1.3 打ち切り誤差

1.3.1 打ち切り誤差（truncation error）

無限または数多くの回数の演算をある有限回の演算で打ち切るときに発生する誤差を，**打ち切り誤差**という。例えば，ある関数 $f(x)$ が，無限級数で表されるとき，有限項で打ち切ることにより発生する。また，数値積分において，積分値を微小面積の足し合わせで近似する際にも発生する。

例 無限級数の計算を考える。例えば，

$$\sum_{i=1}^{\infty} \frac{1}{i^2} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \cdots + \frac{1}{n^2} + \cdots = \frac{\pi^2}{6} \tag{1.17}$$

によって π^2 を計算することを考える。計算機では無限回の演算を行うことはできないので，適当な項 n で和を打ち切る必要がある。このとき，第 $n+1$ 項以降からの寄与が計算結果に入らないことになる。この部分が打ち切り誤差となる。

同様に，指数関数 e^x を級数で計算する場合も，

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots \tag{1.18}$$

$$\simeq 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} \tag{1.19}$$

のように，和を途中までで打ち切って計算することになる。

例 区間 $[a, b]$ における $f(x)$ の積分は，図 1.2 に示すように区間 $[a, b]$ を n 等分し，各区間を底辺とする微小な長方形の面積を足し合わせることで近似できる。

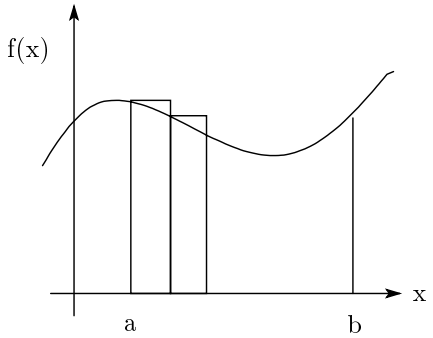

図 1.2: 微小な長方形の足し合わせによる積分の近似

$$\int_a^b f(x)dx \simeq \sum_{i=1}^{n} \frac{b-a}{n} \cdot f\left(a + \frac{b-a}{n}(i-1)\right) \tag{1.20}$$

ここで $n \to \infty$ とすると，右辺の近似値は真の積分値に収束する（$f(x)$ が Riemann 積分可能な場合）。しかし数値計算においては，n を有限で止める必要があるため，誤差が生じる。これも打切り誤差の一種である。

1.3.2　打ち切り誤差の緩和

打ち切り誤差を減らすには，式変形により，もとの数列（あるいは級数）よりも速く収束する数列（級数）に変形すればよい。

例　$e^{10.5}$ を計算する。式 (1.18) を用いると，

$$e^{10.5} = 1 + \frac{10.5}{1!} + \frac{10.5^2}{2!} + \frac{10.5^3}{3!} + \cdots \tag{1.21}$$

となる。この級数は収束するが，分子が指数関数的に増加するため，途中で打ち切ったときの誤差が大きい。例えば，第５項は $(10.5)^4/4! \simeq 506$ であるから，第４項までで打ち切ると少なくともこれだけの誤差が生じる。そこで，

$$\begin{aligned} e^{10.5} &= e^{10} \cdot e^{0.5} \\ &= e^{10} \left(1 + \frac{0.5}{1!} + \frac{0.5^2}{2!} + \frac{0.5^3}{3!} + \cdots \right) \end{aligned} \tag{1.22}$$

のように計算する。式 (1.22) の級数は分子が指数関数的に減少するため，収束が速く，打ち切り誤差が小さい。このことを利用して，$e^{10.5}$ の値を少ない項数で精度良く求めることができる。

同様に，他の級数の計算についても，より速く収束する級数に変換することで打切り誤差を減らすことができる場合がある（第 5 章「数値微分法と加速法」を参照）。

例　数値積分については，より良い近似の公式を使うことで，誤差を減らすことができる。式 (1.20) では，微小な長方形の足し合わせで積分値を近似したが，微小な台形の足し合わせで近似すると，

$$\int_a^b f(x)dx \simeq \sum_{i=1}^{n} \frac{b-a}{n} \cdot \frac{1}{2} \left\{ f\left(a + \frac{b-a}{n}\, i \right) + f\left(a + \frac{b-a}{n}\, (i-1) \right) \right\} \tag{1.23}$$

となる。図 1.2 と図 1.3 の比較からわかるように，式 (1.20) の代わりに式 (1.23) を用いることで，誤差は大幅に減ると考えられる。さらに，台形上部の直線を２次関数で置き換えることで，より誤差の少ない積分公式を作ることも可能である。これらについては，第 4 章「数値積分法」で詳しく説明する。

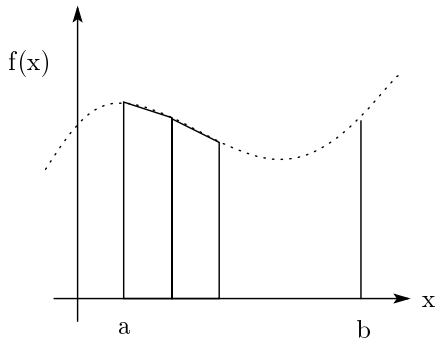

図 1.3: 微小な台形の足し合わせによる積分の近似

1.4 誤差の伝播

ここでは，繰り返し計算により，初期データの誤差がどう伝わっていくか（拡大されていくか）を取り上げる。例として，漸化式の計算を考えてみよう。

漸化式の計算

次の式により，x_0 から始めて，$x_1, x_2, \ldots$ を順に計算するアルゴリズムを考える。

$$x_{n+1} = a_n x_n + b_n$$

ただし，初期値 x_0 と数列 $\{a_n\}$，$\{b_n\}$ は与えられているとする。
このとき，$|a_n|$ の大きさにより，数値計算という立場から見たアルゴリズムの性質は大きく異なる。

1. $|a_n| \geq c > 1$（c はある定数）のとき
   n が大きくなるに連れ，初期誤差は指数的に拡大されていく。これを**不安定なアルゴリズム**という。不安定なアルゴリズムは，数値計算上は役に立たないことが多い。
2. $|a_n| \leq c < 1$ のとき
   初期誤差は縮小されていくので，**安定なアルゴリズム**となる。

例 $n = 0, 2, \ldots, 12$ のそれぞれについて，

$$I_n = \int_0^1 \frac{x^{2n}}{x^2 + 10}\, dx \tag{1.24}$$

を数値計算により求めることを考える。

まず，$n = 1$ のときは，$x = \sqrt{10}\tan\theta$ と置くことにより，

$$I_0 = \int_0^1 \frac{1}{x^2 + 10}\, dx$$

$$
\begin{aligned}
&= \int_0^{\tan^{-1}(1/\sqrt{10})} \frac{1}{10(1+\tan^2\theta)} \cdot \frac{\sqrt{10}}{\cos^2\theta}\, d\theta \\
&= \frac{1}{\sqrt{10}} \int_0^{\tan^{-1}(1/\sqrt{10})} d\theta \\
&= \frac{1}{\sqrt{10}} \tan^{-1}\left(\frac{1}{\sqrt{10}}\right) \qquad (1.25)
\end{aligned}
$$

となり，積分は解析的に計算できる。次に，I_{n-1} がわかったとすると，

$$
\begin{aligned}
I_n &= \int_0^1 \frac{(x^{2n} + 10\,x^{2n-2}) - 10\,x^{2n-2}}{x^2+10}\, dx \\
&= \int_0^1 x^{2n-2}\, dx - 10 \int_0^1 \frac{x^{2n-2}}{x^2+10}\, dx \\
&= \frac{1}{2n-1} - 10 I_{n-1} \qquad (1.26)
\end{aligned}
$$

となるから，I_n も求められることになる。したがって，初期値 (1.25) から始めて，漸化式 (1.26) を用いて $I_1, I_2, I_3, \ldots I_{12}$ を順に計算していくアルゴリズム（前進型の漸化式）が考えられる。

しかし，$|a_n| = 10 > 1$ であるから，このアルゴリズムは数値計算的には不安定である。特に，初期値 (1.25) は無理数 $\frac{1}{\sqrt{10}} \tan^{-1}\left(\frac{1}{\sqrt{10}}\right)$ を 2 進浮動小数点数で表すことによる誤差を含むから，この誤差が各ステップで 10 倍ずつ拡大されてしまう。

そこで，次のように式 (1.26) を変形する。

$$
I_{n-1} = \frac{1}{10}\left(\frac{1}{2n-1} - I_n\right) \qquad (1.27)
$$

さらに，I_{12} を数値積分で正確に求め，これを初期値とすれば，漸化式 (1.27) より，$I_{11}, I_{10}, I_9, \ldots I_0$ の順に計算できる（後退型の漸化式）。このとき，$|a_n| = 1/10 < 1$ であるから，アルゴリズムは数値的に安定となる。

この 2 つの方法を用いて，単精度で $I_0, I_1, \ldots, I_{12}$ を計算した結果を表 1.2 に示す。ここで，$I_n^{(f)}$，$I_n^{(b)}$ がそれぞれ前進型，後退型の漸化式による結果である。また，I_n は高精度の数値積分法である二重指数型数値積分公式（第 4 章参照）を用い，積分 (1.24) を各 n に対して直接計算した結果である。$I_n^{(f)}$，$I_n^{(b)}$ では，下線の部分が I_n と一致する桁を示す。表より，$I_n^{(b)}$ ではすべての n に対して 7 桁まで数値が一致するのに対し，$I_n^{(f)}$ では n が増えるごとに有効な数字がほぼ 1 桁ずつ減り，$n = 6$ 以上では全くでたらめな結果になってしまうことがわかる。これは，この場合に前進型の漸化式が不安定であり，1 ステップごとに誤差が 10 倍に拡大されることの結果である。

以上ではもっとも簡単な漸化式の場合を例にとって説明したが，数値計算アルゴリズムの多くは同様な繰り返し計算を含むため，誤差の伝播や安定性について十分な注意を払うことが必要となる。常微分方程式の解法，偏微分

表 1.2: 漸化式による積分 I_n の計算

n	$I_n^{(f)}$	$I_n^{(b)}$	I_n
0	$\underline{9.685340} \times 10^{-2}$	$\underline{9.685340} \times 10^{-2}$	9.685340×10^{-2}
1	$\underline{3.146594} \times 10^{-2}$	$\underline{3.146591} \times 10^{-2}$	3.146591×10^{-2}
2	$\underline{1.867}385 \times 10^{-2}$	$\underline{1.867415} \times 10^{-2}$	1.867415×10^{-2}
3	$\underline{1.32}6142 \times 10^{-2}$	$\underline{1.325843} \times 10^{-2}$	1.325843×10^{-2}
4	$\underline{1.02}4290 \times 10^{-2}$	$\underline{1.027281} \times 10^{-2}$	1.027281×10^{-2}
5	$\underline{8}.682028 \times 10^{-3}$	$\underline{8.382945} \times 10^{-3}$	8.382945×10^{-3}
6	4.088806×10^{-3}	$\underline{7.079637} \times 10^{-3}$	7.079637×10^{-3}
7	3.603500×10^{-2}	$\underline{6.126699} \times 10^{-3}$	6.126699×10^{-3}
8	-2.936834×10^{-1}	$\underline{5.399674} \times 10^{-3}$	5.399674×10^{-3}
9	2.995657×10^{0}	$\underline{4.826784} \times 10^{-3}$	4.826784×10^{-3}
10	-2.990394×10^{1}	$\underline{4.363733} \times 10^{-3}$	4.363733×10^{-3}
11	2.990870×10^{2}	$\underline{3.981709} \times 10^{-3}$	3.981709×10^{-3}
12	-2.990827×10^{3}	$\underline{3.661163} \times 10^{-3}$	3.661163×10^{-3}

方程式の解法，連立一次方程式の解法など，それぞれの場合にどうやって安定なアルゴリズムを構成するかについては，今後，第 6 章，第 7 章，第 8 章で詳しく学ぶ。

第 1 章の問題

問題 1

(1) 浮動小数点数 ϵ で，計算機上で $1+\epsilon$ を計算して丸めた結果が 1 より大きくなるような（すなわち情報落ちが起きないような）最小の数を**マシンイプシロン**と呼ぶ。IEEE 方式の単精度で，丸めに 0 捨 1 入を使った場合のマシンイプシロンを求めよ。結果は 10 進数で表示せよ。

(2) 同じ条件で倍精度に対するマシンイプシロンを求めよ。

問題 2

$|x| \ll 1$ のとき，以下の各式の計算において，桁落ちが生じる可能性があるかどうか判定せよ。可能性がある場合，どのように式を変形すれば桁落ちが避けられるかを述べよ。ただし，平方根，4 乗根，sin などの関数は高精度に計算できるとする。

(1) $\sqrt{1+x} - \sqrt{1-x}$

(2) $\sin(1+x) - \sin(1-x)$

(3) $\sqrt[4]{1+x} - 1$

第2章　非線形方程式の解法

2.1　二分法

本章では，方程式 $f(x)=0$ の解の計算を取り上げる。$f(x)=0$ が 4 次以下の代数方程式の場合は解析的に解を求める公式が存在するが，$x^5-4x+2=0$ のような 5 次以上の代数方程式の場合，あるいは $e^x-3x=0$ のような超越方程式の場合には一般にこのような公式が存在せず，反復計算によって解を求める必要がある。ここではそのための方法として，二分法とニュートン法について学ぶ。本節では二分法を取り上げる。

特徴　二分法は次の特徴を持つ解法である。

- $f(x)$ は連続関数でさえあればよいため，広い範囲の方程式に対して適用できる。
- 解の誤差の上限が 1 ステップごとに 1/2 になることが保証されており，安定な解法である。
- 収束は遅い（定量的な収束速度については後述）。

原理　二分法の原理は，次の中間値の定理である。

中間値の定理

区間 $[a,b]$ において，$f(a)>0$ かつ $f(b)<0$ ならば,$f(x)=0$ の解が区間内に少なくとも 1 つ存在する $(f(a)<0$ かつ $f(b)>0$ でもよい$)$ 。

$f(a)=0$ あるいは $f(b)=0$ なら $x=a$ あるいは $x=b$ が解であるから，この場合も含めて言い換えると次のようになる。

中間値の定理（言い換え）

$f(a)f(b)\le 0$ なら，$[a,b]$ 内に解が少なくとも 1 つ存在する。

二分法では，まず $f(a_0)f(b_0)\le 0$ を満たす初期区間 $[a_0,b_0]$ を定める。中間値の定理より，この区間は解を少なくとも 1 個含む。次に，区間の中点

$c = (a_0 + b_0)/2$ を計算し，次のようにして解を含む区間の幅を $1/2$ に縮小する。

1. $f(a_0)f(c) \leq 0$ なら，区間 $[a_0, c]$ に解が存在。
2. $f(a_0)f(c) > 0$ なら，区間 $[c, b_0]$ に解が存在。

これを繰り返して解を含む区間の幅を縮めていき，幅が予め定めた解の要求精度 ε 以下になったら終了する。

アルゴリズム　二分法をアルゴリズムの形で書くと，次のようになる。ただし，変数 a0，b0，eps はそれぞれ a_0，b_0，ε を表す。

```
input(a0,b0,eps)
do until |a-b|<eps
   c = (a+b)/2
   if f(a)f(c)<=0 then
      b:=c
   else
      a:=c
   end if
end do
c=(a+b)/2
output(c)
```

収束速度　二分法の第 n ステップでの近似解を

$$c_n = \frac{a_n + b_n}{2} \tag{2.1}$$

とする。真の解を ξ とすると，ξ は区間 $[a_n, b_n]$ 中にあるから，

$$|c_n - \xi| \leq \frac{|a_n - b_n|}{2} = \frac{|a_0 - b_0|}{2}\left(\frac{1}{2}\right)^n \tag{2.2}$$

となる。一般に，

$$|c_n - \xi| \leq Dr^n \quad (|r| < 1) \tag{2.3}$$

のとき，c_n は収束率 r で ξ に *1* **次収束**するという。したがって，二分法による近似解は収束率 $\frac{1}{2}$ で 1 次収束することがわかる。

誤差　解の誤差の上限は毎回 $\frac{1}{2}$ になる。

n=1　0.****
n=2　0.0***
n=3　0.00**

⇒　有効数字は，２進で１桁ずつ増える。

表 2.1: 二分法による $\sqrt{2}$ の計算

n	a_n	b_n	c_n	$f(c_n)$
0	1	2	1.5	0.250000
1	1	1.5	1.25	-0.437500
2	1.25	1.5	1.375	-0.109380
3	1.375	1.5	1.4375	-0.066406
4	1.375	1.4375	1.40625	-0.022460
5	1.40625	1.4375	1.421875	0.021729
6	1.40625	1.421875	1.414063	-0.000430
7	1.414063	1.421875	1.417969	0.010635
8	1.414063	1.417969	1.416016	0.005100
9	1.414063	1.416016	1.415039	0.002336
10	1.414063	1.415039	1.414551	0.000954

数値例　$f(x) = x^2 - 2$，$a_0 = 1$，$b_0 = 2$ として二分法により $\sqrt{2}$ を計算した例を表 2.1 に示す（築山知弘君のレポートより）。c_n の項を見ると，3〜4 反復に 1 桁の割合で正しい値 $1.414213\cdots$ に近づいていることがわかり，収束率 $1/2$ で 1 次収束していることが確認できる。

使用の際の注意事項 1　$f(a_0)f(b_0) \leq 0$ という条件だけでは，区間 $[a_0, b_0]$ に含まれる解は 1 つとは限らない。複数個の解がある場合，二分法はそのうちのどれに収束するかわからないため，予め解が 1 つだけ存在する区間を初期区間とするか，あるいは求まった解が欲しい解かどうか判定する手段を用意するなどの注意が必要である。

使用の際の注意事項 2　一般には，$f(c)$ の計算結果は，ある誤差 δ を含む。もし，

$$f(c) = \delta' > 0$$

において $\delta' \sim \delta$ であると，$f(c)$ が本当に正なのか，それとも本当は負で数値誤差のために正になっているのか分からず，解が $x = c$ のどちら側にあるのかは判定できない。したがって，これ以上の繰り返しは無意味である。$f(c)$ に含まれる誤差の大きさ δ に見当が付いているならば，$|f(c)| < \delta$ を停止条件とするのがよい。

2.2　ニュートン法（単根の場合）

二分法は広い範囲の方程式に対して安定に解を求めることができる解法であるが，前節の例からもわかるように，収束が遅い。そこで，より収束の速

い解法として，ニュートン法を紹介する。

特徴　ニュートン法は次の特徴を持つ解法である。

- $f(x)$ が C^2 級（2 回微分可能かつ 2 階までの微係数が連続な関数）の場合に適用できる。
- 収束は極めて速い（定量的な収束速度については後述）。
- 近似解は必ずしも単調に真の解に近づくとは限らず，発散したり振動したりすることがある。安定に計算を進めるには，真の解に近い初期値から始めることが必要である。
- 関数値 $f(x)$ だけでなく，微係数 $f'(x)$ も計算することが必要である。

原理　ニュートン法では，初期値 x_0 が与えられたとき，その点で $f(x)$ を次のように 1 次関数で近似する。

$$\begin{aligned} f(x) &= f(x_0) + f'(x_0)(x-x_0) + \frac{1}{2}f''(x_0)(x-x_0)^2 + \cdots \\ &= f(x_0) + f'(x_0)(x-x_0) + O((x-x_0)^2) \\ &\simeq f(x_0) + f'(x_0)(x-x_0) \end{aligned} \tag{2.4}$$

ここで，$O(\epsilon)$ は微少量 ϵ と同程度かそれ以下の大きさの数を表す。この近似式に基づいて $f(x)=0$ となる x を求めると，

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{2.5}$$

となる。これを新しい近似解とする。この手続きを繰り返し，近似解 $x_1, x_2, x_3, \ldots$ を次々に求めていく。収束判定は $f(x_n)$ の値で行い，予め定めた小さい数 δ に対して，$|f(x_n)| < \delta$ となったら終了する。

アルゴリズム　ニュートン法をアルゴリズムの形で書くと，次のようになる。ただし，変数 x0，delta はそれぞれ x_0，δ を表す。

```
input(x0,delta)
do until |f(x)|<delta
   x = x0 - f(x0)/f'(x0)
   x0 = x
end do
output(x)
```