により提案された白井の方法について説明した．次に 3 つの方法を提案した (提案手法 1〜提案手法 3)．第 5 章の最後においてはいくつかの数値例によりアフィン演算における全ての除算の方法を比較 · 検証し，提案手法の有効性を示した．第 5 章で提案した除算の応用分野への適用及び δ を理論的に最小に抑える除算の構築が今後の課題となる．

第 6 章では区間演算で評価する関数を多項式に限定した場合の評価方法について考察した．まずこれまでの方法として Horner 法，平均値形式に基づく方法，Alefeld の方法，アフィン演算を使用する方法について説明した．次に 5 つの方法を提案した (提案手法 1〜提案手法 5)．第 6 章の最後ではいくつかの数値例を行うことにより，多項式関数の値域を評価する全ての方法を比較 · 検証し提案手法の有効性を示した．第 6 章で提案した評価方法の応用分野への適用及び一般の非線形関数，2 次元以上の問題に対する拡張性への考察が今後の課題となる．

第 8 章

謝辞

本研究は，著者の学部在学中，修士課程在学中及び博士後期課程在学中に行われたものであります．本研究を始めるきっかけを与えて下さり，これを進めるにあたって終始懇切丁寧な御議論，御指導，御検討を賜り，著者の研究生活において，また研究を離れても常にあたたかい御激励をいただきました指導教授・柏木雅英助教授に深く感謝の意を表します．

また，学部在学中から著者の研究生活をあたたかく見守って下さり，著者の研究においては常に有益な御指導，御助言を賜り，かつ著者の研究生活を物心両面から強力にサポートして下さった大石進一教授に深く感謝いたします．

また，本論文をまとめるに際し適切な御示唆をいただきました山本哲朗教授に深く感謝いたします．

また，著者が常日頃から御指導いただいている先輩として中央大学山村清隆教授，中央大学牧野光則助教授，東京都立短期大学相馬隆郎講師，京都大学辻本諭講師，中谷祐介講師，丸山晃佐講師，荻田武史講師に深く感謝いたします．

また，同じく著者の研究を御指導いただき，研究生活を離れても親身になって御助力いただいている先輩として筑波大学遠藤靖典助教授，芝浦工業大学神沢雄智助教授に深く感謝いたします．

また，本研究を進めるにあたり学会等で有益な御指導，御助言を賜りました電気通信大学山本野人助教授に深く感謝いたします．

また，筆者の学部在学中から3，4，5章のアフィン演算について熱心に御指導してくださり，本研究を進めるに際し，有意義な御議論をさせていただきました金沢工業大学宮田孝富講師に深く感謝いたします．

また，良き同僚として常に多くの御助言，御協力いただきました柏木研究室ならびに大石研究室の先輩，同輩，後輩諸氏に感謝いたします．

最後に，筆者を育て，筆者が生まれてから現在に至るまで様々な面で支えてくださった父・宮島清司氏，母・宮島利江氏に心より感謝申し上げます．

参考文献

[1] N. J. Higham, "Accuracy and Stability of Numerical Algorithms, 2nd ed., " SIAM Publications, Philadelphia, 2002.

[2] Douglas N. Arnold:"The Patriot Missile Failure", URL:http://www.ima.umn.edu/ arnold/disasters/patriot.html

[3] Douglas N. Arnold:"The sinking of the Sleipner A offshore platform", URL:http://www.ima.umn.edu/ arnold/disasters/sleipner.html

[4] Douglas N. Arnold:"The Explosion of the Ariane 5", URL:http://www.ima.umn.edu/ arnold/disasters/ariane.html

[5] U. Kulisch, W. L. Miranker: "Computer Arithmetic in Theory and Practice", Academic Press, New York, 1981.

[6] 山本哲朗: "エッセイ マジソン便り", Computrol No. 12 (コロナ社), pp. 5-8, 1985.

[7] L. B. Rall: "Validation of Numerical Computation", 京都大学数理解析研究所講究録 673, pp. 1-16, 1988.

[8] 中尾充宏: "関数方程式の解の存在に対する数値的検証法", 数学 42, pp. 16-31, 1990.

[9] 中尾充宏: "精度保証付き数値計算の現状と動向", 情報数理 Vol. 31, No. 9, pp. 1177-1190, 1990.

[10] 山本哲朗, 陳小君: "有限次元非線形方程式に対する保証付き数値計算", 情報数理 Vol. 31, No. 9, pp. 1191-1196, 1990.

[11] 山口昌哉, 吉原英昭, 西田孝明: "微分方程式に対する保証付き数値計算", 情報数理 Vol. 31, No. 9, pp. 1197-1203, 1990.

[12] 野田松太郎, 佐々木建昭, 鈴木正幸: "数式処理と数値計算の融合による精度保証", 情報数理 Vol. 31, No. 9, pp. 1204-1211, 1990.

[13] 久保田光一, 伊理正夫: "区間演算を用いた丸め誤差解析", 情報数理 Vol. 31, No. 9, pp. 1212-1219, 1990.

[14] 奥田晃, 棚町芳弘: "大型計算機向き汎用精度保証付き数値計算ソフトウェア", 情報数理 Vol. 31, No. 9, pp. 1220-1227, 1990.

[15] 野寺隆: "パソコンの精度保証付き数値計算ソフトウェア", 情報数理 Vol. 31, No. 9, pp. 1228-1234, 1990.

[16] S. Oishi: "The Self-Validating Numerical Method – A New Tool for Computer Assisted Proofs of Nonlinear Problems –", IEICE Trans. Fundamentals Vol. E75-A, No. 5, pp. 595-612, 1992.

[17] 大石進一:“非線形解析入門”，コロナ社，1997.

[18] 大石進一:“数値計算”，(応用解析セミナー)，裳華房，1999.

[19] 大石進一:“精度保証付き数値計算”，コロナ社，2000.

[20] T. Sunaga: “Theory of Interval Algebra and its Application to Numerical Analysis”, RAAG MEMOIRS Vol. 2, Misc. II, p. 29-, 1958.

[21] O. Aberth: “Computable Analysis”, McGraw-Hill, 1980.

[22] S. Markov, K. Okumura: “The Contribution of T. Sunaga to Interval Analysis and Reliable Computing”, Developments in Reliable Computing (edited by Tibor Csendes, Jozsef Attila University, Szeged, Hungary), ISBN 0-7923-6057-5, 1999.

[23] P. S. Dwyer: “Error of Matrix Computations”, Mathematical Methods for Digital Computation I, II (eds. A. Ralston and H. S. Wilf), John Wiley, I=1960, II=1967.

[24] R. E. Moore: “Interval Analysis”, Prentice-Hall, Englewood Cliffs, New Jersey, 1966.

[25] R. E. Moore: “Methods and Applications of Interval Analysis”, SIAM, Philadelphia, 1979.

[26] Thomas C. Hales:“Proving Theorems in Geometry by Computer”, 11th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics (SCAN 2004), Fukuoka, Japan, October, 2004.

[27] DeRose, Salesin: “Computer Graphics and Computer-Aided Design”, URL:http://www.cs.washington.edu/research/research-samples/graphics.html

[28] Luc Jaulin, Michel Kieffer, Olivier Didrit, Eric Walter:“Applied Interval Analysis”, Springer-Verlag, London, 2001.

[29] “Interval Methods Help a Robot Succeed”, URL:http://www.cs.utep.edu/interval-comp/robot.html

[30] “An Application of Interval Computations to Gravity Featured as One of The Major Scientific Results”, URL:http://www.cs.utep.edu/interval-comp/grav.html

[31] “Double Bubble Minimizes: Interval Computations Help in Solving a Long-Standing Geometric Problem”, URL:http://www.cs.utep.edu/interval-comp/bubble.html

[32] R. E. Moore: “A test for existence of solutions to nonlinear systems”, SIAM J. Numer. Anal., vol. 14, no. 4, pp. 611-615, 1977.

[33] R. E. Moore and S. T. Jones: “Safe starting regions for iterative methods”, SIAM J. Numer. Anal., vol. 14, no. 6, pp. 1051-1065, 1977.

[34] 藤井 康雄, 市田 浩三, 清野 武:“区間演算を利用した最大値探索法”, 情報処理, Vol. 18, No. 11, pp. 1095-1101, 1977.

[35] 市田構造” 区間解析による最適化”, 経済経営論叢 18 巻 3 号 pp. 1-21, 1983.

[36] 市田構造” 区間ニュートン法による最大値探索法”, 経済経営論叢 20 巻 2-3 号 pp. 379-398, 1985.

[37] 柏木雅英, 大石進一: “ベキ級数演算を用いた常微分方程式の精度保証付き計算”, 電子情報通信学会技術研究報告 (TECHNICAL REPORT OF IEICE.), NLP94-91, pp. 65-70, 1995.

[38] Marcus Vinícius A. Andrade, João L. D. Comba and Jorge Stolfi: “Affine Arithmetic”, INTERVAL ’94, St. petersburg, Russia, March, 1994.
URL:http://www.dcc.unicamp.br/˜ stolfi/EXPORT/bibliography/stolfi.html♯ and-com-sto-94-aax

[39] Luiz H. de Figueiredo and Jorge Stolfi: “Self-Validated Numerical Methods and Applications”, Brazilian Mathematics Colloquium monograph, IMPA, Rio de Janeiro, Brazil, July, 1997.
URL: http://www.dcc.unicamp.br/˜ stolfi/EXPORT/bibliography/stolfi.html

[40] Tomoyuki Kikuchi, Masahide Kashiwagi: “Elimination of Non-existence Regions of the Solution of Nonlinear Equations using Affine Arithmetic”, International Symposium on Nonlinear Theory and its Apprications (NOLTA2001), Miyagi, Japan, October, 2001.

[41] Shinya Miyajima, Masahide Kashiwagi: “Existence Examination of Solution in Systems of Nonlinear Equations Applying Affine Arithmetic”, Fourth International Conference on Intelligent Technologies (Intech’03), Chiang Mai, Thailand, December, 2003.

[42] 加藤仁徳:“全解探索における Affine Arithmetic を用いた解の存在領域の除去”, 早稲田大学 2002 年度卒業論文.

[43] 宮田孝富, 柏木雅英:“アフィン演算の乗算について”, 第 10 回 FAN インテリジェント · システム · シンポジウム 講演論文集, 講演受付番号 55, pp. 55-60, 2002.

[44] 柏木雅英, “Affine Arithmetic とその応用”, 日本応用数理学会年会, 1999.

[45] 長嶋 雄一郎 :“Affine Arithmetic の除算の実現”, 早稲田大学 1998 年度卒業論文.

[46] 白井健一，宮田孝富，柏木雅英:“Affine Arithmetic における除算について”，2000 年電子情報通信学会基礎・境界ソサエティ大会講演論文集，pp. 45，2000.

[47] 白井 健一 :“Affine Arithmetic における除算について”，早稲田大学 1999 年度卒業論文.

[48] A. Neumaier:“Interval methods for systems of equations”，Cambrige University Press，1990.

[49] G. Alefeld，J. Herzberger: “Introduction to Interval Computations”，Academic Press，New York，1983.

[50] G. Alefeld: “Bounding the Slope of Polynomial Operators and Some Applications”，Computing 26，pp. 227-237，1981.

[51] Takatomi Miyata，Masahide Kashiwagi:“ On Range Evaluation of Polynomials of Affine Arithmetic ”，International Symposium on Nonlinear Theory and its Apprications (NOLTA2001)，Miyagi，Japan，October，2001.

研究業績

学術論文

- ○宮島信也，宮田孝富，柏木雅英：“アフィン演算における最良乗算について”，電子情報通信学会論文誌，Vol．J86-A，No．2，pp．150-159，February，2003

- ○宮島信也，宮田孝富，白井健一，柏木雅英：“アフィン演算における乗除算について”，電子情報通信学会論文誌，Vol．J86-A，No．3，pp．232-240，March，2003

- ○Shinya Miyajima，Takatomi Miyata，Masahide Kashiwagi：“A New Dividing Method in Affine Arithmetic”，IEICE Trans．Vol．E86-A，No．9，pp．2192-2196，September，2003

- ○Shinya Miyajima，Masahide Kashiwagi：“On Range Evaluation of Polynomials by Applying Interval Arithmetic”，Lecture Notes in Computer Science 2907，Springer-Verlag，pp．254-261，2004

- ○Shinya Miyajima，Masahide Kashiwagi：“On Range Inclusion of Polynomials Applying Interval Arithmetic ”，IEICE Trans．Vol．E87-A，No．3，pp．725-731，March，2004

- ○Shinya Miyajima，Masahide Kashiwagi：“A Dividing Method Utilizing the Best Multiplication in Affine Arithmetic”，IEICE Electronics Express，Vol．1 No．7 pp.176-181，2004

- ○Shinya Miyajima，Masahide Kashiwagi：“A Method Which Finds the Maxima and Minima of a Multivariable Function Applying Affine Arithmetic”，Lecture Notes in Computer Science 3401，Springer-Verlag，pp．424-431，2005

査読付き国際会議論文

- Shinya Miyajima，Masahide Kashiwagi：“Construction of New Dividing Method in Affine Arithmetic”，Proc．2002 International Symposium on Computational Mathematics and Applications (ISCMA2002)，Dalian University of Technology，Dalian，China，p．32，August，2002

- Takatomi Miyata, Shinya Miyajima, Masahide Kashiwagi: "On the best multiplication of Affine Arithmetic", Proc. 2002 International Symposium on Computational Mathematics and Applications (ISCMA2002), Dalian University of Technology , Dalian , China, p. 33, August, 2002

- Shinya Miyajima, Masahide Kashiwagi : "On New Dividing Method in Affine Arithmetic", Proc. 2002 International Symposium on Nonlinear Theory and its Applications (NOLTA2002), Xi'an International Conference Center , PRC , Xi'an , China, pp. 479-482, October, 2002

- Shinya Miyajima, Masahide Kashiwagi : "On Range Evaluation of Polynomials by Applying Interval Arithmetic", Proc. 4th International Conference on "Large-Scale Scientific Computations" (LSSC '03), Bulgarian Red Cross Educational Center, Sozopol, Bulgaria, pp. B-33 - B-34, June, 2003

- Shinya Miyajima, Masahide Kashiwagi : "On the Application of the Best Multiplication to Division in Affine Arithmetic", Proc. Second St.Petersburg Days of LOGIC and COMPUTABILITY, The Euler International Mathematical Institute, St.Petersburg, Russia, p. 38, August, 2003

- Shinya Miyajima, Masahide Kashiwagi : "Existence Examination of Solution in Systems of Nonlinear Equations Applying Affine Arithmetic", Proc. Fourth International Conference on Intelligent Technologies (Intech'03), Chiang Mai Plaza Hotel, Chiang Mai, Thailand, pp. 552-561, December, 2003

- Shinya Miyajima, Masahide Kashiwagi : "A Method Which Finds Maxima and Minima of a Multivariable Function Applying Affine Arithmetic", Proc. Third International Conference on "Numerical Analysis and Applications" (3rd NA&A), Division of Numerical Analysis and Statistics, University of Rousse "Angel Kanchev", Rousse, Bulgaria, p. 31, June, 2004

- Shinya Miyajima, Masahide Kashiwagi : "A Method Which Finds Maxima and Minima of a Multivariable Function Applying Mean Value Form", Proc. 11th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics (SCAN 2004), Nishitetsu Grand Hotel, Fukuoka, Japan, p. 83, October, 2004

- Shinya Miyajima, Takeshi Ogita, Katsuhisa Ozaki, Shin'ichi Oishi: "Fast Error Estimation for Eigenvalues of Symmetric Matrix without Directed Rounding", Proc. 2004 International Symposium on Nonlinear Theory and its Applications (NOLTA 2004), ACROS Fukuoka, Fukuoka, Japan, pp. 167-170, December, 2004

学術講演

- 宮島信也, 宮田孝富, 柏木雅英: "Affine Arithmetic における除算の改良について", 2001 年電子情報通信学会ソサエティ大会講演論文集, 電気通信大学 (東京都調布市), A-2-3, p. 36, 2001 年 9 月

- 宮島信也, 柏木雅英: "Affine Arithmetic における除算の新手法について", 2002 年電子情報通信学会ソサエティ大会講演論文集, 宮崎大学 (宮崎県宮崎市), A-2-23, p. 54, 2002 年 9 月

- 宮島信也, 宮田孝富, 柏木雅英: "Affine Arithmetic における最良乗算の実現", 2002 年情報科学技術フォーラム (FIT2002) 一般講演論文集, 東京工業大学 (東京都太田区), 第 1 分冊, A-21, pp. 41-42, 2002 年 9 月

- 宮島信也, 柏木雅英: "多項式の値域の区間評価について", 2003 年 2 月電子情報通信学会非線形問題研究会 (NLP) 技術研究報告, 北海道大学 (北海道札幌市), Vol. 102, No. 625, NLP2002-95, pp. 13-18, 2003 年 2 月

- 宮島信也, 柏木雅英: "区間演算による多項式の値域の評価について", 2003 年 3 月電子情報通信学会総合大会講演論文集, 東北大学 (宮城県仙台市), A-2-3, p. 37, 2003 年 3 月

- 宮島信也, 柏木雅英: "アフィン演算における最良乗算の除算への応用", 2003 年 6 月電子情報通信学会非線形問題研究会 (NLP) 技術研究報告, 自治会館 (沖縄県那覇市), Vol. 103, No. 137, NLP2003-23, pp. 13-18, 2003 年 6 月

- 宮島信也, 柏木雅英: "アフィン演算を用いた非線形方程式の解の存在検証について", 日本応用数理学会 2003 年度年会講演予稿集, 京都大学 (京都府京都市), pp. 106-107, 2003 年 9 月

- 宮島信也，柏木雅英：“アフィン演算をにおける最良乗算を使用した除算について”，2003 年電子情報通信学会ソサエティ大会講演論文集，新潟大学 (新潟県新潟市)，A-2-1，p．33，2003 年 9 月

- 宮島信也，柏木雅英：“アフィン演算とその応用に関する研究”，2003 年度・計算数学研究会，コーワパークホテル由布院倶楽部 (大分県大分郡湯布院町)，2003 年 10 月

- 宮島信也，柏木雅英：“アフィン演算を利用した非線形方程式の解の存在検証”，2004 年電子情報通信学会総合大会講演論文集，東京工業大学 (東京都太田区)，A-2-23，p．66，2004 年 3 月

- 宮島信也，柏木雅英：“アフィン演算を用いた非線形方程式の全解探索法”，第 33 回数値解析シンポジウム (NAS2004) 講演予稿集，ウェルハートピア熱海 (静岡県熱海市)，pp．112-115，2004 年 5 月

- 尾崎 克久，荻田 武史，宮島信也，大石 進一: “Java による連立一次方程式のための精度保証法”，日本応用数理学会 2004 年度年会講演予稿集，中央大学 (東京都文京区)，pp．324-325，2004 年 9 月

- 宮島信也，柏木雅英：“アフィン演算による多変数関数の最大値探索法”，2004 年電子情報通信学会ソサエティ大会講演論文集，徳島大学 (徳島県徳島市)，A-2-13，p．46，2004 年 9 月