termines which MXLO form is to be used at any time, and the programmer has no control over this.

14) DSO: If $(A_2)<0$, $1 \rightarrow DO_i$.
If $(A_2) \geq 0$, $0 \rightarrow DO_i$.
A_1 specifies the proper discrete output, and A_2 refers only to the arithmetic registers.

15) SK: Start nothing new and ignore the address part of the word. SK will be used whenever it is desirable to send nothing new to the computer, such as after instructions which require more than one word cycle for execution; or to fill the drum when the program does not require the full capacity of the drum.

There are two switch-controlled operations which will be considered here because of their close relation to the instructions. They are Discrete Input and Freeze.

DSI: The Discrete Input operation provides that the (A_2) is to be used if the associated DI switch is OFF. If the switch is ON, "zero" is used instead of (A_2).

Freeze: The Freeze operation which has been described earlier is provided to stop the computations in mid-flight for a short time (so that the instructor can talk with the student pilot or make certain checks) and then to resume computations from the same point. During this "cease computation" period, however, the computer must continue to multiplex out the analog outputs so that the instruments will continue to indicate to indicate the proper quantities.

Programming Considerations

A study was made to determine how Multiple-UDOFT with three-address code will compare with the one-address code UDOFT. For this study, it was assumed that the Multiple-UDOFT will have a minor cycle equal to that of UDOFT, and that the new instructions will require the same number of minor cycles for execution as do the instructions for UDOFT.

It was found that there will be no saving of time with the three-address code for simple sums of variables because only one variable can be obtained from the magnetic-core memory during a given reference cycle. The three-address computer will be at least as fast as UDOFT for polynomial evaluations. For sums of squared variables and for O_{33} quadrature formula evaluations, it provides a saving on the order of 30 per cent of the time required by one-address UDOFT; and for updating a set of three time-different values of the same variable, the saving is on the order of 25–40 per cent. These comparisons were made on the basis of a UDOFT without subroutines, since the Multiple-UDOFT cannot have a subroutine structure (because of the drum).

Under true UDOFT conditions, subroutines would be used. This represents a considerable saving in programming and instruction storage space at the expense of computation time. Thus, from the more realistic comparison of Multiple-UDOFT without subroutines vs UDOFT with subroutines, if was found that Multiple-UDOFT saves on the order of 15 per cent of the time required by UDOFT for a typical "flight" path computation.

The CORDIC Trigonometric Computing Technique*

JACK E. VOLDER†

Summary—**The *CO*ordinate *R*otation *DI*gital Computer (CORDIC) is a special-purpose digital computer for real-time airborne computation. In this computer, a unique computing technique is employed which is especially suitable for solving the trigonometric relationships involved in plane coordinate rotation and conversion from rectangular to polar coordinates. CORDIC is an entire-transfer computer; it contains a special serial arithmetic unit consisting of three shift registers, three adder-subtractors, and special interconnections. By use of a prescribed sequence of conditional additions or subtractions, the CORDIC arithmetic unit can be controlled to solve either set of the following equations:**

$$Y' = K(Y \cos \lambda + X \sin \lambda)$$
$$X' = K(X \cos \lambda - Y \sin \lambda),$$

or

$$R = K\sqrt{X^2 + Y^2}$$
$$\theta = \tan^{-1} Y/X,$$

where K is an invariable constant.

This special arithmetic unit is also suitable for other computations such as multiplication, division, and the conversion between binary and mixed radix number systems. However, only the trigonometric algorithms used in this computer and the instrumentation of these algorithms are discussed in this paper.

* Manuscript received by the PGEC, May 25, 1959. Presented at the Western Joint Computer Conf., San Francisco, Calif.; March 3–5, 1959.
† Convair, a Div. of General Dynamics Corp., Fort Worth, Texas.

Introduction

THE CORDIC computing technique was developed especially for use in a real-time digital computer where the majority of the computation involved the discontinuous, programmed solution of the trigonometric relationships of navigation equations and a high solution rate for the trigonometric relationships of coordinate transformations. A prototype computer, CORDIC I, based on this computing technique, has been designed and constructed at Convair, Fort Worth. Although CORDIC I may be classified as an entire-transfer computer, its design is not based on the conventional "pencil and paper" computing technique of general-purpose computers.

Functional Description

For the sake of simplicity, the trigonometric operations in the CORDIC computer can be functionally described as the digital equivalent of an analog resolver. Similar to the operation of such a resolver, there are two computing modes, ROTATION and VECTORING. In the ROTATION mode, the coordinate components of a vector and an angle of rotation are given and the coordinate components of the original vector, after rotation through the given angle, are computed. In the second mode, VECTORING, the coordinate components of a vector are given and the magnitude and angular argument of the original vector are computed. Similarly, as in the case of resolvers, the computing device of ROTATION plus feedback is employed in the VECTORING mode. The original coordinates are rotated until the angular argument is zero, so that the total amount of rotation required is the negative of the original argument, in which case the value of the X-component is equal to the magnitude of the original vector.

In essence, the basic computing technique used in both the ROTATION and VECTORING modes in CORDIC is a step-by-step sequence of pseudo rotations which result in an over-all rotation through a given angle (ROTATION) or result in a final angular argument of zero (VECTORING).

It is necessary that the angular increments of rotation be computed in a decreasing order. There are several permissible values which may be chosen for the angular magnitude of the first rotation step. The magnitude actually chosen for the first increment is 90°. The expression for a set of coordinate components, Y_1 and X_1, rotated through plus or minus 90° is simply

$$Y_2 = \pm X_1 = R_1 \sin(\theta_1 \pm 90°) \qquad (1)$$

$$X_2 = \mp Y_1 = R_1 \cos(\theta_1 \pm 90°). \qquad (2)$$

The first step is unique in that a perfect rotation step is performed.

The rest of the computing steps can be clarified by examining the relationships, involved in a typical rotation step, which are shown in Fig. 1. Consider two given coordinate components, Y_i and X_i, in the plane coordinate system shown in the figure. In this discussion, the quantity i is equal to the number of the particular step under consideration. The components, Y_i and X_i, are associated with the ith step and describe a vector of magnitude R_i at an angle θ_i from the origin according to the relationship:

$$Y_i = R_i \sin \theta_i \qquad (3)$$

$$X_i = R_i \cos \theta_i. \qquad (4)$$

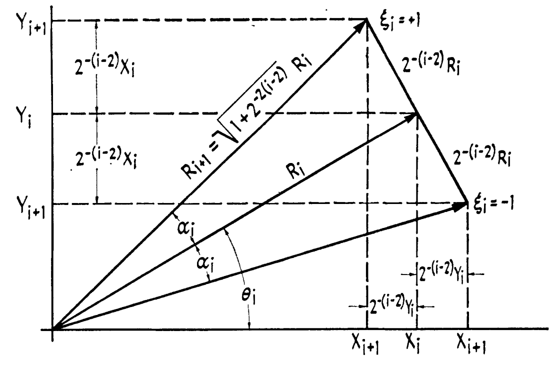

Fig. 1—Typical computing step.

In Fig. 1, the angle α_i is the special magnitude of rotation associated with each computing step. The general expression for α_i where $i > 1$, is

$$\alpha_i = \tan^{-1} 2^{-(i-2)}. \qquad (5)$$

The peculiar magnitude of each α_i is such that a "rotation" of coordinate components through $\pm\alpha_i$ may be accomplished by the simple process of shifting and adding.

The two choices of positive or negative rotation are shown in Fig. 1. The general expression for the rotated components is

$$\begin{aligned} Y_{i+1} &= \sqrt{1 + 2^{-2(i-2)}}R_i \sin(\theta_i \pm \alpha_i) \\ &= Y_i \pm 2^{-(i-2)}X_i \end{aligned} \qquad (6)$$

$$\begin{aligned} X_{i+1} &= \sqrt{1 + 2^{-2(i-2)}}R_i \cos(\theta_i \pm \alpha_i) \\ &= X_i \mp 2^{-(i-2)}Y_i. \end{aligned} \qquad (7)$$

Note that, by restricting the angular rotation magnitude to (5), the right-hand terms of (6) and (7) may be obtained by two simultaneous shift-and-add operations. This is the *fundamental* relationship upon which this computing technique is based.

The computing action of adding (or subtracting) a shifted value of X_i to Y_i to obtain Y_{i+1}, while simultaneously subtracting (or adding) a shifted value of Y_i to X_i to obtain X_{i+1}, is termed "cross-addition."

While the expressions for Y_{i+1} and X_{i+1} are not perfect rotations because of the increase in magnitude by the terms under the radical, either of the two choices of direction produces the same change in magnitude. If, therefore, for each step, the coordinates are always rotated through either a positive or negative α_i, then the increase in magnitude may be considered as a *constant.* This requirement precludes the choice of zero rotation at any step. To identify the choice made in a particular step, the $\pm$ notation may be represented by the binary operator $+\xi_i$ where ξ_i can equal either $+1$ or -1. This substitution produces the general expressions

$$\begin{aligned} Y_{i+1} &= \sqrt{1 + 2^{-2(i-2)}}R_i \sin(\theta_i + \xi_i\alpha_i) \\ &= Y_i + \xi_i 2^{-(i-2)}X_i \end{aligned} \qquad (8)$$

$$\begin{aligned} X_{i+1} &= \sqrt{1 + 2^{-2(i-2)}}R_i \cos(\theta_i + \xi_i\alpha_i) \\ &= X_i - \xi_i 2^{-(i-2)}Y_i \end{aligned} \qquad (9)$$

where

$$\xi_i = +1 \text{ or } -1. \tag{10}$$

Likewise, after the completion of the rotation step in which the $i+1$ terms are obtained, the $i+2$ terms may be computed from these terms with the results

$$Y_{i+2} = \sqrt{1+2^{-2(i-1)}}\sqrt{1+2^{-2(i-2)}}R_i \cdot \sin(\theta_i + \xi\alpha_i + \xi_{i+1}\alpha_{i+1}) \tag{11}$$

$$X_{i+2} = \sqrt{1+2^{-2(i-1)}}\sqrt{1+2^{-2(i-2)}}R_i \cdot \cos(\theta_i + \xi_i\alpha_i + \xi_{i+1}\alpha_{i+1}). \tag{12}$$

Similarly, the pseudo-rotation steps can be continued through any finite, pre-established number of steps without regard to the values assigned to ξ. Consider the initial coordinate components Y_1 and X_1 where

$$Y_1 = R_1 \sin\theta_1 \tag{13}$$

$$X_1 = R_1 \cos\theta_1. \tag{14}$$

By establishing the first and most significant step as a rotation through $\pm 90°$, and by establishing the number of steps as n, the expression for the final coordinate components will be

$$Y_{n+1} = [\sqrt{1+2^{-0}}\sqrt{1+2^{-2}} \cdots \sqrt{1+2^{-2(n-2)}}]R_1 \cdot \sin(\theta_1 + \xi_1\alpha_1 + \xi_2\alpha_2 + \cdots + \xi_n\alpha_n) \tag{15}$$

$$X_{n+1} = [\sqrt{1+2^{-0}}\sqrt{1+2^{-2}} \cdots \sqrt{1+2^{-2(n-2)}}]R_1 \cdot \cos(\theta_1 + \xi_1\alpha_1 + \xi_2\alpha_2 + \cdots + \xi_n\alpha_n) \tag{16}$$

The increase in magnitude of the components for a particular value of n is a constant and will be represented by the letter K. The value selected for n is a function of the desired computing accuracy and can be a constant for a particular computer. For example,

$$\text{if } n = 24, \quad K = 1.646760255. \tag{17}$$

The necessary functional components and information flow for instrumenting the cross-addition are associated with the Y and X registers shown in Fig. 2.

It has not yet been shown how the prescribed sequence of rotation steps can be controlled to effect the desired over-all rotation. By examination of (15) and (16), it may be shown that, for a rotation of a set of coordinate components Y_1 and X_1 through a given angle (as required in the ROTATION mode), it is necessary to obtain the expressions

$$Y_{n+1} = KR_1 \sin(\theta_1 + \lambda) \tag{18}$$

$$X_{n+1} = KR_1 \cos(\theta_1 + \lambda). \tag{19}$$

To obtain the relationships expressed in (18) and (19), it is required that

$$\lambda = \xi_1\alpha_1 + \xi_2\alpha_2 + \cdots + \xi_n\alpha_n \tag{20}$$

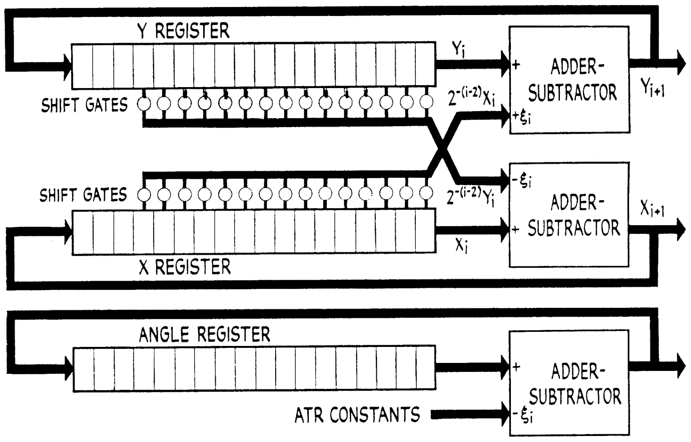

Fig. 2—CORDIC arithmetic unit.

and, as explained previously, for VECTORING it is required that

$$-\theta_1 = \xi_1\alpha_1 + \xi_2\alpha_2 + \cdots + \xi_n\alpha_n. \tag{21}$$

The sequences of (20) and (21) form a special radix representation equivalent to the desired angle, λ or θ where

$$\alpha_1 = 90° \tag{22}$$

$$\alpha_2 = \tan^{-1} 2^{-0} = 45° \tag{23}$$

$$\alpha_3 = \tan^{-1} 2^{-1} \approx 26.5° \tag{24}$$

$$\alpha_i = \tan^{-1} 2^{-(i-2)}. \tag{25}$$

The α terms are referred to as ATR (Arc Tangent Radix) constants, and are precomputed and stored in the computer. The ξ terms are referred to as ATR digits and are determined during each operation.

In the CORDIC computer, the ATR digits are determined sequentially, most significant digit first, and are used to control the conditional action of the adder-subtractors in the arithmetic unit. The following paragraphs contain a description of the manner in which the ATR code representation, $\xi_1\xi_2\xi_3 \cdots \xi_n$, can be determined for any given angle, λ or θ.

First, for any angle, λ or θ, there must be at least one set of values for the ξ operators that will satisfy (20) or (21). Second, a simple technique must be available for determining the ATR code digits that satisfy these equations.

The following relationships are necessary and sufficient for *any* sequence of radix constants to meet the above requirements.

$$|\lambda \text{ or } \theta| \leq \alpha_1 + \alpha_2 + \alpha_3 + \cdots + \alpha_n + \alpha_n \tag{26}$$

$$\alpha_i \leq \alpha_{i+1} + \alpha_{i+2} + \cdots + \alpha_n + \alpha_n. \tag{27}$$

For the satisfaction of the stipulative equations [(10) and (22)], it is required that λ or θ be represented

$$-180° \leq \lambda \text{ or } \theta < +180°. \tag{28}$$

Eq. 28 imposes no special consideration if the two's complement notation is used.

By employing an additional register and adder-subtractor (identified in Fig. 2 as the angle register) the relationship of (18) (ROTATION mode) can be instrumented by: 1) sensing the sign of the angle of ROTATION (or remainder if $i>1$); and 2) either subtracting or adding to the angle the ATR constant corresponding to the particular step. In each step, the relationship instrumented is:

$$|\lambda_{i+1}| = \left| |\lambda_i| - \alpha_i \right|. \tag{29}$$

Execution of the first step of the nulling sequence to (26) results in

$$-\alpha_1 \leq |\lambda| - \alpha_1 \leq \alpha_2 + \alpha_3 + \cdots + \alpha_n + \alpha_n. \tag{30}$$

Application of the relationships of (27) results in

$$|\lambda_2| \equiv \left| |\lambda_1| - \alpha_1 \right| \leq \alpha_2 + \alpha_3 + \cdots + \alpha_n + \alpha_n. \tag{31}$$

Continuation of the nulling sequence through α_n results in

$$|\lambda_{n+1}| \leq \alpha_n. \tag{32}$$

Eq. (32) can be used to prove that the remainder in the angle register converges to zero in the ROTATION mode.

The sequence of operation signs used to null λ to zero is the negative of the equivalent ATR code for the original angle λ_1. More simply, the ATR code digit of each step is equal to the sign of the quantity in the angle register before each step. Therefore, simultaneously with each nulling step in the angle register, the ATR code digit may be used to control the cross-addition step in the Y and X registers (shown in Fig. 2) to effect a rotation of components through an equal angular increment.

The proof of the convergence of the effective angular argument θ_{n+1} to zero, which is necessary in the VECTORING mode, may be obtained by replacing λ by θ in (29) through (32). In VECTORING, the sign of the angle θ_i is obtained by sensing the sign of Y_i. The sequence of signs of Y_i is the negative of the ATR code for the effective rotation performed on the components Y_1 and X_1. During each cross-addition operation in the Y and X register, the corresponding ATR constant can be conditionally added or subtracted, depending on ξ_i, to an accumulating sum in the angle register so that, at the end of the computing sequence, when $\theta_{n+1}=0$, the quantity in the angle register will be equal to the original angular argument θ_1 of the coordinate components Y_1 and X_1.

The step-by-step results of a typical rotation computing sequence are shown in Table I. The two's complement notation is used for all quantities, and, for simplicity, shifted quantities are simply truncated without round-off.

The step-by-step results of a typical vectoring computing sequence are shown in Table II.

Fig. 3 contains the solution flow for solving a typical navigation problem with the CORDIC computing technique. Specifically, this example shows the program sequence necessary for solving for the course angle and the distance-to-destination relationships in great circle steering.

TABLE I
TYPICAL ROTATION COMPUTING SEQUENCE

Y Register	X Register	Angle Register	
$Y_1=0.0101110$	$1.1000101=X_1$	$0.1100101=\lambda$	
$+1.1000101$	-0.0101110	-0.1000000	$\tan^{-1}\infty$
1.1000101	1.1010010	0.0100101	
$+1.1010010$	-1.1000101	-0.0100000	$\tan^{-1} 1$
1.0010111	0.0001101	0.0000101	
$+0.0000110$	-1.1001011	-0.0010010	$\tan^{-1} 2^{-1}$
1.0011101	0.1000010	1.1110011	
-0.0010000	$+1.1100111$	$+0.0001001$	$\tan^{-1} 2^{-2}$
1.0001101	0.0101001	1.1111100	
-0.0000101	$+1.1110001$	$+0.0000101$	$\tan^{-1} 2^{-3}$
1.0001000	0.0011010	0.0000001	
$+0.0000001$	-1.1111000	-0.0000010	$\tan^{-1} 2^{-4}$
1.0001001	0.0100010	1.1111111	
-0.0000001	$+1.1111100$	$+0.0000001$	$\tan^{-1} 2^{-5}$
1.0001000	0.0011110	0.0000000	

TABLE II
TYPICAL VECTORING COMPUTING SEQUENCE

Y Register	X Register	Angle Register	
$Y_1=0.0101110$	$1.1000101=X_1$	0.0000000	
-1.1000101	$+0.0101110$	$+0.1000000$	$\tan^{-1}\infty$
0.0111011	0.0101110	0.1000000	
-0.0101110	$+0.0111011$	$+0.0100000$	$\tan^{-1} 1$
0.0001101	0.1101001	0.1100000	
-0.0110100	$+0.0000110$	$+0.0010010$	$\tan^{-1} 2^{-1}$
1.1011001	0.1101111	0.1110010	
$+0.0011011$	-1.1110110	-0.0001001	$\tan^{-1} 2^{-2}$
1.1110100	0.1111001	0.1101001	
$+0.0001111$	-1.1111110	-0.0000101	$\tan^{-1} 2^{-3}$
0.0000011	0.1111011	0.1100100	
-0.0000111	$+0.0000000$	$+0.0000010$	$\tan^{-1} 2^{-4}$
1.1111100	0.1111011	0.1100110	
$+0.0000011$	-1.1111111	-0.0000001	$\tan^{-1} 2^{-5}$
1.1111111	$0.1111100=KR_1$	$0.1100101=\theta$	

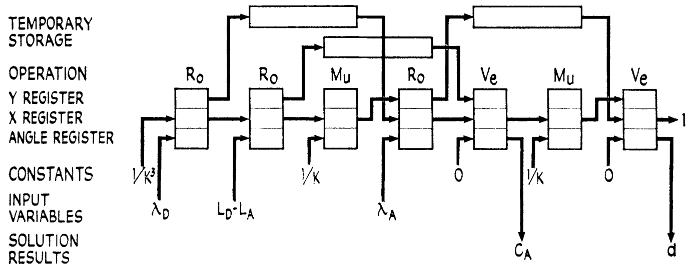

Fig. 3—Solution flow diagram.

Each operation of the computing sequence is represented by a box containing the Y, X and Angle registers. The particular operation performed is abbreviated as follows:

$$R_0 = \text{Rotation},$$
$$V_e = \text{Vectoring},$$
$$M_u = \text{Multiplication}.$$

The explicit equations solved are

$$C_A = \tan^{1} \frac{\cos \lambda_D \sin (L_D - L_A)}{\sin \lambda_D \cos \lambda_A - \cos \lambda_D \sin \lambda_A \cos (L_D - L_A)} \tag{33}$$

$$d = \tan^{-1} \frac{\cos \lambda_D \sin (L_D - L_A)}{\sin C_A [\sin \lambda_D \sin \lambda_A + \cos \lambda_D \cos \lambda_A \cos (L_D - L_A)]} \tag{34}$$

where

C_A = course angle to destination
d = distance to destination
λ_A = present latitude
L_A = present longitude
λ_D = latitude of destination
L_D = longitude of destination.

The form of (33) and (34) cannot be used for establishing the CORDIC solution-flow diagram. It is necessary to express the relationship with some form of rotation operators, such as rotation matrices; a similar change in form is also necessary for establishing analog resolver solution-flow diagrams. The only difference between Fig. 3 and an equivalent analog resolver solution flow diagram is the insertion of multiplication routines to compensate for the magnitude change factor K of each trigonometric operation. Note that, although five trigonometric operations are performed, only two multiplication operations are necessary.

Conclusion

The CORDIC computing technique is especially suitable for use in a special-purpose computer where the majority of the computations involve trigonometric relationships. In general, the ROTATION and VECTORING operations should be considered constant-length routines in which the number of word times per operation is equal to the word length.

While not covered in this paper, similar algorithms have been developed for multiplication, division, conversion between binary and mixed radix systems, extractions of square root, hyperbolic coordinate transformations, exponentiation and generation of logarithms.

It is believed that similar algorithms based on this fundamental concept of computation could be developed for many other computing requirements.