6 数値シミュレーションを支える精度保証技術

大石 進一[1]　荻田 武史[1]

[1] 早稲田大学理工学術院／科学技術振興機構 戦略的創造研究推進事業

数値シミュレーションは，膨大な数値計算の結果の上に成り立っている．そのような数値計算の結果を数学的に厳密な誤差限界とともに与えることを精度保証付き数値計算という．近年の筆者らの成果により，精度保証付き数値計算は，線形計算については近似解を求めるのと比べて数倍の手間で実行できることが多いことが示された．さらに，数値計算自身も必要な精度の計算を適応的にかつ必要最小限に近い計算の手間で行えることが示された．たとえば，連立一次方程式の数値解を倍精度浮動小数点演算で求めても，その精度が10進数換算で16桁（倍精度での最大）はほぼ常に出せる計算方式が実現できる．しかも，問題の難しさ（条件数）に応じた計算量で高速に実行できる．このように，数値線形代数の問題については，精度保証付き数値計算の技術は多くの場合，数値シミュレーションを支える技術として，実に驚くべき進展を遂げた．本稿では，このような現状について概観する．具体的には，筆者らの研究により確立しつつある高速精度保証付き数値計算技術と無誤差数値計算技術についてサーベイを行う．

はじめに

数値計算とは，解析的に解くのが困難な問題を数値的に解く計算やその手法のことであり，その実装にはコンピュータによる浮動小数点演算を用いるのが普通である．浮動小数点演算とは，簡単に言えば，有限桁（たとえば，10進数で16桁程度）の計算のことである．すなわち，何かの計算をするたびに，決められた桁数以下の部分は打ち切られる．したがって，有限桁の計算を重ねると次第に計算誤差が蓄積していくため，最終的に得られた結果がどれくらい正しいかは問題依存であり，正しい結果とはまったく違った数値計算結果が得られることもしばしばある．これに対し，数値計算による結果を数学的に厳密な誤差限界とともに与えることを精度保証付き数値計算という．

一方，物理現象などをモデル化して現れる方程式を数値計算によって解き，その現象のシミュレーションを行うことを数値シミュレーションという．数値シミュレーションは，コンピュータの発展とともにさまざまな分野でその重要性を増し，実験とともに理論を検証するための非常に重要な位置を占めるに至っている．したがって，その結果の精度がどのくらいあるかを示すことが非常に重要であり，精度保証付き数値計算は，その有用なツールとなりつつある．

筆者らは，精度保証付き数値計算としてどのようなものが望ましいかという条件は次のようであると考えている：

- 数値計算の応用分野の研究者が数値シミュレーションを行うために普段用いている計算機とソフトウェア(パッケージ)を変更することなく，精度保証ができるようにする．
- 精度保証のための特別のパッケージを使う必要がないようにする．
- 数値線形計算用のソフトウェアの最適化の効果を失わないように，精度保証をする．
- 問題の条件数(条件数が高いと難しい問題である)に応じて，簡単な問題は高速に，難しい問題は必要なだけの手間で，適応的に解く．
- 簡単な問題に対しては，数値解を得るための時間の数倍の計算時間で，精度保証できるようにする．

近年の筆者らの成果により，線形問題については上記の条件を満たす精度保証法が非常に多くの場合に得られることが示された．このような方法が開発できたのは，筆者らの研究により得られた次の2つのブレークスルー

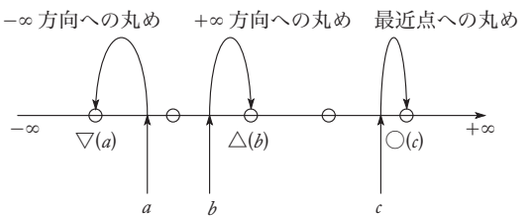

図 -1　丸めモードのイメージ．数直線上の○は離散的に分布している浮動小数点数を表している．

によっている：

(1) 高速精度保証付き数値計算技術

(2) 無誤差数値計算技術

これらについて本稿ではやさしく解説することを目的とする．

高速精度保証付き数値計算

精度保証付き数値計算は，通常のシミュレーションの問題の規模に対応できないことや対応できたとしても計算時間が数万倍にもなり，実用的でないと思われてきた．これは，たとえばガウスの消去法の過程をそのまま逐一区間演算に置き換えて得られる「区間ガウスの消去法」などが用いられてきたことによる．これに対して筆者らは，数値解を得るためのアルゴリズムと精度保証のためのアルゴリズムを独立のものとし，精度保証には異なったアルゴリズムを用いることを考えた．これにより，問題の規模が大きくても，精度保証をすることができるようになった．また，区間演算を基本演算と考えるのではなく，それを行列の積単位で考えることにより，高速精度保証法が得られることを筆者らは示した．本章ではその方法について説明を加える．

IEEE 754 浮動小数点演算規格

精度保証付き数値計算を高速化できた大きな理由の1つは，IEEE 754 浮動小数点演算規格[1]が区間演算を意識して，数学的にも十分吟味して制定されていることである．本規格は，カリフォルニア大学バークレー校の William Kahan を中心として 1985 年に制定された．

IEEE 754 規格は 2 進浮動小数点演算の規格で，単精度（32 ビット）や倍精度（64 ビット）のフォーマットや演算のルール等を定義している．たとえば倍精度であれば，符号（1 ビット），指数部（11 ビット），仮数部（52 ビット）のように決められている．浮動小数点数は正規化することを前提に考えるので，仮数部は実際には 53 ビット分の有効桁を確保できる．したがって，倍精度の演算精度は，$2^{-53} \approx 1.11 \times 10^{-16}$ となるため，10 進数でおよそ 16 桁程度である．規格の詳細は文献に任せるが，IEEE 754 では，浮動小数点数の四則演算を数学的に semimorphism と呼ばれる性質によって定義している．これはまず実数の浮動小数点数への丸めを定義し，各丸めごとに浮動小数点数の四則演算を定義する．IEEE 754 では丸めモードとして

- 最近点への丸め
- $+\infty$ 方向への丸め
- $-\infty$ 方向への丸め
- 原点方向への丸め

の 4 種類がある．今，$\mathbb{F}$ を考えている浮動小数点数の集合，$\mathbb{R}$ を実数の集合とする．このとき，$r \in \mathbb{R}$ に対し，それぞれの丸めモードは以下のように定義される．

最近点への丸め　$\bigcirc : \mathbb{R} \to \mathbb{F}$ は，$\bigcirc(r)$ を r に最も近い $f \in \mathbb{F}$ とする．

$+\infty$ 方向への丸め　$\triangle : \mathbb{R} \to \mathbb{F}$ は，$\triangle(r)$ を $f \geqq r$ を満たす $f \in \mathbb{F}$ で最も小さいものとする．

$-\infty$ 方向への丸め　$\triangledown : \mathbb{R} \to \mathbb{F}$ は，$\triangledown(r)$ を $f \leqq r$ を満たす $f \in \mathbb{F}$ で最も大きいものとする．

原点方向への丸めについては省略する．丸めモードのイメージは図 -1 のようになる．丸めモードを指定したとき，その下での浮動小数点数の四則演算は semimorphism によって定義される．すなわち，たとえば，$f, g \in \mathbb{F}$ に対して，最近点への丸めモードでの加算 $\oplus$ は実数上の加算 $+$ を使って

$$f \oplus g = \bigcirc(f + g)$$

が満たされるように定義される．減算や乗除算についても同様である．

機械区間演算

さて，精度保証の基礎となる考え方の区間解析は，実数を両端が計算機で表現できる区間で置き換える方法である．たとえば，円周率 π は浮動小数点数で厳密に表すことはできないが，$[3.14, 3.15]$ によって円周率の包含を表すことはできる．このときの精度は両端の数の差で与えられる．区間演算は，四則演算に対応する 2 つの

区間の演算である．2つの区間から任意に実数を選んだとき，その演算結果を含む最小の区間として定義される．これを計算機上に実装したのが機械区間演算であり，これは下端の数をさらに $-\infty$ 方向に丸め，上端を $+\infty$ 方向に丸めたものである．たとえば，機械区間演算の和は次のように定義される：2つの区間 $a = [a_l, a_u]$，$b = [b_l, b_u]$ $(a_l, a_u, b_l, b_u \in \mathbb{F})$ に対して

$$a + b := [\nabla(a_l + b_l), \Delta(a_u + b_u)] = [c_l, c_u]$$

と定義される．これを擬似コードの形で書くと次のようになる．擬似コード中の % は，それ以降が行末までコメントであることを意味する．

```
down();          % −∞ 方向への丸めモード
c_l = a_l + b_l; % a + b の下限を計算
up();            % +∞ 方向への丸めモード
c_u = a_u + b_u; % a + b の上限を計算
```

ただし，down() と up() はそれぞれ，CPU の丸めモードを $-\infty$ 方向と $+\infty$ 方向への丸めモードへ変更する命令である．

区間解析の1つの手法は，たとえば，連立一次方程式に対するガウスの消去法において，実数を区間に，四則演算を区間演算に置き換える方法である．これを区間ガウスの消去法という．区間ガウスの消去法によって得られた区間ベクトルは厳密解を含む区間となる．したがって，精度保証付きの連立一次方程式の解法となる．しかし，この方法の計算時間については，丸めモードの変更によって通常の線形計算用パッケージが使えず，そのため最適化が崩れることや，通常，演算子多重定義が用いられることから，区間ガウスの消去法は通常のガウスの消去法の 1,000 倍から1万倍も遅くなる．さらに，計算途中で分母にくる区間が過大評価のためにゼロを含みやすくなり，高々，数百次元の連立一次方程式にしか適用できない．区間解析の基礎については，たとえば文献 5) を参照されたい．

行列区間演算

このような困難を解消するために，演算を行列の積単位で考え，それを精度保証付きで計算する方法を筆者らは提案した[5]．これは $A, B \in \mathbb{F}^{n\times n}$ に対して，次のような非常に簡単な擬似コードで実行できる．

```
down();
L = A * B;   % A * B の下限を計算
up();
U = A * B;   % A * B の上限を計算
```

この計算を行うと

$$L \leqq A * B \leqq U$$

となることが分かる．ただし，不等号は，すべての要素に対して大小関係が成立していることを意味する．これを行列積の高速精度保証アルゴリズムという．

さて，ノルムを用いると，ベクトルや行列の要素の大きさを1つの値（スカラ）で表現できる．たとえば，$\mathbb{R}$ を実数の集合としたとき，$x = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$，$A = (a_{ij}) \in \mathbb{R}^{m\times n}$ に対して，その最大値ノルム $\|x\|_\infty$ および $\|A\|_\infty$ はそれぞれ

$$\|x\|_\infty = \max_{1\leqq i\leqq n} |x_i|$$

および

$$\|A\|_\infty = \max_{1\leqq i\leqq m} \sum_{j=1}^{n} |a_{ij}|$$

となる．したがって，$B = (b_{ij}) \in \mathbb{F}^{n\times n}$ に対して，$\|B\|_\infty$ の上限は次の擬似コードで計算できる．

```
up();
s = max(1≦i≦n) Σ(j=1..n) |b_ij|;
```

すなわち，s が B の最大値ノルムの上限を与える．

ここで，連立一次方程式の数値解の精度保証に有用な次の定理を述べる．

定理 1　連立一次方程式

$$Ax = b \tag{1}$$

を考える．ただし，$A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$ とする．$\tilde{x}$ を $Ax = b$ の近似解，R を A の逆行列の近似，$G = RA - I$ とする．ただし，I は $n \times n$ 単位行列である．もし，$\|G\| < 1$ なら，A^{-1} が存在し，真の解 $x^* = A^{-1}b$ に対して

$$\|x^* - \tilde{x}\| \leqq \frac{\|R\| \cdot \|A\tilde{x} - b\|}{1 - \|G\|} \tag{2}$$

が成り立つ．

この定理でのノルムはベクトルのノルムなら何でもよいが，簡単のため，以下では最大値ノルムであるとする．

式 (2) の右辺に現れる，$\|R\|, \|A\tilde{x} - b\|, \|G\|$ の上限は，いずれも前述の行列積の高速精度保証アルゴリズムや最大値ノルムの上限の精度保証アルゴリズムで計算できる（詳細については，文献 5）などを参照されたい）．この方式を用いると，ガウスの消去法を用いた近似解の計算に対し，その9倍の計算量で精度保証付きの近似解を計算することができる．なお，実際の計算時間においても，近似解を得るのに必要な時間の数倍程度で精度保証できるので，このような精度保証アルゴリズムは高速精度保証と呼ばれるようになった．

これが基本であり，これにさまざまな工夫を施すことによって，やさしい問題に対しては高速で，難しい問題に対してはそれに応じた速度の精度保証アルゴリズムが作れることを以下で見ていこう．

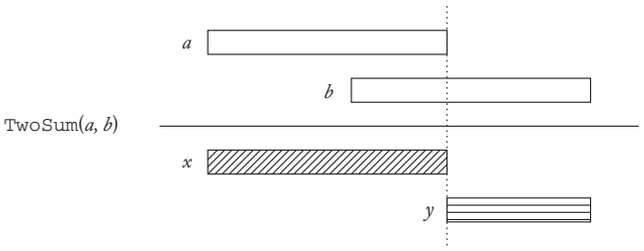

図 -2 **TwoSum** のイメージ

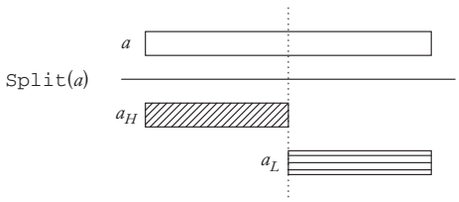

図 -3 **Split** のイメージ

ベクトルの総和の無誤差変換

数値計算では，問題の難しさを条件数という指標で計ることがある．いろいろな条件数の問題を解けるようにするためには，工夫がいる．そのために，ここで，筆者らの高速かつ任意に精度を指定できる内積計算法を紹介する．

1960 年代の初頭より，以下のような非常に注目すべき浮動小数点演算の性質が知られている．議論を簡単にするために IEEE 754 に従う 2 進浮動小数点システムを考える．指数部と仮数部がある程度以上のビット数を持つ浮動小数点数でも同様に成立する．ただし，最近点への丸めモードでの計算とする．考えている浮動小数点数の集合を $\mathbb{F}$，浮動小数点演算の相対精度を $\mathbf{u}$ とする．単精度では，$\mathbf{u} = 2^{-24} \approx 5.96 \times 10^{-8}$，倍精度では，$\mathbf{u} = 2^{-53} \approx 1.11 \times 10^{-16}$ である．

1969 年，Donald E. Knuth は次の事実を示した [3]．$a, b \in \mathbb{F}$ に対して，アルゴリズム TwoSum を次のように定義する：

function $[x, y] = \mathtt{TwoSum}\ (a, b)$
$x = a \oplus b;$
$c = x \ominus a;$
$y = (a \ominus (x \ominus c)) \oplus (b \ominus c);$

ただし，$\oplus, \ominus$ はそれぞれ，(最近点への丸めモードにおける)浮動小数点演算の加算と減算である．このとき

$$a + b = x + y \quad (\mathbf{u}\,|x| \geqq |y|) \tag{3}$$

が成立する．この式は，x は $a + b$ の浮動小数点演算における加算の結果であるため誤差を持つが，その誤差 $y = a + b - (a \oplus b)$ もまた浮動小数点数になることを意味しており，それはアルゴリズム TwoSum を用いると浮動小数点演算のみで計算できる．これを Knuth の定理という．TwoSum のイメージを図 -2 に示す．

1971 年に，乗算に対しても同様な性質が成り立つことを Theodorus J. Dekker が示した [2]．これを説明するためにまず，Dekker による次のアルゴリズムを示す．ただし，$a \in \mathbb{F}$ を仮数部が t ビットの浮動小数点数，$\otimes$ を浮動小数点演算における最近点への丸めのモードでの掛け算とする．

function $[a_H, a_L] = \mathtt{Split}(a)$
$c = (2^{\lceil \frac{t}{2} \rceil} + 1) \otimes a;$
$a_H = c \ominus (c \ominus a);$
$a_L = a \ominus a_H;$

ここで，a_H は a の上位半分のビット，a_L は a の下位半分のビットに対応するものである（ただし，必ずしもビットパターンが一致するわけではない）．このとき

$$a = a_H + a_L \quad (|a_H| \geqq |a_L|) \tag{4}$$

が成立する．この式は，a を a_H と a_L の和に誤差なしで変換できることを意味している．Split のイメージを図 -3 に示す．

これを用いて，Dekker は次のことを示した．まず，アルゴリズム TwoProduct を定義する（1968 年に，Gerhard W. Veltkamp が同様のアルゴリズムを示している）：

function $[x, y] = \mathtt{TwoProduct}(a, b)$
$x = a \otimes b;$
$[a_H, a_L] = \mathtt{Split}(a);$ % $a_H + a_L \leftarrow a$
$[b_H, b_L] = \mathtt{Split}(b);$ % $b_H + b_L \leftarrow b$
$y = a_L \otimes b_L \ominus (((x \ominus a_H \otimes b_H) \ominus a_L \otimes b_H) \ominus a_H \otimes b_L);$

このとき

$$a \times b = x + y \quad (\mathbf{u}\,|x| \geqq |y|) \tag{5}$$

が成立する．この式は，x は $a \times b$ の浮動小数点演算における乗算の結果であるため誤差を持つが，その誤差 $y = a \times b - a \otimes b$ もまた浮動小数点数になることを意味しており，それはアルゴリズム TwoProduct を用いると浮動小数点演算のみで計算できる．これを Veltkamp-Dekker の定理という．

Knuth の定理や Veltkamp-Dekker の定理によって，2 つの浮動小数点数 a, b の加算や乗算は，その近似 x とそれに対する誤差 y という 2 つの浮動小数点数の和に誤差なく変換することができることが分かった．このような変換を，無誤差変換（Error-free Transformation [4]）という．この無誤差変換を用いると，代数的な計算を浮動小

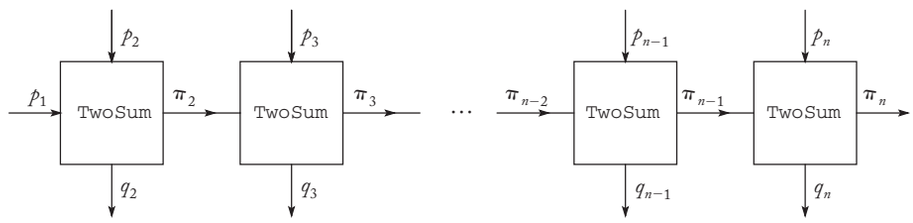

図 -4　ベクトルの総和の無誤差変換アルゴリズム **VecSum**

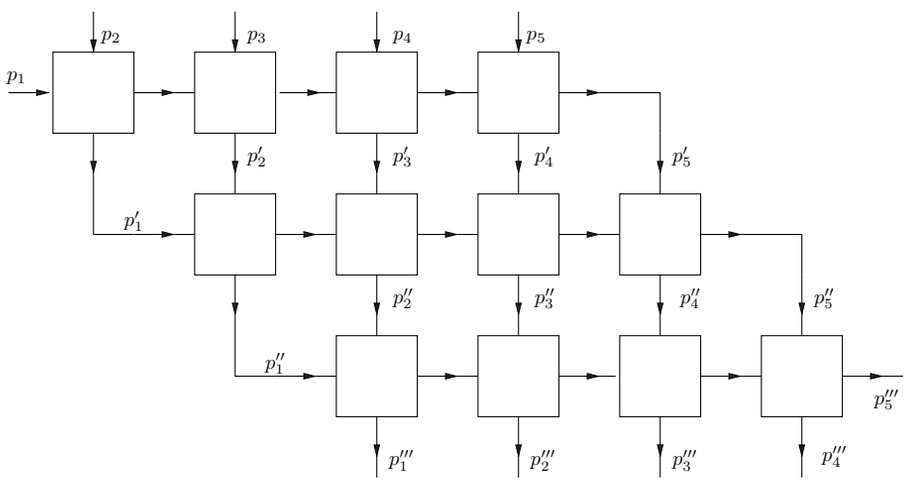

図 -5　**VecSum** の連続的な適用

数点演算に置き換えただけではできなかった高精度な計算を，浮動小数点演算のみを用いて実現することが可能となる．以下では，その一端に触れてみよう．

高精度総和計算

次に，浮動小数点数を要素とするベクトルの総和に対する無誤差変換について述べる．ここで，$p \in \mathbb{F}^n$ に対して

function $p' = \mathrm{VecSum}(p)$
$\pi_1 = p_1$;
for $i = 2 : n$
$[\pi_i, q_{i-1}] = \mathrm{TwoSum}(\pi_{i-1}, p_i)$;
end
$p' = (q_1, q_2, \ldots, q_{n-1}, \pi_n)^T$;

というアルゴリズムで $p' \in \mathbb{F}^n$ を計算したとする（**図 -4** を参照）．このとき

$$\sum_{i=1}^{n} p'_i = \pi_n + \sum_{i=1}^{n-1} q_i = \sum_{i=1}^{n} p_i$$

が成り立つ．この式は，ベクトル p の総和が同じ長さのベクトル p' の総和に誤差なく変換されていることを意味する．ただし，π_n は $\sum_{i=1}^{n} p_i$ の近似，$\sum_{i=1}^{n-1} q_i$ は加算の丸め誤差を集めたものと解釈することができる．

また，VecSum は繰り返し適用することができる（**図 -5** を参照）．これらは総和の無誤差変換であるから

$$\sum_{i=1}^{n} p_i = \sum_{i=1}^{n} p'_i = \sum_{i=1}^{n} p''_i = \cdots$$

が成立することも分かる．

筆者らは次のきわめて重要な性質を発見した[4]．まず，ベクトルの総和の条件数を

$$\mathrm{cond}\left(\sum p_i\right) := \frac{\sum |p_i|}{\left|\sum p_i\right|}$$

と定義する．これは，総和の計算の中で，どれだけ桁落ち（有効数字の減少）が発生するかを表す指標となる．たとえば，通常の浮動小数点演算で総和を再帰的に求めるアルゴリズム

function res = Sum(p)
res = 0;
for $i = 1 : n$
res = res $\oplus p_i$;
end

を用いると，その結果 res $\in \mathbb{F}$ の相対精度は

$$\frac{\left|\mathrm{res} - \sum p_i\right|}{\left|\sum p_i\right|} \leqq \mathcal{O}(\mathbf{u}) \cdot \mathrm{cond}\left(\sum p_i\right)$$

を満たす．ここで，$\mathcal{O}(\mathbf{u})$ は，$\mathbf{u}$ のオーダ（定数倍）であることを意味する．この式の意味を簡単に説明すると，たとえば条件数が 10^7 のとき，右辺はオーダを無視すれ

ば $\mathbf{u}\cdot 10^{7}\approx 10^{-9}$ となり，これは res が 9 桁程度は正しいことを示している．逆に言えば，条件数が $\mathbf{u}^{-1}$ よりも大きい場合は，その結果が 1 桁も正しくないことが推定される．

次に，アルゴリズム SumK を次のように定義する．

function res = SumK(p, K)
for $k = 1 : K-1$
$t = \mathrm{VecSum}(p)$;
end
$\mathrm{res} = t_n \oplus fl(\sum_{i=1}^{n-1} t_i)$;

ただし，$fl(\cdots)$ は括弧の中をすべて浮動小数点演算で実行することを意味する．このとき，SumK によって計算された結果 $\mathrm{res}\in\mathbb{F}$ の相対精度は

$$\frac{|\mathrm{res}-\sum p_i|}{|\sum p_i|}\leqq \mathbf{u}+\mathcal{O}(\mathbf{u}^{K})\cdot \mathrm{cond}(\textstyle\sum p_i)$$

を満たす．この式から，総和については特別に多倍長演算を用意しなくても，SumK を用いれば，通常の浮動小数点演算（各演算の相対精度が $\mathbf{u}$）と比べて，あたかもその K 倍の内部精度（各演算の相対精度が $\mathbf{u}^{K}$）で求めたのと同等の結果が得られることが分かる．すなわち，ユーザに今まで使っていなかった多倍長演算ライブラリを導入してもらわなくても，高精度な総和を計算速度の点でも高速に計算できることが分かる．

また，内積計算に関しては，$x, y\in\mathbb{F}^{n}$ に対して

$$[t_{n+i}, t_i] = \mathrm{TwoProduct}(x_i, y_i),\ i = 1,\ldots, n$$

とすると

$$\sum_{i=1}^{n} x_i y_i = \sum_{i=1}^{2n} t_i$$

となる．よって，内積計算は総和計算の問題に帰着することができるため，内積も高精度に計算できることが分かる．

筆者らは最近になって，さらに，結果の精度を指定すると，その精度までの正しい総和や内積を計算するアルゴリズムを開発した．そして，多くの場合，そのアルゴリズムはこれまでのどのアルゴリズムよりも高速であることを示している．これも，四則演算に対して最近点への丸めモードが実装されている浮動小数点数システムがあれば，その演算だけでアルゴリズム化できる．

こうして，現在では高精度の総和や内積をポータブルかつ非常に高速に計算する手法が確立された．これを利用して，新しい数値計算体系が構成できることを次に見てみよう．

高精度数値計算アルゴリズム

ここで，任意に悪条件な線形問題に対する新しい精度保証付き数値計算法を紹介する．$n\times n$ の正方行列 A を考える．A の条件数を

$$\mathbb{k}(A) := \|A\|\cdot\|A^{-1}\|$$

と定義する．倍精度演算では浮動小数点数の相対精度が $\mathbf{u}\approx 1.11\cdot 10^{-16}$ であることから，$\mathbb{k}(A)$ が 10^{16} より大きいときには，A の逆行列の近似 R を倍精度演算による普通のアルゴリズムで計算しても，通常 $\|RA-I\|>1$ となって精度保証ができなくなる．すなわち，R 自身は数値計算によって求められるが，丸め誤差の影響によって，R が A の逆行列としての情報はほとんど持たないことを示していると思われる．数値計算の破綻である．従来このような場合を検出するときは，使用した数値計算パッケージの警告メッセージに頼るしかなく，実際にはまったく不正確な結果を得ていたとしても，それを正しいと信じるか，多倍長精度演算をトライアンドエラーで用いることくらいしかできなかった．

しかしながら，実は R は A の前処理としての情報を依然として含んでいることを Siegfried M. Rump は 1980 年代に発見した（ただし，解析が不十分であったため，未発表であった）．この性質を利用して，$\|RA-I\|<1$ を満たすような R を高速かつ高精度に求める方法を筆者らは提案した [6]．

その基本的アイディアは，R を

$$R = R_1 + R_2 + \ldots + R_k = \sum_{i=1}^{k} R_i,\ R_i\in\mathbb{F}^{n\times n}$$

のように，要素が浮動小数点数である行列の和として表現することを考えたことにある．ただし

$$|R_i| \geqq 2\mathbf{u}|R_{i+1}|,\ i = 1, 2, \ldots, k-1$$

とする．行列に対する絶対値は，すべての要素に絶対値を取ったものとし，不等号は要素ごとにすべて成立していることを意味する．

次に，このような行列に対して，高精度な行列積を以下のように定義する：行列 $A=\sum_{i=1}^{p} A_i$, $B=\sum_{i=1}^{q} B_i$, $A_i, B_i\in\mathbb{F}^{n\times n}$ に対して $[AB]_j$ は，実数演算で AB を計算し，その正確な結果を指定の精度（j 個の浮動小数点行列の和）に丸めることを意味する．このような高精度な行列積は，前述の四則演算における semimorphism を，内積や行列積のレベルに拡張したものと解釈することができる．これを実現するためには，多倍長精度演算とはまったく違った思想に基づいて内積計算のアルゴリズムを設計しなければならない．

以上の議論に基づいて，アルゴリズムを以下のように定義する：

function $R^{(k)} = \mathbf{AccInv}(A, k)$
$R^{(1)} = \mathbf{inv}(A)$; % 近似逆行列(倍精度)
for $i = 2 : k$
$C = [R^{(i-1)}A]_1$; % 高精度，結果は倍精度
$T = \mathbf{inv}(C)$; % $T \approx C^{-1}$（倍精度）
$R^{(i)} = [TR^{(i-1)}]_i$; % R の更新(高精度)
end

このアルゴリズムの直感的なアイディアは次のようである．まず，$R^{(1)} := R_1$ を A の最初の前処理行列と考え，R_1A を高精度に計算し，結果を倍精度に丸めて

$$C \approx R_1A$$

のようにする．次に，C の(近似)逆行列を通常の倍精度演算で

$$T \approx C^{-1} \approx (R_1A)^{-1}$$

のように求める．そして，TR_1 を高精度に計算し，新たに求めた R_1, R_2 を次の前処理行列 $R^{(2)} := R_1 + R_2$ とすると，結果は

$$R^{(2)} = R_1 + R_2 \approx TR_1 \approx (R_1A)^{-1}R_1 = A^{-1}$$

となることが期待できる．同様に上記の操作を繰り返し，$R^{(k)} := R_1 + R_2 + \ldots + R_k = \sum_{i=1}^{k} R_i$ を求める．

このアルゴリズムの厳密な解析は非常に困難であることが知られているが，筆者らは部分的ながらその解析に成功している[6]．

アルゴリズムの性能を検証するため，実際に Matlab を用いて数値実験を行う．計算環境は CPU: Pentium 4 (2.53GHz), Matlab 7.0.1 (R14) である．係数行列 $A \in \mathbb{F}^{n\times n}$ は，任意の次元数 n と条件数 $\kappa(A)$ を指定できる Rump の行列を用いる．右辺ベクトルは $b = (1, 1, \ldots, 1)^T \in \mathbb{F}^n$ と決める．

まず，$n = 100$，$\kappa(A) \approx 10^{100}$ のとき，次の結果を得た：

```
k = 8
*** residual iteration with verification
loop = 1, max. rel. error = 6.921332e-006
loop = 2, max. rel. error = 4.789042e-011
loop = 3, max. rel. error = 4.264335e-016
```

実行時間は 8.1 秒であった．結果の中で，k は $R = R_1 + R_2 + \ldots + R_k$ が $\|RA - I\| < 1$ を満たしたときの浮動小数点行列の個数，loop は残差反復の回数，max. rel. error は，要素ごとに数値解を相対誤差の意味で精度保証した結果の最大値

$$\max_{1\leqq i\leqq n} \epsilon_i, \left|\frac{\tilde{x} - x^*}{x_i^*}\right| \leqq \epsilon_i$$

である．

次に，$n = 500$，$\kappa(A) \approx 10^{50}$ に対する結果は以下のようになった：

```
k = 5
*** residual iteration with verification
loop = 1, max. rel. error = 3.776758e-009
loop = 2, max. rel. error = 1.023496e-016
```

実行時間は 190 秒であった．

よって，任意に悪条件な問題であっても，精度保証付き数値計算と高精度計算を用いることによって，所望の精度を持つ数値解を得ることが可能となった．

おわりに

これまで述べてきたように，条件数が大きい問題の場合など，従来の数値解を求めるだけの計算では仮数部の一桁も合っていないような非常に悪い数値解が得られているに過ぎないことがある．もちろん，多くの場合に高度な専門知識を持つ研究者がその数値解を吟味して，物理シミュレーションとあまりに違う解を与えていることなどから，その数値解が擬似解であり，棄却すべきのものであると判断していたのであろう．しかし，問題の規模が大きくなればなるほど，また，最先端の研究であればあるほど，そのような目を逃れて紛れ込んでくる数値計算の幻影解を見ていることも多くあったと思われる．精度保証付き数値計算の発達により，(少なくとも数値線形代数の多くの問題は）今やそのような幻影解は精度保証によって検出できるようになり，所望の解を得ることができたかどうかを正確にかつ自動的に見分けがつく時代となった．

本稿で紹介してきた精度保証アルゴリズムは，数値解を得るための計算機環境をそのまま用いることができるようなポータブルなもので構成できる．さらに，数値計算によって得られた結果の精度が不足しているということが精度保証によって分かったとき，倍精度浮動小数点システムなど普段用いている浮動小数システムだけを用いたポータブルで適応的かつ高速な高精度総和計算アルゴリズムや高精度内積計算アルゴリズムにより，ほぼ必要最小限に近い計算の手間で，必要な精度を持つ数値解が精度保証付きで求められるようになりつつある（それを確立していくのが筆者らの最近の研究の重要なテーマである)．

このように，精度を非常に高速に保証できることを契機にして，必要な精度が出ていないときには適応的に必要な精度の解を求めるポータブルなアルゴリズムを作り出すような領域にまで精度保証付き数値計算は進歩しつつある．新しい数値計算学の誕生であるといえると思う．このような理論と技術が広く数値シミュレーションを含む理工学に応用されていくことを筆者らは確信している．

謝辞　本研究の多くは，科学技術振興機構（JST）戦略的創造研究推進事業（CREST）“数値線形シミュレーションの精度保証に関する研究”に補助を得て行われたものである．

参考文献

1) ANSI/IEEE Std 754-1985 : IEEE Standard for Binary Floating-Point Arithmetic, New York (1985).
2) Dekker, T. J. : A Floating-point Technique for Extending the Available Precision, Numer. Math., 18, pp.224-242 (1971).
3) Knuth, D. E. : The Art of Computer Programming : Seminumerical Algorithms, Vol.2, Addison-Wesley, Reading, Massachusetts (1969).
4) Ogita, T., Rump, S. M. and Oishi, S. : Accurate Sum and Dot Product, SIAM J. Sci. Comput., 26, pp.1955-1988 (2005).
5) 大石進一 : 精度保証付き数値計算，コロナ社 (2000).
6) Oishi, S., Tanabe, K., Ogita, T. and Rump, S. M. : Convergence of Rump's Method for Inverting Arbitrarily Ill-conditioned Matrices, J. Comput. Appl. Math., 205, pp.533-544 (2007).

（平成 19 年 7 月 5 日受付）

大石　進一　oishi@waseda.jp
1981 年早稲田大学大学院理工学研究科博士後期課程修了．1980 年から同大理工学部勤務．1989 年教授．2007 年 4 月から同大基幹理工学部応用数理学科所属．電子情報通信学会論文賞 3 回，同学会猪瀬賞，日本応用数理学会論文賞，船井情報科学振興賞，大川出版賞，電気通信普及財団賞等受賞．2004 年から科学技術振興機構（JST）戦略的創造研究推進事業（CREST）「線形数値シミュレーションの精度保証」研究代表者．2005 年から 5 年間文部科学省科学技術研究費特別推進研究「精度保証付き数値計算学の確立」研究代表者．現在，電子情報通信学会 基礎・境界ソサイエティ会長，日本応用数理学会，日本シミュレーション学会各理事．専門は精度保証付き数値計算．

荻田　武史　ogita@waseda.jp
2003 年早稲田大学大学院理工学研究科博士後期課程修了，同年から同大学客員講師を経て，2007 年 4 月から同大学理工学術院客員准教授．また，2005 年から科学技術振興機構研究員．2006 年日本応用数理学会論文賞受賞．専門は精度保証付き数値計算．