1　数値解析

数値解析の目標は, 一口に言えば, 入力値 $x_1, \ldots, x_n$ が与えられた時にある関数値 $y = f(x_1, \ldots, x_n)$ の値を実際に求める事であると言えよう.

中学校で**文字**を使った数式を導入して以来, 我々は**文字**を使った計算に慣れ親しんできた. これは極めて強力な手法である. 例えば, 面積 s の円の半径 r は $s = \pi r^2$ を解いて

$$r = \sqrt{\frac{s}{\pi}}$$

であると直ちに導ける. これは面積 s や半径 r の値に依らず成り立ち, 例えば, ある範囲の面積と半径の値の一覧を表にする等という表現よりも格段に簡潔でしかも明快である.

しかし, **文字**を使った計算にも避けられない限界がある. 我々は $10\mathrm{cm}^2$ の円を書くのに半径は何 cm にすべきかと聞かれたとき, はたと困ることになる[i]. 実際のところ中学以降で習ってきた数学ではこの事を問題にしない. 何故なら**文字**を使った計算の強力さは実際の”計算”の難しさを切り離すことによってこそ得られるものであったからだ[ii].

しかしながら切り離されたからと言って”計算”の困難さはなんら減じてはいない. 例の場合では問題点は 2 点である. 一つは $\sqrt{}$, 即ち平方根の計算が出来ないこと. もう一つは円周率 π の値が分からない[iii]ことである.

これは, 本質的な問題である. 大半の人は, 何かの数値を計算する勉強は小学校で分数の四則演算の書き取りをやったのが最後であろう. ここに至って我々は反省しなければならない, つまるところ**我々が直接出来る計算は有理数の四則演算のみ**なのである[iv].

もちろん直接計算ができないからと言って手も足も出ないわけではない. π のかわりに 3.14 と言う近似値を使って計算すれば良く, 平方根の計算にも妥当な近似計算が存在する. この様な間接的な計算には常に**誤差**がついて回る. 多くの場合我々は真の値を終に計算することができず, 妥当な近似値で手を打たねばならない. 特に, ”近似値”は主観的に近い値というだけではなく何らかの意味で客観的に認められる概念であるべきであり, **誤差**の評価は数値解析の重要な位置を占めている.

---

[i]「電卓を使え」と言うのは現実的であるが今の場合はナンセンスな答えである. 我々は, まさしく, 電卓の行っている処理について学ぼうとしているのである.

[ii]中学校で初めて文字を使った数式を導入される時の戸惑い「実際の値が判らないこのような計算に意味があるのか?」と言うのはある意味正しい疑問である. 実際に値を計算する手法を無くしては控えめに言っても実用性は薄い.

[iii]もしその値を”知っていた”としても計算どころか書き記すのにさえ無限の時間が必要である.

[iv]単に勉強した事がないといった問題ではなく, 本当に手段が無い.

2　誤差

2.1　誤差と近似値

定義 1. 近似値 x と真値 a に対し, その差, $e := x - a$ を a の **(絶対) 誤差**, 真値に対する差の比率, $e_R := \dfrac{x-a}{a}$ を a の**相対誤差**と言う.

fact 1. 絶対誤差と相対誤差の間には

$$e_R = \frac{e}{x-e} = \frac{e}{x} + \frac{e^2}{x(x-e)} = \frac{e}{x} + O\left((\frac{e}{x})^2\right)$$

という関係があるので, 通常 $e << x$ であるから e_R は $\dfrac{e}{x}$ とほぼ一致する.

上の様に定義したが, 実際のところ大抵の場合は, 我々は近似値と誤差の大雑把な評価しか得られない[v]. 例えば円周率 π の近似値 3.14 の誤差 $e := \pi - 3.14$ は正確には判らない, しかしながら $0 < e < 0.01$ である事は判る. 多くの場合, 誤差そのものではなく誤差の絶対値の上からの評価について議論が行われる.

定義 2. 真値 x 近似値 a に対し, $|e| \leq \varepsilon$ 或いは $|e_R| \leq \varepsilon_R$ を満たす ε, ε_R を各々a の**許容 (絶対) 誤差**, **許容相対誤差**と言う.

例:3.14 は許容 (絶対) 誤差 0.01 の π の近似値である.

2.2　誤差の由来

誤差は発生原因により次の 3 種に分類される;

- 入力誤差
- 近似誤差
- 丸め誤差

である. 例を追って説明する.

例 1. $10\mathrm{cm}^2$ の面積を持つ円を描く事を考える. 勿論半径は $\sqrt{\frac{10}{\pi}}\mathrm{cm}$ である.

先ず π の値が真には不明であるので 3.14 で近似する. この $e_{\mathrm{in}} := 10/3.14 - 10/\pi$ が**入力誤差**である. 次に平方根が計算できない為[vi], 4 を中心にした平方根の 3 次までのテイラー展開

$$\sqrt{x} \doteqdot 2 + \frac{1}{4}(x-4) - \frac{1}{64}(x-4)^2 + \frac{1}{512}(x-4)^3 =: f(x)$$

[v] 正確な誤差がわかるなら真値が得られる!

[vi] 実はコンパスを用いれば平方根を作図する事が可能であるが数値計算の例であるのであくまで計算による方法を使う事にする.

で平方根を近似する[vii]．この近似の為，例え入力が正確であっても $e := \sqrt{10/\pi} - f(10/\pi)$ の誤差を生じる．これが**近似誤差**である．通常この誤差は個々の計算法 f の設定時に理論的に評価される．

結果として

$$f(\frac{10}{3.14}) = 2 + \frac{1}{4}(\frac{10}{3.14} - 4) - \frac{1}{64}(\frac{10}{3.14} - 4)^2 + \frac{1}{512}(\frac{10}{3.14} - 4)^3$$

が実際に計算可能な半径の近似値となる．

最終的な誤差は

$$\sqrt{\frac{10}{\pi}} - f(\frac{10}{3.14}) = e + f(\frac{10}{\pi}) - f(\frac{10}{\pi} + e_{\text{in}})$$

となる．e_{in} の値に対し $f(\frac{10}{\pi}) - f(\frac{10}{\pi} + e_{\text{in}})$ がどうなるかを調べる事を計算法 f に関する前進誤差解析，$f(\frac{10}{\pi}) - f(\frac{10}{\pi} + e_{\text{in}})$ の値を特定の値に収めるためには e_{in} の値をどうすべきかを調べる事を後退誤差解析と言う．

上記の $f(10/3.14)$ を厳密に手計算した場合，以上で誤差は尽きるのだが，時間がかかる上に計算ミスをする可能性もかなり高くなり，実際上あまり現実的ではない．故に計算機を用いるのだがこの時に計算機の特質による**丸め誤差**を生じる．これについては次節以降に詳しく論じる．

3　計算機

数値計算で計算機の使用はまず不可欠である．計算機を用いれば非常に高速にしかも正確に計算が可能である．然しながら計算機を用いた計算には特有の誤差が生じる．これは，基本的に「計算機は**有理数**を扱う事ができない」[viii]と言うことに由来する．

3.1　2 進数

計算機の内部では全ての対象がスイッチの ON/OFF で表されているというのは周知の事実であろう．数値も同様であり，いわゆる 2 進数として表現される．例えば 10 進数の 1037 は

$$1037 = 1\times2^{10}+0\times2^{9}+0\times2^{8}+0\times2^{7}+0\times2^{6}+0\times2^{5}+0\times2^{4}+1\times2^{3}+1\times2^{2}+0\times2^{1}+1\times2^{0}$$

であるから

100 0000 1101

[vii] この場合テイラー展開はあまり良い近似法とは言えないのだが，例示としてわかり易いので用いた．

[viii] 本質的に不可能と言うわけではなく，表面上は現在普及している仕様で扱えないというだけである．しかし，後述する整数での場合と同様に，有理数を扱うためには可変なメモリを用いる必要があり効率的でない為，極特殊な場合を除き有理数として扱う事はない．

と言った具合である.

明らかに任意の自然数は**適当な桁の** 2 進数として表現できる. 即ち, どんな自然数も適当な数のスイッチの ON/OFF で表現出来る.

3.2 計算機の数値表現 (整数型)

しかしながら, 計算機において前項の様にして**自然数**を直接扱う事は殆ど無い.

一つの数を表すのに必要なスイッチの数 (メモリの大きさ) が不定であると言うのは, 処理上煩雑な問題を引き起こす為[ix], 普通, 自然数では無く**定まった桁の** 2 進数を扱う. 多くの場合, 一つの数値を表す為に 32 個のスイッチが用いられる[x]. 例えば 10 進数の 1037 は

$$0000\ 0000\ 0000\ 0000\ 0000\ 0100\ 0000\ 1101$$

である. つまり, $2^32 = 4294967296$ 種類の数, 数学的にいえば $\mathbf{Z}/2^{32}\mathbf{Z}$ 上の元を扱うわけであり, この様な物を**整数型**と呼ぶ. この場合 0 から 4294967295 まで (符号無し整数型) 或いは -2147483648 から 2147483647 まで (整数型) の数値が扱える[xi].

fact 2. ところで上の 2 進数による記述はいかにも煩雑である. そこで, 各区切毎に 16 進数[xii]に変換した

$$0\ 0\ 0\ 0\ 0\ 4\ 0\ D$$

と言う表現を用いる事が多い. これは 1037 を 16 進数で表現した物と等しい. また逆の操作も常に可能である. この様に 2 進数と 16 進数は相互の変換が容易であり, また 16 進数を用いた方が遥かに記述が簡潔であるので, 計算機関係では 16 進数を使うことが多い.

[ix] 単に操作が煩雑になるというだけでなく, 本質的な計算量自体に影響を及ぼす.

[x] このスイッチの数は個々の計算機の仕様に依存する. やがて 64bit やそれ以上の計算機が一般的になることもありうる. とは言え, 現在の計算機はほぼ全て 32 個である.

[xi] 通常 $\mathbf{Z}$ と商群 $\mathbf{Z}/2^{32}\mathbf{Z}$ の対応 (代表元の選び方) は

$$[0], [1], [2], [3], \ldots, [2^{32}-2], [2^{32}-1]$$

の様に選ぶのが自然であり, これが符号無し整数型である. 一方で代表元として

$$[-2^{32}], [1-2^{32}], \ldots, [-3], [-2], [-1]$$

を取ることも可能で, 更にはこれらを併せて

$$[0], [1], [2], [3], \ldots, [2^{31}-1], [-2^{31}], [1-2^{31}], \ldots, [-3], [-2], [-1]$$

ともできる. 実はこれが (符号有り) 整数型であり. 整数型の最大の数に 1 を加算した時に負の最大の数になる理由である. この様な負の数の表現方法を 2 の補数表現と言う.

[xii] 16 種類の文字 (通常 $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F$) を使って数値を表現する方法

3.3 科学的記数法

さて, 前述の整数型では 2^{32} 以上の数を扱う事ができない. 出来るだけ少ないスイッチ (この場合数字と言い換えることが出来る) で大きな数を表す工夫が必要である. 日常においてこの様な目的には**科学的記数法**を用いる. 例えば, 1 光年 $9,460,000,000,000,000\mathrm{m}$ を表すのに $9.46 \times 10^{15}\mathrm{m}$ と表す方法である. **科学的記数法**は非常に小さな数を表す時にも又有用である. 例えば, 電子の電荷 $-0.00000000000000000016\mathrm{C}$ は $-1.6 \times 10^{-19}\mathrm{C}$ と表現できる.

即ち, 数値を (10 進数の) 小数で表し, 0 以外の数が現れる最大の桁の右まで小数点を移動し, 適当な桁で丸めた物である. 初めの小数部分を**仮数部**, 10 の肩の数を**指数部**と呼ぶ. 同等の事を 2 **進数で**行ったものが**浮動小数点型**であると言える.

3.4 2 進小数

3.1 において整数の 2 進展開を見たが, 2 の肩の数を負の方向に拡張する事で実数が表現できる. 実際, 任意の実数 x について

$$x = \sum_{k \leq m} s_k 2^k$$

を満たす 0, 1 値の列 $s_m, s_{m-1}, \ldots, s_1, s_0, s_{-1}, s_{-2}, \ldots$ が存在する.

$$s_m s_{m-1} \cdots s_1 s_0 . s_{-1} s_{-2}, \ldots$$

を x の 2 進小数展開と言う. 例えば円周率 π は

$$11.001001000011111101101 0 \cdots$$

である. 特にある桁以降常に 0 が並ぶ物を 2 進有限小数と言う.

計算機で扱うのはこの 2 進有限小数である訳だが, 特に注意すべき事として, 普通の (10 進) 有限小数が 2 進有限小数であるとは限らない. 実際, (10 進有限小数の)0.1 は 2 進有限小数では**ない**[xiii]

特に

$$10 \text{ 進有限小数} \subset 2 \text{ 進有限小数} \subset \text{有理数}$$

である.

3.5 計算機の数値表現 (浮動小数点型)

例えば π を 2 進の科学的記数法で表せば,

$$\boxed{1.10010010000111111011010} \cdots \times 2^{\boxed{1}}$$

[xiii] 2 に変換すると $0.000110001100011 \cdots$ と言う循環小数になる.

となる.(四角で囲まれた数 は 2 進を用いた表現)

基数は常に 2 であるから, 実際は指数部と仮数部を記述すれば数を表現できる. 整数型の時と同じ理由で指数部と仮数部を表す桁数は固定して考え, 計算機の内部ではこれを

$$0 \qquad 0000\ 0001 \qquad 110\ 0100\ 1000\ 0111\ 1110\ 1101$$

と言う風[xiv]に表現している.(初めの 0 は符号が正であることを, 続く大きな塊が指数部を, 残りが仮数部を表現している) この様な数値の表現方法を**浮動小数点型**と言う. また, 同様の表現方法でより多くのスイッチ (大抵は 64 個[xv]) を使ったものを倍精度浮動小数点型と言う. 最近の計算機はハードウェア的に**倍精度浮動小数点型**の計算を効率化しているものが多く, 通常計算機で数値解析をするときはこの型を用いる.

3.6 入力誤差

浮動小数点型で表現可能な数を考えてみよう. 標準的な環境での double 型変数が表現可能な数は

$$\mathbb{F} := \{\frac{\alpha}{2^{53}} \times 2^{\beta} \mid -2^{53} < \alpha < 2^{53}, -1022 \leq \beta \leq 1023\}$$

である. $\mathbb{F}$ は有限集合であり, 計算機で”実数”を扱う時は通常扱いたい実数 x

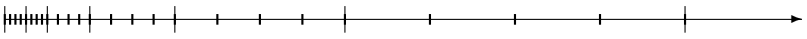

図 1: 浮動小数点型・正規数と非正規数

に最も近い $\mathbb{F}$ の値 $\widetilde{x}$ を使う. つまり, 計算機を用いた場合, 例え入力値の真値がわかっていたとしても殆どの場合**入力誤差**が生じる[xvi]. 一般に仮数部の第一位の数字は 1 である[xvii]から相対誤差が

$$|e_R| = \frac{|x - \widetilde{x}|}{|x|} \leq \frac{2^{-53} \times 2^{\beta}}{|x|} \leq \frac{2^{-53} \times 2^{\beta}}{1 \times 2^{\beta}} = 2^{-53}$$

となる[xviii].

[xiv] 実際は少々異なる. 手元のコンピューターでは

$$0 \qquad 1000\ 0000 \qquad 100\ 1001\ 0000\ 1111\ 1101\ 1011$$

であった. これは, 指数部の表現にオフセット表現を用い. 仮数部の表現にいわゆるケチ表現を用いてるためであり, 本質的には本文の様に理解しておいて問題は無いだろう.

[xv] ここでの事情も整数型のときと同様である. 但し, 符号ビット, 指数部, 仮数部の並び等には実際にバリエーションが存在するので詳細が必要になったならば仕様をる必要がある.

[xvi] 勿論扱う数字が偶然 $\mathbb{F}$ の元であるときは誤差は無い. また,2^{2^1023} より大きな数やはこの限りでない.

[xvii] もし 0 ならば仮数部を 2 倍して指数部を -1 してる筈である

[xviii] 2^{1023} より大きな数や 2^{-1024} より大きな数はこの限りでない.

3.7　丸め誤差

$\mathbb{F}$ は加減乗除について閉じていない. (これは自明でない, 即ち計算結果の大きさが $\max\mathbb{F}$ を超えない場合にも良く起きる. 例えば x と y の指数部が異なればほとんどの場合 $x+y\notin\mathbb{F}$ である) この為, 浮動小数点型では実数として加減乗除した値を最も近い $\mathbb{F}$ に丸めた物を加減乗除の結果として用いる. これにより計算を行う度に誤差が発生する. これを**丸め誤差**と呼ぶ.

4　誤差の伝播

入力誤差に対する最終的な誤差への影響を評価する事を, 即ち前進誤差評価を考える場合, 勿論本格的な評価は各々の計算法に依存するが, 全ての計算は加減乗除により構成されているから, それらの計算における入力誤差と結果の誤差の関係がこの種の評価の前提となっている事がわかる.

端的に言って加減法は絶対誤差を保存し, 乗除法は相対誤差を保存する. 実際, 真値 a,b に対する近似値 x,y の絶対誤差, 相対誤差を各々$e_a,e_{b\mathrm{R}},e_a,e_{b\mathrm{R}}$ とすれば

$$e_{a+b}=x+y-(a+b)=(a+e_a)+(b+e_b)-(a+b)=e_a+e_b$$

$$e_{a\times b\mathrm{R}}=\frac{xy-ab}{ab}=\frac{(a+e_a)(b+e_b)-ab}{ab}=e_{a\mathrm{R}}+e_{b\mathrm{R}}+e_{a\mathrm{R}}e_{b\mathrm{R}}$$

$$e_{a/b\mathrm{R}}=\frac{x/y-a/b}{a/b}=\frac{(a+e_a)/(b+e_b)-a/b}{a/b}=\frac{1}{1+e_{b\mathrm{R}}}(e_{a\mathrm{R}}+e_{b\mathrm{R}})$$

とくに異符号の加法, 同符号の減法の場合には注意が必要で, 相対誤差が

$$e_{a+b\mathrm{R}}=\frac{a}{a+b}e_{a\mathrm{R}}+\frac{b}{a+b}e_{b\mathrm{R}}$$

であるから $a+b$ が a 又は b と比較して小さい場合, 結果の相対誤差が著しく大きくなる. この様な現象を**桁落ち**と言いう.