REVUE D'ANALYSE NUMÉRIQUE ET DE LA THÉORIE DE L'APPROXIMATION, Tome 3, N° 2, 1974, pp. 173–180

A NEW METHOD FOR THE COMPUTATION OF SQUARE ROOT, EXPONENTIAL AND LOGARITHMIC FUNCTIONS THROUGH HYPERBOLIC CORDIC

by

SIN HITOTUMATU

(Kyoto)

§ 1. Introduction

Walther [1] proposed a unified algorithm for elementary functions due to coordinate transformation. In a previous paper [2], the author has discussed mainly the circular case ($m = +1$) and its application to complex arithmetic.

In the present paper, the author would like to discuss the hyperbolic case ($m = -1$), in order to give a modified alogrithm for the computation of square root, exponential and logarithmic functions.

§ 2. The principle of CORDIC in the hyperbolic case

In order to make the paper self-contained, we shall begin with the principle of CORDIC in the hyperbolic case.

Let x, y be the real numbers satisfying $x > |y|$. The *hyperbolic coordinates* (R, A) of a point $P = (x, y)$ is defined by

$$R = (x^2 - y^2)^{1/2},\quad A = \operatorname{arctanh}(y/x)\,;\ x = R\cosh A,\ y = R\sinh A.$$

It is well-known that $R^2A/2$ is the area of the domain surrounded by the x-axis, radius vector OP and the hyperbola $x^2 - y^2 = c$ passing through P (see Fig. 1).

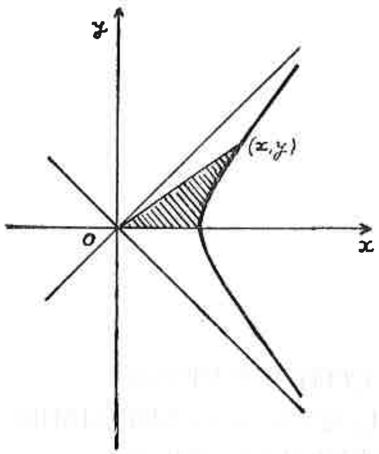

Fig. 1

Now taking a constant $\delta_k (|\delta_k| < 1)$, we shall apply the linear transformation $T(\delta_k) : (x_k, y_k) \to (x_{k+1}, y_{k+1})$ by

$$(1) \qquad T(\delta_k): \begin{aligned} x_{k+1} &= x_k + \delta_k y_k \\ y_{k+1} &= y_k + \delta_k x_k \end{aligned}$$

which yields in the hyperbolic coordinates

$$(2) \qquad \begin{aligned} R_{k+1} &= R_k \times K_k, \ K_k = (1 - \delta_k^2)^{1/2} \\ A_{k+1} &= A_k + \alpha_k, \ \alpha_k = \operatorname{arctanh} \delta_k. \end{aligned}$$

We introduce a third variable z and transform is simultaneously with (1) as

$$(3) \qquad z_{k+1} = z_k - \alpha_k.$$

We take a sequence δ_k, and repeat the transformation $T(\delta_1), T(\delta_2), \ldots, T(\delta_{n-1})$ with (3) starting from the values (x_1, y_1, z_1) until we arrive at (x_n, y_n, z_n). The final values are given by

$$(4) \qquad \begin{aligned} x_n &= K(x_1 \cosh\alpha + y_1 \sinh\alpha) \\ y_n &= K(x_1 \sinh\alpha + y_1 \cosh\alpha) \\ z_n &= z_1 - \alpha \end{aligned}$$

where

$$(5) \qquad K = \prod_{k=1}^{n-1} (1 - \delta_k^2)^{1/2}, \ \alpha = \prod_{k=1}^{n-1} \operatorname{arctanh} \alpha_k.$$

In the practical application, we select one of the following two goals:

I. y or A is forced to be 0.

II. z is forced to be 0.

If the goal was reached at the steps y_n in the case I, we have

$$(6) \qquad x_n = K(x_1^2 - y_1^2)^{1/2}, \ z_n = z_1 + \operatorname{arctanh} \frac{y_1}{x_1} = z_1 + \frac{1}{2} \log \frac{x_1 - y_1}{x_1 - y_1}.$$

Simiarly, if the goal was reached at the step z_n in the case II, we have

$$(7) \qquad \begin{aligned} x_n &= K(x_1 \cosh z_1 + y_1 \sinh z_1), \\ y_n &= K(x_1 \sinh z_1 + y_1 \cosh z_1). \end{aligned}$$

Therefore, we may compute square root, exponential and logarithmic functions by the above process.

If there remains very small residues ε at the final step, it is easy to see that the relative errors in (6) or (7) are of order ε, provided that the rounding errors are negligeable. Remark that for x_n in (6), the truncation error is less than $\varepsilon^2/2x_n^2$, which is of *higher order* than ε.

§ 3. Practice and convergence of CORDIC

In the binary arithmetic, the transformation $T(\delta_k)$ is most easily performed, if we take $\delta_k = + 2^{-k}$, where (1) is computed only by addition, subtraction and shifting without actual multiplication.

For convenience, we put

$$(8) \qquad \varepsilon_k = 2^{-k}, \quad \beta_k = \operatorname{arctanh} \varepsilon_k$$

and select suitably the signature $\delta_k = + \varepsilon_k$. Precisely we take the algorithm

„if $y_k \geqq 0$ then $\delta_k := -\varepsilon_k$ else $\delta_k := \varepsilon_k$;"

in the case I, and

„if $z_k < 0$ then $\delta_k := -\varepsilon_k$ else $\delta_k := \varepsilon_k$;"

in the case II.

Unfortunately, the constants β_k's do *not* satisfy the convergence criterion of WALTHER [1]:

$$(9) \qquad \beta_k - \sum_{j=k+1}^{n-1} \beta_j \leqq \beta_{n-1} \quad (k = 1, 2, \ldots, n-2).$$

In fact, β_k's satisfy an inequality of *opposite* direction in (9). However, as mentioned in [1] and proved in [2], the inequality (9) is *true* if we add the *correction term* $-\beta_l$, $l \leqq 3k+1$ in the left hand side of (9). Therefore it is enough to add some modification in order to guarantee the convergence. Probably the simplest way is to *repeat the process once more* with the same values of ε_k and β_k at the k_i-th step ($i \geqq 1$), where

$$k_0 = 1, \ k_i = 3k_{i-1} + 1 \ (i \geqq 1), \ i.e., \ k = 4, 13, 40, 121, \ldots$$

For the usual accuracy less than 11 decimals, it is enough to repeat at $k = 4$ and 13 only. Precisely speaking, we should write

$$\varepsilon_k = \begin{cases} 2^{-k} & \text{for } 1 \leqq k \leqq 4 \\ 2^{-(k-1)} & \text{for } 5 \leqq k \leqq 14 \\ 2^{-(k-2)} & \text{for } 15 \leqq k \leqq 42 \end{cases}$$

and so on, but we continue use the notation (8).

An alternate modification will be to insert $\varepsilon' = \frac{3}{4}\varepsilon_k$ and the corresponding value $\varepsilon'_k = \operatorname{arctanh} \varepsilon'_k$ at all or necessary steps. We prefer the former one because of the simplicity.

The convergence region in the case II is

$$(10) \qquad |z_1| \leqq B = \sum_{k=1}^{\infty} \beta_k + \beta_4 + \beta_{13} + \ldots = 1.117\ldots$$

and that in the case I is

$$(11) \qquad |y/x_1| \leqq \tanh B = 0.806\ldots,$$

The values of β_k's should be preassigned. Remark that β_k may be replaced by ε_k if ε_k^3 is negligeable, and by $\varepsilon_k + \frac{1}{3}\varepsilon_k^3$ if ε_k^5 is negligeable, so that we need rather few constants in the algorithm.

§ 4. Application to square root

It is easy to see that $x_1 = t + c$, $y_1 = t - c$ gives

$$x_n = K(x_1^2 - y_1^2)^{1/2} = (2K\sqrt{c})\sqrt{t}$$

in (6). Hence taking $\sqrt{c} = 1/2K$, we have $x_n = \sqrt{t}$ in (6). Remark that if we need no logarithmic function simultaneously, the variable z and the constants β_k'*s* are *unnecessary*. Further, by the remark at the end of § 2, we have only to repeat the process with repetetion at $k = 4, 13, \ldots$, *until* $k = n/2$, provided that accuracy of n bits is necessary. The process is as in the flow chart shown in Fig. 2.

The constants are

$$(12) \qquad K = \prod_{k=1}^{\infty}(1 - 2^{-2k})^{1/2} \times \prod_{k=4,13,\ldots}(1 - 2^{-2k})^{1/2}, \quad 1/K = 1.2074970806$$

and

$$c = 1/4K^2 = 0.36451229219\ldots$$

The convergence region is

$$(13) \qquad 0.1068\ldots = e^{-2B} \leqq t/c \leqq e^{2B} = 9.348\ldots$$

which is surely implies the interval $\{1/4 \leqq t \leqq 1\}$ or $\{1/16 \leqq t \leqq 1\}$.

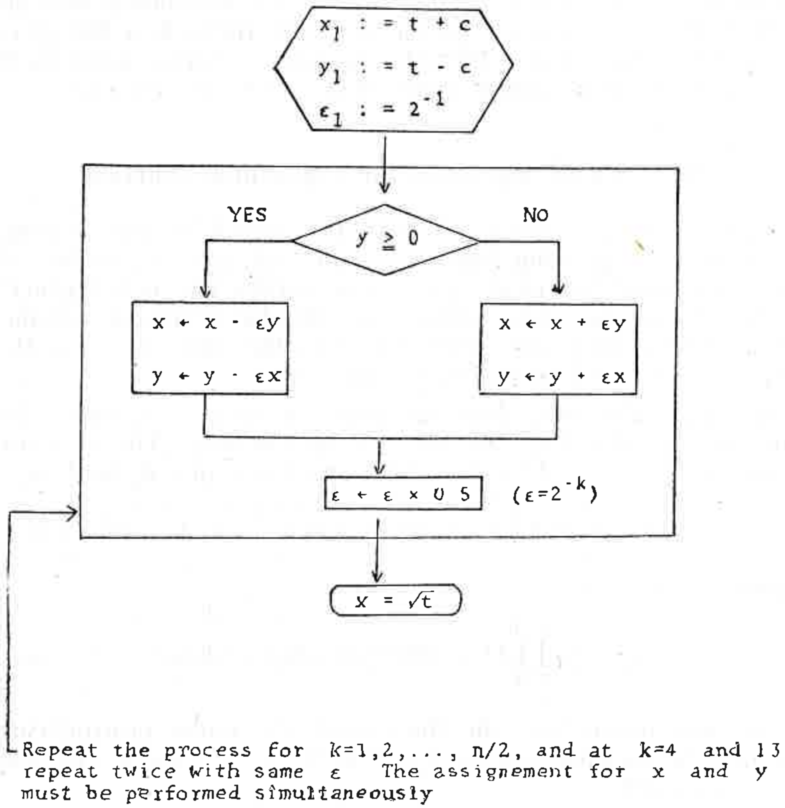

Fig. 2

In the practical program, it will be better to determined the constant c in such a manner that the residue becomes smallest for various values of t in the interval $\{1/4 \leqq t \leqq 1\}$ or $\{1/16 \leqq t \leqq 1\}$. For TOSBAC 3400 in our Institute which has 37 bits in mantissa, the optimal value of c is 0.36451229226.

In the practice, it will be better to normalize t in $\{1/8 \leqq t \leqq 1/2\}$ in order to avoid overflow in the fix point arithmetic. For that case, there is a modified procedure. Start from $k = 2$ ($\varepsilon_2 = 1/4$) with $c = 0.27233292991$ (approximately 3/4 of the previous c), and repeat twice with same ε at $k = 7$ ($= 2 \times 3 + 1$) instead of 4 and 13. The convergence region is $0.33 \doteqdot e^{-B'} \leqq |t/c| \leqq e^{B'} \doteqdot 3$, $B' = B - \beta_1$ which surely covers the interval $\{1/8 \leqq t \leqq 1/2\}$.

According to the results of the numerical experiments, it seems interesting that the repetition up to the final bit (until $k = 38$) gives worse results than to stop at $k = 20$; the maximal relative error in the first case is nearly 3 times bigger than that in the leatter case.

§ 5. A new algorithm for exponential function

Since $e^\alpha = \sinh\alpha + \cosh\alpha$, it will be natural to try to compute e^α by CORDIC case II, starting from $x_1 = y_1 = 1/k$, $z_1 = \alpha$; in fact it surely gives $x_n = e^\alpha$, provided that $z_1 = \alpha$ lies within the convergence region (10). However, in the above discussion, we have always assumed that $x_k > |y_k|$, which does *not* imply the limiting case $x_k = y_k$. Hence we give an alternate proof to guarantee the result.

If $x_1 = y_1$ at the initial step, we have always $x_k = y_k$ under the transformation (1), so that the variable y is unnecessary. The transformation is replaced by $x_{k+1} = x_k(1 + \delta_k)$. Since we have put $\delta_k \tanh \alpha_k$, we see

$$x_{k+1}' = x_k(1 + \tanh\alpha_k) = x_k \operatorname{sech}\alpha_k \exp\alpha_k = x_k(1 - \delta_k^2)^{1/2} \exp\alpha_k,$$

which gives

$$x_n = x_1 \prod_{k=1}^{n-1} (1 - \delta_k^2)^{1/2} \exp(\alpha_k) = K x_1 e^\alpha.$$

Indeed, we can prove that the final result (4) under CORDIC case II is true without the restriction $x_1 < |y_1|$, while in the case I, the restriction (11) is indispensable.

The actual algorithm to compute e^t for real argument t is as follows:

1°. Put $t \cdot \log_2 e = m + s$; m: integer, $-1 < s \leqq 0$

2°. Put $z_1 = s \log_e 2$, $x_1 = 1/K$

where $\log_e 2 = 0.69314718056$ and $1/K$ is given in (12).

3°. Repeat the procedure

„if $z < 0$ then begin $x := x(1 - \varepsilon)$; $z := z + \beta$ end

else begin $x := x(1 + \varepsilon)$; $z := z - \beta$ end;

where $\varepsilon = 2^{-k}$, $\varepsilon = \operatorname{arctanh} 2^{-k}$,

for $k = 1, 2, 3, \ldots$ with repetition at $k = 4$ and 13.

Remark that in the above scaling 1°, s is always negative, so that at the first step, we have always $z \leqq 0$. Hence it will be better to start with

$$z_2 = s \cdot \log 2 + \operatorname{arctanh}(1/2),\ x_2 = 1/2K,\ \varepsilon = 2^{-2},\ k = 2, 3, \ldots$$

In the computation of complex exponential function

$$\exp(x + iy) = e^x(\cos y + i \sin y),$$

we combine with CORDIC in circular case for the trigonometric functions. In that case it will be better to start with $x_1 = 1/KK_+ = 0.36662806930\ldots$, where

$$K_+ = \sum_{k=0}^{\infty} (1 + 2^{-2k})^{1/2}.$$

§ 6. Computation of logarithmic function

Logarithmic function $\log t$ will be computed by the inverse process as in the previous paragraph. However, to avoid actual division, we prefer the following algorithms:

Assume first that the argument t is sufficiently near 1 (precise bound will given later in (15)).

1°. Put

$$x_2 = 1/K_1 = \sqrt{3}/2K = 1.045723138,\ y = t,\ z_2 = 0.$$

Here $K_1 = K/(1 - 2^{-2})^{1/2}$.

2°. Repeat the following process:

„if $x > y$ then begin $x := (1 - \varepsilon)x$; $z := z + \beta$ end

else begin $x := (1 + \varepsilon)x$; $z := z - \beta$ end;"

where $\varepsilon = 2^{-k}$, $\beta = \operatorname{arctanh}\varepsilon$,

for $k = 2, 3, \ldots$, and at $k = 4$ and 13 repeating twice with same ε.

The repetition is completely similar to the one in § 5 except only the branching condition is replaced by $x > y$ instead of $z < 0$. The process brings x as close as y. Finally if $x = y$, then we have

$$x = (K_1/K_1)\, e^\alpha = t,$$

which gives $z = \alpha = \log t$.

Here we start from $k=2$, since $(1-2^{-1})=1/2$ is too small to recover later by other products. Hence the convergence region is restricted in

$$|z_1| \leqq B-\beta_1 = B' \doteqdot 0.568\ldots,$$

(15) $$0.57\ldots = e^{-B'} \leqq |t| \leqq e^{B'} = 1.76\ldots$$

For other values of t, we make scaling $t=2^m \cdot y$ (m being integer), and set $z_2 = m \cdot \log_e 2$ in (14) instead of 0. Here we cannot take $1/2 \leqq \leqq y \leqq 1$ or $1 \leqq y \leqq 2$, but we must choose the mantissa y in

(16) $$3/4 \leqq y \leqq 3/2 \text{ or } 1/\sqrt{2} \leqq y \leqq \sqrt{2}.$$

But this restriction is not serious in the actual programming.

Finally remark that this algorithm is not suitable for the computation of $\log t$ when the argument t is *quite close to* 1. In such a case we should replace the function by $\log(1+s)$ computed by the Taylor series

$$\log(1+s) = s - \frac{s^2}{2} + \frac{s^3}{3} - \ldots,$$

or by other approximation formulas.

Added in Proof:

After I have finished to prepare the present paper I found that the papers [3], [4] *and* [5] *are closely connected with the method proposed here. The author would like to discuss the relations and their evaluations in a separate paper.*

REFERENCES

[1] Walther, J. S., *A unified algorithm for elementary functions*, Spring Joint Computer Conference 379—385, 1971.

[2] Hitotumatu S., *Complex arithmetic through CORDIC*, R.I.M.S. Preprint **138** (1973); will appear in Kodai Math. Sem. Report.

[3] Meggitt, J. E., *Pseudo-division and pseudo-multiplication processes*, IBM J. Res. Dev. **6**, 210—226, (1962).

[4] Specker, W. H., *A class of algorithmus for* ln x, exp x, sin x, cos x, tg^{-1} x *and* ctg^{-1} x, IEEE Trans. E.C. **14**, 85—86, (1965).

[5] Koyanagi, S., Watanabe, K. and Hagiwara, H., *Approximation of elementary functions by micro-programming* (in Japanes), Proc. 14th Annual Meeting of the Information Processing Society of Japan, 171—172, (1973).

Received 1. III. 1974.

Research Institute for
Mathematical Sciences
Kyoto University, Kyoto, Japan