

## 4. 数値計算 Numerical calculation

**数値計算** ニュメリカル キャルキュレイション numerical calculation と言うと、慣れていない人には立派なことをしているように聞こえるかもしれませんが、その意味は、文字どおり「数値を使った計算」と言うことであり、中学校で「文字を数値の代わりに使うこと」**代数計算** アルジェブラ algebraを習う前、小学校の算数でしていたことと同じことです。

小学校の算数の問題でも高学年になれば、「 $\square + 1 = 6$  です。 $\square$ には何が入りますか？」というように、 $\square$ のような記号を「抽象的な数の概念」として扱うことがあるので、コンピュータを使った数値計算は、小学校高学年の算数より低いレベルの話と考えるのも良いでしょう。コンピュータの計算システムは、基本的に数値計算しかすることができません。(ただし、コンピュータ上で動作する「数式処理ソフト」は、ある程度の代数計算をすることができます。)

### 4-1 初等函数と特殊函数 Elementary and special functions

#### 4-1-1 初等函数 Elementary functions

**初等函数** かんすう elementary functions とは、以下の26種類の函数を指します。

**指数函数** exponential function :  $\exp(x) = e^x$  ([補足 4.1.A](#))

**対数函数** logarithmic function :  $\ln x, \log x, \log_e x$  ([補足 4.1.B](#))

**三角函数** trigonometric functions

**正弦函数** sine function :  $\sin x$

**余弦函数** cosine function :  $\cos x$

**正接函数** tangent function :  $\tan x$

**余割函数** cosecant function :  $\csc x = \frac{1}{\sin x}$  (あるいは  $\operatorname{cosec} x$ とも) ([補足 4.1.C](#))

**正割函数** secant function :  $\sec x = \frac{1}{\cos x}$

**余接函数** cotangent function :  $\cot x = \frac{1}{\tan x}$

**逆三角函数** inverse trigonometric functions ([補足 4.1.D](#))

**逆正弦函数** arcsine function :  $\arcsin x, \sin^{-1} x,$

**逆余弦函数** arccosine function :  $\arccos x, \cos^{-1} x,$

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

逆余割函数 arccosecant function :  $\operatorname{arccsc} x$ ,  $\csc^{-1} x$ , ( $\operatorname{arccosec} x$ ,  $\operatorname{cosec}^{-1} x$ )

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

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

双曲線函数 hyperbolic functions ([補足 4.1.E](#))

双曲線正弦函数 hyperbolic sine function :  $\sinh x = \frac{e^x - e^{-x}}{2}$ ,

双曲線余弦函数 hyperbolic cosine function :  $\cosh x = \frac{e^x + e^{-x}}{2}$ ,

双曲線正接函数 hyperbolic tangent function :  $\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ ,

双曲線余割函数 hyperbolic cosecant function :  $\operatorname{csch} x = \frac{2}{e^x - e^{-x}}$ ,

(あるいは  $\operatorname{cosech} x$ とも)

双曲線正割函数 hyperbolic secant function :  $\operatorname{sech} x = \frac{2}{e^x + e^{-x}}$ ,

双曲線余接函数 hyperbolic cotangent function :  $\operatorname{coth} x = \frac{e^x + e^{-x}}{e^x - e^{-x}}$ ,

逆双曲線函数 inverse hyperbolic functions

逆双曲線正弦函数 inverse hyperbolic sine function :

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

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

逆双曲線余弦函数 inverse hyperbolic cosine function : ([補足 4.1.F](#))

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

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

逆双曲線正接函数 inverse hyperbolic tangent function :

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

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

逆双曲線余割函数 inverse hyperbolic cosecant function :

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

**逆双曲線正割函数** inverse hyperbolic secant function :

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

**逆双曲線余接函数** inverse hyperbolic cotangent function :

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

これらの初等函数には、科学者・技術者の社会では「すでに定義されたものとして使って良い」という国際的なルールがあります。 ([補足 4.1.G](#))

初等函数を数式表現の中で用いるときに、紛らわしくない場合には、例えば  $\sin(\theta)$  でなく  $\sin \theta$  のように、変数を囲む括弧 ( ) の記号を使わず表現するのが普通ですが、指数函数の場合には、紛らわしくなくても  $\exp x$  とは書かず  $\exp(x)$  と括弧を付けることも普通です。

高校までに習う他の函数、たとえば  $y = a^x$  ( $x$  が非整数) の形の数式は、一般的なコンピュータの数値計算処理システムでは、初等函数を組み合わせて  $y = \exp(x \ln a)$  の形に翻訳されて計算されます。

コンピュータで初等函数を計算する場合、「<sup>べき</sup>**冪級数展開** power-series expansion」か「**連分数展開** continued-fraction expansion」という方法の使われることが普通です。

## 4-1-2 特殊函数 Special functions

「ガンマ函数」や「ベッセル函数」など、初等函数で表せない函数が**特殊函数** special functions と呼ばれます。例えば**ガンマ函数** Gamma function  $\Gamma(x)$  は、

$$\Gamma(x) \equiv \int_0^{\infty} t^{x-1} e^{-t} dt$$

という積分の形で定義されます (他の定義の仕方もあります)。「**初等函数**は断り無しに使って良いが、**特殊函数**を使う場合には定義式も示す」という暗黙のルールもありました。現代では、ガンマ函数  $\Gamma(x)$  のように良く知られている特殊函数の場合には、定義式が省略される場合もあります。

筆者がコンピュータを利用した「学習」や「実験データ解析」のために実際に利用したことのある特殊函数に、以下のものがあります。

**誤差函数・ガンマ函数・ベータ函数**など

$$\text{誤差函数 error function : } \operatorname{erf}(x) \equiv \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

補誤差関数 complementary error function :  $\operatorname{erfc}(x) \equiv 1 - \operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} e^{-t^2} dt$

尺度化複素補誤差関数 (ファデーバ関数)

scaled complex complementary error function (Faddeeva function) :

$$w(z) \equiv e^{-z^2} \operatorname{erfc}(-iz) = e^{-z^2} \left( 1 + \frac{2i}{\sqrt{\pi}} \int_0^z e^{-t^2} dt \right)$$

フォークト関数 Voigt function :

$$V(x; \sigma, w) \equiv \frac{1}{\sqrt{2\pi}\sigma} \Re \left[ w \left( \frac{x + iw}{\sqrt{2}\sigma} \right) \right] = \int_{-\infty}^{\infty} G(y; \sigma) L(x - y; w) dy$$

ただし  $G(x; \sigma) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$  はガウス型関数 Gaussian function,

$$L(x; w) = \frac{1}{\pi w} \left( 1 + \frac{x^2}{w^2} \right)^{-1} \text{ はローレンツ型関数 Lorentzian function,}$$

$\Re(z)$  は複素数  $z$  の実部 (real part) を表すとする。

フレネル余弦積分 Fresnel cosine :  $C(x) \equiv \frac{1}{2\pi} \int_0^x \frac{\cos t}{\sqrt{t}} dt$

フレネル正弦積分 Fresnel sine :  $S(x) \equiv \frac{1}{2\pi} \int_0^x \frac{\sin t}{\sqrt{t}} dt$

ディガンマ関数 digamma function :  $\psi(x) \equiv \frac{d}{dx} \ln \Gamma(x)$

正則第一種不完全ガンマ関数

regularized incomplete Gamma function of the first kind :

$$P(a, x) \equiv \frac{1}{\Gamma(a)} \int_0^x t^{a-1} e^{-t} dt$$

正則第二種不完全ガンマ関数

regularized incomplete Gamma function of the second kind :

$$Q(a, x) \equiv \frac{1}{\Gamma(a)} \int_x^{\infty} t^{a-1} e^{-t} dt$$

ベータ関数 Beta function :  $B(a, b) \equiv \int_0^1 t^{a-1} (1-t)^{b-1} dt$

不完全ベータ関数 incomplete Beta function :  $B(x; a, b) \equiv \int_0^x t^{a-1} (1-t)^{b-1} dt$

正則不完全ベータ関数 regularized incomplete Beta function :  $I(x; a, b) \equiv \frac{B(x; a, b)}{B(a, b)}$

ベッセル関数 Bessel functions など

第1種ベッセル関数 Bessel function of the first kind :

$$J_{\alpha}(x) \equiv \sum_{m=0}^{\infty} \frac{(-1)^m}{m! \Gamma(m + \alpha + 1)} \left(\frac{x}{2}\right)^{2m+\alpha}$$

第2種ベッセル関数 (ノイマン関数)

Bessel function of the second kind (Neumann functions) :

$$Y_{\alpha}(x) \equiv \frac{J_{\alpha}(x) \cos(\alpha\pi) - J_{-\alpha}(x)}{\sin(\alpha\pi)}$$

第1種変形ベッセル関数 modified Bessel function of the first kind :

$$I_{\alpha}(x) \equiv \sum_{m=0}^{\infty} \frac{1}{m! \Gamma(m + \alpha + 1)} \left(\frac{x}{2}\right)^{2m+\alpha}$$

第2種変形ベッセル関数 (マクドナルド関数)

modified Bessel function of the second kind (Macdonald functions)

$$K_{\alpha}(x) \equiv \frac{\pi}{2} \frac{I_{\alpha}(x) - I_{-\alpha}(x)}{\sin(\alpha\pi)}$$

超幾何関数 Hyper geometric functions など

超幾何関数 Hyper geometric functions :

$${}_2F_1(a, b; c; z) \equiv \sum_{n=0}^{\infty} \frac{(a)_n (b)_n}{(c)_n n!} z^n$$

ただし,  $(q)_n$  はポツホハマー記号 (Pochhammer symbol) で,

$$(q)_n \equiv \begin{cases} 1 & [n = 0] \\ q(q+1)\cdots(q+n-1) & [n > 0] \end{cases}$$

とする。

一般超幾何関数 generalized hypergeometric functions :

$${}_pF_q(a_1, \dots, a_p; b_1, \dots, b_q; z) \equiv \sum_{n=0}^{\infty} \frac{(a_1)_n \cdots (a_p)_n z^n}{(b_1)_n \cdots (b_q)_n n!}$$

ただし  $(x)_n$  はポツホハマー記号 Pochhammer symbol

他の特殊関数の多くも「高校数学では解けない」積分や無限級数の形で定義されます。ところが、数値計算ライブラリでは、特殊関数も<sup>べき</sup>冪級数展開か連分数展開を使って計算されることは、初等関数と同じことです。

コンピュータの数値計算ライブラリを使って関数の計算をするときに、初等関数と特殊関数に特別な区別はありません。不定積分や定積分を解くために、公式集 (e.g. プルドニコ

フら、[1991](#), [1992](#)) や数式処理ソフト、web サイトなどで調べて、「特殊関数で表現される解」が得られるときに、その特殊関数を計算できる数値計算ライブラリが存在すれば、それを使います。そのような「数式の解き方」をするときに、「解が特殊関数を使って表現される問題」であっても、「解が初等関数の組み合わせで表現される『高校数学で解ける』タイプの問題」との間に実質的に違いはありません。

## 4-2 級数展開と連分数展開

### power-series and continued-fraction expansions

#### 4-2-1 冪級数展開 power-series expansion

指数関数  $\exp(x)$  は  $x$  が小さい値の場合には

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

の形で計算できます。たとえば自然対数の底

$$\exp(1) = 2.718\,281\,828\,459\,05\dots$$

について、級数展開で解を求めると、

$$1 + 1 = 2$$

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

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

$$1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} = 2.708\,333\dots$$

$$1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \frac{1}{5!} = 2.716\,666\dots$$

...

$$1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{12!} = 2.718\,281\,828\,286\,17\dots$$

のようになり、級数の項の数を増やすと急速に正しい値に近づきます。

級数展開は、一般的に

$$\begin{aligned} f(x) &= f(0) + xf'(0) + \frac{x^2}{2} f''(0) + \frac{x^3}{3!} f'''(0) + \frac{x^4}{4!} f^{(4)}(0) + \dots + \frac{x^N}{N!} f^{(N)}(0) \\ &= \sum_{j=0}^N a_j x^j \end{aligned} \tag{4.2.1}$$

$$a_j = \frac{1}{j!} \left[ \frac{d^j f(x)}{dx^j} \right]_{x=0} = \frac{1}{j!} f^{(j)}(0) \quad (4.2.2)$$

という形に書けます。このように0を中心とした**テイラー級数 Taylor series**は**マクローリン級数 Maclaurin series**と呼ばれます。0以外の位置を中心としたテイラー級数の方が、早く正確に計算結果を得られる場合もあります。

#### 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」ことは多いのですが、級数展開では収束が遅く、「連分数展開」という方法の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。

$$\tan x = \frac{x}{1 - \frac{x^2}{3 - \frac{x^2}{5 - \frac{x^2}{7 - \dots}}}}$$

たとえば

$$\tan 1 = 1.5574077246549\dots$$

に対して、

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

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 Euclidean algorithm** に似ています。

級数展開が

$$y = a_0 + a_1x + a_2x^2 + \dots$$

と表されるとき、

$$y = a_0 + a_1x \left[ 1 + (a_2/a_1)x + (a_3/a_1)x^2 + \dots \right]$$

$$\begin{aligned}
&= a_0 + \frac{a_1 x}{1} = a_0 + \frac{a_1 x}{1 - \frac{(a_2/a_1)x + (a_3/a_1)x^2 + \dots}{1 + (a_2/a_1)x + (a_3/a_1)x^2 + \dots}} \\
&= a_0 + \frac{a_1 x}{1 - \frac{(a_2/a_1)x[1 + (a_3/a_2)x + (a_4/a_2)x^2 + \dots]}{1 + (a_2/a_1)x + (a_3/a_1)x^2 + \dots}} \\
&= a_0 + \frac{a_1 x}{1 - \frac{(a_2/a_1)x}{\frac{1 + (a_2/a_1)x + (a_3/a_1)x^2 + \dots}{1 + (a_3/a_2)x + (a_4/a_2)x^2 + \dots}}} \\
&= a_0 + \frac{a_1 x}{1 - \frac{(a_2/a_1)x}{1 - \frac{(a_3/a_2 - a_2/a_1)x + (a_4/a_2 - a_3/a_1)x^2 + \dots}{1 + (a_3/a_2)x + (a_4/a_2)x^2 + \dots}}} \\
&= a_0 + \frac{a_1 x}{1 - \frac{(a_2/a_1)x}{1 - \frac{(a_3/a_2 - a_2/a_1)x \{1 + [(a_4/a_2 - a_3/a_1)/(a_3/a_2 - a_2/a_1)]x + \dots\}}{1 + (a_3/a_2)x + (a_4/a_2)x^2 + \dots}}} \\
&= a_0 + \frac{a_1 x}{1 - \frac{(a_2/a_1)x}{1 - \frac{(a_3/a_2 - a_2/a_1)x}{1 + (a_3/a_2)x + (a_4/a_2)x^2 + \dots}}} \\
&= a_0 + \frac{a_1 x}{1 - \frac{(a_2/a_1)x}{1 - \frac{(a_3/a_2 - a_2/a_1)x}{1 + [(a_4/a_2 - a_3/a_1)/(a_3/a_2 - a_2/a_1)]x + \dots}}} \\
&= a_0 + \frac{a_1 x}{1 - \frac{(a_2/a_1)x}{1 - \frac{(a_3/a_2 - a_2/a_1)x}{1 + [(a_4/a_2 - a_3/a_1)/(a_3/a_2 - a_2/a_1)]x + \dots}}}
\end{aligned}$$

のように、順々に変形できます。

ただし、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ないとされています。

### 4-3 方程式の数値的な解法 Root findings

「方程式  $y = f(x)$  を  $x$  について解く」タイプの問題で、 $y$  に具体的な数値をあてはめても良い問題の場合には「数値的に解く方法」が使えます。方程式を数値的に解く方法の代

表的なものは**バイセクション法** (<sup>にぶん</sup>二分法) bisection method と **ニュートン・ラフソン法** Newton-Raphson method です。

#### 4-3-1 バイセクション法 Bisection method

**バイセクション法** (<sup>にぶん</sup>二分法) bisection method は、数値計算技術の中でも、特に重要なテクニックです。方程式  $f(x) = 0$  を解くことについて考えます ( $y = f(x)$  の形の方程式の場合には、 $f(x) - y$  あるいは  $y - f(x)$  を、 $f(x)$  とおきかえます)。方程式の解が  $x \in [a, b]$  の関係を満たし、 $f(a)$  の値と  $f(b)$  の値の符号は異なる ( $f(a) < 0 < f(b)$  または  $f(b) < 0 < f(a)$ ) とします。

はじめに

$$s_a \leftarrow \text{sign}[f(a)]$$

$$s_b \leftarrow \text{sign}[f(b)]$$

とします。ここで  $\text{sign}(x)$  は  $x$  の**符号 signal** を返す関数とします。以下の手順 (1)–(3) を繰り返します。

$$(1) \quad c \leftarrow \frac{a+b}{2} \quad (a \text{ と } b \text{ の中点を } c \text{ とする})$$

$$(2) \quad s_c \leftarrow \text{sign}[f(c)] \quad (\text{中点位置 } c \text{ での函数値の符号を求める})$$

$$(3) \quad \begin{cases} a \leftarrow c & [s_c = s_a] \\ b \leftarrow c & [s_c = s_b] \end{cases} \quad (\text{中点での函数値の符号と一致する端点を中点 } c \text{ でおきかえる})$$

$|b - a|$  の大きさが十分に小さくなるまで (1)–(3) を繰り返します。

初めの区間  $[a, b]$  の中に解が 1 つ以上存在しさえすれば、函数  $f(x)$  が「不連続な函数」であっても「微分不可能な函数」であっても、ほぼ確実に方程式  $f(x) = 0$  の解のうちの 1 つが求められます。

やや単純すぎる例かもしれませんが、数値的な手法 (コンピュータ) を使って  $x \geq 0$  の場合に方程式  $f(x) = x^2 - 2 = 0$  の解を求めることをここらみま。はじめに  $a = 1$ ,  $b = 2$  とすれば、 $f(a) = -1$ ,  $f(b) = 2$  となるので、 $x \in [a, b]$  に解があることがわかります。

1 回目は  $c = 1.5$ ,  $f(c) = 1.5^2 - 2 = 0.25$  となるので、 $b \leftarrow c$  とします。

2 回目は  $c = 1.25$ ,  $f(c) = 1.25^2 - 2 = -0.4375$  となるので、 $a \leftarrow c$  とします。

3 回目は  $c = 1.375$ ,  $f(c) = 1.375^2 - 2 = -0.109375$  となるので、 $a \leftarrow c$  とします。

4 回目は  $c = 1.4375$ ,  $f(c) = 1.4375^2 - 2 = 0.0664062$  となるので、 $b \leftarrow c$  とします。

5 回目は  $c = 1.40625$ ,  $f(c) = 1.40625^2 - 2 = -0.0224609$  となるので、 $a \leftarrow c$  とします。

このように、**繰り返し**<sup>イテレーション</sup> iteration の回数を重ねれば  $c$  の値は、正解  $\sqrt{2} = 1.41421356\dots$  に着実に近づきます。この方法では、解のとりうる値の下限と上限 ( $a$  と  $b$ ) の値も常にわかるので、繰り返しを途中で打ち切ったとしても、その時点でとりうる計算誤差の範囲  $|a - b|$  がわかるので安心です。また、要求される精度が達成されるまで何回の繰り返しのすれば良いか予想できますし、要求される精度が達成された時点でくり返しを中止するように計算制御コードを書くことも難しくありません。

解析的 (代数的) に解きにくい関数  $f(x)$  は、函数値を求めるための計算コストもかかる場合が多く、要求される精度を実現するために関数  $f(x)$  の計算回数が多くなりがちなこと、バイセクション法の欠点でしょう。しかし、単純なコード (プログラム) で実現できて、汎用性が高く (例外処理の必要性が低く)、ほぼ確実に解が得られるなど、明確なメリットがあり、後述するニュートン・ラフソン法を使う前に、知るべきテクニックです。

#### 4-3-2 ニュートン・ラフソン法 Newton-Raphson method

**ニュートン法** Newton method とも呼ばれます ([補足 4.3.2.A](#))。

方程式  $f(x) = 0$  を解く問題について考えます。

関数  $f(x)$  の微分  $f'(x)$  が求められるとします。関数  $f(x)$  が初等関数を組み合わせて表現されていれば微分は必ず解けますし、代数的な表現に「微分形式の数値計算ライブラリが提供されない特殊関数」が含まれていても、[4-4節](#)で述べる数値微分のテクニックを使えば、特定の変数  $x$  に対する函数値を得るための計算コストの2倍程度の計算コストで「微分」は数値的に解けます。

**ニュートン・ラフソン法** Newton-Raphson method で方程式  $f(x) = 0$  の解を求めるとき、「正解から遠くないと思われる  $x$  の値を  $x_0$  とする」ことから始めます。以下の (1), (2) の手順を繰り返します。

$$(1) \quad x \leftarrow x_0 - \frac{f(x_0)}{f'(x_0)}$$

$$(2) \quad x_0 \leftarrow x$$

上の (1) の処理をしたときに、 $|x - x_0|$  の大きさが十分に小さくなれば、繰り返しを終了します。

ここでも [4-3-1節](#)と同じように  $f(x) = x^2 - 2 = 0$  の解を求めることを試みます。

$f'(x) = 2x$  として、初期値としては  $x_0 = 1$  とします。

1回目の更新値は  $x = 1 - (1^2 - 2)/2 = 1.5$ ,

2 回目の更新値は  $x = 1.5 - (1.5^2 - 2)/(2 \times 1.5) = 1.41667\dots$ ,

3 回目の更新値は  $x = 1.41667\dots - (1.41667\dots^2 - 2)/(2 \times 1.41667\dots) = 1.41429\dots$ ,

4 回目の更新値は  $x = \dots = 1.41421356\dots$ ,

のように、正解  $x = \sqrt{2} = 1.41421356\dots$  に近くなるほど、さらに正解への近付き方が急速になります。

函数  $f(x)$  が「滑らかで素直な函数」のときに、ニュートン・ラフソン法は、1 回の更新のたびに有効桁数を 2 倍にする性格を持ちます。一方で函数  $f(x)$  が滑らかだとしても「素直でない」函数の時には、ニュートン・ラフソン法は効率が低くなるだけでなく、まったく使い物にならない場合もあります ([補足 4.3.2.B](#))。

#### 4-4 数値微分 Numerical differential

高校数学の「微分」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、微分の定義式

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h} \quad (4.4.1)$$

だけです。数値微分では「十分に小さい数  $h$ 」を使って

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \quad (4.4.2)$$

とします。

たとえば、「函数  $f(x) = 2^x$  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、

$$f(x) = 2^x = \exp(\ln 2^x) = \exp(x \ln 2)$$

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

から、

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

のように計算するでしょう。

しかし、コンピュータを使えるのであれば、式 ([4.4.2](#)) が利用できます。たとえば  $h = 0.0000001$  として、

$$\frac{f(3.000\,000\,1) - f(3)}{0.000\,000\,1} = \frac{2^{3.000\,000\,1} - 2^3}{0.000\,000\,1} = \frac{8.000\,000\,554\,517\,76\dots - 8}{0.000\,000\,1} = 5.545\,177\,6\dots$$

のように計算すれば、約7桁まで正確な値が得られます。このようにして微分の近似値を**差分 difference**として計算することを、「**数値微分 numerical differential**」と呼びます。この例はまだ簡単な方ですが、函数  $f(x)$  の内容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。

数値微分に用いる「十分に小さい数  $h$ 」として、どの程度の大きさの数を選ぶべきかは函数  $f(x)$  の性質によっても変わり、単純な問題ではありませんが、IEEE 754 の 64 bit (倍精度) 浮動小数点数を用いる場合には、 $h \approx 10^{-8}x$  から  $h \approx 10^{-7}x$  程度の値とするのが普通で、このときに数値微分で得られる値の精度は最も良くなり、十進数での有効桁数が7桁から8桁程度になります ([補足 4.4.A](#))。式 ([4.4.2](#)) の形式は**前進差分 forward difference**と呼ばれます。

数値微分を計算する場合に、式 ([4.4.2](#)) の代わりに

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \tag{4.4.3}$$

として計算する場合があります。

式 ([4.4.3](#)) に示す形式は**中心差分 central difference**とよばれ、式 ([4.4.3](#)) の中心差分を使って計算する方が、式 ([4.4.2](#)) の前方差分を使うより正確な微分値に近い値を求められる傾向があります。64 bit (倍精度) 浮動小数点数を用いる場合に、中心差分を用いるなら、刻み幅は  $h \approx 10^{-5}x$  程度の値とするのが普通で、このときに数値微分で得られる値の精度が最も良くなり、十進数での有効桁数が10桁から11桁程度になります ([補足 4.4.B](#))。

ただし、数値微分を使って微分の近似値を求めようとする場合の多くは函数  $f(x)$  が複雑な数式で表されて、計算コストが「函数  $f(x)$  を何回計算するか」でほぼ決まります。数値微分にニュートン・ラフソン法を用いる場合に、既に  $f(x)$  の値が計算されていることを前提として、式 ([4.4.2](#)) では  $f(x+h)$  の値の計算をすれば良いだけなのに対して、式 ([4.4.3](#)) では  $f(x-h)$  の値も計算しなければならないことがデメリットになる場合もあります。

いずれにしても、コンピュータを使う数値微分の計算が難しくないことは、知っておくべきでしょう。高校数学の知識で複雑な数式の微分を代数的な方法で求めるのは悪くないのですが、その結果が正しいかは、必ずコンピュータを使った数値微分で確認するべきです。

## 4-5 数値積分 Numerical integral (Numerical integration)

数値計算によって定積分

$$S = \int_a^b f(x) dx \quad (4.5.1)$$

を計算すること、あるいはその方法を「**数値積分 numerical integral**」あるいは「**求積法 quadrature**」と呼びます。

典型的な方法では

$$S \approx (b-a)[W_1f(x_1) + W_2f(x_2) + \dots + W_Nf(x_N)] = (b-a) \sum_{j=1}^N W_jf(x_j) \quad (4.5.2)$$

$$x_j = a + (b-a)X_j \quad (4.5.3)$$

という形を使って計算します。

式 (4.5.2), (4.5.3) の中の  $X_j \in \{X_1, X_2, \dots, X_N\}$  は「**標本点**」 sampling point あるいは標本点位置,  $W_j \in \{W_1, W_2, \dots, W_N\}$  は「**重み**」 weight と呼ばれます。式 (4.5.2), (4.5.3) のような表現では標本点位置  $\{X_j\}$  と重み  $\{W_j\}$  は,

$$0 \leq X_1 < X_2 < \dots < X_N \leq 1 \quad (4.5.4)$$

$$W_1 + W_2 + \dots + W_N = 1 \quad (4.5.5)$$

となるように選ばれます。

標本点  $\{X_j\}$  の取り方, あるいは重み  $\{W_j\}$  の決め方の違う様々な数値積分法 (求積法) が知られています。

### 4-5-1 中点法 mid-point method

「数値計算の教科書」のような書籍 (e.g. 伊理・藤野, 1985; Press et al., 2007) に記載されていない場合も多いのですが, 求積法の中で最も基本的で知っておくべき方法は**中点法 mid-point method**とよばれる方法です。この方法では式 (4.5.2), (4.5.3) の形式

$$S \approx (b-a)[W_1f(x_1) + W_2f(x_2) + \dots + W_Nf(x_N)] = (b-a) \sum_{j=1}^N W_jf(x_j) \quad (4.5.2)$$

$$x_j = a + (b-a)X_j \quad (4.5.3)$$

に対して, 標本点位置  $\{X_j\}$  と重み  $\{W_j\}$  を

$$X_j = \frac{j-0.5}{N} \quad (4.5.1.1)$$

$$W_j = \frac{1}{N} \tag{4.5.1.2}$$

とします。

ここで、具体的に

$$S = \int_0^1 \sqrt{1-x^2} dx \tag{4.5.1.3}$$

と表される定積分を解くことについて考えます。

式 (4.5.1.3) の積分の値は [Fig. 4.5.1.1](#) に示した図形の面積（半径 1 の円の 4 分の 1 の扇形の面積）と同じことなので、 $\pi/4 = 0.785\dots$  という値になることは積分を解かなくてもわかります。

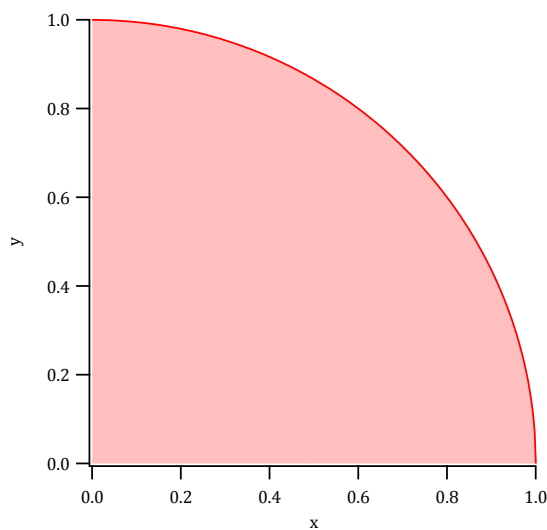


Fig. 4.5.1.1 式 (4.5.1.3) の積分の幾何学的な解釈。塗りつぶした部分の面積を求めるのと同じこと。

高校の数学の問題であれば、式 (4.5.1.3) は

$$\begin{aligned}
 S &= \int_0^1 \sqrt{1-x^2} dx && \stackrel{\substack{= \\ \uparrow \\ x \equiv \sin \theta \\ dx = \cos \theta d\theta \\ x : 0 \rightarrow 1 \\ \theta : 0 \rightarrow \pi/2}}{=} && \int_0^{\pi/2} \sqrt{1-\sin^2 \theta} \cos \theta d\theta = \int_0^{\pi/2} \cos^2 \theta d\theta \\
 & && \stackrel{\substack{= \\ \uparrow \\ \cos^2 \theta = \frac{1+\cos 2\theta}{2}}}{=} && \int_0^{\pi/2} \frac{1+\cos 2\theta}{2} d\theta = \frac{1}{2} \left[ \theta + \frac{\sin 2\theta}{2} \right]_0^{\pi/2} = \frac{\pi}{4}
 \end{aligned} \tag{4.5.1.4}$$

として解くのが「正解」とされるかもしれません。

しかし、一般的に式変形回数が多くなればなるほど「うっかりミス」を犯す確率も高くなるので、ある程度込み入った積分の問題であれば、仮に偶然正解を導けたとしても、その解が正しいか、数値積分を使って確認する必要があります。

次に、中点法でこの問題を解いてみます。初めに標本点の数を5点、 $N=5$ としてみます。中点法で求められる面積は、[Fig. 4.5.1.2](#)で示した図形の面積になるはずであり、「少しずれているかもしれないが、かなり近い値が得られそうだ」とわかるでしょう。

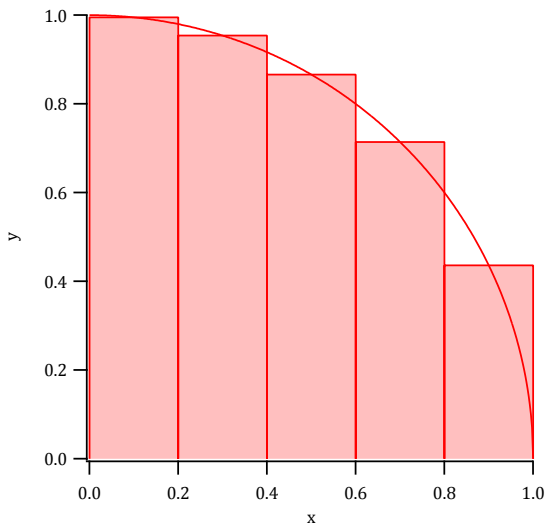


Fig. 4.5.1.2 式 (4.5.1.3) の積分を標本点数5点の中点法で計算する時に求められる面積

実際に電卓や表計算ソフトなどを使って計算すれば、

$$S \approx \frac{\sqrt{1-0.1^2} + \sqrt{1-0.3^2} + \sqrt{1-0.5^2} + \sqrt{1-0.7^2} + \sqrt{1-0.9^2}}{5} = 0.792\dots$$

となり、正解の  $S = \pi/4 = 0.785\dots$  と比較すると、有効数字2桁まで正解が得られていることがわかります。標本点数を10点、 $N=10$ とすれば、 $S \approx 0.788\dots$  という数値が得られます。標本点数を増やせば必ず正解に近づけることも期待できます。

#### 4-5-2 台形法 trapezoid method

普通の数値計算の教科書のような書籍 (e.g. 伊理・藤野, 1985; Press et al., 2007) では、数値積分の方法として、この方法がはじめに紹介されることが多いようです。

この方法も式 (4.5.2), (4.5.3) の形式

$$S \approx (b-a) [W_1 f(x_1) + W_2 f(x_2) + \dots + W_N f(x_N)] = (b-a) \sum_{j=1}^N W_j f(x_j) \quad (4.5.2)$$

$$x_j = a + (b-a)X_j \quad (4.5.3)$$

で表される手法のうちの一例と見ることができて、標本点  $\{X_j\}$  と重み  $\{W_j\}$  を以下のように取ります。

$$X_j = \frac{j-1}{N-1} \quad (4.5.2.1)$$

$$W_j = \begin{cases} \frac{1}{2(N-1)} & [j = 1, N] \\ \frac{1}{N-1} & [j = 2, 3, \dots, N-1] \end{cases} \quad (4.5.2.2)$$

ここでも、前の節と同じように標本点 5 点の台形法で式 (4.5.1.3) の問題（扇形の面積）を解くと、

$$S \approx \frac{\sqrt{1-0^2}}{8} + \frac{\sqrt{1-0.25^2} + \sqrt{1-0.5^2} + \sqrt{1-0.75^2}}{4} + \frac{\sqrt{1-1^2}}{8} = 0.749\dots$$

となり、正解  $\pi/4 = 0.785\dots$  と比較して、中点法の結果 0.792... より、ややずれた値になります。台形法で求められる面積の意味を Fig. 4.5.2.1 に示します。

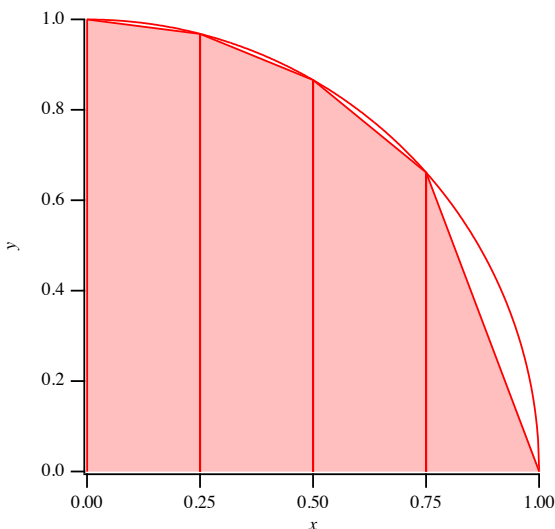


Fig. 4.5.2.1 式 (4.5.1.3) の積分を標本点数 5 点の台形法で計算する時に求められる面積

台形法で積分を解くと、上に凸の曲線では正しい値より必ず小さい値になり、下に凸の曲線では正しい値より必ず大きい値になる特徴があります。

### 4-5-3 シンプソン法 Simpson's method

台形法は分割した区間で被積分函数を一次函数 linear function で近似する考え方に基づいており、シンプソン法では、区分された被積分函数を二次函数 quadratic function で近似する考え方に基づきます。

シンプソン法では、標本点の数は原則的に奇数に限られます。標本点の数が奇数であるとして、式 (4.5.2), (4.5.3) の形式

$$S \approx (b - a) [W_1 f(x_1) + W_2 f(x_2) + \dots + W_N f(x_N)] = (b - a) \sum_{j=1}^N W_j f(x_j) \quad (4.5.2)$$

$$x_j = a + (b - a)X_j \quad (4.5.3)$$

での標本点  $\{X_j\}$  と重み  $\{W_j\}$  を以下のように取ります。

$$X_j = \frac{j-1}{N-1} \quad (4.5.3.1)$$

$$W_j = \begin{cases} 1 & [j = 1, N] \\ \frac{4}{3(N-1)} & [j = 2, 4, \dots, N-1] \\ \frac{2}{3(N-1)} & [j = 3, 5, \dots, N-2] \end{cases} \quad (4.5.3.2)$$

標本点 5 点のシンプソン法で式 (4.5.1.3) の問題（扇形の面積）を解くと、

$$S \approx \frac{\sqrt{1-0^2}}{12} + \frac{\sqrt{1-0.25^2}}{3} + \frac{\sqrt{1-0.5^2}}{6} + \frac{\sqrt{1-0.75^2}}{3} + \frac{\sqrt{1-1^2}}{12} = 0.770\dots$$

となり、正解  $\pi/4 = 0.785\dots$  と比べて、中点法の結果  $0.792\dots$  より、まだ少しずれの大きい値になりますが、台形法の結果  $0.749\dots$  に比べれば、少し正解に近い値になります。

シンプソン法で求められる面積の意味を Fig. 4.5.3.1 に示します。

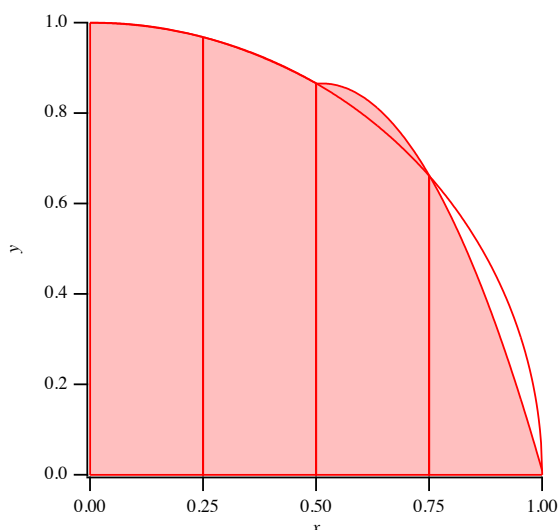


Fig. 4.5.3.1 式 (4.5.1.3) の積分を標本点数 5 点のシンプソン法で計算する時に求められる値の意味。二つの区間に分割して、それぞれの区間について両端点と中点の 3 点を通る二次函数（放物線）で近似する。

実際には、中点法より台形法、さらにシンプソン法の方が正解に早く近づける場合も少なくないのですが、はじめから中点法でなく台形法やシンプソン法を使うべきだというほどではありません。

また、端点で函数値が無限大に発散するような特異性を持つ函数の積分を求める場合に、中点法は使えるのに対して、台形法とシンプソン法はまったく使い物になりません。

#### 4-5-4 ガウス積分 Gaussian Integral

中点法、台形法、シンプソン法は標本点を等間隔に取りましたが、ガウス積分と呼ばれる方法は、等間隔でない標本点の取り方をします。ガウス積分の代表的なものは**ガウス・ルジャンドル積分 Gauss-Legendre quadrature** と呼ばれ、ルジャンドル多項式の値がゼロとなる点を標本点とします。具体的な標本点位置と重みの数値は、アブラモビッツ・ステガンのハンドブック (Abramowitz & Stegun, [1965](#)) に掲載された数表からとるのが定番の方法でしたが、現在ではコンピュータの数値計算ライブラリを使って、自動的に生成させるのが普通です。

アブラモビッツ・ステガンのハンドブックには、5点ガウス・ルジャンドル積分の標本点位置  $\pm X_j$  (区間  $[-1, 1]$  とした時の値) と重み  $W_j$  として [Table 4.5.4.1](#) のような値が記載されています。ガウス・ルジャンドル積分を使う場合には、式 [\(4.5.2\)](#), [\(4.5.3\)](#)

$$S \approx (b-a) [W_1 f(x_1) + W_2 f(x_2) + \dots + W_N f(x_N)] = (b-a) \sum_{j=1}^N W_j f(x_j) \quad (4.5.2)$$

$$x_j = a + (b-a)X_j \quad (4.5.3)$$

の代わりに、

$$S \approx \frac{b-a}{2} \left[ W_{-\frac{N}{2}} f(x_{-\frac{N}{2}}) + W_{-\frac{N}{2}+1} f(x_{-\frac{N}{2}+1}) + \dots + W_{\frac{N}{2}} f(x_{\frac{N}{2}}) \right] = \frac{b-a}{2} \sum_{j=-N/2}^{N/2} W_j f(x_j) \quad (4.5.4.1)$$

$$x_j = \frac{a+b}{2} + X_j \frac{b-a}{2} = \frac{a(1-X_j) + b(1+X_j)}{2} \quad (4.5.4.2)$$

の表現が用いられ、

$$X_{-j} = -X_j \quad (4.5.4.3)$$

$$W_{-j} = W_j \quad (4.5.4.4)$$

の関係が使われます。標本点位置  $\{X_j\}$  と重み  $\{W_j\}$  は、式 [\(4.5.4\)](#), [\(4.5.5\)](#) の代わりに

$$-1 < X_{-\frac{N}{2}} < X_{-\frac{N}{2}+1} < \dots < X_{\frac{N}{2}} < 1 \quad (4.5.4.5)$$

$$W_{-\frac{N}{2}} + W_{-\frac{N}{2}+1} + \dots + W_{\frac{N}{2}} = 2 \quad (4.5.4.6)$$

となるように選ばれるのが普通です。

標本数が  $n$  のガウス・ルジャンドル積分では、標本位置  $\{X_j\}$  は  $n$  次のルジャンドル多項式 Legendre polynomial  $P_n(x)$  の零点 ( $P_n(X_j) = 0$ ) となるように選びます。このとき、重み  $\{W_j\}$  は

$$W_j = \frac{2}{(1 - X_j^2)[P_n'(X_j)]^2} \quad (4.5.4.7)$$

として得られます (Abramowitz & Stegan, 1965, §25.4.30)。

Table 4.5.4.1 5点ガウス・ルジャンドル積分の標本点と重み  
(Abramowitz & Stegan, 1965, Table 25.4)

$\pm X_i$	$W_i$
0.000 000 000 000 000	0.568 888 888 888 889
0.538 469 310 105 683	0.478 628 670 499 366
0.906 179 845 938 664	0.236 926 885 056 189

Table 4.5.4.1 の値を式 (4.5.1.3) の問題 (扇形の面積) に当てはめると、

$$\begin{aligned} S \approx & \frac{0.236\,926\,885\,056\,189}{2} \times \sqrt{1 - \left(\frac{1 - 0.906\,179\,845\,938\,664}{2}\right)^2} \\ & + \frac{0.478\,628\,670\,499\,366}{2} \times \sqrt{1 - \left(\frac{1 - 0.538\,469\,310\,105\,683}{2}\right)^2} \\ & + \frac{0.568\,888\,888\,888\,889}{2} \times \sqrt{1 - \left(\frac{1 - 0.000\,000\,000\,000\,000}{2}\right)^2} \\ & + \frac{0.478\,628\,670\,499\,366}{2} \times \sqrt{1 - \left(\frac{1 + 0.538\,469\,310\,105\,683}{2}\right)^2} \\ & + \frac{0.236\,926\,885\,056\,189}{2} \times \sqrt{1 - \left(\frac{1 + 0.906\,179\,845\,938\,664}{2}\right)^2} \\ = & 0.786\,295\,443\,9173\dots \end{aligned}$$

となり、正しい値  $\pi/4 = 0.785\,398\,163\,397\,44\dots$  にかなり近くなります。

また、アブラモビッツ・ステガンのハンドブックには、標本点の重みを等しくとった時の5点「**ガウス・チェビシェフ Gauss-Chebyshev 積分**」の標本点位置  $\pm X_j$ （ただし区間  $[-1, 1]$  とした時の値）として以下の値が記載されています。

**Table 4.5.4.2** 重みの等しい5点ガウス・チェビシェフ積分の標本点  
(Abramowitz & Stegan, 1965, Table 25.5)

$\pm X_j$
0.832 497 487 0
0.374 541 409 6
0.000 000 000 0

これらの値を式 (4.5.1.3) の問題（扇形の面積）に当てはめると、

$$\begin{aligned}
 S &\approx \frac{1}{5} \times \sqrt{1 - \left(\frac{1 - 0.8324974870}{2}\right)^2} + \frac{1}{5} \times \sqrt{1 - \left(\frac{1 - 0.3745414096}{2}\right)^2} \\
 &\quad + \frac{1}{5} \times \sqrt{1 - \left(\frac{1 - 0.0000000000}{2}\right)^2} + \frac{1}{5} \times \sqrt{1 - \left(\frac{1 + 0.3745414096}{2}\right)^2} \\
 &\quad + \frac{1}{5} \times \sqrt{1 - \left(\frac{1 + 0.8324974870}{2}\right)^2} \\
 &= 0.787873123\dots
 \end{aligned}$$

となり、これもかなり正しい値  $\pi/4 = 0.78539816339744\dots$  に近い値になります。

#### 4-5-5 数値積分のためのその他の方法 Other methods for Numerical Integral

**ガウス積分 Gaussian quadrature**には、[4-5-4節](#)で取り上げた**ガウス・ルジャンドル積分 Gauss-Legendre quadrature**と**ガウス・チェビシェフ積分 Gauss-Chebyshev quadrature**以外に、積分区間が半無限区間の場合に有効な**ガウス・ラゲール積分 Gauss-Laguerre quadrature**や、積分区間が無限区間の場合に有効な**ガウス・エルミート積分 Gauss-Hermite quadrature**もあり、どのような問題を解くかによって使い分けられます。

コンピュータに擬似的な乱数を発生させ、この擬似乱数を使って数値計算をする方法は一般的に**モンテカルロ法 Monte Carlo method**と呼ばれます。この方法は、積分の値を求めるために用いられる場合もあります。モンテカルロ法については別の章で紹介します。

## 4-6 数値計算の技術 Tips for numerical calculations

### 4-6-1 冪乗の計算 Calculation of power

冪乗<sup>べき</sup>として、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ありませんが、計算に無駄が多だけでなく、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 回しなければ正解が得られない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。

### 4-6-2 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なりますが、

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

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

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

のように表される数式を、

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

と変形して計算する例なども良く知られています。

このような計算の方法は**ホーナーの方法** Horner's method と呼ばれます。

### 4-6-3 二次方程式の解 Solution of quadratic equation

二次方程式： $x^2 + 2bx + c = 0$  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、64 bit 浮動小数点数を使っても公式通りの計算のしかただと

$$-b + \sqrt{b^2 - c} = -0.000010000000000001$$

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。

### 4-7 この章の最後に

コンピュータでふつうに扱われる数値は、それほど正確なものではありません。「実数型」と呼ばれる場合もある浮動小数点型の表現で厳密な意味で正確に扱える数は「有理数のうちのごく限られた部分（既約分数の分母が2の整数乗で表される数のうち、一定の大きさの範囲に収まるもの）」だけであり、「それ以外の有理数」や「無理数」は近似的にしか扱えません。（ただし「数式処理ソフト」には IEEE 754-1985 規格より高精度の数値や、有理数を正確に扱えるものもあります。）

コンピュータの普通の計算システムは、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であり、計算結果には必ず誤差が伴います。普通の実験データに伴う誤差の大きさに比べて相対的に小さいので、実験データを処理する場合には気にならない場合が多いかもしれませんが、シミュレーションでは計算ステップが多くなりがちで、計算誤差の累積によって問題の引き起こされる場合も珍しくありません。

プログラミング以前の問題で、計算のしかたに工夫をすれば「近似であるとしても非常に正確な結果」が得られる場合もあり、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を伴う結果になる場合もあります。

数値計算は学問の対象になりうるものかもしれませんが、現実には、むしろ技術的な側面が強いとも言われます。

# 補足

## (補足 4.1.A) 指数関数 (↔)

高校の数学で「自然対数の底」 the base of the natural logarithm として習う定数

$$e = 2.718\ 281\ 828\ 459\ 045\ 235\ 360\ 287\ 471\ 352\ 662\ 497\ 757\dots$$

は、英語圏では Napier's constant (ネイピア定数) と呼ばれることも多かったが、最近は Euler's number と呼ばれることが多くなり、以前 Euler's constant (オイラー定数) と呼ばれていた定数

$$\begin{aligned} \gamma &= \lim_{n \rightarrow \infty} \left( -\ln n + \sum_{k=1}^n \frac{1}{k} \right) \\ &= 0.577\ 215\ 664\ 901\ 532\ 860\ 606\ 512\ 090\ 082\ 402\ 431\ 042\ 159\ 335\ 939\ 92\dots \end{aligned}$$

は Euler-Mascheroni constant (オイラー・マスケローニ定数) と呼ばれる傾向がある。

以下の式で表される双曲線正割関数の展開係数  $\{E_n\}$  は Euler numbers と呼ばれる。

$$\operatorname{sech} t = \frac{1}{\cosh t} = \frac{2}{e^t + e^{-t}} = \sum_{n=0}^{\infty} \frac{E_n t^n}{n!}$$

英語表記では Euler numbers と Euler's number とは区別できるが、日本語に置き換えると区別を付けにくくなる。Euler's number を「オイラー数」と呼び、Euler numbers は「オイラーの展開係数」と呼ぶようにすれば良いかもしれない。中国語では Euler numbers も欧拉数、Euler's number も欧拉数と書かれる。また、位相空間のもつある種の構造を特徴づける位相不変量として Euler characteristic (オイラー標数) というものがあり、Eulerian 多項式の係数として定義される Eulerian number と呼ばれるものもある。かなり紛らわしい状況になっているので、注意した方が良い。

定数  $e = 2.718\ 281\ 828 \dots$  に相当する数学定数は、ヤコブ・ベルヌーイ Jacob Bernoulli (1654-1705) (流体力学のベルヌーイの定理で有名なダニエル・ベルヌーイ Daniel Bernoulli (1700-1782) の伯父) が発見したと言っても良いようだが、ヤコブ・ベルヌーイは「2と3の間の数」ということしかはつきりとはさせておらず、対数との関係も認識していなかったと言われる。

ジョン・ネイピア John Napier (1550-1617) は 1594 年に対数の概念を初めて発見したとも言えるようだが、ネイピアの対数の概念では、定数  $e = 2.718\ 281\ 828 \dots$  に相当する数をはつきりとは使ってはいるわけではない。

レオンハルト・オイラー Leonhard Euler (1707- 1783) は、定数  $e = 2.718\ 281\ 828 \dots$  が「自然対数の底」に相当する数だということをはつきりさせるなど、この定数の重要な性質を明らかにしたことも事実のようであるし、数学の中でもっとも重要な定数とも言える  $e = 2.718\ 281\ 828 \dots$  という定数にこそ、偉大な数学者であるオイラー Euler の名前を冠するべきという考え方があること、Euler のイニシャルが E であることから後世の人が好んで“e”の文字を使ってきたらしいことなどから、Euler's number (オイラー数) と呼ばれることが今後も多くなることが予想される。

日本での「自然対数の底」という呼び方は、オイラーの名前のついた他の定数との紛らわしさを避けるためには悪くない面もあると思われるが、「オイラー数を底とする対数」  $\log_e x$  (自然対数) のことを「自然対数の底を底とする対数」と呼ぶことになってしまう不自然さがある。 (↔)

### (補足 4.1.B) 対数関数の表記 (↔)

高校数学では「自然対数」 natural logarithm を  $\log x$  と表記するが、分野によっては過去に 10 を底とする対数  $\log_{10} x$  のことを「常用対数」 common logarithm と呼んで  $\log x$  と表記する習慣があった。現在は、どの分野でも 10 を底とする対数は  $\log_{10} x$  と書くことが常識的である。数学以外の自然科学・技術分野では、自然対数は  $\ln x$  と表記する場合が多い。 (↔)

### (補足 4.1.C) 余割関数の表記 (↔)

余割関数 cosecant function は  $\operatorname{cosec} x$  と表記されることも多かったが、最近では  $\operatorname{csc} x$  と表記する場合が多い。 (↔)

### (補足 4.1.D) 逆三角関数 (↔)

逆三角関数 inverse trigonometric functions は三角関数の逆関数という意味合いもあるのだが、「三角関数の逆関数」というだけでは値が一意には確定しない。数学的な表現としては「値が一意に確定しなくてもかまわない」場合もあるのだが、逆三角関数の数値計算ライブラリは、値域を明確に定義した形で提供される。普通には、 $\arcsin x$  の値域は  $[-\pi/2, \pi/2]$ 、 $\arccos x$  の値域は  $[0, \pi]$ 、 $\arctan x$  の値域は  $[-\pi/2, \pi/2]$  とされる。

ただし、逆三角関数の数値計算ライブラリを利用する場合には、このこととは別に、実際に数値をあてはめて、どのような値を返してくるか確かめてから使う方が良い。

値が確定しなくても構わないときに  $\arcsin x$ 、 $\arccos x$ 、 $\arctan x$  と表記され、値域が制限され値が確定することを前提とする場合に  $\operatorname{Arcsin} x$ 、 $\operatorname{Arccos} x$ 、 $\operatorname{Arctan} x$  と表記される傾向がある。

逆正接関数  $\arctan x$  は良く使われる関数で、

$$\begin{cases} x = \cos \theta \\ y = \sin \theta \end{cases}$$

の関係が成立するとき  $(x, y)$  から  $\theta$  の値を求められる二変数関数のライブラリが  $\operatorname{Arctan2}(x, y)$  や  $\operatorname{atan2}(x, y)$  などとして提供されるのが普通である。この二変数関数  $\operatorname{Arctan2}(x, y)$  は、値域を  $[-\pi/2, \pi/2]$  とする逆正接関数  $\operatorname{Arctan} x$  を使って、

$$\operatorname{Arctan2}(x, y) = \begin{cases} -\pi & [x < 0, y = 0] \\ \operatorname{Arctan} \frac{y}{x} - \pi & [x < 0, y < 0] \\ -\frac{\pi}{2} & [x = 0, y < 0] \\ \operatorname{Arctan} \frac{y}{x} & [0 < x] \\ \frac{\pi}{2} & [x = 0, 0 < y] \\ \operatorname{Arctan} \frac{y}{x} + \pi & [x < 0, 0 < y] \end{cases}$$

のように表すことができる。 (↔)

### (補足 4.1.E) 双曲線函数 (↔)

双曲線函数 hyperbolic functions は、指数函数を使って表されるので不要のように思われるかもしれないが、例えば双曲線正弦函数 hyperbolic sine function

$$\sinh x = \frac{e^x - e^{-x}}{2}$$

の場合に  $x \simeq 0$  での値を  $[\exp(x) - \exp(-x)]/2$  の形式で計算しようとする、 $\exp(x) \simeq 1$ ,  $\exp(-x) \simeq 1$  となることから、減算によって激しい桁落ち loss of significance が生じて、有効数字が著しく減少してしまう。普通の計算システムでは、双曲線函数を正しく計算するライブラリが提供される。双曲線函数には「使うべきところでは使わなければいけない」性格があり、知っておくべき函数である。 (↔)

### (補足 4.1.F) 逆双曲線余弦函数の値域 (↔)

逆双曲線余弦函数 (inverse hyperbolic cosine function)  $\operatorname{arccosh} x$  のライブラリが提供される場合、その値域は  $[1, \infty)$  に制限されるのが普通である。 (↔)

### (補足 4.1.G) Python 言語での数値計算 (↔)

Python に標準的に付属する数値計算モジュール math でも、Python を使った数値計算のために事実上の標準 (デファクト・スタンダード) となっている NumPy ライブラリでも、三角函数と逆三角函数、双曲線函数、逆双曲線函数は提供される。ただし提供されるのは  $\sin, \cos, \tan, \arcsin, \arccos, \arctan, \sinh, \cosh, \tanh, \operatorname{arcsinh}, \operatorname{arccosh}, \operatorname{arctanh}$  のみで、初等函数に分類され「断り無しに使って良い」とされる  $\csc, \sec, \cot, \operatorname{arccsc}, \operatorname{arcsec}, \operatorname{arccot}, \operatorname{csch}, \operatorname{sech}, \operatorname{coth}, \operatorname{arccsch}, \operatorname{arcsech}, \operatorname{arccoth}$  は提供されない。Python と NumPy を使って数値計算をする場合には、 $\csc x \equiv 1/\sin x$ ,  $\sec x \equiv 1/\cos x$ ,  $\cot x \equiv 1/\tan x$ , ... などの関係を知っておくことが必要になる。 (↔)

### (補足 4.3.2.A) ニュートン法? ニュートン・ラフソン法? (↔)

函数の微分を使って方程式の解を求める手法は、ニュートン法と呼ばれることの方が、まだ多いかもしれない。この手法をニュートン法と呼ぶのでも良いと思われるが、現在実際に使われている方法は、17-18 世紀の英国の科学者 Sir Isaac Newton とは独立に、ほぼ同時期 17-18 世紀の英国の数学者 Joseph Raphson の考案した方法に近いという説がある。このことから、単にニュートン法と呼ぶのではなく、ニュートン・ラフソン法と呼ぶ割合が増える傾向にある。 (↔)

### (補足 4.3.2.B) ニュートン・ラフソン法にとって素直でない函数 (↔)

例えば  $f(x) = x/(1+x^2)$  という函数の形式は、物理現象を記述するための数式として、ときどき現れるタイプのものである。函数  $f(x)$  の微分形式  $f'(x)$  の代数的な表現は、

$$f'(x) = \frac{d}{dx} \left( \frac{x}{1+x^2} \right) \quad \begin{array}{c} = \\ \uparrow \end{array} \quad \frac{1+x^2 - x(2x)}{(1+x^2)^2} = \frac{1-x^2}{(1+x^2)^2} \quad (4.3.2.B.1)$$

$(u/v)' = (u'v - uv')/v^2$

のように、日本の高校数学の知識しか持たずとも得られる。

この函数  $f(x) = x/(1+x^2)$  について、 $f(x) = 0$  となるような解は  $x = 0$  以外に存在しない。

ところが、例えば初期値  $x_0 = 1$  としてニュートン・ラフソン法を始めようとする、 $f'(1) = 0$  となり、「ゼロで割る」計算はできないことから、「はじめからどうにもならない」ことになってしまう。このような函数が「素直でない函数」と呼ばれる。 (↔)

#### (補足 4.4.A) 前進差分による数値微分 (↔)

数値微分に用いる「十分に小さい数  $h$ 」として、どの程度の大きさの数を選ぶべきだろうか？これは函数  $f(x)$  の性質によっても変わるので、必ずしも単純な問題ではない。 $h$  の値は小さくするほど微分の定義に近づくが、 $h$  の値が  $x$  の値に比べて小さすぎれば、 $f(x+h)$  から  $f(x)$  を引いた値 (差) の有効数字の桁数が「<sup>けたお</sup>桁落ち」loss of significance によって少なくなってしまう。

函数  $f(x)$  がテイラー展開 Taylor expansion で近似できる場合には、前進差分と微分の差は

$$\begin{aligned} \frac{f(x+h) - f(x)}{h} - f'(x) &= \frac{1}{h} \left[ f(x) + hf'(x) + \frac{h^2}{2} f''(x) + \dots \right] - \frac{f(x)}{h} - f'(x) \\ &\approx \frac{h}{2} f''(x) \end{aligned} \quad (4.4.A.1)$$

として見積もることができる。このタイプの誤差は**打ち切り誤差** truncation error と呼ばれる。

また、IEEE 754-1985 規格の 64 bit 浮動小数点数を用いる場合、 $f(x+h)$  も  $f(x)$  も仮数部の有効ビット数が 52 bit なので、それぞれ  $2^{-52} f(x) \approx 2.2 \times 10^{-16} f(x)$  程度の誤差を見込まないといけない。この誤差は**丸め誤差** round-off error (rounding error) と呼ばれる。

前進差分  $\frac{f(x+h) - f(x)}{h}$  の計算結果には  $\frac{2 \times 2^{-52} f(x)}{h} = \frac{2^{-51} f(x)}{h}$  の**丸め誤差**が伴うとみるべきであり、**打ち切り誤差**と**丸め誤差**を合わせた誤差値が

$$g(h) = \frac{h}{2} |f''(x)| + \frac{2^{-51} |f(x)|}{h} \quad (4.4.A.2)$$

と表されるとすれば、

$$\frac{dg(h)}{dh} = \frac{1}{2} |f''(x)| - \frac{2^{-51} |f(x)|}{h^2} = \frac{1}{2} |f''(x)| \left[ 1 - \frac{2^{-50} |f(x)|}{h^2 |f''(x)|} \right] \quad (4.4.A.3)$$

となることから、 $g(h)$  の値が最小になるのは、 $h$  の値が

$$h_{\text{opt}} = 2^{-25} \sqrt{\left| \frac{f(x)}{f''(x)} \right|} \approx 3.0 \times 10^{-8} \sqrt{\left| \frac{f(x)}{f''(x)} \right|} \quad (4.4.A.4)$$

の値に近い場合と予想される。例えば  $f(x) = 2^x$ 、 $x = 3$  の場合には、 $f(3) = 8$ 、 $f''(x) = 2^x (\ln 2)^2$  から  $f''(3) \approx 3.84$  と見積もられ、

$$h_{\text{opt}} \approx 2^{-25} \times \sqrt{\frac{8}{3.84}} \approx 4.3 \times 10^{-8} \quad (4.4.A.5)$$

となる。このときの前進差分の誤差の値は

$$g(h_{\text{opt}}) \approx h_{\text{opt}} |f''(3)| \approx 1.7 \times 10^{-7} \quad (4.4.A.6)$$

と見積もられる。

単純化して、IEEE 754-1985 規格の 64 bit 浮動小数点数を使う場合に、前進差分による数値微分では、刻み幅  $h$  としては  $x$  の有効桁数の半分の桁数を持つ  $2^{-26}x \approx 1.4 \times 10^{-8}x$  程度の値を用いれば良く、このときに数値微分の有効数字の桁数は半分になるが、それが最適に近いとみなすことができる。

刻み幅  $h$  を変化させたときに  $f(x) = 2^x$ ,  $x = 3$  についての前進差分による数値微分と、正しい微分値  $f'(3) = 2^3 \ln 2$  との差の値を、現代的な計算システムを利用して得た結果を Fig. 4.4.A.1 に示す。

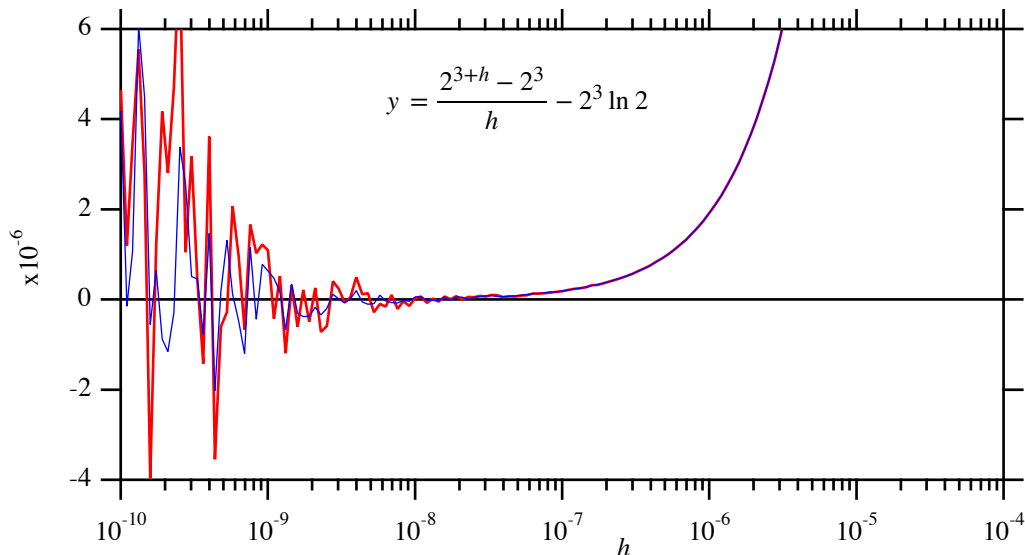


Fig. 4.4.A.1 函数  $f(x) = 2^x$  について、 $x = 3$  での数値微分（前進差分）  $[f(x+h) - f(x)]/h$  を異なる刻み幅  $h$  で計算した値と、正しい微分値  $f'(3) = 2^3 \ln 2$  との差（赤線）。桁落ちの影響を軽減しようとして  $a \leftarrow x+h$ ,  $d \leftarrow a-x$ ,  $f'(x) \approx [f(a) - f(x)]/d$  とした数値微分と  $f'(3)$  の値の差を細い青線で示す。

函数  $f(x) = 2^x$  は下に凸の曲線として表されるので、数値微分の値は必ず正しい値より大きい値になるはずである。Fig. 4.4.A.1 に現れているように、刻み幅  $h$  を細かくしていくと、 $h \approx 10^{-7}$  あたりまでは「打ち切り誤差」が減少することにより、ずれが順調に小さくなるが、それより細かい刻み幅では桁落ちにより不規則な挙動が現れ始める。最も精度の高い結果が得られるのは  $h \approx 10^{-8}$  から  $h \approx 10^{-7}$  の範囲であり、このときの誤差は、 $10^{-7}$  のオーダーになりそうであることも確認できる。

桁落ちの影響を軽減するために、 $a \leftarrow x+h$ ,  $d \leftarrow a-x$ ,  $f'(x) \approx [f(a) - f(x)]/d$  という順で計算すれば良いという説がある。確かにその方法で桁落ちの影響が少しだけ軽減できるらしいことも Fig. 4.4.A.1 に現れているが、丸め誤差による情報の欠落 information loss が生じることには変わらないので、全体的に目立つような効果があるほどではない。 (↔)

#### (補足 4.4.B) 中心差分による数値微分 (↔)

式 (4.4.3) を使って計算される中心差分 central difference の形式で求められる微分の形式の打ち切り誤差は、Taylor 展開の形式から

$$\frac{f(x+h) - f(x-h)}{2h} - f'(x) = \frac{1}{2h} \left[ f(x) + hf'(x) + \frac{h^2}{2} f''(x) + \frac{h^3}{6} f'''(x) + \dots \right]$$

$$\begin{aligned}
& -\frac{1}{2h} \left[ f(x) - hf'(x) + \frac{h^2}{2} f''(x) - \frac{h^3}{6} f'''(x) + \dots \right] - f'(x) \\
& \approx \frac{h^2}{6} f'''(x)
\end{aligned} \tag{4.4.B.1}$$

となる。 $\frac{f(x+h)-f(x-h)}{2h}$  の計算結果には、 $\frac{2 \times 2^{-52} f(x)}{h} = \frac{2^{-51} f(x)}{h}$  の丸め誤差が伴うと見るべきで、

打ち切り誤差と丸め誤差を合わせた値が、

$$g(h) = \frac{h^2}{6} |f'''(x)| + \frac{2^{-51} |f(x)|}{h} \tag{4.4.B.2}$$

と表されるとすれば、

$$\frac{dg(h)}{dh} = \frac{h}{3} |f'''(x)| - \frac{2^{-51} |f(x)|}{h^2} = \frac{h}{3} |f'''(x)| \left[ 1 - \frac{3 \times 2^{-51} |f(x)|}{h^3 |f'''(x)|} \right] \tag{4.4.B.3}$$

から、 $g(h)$  が最小になるのは、 $h$  の値が

$$h_{\text{opt}} \approx \sqrt[3]{3 \times 2^{-51} \left| \frac{f(x)}{f'''(x)} \right|} \approx 1.1 \times 10^{-5} \sqrt[3]{\left| \frac{f(x)}{f'''(x)} \right|} \tag{4.4.B.4}$$

のときであり、 $f(x) = 2^x$ 、 $x = 3$  の場合には  $f(3) = 8$ 、 $f'''(3) = 2^3(\ln 2)^3 \approx 2.66$  から、

$$h_{\text{opt}} \approx \sqrt[3]{3 \times 2^{-51} \times \frac{8}{2.66}} \approx 1.6 \times 10^{-5} \tag{4.4.B.5}$$

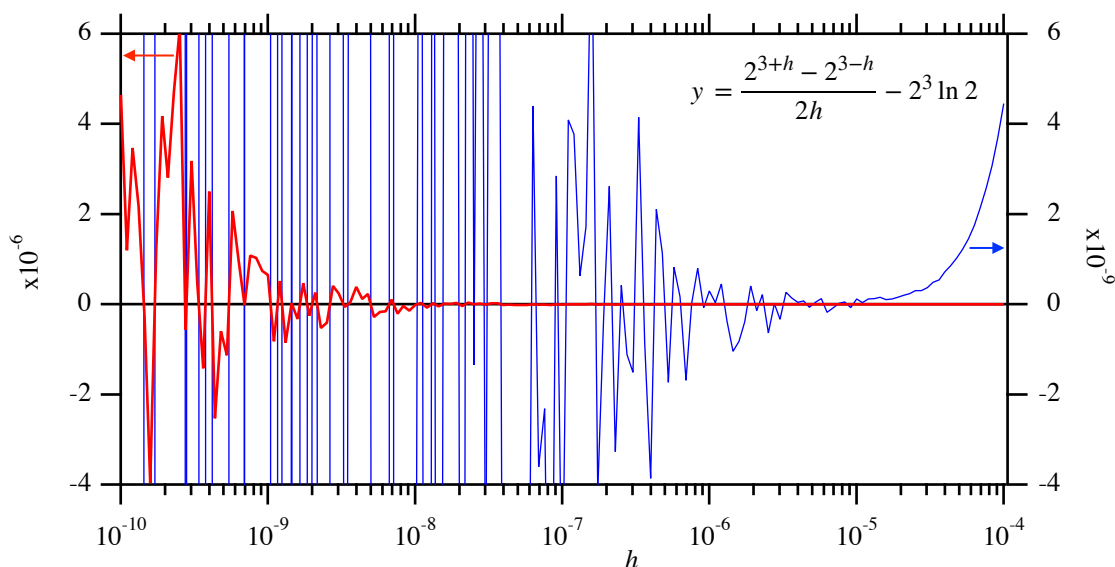
となり、このときの誤差の値は

$$g(h_{\text{opt}}) \approx \frac{h_{\text{opt}}^2}{3} |f'''(3)| \approx 2.2 \times 10^{-10} \tag{4.4.B.6}$$

と予想される。

$f(x) = 2^x$ 、 $x = 3$  という場合について、中心差分  $[f(x+h)-f(x-h)]/(2h)$  と正しい微分値  $f'(3)$  の差が刻み幅  $h$  によってどのように変化するかを [Fig. 4.4.B.1](#) に示す。

[Fig. 4.4.A.1](#) に示した前進差分による数値微分の結果と同じスケールで比較すると、[Fig. 4.4.B.1](#) に示した中心差分による数値微分の結果は  $h > 10^{-7}$  の領域で、明らかに精度が高くなっていることがわかる。また拡大表示を見れば、確かに  $h \simeq 10^{-5}$  付近で最も精度が高くなり、誤差が  $10^{-10}$  程度以下に抑えられていることも確認できる。 ([↔](#))



**Fig. 4.4.B.1**  $f(x) = 2^x$  について  $x = 3$  での中心差分  $[f(x+h) - f(x-h)]/(2h)$  と  $f'(3) = 2^3 \ln 2$  の差。  
 Fig.4.4.A.1 と同じスケールで描いたプロットを赤線, 縦に 1000 倍拡大して描いたプロットを細い青線で示す。

## 参考文献 4

伊理正夫・藤野和建 (1985). 「数値計算の常識」 共立出版 [ISBN-10: 4320013433; ISBN-13: 978-4320013438] (↩)

プルドニコフ・ブリチコフ・マリチェフ (室谷義昭 訳) (1991) 「新数学公式集 I 初等関数」 丸善 [ ISBN-10 : 4621036238, ISBN-13 : 978-4621036235 ] ; Prudnikov, A. P., Brychkov, Yu. A. & Marichev, O. I., “Integrals and Series: Elementary Functions” CRC Press (1998) [ISBN-10 (vol. 1) : 2-88124-089-5, ISBN-13 (vol. 1) : 978-2881240898 ] ; А.П. Прудников, Ю.А. Брычков, и О.И. Маричев, “Интегралы и ряды Элементарная функция” (↩)

プルドニコフ・ブリチコフ・マリチェフ (室谷義昭 訳) (1992) 「新数学公式集 II 特殊関数」 丸善 [ ISBN-10 : 4621036823, ISBN-13 : 978-4621036822 ] ; Prudnikov, A. P., Brychkov, Yu. A. & Marichev, O. I., “Integrals and Series: Special Functions” CRC Press (1998) [ ISBN-10 (vol. 2) : 2881240909; ISBN-13 (vol.2) : 9782881240904 ] ; А.П. Прудников, Ю.А. Брычков, и О.И. Маричев, “Интегралы и ряды специальные функции” (↩)

Prudnikov, A. P., Brychkov, Yu. A. & Marichev, O. I., (1986) “Integrals and Series: Volume 1: Elementary Functions; Volume 2: Special Functions” CRC Press [ ISBN-10 (set) : 2881240976; ISBN-13 (set) : 978-2881240973 ] (↩)

Abramowitz, M. & Stegun, I. A. (1965). “Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables,” Dover Publications, New York. [ ISBN-10 : 0486612724 ; ISBN-13 : 978-0486612720 ] (↩)

Press, W. H., Teukolsky, S. A., Vetterling, W. T. & Flannery, B. P. (2007). "Numerical Recipes 3rd Edition: The Art of Scientific Computing," Cambridge University Press. [ ISBN-10 : 0521880688, ISBN-13 : 978-0521880688 ]. ([↩](#))