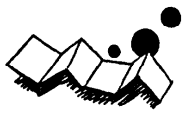
解　説



数値計算の常識†

一　松　　信††

はしがき

電子計算機は，元来は大規模な数値計算を目標として，発明され開発されたはずである．しかし近年ではとくに，広義の情報処理用に広く使用されるようになったため，数値計算自体は，特殊な利用形態として，計算機科学の中では，軽視されているように思われる．じっさいに共同利用大型計算機センターの利用者の大半をしめる数値計算用の使用者中には，随分と「非常識」といってもよい使い方をしている例が，必ずしも稀でないように見える．

本稿は，数値計算小特集の一環として，その種の実例をいくつか示し，問題提起とするものである．

1. 数値計算の常識の内容

標題の語は，まとまった理論というよりも，多分に断片的，雑学的な「生活の知恵」の集積である．しかしその内容は，次のように大まかに分類できるであろう：

Ⅰ　数値計算のための一般的な心得

例　誤差のある数の取り扱い，有限桁演算にともなう諸現象，純粋数学の理論とのくい違い，演算限界の認識，敏感な問題と不安定性

Ⅱ　既存の計算機や計算体系の不備に対する自衛手段

例　算法の工夫・変形，能率や互換性を二の次にした特殊体系の活用，できれば新しい計算機の設計

Ⅲ　能率向上のための一般的工夫

例　計算機内部の演算や動作の理解，収束の加速，適切な算法やライブラリの選択

もっとも計算機及び算法自体の発展にともない，「常識」も著しく変化する．たとえば Runge-Kutta-Gill のもとの算法において，記憶容量を減らす工夫は，今日の計算機ではほとんど必要がないし，丸め誤差の累積を防ぐトリックも，正しく理解しなければ役に立たない[1]．その意味でこれは今日では常微分方程式の数値解法において必ずしも有利な算法とは思われない．また対称行列の固有値を求める Jacobi 法の栄枯盛衰もその一例である．この例のように，昔の本に書かれている諸注意を金科玉条と心得ていると，時代錯誤になりかねない場合さえある．

他方同一の問題に複数個の算法があり，その優劣が重要な場合がある．小さい例だが，誤差関数の評価においては，$x=0$ 付近の公式（整級数など）と $x=\infty$ 付近の公式（漸近展開や連分数）の使い分けが問題になる．中間の x においては，どちらでも十分な精度で答が求められ，両者の差はもっぱら演算速度による．このような場合には，恐らく近い将来，緩い結合の並行処理が実用化され，別々の演算装置にそれぞれのプログラムを入れて並行して計算し，早く答がでた方を採用するようになるかもしれない．あるいは検算のために，ある時間待って（ある限度以上になったらそれを捨てる），他方の答とも比較する，という方式も考えられる．そのようになれば，公式の使い分けに関する理論の多くは無用の長物に近いだろう．

もちろんこのような「ぜいたく」な並行処理が広く実用化されるかどうかは別の問題である．しかしさらに進んで数値計算の各種の算法を生かせるような特殊目的の計算機の設計という課題は，夢をそそる分野である．

上記のⅡは，少々穏当でない語かもしれないが，現在広く使われている IBM/360 の流れを汲む十六進バイト方式が，有効桁数も指数部も狭すぎるだけでなく，数値計算用には本質的に不向きであることが，多くの人々によって指摘されている[2)~4)]．「非 von Neumann 型」よりも，まず「非 IBM 型」計算機を，という声が数値計算の専門家の間に根強いようである．

Ⅲについては，しばしば特定の機械の特定の機能の活用という話に陥りやすく，それらは一般性・互換性

† Common sense in Numerical Computation by Sin HITOTUMATU (Research Institute for Mathematical Sciences, Kyoto University).

†† 京都大学数理解析研究所

に乏しい．ただしある場合にはその種の工夫も重要であるし，ときとして，それが計算機のデリケートな性能試験として有用な場合さえある．本稿ではこの種の話題には深入りできないが，一つ問題提起をしておこう．近年**計算量の理論**が大いに発展してきたが，主記憶容量の限界を意識し，記憶の交換（スワッピング）回数を積極的に考慮に入れた計算量の理論というものは，まだほとんど研究されていないようである．超大型の線型問題では，四則演算回数よりも記憶の交換のほうが時間を制約する場合が多く，それを減らすことが，大幅な能率化をもたらすのが普通である．

2. 数値計算の一般的心得 (1) 有効桁の問題

数値計算の中には，整数の計算のように，十分の語長があれば，まったく誤差なしで計算できる場合もあるが，これはむしろ例外である．普通の数値計算の対象となる数は，誤差を含む連続量の表現であり，その取り扱いには誤差や有効数字の考え方が本質的である．このような知識は手計算時代には，余りにも当然すぎる常識であった．しかし機械計算の普及とともに，かえって無関心な使用者が増えてきたようである．計算尺時代には2桁合えば満足したのに，ポケット電卓時代になると，やたらと無意味な数値を並べたがる学生が増加してきたという意見もある．さらに現在の電子計算機のほとんどは，演算精度が機械的に一律に定められており，しかも暗箱中で大規模な演算が進められているために，途中で何が起こっているかわからないという危険がつきまとう．

手計算時代には，必要な精度より2桁ほど余分にとって計算し，最後の結果を丸めるのが常識であった．しかし現在では計算精度一杯の精度が要求される風潮が強い．10桁で計算して上位3桁のみを答とするというような態度は，しばしば「非現実的なぜいたく計算」と批判されている．もちろん標準組込み関数のようなライブラリ・サブルーチンでは，末位までできるだけ正しい値を出すように努力する必要がある．しかし十六進バイト方式の場合には，最下位は実質的にバッファであり，その値は誤差に埋れていると考えるのが，筆者の経験からみて現実的な態度だと思う．

数値計算の誤差解析において，以前から**桁落ち**が恐れられていた．しかし桁落ちそれ自体は，それほど深刻でない場合が多い．特に桁落ちする桁数の上限があらかじめ見積れる場合には，なまじ変な工夫をするよりも，それに耐えられるだけの長い桁をとって計算するのが，多くの場合安易だが最良の策であるのが普通である．

桁落ちがこわいのは，それが**情報落ち**と続いて生ずる場合である．典型例としては

$$(A+B)-A \qquad (1)$$

という計算である．$|A|$ が $|B|$ に比して著しく大きいとき，Bの情報がAに乗り切れず，Bになるべき(1)の答が0（あるいはBの下位が失われた数）になる．このとき，一時的に(1)を倍長で計算すれば，救われる場合が多い．枢軸選びができない状況の下で，小さい枢軸 a_{kk} で無理に計算を進めなければならないときには，2段階だけ倍長演算を採用すると切り抜けられることが多い[7]，という．この手法は，上記の(1)で $A=1/a_{kk}$ とした形の現象として説明できる．

Wilkinson[5] が繰り返し強調している積和の計算

$$\sum_{k=1}^{n} a_k \times b_k \qquad (2)$$

において，中間の積をすべて倍長数として保持し，最後に単長に丸める方法がよいというのも，中間に桁落ちが生じた折にも，下位の情報が有効に生かせるからである．じつは完全な倍長数でなくても，二進10ビット程度の余分なビットがあるだけで，著しく有効な場合が多い．もともと　単長数×単長数＝倍長数　というのが自然な「数学」のはずである．これを形式的に単長数と統一した無理が，多くの難問を生じているのではないか，といいたくなる．

ところで，普通には数値は，上位の数に主要な意味があり，下位の数は意味が薄いと考えられている．しかし整数論のある種の問題では，むしろ下位の数値が大事な情報をになっている場合が多い．年を表す1982年や，ユリウス日の2,445,279日などについても，何千年にもわたるデータを対象とするのでなく，最近の数年間だけが目的のときは，頭の19や244といった数値は余分な情報であり，それらを除いた下位の数値が本質的である．実際にも，このときにはもとのまま機械的に扱うより，余分な上位の数値を除いて計算したほうが，桁落ちを生ぜずよい値をうることが多い．これは従来余り注意されていなかった論点と思う．

最後に本節の以下の部分は筆者個人の好みであって，反論も多いと思うが，あえて述べてみる．FORTRANなどの実用言語でSINGLE, DOUBLEといった型の区別があるのは，もともと記憶装置が高価で，語の標準長を短くしたかった時代の，やむをえない必要悪だったのではなかろうか．数値計算専用の機

械として，恐らく理想的なのは，完全可変桁数の十進法の計算機であろう．もしそれが実現困難ならば，次善の策として，逆に実用上十分な精度（一例として標準をいえば，仮数部50ビット，指数部14ビット）をとった単一型の数（基底は適宜）とするほうがまだましだと思う次第である．

3. 数値計算の一般的心得 (2) 数学の理論との差

これについては Forsythe の警鐘[6]が有名である．そこには数多くの敏感な問題（初期値やデータの僅かな誤差が答を大きく変化させる例）や，不安定性（小さい誤差を増殖させる現象）の実例があげられている．

以下のような例は，現在では笑い話に近い．その理由を読者諸賢に解説するまでもないだろう．

1° テイラー展開

$$e^{-x}=1-\frac{x}{1!}+\frac{x^2}{2!}-\frac{x^3}{3!}+\frac{x^4}{4!}\cdots\cdots$$

に直接 $x=8$ を代入したら，e^{-8} の値が負になった．

2° $\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\cdots\cdots$

によって，直接に π の近似値を計算しようとした．

3° $\sum_{k=1}^{n}(1/k)$ を数値計算して，$n\to\infty$ のとき10（くらいの値）に収束すると予想した．

4° $\sqrt{0}$ を初期値1から Newton 法で計算したら，0.000027…… といった答がでた．

5° $$J_8(x)=\frac{1}{2\pi}\int_0^{2\pi}\cos(8\theta-x\sin\theta)d\theta \qquad (3)$$

で $x=0$ の値を Romberg 積分で計算したら，答が1になった．

(3)で $J_8(0)$ の正しい答は0である．これは第1回と第2回とでともに値が1になり，そこで計算を自動的に打ち切った失敗である．この種の算法には最小反復回数の指定が不可欠な典型例でもある．

上記の1°, 2° などは算法の選択の失敗例である．実用上有用な算法が，必ずしも数学の理論で最初に導入される定義そのままでないことを，早くから学生達に教育する必要があるのだろう．

線型計算における多くの難点の根本原因は，理論的な一次独立，次元，階数などの基本的諸概念がすべて，ある量 a が0か0でないかという判定に依存していて，実際の数値では a が0に近いが，0なのか0でないのかの判定が判然とできかねる点にあるという指摘がある．実際にこれを考慮に入れた実質的次元，実質的階数などの概念が，今日では線型計算の理論で基本的になってきている．

無限桁の精度をもつ理想化された数学の体系では同一だが，有限桁の数に対する下記のような工夫も，この意味での生活の知恵の一つといえよう．

1° $a>0$, $|b|$ が小のとき，$a-\sqrt{a^2-b^2}$ を $b^2/(a+\sqrt{a^2-b^2})$ と変形する．

2° 指数関数を加法定理で $e^x=e^a\times e^{(x-a)}$ として，$|x-a|$ が小さい場合に還元するとき，a をきっちりした数とするよりも，e^a がきっちりした数になるように a を選ぶ．

3° x が0に近いとき，$\sinh x=(e^x-e^{-x})/2$ は，この右辺でなく，$x+\frac{x^3}{3!}+\frac{x^5}{5!}+\cdots\cdots$ として求める．

他方，答の誤差が必要な限界以下だからといって安心できない場合があることにも注意を要する．通例の位取り記数法において，1.0000001 と 0.9999999 とは，同じく1に対して誤差 $\pm10^{-7}$ であるが，心理的には著しい非対称性をもっている．これは論外だとしても，

$\cos 0=1.000000002$

という答に対して，誤差が 2×10^{-9} で，非常によい答だとして，平然としてよいだろうか．よいこともあるが，$\sqrt{1-\cos\theta}$ が必要なときには ERROR になるだろう．もちろんこの種の判断は，かなり主観的なものになる．通例の計算では，ひとまず無視せざるをえないのかもしれないが，この種の配慮も欠かせない場合があるように思われる．

4. 既存の計算機体系の不備に対する自衛手段

素朴な一例をあげよう．**二項係数**

$$C(n,r)=\binom{n}{r}=\frac{n!}{r!(n-r)!} \qquad (4)$$

を計算したい．それには次のような諸算法が考えられる．

1° 階乗のサブルーチンにより，$n!, r!, (n-r)!$ を求めて，$n!\div r!\div(n-r)!$ とする．

2° $n/1\times(n-1)/2\times\cdots\cdots\times(n-r+1)/r$ （左から順に計算）とする．

3° 実数型で $[n/r]\times[(n-1)/(r-1)]\times\cdots\cdots\times[(n-r+1)/1]$ とする．

4° いわゆるパスカル三角形の形で，漸化式

$$C(n,r)=C(n-1,r)+C(n-1)r-1,$$
$$C(n,0)=C(n,n)=1$$

による．

1°が恐らくもっとも「自然」な方法であろう．しかし現在のたいていの計算機科学者は，もし数学者がこのように教えたならば，「現実を知らぬ空論家」と批難するであろう．能率の点もさることながら，既存の計算機で扱える数の指数部が狭いために，たちまち階乗の値があふれるからである．

しかし果して本当にそうだろうか．これには逆に，既存の計算機に，不自然な工夫を強いるような設計不備が内在しているという批判も，成立するのではなかろうか？

実用上二項分布

$$\binom{n}{r}p^r q^{n-r},\quad q=1-p \qquad (5)$$

の計算にあたって，$n=2,000$，$r=1,600$，$p=0.8$，$q=0.2$ というのは，それほど極端な例ではない．このとき(5)の正しい値は，ごく普通の大きさの数

0.0222966735231……

である．しかしこのとき二項係数，p^k, q^{n-k} はそれぞれほぼ 10^{433}, 10^{-155}, 10^{-280} である．これらは物理的世界では，とてつもない超天文学的数なのかもしれないが，数学の世界では日常ごく普通の大きさの数と思われる．この答を既存の計算機であふれを生じないように正しく求めるためには，非常に凝ったプログラムの工夫を強いられるであろう．これが(5)の形のままごく自然に計算できるような体系[3),4)]が一日も早く実現することを切望するものである．

ついでに上記の 2°～4° の算法にも言及しておく．

2° は整数演算で左から順に計算すれば，あふれが生じない限り，正しく整数値として計算できる．これを

$$[n/1]\times[(n-1)/2]\times\cdots\cdots\times[(n-r+1)/r]$$

としてもよいが，うっかり [ ] 内を整数除法とすると誤った答になる[7)]．実数型なら問題ないが，そのときは 3° のようにするほうがあふれを生じにくい．4° は一個の $C(n,r)$ のみを求めるのには不利であるが，十分の記憶容量があり，答を整数として末位まで正しく求めたいときには，最もよい算法である．

以上は一例にすぎないが，下記のような算法の工夫も，この種の例といえよう．

1°　$\sqrt{x^2+y^2}$ を if $x>y$ then

$x\times\mathrm{sqrt}(1+(y/x)^2)$ else $y\times\mathrm{sqrt}(1+(x/y)^2)$

と変形する．二次方程式の判別式 $\sqrt{b^2-4ac}$ などでも同様である．

2°　連分数 $\overset{\infty}{\underset{k=1}{\Phi}}\frac{a_k}{b_k}$ の逐次の値を次の漸化式で求めるとき，

$$P_k=b_kP_{k-1}+a_kP_{k-2}$$
$$Q_k=b_kQ_{k-1}+a_kQ_{k-2} \qquad (k\geq 2)$$
$$P_0=0,\ Q_0=1,\ P_1=a_1,\ Q_1=b_1$$

P_0, Q_k が大きくなりすぎれば両者を大きな数で割り，小さくなりすぎれば大きな数を掛ける．

指数部の幅が狭いことも既存の計算機の不便な点だが，整数論の計算では，何十桁という整数を最下位まで正しく求めたいという場合が多い．昔の和算家は，必要ならば算盤を何個も並べて，このような多倍長計算を実行したというが，今日それをシミュレートするとき，特に能率よくしようという場合には，特殊なプログラムの工夫を要する．あるいは，この種の計算には，多倍長プログラムよりも，いくつかの大きな素数 p に対する mod p の計算を数個並列に進める方式の計算機が有用なのかもしれない．この種の特殊目的用単能計算機の設計を，もっと積極的に考えてほしいと思う．

5. 誤差の下からの評価

これまで誤差解析は，大半が上からの評価であった．下からの評価というのは，本質的に困難が多いが，近年平野菅保初め，多くの人々によって，断片的ながら興味深い事実が現れ始めてきた．この種の研究は，計算の可能な限界を明示するとともに，ときとして算法の評価に有用な規準を与えるという意義がある．

下からの評価をうるには，$a+b$ を軽率に

$$|a+b|\leqq|a|+|b|$$

とまとめるわけにはゆかない．とくに a, b が逆符号で $a+b$ が著しく小さくなる場合に注意を要する．じっさい A の誤差 a と B の誤差 b とが，元来同一の祖先から生じた単一の起源の誤差で，互いに逆符号になっているときには，$A+B$ の誤差は相殺して著しく小さくなる．重根をもつ3次方程式に Cardano の解法を適用したときが，その典型例である（平野，公式の論文は未発表）．ただし**誤差の相殺**という語は，本来このような意味に限って使用すべきである．たとえば積分

$$\int_{-\infty}^{\infty}e^{-x^2/2}dx=2\int_0^{\infty}e^{-x^2/2}dx=\sqrt{2\pi}$$

を刻み幅 $h=1$ の台形公式により -7 から 7 まで計算すると，誤差 1.5×10^{-8} という信じ難い高精度が得られるが，これは「誤差の相殺」というよりも，端の

値による台形公式の誤差評価から示される結果である.

誤差の下からの評価には,誤差の相殺のほか,いろいろな難問がある.誤差の主要項が消える例としては,

$$\int_0^1 \frac{dx}{1+x^2}=\frac{\pi}{4}$$

を Simpson 公式で求めると,刻み幅 h に対して h^4 の項が消え,誤差の主要項が h^6 に比例するので,異常に精度が高くなる.このような例は,算法の比較にあたって,算法に対する誤った評価を防ぐためにも,十分に心得ておく必要がある.

ある場合には,理論的に下からの評価が可能なことがある.$(1+1/n)^n$ は,e^1 に対して微分方程式 $y'=y$, $y(0)=1$ を刻み幅 $1/n$ で 0 から 1 まで Euler 法で解いた近似値と解釈される.このときには理論上 **Morreau の不等式**

$$\frac{e}{2n+2}<e-\left(1+\frac{1}{n}\right)^n<\frac{e}{2n+1} \qquad (6)$$

が成立することが知られている.しかし $(1+1/n)^n$ の値を電卓で計算すると,しばしば(6)の左側が成立しない「よすぎる答」がえられることがある.これを「完全犯罪」とよぶことがある[8].その原因も解析されている.

算法の評価のための標準問題にも十分注意を要する.線型計算ではしばしば Hilbert 行列 $[a_{ij}]$: $a_{ij}=1/(i+j-1)$ が試験に使われる.しかし Hilbert 行列は,条件指数は大きいが,正値対称,かつ成分が簡単な有理数であり,適当なスケーリングをするかまたは有理数計算によればよい答が得られるなど,特殊性が多すぎて,必ずしも適切な試験行列とはいい難い[9].ほかにも,ある一面のみに注目した伝統的な例では不適切な場合が数多く指摘されている.

むすび

以上,断片的かつ初等的な実例を述べてきた.この種の例は個々としては平凡かもしれないが,案外類似の失敗を犯している場合があるのではなかろうか.

数値計算の利用者の大半は,伝統的に FORTRAN の使用者であり,ある意味で「FORTRAN 公害」ともいうべき現象が見られる.そしてこの種の利用者が意外に保守的(?)であり,新しい算法の普及にも時間がかかり,そのために本当の革新が困難という意見もあるようである.本小特集が,そのために有用なてびきとなることを期待して筆をおく.

参考文献

1) 伊理正夫,松谷泰行: Runge-Kutta-Gill 法について,情報処理, Vol. 9, pp. 103-107 (1967).
2) 一松 信: 新標準浮動小数点体系の提案,情報処理, Vol. 20, No. 9 pp. 793-797 (1979).
3) 松井正一,伊理正夫: あふれのない浮動小数点表示方式,情報処理学会論文誌, Vol. 21, No. 4 pp. 306-313 (1980).
4) 浜田穂積: 二重指数分割に基づくデータ長独立実数値表現法,情報処理学会論文誌, Vol. 22 No. 6, pp. 521-526 (1981).
5) Wilkinson, J. H.: The algebraic eigenvalue problem, Oxford Univ. Press (1965).
6) Forsythe, G. E., 清野 武他訳: 数値計算の陥し穴, ——なぜ数学の教科書だけでは足りないのか?——京都大学大型計算機センター広報(1969); bit (1977) に再掲載.
7) 永坂秀子: 電子計算機と数値計算,朝倉書店,第2章 (1980).
8) 高橋秀俊: ポケット電卓の完全犯罪,数理の散策,日本評論社に収録 (1974).
9) Forsythe, G. E.-Wasow, G. (渋谷,田辺訳): 線型計算の基礎,培風館 (1971).

(昭和56年11月26日受付)