2020

关于计算的报告

CORDIC 算法原理介绍

刘行远（1900012482）

摘 要：介绍了用于计算三角函数的 CORDIC 算法的原理，主要阐述了其与加法公式有关的数学原理。说明了 CORDIC 算法相对于其他算法的优势，以及其产生的背景因素。指出 CORDIC 算法在数学上起源于坐标系的旋转变换，而本质上是三角函数加法公式的引申。阐明了如何利用 CORDIC 算法计算其他函数，并指出过程的共性。说明了传统 CORDIC 算法的局限性。
关键词：CORDIC 算法、三角函数、加法公式

1 引言

CORDIC 算法（Coordinate Rotation Digital Computer），意为坐标旋转数字算法，是一种 J.D.Volder1 于 1959 年首次提出的，用于对三角函数进行计算的方法。CORDIC 算法的核心是基于平面坐标系的旋转变换公式，其与三角函数的加法公式密切相关。同时，CORDIC 算法在具体的运算过程中做出了一定的优化，以与计算机各类运算的效率相匹配。与查表法以及其他利用三角函数性质的算法相比，CORDIC 算法原理简单，且无需储存大量的数据以进行运算，同时效率较高，因此有广泛的应用。

三角函数的运算，在历史上对天文学有着重要的影响。托勒密的《天文学大成》中，就有一张关于弦角关系的弦表，被认为是最早的弦表。到了 15 世纪后，航海业的发展推进了对三角学研究的需求，更加细致的三角函数表被制作出来以供应用。直至上世纪下半叶，我国科研工作者仍常通过各类纸质三角函数表进行三角函数的运算。

上世纪中叶的电子计算机这一发明，使得复杂计算的工作能够交给机器运行。三角函数的计算有多种方法，如二分法、级数展开法。但由于传统算法涉及开平方、乘除等对于计算机来说效率较低的运算，并不完全适合计算机应用。而查表法虽然较为快捷，但需要存储大量的数据，且不能实现较高的精度。而 CORDIC 算法的提出，大大提高了三角函数的计算速度。同时，CORDIC 算法能够推广至其他函数，如双曲三角函数、指数函数等。这使得这种算法成为了这一系列函数较常用的算法。

2 CORDIC 算法原理

CORDIC 算法来源于平面直角坐标系中的坐标旋转变换式:

$$\begin{cases} x' = xcos\theta - ysin\theta \\ y' = xsin\theta + ycos\theta \end{cases}$$

上式将点(x,y)绕原点 O 旋转θ角（逆时针为正，顺时针为负），变换到点(x',y')，变换过程中点到原点 O 的距离不变。

以上变换式常作为 CORDIC 算法的基本原理进行推导，但未说明 CORDIC 算法作为三角函数的算法的本质。CORDIC 算法的核心在于三角函数的加法公式:

$$\begin{cases} cos(\varphi + \theta) = cos\varphi cos\theta - sin\varphi sin\theta \\ sin(\varphi + \theta) = sin\varphi cos\theta + cos\varphi sin\theta \end{cases}$$

以及正切函数的加法公式:

$$tan(\varphi + \theta) = \frac{tan\varphi + tan\theta}{1 - tan\varphi tan\theta}$$

假设有一个数列$t_i = tan\theta_i$，则当$\theta_i \to \theta$时有$t_i \to tan\theta$，因此，只需构造出θ_i使得$\lim\limits_{n\to\infty} \theta_i = \theta$即可。假设$\theta_n = \sum_{i=0}^{n} \varphi_i$，则

$$t_{i+1} = \frac{t_i + tan\varphi_i}{1 - t_i\, tan\varphi_i}$$

由此得到了收敛于待求函数值的数列的递推公式。此时，若数列φ_i 的构造方法是固定的，那么只需计算各个$tan\varphi_i$的值以及每部递推的四则运算，问题得到初步化简。但对于运算中的乘除法的化简，则依赖于φ_i的选取。

CORDIC 算法另一关键在于，选取$\{arctan(2^{-i}) | i = 0,1,\dots\}$中的元素作为$\varphi_i$的值或者相反值。此时，递推式的形式可化为

$$t_{i+1} = \frac{t_i \pm 2^{-k}}{1 \mp t_i\, 2^{-k}}$$

其中每一步递推式中 k 与正负号的取值由θ_i与θ的差值决定。上式中的乘法运算，由于计算机的二进制特性，在实际操作中转化为了对浮点数的移位操作，大大减少了运算量。

对于递推式可进行进一步的简化。构造浮点数序列x_i，y_i，使得$t_i = \frac{y_i}{x_i}$，则有

$$\frac{y_{i+1}}{x_{i+1}} = \frac{y_i \pm x_i 2^{-k}}{x_i \mp y_i\, 2^{-k}}$$

很自然地，可令

$$\begin{cases} x_{i+1} = x_i \mp y_i\, 2^{-k} \\ y_{i+1} = y_i \pm x_i 2^{-k} \end{cases}$$

此时，递推式中仅涉及到加减法与移位运算，使得计算速度有了相当程度的提高。在θ_i与θ的差值达到给定精度要求时，

可停止递推操作，通过$tan\theta = \frac{y}{x}$ 计算出正切值。同时，利用正切值与正余弦的转换，可得出所有的三角函数值。

在完成了主要的计算原理后，重点在于如何构造满足条件的φ_i。在经典 CORDIC 算法中，由等式$arctan2x < 2arctanx$，可令$\varphi_i = \pm arctan(2^{-i})$，此时$|\varphi_{i+1}| < |\varphi_i| < 2|\varphi_{i+1}|$，可以证明，可通过类似二进制的方式将$\theta$表示为$\sum_{i=0}^{\infty}\varphi_i$。但此过程中，需要对每一步的正负号加以判别。已有许多方法对经典 CORDIC 算法进行改良，如平行 CORDIC 算法等，此处不作赘述。

3 对加法公式的利用

CORDIC 算法的重点，在于对三角函数的加法公式的分析。正切函数的加法公式的结构包括两个自变量、自变量的正切函数、对函数值的初等运算，即

$$tan(\varphi + \theta) = \frac{tan\varphi + tan\theta}{1 - tan\varphi tan\theta}$$

在初等函数中，有一类函数的性质与三角函数极其相似，即双曲三角函数：$sinh\,x, cosh\,x, tanh\,x$。双曲三角函数的加法公式的形式与三角函数极为相似，如：

$$tanh(\varphi + \theta) = \frac{tanh\varphi + tanh\theta}{1 + tanh\varphi tanh\theta}$$

因此，可以推想，CORDIC 算法也可推广至双曲三角函数上。令

$$\varphi_i = \pm artanh\left(2^{-i}\right)(i = 1,2 \ldots)$$

同样，也可构造递推列：

$$\begin{cases} x_{i+1} = x_i \pm y_i\, 2^{-k} \\ y_{i+1} = y_i \pm x_i 2^{-k} \end{cases}$$

值得注意的是，双曲正切函数不仅可以用于推导相应的双曲三角函数，且有公式：

$$tanhx = \frac{e^x - 1}{e^x + 1}$$

$$e^x = \frac{1 + tanhx}{1 - tanhx} = \frac{2}{1 - tanhx} - 1$$

由此，可从双曲三角函数得到指数函数的值。因此，通过 CORDIC 算法，可以几乎得到所有基本初等函数的值。与知名度更高的泰勒级数法相比，CORDIC 算法更具有实际可操作性与高效率。

4 结语

本文对于 CORDIC 算法的基本原理进行了介绍，通过三角函数的加法公式导出了计算过程中的递推式，并且对于 CORDIC 算法如何利用电子计算机的运算特性对计算过程进行化简，作出了简单的解释。同时，又对于 CORDIC 算法的可行性进行了阐明，表明如何通过预先给定的量构造出自变量，完善了递推过程的细节。在讲完 CORDIC 算法的原理后，又对其中作为核心原理的加法公式进行了解释，并说明了通过 CORDIC 算法求得其他超越函数的过程。

限于篇幅，本文未能具体说明 CORDIC 算法的各种特性，如收敛性、稳定性、复杂度等。本文仅阐述了 CORDIC 算法的数学原理与部分实现原理，对于算法的具体实现并没有涉及。对于非传统 CORDIC 算法，由于水平所限，没有给出完整的介绍。由于传统 CORDIC 算法每次迭代都需计算下次旋转的方向，算法的效率受到了一定的影响。许多文献已给出了各类优化改进的方法[1][3]，此处不作赘述。

参考文献:

[1]蔡权利,高博,龚敏.基于 FPGA 的 CORDIC 算法实现[J].电子器件,2018,41(05):1242-1246+1256.

[2]姚芳,刘晓婷.历史上最早构造的三角函数表——弦表[J].数学通报,2008,47(11):23-26.

[3]李雪,徐洋洋,邱雅倩,姚亚峰.一种改进 CORDIC 算法的反正切计算[J].电视技术,2019,43(02):14-18.