International Journal of Scientific Research & Engineering Trends
Volume 11, Issue 1, Jan-Feb-2025, ISSN (Online): 2395-566X

MATLAB Implementation of Sine and Cosine Generator Using CORDIC Algorithm

Assistant Professor Mr. Goutam Barma,K. Chakradhar,J. Appa Rao,S. Abhishek
EEE, ACE Engineering College Hyderabad, Telangana, India.

Abstract- The CORDIC (Coordinate Rotation Digital Computer) algorithm is a versatile and efficient iterative method for computing a wide range of mathematical functions, including trigonometric, hyperbolic, exponential, logarithmic, and square root functions. Central to the CORDIC approach is its ability to perform vector rotations in a polar coordinate system, effectively transforming coordinates through a series of predefined angles. These methods eliminates the need for complex multiplications by utilizing simple shift and add operations, making it particularly well-suited for hardware implementations in resource-constrained environments, such as digital signal processors (DSPs) and field-programmable gate arrays (FPGAs).The algorithm operates in several modes, including rotation mode and vectoring mode, allowing it to adapt to various computational requirements. Each iteration reduces the angle by a fixed amount, using pre-computed arctangent values to guide the rotations. The convergence of the algorithm depends on the number of iterations, with higher iterations yielding greater accuracy. CORDIC's unique architecture supports parallel processing, enabling simultaneous calculations of multiple functions, further enhancing its efficiency. Evaluation of trigonometric functions such as sine, cosine and tan has been obtained using MATLAB. This abstract encapsulates the fundamental principles, operational modes, computational advantages, and diverse applications of the CORDIC algorithm, underscoring its significance in modern digital computation and system design.

Index Terms- CORDIC, MATLAB, DSP, FPGA, Trigonometric Functions

I. INTRODUCTION

CORDIC Algorithm

The Coordinate Rotation Digital Computer (CORDIC) is an iterative method for calculation of rotation in 2 dimensional vectors, linear, circular and hyperbolic coordinate system using add and shift operation. CORDIC firstly described in 1959 by Jack E. Volder for sophisticated solution to evaluate the trigonometric function. It was developed to replace the analog navigation computer on the B-58 planes bomber due to desire for high performance and accuracy.

In 1971, J. Walther extended CORDIC algorithm to hyperbolic functions and nowadays found in many applications such as digital signal processing, matrix computation, digital image processing, graphics, robotic and communication. Cochranv also reveals that CORDIC technique is much better especially for scientific calculating applications. Some of the famous applications are digital modulation, direct digital frequency synthesis, inverse and direct kinematics computation for robot manipulation and 3-dimension rotation for animation and graphic. Now-a-days world of information interchange revolves around transmission and viewing real-time-images. Many of the digital signal processing (DSP) applications try to closely simulate real life images. Speed, clarity, and resemblance to real time objects are some of the many issues to be addressed in order to achieve this goal. Trigonometric function calculation is one of the primary tasks performed in DSP applications.

MATLAB

MATLAB (Matrix Laboratory) is a high-performance programming language and interactive environment primarily designed for numerical computation, visualization, and programming. It is widely used in various fields such as engineering, finance, and scientific research due to its powerful capabilities for handling matrices, data analysis, algorithm development, and graphical representation. One of MATLAB's standout features is its optimization for matrix and array operations, allowing users to efficiently perform mathematical calculations. The environment includes a vast library of built-in functions covering areas like linear algebra, statistics, optimization, and signal processing, alongside robust data visualization tools for creating 2D and 3D plots.

General Applications of CORDIC Algorithm

Polar to Rectangular Transformation

A logical extension to sine and cosine computer is a polar to Cartesian coordinate transformer. The transformation from polar to Cartesian space is defined by,

x=rcosθ
y =rsinθ
As pointed out above, the multiplication by the magnitude comes for free using the CORDIC rotator. The transformation is accomplished by selecting the rotation mode with X0 = 0 polar phase and Y0 = 0.The vector result represents the polar input transformed to Cartesian space. The transform has a gain equal to the rotator gain, which needs to be accounted somewhere in the system. If the gain is unacceptable the polar magnitude may be multiplied by the reciprocal of the rotator gain before it is presented to the CORDIC rotator.

General Vector Rotation
The rotation mode CORDIC rotator is also useful for performing general vector rotations, as are often encountered in motion correction and control systems. For general rotation, the 2 dimensional input vector is presented to the rotator inputs. The rotator rotates the vector through the desired angle. The output is scaled by the CORDIC rotator gain, which must be accounted for elsewhere in the system. If the scaling is unacceptable, a pair of constant multipliers is required to compensate the gain.

ARCTANGENT
The arctangent, θ=Atan(y/x) is directly computed using the vectoring mode CORDIC rotator if the angle accumulator is initialized with zero. The argument must be provided as a ratio expressed as a vector(x,y).Since the arctangent result is taken from the angle accumulator the CORDIC rotator growth does not affect the result.

$$Zn = Z0 + \tan\text{-}1(Y0/X0)$$

The vectoring mode CORDIC rotator produces the magnitude of the input vector as a byproduct of computing the arctangent. After the vectoring mode rotation the vector is aligned with the x axis. The magnitude of the vector is therefore the same as the x component of the rotated vector. The magnitude result is scaled by the processor gain which needs to be accounted for elsewhere in the system. The CORDIC implementation represents a significant hardware savings over an equivalent Pythagoras processor. The accuracy of the magnitude result improves by 2 bits for each iteration performed.

II. FORMULA OF CORDIC ALGORITHM

This section would be discussing about the CORDIC computation and its basic principle. CORDIC is known as a simple algorithm to compute a lot of functions such as trigonometric that can be derived or computed from functions using vector rotations. The advantage of CORDIC algorithm is that it provides an iterative method in order to performing vector rotations by coordinate angles by using add and shift operations.

The CORDIC algorithm is classified as a linear convergence algorithm, requiring n-iterations for n bits of accuracy CORDIC algorithm has two types of computing modes Vector rotation and vector translation.

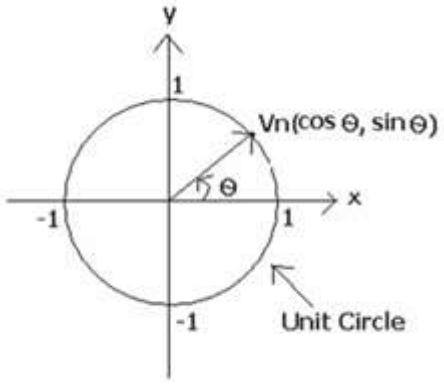

Fig 1 Unit cycle with vector, V_n

The CORDIC algorithm was initially designed to perform a vector rotation, where the vector V with components (X,Y) is rotated through the angle θ yielding a new vector V' with component (X',Y').

$$Vn=[R][V] \quad (1)$$

$$R = \begin{bmatrix} \cos\theta & -sin\theta \\ \sin\theta & cos\,\theta \end{bmatrix} \quad (2)$$

$$R = \begin{bmatrix} \frac{1}{\sqrt{1+tan^2\theta}} & \frac{-\tan\theta}{\sqrt{1+tan^2\theta}} \\ \frac{\tan\theta}{\sqrt{1+tan^2\theta}} & \frac{1}{\sqrt{1+tan^2\theta}} \end{bmatrix} \quad (3)$$

By factoring out the cosine term in (3), the rotation matrix R can be written as

$$R = \left[\frac{1}{\sqrt{1+tan^2\theta}}\right]\begin{bmatrix} 1 & -tan\theta \\ \tan\theta & 1 \end{bmatrix} \quad (4)$$

Interpreted as a product of a scale-factor

$K = \left[\frac{1}{\sqrt{1+tan^2\theta}}\right]$ With a pseudo rotation matrix, given by R_c

$$R_c = \begin{bmatrix} 1 & -tan\theta \\ \tan\theta & 1 \end{bmatrix} \quad (5)$$

$$X_{(i+1)} = K_i(X_i + 2^{-i}.Y_i)$$
$$Y_{(i+1)} = K_i(Y_i - 2^{-i}.X_i)$$

International Journal of Scientific Research & Engineering Trends
Volume 11, Issue 1, Jan-Feb-2025, ISSN (Online): 2395-566X

III. OPERATION OF CORDIC ALGORITHM

CORDIC (Coordinate Rotation Digital Computer) algorithm generally works in two steps which are described below

1. Rotation

In rotation mode, the angle accumulator is initialized with the desired rotation angle.

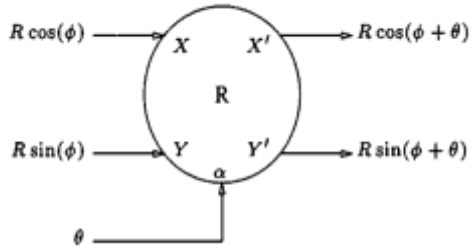

Fig 2 Rotation

The rotation decision at each iteration is made to diminish the magnitude of the residual angle in the angle accumulator. The decision at each iteration is therefore based on the sign of the residual angle after each step. Naturally, if the input angle is already expressed in the binary arctangent base, the angle accumulator may be eliminated. For rotation mode, the CORDIC equations are:

$$X_{i+1} = X_i - Y_i \times d_i \times 2^{-i}$$
$$Y_{i+1} = Y_i + X_i \times d_i \times 2^{-i}$$
$$Z_{i+1} = Z_i - d_i \times \tan^{-1}(2^{-i})$$

Where

$$d_i = -1 \text{ if } Z_i < 0, +1 \text{ otherwise}$$

which provides the following results

$$X_n = A_n[X_0 \cos Z_0 - Y_0 \sin Z_0]$$
$$X_n = A_n[Y_0 \cos Z_0 + X_0 \sin Z_0]$$
$$Z_n = 0$$
$$A_n = \prod_n \sqrt{1 + 2^{-2i}}$$

2. Vectoring

In the vectoring mode the CORDIC vector rotates the input vector through whatever angle is necessary to align the result vector with the x axis.

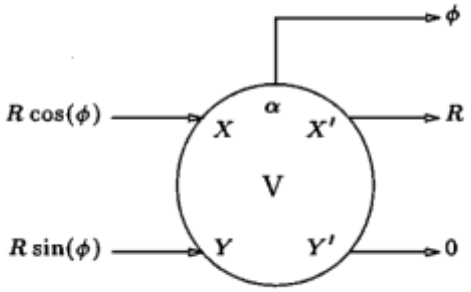

Fig 3 Vectoring

The result of the vectoring operation is a rotation angle and the scaled magnitude of the original vector (the x component of the result).

The vectoring function works by seeking to minimize they component of the residual vector at each rotation. The sign of the residually component is used to determine which direction is to rotate next. If the angle accumulator is initialized with zero, it will contain the traversed angle at the end of the iterations.

$$X_{i+1} = X_i - Y_i \times d_i \times 2^{-i}$$
$$Y_{i+1} = Y_i + X_i \times d_i \times 2^{-i}$$
$$Z_{i+1} = Z_i - d_i \times \tan^{-1}(2^{-i})$$

Where

$$d_i = +1 \text{ if } Y_i < 0, -1 \text{ otherwise}$$

Then:

$$X_n = A_n \times \sqrt{X_0^2 + Y_0^2}$$
$$Y_0 = 0$$
$$Z_n = Z_0 + \tan^{-1}(Y_0/X_0)$$
$$A_n = \prod_n \sqrt{1 + 2^{-2i}}$$

3.3 Iteration Table

The **iteration table** in the CORDIC algorithm is used to track the intermediate values of key parameters during each iteration. It helps in understanding the step-by-step convergence of the algorithm toward the desired result.

Table 1 Predefined Iterative values

i	Tan(α_i)	α_i(degrees)
0	1	45
1	0.5	26.5
2	0.25	14.03
3	0.125	7.125
4	0.0625	3.576
5	0.03125	1.7899
6	0.015625	0.895
7	0.0078125	0.4476
8	0.00390625	0.2238
9	0.001953125	0.1

International Journal of Scientific Research & Engineering Trends
Volume 11, Issue 1, Jan-Feb-2025, ISSN (Online): 2395-566X

IV. BLOCK DIAGRAM

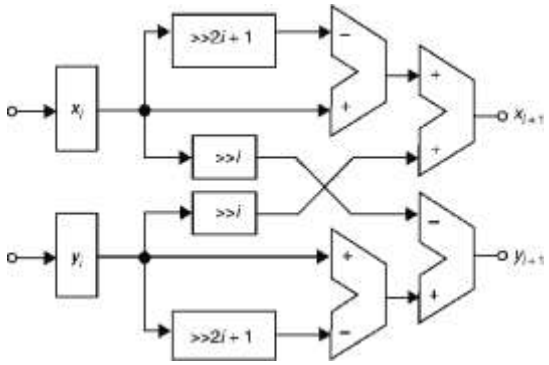

Fig 4 Block Diagram

Input Blocks (Xi, Yi)
In rotation mode, initialization starts with X0 = 1, Y0 = 0, and a given target angle θ. In vectoring mode, X0 and Y0 represent the initial vector coordinates. Iterative updates refine these values toward the desired result.

Shift Operators (>>2i+1, >>i)
Binary right shifts approximate multiplication or division by powers of two. The 2i+1 shift is used for scaling, while the ii shift corresponds to tan(2−i) avoiding explicit multiplication.

Adders/Subtractors
Coordinate updates follow:
Xi+1 = Xi − shifted(Yi)
Yi+1 = Yi + shifted(Xi)
Addition or subtraction is determined dynamically to ensure correct rotation or vectoring behaviour.

Cross Connections
X and Y values interact iteratively, maintaining geometric properties. Shifted Y values update Xi+1 while shifted X values update Yi+1, ensuring convergence.

Decision Logic
Rotation direction depends on the sign of the angle Zi: if Zi>0, rotation is clockwise; otherwise, counter clockwise. The angle update follows:
Zi+1 = Zi − precomputed angle(2−i).

Output Blocks (Xi+1, Yi+1)
In rotation mode, outputs converge to scaled cosθ and sinθ. In vectoring mode, Xi+1 approaches the vector's magnitude, while Zi converges to its angle.

V. CORDIC FLOWCHART

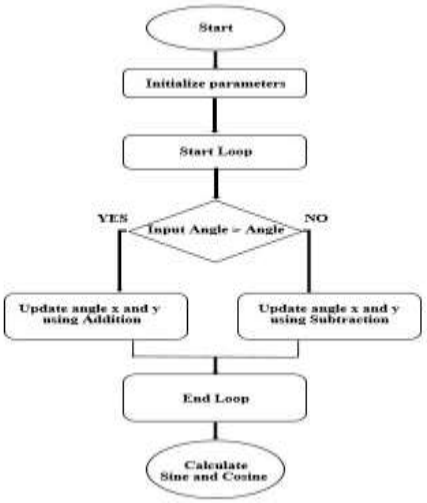

Fig 5 Flowchart describing CORDIC algorithm

Start
The algorithm begins execution.

Initialize Parameters
Initial values for XX, YY, and the target angle θ\theta are set. Precomputed angle values tan(2−i) are prepared for iterative updates.

Start Loop
Begins an iterative process to refine the values of X and Y.

Decision Block
Input Angle > Angle
Determines whether the current angle needs clockwise or counter-clockwise rotation.
YES: Rotate clockwise.
NO: Rotate counterclockwise.

Update X and Y Values
Depending on the decision:
Addition: Used for counterclockwise rotation.
Subtraction: Used for clockwise rotation.
The angle is also updated iteratively:
Zi+1 = Zi − precomputed angle

End Loop
Repeats until the desired precision is reached.

Calculate Sine and Cosine
After iterations, X approximates cosθ and Y approximates sinθ.

VI. PROGRAM

Below provided MATLAB code calculates the trigonometric values for sine and cosine for all values of theta ranging from 0o to 360o and whose magnitude is between 1 and -1.

```
% Initialize parameters
iter = 16; % Number of iterations
angles = 0:1:360; % Input angles from 0 to 360 degrees
% Pre-calculate rotation angles
for i = 1:iter
    th(i) = rad2deg(atan(2^(1-i)));
end
% Initialize output arrays for sine and cosine values
sine_values = zeros(size(angles));
cosine_values = zeros(size(angles));
% Loop over each angle in the range 0 to 360 degrees
for j = 1:length(angles)
    angle_in = angles(j);
    % Adjust angle to fit within -90 to +90 range for CORDIC
    if angle_in <= 90
        adjusted_angle = angle_in;
        sign_x = 1; sign_y = 1;
    elseif angle_in <= 180
        adjusted_angle = 180 - angle_in;
        sign_x = -1; sign_y = 1;
    elseif angle_in <= 270
        adjusted_angle = angle_in - 180;
        sign_x = -1; sign_y = -1;
    else
        adjusted_angle = 360 - angle_in;
        sign_x = 1; sign_y = -1;
    end
    % Initialize variables for CORDIC calculation
    ang = 0; % Current angle
     x = 0.607252935008881; % Initial x-coordinate (scaling factor for convergence)
    y = 0; % Initial y-coordinate
    % Main CORDIC loop
    for i = 1:iter
        if adjusted_angle > ang
            ang = ang + th(i);
            xnew = x - (y / (2^(i-1)));
            ynew = y + (x / (2^(i-1)));
        else
            ang = ang - th(i);
            xnew = x + (y / (2^(i-1)));
            ynew = y - (x / (2^(i-1)));
        end
        x = xnew;
        y = ynew;
    end
    % Apply quadrant signs and store sine and cosine values
    sine_values(j) = sign_y * y;
    cosine_values(j) = sign_x * x;
end
% Plotting the sine and cosine wave
figure;
subplot(2,1,1);
plot(angles, sine_values, 'b', 'LineWidth', 1.5);
title('Sine Wave using CORDIC algorithm');
xlabel('Angle (degrees)');
ylabel('Sine Value');
grid on;
subplot(2,1,2);
plot(angles, cosine_values, 'r', 'LineWidth', 1.5);
title('Cosine Wave using CORDIC algorithm');
xlabel('Angle (degrees)');
ylabel('Cosine Value');
grid on;
% Displaying sample values for sine and cosine at 30 degrees for reference
sine_30 = sine_values(angles == 30);
cosine_30 = cosine_values(angles == 30);
fprintf('Sine at 30 degrees using CORDIC: %.6f\n', sine_30);
fprintf('Cosine at 30 degrees using CORDIC: %.6f\n', cosine_30);
```

VII. RESULTS

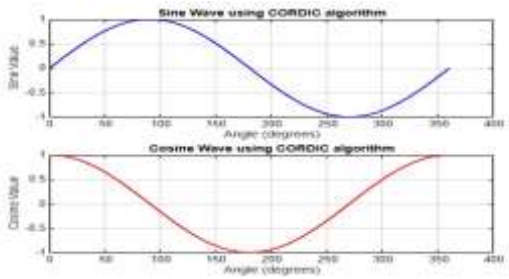

Figure 6 Output for theta varying from 0o to 400o

The given graphs illustrate the sine and cosine waveforms generated using the CORDIC algorithm in MATLAB. The top graph represents the sine wave (blue), while the bottom graph shows the cosine wave (red), both plotted against angles ranging from 0 to 400 degrees. The CORDIC algorithm efficiently computes these trigonometric functions using iterative shift-add operations, eliminating the need for multiplication or division. The smooth periodic nature of both waves confirms the algorithm's accuracy in approximating sine and cosine values. This approach is widely used in digital signal processing and hardware implementations where computational efficiency is crucial.

VIII. CONCLUSION

The CORDIC algorithm stands as a testament to the ingenuity of early computational methods, providing an efficient and versatile means of performing complex mathematical calculations with simple arithmetic operations. Its development by Jack Volder in the 1950s addressed the critical need for efficient trigonometric function computation in resource- constrained environments, a requirement that remains relevant today. CORDIC's ability to perform a wide range of functions, including trigonometric, hyperbolic, and logarithmic calculations, combined with its suitability for hardware implementation, makes it an invaluable tool in digital signal processing, computer graphics, robotics, and many other fields. Despite some limitations, such as its convergence range and precision in fixed-point implementations, the algorithm's low computational overhead and high efficiency continue to make it a preferred choice for real-time and embedded systems. As technology advances, the CORDIC algorithm remains a robust solution for efficient and precise mathematical computation, proving its enduring significance in the landscape of computational mathematics.

REFERENCES

1. Volder J. E., The CORDIC trigonometric computing technique, IRE Transactions on Electronic Computers. (1959) 8, no. 3, 330–334
2. Volder J. E., The birth of CORDIC, Journal of VLSI Signal Processing. (2000) 25, no. 2, 101–105, 2-s2.0-0342550186
3. Maharatna, Alfonso Troya, Swapna Banerjee, Eckhard Grass., Virtually scaling free adaptive CORDIC rotator, IEE Proc.-Comput. Digit. Tech., Vol. 151, No. 6, November 2004
4. Sambit Kumar, Dash Jasobanta Sahoo, Sunita Patel., CORDIC Algorithm and its Applications in DSP, National Institute of Technology Rourkela 2007
5. International Conference on Engineering and Technology (IntCET 2017) AIP Conf. Proc. 1930, 020040-1–020040-7; Published by AIP Publishing. 978-0-7354-1622-2
6. M/S Nivedita Tiharu, Mrs. Jigyasha Maru., A Novel Study of CORDIC Processor and Algorithm, Volume-7,Issue-4, (April-17) ISSN (O) :- 2349-3585