実数の浮動小数点表現と誤差

1　浮動小数点

1.1　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[1]. 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

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

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

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

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. 具体的に $(x)_{10}=(27)_{10}$ として 2 進表示してみる.

[1] $()_\beta$ 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.

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

なので,

$$(27)_{10} = (11011)_2$$

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

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

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

$$(0001)_2 = (1)_{16}$$

となる. よって

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

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

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

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 $(0.1)_{10} = (b_1)_{10}16^{-1} + (b_2)_{10}16^{-2} + \cdots$ の式に 16 をかけると

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

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

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

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

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

したがって

$$(0.1)_{10} = (0.19999\cdots)_{16} \tag{3}$$

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

$$\begin{aligned}(0.1)_{10} &= (0.000110011001100\cdots)_2 \\ &= (0.110011001100\cdots)_2 \times 2^{-3} \\ &= (0.CCC\cdots)_{16} \times 2^{-3}\end{aligned} \tag{4}$$

となる.

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

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

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.

$$\begin{aligned}(a_k a_{k-1}\cdots a_0)_\beta = \{\cdots\{\{(a_k)_{10}\cdot(\beta)_{10} + (a_{k-1})_{10}\}\cdot(\beta)_{10} + (a_{k-2})_{10}\}\cdot(\beta)_{10} \\ + \cdots\}\cdot(\beta)_{10} + (a_1)_{10}\}\cdot(\beta)_{10} + (a_0)_{10}.\end{aligned} \tag{5}$$

β 進表示された純小数の場合は

$$(0.b_1b_2\cdots b_m)_\beta = (b_1b_2\cdots b_m)_\beta \times {(\beta)_{10}}^{-m}$$

とすれば整数のときと同様に計算できる.

2 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.

2.1 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, 切り捨てたられた表現誤差を δ_1 とすると,

$$\delta_1 = z - F$$

となる.z と F を浮動小数点で表示すると,

$$\delta_1 = (0.f_1 f_2 \cdots)_\beta \times (\beta)_{10}{}^{(E)_{10}} - (0.f_1 f_2 \cdots f_m)_\beta \times (\beta)_{10}{}^{(E)_{10}}$$

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

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

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

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

δ_1 として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, $(f_{m+1})_{10} = (\beta)_{10} - 1$. このとき

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

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

$$\begin{aligned}
\delta_{1r} &= \frac{\Big((\beta)_{10}{}^{-m} - (\beta)_{10}{}^{-(m+1)} \Big) \times (\beta)_{10}{}^{(E)_{10}}}{\big((f_1)_{10}(\beta)_{10}{}^{-1} \cdots + (f_m)_{10}\beta_{10}{}^{-m} \big) \times (\beta)_{10}{}^{(E)_{10}}} \\
&\approx \frac{(\beta)_{10}{}^{-m}}{(f_1)_{10}(\beta)_{10}{}^{-1}}. \qquad (6)
\end{aligned}$$

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,

$$\begin{aligned}
\delta_{1r} &\leq \frac{(\beta)_{10}{}^{-m}}{(\beta)_{10}{}^{-1}} \\
&= (\beta)_{10}{}^{-(m-1)} \qquad (7)
\end{aligned}$$

となる.

2.2 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[2]. 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 δ_{ro} とすると,

$$\begin{aligned}\delta_{ro} &\equiv (f_{m+1})_{10}(\beta)_{10}{}^{-(m+1)}\times(\beta)_{10}{}^{(E)_{10}}+\cdots\\ &=\begin{cases}(\beta)_{10}{}^{-m} & \left((f_{m+1})_{10}\geq\frac{(\beta)_{10}}{2}\right)\\ 0 & \left((f_{m+1})_{10}\leq\frac{(\beta)_{10}}{2}-1\right)\end{cases}\end{aligned}$$

である. ここで四捨五入による表現誤差を δ_2 とすると

$$\delta_2 = z-(F+\delta_{ro})$$

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,

$$\begin{aligned}\delta_2 &= (0.f_1f_2\cdots)_\beta\times(\beta)_{10}{}^{(E)_{10}}-((0.f_1f_2\cdots f_m)_\beta\times(\beta)_{10}{}^{(E)_{10}}+\delta_{ro}\\ &=\Big((f_1)_{10}(\beta)_{10}{}^{-1}+\cdots+(f_m)_{10}(\beta)_{10}{}^{-m}+(f_{m+1})_{10}(\beta)_{10}{}^{-(m+1)}+\cdots\Big)\times(\beta)_{10}{}^{(E)_{10}}\\ &\quad-\big((f_1)_{10}(\beta)_{10}{}^{-1}+\cdots+(f_m)_{10}(\beta)_{10}{}^{-m}\big)\times(\beta)_{10}{}^{(E)_{10}}-\delta_{ro}.\end{aligned}$$

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

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

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

$$\delta=\frac{\beta_{10}^{-m}}{2}\times\beta_{10}{}^{E_{10}}$$

となる. $(f_{m+1})_{10}$ の値が $\dfrac{(\beta)_{10}}{2}-1$ 以下の時は繰り上がらずそのときの $(f_{m+1})_{10}(\beta)_{10}{}^{-(m+1)}$ 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 $f_{m+1}=\dfrac{\beta}{2}$ のときであ

[2] 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.

る. 打切り誤差の時と同様に相対誤差 δ_{2r} も求めると

$$\delta_{2r} = \frac{\frac{\beta_{10}^{-m}}{2}\times{\beta_{10}}^{E_{10}}}{(f_1)_\beta{\beta_{10}}^{-1}\cdots+(f_m)_\beta{\beta_{10}}^{-m})\times{\beta_{10}}^{E_{10}}} \approx \frac{\frac{\beta_{10}^{-m}}{2}}{(f_1)_\beta{\beta_{10}}^{-1}} \tag{8}$$

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,

$$\delta_{2r} \leq \frac{\frac{\beta_{10}^{-m}}{2}}{{\beta_{10}}^{-1}} = \frac{\beta_{10}^{-(m-1)}}{2}. \tag{9}$$

2.3 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[3]. 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[4],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例を挙げる.

IBM 方式

IBM 方式とは図 2.1 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[5]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 $16^{-64} \sim 16^{63}$ 範囲である[6]. 10 進表示にすると約 $0.86 \times 10^{-77} \sim 0.72 \times 10^{76}$ である. この方式での表現の相対誤差は $(f_1)_{16} = \cdots = (f_6)_{16} = (F)_{16}$ のとき最も小さくなる[7]. (6) 式より

$$\delta_r \approx \frac{16^{-6}}{15\cdot 16^{-1}} = 16^{-6} \approx 6\times 10^{-8}$$

[3] 16 進法は 2 進法の 4 桁を一つにまとめたものなので実質は 2 進法である.
[4] 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.
[5] 丸めとは切り捨てなどの端数処理のこと.
[6] 16^{-64} は 0 と表現している.
[7] 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.

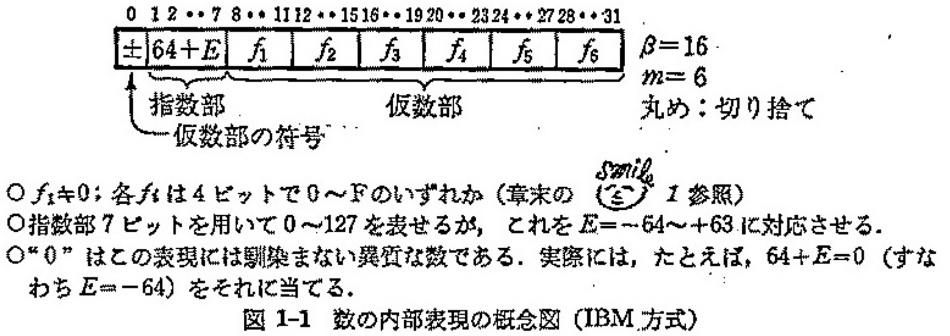

図 2.1: IBM 方式の数の内部表現の概念図 (伊理正夫, 1985:数値計算の常識より)

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,

$$
\begin{aligned}
\delta_r &\approx \frac{16^{-6}}{1 \cdot 16^{-1}} \\
&= 16^{-5} \\
&\approx 10^{-6}
\end{aligned}
$$

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 2.2 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[8]. 10 進表示にすると約 $2.9 \times 10^{-39} \sim 1.7 \times 10^{38}$ となる. この方式での表現の相対誤差は $(f_1 =)(f_2)_2 = \cdots = (f_{24})_2 = (1)_2$ のとき最小になる. (8) 式より

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

8) 2^{-128} は 0 を表現している.

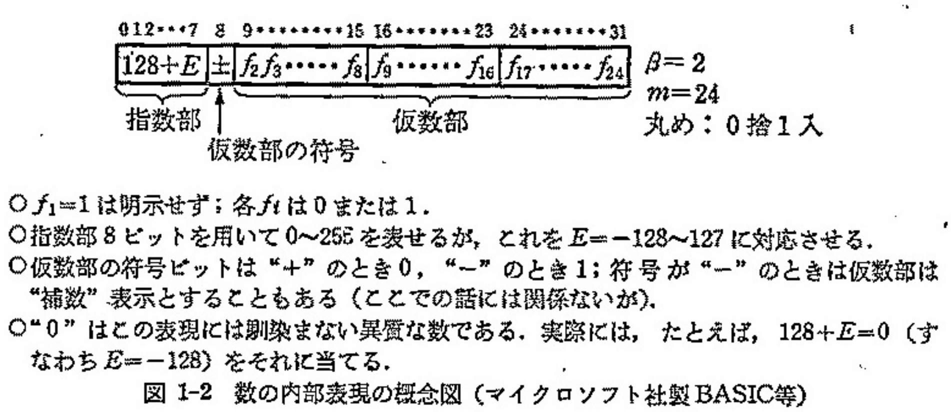

図 2.2: IEEE 方式 (マイクロソフト社製 BASIC 等) の数の内部表現の概念図 (伊理正夫, 1985:数値計算の常識より)

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

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

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.

例 1 の解法

例 1 の誤差は 10 進数と β 進数の相互変換:純小数 出した $(0.1)_{10}$ の 16 進数と 2 進数で IBM 方式のときと IEEE 方式のときになるようにすることで理解できる. (3) 式で小数点以下 7 桁目を切り捨てる.

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

同様に (4) 式で小数点以下 25 桁目を 0 捨 1 入すると,

$$(0.1)_{10} = (0.110011001100110011001101)_2 \times 2^{-3} \quad = (0.CCCCCD)_{16} \times 2^{-3}$$

となる. これらをそれぞれ10進数に戻す. IBM方式の方は(5)式より

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

となって例1の値になる. IEEE方式も同様に

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

となる.

例2の解法

$\displaystyle\sum_{n=1}^{10000} 0.01$ 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

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

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.

例3の解法

IBM方式の場合に $0.1 = (0.199999)_{16} \times 16^0$ を10個足すと

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

となり, 例 1 と同様に 10 進法表示にすると

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

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. 各計算で桁を浮動小数点表示の 25 桁目を 0 捨 1 入する.

1 回目

$$
\begin{array}{cr}
 & 0.110011001100110011001101 \\
+ & 0.110011001100110011001101 \\ \hline
 & 1.100110011001100110011010
\end{array}
$$

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

$$
\begin{array}{lr}
 & 1.10011001100110011001101\cancel{0} \\
+ & 0.11001100110011001100110\cancel{0} \\
\hline
 & 10.0110011001100110011001
\end{array}
$$

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.

$$
\begin{array}{lr}
 & 10.011001100110011001100110\,\cancel{0}_1\ \cancel{1}_0\cancel{1} \\
+ & 0.1100110011001100110011001100\ 1\ \ \ 1\,\cancel{0} \\
\hline
 & 11.00110011001100110011\ 0\ \ \ 1
\end{array}
$$

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

$$
\begin{array}{lr}
 & 11.0011001100110011001101 \\
+ & 0.1100110011001100110011 \\
\hline
 & 100.0000000000000000000000
\end{array}
$$

答えは 1 桁増えたので最期を削る. また, 足すほうも最期を繰り上げる.

4 回目

$$
\begin{array}{lr}
 & 100.000000000000000000000\ \ 0\ \cancel{0} \\
+ & 0.110011001100110011001100\cancel{0}_1\cancel{1}_0\cancel{1} \\
\hline
 & 100.1100110011001100110 1\ \ 0
\end{array}
$$

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.

$$
\begin{array}{lr}
 & 100.1100110011001100110010 \\
+ & 0.1100110011001100110010 \\
\hline
 & 101.1001100110011001100100 \\
+ & 0.1100110011001100110100 \\
\hline
 & 110.0110011001100110011100 \\
+ & 0.1100110011001100110100 \\
\hline
 & 111.0011001100110011010000 \\
+ & 0.1100110011001100110100 \\
\hline
 & 1000.0000000000000000000010
\end{array}
$$

となる. ここで小数点以下 25 桁以上は 0 捨 1 入すると 0.100000000000000000000001 となりこれを 10 進に直すと,

$$
\begin{aligned}
(0.100000000000000000000001)_2 \times 2^1 &= 1 \times 2^1 \times 2^{-1} + 1 \times 2^{-27} \times 2^1 \\
&\approx (1.000000119)_{10}
\end{aligned}
$$

となり while の条件が満たされなくなる.