情報数値解析
第２回

丸め誤差と EFT

第２回・丸め誤差と EFT

- ✔ 講義の目的:
  浮動小数点演算における丸め誤差の発生と影響，丸め誤差の見積もりと制御方法および応用を概観します.
- ✔ 参考文献:
  - ✘ Nicholas J. Higham, Accuracy and Stability of Numerical Algorithms, Society for Industrial Mathematics, 2002.
  - ✘ S.M. Rump, Verification methods: Rigorous results using floating-point arithmetic, Acta Numerica, 19, 287–449, 2010.

講義のポイント

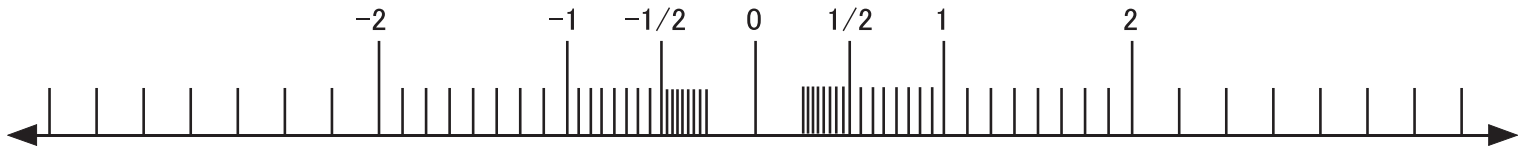
1. 整数の計算は，表現できる数値の範囲を超えない限り何進法でやっても同じ答が得られる．
2. 実数計算では，実数を有限桁の浮動小数点で近似するため「誤差」が生じる．
3. 誤差の発生と注意点．
4. IEEE 標準 754 規格における誤差の見積もり方法．

浮動小数点数

$$\pm \left(1 + \frac{d_1}{2} + \frac{d_2}{2^2} \cdots + \frac{d_n}{2^n}\right) \times 2^m$$

	単精度	倍精度
符合部	1 ビット	1 ビット
指数部	8 ビット	11 ビット
仮数部	23 ビット	52 ビット

- ✔ 符合部は正の場合は 0，負の場合は 1.
- ✔ 仮数部の最初の数は 1 に決まっていることから，実際の仮数部は有効桁数が 1 ビット多く表現できる (**けち表現**，**隠れビット**).
- ✔ 浮動小数点数は数直線に一様に分布していない.



$L = -1, U = 2, n = 4$ の例

絶対誤差と相対誤差

- ✔ 真の値 x と近似値 $\tilde{x}$ との差

$$\tilde{x} - x$$

を**誤差** (*error*) または**絶対誤差** (*absolute error*) と呼ぶ.

- ✔ $x - \tilde{x}$ は『補正』と呼ばれる.
ただし，$\tilde{x} - x$ や絶対値 (またはノルム) の意味での値を誤差または絶対誤差と定義することもある.

- ✔ 誤差が真の値に対して相対的にどのくらいであるのかの比率を表す

$$\frac{\tilde{x} - x}{x}$$

を**相対誤差** (*relative error*) と呼ぶ.

＊確定した用語ではないので，実際使うときには定義を明記することをお勧めします.

丸め誤差

- ✔ 実数 x をそれに近い浮動小数点数 $\tilde{x}$ で近似的に表現することを**丸め** (*rounding* または *round-off*) と呼び，丸めによって生じる誤差を **丸め誤差** (*rounding error* または *round-off error*) 呼ぶ.
- ✔ 計算機内の一連の有限桁演算によって生じた誤差のことを総称して「丸め誤差」と呼ぶこともある.
- ✔ 丸め誤差は計算機内部で有限桁の計算をするために必然的に生じる誤差.
- ✔ 多くの場合，桁数を多くとる (例えば単精度を倍精度に変える) ことで実用上の問題を回避できることが多い.
- ✔ しかし，丸め誤差の影響による桁落ち，情報落ちには注意が必要.

マシンイプシロン

実数 x の丸めを $fl(x)$ と書く．浮動小数点の絶対値最大を F_{max}，絶対値最小を F_{min} とするとき，丸め誤差の絶対値の意味での相対誤差の最大値:

$$\varepsilon_M := \max_{F_{\mathsf{min}} \le |x| \le F_{\mathsf{max}}} \frac{|fl(x) - x|}{|x|}$$

を**マシンイプシロン**(*machine epsilon*) と呼ぶ.

- ✔ 「マシンエプシロン」「計算機イプシロン」「丸めの単位」とも呼ばれる．数値計算においては相対誤差限界の重要な指標となる.
- ✔ IEEE 標準 754 規格では単精度で $\varepsilon_M = 2^{-23} (\approx 1.19 \times 10^{-7})$，倍精度で $\varepsilon_M = 2^{-52} (\approx 2.22 \times 10^{-16})$

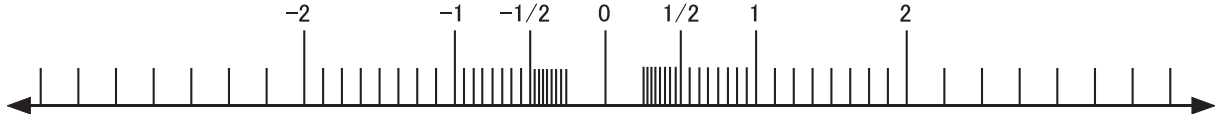
マシンイプシロンあれこれ

- マシンイプシロン ε_M は

$$1 + \varepsilon > 1$$

を満たす最小の正の浮動小数点数と定義することもできる.

- つまり,「1 から次の 1 より大きな浮動小数点数までの距離」のこと.



- 《注意》マシンイプシロンは「浮動小数点数の最小の値」**ではない**.
- マシンイプシロンは「相対的に x と $(1+\varepsilon_M)x$ の区別がつかなくなる最大の正の数」として，収束判定条件や誤差評価で利用される.

丸め誤差の影響: 桁落ち

絶対値がほぼ等しい 2 つ数を加減することで相対誤差が大きくなる現象を**桁落ち**(*cancellation*) と呼ぶ.

桁落ちが生じた場合，計算結果の仮数部の上位数桁が 0 になるために，有効数字の桁数が減り，結果として期待する結果が得られないことがある.

数値計算では，一般に小さい答を大きい数の差として求める演算は避けるべき.

$$\text{例: } x\sqrt{x^2-1}-x^2$$

また，問題によっては桁落ちの起こらない方法を見つけることもできる.

桁落ちの例

$$A\boldsymbol{x} = \boldsymbol{b}$$

$$A = \begin{bmatrix} 64919121 & -159018721 \\ 41869520.5 & -102558961 \end{bmatrix}, \quad \boldsymbol{b} = [1, 0]^T$$

$$\text{真の解 } \boldsymbol{x} = [205117922, 83739041]^T$$

《Cramer の公式》

$$x_1 = a_{22}/(a_{11}a_{22} - a_{12}a_{21}), \quad x_2 = -a_{21}/(a_{11}a_{22} - a_{12}a_{21})$$

IEEE 標準 754 規格の倍精度で計算

$$\Downarrow$$

$$\text{近似解 } \tilde{\boldsymbol{x}} = [102558961, 41869520.5]^T$$

情報落ち

絶対値の小さい数を大きい数に加えてもほとんど影響を与えない現象を**情報落ち**あるいは**積み残し**と呼ぶ.

$$S_1 \equiv \sum_{k=1}^{n} \frac{1}{k^2} = 1 + \frac{1}{4} + \frac{1}{9} + \frac{1}{16} + \cdots + \frac{1}{n^2}$$

n	単精度	倍精度
1000	1.643934850	1.643934567
4000	1.644713880	1.644684098
5000	1.644725320	1.644734087
6000	1.644725320	1.644767414
8000	1.644725320	1.644809075

単精度では，ある n から $1/n^2$ が和にほとんど影響を与えない.

情報落ちの回避例

$$S_2 \equiv \sum_{k=1}^{n} \frac{1}{(n-k+1)^2} = \frac{1}{n^2} + \frac{1}{(n-1)^2} + \frac{1}{(n-2)^2} + \cdots + \frac{1}{4} + 1$$

和を S_1 と逆にとる形に変更すると情報落ちが回避される.

n	単精度	倍精度
1000	1.643934490	1.643934567
4000	1.644684080	1.644684098
6000	1.644767400	1.644767414
8000	1.644809010	1.644809075

その他，情報落ちに関する話題:
http://www.fas.org/spp/starwars/gao/im92026.htm

丸め誤差の制御

丸めモード

IEEE 標準 754 規格では，以下の 4 つが規格化されている．
$\mathbb{F} \subset \mathbb{R}$: 浮動小数点数の集合，$c \in \mathbb{R}$

1. $\triangle : \mathbb{R} \to \mathbb{F}$: 上向きの丸め（round upward）
   c 以上の浮動小数点数の中で最も小さい数に丸める．

2. $\triangledown : \mathbb{R} \to \mathbb{F}$: 下向きの丸め（round downward）
   c 以下の浮動小数点数の中で最も大きい数に丸める．

3. $\square : \mathbb{R} \to \mathbb{F}$: 最近点への丸め（round to nearest）
   c に最も近い浮動小数点数に丸める．ちょうど中間の場合は，仮数部の最後のビットが偶数 (0) である浮動小数点数に丸める．

4. 切り捨て（round taward 0）
   絶対値が c 以下の浮動小数点数の中で c に最も近いものに丸める．

IEEE 標準 754 における四則演算

$\mathbb{F} \subset \mathbb{R}$: 浮動小数点数の集合，$c \in \mathbb{R}$

$\triangle : \mathbb{R} \to \mathbb{F}$: c 以上の最も小さい $\mathbb{F}$ の数への丸め

$\triangledown : \mathbb{R} \to \mathbb{F}$: c 以下の最も大きい $\mathbb{F}$ の数への丸め

$$* \in \{+,\ -,\ \cdot,\ /\},\ \bigcirc \in \{\triangle, \triangledown\}$$

$$x \circledast y = \bigcirc(x * y) \quad \forall x, y \in \mathbb{F}$$

計算機での四則演算の結果 $x \circledast y$ は，(数学的に正しい) 実数としての四則演算の結果 $x * y$ に指定された丸めを行なって得られた数 $\bigcirc(x * y)$ に一致

$$\Downarrow$$

計算機内の四則演算結果を評価することが可能

丸めモードの切り替え

IEEE 標準 754 規格の計算環境においては，丸めの方向を切り替えることによって，計算結果を不等式で評価できる場合がある．

《例》$\boldsymbol{x} = [x_1, \cdots, x_n]^T$, $\boldsymbol{y} = [y_1, \cdots, y_n]^T$ の内積の値 z を評価する．

〔下向きの丸め〕

$$\underline{z} = \sum_{k=1}^{n} x_k y_k$$

〔上向きの丸め〕

$$\overline{z} = \sum_{k=1}^{n} x_k y_k$$

$$\Longrightarrow \quad \underline{z} \leq z \leq \overline{z}$$

詳しくはまた別の機会に説明します．

Error-free 変換

“Error-free” とは

- ✔ 「誤差なし変換」と訳されることがある.
- ✔ （狭い意味では）IEEE 標準 754 の倍精度浮動小数点数の和と積で生じる丸め誤差を情報の欠落なく表現する手法.
- ✔ 現在，様々な応用研究がなされている.
- ✔ 以下は IEEE 標準 754 倍精度（64 ビット）に限定します.
  - ✘ $\mathbb{F}$: 倍精度浮動小数点数の集合
  - ✘ $\varepsilon := 2^{-52}$: マシンエプシロン
  - ✘ $\oplus$: 浮動小数点演算による和
  - ✘ $\ominus$: 浮動小数点演算による差
  - ✘ $\otimes$: 浮動小数点演算による積

Knuth, 1969

$a, b, x, y \in \mathbb{F}$ に対して，アルゴリズム TwoSum を適用したとき，以下が成り立つ．

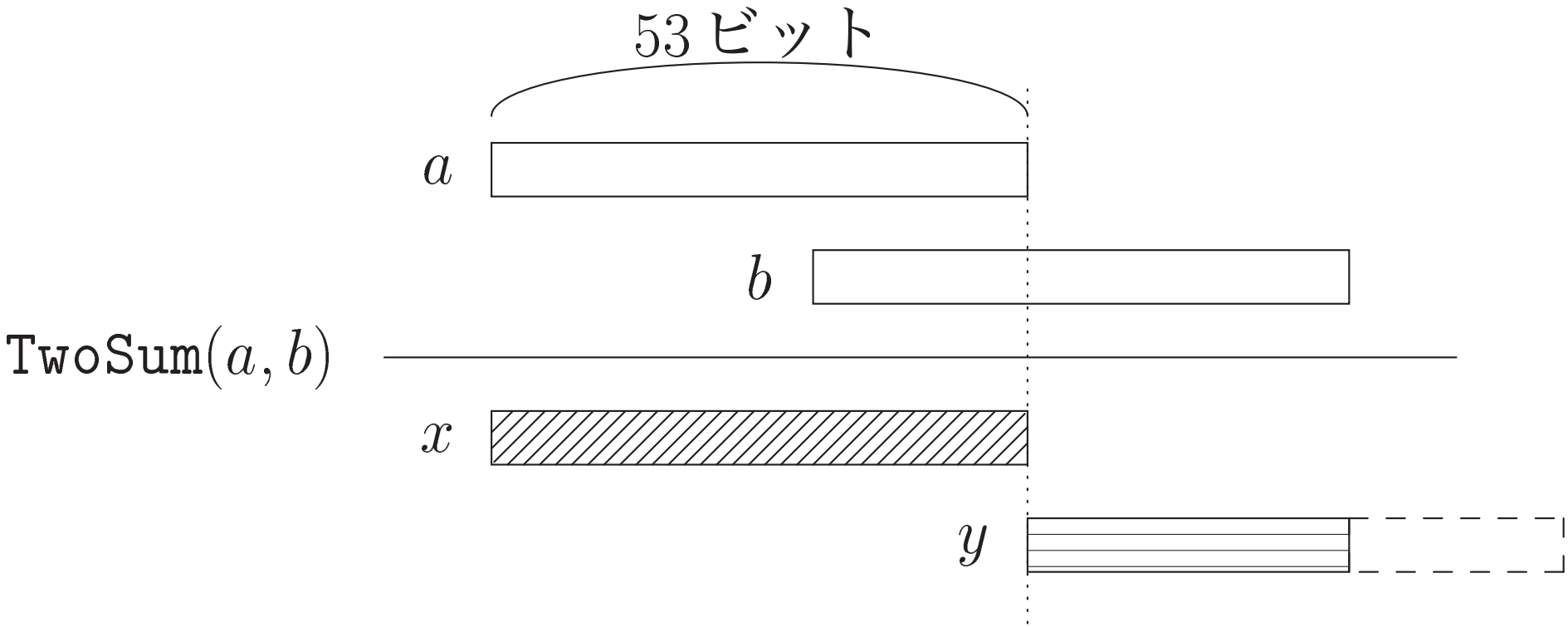
$$a + b = x + y \quad (|y| \leq \varepsilon |x|)$$

```
function [x, y] = TwoSum(a, b)
x = a ⊕ b;
c = x ⊖ a;
y = (a ⊖ (x ⊖ c)) ⊕ (b ⊖ c);
```

x は通常の和の計算結果．y に丸め誤差が**浮動小数点として**格納される．

D. E. Knuth: The Art of Computer Programming, Volume 2, Addison-Wesley, Reading, Massachusetts, 1969.

TwoSum のイメージ



5 回の追加計算で誤差を正確に拾うことができる.

Dekker, 1971

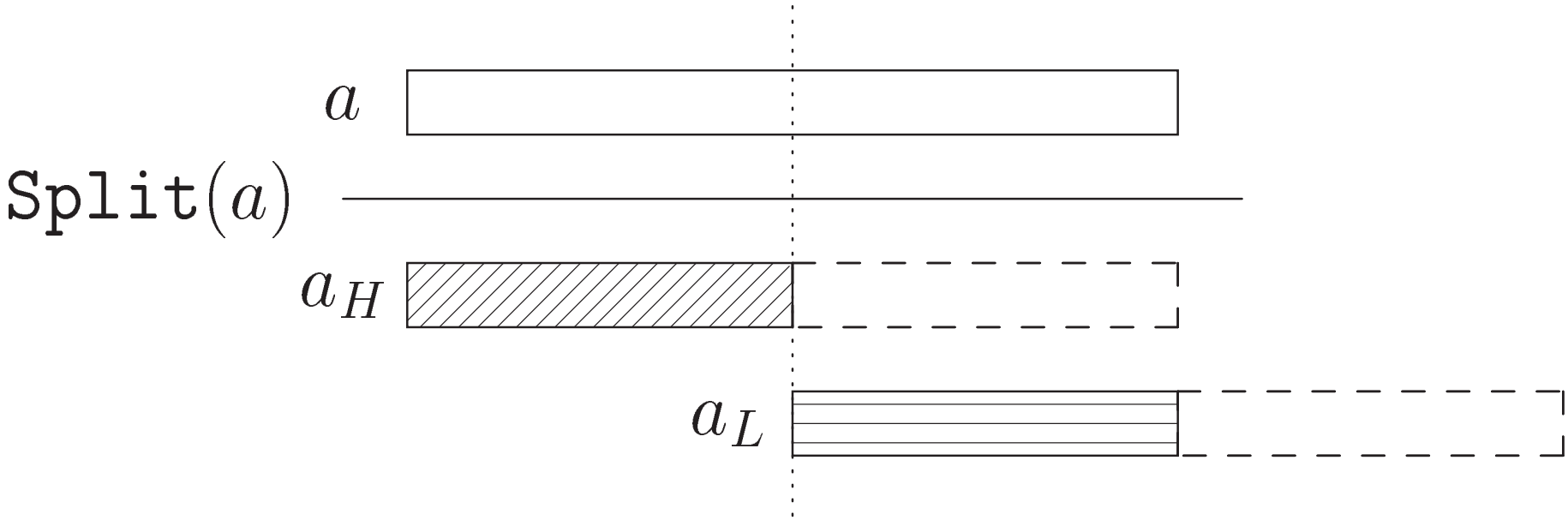
$a \in \mathbb{F}$ に対して，アルゴリズム Split を適用したとき，得られる $a_H, a_L \in \mathbb{F}$ に対して以下が成り立つ．

$$a = a_H + a_L$$

function $[a_H, a_L] = \mathtt{Split}(a)$;
$c = (2^{27} + 1) \otimes a$;
$d = c \ominus a$;
$a_H = c \ominus d$;
$a_L = a \ominus a_H$;

T. J. Dekker: A Floating-point Technique for Extending the Available Precision, Numer. Math. 18 (1971), 224–242.

Split のイメージ



a_H は a の上位仮数部，a_L は a の下位仮数部が正確に格納されている．

Dekker, 1971

$a, b, x, y \in \mathbb{F}$ に対して，アルゴリズム TwoProduct を適用したとき，以下が成り立つ．

$$a \times b = x + y \quad (|y| \leq \varepsilon |x|)$$

$\texttt{function}\ [x, y] = \texttt{TwoProduct}(a, b)$
$x = a \otimes b;$
$[a_H, a_L] = \texttt{Split}(a);$
$[b_H, b_L] = \texttt{Split}(b);$
$e_1 = x \ominus a_H \otimes b_H;$
$e_2 = e_1 \ominus a_L \otimes b_H;$
$e_3 = e_2 \ominus a_H \otimes b_L;$
$y = a_L \otimes b_L \ominus e_3;$

x は通常の積の計算結果．y に丸め誤差が**浮動小数点として**格納される．

FMA 命令

Fused-Multiply-and-Add 命令とは $a, b, c \in \mathbb{F}$ に対して

$$d = \mathrm{FMA}(a, b, c)$$

の結果が $d = \square(a * b + c)$ を保証する命令のこと．Itanium, Power アーキテクチャなどでサポートされている．
この命令が利用できる計算機では TwoProduct を以下の関数で置き換えることができる．

```
function [x, y] = TwoProductFMA(a, b)
x = a ⊗ b;
y = FMA(a, b, −x);
```

x は通常の積の計算結果．y に丸め誤差が**浮動小数点として**格納される．

EFT の展開

浮動小数点演算の加減算・積算における丸め誤差の把握方法を用いた応用例.

- ✔ 内積
- ✔ 行列とベクトルの積
- ✔ 連立１次方程式の解の反復改良
- ✔ 悪条件問題への適用
- ✔ etc.