二重指数分割に基づくデータ長独立実数値表現法†

浜田 穂積††

2進法を基礎とする計算機そのほかのデータ処理装置のための実数値表現方式を提案する．形式はデータの長さに依存せず，精度変換操作が単純で，オーバフロー，アンダフローが発生しない，十分大きい数も小さい数も表現できる方式である．これらの条件を満たす一般的な表現法と，その中で最も単純な規則を持つ形式とを示し，後者を標準案として推奨する．後者によると，±1の近くでの分解能が低下しないように極力努めたので，同じビット数で表現する固定小数点表現と比べ，1ビット分の分解能の悪化に止まり，かつ浮動小数点表現の持つ，小さい数も，大きい数も表現できるという長所も併せ持っている．

実数値表現法として前記の目的を達するため，内部表現のビット列に次の3つの性質を持たせる．(i)すべてのビット列が実数に対応する．(ii)あるビット列の右に1ビット連結する時，元のビット列に対応する区間が2分され，左の区間はビット0，右の区間はビット1を連結したものに対応する．(iii)正数の場合，1から無限大あるいは0に向かって，区間の両端の値の比が，二重指数的に増加する値となるよう分割されている．

これによって実現される表現法を用いると，短いデータでもそれなりにバランスよく実数値を表現できる．また長いデータを扱う処理系と容易に結合できるため，ミニコンピュータ，マイクロコンピュータのための実数値表現法としても適している．

1. はじめに

実数値計算における内部データ表現に浮動小数点表現を用いる理由は，大きい数も小さい数も取り扱えるようにするためであり，現在十分その役割をはたしているといえる．しかしながら一部の計算においては，現在の浮動小数点表現でも表現できないほど大きい数，あるいは小さい数を扱う必要が生じる．この欠点を軽減する案が Kahan など[1),2)] いくつかある．また現在の浮動小数点表現は一定の長さの指数部をとるため，その分だけ表現可能精度が低下し，固定小数点表現と比べて不利になることも好ましくない．これらの点を解決する，実数値の新しい表現方式を考案したので，浮動小数点表現に代る新しい標準の候補として提案する．

2. 設定条件

新しい実数値表現法の設定にあたり，次の条件を必要として考慮した．

条件1：通常の数値表現との間の変換が容易なこと

2進数を基本とし，その指数と仮数の値から簡単な規則で変換できること．したがって，対数で表現するなど，複雑な計算を要するものは考慮しない．

条件2：非常に大の，あるいは小の数も表現可能で，事実上オーバフローもアンダフローもないこと

このためには，実用上重要な1の前後の表現能力を低下させることは許されないから，1の前後はきめ細かく，それから離れるにしたがって大まかに表現する方法をとらざるを得ない．

条件3：データの形式はその長さによらず，かつデータ長の変換が自然に行えるものであること

データの形式は特定の長さのみならず，任意の長さのものに一様に定義されるべきで，かつ長いデータと短いデータとの間の変換が簡単に行える方式であることが実用上不可欠である．それを用いて変換したものが元の値のよい近似になっていることが必要である．簡単な変換規則の一例として，長い形式への変換は後に0のビットを補い，短い形式への変換は左から必要な長さのビットを補い，短い形式への変換は左から必要な長さのビットを切り出す方法がある．この方法を採用したい．

条件4：データ形式を規定する取り決めが不要か，あるいは可能なかぎり少ないこと

現在採用されている浮動小数点表現は，条件3に述べたデータ長のほかにも，正負の数の表現法の違い，基数，指数部の表現など，かなりの取り決め事項が必要である．もし表現法が自然に，かつ必然的に1つのものに落ち着くのであればそれに越したことはない．取り決めがなく1つの形式であれば，異なる計算機の間のデータ互換，あるいはそれ以外にも実数値のディジタル処理を行う装置との相互間のデータ互換が図れる．その実用的利益は多大であろう．

条件5：すべてのビット・パターンが数値に対応し，

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

†† (株)日立製作所システム開発研究所

かつ異なるパターンが異なる数値に対応すること

表現しようとする長さのすべてのビット・パターンを無駄なく数値表現のために利用したい.

条件6：そのデータの表現パターンを固定小数点数とみなした場合と，大小の順序関係が一致すること

これによって，この表現での比較命令が不要となり，固定小数点用の比較命令ですむ.

これら6条件のうち，前3条件を必須とし，後3条件は，可能ならば満たしたいと考えた.

3. 具体的記述

2.の条件2を考慮すれば，指数部の長さが可変であることが必要である．これを部分的に実現したもの[3]完全に実現したもの[4]などがある．条件3，6を考慮すれば，表現の意味する範囲を大きく区切り，重みの大なる情報ほど左に置く必要があると考えられる．指数部は符号の次に重みの大なる情報であるため，符号の次に位置づけ，かつ指数部の長さがそのデータ自身に記述されているべきであると考えられる．これは**図1**に示す2分木への展開で示すことができる．ここでは，まず0あるいは1の連があり，次に連を終了させる1あるいは0となるものは端末の葉であり，連を終了させるビットのないものはさらに2分木が続くように構成されている．数表現のうち，第1ビットは符号にあてるものとして，第2ビット以後について，0あるいは1の連の長さによって，すべてのビット列を分類することができる.

指数部はこの連の長さnに対して，$n-1$ ビットの実質的に指数の値の情報を含むビットをとり，次の順に連結することによって構成すれば，上述の条件を満たすことができる.

nビットの0の連，1，$n-1$ ビットのデータ

あるいは

nビットの1の連，0，$n-1$ ビットのデータ.

指数の値の情報は次の3項目で決定する.

（i） 連を構成するビットが0か1か

（ii） 連の長さn

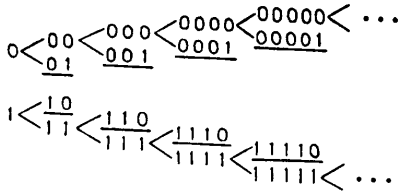

図1 2分木展開
Fig. 1 Binary tree development.

S	L	E	M

S : Sign bit (1 bit)
L : Length designator for E-part. This part consists of bit-string 0^n or 1^n0. 0^n or 1^n shows the 0's or 1's sequence of length n.
E : Exponent part ($n-1$ bits)
M : Mantissa part (the rest)

図2 データ構成
Fig. 2 Data organization.

表1 E部の値の組合せ数
Table 1 Value combination of E-part.

n	value combination of E-part
1	1
2	2
3	4
⋮	⋮
i	2^{i-1}
⋮	⋯

（iii） $n-1$ ビットのデータの値

仮数部は指数部の情報に比べて，重みの小なる情報であるから，指数部の右に置く.

以上で決定したビット配置を **図2** に示す．ここで各部の意味は次の通りである.

S：符号ビット（1ビット，負のとき1）

L：E部の長さ指定部（$n+1$ ビット）

0の連と1，あるいは1の連と0

E：指数部（$n-1$ ビット）

M：仮数部（残りすべて）

である.

nの値とE部のとり得る場合の数を **表1** に示す．表現したい数の指数の値を前記3項目の情報で表わす方法は次の通りである．まず正数の場合を述べる.

実数値xを次の形に表現する.

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

ただし，

eは整数，$1\leq m<2$ (2)

連の長さnはeで決まり，表1の右の欄の値を積算したものにとる．eとL，E部を連ねたビット列との対応の一部を **表2** に示す.

M部は $m-1$ の値を2進展開したものとする．これは，m の2進展開形 1.…の，小数点の左の1という冗長な情報のために無駄なビットを割きたくないからである.

条件5により，すべてのビット・パターンが意味を持つようにしたいが，先行するパターンによっては，

表 2 e と L, E 部のビット・パターンとの対応
Table 2 Correspondence between e and L, E-part bit pattern.

e	L, E-part bit pattern
⋮	⋮
−5	000110
−4	000111
−3	0010
−2	0011
−1	01
0	10
1	1100
2	1101
3	111000
4	111001
⋮	⋮

Underlined string shows L-part.

L, E, M 部が完備できない場合がある．このときはそのパターンの右に必要なだけ0を連結したものと同じ値を持つと解釈する．

右に0を連結しても，L部の限界を決定できない場合がある．それはすべて0のパターンのときである．この場合は $e\to-\infty$ とする極限と考えられるので，0を表わすものとする．すべて0のパターンが値0の表現であるとするのは自然であろう．

負数 x の場合は，前と同様で次の通りとする．

e は整数，$-2\leq m<-1$　　(2)′

xの表現は，S，L，E部については正の場合と同じ方法で決定したものの，0と1を逆にしたものとする．M部は $m+2$ の2進展開形とする．このように表わしたものは，$-x$の表現を固定小数点として解釈した値の2の補数をとったものと一致していることを確かめることができる（証明略）．

負の符号を持つ表現のうち，実数値と対応しない

1000⋯0　　(3)

というパターンは，$e\to\infty$ とした極限に相当しているので，$-\infty$ の表現と解釈することにする．またこのパターンは正数の場合の $e\to\infty$ としたものとも解釈できるので，符号を問題にしない無限大，あるいはKahan[2]，松井[4]などで導入している非数と解釈したい．表 3 に 0〜10 のビット表現を，E部に下線をつけて示す．

4. 形式的記述

3.における表現を形式的に定義する．

実数値を表わそうとする内部表現はビット列でありこの表現であるどのビット列も，すべて実数空間 $\boldsymbol{R}$ の部分区間に対応するものとする．

表 3 2，3の整数の表現例
Table 3 Number representations for some integers.

0	0000000000
1	0100000000
2	0110000000
3	0110100000
4	0110100000
5	0110101000
6	0110110000
7	0110111000
8	0111000000
9	0111000001
10	0111000010

ビット列の要素としてのビット空間を $\boldsymbol{B}$ とし，ある n ビットの列を S とする．

$$\boldsymbol{B}=\{0,\ 1\},\ {}^{\forall}S\in\boldsymbol{B}^n(n\geq1)\qquad(4)$$

S を，実数空間 $\boldsymbol{R}$ の部分区間に写像する関数 I を考える．この関数の値である区間が，ビット列 S に対応するものである．ここで $\boldsymbol{R}$ には便宜上 $\pm\infty$ を含めておく．区間 $I(S)$ は次の通りであるものとする．

$$I(S)=\{x\mid{}^{\exists}a,\ {}^{\exists}b\in\boldsymbol{R},\ a<b,\ a\leq x<b\}\qquad(5)$$

これを次のように略記する．

$$I(S)=[a,b)\qquad(6)$$

S の右に0を連結したものを $S0$ と記す．$S1$ も同様である．このとき関数 I は一般的に次の性質を有するものとする．

$$a<{}^{\exists}c<b,\ I(S0)=[a,c),\ \ I(S1)=[c,b)\qquad(7)$$

c と a, b の関係は S の性質によって各種の定め方がある．I を以下の4段階で定める．

第1段　疎分割

通常用いられているように，次の通りとする．

$$I(0)=[0,\infty),\ \ I(1)=[-\infty,0)\qquad(8)$$

これは符号ビットに相当する．次に，正負それぞれについて ±1 で分割して次の通りとする．

$$I(10)=[-\infty,-1)\qquad(9\text{-}1)$$
$$I(11)=[-1,0)\qquad(9\text{-}2)$$
$$I(00)=[0,1)\qquad(9\text{-}3)$$
$$I(01)=[1,\infty)\qquad(9\text{-}4)$$

第2段　二重指数分割

区間のうち，端点のいずれかに0，あるいは $\pm\infty$ を含むものを開区間，そうでないものを閉区間と呼ぶ．現実に用いられる限られた長さのビット数で，実数域を有効に分割するため，±1 に近いところは細か

く，それからはなれるほど粗く分割することにならざるを得ない．そこで，ここでは開区間について次の二重指数分割を行う．分割の回数は任意であるが，ビット列の長さが一定であるときは(9)の4区間についてそれぞれ，ビット列の長さ－2だけである．この分割はL部の決定に対応する．分割は (9-4) のものについてのみ示す．ほかの3区間については対称性を考慮して決める．1をn個連結したものを 1^n と表わす．0^n も同様である．

$$\left.\begin{aligned} I(01^n0)&=(a^n, a_n2^{2^{n-1}})\\ I(01^n1)&=(a_n2^{2^{n-1}}, \infty)\end{aligned}\right\} \tag{10}$$

二重指数型の係数が a_n に掛かる理由は，L部の長さに対応して，後のE部の長さが固定的に定まるための条件である．

$I(0\,1^n0)$ の右端と $I(0\,1^{n+1}0)$ の左端が等しいので，その条件から

$$a_n2^{2^{n-1}}=a_{n+1} \tag{11}$$

を得る．これを初期値 $a_1=1$ のもとに解いて(10)に代入したもの，およびこれをほかの3区間に適用したものをまとめて示せば次の通りである．

$$I(10^n0)=(-\infty, -2^{2^{n-1}}) \tag{12-1}$$
$$I(10^n1)=(-2^{2^n}-1, -2^{2^{n-1}}-1) \tag{12-2}$$
$$I(11^n0)=(-2-2^{-2^{n-1}+1}, -2-2^{-2^n+1}) \tag{12-3}$$
$$I(11^n1)=(-2-2^{-2^n+1}, 0) \tag{12-4}$$
$$I(00^n0)=(0, 2-2^{-2^n+1}) \tag{12-5}$$
$$I(00^n1)=(2-2^{-2^n+1}, 2-2^{-2^{n-1}+1}) \tag{12-6}$$
$$I(01^n0)=(2^{2^{n-1}}-1-1, 2^{2^n}-1) \tag{12-7}$$
$$I(01^n1)=(2^{2^n}-1, \infty) \tag{12-8}$$

第3段　等比分割

閉区間については，第2段によりその両端の値の比は $2^{2^{n-1}}$ である．そこでこの区間に，次の等比分割を $n-1$ 回適用する．この操作はE部の決定に対応し，E部に $n-1$ ビットを割くことを仮定している．ここでは(7)における c の定め方として規定する．

$$\left.\begin{aligned} &a, b>0 \text{ のとき } c=\sqrt{ab}\\ &a, b<0 \text{ のとき } c=-\sqrt{ab}\end{aligned}\right\} \tag{13}$$

第4段　等差分割

閉区間については，第3段によりその両端の値の比は2である．これに次の等差分割を任意回適用する．この操作はM部の決定に対応する．ここでも(7)における c の定め方として規定する．

$$c=(a+b)/2 \tag{14}$$

以上により，分割のための道具だては出そろった．

次に，任意のビット列 S の，対応する区間を定める手順を示す．S の長さを m ビットとする．

(a)　$m=1$ のとき(8)による．

(b)　$m=2$ のとき(9)による．

(c)　$m\geq 3$ のとき，左から第2ビットから始まる0あるいは1の連の長さ n を数える．

(i)　$n+1=m$ のとき，(12-1, 4, 5, 8) のうちの1つに帰着され，区間が決定する．

・(ii)　$n+2\leq m$ のとき，(12-2, 3, 6, 7) のうちの1つに帰着され，左 $n+2$ ビットに対応する区間が決定する．残り $m-n-2$ ビット分の分割を，必要なだけ等比分割，等差分割の順に行う．

以上で，ビット列 S に対応する区間が定まる．

ビット列 S が実数値の表現として演算のオペランドとして使われるとき，その意味として使われる値は，$I(S)$ の区間の最小値である左端の値であるとする．これは，S の右に無限に0を連結したものの，一点に収束する極限値である．

逆に，実数値のビット列 S による表現を得る方法は次の通りである．実数値 x を m ビットのビット列で表現する場合，m ビットのビット列のすべての場合 2^m 個について，その対応する区間が上述の手順で決定される．このとき，x を含む区間が必ず存在するから，この区間を対応する区間として持つビット列 S が，x の長さ m のビット列による表現である．

5. 一般化と変形

先に述べた表現の規定は，次の一般化ができる．

第1に，L部の長さ $l_L(=n+1)$ とE部の長さ l_E との関係を $l_E=l_L-2$ としたが，

$$l_E=f(l_L) \tag{15}$$

と一般化する．たとえば $l_E=12-l_L$ とすると，IEEE標準案の倍精度のものに類似する．しかしながらここで目指している条件2を満たすためには，l_E は l_L の単調非減少関数とすべきである．この中で最も単純な規則が，上述の $l_E=l_L-2$ である．その意味で **4.** で規定するものを標準案としたい．

次に，**3.** での木構造への展開，あるいは **4.** での二重指数分割を一定段数で打ち切ることである．これは打ち切る段数にもよるが，表現可能な範囲に限界が生じるため，現状に類似する．原理的に限界の生じることが問題にならない場合はこれでよい．

次に変形を述べる．よく行われているように，符号と絶対値で表現することが考えられる．このときは $I(0)$ から展開する方はここでの方法そのままとし，

負については，先頭のビット0を1に変えたものに，区間の方の符号を変えたものと対応づける．このときは0のビット表現が2通り生じて，その代り$-\infty$が定義できなくなる．機械によっては条件6を満たせなくなるなどの問題点が残る．

一般化と変形とは条件4に反する．実現にあたってどれか1つの方式に固定する必要があろう．

なお IEEE 標準案，松井などは非数の表現をいくつか含めている．このような場合のために，いくつかのビットパターンを拡張用に残すべきと主張する人* もいるが，ここで述べた方式は理想的な場合としての規定であって，実際の運用にあたって，ほとんど用いられない値に対応するパターンを拡張用に残すことは考えられる．

6. 評　価

4.で規定した表現が，条件1～6を満たすことは，規定の過程から明白である．

この表現方式においては，一定のビット数で表わすデータの場合，仮数部についても可変長であるから，値によって有効（2進）桁数が変化する．そのため他方式との間で有効桁数の比較を行った．64ビットで表わす場合の結果を**図3**に示す．ここで仮数部のビット数とは$2^{k-1}\leq x<2^k$の範囲の数を表わすパターンの数が2^i個であるときのiで表わす．横軸は$\log_2 k$の尺度で示す．

これによると，1の近くではほかのどの方式より精度がよい．しかし指数部固定長の各方式での表現可能な限界付近では逆に精度が悪くなる．とはいえそこで表現不能にはならず，徐々に精度が悪くはなるが，かなり大きいところまで表現可能である．ちなみに64ビットの場合，表現可能な最も大きい数は$2^{2^{62}-1}$であり，これは10進で約138京桁の数である．このときは有効数字がないに等しい場合であるが，仮数部が備わっている場合で表現できる最大数は$0.75\times 2^{2^{30}}$であり，これでも3億桁以上の数である．表わせる値の範囲としては十分であろう．

精度比較のための仮数部のビット数による評価は，相対誤差に着目したものである．次に絶対誤差に着目して評価する．絶対誤差は分解能の絶対値で比較する．分解能の一定な固定小数点数との比較を，本表現，松井の表現[4]，IBM 360の表現について行ったのが**図4**である．この図によると，固定小数点での表現可能な範囲でいえば，本方式は2ビットの損ということになるが，固定小数点の表現範囲を$-2\leq x<2$とすれば，分解能が1ビット分だけ大となるので，この範囲では本方式が1ビットの損ですむことがわかる．

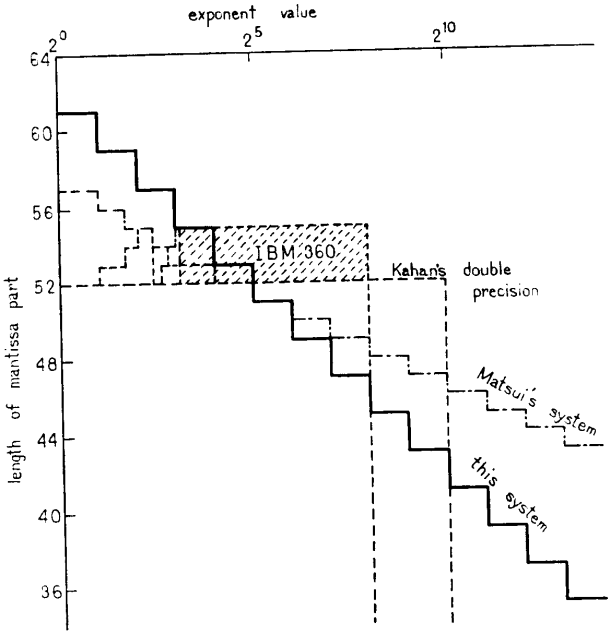

図3　指数の値と仮数部の長さ
Fig. 3　Exponent value vs. length of mantissa part.

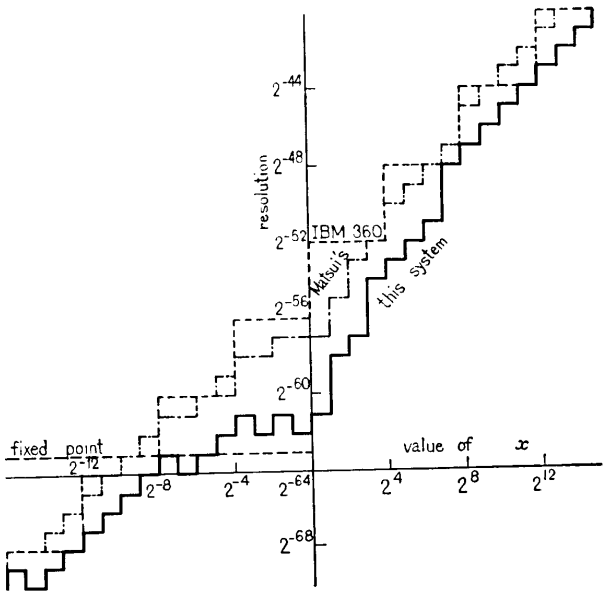

図4　分解能の絶対値
Fig. 4　Absolute resolution.

* 1980年10月に開かれた IFIP-80（メルボルン）でのパネル討論（New Floating Point Standard）において R. P. Brent が述べた，と査読者より指摘を受けたのでここに感謝して付記する．

しかも値が小のところでは分解能は，固定小数点と比べて細かくなり，値の大のところでは，固定小数点では表現できないのに，本方式では表現可能となって，固定小数点表現と浮動小数点表現の両者の長所をかね備えていることがわかる．このことは，ミニコンの場合など，短いデータでも実数値が表現可能であり，それなりに表現できるという結果を示している．指数部固定の浮動小数点表現では，指数の範囲，仮数の精度とも，中途半端になるのと比べ，本方式の有効であることがわかる．またデータ長の大のシステムとのデータの授受も，データ長の変換が容易という性質のためスムーズに行える．

数値とビット・パターンの間の対称性は，0を中心とする加法的対称性，±1を中心とする乗法的対称性ともによい対称性を示している．ただし後者については $\pm 2^i$ のものだけが完全な対称性を示す．

最後に，数値計算における誤差について，概略的な考察を行う．誤差を持つデータによる演算は次の通りであることは知られている．加減算については，結果の絶対誤差はオペランドの絶対誤差の和である．乗除算については，結果の相対誤差はオペランドの相対誤差の和である．そのほかの関数のうち，一変数関数については，結果の絶対誤差はオペランドの値における微係数にオペランドの絶対誤差を乗じたものである．多変数関数については，各オペランドによる上記微係数から生じる絶対誤差の寄与分の和である．

仮数部の長さ一定ということは，相対誤差がほぼ一定ということに対応するが，仮数部の長さ一定という性質が計算結果に一定の信頼を与えるのは，計算が乗除算とごく一部の関数の場合のみである．ところが現実の計算にはこれら以外の演算が多く含まれるため，仮数部一定という条件が数値計算上最も好ましいという結論は得られない．ともかく本方式における計算誤差に関しては，なお今後の研究に待たねばならない．

7. おわりに

2. で述べた6条件を満たす新しい数値表現方式を，3. あるいは 4. で規定するものとして提案した．またこれには 5. に述べる一般化を適用することも可能であるが，汎用計算機に用いるものとしては4.のものが，規則が単純で，表現能力が大であるため最適と考えられる．本文中で考察した点のほかに，本方式の長所である，短いビット数でも数値表現方式としてのかなりの能力があることを生かす場として，AD 変換器の出力表現などに使用して有効であると思う．またこの一応用であるが，オーディオ用 PCM，通信用 CODEC のコードとして適用することが考えられる．

別の見方をすれば，m ビットの列が 2^m 通りしか表現できないという原理的制約の中で，日常重要な±1の近くではきめ細かく，それからはなれるものはとにかく表現可能とする方式であり，我々の通常必要としている要求をバランスよく満たすものと考えられる．

本論文の内容は，昭和55年1月，京都大学数理解析研究所における研究集会「数値計算のアルゴリズム」での，東大工学部・伊理正夫教授の発表に刺激されて思いついた結果である．伊理教授に感謝する．また，懇切なコメントによるご指導をいただいた査読者の方に感謝する．

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 Newsletter, Special Issue, pp. 13-21 (Oct. 1979).
3) Morris, R.: Tapered Floating Point: A New Floating-Point Representation, IEEE Trans Comput, Vol. C-20, No. 12, pp. 1578-1579 (1971).
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).

(昭和56年2月17日受付)
(昭和56年4月27日採録)