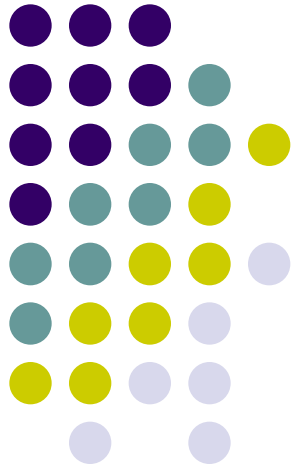
浮動小数点演算の無誤差変換と高精度計算のおはなし

quintia



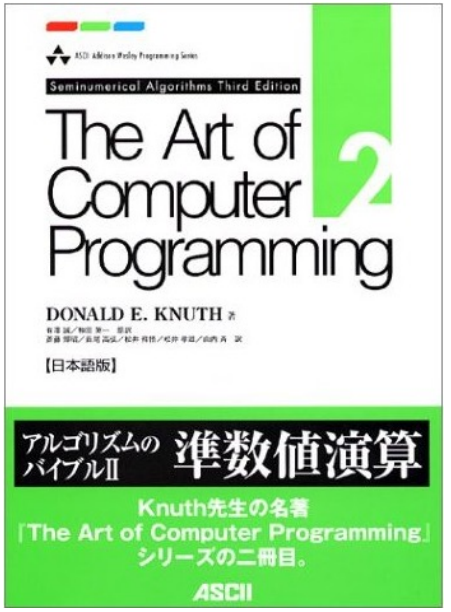
2008/11/22

今回のお題

このつぶやきで決めました

元ネタ

- 数学セミナー 2008年11月号 特集記事
  (日本評論社)



- D.E.Knuth:
  The Art of Computer Programming,
  Vol.2, 1969.

浮動小数点演算について

- 浮動小数点表現の限界
- どういう意味なのか?
- Knuth先生のお言葉

浮動小数点表現の限界

- $(1+2^{60}) - 2^{60}$

  正解は 1

- 浮動小数点(単精度)で演算すると……

  $fl(\ 1 \ +2^{60}\ ) = 2^{60}$

  $fl(\ 2^{60}-2^{60}\ ) = 0$

  $fl(\ )$ は
  「実数での演算じゃないよ
  浮動小数点での演算だよ」
  という印

- 0 になってしまう!

どういう意味なのか?

- 浮動小数点演算では「結合法則」が成立しない場合がある、ということ

$$fl((a + b) + c) \neq fl(a + (b + c))$$

- $fl( )$ の中身は「括弧を外して書いてはいけない」のだ!

Knuth先生のお言葉

- 「$a_1+a_2+a_3$」(略)などの数学的記法は，もともと結合法則が成り立つことを前提として作ってある．プログラマは，結合法則が成り立つ暗黙の前提で考えないように，特に注意しなければならない．

  The Art of Computer Programming 2
  第3版 日本語版 *p217*

無誤差変換

- 無誤差変換の概念
- 浮動小数点演算の誤差を計算する
- 誤差計算式の妙
- Knuth先生のお言葉 ふたたび

無誤差変換の概念

- 無誤差変換というよりは、実数演算の解と、浮動小数点演算の解との誤差という感覚

$$a + b = fl(a + b) + x$$

- x の部分が計算によって生じる誤差
- しかし、これを計算できるのか……？

浮動小数点演算の誤差を計算する

- 計算できる! (D.E.Knuth, 1969)

$$fl(\ (a\ -\ ((a + b)\ -\ (a\ -\ b)))\ +\ (b\ -\ (a\ -\ b))\ )$$

- 乗算については
  T.J.Dekker, (1971) によって
  示された式がある

誤差計算式の妙(1)

$$fl(\ (a\ -\ ((a + b)\ -\ (a\ -\ b)))\ +\ (b\ -\ (a\ -\ b))\ )$$

- 実数計算ならば括弧をはらえるのでちょっと試してみる

$$a - a - b + a - b + b - a + b = 0$$

- 実数計算では必ず0になる式!

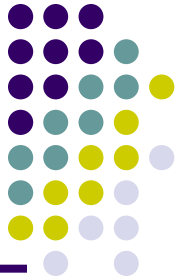
誤差計算式の妙(2)

$$fl(\ (a\ -\ ((a + b)\ -\ (a\ -\ b)))\ +\ (b\ -\ (a\ -\ b))\ )$$

結合法則が成立 ⇨ 値が0 ⇨ 誤差が生じなかった

結合法則が成立しない ⇨ 値が0以外 ⇨ 誤差が生じた

Knuth先生のお言葉 ふたたび



- 「$a_1+a_2+a_3$」(略)などの数学的記法は，もともと結合法則が成り立つことを前提として作ってある．プログラマは，結合法則が成り立つ暗黙の前提で考えないように，特に注意しなければならない．

  The Art of Computer Programming 2
  第3版 日本語版 *p217*

高精度計算の例を簡単に

$(1+2^{60}) - 2^{60}$

$1+2^{60}$	解 2^{60}	誤差 1
$2^{60} - 2^{60}$	解 0	誤差 0

解が0　誤差の総和が1

- 概念としては、解が1ということ

実際にコードを書いてみた

- Javaで書いてみた
- Javaの小数計算
- Delphiで書いてみた

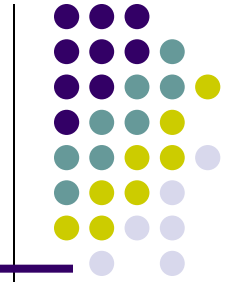
Javaで書いてみた

```
public static void main(String[] args) {
    float a = 1;
    float b = (float)Math.pow(2, 127);

    System.out.println("b: "+b);

    float u = a + b;

    float c = a - b;
    float v = (a-(u-c)) + (b-c);

    System.out.println("u: "+u);
    System.out.println("v: "+
}
```

```
結果
b: 1.7014118E38
u: 1.7014118E38
v: NaN
```

NaNが出てきた!

Javaの小数計算



- Float や Double に以下の様な定義がある
  - POSITIVE_INFINITY (正の無限大)
  - NEGATIVE_INIFINITY(負の無限大)
  - NaN(非数)

- +0.0 と -0.0 が区別されている

- 1.0 / +0.0 > POSITIVE_INFINITY
- 1.0 / -0.0 > NEGATIVE_INFINITY
- 0.0 / 0.0 > NaN

Delphiで書いてみた

```
var
  a,b,c,u,v: Single;
begin
  a:=1.0;
  b:=Power(2.0, 60.0);

  u:=a+b;
  c:=a-b;

  v:=(a-(u-c))+(b-c);

  ShowMessage(Format('%f  ,  %f', [u, v]));
end;
```

近似解 2^{60}

```
結果: 1.15292150460684698E18  ,  1.00
```

誤差 1

- 計画通り!!

ありがとうございました