数理解析研究所講究録
190巻 1973年 156-165

初等函数の統一的計算法
CORDIC による

京大数解研　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) $R = (x^2 + my^2)^{1/2}$,　$A = m^{-1/2} \arctan(m^{1/2} y/x)$

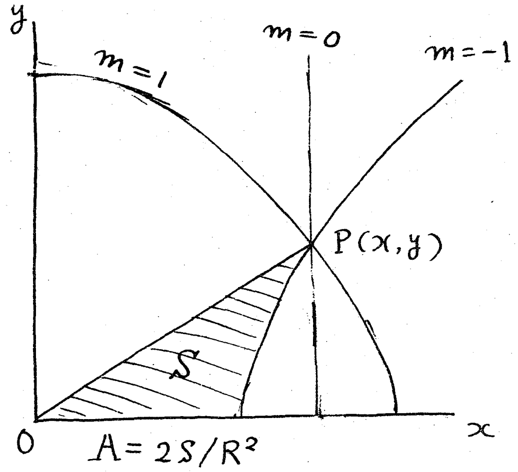
$[x = R\cos(m^{1/2}A),\ y = m^{-1/2} R \sin(m^{1/2} A)]$

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

$A=y/x$，　$y=RA$

となり，$m=-1$ のときは

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



$x=R\cosh A$，$y=R\sinh A$　　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)　$A=2S/R^2$　　　　である．

さて (x, y) をつぎの式で「座標変換」する：

(3)　$x^*=x+my\delta$　，$y^*=y-x\delta$

δ は定数である．このとき「極座標」では

$A^*=A-\alpha$，　$R^*=RK$

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

となる．そこで新しい変数 z を導入し，(3) と同時に変換

(4)　$z^*=z+\alpha$

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

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

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

となる．直交座標にもどすと，つぎのようになる．

$$(6) \quad \begin{aligned} x_n &= K[x_0 \cos(m^{1/2}\alpha) + y_0 m^{1/2} \sin(m^{1/2}\alpha)] \\ y_n &= K[y_0 \cos(m^{1/2}\alpha) - x_0 m^{-1/2} \sin(m^{1/2}\alpha)]. \end{aligned}$$

さて δ_i をうまく選び（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，$y_n \to 0$，または $z_n \to 0$ 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，標準組込み函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. $z \to 0$ とすれば

$$x \to \overset{+}{K}(x_0 \cos z_0 - y_0 \sin z_0) \qquad \overset{+}{K} = \Pi(1 + \delta_i^2)^{1/2}$$

$$y \to \overset{+}{K}(y_0 \cos z_0 + x_0 \sin z_0),$$

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

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

2. $y \to 0$ とすれば，　x 不変，$z \to z_0 + y_0/x_0$

III　$m = -1$ のとき：　1. $z \to 0$ とすれば

$$x \to \overset{-}{K}(x_0 \cosh z_0 + y_0 \sinh z_0) \qquad \overset{-}{K} = \Pi(1 - \delta_i^2)^{1/2}$$

$$y \to \overset{-}{K}(y_0 \cosh z_0 + x_0 \sinh z_0),$$

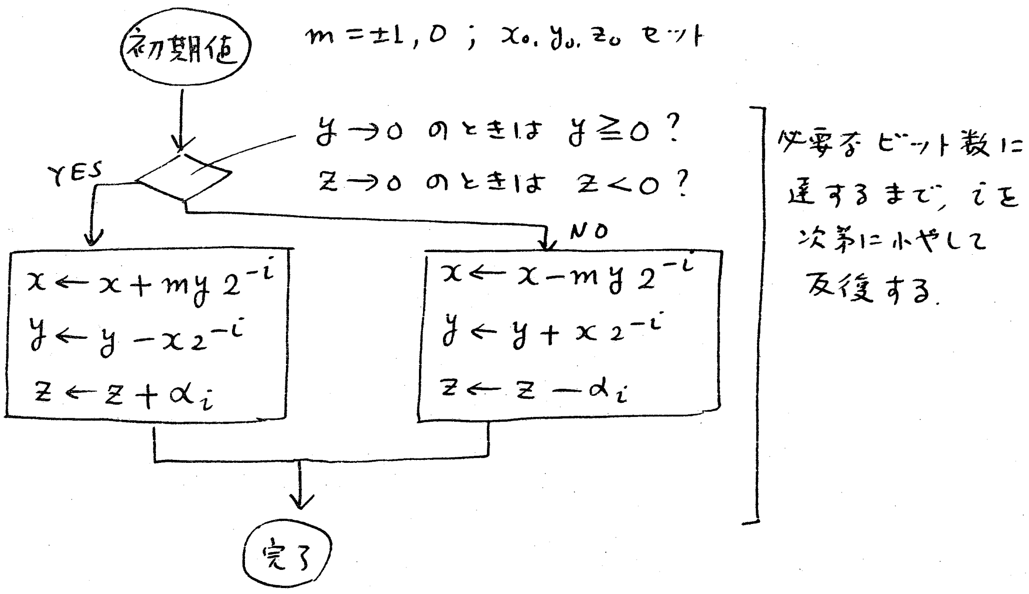
2. $y \to 0$ とすれば，　$x \to \overset{-}{K}\sqrt{x_0^2 - y_0^2}$，$z \to z_0 + \operatorname{arctanh}(y_0/x_0)$

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，下記の表のように助変数や初期値をとればよい：

函数	m	0にする量	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	$2z$
$\cos a$, $\sin a$	1	z	$1/K^+$	0	a	x, y
$\arctan a$	1	y	$\lvert a\rvert<1$: 1	a	0	z
			$a>1$: a	-1	$\pi/2$	
$c+a\times b$	0	z	a	c	b	y
b/a	0	y	a	b	0	z
$(a,b)\to(R,\theta)$	1	y	a/K^+	b/K^+	0	R…x, θ…z
$(R,\theta)\to(a,b)$	1	z	R/K^+	0	θ	a…x, b…y

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，$m=\pm1$ に対しては，2^{-i} に対する α_i ($\arctan 2^{-i}$, $\mathrm{arctanh}\ 2^{-i}$)（や定数 $K^\pm$）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 $my2^{-i}$ は乗法を行なわず，$m=\pm1$ に応じて $x+y2^{-i}$ または $x-y2^{-i}$ とプログラム自体を切りかえる．

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 ± をつけた和で必要な範囲が覆えるか否かできまる．各 i について，条件

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

が成立すれば，初期値 A_0 を

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

の範囲にとるとき，必ず収束することが示される．— 詳しくいえば最後に $|A_n| \leqq \alpha_{n-1}$ 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．$i=0$ からはじめれば，(8) の右辺は $m=1$ のとき $1.74\cdots > \pi/2$ 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，$k \leqq 3i+1$ ならば，つねに

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

が成立する（高次の項までですべて評価しても，$k \leqq 3i+1$ でよい）．したがって $i=1$ からはじめて，$3i+1$ 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 桁程度までな

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，実用上必要な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 $z \to 0$，0 が $y \to 0$ 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，所要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，感謝の詞をのべたい．

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

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

0··Y ; 1··Z ; M, ±1,0

```
DATA  1  1       0.1000000000000000000000E 01    0.                                π/6   0.5235987755982988730770E 00
                 0.8660254037844386467630E 00    0.4999999999999999999995E 00           -0.1251113851930075242291E-22
                   cos(π/6)                        sin(π/6)

DATA  1  1       0.5000000000000000000000E 00   -0.5000000000000000000000E 00      π/4   0.7853981633974483096160E 00
                 0.7071067811865475243998E 00    0.2974635988364792994939E-21           -0.1540806591065663230782E-22
                 [cos(π/4) + sin(π/4)]/2 → 1/√2   [cos(π/4) − sin(π/4)]/2

DATA  1  1  √3/2 0.8660254037844386467640E 00    0.5000000000000000000000E 00      π/12  0.2617993877991494365370E 00
                 0.7071067811865475244017E 00    0.7071067811865475243989E 00           -0.1633104579153969965634E-22
                   cos(π/4)                        sin(π/4)

DATA  1  1       0.                              0.1000000000000000000000E 01  arctan 3/4 0.6435011087932843868030E 00
                -0.5999999999999999999992E 00    0.7999999999999999999994E 00           -0.6385040166858869553003E-25
                 − sin(arctan(3/4)) → −0.6         0.8

DATA  0  1       0.4000000000000000000000E 00    0.3000000000000000000000E 00            0.
                 0.4999999999999999999995E 00   -0.5445049687972300626780E-23            0.6435011087932843868034E 00
                   √(0.4² + 0.3²)                                                          arctan(3/4)

DATA  0  1       0.5000000000000000000000E 00    0.5000000000000000000000E 00            0.
                 0.7071067811865475243997E 00   -0.1846872948733009838116E-22            0.7853981633974483096159E 00
                   √(0.5² + 0.5²)                                                          arctan 1

DATA  1  0       0.3000000000000000000000E 00    0.1000000000000000000000E 00            0.4000000000000000000000E 00
                 0.3000000000000000000000E 00    0.2200000000000000000000E 00            0.
                                                   0.3 × 0.4 + 0.1

DATA  1  0  7/15 0.4666666666666666666670E 00   -0.2000000000000000000000E 00      3/7   0.4285714285714285714290E 00
                 0.4666666666666666666670E 00    0.3597369094440138642414E-21            0.
                                                   3/7 × 7/15 − 1/5

DATA  0  0       0.3000000000000000000000E 00    0.2000000000000000000000E 00            0.
                 0.3000000000000000000000E 00    0.7058607893785836156903E-23            0.6666666666666666666666E 00
                                                                                           0.2/0.3

DATA  1 -1       0.1000000000000000000000E 01    0.                                log2  0.6931471805599453094170E 00
                 0.1249999999999999999997E 01    0.7499999999999999999935E 00           -0.1412666234432096691944E-22
                   cosh(log 2) → 1.25              sinh(log 2) → 0.75

DATA  1 -1       0.5000000000000000000000E 00    0.5000000000000000000000E 00            0.1000000000000000000000E 01
                 0.1359140914229522617693E 01    0.1359140914229522617693E 01           -0.1700412400397961311705E-22
                   e/2                             e/2

DATA  1 -1      -0.1000000000000000000000E 01    0.                                     -0.5000000000000000000000E 00
                -0.1127625965206380785231E 01    0.5210953054937473616336E 00            0.1230012749348499441709E-22
                   − cosh 0.5                      sinh 0.5

DATA  0 -1       0.7500000000000000000000E 00    0.2500000000000000000000E 00            0.
                 0.7071067811865475244003E 00   -0.1105949437016382727608E-22            0.3465735902799726547186E 00
                   √(0.75² − 0.25²) → 1/√2                                                 ½ log 2

DATA  0 -1       0.5000000000000000000000E 00   -0.3000000000000000000000E 00            0.
                 0.3999999999999999999997E 00    0.4221091041326623205554E-23           -0.6931471805599453094169E 00
                   √(0.5² − 0.3²)                                                          − log 2

DATA  0 -1  1/√2 0.7071067811865475244011E 00    0.5000000000000000000000E 00            0.
                 0.4999999999999999999996E 00   -0.1179501934898165139597E-22            0.8813735870195430252320E 00
                   √((1/√2)² − 0.5²)                                                       log(√2 + 1)
```

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．――じっさいPDPで実用にしているらしい．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 $\operatorname{arctanh} 2^{-i}$ 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，じっ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個のレジスタをおき，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, いずれ折をみて, 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, p.379−385