S. HITOTUMATU
KŌDAI MATH. SEM. REP.
26 (1975), 176—186

COMPLEX ARITHMETIC THROUGH CORDIC

(Dedicated to Prof. Y. Komatu for his 60th birthday)

BY SIN HITOTUMATU

Abstract

A unified algorithm for elementary functions due to coordinate transformations, named CORDIC has been first introduced by Volder [1] and later extensively investigated by Walther [2]. Here the author mentions several practical remarks for the application of the algorithm to complex arithmetic including square root.

§ 1. The principle of CORDIC.

In order that the present paper may be self-contained, we first briefly summarize the principle of the algorithm.

1.1. Generalized polar coordinates.

Let (x, y) be the planar orthogonal coordinates for a point P and introduce a *generalized polar coordinate system* (R, A) by

$$(1)\qquad \begin{cases} R=(x^2+my^2)^{1/2} \\ A=m^{-1/2}\arctan(m^{1/2}y/x), \end{cases} \qquad \begin{cases} x=R\cos(m^{1/2}A) \\ y=Rm^{-1/2}\sin(m^{1/2}A). \end{cases}$$

Here m is a fixed constant whose value is one of 1, -1 or 0. We should impose some interpretations when $m=0$ and $m=-1$; precisely, we put

$$A=y/x \quad \text{for} \quad m=0; \qquad A=\operatorname{arctanh}(y/x) \quad \text{for} \quad m-=1.$$

For simplicity, we always assume $x\geqq 0$, and further $x\geqq|y|\geqq 0$ for $m=-1$. It is easily seen that $A=S/2R^2$, where S is the area of the domain surrounded by x axis, the radius vector OP and the curve of constant radius R passing through P.

Received June 4, 1973.

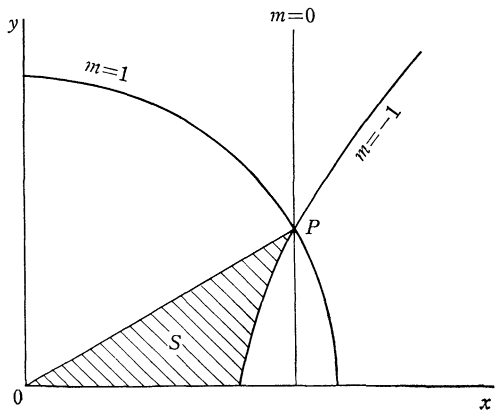

Fig. 1.

1.2. Fundamental transformations.

Take a linear transformation from a point $P_j=(x_j, y_j)$ to $P_{j+1}=(x_{j+1}, y_{j+1})$ given by

$$
\left\{\begin{array}{l} x_{j+1}=x_j+m\delta_j y_j \\ y_{j+1}=y_j-\delta_j x_j \end{array}\right. \tag{2}
$$

where m is the parameter of the coordinate system (1) and δ_j is an arbitrary constant. The transformation (2) gives

$$
A_{j+1}=A_j-\alpha_j
$$

$$
R_{j+1}=R_j \times K_j
$$

in the generalized polar coordinate system (1), where

$$
\alpha_j=m^{-1/2}\arctan(m^{1/2}\delta_j)
$$

$$
K_j=(1+m\delta_j^{\,2})^{1/2}.
$$

Starting from $P_0=(x_0, y_0)$, we iterate the transformations (2) under suitable sequence of constants $\delta_0, \delta_1, \cdots$, until we arrive at $P_n=(x_n, y_n)$. Then we have

$$
A_n=A_0-\alpha
$$

$$
R_n=R_0 \times K
$$

where

$$
\alpha=\sum_{j=0}^{n-1}\alpha_j, \qquad K=\prod_{j=0}^{n-1}K_j. \tag{3}
$$

Now we introduce third variable z and transform it as

(4) $$z_{j+1}=z_j+\alpha_j$$

simultaneously with (2). Then the final values are given by the followings

$$\begin{cases} x_n=K[x_0 \cos(m^{1/2}\alpha)+y_0 m^{-1/2} \sin(m^{1/2}\alpha)] \\ y_n=K[y_0 \cos(m^{1/2}\alpha)-x_0 m^{-1/2} \sin(m^{1/2}\alpha)] \\ z_n=z_0+\alpha \end{cases}$$

where α and K are given by (3).

1.3. The final values.

Let us iterate the above transformations (2) and (4) under suitable sequence of constants $\{\delta_j\}$ in the following two ways:

Case I: A or y is forced to zero.
Case II: z is forced to zero.

If this has been done after n steps, the final results will have the values as in the following table 1, where

$$K_{\pm 1}=\prod_{j=0}^{\infty}(1\pm\delta_j^2)^{1/2}.$$

Noting that

(5) $$\sqrt{(t+c)^2-(t-c)^2}=2\sqrt{c}\sqrt{t},$$

$$\exp t=\cosh t+\sinh t, \qquad \frac{1}{2}\log\frac{t}{c}=\operatorname{arctanh}\frac{t+c}{t-c},$$

the table contains all elementary standard functions such as square root, log, exp, sin, cos and arctan as well as multiplication and division. It is not difficult to see that the iteration of case I for $m=0$ is essentially the non-restorting division algorithm.

Table 1

Case	m	x	y	z
I	1	$K_1\sqrt{x_0^2+y_0^2}$	0	$z_0+\arctan(y_0/x_0)$
I	0	x_0	0	z_0+y_0/x_0
I	-1	$K_{-1}\sqrt{x_0^2-y_0^2}$	0	$z_0+\operatorname{arctanh}(y_0/x_0)$
II	1	$K_1(x_0\cos z_0-y_0\sin z_0)$	$K_1(y_0\cos z_0+x_0\sin z_0)$	0
II	0	x_0	$y_0+x_0z_0$	0
II	-1	$K_{-1}(x_0\cosh z_0+y_0\sinh z_0)$	$K_{-1}(y_0\cosh z_0+x_0\sinh z_0)$	0

1.4. Actual procedure of CORDIC.

Since most of the recent computers use binary system, it is most convenient to choose $\delta_j=\pm 2^{-j}$ (or $\pm 2^{-j-1}$), $j=0, 1, 2, \cdots$ up to necessary number of bits N. When $m=-1$, slight modification is necessary for the convergence (see § 3.1.). We write

$$\varepsilon_j=2^{-j}, \qquad \beta_j=m^{-1/2}\arctan(m^{1/2}2^{-j})$$

$$\delta_j=\pm\varepsilon_j, \qquad \alpha_j=\pm\beta_j \qquad \text{(with same signatures)}.$$

The constants β_j may be precalculated. For convenience, we give the values of arctan 2^{-j} in decimal and in octal form in the appendix of the present paper[1]. Now, we modify the transformations (2) and (4) in the following form:

for $\delta_j>0$

$$(6_+) \qquad \begin{cases} x_{j+1}=x_j+m2^{-j}y_j \\ y_{j+1}=y_j-2^{-j}x_j \\ z_{j+1}=z_j+\beta_j \end{cases}$$

for $\delta_j<0$

$$(6_-) \qquad \begin{cases} x_{j+1}=x_j-m2^{-j}y_j \\ y_{j+1}=y_j+2^{-j}x_j \\ z_{j+1}=z_j-\beta_j \end{cases}$$

In Case I, we select (6_+) if $y_j \geqq 0$, and (6_-) otherwise; in Case II, we select (6_+) if $z_j<0$, and (6_-) otherwise.

It is remarkable that the transformation (6_+) or (6_-) is possible only by addition, subtraction, shifting and read out of constants, but needs no *multiplication*. Thus, the algorithm is quite suitable for micro-computers without hardwares for multiplication and division, or for multiple precision arithmetic.

We also remark that the constants β_j are necessary only for j up to $N/3$, where N is the necessary number of bits, since

$$\beta_j=\varepsilon_j+\frac{m}{3}\varepsilon_j^3+\frac{1}{5}\varepsilon_j^5+\cdots$$

may be replaced by ε_j, if ε_j^3 and higher terms are negligible.

§ 2. CORDIC for $m=1$ and its application to complex arithmetic.

2.1. General remarks.

Using CORDIC for $m=1$, we can easily change orthogonal coordinates into polar coordinates and vice versa.

As Walther [2] has indicated, the convergence condition of CORDIC is given by

$$(7) \qquad \beta_j-\sum_{k=j+1}^{n-1}\beta_k \leqq \beta_{n-1} \qquad (j=0, 1, 2, \cdots).$$

1) The author is much grateful for Mr. S. Yamashita (Fujitsu Co.) who has kindly computed the necessary constants.

When $m=1$, it is easy to verify (7) for $\beta_j=\arctan 2^{-j}$, since $\arctan x$ is *convex* in $x\geqq 0$. Thus the procedure always converges, when the initial argument z_0 is in absolute value less than

$$\sum_{j=0}^{\infty}\arctan 2^{-j}=1.74\cdots>\pi/2,$$

which surely covers the closed *right half plane*. According this fact, it will be more convenient to normalize the argument of a complex number $p+iq$ in the inverval $[-\pi/2, +\pi/2]$, and to permit *negative modulus*; i. e., when $p<0$, we denote in the polar form

$$p+iq=re^{i\theta}, \qquad r=-\sqrt{p^2+q^2}<0, \qquad -\pi/2<\theta<\pi/2.$$

2.2. Multiplication.

The product of two complex numbers $a+ib$ and $p+iq$ is obtained by the following algorithm.

Algorithm 1. 1. *Transformation into polar coordinates.*

(i) Take initial values as

$$x_0=|p|, \quad y_0=\pm\, q \ (+ \text{ if } p\geqq 0, \text{ and } - \text{ if } p<0), \quad z_0=0.$$

(ii) Operate CORDIC of Case I for $m=+1$. Then we have

$$x=K_1\sqrt{p^2+q^2}, \quad y=0, \quad z=\arctan(q/p) \quad \text{(principal value)}.$$

(iii) Replace x by $-x$ if $p<0$.

(iv) In order to modify the absolute value, we multiply to x a constant

$$K_1^{-2}=0.3687561270769\ldots$$

(v) We shall refer by r and θ the final values of x and z respectively.

2. *Rotation.*

(vi) Put

$$x_0=a\times r, \quad y_0=b\times r, \quad z_0=\theta.$$

(vii) Operate CORDIC of Case II for $m=+1$. Then the final values give

$$\left\{\begin{array}{l} x=K_1K_1^{-2}K_1\sqrt{p^2+q^2}\,(a\cos\theta-b\sin\theta)=ap-bq \\ y=K_1K_1^{-2}K_1\sqrt{p^2+q^2}\,(a\sin\theta+b\cos\theta)=aq+bp \end{array}\right. \tag{8}$$

which are the real and the imaginary parts of the product.

Usual method due to the right-most hands of (8) needs four real multiplications to obtain $(a+ib)\times(p+iq)$. By the Algorithm 1, we need 1 multiplication and 1 CORDIC operation in the first step, and 2 multiplications and 1 CORDIC operation in the second step. Therefore the algorithm is less efficient than usual method for single multiplication, even when the real multiplication is computed by CORDIC for $m=0$. However, this may be useful at least in the following cases. First is the case when the multiplier $p+iq$ is previously given by its polar form, e. g. as in the complex Fourier transform. Second is the case when we repeat multiplications with same multiplier as in the Horner's scheme for the computation of a polynomial.

2.3. Division.

To obtain the quotient $(a+ib)/(p+iq)$, it is enough to modify Algorithm 1 as follows. Omit step (iv) and replace step (v) by

$$\text{(v}'\text{)} \qquad r=1/x, \quad \theta=-z.$$

For a single division, however, it will be better to omit step (iv) and to replace step (vi) by

$$\text{(vi}'\text{)} \qquad x_0=a/r, \quad y_0=b/r, \quad z_0=-\theta.$$

The final values are

$$\text{(9)} \qquad \begin{cases} x=K_1(a\cos\theta+b\sin\theta)/K_1\sqrt{p^2+q^2}=(ap+bq)/(p^2+q^2) \\ y=K_1(b\cos\theta-a\sin\theta)/K_1\sqrt{p^2+q^2}=(bp-aq)/(p^2+q^2). \end{cases}$$

Remark that no modification in modulus is necessary, since the constant K_1 cancels out as seen in (9).

Usual method of division due to the right-most hand of (9) requires 6 multiplications and 2 divisions, while the above algorithm needs only 2 CORDIC operations, 1 division and 2 multiplications (or 2 divisions). Thus, this is much more efficient than usual method, if multiplications and divisions are carried out by CORDIC for $m=0$.

In practice, it will be more convenient to make some scalings of the initial values in order to guarantee the accuracy, especially for very large or very small data. However, we emphasize that the above algorithms always *converge* and need no adjustment of arguments. Even when $r=0$ in division, the overflow error will be detected at the stage of real divisions $1/r$ or a/r, b/r, for which we need not pay attention here.

§ 3. CORDIC for $m=-1$ and its application to square root.

3.1. Convergence for $m=-1$.

When $m=-1$, the sequence $\beta_j=\operatorname{arctanh} 2^{-j}$ $(j=1, 2, \cdots)$ does *not* satisfy the convergence condition (7), since arctanh x is *concave* in $x\geqq 0$. However, we have following relation:

$$(10)\qquad \beta_j-\Big(\sum_{k=j+1}^{n-1}\beta_k\Big)-\beta_{3j+1}<\beta_{n-1}\qquad \text{for}\quad \beta_j=\operatorname{arctanh} 2^{-j}\ (j\geqq 1)\,.$$

As there is no proof in Walther [2], we shall give here a brief proof to (10). Since

$$\operatorname{arctanh} x=x+\frac{x^3}{3}+\frac{x^5}{5}+\cdots,\qquad |x|<1$$

and since the linear term satisfies (7) with equality, we have

$$\begin{aligned}\beta_j-\sum_{k=j+1}^{n-1}\beta_k-\beta_{n-1}&=\frac{1}{3}(2^{-3j}-2^{-3(j+1)}-\cdots-2^{-3(n-1)}-2^{-3(n-1)})\\&\quad+\frac{1}{5}(2^{-5j}-2^{-5(j+1)}-\cdots-2^{-5(n-1)}-2^{-5(n-1)})+\cdots\\&<\frac{1}{3}2^{-3j}+\frac{1}{5}2^{-5j}+\cdots<\frac{1}{3}\cdot\frac{2^{-3j}}{1-2^{-2j}}\quad(j\geqq 1)\\&\leqq\frac{4}{9}2^{-3j}<2^{-(3j+1)}<\beta_{3j+1}\,,\end{aligned}$$

which proves (10).

As β_j decreases with j, the convergence condition (7) will be satisfied if we repeat the iteration *once more* with same j at

$$j=4, 13, 40, 121, \cdots.$$

The general formula of the above sequence is given by the recurrence formula

$$j_n=3j_{n-1}+1\,,\qquad j_0=1\,.$$

Usually we need accuracy within 10 decimals, so that the repetition is necessary only at $j=4$ and 13. Repetition at $j=40$ is necessary only for multiple precision up to 120 bits (about 36 decimals), and hence the repetition is not serious in the actual programming of CORDIC.

3.2. Convergence region.

Here we are mainly concern with Case I to obtain square root by (5). Contrary to the case of $m=1$, we must remark carefully the *convergence region* for $m=-1$. The process converges if and only if the initial values (x_0, y_0) satisfy the condition

$$(11)\qquad |\operatorname{arctanh}(y_0/x_0)|<C=\Big(\sum_{j=1}^{\infty}+\sum_{j=4,13,\cdots}\Big)\operatorname{arctanh} 2^{-j}=1.118\cdots.$$

Even when (11) is not satisfied, the process itself stops after finite number of steps, but *non-convergence* means that the value of y does not approach 0, and hence the final values x and z have no meanings.

If we start from

$$x_0=t+c\,,\qquad y_0=t-c\,,$$

(11) gives the restriction

$$Q^{-1}<t/c<Q\,, \qquad Q=e^{2C}=9.35\cdots\,, \qquad Q^{-1}=0.1068\cdots\,.$$

When we compute the square root of a real number, it will be convenient to take the constant c such that

$$K_{-1}2\sqrt{c}=1\,,$$

i. e.

$$c=c_0=1/4K_{-1}^2=0.364512292144\cdots$$

where

$$K_{-1}^2=\Big(\prod_{j=1}^{\infty}\times\prod_{j=4,13,\cdots}\Big)(1-2^{-2j})=0.685847927146\cdots\,,$$

rather than to take $c=1/4$ as is indicated in [2]. Then the process converges for

$$0.039\cdots=Q^{-1}c_0<t<c_0Q=3.41\cdots \tag{12}$$

which surely covers the interval

$$1/4\leqq t\leqq 1 \qquad \text{or} \qquad 1/16\leqq t\leqq 1\,.$$

Outside the interval (12), we need scaling.

If we operate CORDIC of Case I for $m=-1$, we have simultaneously

$$z=\operatorname{arctanh}(y_0/x_0)=\frac{1}{2}\log t-\frac{1}{2}\log c$$

as a byproduct. If this value is unnecessary, it will be better to omit the transformations of z from (6_+) and (6_-), by which the values of

$$\beta_j=\operatorname{arctanh}2^{-j}$$

are no longer necessary for the operations.

3.3. Complex squre root.

The square root of a complex number $p+iq=re^{i\theta}$ is given by

$$\pm\sqrt{r}\,e^{i\theta/2}\,,$$

where we take $r\geqq 0$ as usual.

To compute the square roots of $p+iq$ through CORDIC, we apply the following algorithm.

Algorithm 2. (i) and (ii) are same as in Algorithm 1.

(iii) Put

$$\theta=\begin{cases} z/2 & \text{if } p\geqq 0 \\ z/2-\pi/2 & \text{if } p<0\,,\ z>0 \\ z/2+\pi/2 & \text{if } p<0\,,\ z\leqq 0\,. \end{cases}$$

(iv) In order to guarantee the convergence, we make the following scaling. If $x=0$, then jump directly to (viii). Otherwise, we put

$$x=t\times 4^{l-1}, \qquad 1/16\leqq t\leqq 1, \qquad l \text{ being an integer}^{2)}.$$

(v) Put

$$x_0=t+c, \qquad y_0=t-c$$

where

$$c=c_1=0.32649838486\cdots.$$

The meaning of the constant c_1 will be discussed later.

(vi) Operate CORDIC of Case I for $m=-1$.

(vii) Normalize x by multiplying 2^l. This operation is done by shifting or adjustment of the exponent part.

(viii) Put

$$x_0=x \text{ (as given above)}, \qquad y_0=0, \qquad z_0=\theta.$$

(ix) Operate CORDIC of Case II for $m=+1$.

The final value $x+\imath y$ gives one of the square roots of $p+iq$. Another one is $-x-\imath y$.

Let us explain the meaning of the constant c_1. Since the modulus of the final value through Algorithm 2 is

$$K_1K_{-1}2\sqrt{cK_1r}, \qquad r=\sqrt{p^2+q^2},$$

it seems to be suitable to choose c such that

$$2\sqrt{c}K_{-1}K_1^{3/2}=1. \tag{13}$$

However, the value of c determined by (13) is

$$c=1/4K_{-1}^2K_1^3=0.081624596215\cdots, \tag{14}$$

which is too small to include the interval [1/4, 1] in its convergence region. Hence, we take $c=c_1$ to be 4 times of the constant in (14), which is the above value of c_1. Thus the convergence region is given by

$$0.0212\cdots=c_1Q^{-1}/K_1<r<c_1Q/K_1=1.85\cdots, \qquad r=|p+iq|,$$

which surely contains $1/16\leqq r\leqq 1$. This also explains the above scalings.

Usual method for square roots of a complex number $p+iq$ is due to the formula

$$\pm\sqrt{(p+|p+\imath q|)/2}\pm i\sqrt{(-p+|p+\imath q|)/2},$$

where the signatures are same if $q\geqq 0$ and opposite if $q<0$. This requires 2 multiplications (in order to compute absolute value) and 3 square roots (or 2 square roots and 1 division), while Algorithm 2 needs only 3 CORDIC operations with a slight process of scaling. Hence Algorithm 2 will be much more efficient than usual method, especially when multiplications and divisions are carried out by CORDIC for $m=0$ and square root is computed by usual Newton's iteration.

Several examples show quite satisfactory results both in accuracy and in efficiency. However, we must remark that Algorithm 2 usually gives small real

2) Though this condition gives two values of t and l, each of them gives the same final results. In practice, we may select l such that $l-1$ is even, or that $|l|$ is smaller.

part in the result if $p+iq$ is real negative (i.e., $q=0$, $p<0$), because of the error in the numerical value of $\cos(\pi/2)$ by CORDIC.

Appendix

Table 2

Decimal representations of $\arctan 2^{-j}$

1	0.46364	76090	00806	11621	42562	31461	21440	20285	37054	28612
2	0.24497	86631	26864	15417	20824	81211	27581	09141	44098	38118
3	0.12435	49945	46761	43503	13548	49163	87102	55731	70191	76980
4	0.06241	88099	95957	34847	39791	12985	50511	36062	73887	79749
5	0.03123	98334	30268	27625	37117	44892	49097	70324	95663	72540
6	0.01562	37286	20476	83080	28015	21256	57031	89111	14139	80090
7	0.00781	23410	60101	11129	64633	91842	19928	16212	22811	72501
8	0.00390	62301	31966	97182	76286	65311	42438	71403	57490	11520
9	0.00195	31225	16478	81868	51214	82625	07671	39316	10746	77723
10	0.00097	65621	89559	31943	04034	30199	71729	08516	34197	01581
11	0.00048	82812	11194	89827	54692	39625	64484	86661	92361	13313
12	0.00024	41406	20149	36176	40167	22943	25965	99862	12417	79097
13	0.00012	20703	11893	67020	42390	58646	11795	63009	30829	40901
14	0.00006	10351	56174	20877	50216	62569	17382	91537	85143	53683
15	0.00003	05175	78115	52609	68618	25953	43853	60197	50949	67511
16	0.00001	52587	89061	31576	21072	31935	81269	78851	37429	23814
17	0.00000	76293	94531	10197	02633	88482	34010	50905	86350	74391
18	0.00000	38146	97265	60649	62829	23075	61637	29937	22805	25730
19	0.00000	19073	48632	81018	70353	65369	30591	72441	68714	34216
20	0.00000	09536	74316	40596	08794	20670	68992	31123	90019	63412

Octal representations of $\arctan 2^{-j}$

1	0.35530	63405	30335	51732	12677	44213	66274	16506	40333	41202
2	0.17533	35374	45440	15654	25333	43636	75373	64205	76265	23772
3	0.07752	67246	52605	73334	31310	45714	23717	50421	67570	54712
4	0.03775	25566	71317	67516	15545	44744	55601	61340	65211	43160
5	0.01777	52533	56513	54222	50235	71656	73335	52143	04116	00332
6	0.00777	75252	67355	62465	33574	74305	31000	53735	40220	20357
7	0.00377	77525	25567	35227	13200	30432	43756	76301	27507	32053
8	0.00177	77752	52533	56733	45135	42253	73326	66517	37413	17406
9	0.00077	77775	25252	67356	72456	24653	36526	01045	07164	36220
10	0.00037	77777	52525	25567	35667	12271	32003	17674	72457	46466
11	0.00017	77777	75252	52533	56735	65134	51354	22542	22501	17160
12	0.00007	77777	77525	25252	67356	73556	24562	46533	65335	74736
13	0.00003	77777	77752	52525	25567	35673	52271	22713	20032	00304
14	0.00001	77777	77775	25252	52533	56735	67334	51345	13542	25422
15	0.00000	77777	77777	52525	25252	67356	73567	24562	45624	65336
16	0.00000	37777	77777	75252	52525	25567	35673	56671	22712	27132
17	0.00000	17777	77777	77525	25252	52533	56735	67356	51345	13451
18	0.00000	07777	77777	77752	52525	25252	67356	73567	35562	45624
19	0.00000	03777	77777	77775	25252	52525	25567	35673	56735	22712
20	0.00000	01777	77777	77777	52525	25252	52533	56735	67356	73345

Added in Proof: In Algorithm 2, modulus converges quadratically, so that we may stop after $N/2$ steps to evaluate real square root (see e. g. [3], § 3-8). Such remarks will be published separately.

REFERENCES

[1] J. E. VOLDER, Binary computation algorithms for coordinate rotation and function generation, Convair Report IAR-1 148 Aeroelectronics Group, 1956.
[2] J. S. WALTHER, A unified algorithm for elementary functions, Spring Joint Comp. Conference 1971, p. 379-385.
[3] S. HITOTUMATU, "Numerical Computation of elementary functions", in Japanese, Kyoiku Shuppan, Tokyo, 1974.

RESEARCH INSTITUTE FOR
MATHEMATICAL SCIENCES
KYOTO UNIVERSITY.