数理解析研究所講究録
第215巻 1974年 55-65

CORDIC およびその変形にもとづく初等函数計算

京大数理研　一松 信

0. 概論

昨年この研究集会で，CORDICの紹介をした[5]．その双曲型の場合から，平方根，指数函数，対数函数の新しい計算法を考えたので，報告するが，しかしこれらはむしろ加法定理の応用[3]，ないしはある水準線のトレース[8]と考えたほうが見透しがよい．ここには主として理論的な原理をのべる．能率の比較は，次の小柳氏の講演にゆずる．この種の算法は，ファームウェアの一環であって，超小型機または超大型機に好適であることが，多くの人によって指摘されている．

1. Chen の算法 [8]

(これは研究集会の前日浅谷政明氏から[8]を教わり，予稿にはなかったものである)．

$f(x_0)=z_0$ を計算するために，補助変数 y を導入し，適

当な函数 $F(x,y)$ を作り,

$$F(x_0,y_0)=z_0, \qquad F(x_\omega, z_0)=z_0$$

であるようにする.—— じっさいには F は y について線型

$$F(x,y)=g(x)y+h(x), \qquad g(x_\omega)=1, \quad h(x_\omega)=0$$

とする. (x_0,y_0) から始めて, 変換

$$x_{k+1}=\varphi(x_k,y_k), \qquad y_{k+1}=\psi(x_k,y_k)$$

を反復し, つねに $F(x_k,y_k)=z_0$ であるようにする. この点列 (x_k,y_k) を $F(x_\omega,y_\omega)=y_\omega$ をみたす (x_ω,y_ω) ——とくに上の線型の場合には $g(x_\omega)=1$, $h(x_\omega)=0$ をみたす x_ω ——に近づけると, $z_0=f(x_0)=F(x_0,y_0)=F(x_k,y_k)=F(x_\omega,y_\omega)=y_\omega$ となって, y_ω が $f(x_0)$ の値 z_0 である.

以上はあまりに abstract であるが, たとえば指数函数 e^x に対しては, $F(x,y)=ye^x$, $x_\omega=0$ とし, 変換を

$$x_{k+1}=x_k-\alpha_k, \qquad y_{k+1}=y_k\times e^{\alpha_k}$$

とすればよい. このとき α_k を簡単な数とせず, e^{α_k} を簡単な(たとえば 1 ± 2^{-k}) 数になるように α_k を選ぶ. 対数函数 $\log x$ に対しては, $F(x,y)=y+\log x$, $x_\omega=1$, 変換は

$$x_{k+1}=\beta_k\times x_k, \qquad y_{k+1}=y_k-\log\beta_k$$

とし, β_k を簡単な数(たとえば 1 ± 2^{-k})とする. これはつぎにのべるSTLと本質的に同じになる. (ただし一個所, Chen 独得の名着想がある. 次節参照)

2. STL法[3]

上記のChenの算法を具体化すると，加法定理に基づくSpeckerのSTL (Successive Table Look-up) 法[3]と同じになる．$a_k = \log(1+2^{-k})$ $(k=1, 2, \cdots)$ をあらかじめ計算しておく．使用計算機は2進法のものとする

対数函数　$1/2 \leqq x_0 \leqq 1$ とする．$x := x_0$; $y := 0$ から始めて，つぎの算法を反復する．

(1)　$w := (1+2^{-k}) \times x$; if $w < 1$ then begin

　　$x := w$; $y := y - a_k$ end ;

xが十分1に近づくまで反復するが，$x = 1 - t \doteqdot 0$ ならば $\log(1-t) = -t - t^2/2 - \cdots$ で，t^2以降が無視できれば，$\log x \doteqdot x - 1$ としてよい．ゆえにNビット必要なときは，$k = N/2$まで反復して，あと $x-1$ をyに加えればよい．以上の計算は，シフトと加減算のみで可能である．マイクロプログラムによるlogの計算は，この方式がもっとも速くて確実のようである．

指数函数　$0 \leqq x_0 \leqq \log 2 = 0.693\cdots$ とする．$x := x_0$; $y := 1$ から始めて，つぎの算法を反復する．

(2)　$w := x - a_k$; if $w \geqq 0$ then begin

　　$x := w$; $y := (1+2^{-k}) \times y$ end ;

Nビット必要ならば，N回反復すればよい．ただし半語長程

度の乗法をいとわなければ，$N/2$ 回でやめ，$x \fallingdotseq 1$のとき $e^x \fallingdotseq 1+x$ を利用して，yに $1+x$ を掛ける —— じっさいには xをかけて xyを求め，それをyに加える．こうすれば，少しくらいの誤差はあとで補正できる．Chen は，kが大きければ

$$a_k = \log(1+2^{-k}) = \underbrace{2^{-k} - 2^{-2k-1} + 2^{-3k}/3 - \cdots}_{\fallingdotseq 2^{-k}}$$

であることを利用し，$w \geqq 0$ の判定を減法をせず，xのkビット目が1か否かで判定している．ビット単位の判定が速くできる計算機では，これは有用な能率化であろう．

その昔指数函数の近似において，区間を細分し，分点は簡単な有限小数αの$\log$とした方式があった．(こうするとα倍の乗法で丸め誤差が少なくなる)．上記の方式はこのような区間の細分を極度におしすすめ，最後に1次式で近似した方式とも解釈できる．また乗除算の計算で，2^{-k}を $a_k = \log(1+2^{-k})$ におきかえた擬似乗除算ともみられる．

Meggitt [2] は10進法でこのような擬似乗除算を論じているが，これを2進法に直すのは，かえって簡単である．

3. Non-restorting 式変形 (双曲型 CORDIC)

普通の2進法は，$0.a_1 a_2 \cdots = \sum 2^{-i} a_i$ とし，乗算では1なら加え，0ならそのままとする．これに対して，1

は+1, 0は−1を表わすとし, 1なら加え, 0なら引くという non-restoring 方式がある. これはとくに2の補数表示で正負共通に扱うとき有利である.

前節の算法は, 1なら加え, 0なら何もしない方式であるが, これを1なら加え, 0なら引く, という形に変形することができる. たとえば(2)を変形して, $x:=x_1$, $y:=y_1$ からはじめ, つぎの算法の反復とする:

(3) **if** $x \geqq 0$ **then begin** $x:=x-b_k$; $y:=y\times(1+2^{-k})$ **end**

else begin $x:=x+b_k$; $y:=y\times(1-2^{-k})$ **end**;

ただしこのときには,

$$b_k = \operatorname{arctanh} 2^{-k} = \frac{1}{2}\log\frac{1+2^{-k}}{1-2^{-k}}$$

である. $|x_1| \leqq \log 2$ のとき, $x_\omega = 0$ ならば,

$$y_\omega = y_1 \cdot \tilde{K} \cdot e^{x_1}, \quad \tilde{K} = \prod_{k=1}^{\omega-1}(1-2^{-2k})^{1/2} \fallingdotseq 0.84\cdots$$

$y_1 = 1/\tilde{K}$ ととれば, e^{x_1} 自体をうる.

これはCORDICの双曲型の算法と本質的に同一である. ただし最後に $x \fallingdotseq 0$ になったとき, $e^x \fallingdotseq 1+x$ として, これをかける補正をしないと, 反復計算だけでは収束しない隙間を生ずる. それを防ぐもっとも簡単な手段は, k が

$$k_0=1, \quad k_{i+1}=3k_i+1, \quad \text{すなわち } 4, 13, 40, \cdots$$

において, 同じ定数でもう1回反復することである (厄年留年法と仮称しよう).

log については，(3)の逆として，$x:=x_1$，$y:=0$ からはじめて，つぎの算法を反復する：

(4) $\underline{\text{if}}\ x \leqq t\ \underline{\text{then}}\ \underline{\text{begin}}\ x:=(1+2^{-k})\times x;\ y:=y-b_k\ \underline{\text{end}}$
$\underline{\text{else}}\ x:=(1-2^{-k})\times x;\ y:=y+b_k\ \underline{\text{end}};$

$b_k = \operatorname{arctanh} 2^{-k}$, $\qquad y \to \log t$.

ただし $1-2^{-1}=1/2$ が小さすぎるので，反復は $k=2$ から始める．x_1 は $\sqrt{3}/2\tilde{K} = 1.045723138\cdots$ とする．収束域が $0.57\cdots \leqq t \leqq 1.76\cdots$ と狭いため，$3/4 \leqq t \leqq 3/2$ または $1/\sqrt{2} \leqq t \leqq \sqrt{2}$ と標準化しなければならない([7]参照)．

Non-restoring 方式の長所は，1に対して対称であることと，$b_k = 2^{-k} + 2^{-3k}/3 + \cdots$ であって，Nビット必要ならN/3から先は $b_k = 2^{-k}$ としてよいため，定数が少なくてすむ点であるが，反面1でも0でも必ず掛ける手間がかかる上に，log の1に近い引数のとき，1 ± 2^{-k} をかけて反復するため精度がおちたりして，本来のSTL法に劣るようである．

4. 複素数による $\cos x$，$\sin x$ の計算

指数関数の公式に複素数を代入すると

$$\exp(iz) = \left[\prod_{k=0}^{n} (1+a_k^2)^{-1/2}(1+ia_k)\right]\cdot \exp\left(iz - \sum_{k=0}^{n} \arctan a_k\right)$$

となるから，$z_0 := t$ $(|t| \leqq \pi/2)$ からはじめて

(5) <u>if</u> $z \geqq 0$ <u>then</u> $a := 2^{-k}$ <u>else</u> $a := -2^{-k}$;

$z := z - \arctan a$ ($\pm\arctan 2^{-k}$ を表)

とする反復により，必要なビット数まで反復すると

$$\cos t = \mathrm{Re}\left[K^{-1}\prod_{k=0}^{n}(1+ia_k)\right],\quad \sin t = \mathrm{Im}\left[K^{-1}\prod_{k=0}^{n}(1+ia_k)\right]$$

$$K = \prod_{k=0}^{n}(1+2^{-2k})^{1/2} = 1.72\cdots$$

となる．$\exp(iz) = x + iy$ とすれば，上記の変換は

$$x_{k+1} = x_k - a_k y_k,\quad y_{k+1} = y_k + a_k x_k \quad (a_k = \pm 2^{-k})$$

を同時に並行して適用したものにほかならず，これはCORDIC そのものである([9])．

これを逆に適用すれば，arctan がえられる．ただし arctan については，log を複素変数とした形で，$a_k = 2^{-k}$ または 0 とした restorting 方式の反復も可能である．じっさい [3] にのべられている算法は，その形である．なお arcsin，arccos も同じような方式で，直接に計算することができる（[9] 参照）．

上記の x, y の変換の最初の式だけを $x_{k+1} = x_k + a_k y_k$ にかえたのが，双曲型の CORDIC であり，ここで $x_k = y_k$ として，$x_{k+1} = x_k(1 \pm 2^{-k})$ とすると，§3 でのべた non-restorting 方式の指数函数（および対数函数）の計算である．双曲座標では $|y| < x$ のはずだが，$x = y$ としても

正しいことは，[7] に証明したとおりである。

5、平方根の計算

平方根の計算法は，Newton 反復がもっともよいようで，すでに研究されつくしたようであるが，Chen の算法で，$F = y x^{-1/2}$，$x_\omega = 1$ とする方法が可能である。——なお $F = y/x$，$x_\omega = 1$ とすれば除法になる。ここでは 3. の双曲型 CORDIC によって，偏向成分を 0 に近づけると，x 成分に $\tilde{K}\sqrt{x_1^2 - y_1^2}$ がえられることを利用した方法をのべる（[7]）

$x_1 := t + c$；$y_1 := t - c$ から始めれば，

$$x_n = 2\tilde{K}\sqrt{c}\sqrt{t}$$

となるから，$2\tilde{K}\sqrt{c} = 1$ であるように c をとっておけば，直接に $\sqrt{t}$ をうる。このとき CORDIC の z 成分には

$$\operatorname{arctanh}\frac{y_1}{x_1} = \frac{1}{2}\log\frac{x_1 + y_1}{x_1 - y_1} = \frac{1}{2}\log\frac{t}{c}$$

がえられるけれども，平方根と対数とが同時に必要なことはほとんどないので，平方根の計算用には，z 成分および定数 b_k を捨てて，次のようにすればよい。$x := t + c$，$y := t - c$ から始め，つぎの計算をくりかえす：

(6) **if** $y \geq 0$ **then** $\boxed{x := x - 2^{-k} y;\ y := y - 2^{-k} x}$

else $\boxed{x := x + 2^{-k} y;\ y := y + 2^{-k} x}$;

ただしここで $\boxed{}$ 内は同時に平行して実行する。すなわちあとの式の x は、前の式でおきかえられた x でなく、おきかえられる前の x である。逐次処理しかできなければ、たとえば

$$\underline{\textbf{begin}}\ w := x - 2^{-k}y\ ;\ y := y - 2^{-k}x\ ;\ x := w\ \underline{\textbf{end}}$$

というように解釈しなければいけない。また収束を保証するために、$k = 4, 13\ (40, 121)$ で同じ k で2度計算する。

$$c = 1/4K^2 = 0.3645122921 9 \cdots$$

ただしこれは無限乗積の値である。TOSBAC-3400（仮数部37ビット）では、$1/4 \leqq t \leqq 1$ において、なるべく誤差が全体的に最小になるように半実験的に定めた所

$$c = 0.364512292 26$$

とするのが最適であった。（これは二進十進変換の誤差の影響らしい）。

このとき

$$(x_n^2 - y_n^2)^{1/2} = x_n\left(1 - \frac{y_n^2}{2x_n^2} - \cdots\right)$$
$$= K(x_1^2 - y_1^2)^{1/2}$$

であるから、$y_n^2/2x_n^2$ が無視できるまで、したがってNビット必要なら N/2 回で反復をやめてよい。じっさいTOSBAC-3400の実験でも、$k = 38$ まで反復するより、$k = 19$ で止めたほうが、かえって誤差が少なかった。

以上はまったくシフトと加減算のみで実行する方法である

が，除算をいとわなければ，(6)を何段階かやってから，その近似値 x_k を出発値として，Newton 法にきりかえてもよい。この場合には，初期値を $\sqrt{t}$ の折れ線近似（必ずしも最良近似ではない）で作りだしたものとも考えられる。

この方式の収束域は， $e^{-2B} \leqq t/c \leqq e^{2B}$，
$(B = \sum_{k=0}^{n} \operatorname{arctanh} 2^{-k} \doteqdot 1.117,\quad e^{2B} \doteqdot 9.34\cdots\quad,\ c \doteqdot 0.36\cdots)$
で，$\{1/4 \leqq t \leqq 1\}$ も $\{1/16 \leqq t \leqq 1\}$ も十分に含まれるが，$t=1$ に近いとき，はじめの方の $x_k > 1$ で"あふれ"を生ずるので，$\{1/8 \leqq t \leqq 1/2\}$ に標準化するほうがよい。このように狭域でよければ，$k=2$ $(a_2 = 1/4)$ から始め，c を $k=2$ からの積，したがって前記の値の $3/4$ 倍 $(c = 0.2733842192 0)$ とし，$k=7\,(=2\times3+1)$ において同じ k で反復するような修正も可能である（反復が2回へり，精度もよくなる.）.

反対に収束域を広くしたければ，$a_0 = 3/4$ から始め，$a_k = 2^{-k}$ のうち，a_4 と a_5 の中間に $a_{4.5} = a_4 \times (3/4)$，$a_{13}$ と a_{14} の間に $a_{13.5} = a_{13} \times (3/4)$ を補う（定数 b_k も補いcも変更する）手がある。プログラムは厄介になるが，$c \doteqdot 0.81$，収束域はほぼ $0.014 \leqq t \leqq 50$ となり，$\{1/64 \leqq t \leqq 16\}$ を含むので，16進のまま始めるのに有用である。このときには，a_4 くらいまででやめて，あとを Newton 法にきりかえるほうが有利と思われる.

参考文献

[1] J.E.Volder, Binary computation algorithms for coordinate rotation and function generation, Convair Report IAR-1 148, 1956.

[2] J.E.Meggitt, Pseudo division and pseudo multiplication processes, IBM Research & Devel., 6 (1962), 210-226.

[3] W.H.Specker, A class of algorithms for Ln x, Exp x, Sin x, Cos x, $Tan^{-1}x$, $Cot^{-1}x$, IEEE Trans. on Elec. Comp. 14 (1965), 85-86.

[4] J.S.Walther, A unified algorithm for elementary functions, Spring Joint Comp. Conference 1971, 379-385.

[5] 一松信, 初等函数の統一的計算法 —CORDIC による— 数理解析研講究録 190 (1973), 156−165.

[6] 荻原博・渡辺勝正・小柳滋, マイクロプログラムによる初等関数近似, 第14回情報処理学会 年会予稿集 (1973), 171−172.

[7] S.Hitotumatu, A new method for the comptutation of square root, exponential and logarithmic functions through hyperboloc CORDIC, will appear in Rev. Inst. Comp. Cluj.

[8] Tien Chi Chen, The authomatic computation of exponential, logarithms, ratios and square roots, IBM Reaearch RJ 970(#16884), 1972, p.32

[9] M.D.Perle, Cordic Technique reduces trigonometric function look-up, Computer Design, 1971 June, p.72-78 (但し p.73, 75, 77は広告).