一般化した二重指数分割に基づく数値表現法

富松 剛[†] 金田康正[††]

浜田の提案したURR（Universal Representation of Real numbers）表現は絶対値が1から離れるに従って，急速に精度が悪化するという欠点がある．この欠点を改善するために一般化した二重指数分割に基づく数値表現法を提案する．URR表現は二重指数 $\pm 2^{\pm 2^m}$ で実数を大まかに近似するが，本論文ではこれを $\pm p^{\pm q^m}$ と一般化した．この一般化した表現では p と q を大きくすることで，大きな実数に対してURR表現よりも少ないビット数で近似できるようになる．特に $p=4, q=16$ と選ぶことによって，URR表現に比べ広範囲で精度の良い二重指数分割に基づく数値表現法となる．この一般化した数値表現は数々のURR表現の長所を有し，急速な精度の悪化も抑えている．この数値表現で実際に簡単な数値計算を行い，URR表現に比べ精度が良い本数値表現の有効性を検証した．

A Generalized Numerical Representation Based on Double Exponential Cut

TSUYOSHI TOMIMATSU[†] and YASUMASA KANADA[††]

A precision of URR (Universal Representation of Real numbers) which was proposed by Hamada goes to worse rapidly when the absolute value goes away from 1. We proposed generalized numerical representation based on the double exponential cut for the improvement to the serious drawback of URR. URR approximates real numbers with double exponential form of $\pm 2^{\pm 2^m}$. We generalized the base of double expential cut as $\pm p^{\pm q^m}$ in this paper. With larger p and q, our representation can approximate large numbers, spending lesser bit length than URR. Especially, our numerical representation with parameters of $p=4$, $q=16$ is preciser than URR. The generalized representation also has the same merit as in URR, and it can escape from the demerit in URR. We verified the effectiveness of our representation with simple numerical calculation.

1. はじめに

数値計算におけるあふれの発生は昔から問題となっている．松井・伊理はあふれの発生しにくい数値表現法として指数部を可変とした数値表現法を提案した[1]．この数値表現の指数部の長さは表現する実数の指数の絶対値の値により長さが決定される．そのためあふれは事実上生じない．しかしこの表現方法は単精度や倍精度といった長さの違うフォーマット間において相互変換が困難となる欠点が指摘され，浜田によりURR表現（Universal Representation of Real numbers）が提案された[2)~4)]．このURR表現は二重指数 2^{2^m} による実数区間の分割に基づく数値表現法であり，他の可変長指数部の数値表現と同様にあふれが事実上生じない．

このURR表現の長所を以下にまとめると，

(1) 指数部が可変長であり，事実上あふれが発生しない．
(2) 単精度，倍精度などの長さの違うフォーマット間での変換が容易である．
(3) URR表現の大小関係がビット列を固定小数点と見た場合の大小関係と同じである．
(4) 1付近の精度が非常に良い．

となる．しかしURR表現は1から離れるに従って，急速に精度が悪化するという短所がある．この急速な精度の悪化を抑えた表現方法として，URR表現を改良した中森の三重指数分割[5]が提案された．この表現方法は2の三重指数に基づく分割であるため，表現や演算が非常に複雑になる．また指数部の各部分の必要ビット数から精度を検証したものに横尾，中森の多重指数分割[6)~8)]がある．しかし，多重指数分割は具体的な区間の分割方法は示されていない．

このように，これまで分割の基数を2以外にした

† 東京大学大学院理学系研究科情報科学専攻
Department of Information Science, Graduate School of Science, University of Tokyo
†† 東京大学大型計算機センター
Computer Centre, University of Tokyo

表現方法は報告されていない．先に拡張した二重指数分割表現による数値表現法に関する研究[9]として一般化した二重指数分割に基づく数値表現について述べたが，精度の解析が不十分であったのでその補足も含めて，本論文ではURR表現での二重指数 2^{2^m} の分割を p^{q^m} と一般化した二重指数分割に基づく数値表現（以下，一般化二重指数分割表現と呼ぶ）について述べる．2章でURR表現の基礎となっている二重指数分割に基づく数値表現と，その一般化について述べる．3章で一般化二重指数分割に基づく数値表現法の表現精度について述べる．4章で本表現においてURR表現の欠点を改善した数値表現法を提案する．さらにこの本表現とIEEE表現，IBM表現，松井・伊理の表現，三重指数分割表現との精度の比較を行う．5章では本表現を用いて簡単な数値計算を行いその有効性について検証を行う．

2. 二重指数分割表現の一般化

2.1 URR表現とは

浜田の提案した数値表現であるURR表現は，URR表現のビット列に対応した規則に基づいて実数区間を分割し，表現すべき実数値が分割区間の上限に属すればURR表現を表すビット列に1を連結し，下限に属すれば同様にビット列に0を連結していくという操作を繰り返して数値を表現する方法である．以下に実数区間の分割を簡単に説明する．

ビット列 s が実数値の1つの区間に対応するものとする．

$$s : \{x | a \leq x < b\} \tag{1}$$

これを次の記法を用いて表現する．

$$U(s) = [a, b) \tag{2}$$

このときにビット列 s が正しく示す値は区間の下限の a であると定める．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 $s1$ とする．このとき，s および，a，b の値によって決まる第三の値 c によって $[a, b)$ の区間は分割され，次の対応づけが行われる．

$$U(s0) = [a, c) \quad , \quad U(s1) = [c, b) \tag{3}$$

以上を新しい区間として，再び分割を繰り返す．

このURR表現は分割の最初の段階において，第三の値 c の決定に2を基数とした二重指数 $\pm 2^{\pm 2^m}$ を用いているが，この基数を2以外にした場合の精度の変化を調べるために基数を p, q（p, q は2以上の整数）とし，$\pm p^{\pm q^m}$ の二重指数分割に基づく数値表現に一般化する．この一般化した二重指数分割に基づく数値表現法の性質を調べる．

2.2 一般化二重指数分割に基づく数値表現

以下の分割において，$P^+(m) = p^{+q^m}$，$P^-(m) = p^{-q^m}$ と定義する．また，n 個の0または1の連なりをそれぞれ 0^n および 1^n と表記する．

以下に一般化二重指数分割表現の定義を示す．

2.2.1 大まかな分割

まず，二進整数表現を考慮して，以下のように定義する．

$$\begin{array}{rcrcl} 0 & \leq & U(0) & < & +\infty \\ -\infty & < & U(1) & < & 0 \end{array} \tag{4}$$

続いて，この区間を ± 1 で分割し，以下の範囲を得る．

$$\begin{array}{rcrcl} 1 & \leq & U(01) & < & +\infty \\ 0 & \leq & U(00) & < & 1 \\ -1 & \leq & U(11) & < & 0 \\ -\infty & < & U(10) & < & -1 \end{array} \tag{5}$$

さらに $\pm p$，$\pm 1/p$ で分割し，以下の範囲を得る．

$$\begin{array}{rcrcl} p & \leq & U(011) & < & +\infty \\ 1 & \leq & U(010) & < & p \\ 1/p & \leq & U(001) & < & 1 \\ 0 & \leq & U(000) & < & 1/p \\ -1/p & \leq & U(111) & < & 0 \\ -1 & \leq & U(110) & < & -1/p \\ -p & \leq & U(101) & < & -1 \\ -\infty & < & U(100) & < & -p \end{array} \tag{6}$$

以上で大まかな分割は終了する．このときに，上限と下限の比が p のものは，等比分割へ移る．そうでないものは，第1二重指数分割を行う．

2.2.2 第1二重指数分割

第1二重指数分割は，区間の上限または下限に0または $\pm\infty$ を含むそれぞれの区間について以下の分割を行う．

$$\begin{array}{rcl} +P^+(m) \leq U\left(01^{m+2}\right) < +\infty & & \\ \qquad \text{を} + P^+(m+1) \text{で} & & \\ 0 \leq U\left(00^{m+2}\right) < +P^-(m) & & \\ \qquad \text{を} + P^-(m+1) \text{で} & & \\ -P^-(m) \leq U\left(11^{m+2}\right) < 0 & & \\ \qquad \text{を} - P^-(m+1) \text{で} & & \\ -\infty < U\left(10^{m+2}\right) < -P^+(m) & & \\ \qquad \text{を} - P^+(m+1) \text{で} & & \end{array} \tag{7}$$

この分割を帰納的に行うことで，最終的に上限にも下限にも0または $\pm\infty$ を含まない以下の範囲を得ることができる．

$$\begin{aligned} +P^+(m) &\le U\left(01^{m+2}0\right) < +P^+(m+1) \\ +P^-(m+1) &\le U\left(00^{m+2}1\right) < +P^-(m) \\ -P^-(m) &\le U\left(11^{m+2}0\right) < -P^-(m+1) \\ -P^+(m+1) &\le U\left(10^{m+2}1\right) < -P^+(m) \end{aligned} \quad (8)$$

2.2.3 第2二重指数分割

続いて第2二重指数分割を行う．それぞれ以下の範囲について

$$\begin{aligned} &+P^+(a) \le U(s) < +P^+(b) \text{ を} \\ &\qquad +P^+\left(\frac{a+b}{2}\right) \text{で} \\ &+P^-(a) \le U(s) < +P^-(b) \text{ を} \\ &\qquad +P^-\left(\frac{a+b}{2}\right) \text{で} \\ &-P^-(a) \le U(s) < -P^-(b) \text{ を} \\ &\qquad -P^-\left(\frac{a+b}{2}\right) \text{で} \\ &-P^+(a) \le U(s) < -P^+(b) \text{ を} \\ &\qquad -P^+\left(\frac{a+b}{2}\right) \text{で} \end{aligned} \quad (9)$$

分割を行う．この分割を $\log_2 \log_2 q$ 回行う．

2.2.4 等比分割

等比分割は以下の式によって定義される．上限と下限の比が2になるまで分割を行う．

$$a \le U(s0) < a\sqrt{\frac{b}{a}},\ a\sqrt{\frac{b}{a}} \le U(s1) < b \quad (10)$$

2.2.5 等差分割

等差分割は以下の式によって定義される．十分な精度が得られるまで以下の分割を繰り返し行う．

$$a \le U(s0) < \frac{a+b}{2},\ \frac{a+b}{2} \le U(s1) < b \quad (11)$$

以上で分割は終了する．第1二重指数分割から等差分割までの各分割は必ずしもすべて行う必要はなく，所望の精度が得られたところで打ち切って構わない．以上が一般化二重指数分割表現の定義である．

この定義からも明らかなように $p=2$, $q=2$ のときがURR表現となる．また本数値表現は大小比較や符号反転が容易といったURR表現の長所をそのまま有している．

さらに，分割は偏りなく行われるので

$$a \le U(s) < b \quad (12)$$

のとき

$$\lim_{n\to\infty} U(s0^n) = a \quad (13)$$

は明らかである．そこで一般化二重指数分割表現で分割終了後に引き続くビット列はすべて0と見なすと，本数値表現のビット列は正確に下限の値を示し，また実数0も $U(00\cdots00)$ となり都合がよい．

2.3 分割例

まず $p=4, q=16$ の場合の一般化二重指数分割表現で，実数288を表現する場合の分割例を以下に示す．

・大まかな分割
$4 \le U(011) < +\infty$
・第1二重指数分割
$4 \le U(0110) < 4^{16}$
・第2二重指数分割
$256 \le U(01101) < 4^{16}$
$256 \le U(011010) < 65536$
・等比分割
$256 \le U(0110100) < 4096$
$256 \le U(01101000) < 1024$
$256 \le U(011010000) < 512$
・等差分割
$256 \le U(0110100000) < 384$
$256 \le U(01101000000) < 320$
$288 \le U(011010000001) < 320$

以下0のビット列が精度に応じて続く．

次に $p=4, q=4$ の場合の一般化二重指数分割表現で，実数 $-0.1875 = -3/16$ を表現する場合の分割例を以下に示す．

・大まかな分割
$-1/4 \le U(111) < 0$
・第1二重指数分割
$-1/4 \le U(1110) < -1/256$
・第2二重指数分割
$-1/4 \le U(11100) < -1/16$
・等比分割
$-1/4 \le U(111000) < -1/8$
・等差分割
$-3/16 \le U(1110001) < -1/8$

以下0のビット列が精度に応じて続く．

3. 一般化二重指数分割表現の精度

一般化二重指数分割表現の精度（数値の分解能）は，各分割に要するビット数を計算し，そのビット数より求めることができる．

3.1 各分割における必要ビット数の計算

表現する実数値 x の値が $-p \le x < -1/p$, $1/p \le x < p$ の場合と，それ以外とで必要ビット数が変わるので，2つの場合に分けて解析する．

(1) $-p \le x < -1/p$, $1/p \le x < p$ の場合

まず大まかな分割では，その定義から明らかなよう

に符号部のビットも含めて，3ビットが費される．大まかな分割は以上で終了し，そのときの上限と下限の比は p になるので，そのまま等比分割に移る．

等比分割では上限と下限の比が p から始まるので，これが2になるまでの分割回数は $\log_2 \log_2 p$ 回である．よって等比分割部は $\log_2 \log_2 p$ ビット要する．

等差分割は定義より任意の回数分割を行うので，等差分割部に割り当てられるビットは残り全部となる．この等差分割に割り当てられるビット数が本表現の精度となる．

また，$\log_2 \log_2 p$ 回の分割が整数であることから p の値は以下の形をしていなくてはならないことが分かる．

$$p = 2^{2^{p'}} \quad (p' = 0, 1, 2, \cdots) \tag{14}$$

(2) $x < -p$, $-1/p \leq x < 1/p$, $x \geq p$ の場合

まず大まかな分割と第1二重指数分割では，その定義から明らかなように $m+4$ ビットが費される．

続く第2二重指数分割と等比分割では $x \leq -p$, $-1/p \leq x < 0$, $0 < x < 1/p$, $x \geq p$ の4つの区間に分けて解析を行うが，それぞれ同様の手法で解析が行えるので，$x \geq p$ についてのみ以下に解析を行う．

$x \geq p$ この区間では第1二重指数分割の終了時は以下の範囲を得ることができる．

$$+P^+(m) \leq U(s) < +P^+(m+1) \tag{15}$$

第2二重指数分割の分割方法に従って1回目の分割を行う．

$$\begin{aligned} +P^+(m) &\leq U(s0) < +P^+(m+1/2) \\ +P^+(m+1/2) &\leq U(s1) < +P^+(m+1) \end{aligned}$$

同様に2回目の分割を行う．

$$\begin{aligned} +P^+(m) &\leq U(s00) < +P^+(m+1/4) \\ +P^+(m+1/4) &\leq U(s01) < +P^+(m+2/4) \\ +P^+(m+2/4) &\leq U(s10) < +P^+(m+3/4) \\ +P^+(m+3/4) &\leq U(s11) < +P^+(m+1) \end{aligned}$$

つまり n 回分割後の範囲は，以下のように表すことができる．

$$+P^+\left(m+\frac{j}{2^n}\right) \leq U(s) < +P^+\left(m+\frac{j+1}{2^n}\right) \quad (j = 0, 1, \cdots, 2^n - 1) \tag{16}$$

このときの上限と下限の比は以下のようになる，

$$p^{\left\{q^{m+j/2^n}\left(q^{1/2^n}-1\right)\right\}} \tag{17}$$

等比分割終了時の上限と下限の比が2になる☆ためには，上式のべき乗部分が2のべき乗の形をしていなくてはならない．そのことより第2二重指数分割の分割回数の定義であった $n = \log_2 \log_2 q$ が求まる．同時に n の式より q は以下の形をしていなくてはならない．

$$q = 2^{2^{q'}} \quad (q' = 0, 1, 2, \cdots) \tag{18}$$

よって n 回分割後の上限と下限は以下のようになる．

$$p^{q^m 2^j} \leq U(s) < p^{q^m 2^{j+1}} \tag{19}$$

続いて等比分割に必要なビット数を求める．先の範囲を等比分割の分割方法に従って1回分割を行うと以下のようになる．

$$\begin{aligned} p^{q^m 2^j} &\leq U(s0) < p^{q^m 2^j (1+1/2)} \\ p^{q^m 2^j (1+1/2)} &\leq U(s1) < p^{q^m 2^{j+1}} \end{aligned}$$

同様に2回目の分割を行う．

$$\begin{aligned} p^{q^m 2^j} &\leq U(s00) < p^{q^m 2^j (1+1/4)} \\ p^{q^m 2^j (1+1/4)} &\leq U(s01) < p^{q^m 2^j (1+2/4)} \\ p^{q^m 2^j (1+2/4)} &\leq U(s10) < p^{q^m 2^j (1+3/4)} \\ p^{q^m 2^j (1+3/4)} &\leq U(s11) < p^{q^m 2^{j+1}} \end{aligned}$$

l 回分割後の範囲は，以下のように表すことができる．

$$p^{q^m 2^j \left(1+k2^{-l}\right)} \leq U(s) < p^{q^m 2^j \left(1+(k+1)2^{-l}\right)} \quad (k = 0, 1, \cdots, 2^l - 1) \tag{20}$$

このとき上限と下限の比は2になっていなくてはならないので，

$$p^{q^m 2^{j-l}} = 2 \tag{21}$$

等比分割の分割回数 l は以下のようになる．

$$l = j + m \log_2 q + \log_2 \log_2 p \tag{22}$$

よって l 回分割後の上限と下限は以下のようになる．

$$2^k p^{q^m 2^j} \leq U(s) < 2^{k+1} p^{q^m 2^j} \tag{23}$$

以上より，$x \geq p$ の範囲における所用ビット数は大まかな分割と第1二重指数分割部に $m+4$ ビット，第2二重指数分割部に $\log_2 \log_2 q$ ビット，等比分割部に $j + m \log_2 q + \log_2 \log_2 p$ ビット必要であることが分かる．

同様に解析を行い各分割における必要ビット数をまとめると**表1**のようになる．

☆ 等比分割終了時の比が2ということは等差分割の開始時の比が2ということである．比を2にすることによって等差分割はIEEE表現等のけち表現の仮数部と同様の表現になり扱いやすくなる．

表1 L ビット長数値表現における各段階で必要なビット数
Table 1 Required bits for each separation stages to L bits width representation number.

	$1/p \le x < p$ $-p \le x < -1/p$	$x < -p, x \ge p$ $-1/p \le x < 1/p$
第1	3	m+4
第2	—	$\log_2 \log_2 q$
等比	$\log_2 \log_2 p$	$j + m \log_2 q + \log_2 \log_2 p$
等差	$L - 3 - \log_2 \log_2 p$	$L - 4 - j - m(1 + \log_2 q)$ $- \log_2 \log_2 q - \log_2 \log_2 p$

3.2 表現誤差

一般化二重指数分割の等差分割部はIEEE表現の仮数部に相当する．実際 $x \ge p$ の範囲においては等差分割部の最上位に1を付加することによって1以上2未満の符号付き固定小数点表記と見なすことができる．

さて，一般に数値表現の仮数部と相対誤差の関係は以下のようになる．ある実数 x に対応する代表点 x^* は仮数部の最後の桁の次の桁を丸めた数であるとされる[1]．このとき，

$$e_{rep}(x) = \frac{|x^* - x|}{|x|} \tag{24}$$

を x の相対誤差と呼ぶ．また，

$$u(x) = \max\{e_{rep}(y)| \quad y \text{ は } x \text{ と同じ代表点 } x^* \text{ を持つ}\}$$

を丸めの単位と呼ぶ．つまり丸めの単位は x^* に丸められる実数の中で相対誤差最大のものである．β 進法で仮数部が l 桁のとき丸めの単位は

$$\frac{1}{2}\beta^{1-l}$$

となる．

L ビット長の数値の一般化二重指数分割表現の等差分割部の長さは表1に示したとおりである．ここで，等差分割部の長さを D とすると，等差分割部は最上位のビットを省略したけち表現となっているので事実上 $D+1$ の長さがある．すなわち，それぞれの範囲において 2^{-D-1} の相対誤差で表現が可能であることを指摘しておく．

4. 他の数値表現法との比較

4.1 一般化二重指数分割表現の最適条件

一般化二重指数分割表現は p と q の値を変化させることによって，多様な精度を持った表現が可能である．そのためどのような p と q が良いか判断するための評価基準を設けねばならない．そこで本論文では以下の基準を設定する．

(1) わずかな一部の範囲（1カ所もしくは2カ所）を除いてURR表現と同じか，それ以上の精度であること☆．

(2) 精度の悪化がなるべく緩やかであること．

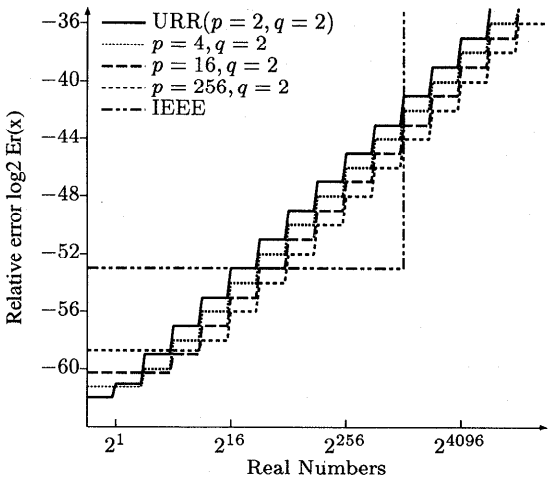


図1 $q=2$ で p を変化させたときの一般化二重指数分割表現の精度の変化
Fig. 1 Precision of the generalized double expontntial representation ($q=2$, p is varying).

この2つの基準により，URR表現の欠点である急速な精度の悪化を抑え，かつURR表現よりも広範囲において精度の良い数値表現方法を選ぶことができる．

次に64ビット長の数値表現に限定させ，p と q を変化させた場合の相対誤差の変化の様子を調べた．p を変化させた様子を図1に，q を変化させた様子を図2に示す．p を大きくすると1付近で精度は落ちるが全体として精度は良くなる．また，q を大きくすると横軸が p を超えたところで急に精度が落ちるが，それ以降の傾きは緩やかになる．

最適な p と q は，条件1より p の値は2か4でなければならない．それよりも大きいと1付近においてURR表現よりも2ビット以上精度が悪くなってしまうからである．また条件2より q の値はなるべく大きい方がよい．

以上の条件を満足している p と q の組合せは

$$(p,q) = \{(2,4),(4,2),(4,4),(4,16)\} \tag{25}$$

になる．この中で精度の悪化の最も緩やかなもの，つまり q の一番大きなものは $p=4$，$q=16$ である．この数値表現を以下 $(4,16)$ 二重指数分割表現と呼ぶことにする．

☆ すべての範囲においてURR表現よりも精度の良い数値表現は存在しない．また，ある範囲でURR表現よりも精度を良くすれば，それは必ず別の範囲で精度の悪化を起こす．この条件は，URR表現よりも総合的に良い精度がえられれば，わずかな範囲での1ビットの悪化はやむをえないものと考えることを意味する．

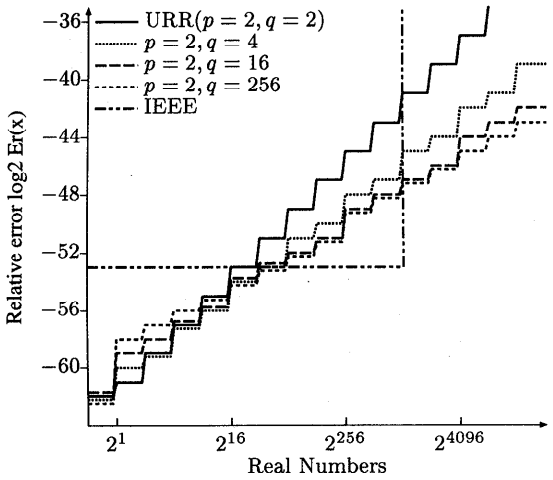


図2 $p=2$ で q を変化させたときの一般化二重指数分割表現の精度の変化

Fig. 2 Precision of the generalized double exponential representation ($p=2$, q is varying).

4.2 他の数値表現法との表現精度の比較

(4, 16) 二重指数分割表現の精度について，他の表現と比較したものが図 **3** である．(4, 16) 二重指数分割表現は URR 表現と比較して 1 付近での精度は 1〜2，4〜16 の 2 つの範囲において 1 ビット精度が悪いが，URR 表現に比べ精度の良い範囲が広く，精度の悪化が緩やかであることが分かる．

5. 数値計算への応用と評価

URR 表現と (4, 16) 二重指数分割表現で実際に簡単な数値計算を行い，計算誤差の振舞いを調べた．

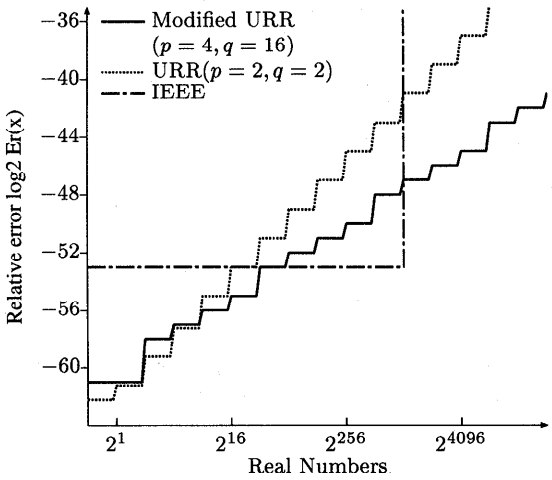
5.1 Graeffe の方法を用いた代数方程式の解

松井・伊理が適用し[1]，浜田も試みた問題[4),10)]（Graeffe の方法による代数方程式の解）と同じ問題に，(4, 16) 二重指数分割表現を適用し，精度の振舞いを比較する☆．

高次代数方程式の Graeffe による解法[11)]は，手順が簡単であるが，計算過程で指数の絶対値の非常に大きい数値を取り扱わねばならず，従来の数値表現では精度良く計算することが困難であった．しかし，可変長指数部の数値表現では指数の絶対値の非常に大きな数値も取り扱うことができるために，Graeffe の方法による評価は有効である．

Graeffe の方法を適用した式は

$$p_1(x) = x^8 - 11x^7 + 45.35x^6 - 88.55x^5$$



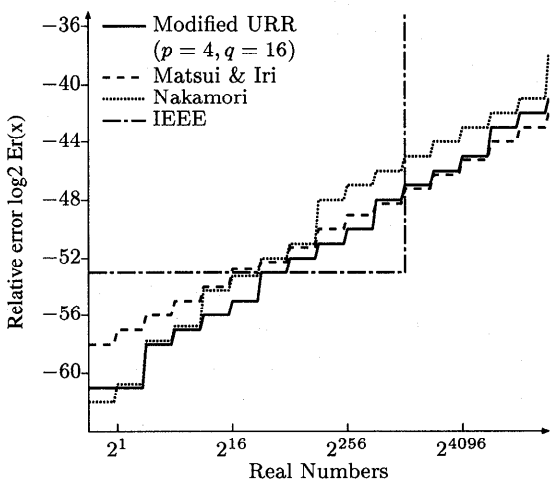


図3 最適な一般化二重指数分割表現（$p=4$, $q=16$）と各種の数値表現の精度の比較

Fig. 3 Precision of the optimum generalized double exponential representaiton ($p=4$, $q=16$) and several other representations.

$$\begin{aligned} &+86.7524x^4 - 42.274x^3 + 10.984x^2 \\ &-1.32x + 0.0576 \\ &= (x-0.1)(x-0.2)(x-0.3) \\ &(x-0.4)(x-1)(x-2) \\ &(x-3)(x-4) \end{aligned} \tag{26}$$

$$\begin{aligned} p_2(x) = {} & x^4 - 10.43857593020614x^3 \\ &+40.58740567587410x^2 \\ &-69.60408570545396x \\ &+44.36715614906059 \\ = {} & (x-2)(x-e) \\ &(x-\sqrt{7.4})(x-3) \end{aligned} \tag{27}$$

☆ 今回の実験に用いたシミュレータは，URR 表現，一般化二重指数分割表現とも計算過程での数値のまるめを切捨てで行った．

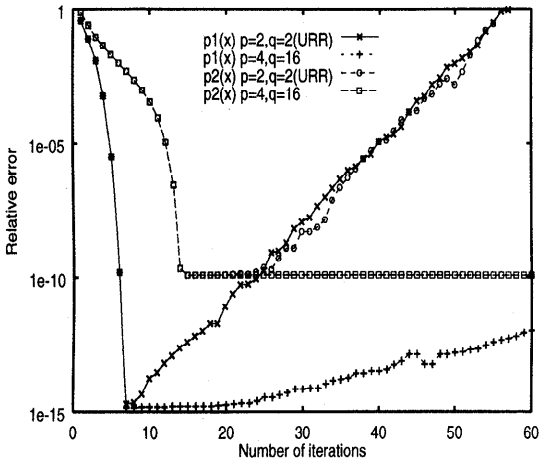


図4 Graeffe の方法を用いた代数方程式の解（$p=4$, $q=16$ の場合と比較）

Fig. 4 Solving single variable polynomial equation by Graeffe's method.

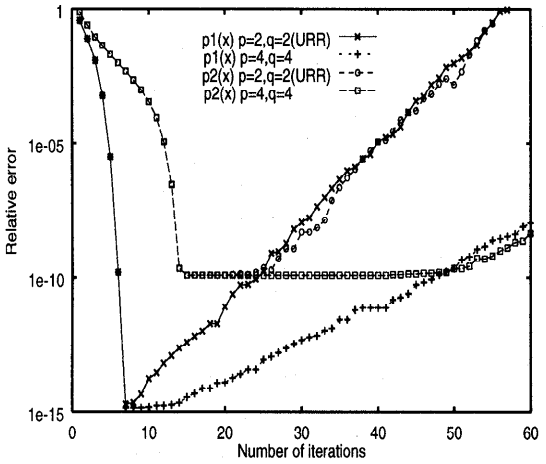


図5 Graeffe の方法を用いた代数方程式の解（$p=4$, $q=4$ の場合と比較）

Fig. 5 Solving single variable polynomial equation by Graeffe's method.

$$\begin{pmatrix} e & = & 2.718281828\cdots \\ \sqrt{7.4} & = & 2.720294102\cdots \end{pmatrix}$$

である．式 (27) は e と $\sqrt{7.4}$ の解が近く，悪条件の場合である．

収束の様子を図4に示す．横軸は Graeffe の方法による反復回数．縦軸は反復で求められた解と真値との相対誤差のうち最大のものである．

$p_1(x)$, $p_2(x)$ の両ケースともとも，IBM 表現では反復回数5回．IEEE 表現では反復回数7回であふれを発生する．

URR 表現では収束した後，さらに反復を繰り返すと相対誤差は急速に悪くなっていく．(4,16) 二重指数分割表現では収束までの振舞いは同じであるが，さらに反復を繰り返しても URR 表現より精度の悪化は緩やかかもしくは，悪化を生じない．その理由は以下のとおりである

Graeffe の方法は繰返しのたびに式の係数を二乗する演算を行う．二乗は相対誤差が2倍になるが，URR 表現では二重指数分割部が1ビット，等比分割部が1ビット長くなるため，等差分割部は2ビット短くなり，1ビット分精度を失ってしまう．しかし，(4,16) 二重指数分割表現では基本的に等比分割部が1ビット伸びるので☆，等差分割部は1ビット短くなる．そのため二乗による精度の悪化はほとんどないといえる．

また，演算を n 回繰り返したあとに，解を得るためには n 回の開平を行う必要がある．開平は相対誤差が半分になるので精度の低下は起きない．なぜなら URR 表現は開平を行うと二重指数分割部が1ビット，等比分割部が1ビット短くなるので等差分割部は2ビット長くなるためである．また，(4,16) 二重指数分割表現においても開平は最悪でも等比分割部が1ビット短くなるため，同様に精度の低下は起きない．つまり結果の最大相対誤差は最大係数の相対誤差と繰返し回数に影響される．

結果の最大相対誤差が悪化しない範囲は，最大係数の等差分割部の長さ☆☆に n ビット加えた等差分割部が解を十分精度良く表現できるときである．

また (4,4) 二重指数分割表現のときの収束の様子を図5に示す．この図から (4,4) 二重指数分割表現では (4,16) 二重指数分割表現のときよりも指数の絶対値の大きな数が精度良く表現できないため，解の悪化はより急速なものとなっていることが分かる．これは指数の絶対値の大きな数値の表現誤差が (4,16) 二重指数分割表現よりも悪いことに原因がある．

5.2 e の計算

次に1付近の精度が良いことがどう影響するのかを調べるために自然対数の底 e の計算を行った．用いた式は以下の式である．

☆ 2^{32}, 2^{512} 等を境に第1二重指数分割部が1ビット，等比分割部が1ビット伸びるのでこの場合は等差分割部は2ビット短くなる．

☆☆ 数値が非常に大きくなると等比分割部が最後まで終らず，結果，等差分割部行われない．しかしこの場合は等比分割部が終了するまでにあと何ビット要するかを負の値の等差分割部の長さとして換算する．たとえば，等比分割部の上限と下限の比が32で終了した場合，等差分割部は -5 ビットと換算する．

表2 e の計算結果の比較
Table 2 Comparison of calculation of e.

n	18
e	2..7182818284590452353
(4,16)	2.718281828459045<u>22402</u>
URR	2.718281828459045<u>22541</u>
IEEE	2.718281828459045<u>0908</u>
IBM	2.718281828459045<u>4</u>
n	19
e	2.7182818284590452353
(4,16)	2.7182818284590452<u>3234</u>
URR	2.7182818284590452<u>3234</u>
IEEE	2.7182818284590450908
IBM	2.718281828459045<u>4</u>
n	20
e	2.7182818284590452353
(4,16)	2.7182818284590452<u>3234</u>
URR	2.7182818284590452<u>3373</u>
IEEE	2.7182818284590450908
IBM	2.718281828459045<u>4</u>

$$e \approx 1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\cdots+\frac{1}{n!}+\cdots \quad (28)$$

$$=1+\frac{1}{1}\left(1+\frac{1}{2}\left(1+\frac{1}{3}\left(1+\cdots+\frac{1}{n}\left(1+\cdots\right)\right)\right)\right) \quad (29)$$

e は式 (28) のように展開されるが，(4,16) 二重指数分割表現の特長である 1 付近の精度が良いことを積極的に利用するために，式 (29) に従って計算を行い，IEEE 表現，IBM 表現，URR 表現との比較を行った．その結果を**表 2** に示す．下線で示す部分が本来の e の値と違う誤差の部分である．

(4,16) 二重指数分割表現は IEEE 表現や IBM 表現と比較して，$n=20$ のときで 2 桁精度が良い．また，(4,16) 二重指数分割表現は 1 付近において 1 ビットの精度を犠牲にしたが，従来の URR 表現と比較しても精度の悪さはほとんど見られないことが分かる．

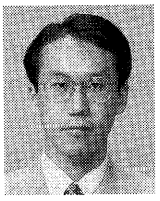
6. まとめ

本論文では URR 表現を一般化した二重指数分割に基づく数値表現を提案した．この一般化二重指数分割表現を用いて二重指数分割に基づく数値表現の表現精度を検証した結果，急速な精度の悪化を抑えた表現方法（$p=4$，$q=16$）があり，この表現は URR 表現よりも広範囲において精度が良いことが分った．また今後の研究課題として，一般化した二重指数分割の特長をいかした数値計算アルゴリズムの検討がある．

参考文献

1) 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).
2) 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).
3) 浜田穂積：新しい数値表現法 URR，第 25 回プログラミングシンポジウム，pp.84–91 (1984).
4) 浜田穂積：ディジタル・システムの数値表現法，*bit*, Vol.20, No.3, pp.299–315 (1988).
5) 中森真理雄，土井 孝：3 重指数分割による数値表現方式について，電子情報通信学会論文誌 A, Vol.J71-A, No.7, pp.1468–1469 (1988).
6) 横尾英俊：自然数の表現の立場から見た多重指数分割浮動小数点表示方式，情報処理学会論文誌，Vol.30, No.6, pp.792–794 (1989).
7) 横尾英俊：自然数の表現に基づく多重指数分割浮動小数点表示方式のクラス，電子情報通信学会論文誌 A，Vol.J72-A, No.12, pp.1998–2004 (1989).
8) 中森真理雄，萩原洋一，高田正之：変動式多重分割による自然数表現法，情報処理学会論文誌，Vol.31, No.6, pp.939–941 (1990).
9) 富松 剛：拡張した二重指数分割表現による数値表現法に関する研究，情報処理学会研究報告 95-HPC-58，Vol.95, No.97, pp.57–62 (1995).
10) 浜田穂積：数値表現法 URR の評価，コンピュータソフトウェア，Vol.1, No.3, pp.241–249 (1984).
11) 長嶋秀世：数値計算法（改訂 2 版），槙書店，pp.160–163 (1986).

(平成 9 年 5 月 15 日受付)
(平成 10 年 1 月 16 日採録)



富松 剛（学生会員）
1971 年生．1994 年工学院大学電子工学科情報工学コース卒業．1996 年東京大学大学院理学系研究科情報科学専攻修士課程修了．同年，同大学院博士課程進学．現在に至る．研究テーマは「数値表現」および「並列数値計算」．

金田 康正（正会員）：pp.519–528「分散メモリ型並列計算機による 2，3，5 基底一次元 FFT の実現と評価」（高橋 大介，金田 康正）を参照．