実数の浮動小数点表現とその誤差 その 1

実際, 我々は計算機を用いて関数 $f(x,y)$ を計算し, 興味のある値 z を獲得する. よって, Δc には $f(x,y)$ の計算の過程で生じた「計算誤差」が含まれている. 今日, 計算機の性能が向上したからといって, この計算誤差が無視できるほどに小さくできるようになったわけではない. 計算誤差の例を以下に 3 つほど挙げる. 各自実際に計算し, 結果を考察されたい.

例 1

$n = 1, 2, \cdots, 10$ に対して $n^2, n^3, 1/n$ を求め, 表を出力する. 実際に手元の計算機で計算すると, 以下の表のようになる. プログラム例は ex1_d.f90 を参照されたい.

```
 1        n           n^2            n^3           1/n
 2        1      1.00000000     1.00000000    1.00000000
 3        2      4.00000000     8.00000000    0.50000000
 4        3      9.00000000    27.00000000    0.33333333
 5        4     16.00000000    64.00000000    0.25000000
 6        5     25.00000000   125.00000000    0.20000000
 7        6     36.00000000   216.00000000    0.16666667
 8        7     49.00000000   343.00000000    0.14285714
 9        8     64.00000000   512.00000000    0.12500000
10        9     81.00000000   729.00000000    0.11111111
11       10    100.00000000  1000.00000000    0.10000000
```

表を見ると, 特におかしなことは起こっていない. しかし, 伊理テキストによると, 同様の計算を大型機の FORTRAN で行うと, $1/10 = 0.1$ とならずに $1/10 =$

0.09999996 となったという[1)].

例 2

0.01 を 10,000 回足すプログラムを作り, 結果を出力する. 伊理テキストによると, パソコン BASIC と大型機 FORTRAN では,

$$\sum_{n=1}^{10000} 0.01 = \begin{cases} 100.003 & (\text{パソコン BASIC}) \\ 99.95277 & (\text{大型機 FORTRAN}) \end{cases} \tag{1.1}$$

という結果になったという. 実際に手元の計算機[2)] で計算すると,

$$\sum_{n=1}^{10000} 0.01 = \begin{cases} 100.00295 & (\text{単精度}) \\ 100.00000000001425 & (\text{倍精度}) \end{cases} \tag{1.2}$$

と言う結果になる. プログラム例は単精度[3)], 倍精度[4)] それぞれ ex2_e.f90, ex2_d.f90 を参照されたい.

例 3

$x = 0.0, 0.1, \cdots, 0.9, 1.0$ に対して x^2 を計算し合計するプログラムを, 伊理テキストのように,

$$\begin{aligned} & x = 0.0\ ;\ s = 0.0\ ; \\ & \text{while } x \leq 1.0 \text{ do} \\ & \text{begin } s = s + x^2\ ;\ x = x + 0.1 \text{ end}\ ; \\ & \text{print } s \end{aligned}$$

というような構造で作成する. 用いて作成する. 伊理テキストによると,

$$\sum_{n=0}^{10} (0.1n)^2 = \begin{cases} 2.85 & (\text{パソコン BASIC}) \\ 3.85 & (\text{大型機 FORTRAN}) \end{cases} \tag{1.3}$$

[1)] 伊理テキスト p.2 に出てくる FORTRAN の FORMAT, F14.8 は, 出力する値を 14 桁分の欄に小数点以下 8 桁の精度で書くという意味である. この 14 桁分の欄には符号, 小数点を含むことに注意されたい.

[2)] 手元の計算機のプロセッサは「Intel(R) Core(TM)2 Duo CPU P8600 @ 2.40GHz」である. FORTRAN コンパイラは gfortran (GNU Fortran (Debian 4.4.5-8) 4.4.5) を用いた.

[3)] 32bit 計算機の場合 32 bit で計算している.

[4)] 32bit 計算機の場合 64 bit で計算している.

という値になったという. 手元の計算機では,

$$\sum_{n=0}^{10}(0.1n)^2 = \begin{cases} 2.8500004 & (\text{単精度}) \\ 3.8499999999999996 & (\text{倍精度}) \end{cases} \tag{1.4}$$

という結果となった. プログラム例は単精度, 倍精度それぞれ ex3_e.f90, ex3_d.f90 を参照されたい. なお, 正答は,

$$\sum_{n=0}^{10}(0.1n)^2 = 3.85 \tag{1.5}$$

であり, これは数列の和の公式,

$$\sum_{k=0}^{n} k^2 = \frac{1}{6}n(n+1)(2n+1) \tag{1.6}$$

から容易に確かめられる[5)]. 正答と比較すると, パソコン BASIC, あるいは単精度計算において, 最後の $x = 1.0$ に対する項の加算をしないらしいことがうかがえる.

参考文献

伊理正夫・藤野和建, 1985: 数値計算の常識, 共立出版

[5)] 式 (1.6) を導出する. 恒等式,

$$(k+1)^3 - k^3 = 3k^2 + 3k + 1$$

を考え, k に $1, 2, \cdots, n$ を代入し, n 個の式の辺々を加えると,

$$(n+1)^3 - 1 = 3(1^2 + 2^2 + \cdots + n^2) + 3(1 + 2 + \cdots + n) + n$$

となる. よって,

$$\begin{aligned} 3(1^2 + 2^2 + \cdots + n^2) &= (n+1)^3 - 3(1 + 2 + \cdots + n) - (n+1) \\ &= (n+1)^3 - 3\frac{n(n+1)}{2} - (n+1) \\ &= \frac{n(2n+1)(n+1)}{2}. \end{aligned}$$

ゆえに,

$$\sum_{k=0}^{n} k^2 = \frac{1}{6}n(n+1)(2n+1).$$