計算機援用証明 II

Computer-Assisted Proofs II

published in Bulletin of the Japan Society for Industrial and Applied Mathematics (Bull. JSIAM), 14(4):44–57, 2004

Siegfried M. Rump
Hamburg-Harburg 工科大学

連絡先: 〒 169-8555 東京都新宿区大久保 3–4–1
早稲田大学大学院 理工学研究科
荻田 武史 (訳者)

Abstract

Following “Computer-Assisted Proofs I”, we will continue to discuss the possibility to ‘prove’ mathematical theorems with the aid of digital computers and present different methods to approach this goal.

Keyword computer-assisted proof, computer algebra, floating point arithmetic, self-validating methods.

キーワード 計算機援用証明，計算機代数学，浮動小数点演算，自己検証的算法.

訳者: 荻田 武史 (早稲田大学大学院 理工学研究科)

3　計算機代数学か自己検証的算法か

計算機代数学的方法の主目的の 1 つは，適切なだけでなく厳密な答えを与えることにある．例として，計算を代数的数体上で行うことはよくあることである [11]．たとえば $\sqrt[3]{17} - \sqrt[5]{5}$ は数百桁の精度で何らかの数値計算による近似によって計算するのではなく，有理数体の適切な代数的拡張の元として計算する．そこにおける演算は「厳密」である．これは，いわゆる決定アルゴリズムの開発を可能にする．決定アルゴリズムとは，イエス・ノーを判定すべき問題に対して，入力データの長さに応じて決まる有限の時間内で確実に答を出すアルゴリズムのことである．

『計算機援用証明 I』において，すでにそのような決定アルゴリズムの例 (Risch による有限の項での積分) について言及した．他に計算機代数アルゴリズムの威力を示す有名でよく知られた例は，Quantifier Elimination [6, 36] や Gröbner 基底である．与えられた多変数連立代数方程式に対して，これらが解けるかどうか，そして根の個数と各根の存在する区間を判定することができる．ここで重要なことは，構成的で高速なアルゴリズム [3, 4] の開発であった．

計算機代数学的方法では，一般にすべての計算を厳密に行う．ただし，誤差範囲を求めれば十分な場合には，それで済ませることもある．その場合には，誤差範囲が広すぎたりまったく求まらない場合に備えて，厳密な計算を用いる別の方法を用意しておくのが一般的である．

厳密な計算にはそれなりの計算時間が必要になる．一方，自己検証的算法 (self-validating method) は計算を浮動小数点演算で行うため高速である．この高速さには「答えが出ない」というリスクは伴うが,「誤った答えが出る」という意味で失敗することは決してない．

これは，自己検証的算法を問題が適切 (well-posed) な場合にしか適用できない理由でもある．たとえば，行列が正則であることを検証することはできるが，それが特異なことを検証することは一般にできない．入力データの任意に小さい変化に対して問題に対する答えが変わるなら問題は不適切 (ill-posed) であり，最もわずかな丸め誤差で答えが変わるおそれがある．しかし，自己検証的算法は「意図して浮動小数点演算を使う」ので，そのような問題は適用の対象外である．

同様に，自己検証的算法により多重固有値の包含 [29]，あるいは多項式の多重根の包含 [32] を計算することができる．しかし，固有値または根が「本当に重複しているかどうか」，すなわち重複度が 1 より大きいかどうかを確かめることはできない．多重根の誤差範囲を定めるのは適切問題であるのに対して，重複度が 1 より大きいかどうかを判定するのは不適切問題なのである．このように自己検証的算法の高速さは，ひとつには浮動小数点演算の (丸めによる) 不正確さと引き換えに実現されていると言ってもよいだろう．

自己検証的算法のひとつの例として，与えられた実行列 A の正則性の検証については『計算機援用証明 I』ですでに述べた．これは，R を前処理行列とす

ると，任意の要素がすべて非負である実ベクトル x に対して

$$|I - RA|x < x \tag{4}$$

は R と A が正則であることを意味するというものである．このように「電子計算機を援用して数学の定理の仮定を検証すること」が，自己検証的算法の主な使用方法である．

これは，行列の正則性という単純な例ではあるが，主張の正当性を意味する．明らかに，式 (4) を検証するために必要なことは，算術演算についての真の誤差限界が計算可能なことだけである．そして，有向丸めの使用はひとつの可能な手段であり，計算機代数学的方法による正確な演算の使用はまた別の選択肢であろう．他に広く使われている手段は区間演算である．区間演算は簡潔な定式化と実装という利点を提供する．しかしながら，区間演算は最良の方法であるというわけではない．次章でその理由について述べる．

4 区間演算

数の集合を表現するとき，たとえば閉区間を利用できる．それは，1958 年に発表された Sunaga の論文 [34] においてすでに記述されている (非常に包括的な論文であるにもかかわらず，それほど良く認知されていない)．当面は，計算機上での丸め誤差と表現を無視し，通常どおり実数区間の集合 $\mathbb{IIR}$ を

$$A \in \mathbb{IIR} \ :\Leftrightarrow \ A \neq \emptyset \quad \text{及び} \quad A = [\underline{a}, \overline{a}] = \{x \in \mathbb{R} : \underline{a} \leq x \leq \overline{a}\}$$

によって定めよう．縮退した $\underline{a} = \overline{a}$ という場合では，区間 $[\underline{a}, \overline{a}]$ はひとつの実数 $a = \underline{a} = \overline{a}$ を表し，a と同一視できる．これはいわゆる点区間 $[a, a]$ を用いた $\mathbb{R}$ の $\mathbb{IIR}$ への自然な埋め込みである．区間演算 $\circ \in \{+, -, \cdot, /\}$ は次のように，区間オペランドの要素間におけるすべての可能な演算の結果を含む最小の区間となるように定義できる: $A, B \in \mathbb{IIR}$ に対して

$$A \circ B := \bigcap \{C \in \mathbb{IIR} : \text{ すべての } a \in A, b \in B \text{ に対して } a \circ b \in C\} \tag{5}$$

ただし，除算では $0 \notin B$ とする．$A = [\underline{a}, \overline{a}]$ 及び $B = [\underline{b}, \overline{b}]$ に対し

$$\begin{array}{rcl} A + B & = & [\underline{a} + \underline{b}, \overline{a} + \overline{b}] \\ A - B & = & [\underline{a} - \overline{b}, \overline{a} - \underline{b}] \end{array} \tag{6}$$

は明らかである．区間の特徴はその直径 $d(A) = d([\underline{a}, \overline{a}]) = \overline{a} - \underline{a}$ によって評価することができる．上式より

$$d(A + B) = d(A - B) = d(A) + d(B) \tag{7}$$

となる．つまり，2 つの区間の和及び「差」の直径は，2 つの区間の直径の和に等しい．後者は過大評価の典型的な例である．

同様に，乗算と除算の取りうる値の範囲は，オペランドが完全に正であるか，負であるか，あるいは零を含むかに依存するいくつかに場合分けによって決定することができる．区間ベクトルは

$$\begin{pmatrix} [\underline{x}_1, \overline{x}_1] \\ \vdots \\ [\underline{x}_n, \overline{x}_n] \end{pmatrix} \in (\mathbb{IIR})^n \quad \text{あるいは} \quad [\underline{x}, \overline{x}] \in \mathbb{IIR}^n$$

によって同等に定義できる．ここで，$[\underline{x}, \overline{x}] \in \mathbb{IIR}^n$ ではベクトルを要素ごとの比較で並べると約束する．また，行列についても同じように考えると，通常の定義でのあらゆる演算を対応する区間演算に置き換えるだけで，ベクトルやスカラの間の演算も定義できる．たとえば，区間の量 $\mathbf{X}, \mathbf{Y} \in \mathbb{IIR}^n$ 及び $\mathbf{A} \in \mathbb{IIR}^{n \times n}$ に対し，$\mathbf{Y} = \mathbf{AX}$ は

$$\mathbf{Y} := \mathbf{AX}, \quad \mathbf{Y}_i := \sum_{i=1}^{n} \mathbf{A}_{i\nu} \mathbf{X}_\nu \quad (1 \le i \le n) \tag{8}$$

によって定義される．ここで，加算と乗算は対応する (スカラの) 区間演算である．この定義は式 (5) に対応している．明らかに

$$\text{すべての } \tilde{A} \in \mathbf{A}, \tilde{x} \in \mathbf{X} \text{ に対して } \tilde{A}\tilde{x} \in \mathbf{Y}$$

である．これは重要な原則，すなわち以下のように定義される包含同調性 (inclusion isotonicity) の具体例の一つである．

包含同調性 (Inclusion isotonicity)

$\tilde{a} \in A, \tilde{b} \in B$ に対して $\tilde{a} \circ \tilde{b}$ が定義されているようなすべての区間量 A, B 及びすべての演算 $\circ \in \{+, -, \cdot, /\}$ について

$$\text{すべての } \tilde{a} \in A, \tilde{b} \in B \text{ に対して } \tilde{a} \circ \tilde{b} \in A \circ B \tag{9}$$

が成り立つ．

よくある区間演算の落とし穴を紹介しよう．$A \in \mathbb{R}^{n \times n}$，$b \in \mathbb{R}^n$ 及び $\mathbf{X} \in \mathbb{IIR}^n$ が与えられたとき，$A^{-1}b \in \mathbf{X}$ を検証するために $b \in A\mathbf{X}$ を調べる．この論理は実は正しくない．なぜなら

$$\forall\, x \in \mathbf{X} : \quad Ax \in A\mathbf{X}$$

であるが，必ずしも

$$y \in A\mathbf{X} \quad \Rightarrow \quad \exists\, x \in \mathbf{X} : \quad y = Ax$$

ではないからである．

これらの区間演算の基礎は幾度となく述べられてきたので，興味のある読者は文献 [1, 21, 31] や，より詳細についてはそれらに引用されている文献を参照してほしい．

複素区間演算も同様に定義できることについて簡単に触れておこう．この場合，関数論的な理由のため，円板，すなわち中心・半径を用いた演算のほうがより適していることが多い (ただし，矩形演算も使う)．$a \in \mathbb{C}, 0 \leq r \in \mathbb{R}$ に対し

$$\langle a, r \rangle := \{z \in \mathbb{C} : \ |a - z| \leq r\} \tag{10}$$

と定義する．そのとき，包含同調性を満たす区間演算は，たとえば文献 [10] の中で

$$\langle a, r \rangle \cdot \langle b, s \rangle := \langle ab, \ |a|s + r|b| + rs \rangle \tag{11}$$

と定義されている．これに対応する複素区間ベクトル・行列演算も式 (8) と同様に定義される．

5　有向丸め

ここまでは，区間の上限・下限が実数で，演算が正確に実行される場合の定義を紹介してきた．浮動小数点演算の結果は一般に浮動小数点数ではないので，電子計算機上での実装には特別な注意を払わなければならない．この問題は，有向丸めを使うことによって解決できる．まず，両端を浮動小数点数とする区間の集合 $\mathbb{IIF}$ が

$$A \in \mathbb{IIF} \ :\Leftrightarrow \ A \neq \emptyset \ \text{ 及び } \ \underline{a}, \overline{a} \in \mathbb{F} \text{ に対し } A = [\underline{a}, \overline{a}] = \{x \in \mathbb{R} : \ \underline{a} \leq x \leq \overline{a}\}$$

によって定義されることを思い出してみよう．ここで，2 つの浮動小数点数による境界 $\underline{a}$ と $\overline{a}$ により，それらの間にあるすべての実数の無限集合が表現されていることに注意する．$\underline{a} = \overline{a}$ と縮退した場合にはこれはひとつの浮動小数点数を表す．つまり任意の浮動小数点数は点区間で表現することが可能であり，言い換えれば，$\mathbb{F} \subseteq \mathbb{IIF}$ と自然な埋め込みが可能である．両端が浮動小数点数であるような区間を用いて計算するとき，包含同調性 (9) を維持するために下限は下向き丸めで計算し上限は上向き丸めで計算するということを除いて，上記と同じ定義が使用できる．たとえば，式 (6) は

$$[\underline{a}, \overline{a}] + [\underline{b}, \overline{b}] = [\mathrm{fl}_\nabla(\underline{a} + \underline{b}), \mathrm{fl}_\Delta(\overline{a} + \overline{b})]$$

及び

$$[\underline{a}, \overline{a}] - [\underline{b}, \overline{b}] = [\mathrm{fl}_\nabla(\underline{a} - \overline{b}), \mathrm{fl}_\Delta(\overline{a} - \underline{b})]$$

で置き換えられる．その他の演算の定義も同じ方針に従う．たとえば，$\mathbf{Y} = \mathbf{AX}$ の定義は変わらず，区間演算における和や積が浮動小数点数の両端を用いて実行され，それらの演算が有向丸めを伴って遂行されるだけである．1993 年までは，すべての区間演算ライブラリは区間行列や区間ベクトルの演算にこの定義を用いていた．

しかしながら，このアプローチは現在の計算機上ではとても低速である．昔は，性能は主に乗算回数で決まっていたのに対し，今日ではデータの配置，キャッ

表 1: 次元数 1000 の行列乗算に対する異なる方式ごとの性能比較

	kji	jik	BLAS
計算時間 (秒)	12.3	1.1	0.68

表 2: 点行列と区間行列の乗算に対するアルゴリズムの性能比較, $n = 1000$

	伝統的な区間演算	PROFIL	新方式	近似計算 (BLAS)
計算時間 (秒)	47.6	8.8	1.3	0.68

シュミス，分岐などが非常に強く計算時間に影響を与える．例として，表 1 は 2 つの 1000×1000 行列の乗算 (近似計算) に対する標準的な PC での性能評価であり，計算のループ順を kji にした場合, jik にした場合，及びレベル 3 BLAS [7] を用いた場合のそれぞれの計算時間 (秒) を表す．レベル 3 BLAS の速度は最適化されたメモリアクセスとブロック化によるものである．

同様にして，区間演算の定義を変えることで性能を向上させられる．この方向の第一段階は，PROFIL ライブラリ [17] である．現在，区間演算の実装で最も高速なものは文献 [27] に載っている方法を使うことである．これは BLAS ルーチンを最大限に活用し，分岐もなく，とても少ない回数の丸めモードの変更しか必要としないものであり，中心・半径を用いた演算を利用する．それには，下端・上端型から中心・半径型への変換やその逆のために，Oishi による巧妙な方法 [25] が用いられている．サイズが 1000×1000 の点行列と区間行列の乗算について，式 (8) に基づく区間演算の伝統的な方式，PROFIL 及び文献 [27] の新方式による性能 (単位は秒) をそれぞれ表 2 に載せる．比較のために，要素が浮動小数点数の行列同士の乗算 (近似計算) に対してレベル 3 BLAS を用いたときに要した時間も再度載せる．

話を先に進める前に，それらすべての区間演算が多くのソフトウェアパッケージにおいて実装されていることについて述べておく．最も使いやすいのは INTLAB [28](Matlab の区間演算ツールボックスとしての最近の実装) であり，INTLAB では新しい定義 [27] が実装されている．文献 [12] が INTLAB の良い入門やチュートリアルとなる．INTLAB は，組み込みの `intval` と呼ばれる新しいデータ型を持ち，演算子多重定義によって `intval` 量と何か他のもの (たとえば，実数や複素数，あるいは実区間や複素区間) との間のあらゆる演算は区間演算として認識され，有向丸めを用いて実行される．たとえば，行列の正則性を確かめる上記のアルゴリズムの INTLAB の実装 (すなわち実行コード) は，次のようになるだろう．

プログラム 1 A の正則性 ($|I - RA|x < x$) を調べる INTLAB プログラム:

```
R = inv(A);
```

```
C = abss(eye(n)-R*intval(A));
x = ones(n,1);
setround(+1)
nonsingular = all( C*x < x )
```

与えられた次元数 n の正方行列 A に対し，まず浮動小数点演算を単純に用いて近似逆行列 R を計算する．その後，型キャスト `intval(A)` によって行列 A が区間行列に変換されるので，区間演算で乗算 `R*intval(A)` が実行される．結果は `intval` 型となり，そのため減算 `eye(n)-R*intval(A)` も区間演算の減算となる．関数 `abss` は，区間行列 X に対して

$$\mathrm{abss}(X) := \max\{|\tilde{X}| : \tilde{X} \in X\}$$

で定義される．ただし，max は要素毎に解釈される．これは

$$|I - RA| \leq C$$

を導く．ベクトル x は要素がすべて 1 のベクトルとして定義され，`setround(+1)` は丸めモードを上向きに変更する．これは，非負の量 C と x の乗算 `C*x` が真の値 $(|I - RA|x)$ の上限であることを意味する．正則性の判定がその後に続く．

`nonsingular=1` という評価は A が正則であることを証明するのに対して，`nonsingular=0` はアルゴリズムが正則性の証明に失敗したことを意味することに注意しよう．間違った答えを与えるのではなく，答えは「わからない」と解釈される．だから，この自己検証的算法は (非常に悪条件な行列に対して) 正則性の証明に失敗するかもしれないが，決して間違った答えは与えない．対照的に，計算機代数アルゴリズムは多数桁演算すなわち無限精度で計算したとき常に正則か特異かを判定する．

この段階で，点データに対して作られた多くの自己検証的算法は，区間データに対応するものに変換できることを注意しておく．上記の事例で区間行列 $A \in \mathbb{IF}^{n\times n}$ が与えられたとする．それは，区間エントリの範囲内にあるすべての実行列 $\tilde{A}$ の集合を表す．問題は，A の範囲内にあるあらゆる行列 $\tilde{A}$ が正則かどうかということである．以下のアルゴリズムを考えてみよう．

プログラム 2 区間行列の正則性を調べる INTLAB アルゴリズム

```
R = inv(mid(A));
C = abss(eye(n)-R*intval(A));
x = ones(n,1);
setround(+1)
nonsingular = all( C*x < x )
```

プログラム 1 との違いは最初の行だけである．ここで，`mid(A)` は A の中心行列を意味する．`nonsingular=1` という結果により，あらゆる行列 $\tilde{A} \in A$ が正則であることを証明している．この「証明」はとても簡潔であり，自己検証的

算法に特有のものである．$\tilde{A} \in A$ を固定されているが任意の (実) 行列とする．そのとき，包含同調性の基本原理 (9) から

$$I - R\tilde{A} \in I - RA \quad \text{したがって} \quad |I - R\tilde{A}| \leq C$$

となりその結果，主張は正しいことがわかる．要点は，入力データから任意にデータを取り出して固定し，包含同調性を適用することである．区間データにとって真であることは，区間データから取り出したあらゆる点データにとっても真である．

区間行列の正則性の証明は自明ではないことについて述べておく．実際，これはNP 困難問題である (文献 [26] を参照)．また，今回の実装は $\|I - RA\|_\infty < 1$ を調べていることに他ならないことも指摘しておく (それはもちろん R と A の正則性を意味するのだが)．ベクトル `x=ones(n,1)` で正則性を証明できなければ，`x` を `C*x` によって更新することにより `x` を反復計算してもよい．これは C の Perron ベクトルを近似する，べき乗法である．

6 自己検証的算法

これまでの議論は，単純かつ強力な原理，すなわち「算術演算が対応する区間演算と入れ替わったなら，必ず真の結果は得られた区間の要素になる」ことに従った．この手法は多くの数値計算アルゴリズム (たとえば線形方程式 $Ax = b$ に対する Gauss 消去法) に適用されうる．線形方程式 $Ax = b$ の例でいうと，実行後，必ず真の解は計算された区間に含まれる．

この手法は「素朴な区間演算 (naive interval arithmetic)」とも呼ばれ，区間演算の最もよくある誤用例である．一般的な応用に対して，このアプローチはほとんど確実に失敗する．その主張，すなわち，最終的な区間の結果が線形方程式の正確な解を含むというのは正しい．しかしながら，このアプローチでは，ピボット要素にゼロを含んでしまい，早々に計算が破綻することがあるし，そうでなくても広すぎる区間解が得られたりする．理由は，区間の依存性である．

本論文では，スペースの都合により自己検証的算法について詳しく述べることができない．興味のある読者は文献 [1, 31, 33] を参照されたい．ただし，少なくともいくつかの重要な原理を述べ，それに基づいて上の方法がなぜ誤用なのかを強調しておきたい．

例題として，行列式の包含を計算することを取り上げる．行列式の近似を計算する「何よりも良い方法」(文献 [14], 式 (14.34)) は，LU 分解から U の対角成分の積を用いることである．これは素朴な区間演算でも実行できる．一方，これに対する自己検証的算法は，以下の実行可能な INTLAB プログラムによって記述される．

```
[L U P] = lu(A);
Linv = inv(L);
```

```
Uinv = inv(U);
B = Linv*intval(P*A)*Uinv;
M = mid(diag(B));
G = midrad(M,abss(sum(B-diag(M),2)));
d = prod(G)/prod(intval(diag(Uinv)))/det(P)
```

ここで，Linv と Uinv は A の近似 LU 分解要素のそれぞれの近似逆行列であり，前処理行列として働く．B は Linv*P*A*Uinv の包含であるので，B の行列式は +/-det(A)*prod(diag(Uinv)) を包み込み，符号は置換行列 P に依存する．その次の 2 つのステートメントは B の Gershgorin 円の包含を計算するが，その積は Linv*P*A*Uinv の固有値の積の包含であり，その結果，行列式の包含と等しくなる．最終行は A の行列式の包含を計算する．ここで，区間演算は 4,6,7 行目だけに現れることに注意しよう．前処理行列 Linv と Uinv は要素が浮動小数点数の行列であり，行列 B(その Gershgorin 円が計算される) は単位行列に近い．

要点は

- 可能なところでは元のデータを使うこと
- 可能なところでは浮動小数点演算と近似値を使うこと
- 起こりうる過大評価が最小になるようなアルゴリズムを設計すること

である．図 1 は，次元数が 10 から 500 の問題に対して行列式の計算の包含を求めるとき，素朴区間 Gauss 消去法による結果 (“×” で表示) 及び上記のアルゴリズム (すなわち，前処理行列を適用した Gershgorin 円によるもの) による結果 (“○” で表示) について，その精度を相対誤差として表す．すべての行列は，要素が $[-1, 1]$ の範囲の一様乱数であるように生成された乱数行列である．たとえば，次元数 500 の行列の行列式は，我々のアルゴリズムでは倍精度計算でおよそ 8 桁まで正しく求められる．素朴な区間演算によるアプローチでは 60 より大きな次元数の行列を取り扱うことができない．これはピボット要素がゼロを含む区間値となるからである．ここで，これは悪条件の行列の問題ではなく (この例で扱ったすべての行列の条件数は，最大でも 5×10^3 より小さい),「単に」データ依存の問題であることに注意しよう．

ここで得られた教訓は，数値計算に対してうまくいくことが必ずしも区間演算によるアプローチに対して良策ではない，ということである．我々は与えられた問題に対して個別の方法を開発する必要がある．標準の数値解法は手助けになるが，通常はさらにいくつかの工夫が必要である．上記の話は，そのひとつの典型的な例である．

区間演算による方法は,「どのような」数値問題を解く場合でも，素朴な区間演算を用いる容易でエレガントな方法として支持されたことがあった．公平な比較のためには，60 年代は計算機とプログラミング言語が黎明期であったことをここで言わなくてはならない．今日，我々が当たり前のように持っている道具と比べれば，これはスポーツカーに対する馬車のようなものである．特に，

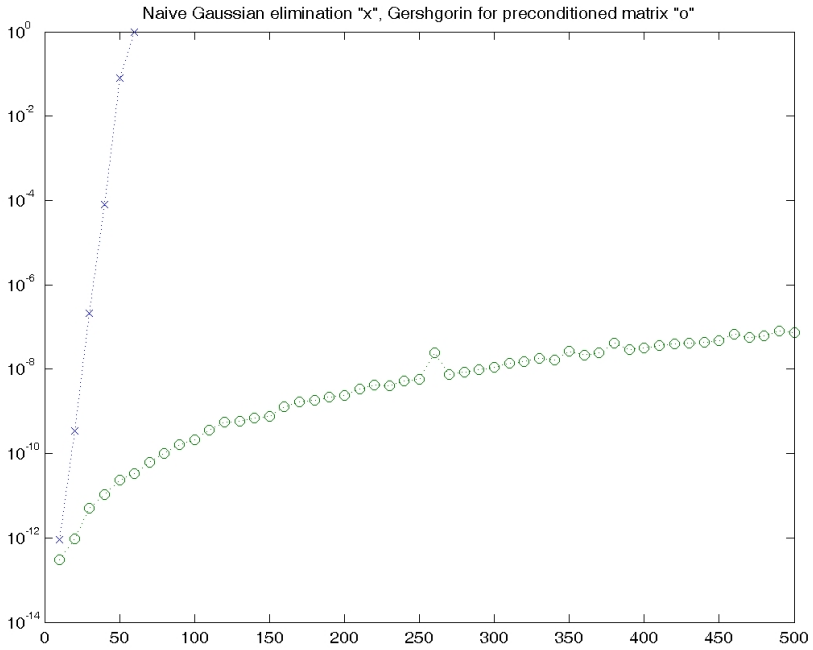

図 1: 素朴な区間演算によるアルゴリズムと自己検証的算法の比較

Matlab のように強力な対話式の数値計算プログラミング環境が広く利用できたわけではなかった．そのため，我々が現在たやすく当たり前のように行っている数値実験は，当時は簡単ではなかったわけである．

だから，他の人がある主張を確かめようとすることができるまで，かなりの時間がかかったのも不思議ではない．それに加えて，区間演算がほとんど利用できなかったということも妨げとなった．そして，計算機はまだ低速であった (文献 [14] の表 9.2 を参照)．

だが，その後，人々はたとえば素朴な区間 Gauss 消去法のような標準的な例を試し，その結果，相当な過大評価があると認めなければならなかった．実際，一般的な場合についても指数関数的な過大評価は珍しくないばかりでなく典型的であった．その結果，区間演算推進者たちが自ら評判を落とす結果になってしまったのはやむを得ないことであり，これが今もなお区間解析の評価が分かれることの理由である．

素朴な区間演算は自動化された前進誤差解析とみなすこともできる．すなわち，個々の操作について最悪の場合を見積もる．Gauss 消去法に対しては，これが Goldstine と von Neumann の有名な論文 [23] ですでになされている．結果はとても悲観的なものであった — 素朴な区間演算の結果のように．この論文がもとで，しばらくの間，丸め誤差の蓄積のために，より大きな線形方程式では電子計算機上で信頼性のある解が得られないという結論にさえ至っていた．ここで,「大きな」というのは次元数が 10 程度を意味した — たとえば，1949 年に

10×10 の線形方程式が IBM 602 上で 4 時間かけて解くことに成功した [38].

この種の自動化された前進誤差解析，すなわち素朴区間解析は個々の操作について最悪の場合の誤差を見積もり，すべての操作を独立したものとして扱う．しかし，丸め誤差は独立したものではなく，また決してランダムに分布するわけでもない (Kahan の文献 [16] を参照).

区間解析はその後発達し，そして素朴な区間演算のアプローチの時代は久しく終わっている．過大評価はあるものの，素朴な区間演算の評価は領域上で関数の値域を見積もるための非常に強力なツールを提供する．そして，この評価は有効で厳密であり，関数の挙動についてのいかなる知識がなくても単純な方法で得られる．これは注目すべき特徴である．たとえば，実行可能な INTLAB のステートメント

```
f=inline('x*cos(x^2)-tanh(x-asin(x/5))'), X=infsup(-1,3), f(X)   (12)
```

は解

```
intval ans = [   -3.9967, 3.9280]
```

を与え，これは与えられた関数 f の値域は表示された範囲内にあることを証明する．このステートメントは INTLAB [30] で実装された厳密な (区間) 標準関数と厳密な入出力 [28] に基づく．真の値域はおよそ $[-3.7156, 1.5509]$ である．(多変数の) 関数の値域に対する厳密な囲い込みは，特に大域的最適化で非常に有用となりうる．

7 おわりに

本論文ではスペースの都合上，自己検証的算法について詳しく述べることができなかった．興味のある読者には文献 [1, 31, 33] やそこで引用されている文献を参照してほしい．特に，例と応用について話すことができなかった．自己検証的算法は多くの未知数を持つ大規模問題，特に計算機援用証明に関連した問題に適用されている．それらの例として

- Lorenz アトラクタの存在検証 [37]
- カオスの存在検証 [22]
- double-bubble 予想 [13]
- Blasius 分析を用いた Orr-Sommerfeld 方程式に対する不安定性の検証 [18]
- Jouanolou 葉層の力学 [5]
- Feigenbaum 定数の検証された包み込み [8]
- Sturm-Liouville 問題の本質的スペクトル以下の固有値の存在 [2]

- タービンの固有振動数 (Kulisch ほか)
- 回路解析のための SPICE プログラム (Rump, 未発表)
- Constance 湖中の極端な流れ (Rump, 未発表)
- 森林計画 [15]

などがある．これらのいくつかについては，文献 [9] により詳細に述べられている．大域的最適化，区域内の非線形方程式の「すべての」ゼロ点，スパースな線形・非線形問題，初期値問題・境界値問題，偏微分方程式などの数値計算的な数学の様々な分野に対する自己検証的算法の文献リストについて，興味のある読者は文献 [33] を参照してほしい．これまでに，多くの日本の研究者が自己検証的算法について研究を行い，成功している (たとえば，[19, 20, 24, 35, 39] を参照)．最先端では，全体で従来の数値計算アルゴリズムと「同じ」計算時間で済むような自己検証的算法を設計するために非常に面白い研究が始まった．これらは，Oishi [25] やその他の研究者によるものである．

我々は「証明とは何か？」という問いと共に出発した．以前言及したように，それに対する答えを与えるつもりではなかった (またそれはできなかった)．しかし，自己検証的算法 (これは浮動小数点演算で実行されるのだが) を使うことが数学的な主張の正当性を検証するための重要な方法とみなすことができるという，いくらかの見解を加えたことを願う．自己検証的算法は決して数学的な証明を置き換えることを意図するものではないが，適切な方法で道具を使用するとき，それは助けになるかもしれない．

最後に，自己検証的算法は従来の数値解法を置き換えるために設計されているわけでもないことを強調したい．この事実は，1) ほとんどの自己検証的算法は検証を始めるために数値計算による良い近似を頼っていること，2) 正反対に，区間演算は浮動小数点演算の結果を包含しなければならないことから，非常に明白である．これは，連立非線形方程式または偏微分方程式のような数値計算問題の直接解に適用される．しかし，自己検証的算法はさらに多くのことを行う．それは，求めた範囲における解の存在と，可能であれば一意性を確かめることである．これは，従来の数値計算アルゴリズムの範囲を越えたものである．

その他の応用範囲は大域的最適化法である．関数の値域の包含 (式 (12) にあるような素朴な区間演算よりも非常に洗練されたものだが) が，ある区域には大域的な最小値が含まれないことを証明することによって，その区域を捨て去るために用いられる．これは，近年のとても興味深い結果と共に非常に有望な分野である．

Matlab のツールボックスである INTLAB は，自己検証的算法に精通し問題を解くためのやさしい手段である．それは，非商用利用であれば我々のホームページ [†1] から自由に入手可能である．INTLAB の非常に良いチュートリアルは文献 [12] である．あらゆるルーチンは，入出力及びルーチンのふるまいを説明

[†1] http://www.ti3.tu-harburg.de/~rump/intlab/

するヘッダと共にともに手に入る．現在までに，推定で 40 以上の国に 3500 人のユーザがいる．

最後に，自己検証的算法は結果の正当性を (必要に応じて) 確かめるための一つの選択肢であることを述べておきたい．その意味で，自己検証的算法に関する研究は計算数学の分野に属していると言える．

謝辞

英語の原稿を翻訳する労をとっていただいた早稲田大学の荻田武史博士に深謝いたします．

参考文献

[1] Alefeld, G., and Mayer, G., Interval analysis: theory and applications, J. Comput. Appl. Math., 121:1–2 2000, 421–464.

[2] Brown, B. M., McCormack, D. K. R., and Zettl, A., On the existence of an eigenvalue below the essential spectrum, Proc. R. Soc. Lond., 455 (1999), 2229–2234.

[3] Buchberger, B., Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal, PhD thesis, Universität Insbruck, 1965.

[4] Buchberger, B., Gröbner bases: an algorithmic method in polynomial ideal theory, Recent Trends in Multidimensional System Theory (edited by Bose, N. K.), Reidel, 1985.

[5] Camacho, C., and de Figueiredo, L. H., The dynamics of the Jouanolou foliation on the complex project 2-space, Ergod. Th. & Dynam. Sys., 21:3 (2001), 757–766.

[6] Collins, G. E., Quantifier elimination for real closed fields by cylindrical algebraic decompositon, Autom. Theor. form. Lang. (edited by Caviness, B.F. et al.), Lect. Notes Comput. Sci., 33 (1975), 134–183.

[7] Dongarra, J. J., Du Croz, J. J., Duff, I. S., and Hammarling, S. J., A set of level 3 Basic Linear Algebra Subprograms, ACM Trans. Math. Software, 16 (1990), 1–17.

[8] Eckmann, J.-P., and Wittwer, P., Computer methods and Borel summability applied to Feigenbaum's equation, Lecture Notes in Physics, 227, Springer Verlag, Berlin Heidelberg New York Tokyo, 1985.

[9] Frommer, A., Proving conjectures by use of interval arithmetic, Perspectives on Enclosure Methods (edited by Kulisch, U. et al.), Springer, Wien, 2001.

[10] Gargantini, I., and Henrici, P., Circular arithmetic and the determination of polynomial zeros, Numer. Math., 18 (1972), 305–320.

[11] Grabmeier, J., Kaltofen, E., and Weispfennig, V., Computer Algebra Handbook, Springer, 2003.

[12] Hargreaves, G., Interval Analysis in MATLAB, Master's thesis, University of Manchester, 2002. http://www.ma.man.ac.uk/~hargreaves/thesis.ps

[13] Hass, J., Hutchings, M., and Schlafly, R., The double bubble conjecture, Elec. Res. Announcement of the Am. Math. Soc., 1:3 (1995), 98–102.

[14] Higham, N. J., Accuracy and Stability of Numerical Algorithms, 2nd edition, SIAM Publications, Philadelphia, 2002.

[15] Jansson, C., Zur Linearen Optimierung mit unscharfen Daten, Dissertation, Universität Kaiserslautern, 1985.

[16] Kahan, W., The Improbability of PROBABILISTIC ERROR ANALYSES for Numerical Computations, manuscript, March 1996, 34 pages. http://www.cs.berkeley.edu/~wkahan/improber.pdf

[17] Knüppel, O., PROFIL / BIAS — A fast interval library, Computing, 53 (1994), 277–287.

[18] Lahmann, J., and Plum, M., A computer-assisted instability proof for the Orr-Sommerfeld equation with Blasius-profile, ZAMM, to appear.

[19] Nakao, M. R., A numerical approach to the proof of existence of solutions for elliptic problems, Japan J. Appl. Math., 5:2 (1988), 313–332, .

[20] Nakao, M. T., Watanabe, Y., Yamamoto, N., and Nishida, T., Some computer assisted proofs for solutions of the heat convection problems, Reliable Computing, 9:5 (2300), 359–372.

[21] Neumaier, A., Interval Methods for Systems of Equations, Encyclopedia of Mathematics and its Applications, Cambridge University Press, 1990.

[22] Neumaier, A., and Rage, Th., Rigorous chaos verification in discrete dynamical systems, Physica D, 67 (1993), 327–346.

[23] Neumann, J. v., and Goldstine, H. H., Numerical inverting of matrices of high order, Bull. Amer. Math. Soc. 53 (1947), 1021–1099.

[24] Oishi, S., Numerical verification of existence and inclusion of solutions for nonlinear operator equations, J. Comput. Appl. Math., 60:1–2 (1995), 171–185.

[25] Oishi, S., and Rump, S. M., Fast verification of solutions of matrix equations, Numer. Math., 90:4 (2002), 755–773.

[26] Poljak, S., and Rohn, J., Radius of nonsingularity, KAM Series 88–117, Charles University, Prague, 1988, 11 pages.

[27] Rump, S. M., Fast and parallel interval arithmetic, BIT, 39:3 (1999), 539–560.

[28] Rump, S. M., INTLAB — INTerval LABoratory, Developments in Reliable Computing (edited by Csendes, T.), Kluwer Academic Publishers, Dordrecht, 1999, 77–104. http://www.ti3.tu-harburg.de/english/index.html

[29] Rump, S. M., Computational error bounds for multiple or nearly multiple eigenvalues, Linear Algebra and its Applications, 324 (2001), 209–226.

[30] Rump, S. M., Rigorous and portable standard functions, BIT, 41:3 (2001), 540–562.

[31] Rump, S. M., Self-validating methods, Linear Algebra and its Applications, 324 (2001), 3–13.

[32] Rump, S. M., Ten methods to bound multiple roots of polynomials, J. Comput. Appl. Math., 156 (2003), 403–432.

[33] Rump, S. M., Computer assisted proofs and self-validating methods, Handbook on Accuracy and Reliability in Scientific Computation (edited by Einarsson, B.), to appear, 55 pages.

[34] Sunaga, T., Theory of an interval algebra and its application to numerical analysis, RAAG Memoirs, 2 (1958), 29–46.

[35] Tanaka, K., Murashige, S., and Oishi S., On necessary and sufficient conditions for numerical verification of double turning points, Numer. Math., 97:3 (2004), 537–554.

[36] Tarski, A., A Decision Method for Elementary Algebra and Geometry, RAND Corp., 1948.

[37] Tucker, W., The Lorenz attractor exists, C. R. Acad. Sci., Paris, Sér. I, Math., 328:12 (1999), 1197–1202.

[38] Verzuh, F. M., The solution of simultaneous linear equations with the aid of the 602 calculated punch, M.T.A.C., 3 (1949), 453–462.

[39] Yamamura, K., An algorithm for representing nonseparable functions by separable functions, IEICE Trans. Fundamentals, E79-A (1996), 1051–1059.

図 目 次

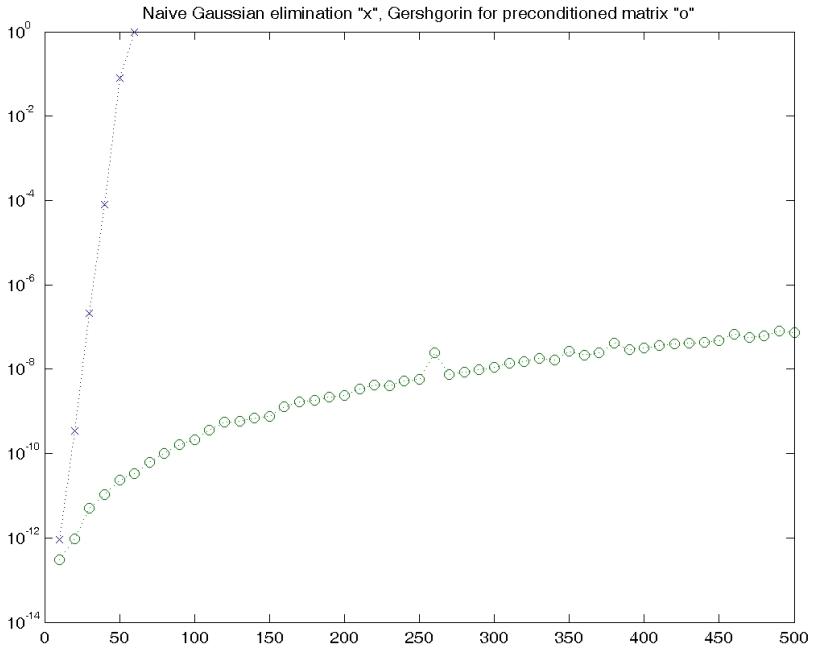

図 1: 素朴な区間演算によるアルゴリズムと自己検証的算法の比較

表 目 次

表 1: 次元数 1000 の行列乗算に対する異なる方式ごとの性能比較

	kji	jik	BLAS
計算時間 (秒)	12.3	1.1	0.68

表 2: 点行列と区間行列の乗算に対するアルゴリズムの性能比較, $n = 1000$

	伝統的な区間演算	PROFIL	新方式	近似計算 (BLAS)
計算時間 (秒)	47.6	8.8	1.3	0.68