Trigonometry/For Enthusiasts/The CORDIC Algorithm

What it is

CORDIC (for **CO**ordinate **R**otation **DI**gital **C**omputer) is a simple and efficient algorithm to calculate trigonometric functions. The only operations it requires are

- Addition,
- Subtraction,
- Multiplications and division by two and
- Table lookup (a table with 64 numbers in it is enough for all the cosines and sines that a handheld calculator can calculate).

Because computers use binary arithmetic internally, multiplications and divisions by two are quick and easy to do. Consequently the CORDIC algorithm allows trigonometric functions to be calculated efficiently with a relatively simple CPU.

Applications

CORDIC is particularly well-suited for handheld calculators, an application for which cost (e.g., chip gate count has to be minimized) is much more important than is speed. Also the CORDIC subroutines for trigonometric and hyperbolic functions (described in Trigonometry Book 2) can share most of their code.

Mode of operation

CORDIC can be used to calculate a number of different functions. This explanation shows how to use CORDIC in rotation mode to calculate the sine and cosine of an angle, and assumes the desired angle is given in radians and represented in a fixed point format. To determine the sine or cosine for an angle β , the y or x coordinate of a point on the unit circle corresponding to the desired angle must be found. Using CORDIC, we would start with the vector v_0 :

$$v_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

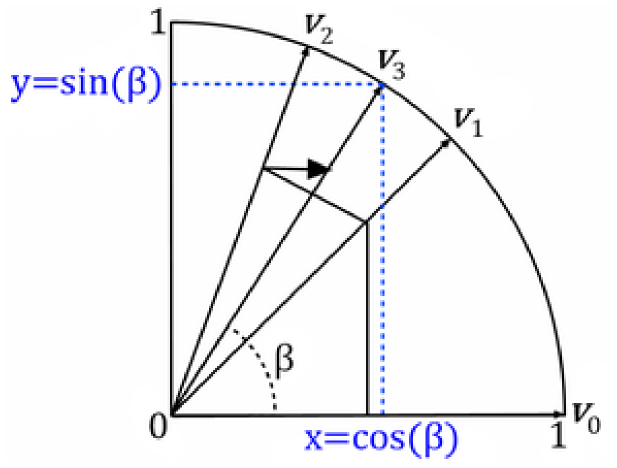
In the first iteration, this vector would be rotated 45° counterclockwise to get the vector v_1 . Successive iterations will rotate the vector in one or the other direction by steps decreasing in size, until the desired angle has been achieved. Step i size is Artg(1/(2$^{(i-1)}$)) where i 1,2,3,...

More formally, every iteration calculates a rotation, which is performed by multiplying the vector v_{i-1} with the rotation matrix R_i:

$$v_i = R_i v_{i-1}$$

The rotation matrix R is given by:

$$R_i = \begin{pmatrix} \cos(\gamma_i) & -\sin(\gamma_i) \\ \sin(\gamma_i) & \cos(\gamma_i) \end{pmatrix}$$



An illustration of the CORDIC algorithm in progress.

Using the following two trigonometric identities

$$\cos(\alpha) = \frac{1}{\sqrt{1+\tan^2(\alpha)}}$$

$$\sin(\alpha) = \frac{\tan(\alpha)}{\sqrt{1+\tan^2(\alpha)}}$$

the rotation matrix becomes:

$$R_i = \frac{1}{\sqrt{1+\tan^2\gamma_i}} \begin{pmatrix} 1 & -\tan\gamma_i \\ \tan\gamma_i & 1 \end{pmatrix}$$

The expression for the rotated vector $v_i = R_i v_{i-1}$ then becomes:

$$v_i = \frac{1}{\sqrt{1+\tan^2\gamma_i}} \begin{pmatrix} x_{i-1} & - & y_{i-1}\tan\gamma_i \\ x_{i-1}\tan\gamma_i & + & y_{i-1} \end{pmatrix}$$

where x_{i-1} and y_{i-1} are the components of v_{i-1}. Restricting the angles γ_i so that $\tan\gamma_i$ takes on the values $\pm 2^{-i}$ the multiplication with the tangent can be replaced by a division by a power of two, which is efficiently done in digital computer hardware using a bit shift. The expression then becomes:

$$v_i = K_i \begin{pmatrix} x_{i-1} & - & \sigma_i 2^{-i} y_{i-1} \\ \sigma_i 2^{-i} x_{i-1} & + & y_{i-1} \end{pmatrix}$$

where

$$K_i = \frac{1}{\sqrt{1+2^{-2i}}}$$

and σ_i can have the values of −1 or 1 and is used to determine the direction of the rotation: if the angle β_i is positive then σ_i is 1, otherwise it is −1.

We can ignore K_i in the iterative process and then apply it afterward by a scaling factor:

$$K(n) = \prod_{i=0}^{n-1} K_i = \prod_{i=0}^{n-1} 1/\sqrt{1+2^{-2i}}$$

which is calculated in advance and stored in a table, or as a single constant if the number of iterations is fixed. This correction could also be made in advance, by scaling v_0 and hence saving a multiplication. Additionally it can be noted that

$$K = \lim_{n\to\infty} K(n) \approx 0.6072529350088812561694$$[1]

to allow further reduction of the algorithm's complexity. After a sufficient number of iterations, the vector's angle will be close to the wanted angle β. For most ordinary purposes, 40 iterations (n = 40) is sufficient to obtain the correct result to the 10th decimal place.

The only task left is to determine if the rotation should be clockwise or counterclockwise at every iteration (choosing the value of σ). This is done by keeping track of how much we rotated at every iteration and subtracting that from the wanted angle, and then checking if β_{n+1} is positive and we need to rotate clockwise or if it is negative we must rotate counterclockwise in order to get closer to the wanted angle β.

$$\beta_i = \beta_{i-1} - \sigma_i \gamma_i. \quad \gamma_i = \arctan 2^{-i},$$

The values of γ_n must also be precomputed and stored. But for small angles, $\arctan(\gamma_n) = \gamma_n$ in fixed point representation, reducing table size.

As can be seen in the illustration above, the sine of the angle β is the y coordinate of the final vector v_n, while the x coordinate is the cosine value.

- See Cordic on wikipedia for more information, including a software implementation.

Hardware implementation

The primary use of the CORDIC algorithms in a hardware implementation is to avoid time-consuming complex multipliers. The computation of phase for a complex number can be easily implemented in a hardware description language using only adder and shifter circuits bypassing the bulky complex number multipliers. Fabrication techniques have steadily improved, and complex numbers can now be handled directly without too high a cost in time, power consumption, or excessive die space, so the use of CORDIC techniques is not as critical in many applications as they once were.

Related algorithms

CORDIC is part of the class of "shift-and-add" algorithms, as are the logarithm and exponential algorithms derived from Henry Briggs' work. Another shift-and-add algorithm which can be used for computing many elementary functions is the BKM algorithm, which is a generalization of the logarithm and exponential algorithms to the complex plane. For instance, BKM can be used to compute the sine and cosine of a real angle x (in radians) by computing the exponential of $0 + ix$, which is $\cos x + i \sin x$. The BKM algorithm is slightly more complex than CORDIC, but has the advantage that it does not need a scaling factor (K).

Origins

The modern CORDIC algorithm was first described in 1959 by Jack E. Volder. It was developed at the aeroelectronics department of Convair to replace the analog resolver in the B-58 bomber's navigation computer.[2]

Although CORDIC is similar to mathematical techniques published by Henry Briggs as early as 1624, it is optimized for low complexity finite state CPUs.

John Stephen Walther at Hewlett-Packard further generalized the algorithm, allowing it to calculate hyperbolic and exponential functions, logarithms, multiplications, divisions, and square roots.[3]

Originally, CORDIC was implemented using the binary numeral system. In the 1970s, decimal CORDIC became widely used in pocket calculators, most of which operate in binary-coded-decimal (BCD) rather than binary.

Hardware

CORDIC is generally faster than other approaches when a hardware multiplier is unavailable (e.g., in a microcontroller based system), or when the number of gates required to implement the functions it supports should be minimized (e.g., in an FPGA).

On the other hand, when a hardware multiplier is available (e.g., in a DSP microprocessor), table-lookup methods and power series are generally faster than CORDIC. In recent years, CORDIC algorithm is used extensively for various biomedical applications, especially in FPGA implementations.

Software

Many older systems with integer only CPUs have implemented CORDIC to varying extents as part of their IEEE Floating Point libraries. As most modern general purpose CPUs have floating point registers with common operations such as add, subtract, multiply, divide, sin, cos, square root, log10, natural log, the need to implement CORDIC in them with software is nearly non-existent. Only microcontroller or special safety and time constraint software applications would need to consider using CORDIC.

History

Volder was inspired by the following formula in the 1946 edition of the CRC Handbook of Chemistry and Physics:

$$K_n R\sin(\theta\pm\phi) = R\sin(\theta) \pm 2^{-n} R\cos(\theta)$$
$$K_n R\cos(\theta\pm\phi) = R\cos(\theta) \mp 2^{-n} R\sin(\theta)$$

with $K_n = \sqrt{1+2^{-2n}}, \tan(\phi) = 2^{-n}$. [2]

Some of the prominent early applications of CORDIC were in the Convair navigation computers CORDIC I to CORDIC III[2], the Hewlett-Packard HP-9100 and HP-35 calculators[4], the Intel 80x87 coprocessor series until Intel 80486, and Motorola 68881[5].

Decimal CORDIC was first suggested by Hermann Schmid and Anthony Bogacki.[6]

Notes

1. J.-M. Muller, Elementary Functions: Algorithms and Implementation, 2nd Edition (Birkhäuser, Boston, 2006), p. 134. (http://perso.ens-lyon.fr/jean-michel.muller/SecondEdition.html)
2. J. E. Volder, "The Birth of CORDIC", J. VLSI Signal Processing **25**, 101 (2000). (https://dx.doi.org/10.1023/A:1008110704586)
3. J. S. Walther, "The Story of Unified CORDIC", J. VLSI Signal Processing **25**, 107 (2000). (https://dx.doi.org/10.1023/A:1008162721424)
4. D. Cochran, "Algorithms and Accuracy in the HP 35", Hewlett Packard J. **23**, 10 (1972).
5. R. Nave, "Implementation of Transcendental Functions on a Numerics Processor", Microprocessing and Microprogramming **11**, 221 (1983).
6. H. Schmid and A. Bogacki, "Use Decimal CORDIC for Generation of Many Transcendental Functions", EDN Magazine, February 20, 1973, p. 64.

References

- Jack E. Volder, *The CORDIC Trigonometric Computing Technique, IRE Transactions on Electronic Computers*, pp330-334, September 1959 (http://www.jacques-laporte.org/Volder_CORDIC.pdf)
- Daggett, D. H., *Decimal-Binary conversions in CORDIC*, IRE Transactions on Electronic Computers, Vol. EC-8 #5, pp335-339, IRE, September 1959
- John S. Walther, *A Unified Algorithm for Elementary Functions*, Proc. of Spring Joint Computer Conference, pp379-385, May 1971 (http://www.jacques-laporte.org/Welther-Unified%20Algorithm.pdf)
- J. E. Meggitt, *Pseudo Division and Pseudo Multiplication Processes*, IBM Journal, April 1962 (http://www.jacques-laporte.org/Meggitt_62.pdf)
- Vladimir Baykov, *Problems of Elementary Functions Evaluation Based on Digit by Digit (CORDIC) Technique*, PhD thesis, Leningrad State Univ. of Electrical Eng., 1972 (http://baykov.de/cordic1972.htm)
- Schmid, Hermann, *Decimal computation.* New York, Wiley, 1974
- V.D.Baykov, V.B.Smolov, *Hardware implementation of elementary functions in computers*, Leningrad State University, 1975, 96p. (http://baykov.de/cordic1975.htm)*Full Text (http://www.umup.narod.ru/1115.zip)
- Senzig, Don, *Calculator Algorithms*, IEEE Compcon Reader Digest, IEEE Catalog No. 75 CH 0920-9C, pp139-141, IEEE, 1975.
- V.D.Baykov,S.A.Seljutin, *Elementary functions evaluation in microcalculators*, Moscow, Radio & svjaz,1982,64p.
- Vladimir D.Baykov, Vladimir B.Smolov, Special-purpose processors: iterative algorithms and structures, Moscow, Radio & svjaz, 1985, 288 pages (http://baykov.de/cordic1985.htm)
- M. E. Frerking, *Digital Signal Processing in Communication Systems, 1994*

- Vitit Kantabutra, *On hardware for computing exponential and trigonometric functions*, IEEE Trans. Computers 45 (3), 328-339 (1996)
- Andraka, Ray, *A survey of CORDIC algorithms for FPGA based computers* (http://www.andraka.com/files/crdcsrvy.pdf)
- Henry Briggs, *Arithmetica Logarithmica.* London, 1624, folio
- *CORDIC Bibliography Site* (http://web.archive.org/web/20001017173921/devil.ece.utexas.edu/)
- *The secret of the algorithms* (http://www.jacques-laporte.org/TheSecretOfTheAlgorithms.htm), Jacques Laporte, Paris 1981
- *Digit by digit methods* (http://www.jacques-laporte.org/digit_by_digit.htm), Jacques Laporte, Paris 2006
- Ayan Banerjee, FPGA realization of a CORDIC based FFT processor for biomedical signal processing (http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6V0X-4313PR1-1&_user=10&_rdoc=1&_fmt=&_orig=search&_sort=d&_docanchor=&view=c&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=c85888a82db1f3b515b4a8d2a529624b), Kharagpur, 2001

Credits

- This page was originally created from The CORDIC Algorithm at Wikipedia.