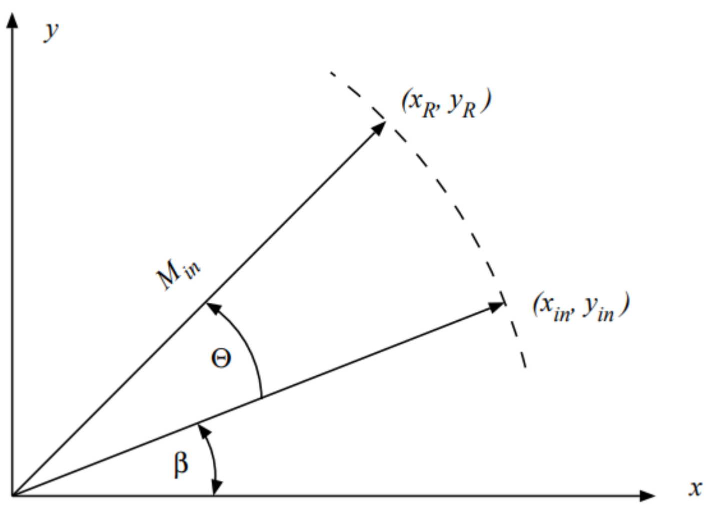
Coordinate Rotation Digit Computer (CORDIC) Algorithm

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

CORDIC Microrotation Equations:

$$x[j+1] = x[j] - \sigma_j 2^{-j} y[j]$$
$$y[j+1] = y[j] + \sigma_j 2^{-j} x[j]$$
$$z[j+1] = z[j] - \sigma_j tan^{-1}(2^{-j})$$

Equations Proof



รูปที่ 1 Vector Rotation

กำหนดให้เวกเตอร์สองมิติเริ่มต้นมีค่า (X, Y) = (x_{in}, y_{in}) โดยทำมุมกับแกน X เริ่มต้นคือ β โมดูลัสของเวกเตอร์คือ M_{in} เมื่อหมุนเวกเตอร์ดังกล่าวไปด้วยมุมคือ θ ผลลัพธ์ที่ได้จากการหมุนคือ (X, Y) = (x_R, y_R) ซึ่งทำมุมกับแกน X คือ θ + β

จากรูปที่ 1 จะได้

$$\cos(\beta) = \frac{x_{in}}{M_{in}} \rightarrow x_{in} = M_{in}\cos(\beta)$$
$$\sin(\beta) = \frac{y_{in}}{M_{in}} \rightarrow y_{in} = M_{in}\sin(\beta)$$

$$x_R = M_{in}\cos(\theta + \beta) \tag{1}$$

$$y_R = M_{in}\sin(\theta + \beta) \tag{2}$$

จากสมการที่ (1) และ (2) เมื่อใช้เอกลักษณ์ตรีโกณ ดังนั้นจะได้

$$\begin{aligned} x_R &= M_{in}(\cos\theta\cos\beta - \sin\theta\sin\beta) \\ &= M_{in}\cos\theta\cos\beta - M_{in}\sin\theta\sin\beta \\ &= x_{in}\cos\theta - y_{in}\sin\theta \end{aligned}$$

$$\begin{aligned} y_R &= M_{in}(\sin\theta\cos\beta + \cos\theta\sin\beta) \\ &= M_{in}\sin\theta\cos\beta + M_{in}\cos\theta\sin\beta \\ &= x_{in}\sin\theta + y_{in}\cos\theta \end{aligned}$$

สามารถเขียนให้อยู่ในรูปแบบของเมตริกซ์ได้ ดังนี้

$$\begin{bmatrix} x_R \\ y_R \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_{in} \\ y_{in} \end{bmatrix} = ROT(\theta) \begin{bmatrix} x_{in} \\ y_{in} \end{bmatrix}$$

การหมุนไปของเวกเตอร์ดังกล่าว สามารถเขียนสมการของผลลัพธ์ได้ ดังนี้

$$x_R = M_{in}\cos(\beta + \theta) = x_{in}\cos\theta - y_{in}\sin\theta \tag{3}$$

$$y_R = M_{in}\sin(\beta + \theta) = x_{in}\sin\theta + y_{in}\cos\theta \tag{4}$$

จากนั้น แบ่ง θ ออกเป็นมุมย่อย ๆ j มุม โดยที่ j มีค่าตั้งแต่ 0 ถึง ∞ คือ α_j

จะได้

$$\theta = \sum_{j=0}^{\infty} \alpha_j$$
$$\boldsymbol{ROT}(\theta) = \prod_{j=0}^{\infty} \boldsymbol{ROT}(\alpha_j)$$

พิสูจน์ กำหนดให้ j = 0, 1 จะได้

$$\prod_{j=0}^{1} \boldsymbol{ROT}(\alpha_j) = \boldsymbol{ROT}(\alpha_0)\boldsymbol{ROT}(\alpha_1) = \begin{bmatrix} \cos(\alpha_0) & -\sin(\alpha_0) \\ \sin(\alpha_0) & \cos(\alpha_0) \end{bmatrix} \begin{bmatrix} \cos(\alpha_1) & -\sin(\alpha_1) \\ \sin(\alpha_1) & \cos(\alpha_1) \end{bmatrix}$$

$$= \begin{bmatrix} \cos\alpha_0 \cos\alpha_1 - \sin\alpha_0 \sin\alpha_1 & -(\sin\alpha_0 \cos\alpha_1 + \cos\alpha_0 \sin\alpha_1) \\ \sin\alpha_0 \cos\alpha_1 + \cos\alpha_0 \sin\alpha_1 & \cos\alpha_0 \cos\alpha_1 - \sin\alpha_0 \sin\alpha_1 \end{bmatrix}$$

จากเอกลักษณ์ตรีโกณมิติ จะได้

$$\prod_{j=0}^{1} \boldsymbol{ROT}(\alpha_j) = \boldsymbol{ROT}\left(\sum_{j=0}^{1} \alpha_j\right) = \begin{bmatrix} \cos(\alpha_0 + \alpha_1) & -\sin(\alpha_0 + \alpha_1) \\ \sin(\alpha_0 + \alpha_1) & \cos(\alpha_0 + \alpha_1) \end{bmatrix}$$

$$\prod_{j=0}^{\infty} \boldsymbol{ROT}(\alpha_j) = \boldsymbol{ROT}\left(\sum_{j=0}^{\infty} \alpha_j\right) = \begin{bmatrix} \cos\left(\sum_{j=0}^{\infty} \alpha_j\right) & -\sin\left(\sum_{j=0}^{\infty} \alpha_j\right) \\ \sin\left(\sum_{j=0}^{\infty} \alpha_j\right) & \cos\left(\sum_{j=0}^{\infty} \alpha_j\right) \end{bmatrix}$$

$$\therefore \boldsymbol{ROT}(\theta) = \prod_{j=0}^{\infty} \boldsymbol{ROT}(\alpha_j), \quad \theta = \sum_{j=0}^{\infty} \alpha_j$$

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

$$x_R[j+1] = x_R[j]\cos(\alpha_j) - y_R[j]\sin(\alpha_j)$$
$$y_R[j+1] = x_R[j]\sin(\alpha_j) + y_R[j]\cos(\alpha_j)$$

เมื่อต้องการหลีกเลี่ยงการคูณกัน

$$x_R[j+1] = \cos(\alpha_j)\,(x_R[j] - y_R[j]\tan(\alpha_j)) \quad (5)$$

$$y_R[j+1] = \cos(\alpha_j)\,(y_R[j] + x_R[j]\tan(\alpha_j)) \quad (6)$$

กำหนดให้ $\tan\alpha_j = \sigma_j(2^{-j}) \rightarrow \alpha_j = tan^{-1}\left(\sigma_j(2^{-j})\right) = \sigma_j tan^{-1}(2^{-j})$ เมื่อ $\sigma_j \epsilon \{-1, 1\}$

จากสมการที่ (5) และ (6) จะได้

$$x_R[j+1] = \cos(\alpha_j)\,(x_R[j] - \sigma_j(2^{-j})y_R[j]) \tag{7}$$

$$y_R[j+1] = \cos(\alpha_j)\,(y_R[j] + \sigma_j(2^{-j})x_R[j]) \tag{8}$$

ดังนั้น

$$\begin{bmatrix} x_R \\ y_R \end{bmatrix} = \prod_{j=1}^{\infty} \boldsymbol{ROT}(\alpha_j) \begin{bmatrix} x_{in} \\ y_{in} \end{bmatrix} = \prod_{j=1}^{\infty} \begin{bmatrix} \cos\alpha_j & -\sin\alpha_j \\ \sin\alpha_j & \cos\alpha_j \end{bmatrix} \begin{bmatrix} x_{in} \\ y_{in} \end{bmatrix}$$

$$= \prod_{j=1}^{\infty} \cos\alpha_j \begin{bmatrix} 1 & -\tan\alpha_j \\ \tan\alpha_j & 1 \end{bmatrix} \begin{bmatrix} x_{in} \\ y_{in} \end{bmatrix}$$

$$= \prod_{j=1}^{\infty} \cos\alpha_j \prod_{j=1}^{\infty} \begin{bmatrix} 1 & -\tan\alpha_j \\ \tan\alpha_j & 1 \end{bmatrix} \begin{bmatrix} x_{in} \\ y_{in} \end{bmatrix}$$

* เนื่องจาก $\tan\alpha_j = \frac{\sigma_j(2^{-j})}{1}$ ดังนั้น $\cos(\alpha_j) = \frac{1}{\sqrt{1+\sigma_j{}^2(2^{-2j})}} = \frac{1}{\sqrt{1+2^{-2j}}} = (1+2^{-2j})^{-1/2}$

จะได้

$$\begin{bmatrix} x_R \\ y_R \end{bmatrix} = \prod_{j=1}^{\infty} (1+2^{-2j})^{-1/2} \prod_{j=1}^{\infty} \begin{bmatrix} 1 & -\sigma_j(2^{-j}) \\ \sigma_j(2^{-j}) & 1 \end{bmatrix} \begin{bmatrix} x_{in} \\ y_{in} \end{bmatrix}$$

กำหนดให้ $K = \prod_{j=1}^{\infty}(1+2^{-2j})^{1/2}$ ซึ่งจะได้ $K \approx 1.6468$

ดังนั้น

$$\begin{bmatrix} x_R \\ y_R \end{bmatrix} = \frac{1}{K} \prod_{j=1}^{\infty} \begin{bmatrix} 1 & -\sigma_j(2^{-j}) \\ \sigma_j(2^{-j}) & 1 \end{bmatrix} \begin{bmatrix} x_{in} \\ y_{in} \end{bmatrix}$$

$$\begin{bmatrix} x_R \\ y_R \end{bmatrix} = \frac{1}{K} \prod_{j=1}^{\infty} \begin{bmatrix} 1 & -\sigma_j(2^{-j}) \\ \sigma_j(2^{-j}) & 1 \end{bmatrix} \begin{bmatrix} x_{in} \\ y_{in} \end{bmatrix}$$

กำหนดให้ z[j] คือมุมที่ยังเหลืออยู่ของการหมุน จะได้ว่า

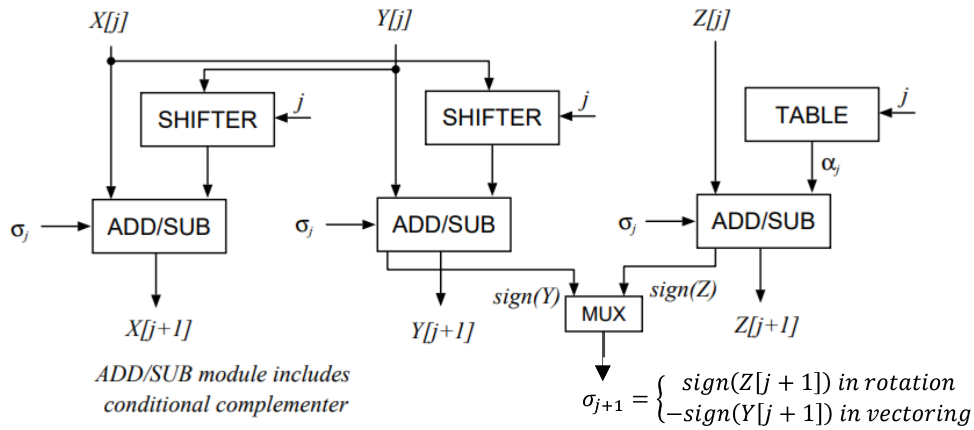
$$z[j+1] = z[j] - \sigma_j tan^{-1}(2^{-j})$$

สามารถเขียน linear combination ได้ดังนี้

$$x[j+1] = x[j] - \sigma_j 2^{-j} y[j]$$
$$y[j+1] = y[j] + \sigma_j 2^{-j} x[j]$$
$$z[j+1] = z[j] - \sigma_j tan^{-1}(2^{-j})$$

โดยที่เมื่อเป็น Rotation Mode; $\sigma_j = \begin{cases} 1 & ,z[j] \geq 0 \\ -1 & ,z[j] < 0 \end{cases}$, Vectoring Mode; $\sigma_j = \begin{cases} 1 & ,y[j] < 0 \\ -1 & ,y[j] \geq 0 \end{cases}$

CORDIC Block Diagram



รูปที่ 2 CORDiC Block Diagram

CORDIC Rotation Mode

Rotation Mode คือ การหมุนเวกเตอร์เริ่มต้น (x_{in}, y_{in}) ด้วยมุม θ

โดยที่กำหนดเงื่อนไขเริ่มต้น ดังนี้

$$z[0] = \theta \quad x[0] = x_{in} \quad y[0] = y_{in}$$

$$\sigma_j = \begin{cases} 1 & , z[j] \geq 0 \\ -1 & , z[j] < 0 \end{cases}$$

เมื่อทำ iteration ของ Microrotation แล้วจะได้ค่าสุดท้าย ดังนี้

$$x_f = K(x_{in}\cos\theta - y_{in}\sin\theta)$$
$$y_f = K(x_{in}\sin\theta + y_{in}\cos\theta)$$
$$z_f = 0$$

หากต้องการหาค่าของ $\cos\theta$ และ $sin\,\theta$ สามารถคำนวณหาได้โดยกำหนดเงื่อนไขเริ่มต้น คือ

$$z[0] = \theta \quad x[0] = \frac{1}{K} \quad y[0] = 0$$

แล้วจะได้ค่าสุดท้าย ดังนี้

$$x_f = \cos\theta$$
$$y_f = \sin\theta$$
$$z_f = 0$$

Homework 2: Rotation Mode

Rotate (x_{in}, y_{in}) by 67˚ using n = 12 Micro-Rotations

Initial Coordinates: x_{in} = 1, y_{in} = 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
```

```
% 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]	σ_j	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]	σ_j	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

$$x_f = 0.2752$$
$$y_f = 0.9695$$
$$z_f = 0.00$$

CORDIC Vectoring Mode

Vectoing Mode คือ คือ การหมุนเวกเตอร์เริ่มต้น (x_{in}, y_{in}) จนกระทั่ง y = 0

โดยที่กำหนดเงื่อนไขเริ่มต้น ดังนี้

$$z[0] = z_{in} \quad x[0] = x_{in} \quad y[0] = y_{in}$$

$$\sigma_j = \begin{cases} 1 & , y[j] < 0 \\ -1 & , y[j] \geq 0 \end{cases}$$

ทำการบวกทบค่าของมุม z ไปจนกระทั่ง y = 0

เมื่อทำ iteration ของ Microrotation แล้วจะได้ค่าสุดท้าย ดังนี้

$$x_f = K(x_{in}^2 + y_{in}^2)^{1/2}$$
$$y_f = 0$$
$$z_f = z_{in} + tan^{-1}(\frac{y_{in}}{x_{in}})$$

Extra-Homework: Vectoring Mode

Initial Vector ($x_{in} = 0.75$, $y_{in} = 0.43$)

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
```

Results:

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

ดังนั้น จะได้ค่าสุดท้าย คือ

$$x_f = 1.4237 \rightarrow Modulus = \frac{x_f}{K} = \sqrt{x_{in}^2 + y_{in}^2} = \frac{1.4237}{1.6468} = 0.8645$$

$$y_f = 0.00$$

$$z_f = tan^{-1}\left(\frac{y_{in}}{x_{in}}\right) = 0.5209 = 29.8°$$