2005年10月26日作成
2011年5月9日更新

４．数値計算　Numerical calculation

４－１　数の表現　Expression of numbers

この節ではコンピュータの内部でどのように数が表現されているかについて，さらに詳しく紹介します。「数」には整数，有理数，実数，複素数などがあり，それに応じてコンピュータ内部の表現のしかたも変わります。ただしコンピュータで扱える数には制限があることに注意しなければならない場合があります。コンピュータでは扱える整数の絶対値の大きさに制限がありますし，厳密な意味では無理数を扱うことができません。

４－１－１　整数　Integer

整数を表現するために何ビット bit を使用するかで扱える範囲が変わります。代表的なビット数は 8, 16, 32 bit です。

1 byte = 8 bit, char （C言語）BYTE（FORTRAN) 0〜255, -128〜127

1 word = 16 bit int (C言語) INTEGER (FORTRAN) -32768〜32767

unsgined int (C言語) 0〜65535

1 double word = 32 bit, longint （C 言語）-2147483648〜2147483647

（参考）

音声，音楽

CD-audio 16 bit, 44.1 kbps，2ch；74分

DVD-audio, Hi-Res audio 24 bit, 192 kbps；4.7GB

静止画

Facebook プロフィール画像 160 × 160〜2.6万画素，1 Byte/色 ⇒ 約 8 kB

Instagram 1080 × 1080〜100万画素，1 Byte/色 ⇒ 約 3 MB

iPhone 7s+ (5.5 inch) 1080×1920〜200万画素，1 Byte/色 ⇒ 約 6 MB

iPad Pro 9.7 inch 2048×1536〜300万画素，1 Byte/色 ⇒ 約 9 MB

写真，印刷

4800×1200dpi=ピクセルサイズ横 0.005 mm，縦 0.02 mm,

４色 1bit/色，L判（127mm×89mm），ピクセル数 約 1億,

⇒ 約 5 MB

動画，アニメーション

デジタルビデオ 640×480 1 Byte/色 フレームレート 15 fps ⇒ 約 110 Mbps

HDTV（ハイビジョン） 1920×1080 フレームレート 30 fps ⇒ 約 1.5 Gbps

4K テレビ 3840×2160 フレームレート 30 fps ⇒ 約 6 Gbps

４－１－２　実数，IEEE 規格　Real number, IEEE standard

コンピュータで整数の表現のしかたはほとんど決まっています。ところが，コンピュータ内部で実数をどのように表現するかには異なる規格があります。現在の主流は IEEE 規格（アイ・トリプル・イーきかく）となっています。なお，コンピュータを使って複素数の計算をすることもできるのですが，複素数をどのように表現するかについては，一般的な規格はないと考えて良いでしょう。

４－１－２－１　単精度実数　Single-precision real number

1985 年くらいまでは 32 bit = 4 byte で一つの実数を表現するのが普通でした。4 byte で表現される実数のことを単精度実数と呼びます。32 bit が以下のように対応づけられます。

0	1	2	3 ... 7	8	9	10	11 ... 30	31
s	e_1	e_2	$e_3 \ldots e_7$	e_8	f_2	f_3	$f_4 \cdots f_{23}$	f_{24}
符号	指数部 + 127				仮数部			

「指数部 + 127」の値が $1 \leq$ (指数部+127) ≤ 254 の範囲の場合には

$$x = \pm\left(1 + f_2 \times \frac{1}{2} + f_3 \times \frac{1}{4} + \cdots + f_{24} \times \frac{1}{2^{23}}\right) \times 2^{\left(e_1 \times 2^7 + e_2 \times 2^6 + \cdots + e_8 - 127\right)}$$

という数を表すこととされています。このような形式で表される数のことは正規数と呼ばれます。

また，$(e_1e_2\cdots e_8)=(00000000)=0$ のときには

$$x=\pm\left(f_2\times\frac{1}{2}+f_3\times\frac{1}{4}+\cdots+f_{24}\times\frac{1}{2^{23}}\right)\times 2^{-126}$$

という数を表すこととされています。この形式で表される数のことは非正規数と呼ばれます。

指数部 $(e_1e_2\cdots e_8)=(11111111)=255$ のときは特別で

$s=0$, $(f_2f_3\cdots f_{24})=(00\cdots0)$ のとき $x=\infty$

$s=1$, $(f_2f_3\cdots f_{24})=(00\cdots0)$ のとき $x=-\infty$

$(f_2f_3\cdots f_{24})\neq(00\cdots0)$ のとき $x=$NAN (not a number；非数)

となります。

正規数では指数部が –126 から 127 の値をとることができるので，単精度実数の正規数の中で絶対値の最も小さい数は

$$1\times 2^{-126}=1.175494\times 10^{-38}$$

絶対値の最も大きい数は

$$\left(1+\frac{1}{2}+\frac{1}{4}+\cdots+\frac{1}{2^{23}}\right)\times 2^{127}=3.40282347\times 10^{38}$$

となります。

4－1－2－2　倍精度実数　Double-precision real number

最近は 64 bit = 8 byte で一つの実数を表現するのがふつうです。

0	1	2...10	11	12	13...62	63
s	e_1	$e_2 \ldots e_{10}$	e_{11}	f_2	$f_3 \ldots f_{52}$	f_{53}
符号	指数部 + 1023			仮数部		

「指数部 + 1023」 $(e_1e_2\cdots e_{11})$ について，

$1=(00000000001)\le(e_1e_2\cdots e_{11})\le(11111111110)=1026$ のときには

$$x=(-1)^s\times\left(1+f_2\times\frac{1}{2}+f_3\times\frac{1}{4}+\cdots+f_{53}\times\frac{1}{2^{52}}\right)\times 2^{\left(e_1\times 2^{10}+e_2\times 2^9+\cdots+e_{11}-1023\right)}$$

という数を表すこととされ，この範囲の数は正規数と呼ばれます。

指数部 $(e_1e_2\cdots e_{11})=(00000000000)=0$ のときは

$$x=\pm\left(f_2\times\frac{1}{2}+f_3\times\frac{1}{4}+\cdots+f_{53}\times\frac{1}{2^{52}}\right)\times 2^{-2046}$$

という数を表すこととされています。このような数値は非正規数とよばれます。

指数部 $(e_1e_2\cdots e_{11})=(11111111111)=2047$ のときは特別で

$s=0$, $(f_2f_3\cdots f_{53})=(00\cdots0)$ のとき $x=\infty$

$s=1$, $(f_2f_3\cdots f_{53})=(00\cdots0)$ のとき $x=-\infty$

$(f_2f_3\cdots f_{53})\neq(00\cdots0)$ のとき $x=$NAN (not a number；非数)

正規数では指数部が -1022 から 1023 の値をとることができますから，倍精度型の正規数の中で絶対値の最も小さい数は

$$1\times 2^{-1022}=2.2250738585072009\times 10^{-308}$$

絶対値の最も大きい数は

$$\left(1+\frac{1}{2}+\frac{1}{4}+\cdots+\frac{1}{2^{52}}\right)\times 2^{1023}=1.7976931348623157\times 10^{308}$$

となります。

コラム：円周率の憶え方

円周率を小数点以下３０桁まで憶える有名な憶え方があります。

π=3.14159265358979323846264338327

「産医師異国に向かふ。産後厄なく，産婦御社に，虫散々闇に鳴く。」

倍精度実数では 3.14159265358979

単精度実数では 3.141593

ですから，倍精度実数で表現できるよりかなり多くの桁数を簡単に憶えられます。

４－２　初等関数と特殊関数の計算

Calculation of elementary and special functions

初等関数 elementary functions とは，以下の関数を指します。

指数関数 exponential function : $\exp(x) = \mathrm{e}^{x}$,

対数関数 logarithmic function : $\ln x$, $\log x$, $\log_{\mathrm{e}} x$

三角関数 trigonometric functions

正弦関数 sine function : $\sin x$,

余弦関数 cosine function : $\cos x$,

正接関数 tangent function : $\tan x$

余割関数 cosecant function : $\operatorname{cosec} x = \dfrac{1}{\sin x}$ （あるいは $\csc x$ とも），

正割関数 secant function : $\sec x = \dfrac{1}{\cos x}$,

余接関数 cotangent function : $\cot x = \dfrac{1}{\tan x}$

逆三角関数 inverse trigonometric functions

逆正弦関数 arcsine function : $\arcsin x$, $\sin^{-1} x$,

逆余弦関数 arccosine function : $\arccos x$, $\cos^{-1} x$,

逆正接関数 arctangent function : $\arctan x$, $\tan^{-1} x$,

逆余割関数 arccosecant function : $\operatorname{arccosec} x$, $\operatorname{cosec}^{-1} x$,

逆正割関数 arcsecant function : $\operatorname{arcsec} x$, $\sec^{-1} x$,

逆余接関数 arccotangent function : $\operatorname{arccot} x$, $\cot^{-1} x$,

双曲線関数 hyperbolic functions

双曲正弦関数 hyperbolic sine function : $\sinh x = \dfrac{e^{x} - e^{-x}}{2}$,

双曲余弦関数 hyperbolic cosine function : $\cosh x = \dfrac{e^{x} + e^{-x}}{2}$,

双曲正接関数 hyperbolic tangent function : $\tanh x = \dfrac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$,

双曲余割関数 hyperbolic cosecant function : $\operatorname{cosech} x = \dfrac{2}{e^{x} - e^{-x}}$,

双曲正割関数 hyperbolic secant function : $\operatorname{sech} x = \dfrac{2}{e^{x} + e^{-x}}$,

双曲余接関数 hyperbolic cotangent function : $\coth x = \dfrac{e^x + e^{-x}}{e^x - e^{-x}}$,

逆双曲線関数 inverse hyperbolic functions

逆双曲正弦関数 inverse hyperbolic sine function :

$$\operatorname{arcsinh} x = \sinh^{-1} x = \ln\left(x \pm \sqrt{x^2+1}\right),$$

$$e^{2y} - 2xe^y - 1 = 0 \Leftrightarrow e^y = x \pm \sqrt{x^2+1}$$

逆双曲余弦関数 inverse hyperbolic cosine function :

$$\operatorname{arccosh} x = \cosh^{-1} x = \ln\left(x \pm \sqrt{x^2-1}\right),$$

$$e^{2y} - 2xe^y + 1 = 0 \Leftrightarrow e^y = x \pm \sqrt{x^2-1}$$

逆双曲正接関数 inverse hyperbolic tangent function :

$$\operatorname{arctanh} x = \tanh^{-1} x = \frac{1}{2}\ln\frac{1+x}{1-x},$$

$$x\left(e^{2y}+1\right) = e^{2y} - 1 \Leftrightarrow e^{2y} = \frac{1+x}{1-x}$$

逆双曲余割関数 inverse hyperbolic cosecant function :

$$\operatorname{arccosech} x = \operatorname{cosech}^{-1} x = \ln\left(\frac{1}{x} \pm \sqrt{\frac{1}{x^2}+1}\right),$$

逆双曲正割関数 inverse hyperbolic secant function :

$$\operatorname{arcsech} x = \operatorname{sech}^{-1} x = \ln\left(\frac{1}{x} \pm \sqrt{\frac{1}{x^2}-1}\right),$$

逆双曲余接関数 inverse hyperbolic cotangent function :

$$\operatorname{arccoth} x = \coth^{-1} x = \frac{1}{2}\ln\frac{1-x}{1+x},$$

たとえば $y = a^x$ の形の数式は， $y = \exp(x \ln a)$ の形で計算されます。

コンピュータで初等関数を計算する場合，べき級数展開か連分数展開を使って計算します。ガンマ関数やベッセル関数など，初等関数で表せない関数を特殊関数と呼びますが，やはりべき級数展開か連分数展開を使って計算される場合が多いようです。つまり，コンピュータを使って計算する場合には，初等関数と特殊関数の間に本質的な区別はありません。

４－２－１　（べき）級数展開
Power-series expansion

指数関数 $\exp(x)$ は x が小さい場合には

$$\exp(x)=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}+\cdots+\frac{x^N}{N!}$$

の形で計算できます。たとえば自然対数の底

$$\exp(1)=2.71828182845905\cdots$$

について,

$$1+1=2$$

$$1+1+\frac{1}{2!}=2.5$$

$$1+1+\frac{1}{2!}+\frac{1}{3!}=2.666\cdots$$

$$1+1+\frac{1}{2!}+\frac{1}{3!}+\frac{1}{4!}=2.708333\cdots$$

$$1+1+\frac{1}{2!}+\frac{1}{3!}+\frac{1}{4!}+\frac{1}{5!}=2.716666\cdots$$

$$\vdots$$

$$1+1+\frac{1}{2!}+\frac{1}{3!}+\cdots+\frac{1}{12!}=2.71828182828617\cdots$$

一般的には

$$f(x)=f(0)+xf'(0)+\frac{x^2}{2!}f''(0)+\frac{x^3}{3!}f'''(0)+\frac{x^4}{4!}f^{(4)}(0)+\cdots+\frac{x^N}{N!}f^{(N)}(0)=\sum_{j=0}^{N}a_j x^j$$

ただし，$a_j=\dfrac{f^{(j)}(0)}{j!}$ という形に書けます。

４－２－２　連分数展開　Continued-fraction expansion

変数 x の値が大きいときでも，「原理的には級数展開が収束する」ということは多いのですが，収束が遅いときに実際の計算は困難です。このような場合に連分数展開という方法が用いられることがあります。

正接関数 $\tan x$ は以下のような形式で表すことができます。

$$\tan x=\cfrac{x}{1-\cfrac{x^2}{3-\cfrac{x^2}{5-\cfrac{x^2}{7-\cdots}}}}$$

たとえば

$\tan 1 = 1.5574077246549\cdots$

に対して,

$$\frac{1}{1-\frac{1}{3}} = 1.5, \quad \frac{1}{1-\frac{1}{3-\frac{1}{5}}} = 1.555\cdots, \quad \frac{1}{1-\frac{1}{3-\frac{1}{5-\frac{1}{7}}}} = 1.55737\cdots$$

などとなります。このような形式で関数の値を計算する方法を連分数展開と呼びます。

級数展開の形式から連分数展開の形式を導く方法は知られています。この方法はユークリッドの互除法に似ています。級数展開が

$$y = a_0 + a_1 x + a_2 x^2 + \cdots$$

と表されるとき,

$$y = a_0 + a_1 x\left[1 + (a_2/a_1)x + (a_3/a_1)x^2 + \cdots\right]$$

$$= a_0 + \frac{a_1 x}{\dfrac{1}{1 + (a_2/a_1)x + (a_3/a_1)x^2 + \cdots}} = a_0 + \frac{a_1 x}{1 - \dfrac{(a_2/a_1)x + (a_3/a_1)x^2 + \cdots}{1 + (a_2/a_1)x + (a_3/a_1)x^2 + \cdots}}$$

$$= a_0 + \frac{a_1 x}{1 - \dfrac{(a_2/a_1)x\left[1 + (a_3/a_2)x + (a_4/a_2)x^2 + \cdots\right]}{1 + (a_2/a_1)x + (a_3/a_1)x^2 + \cdots}}$$

$$= a_0 + \frac{a_1 x}{1 - \dfrac{(a_2/a_1)x}{\dfrac{1 + (a_2/a_1)x + (a_3/a_1)x^2 + \cdots}{1 + (a_3/a_2)x + (a_4/a_2)x^2 + \cdots}}}$$

$$= a_0 + \frac{a_1 x}{1 - \dfrac{(a_2/a_1)x}{1 - \dfrac{(a_3/a_2 - a_2/a_1)x + (a_4/a_2 - a_3/a_1)x^2 + \cdots}{1 + (a_3/a_2)x + (a_4/a_2)x^2 + \cdots}}}$$

$$= a_0 + \frac{a_1 x}{1 - \dfrac{(a_2/a_1)x}{1 - \dfrac{(a_3/a_2 - a_2/a_1)x\left\{1 + \left[(a_4/a_2 - a_3/a_1)/(a_3/a_2 - a_2/a_1)\right]x + \cdots\right\}}{1 + (a_3/a_2)x + (a_4/a_2)x^2 + \cdots}}}$$

$$= a_0 + \frac{a_1 x}{1 - \dfrac{(a_2/a_1)x}{1 - \dfrac{(a_3/a_2 - a_2/a_1)x}{\dfrac{1 + (a_3/a_2)x + (a_4/a_2)x^2 + \cdots}{1 + \left[(a_4/a_2 - a_3/a_1)/(a_3/a_2 - a_2/a_1)\right]x + \cdots}}}}$$

$$= a_0 + \cfrac{a_1 x}{1 - \cfrac{(a_2/a_1)x}{1 - \cfrac{(a_3/a_2 - a_2/a_1)x}{1 - \cfrac{\left[(a_4/a_2 - a_3/a_1)/(a_3/a_2 - a_2/a_1) - a_3/a_2\right]x + \cdots}{1 + \left[(a_4/a_2 - a_3/a_1)/(a_3/a_2 - a_2/a_1)\right]x + \cdots}}}}$$

のように，順々に変形できます。

実際の関数の計算で，級数展開を使うのが良いか，連分数展開を使うのが良いかは簡単には予測できません。

4－3　数値微分　Numerical differential

高校数学で「微分」のところでは，色々な公式を憶えましたが，コンピュータを使った「数値微分」で微分を計算することにすれば，憶えなければいけない公式は，数値微分微分の定義式

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

だけです。

たとえば，「関数 $f(x) = 2^x$ について，$f'(3)$ の値を求めなさい」という問題をどのように解きますか？高校までの数学で教わった通りにするには，

$$f'(x) = \frac{\mathrm{d}}{\mathrm{d}x} \exp(x \ln 2) = (\ln 2) \exp(x \ln 2) = 2^x \ln 2$$

ですから，

$$f'(3) = 2^3 \ln 2 = 5.5451774444795\cdots$$

となります。

しかし，コンピュータを使えるのだったら，微分の定義式の右辺をそのまま利用できます。たとえば $h = 0.0000001$ として，

$$\frac{f(3.0000001) - f(3)}{0.0000001} = \frac{2^{3.0000001} - 2^3}{0.0000001} = 5.5451776148629\cdots$$

のように計算します。このような計算のしかたを，数値微分 numerical differential と呼びます。この例はまだ簡単な方ですが，関数がもっと複雑な場合に，数値微分を使うことにすれば式の上では微分を解かなくても良いので，ずっと楽になる場合がかなり多いのです。

4－4　数値積分　Numerical integral

数値計算によって定積分

$$S=\int_a^b f(x)\,\mathrm{d}x$$

を計算する方法は数値積分 numerical integral あるいは求積法 quadrature と呼ばれます。

一般的に

$$S\cong(b-a)\left[w_1 f\left(a+(b-a)x_1\right)+w_2 f\left(a+(b-a)x_2\right)+\cdots+w_N f\left(a+(b-a)x_N\right)\right]$$

という形を使います。

一番単純なのは中点法 mid-point method とよばれる方法で,

$$S=\frac{b-a}{N}\left[f\left(a+\frac{0.5(b-a)}{N}\right)+f\left(a+\frac{1.5(b-a)}{N}\right)+\cdots+f\left(a+\frac{(N-0.5)(b-a)}{N}\right)\right]$$

という式で計算するものです。

他にも台形法やシンプソン法，ガウスの求積法などいろいろな計算方法があるのですが，中点法ははじめに知っておくべき方法です。台形法とシンプソン法は，積分領域の端点で関数が発散するような場合には使えませんが，中点法はそのような場合にも使えます。

4－5　数値計算の技術　Tips for numerical calculations

4－5－1　ベキ乗の計算　Calculation of power

たとえば x^{12} を計算したいというときに，指数関数と対数関数を使って $\exp(12\ln x)$ とするのは間違いではないですが，計算に無駄が多いだけでなく，計算誤差の影響で正確な結果が得られない場合もあります。単純にはかけ算を 11 回すれば正解が得られるように思うかもしれませんが，この場合

$$t=x\cdot x$$

$$u=t\cdot t$$

$$v=u\cdot u$$

$$y=v\cdot u$$

とすれば，４回のかけ算で正確な結果が得られます。

4－5－2　級数の計算　Calculation of power series

たとえば

$$f(x)=a_0+a_1x+a_2x^2+a_3x^3+a_4x^4$$

の形の計算をこのまますると，10回のかけ算と４回の足し算が必要のように見えますが，

$$f(x)=a_0+x\left(a_1+x\left(a_2+x\left(a_3+xa_4\right)\right)\right)$$

の形で計算すれば４回のかけ算と４回の足し算で済みます。

また

$$f(x)=a_0+a_1x+\frac{a_2x^2}{2}+\frac{a_3x^3}{3!}+\frac{a_4x^4}{4!}$$

のような形を，

$$f(x)=a_0+x\left(a_1+\frac{x}{2}\left(a_2+\frac{x}{3}\left(a_3+\frac{a_4x}{4}\right)\right)\right)$$

と変形して計算する例なども良く知られています。

この計算の方法はホーナー Horner の方法と呼ばれます。

４－５－３　二次方程式の解　Solution of quadratic equation

2次方程式：$x^2+2bx+c=0$ の解は

$$x=-b\pm\sqrt{b^2-c}$$

ですが，$b>0$，$c>0$ で c の値が小さいとき，$-b+\sqrt{b^2-c}$ の方の解は「桁落ち」により精度が下がってしまいます。例えば，$b=0.500005$，$c=0.00001$ のとき，解は $x=1$，0.00001 となるはずですが，倍精度実数を使っても公式通りの計算のしかただと

$$-b+\sqrt{b^2-c}=-0.00001000000000001$$

という少し間違った解になってしまいます。この場合，有理化の逆の変形の仕方をして

$$-b+\sqrt{b^2-c}=-\frac{c}{b+\sqrt{b^2-c}}$$

と変形してから計算すれば桁落ちが起きなくなり –0.00001 という正解が得られます。

４－６　この節の最後に

コンピュータでふつうに扱う数値は，たいして正確なものではありません。また，扱える数は有理数のうちの限られた部分ですから，それ以外の有理数や無理数は近似的にしか扱えません。

“Mathematica” などのいわゆる「数式処理ソフト」は別として，コンピュータは四則演算（足し算，引き算，かけ算，わり算）しかできません。初等関数の計算や微分や積分も数

値を使った四則演算で近似的な計算ができるだけなので，計算結果には必ず誤差がついています。これは実験データに必ず誤差がともなっているのと似ています。

プログラミング以前の問題で，計算のしかたを工夫すれば近似であるとしても非常に正確な結果を得られる場合もありますが，逆に計算のしかたが悪いために計算時間が非常に長くなったり大きな誤差がともなってしまう場合もあります。

数値計算はもちろん学問の対象になりうるものなのですが，現実には，むしろ技術的な側面が強いと言えます。