
# 2. 計算機内部における数の表現法†

浜　田　穂　積††

## 1. はじめに

数の，電子計算機の内部における表現法は，直接アルゴリズムとはいえないが，新しい研究成果がいくつか出現したので，これを中心に解説する．

## 2. 基　数

人類の用いている記数法の大部分が10進法であることは，人の手指が10本であるためであることはいうまでもない．しかしながら電子計算機においては，2進法に基礎を置くのが都合がよい．10進法も2進法を媒介にして表現されるのが普通である．16進法も一部で用いられるが，以下では2進法と10進法を念頭に置くものとする．またここでいう2とか10のことをいう基数を$r$で表わす．

## 3. 整　数

非負の整数$N$を考える．$N$は次のように一意的に表現可能である．

$$N=a_0+a_1r+\cdots+a_kr^k+\cdots \qquad (1)$$

ただし，$a_k$ は $0\leq a_k<r$ の整数である．$a_k$ を記憶装置，あるいはレジスタ上の1桁で表現し，図示する場合は，紙の上の表記法の習慣にあうよう**図-1**のように添字の下降順に示す．1つの整数を表現するには機械によって個々に定めた長さの桁からなる語を用いる．以後1語は$n$桁であるものとする．このとき，$a_k>0\,(^{\exists}k\geq n)$ なる数は1語で表現できない．したがって1語で表現可能な整数は次の範囲に限られる．

$$0\leq N<r^n \qquad (2)$$

$n$桁でこの範囲の整数を表わす場合を，非負の整数としての表現という．負の数を表わす方法は大別して次の4つの方法がある．

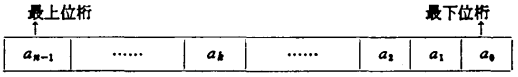


図-1　整数の表現

（1）　符号と絶対値による表現

$$N=(-1)^sN' \quad (s=\{0,1\},N'\geq 0) \qquad (3)$$

として$N$から$(s,N')$を得るとき，$s$を表わす1桁と，$N'$の表現を結合して$n$桁のデータとする．この方法は伝統的な記数法と同じ考え方のため，直感的で理解しやすい．10進法では正符号を1100，負符号を1101で表わすものがあるが，これはパンチカードにおける符号の重ね打ちの名残りである．0の表現が正負二様にできるためにトラブルの生じることもある．文字コードそのものを用いる**ゾーン**形式もある．

（2）　$r-1$ の補数による表現

正しくは「負数を$r-1$の補数により表現する方式」というべきであるが，くどいので通常表記のように言う．次の(3)についても同様である．符号と絶対値表現では加減演算の論理が複雑になるのを解決するために考案されたものである．負数は各桁の $r-1$ の補数で，すなわち $a_k'=r-1-a_k$ で表わす．最上位桁は符号と数値とのあいまいさを排すために符号のみを表わすようにするのが安全である．このとき0は正の，$r-1$ は負の符号を表わす．負数 $-N$ は $r^n-1-N$ で表わされる．すなわち$N$の符号によって次によって表わす．

$$\left.\begin{array}{l} N\geq 0 \text{ のときは} N\text{で}, \\ N\leq 0 \text{ のときは } r^n-1+N \text{ で}. \end{array}\right\} \qquad (4)$$

この方式では演算の結果に $r^n-1$ がいくつ含まれるかを考慮した補正が必要で，現在ではほとんど行われていない．

（3）　$r$の補数による表現

負数の場合に前項の結果に1を加えたもの

$$N<0 \text{ のときは } r^n+N \text{ で} \qquad (5)$$

表わすものをいう．$r^n$の補数による表現というべきか

---

† Internal Representations for Numeric Data by Hozumi HAMADA (Central Research Laboratory, Hitachi Ltd.).

†† (株)日立製作所中央研究所

も知れない．$r^n$ は $n$ 桁の表現では影響を与えないので，前項のごとき補正は不要である．一般に補数による表現は，通常の数の記法からくる直感に合致しないので理解しにくいとして敬遠されがちであるが，演算論理が単純になるなど，多くの合理性を備えている．$r$ の補数による表現は，本章の4つの表現の中で最良のものである．実数の表現においても同じである．補数による表現は加減算には適するが乗除算には適さないとする説もあるが正しくない．

(4)　ゲタバキ表現

数値 $N$ を，$N+M$ の表わす非負の整数の表現で表わすものである．正負ほぼ同数の整数を表現可能とするため，$M$ として $r^n/2$ くらいにとるのが普通である．この $M$ の値がゲタになり，必ず明記しなければならない．ゲタバキ表現自身は整数の表現に用いられることがまれであるが，浮動小数点表現の指数部の表現にしばしば用いられる．特長は，値の順序関係が非負の整数としてのそれと一致することである．

いずれにしても，$n$ 桁の固定長の語で表わせる整数はたかだか $r^n$ 個である．この制約から逃がれるためには，必要なだけの複数個の語を用意して，あたかも $r^n$ 進法を扱うかのようにして計算を行う．これを**多倍長計算**という．多倍長計算においても，$r$ の補数による表現が最も具合がよい．多倍長計算のアルゴリズムもいろいろあるが，本題からそれるので略す．

## 4. 実　数

固定の桁数の語で実数を表わそうとするとき，実数の稠密性を完全に覆うことは原理的に不可能である．内部表現として可能なパタン $r^n$ 個のうち，実数値を表わすのに用いないものを持つこともある．有効なパタンはそれぞれ特定の実数値を正しく表わすものと考えるのが普通である．まれに，ある実数値の区間を表わす場合（後述の非数など）もあるが，むしろ例外と考えられている．ほとんどの実数値については，その値を正しく表わす内部表現を選んで，それによって表わすことになる．この近いものを選ぶ操作を**丸め**という．表現したい値を，表現可能な値で代用するものである．丸めによって生じる，表現したい値と表現された値の差を**丸め誤差**という．この意味において，実数値の個々の表現は実数値の集合を代表していると考えるべきである．しかしながら，演算は常にそのオペランドが正しく表わされていると考える値について行われ，もしその結果の値を正しく表わす表現が存在するならばそれを選ぶという努力が行われている．

一つの表現法において，ある値の表現をその規則から構成しようとしても，表現可能なパタンの集合の中に存在しないとき，その値はその表現法では**表現不可能**であるという．これを溢れともいう．

### 4.1　固定小数点表現

ある実数値 $\delta$ を定め，整数 $N$ の表現（前章のもの）が実数値 $N\delta$ を表現すると考えるのを**固定小数点表現**という．$\delta$ は分解能を表わしていると考えられる．演算は整数におけるものを共用する．$\delta=1$ のとき整数と一致する．$\delta$ を何にするかは任意であるが，多くの場合ある整数 $m$ を定め，$\delta=r^{-m}$ とする．以後 $\delta=r^{-m}$ であるとして述べる．このとき仮想的小数点位置が**図-2** に示す△印となる．$m$ の値は図から $0\leq m<n$ にかぎられるとも思えるが，それ以外でもよい．$m$ の値は演算の陰で考慮されるだけであるが，$m$ に暗黙の了解をとることが困難な場合には明示する．多くの場合，表現する値 $x$ の範囲が $-1<x<1$（あるいは $-1\leq x<1$）となるように，$m=n-1$ とする伝統的な方法を用いる．これは乗算の値が表現不可能な範囲に入らないようにするためである．これらの性質からこの表現法を固定小数点表現という．

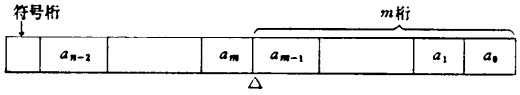


**図-2**　固定小数点表現

固定小数点表現においては，表わしたい値を正しく表わす表現がある場合はよいが，丸め誤差を持つときこれを可能なかぎり小にするには，溢れの生じない範囲で $m$ を可能なかぎり大とするのがよい．この操作を**スケーリング**という．丸め誤差を小にするためにスケーリングに気を使わねばならないことが，固定小数点表現の大きな欠点である．

### 4.2　浮動小数点表現

実数値を固定小数点表現で表わそうとするとき，とり得る値の大きさの程度がほぼ定まっている場合はよいが，どの程度になるかまったく予測がつかないとき，あるいは予測はつくが，非常に大きい値から非常に小さい値までをとり得る場合，暗黙の了解としての $m$ の値を定めることが事実上できない．このときスケーリングのための $m$ の値を可変とし，その値の表現との組にしたデータ構造にするとよい．表現したい値を $x$ として，次式の $e, f$ の組で表現する．

$$x=r^e f \quad (e:\text{整数},\ -r<f<r) \qquad (6)$$

($f$の条件は，補数を用いるとき $-r\leq f<r$ とする.) これによると $e, f$ の組は一意に定まらない．そこでいくつかの組合せのうち丸め誤差の最も小となるように次の条件を$f$につけて一意にする．

$$1\leq|f|<r \qquad (7)$$

($f<0$ で補数のとき $-r\leq f<-1$ とする.) この条件をつけた表現を正規形といい，正規形でないものを非正規形という．(7)によると0は正規形では表現不可能であり (6)によるが，$f=0$であれば$e$は任意である．0の正規形は特別に定めるが，通常$e$としてとり得る最小の値とする．

$e$の表現を指数部，$f$の表現を仮数部という．浮動小数点表現は，それぞれ定まった桁数の指数部と仮数部を連結して1語あるいはその倍数の語のデータ構造とする．例えば**図-3**の通りである．またこの逆に並べる場合もある．指数部の値は仮数部のどこが仮想的小数点位置であるかを示す情報である．仮想的小数点は固定小数点表現の場合と違って，仮数部に相対的に指数部の値によって動きうる．これが**浮動小数点表現**と呼ぶゆえんである．

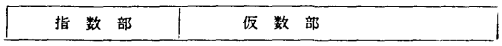


**図-3** 浮動小数点表現の構成

図-3の場合，符号はひとまず仮数の属性と考えられるが，固定小数点表現の場合最上位桁であることが多いので，符号桁のみ指数の左に移す場合が多い．$r=2$でかつ正規化されているという条件がつけば，符号ビットと，符号を除いた仮数部の最上位ビットの計2ビットのうち，とり得るのは2通りで，1ビット分冗長であるから，これを取除く場合がある．これを**ケチ表現**という．ケチ表現をとるときは0の表現に工夫を要する．

浮動小数点表現においては，符号，指数部，仮数部の順，基数，指数部，仮数部それぞれの桁数および負数の表現法，ケチ表現を採用するか否かによってかなりの種類の変形が考えられる．以下に現在行われている代表的なものと，最新の研究成果について具体的に述べる．

(1) IBM S/360 以来の表現[1),2)]

1語 32 ビットで，1語，2語，4語の3形式がある．構成要素は符号 (1ビット)，指数部 (7ビット)，仮数部の順．指数部はゲタの値 65 のゲタバキ2進整数，表現可能な範囲はほぼ $10^{-78}$〜$10^{75}$ である．仮数部は絶対値表現の 16 進で，6桁 (1語)，14桁 (2語)，28桁 (4語) からなる．それ以前 (例えば IBM 7090) は2進を用いていたが，S/360 では 16 進とした．これにより，表現可能な範囲が拡大し，仮数部の実質的桁数も長くなる巧妙な方式と考えられたが，その後の研究によると，精度を定める相対誤差の評価は，$f$の小数点以下の桁数で定まり，$r$の大なる方式はその点で不利であることが判明した．したがって，同じビット数を用いて，表現可能な範囲も同じとした2進による場合と比較して，ケチ方式でないものより1ビット，ケチ方式とは2ビット不利になっている．

(2) IEEE 標準案[1),3)]

先に述べたように，浮動小数点表現は，その形式を定める要因が多く，それらの組合せはかなりのものになる．基数が同じ，語の長さが同じであっても，表現可能な範囲と精度が異なれば計算結果は異なる．そこで統一規格を決める動きとなり，IEEE の標準案が作成された．すでにミニ・コンピュータ，マイクロ・コンピュータでの標準的表現法として使われ始めている．1語 32 ビットを想定し，基本形式 (記憶装置上の表現と考えられる) として単精度 (1語)，倍精度 (2語) の2形式，拡張形式 (レジスタ上の表現と考えられる) として単精度，倍精度の2形式がある．拡張形式はいずれか一つのみを用いることになっている．構成要素は符号 (1ビット)，指数部，仮数部の順．指数部はゲタバキ2進整数，仮数部は絶対値によるケチ表現である．数量的データは**表-1**の通りである．

**表-1** IEEE 標準案

| 形式 | | 指数部の長さ | ゲタの値 | 仮数部の長さ | 表現可能範囲 |
|---|---|---|---|---|---|
| 基本 | 単精度 | 8ビット | 127 | 23ビット | 〜 約 $10^{38}$ |
| | 倍精度 | 11 | 1023 | 52 | 〜 約 $10^{308}$ |
| 拡張 | 単精度 | 11 | 1024 | 31 | 〜 約 $10^{308}$ |
| | 倍精度 | 15 | 16384 | 63 | 〜 約 $10^{4932}$ |

IEEE 標準案の新しい提案は，丸めの方式を4種 (RN, RZ, RP, RM) 認めたこと，および $\pm0, \pm\infty$ その他の非数を導入したことである．逆に問題となりそうな点として，拡張形式から基本形式に戻す場合に溢れが生じること，および戻すタイミングがコンパイラの最適化処理によって変わりうるので，計算結果がやはり常に同じとはならない点である．いずれにして

も，現状との妥協の産物との感はまぬがれない．

（3） 松井・伊理による表現[1),4)]

松井と伊理は，表現可能な数に上限があること，すなわち，溢れの現象が起こることはよくないと考えた．たしかに，計算の最終的結果のみを考えれば，現行の方式での表現範囲で十分かも知れない．しかし計算の途中では必ずしもそうはいえないし，計算途中で溢れの起こるアルゴリズムを計算機が拒否するのはよくない．表現可能な数の上限は指数部に割当てるビット数で決まるが，それに上限を設けないためには可変長にすればよい．そこで**図-4**に示す構成とした．原論文ではデータ長64ビットのものを述べているので，指数部の長さの値を入れる $l$ 部は6ビットとなる．指数部の長さを指定するので指数部も2進ケチ表現にできる．また指数部，仮数部とも符号と絶対値方式としている．0を含む非数の表現は特別のパタンを割当てる．

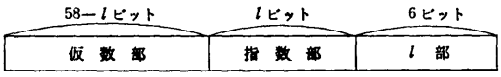


**図-4** 松井・伊理による実数値表現

これまでの方式の共通点である仮数部の長さ一定という性質は，表現誤差の相対値ほぼ一定に対応するが，本方式はそうなっていない．この点に危惧をとな見もあるが，必ずしもそうはいえない．絶対誤差一定える意の系は加減算に，相対誤差一定の系は乗除算に一定の信頼を与える．その他の演算については一概にはいえない．したがって相対誤差一定が絶対的規準ではないからである．

松井・伊理は本方式の長所を示すため，Graeffe の方法による高次方程式の解法が適用でき，よい結果を得ることが可能であることを示した．

（4） 浜田による表現[5)]

先に述べたように，実数値の表現は同一方式内にも複数種用意されるのが普通である．それらの間で，値は変えないで（精度が落ちることもある）形式のみを変換する操作が度々必要となる．この操作を極度に単純化するものを区間分割法で実現したものである．しかも分割点を $\pm1$ から0あるいは $\pm\infty$ に向かって二重指数的に増大（減少）するように選ぶことによって，溢れを事実上生じないようにし，かつ $\pm1$ の付近では十分精度良く表現できるものとなっている．構成は，符号（1ビット），指数部（可変長），仮数部の順．指数部は可変長であるが，その長さを入れるフィールドは特になく指数部が自力でその長さを示すものとなっている．693 の表現を例にとって示す．式(6)による $e, f$ の値の2進表現は次の通りである．

$$\left.\begin{aligned} e&=1001\\ f&=1.010110101 \end{aligned}\right\} \quad (8)$$

このとき，指数の値 $e$ が4桁であることを示すため，5ビットの1の連，1の連の終りを示す1ビットの0（指数の値が負のときは0と1を逆にする），およびケチ表現された $e$ の下3ビットを連結して

111110001　　(9)

とする．この左右に符号ビット0，ケチ表現された $f$ の表現を連結して，693 の表現として次を得る．

0111110001010110101　　(10)

負数の場合，指数の順と，数値の順は逆になるので，(9)のようにして得られたビットパタンの1の補数とする．$-2\leq x<-0.5$，$0.5\leq x<2$ の範囲における指数部は例外的で，次の通りである．これは，$e$ の2進表現と数の指数部のパタンとの類似性を得るために得られた例外である．符号を含めて示す．

$$\left.\begin{aligned} -2\leq x<-1 &\quad : 101\\ -1\leq x<-0.5 &\quad : 110\\ 0.5\leq x<1 &\quad : 001\\ 1\leq x<2 &\quad : 010 \end{aligned}\right\} \quad (11)$$

本表現の長所は次の3点にまとめられる．

（a） 形式がデータの長さによらず，長いデータと短いデータの変換操作が事実上不要である．

（b） 溢れが事実上起こらない．

（c） 固定小数点表現と比べて1ビットの不利に止まる．

### 4.3 その他の表現[6)]

実数の，その他の表現として有理数による表現法をあげておく．**図-5**に示す構成とするのが基本である．

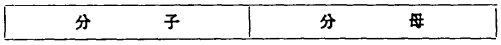


**図-5** 有理数による表現

分子，分母ともに整数の表現による．符号はどちらにつけてもよいが，分母を非負と決めれば1桁分たすかる．分子，分母の長さの配分を可変にするように，分母の長さ部などを設けてもよい．もっと大きい値などを表わすために指数部を置くことも考えられる．この表現の長所は次のものである．

（1） 四則演算に関して閉じている．内部の実質的

演算は加減乗算のみでよい.

(2) 有理数の四則演算の範囲では常に正しい値が得られる.

(3) 浮動小数点表現にある基数依存性がない. 10進で簡単な表現0.1が2進浮動小数点では正しく表わせない.

演算を続けてゆくと分子, 分母とも大きな整数になって表わせなくなる. そのときは連分数展開の打切りによって小さい分子, 分母による近似値を得ることができる. これが丸めに相当する. 実際にはこの丸めがしばしば起こり, しかも時間をかなり消費することになりそうである. その他に, 表現する数の, 実数軸上の稠密性が不均一な点が欠点として挙げられる.

## 5. おわりに

数の表現という基礎的な分野は, すでに解明されつくされているように, 一般には受取られているようであるが, 必ずしもそうでないといえる. 新しい研究成果を採り入れた機械が実現されて, 計算上の制約が軽減され, 信頼度が向上するのに貢献できるようになることを期待したい. また本解説が Ada 等新しい言語の実現とその応用にも役立てば幸いである.

## 参考文献

1) 一松 信: 計算機内部の数の表現, 数学セミナー Vol. 20, No. 1, pp. 34-38 (1981年1月).
2) IBM: IBM System/370 Principles of Operation, GA 22-7000-8 (Oct. 1981).
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年7月).
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II, 情報処理学会論文誌, Vol. 24, No. 2, pp. 149-156 (1983年3月).
6) 前川 守: 数値の有理数表現, bit, Vol. 14, No. 5, pp. 637-639 (1982年4月).

(昭和57年12月1日受付)