Report on the Experiment

No. 1

Subject 数値計算処理における誤差の測定

Date 2022. 11. 23

Weather　Temp ℃ Wet ％

Class　S1
Group
Chief
Partner

No　17
Name　小畠 一泰

Kure National College of Technology

1　目的

数学的に解を得ることが困難な微分や積分を直接計算により数値微分や数値積分を行う方法がある．この方法は，未知の関数や微積分が困難か不可能な関数でも解を得ることが可能である．しかし，関数の近似のため誤差を生じることがある．また，誤差を減少させるために正確な近似を行うと多くの計算時間を必要とする場合がある．

本演習は，数値微分や数値積分を計算機上で行い，その誤差の測定を行う．

2　関数 f2 の確認

コード 1 テストプログラム：関数 f2 の確認処理 (test.c)

```c
#include <stdio.h>
#include "func.h"

int main() {
  int a;
  FILE *fp;
  double t;

  fp = fopen("f2.csv", "wt");
  fprintf(fp, "t,f2,f2d\n");
  for (a = 0; a <= 100; a++) {
    t = (double)a / 100.0;
    fprintf(fp, "%lf,%lf,%lf\n", t, f2(t, 0, 17), f2(t, 1, 17));
  }
  fclose(fp);
  return 0;
}
```

関数 f2 は `func.h` に `double f2(double,int,int);` とあり，それぞれの引数は次の通りである．

- 第 1 引数
  - 時間 t($0 \leqq t \leqq 1$)
- 第 2 引数
  - 1: 0 - 関数値
  - 1: 1 - 微分値
  - 1: 2 - 積分値
- 第 3 引数
  - 出席番号

従ってコード 1 では，1 秒間を 101 分割し，`f2(t)` 並びに `f2'(t)` を求め，CSV に時間 t とともに書き込んでいる．この結果を図 1 に示す．また図 1 より，`f2(t)` が最大値，最小値をとる t において `f2'(t)` も概ね 0 を取ることが見て取れることからプログラムは期待通りに動作していると考えられる．

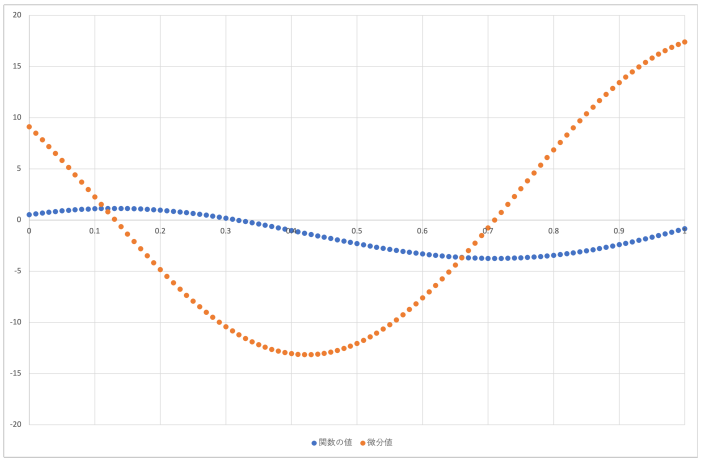


図 1: 関数 f2 の値と微分値

3 数値微分

コード 2 数値微分と微分の比較 (testd.c)

```c
#include <stdio.h>
#include "func.h"
int main() {
  int c, b = 10; // b は区間の分割数
  double x, f21v, f20v, dt2 = 1.0 / (double)b, f[2000], d = 0.0;
  FILE *fp;
  fp = fopen("f2d.csv", "wt");
  for (c = -1; c <= b + 1; c++) {
    x = (double)c / (double)b;
    f[c + 1] = f2(x, 0, 17);
  }
  fprintf(fp, "t,f\'2c,f\'2\n");
  for (c = 0; c <= b; c++) {
    x = (double)c / (double)b;
    f21v = (f[c + 2] - f[c]) / (dt2 * 2.0);
    f20v = f2(x, 1, 17);
    fprintf(fp, "%lf,", x);
    fprintf(fp, "%lf,", f21v);
    fprintf(fp, "%lf\n", f20v);
    d += fabs(f21v - f20v);
  }
  printf("%lf\n", d / (double)b);
  fclose(fp);
}
```

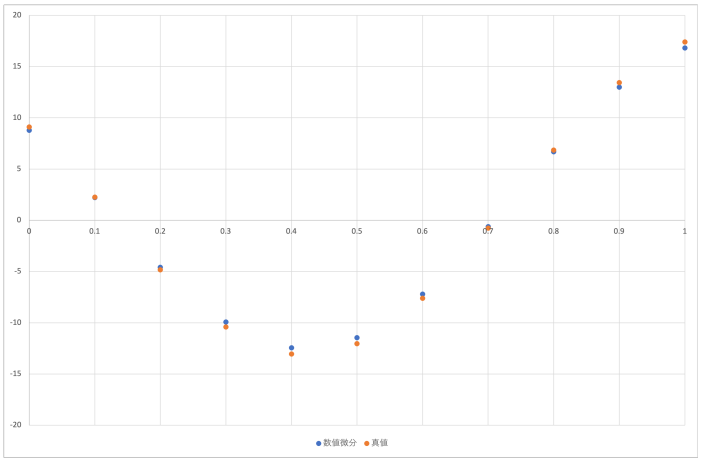


図 2: 数値値分を行なった結果と真値

8 行目から 11 行目で区間の分割数 +2 個分の関数値を配列 f に格納している. その後 13 行目から 21 行目で傾きを求めている.

コード 3 平均誤差の計算 (testd2.c)

```c
#include <math.h>
#include <stdio.h>
#include "func.h"
#define PRECISION 6000
int main() {
  int c, b; // b は区間の分割数
  double x, f21v, f20v, dt2, f[PRECISION + 1000], d;
  FILE *fp;
  fp = fopen("f2d2.csv", "wt");
  for (b = 1; b <= PRECISION; b++) {
    dt2 = 1.0 / (double)b;
    d = 0.0;
    for (c = -1; c <= b + 1; c++) {
      x = (double)c / (double)b;
      f[c + 1] = f2(x, 0, 17);
    }
    for (c = 0; c <= b; c++) {
      x = (double)c / (double)b;
      f21v = (f[c + 2] - f[c]) / (dt2 * 2.0);
      f20v = f2(x, 1, 17);
      d += fabs(f21v - f20v);
    }
    fprintf(fp, "%d,%.20lf\n", b, d / (double)b);
  }
  fclose(fp);
}
```

図 3 より関数 f2 の精度と分割数の関係から, 精度を 10^{-6} 程度にする場合は分割数を 6000 程度にする必要があることがわかる.

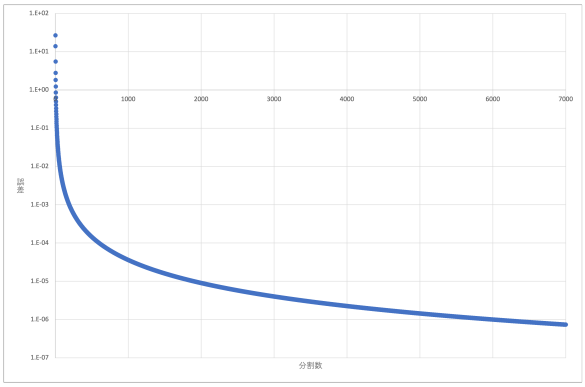


図 3: 分割数と平均誤差の関係

4　数値積分

コード 4 数値積分の誤差を測定 (testi.c)

```c
#include <math.h>
#include <stdio.h>
#include "func.h"
#define PRECISION 2000
int main() {
  int a, b; // b は区間の分割数
  double x, d, s;
  FILE *fp;
  fp = fopen("f2i.csv", "wt");
  for (b = 1; b <= PRECISION; b++) {
    d = 1.0 / (double)b;
    s = 0.0;
    for (a = 0; a < b; a++) {
      x = (double)a / (double)b;
      s += (f2(x, 0, 17) + f2(x + d, 0, 17)) / 2.0 * d;
    }
    fprintf(fp, "%d,%.20lf\n", b, fabs(f2(0.0, 2, 17) - s));
  }
  fclose(fp);
}
```

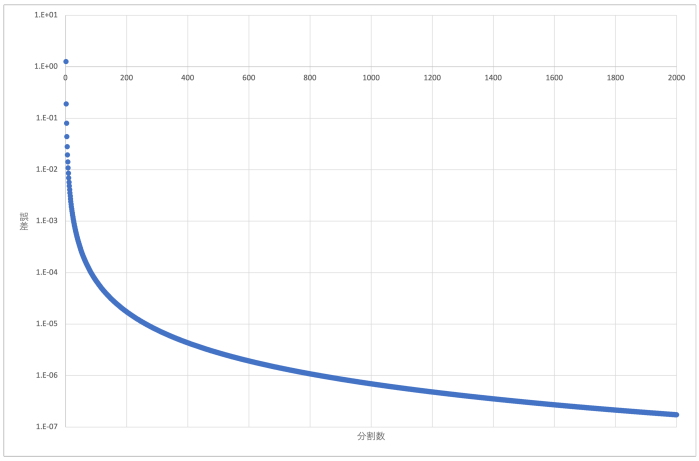


図 4: 数値積分の分割数に対する積分値と真値

台形面積計算による数値積分を行うことで近似を求めている．また f2 の第二引数に 2 をわたすことで積分値を得, 誤差を求めている．図 4 より精度を 10 倍にするためにはより多くの分割数が必要なことがわかる．

5　前進差分法や後退差分法との比較

前進差分法と後退差分法はそれぞれ次の式で表せる.

$$\text{前進差分法}\frac{\partial f}{\partial x} = \frac{f(x_i + \Delta x) - f(x_i)}{\Delta x}$$

$$\text{後退差分法}\frac{\partial f}{\partial x} = \frac{f(x_i) - f(x_i + \Delta x)}{\Delta x}$$

ここで $f(x_i + \Delta x)$ に対してテイラー展開をすると, 前進差分法では

$$\frac{\partial f}{\partial x} = \frac{1}{\Delta x}\left(f(x_i) + \Delta x\frac{\partial f(x_i)}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 f(x_i)}{\partial x^2} + \cdots - f(x_i)\right) = \frac{\partial f(x_i)}{\partial x} + \frac{\Delta x}{2!}\frac{\partial^2 f(x_i)}{\partial x^2} + \cdots$$

$$\begin{aligned}\left(\frac{\partial f}{\partial x}\right)_i &= \frac{1}{\Delta x}\left(f(x_i) + \Delta x\frac{\partial f(x_i)}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 f(x_i)}{\partial x^2} + \cdots - f(x_i)\right) \\ &= \frac{\partial f(x_i)}{\partial x} + \frac{\Delta x}{2!}\frac{\partial^2 f(x_i)}{\partial x^2} + \cdots\end{aligned}$$

移項すると

$$\left(\frac{\partial f}{\partial x}\right)_i - \frac{\partial f(x_i)}{\partial x} = O(\Delta x)$$

同様に後退差分法も行うと, それぞれの誤差は Δx に比例することがわかる.

次に数値微分で行った中心差分を次式に表す.

$$\left(\frac{\partial f}{\partial x}\right)_i = \frac{f_{i+1} - f_{i-1}}{2\Delta x}$$

こちらも f_{i+1}, f_{i-1} に関して同様にテイラー展開を行うと,

$$\begin{aligned}f_{i+1} &= f_i + \Delta x\frac{\partial f(x_i)}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 f(x_i)}{\partial x^2} + \frac{\Delta x^3}{3!}\frac{\partial^3 f(x_i)}{\partial x^3} + \cdots \\ f_{i-1} &= f_i - \Delta x\frac{\partial f(x_i)}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 f(x_i)}{\partial x^2} - \frac{\Delta x^3}{3!}\frac{\partial^3 f(x_i)}{\partial x^3} + \cdots\end{aligned}$$

与式に代入し整理すると,

$$
\begin{aligned}
\left(\frac{\partial f}{\partial x}\right)_i &= \frac{f_{i+1} - f_{i-1}}{2\Delta x} \\
&= \frac{1}{2\Delta x}\left(2\Delta x \frac{\partial f(x_i)}{\partial x} + \frac{2\Delta x^3}{3!}\frac{\partial^3 f(x_i)}{\partial x^3} + \cdots\right) \\
&= \frac{\partial f(x_i)}{\partial x} + \frac{\Delta x^2}{3!}\frac{\partial^3 f(x_i)}{\partial x^3} + \cdots
\end{aligned}
$$

ここで式変形を行い,

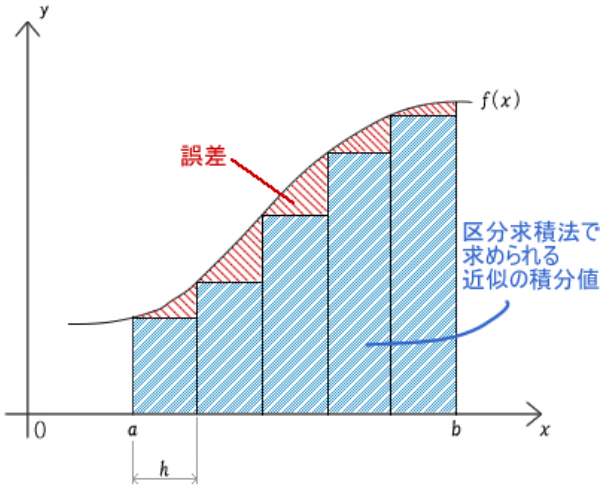
$$
\left(\frac{\partial f}{\partial x}\right)_i - \frac{\partial f(x_i)}{\partial x} = O(\Delta x^2)
$$

従って誤差は Δx^2 に比例することがわかる.

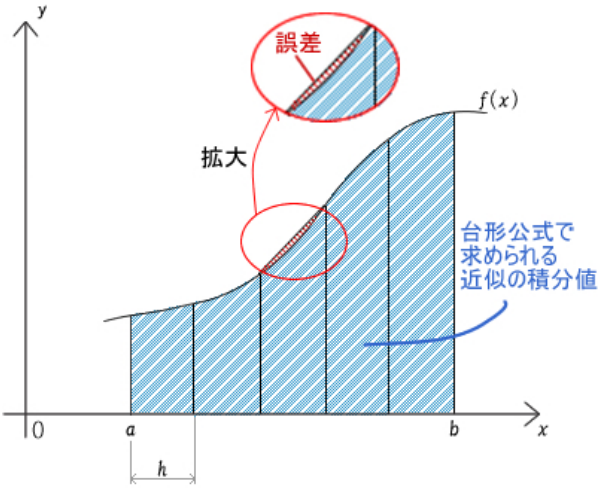
これらの結果から, 中心差分法が前後の値から推測するため最も誤差が小さく汎用性に足ることがわかる. 反対に前進差分法や後退差分法は誤差が大きく, 前の値もしくは後ろの値に強く影響されるため, 元の関数によっては収束せず振動ないし発散が起きてしまうことが伺えた. また後退差分法は予測などの手法には最適であると考えた.

6　長方形の積分との比較

区分求積法と台形公式の簡易図を次に示す. それぞれの図からわかるように台形公式の方はある程度 h が大きくても精度良く求めることができる. 反対に区分求積法では h が大きいと誤差も大きくなってしまうが, 限りなく h を小さくしたときには非常に精度良く値を求めることができる. しかし計算時間も膨大になってしまうため, 関数の性質や計算機の性能, 許容できる計算時間などから選択することが望ましい.



区分求積法



台形公式

図は「第 72 回　微分・積分の数学　数値積分　区分求積法・台形公式［前編］(https://gihyo.jp/dev/serial/01/java-calculation/0072)」より引用

7 参考文献

- 差分法の基礎 三好 隆博 広島大学大学院理学研究科 http://www.icehap.chiba-u.jp/activity/SS2016/textbook/SS2016_miyoshi_FD.pdf