解　説

5. 区間演算を用いた丸め誤差解析†

久保田　光　一††　伊　理　正　夫†††

1. まえがき

数値計算結果に有効数字が何桁あるかというような形で，計算値の精度を保証するには，その結果に含まれる「誤差」の絶対値のなるべく小さな上界が分かればよい．そうすれば，計算値を中に含む区間で，そのなかに無限精度計算の結果の値が存在するようなものが得られるからである．そのための一つの手法として，以前から，(機械) 区間演算 (3. 参照) が使われてきた．しかし，単純に区間演算を実行するだけでは，大規模な問題に対しては，最終的に保証される区間の幅が広すぎて実用の役には立たないということも，また，経験的に知られていた．それに対して，最近，高速自動微分法による丸め誤差評価法と区間演算とを組み合わせれば，保証区間幅を飛躍的に（比にして，10^{-4}～10^{-8} 程度にまで）狭められることが理論的および数値実験により示された[10]．本稿ではその手法を紹介し，さらに代償として増加する所要時間，所要領域がどの程度であるかを述べる．丸めのない区間演算と高速自動微分法との組合せは Yu. V. Matiyasevich (Hilbert の第 10 問題を否定的に解決したこととでも有名な人物) により構成的実数論の基礎付けのために理論的に考察されていたが[12]，ここで述べる方法は彼の方法を丸め誤差解析に応用したものとみなせる[10]．本稿とは異なる接近法として，広すぎる機械区間演算結果を用いながら，区間に関する反復計算などによって間接的に保証区間を狭める方法も提案され，実用化されつつあるが[2),14)]，ここで述べる方法は区間幅を直接狭めるものであり，またある意味で最狭の区間幅を与えるものであるので，より基本的な技法であるといえよう．もちろん，この方法による保証区間を利用して，さらに区間反復計算などを実行することもできる．

本稿では，「誤差」を，浮動小数点数を用いる数値計算に不可避の丸め誤差（計算誤差）に限定して考える．実際の応用の場面においては，離散化誤差なども含めた「誤差」を評価することも重要であるが，そのためにも計算誤差の評価法を確立しておくことが必要なのである．

一方，厳密に精度を保証するという立場を離れるのならば，機械区間演算と組み合わせることなく高速自動微分法による丸め誤差評価値だけを利用しても計算値の精度をかなり正確に知ることができる．とくに，絶対評価と呼ばれる丸め誤差評価値 (4. 式 (4.2) 参照) は丸め誤差の絶対値の上界を近似しているだけでなく，ほとんどの場合，実際に計算値に含まれる丸め誤差の絶対値よりも大きく（過大評価に）なっていることが実験的に確認されている[6),7)]．したがって，本稿で説明する手法はあくまで厳密な精度保証を追求するときのひとつのやり方である．

以下では，まず，浮動小数点演算に対する仮定を述べ (2.)，精度を保証するために従来から使われていた「機械区間演算」を紹介する (3.)．つづいて，高速自動微分法による誤差の近似値の計算について説明し，それと機械区間演算とを組み合わせることによって計算誤差の絶対値の厳密な上界で（計算精度を上げていったとき）漸近的に計算誤差の上限を与える方法が得られることを示す (4.)．さらに，それらの算法の実現法を概説する (5.)．すべてをソフトウェアでシミュレートした場合，簡単なテストプログラムでは，浮動小数点演算を 1 秒間に約 200 万回実行できる計算機上で，機械区間演算は約 4 万回／秒，高速自動微分法のための計算グラフを残す演算が約 2 万回／秒，両者を合わせると約 1 万回／秒の速度で演算を実行できた．演算終了後に丸め誤差評価値を計算する時間も考慮に入れると，浮動小数点演算時間の約 800 倍の時間で保証区間を計算できることになる．より本格的な問

† Rounding Error Analysis with Interval Operations by Koichi KUBOTA (Department of Administration Engineering, Faculty of Science and Technology, Keio University) and Masao IRI (Department of Mathematical Engineering and Information Physics, Faculty of Engineering, University of Tokyo).

†† 慶応義塾大学理工学部管理工学科
††† 東京大学工学部計数工学科

題として，線形方程式系の解法を取りあげ，上と同様に速度を比較する (6.).

2. 浮動小数点演算に対する仮定

本稿で考察の対象とする関数は，数値計算の現実的な環境を大前提として，+，−，*，／，√，exp，log，sin，cos などのその定義域の内部で1階連続微分可能な「基本演算」を次々と用いて値を計算する「手続き」(プログラム) が与えられるようなものとする．(基本演算は実際には浮動小数点演算で実行される.) すなわち，微分方程式などで定義されるような関数を直接取り扱うのではなく，数値計算ができるようにモデル化 (たとえば離散化) して，計算手順が与えられた関数について，その計算値に含まれる誤差を考える．入力データ $\boldsymbol{x}(=(x_1,\cdots,x_n)$ というベクトルで表す) を与えて，分岐・反復のある手順にしたがって計算した結果，計算値 y が得られたとき，その計算の履歴を計算過程と呼び，一般に，図-1 のような形式で表す．計算過程は基本演算を実行してその (中間) 結果を「中間変数」v_j に代入するという「計算ステップ」の列である．もちろん，入力 $\boldsymbol{x}$ の値を変えれば，手続き実行中の分岐先，反復回数などが変化して計算過程の構造—計算ステップの引き数と実引き数との関係や計算ステップの総数—も変化することがある．しかし，入力 $\boldsymbol{x}$ の値が決まれば，誤差のない無限精度演算の履歴として計算過程は一意に定まり，計算結果 y は，基本演算の合成関数として表される $\boldsymbol{x}$ のある関数 f の関数値 $y=f(\boldsymbol{x})$ とみなすことができる．数学的な関数としては同一の (合成) 関数であっても，計算過程が異なれば誤差は異なりうるが，これは，同じものを計算する場合でも数値的に安定な算法とそうでない算法があるということに対応する．

基本演算に関して次のような状況を設定する：

「浮動小数点数を被演算数とする基本演算の結果を浮動小数点数に丸めるために発生する誤差 (「発生誤差」) についてはその大きさの上限だけが分かり，符号，大きさは不明である.」

$$
\begin{aligned}
v_1&\leftarrow\psi_1(u_{11},u_{12})\\
&\vdots\\
v_j&\leftarrow\psi_j(u_{j1},u_{j2})\\
&\vdots\\
(y=)\ v_r&\leftarrow\psi_r(u_{r1},u_{r2})
\end{aligned}
$$

$v_1,\cdots,v_r$: 中間変数；
$\psi_1,\cdots,\psi_r$: 基本演算；
u_{j1},u_{j2}: ψ_j の引数 (入力変数 $x_1,\cdots,x_n$，定数，中間変数 $v_1,\cdots,v_{j-1}$ のいずれかがこれらに対する実引数となる $(j=1,\cdots,r)$).

図-1 計算過程

(中間結果として計算値 v_j を得たときの発生誤差の絶対値の上限は，浮動小数点体系 (浮動小数点数の表現法と丸めの方式) に応じて定まるが，ここではその上限として，いわゆるマシンエプシロン* ε と計算値 v_j との積の絶対値 $|v_j|\cdot\varepsilon$ を用いることにする．ここで，上線は有限精度の数値に対応することを表す.) このような状況設定に対して，発生誤差の符号，大きさなどはある程度分かる場合もあるという主張もあるかもしれないが，そうするとそのような情報を用いて最終的な計算値 $\bar{y}$ が補正できることになる．しかし，そのようにして得られる補正値の精度を保証することを考えれば，そこでは上の状況が成立するであろう．すなわち，ここでは，補正，その補正，そのまた補正，という鎖を断ち切った状況を考えているのである．

計算過程における，発生誤差を明示して δ_j と記せば，各計算ステップは

$$\bar{v}_j\leftarrow\bar{\phi}_j(\bar{u}_{j1},\bar{u}_{j2})+\delta_j$$

と書ける．これを「丸め付き計算過程」と呼ぶことにする (図-2)．上の状況設定は，$|\delta_j|\le|\bar{v}_j|\cdot\varepsilon$ が成立することだけが分かるということである．

丸め付き計算過程について次の仮定を置く：

「有限精度計算の結果として得られる計算過程の構造は，無限精度計算による計算過程の構造と同じである.」

すなわち，有限精度計算による関数値計算の実行可能性を仮定する．有限精度計算では，入力 $\boldsymbol{x}$ の値が同じであっても，精度が低いとその計算過程が無限精度計算のそれと異なることがあるが，この仮定は計算過程が変化しない程度の精度をもった，有限精度計算を考

$$
\begin{aligned}
\bar{v}_1&\leftarrow\bar{\phi}_1(\bar{u}_{11},\bar{u}_{12})=\phi_1(\bar{u}_{11},\bar{u}_{12})+\delta_1\\
&\vdots\\
\bar{v}_j&\leftarrow\bar{\phi}_j(\bar{u}_{j1},\bar{u}_{j2})=\phi_j(\bar{u}_{j1},\bar{u}_{j2})+\delta_j\\
&\vdots\\
(\bar{y}=)\ \bar{v}_r&\leftarrow\bar{\phi}_r(\bar{u}_{r1},\bar{u}_{r2})=\phi_r(\bar{u}_{r1},\bar{u}_{r2})+\delta_r
\end{aligned}
$$

$\bar{v}_1,\cdots,\bar{v}_r$: 中間変数；
$\bar{\phi}_1,\cdots,\bar{\phi}_r$: 基本演算；
$\bar{u}_{j1},\bar{u}_{j2}$: $\bar{\phi}_j$ の引数 (入力変数 $x_1,\cdots,x_n$，定数，中間変数 $\bar{v}_1,\cdots,\bar{v}_{j-1}$ のいずれかがこれらに対する実引数となる $(j=1,\cdots,r)$)；
$\delta_1,\cdots,\delta_r$: 発生誤差.

図-2 丸めつき計算過程

* ある浮動小数点体系において，$1\oplus\varepsilon>1$ を満たす最小の浮動小数点数 ε を，その浮動小数点体系のマシンエプシロンと呼ぶ．ここで，$\oplus$ は浮動小数点数の加算である.

えるということである．この仮定の下では，精度と無関係に定まる計算過程を基本演算の合成関数fに対応付けることができる．

以上の仮定の下で精度を保証するということは，冒頭で述べたように，無限精度演算を実行してその履歴により定まる合成関数 f に入力データ x を与えたときの値 $f(x)$ を含む(なるべく狭い)区間を見出すことである．計算値 $\bar{y}$ に関する誤差評価値が Δ であれば，

$$[\bar{y}-\Delta, \bar{y}+\Delta] \ni f(x)$$

であることがいえ，この区間の幅 2Δ が $\bar{y}$ に保証される精度であるということになる．

3. 機械区間演算

無限精度演算結果を含む保証区間を与えるために従来よく用いられていた「機械区間演算」について説明する[1]．この機械区間演算を通常の浮動小数点演算の代わりに実行すると，無限精度計算結果を確実に含む区間を次々生成して，結果としてひとつの実数区間が得られ，その区間が $f(x)$ を含むことが保証できる．

機械区間演算とは，いわゆる区間演算—実数区間を被演算区間とし，区間内の任意の実数(の組)の実数演算結果をすべて含むような最小実数区間を結果とする演算—を用いて次のように定義できる．すなわち，両端点が浮動小数点数であるような実数区間を特に「機械区間」と呼ぶことにすれば，機械区間演算は，「機械区間を被演算区間とする実数区間演算の結果を丸めてそれを含む最狭の機械区間を結果とする演算」である．実数区間の全体を I，機械区間の全体を I_M，2項の実数演算を ϕ とすれば，ϕ に対応する区間演算 Φ は

$X, Y \in I$ のとき

$$\Phi(X, Y) = \{\phi(x, y) \mid x \in X, y \in Y\}$$

であり，機械区間演算 Φ_M は上の Φ を用いて

$X_M, Y_M \in I_M$ のとき

$$\Phi_M(X_M, Y_M) = \cap \{Z \mid \Phi(X_M, Y_M) \subseteq Z \in I_M\}$$

と表される．

ここで，2.での仮定「有限精度計算による関数値計算の実行可能性」と機械区間演算との関係をみよう．計算手続きにしたがって，機械区間演算により計算を完遂できた場合には，そこで用いた浮動小数点体系において，2.の仮定が成立することが保証される．つまり，絶対値がある上限以下であれば発生誤差がどのように変動しようとも，その浮動小数点体系では計算過程の構造が不変であるということである．しかし，手続き実行中に，0を含む区間による除算や，重なりがあるために大小関係の決まらない二つの機械区間の比較などが現れて，計算が続行できなくなる場合がある．その場合には，その浮動小数点体系では，その計算手順が数値的に確定するとは限らないことを意味する．

この機械区間演算結果は，原理的には，数値計算結果の精度を保証できるが，その保証範囲が広がりやすく，大規模な関数の計算に対しては，実際には役に立たないことも多い．

4. 計算誤差の推定値

丸め付き計算過程において，発生誤差 δ_j の計算値 $\bar{y}$ への寄与の総和

$$L_y \overset{\text{def}}{=} \sum_{j=1}^{r} \frac{\partial f}{\partial v_j} \cdot \delta_j \tag{4.1}$$

は，$\bar{y}$ に含まれる誤差 $\bar{y}-f(x)$ の線形近似式として古くから知られている[16]．δ_j の値は特定できないが，仮定 $|\delta_j| \leq |v_j| \cdot \varepsilon$ により，この式の絶対値の上界は

$$A_y \overset{\text{def}}{=} \sum_{j=1}^{r} \left|\frac{\delta f}{\delta v_j}\right| \cdot |v_j| \cdot \varepsilon \tag{4.2}$$

と評価できる．この A_y は「絶対評価」と呼ばれ，$|L_y| \leq A_y$ を満たす．以前は，全ての v_j について $\partial f/\partial v_j$ の値を計算することは手間の面からみて非現実的であったが，高速自動微分法[3)-5),8),9)]により高速に正確に自動的にそれらの値を計算できるようになったので，今や A_y も実際に計算できる量である．したがって，計算値 $\bar{y}$ とこの A_y とから $[\bar{y}-A_y, \bar{y}+A_y]$ なる区間を作れば，この中に $f(x)$ がほぼ確実に含まれるはずであり，実際に数値実験によっても A_y が誤差の上界を近似しているということが確かめられている[6),7),11),13)]．しかしながら，厳密な精度保証の立場からすると，理論的には少々問題がある．それは，式(4.1)がそもそも線形近似式であること，そして，A_y の計算自身にも浮動小数点演算を用いていることの二つの理由により，A_y は $|\bar{y}-f(x)|$ の上限の近似値ではあるが上界であるとは限らないからである．(S. Linnainmaa は誤差 $\bar{y}-f(x)$ の δ_j に関する Taylor 展開係数を計算し，高次の近似式も提案している[11)]．しかし，その近似値も上界とは限らない)．

3.の機械区間演算によれば $f(x)$ を含むことが保証される区間 Y を計算できるが，さらにそれと式(4.2)の絶対評価とを組み合わせると，$f(x)$ を含みかつ十分に狭い保証区間が得られる[10)]．原理的には微分の“中間値の定理”の自然で容易な応用であり，その算

法は至って簡単である．まず，機械区間演算を実行し，中間変数に対応する機械区間 $V_1, \cdots, V_r$ を計算する；高速自動微分法による $\partial f/\partial v_j$ の計算も機械区間演算により実行し，$\partial f/\partial v_j$ を含む機械区間 W_j を計算する $(j=1, \cdots, r)$；発生誤差 δ_j を含む区間 $[-|V_j|\cdot\varepsilon, |V_j|\cdot\varepsilon]$ を定め*，最後に，式(4.2)を，

$$L_Y \overset{\text{def}}{=} \bigoplus_{j=1}^{r} W_j \otimes [-|V_j|\cdot\varepsilon, |V_j|\cdot\varepsilon] \qquad (4.3)$$

のように，機械区間演算を用いて計算する（⊕は機械区間演算の総和，⊗は同じく乗算を示す）；この結果の機械区間 L_Y の絶対値が誤差の絶対値の上限に近い上界になる．これを

$$A_Y \overset{\text{def}}{=} |L_Y| \qquad (4.4)$$

と書けば，浮動小数点演算の結果 $\bar{y}$ に対して，$[\bar{y}-A_Y, \bar{y}+A_Y]$ が $f(x)$ を含むことが保証される．$\partial f/\partial v_j \neq 0$ という仮定のもとでは，マシンエプシロンで代表される浮動小数点体系の精度が高くなるにつれ，A_Y は実際に発生しうる丸め誤差の上限に漸近することも証明されている[10]．また，$\bar{y}$ を計算するのに実行した総演算回数 r の高々定数倍回の機械区間演算の実行により A_Y を計算できるということに注意することも重要である．すなわち，定数倍の時間と，計算時間に比例した領域とを用いることによって，数値計算結果を高精度で保証できるのである．

計算量が大きくなるという点からは実際的でないが，もっと凝った保証方法も考えられる．それは，機械区間演算と浮動小数点演算とを同時に並行して実行し，中間変数 v_j の値とそれを含む機械区間 V_j が定められた時点で，v_j に関して上の A_Y に相当する A_{V_j} を計算し，v_j を含む新たな機械区間として

$$V_j \leftarrow V_j \cap [v_j - A_{V_j}, v_j + A_{V_j}]$$

を採用するという方法である．この方法は計算過程の総ステップ数を r として，$O(r^2)$ の手間を要する．

なお，式(4.1)の評価として，「確率評価」と呼ばれるものもある．それは，丸めの方式が四捨五入の場合は，δ_j を $[-|v_j|\cdot\varepsilon, |v_j|\cdot\varepsilon]$ の上の一様分布に従う確率変数とみなしたときの式(4.1)の標準偏差値

$$P_v \overset{\text{def}}{=} \varepsilon \cdot \sqrt{\frac{1}{3}\sum_{j=1}^{r}\left(\frac{\partial f}{\partial v_j}\cdot v_j\right)^2} \qquad (4.5)$$

のことであり，P_v が実際に発生している丸め誤差の大きさをよく近似していることも実験的に確かめられている[6),7)]．

* 機械区間 $V_j \overset{\text{def}}{=} [v_j{}^l, v_j{}^h]$ の絶対値 $|V_j|$ は $|V_j| \overset{\text{def}}{=} \max\{|v_j{}^l|, |v_j{}^h|\}$ で定義する．

5. 保証つき誤差評価法の実現法

上述の評価を実際に計算するためには，機械区間演算と高速自動微分法とを，少なくともソフトウェアで，実現する必要がある．ここではその一例をあげる．

機械区間演算を忠実に実現するためには，ハードウェア支援あるいはより高精度の浮動小数点演算を必要とするが，ここでは，通常の浮動小数点演算により機械区間演算を模倣する．すなわち，実数区間演算の定義式（図-3）をそのまま浮動小数点演算で計算し，得られた機械区間の両端を広げて，最狭とは限らないが演算結果を確実に含む，機械区間を得る．

高速自動微分法を実行するには，基本演算を一回実行するたびにその演算の種類と被演算数とを履歴に残し，計算グラフ[4)-7),9)]（計算過程を表現する無閉路有向グラフ）を構築すればよい．計算が終了した時点で計算グラフを逆にたどって偏導関数 $\partial f/\partial v_j$ を計算しながら，式(4.2)，(4.3)を計算することができる．

以上を簡便に実現するため，プログラム言語 C++ の演算子多義化機能を用いて新たなデータ型を定義し，そのデータ型のためのライブラリを用意した．このデータ型を用いることにより，見掛けは倍精度（浮動小数点数）の変数／配列に関するプログラムとほぼ同じでありながら，以下の5種類の演算を遂行できる：

（1） 倍精度演算を実行する（もとのまま）；

（2） 両端点が倍精度浮動小数点数の機械区間演算を実行する；

（3） 高速自動微分法のために計算過程の履歴を残す；

（4） （2）の機械区間演算を実行し履歴を残す；

（5） 前章最後に述べたように，（4）に加えて倍精度演算も実行し，機械区間とその中に含まれる浮動小数点数とを組にして記憶し，その時点までの誤差評価を行うことによって，演算結果を含みかつより狭い機械区間を定める．

$X=[x_1, x_2], Y=[y_1, y_2]$ とするとき

$$X+Y=[x_1+y_1, x_2+y_2]$$
$$X-Y=[x_1-y_2, x_2-y_1]$$
$$X*Y=[\min\{x_1\cdot y_1, x_1\cdot y_2, x_2\cdot y_1, x_2\cdot y_2,\},$$
$$\max\{x_1\cdot y_1, x_1\cdot y_2, x_2\cdot y_1, x_2\cdot y_2,\}]$$
$$X/Y=[\min\{x_1/y_1, x_1/y_2, x_2/y_1, x_2/y_2,\},$$
$$\max\{x_1/y_1, x_1/y_2, x_2/y_1, x_2/y_2,\}]$$
（ただし，$Y \ni 0$）

図-3 区間演算における四則演算の計算法

6. 数値実験

5. の実現法（データ型とライブラリ）を用いれば，式(4.3)を計算できるが，まず，すでに報告されている数値実験結果[10]を紹介する．

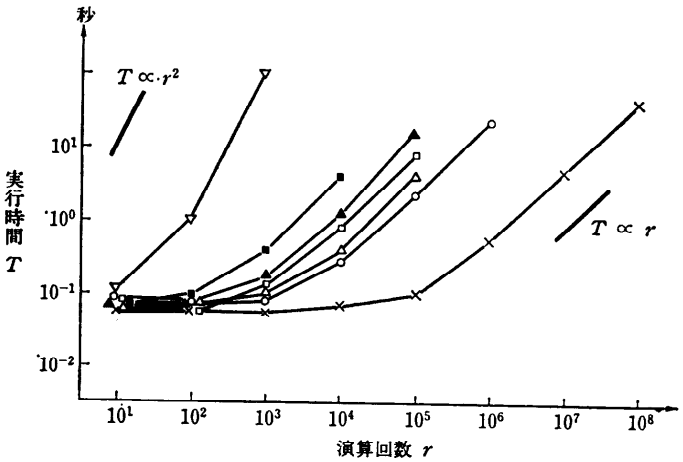
総演算回数が数十というような小規模な関数では，単なる機械区間演算の結果 Y の幅の 1/2，式(4.2)の A_ν，式(4.4)の A_Y のいずれもほぼ同じ大きさであり，それらが与える（無限精度演算結果の存在を保証する）区間の幅にはほとんど差がない．しかし，LU 分解によって 10 元線形方程式系を解く際には総演算回数が 10^3 回近くになり，その場合には保証区間幅の比（(Y の幅)/(2·A_Y)）が 10^4〜10^8 にも達し，A_Y は Y に比べて 10 進で 4 桁から 8 桁も高精度の保証区間を与える場合がある．

また，機械区間演算に関しては，山本ら[15]によると，機械区間演算の倍精度加算は，単精度・倍精度浮動小数点演算の加算時間の 30 倍程度で実現できるという．

6.1 予備実験

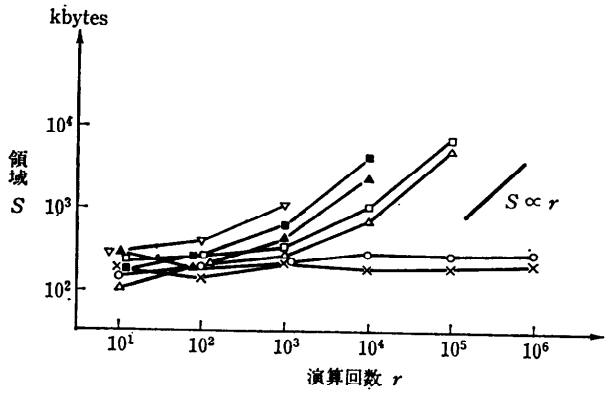
予備実験として，5. における実現法で(1),…,(5)の各種演算を加算，乗算で実行し，倍精度浮動小数点演算に比べてどの程度速い(遅い)のかを調べた．使用計算機は SUN Microsystems 社の Sparc Station 1*，プログラム言語は GNU C++，最適化(を試みる)オプションを指定した．

実験では，乗算を 10^1, 10^2, 10^3, …回繰り返して実行したときの CPU 時間（ユーザ使用時間とシステム使用時間の和）と領域とを測定した．結果を図-4 に示す（加算についてもほぼ同様の結果が得られている）．この結果からは，(1)の倍精度演算は約 200 万回/秒，(2)の機械区間演算は約 4 万回/秒，(3)の計算グラフ作成を演算と同時に行うと約 2 万回/秒，(4)では約 1 万回/秒であることが分かる．したがって，約 200 倍の時間とそれに見合う領域を費やせば，式(4.3)の A_Y の計算準備が整えられることが分かった．A_ν, A_Y を計算するには，さらに，高速自動微分法を実行するための時間（そ



(a)　乗算の実行時間

×(1)　浮動小数点演算
○(2)　機械区間演算
△(3)　浮動小数点演算の履歴を残す演算
▲(3′)　(3)かつ最後に式(4.2)の A_ν を計算する
□(4)　(2)かつ履歴を残す演算
■(4′)　(4)かつ最後に式(4.3)の A_Y を計算する
▽(5)　(4)かつ毎回式(4.3)相当を計算する



(b)　乗算のための領域

×(1)　浮動小数点演算
○(2)　機械区間演算
△(3)　浮動小数点演算の履歴を残す演算
▲(3′)　(3)かつ最後に(4.2)式の A_ν を計算する
□(4)　(2)かつ履歴を残す演算
■(4′)　(4)かつ最後に(4.3)式の A_Y を計算する
▽(5)　(4)かつ毎回式(4.3)相当を計算する

図-4

* メモリ 12 Mbyte，スワップ領域 28 Mbyte.

れまでにかかった時間の高々定数倍）を必要とする（図-4 中の(3′)および(4′)）．今回の実験では，最終的に保証区間を得るためには，1 演算あたり約 1/2,500 秒すなわち浮動小数点演算の約 800 倍の時間が必要であったことになる．領域については，履歴を残す場合（(3)，(3′)，(4)，(4′)，(5)）には計算時間に比例する領域が必要であるが，ここでの実現法では，1 演算あたり 60 バイト弱であった．

保証区間幅に関しては，単純計算を繰り返しているだけで発生誤差の打ち消しなどが起こらないので，どれもほとんど差がない．

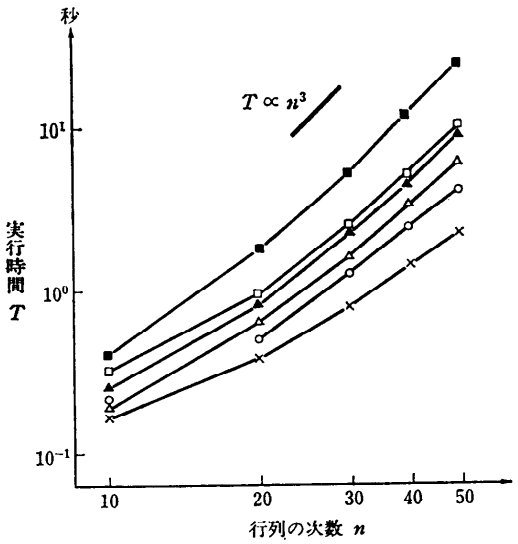
次節で示す実験結果と異なり倍精度演算が他のものに比べて速いのは，ここで用いたプログラムが，レジスタ宣言が有効に働いているなど，倍精度演算の実行に有利になっているからである．（もちろん，倍精度演算にベクトルプロセッサなどを使用できる状況では倍精度演算が有利になり，実行時間の差がさらに開く可能性もあるが，比較の基準点が異なるのでここでは言及しない．）

6.2 線形方程式系の解に含まれる誤差

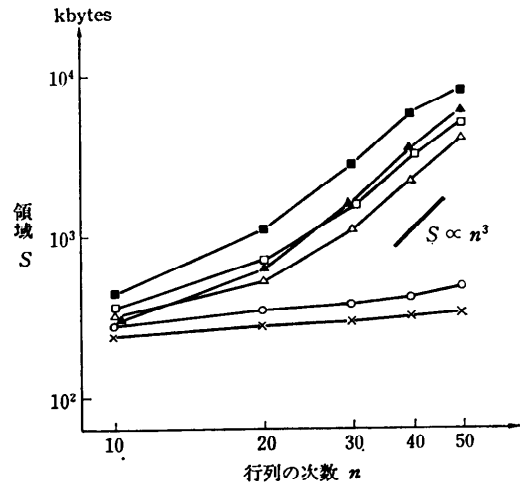
LU 分解を用いる線形方程式系の解に関する誤差評価について，代償として支払うべき時間，領域について調べた．使用計算機などは 6.1 と同じである．

n 次正方行列 A と n 次元ベクトル $\boldsymbol{b}$ に関する線形方程式系 $A\boldsymbol{x}=\boldsymbol{b}$ の解 $\boldsymbol{x}$（の第一成分）に含まれる誤差を評価する．解 $\boldsymbol{x}$ は，$\boldsymbol{f}(A, \boldsymbol{b}) \overset{\text{def}}{=} A^{-1}\boldsymbol{b}$ で定義される（合成）関数 $\boldsymbol{f}$ の計算値である．$\boldsymbol{f}$ を計算する手続きとしては，ここでは部分枢軸選択付き LU 分解法を採用する．成分が $[-1, 1]$ 上の一様分布に従う行列を A，ベクトルを $\boldsymbol{b}$ とし，それらの次数を 10, 20, 30, 40, 50 と変えて 5. の (1), …, (4) の演算を実行した．実行時間・領域を図-5 に示す（評価値(4.2)，(4.4)の計算時間も含めたものを (3′)，(4′) と表す）．実行時間については，単純な倍精度演算とその他との差が予備実験のときほどは開かず，倍精度演算による実行時間の 10 倍程度の時間で精度保証ができたことが分かる．また，このとき，1 演算あたり 60 バイト弱の領域を使用したことも分かる．

保証区間幅について，結果の一部を表-1 に示す．20次の乱数行列に関しては，倍精度浮動小数点演算による計算値約 −0.236 に対して，機械区間演算では 10 進数で上位 6 桁，式(4.4)の Ay では上位 11 桁を保証していることが分かる．50次の乱数行列に関しては，計算値約 1.47 に対して，機械区間演算では 10^7 の幅



部分枢軸選択付 LU 分解による線形方程式系の解に必要な時間
×(1) 浮動小数点演算
○(2) 機械区間演算
△(3) 浮動小数点演算の履歴を残す演算
▲(3′) (3)かつ最後に式(4.2)の A_y を計算する
□(4) (2)かつ履歴を残す演算
■(4′) (4)かつ最後に式(4.3)の Ar を計算する



部分枢軸選択付 LU 分解による線形方程式系の解に必要な領域
×(1) 浮動小数点演算
○(2) 機械区間演算
△(3) 浮動小数点演算の履歴を残す演算
▲(3′) (3)かつ最後に式(4.2)の A_y を計算する
□(4) (2)かつ履歴を残す演算
■(4′) (4)かつ最後に式(4.3)の Ar を計算する

図-5

の保証区間が得られるだけで，ほとんど無意味になってしまっているが，A_Y では約 ±0.8 の保証区間を得ることができた．しかし，式(4.2)の線形近似による誤差評価値 A_y に比べて，A_Y が極端に大きくなってしまっている．その理由は，上の計算において，倍精度計算を基礎とする機械区間演算を用いたのでは式(4.3)の中の W_j が0を含む，すなわち，符号が確定しないという状況が発生してしまって，A_Y が A_y に近くなるという最適性を理論的に保証するための前提である $\partial f/\partial v_j \neq 0$ が成り立ちにくくなっているからである．基礎となる演算の精度を上げていくと，A_Y は A_y に漸近するが，機械区間演算の幅は A_y には近付かないというのが，A_Y の最適性の意味である[10]．

表-1 保証区間幅の比較

$Ax=b$ を枢軸選択付 LU 分解で解いたときの解 x の第1成分の値とその保証区間幅の一例．ただし．A, b は，その成分が [−1, 1] 上の一様分布に従う乱数行列，乱数ベクトルである．

	20 次	50 次
倍精度計算値	$-2.36437520805967\times10^{-1}$	$1.476932775059060\times10^{0}$
$2\cdot A_y$	2.63446×10^{-13}	7.65756×10^{-11}
機械区間演算結果 Y の幅	7.21629×10^{-7}	4.87982×10^{7}
$2\cdot A_Y$	2.63448×10^{-13}	1.51378×10^{0}

7. おわりに

機械区間演算と，高速自動微分法を用いた丸め誤差評価の方法を組み合わせることによって，単純に機械区間演算を実行した結果に比べて（無限精度演算結果の存在を保証する）区間の幅を狭めうることを示した．この方法は，小規模な問題に対してはあまり効果がないが，線形方程式系の解について調べた結果からみられるように，大規模な問題に対してはその効果がきわめて大きい（区間幅が狭まる）場合もある．

実験的に示した時間・領域の代償が大きいかどうかの評価は，状況により異なるであろう．ここで示したのは，数値計算結果に厳密な意味での精度保証が不可欠になった場合には，800 倍程度の時間とそれに見合った領域を費やせば，そのような精度保証が可能であるということである．

今回の実現法では，精度保証の代償として，誤差評価値計算まで含めて浮動小数点四則演算の 800 倍程度の時間を支払うことになったが，機械区間演算の実現法の改良，履歴を残す演算実現法の最適化などを図ることにより，少なくともスカラ計算機上では，より小さな代償で済ませることができると期待される．

なお，1. でも触れたとおり，厳密性に固執せずに実用的な立場からすると，誤差が小さいときには $A_Y/A_y \simeq 1$ であることが実験的に確かめられているので，絶対評価 A_y によって精度保証をしてもよい．（あるいは，式(4.5)の P_y でもおよその精度を知ることができる．）厳密性を追求するならばここで述べた方法により，反復改良でなく直接的に，精度保証ができるということである．

参考文献

1) Alefeld., G. and Herzberger, J.: Introduction to Interval Computations, Academic Press, New York (1983).
2) Bohlender, G., Ullrich C., Gudenberg, J. W. and Rall, L. B.: Pascal-SC—A Computer Language for Scientific Computation, Academic Press, Orlando (1987).
3) Griewank, A.: On Automatic Differentiation, Iri, M. and Tanabe, K. (eds.), Mathematical Programming—Recent Developments and Applications, Kulwer Academic Publishers, pp. 83-107 (1989).
4) Iri, M.: Simultaneous Computation of Functions, Partial Derivatives and Estimates of Rounding Errors—Complexity and Practicality, Japan Journal of Applied Mathematics, Vol. 1, No. 2, pp. 223-252 (1984).
5) Iri, M. and Kubota, K.: Methods of Fast Automatic Differentiation and Applications, Research Memorandum RMI 87-02, Department of Mathematical Engineering and Instrumentation Physics, University of Tokyo (1987).
6) 伊理正夫，土谷 隆，星 守：偏導関数計算と丸め誤差推定の自動化の大規模非線形方程式系への応用，情報処理，Vol. 26, No. 11, pp. 1411-1420 (1985).
7) Iri, M., Tsuchiya, T. and Hoshi, M.: Automatic Computation of Partial Derivatives and Rounding Error Estimates with Applications to Large-scale Systems of Nonlinear Equations, Journal of Computational and Applied Mathematics, Vol. 24, pp. 365-392 (1988).
8) Kim, K. V., Nesterov, Yu. E. and Cherkassky, B. V.: An Algorithm for Fast Differentiation and Its Applications, Abstracts of the 12th IFIP Conference on System Modelling and Optimization (September 2-6, 1985, at Buda-

pest, Hungary), pp. 181-182 (1985).
9) 久保田光一，伊理正夫：高速自動微分法の定式化と計算複雑度の解析，情報処理学会論文誌，Vol. 29, No. 6, pp. 551-560 (1988).
10) 久保田光一，伊理正夫：高速自動微分法と区間解析とを用いた丸め誤差推定，情報処理学会論文誌，Vol. 30, No. 7, pp. 807-815 (1989).
11) Linnainmaa, S.: Taylor Expansion of the Accumulated Rounding Error, BIT, No. 16, pp. 146-160 (1976).
12) Matiyasevich, Yu. V.: Veshchestvennye Chisla i ÈVM, Kibernetika i Vychislitel'naya Tekhnika, Vypusk 2, pp. 104-133 (1986).
13) Miller, W. and Wrathall, C.: Software for Round-off Analysis of Matrix Algorithms, Academic Press, New York (1980).
14) Rall, L. B.: Validation of Numerical Computation, 数理解析研究所講究録673「自己検証的算法とその応用」，京都大学数理解析研究所，pp. 1-16 (1988年11月).
15) 山本哲朗，陳 小君，菅野幸夫：精度保証付き数値計算について，情報処理学会研究報告，88-NA-27-1 (情報研報 Vol. 88, No. 91), pp. 1-11 (1988).
16) Wilkinson, J. H.: Rounding Errors in Algebraic Processes, Prentice-Hall Inc., Englewood Cliffs, New Jersey (1963).

(平成2年4月11日受付)