Vol. 34 No. 7 情報処理学会論文誌 July 1993

初等関数の高速計算法

黄 栄輝[†,††] 後藤英一[†,†††] 吉田宣章[†]

初等関数の計算は，その重要性にもかかわらず，高速数値計算では比較的なおざりにされてきた領域である．この領域におけるもっとも重要な初期の革新のひとつは CORDIC アルゴリズムの発見である．しかし，ハードウェアで実現されても，CORDIC アルゴリズムは必要な速度を提供するには至っていない．これは CORDIC がビット順次アルゴリズムであるためである．スーパコンピューティングの分野では，べき級数や他の多項近似が一般的である．計算の高速化に表がよく用いられる．メモリー技術のめざましい進歩によって，もっと大きな表をメモリーに入れることが可能になっている．本論文の目的は，表容量の増加によって，逆数，平方根，指数関数，正弦，余弦，対数，逆三角関数などの初等関数の計算速度がどのように向上するかを検討することである．これらの関数を単精度で計算するために我々は ATA (Add-Table lookup-Add) 法を導入する．それから，倍精度にその結果を拡張することにする．計算時間と精度の問題も取り扱う．

Fast Evaluation of Elementary Functions

W. F. Wong,[†,††] Eiichi Goto[†,†††] and Nobuaki Yoshida[†]

The computation of the elementary functions, despite its importance, is a relatively neglected area of high speed numerical computing. One of the most important early breakthrough in this area is the discovery of the CORDIC algorithm. However, even when realized in hardware, the CORDIC algorithm is still unable to deliver the necessary speed. This is due mainly to its linear convergence. In the area of supercomputing, power series or other polynomial approximations are commonly used. To speed up the computations, tables are often used. With the tremendous gain in memory technology, it is now possible to put much larger tables into memory. It is the purpose of this paper to consider how increased table storage will improve the computations of reciprocal, square rooting, logarithm, exponential, arc tangent, sine and cosine. We will introduce the ATA (Add-Table lookup-Add) method for computing these functions in single precision. We will then extend the result for double precision. The timing and accuracy issues will also be addressed.

1. はじめに

今日の数値計算における問題の多くでは，初等関数が重要な役割を果たしている．例えば，平方根および三角関数は，複雑なコンピュータ・グラフィックスや画像処理に重要性を増している．しかし，今日もっとも高速なコンピュータでさえ初等関数の計算は比較的に遅いことが知られている[6]．それは，この領域では逆数計算以外，ハードウェアのサポートが行われていないためである．関数の計算速度を上げる問題におけるもっとも初期の成功のひとつは CORDIC アルゴリズムの発見である[8),10)]．連続した桁送り，加減算，および表探索を用いて，重要な初等関数すべての計算を行うことができる．この方法の長所は，ハードウェア要件が低く，簡単であることである．しかし，CORDIC は本質的にはビット順次アルゴリズムで遅いため，その利用が現在のところポケット計算機に限られている．より大きな計算機では，関数計算の高速化は，べき級数あるいは収束の速い近似多項式の利用によってなされている[1),3),4)]．表も用いられるが[7)]，作表と加算乗算などの並列化は図られていない．

コンピュータ技術においてもっとも目覚ましい成功のひとつは RAM (Random Access Memory) の分野にみられる．数年単位で，メモリーチップの記憶容量は4倍ずつ大きくなっている．16メガビットのチップが製造ラインに乗るのも間近である．次世代の64メガビット，256メガビットのメモリーチップの研究もすでに進んでいる．近年のもうひとつの革新は，容

† 理化学研究所，後藤特別研究室
Goto Laboratory, The Institute of Physical and Chemical Research (RIKEN)
†† シンガポール国立大学 DISCS
Department of Information Systems and Computer Science, Faculty of Science, National University of Singapore
††† 神奈川大学理学部情報科学科
Department of Information Science, Faculty of Science, Kanagawa University

量はやや小さいものの非常に高速なメモリーチップを生産する BiCMOS 技術の利用である[5]．このため，初等関数計算の高速化に大きな（メガバイト）表を利用することが可能となりつつある．この論文では，我我が ATA (Add-Table lookup-Add) と呼ぶ新しい手法を使ったアプローチの効用を示す．浮動小数点数 (IEEE 規格) については，この方法は現在まで提案されたものの中で最高速のアルゴリズムであって，単精度の仮数部については1乗算，倍精度については2乗算程度の計算時間になる．

第2章では，ATA 法を導入する．この方法を使うと，逆数，平方根，指数関数，正弦，余弦，対数，および逆正接を，IEEE 規格単精度の仮数部に対して扱うことができる．第3章では，ATA 法を基本とした ATA-M 法を導入して，IEEE 規格倍精度の仮数部を扱う．指数部の扱いは単精度・倍精度の両方に対して同様であり，それは第4章で説明される．第5章ではニュートン法に触れる．最後に第6章で論を結ぶ．

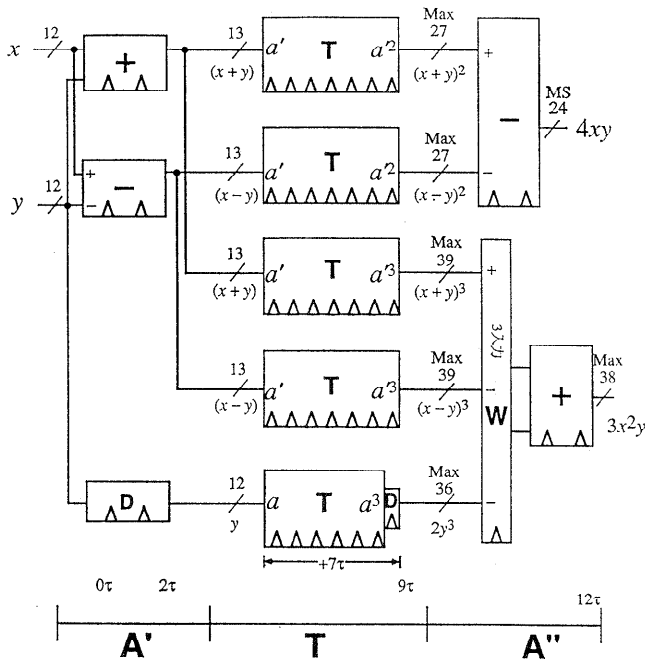
アルゴリズムの速度を推定するために，我々は，表探索や乗算など様々な基本演算の実行に要する時間についてある前提をたてた．これらの前提は，付録に示してある．実装技術上の問題を回避するため，ECL電流スウィッチ遅延 τ[2] を単位として遅延時間を数える．アルゴリズムによって得られる速度の利得をより正確にわかっていただくために，完全な倍精度乗算を実行する時間 τ_{FULL} について約16τと推定したことを覚えておいていただきたい．また，本論文では，2のべき乗の定数による乗算が必要な時は常に，それはハードウェア上の配線による算術シフトでできると考えて，計算時間は無視できるとした．

2. ATA 法

メモリー技術の進歩を念頭に置きながら，初等関数計算という古い問題を検討していこう．強力なチップを以てしても，初等関数を完全に表化できるには至っていない．単精度関数でも $2^{32}\times32$ ビット（=16ギガバイト）の表が必要になる．いうまでもなく，倍精度関数では，まったく不可能である．

しかし，メガバイトの容量のある表を用いることで，これらの関数の計算を高速化できると我々は示す．我々がこれを達成するために用いる方法は ATA (Add-Table lookup-Add) 法と呼ぶことにする．この名称が暗示するとおり，一般的にこの方法はデータの加減算を用いる．その計算結果は表のアドレスとして用いる．表から読み出したものはさらに加減算される．時間のかかる乗算は使わない．この一般的な方法を三つの例について説明する．

図1から**図3**までは三つの例の概要を示している．以下の説明を読む間，参照していただきたい．図の中では，各演算を表す四角の中で，小さい三角形を使って，その演算に要する τ の数を示している．各データの線の最大ビット長も示している．しかし，大概は，すべてのビットが必要なわけではない．特に注意すべきことは，時間区切りごとに並行して行われる演算である．図の中で，「T」は表引きを表し，「D」は遅延，そして「W」はウオレストリーによる加算器を表している．



説明：
D：遅延　W：ウオレスのトリー　$a(a')$：12（13）ビットのアドレス
T：表引き　+：桁上げ先見加算器　MS：最下位
n：n ビットのパス　A′：前段加算　A″：後段加算
：1τの遅延

図1 $4xy$ と $3x^2y$ を計算する ATA 回路
Fig. 1 ATA circuit for computing $4xy$ and $3x^2y$.

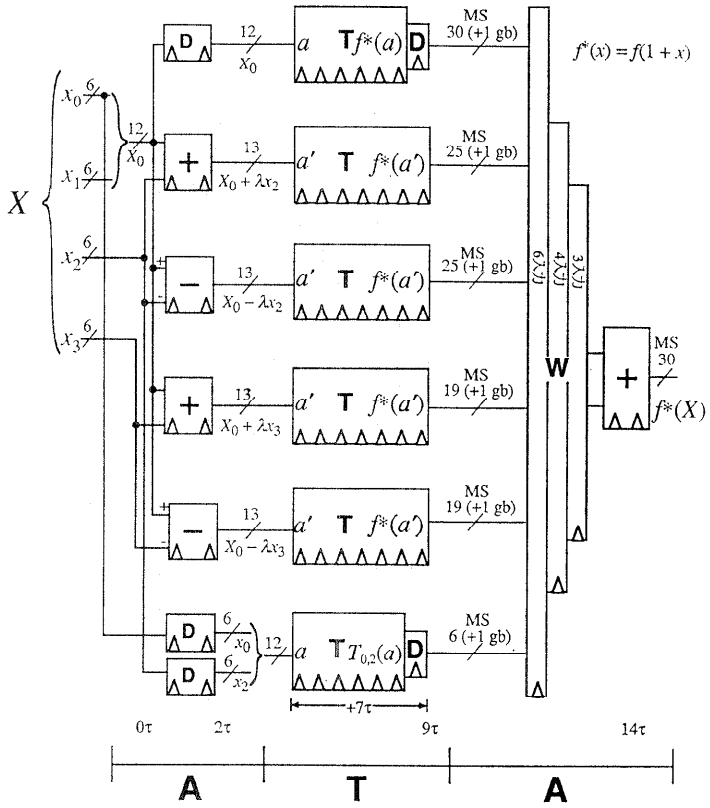


図 2 30 ビット正規仮数部の計算する ATA 回路
Fig. 2 ATA circuit for 30 bit normalized mantissa.

2.1 平方表による乗算

図1の上部は次の ATA 恒等式を使った乗算回路である.

$$4xy=(x+y)^2-(x-y)^2 \qquad (1)$$

まず「加算」の段階では, $(x+y)$ と $(x-y)$ の和がそれぞれ, 同時に並行して計算される. 次に「表引き」の段階で, $(x+y)^2$ と $(x-y)^2$ を得る. 最後の「加算」の段階で, 目的の結果が出る.

図1では x と y が 12 ビット, 表は 13 ビットのアドレスの表である. 24 ビットの積を得る計算時間は付録の方法で計算すると 11τ になる. 一方, 組合せ乗算回路の遅延時間は 8τ である. したがって, 本論文では乗算にはより速い組合せ乗算回路を使うものとする. しかし, もしメモリーの速度がより速くなれば, ATA 乗算の方が速くなるかもしれない.

2.2 立方表による $3x^2y$ の計算

後で使う $3x^2y$ を立方表を使い次の ATA 恒等式によって計算できる.

$$2(3x^2y)=(x+y)^3-(x-y)^3-2y^3 \qquad (2)$$

x と y が 12 ビットの長さ, そして表からの出力と最終結果がともに 24 ビットの長さの場合, ゲート遅延時間は 12τ となる. 一方, もし組合せ乗算回路を使えば少なくとも 18τ の時間がかかる. したがって, ATA 法の方が速い.

2.3 関数の乗算を用いない ATA 評価法

X を $0\le X<1$ なる 30 ビット2進小数として, $X=x_0+\lambda x_1+\lambda^2 x_2+\lambda^3 x_3+\lambda^4 x_4$ と表す. ただし $\lambda=2^{-6}$ であり, x_n, $0\le n\le 4$, は6ビットの2進小数である. IEEE 単精度の正規化された仮数部は $1+X$ なる形を持つ. このうち x_4 は最下位6ビットの内部ガードビットを表す. 特に, 丸められた外部入力では $x_4=0$ となる. ここで

$$1+X=X_0^*+\delta X,$$
$$X_0^*=1+x_0+\lambda x_1,$$
$$\delta X=\lambda^2 x_2+\lambda^3 x_3+\lambda^4 x_4$$

とおけば, 6ビット小数に関して四次までテイラー展開と合致する, 次の ATA 公式が得られる.

$$f(1+X)=f(X_0^*+\delta X)=\sum_{n=0}^{\infty}\frac{f^{(n)}(X_0^*)}{n!}(\delta X)^n$$
$$=f(X_0^*)+\frac{\lambda}{2}(f(X_0^*+\lambda x_2)-f(X_0^*-\lambda x_2))$$
$$+\frac{\lambda^2}{2}(f(X_0^*+\lambda x_3)-f(X_0^*-\lambda x_3))$$
$$+\lambda^4(T_{0,2}(x_0,x_2)+T_{0,4}(x_0,x_4))+O(\lambda^5) \qquad (3)$$

ただし $T_{0,2}$ と $T_{0,4}$ は 12 ビット引数を持つ下記の $O(\lambda^4)$ の補正表である.

$$T_{0,2}(x_0,x_2)=\frac{x_2^2}{2}f''(1+x_0)-\frac{x_2^3}{6}f'''(1+x_0) \qquad (4)$$

$$T_{0,4}(x_0,x_4)=f'(1+x_0)x_4 \qquad (5)$$

図2は(3)を評価するための回路である. 図に見るとおり, 乗算器は使っていない. 最初の五つの項は, 加算と減算の組合せと, その結果を使っての表引きで得られる. $T_{0,2}(x_0,x_2)$ を得るために, x_0 と x_2 の各6ビット, 計 12 ビットを使ってアドレスを構成する. このアドレスを使って, $T_{0,2}(x_0,x_2)$ が表から

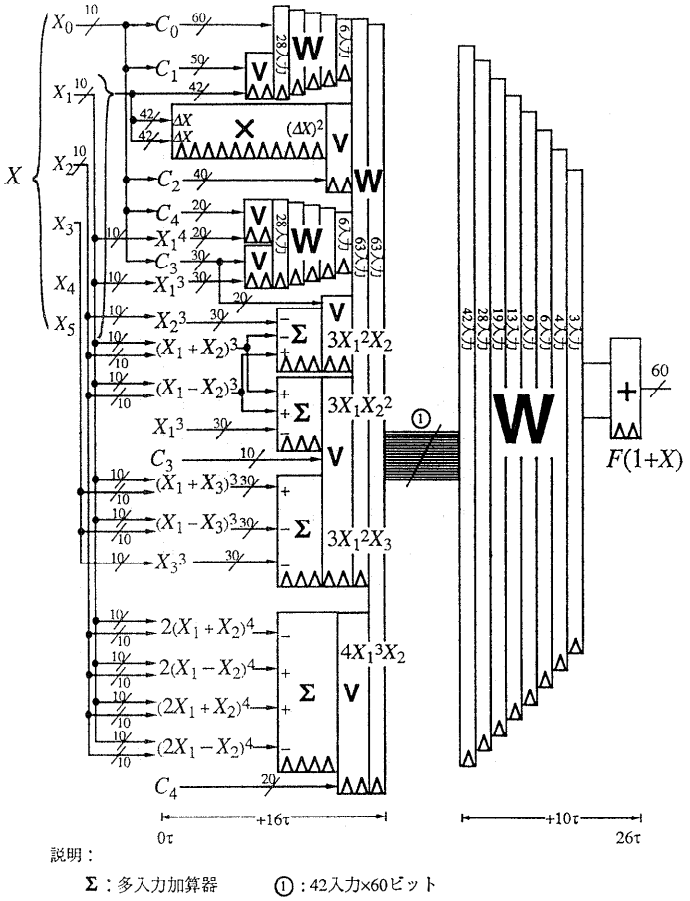


図 3 60 ビット正規仮数部の四次多項式近似級数を計算する ATA 回路
Fig. 3 ATA circuit for computing 4th order approximation polynomial of 60 bit normalized mantissa.

読み出される．24 ビットの仮数の時，$x_4=0$ なので $T_{0,4}(x_0, x_4)$は不要であるので図にない．ゲート遅延時間は 14τ となる．それは倍精度1乗算の時間 (τ_{FULL}) と同程度である．

表 1 24 ビットの入力に対する 2^{24} 通り全数計算の結果
Table 1 Complete check of 24 bit mantissa input.

関　数	Min(−log₂(絶対誤差))
逆　数	27.3 ビット
平 方 根	31.6 ビット
逆数平方根	32.6 ビット
指 数 関 数	29.9 ビット
$\cos\left(\frac{\pi X}{2}\right)$	29.7 ビット
$\sin\left(\frac{\pi X}{2}\right)$	30.3 ビット
自 然 対 数	29.1 ビット
逆 正 接	30.8 ビット

我々は $\lambda=2^{-6}$ の全組合せで様々な初等関数を24ビットの仮数 ($1\leq X<2$) すべてについて全数検査し，必要な入力範囲内の結果を**表1**にまとめた．

3. ATA-M (ATA-Multiply) 法

IEEE 規格浮動小数点数の仮数部は，X ($0\leq X<1$) を二進小数として $1+X$ と表される．$\Lambda=2^{-M}$ と置いて，X を $X=\sum_{i=0} \Lambda^i X_i$ のように展開する ($0\leq X_i<1$)．X を，

$$1+X=X_0^*+\Delta X$$

と組み直す．ただし

$$X_0^*=1+X_0+2^{-(M+1)},$$
$$\Delta X=\Lambda X_1'+\sum_{i=2} \Lambda^i X_i$$
$$X_1'=(X_1-1/2)$$

である．ここで X_0^* は二進形で X_0 の上位に1，下位に1を付けたものになるが，これらは固定されているので，X_0 と同じく M ビットで指定することができる．初等関数 $f(1+X)$ を求めるために，X_0^* の近傍 ($-2^{-(M+1)}\leq\Delta X<2^{-(M+1)}$) において次の N 次多項式 $F(1+X)$ による近似計算を行う：

$$F(1+X)=C_0+C_1\Delta X+C_2(\Delta X)^2+\sum_{n=3}^{N} C_n(\Delta X)^n \quad (6)$$

ただし C_n は X_0^* の関数である．C_n は表を引く．$(\Delta X)^2$ は1回の乗算なので，表を引くのと同時に行う．$(\Delta X)^n$ ($n\geq 3$) は2回以上の乗算になるので後述の ATA 公式を使って計算する．C_n と $(\Delta X)^n$ を掛けるので ATA-M と呼ぶ．式(6)を書きなおすと，

$$\begin{aligned}F(1+X)=C_0+&\\ &C_1\Delta X+\\ &C_2(\Delta X)^2+\\ &C_3(\Lambda^3X_1^3+\Lambda^4(3X_1^2X_2)+\Lambda^5(3X_1^2X_3+\\ &\quad 3X_1X_2^2)+\Lambda^6(3X_1^2X_4+6X_1X_2X_3))+\\ &C_4(\Lambda^4X_1^4+\Lambda^5(4X_1^3X_2)+\\ &\quad \Lambda^6(4X_1^3X_3+6X_1^2X_2^2))+\\ &C_5(\Lambda^5X_1^5+\Lambda^6(5X_1^4X_2))+\\ &C_6\Lambda^6X_1^6+\\ &O(\Lambda^7)\end{aligned} \quad (7)$$

となる．Λ^6 の位までの展開式を近似した式(7)を計

算するために，(2)および次の ATA 恒等式を用いることを提案する．

$$8(3xyz)=(x+y+z)^3-(-x+y+z)^3 -(x-y+z)^3-(x+y-z)^3 \quad (8)$$

$$16(3x^3y)=-2(x+y)^4+2(x-y)^4 +(2x+y)^4-(2x-y)^4 \quad (9)$$

$$4(3x^2y^2)=(x+y)^4+(x-y)^4-2x^4-2y^4 \quad (10)$$

$$8(15x^4y)=(2x+y)^5-(2x-y)^5-4(x+y)^5 +4(x-y)^5+6y^5 \quad (11)$$

特に

$$C_n=f^{(n)}(X_0^*)/n!$$

とおけばテイラー級数に相当する．この場合，近似式 $F(1+X)$ の誤差はおよそ

$$|(\Delta X)^{N+1}f^{(N+1)}(X_0^*)/(N+1)!| \leq |2^{-(M+1)(N+1)} \cdot f^{(N+1)}(X_0^*)/(N+1)!|$$

によって抑えられている．しかし関数の数値計算では，低い次数で良い精度を得るために，チェビシェフ近似を使うのが通例である[12]．今回の論文では，最良の近似式を求めることまでは行わずに，テイラー展開の $(\Delta X)^{N+1}$ の項にチェビシェフ近似を施すことにとどめる．すなわち，$(\Delta X)^{N+1}$ を N 次多項式に収めるようにするには，

$$C_n=f^{(n)}(X_0^*)/n!-2^{(M+1)n+1-(N+1)(M+2)} \cdot t_n f^{(N+1)}(X_0^*)/(N+1)!$$

とすればよい．ただし t_n は $N+1$ 次チェビシェフの多項式：

$$T^{(N+1)}(x)=2^N x^{N+1}+\sum_{n=0}^{N-1} t_n x^n \quad (12)$$

の n 次の項の係数である．$T^{(N+1)}(x)$ の性質により，誤差が

$$|2^{-(M+2)(N+1)+1}f^{(N+1)}(X_0^*)/(N+1)!| \quad (13)$$

程度にまで下げられることが分かる．なお，$|f^{(N+1)} \cdot (X_0^*)/(N+1)!|$ の値は，本論文で扱う関数で，$1 \leq X_0^* < 2$ の範囲では，たかだか $|(1/X)^{(N+1)}/(N+1)!|_{X=1}=1$ である．具体的に倍精度浮動小数点計算の場合，$M=10$, $N=4$ とすれば，誤差が 2^{-59} で抑えられる．このとき式(6)で C_5, C_6 はいらない．また，C_3, C_4 の A^6 の項もいらない．図3でこの場合の ATA-M 回路を示す．$M=8$, $N=6$ とすると表は小さくなるがウオレスのトリーが複雑になる．どれが良いかを決めるためには，チップの製造技術を考慮した，より詳細な解析が必要となる．

4. 符号と指数部を含む計算

IEEE 754 バイナリー浮動小数点規格[11]の定義にある主なパラメータを考えてみよう．単精度浮動小数点数 X の場合は，$s \in \{0,1\}$, $-127<e<128$, $0 \leq f<2^{23}$ で，倍精度 X の場合は，$s \in \{0,1\}$, $-1023<e<1024$, $0 \leq f<2^{52}$ の三つのフィールドで，次のように表現される．

$$X=(-1)^s 2^e(1 \cdot f)$$

s は符号ビット，e は指数，そして $1 \cdot f$ は仮数として知られている．実際の規格では，e はバイアス形式で表されているが，単純化のために真の値として取り扱う．IEEE 規格は無限大，非数，非正規化数などの量も定義している．しかし，規格はそのような形の入力を受け取った場合の初等関数の振る舞いについては何も定義していない．また，例えば負の数の対数や平方根のような，不適当な演算に対する出力も定義されていない．我々はこれを重要かつ興味深い規格拡張の問題と考えるが，この論文の範囲を越えているので扱わないことになる．

2章と3章で，仮数部の計算を説明したが，この章では符号と指数部を含む計算への拡張を検討する．次の説明では，単精度が中心になるが，倍精度の場合にも，違う表を使えば，方法は全く同じである．

実際の計算では，相対誤差の問題が重要になる．IEEE 規格の定義から，最下位の桁 (unit in the last place—ulp) の単位は，単精度の場合に 2^{-24} で，倍精度の場合に 2^{-53} である．精度については，相対誤差が 0.5 ulp 以下であることが要求される．つまり，もし α が無限精度の厳密な答で，α' が我々のアルゴリズムの結果であるなら，次の式が成り立つことが主張できなければならない．

$$\frac{|\alpha-\alpha'|}{\alpha}<0.5\,\mathrm{ulp}$$

単精度の場合，$|f(X_0^*)| \gg \lambda$ かつ $|f^{(n)}(X_0^*)| \ll 1/\lambda$, $(n \geq 1)$ ならば，式(3)により，我々の方法がこれを満たすことは明らかである．実際，逆数，平方根，指数関数ではこうなっている．しかし，正弦，余弦，対数，逆正接の場合には，$f(X_0^*)=0$ となることがあるので問題である．これらの取扱いについては以下で個々に述べる．

4.1 逆　　数

$X=(-1)^s 2^e(1 \cdot f)$ を上に定義した単精度数であるとしよう．

$1\leq(1\cdot f)<2$ のとき $0.5<(1\cdot f)^{-1}\leq 1$ となり，結果を正規化することが必要である．

また，アルゴリズムの速度が，$(1\cdot f)^{-1}$ の計算によって支配されていることは簡単にわかる．2.3 節に示されているとおり，これは 14τ を要する．倍精度の場合に 25τ になる．

4.2 平方根および逆数平方根

平方根の場合も除算の場合に非常によく似ている．いままでと同様に，$X=(-1)^s 2^e(1\cdot f)$ を単精度数としよう．平方根 X は次のような分け方に従って，$X'=(-1)^s 2^{e'}(1\cdot f')$ で表されるもうひとつの単精度浮動小数点数となる．

$$e'=\begin{cases}\dfrac{e}{2} & \text{if } e \text{ is even}\\[2ex] \dfrac{e-1}{2} & \text{if } e \text{ is odd}\end{cases} \tag{14}$$

$$(1\cdot f')=\begin{cases}\sqrt{1\cdot f}+\varepsilon & \text{if } e \text{ is even}\\ \sqrt{2\times 1\cdot f}+\varepsilon & \text{if } e \text{ is odd}\end{cases} \tag{15}$$

我々は最初に $(1\cdot f')$ を固定した．もし e が偶数であれば，$1\leq(1\cdot f)<2$ となり，$1\leq(1\cdot f')<\sqrt{2}<2$ となることを示している．

もし e が奇数であるなら，$2\leq 2\times(1\cdot f)<4$ となり，$\sqrt{2}\leq(1\cdot f')<2$ となることを示している．したがって，$1\leq(1\cdot f')<2$ となるのであり，つまり，$(1\cdot f')$ が正規化された形となる．e' は整数であってほしいため，(14)式で行った調整が必要である．

表1では単精度の場合に $1\leq X<2$ について $\sqrt{X}$ を計算する時の精度を示した．我々は $1\leq X<4$ を必要とするから，ATA 法を用いて $\sqrt{X}$，$2\leq X<4$ を計算する時の精度も検査した．

速度は，除算と同様，単精度の場合に 14τ，倍精度の場合に 25τ となる．

逆数平方根の計算に上記のものがどう拡張されるかは容易にわかる．指数は予備桁シフトの後に単純否定することによって対応でき，仮数は別の表の組を用いた同様な ATA から得ることができる．

4.3 指数関数

指数関数の場合においては，我々が用いることのできる最も単純な恒等式は

$$\exp(X+Y)=\exp(X)\exp(Y) \tag{16}$$

残念ながら，これでは指数は都合よく処理できない．そこでまず X を指数部が1になるように非正規化する．そして仮数部の整数部を p として小数部を q とする．これを行うために，任意のビット数だけオペランドをシフトできるような可変シフト器が必要となる．そのようなシフト器は 2τ の時間を取るであろうと見積る．

$\exp(2^{10})$ が単精度表現においてオーバフローするため，$p<2^{10}$ でなければならない．したがって $\exp(p)$ は表から引くことができる．$\exp(q)$ の方は前述の ATA 法を用いる．こうして得られた $\exp(p)$ と $\exp(q)$ を(16)によって乗ずる．

速度の点でも，標準的な ATA 以外にも，ひとつの乗算を用いる．したがって，全所要時間は単精度の場合に 32τ で，倍精度の場合に 43τ である．

4.4 正弦と余弦

正弦と余弦の計算における大きな問題のひとつは引き数から主値を導出することである．ほとんどのアルゴリズムは引き数の範囲に制限を設けてこの問題を避けている．しかし，我々は，引き数に制限を設けなくとも，アルゴリズムが正しい値を戻せるようにしたい．

$0\leq\theta<\dfrac{\pi}{2}$ の時に $Y=\theta+\dfrac{m\pi}{2}$ であるとしよう．すると

$$\begin{aligned}\exp\left(\frac{i2Y}{\pi}\right)&=\cos\left(\frac{2\theta}{\pi}+m\right)+i\sin\left(\frac{2\theta}{\pi}+m\right)\\&=\cos(\theta'+m)+i\sin(\theta'+m)\end{aligned} \tag{17}$$

ここで $0\leq\theta<1$ は ATA への入力である．したがって，この ATA 操作に $\dfrac{2}{\pi}$ の桁移動数を補正させて，その入力 θ' から実際に $\exp(i\theta)$ を計算することが必要である．m の最下位の2個のビットを書き留めた後，$n=0,1,2$ あるいは3として $m=4k+n$ とすることができる．n はしたがって m の最下位ビット2個であり，そしてこれらから，もし $n=0,1,2$ あるいは3であれば，それぞれ Y が $\theta+2k\pi$，$\theta+2k\pi+\dfrac{\pi}{2}$，$\theta+2k\pi+\pi$，あるいは $\theta+2k\pi+\dfrac{3\pi}{2}$ であるというふうになる．例えば，もし $n=1$ であれば $m=4k+1$ となり，これはまた $Y=\theta+\dfrac{(4k+1)\pi}{2}=\theta+2k\pi+\dfrac{\pi}{2}$ であることを意味する．これから，$0\leq z<2\pi$ については $\exp(iz)=\exp(i(z+2k\pi))$ となる．したがって，s_0 と s_1 の値に従って象限調整を行う必要がある．これらの調整は正弦および余弦のよく知られた性質により明らかである．

必要となる操作は，$\pi/2$ で乗算することと ATA 操作することである．ATAN2 の場合と同様に，象限調整は ATA の最終局面で行うことができる．よって，所要時間は単精度の場合に 32τ で，倍精度の場

合に 43τ である．

ここで相対誤差の問題を論ずる．範囲の調整を行ったので，その結果 $\cos(\pi X/2)$ と $\sin(\pi X/2)$ の計算は $0\leq X<1$ の範囲内において要求されたということを思い出してほしい．したがって，引き数が零に接近するときの正弦や引き数が $\pi/2$ に接近するときの余弦の計算が必要となってくることが有り得る．この時 x_i の中のビットが零になり，ゆえに我々が用いていた多項式展開の低次の項が零になるので，多項式近似の有効性が減ってくるために起きる．この問題は，有効ビット数が増すようにビットをずらすことによって対処できる．すなわち，引き数を $X'=1+2^{-n}X+\varepsilon$ としよう．ただし単精度の場合には $\varepsilon=O(2^{-24})$，倍精度の場合には $\varepsilon=O(2^{-53})$ である．やりたいことは，多項式近似において X の重要度を増すことである．これを行うために，最大項の1を取り除いてから，$2^{-n}X$ の表現を，4ビットずつずらして行って，16進表現において少数点の後の最初の16進数が零でなくなるように再正規化する．そのような4ビットシフトの回数を k としよう．単精度の場合，$k=1,2$ か3のときには，(3)のようにATA法を使う．ただしその際に用いる表は異なっていて，新しい指数部に合わせたものを用いる．k が3を越えているときには，その k に合わせた A を使って ATA-M を直接使う．以前と同様，これらの調節された数の乗算は，算術シフトによってなされる．対数の倍精度計算の場合も，この方法で扱われる．これらの計算において，我々は16進法に基づいてきたので，IEEE 規格に従った形で結果を得るためには，計算の最終段階で，零桁から4桁の2進シフトを行う必要があることに注意しなければならない．

4.5 対　　数

ここでは，計算する対数は自然対数であるものとする．他の底に対する対数は簡単に誘導することができる．$X=(-1)^s2^e(1\cdot f)$ を単精度数であるとしよう．ただし，負数の対数は実領域で定義されていないため，$s=0$ である．すると，X の対数は次のようになる．

$$\log(X)=e\log(2)+\log(1\cdot f) \qquad (18)$$

しかし，(18)式の適用において，多少の注意が必要である．X が非常に1に近い時は，その結果は非常に小さなものになる．X が1に近いが $X<1$ である時は，e は -1 になるが，$(1\cdot f)$ は2に近くなる．これによって $\log(1\cdot f)$ は $\log(2)$ に非常に近くなる．このため，アルゴリズム計算においてよく知られている桁落ちの問題が起こるのである．X が1に近いが $X\geq 1$ である場合，e が0となるため何の問題も起こらない．

上記の問題を回避するためには，Xの指数をまずチェックし，それを非正規化して $X'=2^{e'}(1\cdot f')$ とし，$0.5\leq 1\cdot f'<2$ およびその指数が，X が1に近い時に，-1 になるようにすることである．そうしてから(18)式をこの新しい数にあてはめるのである．単精度に関しては $0.5\leq X<2$ について $\log(X)$ の計算全数検査し，誤差が小さいことを確認した．

速度の点では，(18)式で要求される加算は，ATA最終局面の加算と同時に実行できるものと考えられる．したがって，このアルゴリズムは平方根計算に対しても同じ時間，つまり単精度の場合に14τ を費やすと推定される．倍精度の場合に 25τ である．

Xが1に近いときに相対誤差が大きくなる．この問題は，正弦と余弦で述べたのと同じ方法で解決できる．

4.6 逆　正　接

逆正接の計算を二つの副問題に分けてみる．ひとつは $0\leq X<1$ の時の X の逆正接を計算することである．もうひとつは $X>1$ の時の X の逆正接を計算することである．後者は下記の関係によって前者に帰着する．

$$\arctan(X)=\frac{\pi}{2}-\arctan\left(\frac{1}{X}\right) \qquad (19)$$

速度の点では，$0\leq X<1$ の時は ATA 操作が一度必要であるが，一般的な X については，ATA 操作が二度，つまり単精度の場合に 28τ，倍精度の場合に 50τ 必要である．

Xが0に近いときに相対誤差が大きくなる．この問題は，正弦と余弦で述べたのと同じ方法で解決できる．

4.7 引き数2個の関数

X^Y，ATAN2(Y,X)，X/Y の計算は指数，対数，逆正接の計算に乗算を組み合わせて行われる．特に速い ATA 法は見つからなかった．

5. ニュートン反復法

逆数や平方根を計算するためによく使われる方法に，ニュートン反復法がある．この方法は二次で収束する．しかし，それは逆数の計算に1回の表引きと最低2回の乗算を必要とするので（平方根の場合は3回の乗算），倍精度計算までは ATA と ATA-M 法の方が速いと結論する．

6. おわりに

この論文では，我々は逆数，平方根，指数関数，正弦，余弦，対数と逆正接を IEEE 単・倍両精度浮動小数点規格で計算を行うためのアルゴリズムを提示した．このアルゴリズムは我々の ATA 法に基づく大きな表の使用を前提としている．ATA の特徴は，加減算および表引きが並行に行われることである．さらに，各々の算術の要素の精度が必要十分な長さに調節されるので，最適な計算時間や資源利用が得られる．この二つの特徴はソフトウェアでは容易に達成できない．

表 2 にこれらのアルゴリズムのスペース要件および速度推定をまとめた．付録Aに示す計算時間モデルに基づいて，我々は，これらのアルゴリズムは非常に高速であると結論する．ハードウェア資源を説明するため，表2に，N_S を使って必要な全表容量をビット単位で示す．図2から $N_S=2^{13}\times 66+2^{12}\times 38=696,320$ ビットを得る．図3では，使用される表を2種類に分けることができる．ひとつはすべての関数に共通するもので，もうひとつは各々の関数固有のものである．N_{ATA} と N_{DF} をそれぞれ前者と後者の大きさとしよう．図3から，$N_{ATA}=491,520$ ビットおよび $N_{DF}=184,320$ ビットを得る．本文で述べたように，関数を完全に計算するためにはしばしば単なる N_S や N_{DF} よりも多くの表が必要になる．また，正弦，余弦，対数および逆正接で，必要な表が，平方根などの表の4倍大きいのは，相対誤差に対する精度の要求から来ていることに注意されたい．絶対誤差でよければ，表は更に減らすことができる．それが総計の下に括弧で示されている．

表 2 ATA と ATA-M 法のアルゴリズムのメモリー量と速度
Table 2 Table sizes and speeds of ATA and ATA-M.

関数	精度	表サイズ（ビット）	速度 (τ_{FULL})
逆数	単	$1\times N_S$	0.88
平方根	単	$1\times N_S$	0.88
逆数平方根	単	$1\times N_S$	0.88
指数関数	単	$1\times N_S+30\times 2^{10}$	2.00
正弦	単	$4\times N_S$	2.00
余弦	単	$4\times N_S$	2.00
対数	単	$4\times N_S+30\times 2^8$	0.88
逆正接	単	$4\times N_S$	1.75
逆数	倍	$1\times N_{DF}$	1.63
平方根	倍	$1\times N_{DF}$	1.63
逆数平方根	倍	$1\times N_{DF}$	1.63
指数関数	倍	$1\times N_{DF}+60\times 2^{12}$	2.75
正弦	倍	$4\times N_{DF}$	2.07
余弦	倍	$4\times N_{DF}$	2.07
対数	倍	$4\times N_{DF}+60\times 2^{12}$	1.63
逆正接	倍	$4\times N_{DF}$	3.26
総計		18,634,240 ビット	
（絶対誤差の場合 8,066,560 ビット）			

スーパコンピュータの特徴は，パイプラインをフルに使うことである．本論文に示した ATA と ATA-Mの図面は，すべて完全にパイプライン化されているので，そのままスーパコンピュータに適用できる．なお，ラッチ遅れ時間がほとんどないシリーズ回路[14]を使用すれば，現在のスーパコンピュータよりパイプラインクロックを数倍高速化することも可能であろう．

ここで次の ATA 公式を考える．

$$4(3x^2y)=(2x+y)^3-(2x-y)^3-2y^3 \qquad (20)$$

これは(2)と同じ結果を出すが，(2)は $n+1$ ビットのアドレスの表で済むのに，(20)は $n+2$ ビットのアドレスの表が必要である．本論文に示した ATA 公式も最良のものでないかもしれない．ATA 公式の改良と2変数関数への拡張の可能性は将来に残された問題である．

謝辞 貴重なご意見を頂いた筑波大学の中澤喜三郎教授ならびに東京大学の小柳義夫教授に感謝します．

参考文献

1) Chen, T. C.: Automatic Computation of Exponentials, Logarithms, Ratios and Square Roots, *IBM J. Res. Dev.*, pp. 380-388 (1972).
2) Fairchild Corp., *F 100 K ECL Data Book.* (1982).
3) Gal, S.: Computing Elementary Functions: A New Approach for Achieving High Accuracy and Good Performance, *Lecture Notes in Comp. Sc. 235*, pp. 1-16 (1986).
4) Koren, I. and Zinaty, O.: Evaluating Elementary Functions in a Numerical Coprocessor Based on Rational Approximations, *IEEE Trans. on Comp.*, Vol. 39, pp. 1030-1037 (1990).
5) Nambu, H. et al.: A 1.5 ns, 64 Kb ECL-CMOS SRAM, *Digest of Technical Papers of the 1991 Symp. on VLSI Circuits*, pp. 11-12 (1991).
6) Nagai, T. and Hatano, Y.: Performance Evaluation of Mathematical Functions, *Supercomputer*, Vol. Ⅷ, No. 6, pp. 46-56 (Nov. 1991).
7) Tang, P. T. P.: Table-lookup Algorithms for Elementary Functions and Their Error Analysis, Argonne National Lab. Report MCS-P 194-1190 (1991).
8) Volder, J. E.: The CORDIC Trigonometric Computing Technique, *IRE Trans. Elec. Comp.*, Vol. EC-9, pp. 227-231 (1960).
9) Wallace, C. S.: A Suggestion for Parallel Multipliers, *IRE Trans. Elec. Comp.*, Vol. EC-13, pp. 14-17 (1964).
10) Walther, J. S.: A Unified Algorithm for Ele-

mentary Functions, *Proc. of Spring Joint Computer Conf.*, pp. 379-385 (1971).
11) IEEE Standard for Binary Floating Point Arithmetic, ANSI/IEEE Std 754-1985, IEEE (1985).
12) Hastings, C.: *Approximations for Digital Computers*, Princeton University Press (1955). (日本語訳 竹内 均：近似計算法，東京図書).
13) Harata, Y. et al.: A High Speed Multiplier Using a Redundant Binary Adder Tree, *IEEE J. Solid-State Circuits*, Vol. SC-22, No. 1, pp. 28-34 (1987).
14) 後藤英一，上川井良太郎，ウイリ・ヒュー：論理回路，日本特許公開番号 平 3-262327 (1991).

付録A 計算時間モデル

この付録では，我々のアルゴリズムについて行った計算時間の概算のもととなったモデルを説明する．基本時間は，標準的 OR/NOR の4入力あるいは XOR の2入力のゲート伝播遅延 τ である．

A1: ROM（読出し専用メモリー）

n-ビットアドレスのメモリー，つまりサイズが 2^n のメモリーについて，アクセス時間 τ_{ROM} は下記のとおりである．

$$\tau_{ROM}=\left\lceil\frac{n}{2}\right\rceil\tau \qquad (A1.1)$$

A2: 桁上げ先取り加算

ECL や TTL チップを用いる桁上げ先見加算の実現方法において，従来行われてきた方法では二つの余計なゲートの段階を含み，それが余計な 4τ の遅延の原因となっていた．これは恐らく，ピンや機能ブロックの数を減らす必要から生じたのであろう．しかし，我々は，コレクタ AND やエミッタフォロワを積極的に使用することによって，64 ビットまでのオペランドに対しては桁上げ先見加算が 2τ で達成できると考える．すなわち

$$\tau_{CLA}=2\tau \qquad (A2.1)$$

A3: ウオレストリーを用いたマルチ・オペランドの加算

マルチ・オペランドの加算は加算器のウオレストリーを用いて効率的に行うことができる[9]．$N_W(i)$ を，高さ i のウオレストリーの中で加算される入力数であるとしよう．トリーの各ノードは3個を加算し2出力をだす (3, 2) 加算器であるから，桁上げおよび合計は下記のようになる．

$$N_W(i+1)=\left\lfloor N_W(i)\times\frac{3}{2}\right\rfloor \qquad (A3.1)$$

表 A1 オペランド数に対するウオレストリー高さ
Table A1 Heights of the Wallace Tree for various operand sizes.

i	0	1	2	3	4	5	6	7	8	9	10
N_W	2	3	4	6	9	13	19	28	42	63	94

表 A2 乗算時間
Table A2 Multiplication times.

構 成	8×8	16×16	56×56
τ_{RECODE}	2τ	2τ	2τ
$\tau_{WALLACE}$	2τ	4τ	8τ
τ_{CLA}	2τ	2τ	2τ
合 計	6τ	8τ	12τ

$N_W(0)=2$ のときは**表A1**を得る．

各 (3, 2) 加算器は計算に1ゲート段階を要するから，ウオレストリーを用いて $N_W(i)$ オペランドの加算を行うのに要する時間はしたがって下記のようになる．

$$\tau_{WALLACE}=i \qquad (A3.2)$$

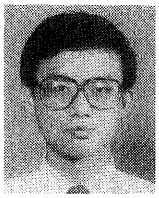
A4: 乗 算

高速の乗算は通常3段階で行われる．被加数の数を減らすためのリコード乗算，ウオレストリーによるマルチオペランド加算，そして最後に桁上げ先取り加算器を用いた桁上げの伝播である．4進リコード乗算を行うと，加算される被加数の数が半分になる．よって，$m\times m$ 乗算については，m 被加数でなく，リコーディングによって被加数の数を $\lceil m/2\rceil+1$ に減る．ウオレストリーの中の $\lceil m/2\rceil+1$ のマルチオペランドを合計する計算時間は上記に説明した．表 A1 は我々の計算時間モデルによって与えられる乗算時間の例を示している．しかし，最後の桁上げ先取り加算については，オペランドが m ビットで切り捨てられるものとしていることは注意すべきである．

完全な倍精度乗算が必要なときは，112 ビット以上の完全な桁上げ同化 (complete carry assimilation) が行われるように，56×56 の乗算の τ_{CLA} は 4τ にしておくべきである．また，(1ビット) の正規化桁シフト (normalization shifting) には 1τ を加算し，60 ビットの加算にはさらに 2τ 加算して，保護ビットについて丸めるようにする．そのとき全遅延時間は 16τ となる．

なお，冗長2進数表現を用いた加算法が最近提案されている[13]．この方法によれば回路設計が規則的にできるが，計算速度は幾分遅くなる．

(平成4年11月2日受付)
(平成5年5月12日採録)

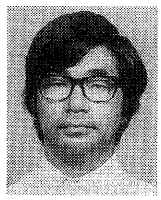


黄　栄輝

1964年生．1988年シンガポール国立大学情報システム計算機学科卒業．1991年同学科修士課程修了．1989年同学科助手．1990年新技術事業団創造科学技術推進事業後藤磁束量子情報プロジェクト研究員．1991年10月より理化学研究所後藤特別研究室研究員．主に計算機アーキテクチャの研究に従事．ACM，IEEE各会員．

後藤　英一（正会員）

1931年生．1953年東京大学理学部物理学科卒業．1958年同大大学院修了．1958年同大理学部助手，1959年同助教授．1962年理学博士．1968年理化学研究所情報科学研究室主任研究員（非常勤）．1970年東京大学理学部教授．1986年新技術事業団創造科学技術推進事業後藤磁束量子情報プロジェクトリーダー（兼務）．1991年4月より神奈川大学理学部教授．同年5月より理化学研究所後藤特別研究室特別招聘研究員（兼務）．現在，熱機関，量子限界感度磁束計，完全磁気遮蔽，計算機アーキテクチャ，無発熱計算，計算理工学等の研究に従事．



吉田　宣章（正会員）

1955年生．1978年東京大学理学部物理学科卒業．1983年同大大学院博士課程修了．理学博士．同年理化学研究所サイクロトロン研究室特別研究生．1985年東京大学理学部情報科学科助手．1991年7月より理化学研究所後藤特別研究室研究員．原子核物理学，数値解析，計算機アーキテクチャの研究に従事．日本物理学会会員．