数学入門公開講座

平成4年8月4日(火)から8月13日(木)まで

京都大学数理解析研究所

講師及び内容

1. 確率論の話題から（7時間）

京都大学数理解析研究所・助教授　楠　岡　成　雄

確率論に関連した4つの話題（シャノンの情報理論、確率制御の話題、シミュレーテッド・アニーリング、乱数の理論）について話を行う予定である。どのように問題をとらえるかといった考え方について話の力点を置いて話していく。

2. 「保証書」付き数値計算法（7時間）

京都大学数理解析研究所・助教授　室　田　一　雄

計算機のできる数値計算は、所栓は誤差を含んだ近似計算である。それでも、その計算結果に従って橋は作られ列車は走る。数値計算誤差につきまとう不安感を一掃し、ぼんやりとした計算結果に基づいてはっきりとした結論を導き出す手法について解説する。

3. 「時間とミクロ世界・マクロ世界」（7時間）

京都大学数理解析研究所・助教授　小　嶋　　泉

「時間とは何か？人が問わなければ、私はそれを知っている。だが人に説明しようとすると、私は最早それを知っていない」――時間を巡る謎のとらえどころのなさは、聖アウグスティヌスのこの有名な言葉によく表わされている。自然法則を書き下す上で時間概念は不可欠であり、今世紀の物理学革命も時空間概念の根本的変革と軌を一にして達成された。それによってこの「謎」はどう解き明されたのか？時間概念を軸に、基本的な物理法則とその論理構造を概観し、そこから、自然の運動・構造・歴史を理論的・数学的に記述することの意味を改めて考え直してみたい。

4. グラフと組合せ論（7時間）

京都大学数理解析研究所・助手　松　本　　眞

グラフ理論や組合せ論の世界は、まだいまいち体系的ではありませんが、たとえばかつての四色問題などのように、誰でも問題の内容がすぐ理解できるのに、証明はいまだ満足にされていないような楽しい予想が小石のように転がっています。こうした楽しそうな部分をつまみぐいして、平易にグラフと組合せ論の紹介をします。

時　間　割

日 / 時間	8月4日(火)	5日(水)	6日(木)	7日(金)	8日(土) 9日(日)	10日(月)	11日(火)	12日(水)	13日(木)
13:15～15:00	楠岡	楠岡	楠岡	楠岡	休講	小嶋	小嶋	小嶋	小嶋
15:00～15:15	休憩					休憩			
15:15～17:00	室田	室田	室田	室田		松本	松本	松本	松本

2．「保証書」付き数値計算法 （7時間）

京都大学数理解析研究所・助教授 室 田 一 雄

1992, AUGUST 4,5,6,7　　15:15-17:00

「保証書」付き数値計算法

室田一雄

有限桁演算に基づく数値計算の結果は所詮は離散化誤差や丸め誤差を含む近似値である．そのような数値計算結果の精度を厳密な意味で保証する手法が近年飛躍的に発展し，精度保証付き数値計算法と呼ばれている．従来，区間演算による手法は実用性をもたないと考えられてきたが，これを高速自動微分法や不動点定理と組み合せることによって，計算量と保証精度の両面で実用的な精度保証付き数値計算法が得られる．本講義の目的は，最近の進歩によってにわかに実用性を帯びてきた区間解析の手法を紹介することである．

1 区間演算

数値計算の結果は丸め誤差に満ちている．例えば，$x = \pi\ (= 3.14159\cdots)$ と $y = \mathrm{e}\ (= 2.71828\cdots)$ に対して $\phi(x,y) = x/y$ を知りたいとき，10進3桁4捨5入で計算すると $x/y \sim 3.14/2.72 \sim 1.154\cdots \sim 1.15$ となる．この計算値 1.15 は真値 $\pi/\mathrm{e}\ (= 1.155727\cdots)$ の近似値であるが，その精度はどの程度保証できるであろうか．

区間解析は，数値計算結果に含まれる丸め誤差の厳密な限界を得るための手法である．区間解析では，数値に含まれる誤差を考慮して，実数 x をその下限値 $\underline{x}$ と上限値 $\overline{x}$ の組 (すなわち区間) $X = [\underline{x}, \overline{x}]$ $(\underline{x} \le x \le \overline{x})$ で表現し，すべての演算を区間に対する演算 (区間演算) で置き換える．すなわち，実数 x, y に対する演算 $\phi(x,y)$ に対応して，区間 $X = [\underline{x}, \overline{x}]$, $Y = [\underline{y}, \overline{y}]$ に対する区間演算 $\Phi(X,Y) = \{\phi(x,y) \mid x \in X, y \in Y\}$ を定義する．例えば，四則演算に対する区間演算は次のようになる (除算では $0 \notin Y$ とする):

$$
\begin{aligned}
X + Y &= [\underline{x} + \underline{y}, \overline{x} + \overline{y}] \\
X - Y &= [\underline{x} - \overline{y}, \overline{x} - \underline{y}] \\
X * Y &= [\min\{\underline{x} * \underline{y}, \overline{x} * \overline{y}, \underline{x} * \overline{y}, \overline{x} * \underline{y}\}, \max\{\underline{x} * \underline{y}, \overline{x} * \overline{y}, \underline{x} * \overline{y}, \overline{x} * \underline{y}\}] \\
X / Y &= [\min\{\underline{x} / \underline{y}, \overline{x} / \overline{y}, \underline{x} / \overline{y}, \overline{x} / \underline{y}\}, \max\{\underline{x} / \underline{y}, \overline{x} / \overline{y}, \underline{x} / \overline{y}, \overline{x} / \underline{y}\}]
\end{aligned}
$$

また, $|X| = \max(|\underline{x}|, |\overline{x}|)$ と定義する.

計算機で扱える区間 $X = [\underline{x}, \overline{x}]$ は, 上下限がその計算機で表現できる浮動小数点数に含まれるものに限られる. このような区間を機械区間と呼び, その全体を $\mathcal{I}_{\mathrm{M}}$ と記す. 一般には, $X, Y \in \mathcal{I}_{\mathrm{M}}$ であっても $\Phi(X,Y) \in \mathcal{I}_{\mathrm{M}}$ とは限らないので, 実際に実行できる区間演算は $\Phi_{\mathrm{M}}(X,Y) = \cap\{Z \mid \Phi(X,Y) \subseteq Z \in \mathcal{I}_{\mathrm{M}}\}$ である. これを機械区間演算と呼ぶ.

上の例で, 10進3桁の数が計算機上で表現できるとすると, $x = \pi$, $y = \mathrm{e}$ に対応する機械区間として $X = [\underline{x}, \overline{x}] = [3.14, 3.15]$, $Y = [\underline{y}, \overline{y}] = [2.71, 2.72]$ をとることができる. このとき, $\Phi(X,Y) = [\underline{x}/\overline{y}, \overline{x}/\underline{y}] = [1.154417\cdots, 1.16236\cdots]$, $\Phi_{\mathrm{M}}(X,Y) = [1.15, 1.17]$ である.

以下では, 実数 x, y に対して定義された演算 $\phi(x,y)$ に対応して, 丸め誤差を含んだ演算 $\tilde{\phi}(x,y)$ が実行でき, さらに,

$$\Phi(X,Y) \supseteq \{\phi(x,y) \mid x \in X, y \in Y\},\ \Phi(X,Y) \supseteq \{\tilde{\phi}(x,y) \mid x \in X, y \in Y\}, \tag{1.1}$$

を満たす“区間演算”Φ が実行できるものと仮定する (一般の N 項演算に対しても同様). 機械区間演算 Φ_{M} は, 定義により, この第一の条件を満たしていることに注意.

関数 $f(x_1, \ldots, x_n)$ の計算手続きが通常のプログラムのような形で与えられているときに, 関数値 $f(x_1, \ldots, x_n)$ を含む厳密な (機械) 区間を計算するには, 計算手続き中の変数を区間変数で, 演算を対応する区間演算で, 入力変数 x_j を幅が0の区間 $[x_j, x_j]$ で置き換えて計算を進めていけばよい. このような素朴な区間解析法は既に1960年代に考案されたが, これを大規模計算に適用してみると, 多くの場合に無意味な程幅の広い保証区間を与えることが経験された. その結果, 厳密ではあるが精密でない区間解析は実用的でないとして, 区間解析の手法は長い間顧みられないこととなった.

しかし, 今, 時代は変わろうとしている ….

2 計算グラフと高速自動微分法

高速自動微分法は, n 個の変数 x_j $(j = 1, \ldots, n)$ のスカラー値関数 $f(x_1, \ldots, x_n)$ の計算手続きが与えられているときに, その勾配 $\nabla f = (\partial f / \partial x_j)$

や Hesse 行列 $H = (\partial^2 f/\partial x_i \partial x_j)$ などを差分近似に依らずに求める手法で，つぎのような特長をもつ: (1) 計算の高速性: ∇f の値や，任意の定数行ベクトル y と H との積 $y^{\mathrm{T}}H$ の値を，f を計算するために必要な手間の定数倍 (n に無関係) の手間で計算する．(差分近似に基づく方法による ∇f の計算が通常 $n+1$ 回以上の関数値計算を必要とするのに比べて有利である．) (2) 値の正確さ: ∇f や $y^{\mathrm{T}}H$ の値の計算精度は，いったん数式の形で求められた ∇f や $y^{\mathrm{T}}H$ に数値を代入して計算される値と同程度である．(3) 自動化可能性: 関数の計算手続き (条件分岐や反復などを含んでよい) が与えられれば，その導関数値を自動的に計算する手続きを，計算機上で実現できる．

2.1 基本算法

入力変数 $x_1, \ldots, x_n$ の値が与えられたとき，$f(x_1, \ldots, x_n)$ は，$x_1, \ldots, x_n$ から始めて，表2.1のような基本演算 ϕ_k の結果を中間変数 v_k に記憶するという計算ステップを繰り返して，

$$v_k = \phi_k(u_{kp} \mid p = 1, \ldots, N_k) \quad (k = 1, \ldots, r) \tag{2.1}$$

のように計算されるとする．ここで，N_k は仮引き数 u_{kp} の個数 (表2.1の基本演算では $N_k \leq 2$) であり，u_{kp} の実引き数は，入力変数 x_j，あるいは v_k より前の中間変数 v_l $(l < k)$ のいずれかである[1]．f の値は v_r に計算される．(2.1) を f の計算過程と呼び，関数定義と区別する．関数 f が条件分岐などを含んだ形で定義されている場合には，計算過程は入力変数の値に応じて定まる履歴であることに注意．なお，基本演算 ϕ_k の u_{kp} に関する偏導関数 $\phi_{kp} = \partial\phi_k/\partial u_{kp}$ を要素的偏導関数と呼ぶ．

計算過程を微分して得られる関係式

$$\mathrm{d}v_k = \sum_{p=1}^{N_k} \phi_{kp} \cdot \mathrm{d}u_{kp} \quad (k = 1, \ldots, r) \tag{2.2}$$

から $\mathrm{d}v_k$ $(k = 1, \ldots, r-1)$ を消去して $\mathrm{d}v_r = \sum_{j=1}^{n} \xi_j \mathrm{d}x_j$ の形の式を導けば，$\xi_j = \partial f/\partial x_j$ である．(2.2) は，$\mathrm{d}x = (\mathrm{d}x_1, \ldots, \mathrm{d}x_n)^{\mathrm{T}}$，$\mathrm{d}v =$

[1]実引き数が定数の場合は，これを入力変数の一つと見なすことにする．本節で扱う高速自動微分法の範囲では，定数を別に扱った方が算法の効率はよくなるが，後に述べる区間解析による丸め誤差解析のためには入力変数に含めておく方が便利である

表 2.1: 基本演算 ϕ_k, 要素的偏導関数 ϕ_{kp}, 要素的 2 階偏導関数 ϕ_{kpq}

ϕ_k	ϕ_{k1}	ϕ_{k2}	ϕ_{k11}	ϕ_{k12}, ϕ_{k21}	ϕ_{k22}
$-u_{k1}$	-1	—	0	—	—
$u_{k1}+u_{k2}$	1	1	0	0	0
$u_{k1}-u_{k2}$	1	-1	0	0	0
$u_{k1}*u_{k2}$	u_{k2}	u_{k1}	0	1	0
u_{k1}/u_{k2}	$1/u_{k2}$	$-u_{k1}/{u_{k2}}^2$	0	$-1/{u_{k2}}^2$	$2u_{k1}/{u_{k2}}^3$
$\exp u_{k1}$	$\exp u_{k1}$	—	$\exp u_{k1}$	—	—
$\log u_{k1}$	$1/u_{k1}$	—	$-1/{u_{k1}}^2$	—	—
$\sin u_{k1}$	$\cos u_{k1}$	—	$-\sin u_{k1}$	—	—
$\cos u_{k1}$	$-\sin u_{k1}$	—	$-\cos u_{k1}$	—	—

$(\mathrm{d}v_1, \ldots, \mathrm{d}v_r)^{\mathrm{T}}$ とおくと,

$$A \begin{bmatrix} \mathrm{d}\boldsymbol{x} \\ \mathrm{d}\boldsymbol{v} \end{bmatrix} = \begin{bmatrix} M & | & L \end{bmatrix} \begin{bmatrix} \mathrm{d}\boldsymbol{x} \\ \mathrm{d}\boldsymbol{v} \end{bmatrix} = 0 \tag{2.3}$$

の形に書ける. ここで, $A = [M \mid L] = (a_{kj})$ は $r \times (n+r)$ 行列, $M = (m_{kj})$ は $r \times n$ 行列, $L = (l_{kj})$ は対角要素が -1 の r 次左下三角行列である. 例えば, x_j が u_{k1} の実引き数なら $m_{kj} = \phi_{k1}$, v_j が u_{k1}, u_{k2} の両方の実引き数なら $l_{kj} = \phi_{k1} + \phi_{k2}$ である. 計算過程の遂行後には A の要素の値が確定していることに注意せよ.

例 2.1 $v_1 = x_1 - x_2, v_2 = x_2 * x_3, v_3 = v_1/v_2, v_4 = \exp v_3$ は $f(x_1, x_2, x_3) = \exp((x_1 - x_2)/(x_2 * x_3))$ の計算過程であり,

$$A = \left[\begin{array}{ccc|ccc} 1 & -1 & & -1 & & \\ & x_3 & x_2 & & -1 & \\ & & & 1/v_2 & -{v_1}/{v_2}^2 & -1 \\ & & & & \exp v_3 & -1 \end{array}\right]$$

である. □

行列 A において, 第 $(r-1)$ 行の $a_{r,n+r-1}$ 倍を第 r 行に足し込んで $a_{r,n+r-1} = 0$ とし, 次に第 $(r-2)$ 行の $a_{r,n+r-2}$ 倍 (更新後の値) を第 r 行に足し込ん

で $a_{r,n+r-2}=0$ とし，…というように $a_{r,n+k}$ $(k=r-1,r-2,\ldots,1)$ を順次零とする計算，すなわち，$k=r-1,r-2,\ldots,1$ に対して

$$a_{rj} := a_{rj} + a_{r,n+k}a_{kj} \quad (j=1,\ldots,n+k) \tag{2.4}$$

によって，最終的に $a_{rj}=\xi_j=\partial f/\partial x_j$ $(j=1,\ldots,n)$ となる．実際には，入力変数 x_j $(j=1,\ldots,n)$ と中間変数 v_k $(k=1,\ldots,r)$ に対応する作業変数 sx_j $(j=1,\ldots,n)$, sv_k $(k=1,\ldots,r)$ を用意し，次のように計算する ((2.4) の $a_{r,n+k}$ が (2.5) の sv_k に対応する)．仮引き数 u_{kp} の実引き数に対応する作業変数を $s(u_{kp})$ と表す．

高速微分の基本算法 (計算過程による表現): sv_r を 1 に，その他の作業変数を 0 に初期化し，$k=r,r-1,\ldots,1$ に対して

$$s(u_{kp}) := s(u_{kp}) + \phi_{kp}\cdot sv_k \quad (p=1,\ldots,N_k) \tag{2.5}$$

とすると，$sx_j=\partial f/\partial x_j$ $(j=1,\ldots,n)$. □

ここで，要素的偏導関数 ϕ_{kp} は基本演算 ϕ_k と同程度の手間で計算でき (→表 2.1)，(2.5) の計算は $(\sum_{k=1}^{r} N_k)$ 回の加算と乗算でできる．従って，N_k が n に依らない数で抑えられていれば，f と ∇f の両方の計算に要する手間 $L(f,\nabla f)$ は f だけの計算に要する手間 $L(f)$ の数倍，すなわち，$L(f,\nabla f)\le C\cdot L(f)$ であって，C は入力変数の数 n に依らない．

例 2.2 例 2.1に対して上の計算 (2.5) は次のようになる: $sv_4:=1$, $sv_1:=0$, $sv_2:=0$, $sv_3:=0$, $sx_1:=0$, $sx_2:=0$, $sx_3:=0$; $sv_3:=sv_3+(\exp v_3)*sv_4$; $sv_1:=sv_1+(1/v_2)*sv_3$, $sv_2:=sv_2-(v_1/{v_2}^2)*sv_3$; $sx_2:=sx_2+x_3*sv_2$, $sx_3:=sx_3+x_2*sv_2$; $sx_1:=sx_1+1\cdot sv_1$, $sx_2:=sx_2-1\cdot sv_1$. 実際には，さらに個々の基本演算の性質を利用して，指数関数演算のステップ $(k=4)$ は $sv_3:=sv_3+v_4*sv_4$ の形で，除算のステップ $(k=3)$ は $sv_1:=sv_1+(1/v_2)*sv_3$, $sv_2:=sv_2-(v_3/v_2)*sv_3$ の形で計算するのがよい． □

計算過程は計算グラフとしても表現される．計算グラフ G の頂点 v は計算過程に現れる入力変数，中間変数を表す．枝 a は計算過程中の仮引き数に対応する．a が仮引き数 u_{kp} に対応するとき，a の終点 $\partial^- a$ は中間変

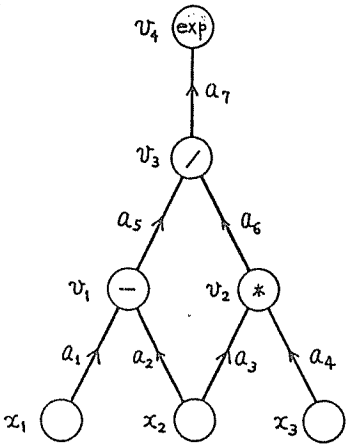

図 2.1: 例 2.1の f の計算グラフ G

数 v_k を表す頂点，始点 $\partial^+ a$ は u_{kp} の実引き数を表す頂点であり，要素的偏導関数 ϕ_{kp} が $d(a)$ としてこの枝に付随している．また，$\delta^- v_k = \{a \mid \partial^- a = v_k\}$，$\delta^+ v_k = \{a \mid \partial^+ a = v_k\}$ とおく．例 2.1の計算グラフを図 2.1に示す．頂点集合 $= \{x_1, x_2, x_3, v_1, v_2, v_3, v_4\}$，枝集合 $= \{a_1, \ldots, a_7\}$ で，$\partial^+ a_1 = x_1$，$\partial^- a_1 = v_1$，$\delta^- v_1 = \{a_1, a_2\}$，$\delta^+ v_1 = \{a_5\}$ などである．

計算グラフ G は，有向閉路を含まないグラフ (非巡回グラフ) であり，G 上で v_k から v_l に有向道があることは v_l より v_k を先に計算すべきことを，逆に，有向道がないことはその必要のないことを表している．例えば，図 2.1において v_2 より v_1 を先に計算する必要はない．このように，計算グラフは中間変数の間の本質的な依存関係 (半順序関係) を明確に表現しており，その半順序に矛盾しない限り，計算過程 (2.1) の基本演算の順序を変えることができる．

計算グラフの概念は数値計算の構造解析の様々な局面で有用であるが，上述の基本算法 (2.5) も計算グラフ G 上の計算として次のように記述できる．

高速微分の基本算法 (計算グラフによる表現): sv_r を 1 に，その他の作業変数を 0 に初期化し，$k = r, r-1, \ldots, 1$ に対して

$$s(\partial^+ a) := s(\partial^+ a) + d(a) \cdot sv_k \quad (a \in \delta^- v_k) \tag{2.6}$$

とすると， $sx_j = \partial f/\partial x_j$ $(j = 1, \ldots, n)$. □

また， v_k の摂動に対する f の変化率を $\partial f/\partial v_k$ とするとき，

$$sx_j = \frac{\partial f}{\partial x_j} = \sum_{P:x_j \to f} \prod_{a \in P} d(a), \quad sv_k = \frac{\partial f}{\partial v_k} = \sum_{P:v_k \to f} \prod_{a \in P} d(a) \qquad (2.7)$$

($\sum_{P:v\to f}$ は v から f に至るあらゆる有向道 P についての和) が成り立つ.

2.2 応用

計算過程に従った数値計算の結果である関数値 $f(\boldsymbol{x})$ には，丸め誤差や超越関数を近似式でおきかえることによる誤差などの計算誤差 δf が含まれている．入力変数 x_j に含まれる誤差を δx_j，基本演算 ϕ_k で発生する誤差を δ_k とすると，誤差が小さいとき，

$$\delta f \sim \sum_{k=1}^{r} \frac{\partial f}{\partial v_k}\delta_k + \sum_{j=1}^{n} \frac{\partial f}{\partial x_j}\delta x_j \qquad (2.8)$$

と近似される (厳密な評価は式 (3.5))．この式の意味を (2.7) によって計算グラフ上に翻訳すると，各頂点 (v_k や x_j) で発生した誤差がいろいろな有向道 P に沿って $\prod_{a\in P} d(a)$ 倍に拡大されて頂点 f に到達するということである．現在の計算環境で妥当な仮定: $|\delta_k| \le \varepsilon|v_k|$ (ε はいわゆるマシンエプシロン) の下では，この第 1 項は

$$\delta_{\mathrm{A}} f = \sum_{k=1}^{r} \left|\frac{\partial f}{\partial v_k} \cdot v_k\right| \varepsilon \qquad (2.9)$$

で抑えられる (この評価式は実際の数値計算誤差の大きさの程度をよく反映していることが経験的に知られている)．ここで，係数 $\partial f/\partial v_k$ は高速微分法の副産物として容易に計算される (式 (2.7) 参照) ので，この評価式を実際的な大規模計算に利用することができる (停止規則 (4.3) 参照).

3 関数値計算の精度保証

関数 $f(x_1, \ldots, x_n)$ の計算手続きが通常のプログラムのような形で与えられているときに，関数値 $f(x_1, \ldots, x_n)$ を含む (機械) 区間を計算する手法を考える.

入力変数 $x_1,\ldots,x_n$ に対応する計算過程 ((2.1) 式参照) を

$$v_k = \phi_k(u_k) \qquad (k=1,\ldots,r) \tag{3.1}$$

とする (ただし, $u_k = (u_{kp} \mid p=1,\ldots,N_k)$). 既に述べたように, 関数値に含まれる丸め誤差は実用上十分な精度で (2.9) で見積ることができるが, この評価式は誤差が小さいことを前提とした線形近似式であり, (2.9) の計算自体も丸め誤差を含むので, 厳密な意味での精度保証を与えない.

厳密な保証区間を得るには, 関数の計算手続き中の変数を区間変数で, 演算を対応する区間演算で, 入力変数 x_j を幅が 0 の区間 $[x_j,x_j]$ で置き換えて計算すればよい. このとき, 計算手続き中の "$v < w$?" のような分岐条件は, 区間変数に関する "$[\underline{v},\overline{v}] < [\underline{w},\overline{w}]$?" のような条件に置き換えられるが, これが判定可能 ($\overline{v} < \underline{w}$ または $\underline{v} \geq \overline{w}$) であって, 分岐先が確定すると仮定する. この仮定 (関数値の計算可能性の仮定) の下では, 区間演算による計算過程 (計算グラフ), 丸め誤差を含む演算による計算過程 (計算グラフ) はともに, 丸め誤差のない理想的な計算過程 (計算グラフ) と同じ構造をもち, f に対応する区間 F は真の関数値を含むことがわかる (式 (3.8) 参照).

このような素朴な区間解析法は既に 1960 年代に考案されたが, これを大規模計算に適用してみると, 多くの場合に無意味な程幅の広い保証区間を与えることが経験された.

より精密な保証区間は, 平均値の定理

$$f(y) - f(x) \in f'(X)\cdot(y-x) \quad (x,y \in X)$$

(ただし $f'(X) = \{f'(x) \mid x \in X\}$) に基づいて導かれ, 区間演算と高速微分法の応用として, 以下のように計算される (関数値の計算可能性を仮定する). 丸め誤差のない理想的な計算過程を (3.1) で表すものとし, 丸め誤差を含んだ計算過程を

$$\tilde{v}_k = \tilde{\phi}_k(\tilde{u}_k) = \phi_k(\tilde{u}_k) + \delta_k \qquad (k=1,\ldots,r) \tag{3.2}$$

(ただし, $\tilde{u}_k = (\tilde{u}_{kp} \mid p=1,\ldots,N_k)$) と書くことにする. 上式で定義される δ_k は第 k 計算ステップで発生する丸め誤差を表している. 平均値の定理

表 3.1: 要素的偏導関数 ϕ_{kp} に対応する区間 Φ_{kp} $(V_k = \Phi_k(U_k))$

Φ_k	Φ_{k1}	Φ_{k2}
$-U_{k1}$	$[-1,-1]$	—
$U_{k1}+U_{k2}$	$[1,1]$	$[1,1]$
$U_{k1}-U_{k2}$	$[1,1]$	$[-1,-1]$
$U_{k1}*U_{k2}$	U_{k2}	U_{k1}
U_{k1}/U_{k2}	$[1,1]/U_{k2}$	$[-1,-1]*V_k/U_{k2}$
$\exp U_{k1}$	V_k	—
$\log U_{k1}$	$[1,1]/U_{k1}$	—

により，

$$\tilde{v}_k - v_k = \phi_k(\tilde{u}_k) - \phi_k(u_k) + \delta_k = \sum_{p=1}^{N_k} \tilde{\phi}_{kp} \cdot (\tilde{u}_{kp} - u_{kp}) + \delta_k, \qquad (3.3)$$

$$\tilde{\phi}_{kp} = \left.\frac{\partial \phi_k}{\partial u_{kp}}\right|_{u_k + \theta_k(\tilde{u}_k - u_k)}, \quad 0 < \theta_k < 1, \qquad (3.4)$$

が成り立つ．(2.2) から (2.7), (2.8) が導かれたように，(3.3) から

$$\tilde{f} - f = \sum_{k=1}^{r} w_k \delta_k, \quad w_k = \sum_{P: v_k \to f} \prod_{a \in P} \tilde{\phi}(a) \qquad (3.5)$$

が導かれる．ただし，計算グラフ上の枝 a が仮引き数 u_{kp} を表すとき，$\tilde{\phi}(a) = \tilde{\phi}_{kp}$ と定義し，$\sum_{P:v_k \to f}$ は v_k から f に至るあらゆる有向道 P についての和を表す．高速微分の基本算法 (2.6) に対応して，w_k は次のように計算される: $w_r := 1$, $w_k := 0$ $(k = 1, \ldots, r-1)$ と初期化し，$k = r, r-1, \ldots, 1$ に対して

$$w(\partial^+ a) := w(\partial^+ a) + \tilde{\phi}(a) \cdot w_k \quad (a \in \delta^- v_k) \qquad (3.6)$$

とする．ここで，$\partial^+ a = v_j$ のとき $w(\partial^+ a) = w_j$ と定義し，$\partial^+ a$ が入力変数 (や定数) のときには，その a に対する計算は行わないと約束する．

計算過程 (3.1) を (機械) 区間演算で置き換えたものを

$$V_k = \Phi_k(U_k) \qquad (k = 1, \ldots, r) \qquad (3.7)$$

(ただし, $\mathbb{U}_k = (U_{kp} \mid p = 1, \ldots, N_k)$) とすると, (1.1) により

$$v_k \in V_k, \quad \tilde{v}_k \in V_k \qquad (k = 1, \ldots, r) \tag{3.8}$$

が成り立ち, とくに $F = V_r$ が f の保証区間を与える (上で素朴な区間解析法と呼んだものである).

要素的偏導関数 ϕ_{kp} を区間演算で計算したものを Φ_{kp} とする (→表 3.1) と, $0 < \theta_k < 1$ より $u_{kp} + \theta_k(\tilde{u}_{kp} - u_{kp}) \in U_{kp}$ が成り立つので, (1.1) と (3.4) により $\tilde{\phi}_{kp} \in \Phi_{kp}$ である. 従って, (3.6) において w_k を区間変数 W_k で, $\tilde{\phi}_{kp} = \tilde{\phi}(a)$ を $\Phi_{kp} = \Phi(a)$ で置き換えれば, $w_k \in W_k$ となる. さらに, 第 k 計算ステップで発生する丸め誤差 δ_k に対する保証区間 D_k が, 例えばマシンエプシロン ε を用いて $D_k = [-\varepsilon \cdot |V_k|, \varepsilon \cdot |V_k|]$ のように設定できるとすれば, (3.5) の第 1 式を区間演算で置き換えた式から f の保証区間がわかる.

以上の考察から, 計算機で表現できる数 x_j $(j = 1, \ldots, n)$ に対する関数値 $f(x_1, \ldots, x_n)$ は, 次のようにして計算される区間 L と丸め誤差を含んだ計算値 $\tilde{f}$ を用いて, $\tilde{f} - f \in L$ と保証される. 従って, $|L|$ が $\tilde{f}$ に含まれる丸め誤差の厳密な上界を与える (次節の例 5.1参照).

関数値の保証区間の計算法: f の計算手続きを区間演算に置き換え, 幅が 0 の区間 $[x_j, x_j]$ を入力として実行する (この際, 関数値の計算可能性を確認する); $W_r := [1, 1]$, $W_k := [0, 0]$ $(k = 1, \ldots, r-1)$ と初期化し, $k = r, r-1, \ldots, 1$ に対して [$W(\partial^+ a) := W(\partial^+ a) + \Phi(a) \cdot W_k$ $(a \in \delta^- v_k)$] として W_k を計算する ((3.6) の直後の約束に従う); 発生誤差 δ_k を含む区間を $D_k = [-\varepsilon \cdot |V_k|, \varepsilon \cdot |V_k|]$(など) と定め, 区間 $L = \Sigma_{k=1}^{r} W_k * D_k$ を計算する. □

このように得られる保証区間は, 大規模な計算に対しても実用的に意味のある精度を与える. また, 区間演算の手間は通常の浮動小数点演算の定数倍でできるので, その計算の手間は f の計算に必要な手間の定数倍である.

問. 入力変数の幅 $\delta x_j \geq 0$ $(j = 1, \ldots, n)$ が与えられたとき, $F \equiv \{f(y_1, \ldots, y_n) \mid y_j \in [x_j - \delta x_j, x_j + \delta x_j]\ (j = 1, \ldots, n)\}$ を含む区間を計算するには, 上の "関数値の保証区間の計算法" をどのように変更すればよいか.

[略解: 入力変数 x_j に対しても区間変数 W_j' を導入する．算法中の [$W(\partial^+ a)$ $:= W(\partial^+ a) + \Phi(a) \cdot W_k$ $(a \in \delta^- v_k)$] において，$\partial^+ a$ が x_j のときには，$W(\partial^+ a) = W_j'$ と約束する．$L = \Sigma_{j=1}^n W_j' * [-\delta x_j, \delta x_j] + \Sigma_{k=1}^r W_k * D_k$ とすると，$F \subseteq [\tilde{f}, \tilde{f}] - L$.]

4 非線形方程式の解法

$x \in \mathbb{R}$ に関する非線形方程式

$$f(x) = 0 \tag{4.1}$$

を考える．解 $x = \alpha$ を有限回の四則演算で求めるのは一般に不可能であり，通常は，適当な初期値 $x^{(0)}$ から出発してなんらかの反復法

$$x^{(\nu+1)} = x^{(\nu)} + \Delta x^{(\nu)} \quad (\nu = 0, 1, \ldots) \tag{4.2}$$

によって α に収束する近似解列 $\{x^{(\nu)}\}$ を数値的に生成する．

近似解列 $\{x^{(\nu)}\}$ は，有限桁の演算によって計算される $f(x)$ の値などに基づいて生成されるので，必然的に丸め誤差を含み，その結果，数学的な意味で $\lim_{\nu \to \infty} x^{(\nu)} = \alpha$ となるようにはできない．使用する計算機などによって自然に定まる丸め誤差限界 (→式 (2.9)) を δf とするとき，$|f(x)|$ が δf 程度以下ならば $f(x) = 0$ と見なさざるをえない．すなわち，数値計算の立場からは，方程式 (4.1) を解くというのは $|f(x)| \leq \delta f$ を満たす x を求めることであると考え，

$$|f(x^{(\nu)})| \leq \delta f \tag{4.3}$$

を反復法の停止規則にするのが健全である．

数値計算で用いられる反復法 (4.2) は

$$x^{(\nu+1)} = g(x^{(\nu)}) \quad (\nu = 0, 1, \ldots) \tag{4.4}$$

とかけるものが多い ($g(x)$ は $D \subseteq \mathbb{R}$ 上で定義されているとする)．g の不動点 ($\alpha = g(\alpha)$ である α) が $f(x) = 0$ の解に対応する．

次の定理 (Brouwer の定理の 1 次元の場合) は緩い仮定のもとで不動点の存在を保証する．

定理 4.1 $g(x)$ が $\mathbb{R}$ の閉区間 D で定義された連続関数で，$x \in D$ に対し $g(x) \in D$ ならば，g は D 内に不動点をもつ. □

最も基本的な非線形方程式の反復解法は，

$$x^{(\nu+1)} = x^{(\nu)} - f(x^{(\nu)})/f'(x^{(\nu)}) \tag{4.5}$$

で定義される Newton 法である (Newton-Raphson 法とも呼ばれる). これは，式 (4.2) における修正量 $\Delta x^{(\nu)}$ を，$f(x)$ の $x = x^{(\nu)}$ 付近での線形近似に基づいて定めたものである. 実際，$f(x) \sim f(x^{(\nu)}) + f'(x^{(\nu)})(x - x^{(\nu)})$ と近似されるが，$x = \alpha$ とおくと左辺は 0 であるから $\alpha \sim x^{(\nu)} - f'(x^{(\nu)})^{-1} f(x^{(\nu)})$ となる. この右辺を新たな近似解としたものが式 (4.5) である. 幾何学的には，曲線 $y = f(x)$ の点 $(x^{(\nu)}, f(x^{(\nu)}))$ における接線と x 軸との交点の x 座標が $x^{(\nu+1)}$ である.

例 4.1 3 次方程式 $f(x) = x^3 - 3x + 3 = 0$ は唯一の実根 $\alpha_1 = -2.1038034$ をもつ. $x^{(0)} = -3$ として Newton 法を適用すると，下表のように速やかに $x^{(\nu)} \to \alpha_1$ となる. 仮数部 16 進 6 桁 (単精度) 計算では $f(x)$ $(x \sim \alpha_1)$ の計算精度 δf は 10^{-5} 程度であるから，$x^{(4)} = -2.10380$, $f(x^{(4)}) = -9.5 \times 10^{-7}$ で反復を停止する.

ν	$x^{(\nu)}$	$f(x^{(\nu)})$
0	−3.00000	−1.50 E+1
1	−2.37500	−3.27 E+0
2	−2.14001	−3.80 E−1
3	−2.10458	−8.01 E−3
4	−2.10380	−9.54 E−7

□

Newton 法においては導関数値を計算する必要があるが，f が多項式

$$f(x) = \sum_{k=0}^{n} a_{n-k} x^k \tag{4.6}$$

の場合には，その関数値や (高階の) 導関数値，すなわち

$$f(x + \zeta) = \sum_{k=0}^{n} c_{n-k} \zeta^k \tag{4.7}$$

の展開係数 $c_{n-k} = f^{(k)}(x)/k!$ を以下のように効率よく計算することができる．実際の計算では上添字 $^{(l)}$ を無視して，一つの 1 次元配列 $b[k]$ $(k = 0, 1, \ldots, n)$ だけを用いる．

[組立除法 (Horner 法)]

1°: $b^{(-1)}[k] := a_k \quad (k = 0, 1, \ldots, n)$

2°: $l = 0, 1, \ldots, n$ に対して順次 [$b^{(l)}[0] := a_0$; $b^{(l)}[k] := b^{(l)}[k-1]x + b^{(l-1)}[k] \quad (k = 1, \ldots, n-l)$] を計算する． □

算法の終了時に $b[n-k] = b^{(k)}[n-k] = c_{n-k}$ $(k = 0, 1, \ldots, n)$ であり，乗算回数は全体で $n(n+1)/2$ になる．(算法の 2° で $l = 0, 1, \ldots, m$ とすれば，$b[n-k] = c_{n-k}$ $(k = 0, 1, \ldots, m)$ であることに注意．)

例 4.2 3 次式 $f(x) = x^3 - 3x + 3$ を $x = -2.11$ の周りで (4.7) の形に展開する．10 進 3 桁 4 捨 5 入計算の組立除法は次のようになる．

	$k=0$	$k=1$	$k=2$	$k=3$
$b^{(-1)}[k]$	1.00	0.00	-3.00	3.00
$(b^{(0)}[k-1]x)$		-2.11	4.45	-3.06
$b^{(0)}[k]$	1.00	-2.11	1.45	$\underline{-0.06 = c_3}$
$(b^{(1)}[k-1]x)$		-2.11	8.90	
$b^{(1)}[k]$	1.00	-4.22	$\underline{10.4 = c_2}$	
$(b^{(2)}[k-1]x)$		-2.11		
$b^{(2)}[k]$	1.00	$\underline{-6.33 = c_1}$		

従って，$f(\zeta - 2.11) = \zeta^3 - 6.33\zeta^2 + 10.4\zeta - 0.06$ である． □

5 方程式の解の精度保証

非線形方程式 $f(x) = 0$ の解 $x = \alpha$ は，一般には反復法の収束先として (いわば無限回の計算ステップを経て) 計算される (→前節) ので，前述の精度保証手法をそのまま適用することはできない．そこで，Brouwer の不動点定理 (定理 4.1) のような数学的結果を援用して，関数値の精度を近似解の精度に転換する．

連続関数 $g(x)$ による反復法 (4.4): $x^{(\nu+1)} = g(x^{(\nu)})$ に対応して，区間 $X \subseteq \mathbb{R}$ を区間に写す区間写像 $G(X)$ による区間反復法

$$X^{(\nu+1)} = G(X^{(\nu)}) \qquad (\nu = 0, 1, \ldots) \tag{5.1}$$

を考える．ここで，$G(X) \supseteq \{g(x) \mid x \in X\}$ とする．つぎの補題 (ときに Moore テストと呼ばれる) は，区間反復法による精度保証の基礎を与える．

補題 5.1 (1) $G(X) \cap X = \emptyset$ ならば g は X に不動点をもたない．
(2) $G(X) \subseteq X$ ならば，ある $\alpha \in X$ が存在して $g(\alpha) = \alpha$.

[証明] (1) $g(\alpha) = \alpha \in X$ とすると $g(\alpha) \in G(X)$. (2) 任意の $x \in X$ に対して $g(x) \in G(X) \subseteq X$ なので， Brouwer の不動点定理 (定理 4.1) が適用できる． □

区間写像 $G(X)$ には種々のものが提案されているが，ここでは， Newton 法の反復式 (4.5) に Brouwer の定理を適用して，解の存在を保証してみよう．

例 5.1 例 4.1に引き続き， 3 次方程式 $f(x) = x^3 - 3x + 3$ を考え， $\alpha_1 = -2.1038034$ の近似値 $x^{(0)} = -2.11$ が得られたとする． $X = [x^{(0)} - \delta x, x^{(0)} + \delta x] = [-2.14, -2.08]$ $(\delta x = 0.03)$， $g(x) = x - f(x)/f'(x)$ (式 (4.5) 参照) に対して， $g(X) \subseteq X$ (ただし $g(X) = \{g(x) \mid x \in X\}$) が成り立つことを前述の手法で確かめよう．このとき， Brouwer の不動点定理により， $\alpha_1 \in X$ が結論できる．

$g(x)$ の計算過程は， $f(x), f'(x)$ を組立除法で計算した後に， $v = b^{(0)}[3]/b^{(1)}[2]$， $g = x - v$ とするものとする．すなわち， $b_k = b^{(-1)}[k]$ $(k = 0, 1, \ldots, 3)$ とおくと， $b_0 = 1$, $b_1 = 0$, $b_2 = -3$, $b_1 = 3$, $v_1 = x * b_0$, $v_2 = v_1 + b_1$, $v_3 = x * v_2$, $v_4 = v_3 + b_2$, $v_5 = x * v_4$, v_6 $(= f = b^{(0)}[3])$ $= v_5 + b_3$, $v_7 = x * b_0$, v_8 $(= b^{(1)}[1])$ $= v_7 + v_2$, $v_9 = x * v_8$, v_{10} $(= f' = b^{(1)}[2])$ $= v_9 + v_4$, v_{11} $(= f'/f = v)$ $= v_6/v_{10}$, v_{12} $(= g)$ $= x - v_{11}$.

まず，例 4.2のように， 10 進 3 桁 4 捨 5 入計算の組立除法で $\tilde{b}^{(0)}[3] = -0.06$, $\tilde{b}^{(1)}[2] = -10.4$ を計算し，さらに， $\tilde{v} = -0.00577$， $\tilde{g} = \tilde{g}(x^{(0)}) = -2.10$ を得る．次に， $X = [-2.14, -2.08]$ に対して組立除法をおこなうと

	$k=0$	$k=1$	$k=2$	$k=3$
$b^{(-1)}[k]$	$[1.00, 1.00]$	$[0.00, 0.00]$	$[-3.00, -3.00]$	$[3.00, 3.00]$
		$[-2.14, -2.08]$	$[4.32, 4.58]$	$[-3.39, -2.74]$
$b^{(0)}[k]$	$[1.00, 1.00]$	$[-2.14, -2.08]$	$[1.32, 1.58]$	$[-0.39, 0.26]$
		$[-2.14, -2.08]$	$[8.65, 9.16]$	
$b^{(1)}[k]$	$[1.00, 1.00]$	$[-4.28, -4.16]$	$[9.97, 10.8]$	

となる．これより，$g(X) \subseteq [-2.14, -2.08] - [-0.390, 0.260]/[9.97, 10.8] \subseteq [-2.14, -2.08] - [-0.0392, 0.0261] \subseteq [-2.17, -2.04]$ と計算されるが，$g(X) \subseteq X$ を主張することはできない．

そこで，前述の手法 (入力区間幅が 0 でない場合) を用いて，さらに精密な $g(X)$ の評価を導こう．x, v, g および $b^{(l)}[k]$ に対応する区間変数 $W(x)$, $W(v)$, $W(g)$, $W(b^{(l)}[k])$ を用意し，$W(g) = [1.00, 1.00]$ から始めて前述の算法を適用して 10 進 3 桁で計算する．$W(v) = [-1.00, -1.00]$,

	$k=1$	$k=2$	$k=3$
$W(b^{(0)}[k])$	$[-0.477, -0.382]$	$[0.188, 0.220]$	$[-0.101, -0.0925]$
$W(b^{(1)}[k])$	$[-0.00565, 0.00848]$	$[-0.00396, 0.00264]$	

(この表の右下から順に計算される) となる．$W(x)$ は，$v_{12}, v_9, v_7, v_5, v_3, v_1$ からの寄与 $\Delta_{12} = [1, 1]*[1.00, 1.00] = [1.00, 1.00]$, $\Delta_9 = [-4.28, -4.16]*[-0.00396, 0.00264] \subseteq [-0.0113, 0.0170]$, $\Delta_7 = [1, 1]*[-0.00565, 0.00848] = [-0.00565, 0.00848]$, $\Delta_5 = [1.32, 1.58]*[-0.101, -0.0925] \subseteq [-0.160, -0.122]$, $\Delta_3 = [-2.14, -2.08]*[0.188, 0.220] \subseteq [-0.471, -0.391]$, $\Delta_1 = [1.00, 1.00]*[-0.477, -0.382] = [-0.477, -0.382]$, の和であり，$W(x) = (((((\Delta_{12} + \Delta_9) + \Delta_7) + \Delta_5) + \Delta_3) + \Delta_1) \subseteq [-0.126, 0.135]$ と計算される．なお，この部分和は，順に，$[0.988, 1.02]$, $[0.982, 1.03]$, $[0.822, 0.908]$, $[0.351, 0.517]$, $[-0.126, 0.135]$ である．

この計算結果より，$g(X) - \tilde{g} \subseteq W(x) * [-\delta x, \delta x] + \Sigma_k W_k * D_k$ において $|W(x) * [-\delta x, \delta x]| \leq 0.00405$, $\Sigma_k |W_k * D_k| \leq 0.0130$ となる (発生誤差 δ_k は最下位有効桁の半分である) ので，$|g(X) - (-2.10)| \leq 0.0171$，すな

わち， $g(X) \subseteq [-2.10 - 0.0171, -2.10 + 0.0171] \subseteq X$ が成り立つ．従って， Brouwer の不動点定理を適用できる． □

微分方程式や積分方程式のような関数方程式を数値的に解く場合には，まず，微分を差分で置き換えたり積分を有限和で置き換えたりすることによって元の問題を有限次元の非線形方程式で近似し，さらに，その方程式の解を近似的に求めることになる．本節で述べた手法は，この第二段階に関する精度保証を与えるものであるが，さらにその考え方を発展させて，第一段階で生じる離散化誤差をも含めた精度保証手法が提案されている．

「保証書」付き数値計算法 (補足資料)

京都大学数理解析研究所　室田一雄

1　数値の表現

計算機の中では $\pi = 3.14159265358979\cdots$ のような無限桁の実数をそのまま扱うことはできないので，これを適当な有限桁の数，いわゆる浮動小数点数で近似的に表現する．浮動小数点数の体系は計算機ごとに決っており，“β 進 n 桁”と呼ばれる体系においては，$x_{\mathrm{f}} = 0$ または

$$x_{\mathrm{f}} = \pm f \times \beta^m, \quad f = \frac{x_1}{\beta} + \frac{x_2}{\beta^2} + \cdots + \frac{x_n}{\beta^n} \tag{1.1}$$

の形の数 x_{f} が表現可能な数である．ここで，x_k $(k = 1,\ldots,n)$ は $1 \le x_1 \le \beta - 1$, $0 \le x_k \le \beta - 1$ $(k = 2,\ldots,n)$ を満たす整数で[1]，m は予め決まっている範囲 $L \le m \le U$ 内の整数である．f を x_{f} の仮数部 (mantissa) あるいは小数部 (fraction)，m を x_{f} の指数部 (exponent) という．零でない浮動小数点数の絶対値の最大値 $F_{\max}$，最小値 $F_{\min}$ は

$$F_{\max} = \beta^U(1 - \beta^{-n}), \quad F_{\min} = \beta^{L-1}$$

で与えられる．

浮動小数点体系 は 4 つのパラメタ，すなわち，基数 (あるいは底) β, 桁数 n, 最小指数 L, 最大指数 M で特徴付けられる．大型計算機の単精度では $\beta = 16$, $n = 6$, $L = -64$, $U = 63$, 倍精度では $\beta = 16$, $n = 14$, $L = -64$, $U = 63$, また，パーソナルコンピューターやワークステーションの単精度では $\beta = 2$, $n = 24$, $L = -126$, $U = 127$, 倍精度では $\beta = 2$, $n = 53$, $L = -1022$, $U = 1023$ としているものが多い．

実数 x はそれに近い浮動小数点数 x_{f} で近似的に表現される．これを丸めと呼び，丸めによって生じる誤差を丸め誤差と呼ぶ．$F_{\min} \le |x| \le F_{\max}$ のとき，$x_{\mathrm{f}}^{(1)} \le |x| \le x_{\mathrm{f}}^{(2)}$ を満たす浮動小数点数で最大のもの $x_{\mathrm{f}}^{(1)}$ と最小

[1] このように $x_1 \neq 0$ を課す表現を正規化された表現とよぶ．$x_1 = 0$ を許す体系もあるが，ここでは正規化を仮定する．パーソナルコンピューターやワークステーションでは，正規化しない表現を用いているものもある．詳細は D. Stevenson: “A proposed standard for binary floating-point arithmetic,” *Computer*, Vol. 14 (1981), March, pp.51–62, などを参照されたい．

のもの $x_{\mathrm{f}}^{(2)}$ がとれるので，適当に定めた規則に従って $x_{\mathrm{f}}^{(k)}$ $(k=1,2)$ のうちのどちらかを $|x|$ の近似値に採用し，それに x の符号をつけて x_{f} とするのが普通である．$x_{\mathrm{f}}^{(1)}=|x_{\mathrm{f}}|$ とする方式を切捨て，$x_{\mathrm{f}}^{(k)}$ $(k=1,2)$ のうち $|x|$ に近い方を $|x_{\mathrm{f}}|$ とする方式を“四捨五入”($(\beta/2-1)$ 捨 $(\beta/2)$ 入) と呼ぶ．[2]

実数 x が $F_{\min}\le|x|\le F_{\max}$ の範囲にあるとき，

$$x_{\mathrm{f}}=x(1+\varepsilon_x),\qquad |\varepsilon_x|\le\varepsilon_{\mathrm{M}} \tag{1.2}$$

が成り立つ．ここで，ε_{M} は浮動小数点体系と丸めの方式で決まる数であり，**マシンエプシロン** (machine epsilon) と呼ばれる，相対誤差限界を表す重要な指標である．β 進 n 桁で切捨てのとき $\varepsilon_{\mathrm{M}}=\beta^{1-n}$，“四捨五入”のとき $\varepsilon_{\mathrm{M}}=\beta^{1-n}/2$ であり，いずれの場合も，ε_{M} は，計算機上で $1+\varepsilon$ を計算して浮動小数点数に丸めた結果が 1 より大きくなるような ε の最小値に等しい．

実数 x の絶対値が $F_{\min}\le|x|\le F_{\max}$ の範囲にないときは，これを近似できる浮動小数点数はないと考えるのが自然である．とくに $|x|>F_{\max}$ のとき**オーバーフロー** (overflow)，$|x|<F_{\min}$ のとき**アンダーフロー** (underflow) と呼ぶ．[3]

現在ほとんどの計算機で採用されている浮動小数点体系にも指数部の範囲が狭いなどの不便な点があり，新しい数値表現法が提案されている．

2　誤差の発生と伝播

数値計算の誤差は，極限を有限で打ち切ったり近似式を用いたりする計算法そのものから生じる誤差と，実数値を有限桁の浮動小数点数の形で扱うために生じる誤差の二つに大別される．後者は丸め誤差，前者は離散化誤差，打ち切り誤差，公式誤差，理論誤差，近似誤差などの名で呼ばれる．ここでは，四則演算などの基本演算における丸め誤差を考察する．

二つの実数 x,y に対する基本演算 ϕ の結果 $z=\phi(x,y)$ を計算機上で求めようとすると，まず，x,y が浮動小数点数 $x_{\mathrm{f}},y_{\mathrm{f}}$ に丸められてから演算

[2]厳密にいうと，いわゆる“四捨五入”($(\beta/2-1)$ 捨 $(\beta/2)$ 入) は β が偶数のときに定義され，その結果は $|x|$ に近い方に一致する．

[3]実際の計算機では，$|x|<F_{\min}$ のときに $x_{\mathrm{f}}=0$ とするものが多い．

ϕ が近似的に施され，最後に，その結果が浮動小数点数 z_{f} に丸められる．このように，数学世界の $z=\phi(x,y)$ に対応して，計算機上では $z_{\mathrm{f}}=\tilde{\phi}(x_{\mathrm{f}},y_{\mathrm{f}})$ となる．

$$z_{\mathrm{f}}-z=[\tilde{\phi}(x_{\mathrm{f}},y_{\mathrm{f}})-\phi(x_{\mathrm{f}},y_{\mathrm{f}})]+[\phi(x_{\mathrm{f}},y_{\mathrm{f}})-\phi(x,y)] \tag{2.3}$$

の右辺第1項

$$\delta=\tilde{\phi}(x_{\mathrm{f}},y_{\mathrm{f}})-\phi(x_{\mathrm{f}},y_{\mathrm{f}}) \tag{2.4}$$

は，x,y 自身が浮動小数点数である場合にも生じる誤差であり，演算 ϕ における発生誤差 と呼ばれることがある．現在の計算機においては，通常，

$$|\delta|\le c\,\varepsilon_{\mathrm{M}}\,|\tilde{\phi}(x_{\mathrm{f}},y_{\mathrm{f}})| \tag{2.5}$$

が成り立っているとしてよい (c は演算に応じて決まる数で，$c\le 2$ ととれることが多い)．ただし，(2.3) の第2項も含めた $|z_{\mathrm{f}}-z|\le c\,\varepsilon_{\mathrm{M}}\,|z|$ や $|z_{\mathrm{f}}-z|\le c\,\varepsilon_{\mathrm{M}}\,|z_{\mathrm{f}}|$ のような形の相対精度を保証する不等式は必ずしも成立しないことに注意．とくに，x と y がほぼ等しいときには，x_{f} と y_{f} の仮数部の大部分が打ち消し合って z_{f} の精度が著しく落ちる．この現象は桁落ちと呼ばれている．

例 **2.1** 2次方程式 $ax^2+bx+c=0$ $(a\neq 0)$ の解 (根) は

$$\alpha_+=\frac{-b+\sqrt{b^2-4ac}}{2a},\qquad \alpha_-=\frac{-b-\sqrt{b^2-4ac}}{2a}$$

という公式で与えられる．$a=1, b=200, c=1$ のとき，真の値は $\alpha_+=-0.0050001250062\cdots$, $\alpha_-=-199.99499987\cdots$ である．

ところが，2進23桁 (0捨1入), $\varepsilon_{\mathrm{M}}=2^{-23}=1.192\cdots\times 10^{-7}$, で計算した結果の出力は，$\tilde{\alpha}_+=0.0000000\mathrm{E}{+}00$, $\tilde{\alpha}_-=-199.99501$ となる (ここで，E は10のべきを表す)．α_+ の分子の減算において桁落ちが起きているため，その精度が著しく低下している．

$\alpha_\pm$ を精度良く求めるには，

$b\ge 0$ のとき: $\alpha_+=\dfrac{-2c}{b+\sqrt{b^2-4ac}}$, $\alpha_-=\dfrac{-b-\sqrt{b^2-4ac}}{2a}$;

$b<0$ のとき: $\alpha_+=\dfrac{-b+\sqrt{b^2-4ac}}{2a}$, $\alpha_-=\dfrac{-2c}{b-\sqrt{b^2-4ac}}$

のように式変形 (分子の有理化) をしてから数値を代入すればよい. これにより桁落ちを回避できる. 実際, この計算式を用いると, 2 進 23 桁 (0 捨 1 入) 計算で, $\tilde{\alpha}_+ = -5.0001251\mathrm{E}-03$, $\tilde{\alpha}_- = -199.99501$ が得られる.

この例は, 数学の世界で正しい公式でも丸め誤差のある数値計算の世界では別種の注意が必要であることを端的に示している. □

絶対値の大きな浮動小数点数 x_{f} に絶対値の小さな数 y_{f} を足す計算 $x_{\mathrm{f}} + y_{\mathrm{f}}$ においては, 演算結果を浮動小数点数に丸めたものが x_{f} に等しくなって, y_{f} の情報が反映されないことが起こる. これを情報落ちあるいは積み残しなどと呼ぶことがある.

例 **2.2** 無限級数 $S = \sum_{k=1}^{\infty} 1/k^2 = \pi^2/6 = 1.6449340668\cdots$ の部分和 $S_n = \sum_{k=1}^{n} 1/k^2$ を

$$S_n = \left(\cdots\left(\left(\frac{1}{1^2} + \frac{1}{2^2}\right) + \frac{1}{3^2}\right) + \cdots + \frac{1}{n^2}\right)$$

の形で 2 進 23 桁 (0 捨 1 入) で計算すると, $n \geq N = 4096$ に対して $S_n = 1.6447253$ となって増加しなくなる. これに対し,

$$S_n = \left(\cdots\left(\left(\frac{1}{n^2} + \frac{1}{(n-1)^2}\right) + \frac{1}{(n-2)^2}\right) + \cdots + \frac{1}{1^2}\right)$$

として小さい項から足すようにすると, 精度のよい S の近似値を得ることができる. 例えば, $S_{10000} = 1.6448341$, $S_{100000} = 1.6449241$, $S_{1000000} = 1.6449330$ と計算される. □

以上