# 「精度保証付き数値計算の基礎」チュートリアル
# 数学関数の精度保証

柏木 雅英
kashi@waseda.jp
http://verifiedby.me/

早稲田大学 基幹理工学部 応用数理学科

2018 年 9 月 10 日

# 概要

- 精度保証付き数値計算
- 区間演算
- 数学関数について
- kv ライブラリ
- 数学関数の実装
  - 指数関数
  - 対数関数
  - 三角関数
  - 逆三角関数

# 精度保証付き数値計算

## 精度保証付き数値計算

近似解だけでなく、その解の数学的に厳密な誤差評価をも計算する数値計算法。

## 必要な技術

- 区間演算 (切り捨てと切り上げを併用して計算に混入した誤差を把握する。関数の像の評価も行う。)
- 不動点定理 (不動点定理の成立の十分条件を区間演算で確認することによって、方程式の解の存在と存在範囲を保証する)
- 自動微分も重要。

# 計算機の実数の表現 (IEEE 754 倍精度浮動小数点数)

## 実数の表現

- $6.02 \times 10^{23}$ のような、「1 に近い数 $\times$ ベキ数」の形で記憶する。浮動小数点形式という。
- 2 進数で記憶し、64bit=8byte を使う。
- 64bit を以下のように分割し、

| $s$ | $e_{10}$ | $e_9$ | $\cdots$ | $e_0$ | $m_{51}$ | $m_{50}$ | $\cdots$ | $m_0$ |
|---|---|---|---|---|---|---|---|---|

$e$ を 2 進整数 $(e = \sum_{i=0}^{10} e_i 2^i)$、$m$ を 2 進小数 $(m = \sum_{i=0}^{51} m_i 2^{i-52})$ として

$$(-1)^s \times (1 + m) \times 2^{e-1023}$$

で計算される数値と対応する。

- 計算精度 (有効数字) は、$2^{-52} \simeq 2.22 \times 10^{-16} \simeq 10^{-15.65}$ なのでおよそ 16 桁。

## IEEE 754 倍精度フォーマットの詳細

| $s$ | $e_{10}$ | $e_9$ | $\cdots$ | $e_0$ | $m_{51}$ | $m_{50}$ | $\cdots$ | $m_0$ |
|---|---|---|---|---|---|---|---|---|

$$e = \sum_{i=0}^{10} e_i 2^i, \quad m = \sum_{i=0}^{51} m_i 2^{i-52}$$

| | $m = 0$ | $m \neq 0$ |
|---|---|---|
| $e = 0$ | $\pm 0$ | $(-1)^s \times (0 + m) \times 2^{-1022}$ (非正規化数) |
| $1 \leq e \leq 2046$ | $(-1)^s \times (1 + m) \times 2^{e-1023}$ (正規化数) | |
| $e = 2047$ | $\pm\infty$ | NaN (Not a Number) |

| 0 | $2^{-1074} \leq x < 2^{-1022}$ | $2^{-1022} \leq x < 2^{1024}$ | $+\infty$ |
|---|---|---|---|
| | 非正規化数 | 正規化数 | |

# 区間演算 (1/5)

## 区間演算

- 精度保証付き数値計算の基本技術。
- 数値を、計算機で表現可能な浮動小数点数を両端に持つ閉区間 $X = [a, b]$ で表現する。
- 区間同士の演算は、集合値演算として計算結果として有り得る値を包含するように行う。
- IEEE754 標準で定義されている「方向付き丸め」を利用する。
- 「丸め誤差の把握」と「関数の値域の評価」の 2 つの役割がある。

# 区間演算 (2/5)

- $X = [a, b], Y = [c, d]$
- $\underline{\cdot}, \overline{\cdot}$ はそれぞれ下向き丸め、上向き丸め

- 加算 $X + Y = [a \underline{+} c, b \overline{+} d]$
- 減算 $X - Y = [a \underline{-} d, b \overline{-} c]$
- 乗算 $X \times Y =$

| | $d < 0$ | $c \le 0, d \ge 0$ | $c > 0$ |
|---|---|---|---|
| $b < 0$ | $[b \underline{\times} d, a \overline{\times} c]$ | $[a \underline{\times} d, a \overline{\times} c]$ | $[a \underline{\times} d, b \overline{\times} c]$ |
| $a \le 0, b \ge 0$ | $[b \underline{\times} c, a \overline{\times} c]$ | $[\min(a \underline{\times} d, b \underline{\times} c), \max(a \overline{\times} c, b \overline{\times} d)]$ | $[a \underline{\times} d, b \overline{\times} d]$ |
| $a > 0$ | $[b \underline{\times} c, a \overline{\times} d]$ | $[b \underline{\times} c, b \overline{\times} d]$ | $[a \underline{\times} c, b \overline{\times} d]$ |

- 除算 $X/Y =$

| | $d < 0$ | $c > 0$ |
|---|---|---|
| $b < 0$ | $[b \underline{/} c, a \overline{/} d]$ | $[a \underline{/} c, b \overline{/} d]$ |
| $a \le 0, b \ge 0$ | $[b \underline{/} d, a \overline{/} d]$ | $[a \underline{/} c, b \overline{/} c]$ |
| $a > 0$ | $[b \underline{/} d, a \overline{/} c]$ | $[a \underline{/} d, b \overline{/} c]$ |

($Y \not\ni 0$ の場合のみ定義される)

- 平方根 $\sqrt{X} = [\underline{\sqrt{a}}, \overline{\sqrt{b}}]$

# 区間演算 (3/5)

## 10 進数有効数字 3 桁で $(1 \div 3) \times 3$ を計算すると $\cdots$

- 普通の数値計算

$$1 \div 3 = 0.333$$
$$0.333 \times 3 = 0.999$$

四捨五入による誤差がでる。

- 区間演算

最初は幅のない区間からスタート。

$$[1, 1] \div [3, 3] = [0.333, 0.334]$$
$$[0.333, 0.334] \times [3, 3] = [0.999, 1.01]$$

常に区間内に真の値を含みながら計算が進む。

# 2 次方程式の丸め誤差

## 2 次方程式

2 次方程式 $ax^2 + bx + c = 0$ の解の公式は、

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

## 大きな誤差が見られる例

$$a = 1,\ b = 10^{15},\ c = 10^{14}$$

のときの大きい方の解を計算する。

- (解の公式) $\dfrac{-b + \sqrt{b^2 - 4ac}}{2a} = -0.125$
- (解の公式の分子を有理化)
  $\dfrac{2c}{-b - \sqrt{b^2 - 4ac}} = -0.10000000000000002$

# 2 次方程式の例を区間演算で計算する

## 2 次方程式

2 次方程式 $ax^2 + bx + c = 0$ の解の公式は、

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

## 大きな誤差が見られる例

$$a = 1,\ b = 10^{15},\ c = 10^{14}$$

のときの大きい方の解を計算する。

- (解の公式) $\dfrac{-b + \sqrt{b^2 - 4ac}}{2a} = [-0.1875, -0.0625]$
- (解の公式の分子を有理化) $\dfrac{2c}{-b - \sqrt{b^2 - 4ac}} =$
  $[-0.10000000000000004, -0.09999999999999991]$

# 区間演算 (4/5)

## 丸めの向きの変更方法

- IEEE754 で丸めの向きを変更可能にすることが要請されているが、その方法は CPU やコンパイラによって違う。
- X86 だと FPU と SSE2 の 2 種類の演算器で丸めの向きの変更方法が異なるなど複雑。
- 最近は C99 準拠のコンパイラが増えたので、`fenv.h` と `fesetround` を使えば簡単になった。

```
#include <iostream>
#include <fenv.h>
int main()
{
    double x=1, y=10, z;
    std::cout.precision(17);

    fesetround(FE_TONEAREST);
    z = x / y;
    std::cout << z << std::endl;
    fesetround(FE_DOWNWARD);
    z = x / y;
    std::cout << z << std::endl;
    fesetround(FE_UPWARD);
    z = x / y;
    std::cout << z << std::endl;
}
```

```
0.10000000000000001
0.099999999999999991
0.10000000000000001
```

# 区間演算 (5/5)

## IEEE754 における計算精度と丸めの向きの変更

| | 完全精度 (0.5ulp) | 丸めの向きの変更の可否 |
|---|---|---|
| 加減乗除、平方根 | OK | OK |
| それ以外の関数 (exp など) | NG | NG |

平方根以外の関数は全く信用できない。従って、精度保証付き数値計算を実現するには数学関数を自作する必要がある。

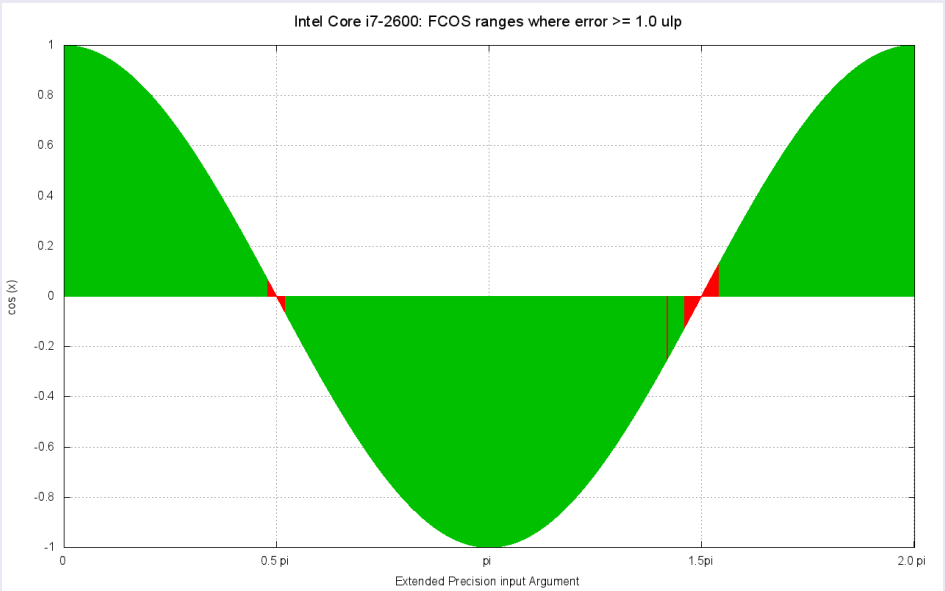
# Intel FPU の数学関数の精度

Intel overstates FPU accuracy (http://notabs.org/fpuaccuracy/) より

## Pentium プロセッサのマニュアルの記述

On the Pentium processor, the worst case error on functions is less than 1 ulp when rounding to the nearest-even and less than 1.5 ulps when rounding in other modes.

## FCOS で誤差が 1ulp 以上の部分



# 精度保証付き数学関数の実装例 (1/3)

## INTLAB

- `http://www.ti3.tu-harburg.de/rump/intlab/`
- S. M. Rump による、MATLAB で動く精度保証付き数値計算用ソフトウェア
- 精度保証付き区間演算が可能。table lookup を用いて、MATLAB 用としてはある程度高速。double を端点に持つ区間専用。
- 正しさの完全な証明はなされていない。

# 精度保証付き数学関数の実装例 (2/3)

## crlibm

- `http://lipforge.ens-lyon.fr/www/crlibm/`
- 完全精度 (誤差 0.5ulp) を達成する数学関数ライブラリ。上向き丸めと下向き丸めの計算も可能。
- 高速性も追求されている。正しさを証明する長大な論文あり。
- 残念ながら 2010 年 3 月 4 日の crlibm-1.0beta4.tar.gz を最後に更新を停止。プロジェクトページもアクセス出来なくなっている。
- 形式的証明 (formal proof) とコードの自動生成を目指した、後継の MetaLibm (`http://www.metalibm.org/`) というプロジェクトが立ち上がっている。

# 精度保証付き数学関数の実装例 (3/3)

## MPFR

- https://www.mpfr.org/
- 内部に gmp (http://gmplib.org/) を用いた、任意精度浮動小数点計算のためのライブラリ。
- 任意精度の最近点丸め、上向き、下向き丸めの可能な数学関数を持つ。
- 広く使われており信頼性は高いが、内部計算は多倍長整数に頼っているので計算速度は速くない。

# kv ライブラリ

## kv ライブラリ

- http://verifiedby.me/kv/ で公開中。
- 作成開始は 2007 年秋頃。公開開始は 2013 年 9 月 18 日。最新版は version 0.4.44。
- 言語は C++。boost C++ Libraries も必要。
- 全てヘッダファイルで記述されており、インストールはヘッダファイルをどこかに配置するだけ。
- オープンソースである。精度保証付き数値計算の結果が「証明」であると主張するならば、計算に使われたプログラムは必ず公開されているべき。
- 計算に使う数値の型は double に制限されていない。C++のテンプレート機能を用いて容易に変更することが出来る。
- (数値型) 区間演算 (多数の数学関数含む)、4 倍精度 (double-double) 演算、MPFR ラッパー、複素数演算、自動微分、affine arithmeric、ベキ級数演算、とそれらの組み合わせ。
- (アプリケーション) Krawczyk 法による非線形方程式の精度保証、非線形方程式の全解探索、常微分方程式の初期値問題、常微分方程式の境界値問題、数値積分、特殊関数、他

# 区間数学関数の実装のポリシー

- 加減乗除と平方根の方向付き丸めが可能な環境を想定し、それだけを用いてアルゴリズムを構成する。
- Taylor 展開など平易な数学のみを用いて、「誰が見てもこれは正しいだろう」と思えるようなものとする。簡単に実装可能であるようなアルゴリズムを目指す。
- 計算結果の区間幅は、1ulp(下限と上限が隣同士の浮動小数点) であることを目指さない。大きめの区間が帰ってくるが、その中に真値が間違いなく含まれていることを重視する。
- 高速性よりも信頼性を重視する。
- 加減乗除と平方根の方向付き丸めが可能でありさえすれば、区間の両端の数は倍精度浮動小数点数 (double) でなくてもよい。計算精度は machine epsilon を参照して適切に制御する。

# 指数関数 (exp)

## 幅の広い入力区間 $I = [x, y]$ に対する $\exp(I)$

exp は単調増加なので [$\exp(x)$ の下限, $\exp(y)$ の上限] を返す。

## 点入力 (幅 0 の区間) に対する $\exp(x)$

- $x = \infty$ なら $[\mathrm{F}_{\max}, \infty]$
- $x = -\infty$ なら $[0, \mathrm{S}_{\min}]$
- $x$ を $x = a + b, \quad a \in \mathbb{N}, \quad -\frac{1}{2} \le b \le \frac{1}{2}$ のように分解し、$\exp(x) = \mathrm{e}^a \exp(b)$ を計算。

## $-\frac{1}{2} \le b \le \frac{1}{2}$ に対する $\exp(b)$

$$\exp(b) \in 1 + b + \frac{1}{2!}b^2 + \frac{1}{3!}b^3 + \cdots + \frac{1}{n!}\left[\mathrm{e}^{-\frac{1}{2}}, \mathrm{e}^{\frac{1}{2}}\right] b^n$$

を区間演算で計算。($n$ は double なら 15 程度)

# 対数関数 (log)

## 幅の広い入力区間 $I = [x, y]$ に対する $\log(I)$

log は単調増加なので [$\log(x)$ の下限, $\log(y)$ の上限] を返す。

## 点入力 (幅 0 の区間) に対する $\log(x)$

- $x = 0$ なら $[-\infty, -\mathrm{F}_{\max}]$、$x = \infty$ なら $[\mathrm{F}_{\max}, \infty]$
- $x$ を $x = 2^a \cdot b, \quad 2a \in \mathbb{N}, \quad 2(\sqrt{2}-1) \simeq 0.83 \leq b \leq 4 - 2\sqrt{2} \simeq 1.17$ のように分解し、$\log(x) = a\log(2) + \log(b)$ を計算。

## $\log(b)$ の計算

$$\log(b) \in 0 + (b-1) - \frac{1}{2}(b-1)^2 + \frac{1}{3}(b-1)^3 - \frac{1}{4}(b-1)^4 + \cdots$$
$$+ (-1)^{n-1}\frac{1}{n}\mathrm{hull}(1, b)^{-n}(b-1)^n$$

を区間演算で計算。($n$ は double なら 22 程度)

# 三角関数 (sin, cos) (1/3)

## 幅の広い区間 $I = [x, y]$ に対する $\sin(I), \cos(I)$

- $x$ または $y$ が $\infty$ または $-\infty$ なら、$[-1, 1]$ を返す。
- $I' = [x', y'] = I - 2n\pi$ として、$-\pi \le x' \le \pi$ となるように正規化。
- $y' - x' \ge 2\pi$ なら、$[-1, 1]$ を返す。
- 以下を計算。

$$\sin(I') \in \mathrm{hull}(\ \sin x', \sin y', -1(I' \text{ が } -\tfrac{\pi}{2}, \tfrac{3}{2}\pi \text{ を含むときのみ}), 1(I' \text{ が } \tfrac{\pi}{2}, \tfrac{5}{2}\pi \text{ を含むときのみ}))\cap[-1, 1]$$

$$\cos(I') \in \mathrm{hull}(\ \cos x', \cos y', -1(I' \text{ が } -\pi, \pi, 3\pi \text{ を含むときのみ}), 1(I' \text{ が } 0, 2\pi \text{ を含むときのみ}))\cap[-1, 1]$$

# 三角関数 $(\sin, \cos)$ (2/3)

## $-\pi \leq x \leq \pi$ に対する $\sin(x), \cos(x)$

| | $\sin(x)$ | $\cos(x)$ |
|---|---|---|
| $-\pi \leq x \leq -\frac{3}{4}\pi$ | $-\sin(x+\pi)$ | $-\cos(x+\pi)$ |
| $-\frac{3}{4}\pi \leq x \leq -\frac{\pi}{2}$ | $-\cos(-\frac{\pi}{2}-x)$ | $-\sin(-\frac{\pi}{2}-x)$ |
| $-\frac{\pi}{2} \leq x \leq -\frac{\pi}{4}$ | $-\cos(x+\frac{\pi}{2})$ | $\sin(x+\frac{\pi}{2})$ |
| $-\frac{\pi}{4} \leq x \leq 0$ | $-\sin(-x)$ | $\cos(-x)$ |
| $0 \leq x \leq \frac{\pi}{4}$ | $\sin(x)$ | $\cos(x)$ |
| $\frac{\pi}{4} \leq x \leq \frac{\pi}{2}$ | $\cos(\frac{\pi}{2}-x)$ | $\sin(\frac{\pi}{2}-x)$ |
| $\frac{\pi}{2} \leq x \leq \frac{3}{4}\pi$ | $\cos(x-\frac{\pi}{2})$ | $-\sin(x-\frac{\pi}{2})$ |
| $\frac{3}{4}\pi \leq x \leq \pi$ | $\sin(\pi-x)$ | $-\cos(\pi-x)$ |

$0 \leq x \leq \frac{\pi}{4}$ の範囲の $\sin(x)$ または $\cos(x)$ の計算に帰着。

# 三角関数 (sin, cos) (3/3)

## $0 \leq x \leq \frac{\pi}{4}$ に対する $\sin(x), \cos(x)$

$$\sin x \in x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \cdots + \frac{1}{n!}[-1,1]x^n$$

$$\cos x \in 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 - \cdots + \frac{1}{n!}[-1,1]x^n$$

を区間演算で計算。($n$ は double なら 17 程度)

# 三角関数 (tan)

## 幅の広い入力区間 $I = [x, y]$ に対する $\tan(I)$

- $x$ または $y$ が $\infty$ または $-\infty$ なら，$[-\infty, \infty]$ を返す。
- $I' = [x', y'] = I - n\pi$ として、$-\frac{\pi}{2} < x' < \frac{\pi}{2}$ となるように正規化。
- $y' > \frac{\pi}{2}$ ならば、$[-\infty, \infty]$ を返す。
- [$\tan(x')$ の下限, $\tan(y')$ の上限] を返す。

## 点入力 (幅 0 の区間) に対する $\tan(x)$

単純に $\frac{\sin(x)}{\cos(x)}$ を区間演算で計算。

# 逆三角関数 (arctan)

## 幅の広い入力区間 $I = [x, y]$ に対する $\arctan(I)$

arctan は単調増加なので [$\arctan(x)$ の下限, $\arctan(y)$ の上限] を返す。

## 点入力 (幅 0 の区間) に対する $\arctan(x)$

| | |
|---|---|
| $x \geq \sqrt{2}+1$ | $\frac{\pi}{2} - \arctan(\frac{1}{x})$ |
| $\sqrt{2}-1 \leq x \leq \sqrt{2}+1$ | $\frac{\pi}{4} + \arctan(\frac{x-1}{x+1})$ |
| $-(\sqrt{2}-1) \leq x \leq \sqrt{2}-1$ | $\arctan(x)$ |
| $-(\sqrt{2}+1) \leq x \leq -(\sqrt{2}-1)$ | $-\frac{\pi}{4} + \arctan(\frac{1+x}{1-x})$ |
| $x \leq -(\sqrt{2}+1)$ | $-\frac{\pi}{2} - \arctan(\frac{1}{x})$ |

により、$x$ の変域を $|x| \leq \sqrt{2}-1 \simeq 0.41$ に限定する。

## $|x| \leq \sqrt{2}-1 \simeq 0.41$ に対する $\arctan(x)$

$$\arctan(x) \in x - \frac{1}{3}x^3 + \frac{1}{5}x^5 - \frac{1}{7}x^7 + \cdots + \frac{1}{n}[-1,1]x^n$$

を区間演算で計算。($n$ は double なら 38 程度)

# 逆三角関数 (arcsin)

## 幅の広い入力区間 $I = [x, y]$ に対する $\arcsin(I)$

arcsin は単調増加なので [$\arcsin(x)$ の下限, $\arcsin(y)$ の上限] を返す。

## 点入力 (幅 0 の区間) に対する $\arcsin(x)$

- $x < -1$ または $x > 1$ ならエラー
- $x = \pm 1$ のときは直接 $\pm\frac{\pi}{2}$ を返す。
- $-1 < x < 1$ のときは、

$$\arcsin(x) = \arctan\left(\frac{x}{\sqrt{1-x^2}}\right)$$

を区間演算で計算。

# 逆三角関数 (arccos)

## 幅の広い入力区間 $I = [x, y]$ に対する $\arccos(I)$

arccos は単調減少なので [$\arcsin(y)$ の下限, $\arcsin(x)$ の上限] を返す。

## 点入力 (幅 0 の区間) に対する $\arccos(x)$

- $x < -1$ または $x > 1$ ならエラー
- $x = -1, 1$ のときは直接 $\pi, 0$ を返す。
- $-1 < x < 1$ のときは、$\arccos(x) = \frac{\pi}{2} - \arctan\left(\frac{x}{\sqrt{1-x^2}}\right)$ を計算すればよいが、桁落ちを避けるため、$\alpha = \frac{x}{\sqrt{1-x^2}}$ を計算したあと次の表で計算する。

| | |
|---|---|
| $\alpha \geq \sqrt{2}+1$ | $\arctan(\frac{1}{\alpha})$ |
| $\sqrt{2}-1 \leq \alpha \leq \sqrt{2}+1$ | $\frac{\pi}{4} - \arctan(\frac{\alpha-1}{\alpha+1})$ |
| $-(\sqrt{2}-1) \leq \alpha \leq \sqrt{2}-1$ | $\frac{\pi}{2} - \arctan(\alpha)$ |
| $-(\sqrt{2}+1) \leq \alpha \leq -(\sqrt{2}-1)$ | $\frac{3\pi}{4} - \arctan(\frac{1+\alpha}{1-\alpha})$ |
| $\alpha \leq -(\sqrt{2}+1)$ | $\pi + \arctan(\frac{1}{\alpha})$ |

# 残りは...

- expm1
- log1p
- atan2
- 双曲線関数
- 逆双曲線関数

については、教科書を参照して下さい。

# おわりに

- kv ライブラリ (`http://verifiedby.me/kv/`) で実装されている区間数学関数のアルゴリズムについて解説した。
- kv ライブラリでは、double だけでなく double-double や mpfr を端点に持つ区間に対しても、本アルゴリズムで精度保証付き数学関数が計算できる。(mpfr の持っている数学関数は使わない。)
- double に対する区間演算だけでよければ、kv ライブラリのおまけの「シンプルな区間演算ライブラリ」(`http://verifiedby.me/kv/simple/index.html`) が、単一ファイルのインクルードのみで使えて簡単。
- 精度保証付き数値計算のご利用、あるいは kv ライブラリ (`http://verifiedby.me/kv/`) のご利用をお待ちしております!