CORDIC

JUAN CARLOS LLAMAS NÚÑEZ 3ºDG MAT-INF

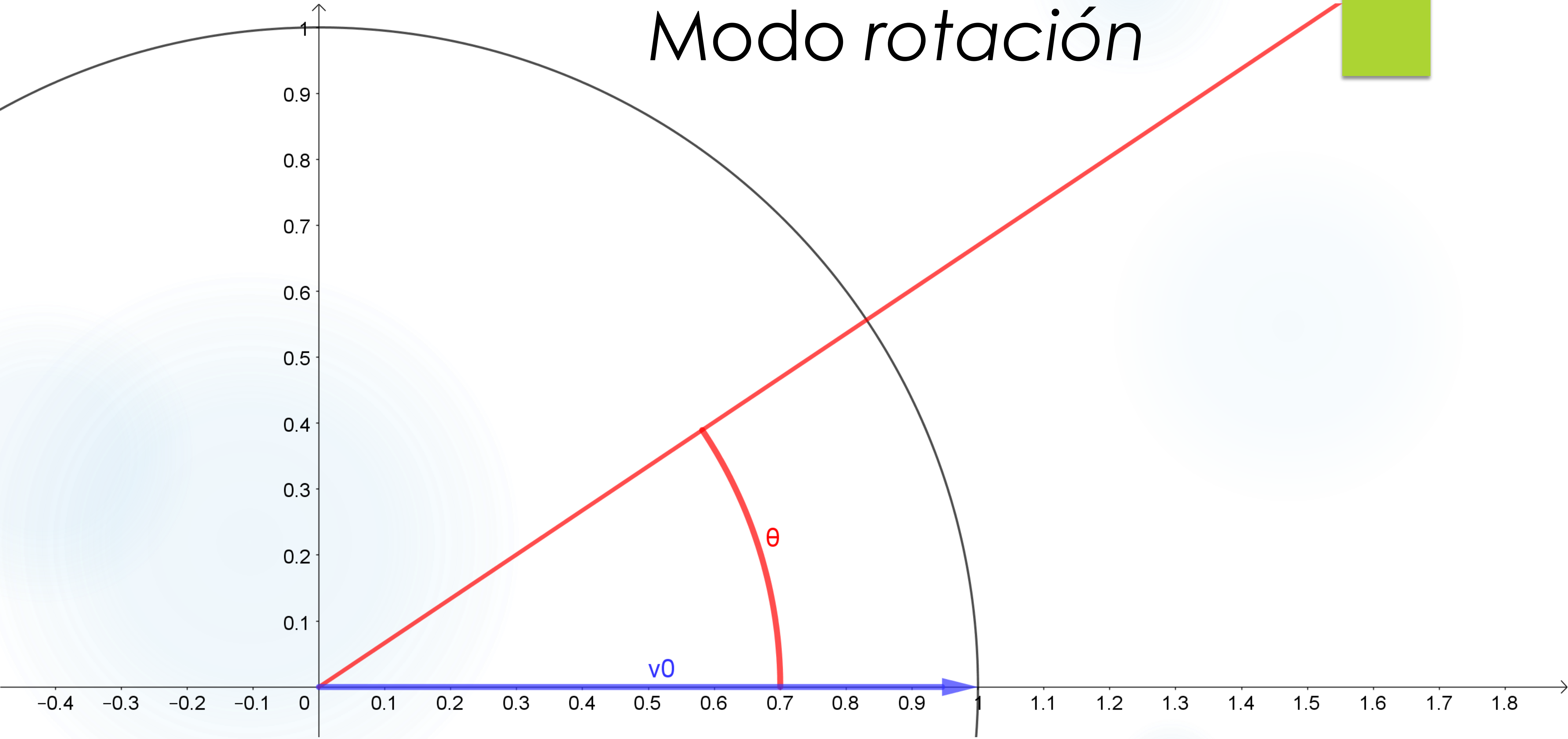
¿Qué es el algoritmo CORDIC?

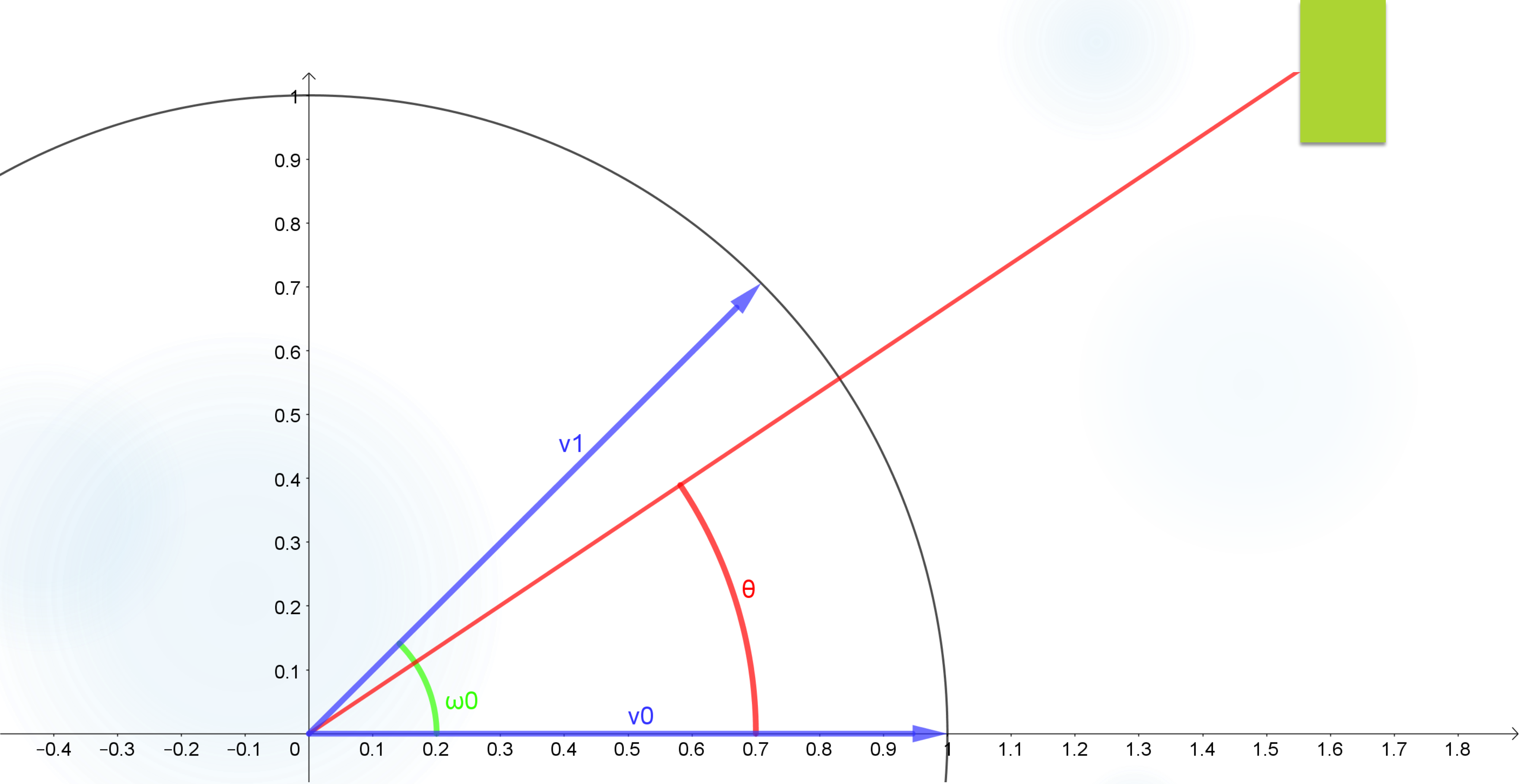
- CORDIC (COordinate Rotation DIgital Computer) computadora digital de rotación de coordenadas
- Cálculo de funciones trigonométricas e hiperbólicas
- Diseñado por Jack E.Volder (1956)
- Posteriores generalizaciones
- Evita el uso de un modulo multiplicador externo

Modos de operación

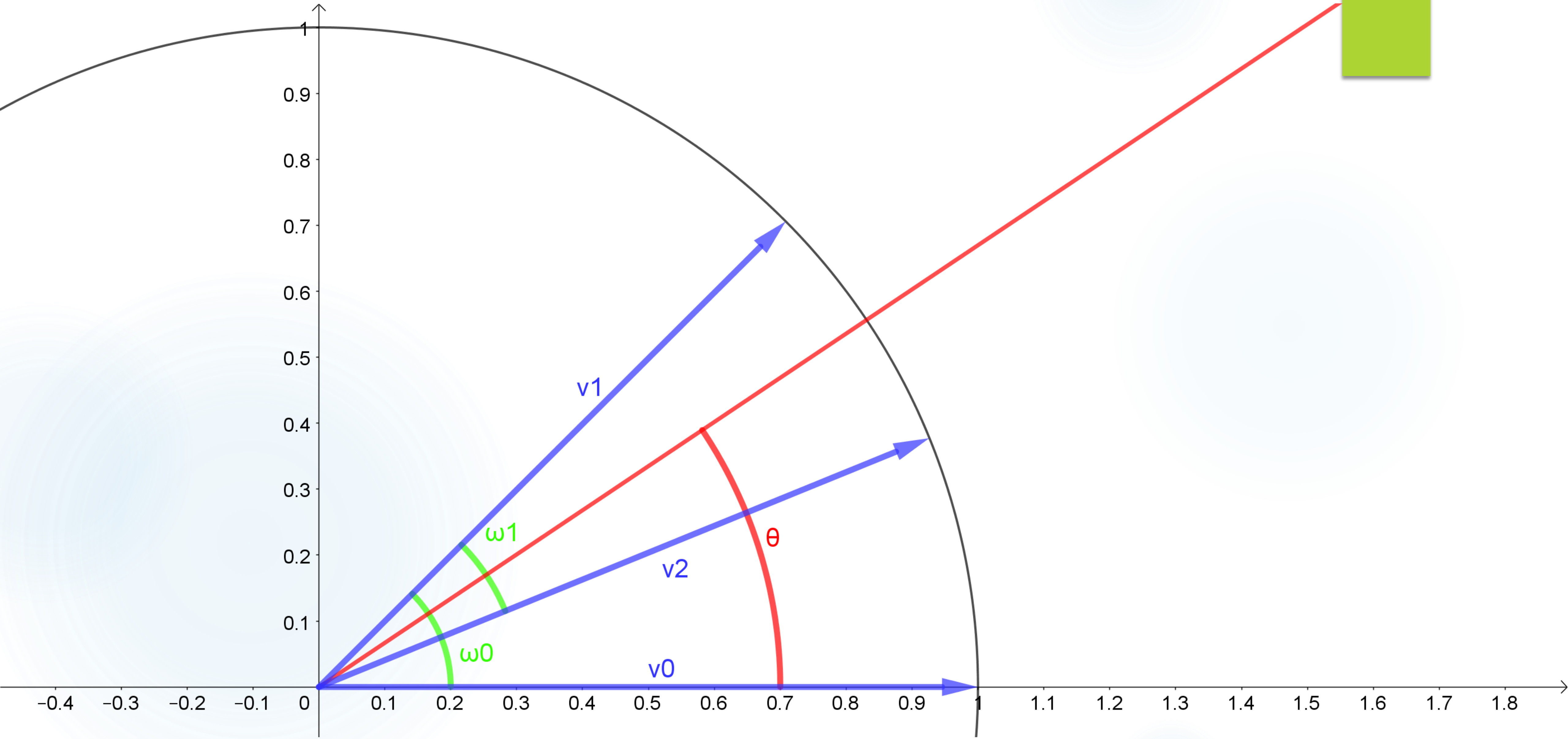
- Modo *rotación* vs. Modo *vectorización*
- Modo *rotación* : recibimos un ángulo y calculamos su seno y coseno
- Modo *vectorización* : recibimos un vector y calculamos su ángulo respecto al eje X

Modo *rotación*

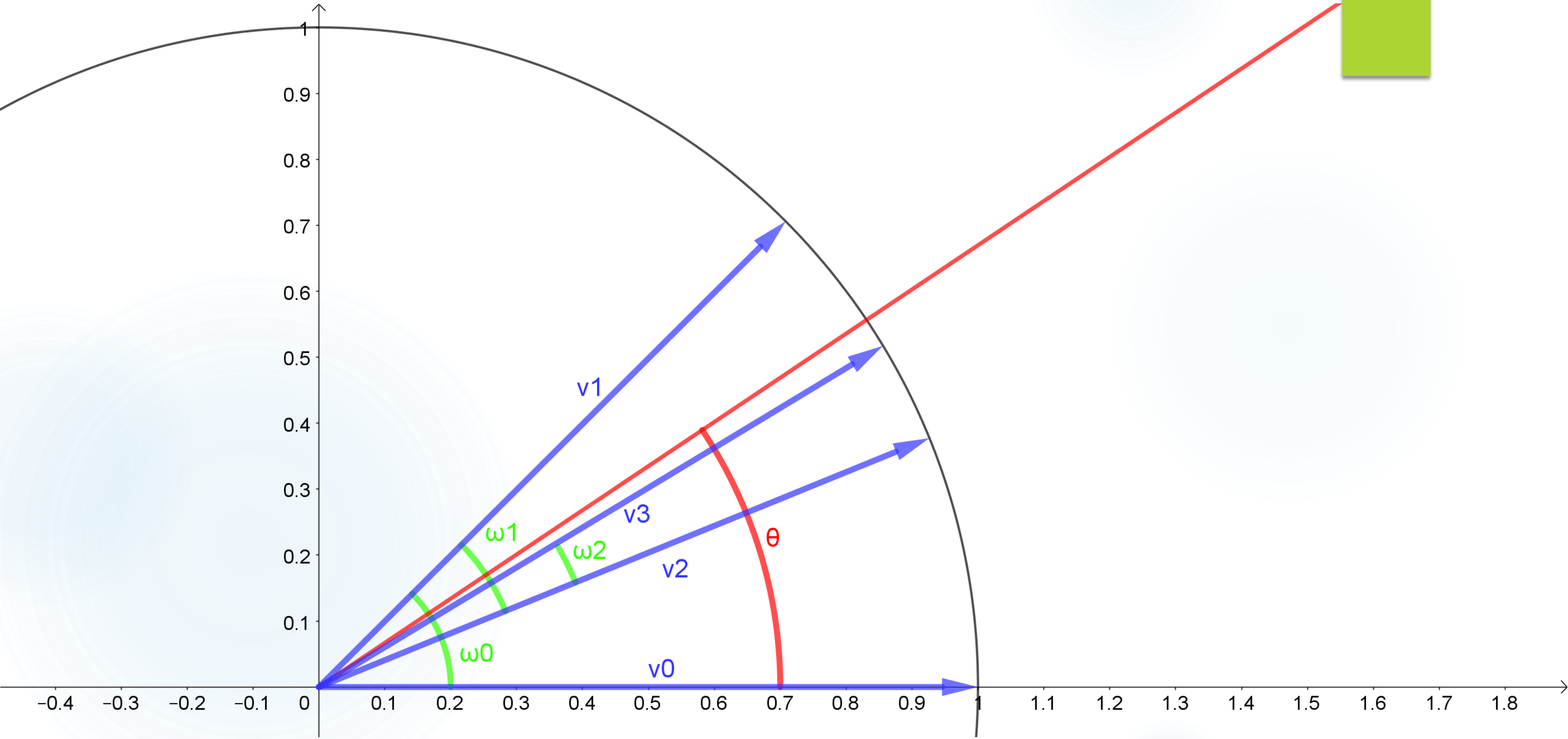




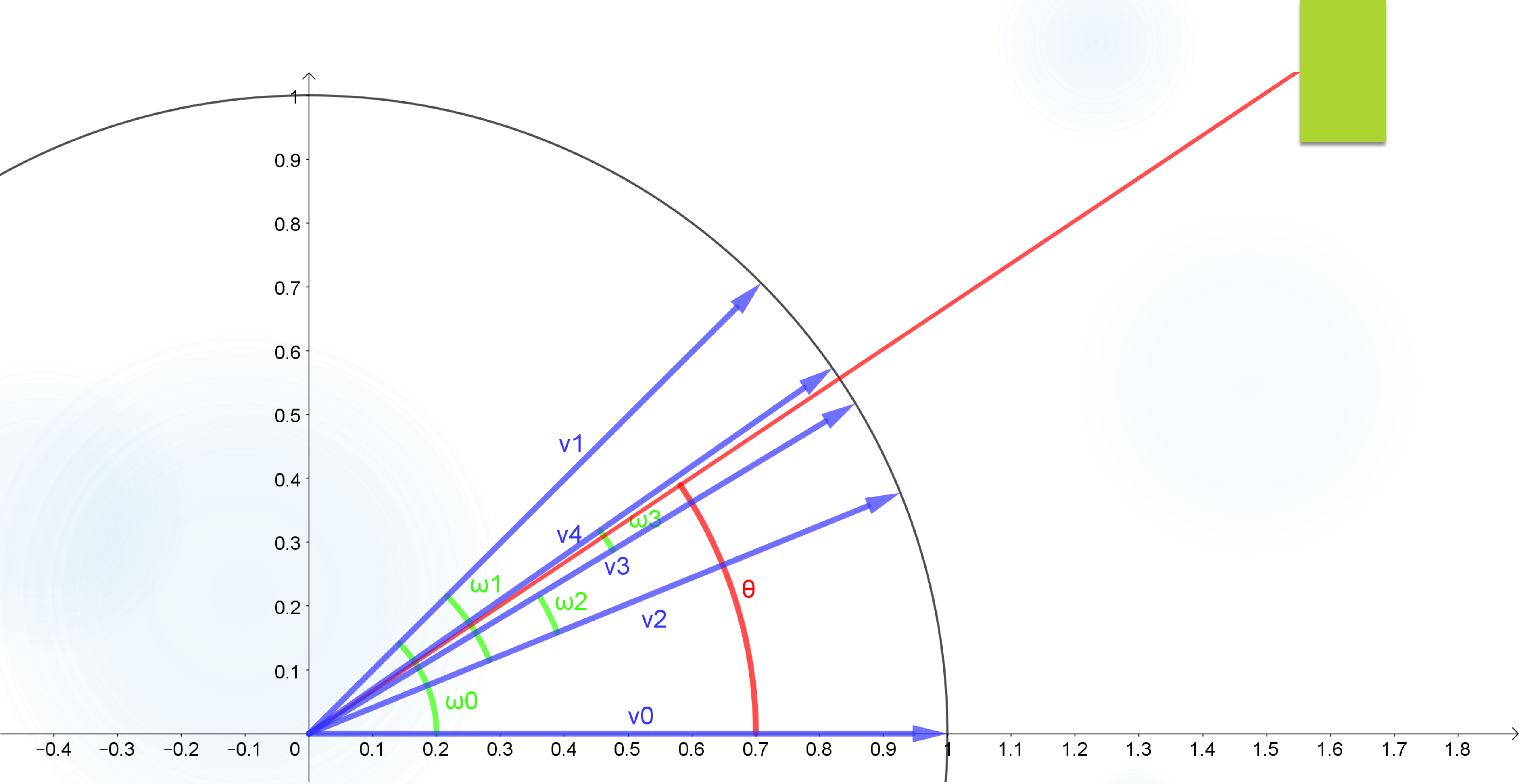
v1
θ
ω0
v0



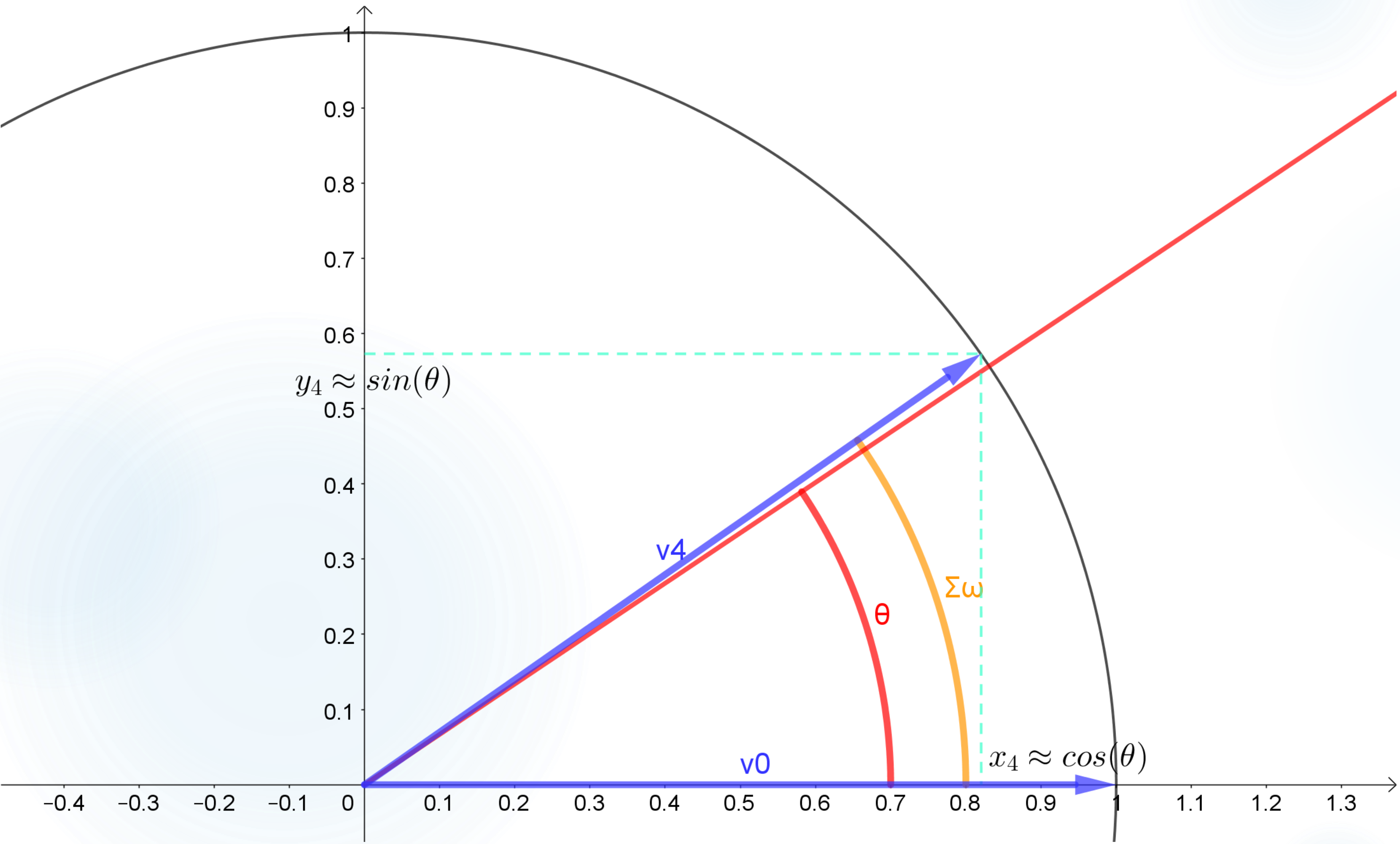
v1
v2
v0
ω1
ω0
θ



1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
−0.4
−0.3
−0.2
−0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
v1
v3
v2
v0
ω1
ω2
ω0
θ



v1
v4
v3
v2
v0
ω0
ω1
ω2
ω3
θ



1
0.9
0.8
0.7
0.6
$y_4 \approx sin(\theta)$
0.5
0.4
0.3
0.2
0.1
v4
θ
Σω
v0
$x_4 \approx cos(\theta)$
−0.4 −0.3 −0.2 −0.1 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1.1 1.2 1.3

Modo *rotación*

- Elección de los ángulos de rotación w_i

$$\delta_i = +1 \text{ o } -1, \text{ sentido de rotación}$$
$$|\omega_i| = arctan(2^{-i})$$

- Sentido de rotación

$$z_n = \theta - \sum_{i=0}^{n} \omega_i$$

$$\delta_i = \begin{cases} +1, & \text{si} \quad z_i \geq 0 \\ -1, & \text{si} \quad z_i < 0 \end{cases}.$$

Recurrencias

$$x_0 = 1, \quad y_0 = 0, \quad z_0 = \theta$$

$$\begin{cases} x_{i+1} = x_i - \delta_i 2^{-i} y_i \\ y_{i+1} = y_i + \delta_i 2^{-i} x_i \\ z_{i+1} = z_i - \delta_i arctan(2^{-i}) \end{cases}$$

- Solamente sumas, restas y desplazamientos hacia la izquierda
- Podemos almacenar los valores de las arcotangentes en una LUT

Tras n iteraciones…

$$\sum_{i=0}^{n-1} \omega_i \approx \theta$$

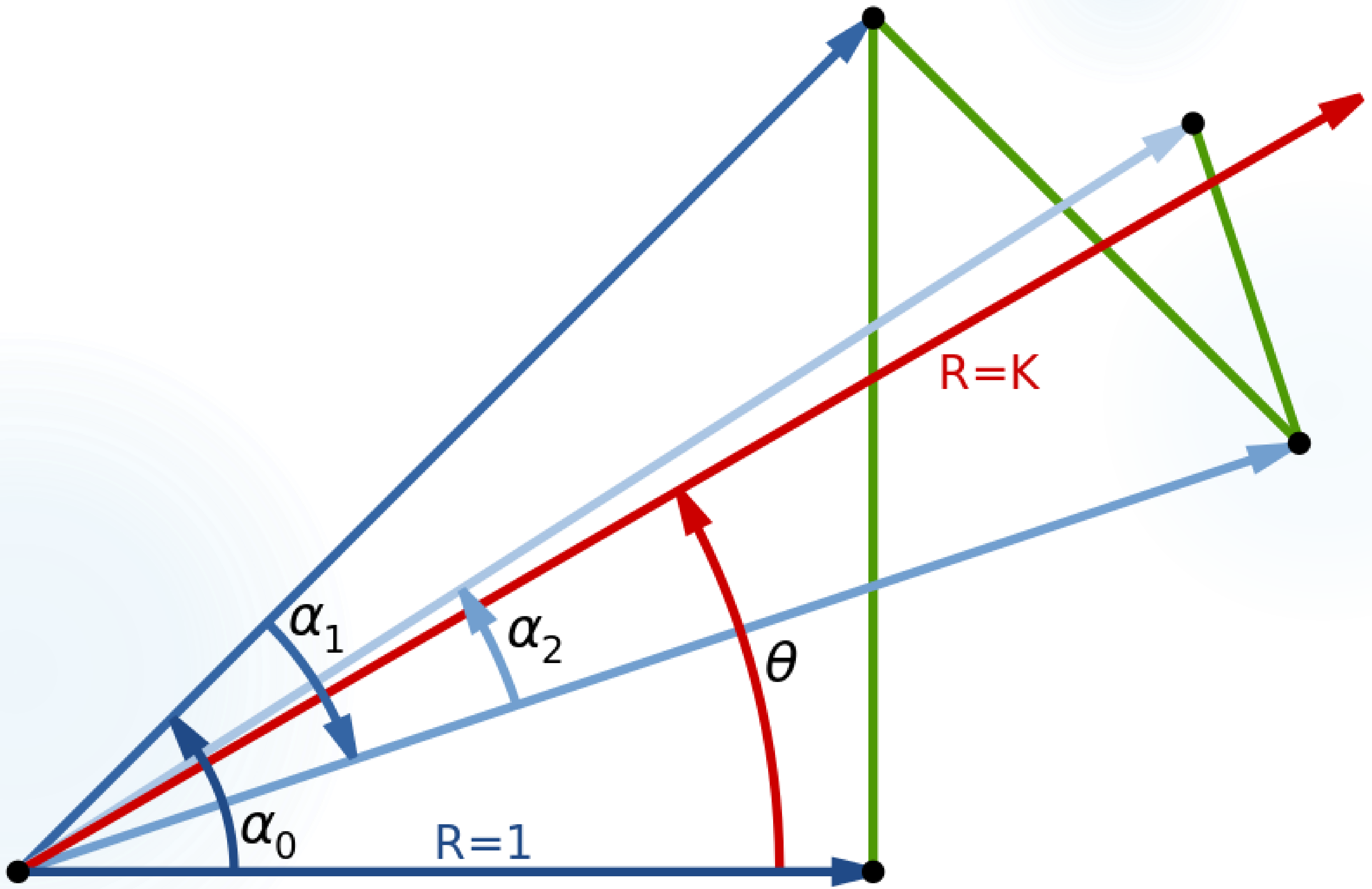
$$K(n)\ x_n = cos(\sum_{i=0}^{n-1} \omega_i) \approx cos(\theta)$$

$$K(n)\ y_n = sin(\sum_{i=0}^{n-1} \omega_i) \approx sin(\theta)$$

¿Qué hacemos con K(n)?

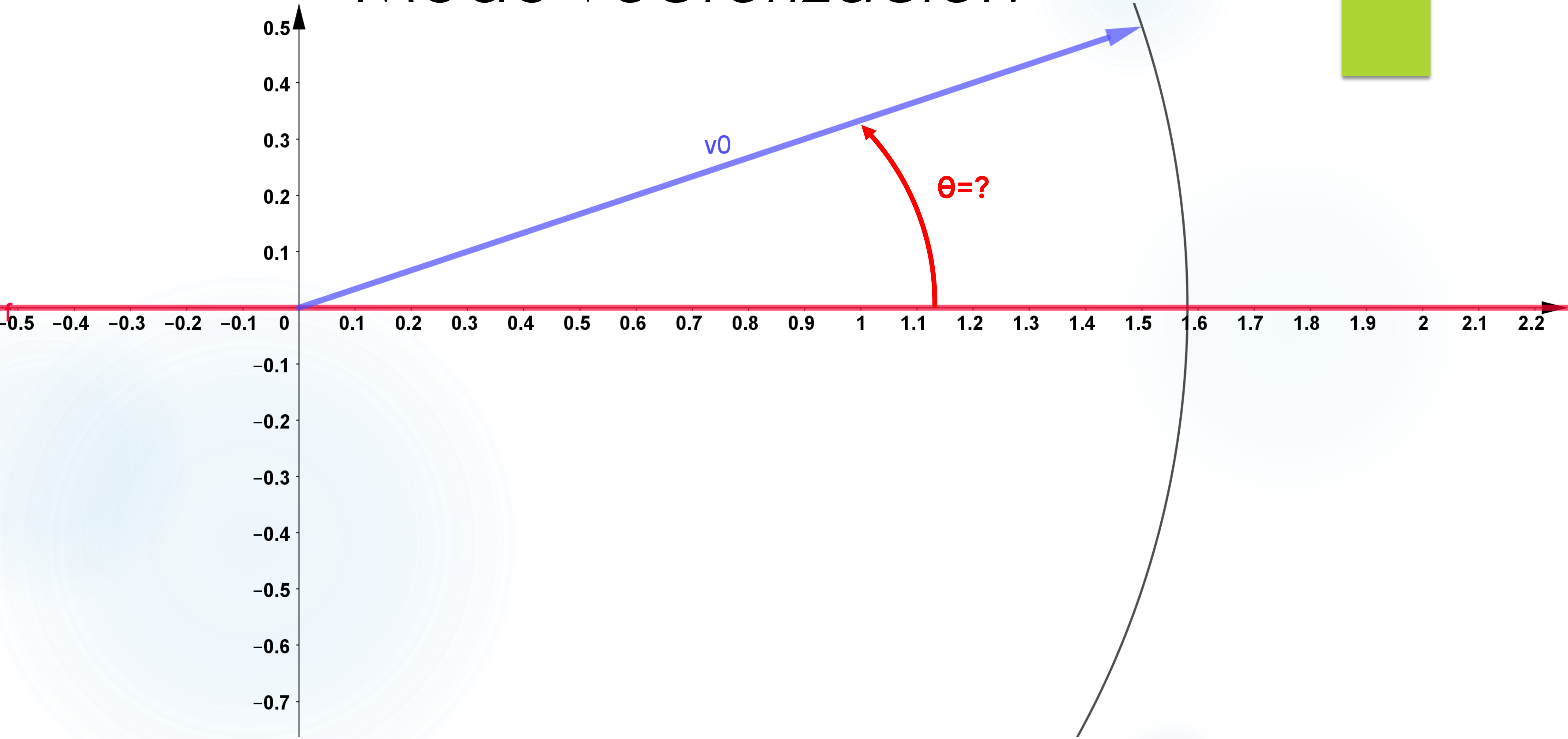
- Podemos inicializar el algoritmo con $$\hat{v}_0 = K(n)v_0 = \begin{pmatrix} K(n) \\ 0 \end{pmatrix}$$

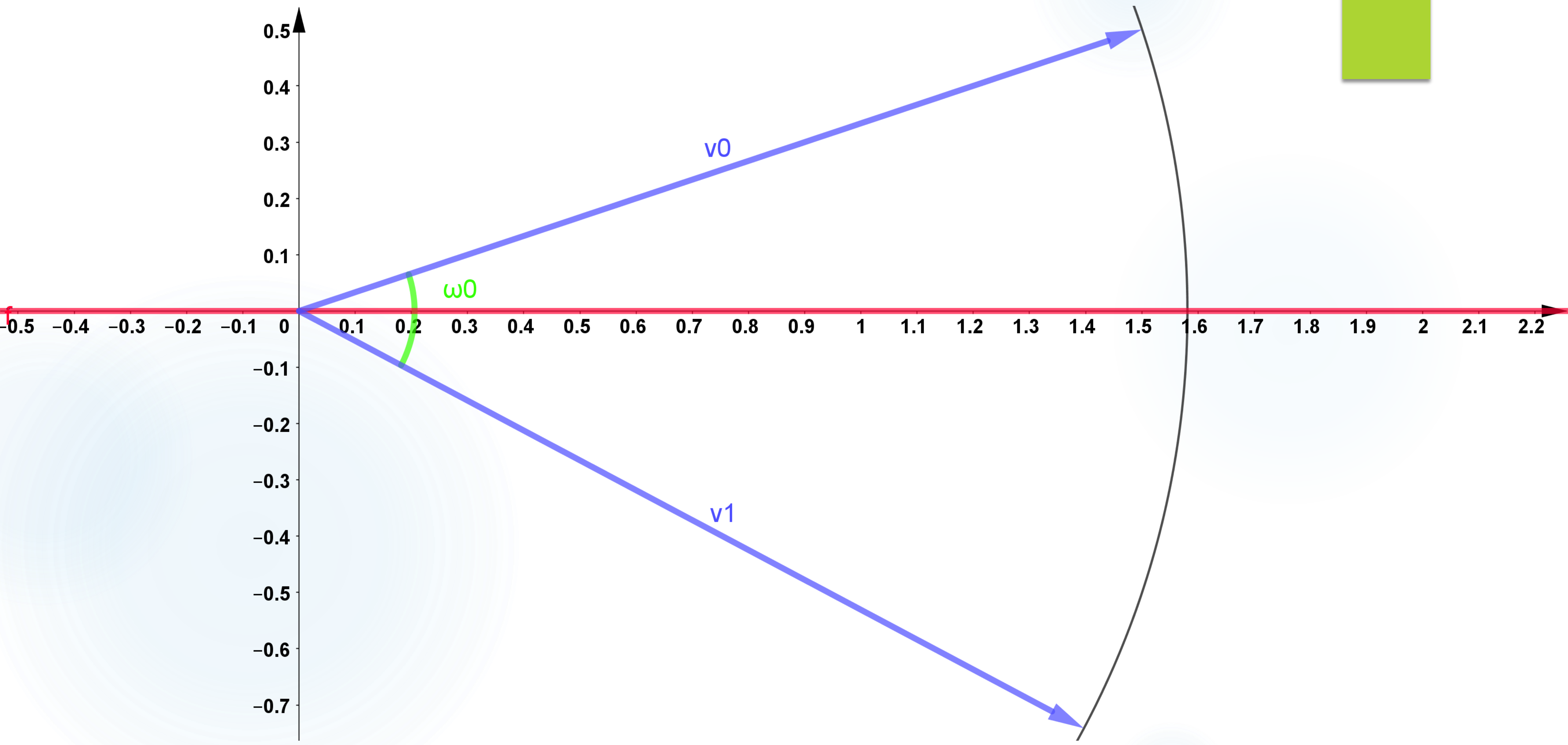
- Podemos recordar que el resultado está multiplicado por 1/K(n)

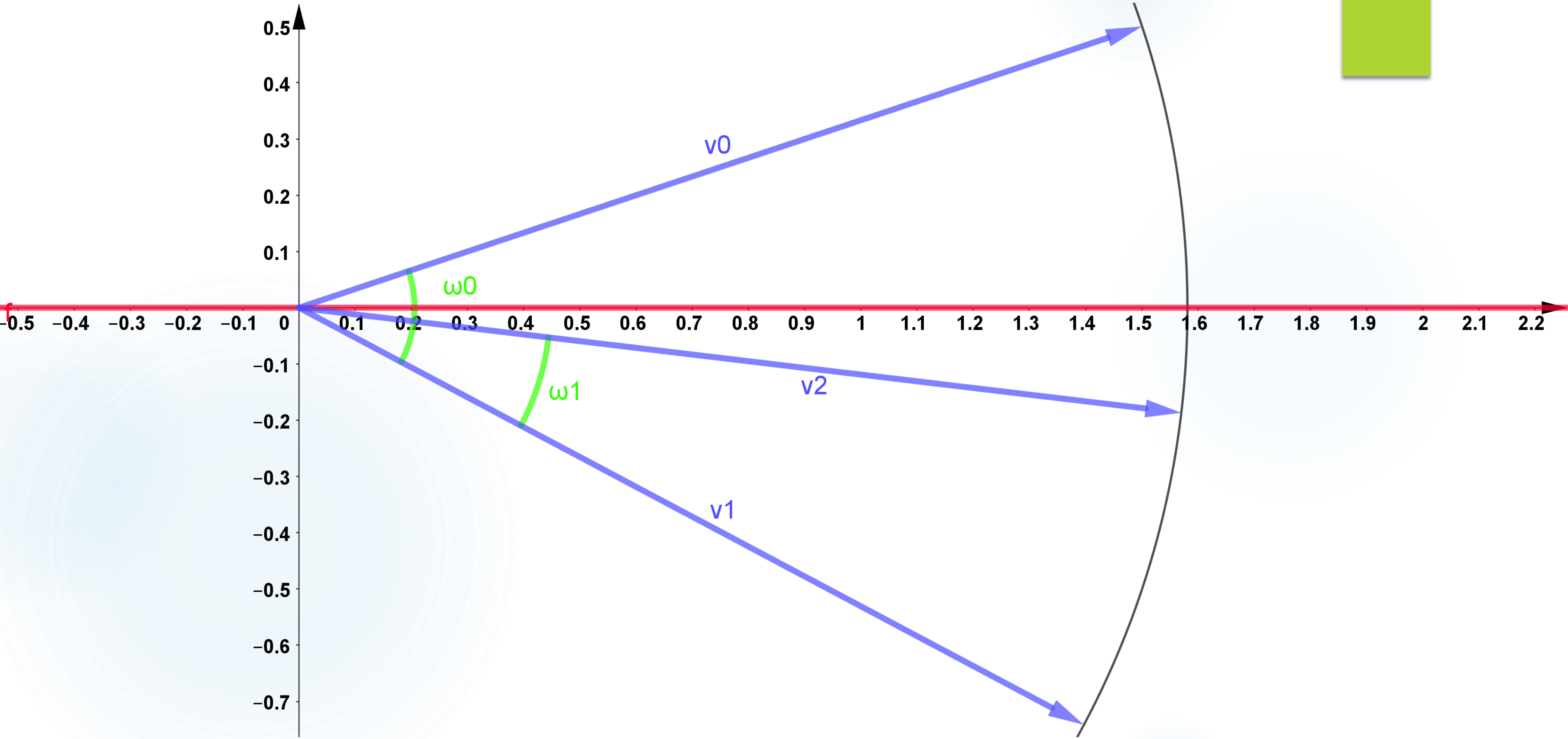


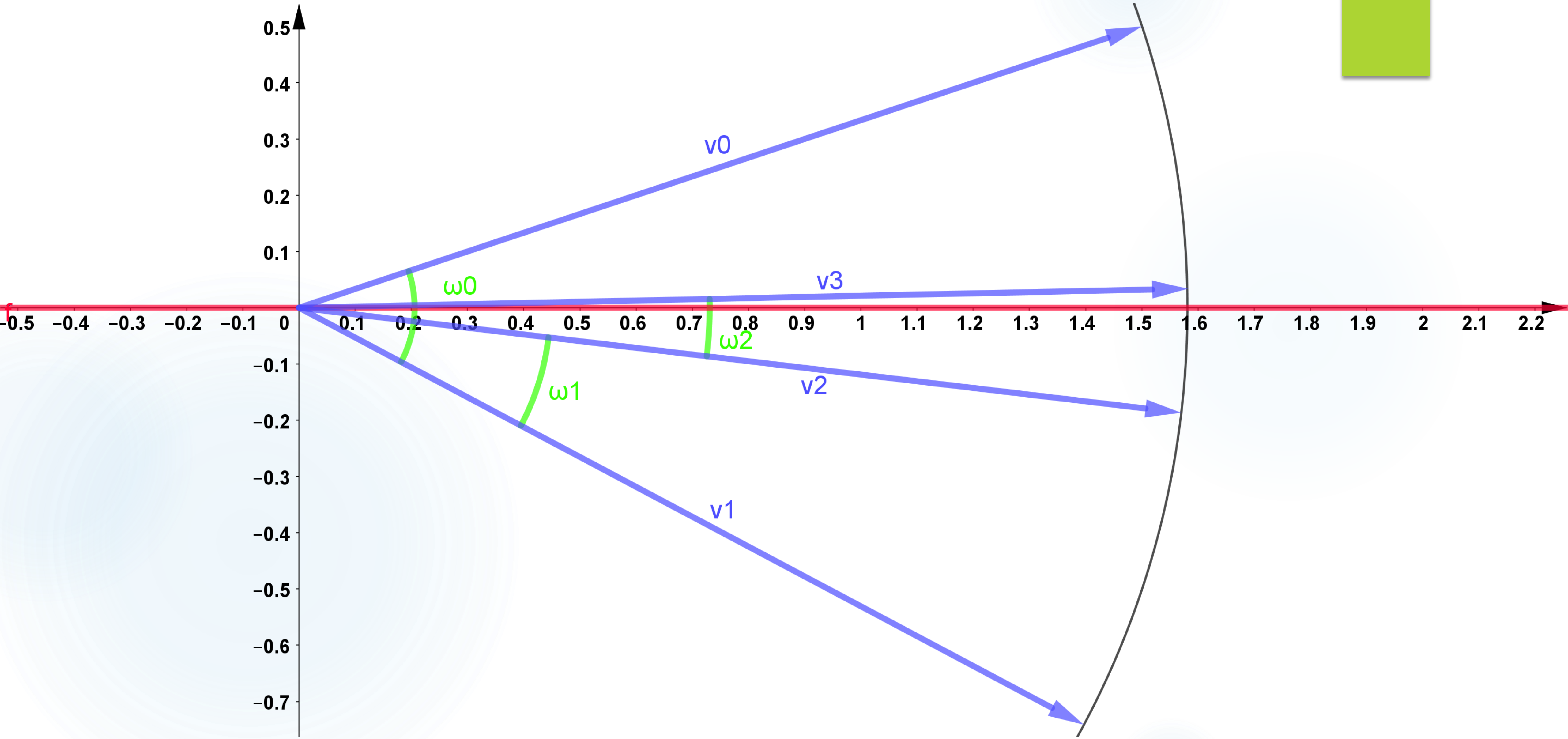
R=K
α_1
α_2
θ
α_0
R=1

Modo vectorización

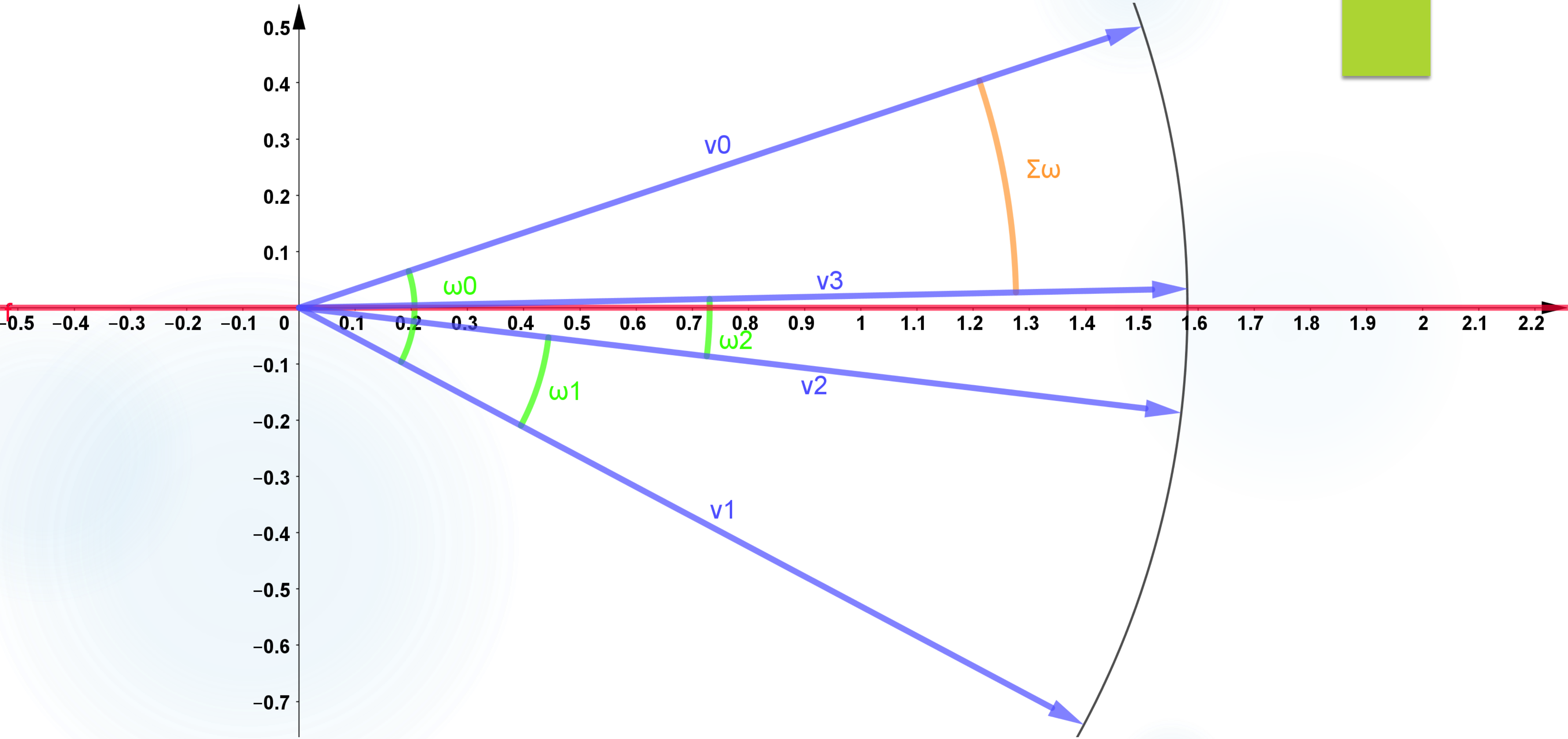


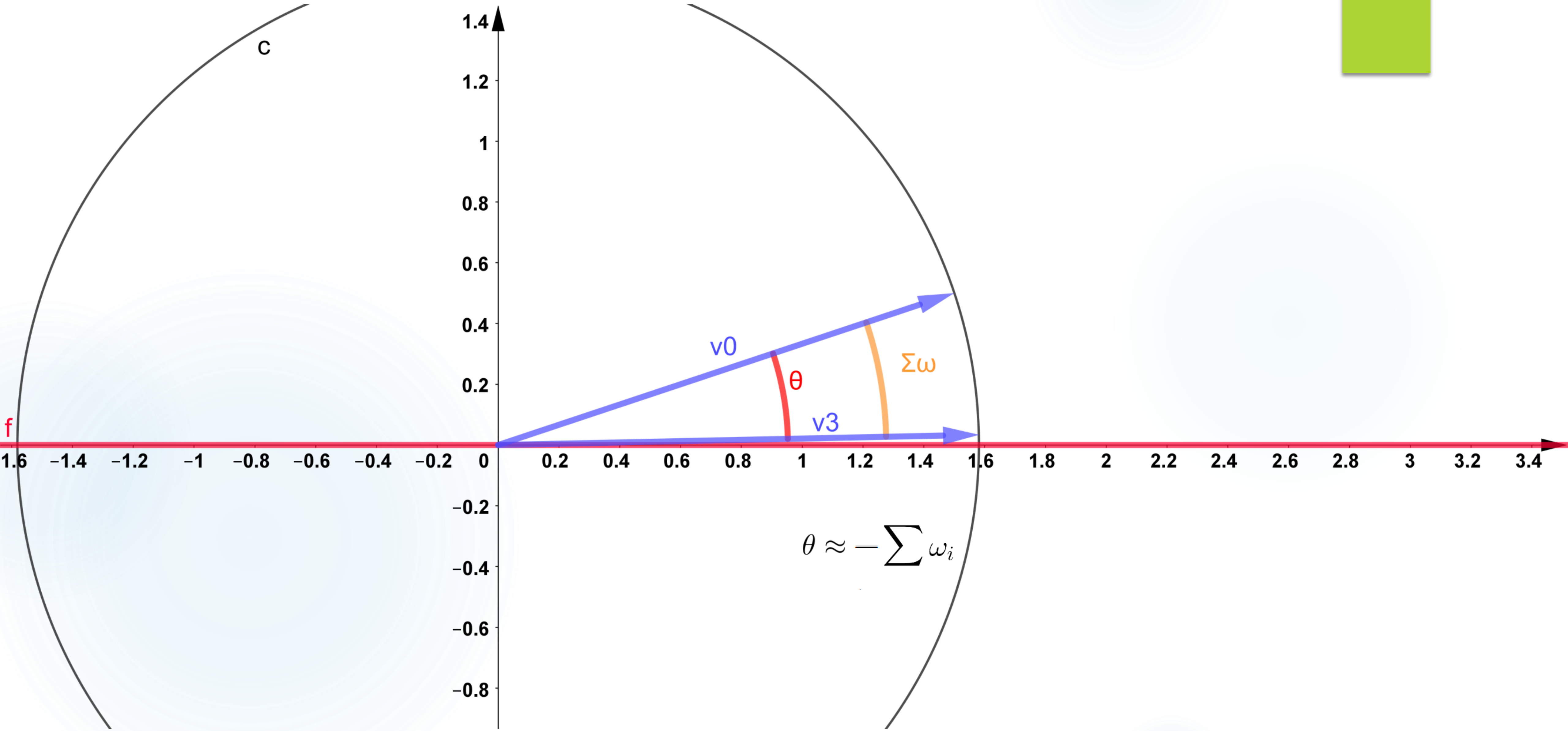

v0
v1
ω0
0.5
0.4
0.3
0.2
0.1
0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
−0.5
−0.4
−0.3
−0.2
−0.1
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
2
2.1
2.2


v0
v1
v2
ω0
ω1
f



v0
v1
v2
v3
ω0
ω1
ω2


v0
Σω
v3
ω0
ω2
v2
ω1
v1



$$\theta \approx -\sum \omega_i$$

Modo vectorización

- Sentido de rotación

$$z_n = -\sum_{i=0}^{n} \omega_i$$

$$\delta_i = \begin{cases} +1, & \text{si} \quad y_i \leq 0 \\ -1, & \text{si} \quad y_i > 0 \end{cases}$$

- X_{i+1}, y_{i+1} , z_{i+1} se calculan igual que en el modo *rotación*

$$x_0 = v_0^1, \quad y_0 = v_0^2, \quad z_0 = 0$$

$$\begin{cases} x_{i+1} = x_i - \delta_i 2^{-i} y_i \\ y_{i+1} = y_i + \delta_i 2^{-i} x_i \\ z_{i+1} = z_i - \delta_i arctan(2^{-i}) \end{cases}$$

Tras n iteraciones…

$$y_n \approx 0$$

$$x_n = \frac{1}{K(n)}\sqrt{x_0^2 + y_0^2}$$

$$z_n \approx arctan(\frac{y_0}{x_0})$$

Aplicaciones

- Calculo de senos y cosenos
- Calculo de arcotangentes y módulo de vectores
- Cambio de coordenadas

Generalizaciones

$$\begin{cases} x_{i+1} = x_i - m\delta_i 2^{-i} y_i \\ y_{i+1} = y_i + \delta_i 2^{-i} x_i \\ z_{i+1} = z_i - \delta_i f(2^{-i}) \end{cases}$$

- Nuevos parámetros f(x) y m
- Si m = 1 y f(x) = arctan(x), CORDIC básico

$$\begin{cases} x_{i+1} = x_i \\ y_{i+1} = y_i + \delta_i 2^{-i} x_i \\ z_{i+1} = z_i - \delta_i 2^{-i} \end{cases}$$

- Si m = 0 y f(x) = x, CORDIC **lineal**
- Si m = -1 y f(x) = arctanh(x), CORDIC **hiperbólico**

$$\begin{cases} x_{i+1} = x_i + \delta_i 2^{-i} y_i \\ y_{i+1} = y_i + \delta_i 2^{-i} x_i \\ z_{i+1} = z_i - \delta_i arctanh(2^{-i}) \end{cases}$$

Mayor capacidad de cómputo

- CORDIC **lineal** en modo *rotación*: productos
- CORDIC **lineal** en modo *vectorización*: divisiones
- CORDIC **hiperbólico** en modo *rotación*: senos y cosenos hiperbólicos
- CORDIC **hiperbólico** en modo *vectorización*: raíces y arcotangentes hiperbólicas

Y combinándolos apropiadamente…

$$tan(x) = \frac{sin(x)}{cos(x)}, \quad cotan(x) = \frac{cos(x)}{sin(x)}, \quad tanh(x) = \frac{sinh(x)}{cosh(x)},$$

$$cotanh(x) = \frac{cosh(x)}{sinh(x)}, \quad sec(x) = \frac{1}{cos(x)}, \quad cosec(x) = \frac{1}{sin(x)},$$

$$sech(x) = \frac{1}{cosh(x)}, \quad cosech(x) = \frac{1}{sinh(x)}, \quad exp(x) = sinh(x) + cosh(x),$$

$$ln(w) = 2arctanh(y/x) \quad \text{con} \quad x = w + 1 \quad \text{e} \quad y = w - 1 \quad \text{o}$$

$$\sqrt{w} = \sqrt{x^2 - y^2} \ \text{ para } \ x = w + 1/4 \quad \text{e} \quad y = w - 1/4\ .$$

Sin embargo…

- Convergencia en dominio de parámetros reducidos

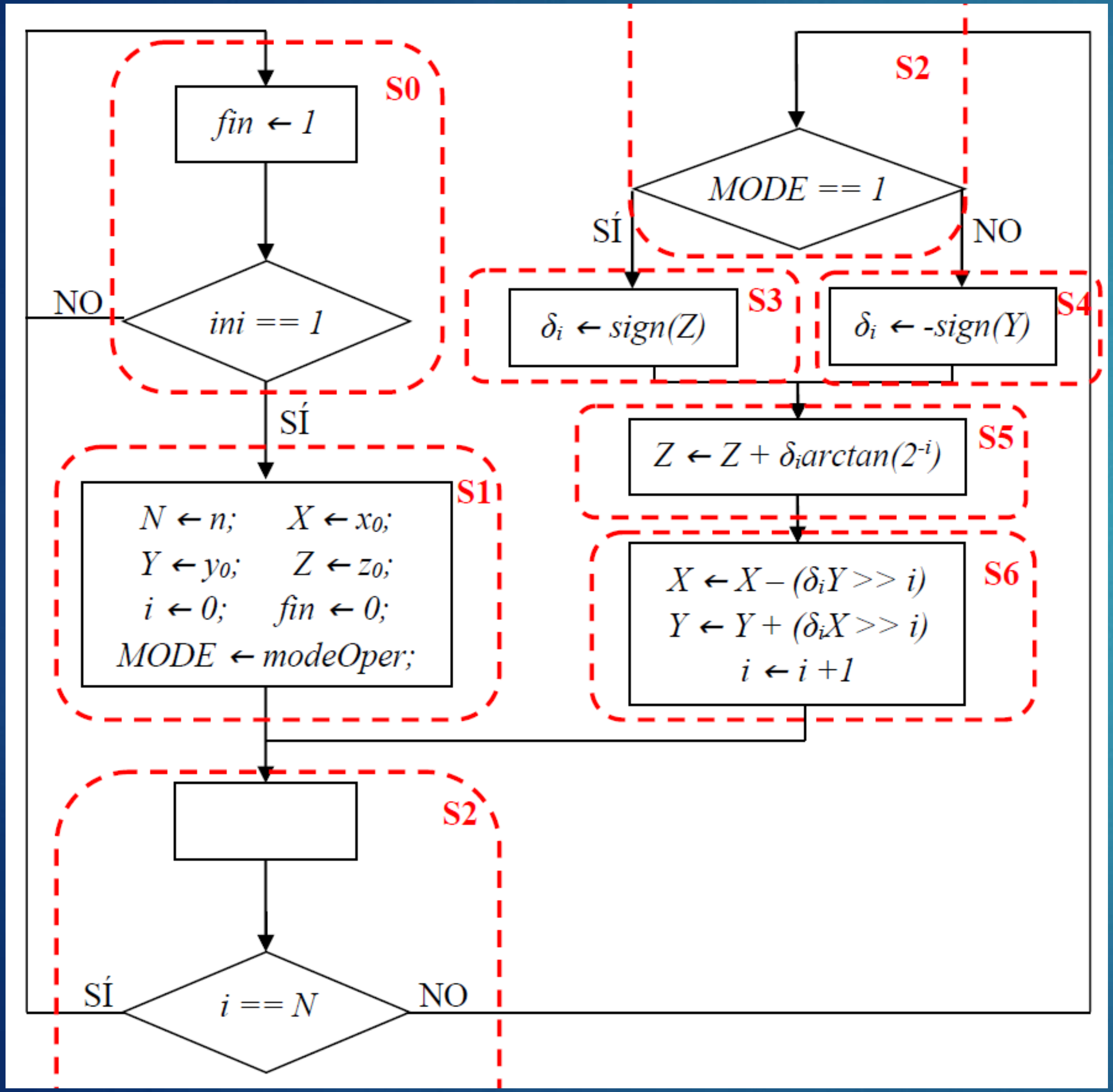
- Se puede mejorar haciendo iteraciones dobles o utilizando igualdades algebraicas

- Número de iteraciones: en la práctica alrededor de 40

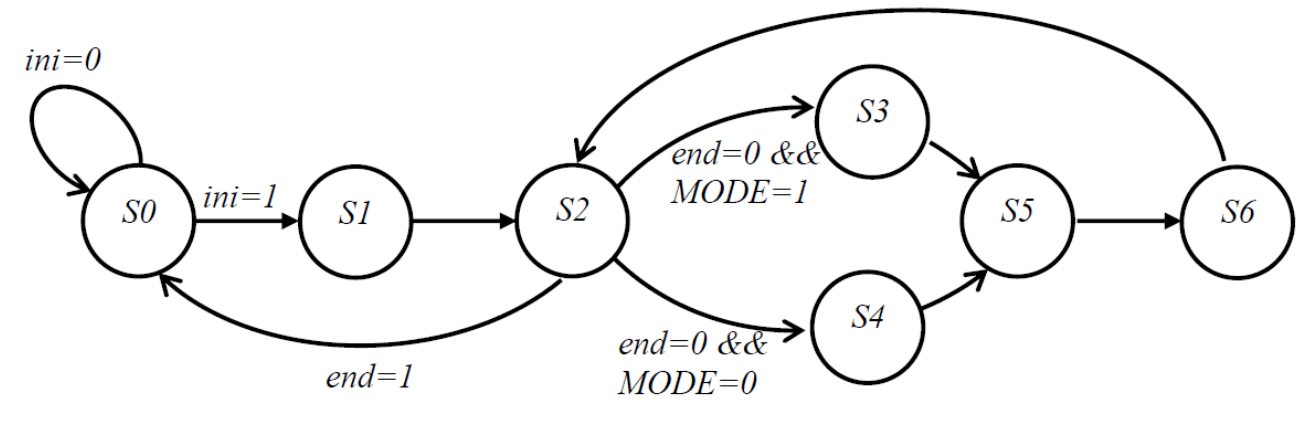
Implementación CORDIC básico

Algorithm 1: CORDIC

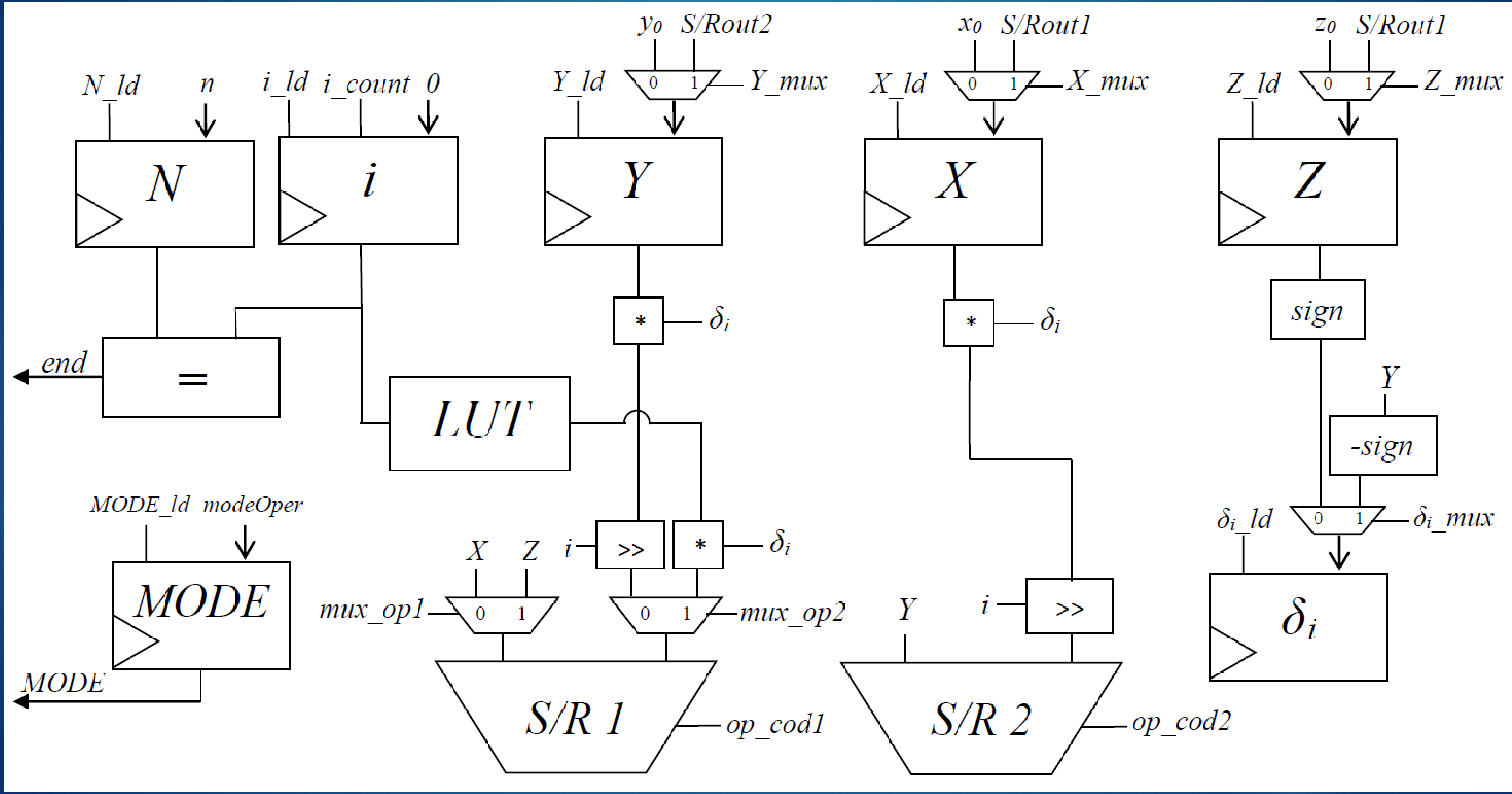
$N \leftarrow n;$
$X \leftarrow x_0;$
$Y \leftarrow y_0;$
$Z \leftarrow z_0;$
$i \leftarrow 0;$
$MODE \leftarrow modeOper;$
while $i < N$ **do**
  if $MODE == 1$ **then**
    $\delta_i \leftarrow sign(Z);$
  else
    $\delta_i \leftarrow -sign(Y);$
  end
  $Z \leftarrow Z + \delta_i arctan(2^{-i});$
  //Asignaciones simultáneas;
  $X \leftarrow X - (\delta_i Y >> i);$
  $Y \leftarrow Y + (\delta_i X >> i);$
  $i \leftarrow i + 1;$
end



S0
fin ← 1
ini == 1
NO
SÍ
S1
N ← n; X ← x0;
Y ← y0; Z ← z0;
i ← 0; fin ← 0;
MODE ← modeOper;
S2
SÍ
i == N
NO
S2
MODE == 1
SÍ
NO
S3
δi ← sign(Z)
S4
δi ← -sign(Y)
S5
Z ← Z + δiarctan(2-i)
S6
X ← X – (δiY >> i)
Y ← Y + (δiX >> i)
i ← i +1



ini=0
S0
ini=1
S1
S2
end=0 &&
MODE=1
S3
end=0 &&
MODE=0
S4
S5
S6
end=1



N_ld
n
N
i_ld
i_count
0
i
end
=
LUT
Y_ld
y_0
S/Rout2
Y_mux
Y
*
δ_i
X_ld
x_0
S/Rout1
X_mux
X
*
δ_i
Z_ld
z_0
S/Rout1
Z_mux
Z
sign
Y
-sign
δ_i_ld
δ_i_mux
δ_i
MODE_ld
modeOper
MODE
MODE
X
Z
i
>>
*
δ_i
mux_op1
mux_op2
S/R 1
op_cod1
Y
i
>>
S/R 2
op_cod2

Estado	N_ld	i_ld	i_count	MODE_ld	Y_ld	Y_mux	X_ld	X_mux	Z_ld	Z_mux	δ_i_ld	δ_i_mux	mux_op1	mux_op2	op_cod1	op_cod2
S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S1	1	1	0	1	1	0	1	0	1	0	0	0	0	0	0	0
S2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
S4	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
S5	0	0	0	0	0	0	0	0	1	1	0	0	1	1	+	0
S6	0	0	1	0	1	1	1	1	0	0	0	0	0	0	−	+