# Coordinate Rotation Digit Computer (CORDIC) Algorithm

CORDIC Algorithm เป็นอัลกอลิทึมที่สามารถใช้คำนวณค่าของฟังก์ชันตรีโกณมิติ ไฮเพอร์โบลิก ราก การคูณ การหาร เอกส์โพเนนเชียล และ ล็อกกาลิทึมด้วยฐานที่สามารถกำหนดเองได้ โดยการใช้เพียงแค่ การบวกลบ และการเลื่อนบิตเท่านั้น

## CORDIC Microrotation Equations:

# Equations Proof

A diagram of a triangle

Description automatically generated

รูปที่ 1 Vector Rotation

กำหนดให้เวคเตอร์สองมิติเริ่มต้นมีค่า (X, Y) = (xin, yin) โดยทำมุมกับแกน X เริ่มต้นคือ β โมดูลัสของเวกเตอร์คือ Min เมื่อหมุนเวกเตอร์ดังกล่าวไปด้วยมุมคือ θ ผลลัพธ์ที่ได้จากการหมุนคือ (X, Y) = (xR, yR) ซึ่งทำมุมกับแกน X คือ θ + β

จากรูปที่ 1 จะได้

(1)

(2)

*จากสมการที่* (1) และ (2) เมื่อ*ใช้เอกลักษณ์ตรีโกณ ดังนั้น*จะได้

สามารถเขียนให้อยู่ในรูปแบบของเมตริกซ์ได้ ดังนี้

การหมุนไปของเวกเตอร์ดังกล่าว สามารถเขียนสมการของผลลัพธ์ได้ ดังนี้

(3)

(4)

*จากนั้น แบ่ง θ ออกเป็นมุมย่อย ๆ* j มุม โดยที่ j มีค่าตั้งแต่ 0 ถึง *คือ* αj

จะได้

พิสูจน์ กำหนดให้ j = 0, 1 จะได้

จากเอกลักษณ์ตรีโกณมิติ จะได้

ดังนั้นแล้ว จากสมการที่ (3) และ (4) เมื่อ *θ* ถูกแบ่งเป็นมุม αj  แล้ว จะได้

เมื่อต้องการหลีกเลี่ยงการคูณกัน

(5) (6)

กำหนดให้ เมื่อ

จากสมการที่ (5) และ (6) จะได้

(7) (8)

ดังนั้น

\* เนื่องจาก ดังนั้น

จะได้

กำหนดให้ ซึ่งจะได้

ดังนั้น

*กำหนดให้* z[j] คือมุมที่ยังเหลืออยู่ของการหมุน จะได้ว่า

*สามารถเขียน* linear combination ได้ดังนี้

โดยที่เมื่อเป็น Rotation Mode; , Vectoring Mode;

# CORDIC Block Diagram

A diagram of a program

Description automatically generated

รูปที่ 2 CORDiC Block Diagram

# CORDIC Rotation Mode

Rotation Mode คือ การหมุนเวกเตอร์เริ่มต้น (xin, yin) ด้วยมุม θ

โดยที่กำหนดเงื่อนไขเริ่มต้น ดังนี้

เมื่อทำ iteration ของ Microrotation แล้วจะได้ค่าสุดท้าย ดังนี้

หากต้องการหาค่าของ และ สามารถคำนวณหาได้โดยกำหนดเงื่อนไขเริ่มต้น คือ

แล้วจะได้ค่าสุดท้าย ดังนี้

# Homework 2: Rotation Mode

Rotate (xin, yin) by 67◦ using n = 12 Micro-Rotations

Initial Coordinates: xin = 1, yin = 0.125

## MATLAB implementation

% ======================================================================= %

% This program was built by Sirapop Saengthongkam to study Cordic

% algorithm.

%

% This program can compute CORDIC algorithm with 2 Mode

% Mode = 0: Vectoring Mode --> Input: Xin, Yin, Zin = Angle

% / Output: Xf/K = M, Zf = arctan(Yin/Xin)

% Mode = 1: Rotation Mode --> Input: Xin = 1/K, Yin = 0, Zin = Angle

% / Output: Xf = cos(Angle), Yf = sin(Angle)

% n is Iteration index if n is increase then the accuracy is increase.

% ======================================================================= %

clear; clc; close all;

% Constant

K = 1.6468; % K = sqrt(1+(2^-2n))

% Initial Conditions

Mode = 1; % 0 is Vectoring Mode, 1 is Rotation Mode.

Xin = 1; % Initial Coordinate-x

Yin = 0.125; % Initial Coordinate-y

Zin = 67; % Initial Angle

n = 12; % Iteration index

% Pre-Calculation

Theta = Zin \* pi/180;

X = zeros(n,1);

X(1) = Xin;

Y = zeros(n,1);

Y(1) = Yin;

Z = zeros(n,1);

Z(1) = Theta;

sigma = zeros(n,1);

if (Mode)

if (Z(1) < 0)

sigma(1) = -1;

else

sigma(1) = 1;

end

else

if (Y(1) < 0)

sigma(1) = 1;

else

sigma(1) = -1;

end

end

% CORDIC - Iteration

for j = 1:n

[signX, X(j+1)] = ADD\_SUB(X(j), SHIFTER(Y(j), j-1), sigma(j), 0);

[signY, Y(j+1)] = ADD\_SUB(Y(j), SHIFTER(X(j), j-1), sigma(j), 1);

[signZ, Z(j+1)] = ADD\_SUB(Z(j), arctanLUT(j-1), sigma(j), 0);

sigma(j+1) = MUX2to1(signY, signZ, Mode);

end

% Display Values

j = (0:1:n)';

if (Mode)

T = table(j, Z, sigma, X, Y)

if ((Xin == 1/K)&&(Yin == 0))

fprintf("======================================================\n")

fprintf("\t\t\tXf = cos(%.1f°) = %.4f\n\t\t\t" + ...

"Yf = sin(%.1f°) = %.4f\n", Zin, X(n+1), Zin, Y(n+1))

fprintf("======================================================\n")

end

else

T = table(j, Y, sigma, X, Z)

if (Zin == 0)

fprintf("======================================================\n")

fprintf("\tXf = Modulus = %.4f\n\t" + ...

"Zf = arctan(%.4f/%.4f) = %.4f = %.1f°\n", X(n+1)/K, Yin, ...

Xin, Z(n+1), Z(n+1)\*180/pi)

fprintf("======================================================\n")

end

end

**Results:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| j | z[j] | σj | x[j] | y[j] |
| 0 | 1.1694 | 1 | 1.0000 | 0.1250 |
| 1 | 0.3840 | 1 | 0.8750 | 1.1250 |
| 2 | -0.0797 | -1 | 0.3125 | 1.5625 |
| 3 | 0.1653 | 1 | 0.7031 | 1.4844 |
| 4 | 0.0409 | 1 | 0.5176 | 1.5723 |
| 5 | -0.0215 | -1 | 0.4193 | 1.6046 |
| 6 | 0.0098 | 1 | 0.4695 | 1.5915 |
| 7 | -0.0059 | -1 | 0.4446 | 1.5988 |
| 8 | 0.0020 | 1 | 0.4571 | 1.5954 |
| 9 | -0.0019 | -1 | 0.4508 | 1.5972 |
| 10 | 0.0000 | 1 | 0.4540 | 1.5963 |
| 11 | -0.0010 | -1 | 0.4524 | 1.5967 |
| 12 | -0.0005 | -1 | 0.4532 | 1.5965 |

ผลลัพธ์หลังจาก normalize ด้วย K = 1.6468 ดังตารางด้านล่างนี้

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| j | z[j] | σj | x[j] | y[j] |
| 0 | 1.1694 | 1 | 0.6072 | 0.0759 |
| 1 | 0.3840 | 1 | 0.5313 | 0.6831 |
| 2 | -0.0797 | -1 | 0.1898 | 0.9488 |
| 3 | 0.1653 | 1 | 0.4270 | 0.9014 |
| 4 | 0.0409 | 1 | 0.3143 | 0.9547 |
| 5 | -0.0215 | -1 | 0.2546 | 0.9744 |
| 6 | 0.0098 | 1 | 0.2851 | 0.9664 |
| 7 | -0.0059 | -1 | 0.2700 | 0.9709 |
| 8 | 0.0020 | 1 | 0.2776 | 0.9688 |
| 9 | -0.0019 | -1 | 0.2738 | 0.9699 |
| 10 | 0.0000 | 1 | 0.2757 | 0.9693 |
| 11 | -0.0010 | -1 | 0.2747 | 0.9696 |
| 12 | -0.0005 | -1 | 0.2752 | 0.9695 |

ดังนั้น จะได้ค่าสุดท้าย คือ

# CORDIC Vectoring Mode

Vectoing Mode คือ คือ การหมุนเวกเตอร์เริ่มต้น (xin, yin) จนกระทั่ง y = 0

โดยที่กำหนดเงื่อนไขเริ่มต้น ดังนี้

ทำการบวกทบค่าของมุม z ไปจนกระทั่ง y = 0

เมื่อทำ iteration ของ Microrotation แล้วจะได้ค่าสุดท้าย ดังนี้

# Extra-Homework: Vectoring Mode

Initial Vector (xin = 0.75, yin = 0.43)

y forced to zero in n = 12 Micro-Rotations

## MATLAB implementation

% ======================================================================= %

% This program was built by Sirapop Saengthongkam to study Cordic

% algorithm.

%

% This program can compute CORDIC algorithm with 2 Mode

% Mode = 0: Vectoring Mode --> Input: Xin, Yin, Zin = Angle

% / Output: Xf/K = M, Zf = arctan(Yin/Xin)

% Mode = 1: Rotation Mode --> Input: Xin = 1/K, Yin = 0, Zin = Angle

% / Output: Xf = cos(Angle), Yf = sin(Angle)

% n is Iteration index if n is increase then the accuracy is increase.

% ======================================================================= %

clear; clc; close all;

% Constant

K = 1.6468; % K = sqrt(1+(2^-2n))

% Initial Conditions

Mode = 0; % 0 is Vectoring Mode, 1 is Rotation Mode.

Xin = 0.75; % Initial Coordinate-x

Yin = 0.43; % Initial Coordinate-y

Zin = 0; % Initial Angle

n = 12; % Iteration index

% Pre-Calculation

Theta = Zin \* pi/180;

X = zeros(n,1);

X(1) = Xin;

Y = zeros(n,1);

Y(1) = Yin;

Z = zeros(n,1);

Z(1) = Theta;

sigma = zeros(n,1);

if (Mode)

if (Z(1) < 0)

sigma(1) = -1;

else

sigma(1) = 1;

end

else

if (Y(1) < 0)

sigma(1) = 1;

else

sigma(1) = -1;

end

end

% CORDIC - Iteration

for j = 1:n

[signX, X(j+1)] = ADD\_SUB(X(j), SHIFTER(Y(j), j-1), sigma(j), 0);

[signY, Y(j+1)] = ADD\_SUB(Y(j), SHIFTER(X(j), j-1), sigma(j), 1);

[signZ, Z(j+1)] = ADD\_SUB(Z(j), arctanLUT(j-1), sigma(j), 0);

sigma(j+1) = MUX2to1(signY, signZ, Mode);

end

% Display Values

j = (0:1:n)';

if (Mode)

T = table(j, Z, sigma, X, Y)

if ((Xin == 1/K)&&(Yin == 0))

fprintf("======================================================\n")

fprintf("\t\t\tXf = cos(%.1f°) = %.4f\n\t\t\t" + ...

"Yf = sin(%.1f°) = %.4f\n", Zin, X(n+1), Zin, Y(n+1))

fprintf("======================================================\n")

end

else

T = table(j, Y, sigma, X, Z)

if (Zin == 0)

fprintf("======================================================\n")

fprintf("\tXf = Modulus = %.4f\n\t" + ...

"Zf = arctan(%.4f/%.4f) = %.4f = %.1f°\n", X(n+1)/K, Yin, ...

Xin, Z(n+1), Z(n+1)\*180/pi)

fprintf("======================================================\n")

end

end

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| j | y[j] | σj | x[j] | z[j] |
| 0 | 0.4300 | -1 | 0.7500 | 0.0000 |
| 1 | -0.3200 | 1 | 1.1800 | 0.7854 |
| 2 | 0.2700 | -1 | 1.3400 | 0.3218 |
| 3 | -0.0650 | 1 | 1.4075 | 0.5667 |
| 4 | 0.1109 | -1 | 1.4156 | 0.4424 |
| 5 | 0.0225 | -1 | 1.4226 | 0.5048 |
| 6 | -0.0220 | 1 | 1.4233 | 0.5360 |
| 7 | 0.0002 | -1 | 1.4236 | 0.5204 |
| 8 | -0.0109 | 1 | 1.4236 | 0.5282 |
| 9 | -0.0053 | 1 | 1.4236 | 0.5243 |
| 10 | -0.0025 | 1 | 1.4237 | 0.5224 |
| 11 | -0.0011 | 1 | 1.4237 | 0.5214 |
| 12 | -0.0005 | 1 | 1.4237 | 0.5209 |

**Results:**

ดังนั้น จะได้ค่าสุดท้าย คือ