1 Ogólnie o algorytmie CORDIC

Postanowiłem napisać ten artykuł, gdyż materiały, z który to rozgryzałem nie były najlepsze. Często nie podawały wprost założeń jakie trzeba spełnić. Jak te ograniczenia obejść itp. W dodatku w języku polskim praktycznie nie ma materiałów na ten temat. Dlatego bardzo było by mi miło gdybyś napisał mi, czy i jak Ci ten materiał pomógł. Teraz do rzeczy.

Algorytm CORDIC pozwala na obliczenie wartości kilku funkcji matematycznych bez wykonywania mnożenia

- sinus $f(x) = \sin x$
- kosinus $f(x) = \cos x$
- arkus tangens $f(x) = \operatorname{arctg} x$
- sinus hiberboliczny $f(x) = \sinh x$
- kosinus hiberboliczny $f(x) = \cosh x$
- area tangens hiberboliczny $f(x) = \operatorname{artgh} x$

Pozwala także na obliczenie mnożenia i dzielenia bez wykonywania go wprost. Istnieją też różne modyfikacje pozwalające ponoć liczyć $f(x) = \arcsin x$ oraz $f(x) = \arccos x$, $f(x) = \ln x$ i inne funkcje. Tego jednak nie sprawdzałem.

Algorytm CORDIC jest pewną alternatywą dla szeregów Taylora. Opiera się tylko o dodawanie, odejmowanie, przesuwanie bitowe i potrzebuje pamięci aby spamiętać kilkanaście liczb. W tym miejscu już widać, że pierwszym wymaganiem jest to aby liczby były w postaci bitowej. Co w praktyce w zasadzie zawsze jest spełnione. Nie ma znaczenia czy operujemy na liczbach stałoprzecinkowych czy zmiennoprzecinkowych.

Są jednak pewne ograniczenia co do zbieżności.

Nazwa funkcji	Przedział teoretyczny zbieżności	Sugerowane użycie
$\sin x, \cos x$	$x \in (-99,7^\circ; 99,7^\circ)$	$x \in (-\frac{\pi}{2}, \frac{\pi}{2})$
$\operatorname{arctg} x$	$x \in \mathbb{R}$	$x \in \mathbb{R}$
$a \cdot b$	$a \in \mathbb{R}, b \in [-2,2]$	$a, b \in (-2,2)$
$\frac{a}{b}$	$a \in \mathbb{R}, b \in \mathbb{R} \wedge \frac{a}{b} \in [-2,2]$	$a, b \in (-2,-1] \cup [1,2)$
$\sinh x, \cosh x$	$x \in (-1,11; 1,11)$	$x \in (-1,11; 1,11)$
$\operatorname{artgh} x$	$x \in (-1; 1)$	$x \in (-1; 1)$

Możliwości obliczeniowe po połączeniu poszczególnych wariantów algorytmu, skorzystaniu z tożsamości matematycznych lub innych modyfikacjach.

Nazwa funkcji	Przedział rozszerzony	W jaki sposób
$\sin x, \cos x$	$x \in \mathbb{R}$	trygonometryczne wzory redukcyjne
$a \cdot b$	$a, b \in \mathbb{R}$	wykorzystanie własności liczby zmiennoprzecinkowej
$\frac{a}{b}$	$a, b \in \mathbb{R} \wedge b \neq 0$	wykorzystanie własności liczby zmiennoprzecinkowej
$\sinh x, \cosh x$	$x \in \mathbb{R}$	rekurencyjne wykorzystanie tożsamości i mnożenia $\sinh 2x = 2 \sinh x \cosh x$ $\cosh 2x = \cosh^2 x + \sinh^2 x$

Dla funkcji sinus i kosinus szczególne kąty typu: $0^\circ, 90^\circ, 180^\circ, 270^\circ$ wykryć i potraktować oddzielnie.

Algorytm oblicza funkcję sinus i kosinus jednocześnie. Jest zatem szczególnie przydatny podczas obliczań obrotów na płaszczyźnie lub w przestrzeni trójwymiarowej. Nie jestem pewny, ale prawdopodobnie w kartach graficznych może być stosowany. Ponadto może być bardzo użyteczny w sprzęcie, w którym nie mamy do dyspozycji układów mnożących. Grupą szczególnie zainteresowanych będą programiści niskopoziomowi, elektronicy i mikroelektronicy.

Co ciekawe Algorytm CORDIC pozwala także na obliczanie mnożenia i dzielenia bez wykonywania mnożenia i dzielenia. Tu także są pewne ograniczenia, ale poprzez odpowiednie użycie można je obejść. Oczywiście dzięki temu możemy w sytuacjach koniecznych wykonać mnożenie nie wykonując mnożenia wprost.

W przypadku funkcji hiperbolicznych $\sinh x$ oraz $\cosh x$ przedział zbieżności jest ograniczony tylko do około $(-1,11; 1,11)$. Można podać skąd te wartości pochodzą, ale jest to skomplikowane i nieistotne, bo i tak będę proponował zmniejszenie tego przedziału do $[-1, 1]$. Wykorzystując tożsamości matematyczne można to rozszerzyć na większy przedział, potrzebne jednak będzie wykonywanie mnożenia, które może być realizowane inną wersją/instancją algorytmu, tą wykonującą mnożenie bez mnożenia.

Algorytm CORDIC jest algorytmem iteracyjnym. Cechuje się tym, że kolejna iteracja niekoniecznie musi dać dokładniejszy wyniki, niż poprzednia. Co więcej na poprawę "rekordu" możemy krótko lub długo czekać. Jednak w długiej perspektywie algorytm jest zbieżny. Problem jednak polega na tym, że aby uniknąć mnożenia trzeba z góry założyć ile tych iteracji wykonamy.

Istnieje możliwość obliczenia dodatkowych funkcji matematycznych pośrednio.

Funkcja matematyczna	Uwagi
$\operatorname{tg} x = \frac{\sin x}{\cos x}$	trzeba dzielić
$\operatorname{ctg} x = \frac{\cos x}{\sin x}$	trzeba dzielić
$\operatorname{arcctg} x = \frac{\pi}{2} - \operatorname{arctg} x$	tylko odejmowanie
$\operatorname{arctg}_2(y,x) = \begin{cases} \operatorname{arctg}\frac{y}{x} & , x > 0 \\ \operatorname{arctg}\frac{y}{x} + \pi & , x < 0 \wedge y \geqslant 0 \\ \operatorname{arctg}\frac{y}{x} - \pi & , x < 0 \wedge y < 0 \\ \frac{\pi}{2} & , x = 0 \wedge y > 0 \\ -\frac{\pi}{2} & , x = 0 \wedge y < 0 \\ \text{nieokreślone} & , x = 0 \wedge y = 0 \end{cases}$	dodawanie lub odejmowanie π
$\operatorname{tgh} x = \frac{\sinh x}{\cosh x}$	trzeba dzielić
$\operatorname{ctgh} x = \frac{\cosh x}{\sinh x}$	trzeba dzielić
$e^x = \sinh x + \cosh x$	tylko dodawanie, o ile mnożenia nie było w samym $\sinh x$ lub $\cosh x$
$\sqrt{x^2 + y^2}$	trzeba mnożyć
$\sqrt{x^2 - y^2}$	trzeba mnożyć
$\ln x = 2 \operatorname{artgh} \frac{x-1}{x+1}$	trzeba dzielić, zauważmy, że dla $x > 0 \Leftrightarrow \frac{x-1}{x+1} \in (-1, 1)$
$\log_a x = \frac{\ln x}{\ln a}$	trzeba dzielić
$a^x = e^{x \ln a}$	trzeba mnożyć
$\sqrt{x} = \sqrt{\left(x + \frac{1}{4}\right)^2 - \left(x - \frac{1}{4}\right)^2}$	trzeba mnożyć

Skupmy się teraz tylko nad zasadą działania w trybie rotacyjnym. Pozwalającym obliczać sinus i kosinus. Tę zasadę zmodyfikujemy nieco w trybie wektorowym, która pozwoli nam obliczyć $\operatorname{arctg} x$

2 Algorytm CORDIC w trybie rotacyjnym (rotation mode)

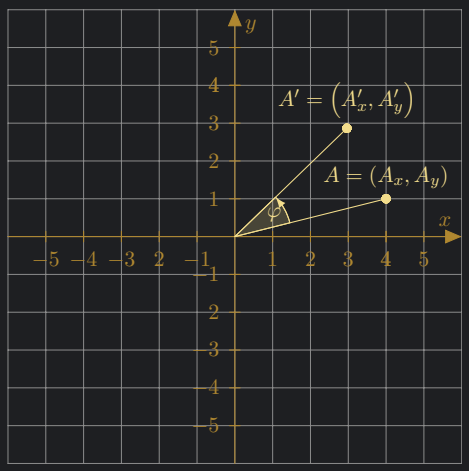
2.1 Obrót na płaszczyźnie o kąt φ przy nieruchomym układzie współrzędnych

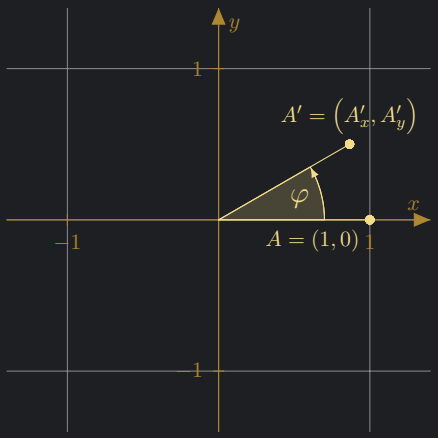
Przypomnijmy, że obrót na płaszczyźnie można zrealizować przy pomocy macierzy obrotu.

$$\begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}$$

Chcąc obrócić punkt $A = (A_x, A_y)$ o kąt $\varphi \in \mathbb{R}$ w nieruchomym układzie współrzędnych, należy

$$\begin{bmatrix} A'_x \\ A'_y \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \cdot \begin{bmatrix} A_x \\ A_y \end{bmatrix}$$



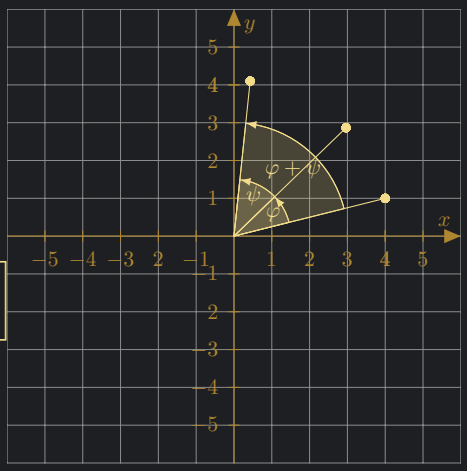


Zauważmy, że jeśli obracalibyśmy punkt o współrzędnych $A = (1, 0)$, to w wyniku obrotu otrzymamy punkt o współrzędnych $A' = (\cos\varphi, \sin\varphi)$.

$$\begin{bmatrix} \cos\varphi \\ \sin\varphi \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Wykonując kolejno dwa obroty, najpierw o kąt φ, a potem o kąt ψ skutek będzie taki sam jak wykonanie obrotu od razu o kąt $\varphi + \psi$. Potwierdza to też rachunek macierzowy.

$$\begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \cdot \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} = \begin{bmatrix} \cos(\varphi+\psi) & -\sin(\varphi+\psi) \\ \sin(\varphi+\psi) & \cos(\varphi+\psi) \end{bmatrix}$$



2.2 Przekształcenie

Wróćmy do dowolnego punktu $A = (A_x, A_y)$. Efekt po obrocie zapiszmy w postaci układu równań.

$$\begin{cases} A'_x = A_x \cos\varphi - A_y \sin\varphi \\ A'_y = A_x \sin\varphi + A_y \cos\varphi \end{cases}$$

Jeżeli przyjmiemy, że wykonujemy obroty w ramach kątów $\varphi \in (-\frac{\pi}{2}, \frac{\pi}{2})$, to wówczas $\cos\varphi \neq 0$. W takim razie można w prawych stronach wyciągnąć przed nawias $\cos\varphi$, a właściwie to za nawias.

$$\begin{cases} A'_x = A_x \cos\varphi - A_y \sin\varphi \\ A'_y = A_x \sin\varphi + A_y \cos\varphi \end{cases} \quad \Rightarrow \quad \begin{cases} A'_x = (A_x - A_y \operatorname{tg}\varphi)\cos\varphi \\ A'_y = (A_y + A_x \operatorname{tg}\varphi)\cos\varphi \end{cases}$$

Zauważmy, że skoro $\varphi \in (-\frac{\pi}{2}, \frac{\pi}{2})$, to zachodzi tożsamość

$$\frac{1}{\sqrt{1+\operatorname{tg}^2\varphi}} = \frac{1}{\sqrt{\frac{\cos^2\varphi+\sin^2\varphi}{\cos^2\varphi}}} = \frac{1}{\frac{1}{|\cos\varphi|}} = |\cos\varphi| = \cos\varphi$$

$$\begin{cases} A'_x = (A_x - A_y \operatorname{tg}\varphi)\cos\varphi \\ A'_y = (A_y + A_x \operatorname{tg}\varphi)\cos\varphi \end{cases} \quad \Rightarrow \quad \begin{cases} A'_x = (A_x - A_y \operatorname{tg}\varphi)\frac{1}{\sqrt{1+\operatorname{tg}^2\varphi}} \\ A'_y = (A_y + A_x \operatorname{tg}\varphi)\frac{1}{\sqrt{1+\operatorname{tg}^2\varphi}} \end{cases}$$

2.3 Obrót jako suma mniejszych obrotów

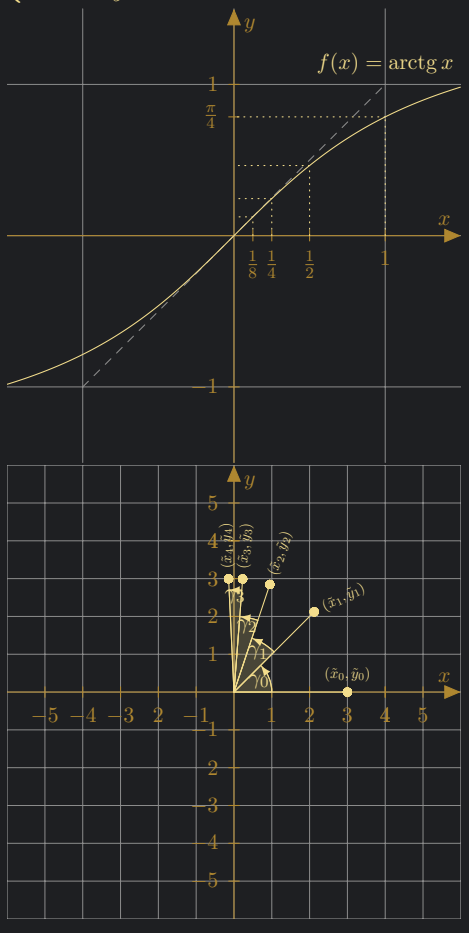
Jeżeli kąt φ zapiszemy jako sumę n innych kątów, tzn.

$$\varphi = \varphi_0 + \varphi_1 + \varphi_2 + \ldots + \varphi_{n-1}$$

To wówczas można zapisać następujący układ rekurencji, którego każda iteracja odpowiada za obrót punktu $(\tilde{x}_k, \tilde{y}_k)$ o kąt φ_k.

$$\begin{cases} \tilde{x}_{k+1} = (\tilde{x}_k - \tilde{y}_k \operatorname{tg}\varphi_k)\frac{1}{\sqrt{1+\operatorname{tg}^2\varphi_k}} \\ \tilde{y}_{k+1} = (\tilde{y}_k + \tilde{x}_k \operatorname{tg}\varphi_k)\frac{1}{\sqrt{1+\operatorname{tg}^2\varphi_k}} \\ \tilde{x}_0 = A_x \\ \tilde{y}_0 = A_y \end{cases}$$

Po n krokach otrzymujemy dobre przybliżenie obrazu punktu A czyli punkt $A' \approx (\tilde{x}_n, \tilde{y}_n)$. Zauważmy, że funkcja arctg jest funkcją nieparzystą, tzn. $\operatorname{arctg}(-x) = -\operatorname{arctg} x$. Zatem jeśli znamy arctg dla pewnego x to znamy także arctg dla $-x$. Wystarczy zmienić znak arctg na przeciwny. Zastanówmy się nad zbiorem takich kątów γ_k, które dodając lub odejmując mogłyby zbliżyć się dowolnie blisko zadanemu kątowi φ przy odpowiednio dużej liczbie kroków.



Widzimy, że w każdej iteracji trzeba wykonać mnożenie $\tilde{y}_k$ z $\operatorname{tg}\varphi_k$. Chcemy uniknąć mnożenia. Jedyne na co się godzimy to mnożenie przez potęgę 2 tzn. przez 2^m, gdzie $m \in \mathbb{Z}$. Mnożenie liczby a w systemie dwójkowym przez 2^m odpowiada przesuwaniu bitów liczby a w lewo lub w prawo. Jeśli m jest dodatnie to wówczas przesuwamy bity w lewo o m pozycji. Jeśli m jest ujemne wówczas przesuwamy bity w prawo o $-m$ pozycji. Niech zatem to będzie taki zbiór kątów, dla których $\operatorname{tg}\gamma_k$ będzie potęgą 2. Rozpatrzmy zbiór kątów dla, których $\operatorname{tg}\gamma_k = 2^{-k}$, gdzie k jest liczba naturalną. Matematycznie taki zbiór można zapisać tak:

$$\left\{\gamma_k \in \left(0, \frac{\pi}{4}\right] : \gamma_k = \operatorname{arctg} 2^{-k} \quad \wedge \quad k \in \mathbb{N}\right\}$$

Gdybyśmy z góry wcześniej wyznaczyli wartości tych kątów do pewnego niekoniecznie dużego n, np. $n = 40$ i je zapamiętali, to wówczas zamiast wykonywać kosztowne mnożenie przesuwalibyśmy bity liczby.

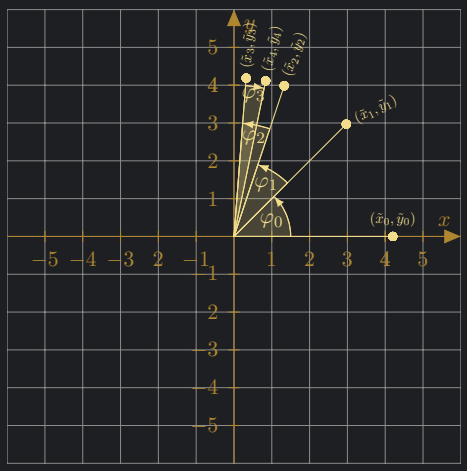
k	0	1	2	3	4	5	6	7	8	...
γ_k	arctg 1	arctg $\frac{1}{2}$	arctg $\frac{1}{4}$	arctg $\frac{1}{8}$	arctg $\frac{1}{16}$	arctg $\frac{1}{32}$	arctg $\frac{1}{64}$	arctg $\frac{1}{128}$	arctg $\frac{1}{256}$	...
γ_k	$45,0°$	$26,57°$	$14,04°$	$7,13°$	$3,58°$	$1,79°$	$0,9°$	$0,45°$	$0,22°$	...
γ_k	0,785	0,464	0,245	0,124	0,062	0,031	0,016	0,008	0,004	...

$$\varphi_k = \epsilon_k \gamma_k, \quad \text{gdzie} \quad \epsilon_k \in \{-1, 1\}$$

Należy się zastanowić czy każdy kąt z przedziału $\varphi \in (-\frac{\pi}{2}, \frac{\pi}{2})$ można przestawić jako sumę tych kątów, gdzie dany kąt może być wzięty z plusem lub z minusem. Przykładowo kąt

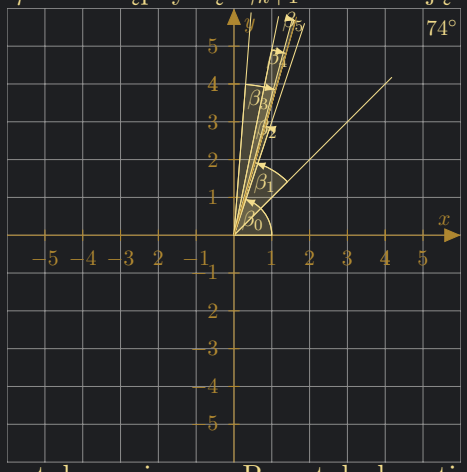
$$74° \approx \gamma_0 + \gamma_1 + \gamma_2 - \gamma_3 - \gamma_4 - \gamma_5 + \gamma_6$$

$$74° \approx 45,0° + 26,57° + 14,04° - 7,13° - 3,58° - 1,79° + 0,9°$$



Dla każdego kąta można znaleźć taki ciąg plusów i minusów jaki należy dołożyć do kątów γ_k aby ten ciąg dążył do zadanego kąta. Pozostawiam to bez dowodu.
Warto zaznaczyć, że w długiej perspektywie iteracji wyniki się poprawiają, lecz w krótkim zakresie parę kolejnych iteracji może dać gorszy wynik zanim trafimy na iterację, która wynik poprawi.

Pojawia się pytanie jak wybierać te znaki? Odpowiedź jest prostsza, niż się wydaje. Jak aktualnie naliczona suma kątów $\varphi_0 + \varphi_1 + \ldots + \varphi_k$ jest mniejsza od szukanego kąta φ to następny kąt γ_{k+1} trzeba dodać. Jak bieżąca suma jest większa od danego kąta φ to następny kąt γ_{k+1} trzeba odjąć. Tę zasadę można odwrócić i dany kąt φ z kroku na krok modyfikować, aż będzie dostatecznie blisko 0 lub gdy uznamy, że osiągnęliśmy maksymalną liczbę kroków n. Niech β_k oznacza kąt, po kroku k, jaki pozostał jeszcze do obrócenia, aby znaleźć się w punkcie A', będącym obrazem punkt A.



$$\begin{cases} \beta_{k+1} = \beta_k - \varphi_k \\ \beta_0 = \varphi \end{cases}$$

Gdzie po n krokach $\beta_n \approx 0$. Zauważmy także, że $\varphi_k = \epsilon_k \gamma_k = \operatorname{sgn}(\beta_k)\gamma_k$, a co za tym idzie

$$\operatorname{tg} \varphi_k = \operatorname{tg}(\epsilon_k \gamma_k) = \epsilon_k \operatorname{tg} \gamma_k = \operatorname{sgn}(\beta_k) \operatorname{tg} \gamma_k = \operatorname{sgn}(\beta_k) 2^{-k}$$

W ten sposób problem mnożenia $\tilde{y}_k \cdot \operatorname{tg} \varphi_k$ oraz $\tilde{x}_k \cdot \operatorname{tg} \varphi_k$ został rozwiązany. Pozostała kwestia mnożenia przez $\frac{1}{\sqrt{1+\operatorname{tg}^2 \varphi_k}}$.

2.4 Mnożenie przez $\frac{1}{\sqrt{1+\operatorname{tg}^2 \varphi_k}}$

Zaobserwujmy co się dzieje podczas dwóch kolejnych iteracji.

$$\begin{cases} \tilde{x}_1 = (\tilde{x}_0 - \tilde{y}_0 \operatorname{tg} \varphi_0) \frac{1}{\sqrt{1+\operatorname{tg}^2 \varphi_0}} \\ \tilde{y}_1 = (\tilde{y}_0 + \tilde{x}_0 \operatorname{tg} \varphi_0) \frac{1}{\sqrt{1+\operatorname{tg}^2 \varphi_0}} \end{cases} \Rightarrow \begin{cases} \tilde{x}_2 = (\tilde{x}_1 - \tilde{y}_1 \operatorname{tg} \varphi_1) \frac{1}{\sqrt{1+\operatorname{tg}^2 \varphi_1}} \\ \tilde{y}_2 = (\tilde{y}_1 + \tilde{x}_1 \operatorname{tg} \varphi_1) \frac{1}{\sqrt{1+\operatorname{tg}^2 \varphi_1}} \end{cases}$$

$$\begin{cases} \tilde{x}_2 = [(\tilde{x}_0 - \tilde{y}_0 \operatorname{tg} \varphi_0) - (\tilde{y}_0 + \tilde{x}_0 \operatorname{tg} \varphi_0) \operatorname{tg} \varphi_1] \frac{1}{\sqrt{1+\operatorname{tg}^2 \varphi_0}\sqrt{1+\operatorname{tg}^2 \varphi_1}} \\ \tilde{y}_2 = [(\tilde{y}_0 + \tilde{x}_0 \operatorname{tg} \varphi_0) + (\tilde{x}_0 - \tilde{y}_0 \operatorname{tg} \varphi_0) \operatorname{tg} \varphi_1] \frac{1}{\sqrt{1+\operatorname{tg}^2 \varphi_0}\sqrt{1+\operatorname{tg}^2 \varphi_1}} \end{cases}$$

Możemy zatem skupić się na rekurencji poniżej, w której po wykonaniu n iteracji otrzymamy przybliżenie obraz punkt A, czyli punkt A'.

$$\begin{cases} x_{k+1} = x_k - y_k \operatorname{tg} \varphi_k \\ y_{k+1} = y_k + x_k \operatorname{tg} \varphi_k \\ x_0 = A_x \\ y_0 = A_y \end{cases} \quad \rightarrow \quad \begin{cases} \tilde{x}_n = x_n \prod_{k=0}^{n-1} \frac{1}{\sqrt{1+\operatorname{tg}^2 \varphi_k}} \\ \tilde{y}_n = y_n \prod_{k=0}^{n-1} \frac{1}{\sqrt{1+\operatorname{tg}^2 \varphi_k}} \end{cases}$$

Zauważmy iż $\operatorname{tg}^2 \gamma_k = \operatorname{tg}^2 (-\gamma_k) = \operatorname{tg}^2 (\varphi_k)$, oznacza to, że wszystkie $\operatorname{tg}^2 \varphi_k$ są znane z góry, bo znane są z góry wartości kątów γ_k. Rozpatrzmy funkcję $K : \mathbb{N}^+ \to \mathbb{R}$, która w zasadzie jest ciągiem, którego pierwszy wyraz jest o indeksie 1. Ciąg jest dany wzorem

$$K_n = \prod_{k=0}^{n-1} \frac{1}{\sqrt{1+\operatorname{tg}^2 \varphi_k}} = \frac{1}{\sqrt{\prod_{k=0}^{n-1} (1+\operatorname{tg}^2 \varphi_k)}} = \frac{1}{\sqrt{\prod_{k=0}^{n-1} (1+2^{-2k})}}$$

Można policzyć jednokrotnie wyrazy tego ciągu i zapamiętać.

n	1	2	3	4	5	6	7	8	...
K_n	0,7071	0,6325	0,6135	0,6088	0,6076	0,6074	0,6073	0,6073	...

Zamiast obliczać rekurencję z $\tilde{x}_k$ i $\tilde{y}_k$ możemy obliczać rekurencję z x_k oraz y_k, a na koniec raz pomnożyć przez K_n. Wygląda na to, że jednego mnożenia trzeba będzie zatem dokonać.

Okazuje się, że jednak tego mnożenia także można uniknąć. Zauważmy, że zamiast mnożyć na końcu to można by mnożyć na początku. Wywnioskować to można spoglądając na rozpisane wcześniej 2 iteracje.

$$\begin{cases} x_{k+1} = x_k - y_k \operatorname{tg} \varphi_k \\ y_{k+1} = y_k + x_k \operatorname{tg} \varphi_k \\ x_0 = \tilde{x}_0 = A_x \\ y_0 = \tilde{y}_0 = A_y \\ \begin{cases} \tilde{x}_n = x_n K_n \\ \tilde{y}_n = y_n K_n \end{cases} \end{cases} \quad \rightarrow \quad \begin{cases} x_{k+1} = x_k - y_k \operatorname{tg} \varphi_k \\ y_{k+1} = y_k + x_k \operatorname{tg} \varphi_k \\ x_0 = \tilde{x}_0 = A_x K_n \\ y_0 = \tilde{y}_0 = A_y K_n \\ \begin{cases} \tilde{x}_n = x_n \\ \tilde{y}_n = y_n \end{cases} \end{cases}$$

Możemy przestać skupiać się na zmiennych akcentowanych tyldą $\tilde{x}_k$ oraz $\tilde{y}_k$, a skupić na zmiennych nieakcentowanych tyldą, czyli x_k oraz y_k. Jak już było wspomniane $\operatorname{tg} \varphi_k = \operatorname{sgn}(\beta_k) 2^{-k}$

$$\begin{cases} x_{k+1} = x_k - y_k \operatorname{tg} \varphi_k \\ y_{k+1} = y_k + x_k \operatorname{tg} \varphi_k \\ \beta_{k+1} = \beta_k - \operatorname{sgn}(\beta_k) \gamma_k \\ x_0 = A_x K_n \\ y_0 = A_y K_n \\ \beta_0 = \varphi \end{cases} \quad \rightarrow \quad \begin{cases} x_{k+1} = x_k - \operatorname{sgn}(\beta_k) y_k 2^{-k} \\ y_{k+1} = y_k + \operatorname{sgn}(\beta_k) x_k 2^{-k} \\ \beta_{k+1} = \beta_k - \operatorname{sgn}(\beta_k) \gamma_k \\ x_0 = A_x K_n \\ y_0 = A_y K_n \\ \beta_0 = \varphi \end{cases}$$

2.5 Użycie algorytmu w celu obliczenia $\sin \varphi$ oraz $\cos \varphi$

Teraz tak jak powiedziane było na początku. Jeśli punkt $A = (A_x, A_y)$ będzie punktem $A = (1, 0)$ to w wyniku otrzymamy punkt o współrzędnych $A' = (\cos \varphi, \sin \varphi)$. Decydując się na konkretną liczbę iteracji np. $n = 40$ z góry wiadomo, że

$$\begin{cases} x_0 = K_n \\ y_0 = 0 \\ \beta_0 = \varphi \end{cases} .$$

Ograniczeniem jest to, że z góry na starcie musimy zdecydować ile zrobimy iteracji tego algorytmu. W ten sposób utworzyliśmy równania rekurencyjne, które pozwalają jednocześnie obliczyć wartość funkcji sinus i kosinus dla dowolnego kąta. Wzorami redukcyjnymi sprowadzamy dowolny kąt do kąta z przedziału $(-\frac{\pi}{2}, \frac{\pi}{2})$, a następnie wykonujemy n iteracji poniższego układu.

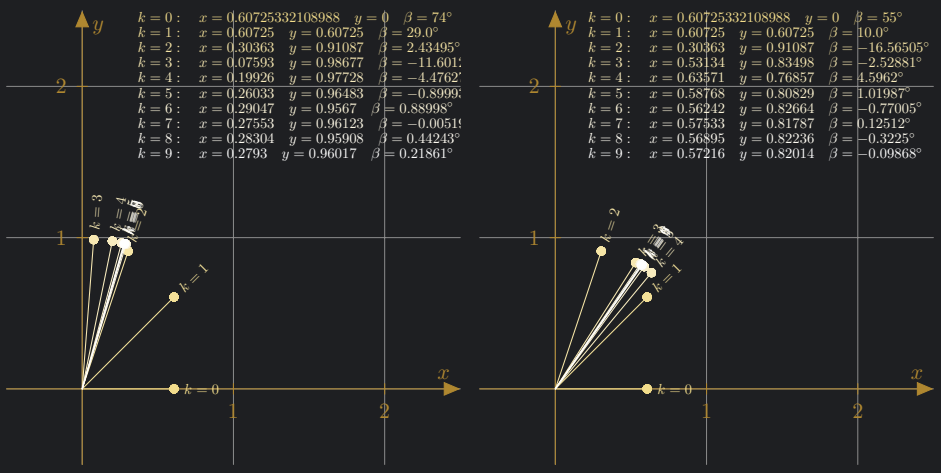
$$\begin{cases} x_{k+1} = x_k - \operatorname{sgn}(\beta_k) y_k 2^{-k} \\ y_{k+1} = y_k + \operatorname{sgn}(\beta_k) x_k 2^{-k} \\ \beta_{k+1} = \beta_k - \operatorname{sgn}(\beta_k) \gamma_k \\ x_0 = K_n \\ y_0 = 0 \\ \beta_0 = \varphi \end{cases}$$

Przybliżone wartości funkcji sinus i kosinus to

$$\begin{cases} \cos\varphi \approx x_k \\ \sin\varphi \approx y_k \end{cases}$$

2.6 Przykłady trybu rotacyjnego

Poniżej zwizualizowane i rozpisane są dwa przykłady. Jeden dotyczy obliczania $\sin 74°$ oraz $\cos 74°$, a drugi obliczenia $\sin 55°$ oraz $\cos 55°$



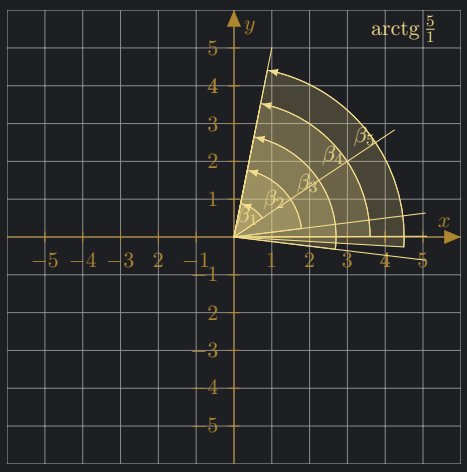
3 Algorytm CORDIC w trybie wektorowym (vectoring mode)

W trybie rotacyjnym zadany punkt $A = (1, 0)$ był obracany kolejno przez kąty φ_k, dla $k = 0, 1, \ldots, n-1$, aż osiągnięty zostanie punkt $A' = (\cos x, \sin x)$. W trybie wektorowym zaczynam od dowolnego punktu $A = (A_x, A_y)$, dla którego nie znamy kąta φ. Obracamy o kolejne kąty φ_k aż współrzędna y obrazu będzie w przybliżeniu równa $A'_y \approx 0$. Współrzędna x obrazu będzie wówczas równa $A'_x \approx \sqrt{A_x^2 + A_y^2}$, wynika to z twierdzenia Pitagorasa. Tryb wektorowy pozwala na obliczenie wartości funkcji arctg, a dokładniej to arctg $\frac{A_y}{A_x}$. Wyjdźmy od postaci

$$\begin{cases} x_{k+1} = x_k - y_k \operatorname{tg}\varphi_k \\ y_{k+1} = y_k + x_k \operatorname{tg}\varphi_k \\ x_0 = A_x K_n \\ y_0 = A_y K_n \end{cases}$$

Zauważmy, że dla kątów $\varphi \in (-\frac{\pi}{2}, \frac{\pi}{2})$, współrzędna $x_k > 0$ to o jaki kąt należy obrócić w danej iteracji zależy od znaku y_k. Kąt obrotu w danej iteracji to φ_k. Jeżeli y_k jest dodatnie to należy obrócić o kąt φ_k ujemny. Jak y_k jest ujemne to należy obrócić o kąt φ_k dodatni. Wynika nam z tego

$$\varphi_k = \epsilon_k \gamma_k = -\operatorname{sgn}(y_k)\gamma_k$$



Natomiast kąt β_k to będzie suma wykonanych obrotów φ_k z przeciwnym znakiem. Jest tak dla tego, że suma katów obrotu sprowadzenia punkt $A = (A_x, A_y)$ do punkt $A' = (A'_x, 0)$ jest kątem o przeciwnym zwrocie do kąta od dodatniej półosi osi x do promienia wodzącego A. Otrzymujemy następująca rekurencję

$$\begin{cases} \beta_{k+1} = \beta_k - \varphi_k \\ \beta_0 = 0 \end{cases}$$

Układ rekurencji dla tego algorytmu jest następujący

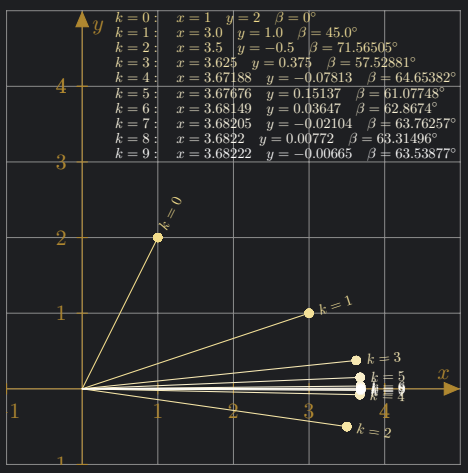
$$\begin{cases} x_{k+1} = x_k - y_k \operatorname{tg}\varphi_k \\ y_{k+1} = y_k + x_k \operatorname{tg}\varphi_k \\ \beta_{k+1} = \beta_k - \varphi_k \\ x_0 = A_x K_n \\ y_0 = A_y K_n \\ \beta_0 = 0 \end{cases} \rightarrow \begin{cases} x_{k+1} = x_k + \operatorname{sgn}(y_k) y_k \operatorname{tg}\gamma_k \\ y_{k+1} = y_k - \operatorname{sgn}(y_k) x_k \operatorname{tg}\gamma_k \\ \beta_{k+1} = \beta_k + \operatorname{sgn}(y_k)\gamma_k \\ x_0 = A_x K_n \\ y_0 = A_y K_n \\ \beta_0 = 0 \end{cases} \rightarrow \begin{cases} x_{k+1} = x_k + \operatorname{sgn}(y_k) y_k 2^{-k} \\ y_{k+1} = y_k - \operatorname{sgn}(y_k) x_k 2^{-k} \\ \beta_{k+1} = \beta_k + \operatorname{sgn}(y_k)\gamma_k \\ x_0 = A_x K_n \\ y_0 = A_y K_n \\ \beta_0 = 0 \end{cases}$$

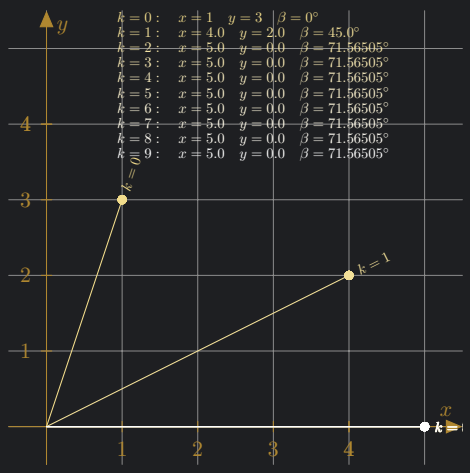
Pojawia się jednak problem bo skoro A_x i A_y jest dowolne to musimy wykonać mnożenie $A_x K_n$ oraz $A_y K_n$. W tym wypadku też jest na to sposób, aby uniknąć tego mnożenia. Mianowicie po prostu o nim zapomnijmy :-). Skutkować to będzie tylko tym, że współrzędna x obrazu będzie błędna, a dokładniej to $A' = \left(\frac{1}{K_n}\sqrt{A_x^2 + A_y^2}, 0\right)$. Zwykle przyjmujemy, że $A_x = 1$, ale nie jest to konieczne. Jeśli przyjmiemy $A_x = 1$, to wtedy będziemy obliczać $\operatorname{arctg} A_y$. Po n iteracjach odczytujemy znalezioną wartość funkcji arctg

$$\operatorname{arctg}\frac{A_y}{A_x} \approx x_n$$

3.1 Przykłady trybu wektorowego

Poniżej zwizualizowane i rozpisane są dwa przykłady. Pierwszy dotyczy obliczania $\operatorname{arctg} 2$, a drugi obliczenia $\operatorname{arctg} 3$.





3.2 Algorytm obliczania funkcji $f(x,y) = \sqrt{x^2+y^2}$

W trybie wektorowym istnieje także możliwość obliczenia pierwiastka kwadratowego. Jest jednak pewne ale, mianowicie nie można tego wykorzystać do obliczenia funkcji $g(x) = \sqrt{x}$. No chyba, że znamy rozkład x na a i b postaci $x = a^2 + b^2$. Zobaczmy, że nawet, gdyby przyjąć $x = 0$ w funkcji $f(x,y) = \sqrt{x^2+y^2}$, to $f(0,y) = \sqrt{0+y^2} = |y|$. Wychodzi na to, że znając y możesz policzyć jego wartość bezwzględna, czyli żaden z tego pożytek. Co więcej wymagało by to mnożenia na starcie, bo w tym przypadku nie możemy zignorować mnożenia K_n. Zignorowanie skutkowało by tym, że końcowy wynik byłby podzielony przez K_n.

W przypadku obliczania funkcji $f(x,y) = \sqrt{x^2+y^2}$, bez mnożenia się nie obejdziemy. Jeśli mnożenie jest dostępne to wtedy faktycznie możemy obliczać wartości funkcji $f(x,y) = \sqrt{x^2+y^2}$.

4 Podsumowanie - tryb rotacyjny i tryb wektorowy

Zestawiając ze sobą omówione dwa tryby, po chwili zauważamy, iż zarówno tryb rotacyjny jak i tryb wektorowy możemy zapisać wspólnie.

Tryb rotacyjny	Tryb wektorowy
$\begin{cases} x_{k+1} = x_k - \operatorname{sgn}(\beta_k) y_k 2^{-k} \\ y_{k+1} = y_k + \operatorname{sgn}(\beta_k) x_k 2^{-k} \\ \beta_{k+1} = \beta_k - \operatorname{sgn}(\beta_k)\gamma_k \\ x_0 = A_x K_n \\ y_0 = A_y K_n \\ \beta_0 = \varphi \end{cases}$	$\begin{cases} x_{k+1} = x_k + \operatorname{sgn}(y_k) y_k 2^{-k} \\ y_{k+1} = y_k - \operatorname{sgn}(y_k) x_k 2^{-k} \\ \beta_{k+1} = \beta_k + \operatorname{sgn}(y_k)\gamma_k \\ x_0 = A_x K_n \\ y_0 = A_y K_n \\ \beta_0 = 0 \end{cases}$

$$\begin{cases} x_{k+1} = x_k - d_k y_k 2^{-k} \\ y_{k+1} = y_k + d_k x_k 2^{-k} \\ \beta_{k+1} = \beta_k - d_k \gamma_k \end{cases}$$

Tryb rotacyjny

$$\begin{aligned} d_k &= \operatorname{sgn}(\beta_k) \\ x_0 &= K_n \\ y_0 &= 0 \\ \beta_0 &= \varphi \\ \cos\varphi &\approx x_n \\ \sin\varphi &\approx y_n \end{aligned}$$

Tryb wektorowy

$$\begin{aligned} d_k &= -\operatorname{sgn}(y_k) \\ x_0 &= 1 \\ y_0 &= A_y \\ \beta_0 &= 0 \\ \operatorname{arctg} A_y &\approx \beta_n \end{aligned}$$

W następnej części omówiony zostanie tryb liniowy.