

初等函数の統一的計算法 CORDIC による

京大教解研 一松 信

0. 要旨

初等函数（むしろ組込み標準函数）の計算手法はほぼ確立している。しかし近年「非近似式」的な手法がいろいろ考えられており、特殊目的には有用である。

最近 Walther [1] は、座標変換にもとづく CORDIC という統一的算法を提唱した。これはとくに乗除算の金物をもたないミニコンの場合に有利である。 $\arctan$ の計算が乗除算（同じ方法による）よりも早いとなると、複素数の乗除算も、極座標に直して実行するほうが早いかもしれない。いろいろとおもしろい応用が考えられるので、とりあえず概要を紹介したい。

1. 計算の原理

平面の座標 (x, y) の点 P （便宜上 $|y| \leq x$ とする）に対して、拡張された極座標 R, A を次の式で定義する：

$$(1) \quad R = (x^2 + m y^2)^{1/2}, \quad A = m^{-1/2} \arctan(m^{1/2} y/x) \\ [x = R \cos(m^{1/2} A), \quad y = m^{-1/2} R \sin(m^{1/2} A)]$$

ただし m は 定数 $+1, 0, -1$ であり, $m=0$ のときは $m \rightarrow 0$ の極限をとる. 具体的に

は, $m=0$ のときは

$$A = y/x, \quad y = RA$$

となり, $m=-1$ のときは

$$A = \operatorname{arctanh} y/x$$

$x = R \cosh A, \quad y = R \sinh A$ となる. (x, y) を通る $R =$

一定の曲線 ($m=1$ なら円, $m=0$ なら垂直線, $m=-1$ なら直角双曲線) と $\overline{OP}$, x 軸で囲まれる部分の面積を S とすると

$$(2) \quad A = 2S/R^2 \quad \text{である}$$

さて (x, y) をつぎの式で「座標変換」する:

$$(3) \quad x^* = x + m y \delta, \quad y^* = y - x \delta$$

δ は定数である. このとき「極座標」では

$$A^* = A - \alpha, \quad R^* = R K$$

$$\alpha = m^{-1/2} \operatorname{arctan}(m^{1/2} \delta), \quad K = (1 + m \delta^2)^{1/2}$$

となる. そこで新しい変数 z を導入し, (3) と同時に変換

$$(4) \quad z^* = z + \alpha$$

を行なう. さて初期値 (x_0, y_0, z_0) から適当な列 δ_i (α_i, K_i をそれに対する値とする) をとって変換 (3), (4) を n 回くりかえして, (x_n, y_n, z_n) をえたとすると, 「極座標」では

$$(5) \quad A_n = A_0 - \alpha, \quad R_n = R_0 \cdot K, \quad z_n = z_0 + \alpha$$

$$\alpha = \alpha_0 + \dots + \alpha_{n-1}, \quad K = K_0 \cdot K_1 \cdots K_{n-1}$$

となる。直交座標にもどすと、つぎのようになる。

$$(6) \quad x_n = K [x_0 \cos(m^{1/2} \alpha) + y_0 m^{1/2} \sin(m^{1/2} \alpha)]$$

$$y_n = K [y_0 \cos(m^{1/2} \alpha) - x_0 m^{-1/2} \sin(m^{1/2} \alpha)].$$

さて δ_i をうまく選ぶ (実用上では $\pm 2^{-i}$ とする), それに応ずる α_i を求めておき, $y_n \rightarrow 0$, または $z_n \rightarrow 0$ とすると, 他の変数の値は下記の表の値に近づく. この中に, 標準組込み関数がすべて含まれていることに注意する.

I $m=1$ のとき: 1. $z \rightarrow 0$ とすれば

$$x \rightarrow K^+(x_0 \cos z_0 - y_0 \sin z_0) \quad K^+ = \pi(1 + \delta_i^2)^{1/2}$$

$$y \rightarrow K^+(y_0 \cos z_0 + x_0 \sin z_0),$$

$$2. \quad y \rightarrow 0 \quad \text{とすれば,} \quad x \rightarrow K^+ \sqrt{x_0^2 + y_0^2}, \quad z \rightarrow z_0 + \arctan(y_0/x_0)$$

II $m=0$ のとき: 1. $z \rightarrow 0$ とすれば x 不変, $y \rightarrow y_0 + x_0 z_0$

$$2. \quad y \rightarrow 0 \quad \text{とすれば,} \quad x \text{ 不変,} \quad z \rightarrow z_0 + y_0/x_0.$$

III $m=-1$ のとき: 1. $z \rightarrow 0$ とすれば

$$x \rightarrow K^-(x_0 \cosh z_0 + y_0 \sinh z_0) \quad K^- = \pi(1 - \delta_i^2)^{1/2}$$

$$y \rightarrow K^-(y_0 \cosh z_0 + x_0 \sinh z_0),$$

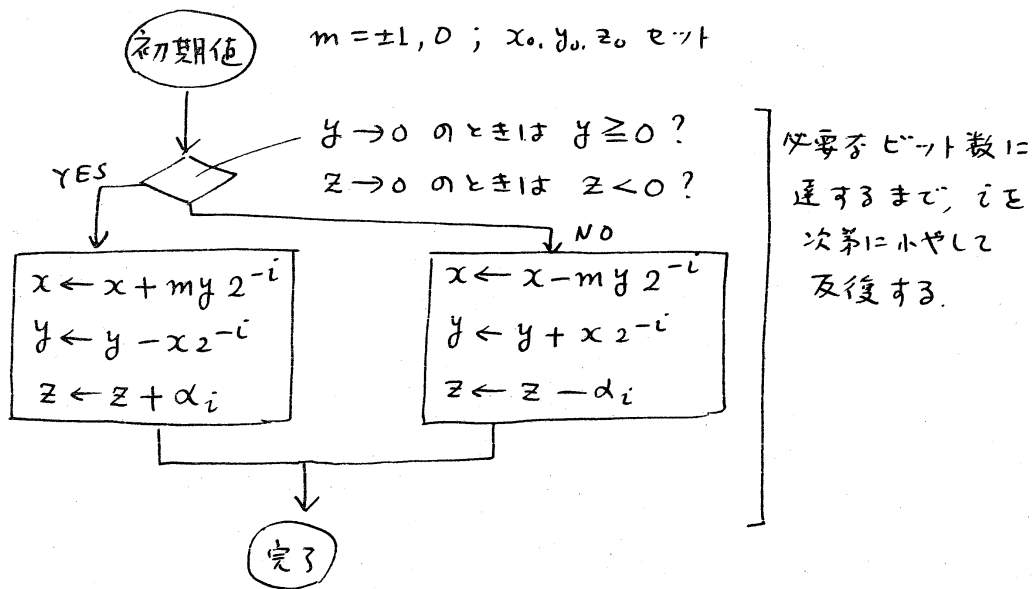
$$2. \quad y \rightarrow 0 \quad \text{とすれば,} \quad x \rightarrow K^- \sqrt{x_0^2 - y_0^2}, \quad z \rightarrow z_0 + \operatorname{arctanh}(y_0/x_0)$$

2. 具体的な算法.

具体的に各種の初等函数を計算するには、下記の表のように
に助変数や初期値をとればよい:

函数	m	0に近 量	x_0	y_0	z_0	答
$\sqrt{a}$	-1	y	$(1/k^-)(a+1/4)$	$(1/k^-)(a-1/4)$	0	x
e^a	-1	z	$1/k^-$	$1/k^-$	a	x, y
$\log a$	-1	y	$a+1$	$a-1$	0	$2z$
$\cos a$	1	z	$1/k^+$	0	a	$\begin{cases} x \\ y \end{cases}$
$\sin a$						
$\arctan a$	1	y	$\begin{cases} a < 1 & 1 \\ a > 1 & a \end{cases}$	$\begin{cases} a \\ -1 \end{cases}$	$\begin{cases} 0 \\ \pi/2 \end{cases}$	z
$c+a \times b$	0	z	a	c	b	y
b/a	0	y	a	b	0	z
$(a, b) \rightarrow (R, \theta)$	1	y	a/k^+	b/k^+	0	$\begin{cases} R \rightarrow x \\ \theta \rightarrow z \end{cases}$
$(R, \theta) \rightarrow (a, b)$	1	z	R/k^+	0	θ	$\begin{cases} a \rightarrow x \\ b \rightarrow y \end{cases}$

もちろん前もって引き数の還元 (たとえば $\sin, \cos$ では
 $0 \sim \pi/2$ に直す) が必要である. $m=0$ のときは $\alpha=\delta, k=1$
であるが, $m=\pm 1$ に対しては, 2^{-i} に対する α_i ($\arctan 2^{-i}$,
 $\operatorname{arctanh} 2^{-i}$) ^(や定数 $R^\pm$) をあらかじめ計算しておく. (3) の計算は, 2
進法ならば, 四則と加減算だけでできる. そしてつぎの流れ
図のように反復する.



ただし $my 2^{-i}$ は乗法を行わず, $m = \pm 1$ に応じて $x + y 2^{-i}$ または $x - y 2^{-i}$ とプロセッサ自体を切りかえる。

原論文では精度を保つため, $i = 0$ (または 1) からでなく, 初期値に応じて, その指数部を i の初期値に使うなどの工夫がこらされている。

$m = 0$ のときの乗除算は, 実質的には non-restoring 法によるビット毎の乗除算と同じことである。

3. 収束条件

収束条件は, $|\delta_i|$ に対する $|\alpha_i|$ (以下これを α_i と略記する) について, その土まうけた和で必要な範囲が覆えるかどうかできる。各 i について, 条件

$$(7) \quad \alpha_i - \sum_{j=i+1}^{n-1} \alpha_j \leq \alpha_{n-1}$$

が成立すれば、初期値 A_0 は

$$(8) \quad |A_0| \leq \sum_{j=0}^{n-1} \alpha_j + \alpha_{n-1}$$

の範囲にとるとき、必ず収束することが示される。——詳しく
いえば最後は $|A_n| \leq \alpha_{n-1}$ となり、その精度以内で正しく求
められる。

$m=0$ なら、 $\alpha_i = 2^{-i}$ とすると、(7) は等号で成立する。
 $m=1$, $\alpha_i = \arctan 2^{-i}$ なら、 $x > 0$ で $\arctan x$ が上に凸
なので、(7) は成立する。したがって $i=0$ (または 1) から、
順次 $\delta_i = \pm 2^{-i}$ としてよい。 $i=0$ からはいじめれば、(8) の
右辺は $m=1$ のとき $1.74 \dots > \pi/2$ で、十分 $0 \sim \pi/2$
を覆う。しかし $m=-1$ のときには、 $\alpha_i = \operatorname{arctanh} 2^{-i}$
とすると、(7) は成立しない ($>$ となる)。これでは困るの
であるが、テイラー展開した 1 次の項は (7) で等号なので、
3 次の項を消すために余分の項を加えると、 $\alpha_i = \operatorname{arctanh} 2^{-i}$
とするとき、 $k \leq 3i+1$ ならば、つねに

$$(7') \quad \alpha_i - \sum_{j=i+1}^{n-1} \alpha_j - \alpha_k < \alpha_{n-1}$$

が成立する (高次の項をすべて評価しても、 $k \leq 3i+1$ でよい)。

したがって $i=1$ からはいじめ、 $3i+1$ に相当する 4, 13, 40,
121, ... のときだけ、もう一度同じ δ_i, α_i の値で反復すれ
ば、収束する。10 桁程度ならば 4 と 13, 40 桁程度まで

らば、40 を加えただけですむから、実用上はさして困難はない。

引き数の還元をうまくすれば、実用上必要を上記の諸函数の変域は、すべてこのような収束域^中に含ませることができる。

4. 実験結果

このシミュレーションを FORTRAN で書いて実験した例を別表に掲げる。DATA の次の最初の数字は 1 が $z \rightarrow 0$, 0 が $y \rightarrow 0$ とする指示であり、次の数字は m である。つぎの値は、初期値 x_0, y_0, z_0 であり、下の行は結果の x, y, z である。ただし $m=1$ のときは K^+ , $m=-1$ のときは K^- で割った値にしてある。各 DATA や結果の意味を記入しておいた。E-21, E-22 などのついた値は、0 とみなすべき量である。これは倍長計算 (74 ビット) で実施したが、単長でも、毎回のビット数の範囲で正しく答がでている。

なおこの実験にあたって、定数 $\arctan 2^{-i}$, $\operatorname{arctanh} 2^{-i}$ の高精度 (10 進 50 桁) の値を計算してくれた日大・山下真一郎氏に、感謝の詞をのべてたい。

$$1/K^+ = 0.85878 \ 53364 \ 80427 \ 52634 \ 47$$

$$1/K^- = 1.20749 \ 70677 \ 63072 \ 128877$$

5. 実用価値と応用の可能性

この方法の長所は、加減算とシフトと判断（正負の）だけでできることである。したがって乗除算の金物をもたないミニコン用にとくに適している。—— じつさい PDP で実用に行っているらしい。欠点は定数を多く要することであるが、これは読出し専用記憶でよい。また $\arctanh 2^{-i}$ も $\arctan 2^{-i}$ も、 $2^{-3i}/3$ 以上の項を無視すれば、 2^{-i} で近似できるから、じつさいには必要のビット数の $1/3$ （60ビットなら 20個）まで用意すればすむ。FORTRAN でのシミュレーションでは困難だが、 x, y, z 用の 3 個のレジスタを置き、同時並行演算をすれば、非常に早くできる。

他の用途として、科学技術計算用の超大型機用に、専用の回路をつけるか、あるいは専用のソフトウェア・パッケージをつけることも考えられる。定数さえ用意すれば、必要なビット数まで同一の反復ですむから、四則演算の金物のない超高精度計算（たとえば 4 倍長用）などには、かえって他の近似式などによるより早いかもしれない。

また直交座標 $\longleftrightarrow$ 極座標の変換が一度でできるし、

$$x_0 \cos a - y_0 \sin a, \quad y_0 \cos a + x_0 \sin a$$

の計算が一度にできるから、複素数の乗除算に有利である。

とくに除法は、直交座標のまま ^(で実行する) よりも最高半分の時間ですむ

可能性がある。——これらのテストも、いずれ折返みて、やってみたいと思う。

この着想は、個々には以前からあったらしいが、このような観点でまとめたアイディアには敬服のほかはない。

参考文献

- [1] J. S. Walther, A unified algorithm for elementary functions, Spring Joint Comp. Conf. 1971, p379-385