The CORDIC Method

Computer Architecture

September 18, 2018

© 2018 Oprițoiu Flavius. All Rights Reserved.
Introduction

Objectives:

- Understand the CORDIC method for trigonometric functions evaluation

**Coordinate Rotation Digital Computer method:**
- Developed by J Volder in 1959 while working to improve the navigation system of the B-58 bomber [Vold00]
- Used for approximation of $\cos(\theta)$ and $\sin(\theta)$ for angles belonging to the first or the fourth unity circle quadrants
- Generalized to allow computation of logarithms, exponentials and square roots [Walt00]
- Favorable for hardware implementation
The CORDIC method approximates $\cos(\theta)$ and $\sin(\theta)$, by rotating a unity vector, initially positioned along abscissa.

Rotation of the initial vector by angle $\theta$ brings the vector in its final position for which abscissa equals $\cos(\theta)$ and ordinate equals $\sin(\theta)$, respectively, as illustrated below:

The CORDIC method can be used for angles $\theta$ in interval $-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$.

© 2018 Opritoiu Flavius. All Rights Reserved.
Rotation of a vector

The vector is rotated in a series of *micro-rotations*. At each micro-rotation step, the vector is rotated by a pre-configured angle increment. Moreover, for each step, the rotation can be either positive or negative.

The angle increments are chosen at design time in order to facilitate abscissa and ordinate update.

The CORDIC method has linear convergence meaning it requires \( N + 1 \) iterations for obtaining a result with an \( N \)-bit precision.

© 2018 Oprițoiu Flavius. All Rights Reserved.
CORDIC micro-rotation

The $i$-th micro-rotation, depicted below, realizes a rotation by angle $\alpha_i$ transforming vector $\overrightarrow{OP(i)}$ into vector $\overrightarrow{OP(i+1)}$:

The coordinates update is expressed as:

\[
\begin{bmatrix}
x(i+1) \\
y(i+1)
\end{bmatrix} = \frac{1}{\sqrt{1 + \tan^2(\alpha_i)}} \begin{bmatrix}
1 & -\sigma_i \tan(\alpha_i) \\
\sigma_i \tan(\alpha_i) & 1
\end{bmatrix} \begin{bmatrix}
x(i) \\
y(i)
\end{bmatrix}
\]

in which $\sigma_i$ is 1 for a positive rotation and -1 for a negative one.
Micro-rotation angles, $\alpha_i$, are selected so that $\tan(\alpha_i) = 2^{-i}$. As a result, multiplication by $\tan(\alpha_i)$ is implemented as rightshifting.

After $n$ micro-rotations were performed, the $x$ and $y$ coordinates become:

$$\begin{bmatrix} x(n) \\ y(n) \end{bmatrix} = \frac{1}{k_{n-1} \cdots k_0} \begin{bmatrix} 1 & -\sigma_{n-1}2^{-n+1} \\ \sigma_{n-1}2^{-n+1} & 1 \end{bmatrix} \cdots \begin{bmatrix} 1 & -\sigma_02^{-0} \\ \sigma_02^{-0} & 1 \end{bmatrix} \begin{bmatrix} x(0) \\ y(0) \end{bmatrix}$$

in which $x(0) = 1$, $y(0) = 0$ and $k_i = \sqrt{1 + 2^{-2i}}$
The method uses variable $z$ for tracking the angular error, and it is initialized to angle $\theta$. Variable $z$ is updated, at each step, by:

$$z(i) = z(i - 1) - \sigma_i \arctan(2^{-i+1}), \text{ for } i > 0$$

Coefficients $\sigma_i$ are chosen based on the sign of $z_{i-1}$, so that:

$$\sigma_i = \begin{cases} -1 & \text{if } z(i - 1) < 0, \\ 1 & \text{if } z(i - 1) \geq 0 \end{cases}$$

By updating variable $z$ in this manner, it will converge towards $0$ concurrent with variable $x$’s convergence towards $\cos(\theta)$. 

© 2018 Oprițoiu Flavius. All Rights Reserved.
Scaling factors

According to equation (1), after all micro-rotation steps were performed, the \( x(n) \) and \( y(n) \) coordinates need to be scaled down by factor \( K = k_{n-1}k_{n-2}...k_0 \).

Typically, the number of steps of the CORDIC approximation is fixed and, as a result, all factors \( k_i \) are constant and can be pre-computed.

Instead of dividing \( x(n) \) and \( y(n) \) by \( K \) in (1), coordinates \( x(0) \) and \( y(0) \) can be initialized to:

\[
\begin{align*}
x(0) &= \frac{1}{K} \\
y(0) &= 0
\end{align*}
\]
CORDIC algorithm for $\cos(\theta)$

1: procedure $\text{Cos}(\theta)$ \hspace{1cm} $\triangleright$ $n$-step approximation, $n$ fixed
2: \hspace{1cm} $x(0) \leftarrow \frac{1}{K}$ \hspace{1cm} $\triangleright$ pre-compute factor $K$
3: \hspace{1cm} $y(0) \leftarrow 0$; $z(0) \leftarrow \theta$ \hspace{1cm} $\triangleright$ with $0 \leq \theta \leq \frac{\pi}{2}$
4: \hspace{1cm} for $i = 0$ to $n - 1$ do
5: \hspace{2cm} if $z(i) \geq 0$ then
6: \hspace{3cm} $x(i + 1) \leftarrow x(i) - (2^{-i}y(i))$
7: \hspace{3cm} $y(i + 1) \leftarrow y(i) + (2^{-i}x(i))$
8: \hspace{3cm} $z(i + 1) \leftarrow z(i) - \arctan(2^{-i})$
9: \hspace{2cm} else
10: \hspace{3cm} $x(i + 1) \leftarrow x(i) + (2^{-i}y(i))$
11: \hspace{3cm} $y(i + 1) \leftarrow y(i) - (2^{-i}x(i))$
12: \hspace{3cm} $z(i + 1) \leftarrow z(i) + \arctan(2^{-i})$
13: \hspace{2cm} end if
14: \hspace{1cm} end for
15: \hspace{1cm} return $x(n)$
16: end procedure
Fixed-point fractional numbers

Point coordinates on the unity circle, $x$ and $y$, can have values in interval $[-1, 1]$. In addition, for the CORDIC method, angle $\theta$ belongs to interval $-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$, meaning $\theta$ has values in interval $[-1.571, 1.571]$.

Because floating-point operations are costly, fixed-point fractional numbers are used instead. Multiplication of fixed-point numbers, by $2^{-i}$, is an $i$-bit rightshift.

Thus, for a CORDIC approximation with 15 bits of precision, the following fixed-point format can be used:

$$X = x_{15}x_{14}.x_{13}x_{12}...x_1x_0$$

For the format above, $x_{15}$ is the sign bit, $x_{14}$ is a unitary bit and the remaining bits, from $x_{13}$ to $x_0$, are fractional bits. Conversion of a floating-point value into the proposed fixed-point format is realized by truncating the number to at most 14 fractional positions.
In computer systems, fixed-point data is, typically, of integer type. The 16-bit integer \((x_{15}x_{14}x_{13}x_{12}...x_1x_0.)\) that correspond to a value represented in the proposed format \((x_{15}x_{14}.x_{13}x_{12}...x_1x_0)\) is obtained by multiplying the value in the proposed format by \(2^{14}\) and truncating the result to its integer part.

The CORDIC method operating with 16-bit integer values will have variables \(x(i), y(i)\) and \(z(i)\) represented as 16-bit integers. Moreover, constant \(\frac{1}{K}\), angle \(\theta\) and terms \(\arctan(2^{-i})\) will be converted to the 16-bit integers, as well.

At the completion of the procedure, the 16-bit integer value of \(x(16)\) will be converted back into a floating-point number by dividing it by \(2^{14}\).