

計算物理学B

第6回

野垣 康介、藤本 悠輝

講義予定

10/07 両名:四則演算

10/14 野垣:制御文(for, if)

10/21 野垣:関数

10/28 藤本:配列(numpy)

11/04 藤本:可視化(matplotlib)

11/11 野垣:数値微分

11/18 藤本:数値積分

～中間レポート～

12/09 野垣:モンテカルロ1

12/16 野垣:モンテカルロ2

12/23 藤本:微分方程式1

01/13 藤本:微分方程式2

01/20 藤本:微分方程式3

01/27 野垣:最適化

02/03 藤本:機械学習

～期末レポート～

はじめに

授業で用いるリンクをまとめておきます。

Google Colab

<https://colab.research.google.com/?hl=ja>

GitHub

https://github.com/nogaki/Computational_Physics_B

GitHub (今週の教材)

https://github.com/nogaki/Computational_Physics_B/tree/main/week6

はじめに

GitHubからGoogle Colabへのノートブックの取り込みについて

Colabの画面で

[ファイル]→[ノートブックを開く]→タブ[GitHub]をクリック

GitHub上のノートブックのurlを打ち込むと、直接取り込めます。

https://github.com/nogaki/Computational_Physics_B/tree/main/week6/week6.ipynb

[注意!]この状態では、GitHub上のファイルを開いただけで、
個人のドライブに保存されていません！
[ファイル]→[ドライブにコピーを保存]を実行すること。

数値微分と誤差

関数のテイラー展開 $|h| \ll 1$

$$f(x \pm h) = f(x) \pm f'(x)h + \frac{f''(x)}{2!}h^2 + \mathcal{O}(h^3)$$

を用いると数値微分は

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h)$$

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h^2)$$

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + \mathcal{O}(h^2)$$

とすれば簡単に求まりそうだ？

誤差との闘い！

計算機での数値の表現

計算機上のデータは全て0と1の羅列に過ぎない

00101001000111101101011...

0または1を取りうる、データの最小単位をbitと呼ぶ。

現代の計算機では、整数も小数も64bitが標準的。
(Pythonの整数型は例外)

1 byte = 8bit。

1 GB = 1024 MB = $(1024)^2$ KB
= $(1024)^3$ B = $8 \times (1024)^3$ bit

整数の表現

人間界の10進数の整数は0と1で表現不能

→2進数表示: 計算機界のスタンダード

$$13 \rightarrow (0 \ 1 \ 1 \ 0 \ 1)_2$$

$$25 \rightarrow (1 \ 1 \ 0 \ 0 \ 1)_2$$

$$2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0$$

の の の の の

桁 桁 桁 桁 桁

8bitの符号付き整数の表現 (64bitでも同じ)

$$00 \ 100 \ 101 = 37$$

↑
符号 絶対値

-128から127までを表現できる

整数の表現

$M+1$ bitの整数の10進数への変換



符号

絶対値 (M bit)

符号: 0なら正の数、1なら負の数

絶対値: M bitの2進数表現。0から 2^M-1

負の数の場合は、 2^M から絶対値を引いた数に
負号をつけた数として解釈する。(2の補数表現)

例) 8bit整数で符号 $\rightarrow 1$ 絶対値 $\rightarrow 32$ なら、 $-(128-32) = -96$ を表す

浮動小数の表現

小数も2進数表示が可能

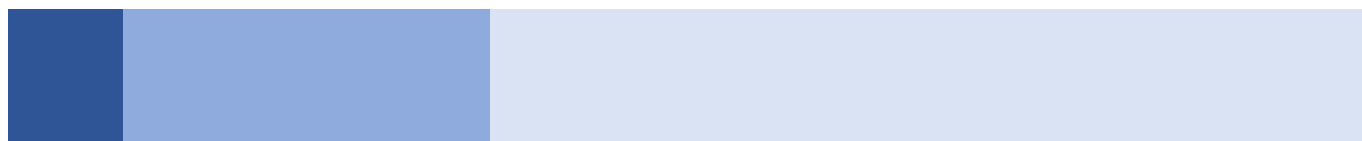
$$15.5 \rightarrow (1111.1)_2 = (0.11111)_2 \times 2^4$$

$$3.3 \rightarrow (11.011)_2 = (0.11011)_2 \times 2^4$$

2進数表示して、 2^{-1} の位が1になるように適切に指数をつける

$$\pm(0. \cdots)_2 \times 2^E \text{ 指数部}$$

符号 仮数部



符号

指数部

仮数部 (小数点以下)

1 bit

11 bit

52 bit

浮動小数の表現

仮数部の桁数が52であるように、計算機上の浮動小数で表現できる桁数には限りがある。



数学的には実数は稠密

ほとんど全ての実数は計算機上で正確に表現できない!

浮動小数の表現

0.1 を2進数表示すると…

$$0.1 = (0.00011001100110011\cdots)_2$$

0011 が繰り返される循環小数になる。

計算機では、あるところで打ち切られ、それ以降は記憶されない。

```
a = 0.1  
print(f"{a:.55f}")
```

0.1000000000000000000055511151231257827021181583404541015625

0.1すら正確に表現できない！

足し算と打ち切り誤差

桁落ちに注意する必要がある別の場面として、
(桁違いに) 大きさの異なる数同士の足し算がある。

```
sum = 0.0
for _ in range(10000000):
    sum = sum + 0.01

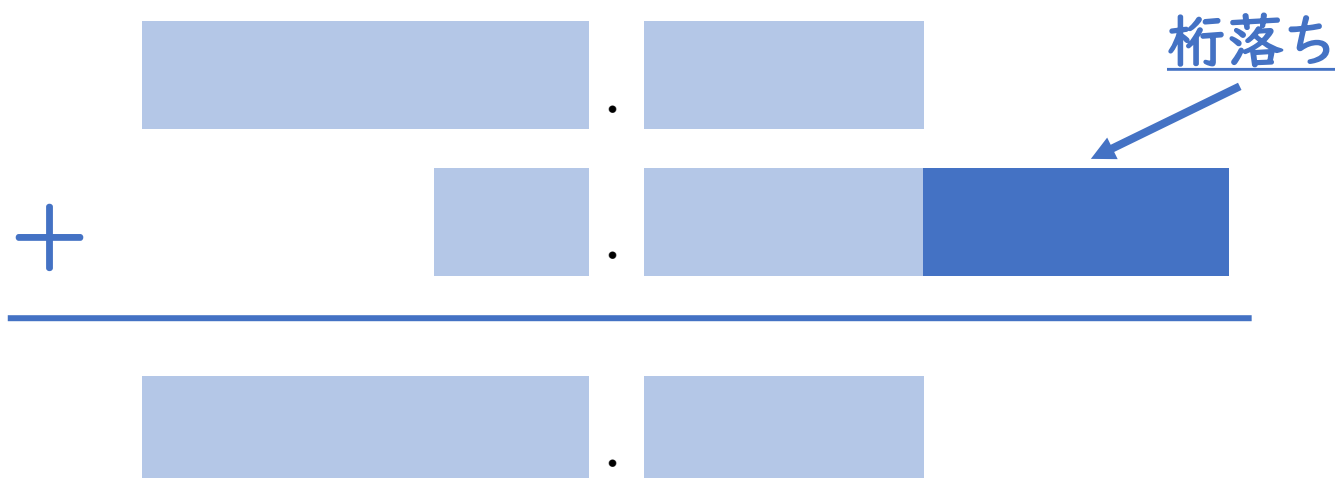
print(sum)
```

99999.99998630969

足されていない成分がある…?

足し算と桁落ち

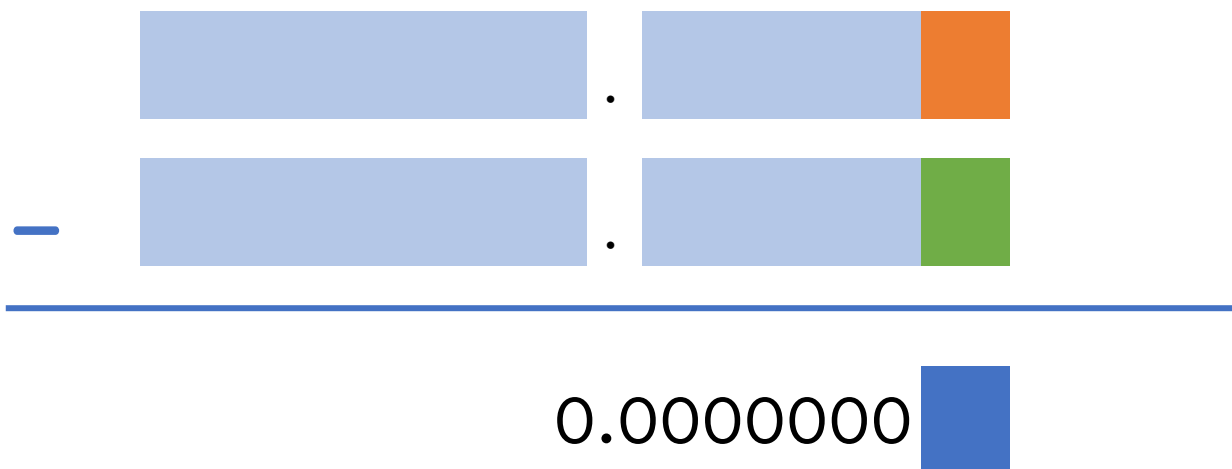
大きな数と小さな数の足し算



大きさの異なる数字の足し算に注意

引き算と桁落ち

ほとんど同じ大きさの数同士の引き算



引き算に用いるデータは途中で打ち切られているため、結果の有効数字が著しく低下する。

大きさの同じ数字の引き算に注意

数値微分と桁落ち

数値微分の公式

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h)$$

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h^2)$$

公式の精度を上げる→hを小さく

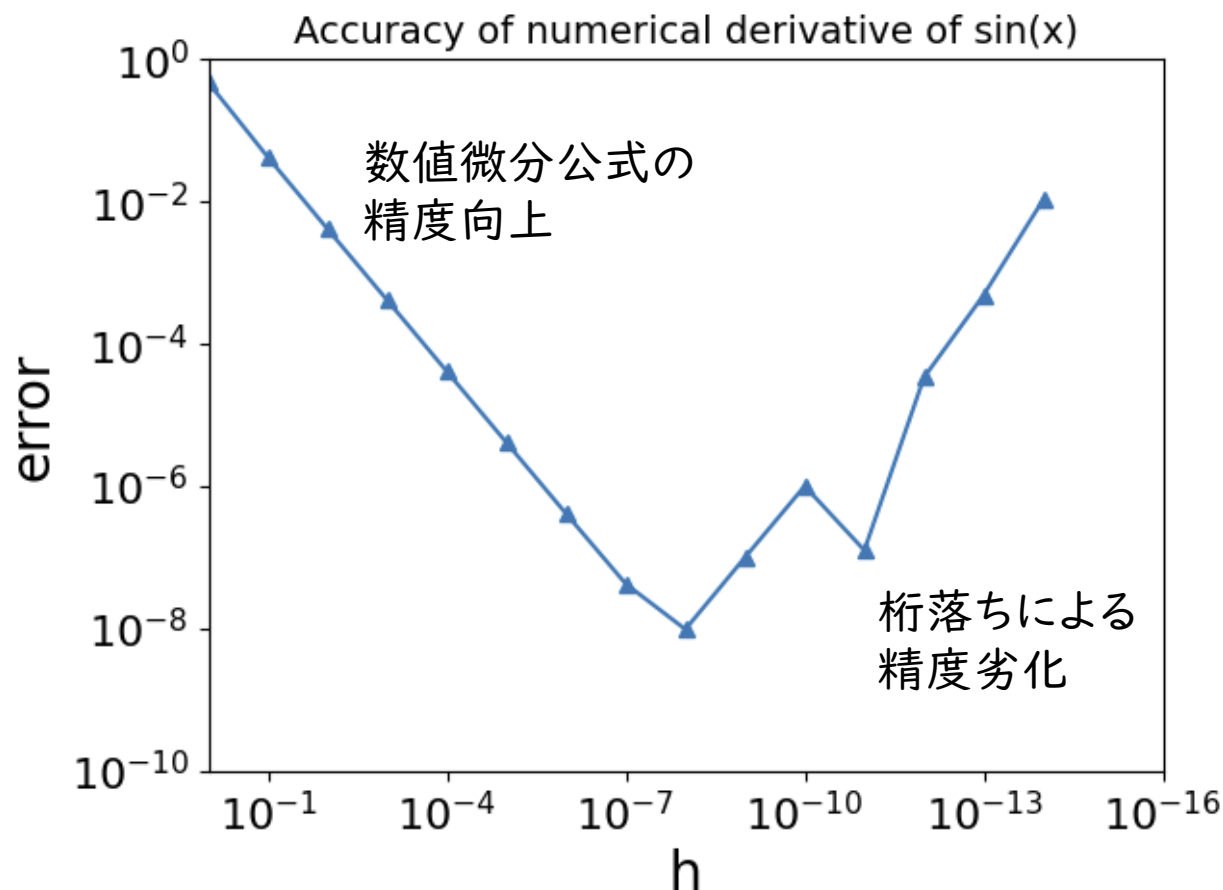
v.s

分子の引き算を安全に行う→hを大きく

丁度よいhを選ぶことが重要

数値微分と桁落ち

$\sin(x)$ の $x=0.3\pi$ における数値微分
errorは $\cos(0.3\pi)$ との誤差



数値微分と桁落ち

64 bit の浮動小数点では、10進数で 10^{-16} 程度の精度を持つ

公式 $f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h)$ を用いる場合は、精度の半分の $h = 10^{-8}$ と選ぶのが最適であることが多い。

$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h^2)$ を用いる場合は、精度の1/3の $h = 10^{-5}$ と選ぶのが最適であることが多い。

あくまで経験則。過信は禁物

実習タイム

例題

- 適当な多項式関数を数値微分し、解析解と比較せよ。
- 激しく変化する関数 $\sin(10^{-4}x)$ を数値微分し、最適な h の値が変わることを確認せよ。
- 以下の3つの公式によって、計算精度が変化するか確認せよ。

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h)$$

$$f'(x) = \frac{f(x) - f(x-h)}{h} + \mathcal{O}(h)$$

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h^2)$$