応用複素関数 第 11 回

～ 数値積分 (1) ～

かつらだ まさし
桂田 祐史

2022 年 6 月 28 日

目次

本日の内容・連絡事項

- 先週、レポート課題を置いてある WWW サーバーに外部から攻撃がありました。そのため一時的に WWW サーバーを停止する措置を取りました。そのため、レポート課題の〆切を 1 週間延長しています。今後は Oh-o! Meiji にも課題文などを置くようにするつもりです。
- 既に話したことがありますが、昨年度と比べると進行がやや遅れています。大事なところで時間切れは避けたいと考えるので、板書はやめて、スライドを写しての説明に切り替えます。
- 今回から 3 回数値積分をテーマとする。今回はお話が主体 (あまり理屈は出て来ない) で、気軽に聞けると思います。講義ノート [1] を用意してあります。
- 数値積分について次回レポート課題 3 を発表する予定です。
- 期末レポート課題についても次回発表する予定です。

5 数値積分　5.1 はじめに

定積分 $I = \int_\Omega f(x)\,dx$ の値を数値計算で近似的に求めることを**数値積分** (numerical integration) という。

導関数を知っている関数を組み合わせた関数の導関数は (必ず) 計算できる。原理は簡単で、計算の効率を追求する話もある (自動微分 (automatic differentiation))。

しかし積分の計算はしばしば難しい。それで数値積分の出番となる。

5.2 次元ごとに方法を変える

まず次元、すなわち $\Omega \subset \mathbb{R}^n$ の n, あるいは f の変数の個数 ($x = (x_1, \cdots, x_n)$ の n).

1. 1 次元の場合、色々優れた公式 (後述) がある。
次元がとても低ければ、重複積分にして、これらの公式が使えるかもしれない。
2. 高次元の場合、**モンテ・カルロ法** (Monte Carlo methods) くらいしかやりようがない。
Ω に一様分布する乱数 $x_1, \cdots, x_N$ を用いて

$$I_N := \frac{1}{N} \sum_{k=1}^{N} f(x_k) \times (\Omega \text{ の体積})$$

で $I = \int_\Omega f(x)\,dx$ を推定する。$f \in L^2(\Omega)$ という緩い仮定のもとで

$$\text{推定値の標準偏差} = O\left(\frac{1}{\sqrt{N}}\right)$$

が知られている。つまり「誤差を 1 桁小さくするには、N を 100 倍にする」…ゾッとするほど低効率？そもそも扱っている問題が違うので比較する方がおかしい。**モンテ・カルロ法は最後の手段。**
(それなのに、$n = 1, 2$ の問題をモンテ・カルロ法の例として説明することが…)

3. 中間の次元、1 次元の公式への帰着が難しい、モンテカルロ法の精度では不満足という場合、**準モンテカルロ法** (quasi-Monte Carlo methods), **数論的数値積分法** 杉原・室田 [2] が参考になる。以下では、$\{x\}$ で x の小数部分を表す。
   1. low-discrepancy sequences (超一様分布列) の利用
   2. method of good lattice points (「優良格子点法」), E. Hlawka, N. M. Korobov
      $n \le 4$ 程度で使える (?)。$\Omega = [0,1]^n$ のとき
      $$I_N = \frac{1}{N}\sum_{k=0}^{N-1} f\left(\left\{kg_1^{(N)}/N\right\},\cdots,\left\{kg_n^{(N)}/N\right\}\right).$$
      $g_i^{(N)}$ $(i=1,\cdots,n)$ は例えばフィボナッチ数列を用いて決める。
   3. Haselgrove 法 (Haselgrove [3], Sugihara-Murota [4])
      $$I_N = \frac{1}{N}\sum_{k=1}^{N} w_q\left(\frac{k}{N}\right) f\left(\{k\alpha_1\},\cdots,\{k\alpha_n\}\right).$$
      ここで $\alpha_1, \ldots, \alpha_n$ は $\mathbb{Q}$ 上 1 次独立な代数的無理数で
      $$w_q(x) := \frac{(2q+1)!}{q!q!}x^q(1-x)^q.$$

5.3 脱線: Weyl の定理

定理 11.1 (Weyl の定理)

1, $\alpha_1, \cdots, \alpha_s$ が $\mathbb{Q}$ 上 1 次独立ならば、$f \in C([0,1]^s)$ に対して

$$\int_{[0,1]^s} f(x)\,dx = \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} f(\{k\alpha_1\}, \cdots, \{k\alpha_s\}).$$

(この定理の証明をレポートするのはアリかも。Fourier 解析的。)

例えば $s = 1$ ならば、α_1 は無理数ということで、$\alpha_1 = \sqrt{2}$ の場合に適用できて

$$\int_0^1 f(x)\,dx = \lim_{N\to\infty} \sum_{k=1}^{N} f\left(\left\{k\sqrt{2}\right\}\right).$$

$$\{\sqrt{2}\} = 0.41421356\cdots, \quad \{2\sqrt{2}\} = 0.82842712\cdots,$$

5.4 1次元の公式 概観

1次元の定積分

$$I = I(f) = \int_a^b f(x)\,dx$$

に対しては、優秀な方法が色々ある。

1. **補間型数値積分公式**
   被積分関数の**補間多項式**を用いる。
   1. 等間隔標本点 (Newton-Cotes の公式 (Newton-Cotes rules))
      (以下では、定番の中点公式, 台形公式, Simpson 公式を紹介する。)
   2. 区間の端点近くで密な標本点
      特に直交多項式の零点を標本点とする **Gauss 型数値積分公式** ▸ おまけへの Link
2. **変数変換型数値積分公式**
   IMT 公式, 今回取り上げる **DE 公式** (double exponential formula)

(1) $$I_N = I_N(f) := \int_a^b f_N(x)\,dx.$$

ここで f_N は f の**補間多項式**、すなわち $\{x_j\}_{j=1}^N$ を標本点として

(2) $$f_N(x) \in \mathbb{R}[x], \quad \deg f_N(x) \le N-1, \quad f_N(x_j) = f(x_j) \quad (j = 1, \cdots, N).$$

φ_j $(j = 1, \cdots, N)$ を

$$\varphi_j(x) \in \mathbb{R}[x], \quad \deg \varphi_j(x) \le N-1, \quad \varphi_j(x_k) = \delta_{jk} \quad (j, k = 1, \cdots, N)$$

を満たす関数とするとき (これは一意的に存在する)、

$$f_N(x) = \sum_{j=1}^N f(x_j)\varphi_j(x)$$

と書き換えられる。ゆえに

(3) $$I_N = \sum_{j=1}^N f(x_j) w_j, \quad w_j := \int_a^b \varphi_j(x)\,dx.$$

w_j は、$\{x_j\}$ で定まり、f にはよらないので、事前に求めておくことができる。

5.5.2 等間隔標本点による補間型数値積分公式

$N=1$ の場合は**中点公式** (midpoint rule)

$$I_1 = I_1(f) = f\left(\frac{a+b}{2}\right)h, \quad h := b-a.$$

f を定数関数で近似している。f が 1 次多項式ならば $I(f) = I_1(f)$.

$N=2$ の場合は**台形公式** (trapezoidal rule)

$$I_2 = I_2(f) = \frac{h}{2}\left(f(a)+f(b)\right), \quad h := b-a.$$

f を 1 次関数で近似している。f が 1 次多項式ならば $I(f) = I_2(f)$.

$N=3$ の場合は **Simpson 公式** (Simpson's rule)

$$I_3 = I_3(f) = \frac{h}{3}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right), \quad h := \frac{b-a}{2}.$$

f を 2 次関数で近似している。f が 3 次多項式ならば $I(f) = I_3(f)$.

$N=4$ の場合は **Simpson$\frac{3}{8}$ 公式** (Simpson's 3/8 rule — 多分だれも使わない)

$$I_4 = I_4(f) = \frac{3h}{8}\left(f(a)+3f\left(\frac{2a+b}{3}\right)+3f\left(\frac{a+2b}{3}\right)+f(b)\right), \quad h := \frac{b-a}{3}.$$

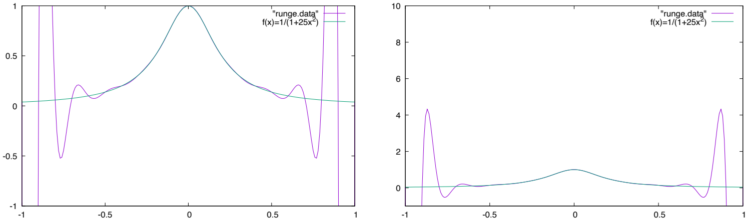
f を 3 次関数で近似している。f が 3 次多項式ならば $I(f) = I_4(f)$.

N を大きくすると良い公式になる？**NO!**

Runge の現象

$$I=\int_{-1}^{1} f(x)\,dx, \quad f(x)=\frac{1}{1+25x^2}.$$

$[-1,1]$ を 20 等分した点を標本点にした場合 $(N=21)$、補間多項式 f_{21} のグラフを見ると、区間の端点近くで暴れている。



f_{21} は f とは似ても似つかない。N を大きくするともっとひどくなる。

大きく分けて2つの対処法がある。

(a) 区間を小区間に分割し、小区間で次数が低い多項式を用いる。
(区間全体で1つの補間多項式を使うことはあきらめる)。
この考え方はあちこちで顔を出す。
- **スプライン近似**
- 有限要素法の**有限要素空間** (近似解、試験関数の属する空間)
- 数値積分の**複合数値積分公式**

(b) **直交多項式**の根を標本点とする補間多項式を利用する。
(直交多項式の根は、区間の端点の近くに密集している。)
- **Gauss 型数値積分公式**
  n 次の公式で、$2n-1$ 次の多項式の積分を誤差なく計算できる。

5.7 複合数値積分公式

- 複合中点公式 ($[a,b]$ を N 等分して、各小区間で中点公式を用いる。)

$$\text{(4)} \qquad M_N := h\sum_{j=1}^{N} f\left(a+\frac{j-1}{2}h\right), \quad h := \frac{b-a}{N}.$$

- 複合台形公式 ($[a,b]$ を N 等分して、各小区間で台形公式を用いる。)

$$\text{(5)} \qquad T_N := h\left(\frac{1}{2}f(a)+\sum_{j=1}^{N-1} f(a+jh)+\frac{1}{2}f(b)\right), \quad h := \frac{b-a}{N}.$$

- 複合 Simpson 公式 ($[a,b]$ を m 等分して、$[a_j,b_j]$ で Simpson 公式を使う。)

$$\text{(6)} \quad S_{2m} = \frac{h}{3}\left(f(a)+2\sum_{j=1}^{m-1} f(a+2jh)+4\sum_{j=1}^{m} f(a+(2j-1)h)+f(b)\right),$$

$$h := \frac{b-a}{2m}.$$

普通は「複合」を省略して、それぞれ単に「中点公式」、「台形公式」、「Simpson 公式」と呼ぶ。「公式」のところを「則」という人もいる。

5.8 数値実験例

C 言語によるサンプル・プログラムを用意した。次のようにしてネット経由でダウンロードしてコンパイル＆実行してみよう。

Mac のターミナルで以下の 4 つのコマンドを実行

```
curl -O http://nalab.mind.meiji.ac.jp/~mk/complex2/prog20220628.tar.gz
tar xzf prog20220628.tar.gz
cd prog20220628
make
```

1. `curl -O` *URL* は、ファイルをインターネット経由で入手するコマンド。
2. `.tar.gz` は、`tar` で複数のファイルを 1 つのアーカイブ・ファイルにまとめ、`gzip` で圧縮したファイルの拡張子である。復元するには `tar xzf` ファイル名 とする。
3. `prog20220628` というディレクトリィが現れるので、そこに cd (change directory) する。
4. `make` する (`Makefile` に記述した指示に従って、コンパイルなどの作業を行う)。今回は、いくつかの C のプログラムをコンパイルして、実行する。

→ 画面に色々なグラフが出て来るはず。

5.8 数値実験例　数値積分公式のコード

今年度は試験的に Python で書いたサンプル・プログラム (Jupyter Notebook 用) を用意しました。

http://nalab.mind.meiji.ac.jp/~mk/complex2/20220628ni.ipynb

```c
/* nc.c --- Newton-Cotes の積分公式: 中点公式, 台形公式, Simpson 公式 */
typedef double ddfunction(double);

/* 関数 f の [a,b] における積分の複合中点公式による数値積分 M_N */
double midpoint(ddfunction f, double a, double b, int N)
{
  int j;
  double h = (b - a) / N, M = 0.0;
  for (j = 1; j <= N; j++) M += f(a + (j - 0.5) * h);
  M *= h;
  return M;
}

/* 関数 f の [a,b] における積分の複合台形公式による数値積分 T_N */
double trapezoidal(ddfunction f, double a, double b, int N)
{
  int j;
  double h = (b - a) / N, T = (f(a) + f(b)) / 2;
  for (j = 1; j < N; j++) T += f(a + j * h);
  T *= h;
  return T;
}

/* 関数 f の [a,b] における積分の複合 Simpson 公式による数値積分 S_{N} */
double simpson(ddfunction f, double a, double b, int N)
{
  int m = N / 2;
  return (trapezoidal(f, a, b, m) + 2 * midpoint(f, a, b, m)) / 3;
}
```

5.8 数値実験例　滑らかな関数の場合

$$I = \int_0^1 e^x \, dx$$ を中点公式, 台形公式, Simpson 公式で計算すると

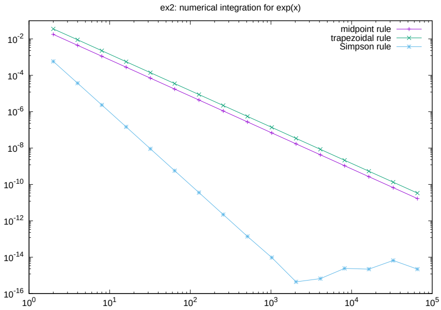


図 1: 横軸は $N = 2, 4, \cdots, 2^{16} = 65536$; 縦軸は誤差 (両軸とも対数目盛)

誤差は、中点公式, 台形公式では $O\left(\frac{1}{N^2}\right)$, Simpson 公式では $O\left(\frac{1}{N^4}\right)$).

用語紹介 (m 位の公式) と余談

数値積分公式が m 位の公式 (m 次の精度) であるとは、関数 f の数値積分公式の誤差を $E(f)$ と書くとき

$$E(x^k) = 0 \quad (k = 0, 1, \cdots, m), \quad E(x^{m+1}) \neq 0$$

が成り立つことをいう。(m 次多項式の積分が誤差なく計算できる。)

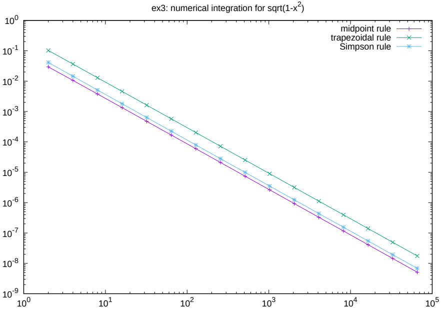
滑らかな関数の定積分を m 位の数値積分公式で計算したときの誤差は、$O\left(\frac{1}{N^{m+1}}\right)$ となる。

実は、中点公式 ($N = 1$) と台形公式 ($N = 2$) は、ともに 1 位の公式で、Simpson 公式 ($N = 3$) は 3 位の公式である。

(一般化できて) N 個の標本点を使う $N-1$ 次の補間多項式で作った数値積分公式は、少なくとも $N-1$ 位であるが、実は N が奇数のときは N 位の公式である (1 位分お得)。

5.8 数値実験例　滑らかでない関数の場合

$I = \int_0^1 \sqrt{1-x^2}\,dx$ を中点公式, 台形公式, Simpson 公式で計算すると



連続ではあるが、滑らかではない関数 (実際 $x=1$ で微分可能でない) に対しては、誤差は N の増加とともに減少する (0 に近く) が、滑らかな被積分関数の場合と比べると遅い。

また、中点公式, 台形公式, Simpson 公式で差が出ない。高次の公式の優位性はない (こういう結果は割と一般的に見られる)。

5.8 数値実験例　滑らかな関数 vs 滑らかでない関数

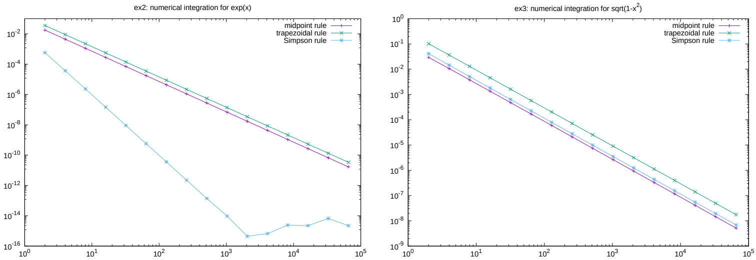
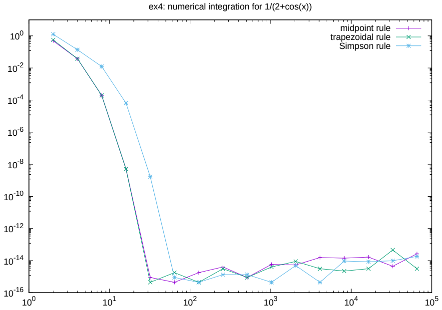


図 2: $\int_0^1 e^x \, dx$ と $\int_0^1 \sqrt{1-x^2} \, dx$ の数値積分の結果

5.9 不思議な好結果 (1) 滑らかな周期関数の周期積分

$I = \displaystyle\int_0^{2\pi} \frac{dx}{2+\cos x} \quad \left(= \frac{2\pi}{\sqrt{3}}\right)$ を中点公式, 台形公式, Simpson 公式で計算すると



$N = 32$ で中点公式、台形公式、Simpson 公式の誤差は、それぞれ 4.44×10^{-16}, -1.78×10^{-16}, 1.70×10^{-9} (中点 ≒ 台形 ≪Simpson).

C 言語処理系の double 型は 10 進法に換算して 16 桁弱の精度なので、(わずか) $N = 32$ の台形公式ではあるが、満足すべき結果が得られたと言える。

台形公式の定義式は

$$T_N = h\left(\frac{1}{2}f(a) + \sum_{j=1}^{N-1} f(a+jh) + \frac{1}{2}f(b)\right), \quad h := b-a$$

であったが、周期関数の周期積分の場合は $f(a) = f(b)$ が成り立つので

$$T_N = h\sum_{j=1}^{N} f(a+jh) = h\sum_{j=0}^{N-1} f(a+jh) \quad \text{(長方形公式？)}.$$

一方、中点公式は

$$M_N = h\sum_{j=1}^{N} f(a+(j-1/2)h)$$

であった。周期関数の周期積分については、どちらも N 等分点での被積分関数の値の和に h をかけたものである ($h/2$ ずれているけれど)。その意味では台形公式と中点公式には、本質的な違いはないことが分かる。

$I = \displaystyle\int_{-\infty}^{\infty} f(x)\,dx$ に対して、$h > 0$, $N \in \mathbb{N}$ を適当にえらんで

$$I_h := h \sum_{n=-\infty}^{\infty} f(nh), \quad I_{h,N} := h \sum_{n=-N}^{N} f(nh)$$

とおく。I_h, $I_{h,N}$ も**台形公式**と呼ばれる。

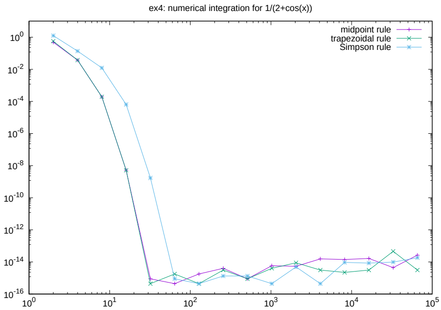
ガウス積分 $I = \displaystyle\int_{-\infty}^{\infty} e^{-x^2}\,dx \ (= \sqrt{\pi}\,)$ について

N	h	$I - I_{h,N}$
6	1	-1.833539×10^{-4}
12	0.5	$-2.220446 \times 10^{-16}$
24	0.25	$-4.440892 \times 10^{-16}$

(おおざっぱに言って $x \in [-6, 6]$ の範囲を計算したことになっている。$x = \pm 6$ のとき $e^{-x^2} \fallingdotseq 2.3 \times 10^{-16}$ で、すでに十分小さい。)

$N = 12$ (25 個の標本点) で、満足行く精度の近似値 (ほぼ処理系の浮動小数点数の精度一杯) が得られている。

$$I = \int_0^{2\pi} \frac{dx}{2+\cos x} \quad \left(= \frac{2\pi}{\sqrt{3}}\right)$$ を中点公式, 台形公式, Simpson 公式で計算すると



$N=32$ で中点公式、台形公式、Simpson 公式の誤差は、それぞれ 4.44×10^{-16}, -1.78×10^{-16}, 1.70×10^{-9} (中点 $\fallingdotseq$ 台形 $\ll$Simpson).

C 言語処理系の double 型は 10 進法に換算して 16 桁弱の精度なので、(わずか) $N=32$ の台形公式で満足すべき結果と言える。

$I = \int_{-\infty}^{\infty} f(x)\,dx$ に対して、$h > 0$, $N \in \mathbb{N}$ を適当にえらんで

$$I_h := h \sum_{n=-\infty}^{\infty} f(nh), \quad I_{h,N} := h \sum_{n=-N}^{N} f(nh)$$

とおく。I_h と $I_{h,N}$ も**台形公式**と呼ばれる。

ガウス積分 $I = \int_{-\infty}^{\infty} e^{-x^2}\,dx \ (= \sqrt{\pi}\,)$ について

N	h	$I - I_{h,N}$
6	1	-1.833539×10^{-4}
12	0.5	$-2.220446 \times 10^{-16}$
24	0.25	$-4.440892 \times 10^{-16}$

(おおざっぱに言って $x \in [-6, 6]$ の範囲を計算したことになっている。$x = \pm 6$ のとき $e^{-x^2} \fallingdotseq 2.3 \times 10^{-16}$ で、すでに十分小さい。)

$N = 12$ (25 個の標本点) で、満足行く精度の近似値 (ほぼ処理系の浮動小数点数の精度一杯) が得られている。

与えられた定積分 $\int_a^b f(x)\,dx$ を、上で紹介した

- (A) 滑らかな周期関数の周期にわたる積分 $\int_0^T f(x)\,dx$

  (台形公式 $T_N = h\sum_{j=0}^{N-1} f(jh)$, $h := T/N$ を適用)

- (B) $x \to \pm\infty$ のときの減衰の速い解析的な関数 f の $\mathbb{R}$ 上の積分 $\int_{-\infty}^{\infty} f(x)\,dx$

  (台形公式 $I_{h,N} := h\sum_{n=-N}^{N} f(nh)$ を適用)

などの非常にうまく行く場合の形に変数変換で直して、それから数値積分する、というアイディアが提出された (1970 年頃)。

IMT 公式 (伊理・森口・高澤 1970 年) は、(A) に帰着させるものである。

DE 公式 (double exponential formula, 二重指数関数型数値積分公式, 高橋・森 1974 年) は、(B) に帰着させるものである。

5.10.2 二重指数関数型数値積分公式の紹介

高橋・森は以下のような数値積分公式を提案した。解析関数 f の定積分

$$I = \int_{-1}^{1} f(x)\,dx$$

に対して

(7) $$x = \varphi_1(t) := \tanh\left(\frac{\pi}{2}\sinh t\right) \quad (t \in \mathbb{R})$$

により変数変換 (置換積分) をする。

$$I = \int_{-\infty}^{\infty} f\left(\varphi_1(t)\right)\varphi_1'(t)\,dt.$$

この I に対して、前回紹介したような台形公式を適用する。

$$I_{h,N} = h\sum_{n=-N}^{N} f\left(\varphi_1(nh)\right)\varphi_1'(nh) \quad (h > 0,\ N \in \mathbb{N}).$$

この公式を**二重指数関数型数値積分公式** (double exponential formula) と呼ぶ。以下では、**DE 公式**と呼ぶことにする。

5.10.2 二重指数関数型数値積分公式の紹介 (続き)

$t \to \infty$ のとき $\varphi_1(t) \to 1$, $t \to -\infty$ のとき $\varphi_1(t) \to -1$ であるが、ともに非常に速く収束する。

また $t \to \pm\infty$ のとき $\varphi_1' \to 0$ も非常に速い。

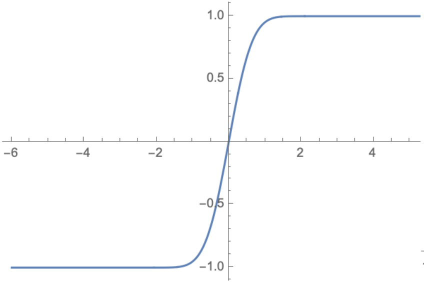


図 3: φ_1 は $\mathbb{R}$ を $(-1,1)$ に写す

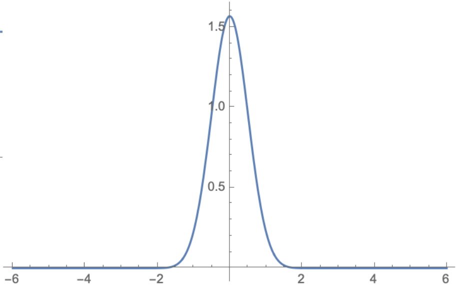


図 4: φ_1' は $t \to \pm\infty$ のとき急速に減衰

5.10.2 二重指数関数型数値積分公式の紹介 (続き)

　高橋・森は、φ_1 が (ある定数 C に対して)

$$\text{(8)}\qquad \left|f\left(\varphi_1(t)\right)\varphi_1'(t)\right| \sim \exp\left(-C\exp|t|\right) \quad (t\to\pm\infty)$$

を満たすように選んだ。**高橋・森の誤差解析手法**に基づき、ある種の最適性があると判断したためという。これが「二重指数関数型数値積分公式」という名前の由来である。

　上では積分区間が $[-1,1]$ の場合を説明したが、一般の有界閉区間上の積分 $\displaystyle\int_a^b f(u)\,du$ の場合は、変数変換 $u=a+\frac{b-a}{2}(x+1)$ $(x\in[-1,1])$ を利用すれば良い。

　非有界区間の定積分 $I=\displaystyle\int_{-\infty}^{\infty} f(x)\ dx$ (f の減衰が遅い場合), $I=\displaystyle\int_0^{\infty} f(x)\ dx$ などについても、(8) が成り立つような具体的な変数変換 $x=\varphi(t)$ $(t\in\mathbb{R})$ が提案されている (桂田 [1] を見よ)。

5.10.3 サンプル・プログラムの入手・実行

サンプル・プログラムを入手した人は、既に DE 公式のサンプル・プログラムを持っている。

再録: Mac のターミナルで以下の 4 つのコマンドを実行

```
curl -O http://nalab.mind.meiji.ac.jp/~mk/complex2/prog20220628.tar.gz
tar xzf prog20220628.tar.gz
cd prog20220628
make
```

以上をしてあれば (同じディレクトリィで)

```
./example6
./example6kai
```

とすれば良い。

5.10.4 DE 公式の数値例 (1) $\int_{-1}^{1}\sqrt{1-x^2}\,dx$ に再挑戦

中点公式、台形公式、Simpson 公式でうまく計算できなかった (その結果は `open ex3.png` で見られる)

$$I=\int_{-1}^{1}\sqrt{1-x^2}\,dx=\pi/2=1.5707963267948966192\cdots$$

を DE 公式で計算すると

```
% ./example6
DE 公式による数値積分
test1 (sqrt(1-x^2) の積分)
h=1.000000, N=   3, I_hN=       1.7125198292703636, I_hN-I=1.417235e-01
h=0.500000, N=   6, I_hN=       1.5709101233831166, I_hN-I=1.137966e-04
h=0.250000, N=  12, I_hN=       1.5707963267997540, I_hN-I=4.857448e-12
h=0.125000, N=  24, I_hN=       1.5707963267948970, I_hN-I=4.440892e-16

(後略)
```

$N=24$ (49 項) のときに、使用している浮動小数点数 (10 進 16 桁弱) の精度の近似値が得られた (実は $N=18$ で $I-I_{h,N}\fallingdotseq 2.22\times 10^{-16}$)。

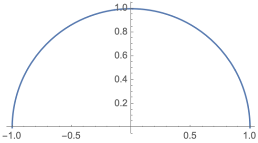
5.10.4 DE 公式の数値例 (1) $\int_{-1}^{1}\sqrt{1-x^2}\,dx$ に再挑戦

```
In[1]:= f[x_] := Sqrt[1 - x^2]
                 平方根

In[2]:= phi1[t_] := Tanh[Pi / 2 Sinh[t]]
                    双…   円周率 双曲線正弦

In[4]:= g1 = Plot[f[x], {x, -1, 1}, AspectRatio → Automatic]
             プロット                  縦横比        自動
```

Out[4]=




```
In[6]:= g2 = Plot[f[phi1[t]] × phi1'[t], {t, -3, 3}, AspectRatio → Automatic]
             プロット                                   縦横比        自動
```

Out[6]=

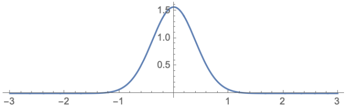


図 5: $f(x)=\sqrt{1-x^2}$ と変数変換後 $f(\varphi_1(t))\varphi_1'(t)$ のグラフ

直観的に分かる？

5.10.5 DE 公式の数値例 (2) $\displaystyle\int_{-1}^{1}\frac{dx}{\sqrt{1-x^2}}$

驚くべきことに広義積分 $\displaystyle\int_{-1}^{1}\frac{dx}{\sqrt{1-x^2}}$ $(=\pi)$ (端点で分母が 0) を計算できる。

example6 の test 2 では、$N=6$, $h=1/2$ のとき、誤差が 1.8×10^{-8} という結果が得られる。N を大きくしてもそれ以上精度が改善されないが、それはいわゆる**桁落ち現象**による ($x\fallingdotseq\pm 1$ のとき $1-x^2$ の有効桁がたくさん失われる)。

桁落ちが起こらない工夫をした example6kai の計算結果は次のようになる。

```
% ./example6kai

(中略)

test2 (1/sqrt(1-x^2) の (-1,1) での積分)
h=1.000000, N=   4, I_hN=       3.1435079789309328, I_hN-I=1.915325e-03
h=0.500000, N=   8, I_hN=       3.1415926733057051, I_hN-I=1.971591e-08
h=0.250000, N=  16, I_hN=       3.1415926535897940, I_hN-I=8.881784e-16
```

$N=16$ (33 項) で、誤差が 10^{-15} を下回っている。満足すべき結果である。

5.10.5 DE 公式の数値例 (2) $\int_{-1}^{1} \frac{dx}{\sqrt{1-x^2}}$

```
In[9]:= f2[x_] := 1 / Sqrt[1 - x ^ 2]
```

平方根

```
In[10]:= g3 = Plot[f2[phi1[t]] × phi1'[t], {t, -3, 3}, AspectRatio → Automatic]
```

プロット　縦横比　自動

Out[10]=

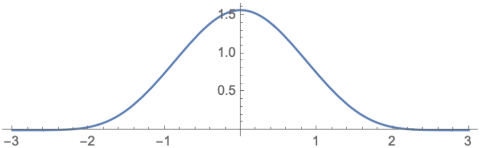


図 6: $f(x) = \dfrac{1}{\sqrt{1-x^2}}$ の場合の変数変換後の $f(\varphi_1(t))\varphi_1'(t)$ のグラフ

ゆっくり考えてみることを勧める

§5 .9「不思議な好結果」は、特殊なケースに限られて、実際の役には立たないように思えたかもしれない。しかし、色々な定積分が、二重指数関数型変数変換によって、「不思議な好結果」をもたらす定積分に変換できることが分かった。

こうなると

なぜ (不思議なくらい) 好結果をもたらすか？

知りたくなる。

この講義に残された時間は少ないが、高橋・森の誤差解析手法の門をくぐってみよう。

おまけ: Gauss 型数値積分公式

この型の数値積分公式を考えるときは、考える定積分を

$$I = \int_a^b f(x)w(x)\,dx$$

の形のものにするのが普通である。w は重み関数と呼ばれる非負関数である。

$1, x, x^2, \cdots$ を内積

$$\langle \varphi, \psi \rangle := \int_a^b \varphi(x)\psi(x)w(x)\,dx$$

について直交化して直交関数系 $\{P_n(x)\}_{n\geq 0}$ を作る。

n 次多項式 $P_n(x)$ の零点 $\{x_j\}_{j=1}^n$ を標本点に採用すると、非常に高性能 ($2n-1$ 次の多項式に対して真の値) の公式が得られる。

参考文献 I

私はモンテカルロ法については、何も知らないので、参考書の紹介は差し控える。

数値積分について詳しいテキストとしては、杉原・室田 [2], 森 [5] をお勧めする。(以下は関数論としては脱線かも) Haselgrove の方法は、杉原・室田両氏の論文 [4] で、とても使いやすく改良された。優良格子点法については、Niederreiter [6] も見るとよい。

[1] 桂田祐史：数値積分解説, `http://nalab.mind.meiji.ac.jp/~mk/complex2/numerical-integration-2017.pdf` (2017).

[2] 杉原 正顯（まさあき）, 室田（むろた）一雄：数値計算法の数理, 岩波書店 (1994).

[3] Haselgrove, C. B.: A Method for Numerical Integration, *Mathematics of Computation*, Vol. 15, No. 76, pp. 323–337 (1961).

[4] Sugihara, M. and Murota, K.: A Note on Haselgrove's Method for Numerical Integration, *Mathematics of Computation*, Vol. 39, No. 160, pp. 549–554 (1982).

[5] 森 正武（まさたけ）：数値解析 第 2 版, 共立出版 (2002/2/25), 第 1 版は 1973 年に出版された。

参考文献 II

[6] Niederreiter, H.: 数論の応用, 数学セミナー, 1984 年 11 月号, pp. 90–96 (1984).