数値計算法 第13回

数値積分 (1)

濱本 雄治 [1]

情報工学部 情報通信工学科

2025 年 7 月 14 日

[1] hamamoto@c.oka-pu.ac.jp

復習: Newton 補間

- 1 点 (x_0, y_0) を通る 0 次 Newton 補間多項式

$$p_0(x) = m_0 \qquad (m_0 = f[x_0] \equiv y_0)$$

- 2 点 $(x_0, y_0), (x_1, y_1)$ を通る 1 次 Newton 補間多項式

$$p_1(x) = m_0 + m_1(x - x_0) \qquad \left(m_1 = f[x_0, x_1] \equiv \frac{f[x_0] - f[x_1]}{x_0 - x_1}\right)$$

- 3 点 $(x_0, y_0), (x_1, y_1), (x_2, y_2)$ を通る 2 次 Newton 補間多項式

$$p_2(x) = m_0 + m_1(x - x_0) + m_2(x - x_0)(x - x_1)$$
$$\left(m_2 = f[x_0, x_1, x_2] \equiv \frac{f[x_0, x_1] - f[x_1, x_2]}{x_0 - x_1}\right)$$

前回の小テストの解説

データ点 $(0,0),(1,1),(2,3)$ から差分表を作り、2 次の Newton 補間多項式 $p_2(x)$ を求めよ。

$$
\begin{array}{ccccccc}
\boxed{0} & \to & \boxed{0} & \searrow & & & \\
 & & & & \boxed{1-0=1} & \searrow & \\
\boxed{1} & \to & \boxed{1} & \nearrow\searrow & & & \boxed{\dfrac{2-1}{2-0}=\dfrac{1}{2}} \\
 & & & & \boxed{3-1=2} & \nearrow & \\
\boxed{2} & \to & \boxed{3} & \nearrow & & &
\end{array}
$$

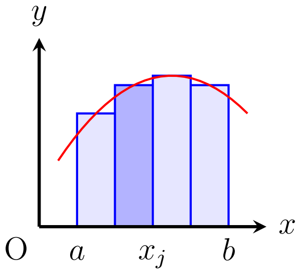
$$
\begin{aligned}
p_2(x) &= m_0 + m_1(x-x_0) + m_2(x-x_0)(x-x_1) \\
&= 0 + 1\cdot(x-0) + \frac{1}{2}(x-0)(x-1) = \frac{1}{2}x^2 + \frac{1}{2}x
\end{aligned}
$$

補間多項式の一意性から、Lagrange 補間と一致することに注意

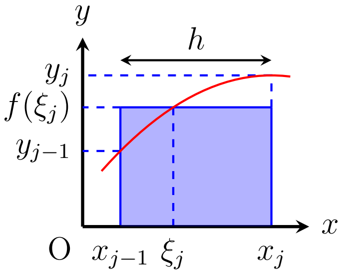
定積分

- 積分区間 $[a,b]$ を n 等分して $h=\dfrac{b-a}{n}$ とすると

$$x_j = x_0 + jh \quad (j=0,\cdots,n), \qquad a = x_0 < \cdots < x_n = b$$



$\xrightarrow{\text{拡大}}$



- j 番目の長方形の高さを $f(\xi_j)$ $(x_{j-1} \le \xi_j \le x_j)$ とすると

$$\int_a^b f(x)\,\mathrm{d}x = \lim_{n\to\infty} h \sum_{j=1}^{n} f(\xi_j), \qquad \left(f(\xi_j) = \frac{\alpha y_{j-1} + \beta y_j}{\alpha+\beta} \right)$$

数値積分

- $n \to \infty$ の代わりに、十分大きな $n \gg 1$ に対して

$$\int_a^b f(x)\,\mathrm{d}x \simeq h \sum_{j=1}^{n} f(\xi_j), \qquad \left(f(\xi_i) = \frac{\alpha y_{j-1} + \beta y_j}{\alpha + \beta} \right)$$

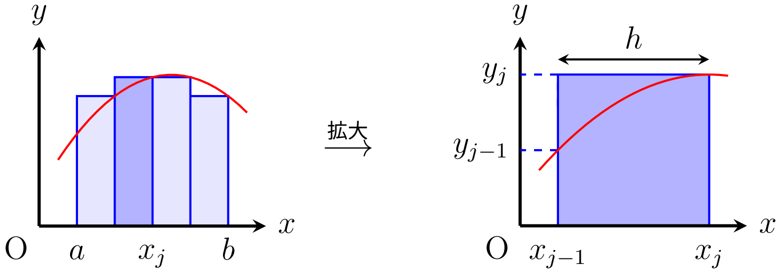
- 数値積分法の種類

手法	α	β	長方形の高さ
区分求積法	0	1	$f(\xi_j) = y_j$
台形公式	1	1	$f(\xi_i) = \dfrac{y_{j-1} + y_j}{2}$
Simpson の公式	1	2	$f(\xi_{j-1}) = \dfrac{y_{j-2} + 2y_{j-1}}{3}$
	2	1	$f(\xi_j) = \dfrac{2y_{j-1} + y_j}{3}$

区分求積法

- $f(\xi_j) = y_j$ として

$$\int_a^b f(x)\,\mathrm{d}x \simeq h \sum_{j=1}^{n} y_j$$

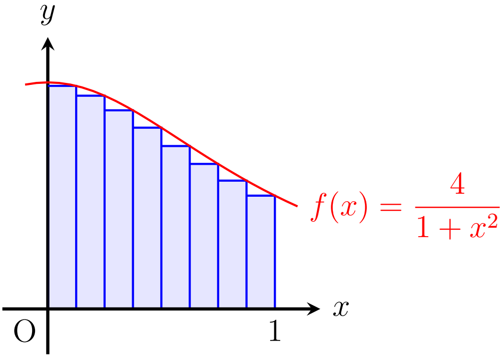


区分求積法の例

- 定積分

$$S = \int_0^1 \frac{4}{1+x^2}\mathrm{d}x \quad \left(= \int_0^{\pi/4} \frac{4}{1+\tan^2\theta}\frac{\mathrm{d}\theta}{\cos^2\theta} = \pi \right)$$

- 8 分割のとき



区分求積法の計算例

- division.c

```
#include <stdio.h>
int main(){
  double xmin = 0.0;
  double xmax = 1.0;
  for(int num=8; num<=1024; num*=2){
    double h = (xmax - xmin) / num;
    double sum = 0.0;

    for(int i=1; i<=num; ++i){
      double x = xmin + i * h;
      sum += 4.0 / (1.0 + x * x);
    }
    printf("%d %f\n", num, sum*h);
  }
}
```

- 実行結果

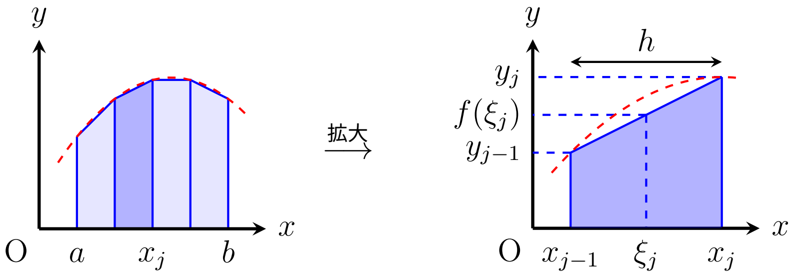
```
$ gcc division.c -lm
$ ./a.out
8 3.013988
16 3.078442
32 3.110180
64 3.125927
128 3.133770
256 3.137684
512 3.139639
1024 3.140616
```

収束が遅い

台形公式

- $f(\xi_j) = \dfrac{y_{j-1} + y_j}{2}$ として

$$\int_a^b f(x)\,\mathrm{d}x \simeq h \sum_{j=1}^{n} \frac{y_{j-1} + y_j}{2} = h \left(\frac{y_0}{2} + \sum_{j=1}^{n-1} y_j + \frac{y_n}{2} \right)$$

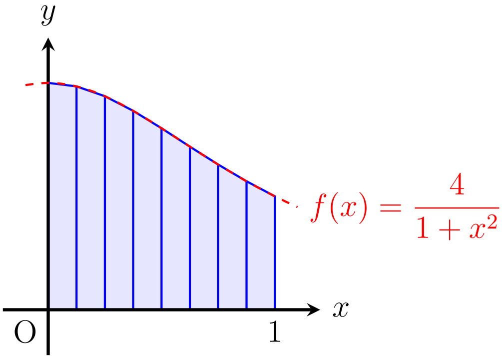


台形公式の例

- 定積分

$$S = \int_0^1 \frac{4}{1+x^2}\mathrm{d}x$$

- 8 分割のとき



台形公式の計算例

- `trapez.c`

```
#include <stdio.h>
double func(double x){
  return 4.0 / (1.0 + x * x);
}
int main(){
  double xmin = 0.0; double xmax = 1.0;
  for(int num=8; num<=1024; num*=2){
    double h = (xmax - xmin) / num;
    double sum = func(xmin) + func(xmax);
    for(int i=1; i!=num; ++i){
      double x = xmin + i * h;
      sum += 2 * func(x);
    }
    printf("%d %f\n", num, 0.5 * sum * h);
  }
}
```

- 実行結果

```
$ gcc trapez.c -lm
$ ../a.out
8 3.138988
16 3.140942
32 3.141430
64 3.141552
128 3.141582
256 3.141590
512 3.141592
1024 3.141592
```

Simpson の公式

- 台形公式では 2 点 $(x_{j-1}, y_{j-1}), (x_j, y_j)$ を Lagrange 補間

$$f_1(x) = y_{j-1}\frac{x - x_j}{x_{j-1} - x_j} + y_j\frac{x - x_{j-1}}{x_j - x_{j-1}} = \frac{y_j - y_{j-1}}{h}(x - x_{j-1}) + y_{j-1}$$

$$\int_{x_{j-1}}^{x_j} f_1(x)\mathrm{d}x = \frac{h}{2}(y_{j-1} + y_j)$$

- 3 点 $(x_{j-1}, y_{j-1}), (x_j, y_j), (x_{j+1}, y_{j+1})$ を Lagrange 補間すると

$$\begin{aligned}
f_2(x) &= y_{j-1}\frac{(x - x_j)(x - x_{j+1})}{(x_{j-1} - x_j)(x_{j-1} - x_{j+1})} + y_j\frac{(x - x_{j-1})(x - x_{j+1})}{(x_j - x_{j-1})(x_j - x_{j+1})} \\
&\quad + y_{j+1}\frac{(x - x_{j-1})(x - x_j)}{(x_{j+1} - x_{j-1})(x_{j+1} - x_j)} \\
&= \frac{1}{2h^2}[y_{j-1}(x - x_j)(x - x_j - h) - 2y_j(x - x_j + h)(x - x_j - h) \\
&\quad + y_{j+1}(x - x_j + h)(x - x_j)]
\end{aligned}$$

- $u = x - x_j$ とおいて区間 $[x_{j-1}, x_{j+1}]$ で $f_2(x)$ を積分すると

$$
\begin{aligned}
&\int_{x_{j-1}}^{x_{j+1}} f_2(x)\mathrm{d}x \\
&= \frac{1}{2h^2}\int_{-h}^{h} [y_{j-1}u(u-h) - 2y_j(u^2-h^2) + y_{j+1}u(u+h)]\mathrm{d}u \\
&= \frac{1}{h^2}\int_{0}^{h} [y_{j-1}u^2 - 2y_j(u^2-h^2) + y_{j+1}u^2]\mathrm{d}u \\
&= \frac{1}{h^2}\left\{(y_{j-1} - 2y_j + y_{j+1})\left[\frac{u^3}{3}\right]_0^h + 2y_jh^2\Big[u\Big]_0^h\right\} \\
&= h\frac{y_{j-1} + 4y_j + y_{j+1}}{3} \\
&= h\frac{y_{j-1} + 2y_j}{3} + h\frac{2y_j + y_{j+1}}{3} \qquad \text{(Simpson の公式)}
\end{aligned}
$$

- 区間 $[x_{j-1}, x_j], [x_j, x_{j+1}]$ をまとめて扱うため、分割数 n は偶数

合成 Simpson 公式

- 区間 $[a, b]$ での定積分は

$$
\begin{aligned}
\int_a^b f(x)\,\mathrm{d}x &\simeq h\left(\frac{y_0 + 2y_1}{3} + \frac{2y_1 + y_2}{3} + \frac{y_2 + 2y_3}{3} + \frac{2y_3 + y_4}{3}\right. \\
&\qquad \left. + \cdots + \frac{y_{n-2} + 2y_{n-1}}{3} + \frac{2y_{n-1} + y_n}{3}\right) \\
&= h\left[\frac{y_0}{3} + \frac{4}{3}(y_1 + y_3 + \cdots + y_{n-1})\right. \\
&\qquad \left. + \frac{2}{3}(y_2 + y_4 + \cdots + y_{n-2}) + \frac{y_n}{3}\right]
\end{aligned}
$$

Simpsonの公式の計算例

- simpson.c

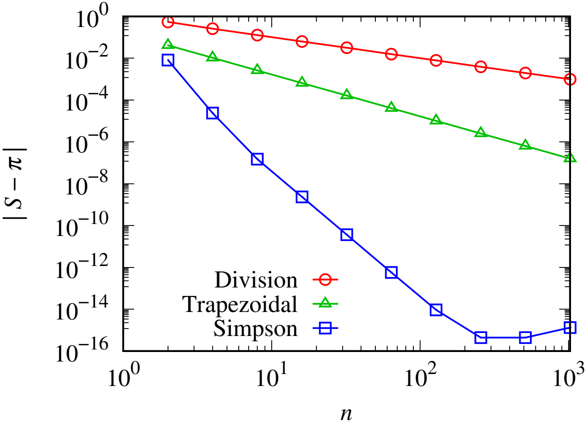
```c
#include <stdio.h>
#include <math.h>
double func(double x){
  return 4.0 / (1.0 + x * x);
}
int main(){
  double xmin = 0.0; double xmax = 1.0;
  for(int num=2; num<=1024; num*=2){
    double h = (xmax - xmin) / num;
    double sum = func(xmin) + func(xmax);
    for(int j=1; j!=num; ++j){
      double x = xmin + j * h;
      sum += (j%2==0? 2: 4) * func(x);
    }
    printf("%d %f\n", num, sum * h / 3);
  }
}
```

- 実行結果

```
$ gcc simpson.c
$ ./a.out
2 3.133333
4 3.141569
8 3.141593
16 3.141593
32 3.141593
64 3.141593
128 3.141593
256 3.141593
512 3.141593
1024 3.141593
```

収束性の比較

- `M_PI=3.14159265358979323846` からの絶対誤差



  - 分割数 n の増加とともに誤差が指数関数的に減少
  - Simpson 公式では $n = 256$ で `double` 型の精度限界 $\sim 10^{-16}$ に到達

Newton-Cotes の公式

- Simpson の公式では 2 個の区間 $[x_{j-1}, x_j], [x_j, x_{j+1}]$ をまとめて 3 点 $(x_{j-1}, y_{j-1}), (x_j, y_j), (x_{j+1}, y_{j+1})$ を Lagrange 補間
- m 個の隣接する区間の $m+1$ 点 $(x_0, y_0), (x_1, y_1), \cdots, (x_m, y_m)$ を Lagurange 補間すると

$$f_m(x) = \sum_{i=0}^{m} y_i \prod_{j=0\,(j\neq i)}^{m} \frac{x - x_j}{x_i - x_j}$$

- 区間 $[x_0, x_m]$ で $f_m(x)$ を積分すると

$$\int_{x_0}^{x_m} f_m(x)\,\mathrm{d}x = \sum_{i=0}^{m} w_i y_i \qquad \text{(Newton-Cotes の公式)}$$

ただし $w_i \equiv \displaystyle\int_{x_0}^{x_m} \prod_{j=0\,(j\neq i)}^{m} \frac{x - x_j}{x_i - x_j}\,\mathrm{d}x$