数値計算における誤差 — 浮動小数点演算と丸め誤差

静岡理工科大学 理工学部 情報システム学科
幸谷智紀
tkouya@cs.sist.ac.jp

1999 年 12 月 18 日 (土)

1 初めに

通常の解析学は，実数体 $\mathbb{R}$ とその拡大である複素数体 $\mathbb{C}$ における演算を基礎に成立している理論体系である。特に $\mathbb{R}$ の

1. 四則演算の完備性・連続性
2. 加減・乗除の交換法則・結合法則

という性質が重要である。

しかし，現在の数値計算では仮数部の桁数が有限である浮動小数点数を使用して行われる。そのために，演算結果を離散的な浮動小数点数に変換するための操作，所謂「丸め (round-off)」が行われる。この丸めによって生じる真の演算結果との「ずれ」を「丸め誤差 (round-off error)」と呼ぶ。この丸め誤差の発生によって，実数体の重要な性質の一つである結合法則が成立しなくなる。ために，今も昔も，１つの数値計算アルゴリズムの研究は，実数又は複素数体上で理論を構築し，計算機上でアルゴリズムを動作させ，そのアルゴリズムが丸め誤差を伴う計算においても十分実用に耐えうるかどうかを検証する，というのが一つのパターンになっている。

理論は実数体上で，しかし実際には丸め誤差を伴う浮動小数点数の集合上で演算を行わなければならない。丸め誤差による影響は，精度の問題のみならず，計算量にも影響を及ぼすことがある。それ故に，数値計算は実験工学として扱わねばらならい，というのが我々計算屋の共通認識である。

このあたりの事情を述べた箇所を森 [5] の本から引用する。

> このように計算機による数値計算の中で丸め誤差は様々な困難を生ずるが，一方これを避けて通るわけにはゆかない。(中略)
>
> しかし，一般に丸め誤差は小さく，また数値計算で用いられるアルゴリズムはほとんど例外なく実数を対象にした原理のうえに構成されている。従って，本書では主としてまず丸め誤差を除いて解析を行い，必要に応じて丸め誤差の影響を考慮に入れるという方法をとる。

森のこの書は，丸め誤差についての解析を随所で行っており，希有な入門書である。が，その解析をことごとく「この評価は過大すぎて実際には役に立たない」と否定してしまっている。「一般に丸め誤差は小さ」いためであるが，この辺りにも工学的な視点の必要性が見て取れる。

本文書は，数値計算における丸め誤差についての講演のアブストラクトである。実際の数値例については，講演時に示す。

2 歴史的経緯

数値計算における丸め誤差の影響を最初に調べたのは，J. von Neumann と H. H. Goldstine，そして A. M. Turing であると言われている [1]。計算機の発明に深く関わった von Neumann と Turing が，既に丸め誤差に関心を持っていたということは興味深い。

この仕事をほぼ完成させた一番の功労者は，J. H. Wilkinson(1919–1986) である。以下，Higham の本 [1] から，Wilkinson を紹介している部分を引用して紹介する。

> Wilkinson は，Turing 同様，ケンブリッジ大

学卒の数学者であり，英国物理学研究所 (National Physical Laboratory) で Turing の助手を務めていた。Turing の死後，Wilkinson は ACE(Automatic Computing Engine) のパイロット版を作るグループを率い，ACE のハードウェアとソフトウェアの設計と構築を行った。その後，作り上げた ACE を使い，様々な数値計算の手法を考案し研究した。1950 年代か 60 年代にかけて，後退誤差解析 (backward error analysis) という手法を編みだし，"Rounding Errors in Algebraic Processes"[9] と"The Algebraic Eigenvalue Problem"[8] という本を出版した。どちらも，直ちにその分野の古典と位置づけられることになった。(中略)　後者については，後の L. Fox 教授が「ほぼ間違いなく，数値解析において最も重要かつ広く読まれるべきものだ」と述べている。Wilkinson はまた，数学ソフトウェアの発展にも多大な貢献をした。"Handbook for Automatic Computation, Volume II: Linear Algebra"を Reinsch と共に編纂し，高品質かつ正確に文書化されたソフトウェアを収録した。この本は，その後の，NAG ライブラリ，EISPACK, LINPACK，LAPACK といったソフトウェアのプロジェクトに強い影響を与えることになった。

日本では，日本大学の宇野敏雄と富士通を中心とするグループと東京大学の森口繁一・伊理正夫を中心とするグループの活動が目立っていた。後者のグループの活動は今も「数値計算の精度保証」という一つの分野で引き続き行われている。

3　浮動小数点数と丸め方式

計算機で使用される浮動小数点数は，以下のような構成になっている。

$$\pm M \times \beta^E \tag{1}$$

ここで，M を仮数部 (mantissa)，E を指数部 (exponent)，β をこの浮動小数点数の基数 (base) と呼ぶ。M は常にある一定値 ($1 > M > \beta^{-1}$ か $\beta > M > 1$) 内に収まるよう正規化 (normalize) される[1]。正規化のために，指数部が調整されることになる。

かつて大型計算機が跳梁跋扈していた当時，浮動小数点数の規格は各社バラバラであったようである。IBM 互換機路線を取っていた富士通や日立の計算機では，16 を基数とした浮動小数点数を使用していたし，2 を基数としていた機種もあった。近年は，パソコンやワークステーションといった，浮動小数点演算回路も組み込んだ CPU を積んだ計算機が多く使用されるようになっている。この CPU の多くは IEEE754 規格に準拠した浮動小数点数 ($\beta = 2$) を使っている。この規格についての詳細は別に譲り [2]，ここでは丸め方式についてだけ触れることにする。

IEEE754 規格の丸めは四種類ある。

RZ モード　0 方向への丸め (切り捨て)

RM モード　$-\infty$ 方向への丸め (正数は切り捨て，負数は切り上げ)

RP モード　$+\infty$ 方向への丸め (正数は切り上げ，負数は切り捨て)

RN モード　0 捨 1 入，但し仮数部が偶数になるようにする

特に指定しない限り，通常の丸めは RN モードで行われる。モード変更を行うのは非常に簡単で計算量のコストもかからないので，うまく利用すると丸め誤差の評価が可能になる。同じ計算アルゴリズムでも，モード変更することによって丸め誤差の大きさが変化するため，計算結果に含まれる丸め誤差の桁数分だけ，数字が違ってくるからである。この事実を利用した研究としては，古くは山下 [10]，筆者 [4] のものがある。区間演算 (Interval Analysis) の観点から洗い直し，この事実に突き当たった研究としては大石 (早大) のものがある。

4　桁落ちによる丸め誤差の拡大と誤差のキャンセル

桁落ち (Loss of significant digits) は，数値計算において最も注意すべき現象として取り上げられる。

[1] underflow 近辺では正規化されない場合もある。

その割には素朴な 2 次方程式の解公式で説明されるだけで，他の数値計算アルゴリズムを例にきちんとした実例を示しているものは少ない。桁落ちは様々な場面で発生し，相対誤差を増大させる一番の原因となるものである。

この桁落ちは，精度の残っている上の桁が消失し，相対誤差が悪化する現象のことである。言い換えれば，桁落ちの起きる計算は，精度があるからこそ引き起こされるものであるため，これを避ける抜本的な改善策は，アルゴリズムを別のものにするか，仮数部の桁数を増やすしかない。

これとは別に，丸め誤差を縮小させる現象もある。これを**誤差のキャンセル**と呼ぶ [7]。これを，あくまで希な現象であると捕らえる向きが多いのはやむを得ないとは思うものの，誤差のキャンセルを考慮しないと説明のつかない事実があることも，強調しておきたい。

5 代数方程式

代数方程式

$$x^n + a_{n-1}x^{n-1} + \cdots + a_0 = 0 \tag{2}$$

が，全て実根 $\alpha_n, \alpha_{n-1}, \ldots \alpha_0$ を持つ場合を考える。

$$\Pi_{i=1}^{n}(x - \alpha_i) = 0 \tag{3}$$

このとき，各根の相対誤差は次のように評価できることが知られている。

$$\alpha_i\text{の相対誤差} \approx \Pi_{j=1,j\neq i}^{n} \left| \frac{\alpha_i - \alpha_j}{\max(|\alpha_i|, |\alpha_j|)} \right| \tag{4}$$

これは，各根が近接していると精度が悪化することを示している。

また，Newton 法を使いつつ，根を求める毎に代数方程式の次数を落としていこうとすると，

絶対値の大きい根から求める場合 低次の係数から順次求めていく

絶対値の小さい根から求める場合 高次の係数から順次求めていく

必要があることが知られている。

6 連立一次方程式

次のような連立一次方程式を考える。

$$Ax = b \tag{5}$$

ここで，A は n 次正方行列，b, x は n 次元ベクトルとする。

この方程式の A, b にそれぞれ誤差が混入した

$$(A + \Delta A)(x + \Delta x) = b + \Delta b \tag{6}$$

という方程式の解 $x + \Delta x$ の誤差 Δx は，次の評価式で与えられる [5]。

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\kappa(A)}{1 - \kappa(A)\frac{\|\Delta A\|}{\|A\|}} \left(\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|b\|} \right) \tag{7}$$

ここで $\kappa(A) = \|A^{-1}\|\|A\|$ を条件数と呼び，これが大きいとき，悪条件 (ill-conditioned) と呼ぶ。

以上は一般常識であるが，行列によってはこの評価式に当てはまらないこともある。悪条件である行列 A の例として，一頃はよく用いられた Hilbert 行列は，行列のサイズが大きくなるに従って条件数も増加する。しかし，実際にその条件数を計算してみると，ある一定以上のサイズになると理論値よりも小さい値で停留する。これは Hilbert 行列の要素に混入した丸め誤差によって，Wilkinson 流にいうなら，計算途中の丸め誤差が問題そのものの perturbation を引き起こして，特に $\|A^{-1}\|$ の縮小が止まってしまうことに起因している。ために，浮動小数点数の仮数部の桁数に依存して条件数も決定されてしまうことになる。

7 行列の固有値問題

行列 A の固有値 λ

$$Ax = \lambda x \tag{8}$$

を求める問題は，代数方程式の解を求める問題に帰着される。しかし，先に示したように，解の近接度合いによって精度が悪化する可能性があることから，今では陽に代数方程式の係数を求めずに Strum 列を形成して求めたり，行列のまま反復を重ねることによって固有値を求める手法が良いとされている。この場合，固有値の精度は近接度合いではなく，連立一次方程式

と同様，条件数によって決まってくる。特に絶対値最小の固有値の精度が悪くなる。これは代数方程式の場合とは全く逆である。それを利用して，代数方程式をCompanion行列と見立てて，固有値問題のアルゴリズムを適応するという手法も，実用化されているようである [1]。

また，2 次以上の Jordan ブロックを持つ行列の場合，その Jordan ブロックを形成する固有値の精度はちょうどブロックの次数だけ悪化する。しかし，対角化可能な部分の固有値の精度に影響は及ぼさない。ここに誤差のキャンセルの効用が現れている。

8 常微分方程式の初期値問題

次のような常微分方程式の初期値問題を考える。

$$\begin{cases} \frac{dy}{dx} &= f(x,y) \\ y(x_0) &= y_0 \end{cases} \tag{9}$$

これに対する数値解法として一番よく使用されているものとして，陽的Runge-Kutta 法やその拡張である埋め込み型公式がある。但し，これらの方法は「A 安定でない」ため，Stiff 問題に対しては，刻み幅 h が一定値以上になると計算の途中で overflow してしまう。それを解決するために陰的 Runge-Kutta 法というものが考案された。

このように，公式は安定性を高め，次数を上げる方向で進歩してきたのだが，そのような方法ではどうしても解決が難しい問題も存在する。

一つは，極を持つ問題，あるいは極に近い振る舞いをする点を持つ問題である。この場合は，極の近傍で解が急激に変動するため，極点を過ぎた直後から近似解の精度が著しく悪化し，公式の次数を上げるだけでは制度の改善が出来ず，ついには計算が続行できなくなってしまう。従って，Runge-Kutta 法のように次数固定の方法ではなく，可変次数の Taylor 展開法や補外法を利用しつつ，浮上小数点数の仮数部の桁数も増やさないことには精度の改善は望めないと考えられる。

もう一つは，Chaos 研究では著名な Lorentz モデルやそれの簡易版である Rössler モデルといった問題である。これは，パラメータが一定値になると，数値解の丸め誤差が徐々に増大していき，そのうち軌道を描いた図でも判別できるほど，ずれが顕在化する。これも直接的には桁落ちが主因であると思われる。そして解決策としては，やはり仮数部の桁数を増やすしかなさそうである。

9 最後に

以上見てきたように，数値解析においては，丸め誤差についての考察が不可欠である。現在では，ハードウェアの性能向上が著しいため，仮数部の桁数を増やした多倍長計算が当たり前のように行われるようになっている。従って，これからの時代は全ての数値計算アルゴリズムは多倍長演算で実行されるものである，と考える必要がある。

しかし，多倍長計算は見積もるのが厄介な丸め誤差を計算桁末尾に追いやる方便として利用されてきたきらいがあった。現実には有限桁演算である以上，仮数部の桁数を増やしたところで，丸め誤差を 0 にすることは不可能であり，丸め誤差を著しく増大させる悪条件な問題はこれからも存在し続ける。誤差解析の研究はこれから先も，メインストリームではないにしろ，絶やすことなく続けられなければならない。少なくとも私はそのように考えている[2]。

参考文献

[1] N. J. Higham, Accuracy and Stability of Numerical Algorithms, SIAM, 1996.

[2] インターフェース編集部 編, 数値演算プロセッサ, CQ 出版社, 1987.

[3] 伊理正夫・藤野和建, 数値計算の常識, 共立出版, 1985.

[4] 幸谷智紀, 常微分方程式および固有値問題における高精度計算法の研究, 博士論文 (日本大学), 1997.

[5] 森正武, 数値解析, 共立出版, 1973.

[6] 森口繁一, 数値計算工学, 岩波書店, 1989.

[7] 永坂秀子, 計算機と数値解析, 朝倉書店, 1980.

[8] J. H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford University Press, Reprinted 1992.

[9] J. H. Wilkinson, Rounding Errors in Algebraic Processes, Dover Publications, Inc., 1994.

[10] 山下眞一郎, 浮動小数点演算の誤差, bit 別冊 数値計算における誤差, 1976.

[2] 実行が伴っていない面は否定しない。