二重指数分割に基づくデータ長独立実数値表現法 II†

浜田穂積††

実数値の2進表現として，先に次の特徴を有する表現法を提案した．(i)形式はデータの長さに依存せず精度変換操作が単純，(ii)オーバフロー，アンダフローが発生せず，十分大きい数も十分小さい数も表現可能，(iii)固定小数点表現と比べて，分解能の点で1ビットの不利にすぎない．しかしながら，2進数としての指数の値の2進表現と本表現の指数部とが完全には対応せず1だけずれていて論理を少し複雑にしていたのを修正した．これによっても，上記の特徴は保たれることがわかった．丸めについてはIEEE標準案での提案のすべてを容易に満たすことができる．非数については，IEEE標準案と松井・伊理の提案したもののうち，0, +0, −0, ∞, +∞, −∞ は2進による表現パタンの極限的なものを，表現しようとする数値との自然な連続として定義することができる．これら非数をも含めた数の四則演算を，対称性をよく保って定義することができる．ただし，松井・伊理の定義した？，＋？，－？については特別な数値ではなく，表現の詳しさに関する情報であって，本表現の意味となじまないため定義しないほうが望ましいと考え，該当の場合は有意性なしの例外処理を期待する．データの特性パラメータが本表現によれば長さだけであるので，プログラミング言語での指定も容易となる．この場合の具体的提案も行った．

1. まえがき

実数値の計算機内部における表現形式として，先に二重指数分割に基づくデータ長独立実数値表現法[5]（以後たんに前論文というときこれを指すことにする）を提案した．これはIEEE標準案[2),3)]を不満とし，オーバフロー，アンダフロー問題の解消を意図した松井・伊理による表現法[4]に，実用的に最も重要なデータ長変換を容易にする特性を付した，という文脈の上にある．前論文では，一般的形式と標準としての提案方式とを述べた．この提案方式による計算論理が若干複雑になる点を改良する方式を，一般的形式のなかに見いだしたので，新たに標準方式として提案する．

また，IEEE標準案で概念として提案された非数と松井・伊理によって規定された非数を含む演算に関しては，前論文で言及しなかったので，これらの点について述べる．

2. 変更された実数値表現法の定義

2.1 設定条件

ここで述べる実数値表現法を設定するにあたっての条件は前論文のそれと本質的に同じであるが，念のためここに再録する．まず次の3条件を必須とする．

条件1：2進数としての指数，仮数から本表現へ，またその逆の変換が容易であること．このことは対数関数など，複雑な処理を用いないことを意味する．

条件2：オーバフロー，アンダフローが事実上起こらないこと．

条件3：表現仕様がデータの長さに依存せず，長さの異なるデータ間の相互変換が容易であること．

次の3条件は可能なら満たしたいものである．

条件4：形式の取決め事項が不要，あるいはなるべく少ないこと．取決めのためのパラメータは少ないほどよく，ある一つの形式に自然に落ち着くものでありたい．

条件5：固定長としたとき，すべてのパタンが異なる数値に対応すること．

条件6：値の大小関係が，固定小数点と考えた場合のそれと一致すること．

2.2 慣例的定義

表そうとする数をxとする．これを

$$x=2^e\times f \qquad (1)$$

と，二つの数eとfで表現する．ここで値を一意的にするため，e, f に次の条件を設ける．まず $x>0$ の場合にかぎって考える．

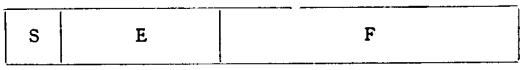
$$e: \text{整数} \qquad (2)$$

$$1\leq f<2 \qquad (3)$$

このとき条件1，3を満たすために，次の三つのデータをこの順に置く．この点に関するかぎり，多くの

† Data Length Independent Real Number Representation Based on Double Exponential Cut II by HOZUMI HAMADA (Central Research Laboratory, Hitachi Ltd.).

†† (株)日立製作所中央研究所



S : sign bit
E : exponent part
F : fraction part

図 1 新実数値表現法の構成
Fig. 1 Construction of new real number representation.

浮動小数点表現と同じものである．これを**図1**に示す．

（i） 符号部（数xの符号を表す1ビット）
（ii） 指数部
（iii） 仮数部

仮数 f の値の2進数表現を次のとおりとする．

$$f=1.f_1f_2\cdots f_j\cdots \tag{4}$$

このとき $f_1f_2\cdots f_j\cdots$ を仮数部のビット・パタンとする．条件2を満たすためには指数部を可変長にせざるをえない．可変長のデータ・フィールドの長さに自己記述能力をもたせる次の方法を用いる．なお以後二重指数表現を多用するので，印刷上の誤まりの発生を防ぐため，次の表現を用いることにする．

$$2^n \Rightarrow \exp(n) \tag{5}$$

まず $e>0$ のとき，e がちょうど2進 $m(>0)$ 桁で表せる範囲は次のとおりである．

$$2^{m-1}\leq e\leq 2^m-1 \tag{6}$$

これに(3)の f の範囲も含めて，xの範囲で表せば

$$\exp(2^{m-1})\leq x<\exp(2^m) \tag{7}$$

である．$e<0$ のときは(7)との対称性を尊重して

$$\exp(-2^m)\leq x<\exp(-2^{m-1}) \tag{8}$$

とすべきであり，これを e の範囲に戻せば

$$-2^m\leq e<-2^{m-1}-1 \tag{9}$$

となる．(9)から，e の表現を2の補数表現と考えるのが合理的である．したがって仮数 f についても $x<0$ のとき

$$-2\leq f<1 \tag{3}'$$

とするのが合理的である．(6)，(9)には $e=0,-1$ の場合が含まれていない．これは $m=0$ と解釈する．これらを含めて，e がちょうど2進 m 桁で表される場合の整数 e の内部表現を次のとおりとする．

$$\left.\begin{array}{ll} e\geq 0 \text{ のとき} & 0\cdots 01e_{m-1}\cdots e_2e_1 \\ e<0 \text{ のとき} & 1\cdots 10e_{m-1}\cdots e_2e_1 \end{array}\right\} \tag{10}$$

ここで定めた m と $e_{m-1}\cdots e_2e_1$ とから，m の識別ビット列を左に追加して指数部を次のとおりに構成する．

$$\left.\begin{array}{ll} e\geq 0 \text{ のとき} & \overbrace{1\cdots 10}^{m+1\text{個}}\ e_{m-1}\cdots e_2e_1 \\ e<0 \text{ のとき} & 0\cdots 01\ e_{m-1}\cdots e_2e_1 \end{array}\right\} \tag{11}$$

(11)のビット列の符号のない固定小数点2進数としての大小順は，e の大小順に一致する．x の大小順は，仮数部を含めて考えても e の大小順に一致するから，条件6を満たす．

$x<0$ のときも上とほぼ同様で，(7)の代りに

$$-\exp(2^m)\leq x<-\exp(2^{m-1}) \tag{7}'$$

とし，また(8)の代りに次のとおりとする．

$$-\exp(-2^{m-1})\leq x<-\exp(-2^m) \tag{8}'$$

$x<0$ では xの順と e の順が逆であることを考慮して(11)の順を逆にすればよいが，これは1の補数をとることによって得られるから，(11)の代りに

$$\left.\begin{array}{ll} e\geq 0 \text{ のとき} & \overbrace{0\cdots 01}^{m+1\text{個}}\ \overline{e_{m-1}}\cdots\overline{e_2}\overline{e_1} \\ e<0 \text{ のとき} & 1\cdots 10\ \overline{e_{m-1}}\cdots\overline{e_2}\overline{e_1} \end{array}\right\} \tag{11}'$$

とする．ここで上線は各ビットの0と1の反転操作を表す．

ここで規定したものと，前論文におけるものとの内容上の相違は，(10)と(11)との関係が，前論文ではこのように直接的なビット操作で得られない点にある．

2.3 形式的定義

前節の慣例的定義法では，条件 3, 4, 5, 6 が満たされているか否かが直観的に理解しにくい．そこでこれと同じ内容を以下に示す区間分割に基づく方法で形式的に定義する．この定義法は前論文で用いたものと同じものである．本表現法の定義として，数学的にはこちらを正式とし，前節のものは理解の便のためにのみ用いるものとする．

まず一般論として次の事項から規定する．

任意のビット列 S は実数値の一つの区間に対応するものとする．すなわち

$$S:\ \{x\,|\,a\leq x<b\}$$

これを次の記法を用いて示す．

$$I(S)=[a,b) \tag{12}$$

S が正しく表すと考える値は区間の下限 a であるものとする．S の右にビット0を連結したものを $S0$，ビット1を連結したものを $S1$ と記す．このとき，S の形および a,b の値によって定まる第3の値 c によって区間 $I(S)$ が2分されて，次の対応づけが行われるものとする．

$$I(S0)=[a,b),\quad I(S1)=[c,b) \tag{13}$$

このことが，条件3, 4, 5, 6を満たすことを示している．以下の記述で用いる次の表現を定義する．

0^k: k 個の0の連結

1^k も同様である．

さて，区間の分割を次の4段階の手順で行う．

（Ⅰ） 疎分割

$$\left.\begin{aligned} I(1)&=[-\infty, 0)\\ I(0)&=[0, +\infty)\end{aligned}\right\} \tag{14}$$

から出発する．これらをそれぞれ $-1, 1$ で分割して次を得る．

$$\left.\begin{aligned} I(10)&=[-\infty, -1)\\ I(11)&=[-1, 0)\\ I(00)&=[0, 1)\\ I(01)&=[1, +\infty)\end{aligned}\right\} \tag{15}$$

さらにこれらをそれぞれ -2, -0.5, 0.5, 2 で分割して次を得る．

$$\left.\begin{aligned} I(100)&=[-\infty, -2)\\ I(101)&=[-2, -1)\\ I(110)&=[-1, -0.5)\\ I(111)&=[-0.5, 0)\\ I(000)&=[0, 0.5)\\ I(001)&=[0.5, 1)\\ I(010)&=[1, 2)\\ I(011)&=[2, +\infty)\end{aligned}\right\} \tag{16}$$

(16)の8式のうち，ビット列の第2ビットから右に見て，0あるいは1の連が止まっている，第2, 3, 6, 7の4例ではただちに(Ⅳ)の等差分割に進むが，その他の4例の場合は次の二重指数分割を行う．

（Ⅱ） 二重指数分割

ここでは $m>0$ の場合について帰納的に

$I(10^{m+1})=[-\infty, -\exp(2^{m-1}))$ を $-\exp(2^m)$ で

$I(11^{m+1})=[-\exp(-2^{m-1}), 0)$ を $-\exp(-2^m)$ で

$I(00^{m+1})=[0, \exp(-2^{m-1}))$ を $\exp(-2^m)$ で

$I(01^{m+1})=[\exp(2^{m-1}), +\infty)$ を $\exp(2^m)$ で

分割する．これにより，0あるいは1の連の止まっている次のものを得る．

$$\left.\begin{aligned} I(10^{m+1}1)&=[-\exp(2^m), -\exp(2^{m-1}))\\ I(11^{m+1}0)&=[-\exp(-2^{m-1}), -\exp(-2^m))\\ I(00^{m+1}1)&=[\exp(-2^m), \exp(-2^{m-1}))\\ I(01^{m+1}0)&=[\exp(2^{m-1}), \exp(2^m))\end{aligned}\right\} \tag{17}$$

（Ⅲ） 等比分割

(17)の四つの場合について，

$$c=\pm\sqrt{ab}\ \text{（符号は}a\text{のそれと同じ）} \tag{18}$$

で分割する操作を $m-1$ 回行う．これにより区間の両端の比は2となる．

（Ⅳ） 等差分割

区間の両端の値の比が2であるものについて，

$$c=(a+b)/2 \tag{19}$$

による分割を任意回行う．

以上の分割法は，ビット列と区間との対応関係の生成法であって，所望の長さのビット列が得られれば，必ずしもすべての手順を完了しなくてもよく，途中で打ち切ってよい．

ここで規定した定義によって得られるビット列と，従来の表現における符号部，指数部，仮数部との関係はほぼ次のとおりである．

(14)によって得られる1ビット：符号部

(15), (16)と(Ⅱ), (Ⅲ)で新たに得られるビット列：指数部

(Ⅳ)によって新たに得られるビット列：仮数部

3. 表現法に固有の精度に関する評価

多くの浮動小数点表現では，仮数部が数 x の大きさによらず一定の長さをとるので，表現法に固有の精度は相対誤差がほぼ一定という形で述べられる．これを量でいうときは用いられる基数における仮数部の桁数である．仮数部の桁数の変わりうる松井・伊理方式[4]とか本表現法ではより細かく調べてみる必要がある．ここでは誤差として，表現可能な最小単位の半分のものがあるとして，相対誤差 E_r と絶対誤差 E_a について考察する．なお分解能という測度も考えられるが，これは上述の表現可能な最小単位とするのが妥当であり，以下に述べる誤差を2倍したものと考えればよいので，分解能の考察は省略する．

64ビット・データの相対誤差 E_r について，典型的な4種の表現についてのものを**図2**に示す．横軸は表現しようとする値 x の，縦軸は相対誤差 E_r の，それぞれの絶対値の2を底とする対数で示す．図2はかなり複雑に重なりあって見にくいので単純化を試みることにする．同じ形が左右方向に並ぶ場合1本の横線で結んだものと置き換えてみる．このような場合最も不利な条件で考えねばならないので，おのおのの部分の上縁を連ねる．これを行うとほぼ左右対称となる．そこで右半分だけをとって，かつ横軸の尺度のさらに

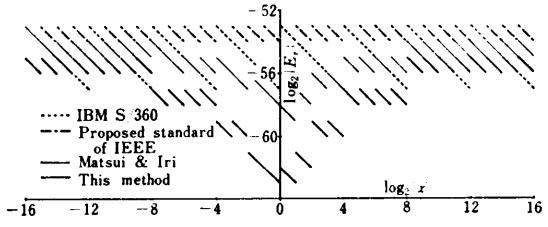


図 2 表現法固有の相対誤差

Fig. 2 Characteristic relative error on several representations.

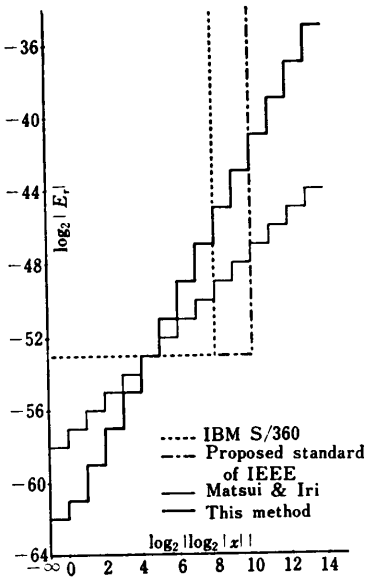


図 3 表現範囲と相対誤差
Fig. 3 Relation between represented value and characteristic relative error.

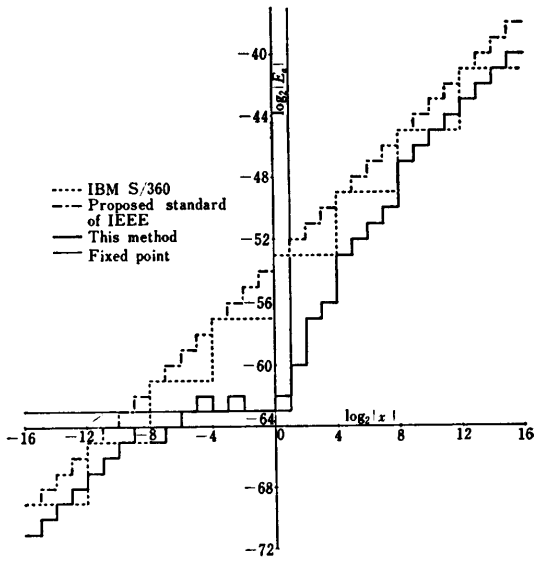


図 4 表現法固有の絶対誤差
Fig. 4 Characteristic absolute error on several representations.

2を底とする対数をとる．こうして得られたのが**図3**である．これは前論文とほぼ同じ傾向である．

次に絶対誤差 E_a を**図4**に示す．なおこれには松井・伊理方式に代えて，64ビットで $-2 \leq x < 2$ の範囲を表す固定小数点表現の場合を追加した．これによると，前論文の場合と同じく本表現法が固定小数点表現と比べて不利であるのは3か所だけで，しかもいずれにおいても1ビットの不利にすぎず，なおかつ浮動小数点表現のもつ長所，すなわち大きい数を表現でき，小さい数はより細かく表現可能という長所をあわせもつことがわかる．このことは別の観点から見るとき，ビット数の小なるデータもかなりの表現能力をもつことを示している．

以上の考察により，本表現は汎用大型計算機での実数値計算のみならず，ミニコンピュータ，マイクロコンピュータにも使え，また広義のアナログ・ディジタル変換におけるディジタル側表現にも適する．これによりすべてのディジタル機器における実数値表現が一つの表現法で統一できるから，これら機器を有機的に結合して，自由な計算処理を行う体系の構築が可能となる．

4. 丸　　め

IEEE 標準案では丸めについて具体的に提案している．本表現法における丸めは本質的にこれに新たな変更を加えるものではない．ただし，本表現法における丸めは，当該の値の表現を固定小数点表現と見なしたものについての丸めと同等の処理をするものとする．このことは次の二つの理由により合理的と考える．

（i） すべてのビット・パタンが異なる値を表す．

（ii） 値の順序関係は固定小数点表現のそれと同じである．

IEEE 標準案でいう丸め法 RN(round to nearest) では距離を考慮しなければならない．上述の固定小数点表現における距離と本表現法における距離は次の三つの場合の丸め位置に分類できる．

（a） 等差分割により得られるビット位置

（b） 等比分割により得られるビット位置

（c） 二重指数分割により得られるビット位置

丸め位置が(a)であるときは従来どおりで問題ない．(b)であるときは，距離は指数的，(c)であるときは二重指数的ノルムで計ることになる．上述の丸め方式は処理の簡単さを重視したものであり，(b),(c)の場合を厳密に線形ノルムで処理すべきと考える場合の複雑さに優先されるべきものと考える．

本表現法は，負数の表現に関して2の補数の系統に属するものである．従来，実数値表現においては，丸め操作の点で正負の対称性のよい符号と絶対値のほうが優れているとする意見が支配的であった[1]と思う．ここでは丸め操作の符号による対称性についての考察

をする．この章の以下の議論において，「切り捨て」とは表現のビット・パタンにおいて丸め位置の右側を無条件に捨てる操作を表し，「切り上げ」とは丸め位置の右がすべて0である場合を除いて丸め位置に1を加えてから切り捨てる操作を表すものとする．

IEEE 標準案で規定する4種の丸め操作について，2の補数系と，符号と絶対値系の違いの評価をする．

(i) RN(round to nearest)

切り上げになるか，切り捨てになるかは，下位に2ビットを追加して最下位ビットを含め3ビットの組合せによって決まる．丸めの結果を偶数とするか，奇数とするかの条件がつく場合は，両系における処理の違いがまったくない．

(ii) RZ(round to zero)

符号と絶対値系では符号によらず切り捨てとする．2の補数系では，正数の場合切り捨て，負数の場合切り上げを行う．この場合の処理の複雑さが，符号と絶対値方式が優れているとする根拠を与えている．しかしながら，誤差解析によれば，ごく特殊な場合を除いて，RN のほうが優れている[6]ことが知られているので RZ における有利さが全体を支配するだけの根拠は薄い．

(iii) RP(round to plus infinity)

符号と絶対値系では，負数は切り捨て，正数は切り上げる．2の補数系では符号によらず切り上げる．

(iv) RM(round to minus infinity)

符号と絶対値系では，正数は切り捨て，負数は切り上げる．2の補数系では符号によらず切り捨てる．

以上の考察のとおり，本質的には両系の非対称性は存在しない．処理の容易さを追求する立場からいえば，RN では両系とも同じ，RZ では符号と絶対値系が有利，RP, RM では2の補数系が有利という結果が得られる．

5. 非　　数

ここではデータの長さを n ビット固定と考える．次の六つの極端な場合を考える．

(i) 10…00 : $-\infty$

(ii) 10…01 : $-\exp(2^{n-3})$

(iii) 11…11 : $-\exp(-2^{n-3})$

(iv) 00…00 : 0

(v) 00…01 : $\exp(-2^{n-3})$

(vi) 01…11 : $\exp(2^{n-3})$

0は(v)において $n\to\infty$ とした極限と考えられ，すべてが0の列で表現することは自然である．(14)の定義はこの点を考慮してなされたものである．

オーバフロー，アンダフローをなくしたいが，ビット・パタンの個数は有限であるから厳密には不可能である．たとえば絶対値の小さい数同士の乗算を行えばより絶対値の小なる結果が得られる…と，いずれは表現不可能な値になろう．そこで表現可能な最も絶対値小のパタンを極限的な値と解釈し，それより絶対値小にはなりえないことにする．これは結局アンダフローと同じであるが，その値が現実の計算ではまず絶対にといっていいほど起こりにくいものであって，たとえ起こったとしてもそういう状態にあることがわかればよいという程度のものであれば，アンダフローと考えなくてもよいであろう．ちなみに $n=32$ としたときの，表現可能な絶対値最小の値は $2^{-2^{29}}$ で，10進数では小数点の後に 1.61×10^{8} 個あまりの0を並べた後に初めて0でない数の出現するような値であって，上述の仮定を十分満たしているといえよう．オーバフローについてもまったく同様である．これらの場合の表現を，通常の解釈ではなく，特別に状態と考え非数と呼ぶ．非数の概念は IEEE 標準案で提案され，松井・伊理方式でも採用されている．両方式におけるこれら非数の表現は，通常の数の表現からの自然な連続というには多少の無理があるが，本表現法では以下のように自然な解釈ができる．すなわち先の(v)を$+0$, (iii)を-0, (iv)を$+\infty$, (ii)を$-\infty$ と定める．こうするとき，(i)は $-\infty$ とするよりたんなる符号のない ∞ として扱うほうが便利である．IEEE標準案でいうaffine mode ではこの ∞ を使わないことになる．

以上をまとめて次のとおりに定義する．

$$\left.\begin{array}{l}10\cdots00:\ \infty\\ 10\cdots01:\ -\infty\\ 11\cdots11:\ -0\end{array}\right\}\qquad\left.\begin{array}{l}00\cdots00:\ 0\\ 00\cdots01:\ +0\\ 01\cdots11:\ +\infty\end{array}\right\}\tag{20}$$

IEEE 標準案ではこの6種のすべてが揃っているわけではないが，この6種は必須であると思う．

最後に，松井・伊理方式で定義した？などについてであるが，これは数値の極限状態ではなく，有効桁数が極端に失われたことを示すためのものである．これについては本表現方式の意図になじまないため，定義しないことにする．この問題の発生する場合の処理については次章で述べる．

丸めによって非数でなかったものが非数になる場合

表 1 加算の結果
Table 1 Results of addition.
(ns: no significance)

1st. op. \ 2nd. op.	+num	−num	+0	0	−0	+∞	∞	−∞
+num	+num	+num 0 −num	+num	+num	+num	+∞	∞	−∞
−num	+num 0 −num	−num	−num	−num	−num	+∞	∞	−∞
+0	+num	−num	+0	+0	ns	+∞	∞	−∞
0	+num	−num	+0	0	−0	+∞	∞	−∞
−0	+num	−num	ns	−0	−0	+∞	∞	−∞
+∞	+∞	+∞	+∞	+∞	+∞	+∞	ns	ns
∞	∞	∞	∞	∞	∞	ns	ns	ns
−∞	−∞	−∞	−∞	−∞	−∞	ns	ns	−∞

表 2 減算の結果
Table 2 Results of subtraction.
(ns: no significance)

1st. op. \ 2nd. op.	+num	−num	+0	0	−0	+∞	∞	−∞
+num	+num 0 −num	+num	+num	+num	+num	−∞	∞	+∞
−num	−num	+num 0 −num	−num	−num	−num	−∞	∞	+∞
+0	−num	+num	ns	+0	+0	−∞	∞	+∞
0	−num	+num	−0	0	+0	−∞	∞	+∞
−0	−num	+num	−0	−0	ns	−∞	∞	+∞
+∞	+∞	+∞	+∞	+∞	+∞	ns	ns	+∞
∞	∞	∞	∞	∞	∞	ns	ns	ns
−∞	−∞	−∞	−∞	−∞	−∞	−∞	ns	ns

表 3 乗算の結果
Table 3 Results of multiplication.
(ns: no significance)

1st. op. \ 2nd. op.	+num	−num	+0	0	−0	+∞	∞	−∞
+num	+num +0 +∞	−num −0 −∞	+0	0	−0	+∞	∞	−∞
−num	−num −0 −∞	+num +0 +∞	−0	0	+0	−∞	∞	+∞
+0	+0	−0	+0	0	−0	ns	ns	ns
0	0	0	0	0	0	ns	ns	ns
−0	−0	+0	−0	0	+0	ns	ns	ns
+∞	+∞	−∞	ns	ns	ns	+∞	∞	−∞
∞	∞	∞	ns	ns	ns	∞	∞	∞
−∞	−∞	+∞	ns	ns	ns	−∞	∞	+∞

表 4 除算の結果
Table 4 Results of division.
(ns: no significance)

1st. op. \ 2nd. op.	+num	−num	+0	0	−0	+∞	∞	−∞
+num	+num +0 +∞	−num −0 −∞	+∞	∞	−∞	+0	0	−0
−num	−num −0 −∞	+num +0 +∞	−∞	∞	+∞	−0	0	+0
+0	+0	−0	ns	ns	ns	+0	0	−0
0	0	0	ns	ns	ns	0	0	0
−0	−0	+0	ns	ns	ns	−0	0	+0
+∞	+∞	−∞	+∞	∞	−∞	ns	ns	ns
∞	∞	∞	∞	∞	∞	ns	ns	ns
−∞	−∞	+∞	−∞	∞	+∞	ns	ns	ns

がある．また逆にデータ長を延長すれば非数が非数でなくなる場合がある．非数に，データの長さに依存しない意味づけを与えようと思えば，これらの現象は不合理である．しかし，ここで述べた非数が，表現のうちで極限的なものに意味を与えたものであるから，上述の現象は，データ長の延長，短縮にあたって，数値とみなした自然な処理を行うことでまったく問題ないと思う．

符号と絶対値系と，2の補数系の優劣の問題を非数の点から考察すると次のとおりである．非数として0，∞ の2種の符号でないものを導入しようとすると，符号と絶対値系は不自然になる．2の補数系は上に示したとおり自然に導入でき，正と負における表現可能な数の集合の対称性をくずさない．したがって，非数として考えるものをこの章で述べたものの範囲に限れば2の補数系のほうが優れている．

6. 非数を含む四則演算

非数として0および∞を導入することによって，四則演算は一応閉じたものとなる．ただし，乗除算で +0，−0，+∞，−∞ が発生するから，それらをすべて含んだ値の間の四則演算を定義しなければならない．これを**表 1〜4** に示す．この表において $+r$ は正の数で非数でないもの，$-r$ も同様とする．松井・伊理によると，+∞ と +0 の積などで +?，+∞ と −0 の積などで −?，+∞ と −∞ の和などで ? が発生するとしている．本表現では先に述べたようにこれらを定義しない．これらの場合においては有意性なしの結果が得られるものとし，表の該当欄には ns(no significant) と記してある．機械による操作では例外処理としたい．

7. プログラミング言語との関係

プログラミング言語で用いる実数データの型，あるいは属性の規定法は言語によって少しずつ異なる．ひと通りの規定法しかない言語の例は Basic, Pascal などである．とくに Basic は実数データのみを扱うため，宣言に用いる言葉もない．これらの言語では実数

データの間の区別はないので，個々の処理系で定める特定のものに固定される．本章ではこの場合を考察の対象外とする．

FORTRAN では REAL, DOUBLE PRECISION の言葉でデータの長さを区別している．これらの関係は後者のデータが前者の2倍の記憶域を用いるというものであって，必ずしも字義通りに2倍の精度を有するわけではない．おのおのの場合に精度と表現可能な値の範囲が何であるかは定まっていない．これらの決定は処理系の作成者に任されている．

FORTRAN に上述の意味での不明確な点のあることの反省は，PL/I, Ada 等で部分的に解決された．PL/I では，少なくとも確保すべき仮数の桁数を FLOAT(d) という形式で指定するが，表現可能な範囲は指定できない．Ada では

digits d **range** l..r (21)

として表現可能範囲の指定を可能とした．しかし，アンダフローの限界付近の規定はできない．

これらの言語の規定方法は，指数部と仮数部のおのおののフィールドが一定の長さで表される従来の表現法に立脚している．本表現法は指数と仮数の情報が可変長であるため多少様子が異なる．しかしながら，本表現法においては，データ長を規定することによりすべてのデータの特性が定まるので，むしろ具合がよい．そこで

$$\mathrm{real}(d) \tag{22}$$

と表現することによって少なくとも d ビットは確保すると定めるという方法がありうる．しかるにこれではあまりに機械の性質を直接的に用いるので望ましくないと考えるなら，パラメータをもう一つ増して

$$\mathrm{real}(p, q) \tag{23}$$

と表現し，$10^{-q} \leq |x| \leq 10^{q}$ の範囲では少なくとも10進 p 桁の精度を確保するという定め方も可能である．

プログラミング言語によってオーバフロー，アンダフローに対する対策を行うことのできるものがある．典型的なものは PL/I で，条件前置語 OVERFLOW, UNDERFLOW によって，これらの条件が発生するか否かの検出をさせたり，やめさせたりできるほか，検出した場合のプログラムによる処理の指定ができる．しかるに本表現法を PL/I に適用する場合を考えればこれらオーバフロー，アンダフローの概念は不要となる．その他にゼロによる除算も同様である．これら非数を結果とする場合をまとめて検出するシステムを設計できるとともに，非数を表すシンボルとしての定数名を定義する必要が生じてくる．

8. あとがき

現行の浮動小数点表現法に対して指摘されている欠点をほぼすべて取り除くことのできる新しい実数値表現法を考案した．本表現法は最初に設定した6条件をすべて満たすことができた．これらを含めて，次の8項目が本表現法の特長である．

(i) 2進数としての指数と仮数との間で容易に相互交換が可能である．とくに，前論文に示したものと比べ，指数の2進パタンを直接表現に対応させうることができるように改良したことが，本論文の新規な点である．

(ii) オーバフロー，アンダフローが事実上発生しない．

(iii) 表現の仕様がデータの長さによらず一様で，そのため，長さの異なる系を相互に簡単に結合できる．

(iv) 形式の取決め事項が不要である．データの特性を定めるパラメータはデータの長さのみである．

(v) データの長さを固定するとき，すべてのパタンは異なる数値に対応し，無駄がない．

(vi) 値の大小関係が，固定小数点数と考えた場合のそれと一致する．

(vii) 固定小数点表現と比べ，絶対誤差はたかだか1ビットの不利でしかない．

(viii) データの延長で，任意の実数値を限りなく近似可能である．

丸めに関しては基本的に IEEE 標準案に従うことが可能であるため特別な提案はない．

非数については，IEEE とは異なり，0と+0, −0, ∞と+∞, −∞を同時に設定できる．これは松井・伊理の提案のなかに含まれるが，データの表現は通常の数との自然な連続の上にある点で実用的には種々のメリットを生じる．また丸めと非数に関して，正負の表現の対称性等を考慮すれば，符号と絶対値を基礎にする表現法より，2の補数を基礎にする表現法のほうが優れている．

非数を含む数の四則演算に関しては，特定の非数の組合せについて，結果に関する情報が十分得られないため，有意性なしの例外処理が望ましい場合が起こる．それ以外の組合せではよい対称性を示している．

本表現を用いる場合の，プログラミング言語上での

精度指定の方法は，前記(iv)により機械独立かつ言語独立な表現法を設定可能である．これに関する具体的な提案をすることができた．

謝辞 本表現法についての議論を通じて，京都大学萩原宏教授，一松信教授，東京農工大学高橋延匡教授，立教大学島内剛一教授，慶応義塾大学高橋秀俊教授から，有益なご助言と激励の言葉をいただいた．ここに衷心より感謝の意を表する．

参考文献

1) 一松 信：新標準浮動小数点体系の提案，情報処理，Vol. 20, No. 9, pp. 793-797 (1979).
2) Kahan, W. and Palmer, J.: On a Proposed Floating-Point Standard, *ACM SIGNUM Newsl.*, Special Issue, pp. 13-21 (Oct. 1979).
3) Stevenson, D. et al.: A Proposed Standard for Binary Floating-Point Arithmetic, Draft 8.0 of IEEE, *Computer*, pp. 51-62 (Mar. 1981).
4) 松井正一，伊理正夫：あふれのない浮動小数点表示方式，情報処理学会論文誌，Vol. 21, No. 4, pp. 306-313 (1980).
5) 浜田穂積：二重指数分割に基づくデータ長独立実数値表現法，情報処理学会論文誌，Vol. 22, No. 6, pp. 521-526 (1981).
6) Wilkinson. J. H. (一松 信，四条忠雄訳)：基本的演算における丸め誤差解析，p. 186，培風館，東京 (1974).

(昭和57年6月2日受付)
(昭和57年9月6日採録)