シミュレーション概論ノート第 4 講: 数値積分について

1 全般的注意

連続関数 $f(x)$ が与えられた場合に

$$\int_a^b dx f(x) \tag{1}$$

を求める事は実は大変であり、殆んどの場合は初等関数ではおろか特殊関数としても表現できない。そこで数値積分を用いる必要が多くの場合生じる。原理は離散点での関数の値を求めてその和を取るということであるが実際の運用では様々な方法・工夫がある。

最も簡単な手法として知られているのは既に丸め誤差の例で用いた台形公式である。今回はその刻みの最適化について考えてみる。次にシンプソンの公式について説明する。台形公式が離散点の間を 1 次関数で近似したのに対してこの方法では 2 次関数で近似する。シンプソン公式は古典的な数値解析の本では必ず触れられている手法である。最後に 2 重指数関数型積分法 (Double Exponential method=DE 法) について触れる。[1] この方法は現在では数値解析の専門家の間で標準的手法として用いられている。DE 法は特に端点に特異性のある積分や半無限区間の積分に強く、また次第に精度を向上させていく反復型の自動プログラムも作成しやすい。その便を図るために原理的な面だけ説明をしておこう。

2 台形公式

台形公式の概要は既に説明したので今回はより実際的な点を考慮してアルゴリズムを説明しよう。即ち、分点の数をいかに選び、どう効率的なプログラムを組むかという点である。

まず分点数を最適化するためには積分値がある程度収束したところで計算を打ち切る操作をプログラム中に組み込む必要がある。何も考えずに刻みを細かくしていくと丸め誤差によってかえって真の値からかけ離れた結果を得ることになる。次に分点の数を増やした場合にどう効率的にプログラムを組むかという問題に対しては分点を前の分点の倍取るという操作を続け、古い分点に対しては既に得られた計算結果を利用することで計算時間を半分にすることができる。

この分点の取り方についてもう少し詳しく説明をすれば以下の様になる。分割数 $N = 2^n$ における台形公式による答えを T_n とする。この時、台形の幅 h は $h = (b-a)/N$ で与えられる。すると

$$\begin{aligned} T_n &= \frac{b-a}{2^n}\{\frac{f(a)+f(b)}{2} + \sum_{j=1}^{2^n-1} f(a+j\frac{b-a}{2^n})\} \\ &= \frac{b-a}{2^n}\{\frac{f(a)+f(b)}{2} + \sum_{j=1}^{2^n-1} f(a+2j\frac{b-a}{2^{n+1}})\}, \\ T_{n+1} &= \frac{b-a}{2^{n+1}}\{\frac{f(a)+f(b)}{2} + \sum_{j=1}^{2^{n+1}-1} f(a+j\frac{b-a}{2^{n+1}})\} \end{aligned} \tag{2}$$

[1] しかし実際に講義ではプログラムを組まない。興味がある人は 1998 年の講義ノートを参照せよ。

となる。従って

$$T_{n+1} = \frac{T_n}{2} + \frac{b-a}{2^{n+1}} \sum_{j=1}^{2^n-1} f(a + (2j-1)\frac{b-a}{2^{n+1}}) \tag{3}$$

という漸化式を得る。ここで $T_0 = (b-a)(f(a)+f(b))/2$ とおけば逐次的に計算できる。

以上の事を頭に入れてアルゴリズム考えると以下の通りにすればいいことは容易に理解できる。

1. $f(x)$ を与える。ϵ を設定する。初期値として分割数 2 より $n = 1$ 及び $h = b - a$, $T = h(f(a) + f(b))/2$.
2. 以下

   n=2n

   h=h/2

   s=0

   $i = 1, 3, \cdots, n-1$ の順に

   $s = s + f(a + ih)$

   繰り返す

   $newT = T/2 + hs$

   if $|newT - T| < \epsilon |newT|$ goto (3)

   $T = newT$

   を繰り返す
3. $newT$ を答とする。

例題として $f(x) = e^x$ として区間 $[0, 1]$ で積分を実行してみる。$\epsilon = 10^{-6}$ と設定してみて、上のアルゴリズムをそのまま Fortran に翻訳すると

```
external f

eps=1.0e-6

b=1.0
a=0.0
n=1

h=b-a

t=h*(f(a)+f(b))/2.0

do k=1,15

  n=2*n
  h=h/2.0
```

```
            s=0.0

            do i=1,n-1,2
             s=s+f(a+real(i)*h)
            enddo

            tn=t/2.0+h*s

            if (abs(tn-t).lt.eps*abs(t)) goto 1

            t=tn
          enddo

 1        write(*,*) tn,k
          end

          real function f(x)
          f=exp(x)
          end
```

となる。この問題では 10 回の繰り返しで望みの精度が得られて 1.71828 という値を得る。勿論、この答えは $e - 1 = 1.7182818\cdots$ と比較して設定通り 6 桁の正しい結果を得る。

台形公式を上のアルゴリズムに従って C でプログラムを組むと次の様になる。

```
#include <stdio.h>
#include <math.h>

float f(float);

main()
{
  int i, k, n;
  float a, b, h, s, t, x, tn, eps;

  eps = 1.0e-6;

  b = 1.0;
  a = 0.0;

  n = 1;
```

```
  h = b - a;

  t = h * (f(a) + f(b)) / 2.0;

  for(k = 1; k <= 15; k++) {
    n = 2 * n;
    h = h / 2.0;
    s = 0.0;
    for(i = 1; i <= n - 1; i += 2) {
      s = s + f(a + (float)i * h);
    }

    tn = t / 2.0 + h * s;

    if(fabs(tn - t) < eps * fabs(t)) break;

    t = tn;
  }

  printf("%12.8f %d\n", tn, k);
}

float f(float x)
{
  return exp(x);
}
```

Fortran90 自由形式の台形公式を使ったプログラム（倍精度）は以下の通りである。

```
real(8),external::f
real(8)::eps,a,b,h,s,tn,t
eps=1.0d-10
b=1.0_8
a=0.0_8
n=1
h=b-a
t=h*(f(a)+f(b))/2.0_8

do  k=1,30

    n=2*n
    h=h/2.0_8
    s=0.0_8
```

```
      do  i=1,n-1,2

            s=s+f(a+i*h)

      enddo

    tn=t/2.0_8+h*s
    if (abs(tn-t) < eps*abs(t)) exit
    t=tn

enddo
print*,tn,k

end

real(8) function  f(x)
real(8)::x
f=exp(x)
end
```

単純な置き換えに従っているので説明は不要である。

問１ $\int_0^2 dx \cos(x)$ を求めよ。(答え $\sin(2) \simeq 0.909297407$)

3　シンプソン公式

台形公式では被積分関数を折れ線近似した。この補間関数[2] の近似の精度を上げれば精度の高い結果が得られる事が期待できる。そこで折れ線近似を２次関数で近似することを考える。

２次関数で補間する事を論じるには３点の場合を考えてからそれを一般化すればいい。３点 x_1, x_2, x_3 を等間隔としてその点における関数値をそれぞれ f_1, f_2, f_3 とする。その間を $f(x) = a(x-x_2)^2 + b(x-x_2) + c$ で補間することを考えよう。$h = x_3 - x_2$ とおくと明らかに

$$\begin{aligned} f_1 &= ah^2 - bh + c \\ f_2 &= c \\ f_3 &= ah^2 + bh + c \end{aligned} \tag{4}$$

が成り立つ。これらを解くと直ちに

$$a = \frac{f_1 + f_3 - 2f_2}{2h^2}, \quad b = \frac{f_3 - f_1}{2h}, \quad c = f_2 \tag{5}$$

が得られるので $y = x - x_2$, $g(y) = f(x)$ に対して

$$\int_{-h}^{h} g(y)dy = \frac{h}{3}[f_1 + 4f_2 + f_3] \tag{6}$$

[2] 関数の補間についてはいずれ詳しく学習する予定。

を得る。３点を一般化する場合には積分の近似値 S_N を

$$\begin{aligned} S_N &= \frac{h}{3}\sum_{j=0}^{N-1}[f(x_{2j})+f(x_{2j+2})+4f(x_{2j+1})] \\ &= \frac{h}{3}[f(a)+f(b)+4\sum_{j=1}^{N}f(x_{2j-1})+2\sum_{j=1}^{N-1}f(x_{2j})] \end{aligned} \tag{7}$$

とおけばいい。上式で積分を近似する方法をシンプソンの公式といい誤差が N^{-4} に比例することが知られている。(その評価については各自の練習問題とする)。

さてシンプソン公式を素直に翻訳してプログラムを組むことは易しいが効率は悪い。前節を参考にして漸化式を用いるとより効率的なアルゴリズムになる。つまり分割数を $N=2^n(n=1,2,\cdots)$ として n を順次増やして答えが p 桁変化しなくなった時点で終了する。

$N=2^n$ のときの台形則、及びシンプソン則による近似値をそれぞれ T_n, S_n とする。すると双方の間には

$$S_{n+1}=\frac{4}{3}T_{n+1}-\frac{1}{3}T_n \tag{8}$$

という関係がある。実際、$h=(b-a)/2^{n+1}$ に対して

$$\begin{aligned} \frac{4}{3}T_{n+1}-\frac{1}{3}T_n &= \frac{4h}{3}[\frac{f(a)+f(b)}{2}+\sum_{j=1}^{N}f(a+jh)] \\ &\quad -\frac{2h}{3}[\frac{f(a)+f(b)}{2}+\sum_{j=1}^{N-1}f(a+2jh)] \\ &= \frac{h}{3}[f(a)+f(b)+4\sum_{j=1}^{N}f(x_{2j-1})+2\sum_{j=1}^{N-1}f(x_{2j})] \end{aligned} \tag{9}$$

となる。従って台形公式から帰納的に S_n を求めることが出来る。

アルゴリズムは

1. $\epsilon=10^{-p}$,

   $N:=2$,

   $h:=(b-a)/2$,

   $T:=h\{f(a)+f(b)+2f((a+b)/2)\}/2$,

   $S:=h\{f(a)+f(b)+4f((a+b)/2)\}/3$

2. Loop

   $N:=2N, h=h/2$

   $s=0$

   - Loop($i=1,3,5,\cdots,N-1$)
     s=s+f(a+ih)
   - 繰り返す

newT=T/2+h s

newS=(4newT-T)/3

If $|newS - S|/|newS| < \epsilon$ goto 3

$T = newT, S = newS$

繰り返す。

3. newS を答えとする。

である。

従って Fortran でプログラムを書くと

```
      external f

      eps=1.0e-6
      n=2

      b=1.0
      a=0.0

     h=(b-a)/2.0

     xi=(a+b)/2.0
     t=h*(f(a)+f(b)+2.0*f(xi))/2.0
     ss=h*(f(a)+f(b)+4.0*f(xi))/3.0

     do k=1,20
      n=2*n
      h=h/2.0
      s=0.0
      do i=1,n-1,2
        s=s+f(a+real(i)*h)
      enddo

      tn=t/2.0+h*s
      sn=(4.0*tn-t)/3.0

      if (abs(sn-ss).lt.eps*abs(sn)) goto 3

      write(*,*) sn-ss,sn,k
       t=tn
       ss=sn

    enddo
```

```
 3      write(*,*) sn,k
        end

        real function f(x)
        f=exp(x)
        end
```

Cでは

```
#include <stdio.h>
#include <math.h>

float f(float);

main()
{
  int i, k, n;
  float a, b, h, s, t;
  float xi, ss, sn, tt, tn, eps;

  eps = 1.0e-6;
  n = 2;

  b = 1.0;
  a = 0.0;

  h = (b - a) / 2.0;

  xi = (a + b) / 2.0;
  t = h * (f(a) + f(b) + 2.0 * f(xi)) / 2.0;
  ss = h * (f(a) + f(b) + 4.0 * f(xi)) / 3.0;

  for(k = 1; k <= 20; k++) {
    n = 2 * n;
    h = h / 2.0;
    s = 0.0;
    for(i = 1; i <= n - 1; i += 2) {
      s = s + f(a + (float)i * h);
    }

    tn = t / 2.0 + h * s;
    sn = (4.0 * tn - t) / 3.0;
```

```
    if(fabs(sn - ss) < eps * fabs(sn)) break;

    t = tn;
    ss = sn;
  }

  printf("%12.8f %d\n", sn, k);
}

float f(float x)
{
  return exp(x);
}
```

である。

Fortran 90 自由形式でのシンプソン公式を使ったプログラム（倍精度）は以下の通り：

```
real(8),external::f
real(8)::eps,a,b,h,xi,t,ss,tn,sn,s
eps=1.0d-10
n=2
b=1.0_8
a=0.0_8
h=(b-a)/2.0_8
xi=(a+b)/2.0_8
t=h*(f(a)+f(b)+2.0_8*f(xi))/2.0_8
ss=h*(f(a)+f(b)+4.0_8*f(xi))/3.0_8


do k=1,20

   n=2*n
   h=h/2.0_8
   s=0.0_8

   do  i=1,n-1,2

       s=s+f(a+i*h)

   enddo
```

```
   tn=t/2.0_8+h*s
   sn=(4.0_8*tn-t)/3.0_8
   if(abs(sn-ss) < eps*abs(sn)) exit
   print*,sn-ss,sn,k
   t=tn
   ss=sn

enddo

print*,sn,k
end

real(8)function f(x)
real(8)::x
f=exp(x)
end
```

注意すべきはアルゴリズム中における s と S の混同を避けなければいけない点である。望むべくはアルゴリズムを書く段階で混乱がないように異なった notation を用いるのがよい。いずれにしても正しくプログラムされていれば 4 回のループで計算が終了し、その収束の早さは台形公式に比べて格段に改善されている。

問 2 $\int_0^2 dx \cos(x)$ をシンプソン公式を用いて求めよ。また $\displaystyle\int_0^{1/4} \frac{dx}{\sqrt{(1+4x^2)(1+3x^2)}}$ を台形公式、シンプソン公式を用いて求めてみよ。(答え：2 番目のものは 0.233859524)

このように補間を上げて積分を実行することでよりよい計算が実行できるのは当然のことと言えよう。この方向ではニュートンーコーツ法と呼ばれる手法 (補間関数として 8 次式を使う) があり成功を収めているがここでは触れない。

4 DE 公式による数値積分

さて台形公式とシンプソン公式だけ知っていれば充分であろうか。またこれでは Mathematica 等の数式処理ソフトを持っていれば充分の様な感じも受けるのではなかろうか。ところが実際には次の問題は台形公式、シンプソン公式、数式処理ソフトのいずれもそのままでは実行できない問題である。

問 3 $\int_{-1}^{1} dx/\sqrt{1-x^2}$ を台形公式、シンプソン公式を用いて求めてみよ。

この問いを実行すると計算機はオーバーフローを起こして計算は止まってしまう。原因は明白である。即ち、端点での発散を計算機が拾ってしまうからである。一方、この計算結果は $x = \sin\theta$ とおけば直ちにわかるように π という収束値を持つ。また（半）無限区間での積分に対しても従来の方法では両端を順に動かしてその収束性を見るような原始的手法しか使えない。

このような問題点を全て解決して更に一般の場合も精度が良く、計算効率の高い欲張った方法はないのだろうか。実はあるのである。それが DE 公式 (Double Exponential Formula) である。

問 4 次の積分を計算するアルゴリズムについて考える。

$$I_n = \int_0^1 dx x^n e^{x-1} \quad (n = 1, 2, \cdots)$$

(1) 部分積分を用いて I_n を計算する漸化式を導け。

(2) (1) の漸化式を用いて $I_1, \cdots, I_{15}$ 迄求めよ。

(3) $\lim_{n\to\infty} I_n$ の値と比較することから上のアルゴリズムは数値的に不安定であることを示せ。

(4) 安定なアルゴリズムを求め、$I_1, \cdots, I_{15}$ の良い近似値を求めよ。

4.1 DE 公式の原理

関数 $f(x)$ の定積分を数値的に求める場合、もっとも簡単な方法は台形公式に基づくものであることは言うまでもない。台形公式を有限区間の普通の積分に適用すると精度が悪くて実用に耐えないが無限区間の積分

$$I = \int_{-\infty}^{\infty} dt g(t) \tag{10}$$

に適用し、等間隔の刻み h で分布する分点で

$$I_h = h \sum_{i=-\infty}^{\infty} g(ih) \tag{11}$$

とすると高精度の結果が得られることが知られている。

問 5 $I = \int_{-\infty}^{\infty} dt e^{-t^2} = \sqrt{\pi}$ を (11) を用いて求めてみよ ($h = 0.5$)。

従って (1) を求めるのに変数変換

$$x = \varphi(t) \tag{12}$$

によって積分区間を $(-\infty, \infty)$ の積分

$$I = \int_{-\infty}^{\infty} f(\varphi(t))\varphi'(t) \tag{13}$$

にしておいてから台形公式を用いると精度の高い結果を得る。(13) に台形公式を用いると

$$I_h = h \sum_{i=-\infty}^{\infty} f(\varphi(ih))\varphi'(ih) \tag{14}$$

とすることになる。但し、現実には無限和は取れないので有限項で打ち切って

$$I_h^{(N)} = h \sum_{i=-N_-}^{N_+} f(\varphi(ih))\varphi'(ih) : \quad N = N_+ + N_- + 1 \tag{15}$$

を実行する。

次に $\varphi(t)$ をどのように選ぶといいかという問題があるが与えられた積分が

$$I = \int_{-1}^{1} d\xi f(\xi) \tag{16}$$

を考えて[3]

$$\xi = \varphi(t) = \tanh(\frac{\pi}{2}\sinh t) \tag{17}$$

と取ればいいことが知られている。従って (15) は

$$I_h^{(N)} = h \sum_{i=-N_-}^{N_+} f(\tanh(\frac{\pi}{2}\sinh(ih)))\frac{\frac{\pi}{2}\cosh(ih)}{\cosh^2(\frac{\pi}{2}\sinh(ih))} \tag{18}$$

となる。変換 (17) を用いると (13) の被積分関数は $|t|$ の大きくなると

$$f(\varphi(t))\varphi'(t) \sim A\exp[-c\exp|t|] \tag{19}$$

となることが知られている[4] 。この (19) こそが DE 公式 (Double Exponential Formula) と呼ばれる所為である。

この方法は端点に特異点があっても有効な方法であることが知られている。区間中に特異点が離散的に存在する場合はそこを端点に選び積分を実行できる。

4.2 DE 公式による自動積分

自動積分とは被積分関数、区間、許容誤差を与えたときに結果として積分の近似値を期待される精度で与えるプログラムのことである。

ところで既に繰り返し見た通り計算時間の節約のためには刻み幅を変えた場合に元の結果を使うことが必要になってくる。従って刻み幅を h から $h/2$ に変えた場合に

$$I_{h/2} = \frac{1}{2}\{I_h + h\sum_{j=-\infty}^{\infty} f(\varphi((2j+1)h/2))\varphi'((2j+1)h/2)\} \tag{20}$$

という形でプログラムを書くべきである。また誤差 $\Delta I_h = I - I_h$ は $\Delta I_h \sim \exp(-C/h)$ であることが知られているので刻みを半分にすると $\Delta I_{h/2} \sim \exp(-2C/h) \sim (\Delta I_h)^2$ となるので有効桁数はほぼ倍になる。このように DE 公式の収束は極めて早い。

この事実を積分の打ち切りの判定条件に利用することができる。即ち $I_{h/2}$ は充分 I_h より真の値に近いので $\Delta I_h \sim I_{h/2} - I_h$ となり、$\sqrt{|\Delta I_{h/2}|} \sim |I_{h/2} - I_h|$ が成立する。従って与えられた許容誤差を ϵ として

$$|I_{h/2} - I_h| < c_1\sqrt{\epsilon} \tag{21}$$

を打ち切りの条件と出来る。ここで c_1 は 1 程度の適当な定数で安全を見込んで $c_1 = 0.2$ とでもしておけばいい。

DE 公式では無限和を打ち切る操作が必要になる。積分 (13) をある $t = \tau$ で打ち切ると (19) から打ち切られた部分は

$$\int_\tau^\infty dt f(\varphi(t))\varphi'(t) \sim A\int_\tau^\infty dt\exp[-e^t] \le \frac{A}{e^\tau}\int_{e^\tau}^\infty e^{-s}ds \sim e^{-\tau}f(\varphi(\tau))\varphi'(\tau) \tag{22}$$

従って甘く見積もって

$$|f(\varphi(\tau))\varphi'(\tau)| < \epsilon \tag{23}$$

[3] $x = (b-a)\xi/2 + (b+a)/2$ を用いると (1) と (16) は相互変換可能である。

[4] 証明は手書き原稿を見よ。

を満たす $t=\tau$ で打ち切ればいい。

DE 公式でプログラムを組む場合に注意しなければならないのはオーバーフローと桁落ちである。オーバーフローは (18) の分母にある $\cosh^2((\pi/2)\sinh(ih))\sim(1/4)\exp[\frac{\pi}{2}\exp|ih|]$ から生じ得る。例えば 32 ビット１６進計算をすると浮動小数点数の最大値は既に述べた様に 7.2×10^{72} であるがこれは $|ih|\sim4.73$ に過ぎない。従って分点はオーバーフローの上限で打ち切る必要がある。

次に端点での特異性が強い時に桁落ちが起こるのでそれを避けるのに特殊な工夫が必要である。具体的には

$$I=\int_{-1}^{1}dxf(x),\quad f(x)=(1-x^2)^{\alpha},\alpha>-1 \tag{24}$$

を考えてみよう。端点における特異性が強いとは α が-1 に近いという意味である。$\alpha<0$ であれば (13) は $f(\varphi(t))$ が増大するが (19) にあるような $\varphi'(t)$ の強い減衰によって発散は押えられる。しかし、α が-1 に近いと f の増大が著しくなりそのままの形では DE 公式は使えない。[5] しかし問３程度の端点の特異性であれば直接 DE 公式を適用できるので実際に計算を行ない有効性を確かめてみよう。具体的プログラムは森の本の他に簡略化されたものが 1998 年の本講義ノートで公開されている。参考にされたい。

5　補足：２重指数関数型公式の収束性について

本講で述べた様に無限区間の台形公式に高度収束性に基づき DE 公式は定式化してある。つまり、関数 $f(x)$ の定積分を数値的に求める場合、もっとも簡単な方法は台形公式に基づくものであることは言うまでもない。台形公式を有限区間の普通の積分に適用すると精度が悪くて実用に耐えないが無限区間の積分

$$I=\int_{-\infty}^{\infty}dtg(t) \tag{25}$$

に適用し、等間隔の刻み h で分布する分点で

$$I_h=h\sum_{i=-\infty}^{\infty}g(ih) \tag{26}$$

とすると高精度の結果が得られることが知られている。このことの理由を考えてみよう[6] 。より詳しくは離散化誤差

$$E_D=I_h-I \tag{27}$$

が $1/h$ に関して指数関数的に急激に現象する。但し以下の議論で $D(d)=\{x\in\mathbf{C}||Imz|<d\}$ は複素平面上の幅 $2d$ の実軸に平行な帯状領域を表す。

定理１。 $D(d)$ で正則な関数 $f(z)$ が次の２条件を満たす。

(1) 任意の c $(0<c<d)$ に対して

$$\Lambda(f,c)=\int_{-\infty}^{\infty}dx\{|f(x+ic)|+|f(x-ic)|\}$$

が存在し、更に

$$\lim_{c\to d-0}\Lambda(f,c)<\infty$$

[5] 無論、その点についても工夫がされていて高い精度で計算できる方法があるが本講のレベルを越えるので省略する。詳細は森正武著「FORTRAN77 数値計算プログラミング（岩波 1987)」を参照のこと。

[6] 議論は杉原正顯、室田一雄「数値計算法の数理」（岩波）による。

を満たす。

(2) 任意の c $(0 < c < d)$ に対して

$$\lim_{x\to\pm\infty}\int_{-c}^{c} dy|f(x+iy)| = 0$$

である。

以上の 2 条件を満たすときに、任意の $h > 0$ に対して

$$|E_D| \leq \frac{\exp(-2\pi d/h)}{1-\exp(-2\pi d/h)}\Lambda(f.d-0) \tag{28}$$

が成立する。

（証明） 実軸に平行な $(-(N+1/2)h \pm ic, (N+1/2)h, \pm ci)$ と虚軸に平行な $(\pm(N+1/2)h - ci, \pm(N+1/2)h + ci)$ で囲まれる積分路 R に沿って関数 $f(z)\cot(\pi z/h)$ を積分する。このとき、$\cot(\pi z/h)$ の $z = kh$ (k は整数）における留数が h/π である。実際、極 $z_0 = kh$ のまわりで $\cot(\pi z/h) \simeq h\cos^2(\pi z_0/h)/(\pi(z-z_0))$ と展開できるので留数は $Res[\cot(\pi z/h)] = h/\pi$ である。従って

$$\frac{1}{2i}\int_R dz f(z)\cot(\frac{piz}{h}) = h\sum_{k=-N}^{N} f(kh)$$

である。ここで $N \to \infty$ で右辺は I_h となるので、条件 (2) から実軸に平行な積分の寄与のみが残る。よって

$$I_h = \frac{1}{2i}\int_{-\infty}^{\infty} dx[-f(x+ic)\cot\frac{\pi(x+ic)}{h} + f(x-ic)\cot\frac{\pi(x-ic)}{h}]$$

である。一方、条件 (2) を使えば

$$I = \frac{1}{2}\int_{-\infty}^{\infty} dx\{f(x+ic) + f(x-ic)\}$$

と書き直せるので

$$I_h - I = \int_{\infty}^{\infty} dx[f(x+ic)g(x+ic) - f(x-ic)g(x-ic)]$$

となる。但し

$$g(x) = 2i(1-\cot\frac{\pi z}{h}) = \frac{\exp\frac{2\pi x}{h}}{1-\exp\frac{2\pi x}{h}}$$

である。従って

$$I_h - I \leq g(ic)\int_{-\infty}^{\infty} dx\{|f(x+ic)| + |f(x-ic)|\} = g(ic)\lambda(f,c)$$

となる。ここで $c \to d-0$ とおくと定理は証明されたことになる。（証明終り）。

このように無限区間では台形公式が非常に強い収束性を持つことが証明された。無限区間上の台形公式 (26) は無限和なので、実際には適当な項数で打ち切った有限和

$$I_h^* = h\sum_{k=-N}^{N} f(kh) \tag{29}$$

を用いる。積分式 I_h^* の誤差は離散化誤差 E_D と打ち切り誤差

$$E_I = I_h^* - I_h \tag{30}$$

の和である。従って $|E_I|/|E_D| \sim 1$ となるように N を選べば良い。

打ち切り誤差は

$$E_I = -h \sum_{|k|>N} f(kh)$$

で表されるので、その大きさは $f(x)$ の $x \to \pm\infty$ での減衰率に支配される。そこで $w(z)$ を $D(d)$ で 0 にならない正則関数として $w(z)$ に関して有界な関数の全体 $H(w,d)$ を考える。但し

$$H(w,d) = \{f(z) | f(z) : \text{regular} \in D(d); H \equiv \sup_{z \in D(d)} |f(z)/w(z)| < \infty\}$$

とし よう。$f(z) \in H(w,d)$ に対して

$$|E_I| \le hH \sum_{|k|>N} w(kh)$$

が成立する。また $\Lambda(f,c)$ の定義から $|f| \le H|w|$ を用いて

$$\Lambda(f,c) \le H\Lambda(w,c)$$

も成立する。これらと定理 1. を組み合わせると

定理 2. $f(z) \in H(w,d)$ に対して

$$|I_h^* - I| \le |E_D| + |E_I| \le H[g(d)\Lambda(w, d-0) + h \sum_{|k|>N} w(kh)] \tag{31}$$

が成立する。但し $\Lambda(w, d-0) < \infty$ であるとする。

ここで $d < \pi/2$ として天下り的だが

$$w(z) = w_{DE}(z) = \frac{d}{dz} \tanh(\frac{\pi}{2} \sinh z) = \frac{\frac{\pi}{2}\cosh(z)}{\cosh^2(\frac{\pi}{2}\sinh(z))} \tag{32}$$

とおいてみよう。このとき、$x \to \pm\infty$ で $\cosh x \simeq \sinh x \simeq \exp[|x|]/2$ であるから

$$w_{DE}(x) \simeq \pi \exp[-\frac{\pi}{2}\exp[|x|] + |x|] \tag{33}$$

のように 2 重指数関数的に減衰する。このことを利用して $|E_D| \simeq |E_I|$ となるように (28) 式に基づき $2\pi d/h = (\pi/2)\exp(Nh)$ と選ぶと

$$|I_h^* - I| \le C'H \exp[-CN/\ln N] \tag{34}$$

の形の評価式が得られる。これは分点数を倍にすると有効桁数が倍近くになることを意味する。

このように 2 重指数関数的に減衰する被積分関数に対して台形公式は有効である。この事実をうまく利用するために積分区間を $(-\infty, \infty)$ の積分

$$I = \int_{-\infty}^{\infty} f(\varphi(t))\varphi'(t) \tag{35}$$

にしておいてから台形公式を用いよう。(13) に台形公式を用いると

$$I_h = h \sum_{i=-\infty}^{\infty} f(\varphi(ih))\varphi'(ih) \tag{36}$$

とすることになる。但し、現実には無限和は取れないので有限項で打ち切って

$$I_h^{(N)} = h \sum_{i=-N_-}^{N_+} f(\varphi(ih))\varphi'(ih) : \quad N = N_+ + N_- + 1 \tag{37}$$

を実行する。

次に $\varphi(t)$ をどのように選ぶといいかという問題があるが与えられた積分が

$$I = \int_{-1}^{1} d\xi f(\xi) \tag{38}$$

を考えて[7]

$$\xi = \varphi(t) = \tanh(\frac{\pi}{2}\sinh t) = \frac{\frac{\pi}{2}\cosh(ih)}{\cosh^2(\frac{\pi}{2}\sinh(ih))} \tag{39}$$

と取ればいいことが知られている。従って (37) は

$$I_h^{(N)} = h \sum_{i=-N_-}^{N_+} f(\tanh(\frac{\pi}{2}\sinh(ih)))\frac{\frac{\pi}{2}\cosh(ih)}{\cosh^2(\frac{\pi}{2}\sinh(ih))} \tag{40}$$

となる。変換 (39) を用いると (35) の被積分関数は $|t|$ の大きくなると

$$f(\varphi(t))\varphi'(t) \sim A\exp[-c\exp|t|] \tag{41}$$

となることは上に示した通りである。この (41) こそが DE 公式 (Double Exponential Formula) と呼ばれる所為である。

この方法は端点に特異点があっても有効な方法であることが知られている。区間中に特異点が離散的に存在する場合はそこを端点に選び積分を実行できる。

[7] $x = (b-a)\xi/2 + (b+a)/2$ を用いると $\int_a^b f(x)dx$ と (16) は相互変換可能である。