実数の浮動小数点表現と誤差その**2**

浮動小数点

β 進数

計算機の中での実数の表現は“浮動小数点”の形であらわされる. その形は β 進数と 10 進数を併用した次の式で表される.

$$\pm(0.f_1f_2\cdots f_m)_\beta\times(\beta)_{10}{}^{\pm(E)_{10}}. \tag{1}$$

ここで $0.f_1f_2\ldots f_m$ は掛け算ではなく, $0.123\ldots$ のような数字の羅列を示す[1)].

この表記の β 進数で表された部分を仮数部 (mantissa) と呼ぶ. ここでの f_i は 0 から $\beta-1$ までの整数で $f_1\neq 0$ としている[2)]. 10 進表示された $\pm(E)_{10}$ のことを指数部 (exponent) と呼ぶ. この $(E)_{10}$ には 0 または正の整数が入る. β 進数も用いられている浮動小数点で表記された式を馴染みのある 10 進数のみの表記に戻すには

$$\begin{aligned}&\pm\,(0.f_1f_2\cdots f_m)_\beta\times(\beta)_{10}{}^{\pm(E)_{10}}\\&=\pm\left((f_1)_{10}(\beta)_{10}{}^{-1}+(f_2)_{10}(\beta)_{10}{}^{-2}+\cdots+(f_m)_{10}(\beta)_{10}{}^{-m}\right)\times(\beta)_{10}{}^{\pm(E)_{10}}\end{aligned} \tag{2}$$

を用いる.

10 進数と β 進数の相互変換:整数

10 進表示された整数 $(x)_{10}$ を $(a_ka_{k-1}\cdots a_0)_\beta(a_i=0,1,\cdots,\beta-1)$ と表記された β 進数に変換するときは (2) 式より

$$(x)_{10}=(a_k)_{10}(\beta)_{10}{}^{(k)_{10}}+(a_{k-1})_{10}(\beta)_{10}{}^{(k-1)_{10}}+\cdots+(a_1)_{10}(\beta)_{10}+(a_0)_{10} \tag{3}$$

[1)] 16 進数など,β が 10 より大きい場合は,$10,11,12\ldots$ を表す文字として $A,B,C\ldots$ を用いる.

[2)] $()_\beta$ と書かれた場合その中は β 進数で表記される. たとえば 10 進表示された値として 0.15625 という数値を考える. これを仮数部の様に表記すれば $(0.15625)_{10}$ となる. この数値は 16 進表示では $(0.28)_{16}$ と表記する. この値はどちらも同じ値である. つまり f_i は β によって値が変化する.

となるので $(x)_{10}$ を $(\beta)_{10}$ で割った余りを順に求めればよいということになる[3)]. 具体的に $(x)_{10} = (27)_{10}$ として 2 進表示してみる.

$$
\begin{array}{rll}
2) & 27 & \\
\hline
2) & 13 & \text{余り } 1 = a_0 \\
\hline
2) & 6 & \text{余り } 1 = a_1 \\
\hline
2) & 3 & \text{余り } 0 = a_2 \\
\hline
 & 1 = a_4 & \text{余り } 1 = a_3
\end{array}
$$

なので,

$$(27)_{10} = (11011)_2 \tag{4}$$

となる. 16 進表示は同じ方法でも求められるが 2 進表示が求められているときは下から 4 桁ごとに区切って, それぞれを 16 進表示に変換してもいい. 同じ例の場合

$$(1011)_2 = (11)_{10} = (B)_{16} \tag{5}$$

となるので残っている $(1)_2$ の部分はそのまま $(0001)_2$ としていいので

$$(0001)_2 = (1)_{16} \tag{6}$$

となる. よって

$$(27)_{10} = (1B)_{16}. \tag{7}$$

10 進数と β 進数の相互変換:純小数

10 進数の純小数 $(y)_{10}$ の β 進表示への変換は (2) 式より

$$(y)_{10} = (b_1)_\beta (\beta)_{10}{}^{-(1)_{10}} + (b_2)_\beta (\beta)_{10}{}^{-(2)_{10}} + \cdots + (b_m)_\beta (\beta)_{10}{}^{-(m)_{10}} \tag{8}$$

であるから, 小数部分を $(\beta)_{10}$ 倍してその整数部分を取り出していくことで求められる. 具体的に $(y)_{10} = (0.1)_{10}$ としたときの 16 進表示を求めてみる. まず $(0.1)_{10} = (b_1)_{16}(16)_{10}^{-(1)_{10}} + (b_2)_{16}(16)_{10}^{-(2)_{10}} + \cdots$ の式に $(16)_{10}$ をかけると

$$(1.6)_{10} = (b_1)_{16} + (b_2)_{16}(16)_{10}^{-(1)_{10}} + (b_3)_{16}(16)_{10}^{-(2)_{10}} + \cdots . \tag{9}$$

b_i は 0 から $(\beta)_{10} - 1$ までの整数, つまり b_i の最大は 15 であるので b_1 より後ろの項の計は 1 以下になる. よって, $(b_1)_{16} = (1)_{16}$ となる. 次に両辺から b_1 を引いて同様に行うと

$$(9.6)_{10} = (b_2)_{16} + (b_3)_{16}(16)_{10}^{-(1)_{10}} + \cdots . \tag{10}$$

[3)] (3) を辺々 β で割って確認せよ.

よって, $(b_2)_{16} = (9)_{16}$ となる. これを繰り返していくと

$$(b_2)_{16} = (b_3)_{16} = \cdots = (9)_{16}. \tag{11}$$

したがって

$$(0.1)_{10} = (0.19999\cdots)_{16} \tag{12}$$

2 進数にするには同様にやってもできるが整数のときと同じく 16 進表示から求める. 16 進表示された式を 2 進表示すると

$$\begin{aligned}(0.19999\cdots)_{16} &= (0.000110011001100\cdots)_2 \\ &= (0.110011001100\cdots)_2 \times (2)_{10}^{-(3)_{10}} \\ &= (0.\mathrm{CCC}\cdots)_{16} \times (2)_{10}^{-(3)_{10}}\end{aligned} \tag{13}$$

となる. なお, 見やすくするために, 3 行目で仮数部のみ 16 進表示にした.

10 進数と β 進数の相互変換:数値的な計算法

β 進表示された整数を 10 進数に戻す時には

$$(a_k a_{k-1}\cdots a_0)_\beta = (a_k)_{10}(\beta)_{10}{}^{(k)_{10}} + (a_{k-1})_{10}(\beta)_{10}{}^{(k-1)_{10}} + \cdots + (a_1)_{10}(\beta)_{10} + (a_0)_{10} \tag{14}$$

をそのまま計算していては効率が悪い. このままの場合は $(\beta)_{10}$ の乗算に $\dfrac{k(k+1)}{2}$ 回必要になる. そこで右辺にホーナー (Horner) 法を用いることで乗算の数を k 回まで下げられる.

$$\begin{aligned}(a_k a_{k-1}\cdots a_0)_\beta = \\ \{\cdots\{\{(a_k)_{10}\cdot(\beta)_{10} + (a_{k-1})_{10}\}\cdot(\beta)_{10} + (a_{k-2})_{10}\}\cdot(\beta)_{10} \\ + \cdots\}\cdot(\beta)_{10} + (a_1)_{10}\}\cdot(\beta)_{10} + (a_0)_{10}.\end{aligned} \tag{15}$$

β 進表示された純小数の場合は

$$(0.b_1 b_2 \cdots b_m)_\beta = (b_1 b_2 \cdots b_m)_\beta \times (\beta)_{10}{}^{-(m)_{10}} \tag{16}$$

とすれば整数のときと同様に計算できる.

表現誤差

実際の値は数直線上のどんなに狭い部分にも無限個の実数が含まれているため浮動小数点表示には表現の誤差が含まれる.

切り捨て

浮動小数点での表示を m 桁までできたとする. そのときそれより先の $m+1$ 以上の桁を切り捨てて表示したとするとその切り捨てた分が誤差になる. 今, 実際の値を z, 浮動小数点で表示できる部分を F, 切り捨てられた表現誤差を δ_1 とすると,

$$\delta_1 = z - F \tag{17}$$

となる. z と F を浮動小数点で表示すると,

$$\delta_1 = (0.f_1f_2\cdots)_\beta\times(\beta)_{10}{}^{(E)_{10}} - (0.f_1f_2\cdots f_m)_\beta\times(\beta)_{10}{}^{(E)_{10}} \tag{18}$$

となる. ここで簡単のため仮数部も指数部も正の値で考えている. (2) 式から

$$\begin{aligned}\delta_1 &= \Big((f_1)_{10}(\beta)_{10}{}^{-(1)_{10}} + \cdots + (f_m)_{10}(\beta)_{10}{}^{-(m)_{10}} + (f_{m+1})_{10}(\beta)_{10}{}^{-(m+1)_{10}} + \cdots\Big)\times(\beta)_{10}{}^{(E)_{10}} \\ &\quad - \Big((f_1)_{10}(\beta)_{10}{}^{-(1)_{10}} + \cdots + (f_m)_{10}(\beta)_{10}{}^{-(m)_{10}}\Big)\times(\beta)_{10}{}^{(E)_{10}} \\ &= (f_{m+1})_{10}(\beta)_{10}{}^{-(m+1)_{10}}\times(\beta)_{10}{}^{(E)_{10}} + (f_{m+2})_{10}(\beta)_{10}{}^{-(m+2)_{10}}\times(\beta)_{10}{}^{(E)_{10}} + \cdots .\end{aligned} \tag{19}$$

仮に m が十分大きく $(\beta)_{10}{}^{-(m+2)_{10}}$ 以降の項を極小だとする. そのとき, 誤差の値が $m+1$ 桁目の値で決まるとみなすと,

$$\delta_1 \approx (f_{m+1})_{10}(\beta)_{10}{}^{-(m+1)_{10}}\times(\beta)_{10}{}^{(E)_{10}}. \tag{20}$$

δ_1 が取り得る最大の値を取るのは $(f_{m+1})_{10}$ が最大値を取ったときである. よって, $(f_{m+1})_{10} = (\beta)_{10} - (1)_{10}$. このとき

$$\begin{aligned}\delta_1 &\le ((\beta)_{10} - (1)_{10})(\beta)_{10}{}^{-(m+1)_{10}}\times(\beta)_{10}{}^{(E)_{10}} \\ &= \Big((\beta)_{10}{}^{-(m)_{10}} - (\beta)_{10}{}^{-(m+1)_{10}}\Big)\times(\beta)_{10}{}^{(E)_{10}}.\end{aligned} \tag{21}$$

δ_1 の相対誤差は δ_1 を F で割ることで求められるので相対誤差を δ_{1r} とすると

$$\begin{aligned}\delta_{1r} &= \frac{\Big((\beta)_{10}{}^{-(m)_{10}} - (\beta)_{10}{}^{-(m+1)_{10}}\Big)\times(\beta)_{10}{}^{(E)_{10}}}{\Big((f_1)_{10}(\beta)_{10}{}^{-(1)_{10}} + \cdots + (f_m)_{10}(\beta)_{10}{}^{-(m)_{10}}\Big)\times(\beta)_{10}{}^{(E)_{10}}} \\ &\approx \frac{(\beta)_{10}{}^{-(m)_{10}}}{(f_1)_{10}(\beta)_{10}{}^{-(1)_{10}}}.\end{aligned} \tag{22}$$

相対誤差の取り得る最大の値は $(f_1)_{10} = (1)_{10}$ のときである. よって,

$$\begin{aligned}\delta_{1r} &\le \frac{(\beta)_{10}{}^{-(m)_{10}}}{(\beta)_{10}{}^{-(1)_{10}}} \\ &= (\beta)_{10}{}^{-(m-1)_{10}}\end{aligned} \tag{23}$$

となる.

四捨五入

浮動小数点の形で正確に表せる数の中間に境目を置いて, β 進法で四捨五入のようなことをする場合を考える[4]. 今, $m+1$ 桁目の値を四捨五入することを考える. このとき四捨五入される $m+1$ 桁目の値を z_2、四捨五入された後の値を z_2' とすると,

$$z_2 \equiv (f_{m+1})_{10}(\beta)_{10}{}^{-(m+1)_{10}} \times (\beta)_{10}{}^{(E)_{10}} + \cdots \tag{24}$$

$$z_2' = \begin{cases} (\beta)_{10}{}^{-(m)_{10}} \times (\beta)_{10}{}^{(E)_{10}} & \left((f_{m+1})_{10} \geq \frac{(\beta)_{10}}{(2)_{10}}\right) \\ 0 & \left((f_{m+1})_{10} < \frac{(\beta)_{10}}{(2)_{10}}\right) \end{cases} \tag{25}$$

である. ここで四捨五入による表現誤差を δ_2 とすると

$$\delta_2 = z - (F + z_2) \tag{26}$$

となる. 切り捨ての時と同様に浮動小数点で表したとすると,

$$\begin{aligned} \delta_2 &= (0.f_1 f_2 \ldots)_\beta \times (\beta)_{10}{}^{(E)_{10}} - ((0.f_1 f_2 \ldots f_m)_\beta \times (\beta)_{10}{}^{(E)_{10}} + z_2) \\ &= \left((f_1)_{10}(\beta)_{10}{}^{-(1)_{10}} + \cdots + (f_m)_{10}(\beta)_{10}{}^{-(m)_{10}} + (f_{m+1})_{10}(\beta)_{10}{}^{-(m+1)_{10}} + \cdots\right) \times (\beta)_{10}{}^{(E)_{10}} \\ &\quad - \left((f_1)_{10}(\beta)_{10}{}^{-(1)_{10}} + \cdots + (f_m)_{10}(\beta)_{10}{}^{-(m)_{10}}\right) \times (\beta)_{10}{}^{(E)_{10}} - z_2. \end{aligned} \tag{27}$$

今, $(f_{m+1})_{10} = \dfrac{(\beta)_{10}}{(2)_{10}}$ とする. このとき z_2 は繰り上がって,

$$\begin{aligned} \delta_2 &= \left((f_1)_{10}(\beta)_{10}{}^{-(1)_{10}} + \cdots + (f_m)_{10}(\beta)_{10}{}^{-(m)_{10}} + \frac{(\beta)_{10}}{(2)_{10}}(\beta)_{10}{}^{-(m+1)_{10}} + \cdots\right) \times (\beta)_{10}{}^{(E)_{10}} \\ &\quad - \left((f_1)_{10}(\beta)_{10}{}^{-(1)_{10}} + \cdots + (f_m + 1)_{10}(\beta)_{10}{}^{-(m)_{10}}\right) \times (\beta)_{10}{}^{(E)_{10}} \\ &= \left(\frac{(\beta)_{10}}{(2)_{10}}(\beta)_{10}{}^{-(m+1)_{10}} - (1)_{10}(\beta)_{10}{}^{-(m)_{10}}\right) \times (\beta)_{10}{}^{(E)_{10}} + \cdots \\ &= -\frac{(\beta)_{10}}{(2)_{10}}(\beta)_{10}{}^{-(m+1)_{10}} \times (\beta)_{10}{}^{(E)_{10}} + \cdots \end{aligned} \tag{28}$$

となる. 切り捨てのときと同様に $(\beta)_{10}{}^{-(m+2)_{10}}$ 以降の項が無視できるとすると,

$$\delta_2 \approx \frac{\beta_{10}^{-(m)_{10}}}{2} \times \beta_{10}{}^{(E)_{10}} \tag{29}$$

となる. $(f_{m+1})_{10}$ の値が $\dfrac{(\beta)_{10}}{(2)_{10}} - (1)_{10}$ 以下の時は繰り上がらずそのときの $(f_{m+1})_{10}(\beta)_{10}{}^{-(m+1)_{10}}$ が表現誤差になるので最終的に誤差の値が最大になるのは $(f_{m+1})_{10} = \dfrac{(\beta)_{10}}{(2)_{10}}$ のと

[4] ここでの四捨五入とは中間に境目をおいて値がその境目以上のときは繰り上げ, それより低いときは切り捨てを行う丸めのこと.

きである. 打切り誤差の時と同様に相対誤差 δ_{2r} も求めると

$$\begin{aligned}\delta_{2r} &= \frac{\frac{(\beta)_{10}^{-(m)_{10}}}{(2)_{10}}\times(\beta)_{10}{}^{(E)_{10}}}{((f_1)_{10}(\beta)_{10}{}^{-(1)_{10}}+\cdots+(f_m)_{10}(\beta)_{10}{}^{-(m)_{10}})\times(\beta)_{10}{}^{(E)_{10}}} \\ &\approx \frac{\frac{(\beta)_{10}^{-(m)_{10}}}{(2)_{10}}}{(f_1)_{10}(\beta)_{10}{}^{-(1)_{10}}} \end{aligned} \tag{30}$$

となる. 相対誤差の取り得る最大の値は $(f_1)_{10}=(1)_{10}$ のときである. よって,

$$\begin{aligned}\delta_{2r} &\le \frac{\frac{(\beta)_{10}^{-(m)_{10}}}{(2)_{10}}}{(\beta)_{10}{}^{-(1)_{10}}} \\ &= \frac{(\beta)_{10}^{-(m-1)_{10}}}{(2)_{10}}. \end{aligned} \tag{31}$$

具体例

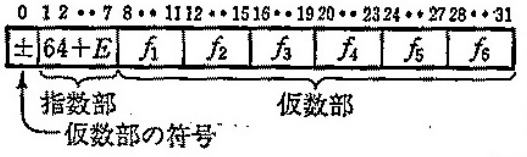
パソコンで使う場合は 2 進数かあるいは 16 進数を用いる[5)]. 1 つの数は定まったビット数の 1 語に収められることになっているため[6)], 1 語で表現しうる数の種類はこのビット数に応じて高々2^{32} 個とか 2^{64} 個とか言うように限られたものになる. そのときには表現誤差が含まれた形になる. 以下に 32 ビット語の場合の代表的な 2 つの例を挙げる.

IBM 方式

IBM 方式とは図 1 の概念図のような形で数が表現されている表現方式である.

IBM 方式は (1) 式において $\beta=16$, $m=6$ とし, 丸め[7)]を切り捨て方式にしている. この形で表現できる数は, 絶対値で約 $16^{-64}\sim 16^{63}$ の範囲である[8)]. 10 進表示にすると約 $0.86\times10^{-77}\sim 0.72\times10^{76}$ である. この方式での表現の相対誤差は

[5)] 16 進数は 2 進数の 4 桁を一つにまとめたものなので実質は 2 進数である.
[6)] ビットはコンピュータの最小単位で 2 進法の 1 桁のこと.
[7)] 丸めとは切り捨てなどの端数処理のこと.
[8)] 16^{-64} は 0 と表現している.



○ $f_1 \neq 0$; 各 f_i は 4 ビットで 0～F のいずれか（章末の Smile 1 参照）
○指数部 7 ビットを用いて 0～127 を表せるが，これを $E=-64 \sim +63$ に対応させる．
○“0”はこの表現には馴染まない異質な数である．実際には，たとえば，$64+E=0$（すなわち $E=-64$）をそれに当てる．

図 1-1　数の内部表現の概念図（IBM 方式）

図 1: IBM 方式の数の内部表現の概念図 (伊理正夫, 1985:数値計算の常識より)

$(f_1)_{16} = \cdots = (f_6)_{16} = (\mathrm{F})_{16}$ のとき最も小さくなる[9]. (22) 式より

$$\delta_r \approx \frac{16^{-6}}{15 \cdot 16^{-1}} \tag{32}$$

$$= 16^{-6} \tag{33}$$

$$\approx 6 \times 10^{-8} \tag{34}$$

となる. また,$(f_1)_{16} = (1)_{16}$,$(f_2)_{16} = \cdots = (f_6)_{16} = 0$ のとき最も大きくなる. 同様にやると,

$$\delta_r \approx \frac{16^{-6}}{1 \cdot 16^{-1}} \tag{35}$$

$$= 16^{-5} \tag{36}$$

$$\approx 10^{-6} \tag{37}$$

となる.

IEEE 方式 (マイクロソフト社製 BASIC 等)

マイクロソフト社製 BASIC 等の方式は IEEE 方式と呼ばれる表現方式である. この方式は図 2 の概念図のような形で数が表現される.

IEEE 方式は (1) 式において $\beta = 2$,$m = 24$ とし, 丸めを四捨五入 (2 進法なので 0 捨 1 入) とする. 2 進法で表現されたことで $(f_1)_\beta \neq 0$ の条件から $(f_1)_2$ は自動的に $(1)_2$ に決まる. そのため $(f_1)_2$ には情報がないことになり省略できるなどの利点がある. この形で表現できる数は, 絶対値で約 $2^{-128} \sim 2^{127}$ の範囲である[10]. 10 進

[9] 16 進法では慣習で $0, 1, \cdots, 9$ の他に $10, 11, 12, 13, 14, 15$ に相当するもとのとして A,B,C,D,E,F を使う.
[10] 2^{-128} は 0 を表現している.

○f_1=1は明示せず；各f_iは0または1．
○指数部8ビットを用いて0～255を表せるが，これをE=−128～127に対応させる．
○仮数部の符号ビットは“+”のとき0，“−”のとき1；符号が“−”のときは仮数部は“補数”表示とすることもある（ここでの話には関係ないが）．
○“0”はこの表現には馴染まない異質な数である．実際には，たとえば，128+E=0（すなわちE=−128）をそれに当てる．

図 1-2　数の内部表現の概念図（マイクロソフト社製BASIC等）

図 2: IEEE 方式 (マイクロソフト社製 BASIC 等) の数の内部表現の概念図 (伊理正夫, 1985:数値計算の常識より)

表示にすると約 $2.9 \times 10^{-39} \sim 1.7 \times 10^{38}$ となる. この方式での表現の相対誤差は $(f_1)_2 = (f_2)_2 = \cdots = (f_{24})_2 = (1)_2$ のとき最小になる. (30) 式より

$$\begin{aligned} \delta_r &\approx \frac{\frac{2^{-24}}{2}}{\frac{2}{2}} \\ &= 2^{-25} \\ &\approx 3 \times 10^{-8} \end{aligned} \tag{38}$$

最大になるのは $(f_1)_2 = (1)_2$,$(f_2)_2 = \cdots = (f_{24})_2 = 0$ のときで最小の時と同様に求めると

$$\begin{aligned} \delta_r &\approx \frac{\frac{2^{-24}}{2}}{\frac{1}{2}} \\ &= 2^{-24} \\ &\approx 6 \times 10^{-8} \end{aligned} \tag{39}$$

である. 表現の相対誤差がほぼ一定であるのが 16 進法に比べて著しい長所の一つである.

GFD ワークノート「実数の浮動小数点表現の誤差その 1 」の例題の解法について考察する.

例 1 の解法

例 1 では, 大型計算機の FORTRAN で計算を行うと, 0.09999996 となった. 例 1 の誤差は $(0.1)_{10}$ の IBM 方式で表示したときと IEEE 方式で表示したときを見比べる

ことで理解できる. (12) 式で小数点以下 7 桁目を切り捨てる.

$$(0.1)_{10} = (0.199999)_{16} \times 16^0. \tag{40}$$

同様に (13) 式で小数点以下 25 桁目を 0 捨 1 入すると,

$$(0.1)_{10} = (0.11001100110011001100110 1)_2 \times 2^{-3} \quad = (0.\mathrm{CCCCCD})_{16} \times 2^{-3} \tag{41}$$

となる. これらをそれぞれ 10 進数に戻す. IBM 方式の方は (15) 式より

$$\begin{aligned}(0.199999)_{16} &= (199999)_{16} \times 16^{-6} \\ &= (((((1 \times 16 + 9) \times 16 + 9) \times 16 + 9) \times 16 + 9) \times 16 + 9) \times 16^{-6} \\ &= \frac{1677721}{16777216} \\ &\approx (0.09999996424)_{10}\end{aligned} \tag{42}$$

となって大型計算機で計算した値と一致する. 同様に IEEE 方式も計算する. 式 (13) を用いると,

$$\begin{aligned}(0.\mathrm{CCCCCD})_{16} \times 2^{-3} &= (\mathrm{CCCCCD})_{16} \times 2^{-27} \\ &= (((((12 \times 16 + 12) \times 16 + 12) \times 16 + 12) \times 16 + 13) \times 2^{-27} \\ &= 13421773 \times 7.450580597 \times 10^{-9} \\ &\approx (0.1000000015)_{10}\end{aligned} \tag{43}$$

となる.

例 2 の解法

0.01 を 10000 回足すプログラムを行うと, パソコン BASIC では 100.003, 大型計算機 FORTRAN では 99.95277 という問題がある. この計算で起きる相対誤差を見積もってみる. $\sum_{n=1}^{10000} 0.01$ の計算において n 項目までの部分和の大きさが $0.01n$ であり, ε の相対誤差が毎回生じたとすると

$$\begin{aligned}\sum_{n=1}^{10000} 0.01n\varepsilon &= 0.01\varepsilon \frac{(10000)(10000+1)}{2} \\ &\cong 0.01 \times \frac{(10000^2\varepsilon)}{2} \\ &= 5 \times 10^5 \varepsilon\end{aligned} \tag{44}$$

ほどの誤差が累積する. IBM 方式では $\varepsilon = 6 \times 10^{-8} \sim 10^{-6}$ の間なので, この値は $0.03 \sim 0.5$. IEEE 方式では $\varepsilon = 3 \times 10^{-8} \sim 6 \times 10^{-8}$ として, この値は $0.015 \sim 0.03$

ほどとなる. よって, IBM 方式では $100 - 0.5 = 99.95$,IEEE 方式では $100 + 0.03 = 100.003$ で例 2 の結果とほぼ一致する[11]).

例 3 の解法

0.0 から 1.0 までの x^2 を足し合わせるプログラムにおいて, パソコン BASIC では 2.85, 大型計算機 FORTRAN では 3.85 という結果が出た. BASIC で最後まで足されなかった理由を考察する. IBM 方式の場合に $0.1 = (0.199999)_{16} \times 16^0$ を 10 個足すと

$$
\begin{array}{rr}
 & 0.199999 \\
+ & 0.199999 \\
\hline
 & 0.333332 \\
+ & 0.199999 \\
\hline
 & 0.4CCCCB \\
+ & 0.199999 \\
\hline
 & 0.666664 \\
+ & 0.199999 \\
\hline
 & 0.7FFFFD \\
+ & 0.199999 \\
\hline
 & 0.999996 \\
+ & 0.199999 \\
\hline
 & 0.B3332F \\
+ & 0.199999 \\
\hline
 & 0.CCCCC8 \\
+ & 0.199999 \\
\hline
 & 0.E66661 \\
+ & 0.199999 \\
\hline
 & 0.FFFFFA
\end{array}
$$

となり, 例 1 と同様に 10 進法表示にすると

$$
\begin{aligned}
(0.\mathrm{FFFFFA})_{16} &= (\mathrm{FFFFFA})_{16} \times 16^{-6} \\
&= (((((15 \times 16 + 15) \times 16 + 15) \times 16 + 15) \times 16 + 15) \times 16 + 10) \times 16^{-6} \\
&\approx (0.9999996424)_{10} \qquad (45)
\end{aligned}
$$

[11]) パソコン BASIC では IBM 方式を使用していて、大型計算機 FORTRAN では IEEE 方式を使用しているような書き方になっているが、実際浮動小数点をどのように扱っているのかは不明. コンパイラ、OS、CPU のどれかが扱い方を決めている可能性があるが、詳しくはよくわからない.

となり 1 より小さいので while の条件は満たされている.

同様に IEEE 方式でも同様の計算を行う. 各計算で浮動小数点表示の 25 桁目を 0 捨 1 入する.

1 回目

$$\begin{array}{rr} & 0.11001100110011001100 1101 \\ + & 0.11001100110011001100 1101 \\ \hline & 1.10011001100110011001 1010 \end{array}$$

答えは 1 桁増えたので最後の 0 を削る, また, その値に合わせて足すほうも削る.

2 回目

$$\begin{array}{rr} & 1.10011001100110011001 101\not{0} \\ + & 0.11001100110011001100 1\not{0} \\ \hline & 10.01100110011001100110 011 \end{array}$$

同様に 1 桁増えたので最後を繰り上げる. その値に合わせて足すほうも削る.

$$\begin{array}{rr} & 10.01100110011001100110 \not{0}_1\ \not{1}_0\not{1} \\ + & 0.11001100110011001100\ 1\quad 1\not{0} \\ \hline & 11.00110011001100110011\ 0\quad 1 \end{array}$$

桁が増えなかったのでこのままの答えを使って計算する.

3 回目

$$\begin{array}{rr} & 11.00110011001100110011 101 \\ + & 0.11001100110011001100 11 \\ \hline & 100.00000000000000000000 0 \end{array}$$

答えは 1 桁増えたので最後を削る. また, 足すほうも最後を繰り上げる.

4 回目

$$
\begin{array}{rr}
 & 100.00000000000000000000\ 0\not{0} \\
+ & 0.11001100110011001100\not{1}\not{1}\not{0}\not{1} \\
\hline
 & 100.11001100110011001101\ 0
\end{array}
$$

答えは最後まで繰り上がらないのでこれ以降は続けて書く.

$$
\begin{array}{rr}
 & 100.11001100110011001010 \\
+ & 0.11001100110011001010 \\
\hline
 & 101.10011001100110011001 00 \\
+ & 0.11001100110011001010 \\
\hline
 & 110.01100110011001100110 \\
+ & 0.11001100110011001010 \\
\hline
 & 111.00110011001100110100 \\
+ & 0.11001100110011001010 \\
\hline
 & 1000.00000000000000000000010
\end{array}
$$

となる. ここで小数点以下 25 桁以上は 0 捨 1 入すると 0.10000000000000000000000001 となりこれを 10 進数に直すと,

$$
\begin{aligned}
(0.10000000000000000000000001)_2 \times 2^1 &= 1 \times 2^1 \times 2^{-1} + 1 \times 2^{-27} \times 2^1 && (46) \\
&\approx (1.000000119)_{10} && (47)
\end{aligned}
$$

となり while の条件が満たされなくなるため, 計算が途中で終わっている.

参考文献

伊理正夫, 藤野和建, 1985:数値計算の常識, 共立出版株式会社, pp174, ISBN 4-320-01343-3