ハイパフォーマンス
コンピューティング 66－6
（１９９７．５．９）

新しい可変長指数部浮動小数点数表現形式の提案

須田 礼仁, 小柳 義夫
東京大学 理学部 情報科学科

可変長指数部浮動小数点数表現形式とは指数の大きさによって指数部の長さが変化することにより、通常の浮動小数点数よりも高い精度や広い表現範囲が実現できる浮動小数点数の諸形式である。本論文では従来の方式の問題点を考察し、誤差解析やハードウェアへ実装などを考慮して、表現形式の設計基準の中に (1) 変換にシフトが不要である (2) 指数部のビット長の変化が最大 1 であるの 2 点を盛り込むことを提案する。そしてこの基準を満たす表現形式をいくつか提示し、従来のものをも含めて精度や表現範囲を比較検討した。また、Kraft の不等式を用いた二重指数分割方式に関するいくつかの理論的な結果を示した。

New Floating Point Number Formats with Variable Length Exponent

Reiji Suda, Yoshio Oyanagi
The University of Tokyo

Some researchers proposed floating point number formats that attain higher precision and wider representation range than the usual formats by varing the length of the exponent. This paper consider the problems of those formats, and proposes to require that (1) encoding and decoding need no shifter and (2) the length of the exponent increases by at most one bit. Some new formats that meets those requirements are presented, and their precisions are evaluated. Some analytical results obtained from the Kraft's inequation about the double exponential cut formats are also presented.

§1 はじめに

これまでに数多くの浮動小数点数形式が提案されてきた。それぞれの形式は精度、表現範囲、ハードウェアやソフトウェアでの実装の容易さなどに違いがある。我々が注目したのは、Tapered Floating Point [1], 松井・伊理形式 [2]、浜田の URR [3, 5, 7] など、通常の浮動小数表現の指数部分 (全体が有限長であればそれに応じて仮数部も) の長さが指数の値によって変化し、通常の浮動小数点数形式に比べて 1 の付近では精度が高く、表現できる範囲も広くなっている諸形式である。本論文ではこれらをまとめて VLE (Variable Length Exponent) と呼ぶことにする。これらのうち最初の二つ、つまり 1971 年の Tapered Floating Point と 1980 年の松井・伊理形式 (以下、まとめて TFP と呼ぶ) はそれぞれの形式によって決まる指数部の長さの上限があり、全体のビット長によって「指数部長」を示すビットの長さが異なっていた。これに対して 1981 年の浜田による URR は全体のビット長によらずに表現形式が決まる点で画期的であった。URR に対して指数の絶対値が大きくなると速やかに精度が悪化するという問題が指摘され、これに対して表現形式の変更 [4, 13] や多重指数分割 [8, 9, 10, 12] によって改善が図られた (以下ではこれらの含めて URR と呼ぶ)。ハードウェアへの実装についても実験的なものがいくつかあるが、現在のところ商用計算機に広く行き渡るには及んでいない。

§1.1 これまでの VLE の問題点

VLE が実用化に至っていないのにはいくつか理由があると考えられる。最大の原因は戦略的な宣伝活動を行なわなかったことにあるかもしれない。しかし URR を計算用の数値表現として用いるためには、それ以外にも越えるべきハードルがいくつかあった。第一に、VLE を用いた計算の誤差解析の手法が確立していないことが挙げられる。VLE は精度が表現する数によって変化し、例えば IEEE 標準に比べて精度が高い範囲も存在する。しかし誤差解析の手法が確立していなかったため、VLE を用いた場合に計算精度がどのように変わるのか、どのような場合に得をしてどのような場合に損をするのかが明らかでなかった。しかも多くの VLE は IEEE 標準形式などに比べて精度が高い範囲が狭く、広い表現範囲を以外にはあまり利益がないと考えられてきたように思われる。理論的な立場で、あるいは実験を通して TFP の精度を通常の浮動小数点数表現と比較する研究も行なわれている [11] が、精度の比較だけでは誤差解析としては極めて不十分である。このような事情から VLE が実用に耐える精度を持った表現形式ではないという印象を持たれたため、実用機に VLE が採用されなかったのかも知れない。

第二に、ハードウェアでの実装が必ずしも容易ではなかったことが挙げられる。VLE は「標準的でない」浮動小数点数表現形式の中では比較的簡単で、ハードウェアの設計もそれほどの困難はない。しかし設計の容易さとハードウェアコストとは必ずしも一致しない。URR は形式の変換のためにシフタをかなりの数追加しなければならないが、シフタは乗算器の次にチップ面積を消費する回路である。そのため URR を採用するとチップ面積が増大して歩留まりを悪くするし、それを避けてチップ面積を以前と同じに収めようとすれば、何らかの別の回路を犠牲にしなければならなくなる。この問題は実用機での URR の採用の妨げになったかも知れない。

第三の問題は、これまでの URR では 1 異なる指数を表現するビット長が 2 ビット以上異なる場合があったことである。例えば浜田の URR では指数が −2 の時は指数部は 001 で 3 ビットの長さがあるが、指数が −3 の時は 00011 で 5 ビットあり、長さが 2

ビット異なる。このなんでもないようなことが、実は面倒な問題を引き起こすのである。まず、$(a+b)/2$ あるいは $a/2+b/2$ の計算結果が a と b の間に入らないという現象が起きる。例えば全体が 8 ビットとし、a のビット表現が 00010011, b が 00010001 とする（最初のビットは正の符号を表す）。このとき $a/2$ は 00001101 に、$b/2$ は 00001100 になって和は 00010100 となる。これは a と b のどちらよりも大きい。これは二分法などを使用する場合に重大な問題を引き起こす。同様の問題は、仮数部を 2 進数以外にとる場合にも発生する。ビット長の変化が 1 ビット以下であればこの現象は $a=b$ の場合にしか起こらないので、ちゃんとプログラムしてある二分法では問題にならない。ただし、この問題はソフトウェアで回避することができる。すなわち、$|a|<1$ かつ $|b|<1$ であれば $(a+b)/2$ を、$|a|>1$ かつ $|b|>1$ であれば $a/2+b/2$ を、それ以外の場合にはどちらかを使用するのである。しかし計算内容に比べてあまりに面倒な手続きである。

2 ビット以上の指数部長の変化がもたらすもう一つの問題は、絶対誤差が値の絶対値に関して単調でないということである。浜田の URR の例を続けるが、全体が l ビットであれば、指数が -2 の数の最下位ビットは 2^{-l+1} の桁であるが、指数が -3 の数の最下位ビットは 2^{-l+2} の桁になる。通常の浮動小数点数では、複数の値の絶対誤差の最大値を与えるのは、値の絶対値が最大値のものであったが、URR ではこれが成立しない。有名な Wilkinson の LU 分解の丸め誤差解析も、この原理に基づいて表示されているが、従来の URR ではこの結果が使用不能になってしまう。また、浮動小数点数の計算の最大の問題に桁落ちがある。しかし絶対誤差が絶対値に対して単調であれば、桁落ちが生じた時には計算結果の絶対値が小さくなっているため決して丸めは行なわれない。つまり、通常の浮動小数点数では桁落ちは相対誤差の分母が小さくなっているだけで分子は決して大きくなっていないのである。しかし URR では桁落ちに際して相対誤差と絶対誤差の両方が悪くなってしまうことがある。また、通常は総和演算は絶対値の小さいものから順に足すと丸め誤差が小さいと期待されるというのは良く知られている。しかし、絶対誤差が絶対値に対して単調でないと、途中の和が絶対誤差が大きいところにぶつかってしまうとこの結論も成り立たないことになる。さらに、丸め誤差を補償して総和演算をほぼ 2 倍の精度で計算する方法があるが、これも桁落ちでは丸めがないことを利用しているので URR では使えない。固定長の URR しか使えないハードウェアで多倍長の計算を行なう場合にも同様の問題が生じ、通常の浮動小数点数で使うことのできる技法が通用しなくなる。

以上のように、URR は計算用の数値表現としてはいろいろと問題を含んでいる。URR を提案した浜田氏自身、数値計算に使用するよりもデータを交換する際の標準形式として価値が高いと考えておられるようである。しかし VLE の特徴である 1 付近での高い精度と広い表現範囲とは、うまく利用すれば数値計算でも有効であると期待される。我々は上記のような考察に基づいて、VLE の設計基準を見直すことにした。本論文ではその設計基準とそれを満たす表現形式、精度の比較評価などについて述べる。実際には誤差解析をどうするかが最大の問題であると認識しているが、本論文では絶対誤差の近似表示と、誤差解析の基本方針について簡単に述べることしかできない。誤差解析については今後研究を進めていく予定にしている。

§2 新しい VLE の設計基準

§2.1 従来の設計基準

従来の URR の設計基準は、事実上浜田 [5] で提示されたものに従っていた。これを一部簡略化して引用すると、

1. 指数・仮数の 2 進数表現と浮動小数表現との間の変換が容易であること。
2. あふれが事実上起こらないこと。
3. 仕様がデータ長に依存しないこと。
4. 形式を決定するパラメタが少ないこと。
5. すべてのパタンが異なる数値に対応すること。
6. 値の大小関係が辞書順に一致すること。

となる。表現形式を VLE に限ればこれらの基準のうち 1 と 2 は必ず満たされる。その他のものも望ましい性質ではあるが、それを採用すると別の点で不利になるのであれば捨てても良いと我々は考えた。

まず基準の 3 であるが、これは TFP が満たさない。TFP は「指数部の表現効率が最も良い」形式であるが、実際に「指数部の表現効率が最も良い」のは指数の絶対値が充分に大きい場合であって、実用上問題になる範囲で TFP が基準 3 を満たす他の形式に比べて精度が高いとはかならずしも言えない。さらに、指数が一定の範囲にある場合に例外的な表現を許すことにすれば、基準 3 を満たす表現形式の範囲は事実上いくらでも広げることができる (URR でも指数が 0 と -1 の場合には例外的な処理をしなければならない)。このように条件を緩めれば、基準 3 は設計基準として採用してなんら不都合はない。しかし必要な表現範囲が限られている場合には、基準 3 は表現形式の定義を多少簡単にする以外の役には立たない。このため、無限に広い表現範囲が要求されている場合には基準 3 を採用し、有限の表現範囲が要求されている場合には却下するのが良いと考えられる。

基準の 4 はいわば美学の問題である。これは確かに望ましい性質ではあるが、形式を極めて単純なものに制約するように働くので、さまざまな表現形式を比較検討する妨げになるように思われる。このため、この基準は採用しない方が良いと考えられる。

基準の 5 を満たさない形式では「無駄な」ビットパタンが生じる。このような形式では表現効率が悪くなっているので、無駄なパタンを活用すればより精度の高い表現が得られるはずである。そこで今回はこの基準を採用することにした。しかし無駄なビットを許すと、指数長の変化を 1 ビット以内に抑えることが容易に実現できる。このためこれについては今後検討する必要がある。

基準の 6 は棄却する。この条件を満たすためには指数部は仮数部の上位になければならない。しかしこの配置では指数・仮数の 2 進数表現と浮動小数表現との変換の際にシフタが必ず必要になる。指数・仮数の 2 進数表現では指数の最下位ビットと仮数の最上位ビットとが固定されているのに、基準 6 を満たす浮動小数表現では逆であるからである。前述の通り、シフタはハードウェアコストが比較的高い回路である。そこで、我々は敢えてこの基準 6 を棄却する。そしてシフタを必要としないように、指数部を仮数部の下位に配置することにする。

しかしこの選択にはいろいろと欠点もある。この配置では、大小比較を行なうには浮動小数点数を指数・仮数の 2 進数表現に変換しなければならなくなってしまう。このため比較器は他の表現よりもかなり複雑で、時間が倍、チップ面積が 3 倍程度必要となると予想される。また、従来の URR の配置であれば、長さの異なるデータ間の変換が極めて簡単であったが、我々の提案する配置の場合にはやはり指数部と仮数部を一旦分離しなければならない。この意味では、我々の提案する配置では強い意味での表現のデータ長独立性が失われていると考えられるかも知れない。このような欠点があるため、我々の選択に対して反対意見もあることであろう。しかし、いずれにせよ整数と同じ比較器とシフトのない変換器のどちらかを選択しなければならない。我々は後者を選択したということである。

§2.2 新しい設計基準

我々は以上のように、これまでの VLE の持っていた問題点 (のいくつか) と、これまでの VLE の設計基準とを評価した。これに基づいて、我々は新しい VLE の設計基準として、以下のものを提案する。基準は以下に示す 6 つあり、そのうち最初の 3 つは必須条件で、あとの 3 つは要望条件である。

1. 表現形式は指数・仮数の 2 進数表現に直接基づくものであること。
2. 指数は 2 進数としての指数であること。1 異なる 2 つの指数を表現するビット長の差は 1 以下であること。
3. 変換にシフタを必要としないこと
4. 同じビット長の一般的な浮動小数点数表現形式に比べ、1 の付近の広い範囲で精度が高いこと。
5. 固定長の場合、すべてのパタンが異なる数値に対応すること。
6. 仕様がデータ長に依存しないこと。

従来の VLE のうち、TFP は 3 つの必須条件と 5 を満たしており、4 もそれなりに評価できる。URR は必須条件である 2 と 3 が満たされていないが、5 と 6 を満たしており、4 もさまざまな手法で改善されている。

以下ではこの設計基準を満たす VLE をいくつか紹介して比較評価を行なうが、提案する設計基準を満たす VLE は無限に存在し、本論文で紹介するものはそのいくつかの例に過ぎない。論文の題名は VLE の提案となっているが、本節で説明してきたこの設計基準の提案が、実際には本論文の最大のポイントである。

§3 新しいデータ長独立 VLE の構成

この節では我々の提案する 6 つの設計基準を満たす新しい VLE を構成する。ただし、基準 4 は相対的な問題なので、基準 1–3 および 5, 6 を満たすものをいくつか提案し、それらを後に基準 4 の視点で比較を行なうということにする。

§3.1 方式 1 (2 進数) の定義

すでに説明したように、変換にシフトが必要なくなるためには指数部が仮数部の下位に置かれていなければならない。よって、形式の全体としては最上位に符号 1 ビット、最下位に可変長の指数部、その間の残りの部分が仮数部が配置される。仮数は $[1,2)$ の範囲に正規化され、指数部は指数の 2 の補数表現に基づいて定義される。

まず一つの指数部の表現形式を定義する。これを仮に方式 1 と呼ぶことにする。

指数部は下位から順に、開始部、中間部、終端部からなる。開始部は 2 ビットからなる。下の 1 ビットは指数の符号を表す。上の 1 ビットは指数が -3 から 2 の範囲の時には 1, それ以外では 0 である。これを以下では U ビットと呼ぶ。この U ビットが 1 (これを $U = 1$ と書くことにする) の時は中間部はなく、開始部の直後に指数の値によって決まる終端部がすぐに続く。$U = 0$ の時には、終端部は指数の 2 の補数表現での最上位の数ビットから決定される。終端部をビットパタンを表 1 に示す (表をコンパクトにするため、指数を E, 指数の最上位の数ビットを MSE と略す)。$U = 0$ の場合に、指数の 2 の補数表現のうち、終端部を決定するのに使用されなかった下位のビットがある場合は、それを $b_{l-1}b_{l-2}\cdots b_1b_0$ とする。このとき中間部はこれらのそれぞれのビットの前に 0 を挿入して $b_{l-1}0b_{l-2}0\cdots b_10b_00$ と定義される。

表 1. 方式 1 の終端部

$U=1$		$U=0$	
E	終端部	MSE	終端部
2	10	101	101
1	00	100	001
0	1	11	11
-1	0	00	01
-2	11	011	111
-3	01	010	011

以上が方式 1 の定義である。中間部はちょうど浜田の URR の指数部長を示すビット列と指数を示すビット列をインタリーブさせたようになっている。これにより、指数の 2 の補数表現で下から k ビットめのビットがいつでも指数部の $2k+2$ ビットめにあるようになり、変換にシフトが不要となった。中間部の長さは 2 の倍数であるが、中間部の長さが 2 ビット長くなる時には終端部が 1 ビット短くなるので、指数部全体の長さの変化はいつでも 1 ビット以内に収まる。指数部の表現はデータ長に依存しないことは明らかであるし、使用していないビットパタンが存在しないことも証明可能である。以上のことから、この方式 1 は我々の提案する設計基準を (比較の必要な 4 を除いて) 満たすことがわかる。

§3.2 方式 2 (4 進数) と方式 3 (8 進数) の定義

方式 1 はさまざまな方法で変更を加えることができる。一つの重要な方法は、中間部の形式を変更することである。指数 e が充分大きい範囲では、指

数部の長さはほぼ $2\log_2 e$ になるが、この係数 2 は中間部の形式で決まっている。方式 1 では終端部に使用されなかった指数の 2 の補数表現のビットの間すべてに 0 を挿入したため、指数部の長さがほぼ 2 倍になった。この 0 の挿入を 2 ビットごと、あるいは 3 ビットごとにすることで指数部の長さを短くすることが可能である。これらは指数を 4 進数、あるいは 8 進数で処理していると考えることもできる。このような改良は URR に対して [4, 13] で行なわれているものと本質的に同じである。

表 2. 方式 2, 3 の終端部 (指数が正の部分)

方式 2				方式 3			
$U=1$		$U=0$		$U=1$		$U=0$	
E	終端部	MSE	終端部	E	終端部	MSE	終端部
7	00001	111	00001	7	000001	111	000001
6	10001	110	10001	6	100001	110	100001
5	11001	101	1001	5	10001	101	10001
4	01001	100	101	4	1001	100	1001
3	1101	11	111	3	1101	11	111
2	0101	10	011	2	0101	10	101
1	111			1	111	1	011
0	011			0	011		

このように指数を 4 進数または 8 進数として中間部を定義する方式をそれぞれ方式 2, 方式 3 として定義する。これらの方式では開始部は指数の符号の 1 ビットのみとなった。ただし U ビットは符号のすぐ上のビットとして同様に定義される。中間部は終端部に入らなかった指数の 2 の補数表現のビット列に、2 ビットまたは 3 ビットおきに 0 を挿入する。終端部は指数が正の部分については表 2 で定義する。指数が負の部分も方式 1 のように、正の部分に準じて定義すれば良い。これらの方式も方式 1 と同様の性質を持っており、設計基準を満たしていることがわかる。

§3.3 各方式の比較

表 3. 指数部長と表現可能な最大指数

l	TFP	方式 1	方式 2	方式 3	浜田	IEEE
2					0	
3		0			1	
4		3	1	1		
5		5	3	4	3	
6		7	7	5		
7	1	11	19	7	7	
8	3	15	23	31		127
9	7	23	31	39	15	
10	15	31	79	47		
11	31	47	95	63	31	1023
12	63	63	127	255		
13	127	95	319	319	63	
14	255	127	383	383		
15	511	191	511	511	127	
16	1023	255	1279	2047		

この節では設計基準の 4 である、他の表現との精度の違いを比較評価する。表 3 は設計基準の必須条件を満たす TFP (69 bit) と提案した 3 方式および参考にするための浜田の URR と IEEE 標準形式について、指数部長 l とその時に表現できる最大の指数との関係を示して比較している。

比較の仕方はいくつか考えられる。第一に、特定の長さまでの指数部で表現できる範囲である。表 3 はこれを示しているが、問題は「特定の長さ」として何を採用するかである。数値計算を想定するならば、IEEE 標準の指数部長である 11 ビットというのは意味があるであろう。ここで比べると、方式 2 が他より優れている。ちなみにここで得られた 95 という値は、参考文献に挙げた研究のどの方式よりも大きい。これは提案方式では最短の指数部長が比較的長いことに関係があり、さらに指数部長の最小値を大きくすれば 11 ビットで表現できる範囲をもっと広げることも可能である。VLE は 16 ビット以下などかなり短いデータ長でもそれなりの精度と表現範囲を提供できる点でも重要で、このような性質は DSP などのアナログ情報処理で有効に利用できるものと考えられる。このような場合には 15 ビット、11 ビットなどが考えられるが、この場合にはデータ長にあわせて最短指数部長を 5 に定義しなおした TFP が最も良い。

第二の比較は、特定の範囲の指数を表現するのに必要なビット長である。これはある範囲のアプリケーションを想定する場合に意味があると考えられる。仮に IEEE 標準の単精度、倍精度で要求されている 127 と 1023 を当てはめてみると、最小になるのはそれぞれ 12 ビットおよび 16 ビットで、方式 2 と 3 がこれを満たす。アナログ処理を想定して 15 としてみると、やはり方式 2 が最小の 7 ビットを与えている。しかし、URR の特徴の一つであった「固定小数点数の代わりに用いても 1 ビットしか精度が劣らない」という視点から、0 (つまり最短指数部長) を取ってみると、浜田の方式が圧倒的に優れている。実は、最短指数部長が 2 となる設計基準を満たす指数部の表現方式は、浜田の URR で指数部長を表すのに用いられている、いわゆる 1 進数表記しかないことが簡単にわかる。しかもこの最短指数部長は指数部が大きいところでの指数部長と密接な関係があって、極限での表現効率が良い形式では必然的に最短指数部長は大きくなるのである。ただし設計基準の 2 と 4 を満たさない場合はそうとは限らない。従来の URR が実際そうであったし、方式 2 を少し変更して無駄なパタンを許すことにすれば最短指数部長を 3 にすることは容易である。

以上の比較からすると、表 3 による比較では方式 2 が他のものよりも優れているという結論になりそうである。方式 3 の方がよいデータを出している部分も多少あるが、本論文で比較した方式の中ではやはり方式 2 が最良と言って差し支えないであろう。ただし設計基準を満たす表現形式はここでは取り上げたものがすべてではない。

§3.4 Kraft の不等式

我々の提示した設計基準を満たす VLE は、ここで提示したものですべてではない。そのようなものはこれ以外にも無限に存在しており、ここでは、いわば最も簡単なものを例示したに過ぎないのである。

実用を目的として VLE を定義しようという場合には、どれだけの範囲の数がどれだけの指数部長で表現できるように設計するかについて徹底的に検討する必要があるであろう。この際制約となるものに [10] などでも利用されている Kraft の不等式がある。指数部長 x の指数が $n(x)$ 個あるとする。このとき最短指数部長を β, データ長 (のうち指数部に使用できるもの) が L ビットであるとすると

$$\sum_{x=\beta}^{L} n(x)2^{L-x} \leq 2^L$$

でなければならない。これが精度と表現範囲との関係の制約となる。

二重指数分割を用いる場合には、指数部長 x での最大指数はおよそ $2^{(x-\beta)\gamma}-1$ のようになる。ここで β は最短指数部長、γ は 2 進数表現に対する漸近的な表現効率で、TFP では $\gamma=1$, 浜田の URR や方式 1 では $\gamma=1/2$, 方式 2, 3 ではそれぞれ $\gamma=2/3$, $3/4$ である。この式と Kraft の不等式から

$$n(x) = 2^{(x-\beta)\gamma}\left(1-2^{-\gamma}\right)$$

となる。これと Kraft の不等式を合わせると、二重指数分割による VLE に関してさまざまなことがわかる。まず $\gamma<1$ では $L\to\infty$ とすることができて、URR が可能である。Kraft の不等式は

$$\frac{1-2^{-\gamma}}{1-2^{\gamma-1}} \leq 2^{\beta}$$

となる。これから、効率 γ が下から 1 に近付くと分母が 0 に近付き、最短指数部長 β が大きくなることがわかる。γ が 1 に近いほど指数が大きい場合の表現範囲は広がるわけであるが、その場合には指数が小さいところでの精度は落ちるというトレードオフが存在することがわかる。そして $\gamma=1$ では

$$\beta \geq \log_2(L-\beta+1)-1$$

となって β は L に依存し、URR はできないことが証明できる†。

$\gamma<1$ の場合、指数部長 x 以下で表現できる最大の指数は

$$2^{x\gamma}\left(\frac{1-2^{-\gamma}}{1-2^{\gamma-1}}\right)^{-\gamma}-1$$

となる。表現範囲を示しているこの式は単調関数ではなく、IEEE の単精度・倍精度に対応する $x=7$ と $x=10$ を代入してみると、$\gamma=i/(i+1)$ という形のものでは $\gamma=4/5$ で表現範囲が最大になるという興味深い結果を得る。ただしこれは解析的な結果で、2 進数表現としてこのようなものが存在するかどうかは、特に我々の設計基準を課する場合には、別の問題である。

ここで $n(\beta)$ のみ特別扱いをして $n(\beta)=n_0$ とおくと、長さ x 以下で表現できる指数の最大は

$$n_0+2^{(x-1)\gamma}\left(n_0+\frac{2^{\frac{1}{\gamma}}-1}{2-2^{\frac{1}{\gamma}}}\right)^{-\gamma}-1$$

となる。ここから $\gamma<1$ の範囲では、n_0 を大きくする (イメージ的には、底を広くする) と表現範囲が広くなることがわかる。しかしこのとき

$$n_0+\frac{2^{\frac{1}{\gamma}}-1}{2-2^{\frac{1}{\gamma}}} \leq 2^{\beta}$$

という関係があることから、最短指数部長は長くなっている。ここでも、表現範囲を広げようとすると最短指数部長が大きくなってしまうということがわかる。

なお、二重指数分割による VLE の絶対誤差はここで用いた γ と $\varepsilon=2^{\beta-L}$ を用いて

$$E(x) \approx \varepsilon|x||\log_2|x||^{\gamma}$$

と表現できる。この式は $|x|$ が $[1/2,2)$ の範囲ではかなりの誤差を含んでいるが、誤差の最小化を考える場合にはかなり役に立つ重要な式である。

§3.5 実用計算のための精度と表現範囲

実用計算を考えた場合には、指数が比較的小さい範囲では精度が、指数が大きいところでは表現範囲が問題であるはずである。このことは、指数が小さい範囲では高い精度を維持するために TFP のように $\gamma=1$ が要求されるが、指数が大きいところではデータ長独立性が得られる $\gamma<1$ が望ましいことを意味するのではないだろうか。多重指数分割を用いると指数が大きくなると γ が 1 に近付くが、これは方向性が逆であるように思われる。

指数が小さい範囲では $\gamma=1$ で、大きくなると $\gamma<1$ となるような表現方式をここでは考える。$\gamma=1$ の範囲では指数部長 x に対して最大指数が $2^{x-\beta}-1$ のようになるが、指数が 1023 までこれが成立する最小の β は Kraft の不等式を用いると 3 (指数の符号を含めると 4) となることがわかる。

この条件を満たす方式 4 を以下のように定義することができる。指数が 0 から 3 の範囲では表 4 の通りの指数部表現を用いる。指数が 4 から 1023 までは最下位 2 ビットを 10 とし、その上 3 ビットに指数部長から 7 を引いたもの、その上に指数の 2 進数表現から常に 1 である最上位の 1 ビットを除いたものを置く。指数が 1024 以上では最下位 3 ビットを 100 とし、その上に指数の 2 進数表現の下 9 ビットを配置、

表 4. 方式 4 の指数部

E	指数部
3	111000
2	011000
1	01000
0	0000

† これは二重指数分割の場合の解析で、多重指数分割を行なえば漸近効率が 1 となる URR は可能である。しかし、多重指数分割を行なうと指数部長の表現が複雑になり、設計基準 3 を満たすためには、指数部長のビット長の変化の種類に従って複数の終端部を定義しなければならないという問題がある。

さらにその上に指数の 2 進数表現の上位を方式 2 で配置する。この方式は設計基準をすべて満たしており、また指数が 1023 以下では $\gamma = 1$, それ以上では $\gamma = 2/3$ となっている。

この表現での指数部長と最大指数との関係の主なものを表 5 に示しておいた。指数が 127 および 1023 となるのはそれぞれ 11 および 14 ビットで、表 3 で比較した他の方式よりもかなり効率がよいことがわかる。指数部長が 8 以下では方式 2 の方が方式 4 よりも若干精度が高い。この部分について方式 4 の精度を改良することは不可能ではない。しかし、あまり表現形式の定義が複雑になるのも問題である。

表 5. 方式 4 の精度

l	最大指数
4	0
8	15
11	127
14	1023
15	2559
16	3071

この方式 4 は TFP と URR のあいの子のような、あまりきれいではない方法である。しかし指数が 1023 以下ではかなりよい成績を示し、それ以上ではデータ長独立で広い表現範囲を与える。このように、重要な部分の精度とデータ長独立性とをなんとか両立させており、実用計算に使用するにはこのような形式が必要であるのかも知れない。

方式 4 は 32 ビットでも 10 進数で 29,072,700 桁の数を表すことができる。しかし、本当にこのような数を表現する必要があるのであろうか。そもそもこのような数の必要性と実装のコストとが釣り合わなかったために VLE は実用化に至っていないのかも知れない。少なくとも、10 進 1 億桁の数が扱える浮動小数点数よりは、10 進 30 桁の精度の浮動小数点数の方がはるかに有用であるに違いない。ハードウェアの実装を考えても、指数部の長さが抑えられている方が設計に関してもコストに関しても有利である。本当に有用な指数の範囲はどれほどのものであるのか、今後議論を深めてゆく必要があると思われる。

§4 おわりに

本研究では、VLE の持っている利点を実用計算に利用すべく、従来の VLE を実用化する場合の問題点を改めて考察し、それに基づいて新しい VLE の設計基準を提案した。論文の題名は VLE の提案となっているが、むしろこの設計基準の提案が本論文の最大のポイントである。その特徴は、誤差解析をやりやすくし実用上の問題を避けるために指数部長の変化を 1 ビット以下に制限したことと、ハードウェア実装におけるコストの軽減のために変換にシフトを必要としない形式を要求したことである。本論文ではこれを満たす URR を 3 つ提示し、高い精度が得られる表現範囲を比較した。その結果、提案方式の一つである、指数を 4 進数で扱った方式 2 が相対的に優れているという結果を得た。また、Kraft の不等式を利用して、二重指数分割による VLE の精度と表現範囲とのトレードオフについて論じ、絶対誤差の簡略化した表示を示した。また、TFP と URR を組み合わせた方式について考察した。

今回は誤差解析については述べなかった。Demmel [6] も指摘しているように、VLE では「任意の入力に対して一定以上の精度の結果を出す」という意味で「信頼できる」アルゴリズムを作ろうとすると、指数の範囲を制御するような複雑なコードがさけられない。このため VLE においては目標を「任意の入力に対して精度の評価つきで結果を出す」あるいは「指定された範囲の入力に対して一定以上の精度の結果を出す」のように設定し直すべきであろう。このような立場に立って今後 VLE の誤差解析の研究を進めてゆきたいと考えている。

参考文献

[1] Robert Morris, "Tapered Floating Point: A New Floating-Point Representation," *IEEE Trans. Comp.*, Vol. 20, No. 12, (1971) pp. 1578–9.

[2] 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, (1980) pp. 306–313.

[3] 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, (1981) pp. 521–526.

[4] 高田正之、西村恕彦、「浜田方式数値表現の変形とその評価」、情報処理学会第 26 回全国大会、(1983) pp. 1231–1232.

[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, (1983) pp. 149–156.

[6] J. W. Demmel, "On Error Analysis in Arithmetic with Varing Relative Precision," *Proc. 8th Symp. Comp. Arith.*, (1987) pp. 148–152.

[7] H. Hamada, "A New Real Number Representation and Its Operation," *Proc. 8th Symp. Comp. Arith.*, (1987) pp. 153–157.

[8] 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, (1988) pp. 1468–1469.

[9] 中森真理雄、「変動式多重指数分割による数値表現方式」、電子情報通信学会論文誌 A, Vol. J72-A, No. 6, (1989) pp. 1009–1011.

[10] 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, (1989) pp. 792–794.

[11] A. M. Azmi and F. Lombardi, "On a tapered floating point system," *Proc. 9th Symp. Comp. Arith.*, (1989), pp. 2–9.

[12] Hidetoshi Yokoo, "Overflow/Underflow-Free Floating-Point Number Representation with Self-Delimiting Variable-Length Exponent Field," *IEEE Trans. Comp.*, Vol. 41, No. 8, (1992) pp. 1033–1039.

[13] 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、(1995) pp. 57–62.