

CORDIC

Coordinate Rotation Digital Computer

Sistemas Digitales (66.17)



Introducción

- Fue descrito por primera vez en 1959 por Jack E. Volder
- Desarrollado en el departamento de aeroelectrónica de Convair
- Más tarde John Stephen Walther, en Hewlett-Packard, generalizó el algoritmo
- Originalmente fue implementado usando el sistema binario. En los 70's su variante decimal se empezó a usar fuertemente en calculadoras de mano

Introducción

- Es utilizado sobre todo en dispositivos en los que no hay disponibilidad de multiplicadores, por ejemplo en microcontroladores y FPGA's pequeñas
- Sólo utiliza las operaciones de suma, resta y desplazamiento
- Se lo utiliza para la rotación de vectores, el cálculo de funciones trigonométricas (sen, cos, tan, etc), la transformación de coordenadas, etc

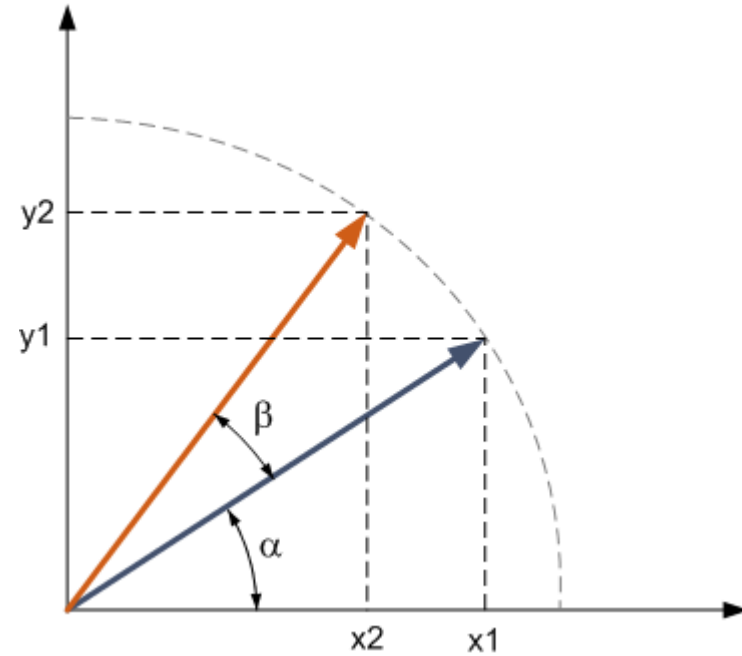
Ecuaciones

①

$$\begin{aligned}x_1 &= A.\cos(\alpha) \\ y_1 &= A.\sen(\alpha)\end{aligned}$$

②

$$\begin{aligned}x_2 &= A.\cos(\alpha + \beta) \\ y_2 &= A.\sen(\alpha + \beta)\end{aligned}$$

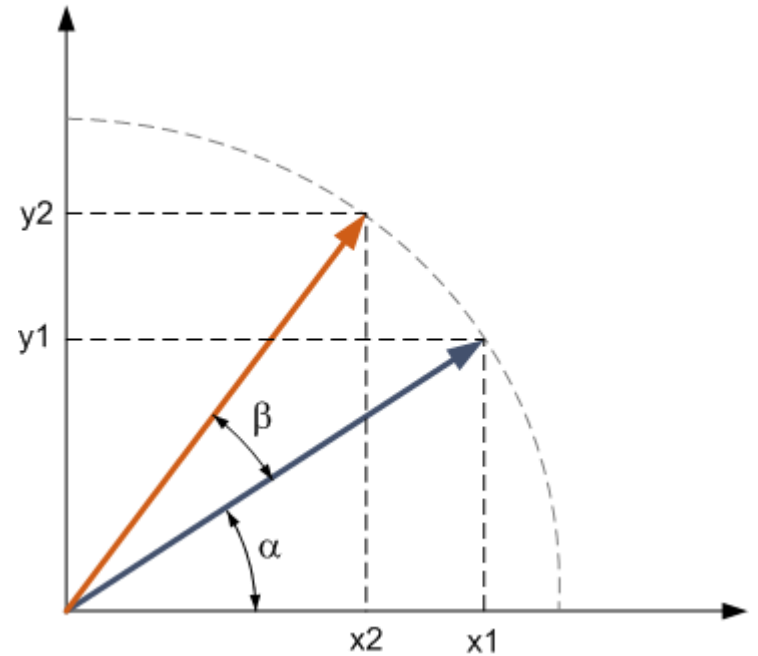


Ecuaciones

Desarrollando el coseno y el seno de la ecuación 2:

3

$$\begin{aligned}x_2 &= A.\cos(\alpha).\cos(\beta) - A.\sen(\alpha).\sen(\beta) \\y_2 &= A.\sen(\alpha).\cos(\beta) + A.\cos(\alpha).\sen(\beta)\end{aligned}$$



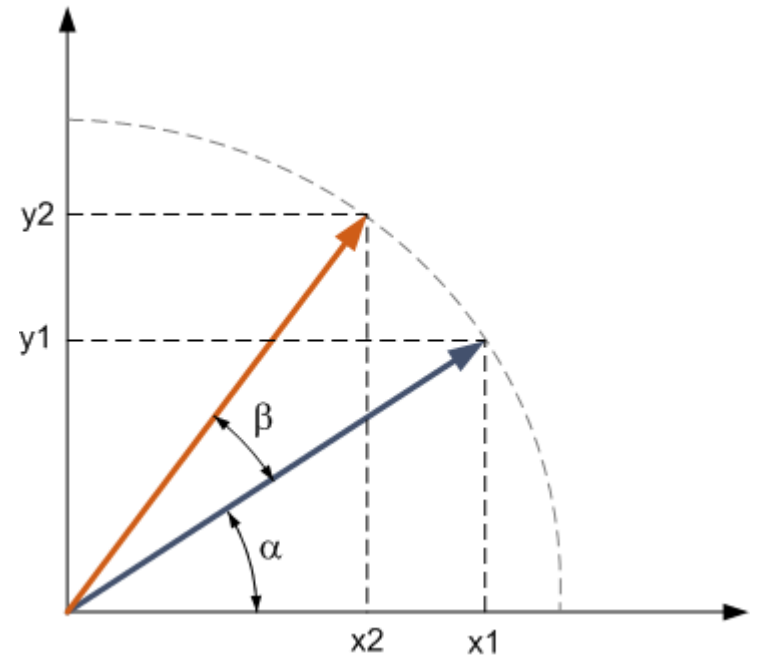
Ecuaciones

Sacando $\cos(\beta)$ como factor común en la ecuación 3:

4

$$x_2 = \cos(\beta) \cdot (A \cdot \cos(\alpha) - A \cdot \operatorname{sen}(\alpha) \cdot \operatorname{tg}(\beta))$$

$$y_2 = \cos(\beta) \cdot (A \cdot \operatorname{sen}(\alpha) + A \cdot \cos(\alpha) \cdot \operatorname{tg}(\beta))$$



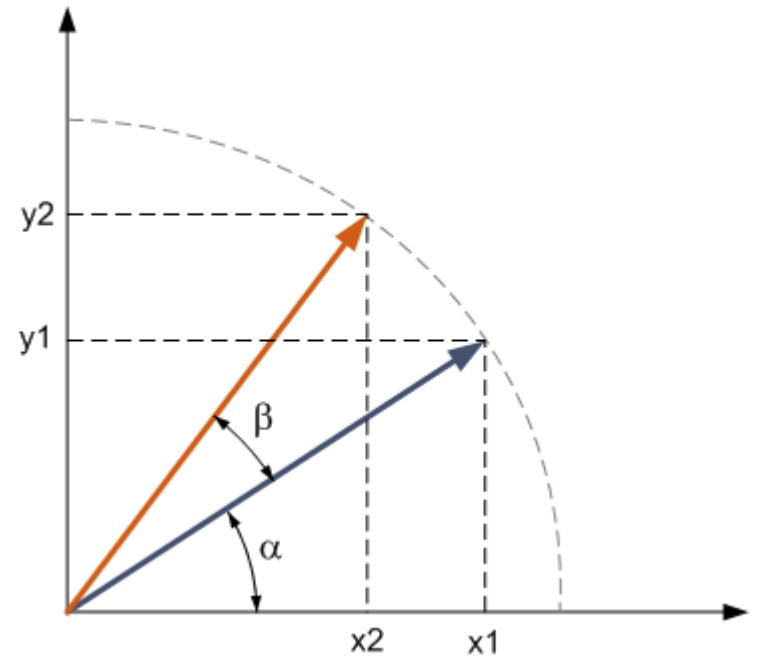
Ecuaciones

Sustituyendo la ecuación 1 en 4:

④

$$x_2 = \cos(\beta) \cdot (A \cdot \cos(\alpha) - A \cdot \sin(\alpha) \cdot \tan(\beta))$$

$$y_2 = \cos(\beta) \cdot (A \cdot \sin(\alpha) + A \cdot \cos(\alpha) \cdot \tan(\beta))$$



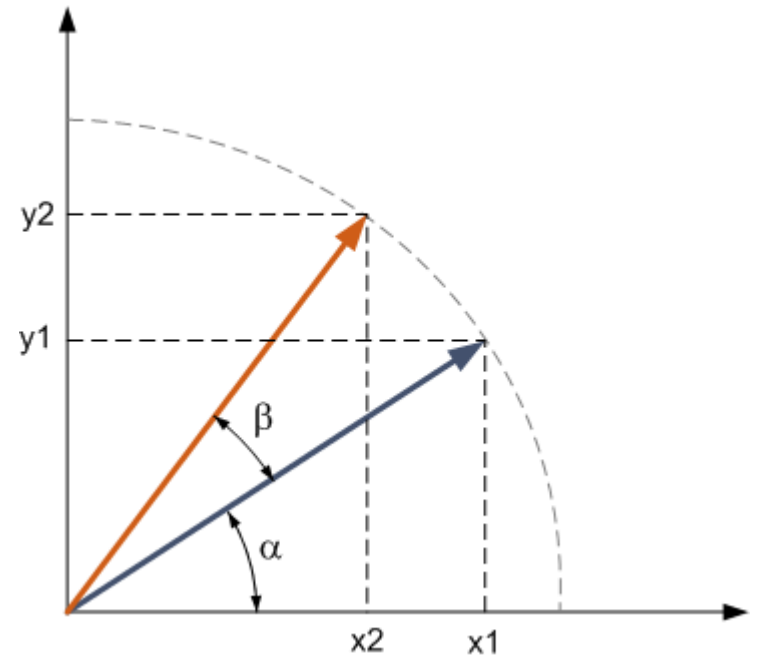
Ecuaciones

Sustituyendo la ecuación 1 en 4:

④

$$x_2 = \cos(\beta) \cdot (A \cdot \cos(\alpha) - A \cdot \sin(\alpha) \cdot \tan(\beta))$$

$$y_2 = \cos(\beta) \cdot (A \cdot \sin(\alpha) + A \cdot \cos(\alpha) \cdot \tan(\beta))$$



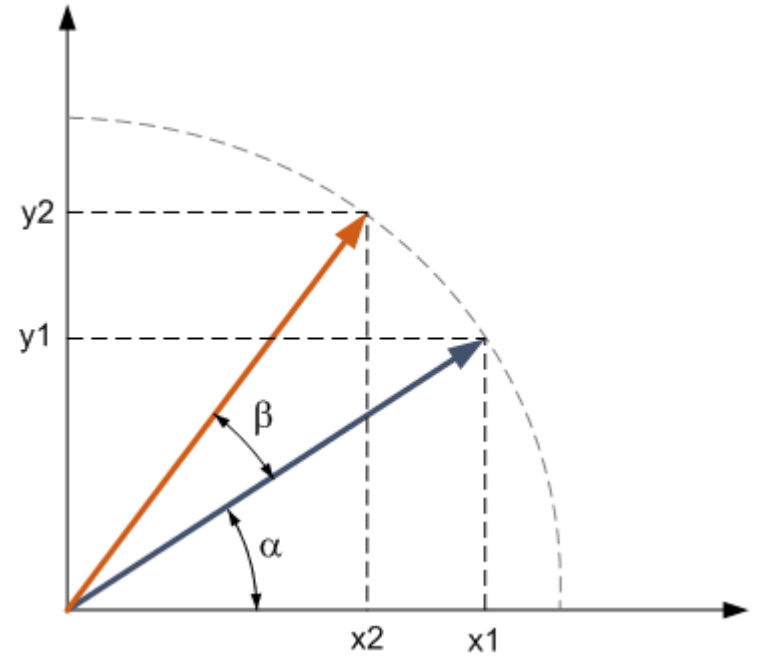
Ecuaciones

Sustituyendo la ecuación 1 en 4:

$$x_2 = \cos(\beta) \cdot (x_1 - \operatorname{tg}(\beta) \cdot y_1)$$

$$y_2 = \cos(\beta) \cdot (y_1 + \operatorname{tg}(\beta) \cdot x_1)$$

Givens



Ecuaciones

Hasta este momento no hemos llegado a nada interesante. Ahora, pensemos por un momento qué pasaría si los ángulos en los que puede rotar el vector se restringen de tal manera que:

$$\operatorname{tg}(\beta) = \pm 2^{-i}$$

¿Se lograría algún beneficio con esto?

¡La multiplicación en el término que incluye la tangente se reduce a una simple operación de desplazamiento!

Ecuaciones

Por lo dicho:

$$\begin{aligned} \operatorname{tg}(\beta) = \pm 2^{-i} &\Rightarrow \beta = \operatorname{tg}^{-1}(\pm 2^{-i}) \Rightarrow \cos(\beta) = \cos(\operatorname{tg}^{-1}(\pm 2^{-i})) \\ &\Rightarrow \cos(\beta) = \cos(\operatorname{tg}^{-1}(2^{-i})) \end{aligned}$$

Teniendo en cuenta que: $\cos(\operatorname{tg}^{-1}(x)) = \frac{1}{\sqrt{1+x^2}}$

$$\Rightarrow \cos(\beta) = \cos(\operatorname{tg}^{-1}(2^{-i})) = K_i = \frac{1}{\sqrt{1+2^{-2i}}}$$

Luego, reemplazando en las ecuaciones de Givens:

$$x_{i+1} = K_i \cdot (x_i - y_i \cdot d_i \cdot 2^{-i})$$

$$y_{i+1} = K_i \cdot (y_i + x_i \cdot d_i \cdot 2^{-i})$$

Donde: $d_i = \pm 1$

Ecuaciones

Si retiramos la constante de las ecuaciones se tiene un algoritmo que tiene sólo operaciones de desplazamiento y suma

$$\begin{aligned}x_{i+1} &= x_i - y_i \cdot d_i \cdot 2^{-i} \\ y_{i+1} &= y_i + x_i \cdot d_i \cdot 2^{-i}\end{aligned}$$

De esta manera el algoritmo sufre una ganancia que está dada por la productoria de la inversa de cada uno de los K_i 's

$$A_n = \prod_n \sqrt{1 + 2^{-2i}}$$

La ganancia depende de la cantidad de iteraciones. Cuando se tiende a infinito se aproxima a 1,647


Ecuaciones (Ganancia del CORDIC)

$$A_n = \prod_n \sqrt{1 + 2^{-2i}}$$

i	$x=2^{-2i}$	Raiz(1+x)	A_n	i	$x=2^{-2i}$	Raiz(1+x)	A_n
0	1	1.4142135623731000	1.414213562	14	3.7253E-09	1.0000000018626400	1.64676025709862000000
1	0.25	1.1180339887498900	1.58113883	15	9.3132E-10	1.0000000004656600	1.64676025786545000000
2	0.0625	1.0307764064044200	1.629800601	16	2.3283E-10	1.0000000001164200	1.64676025805716000000
3	0.015625	1.0077822185373200	1.642484066	17	5.8208E-11	1.0000000000291000	1.64676025810509000000
4	0.00390625	1.0019512213675900	1.645688916	18	1.4552E-11	1.0000000000072800	1.64676025811707000000
5	0.00097656	1.0004881620988800	1.646492279	19	3.638E-12	1.0000000000018200	1.64676025812007000000
6	0.00024414	1.0001220628628300	1.646693254	20	9.0949E-13	1.0000000000004500	1.64676025812082000000
7	6.1035E-05	1.0000305171124800	1.646743507	21	2.2737E-13	1.0000000000001100	1.64676025812100000000
8	1.5259E-05	1.0000076293654300	1.64675607	22	5.6843E-14	1.0000000000000300	1.64676025812105000000
9	3.8147E-06	1.0000019073468100	1.646759211	23	1.4211E-14	1.0000000000000100	1.64676025812106000000
10	9.5367E-07	1.0000004768370400	1.646759996	24	3.5527E-15	1.0000000000000000	1.64676025812106000000
11	2.3842E-07	1.0000001192092800	1.64676019268469000000	25	8.8818E-16	1.0000000000000000	1.64676025812107000000
12	5.9605E-08	1.0000000298023200	1.64676024176197000000	26	2.2204E-16	1.0000000000000000	1.64676025812107000000
13	1.4901E-08	1.0000000074505800	1.64676025403129000000	27	5.5511E-17	1.0000000000000000	1.64676025812107000000

Ecuaciones

El algoritmo incluye una tercera ecuación que describe el acumulador angular:

$$z_{i+1} = z_i - d_i \cdot \text{tg}^{-1}(2^{-i})$$


i	radianes	grados
0	0.78539816	45
1	0.46364761	26.5650512
2	0.24497866	14.0362435
3	0.12435499	7.12501635
4	0.06241881	3.57633437
5	0.03123983	1.78991061
6	0.01562373	0.89517371
7	0.00781234	0.44761417
8	0.00390623	0.2238105
9	0.00195312	0.11190568
10	0.00097656	0.05595289
11	0.00048828	0.02797645
12	0.00024414	0.01398823
13	0.00012207	0.00699411
14	6.1035E-05	0.00349706
15	3.0518E-05	0.00174853

Modos de operación

El algoritmo puede operar en dos modos diferentes:

- **Modo rotación**

- Rota un vector en un ángulo especificado
- El acumulador angular se inicializa con el ángulo a rotar
- La decisión de rotación en cada iteración se lleva a cabo de tal manera de disminuir el ángulo residual en el acumulador angular (se utiliza su signo)

- **Modo vector**

- Rota un vector hacia el eje de coordenadas x, guardando los ángulos requeridos para lograrlo
- Busca minimizar la componente y del vector residual
- La dirección de rotación se decide por el signo de la componente y residual

Modos de operación

- **Modo rotación**

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$

$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$

$$z_{i+1} = z_i - d_i \cdot \text{tg}^{-1}(2^{-i})$$

Donde: $d_i = -1$ si $z_i < 0$, en otro caso $+1$

Finalmente se tiene:

$$x_n = A_n(x_0 \cdot \cos(z_0) - y_0 \cdot \text{sen}(z_0))$$

$$y_n = A_n(y_0 \cdot \cos(z_0) + x_0 \cdot \text{sen}(z_0))$$

$$z_n = 0$$

Modos de operación

- **Modo vector**

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$

$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$

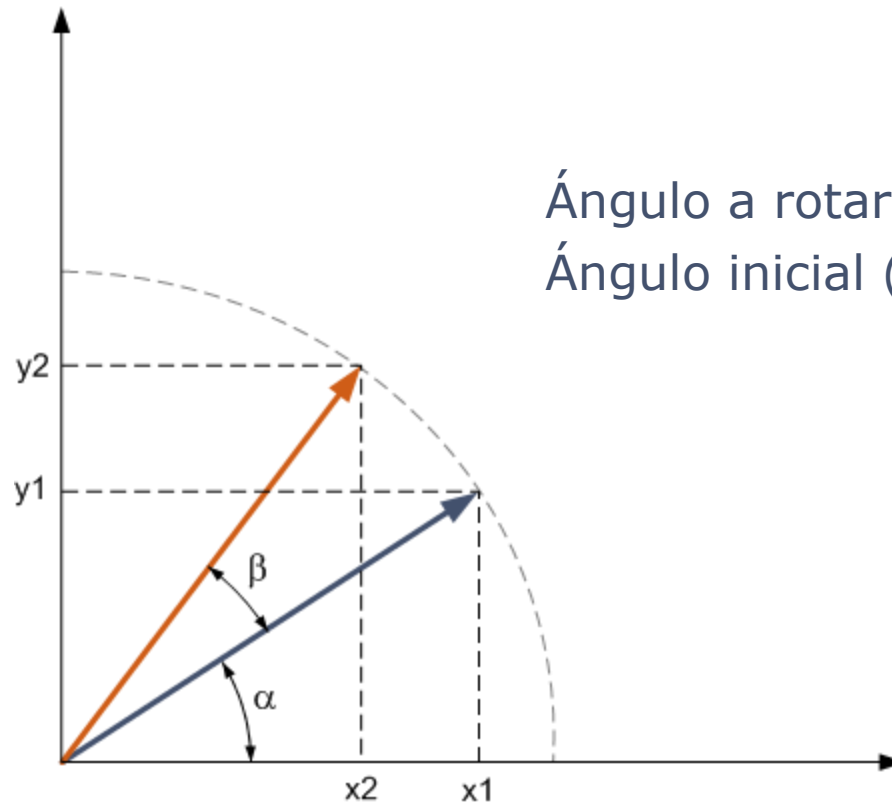
$$z_{i+1} = z_i - d_i \cdot \text{tg}^{-1}(2^{-i})$$

Donde: $d_i = +1$ si $y_i < 0$, en otro caso -1

Finalmente se tiene:

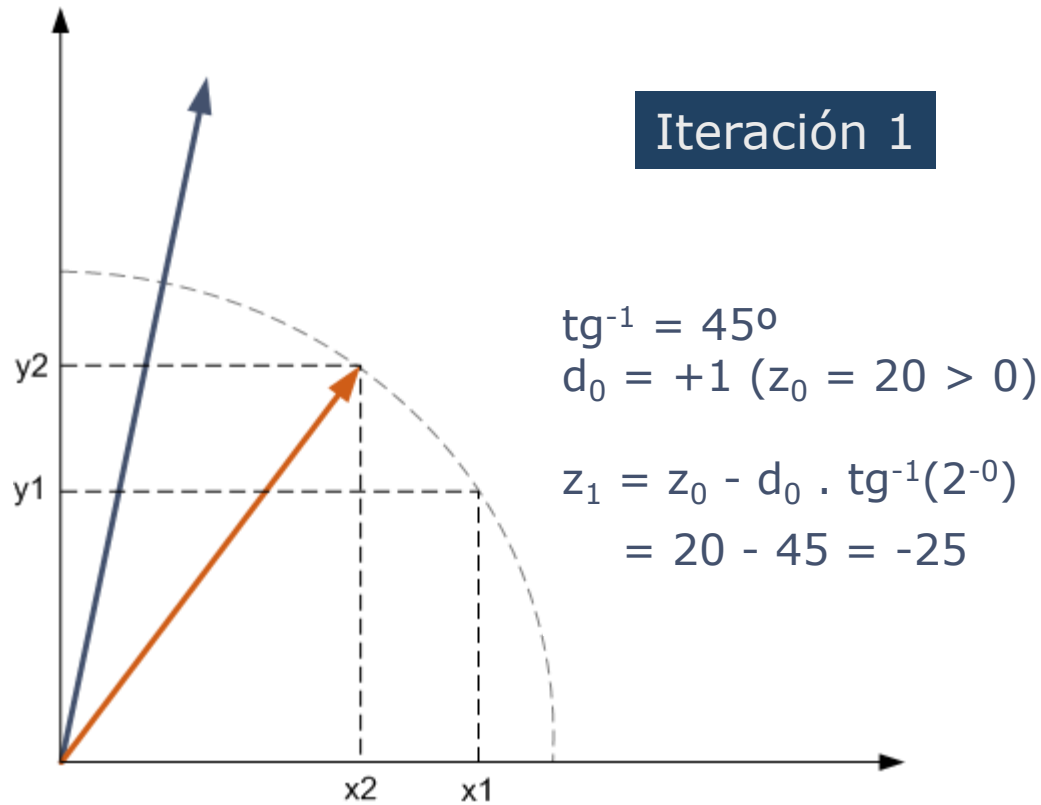
$$x_n = A_n \cdot \sqrt{x_0^2 + y_0^2}$$
$$y_n = 0$$
$$z_n = z_0 + \text{tg}^{-1} \left(\frac{y_0}{x_0} \right)$$

Ejemplo



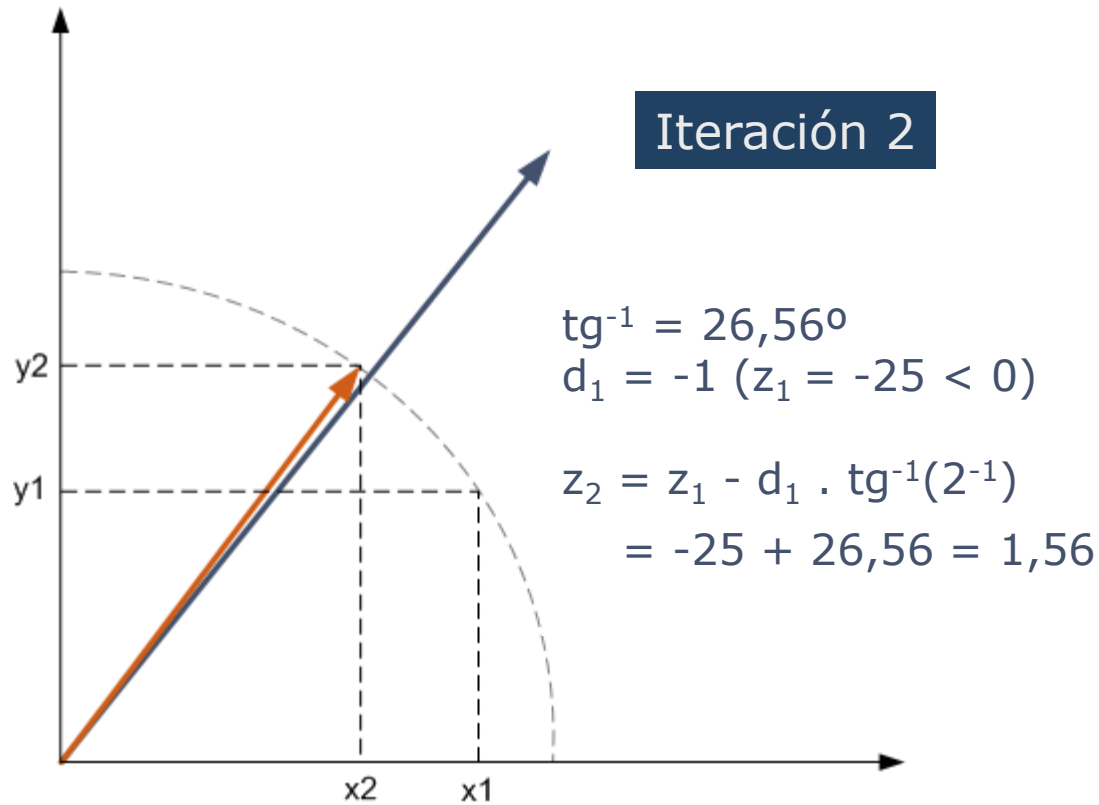
Ángulo a rotar (β): 20°
Ángulo inicial (α): 33°

Ejemplo



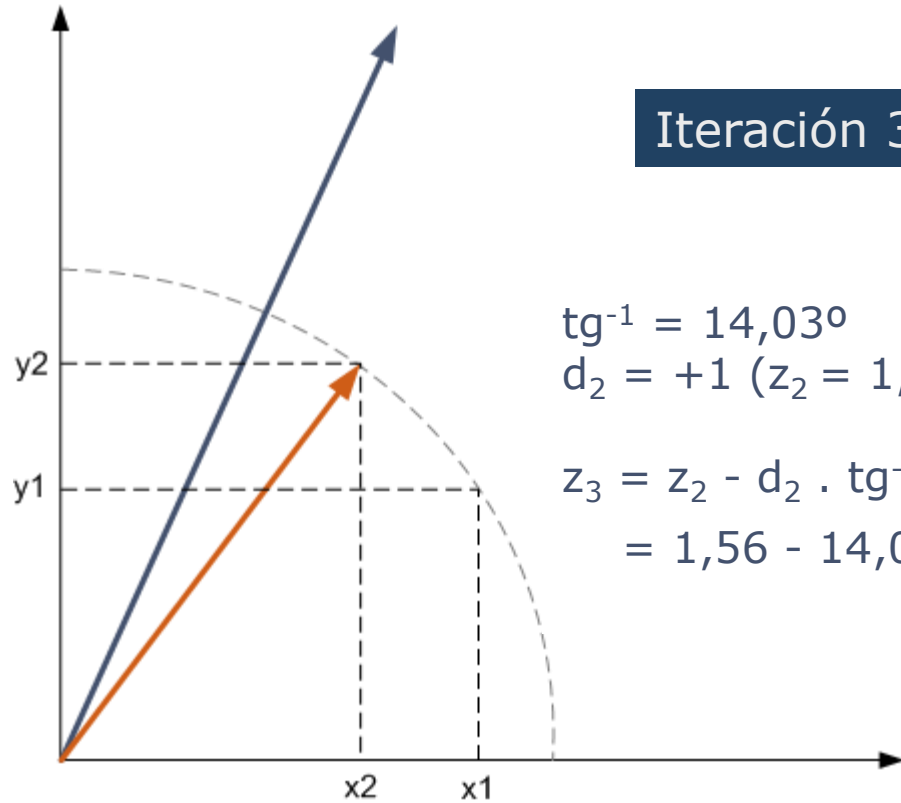
i	radianes	grados
0	0.78539816	45
1	0.46364761	26.5650512
2	0.24497866	14.0362435
3	0.12435499	7.12501635
4	0.06241881	3.57633437
5	0.03123983	1.78991061
6	0.01562373	0.89517371
7	0.00781234	0.44761417
8	0.00390623	0.2238105
9	0.00195312	0.11190568
10	0.00097656	0.05595289
11	0.00048828	0.02797645
12	0.00024414	0.01398823
13	0.00012207	0.00699411
14	6.1035E-05	0.00349706
15	3.0518E-05	0.00174853

Ejemplo



i	radianes	grados
0	0.78539816	45
1	0.46364761	26.5650512
2	0.24497866	14.0362435
3	0.12435499	7.12501635
4	0.06241881	3.57633437
5	0.03123983	1.78991061
6	0.01562373	0.89517371
7	0.00781234	0.44761417
8	0.00390623	0.2238105
9	0.00195312	0.11190568
10	0.00097656	0.05595289
11	0.00048828	0.02797645
12	0.00024414	0.01398823
13	0.00012207	0.00699411
14	6.1035E-05	0.00349706
15	3.0518E-05	0.00174853

Ejemplo



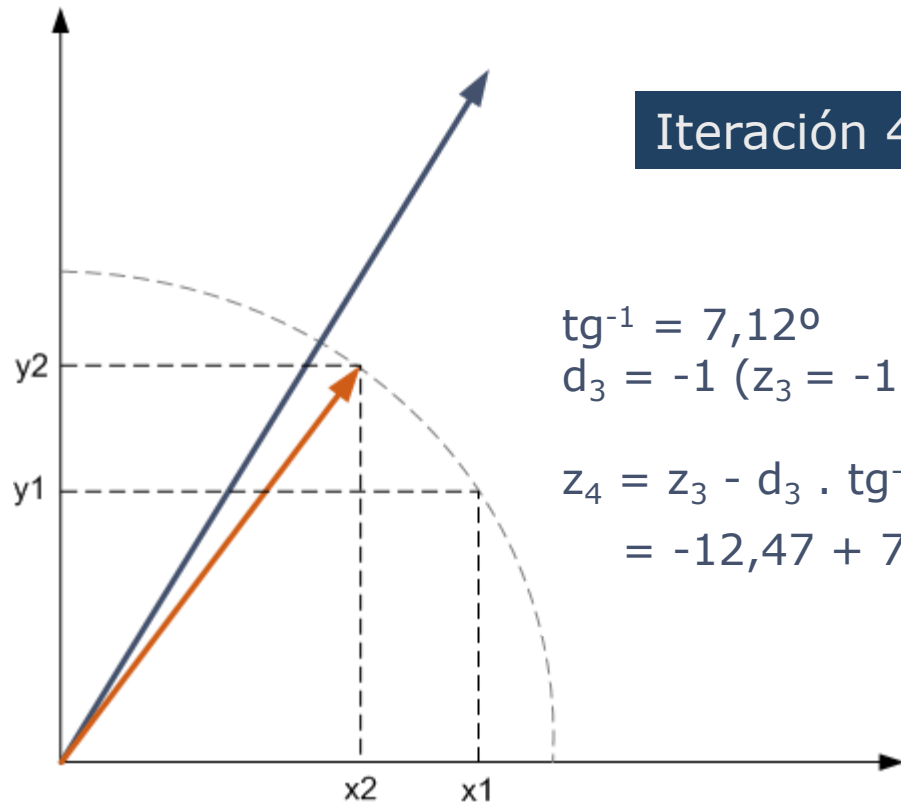
Iteración 3

$$\begin{aligned} \text{tg}^{-1} &= 14,03^\circ \\ d_2 &= +1 \quad (z_2 = 1,56 > 0) \end{aligned}$$

$$\begin{aligned} z_3 &= z_2 - d_2 \cdot \text{tg}^{-1}(2^{-2}) \\ &= 1,56 - 14,03 = -12,47 \end{aligned}$$

i	radianes	grados
0	0.78539816	45
1	0.46364761	26.5650512
2	0.24497866	14.0362435
3	0.12435499	7.12501635
4	0.06241881	3.57633437
5	0.03123983	1.78991061
6	0.01562373	0.89517371
7	0.00781234	0.44761417
8	0.00390623	0.2238105
9	0.00195312	0.11190568
10	0.00097656	0.05595289
11	0.00048828	0.02797645
12	0.00024414	0.01398823
13	0.00012207	0.00699411
14	6.1035E-05	0.00349706
15	3.0518E-05	0.00174853

Ejemplo



Iteración 4

$$\text{tg}^{-1} = 7,12^\circ$$

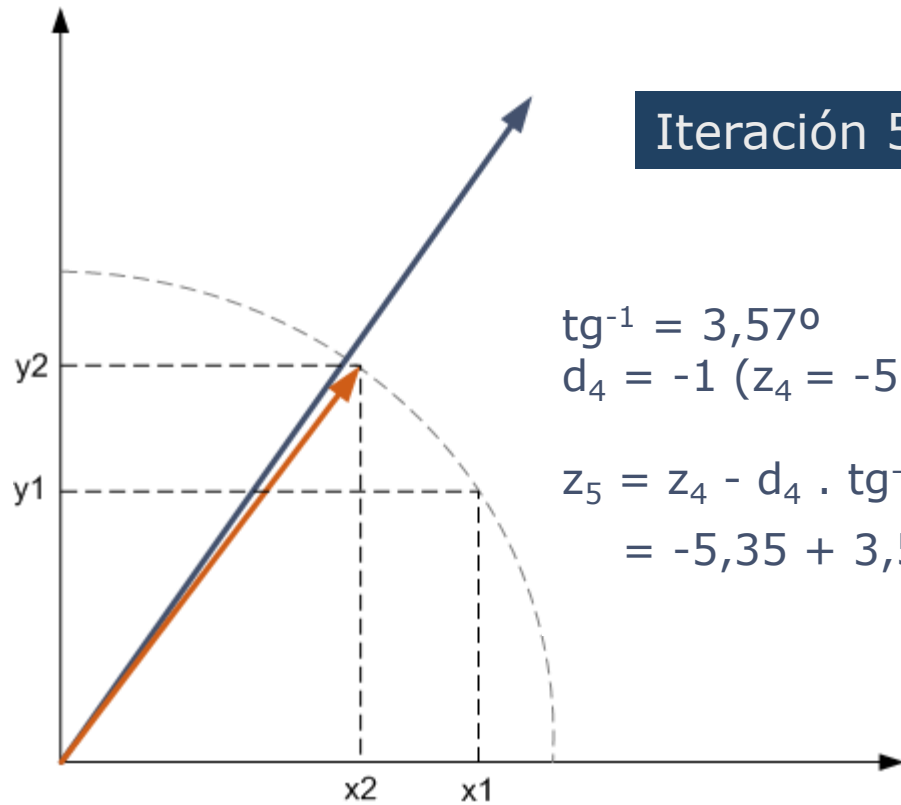
$$d_3 = -1 \quad (z_3 = -12,47 < 0)$$

$$z_4 = z_3 - d_3 \cdot \text{tg}^{-1}(2^{-3})$$

$$= -12,47 + 7,12 = -5,35$$

i	radianes	grados
0	0.78539816	45
1	0.46364761	26.5650512
2	0.24497866	14.0362435
3	0.12435499	7.12501635
4	0.06241881	3.57633437
5	0.03123983	1.78991061
6	0.01562373	0.89517371
7	0.00781234	0.44761417
8	0.00390623	0.2238105
9	0.00195312	0.11190568
10	0.00097656	0.05595289
11	0.00048828	0.02797645
12	0.00024414	0.01398823
13	0.00012207	0.00699411
14	6.1035E-05	0.00349706
15	3.0518E-05	0.00174853

Ejemplo



Iteración 5

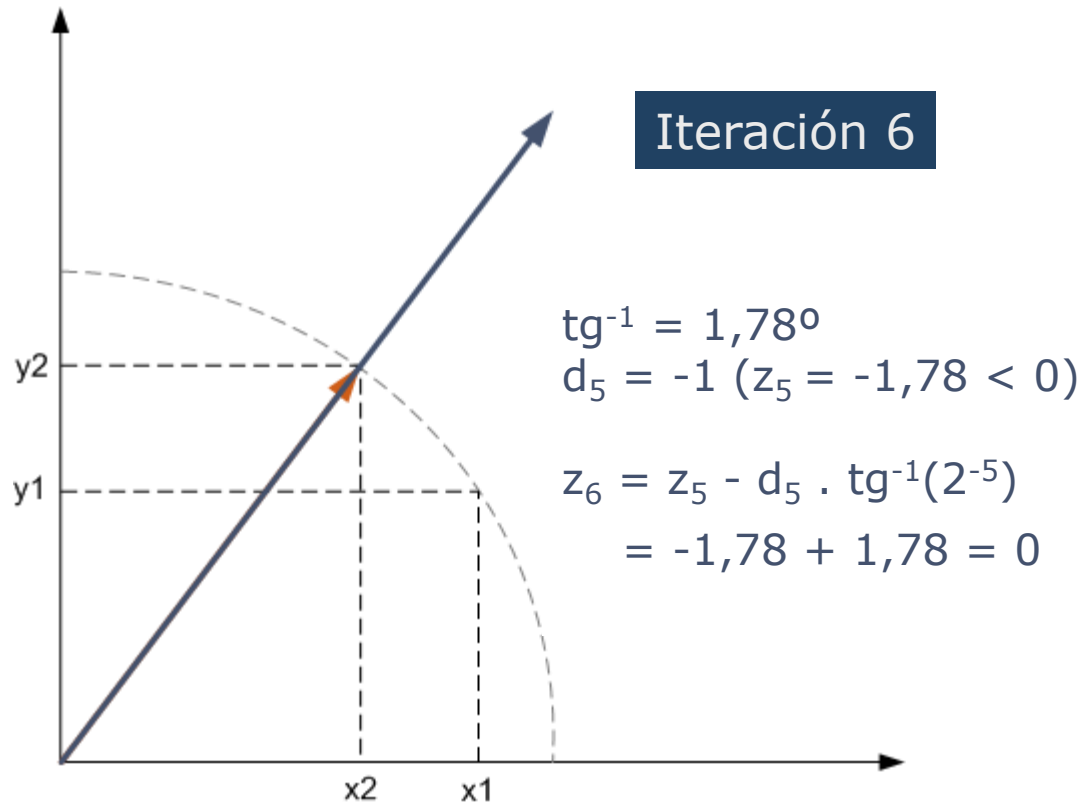
$$\text{tg}^{-1} = 3,57^\circ$$

$$d_4 = -1 \quad (z_4 = -5,35 < 0)$$

$$\begin{aligned} z_5 &= z_4 - d_4 \cdot \text{tg}^{-1}(2^{-4}) \\ &= -5,35 + 3,57 = -1,78 \end{aligned}$$

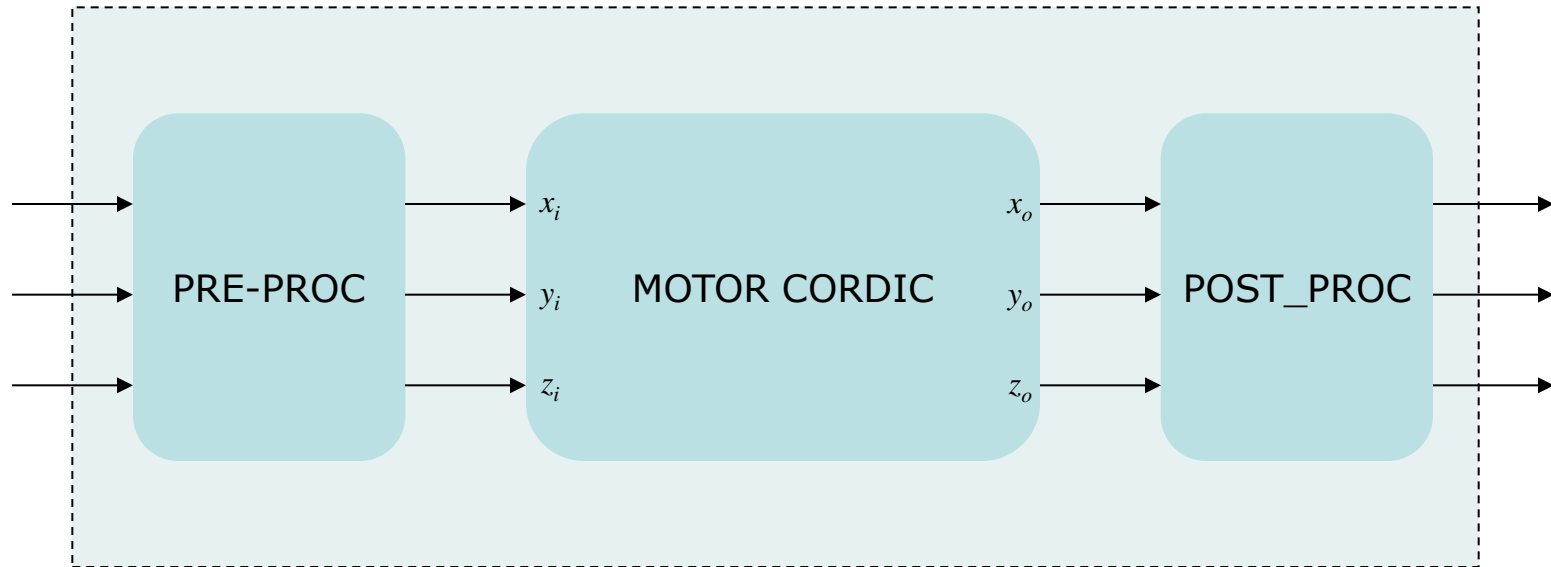
i	radianes	grados
0	0.78539816	45
1	0.46364761	26.5650512
2	0.24497866	14.0362435
3	0.12435499	7.12501635
4	0.06241881	3.57633437
5	0.03123983	1.78991061
6	0.01562373	0.89517371
7	0.00781234	0.44761417
8	0.00390623	0.2238105
9	0.00195312	0.11190568
10	0.00097656	0.05595289
11	0.00048828	0.02797645
12	0.00024414	0.01398823
13	0.00012207	0.00699411
14	6.1035E-05	0.00349706
15	3.0518E-05	0.00174853

Ejemplo



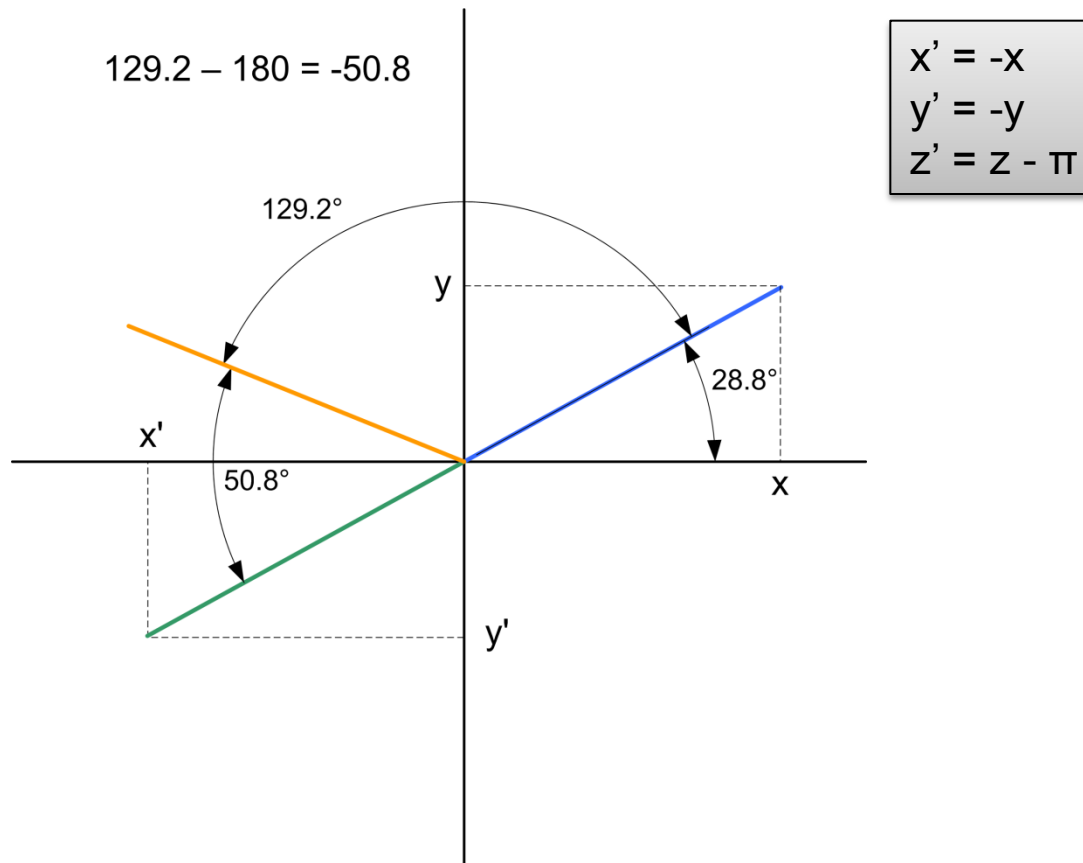
i	radianes	grados
0	0.78539816	45
1	0.46364761	26.5650512
2	0.24497866	14.0362435
3	0.12435499	7.12501635
4	0.06241881	3.57633437
5	0.03123983	1.78991061
6	0.01562373	0.89517371
7	0.00781234	0.44761417
8	0.00390623	0.2238105
9	0.00195312	0.11190568
10	0.00097656	0.05595289
11	0.00048828	0.02797645
12	0.00024414	0.01398823
13	0.00012207	0.00699411
14	6.1035E-05	0.00349706
15	3.0518E-05	0.00174853

Bloques de pre-procesamiento y post_procesamiento



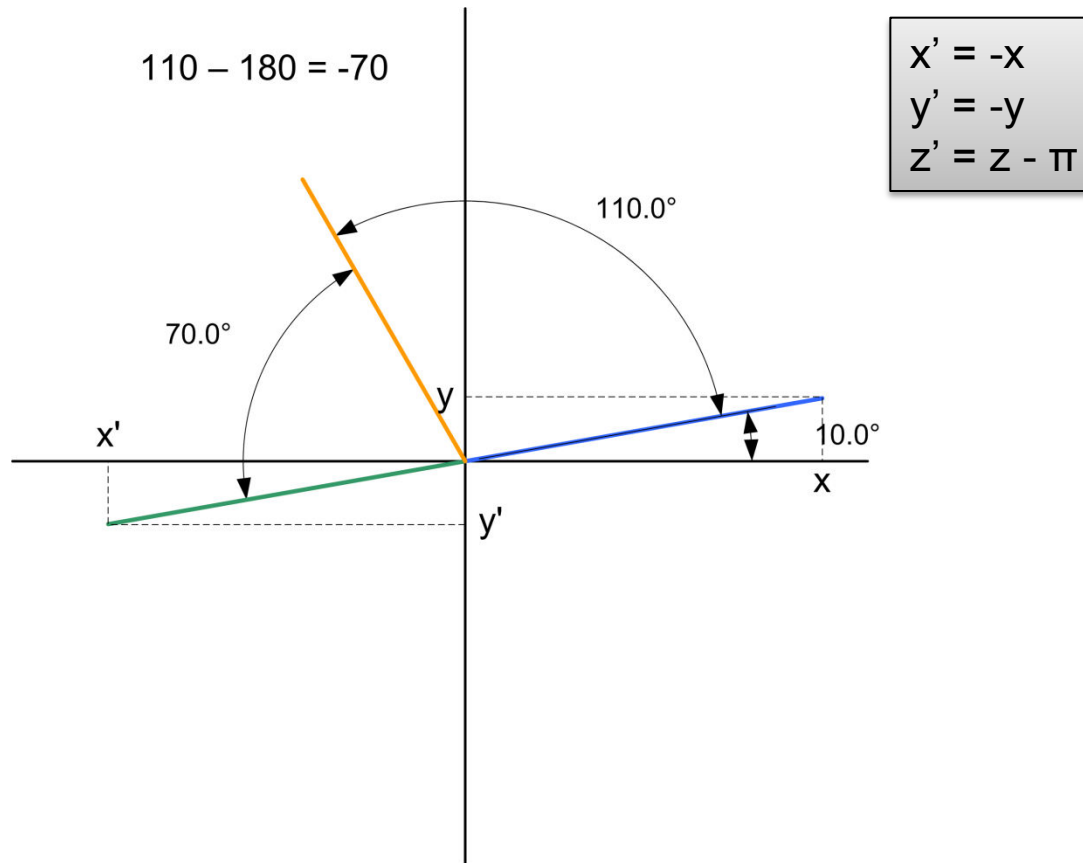
Bloques de pre-procesamiento y post_procesamiento

- **Preprocesamiento ($90 < \text{ang} < 180$)**



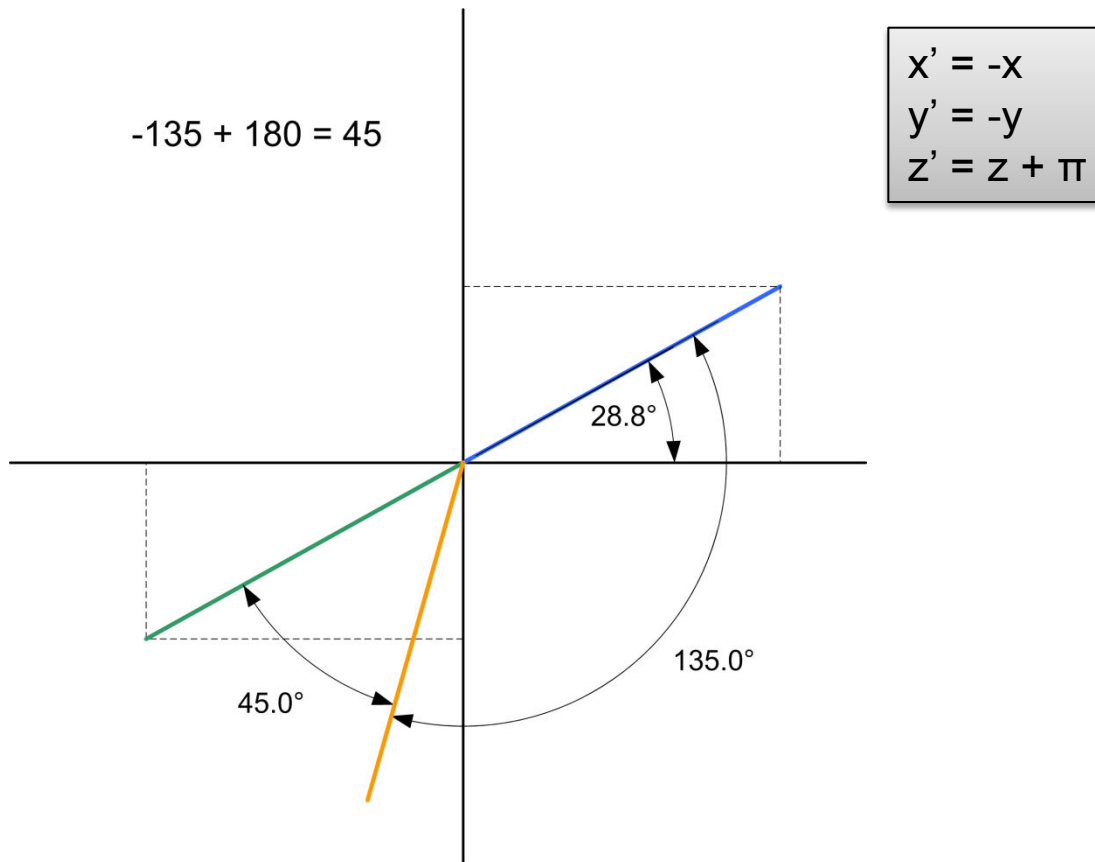
Bloques de pre-procesamiento y post_procesamiento

- **Preprocesamiento ($90 < \text{ang} < 180$)**



Bloques de pre-procesamiento y post_procesamiento

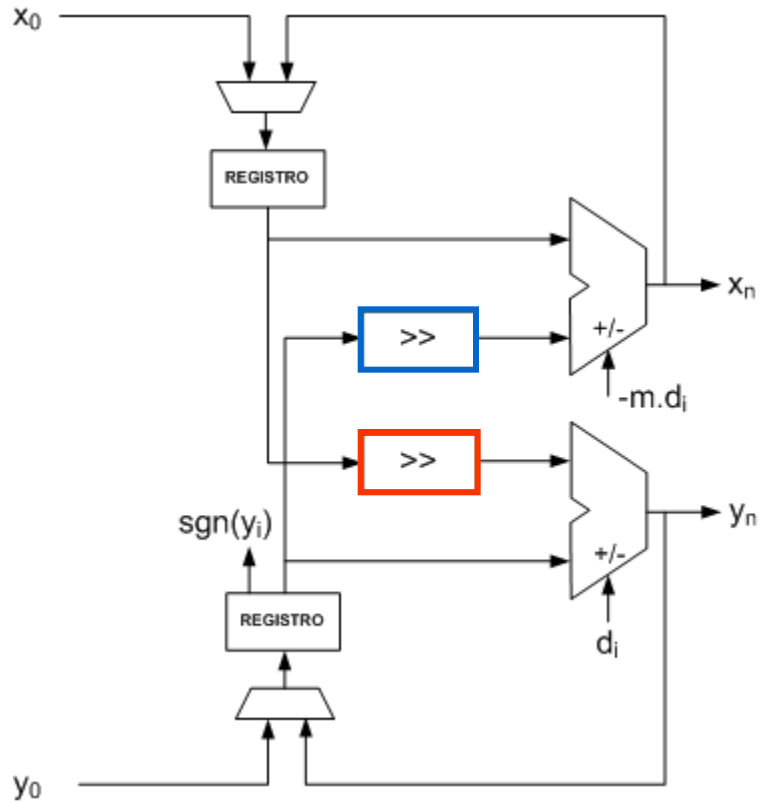
- Preprocesamiento ($180 < \text{ang} < 270$)



Arquitecturas

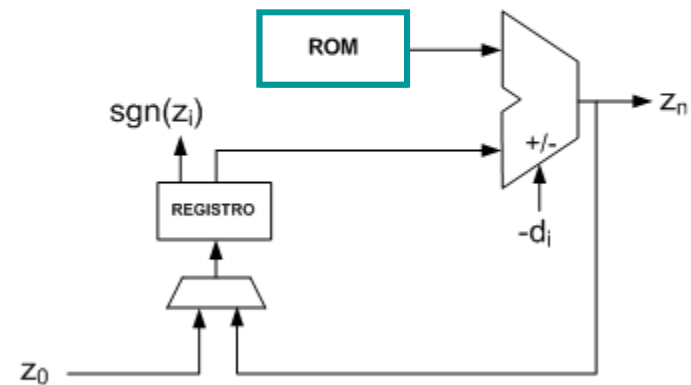
- ❖ Iterativa
- ❖ Unrolled
- ❖ Pipeline unrolled

Arquitectura iterativa



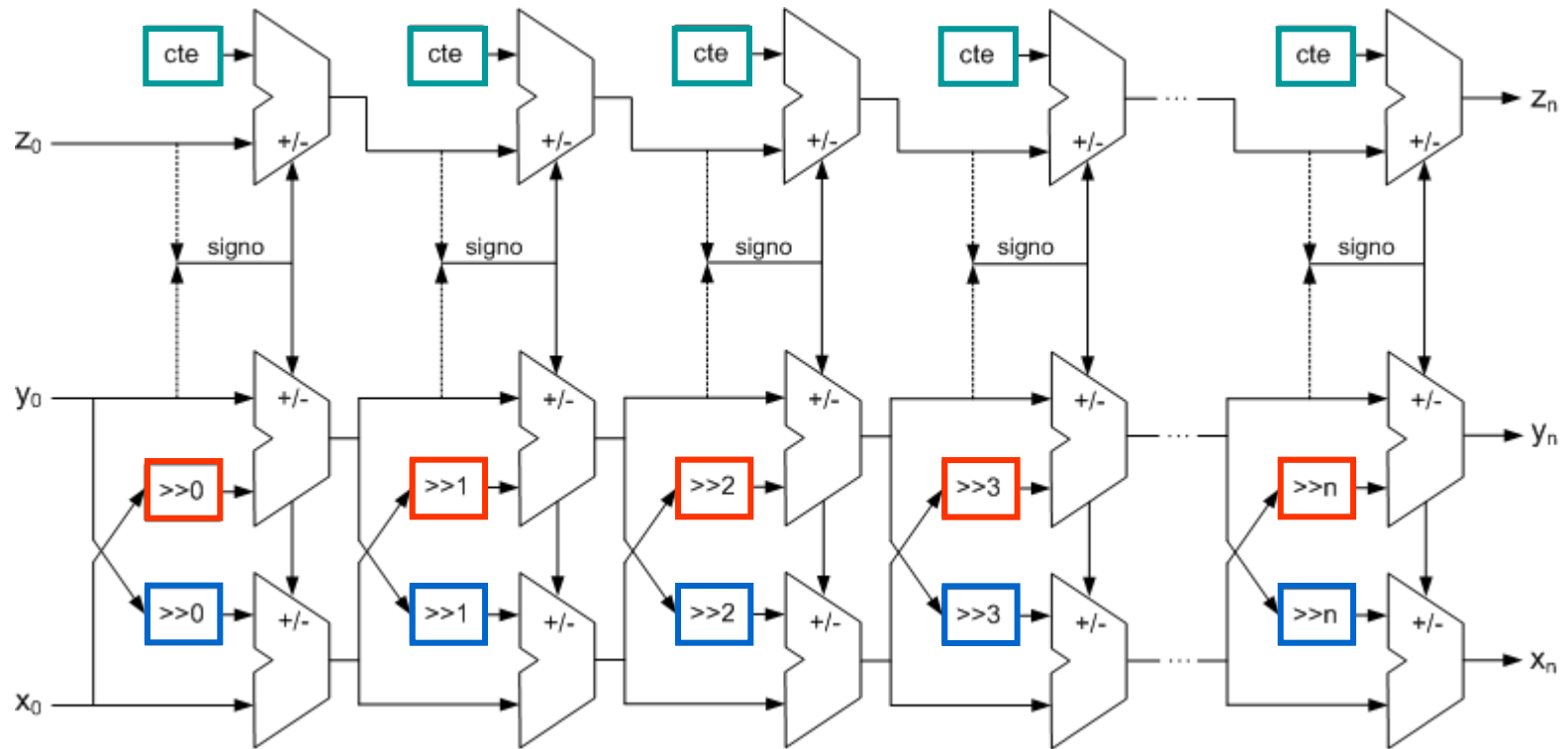
$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$

$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$

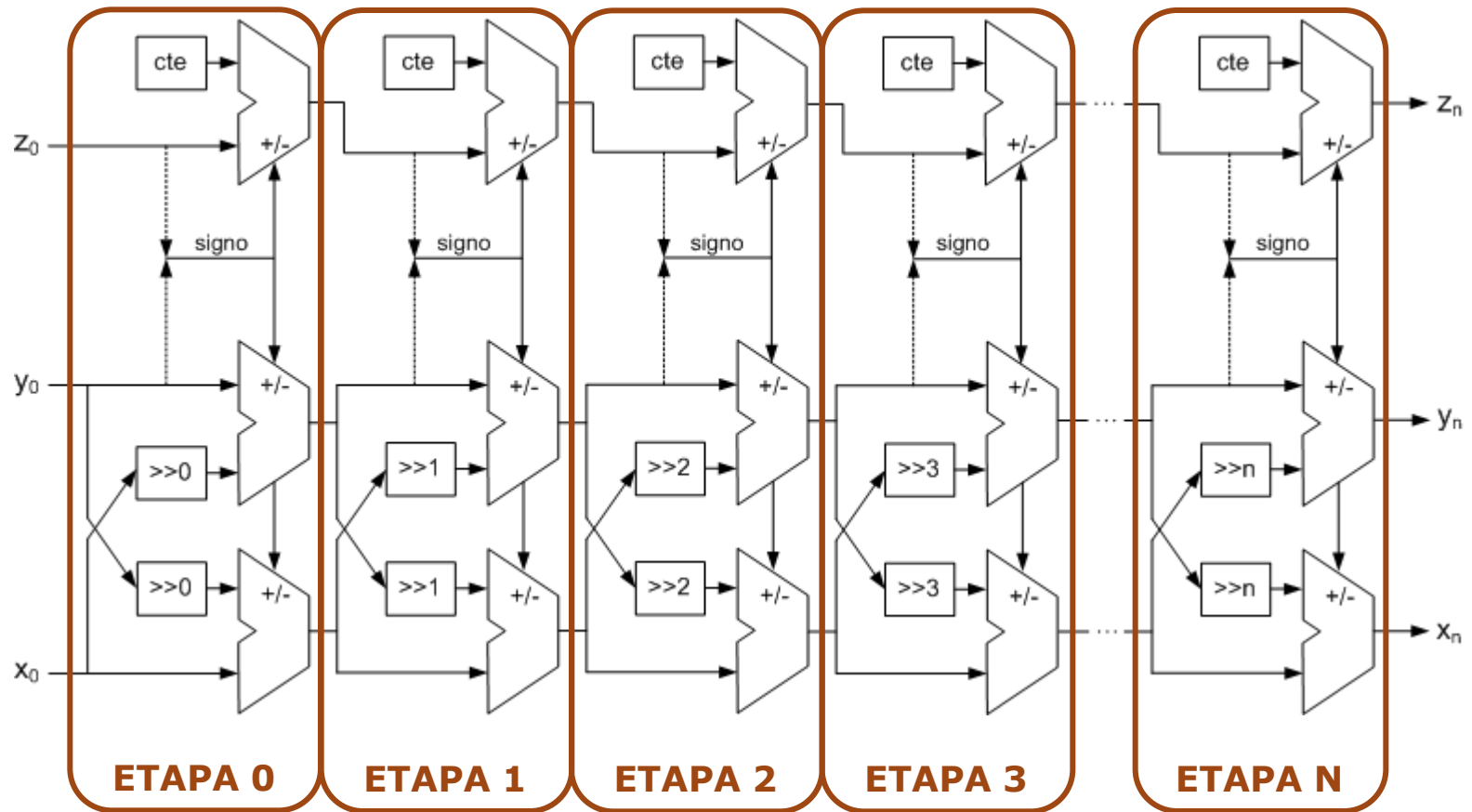


$$z_{i+1} = z_i - d_i \cdot \text{tg}^{-1}(2^{-i})$$

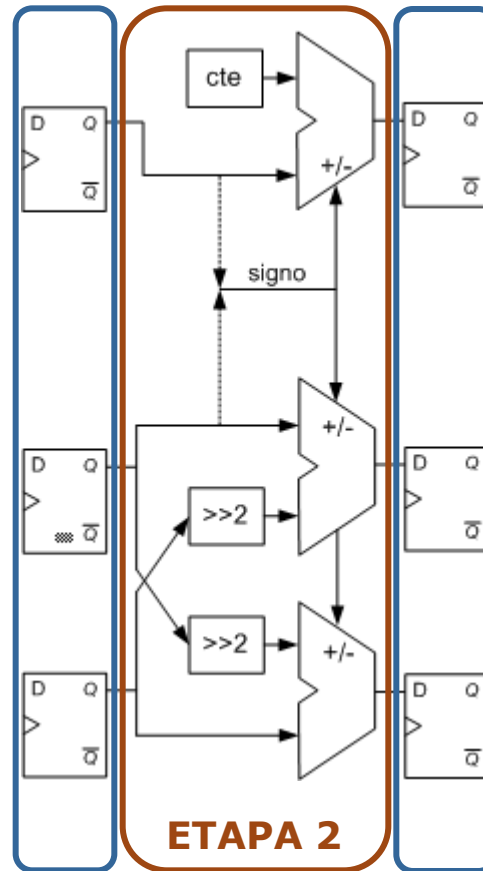
Arquitectura Unrolled



Arquitectura Pipeline Unrolled



Arquitectura Pipeline Unrolled



Ejercicios

- 1) Determinar cómo calcularía el seno y el coseno de un ángulo dado utilizando el algoritmo CORDIC.
- 2) Determinar cómo transformaría coordenadas polares en cartesianas utilizando el algoritmo CORDIC.
- 3) Determinar cómo calcularía la arcotangente de un ángulo dado utilizando el algoritmo CORDIC.
- 4) Determinar cómo transformaría coordenadas cartesianas en polares utilizando el algoritmo CORDIC.

Ejercicios

OPERATION	MODE	INITIALIZE	DIRECTION
Sine, Cosine	Rotation	$x=1/A_n, y=0, z=\alpha$	Reduce z to Zero
Polar to Rect.	Rotation	$x=(1/A_n)X_{mag}, y=0, z=X_{phase}$	Reduce z to Zero
General Rotation	Rotation	$x=(1/A_n)x_0, y=(1/A_n)y_0, z=\alpha$	Reduce z to Zero
Arctangent	Vector	$x=(1/A_n)x_0, y=(1/A_n)y_0, z=0$	Reduce y to Zero
Vector Magnitude	Vector	$x=(1/A_n)x_0, y=(1/A_n)y_0, z=0$	Reduce y to Zero
Rect. to Polar	Vector	$x=(1/A_n)x_0, y=(1/A_n)y_0, z=0$	Reduce y to Zero
Arcsine, Arccosine	Vector	$x=(1/A_n), y=0,$ $arg=\sin \alpha$ or $\cos \alpha$	Reduce y to Value in arg Register

FIN