日本応用数理学会論文誌
Vol. 9, No. 2, 1999, pp. 51～64

乗法的浮動小数点除算アルゴリズムの誤差限界モデル

網崎　孝志*　　長嶋　雲兵**
* 島根大学総合理工学部数理・情報システム学科　　** 物質工学工業技術研究所

Accuracy Models for Floating-Point Divide Operation That Uses Multiplicative Algorithms

Takashi Amisaki*　　Umpei Nagashima**
* Shimane University　　** National Institute of Materials and Chemical Reactions

Abstract. This report compares the behaviors of three multiplicative algorithms for floating-point divide operation. They are the Newton-Raphson method, Goldschmidt's algorithm, and another method that simply evaluates the Taylor series expansion of a reciprocal. Goldschmidt's algorithm is based on the same series but differs from the alternative method in a manner of evaluating the series. The behaviors of the three methods are compared using two kinds of models for each method: a performance model, which describes latency, and an accuracy model, which describes the upper bound for the error of the quotient. Particular emphasis is placed on development of the accuracy models. Validity of the accuracy models were empirically verified with numerical tests. It is shown that, with a practical choice of the number of iterations, k, the magnitude of the relative error is bounded by 3×2^{-p}, $(2k+1)\times2^{-p}$, and $(k+1)\times2^{-p}$, where p is the size of the mantissa represented in bits, for the Newton-Raphson method, Goldschmidt's algorithm, and the alternative method, respectively (results on a floating-point unit with a multiply-add-fused configuration). For the Newton-Raphson method, any number of additional iterations further reduce the bound to $(8/3)\times2^{-p}$. The performance models indicate that, on a pipelined floating-point unit, Goldschmidt's algorithm and the alternative method are equally faster than the Newton-Raphson method. As a result, in general, the alternative method is promising in the case of $k \leq 3$, a very realistic value.

1　はじめに

浮動小数点除算は, ほとんどすべての科学技術計算に必要な基本演算である. 一般に, 加減算や乗算に比べると, その出現頻度は低いが, それでも, 除算性能がシステム全体の性能に大きな影響を与えることが報告されている [9]. このため, 現代の汎用マイクロプロセッサの多くは, その浮動小数点演算部 (floating-point unit; FPU) に, 除算を高速に行うための何らかのハードウェア的措置が講じられている [11]. また, 除算性能のより一層の向上のため, 数多くの研究がなされている (例えば, [10] 参照). ところで, FPU の設計にあたっては, 「面積・性能トレードオフ」が極めて重要である [11]. すなわち, システム全体の性能を高めるためには, 限られたチップ面積を考慮した上で, 加減算, 乗算, 除算をバランスよく実現する必要がある. 汎用の FPU を設計する場合には, この最適配分を決定することは非常に難しい問題であるが, 応用分野が特定されている場合は, その作業が比較的容易となることがある. 例えば, 非経験的分子軌道シミュレーションには莫大な量の浮動小数点演算が必要となるが, 除算の出現頻度は極めて低い. このため, このシミュレーションのためだけの特別用途の FPU を設計するのであれば, 除算性能を追求するよりも, 乗算器と加算器の性能を少しでも向上させるほうが得策となる. 実際, そのような設計方針のもとに, 専用目的の並列計算機が設計されている [8].

そして, このように, アプリケーションに特別なプロセッサを開発することは, 決して例外的なことではなく, 大規模数値計算の高速化のための有効な手段のひとつとして注目されている.

浮動小数点除算のアルゴリズムは, その主演算が減算に基づくものと, 乗算に基づくものの二つに大別される [10,11]. 後者を乗法的アルゴリズムというが, こちらを利用した方が除算のための特別なハードウェアのコストが少ないか, あるいは場合によっては, 不要となる. これは, 乗法的手法は基本的には, どの FPU にも備えられている 乗算器 と 加算器 を使って実現されるからである. 代表的な乗法的手法としては, Newton-Raphson 法と Goldschmidt のアルゴリズムの二つがある [3,4,10,11]. 前者は, 例えば, IBM RS/6000 の FPU [7], 後者は Sparc システム用浮動小数点演算コプロセッサ TMS390C602A [2] で採用されている. 無限精度の演算を行えば同一の商が得られるという意味で, 両手法は数学的に同一である. しかし, 浮動小数点演算において両者は異なる振舞いを示す. すなわち, パイプライン化された FPU においては, 後者の方が高速である. その反面, 概して, 前者の方がより正確な商を与える. 一般に, Newton-Raphson 法は, 結果の正確さに関して "自己補正的" であるといわれている. すなわち, "丸め誤差が蓄積する" ことなく, 反復回数を増やせば解の精度が向上する. 一方の Goldschmidt のアルゴリズムは, 逆数の Taylor 級数展開を計算するためのひとつ方式である. Goldberg の解説 [4] にあるように, この級数を素直に計算したのでは, 精度上問題があると考えられている. Goldschmidt のアルゴリズムは, この点を改善するための手法であり, ある意味で弱い自己補正能力をもつ手法である.

本報告では, 除算のための特別なハードウェアを備えていない FPU 上での, Newton-Raphson 法, Goldschmidt のアルゴリズム, ならびに Taylor 級数を素朴に計算する手法という三種の乗法的アルゴリズムの振舞いを解析する. 各手法本来の基本的性質を解明するため, ここでは IEEE 規格 754 [5] の要請を考慮しない. すなわち, 得られる商は, 無限精度の除算結果に近ければ, 必ずしもそれを正しく丸めたものでなくともよいとの立場をとる. このようにして開発されたアルゴリズムは, 除算機能を備えていない FPU を用いてソフトウェア除算を行う場合などに, 直ちに 利用でき, 前述のようなアプリケーションにおいて有効である. さらに, 三種のアルゴリズムとも, 文献 [10] に紹介されているような手法を用いて, IEEE 丸めをおこなうように改変することも可能である. その場合にも, 本報告で述べる結果は非常に有益である.

本報告では, 各手法の除算速度を表す "性能モデル" と, 商の正確さを表す "誤差限界モデル" を構築し, 後者については数値実験によりその妥当性の検証を行う. 特に, この誤差限界モデルは, 極めて単純なものではあるが, 現実的な場面における各手法の精度を端的かつ精密に説明することができる. これらのモデルを用いて, FPU の構成によっては, Taylor 級数を素直に計算するほうが, Newton-Raphson 法や Goldschmidt のアルゴリズムよりも有利になる場合があることを示す. 一般に, 乗法的手法を用いたハードウェア除算には, Newton-Raphson 法や Goldschmidt のアルゴリズムが用いられており, 我々の知識の範囲内では, Taylor 展開に対する素朴な手法の性能を FPU レベルで解析した結果は報告されていない. ここに示す結果は, 前述のような状況設定においてはもちろんのこと, 乗法的アルゴリズムのハードウェア実現においても非常に有用であると思われる.

2 アルゴリズム

二種類の FPU の上での, 浮動小数点数の除算 $q = a/b$ を考える. ひとつは乗算器と加算器が分離した構造をもつ FPU であり, 乗算と加算が独立に行える. この構成を分離型 (independent-add-and-multiply; IAM) という. もう一方は, いわゆる積和器構造をもつ FPU であり, その主演算は, 三つのオペランドのうちの二つに対して乗算を行い, その結果と残りのオペランドとの加算を行うというもので, この複合演算は不可分に行われる. この構成を積和型 (multiply-add-fused; MAF) ということ

(A) procedure Newton_IAM
(0) $x_0 \leftarrow r_b$
(1) for $i = 0$ to $k-1$ do
(2) $\quad s_i \leftarrow b \times x_i$
(3) $\quad s_i \leftarrow 2 - s_i$
(4) $\quad x_{i+1} \leftarrow x_i \times s_i$
(5) endfor
(6) $x_k \leftarrow a \times x_k$
end.

(B) procedure Newton_MAF
(0) $x_0 \leftarrow r_b$
(1) for $i = 0$ to $k-1$ do
(2) $\quad s_i \leftarrow 2 - b \times x_i$
(3) $\quad x_{i+1} \leftarrow 0 + x_i \times s_i$
(4) endfor
(5) $x_k \leftarrow 0 + a \times x_k$
end.

Fig. 1. Pseudo-code for the Newton-Raphson method on each FPU configuration. In either case, the quotient is finally obtained as x_k. (A) Independent-add-and-multiply configuration. (B) Multiply-add-fused configuration.

にする. なお, いずれの構成においても第 1 章で述べたような制約をおく. すなわち, FPU は乗算器と加算器で構成され, 除算のための特別な機構は備えてはないとする. 本章の後半で行う性能解析はこの制約の影響を受けるが, 第 3 章に述べる誤差解析の結果はこの制約に左右されない.

Newton-Raphson 法

除算 a/b は, $(1/b) \times a$ として計算することができる. そこで, 方程式

$$f(x) = 1/x - b = 0$$

の根 $x = 1/b$ を Newton-Raphson 法により求め, その結果に a を乗じて a/b を得る. $1/b$ を計算するには, 適当な初期近似値 $x_0 \leftarrow r_b \simeq 1/b$ から出発して, 以下の Newton ステップを繰り返す:

$$x_{i+1} \leftarrow x_i - \frac{f(x_i)}{f'(x_i)} = x_i \times (2 - b \times x_i). \tag{2.1}$$

r_b の値はテーブル参照により求める. 分離型および積和型いずれの FPU においても, 式 (2.1) に忠実に除算の手続きを実現すると, Fig. 1 に示した疑似コードのようになる. いずれのコードでも商 q は x_k に与えられる. 今後示すコードでも同様である. また, ここに示したように, 積和型の FPU では加算や乗算は単独では行えず, 積和演算のみが利用できるとする.

Goldschmidt のアルゴリズム

初期値 r_b に相対誤差 ξ が含まれるとする. すなわち, $r_b = (1/b)(1+\xi)$ である. これは, b ではなく $b/(1+\xi)$ の正確な逆数である. そこで $r_b \times a/(1+\xi)$ とすれば a/b を求めることができる. この $a/(1+\xi)$ を $\xi = 0$ のまわりで Taylor 級数に展開すると

$$r_b \cdot \frac{a}{1+\xi} = \frac{a}{b}(1+\xi)(1-\xi)(1+\xi^2)(1+\xi^4)\cdots(1+\xi^{2^{k-1}})(1+\xi^{2^k})\cdots \tag{2.2}$$

を得る. Goldschmidt のアルゴリズムは原理的にはこの展開に基づいた手法であるが, 実際には以下のようにして a/b を計算する. 分母 b と分子 a それぞれに初期値 $r_b = (1/b)(1+\xi)$ を乗じ, その結果をそれぞれ, y_0, x_0 とおく. すなわち,

$$y_0 \leftarrow b \times r_b = 1 + \xi, \qquad x_0 \leftarrow a \times r_b = \frac{a}{b}(1+\xi).$$

(A) procedure Goldschmidt_IAM
(0) $x_0 \leftarrow r_b$
(1) $y_0 \leftarrow x_0 \times b$
(2) $x_0 \leftarrow x_0 \times a$
(3) for $i = 0$ to $k-1$ do
(4) $\quad s_i \leftarrow 2 - y_i$
(5) $\quad y_{i+1} \leftarrow y_i \times s_i$
(6) $\quad x_{i+1} \leftarrow x_i \times s_i$
(7) endfor
end.

(B) procedure Goldschmidt_MAF
(0) $x_0 \leftarrow r_b$
(1) $s_0 \leftarrow 2 - x_0 \times b$
(2) $y_0 \leftarrow 0 + x_0 \times b$
(3) $x_0 \leftarrow 0 + x_0 \times a$
(4) for $i = 0$ to $k-1$ do
(5) $\quad s_{i+1} \leftarrow 2 - y_i \times s_i$
(6) $\quad y_{i+1} \leftarrow 0 + y_i \times s_i$
(7) $\quad x_{i+1} \leftarrow 0 + x_i \times s_i$
(8) endfor
end.

Fig. 2. Pseudo-code for Goldschmidt's algorithm on each FPU configuration. In either case, the quotient is finally obtained as x_k. (A) Independent-add-and-multiply configuration. The line (5) must be executed only when $i < k-1$. (B) Multiply-add-fused configuration. The lines (5) and (6) must be executed only when $i < k-1$ and when $i < k-2$, respectively.

次に, $s_0 \leftarrow 2 - y_0 = 1 - \xi$ を分母, 分子に乗じ, $y_1 = 1 - \xi^2$, $x_1 = (a/b)(1 - \xi^2)$ を得る. 一般に, 第 i 回目の反復において, 以下の三つの演算を行う:

$$s_i \leftarrow 2 - y_i, \qquad y_{i+1} \leftarrow y_i \times s_i, \qquad x_{i+1} \leftarrow x_i \times s_i.$$

x_i の値は式 (2.2) の右辺で因子 $(1+\xi^{2^i})$ 以降を省略したものとなるが, 反復回数 k を $y_k = 1$ とみなせるようにとると, x_k には a/b が得られる. これが Goldschmidt のアルゴリズムである (Fig. 2). なお, 積和型の FPU の場合は, 第 i 回めの反復において, s_i の代わりに s_{i+1} を

$$s_{i+1} \leftarrow 2 - y_i \times s_i$$

としてあらかじめ求めておけば, y_{i+1} を求める際の乗算との依存関係を解消することができる. 同一の演算 $y_i \times s_i$ が二度行われることになるが, このほうが全体の計算が高速化される (Fig. 2, Line 5).

素朴な Taylor 級数法

関数の多項式近似に Taylor 級数展開を用いることは, 通常は, あまり洗練された手法ではないが, それでも多くの場面で用いられている. 除算 a/b に, この近似法を適用するならば, $b = z - \delta$ とおいて $1/(z-\delta)$ を z のまわりで δ のベキ級数に展開するのが一般的であろう. その結果を整理すると,

$$\frac{a}{b} = \frac{a}{z} \cdot \left(1 + \frac{\delta}{z}\right)\left(1 + \frac{\delta^2}{z^2}\right)\left(1 + \frac{\delta^4}{z^4}\right) \cdots \left\{1 + \left(\frac{\delta}{z}\right)^{2^{i-1}}\right\}\left\{1 + \left(\frac{\delta}{z}\right)^{2^i}\right\} \cdots \tag{2.3}$$

を得る. なお, これは式 (2.2) で $\xi = -\delta/z$ としたものと同一である. 本報告ではこの表式に忠実に Taylor 展開を計算する方法を素朴な Taylor 級数法とよぶことにする. すなわち, まず, $r_z = 1/z$ の値をテーブル参照により求めた後,

$$y_0 \leftarrow 1 - r_z \times b = \delta/z, \qquad x_0 \leftarrow r_z \times a = a/z$$

(A) procedure Taylor_IAM
(0) $x_0 \leftarrow r_z$
(1) $t \leftarrow x_0 \times b$
(2) $y_0 \leftarrow 1 - t$
(3) $x_0 \leftarrow x_0 \times a$
(4) for $i = 0$ to $k - 1$ do
(5) $\quad s_i \leftarrow 1 + y_i$
(6) $\quad y_{i+1} \leftarrow y_i \times y_i$
(7) $\quad x_{i+1} \leftarrow x_i \times s_i$
(8) endfor
end.

(B) procedure Taylor_MAF
(0) $x_0 \leftarrow r_z$
(1) $y_0 \leftarrow 1 - x_0 \times b$
(2) $x_0 \leftarrow 0 + x_0 \times a$
(3) for $i = 0$ to $k - 1$ do
(4) $\quad y_{i+1} \leftarrow 0 - y_i \times y_i$
(5) $\quad x_{i+1} \leftarrow x_i + x_i \times y_i$
(6) endfor
end.

Fig. 3. Pseudo-code for the naive Taylor series method on each FPU configuration. In either case, the quotient is finally obtained as x_k. (A) Independent-add-and-multiply configuration. The line (6) is executed only when $i < k - 1$. (B) Multiply-add-fused configuration. The line (4) is executed only when $i < k - 1$.

Table 1. Performance model* for each implementation

Methods	FPU configurations	
	IAM	MAF
Newton	$(2k+1)L_m + kL_a$	$(2k+1)L_{ma}$
Goldschmidt	$(k+1)L_m + kL_a$	$(k+1)L_{ma} + k$
Taylor	$(k+1)L_m + 2L_a + k - 1$	$(k+1)L_{ma} + 1$

*Each model describes latency in clock times: k is the number of iterations, L_m and L_a are the latencies of a multiplication and addition, respectively, on IAM configuration, and L_{ma} is the latency of a multiply-add operation.

とする. その後, 第 i 回目の反復で以下の操作を行う:

$$y_{i+1} \leftarrow y_i \times y_i, \qquad x_{i+1} \leftarrow x_i \times (1 + y_i).$$

なお, 積和型の FPU 上では, 積, 和の順で演算がなされるように

$$x_{i+1} \leftarrow x_i + x_i \times y_i$$

と計算することにする. 以上の手続きを Fig. 3 に疑似コードで示した.

性能モデル

以上, Figs. 1–3 に示した 6 種の実現の除算計算時間 (レイテンシ) を Table 1 に示した. ただし, いずれの構成の FPU もパイプライン化されており, 全ての演算のスループットは 1 マシンクロックであるとする. また, 分離型の FPU での乗算と加算のレイテンシはそれぞれ L_m と L_a, 積和型の FPU では積和のレイテンシは L_{ma} クロックサイクルであるとする. なお, Table 1 に示した値には, 初期値を求めるためのテーブル参照の他, 丸め, 正規化, 指数部の算出などに必要な時間は含んでいない.

Newton-Raphson 法では, その二つの乗算の間に依存関係が存在し, Table 1 に示したように, 他の手法に比べて低速となる. 一方, 素朴な Taylor 級数法と Goldschmidt のアルゴリズムは, 乗算に関す

る部分では全く同一の値をとり, 全体としてもほぼ同程度の性能を示す. 分離型においては, $k \geq 3$ かつ $L_a \geq 1 + (k-2)$ が成立するとき前者が, そうでないとき後者がそれぞれ若干高速である. 積和型の場合は, 現実的なテーブルサイズ ($k \geq 2$) を考えると, 素朴な Taylor 級数法がわずかに高速である.

前述のように, 本報告の目的のひとつは, 特別なハードウェアが仮定できない状況下で, 除算に対する乗法的手法の性能解析を行うことである. この制約を取り除くと, 分離型 FPU に対する結果は Table 1 に示したものとは異なってくる. すなわち, $2 - u \times v$ という複合演算をひとつの原子演算として行うように乗算器を拡張すれば, 除算性能が著しく向上する可能性がある. この拡張は, 比較的小規模のハードウェア資源の追加により実現可能である. しかし, 同様な改善を素朴な Taylor 級数法に対して行うためには, $1 + u \times u$ が行えるように乗算器を拡張する必要があるが, これには比較的大量のハードウェア投資が必要である. このような改善を行った Goldschmidt のアルゴリズムと, 改善を行っていない Taylor 級数法を比較すると, 前者が若干高速となる. なお, 積和型の FPU では, このような改善はいずれの手法においても意味をなさない. また, その他の有効なハードウェア支援としては, 初期値のテーブルを FPU 内に設置する, 各ステップでの一連の命令をあたかもひとつの原子命令として実行できるように乗算器を拡張するなど [11] があげられるが, これらは三者に同じように寄与する.

3 誤差解析

Newton-Raphson 法, Goldschmidt のアルゴリズム, 素朴な Taylor 級数法の三つの手法は, 数学的には同一の手法である. すなわち, すべての演算が無限精度で行えるのであれば, 各手法が生成する系列 $\{x_i\}$ は完全に一致する.† 初期値 r_b あるいは r_z をテーブル検索によって与えることにし, その際のキーとして b の仮数部の上位 n ビットを用いることにする. そのとき, 初期値の有効桁数は $\log(2^{n+1}+1)$ ビットであり, x_k の有効桁数 m_k は, いずれの手法でも

$$m_k \geq 2^k \log(2^{n+1}+1) \tag{3.4}$$

で与えられる.‡ 一方, 浮動小数点演算が用いられる場合は, 三者はそれぞれ異なる系列を生成する. 以後, 本章では, 浮動小数点演算により混入する誤差を考慮した誤差解析を行う.

まず最初に, 表記法や諸定義, 仮定について述べる. 浮動小数点数の加算, 減算, 乗算を, それぞれ, $\oplus$, $\ominus$, $\otimes$ で表す. また, 浮動小数点数の仮数部のサイズは p ビットであるとする. 一般に, 1 回の浮動小数点演算により混入する **相対誤差** の大きさの上限は, 丸め誤差のそれに等しく 2^{-p} である [6,12] :

$$x \odot y = (x \bullet y)(1+\eta), \quad |\eta| \leq 2^{-p} \tag{3.5}$$

ここで, $\odot$ は $\oplus$, $\ominus$, $\otimes$ のいずれか, $\bullet$ はそれに対応する厳密な演算である. 本報告での誤差解析は, 基本的には, この性質を利用し, 伝播する誤差の大きさの上限を推定するもので, いわゆる前進誤差解析である. また, 本報告の誤差解析においては, 積和型の構成をとる FPU を以下のように定義する: "積和型の FPU" とは, 積和演算のうち, 乗算で生ずる相対誤差の大きさがみかけ上 $2^{-p'}$ でおさえられ, かつ $2^{-p'} \ll 2^{-p}$ であるような FPU をいう. このとき, 積和演算のうち, 乗算による誤差は無視できて, 1 回の積和演算により混入する誤差の大きさの相対値の上限はほぼ 2^{-p} である. また, 式 (3.4) は無限精度の演算が行える場合の関係式であるため, m_k の値は p よりも若干大きめにとるのが一般

†無限精度の演算を用いる場合の一致性は, Goldschmidt のアルゴリズムと素朴な Taylor 級数法に関しては, 第 2 章に示したとおりである. また, これらの手法と Newton-Raphson 法についても, 同様の一致性を示すことができる [4, pp.A-25–26].

‡m_k の定義という意味では等号を用いる方が適切であるが, 後の記述を簡単にするため不等号を用いた.

的である. そこで, m_k の値を明記せずに "関係式 (3.4) を満足するような k" というときは, p よりも数ビット以上大きくとった m_k 並びにある与えられた n について, この関係式が成立するような整数 k をさすことにする. なお, 以下の誤差解析においては, $1/2 \le b < 1$ を仮定する.

Newton-Raphson 法での x_i の相対誤差 ϵ_i を $x_i = (1/b)(1+\epsilon_i)$ で定義する. また, 式 (2.1) の操作のうち, 第一演算および第二演算により混入する相対誤差 σ を

$$2 \ominus_2 b \otimes_1 x_i = 1 - \epsilon_i - \zeta_1 + \zeta_2 = (1-\epsilon_i)(1+\sigma)$$

のように定義する. ただし, ζ_1 と ζ_2 はそれぞれ演算 $\otimes_1$ と $\ominus_2$ で混入する**絶対誤差**である. すると, $b \otimes_1 x_i = 1 + \epsilon_i + \zeta_1 < 1$ の場合は, $|\zeta_1| \le 2^{-p-1}$ かつ $|\zeta_2| \le 2^{-p}$ であり, $|\sigma| \le 1.5 \times 2^{-p}$ が成立する. 一方, $1 + \epsilon_i + \zeta_1 \ge 1$ であれば, $|\zeta_1| \le 2^{-p}$ かつ $\zeta_2 = 0$ であり,

$$|\sigma| \le \frac{|\zeta_1| + |\zeta_2|}{1-\epsilon_i} \le \frac{|\zeta_1|}{1-|\epsilon_0|} = 2^{-p} + 2^{-p-n-1} \le 1.5 \times 2^{-p}.$$

このように, いずれの場合も $|\sigma| \le 1.5 \times 2^{-p}$ が成立する. なお, 前述のように, 積和型 FPU の場合は, $|\sigma| \le 2^{-p}$ である. Newton-Raphson 法では, 商 q は最終的に

$$q = a \otimes_4 x_{k-1} \otimes_3 (2 \ominus_2 b \otimes_1 x_{k-1}) \tag{3.6}$$

として計算される. よって, その商 q の相対誤差 θ_N について以下の関係が成立する:

$$1 + \theta_N = (1 - \epsilon_{k-1}^2)(1+\sigma)(1+\eta_3)(1+\eta_4).$$

ところで, σ, η_3, η_4 の大きさの上限はいずれも 2^{-p} 程度である. したがって, k が小さければ $\theta_N \simeq -\epsilon_{k-1}^2$ であり, q の有効桁数は k を 1 増やす毎にほぼ倍増する. その意味で, Newton-Raphson 法は自己補正的である. 一方, k を, 関係式 (3.4) を満足するように選べば, そのとき $\epsilon_{k-1}^2 \ll 2^{-p}$ であるから, 以下の誤差限界が得られる ($\sigma\eta_3$ など大きさが 2^{-2p} 程度以下の項は無視する):

$$|\theta_N| \le \begin{cases} 3.5 \times 2^{-p} & \text{(分離型 FPU の場合)} \\ 3 \times 2^{-p} & \text{(積和型 FPU の場合)} \end{cases}. \tag{3.7}$$

このように, 関係式 (3.4) を満足するように反復数 k を選べば, Newton-Raphson 法により得られる商の誤差限界は k に依存しない.

Goldschmidt アルゴリズムの場合については, 以下のように考えることができる. まず, x_i の相対誤差 ϵ_i を, $x_i = x_i^*(1+\epsilon_i)$ で定義する. ただし, $x_i^* = (a/b)(1-\xi^{2^i})$ である. このように今後, 無限精度の演算により得られた値を, その肩にアステリスク $(^*)$ をつけて表すことにする. なお, Newton-Raphson 法のときとは ϵ_i の定義が異なるので注意されたい. y_i の相対誤差 φ_i も同様に, $y_i = y_i^*(1+\varphi_i) = (1-\xi^{2^i})(1+\varphi_i)$ により定義する. ステップ i で y_i は以下のように更新される:

$$y_{i+1} = y_i^*(1+\varphi_i) \otimes_3 \{2 \ominus_1 y_i^*(1+\varphi_i)\} = y_i^*(1+\varphi_i)\{2 - y_i^*(1+\varphi_i)\}(1+\eta_{1,i})(1+\eta_{3,i})$$

ここで, $\eta_{1,i}$ は第 i ステップの演算 $\ominus_1$ により混入する相対誤差を表し, $\eta_{3,i}$ についても同様である. この式に $y_i^* = 1 - \xi^{2^i}$ を代入し, 大きさが 2^{-2p} 程度以下の微小項を省略すると,

$$\varphi_{i+1} \simeq \eta_{1,i} + \eta_{3,i} + \frac{2\xi^{2^i}}{1+\xi^{2^i}}\varphi_i. \tag{3.8}$$

このように y_i の誤差限界は反復数に依存しない．一方，x_i は以下のように更新される:

$$x_{i+1} = x_i^*(1+\epsilon_i) \otimes_2 \{2 \ominus_1 y_i^*(1+\varphi_i)\} = x_i^*(1+\epsilon_i)\{2 - y_i^*(1+\varphi_i)\}(1+\eta_{1,i})(1+\eta_{2,i}).$$

先程と同様に，大きさが 2^{-2p} 程度以下となる項を省略すれば，以下の漸化関係が得られる:

$$\epsilon_{i+1} \simeq \epsilon_i + \eta_{1,i} + \eta_{2,i} - \varphi_i + \frac{2\xi^{2^i}}{1+\xi^{2^i}}\varphi_i \tag{3.9}$$

ところで，$\xi \simeq 2^{-n-1}$ であり，現実的な n の値を考えると，式 (3.8) と式 (3.9) の右辺最終項の大きさは他の項に比べて極めて小さい．さらに，演算 $\ominus_1$ に伴う誤差が，これら二つの式の両方に出現するが，アルゴリズムの全ステップにわたって考えれば，これらは互いに打ち消しあう．Goldschmidt のアルゴリズムの特徴のひとつは，ある意味で自己補正的なことであった [4]．すなわち，反復の初期においては，分母 (y_i) と分子 (x_i) に乗ずる値 (Fig. 2 の s_i) が少々不正確であってもかまわないことであるが，両式の η_1 が打ち消しあうことは，この特徴を形式的に示すものである．結局，誤差限界に関する漸化関係 $|\epsilon_{i+1}| \le |\epsilon_i| + 2 \times 2^{-p}$ が得られる．また，$x_0 = r_b \otimes b$ であり，この演算による誤差も考慮すると，Goldschmidt のアルゴリズムで求めた商の相対誤差 θ_G について

$$|\theta_G| \le (2k+1) \times 2^{-p} \tag{3.10}$$

が成立する．なお，ここでの k も，式 (3.4) を満足するような最小整数の k の意味で用いている．

最後に，素朴な Taylor 級数法での誤差を考える．x_i の相対誤差 ϵ_i と y_i の相対誤差 φ_i を，Goldschmidt のアルゴリズムの場合と同様に定義する．すなわち，$x_i = x_i^*(1+\epsilon_i) = (a/b)\{1-(\delta/z)^{2^i}\}(1+\epsilon_i)$，$y_i = y_i^*(1+\varphi_i) = \delta^{2^i}(1+\varphi_i)$ である．分離型の FPU では，各ステップで y_i は

$$y_i = y_{i-1}^*(1+\varphi_{i-1}) \otimes_1 y_{i-1}^*(1+\varphi_{i-1})$$

として更新される．現実的には，反復数 k が無用に大きくとられることはなく，φ_i の大きさの上限は 2^{-p} の数倍程度と考えてよい．また，x_i は各ステップで以下のように更新される:

$$x_{i+1} = x_i^*(1+\epsilon_i) \otimes_3 \{1 \oplus_2 y_i^*(1+\varphi_i)\} = x_i^*(1+\epsilon_i)\{1 + y_i^*(1+\varphi_i)\}(1+\eta_2)(1+\eta_3).$$

大きさが 2^{-2p} 程度以下の微小項を無視すれば，x_i の相対誤差 ϵ_i について以下の関係を得る:

$$\epsilon_{i+1} \simeq \epsilon_i + \eta_2 + \eta_3 + \frac{y_i^*\varphi_i}{1+y_i^*}$$

ところで，$y_i^* = (-\xi)^{2^i}$ であるから，この右辺第四項の大きさは他の項に比べて非常に小さく無視できる．また，$x_0 = r_z \otimes a$ であるから，素朴な Taylor 級数法で求めた商 q の相対誤差 θ_T の大きさの上限は

$$|\theta_T| \le (2k+1) \times 2^{-p} \qquad \text{(分離型 FPU の場合)}. \tag{3.11}$$

ここで，Goldschmidt のアルゴリズムの場合と同様，k は式 (3.4) を満足するような最小の整数である．これは，y_i の大きさがいわゆる計算機イプシロン以下となると，その後，何回反復しようとも，正確に $s_i = 1 \oplus y_i = 1$ が成立するからである．

一方，積和型の FPU を用いて素朴な Taylor 級数法で除算を行う場合は，y_0 の誤差 φ_0 に注意が必要である．y_0 は以下のように計算される:

$$y_0 = 1 \ominus_2 r_z \otimes_1 b = \left\{\frac{\delta}{z}(1+\eta_1) - \eta_1\right\}(1+\eta_2).$$

これまでと同様に, 大きさが 2^{-2p} 程度以下の項を無視し, また $|z/\delta| = |1/\xi| \gg 1$ を考慮すると, $\varphi_0 \simeq \eta_1/\xi$ を示すことができる. また, 各ステップで x_i は以下のように更新される:

$$x_{i+1} = x_i^*(1+\epsilon_i) \oplus_3 x_i^*(1+\epsilon_i) \otimes_2 y_i^*(1+\varphi_i) = x_i^*(1+\epsilon_i)\{1 + y_i^*(1+\varphi_i)(1+\eta_2)\}(1+\eta_3).$$

これより, これまでと同様の近似を行えば,

$$\epsilon_{i+1} \simeq \begin{cases} \epsilon_i + \eta_3 & (i > 0), \\ \epsilon_i + \dfrac{\eta_1}{1-\xi} + \eta_3 & (i = 0). \end{cases}$$

を示すことができる. ところで, 積和型の FPU では, $|\eta_1| \leq 2^{-p'} \ll 2^{-p}$ であること, および, x_0 を求めるための演算 $r_z \otimes a$ の誤差を考慮すれば, 相対誤差 θ_T について以下の上限を示すことができる:

$$|\theta_T| \leq (k+1) \times 2^{-p} \qquad \text{(積和型 FPU の場合)}. \tag{3.12}$$

これまでと同様に, k は式 (3.4) を満足するような最小の整数である. なお, 素朴な Taylor 級数法の場合, 他の二つの手法とは異なり, 分離型の FPU と積和型の FPU では, そのコードが大きく異なる (Fig. 3). 積和型 FPU 用のコードを分離型 FPU の上で実行した場合, 上記の η_1 が無視できないため, $|\theta_T| \leq (k+2) \times 2^{-p}$ となる.

以上, ここまでの結果をまとめると, Newton-Raphson 法での誤差の上限は, 繰り返し数 k によらず一定であり, しかも, その値は 3×2^{-p} から 3.5×2^{-p} と小さい. 精度上, 全般的にみて最も優れた手法である. Goldschmidt のアルゴリズムおよび分離型の構成をとる FPU での素朴な Taylor 級数法では, 誤差の上限が $(2k+1) \times 2^{-p}$ で与えられ, 精度上は優れた手法とはいえない. 一方, 積和型の FPU での素朴な Taylor 級数法では, その誤差の上限が $(k+1) \times 2^{-p}$ で与えられ, k の値を小さくとれるのであれば, すなわち, 初期値テーブルのサイズをそれにみあって大きくとれるのであれば, Newton-Raphson 法と同程度あるいはそれ以上に正確な手法といえる. ただし, 同じコードを分離型の FPU で実行した場合, 誤差限界は $(k+2) \times 2^{-p}$ となる.

4 数値実験

第 3 章に示した誤差限界モデルは, いずれもその導出にあたって, いくつかの近似を行っている. そこで, これらのモデルの妥当性を確認するために数値実験を行った.

第 2 章に示した 6 種のコードを, C 言語のプログラムとして, 以下の環境で実現した: PentiumPro, Linux-2.0.31, gcc-2.7.2.3, libc-5.4.33. このプラットフォームでは, C の long double 型の仮数部のサイズは 64 ビットである. そこで, 各浮動小数点演算を long double 型で行い, その結果をその都度 double 型に丸めることにより分離型の FPU ($p = 53$) をシミュレートした. 一方, 積和型の FPU は, 積和をそれぞれ long double 型でおこない, その結果を double に丸めることにより, シミュレートした ($p = 53, p' = 64$).

精度の評価は以下のように行った. 区間 $[1/2, 1)$ を 2048 個に等分割し, 各部分区間からランダムに選出した除数 b とランダムに選んだ被除数 $a \in [1/2, 1)$ をもって, 2048 個の商 $q = a/b$ を計算した. この操作を 512 個の a について行った. この総計 1,048,576 個の商 q と, そのそれぞれを long double 型の演算 a/b で求めた値 q^* をもって, 以下の量 D を計算し, それを各手法の精度の指標とした:

$$\log_2 D = 53 - \max\left\{53, -\log_2 \left|\frac{q-q^*}{q^*}\right|\right\}.$$

Table 2. Accuracy of each implementation measured by D^*

k	n	Newton		Goldschmidt		Taylor		
		IAM	MAF	IAM	MAF	IAM	MAF	MAF/IAM**
1	29	2.867	2.487	2.907	2.412	2.907	1.976	2.907
2	14	3.321	2.827	4.925	4.576	4.476	2.854	3.761
3	7	3.422	2.867	5.731	5.532	5.741	3.694	4.473
4	3	3.331	2.758	6.163	6.033	6.803	4.519	5.138
5	1	3.191	2.761	7.926	6.317	6.999	4.662	5.671

*D for a double precision number is defined as $\log_2 D = 53 - \max\{53, -\log_2|(q-q^*)/q^*|\}$, where q and q^* are the quotients obtained with each method concerned and with long double divide operations, respectively. The quantity D is a function of the number of iteration k and the size of the key for the look-up table, n.

**The code for the MAF configuration is executed on the IAM configuration.

この D は, $|\theta| \times 2^p$ の上限の推定値であり, $\log_2 D$ は "失われる有効桁数" をビットを単位として表したものと考えることができる. また, D は繰り返し数 k と初期値テーブル (n-bits-in p-bits-out) のキーサイズ n の関数である.

各 k の値に対する n の値は, 式 (3.4) において $m_k = 60$ としたときに, この関係式を成立させる最小の整数とした. すると, $k = 1, 2, 3, 4, 5$ で n の値はそれぞれ 29, 14, 7, 3, 1 となる. このような組 (k, n) での各実現の D の値を Table 2 に示した. Newton-Raphson 法の場合は, どの (k, n) の組でも, 分離型の FPU であれば $D < 3.5$, 積和型の FPU であれば $D < 3.0$ となっており, また, その値は k とはほぼ独立であるといえる. このように, 式 (3.7) のモデルで数値実験の結果を説明できる. Goldschmidt のアルゴリズムの場合は, いずれの (k, n) でも $D < 2k + 1$ となっており, 分離型, 積和型いずれの構成においても, 式 (3.10) のモデルでうまく説明できている. 素朴な Taylor 級数法による場合も, モデルから予測すると, FPU が分離型であれば, $D \leq 2k + 1$, 積和型であれば $D \leq k + 1$, 積和型のコードを分離型 FPU 上で実行した場合は $D \leq k + 2$ となるはずであるが, 確かに, そのような結果が得られている.

このように, 各モデルで Table 2 に示した結果を大まかには説明できるが, 一部, 説明できないような振舞いもある. 最も顕著なものとしては, Newton-Raphson 法以外の手法では, k が大きくなると D の値は モデルにより予測される値 ($2k+1$ など) よりも大幅に小さくなることがあげられる. ここでの数値実験で得られた D の値が, モデルで予測される値に最も近くなるのは, 誤差の原因となりうる演算の全てにおいて最大誤差が発生したときである. k が大きいと, すなわち, 演算数が大きいと, そのような確率は非常に小さくなる. これが, k が大きくなると D の値はモデルにより予測される値よりも大幅に小さくなることの原因と思われる. この隔たりを改善するには, しらみつぶし方式の数値実験を行うか, あるいは, k が大きいときの振舞いを説明できるよう確率的なモデルへと詳細化するかのいずかの方法が考えられる. しかし, ここでの目的は, 数値実験の結果を完全に説明できる誤差限界モデルを開発することではなく, 誤差モデルの妥当性を数値的に検討することであるから, これらの改善は不要と思われる.

これまでに開発してきた誤差限界モデルの多くは, k の関数として表されていた. その k は, ある n が与えられたときに, 関係式 (3.4) を満足する最小整数の k であった. 以下, 反復数をその k よりも多くとった場合の各手法の振舞いについて述べる. Fig. 4 は, $k = 3$ の場合を例にとり, 各手法の D の値を n の関数として示したものである. もし, 反復数を前述の k よりも多くとっても精度は悪

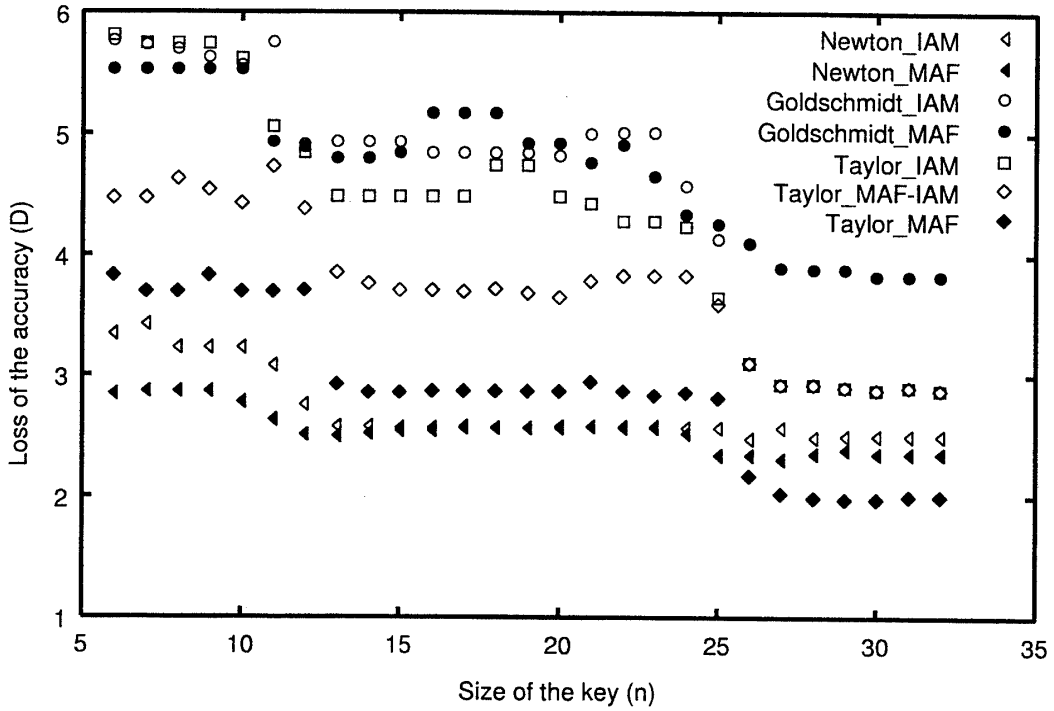

Fig. 4. Loss of the accuracy, D, for each implementation as a function of the size of a look-up table. An n-bits-in p-bits-out table is assumed. "Taylor_MAF-IAM" represents the code for MAF (Taylor_MAF) is executed on the IAM configuration. Since data are shown only for cases wherein $k = 3$, this figure does not necessarily represent the actual accuracy of each implementation for $k \neq 3$. See text for details.

化も改善もしないという命題が成立するのであれば, この図から $k = 1$ や $k = 2$ についての D の値も読みとれるはずである. 実際, 多くの実現ではこの命題が成立する. 例えば, 積和型の FPU での素朴な Taylor 級数法では, $6 \leq n \leq 10$, $11 \leq n \leq 23$, $26 \leq n \leq 32$ での D の値が, Table 2 に示した $k = 3, 2, 1$ での値とほぼ一致している. しかし, 積和型の FPU に対する Goldschmidt のアルゴリズムについては, この命題とモデル (3.10) により $D \leq 3$ が予測されるが, $n = 30$ での D の値はおよそ 3.8 となっている. これは, 積和型の FPU に対するコードでは, s_i が 1 に極めて近くなったときに, 前述の分母分子での誤差の打ち消しあいが行われないためであり, 繰り返しを余分に行うと精度が低下するという現象が発生する. なお, データは示していないが, さらにそれ以上反復しても, D の値は一定で変化しない.

さらに興味深いのは Newton-Raphson 法の場合である. Fig. 4 では, $n = 12$ の辺りで D が大きく減少している. この現象は式 (3.7) のモデルでは説明できない. しかし, より詳細な誤差解析を行うことにより Newton-Raphson 法での誤差の振舞いをほぼ完全に説明することが可能である (付録 A 参照). それによると, k の値を式 (3.4) を満足する最小の整数より 1 以上大きくとれば, すなわち, 1 回以上余分に反復すれば, 得られる商の相対誤差の大きさの上限は $(8/3) \times 2^{-p}$ となる.

5 おわりに

除算は, 多くのハードウェア資源をさいて実現することもできる. その場合, 本報告で紹介した乗法的手法ではなく, 減法的手法を用いることも多い. 事実, 現代の多くのマイクロプロセッサでは後

者が用いられている. ただし, そのような場合には, 多くのチップ面積を除算のために割くことになる. 冒頭で述べたように, 汎用のマイクロプロセッサを設計するにあたっては, このような面積・性能トレードオフは極めて重要な問題である.

本報告では, これとは対象的に, 除算の頻度が極めて低いようなアプリケーションのために, 除算機能を備えていない FPU 上でソフトウェアにより除算を実現するような場合, あるいは最小限のハードウェアコストで除算を実現するような場合を想定した. 大規模科学技術計算の高速化に専用ハードウェアが注目されている現在, このような状況は, 今後, 多くの分野で発生するものと思われる. 本報告では, そのような状況下において, 除算を乗法的アルゴリズムにより実現した場合の相対誤差の大きさの上限を表すモデル (誤差限界モデル) を導出し, 数値実験により, その妥当性を検証した. さらに本報告では, 逆数の Taylor 展開級数を素朴に計算することにより除算を行う手法が, 従来よく用いられてきた Goldschmidt のアルゴリズムよりも, 積和型 FPU 上では, 高精度であることを示した. パイプライン化された FPU においては, この素朴な Taylor 級数法は Goldschmidt のアルゴリズムと同程度に, Newton-Raphson 法よりも高速である.

本報告で開発した誤差限界モデルが記述するものは, 失われる有効桁数ではなく, その 2 のベキであり, その意味で非常に精密なモデルであるといえる. それらのモデルを用いて, Newton-Raphson 法での誤差の上限は 繰り返し数 k によらず一定 ($3.0 \times 2^{-p} - 3.5 \times 2^{-p}$, 余分に反復すれば $(8/3) \times 2^{-p}$) であること, 積和型の FPU での素朴な Taylor 級数法では, その誤差の上限が $(k+1) \times 2^{-p}$ で与えられること, それ以外の手法の誤差の上限は $(2k+1) \times 2^{-p}$ で与えられることを示した. さらに, これらのモデルの導出の過程で, 例えば, Goldschmidt のアルゴリズムの特徴 (分母分子に乗ずる因子が少々不正確でもかまわないこと) などを形式的に示した. 以上の結果は, ソフトウェア除算だけでなく, これらの乗法的アルゴリズムをハードウェアで実現する場合にも有益であると思われる. また, Newton-Raphson 法には各種の変法があるが [1], そのような手法の誤差の振舞いも, おおまかには, 本報告の誤差限界モデルで説明できると考えている. ただし, 詳細においての差異については検討の余地がある. 今後, それらに対する詳細な誤差限界モデルを構築することも, 基本演算としての除算を実装する上で有意義であろう.

参考文献

[1] Crandall, R. E., "Topics in Advanced Scientific Computation", Springer-Verlag, New York, 1996, pp.1–7.

[2] Darley, M., Kronlage, B., Bural, D., Churchill, B., Pulling, D., Wang, P., Iwamoto, R., and Yang, L., The TMS390C602A floating-point coprocessor for Sparc systems, IEEE Micro, 10(1990), 36–47.

[3] Flynn, M.J., On division by functional iteration, IEEE Trans. Computers, C-19(1970), 702–707.

[4] Goldberg, D., Computer Arithmetic, Appendix A in J.L. Hennessy and D.A. Patterson, "Computer Architecture: A Quantitative Approach", Morgan Kaufmann, Palo Alto, 1990.

[5] IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std. 754-1985.

[6] Knuth, D.E., The Art of Computer Programming. Volume 2, Seminumerical Algorithms, 2nd Ed., Addison-Wesley, Reading, Mass, 1989, 213–217.

[7] Montoye, R.K., Hokenek, E., and Runyon, S.L., Design of the IBM RISC System/6000 floating-point execution unit, IBM J. Res. Develop., 34(1990), 59–70.

[8] 長嶋雲兵, 小原 繁, 村上和彰, 網崎孝志, 北村一泰, 高島 一, 北尾 修, 田辺和俊, 稲畑深二郎, 山田想, 宮川宣明, 超高速分子軌道計算専用機 MOE の開発, 化学とソフトウェア, 19(1997), 129–135.

[9] Oberman, S.F., and Flynn, M.J., Design issues in division and other floating-point operations, IEEE Trans. Computers, 46(1997), 154–161.

[10] Oberman, S.F., and Flynn, M.J., Division algorithms and implementations, IEEE Trans. Computers, 46(1997), 833–854.

[11] Soderquist, P., and Leeser, M., Area and performance tradeoffs in floating-point divide and square-root implementations, ACM Computing Surveys, 28(1996), 518–564.

[12] Ralston, A., and Rabinowitz, P., A First Course in Numerical Analysis, McGraw-Hill, New York, 1978, 15–19.

A 反復数を多くとった場合の Newton-Raphson 法の誤差限界

ある n が与えられたとき, p に比し余裕をもって定めた m_k を用いて, 関係式 (3.4) から k の値を決定するとき, 最小の k よりも大きな k を用いた場合を考える.

第 i 番目の Newton ステップでの操作

$$x_{i+1} = x_i \otimes_3 (2 \ominus_2 b \otimes_1 x_i)$$

において, x_i の相対誤差を ϵ_i とする. すなわち, $x_i = (1/b)(1+\epsilon_i)$ である. また, 演算 $\otimes_1, \ominus_2, \otimes_3$ で混入する絶対誤差を, それぞれ $\zeta_1, \zeta_2, \zeta_3$ とする. また, 乗算と加減算に IEEE 754 最近偶数値丸めを仮定する. 以下, IAM 構成について説明するが, MAF についても同様である.

ある n と適切な m_k の値が与えられたとき, 条件 (3.4) を満たすような最小の k を用いると, 誤差限界モデル (3.7) により, 上記の操作で $|\epsilon_k| \leq 2.5 \times 2^{-p}$ である (最後に被除数 a を乗ずる操作を含めていないことに注意). そこで, $|\epsilon_i| \leq 2.5 \times 2^{-p}$ となった後, さらに Newton ステップを反復することを考える. この場合, ϵ_i の大きさにより, ϵ_{i+1} の大きさには 4 通りの可能性がある (Table A1). ここでは, $5/2 \geq \epsilon \times 2^p > 1$ の場合を例に説明する. 前述のように IEEE 規格 754 を仮定するので, 演算結果は無限精度のものを正しく (最近偶数値に) 丸めたものである. よって, $b \otimes_1 x_i = 1 + \epsilon_i + \zeta_1 = 1 + 2 \times 2^{-p}$, $2 \ominus_2 b \otimes_1 x_i = 1 - 2 \times 2^{-p}$. これより ϵ_{i+1} の値は $\epsilon_{i+1} = (x_{i+1} - 1/b) \times b = (1+\epsilon_i) \times (1 - 2 \times 2^{-p}) + b\zeta_3 - 1$. これが最小となるのは $\epsilon_i \downarrow 1 \times 2^{-p}$ のとき, 最大となるのは $\epsilon_i = 1.5 \times 2^{-p}$ のときである. また, $b = 2/3$ のときに $b|\zeta_3|$ の値が最大となる. これは, ζ_3 を, $x_i - x_i \times 2^{-p+1}$ を正しく丸める際の丸め誤差と考えればわかりやすい. これより, $7/6 \geq \epsilon_{i+1} \times 2^p > -5/3$. ただし, 2^{-2p} 程度の項は無視した. このように考えてゆくと, 最悪の場合, ϵ_i の値は, $(7/6) \times 2^{-p}$ のあたりと $(-5/3) \times 2^{-p}$ のあたりを繰り返すことになる (Table A1 参照). これに, 被除数 a を乗ずる操作を考慮すると, 商の誤差限界は $(8/3) \times 2^{-p}$ であり, k をいくら大きくとっても, その値は減少しない.

Table A1. Error Bounds for Arithmetic Operations That Occur in Each Newton's Step

ϵ_i	$b \otimes_1 x_i$	$2 \ominus_2 b \otimes_1 x_i$	$\lvert\zeta_3\rvert$	ϵ_{i+1}
$5/2 \geq \epsilon_i \times 2^p > 1$	$1 + 2^{-p+1}$	$1 - 2^{-p+1}$	$\leq 2^{-p}$	$7/6 \geq \epsilon_{i+1} > -5/3$
$1 \geq \epsilon_i \times 2^p \geq -1/2$	1	1	0	ϵ_i
$-1/2 > \epsilon_i \times 2^p > -3/2$	$1 - 2^{-p}$	1	0	ϵ_i
$-3/2 \geq \epsilon_i \times 2^p \geq -5/2$	$1 - 2^{-p+1}$	$1 + 2^{-p+1}$	$\leq 2^{-p}$	$7/6 \geq \epsilon_{i+1} \times 2^p \geq -7/6$

網崎孝志 (非会員) 〒690-8504 松江市西川津町 1060

昭和58年 大阪大学大学院薬学研究科前期課程修了. 同年 鳥取大学医学部技官. 平成3年 島根大学理学部助手, 講師 を経て, 平成7年 島根大学総合理工学部助教授. 平成4年 博士 (薬学). 計算化学, 並列分散処理, 最適化法の研究に従事. 日本薬学会, ACM など各会員.

長嶋雲兵 (正会員) 〒305-8565 つくば市東 1-1

昭和58年 北海道大学大学院博士後期課程修了. 理学博士. 同年 岡崎分子科学研究所助手. 平成4年 お茶の水女子大学助教授. 平成8年 同教授. 平成10年 物質工学工業技術研究所理論化学研究室長. 理論化学, 並列分散処理, 性能評価の研究に従事. 日本化学会, IEEE, ACM 各会員.

(1998年8月17日受付)