Fast computation of arctangent functions for embedded applications: A comparative analysis

Conference Paper · June 2011

DOI: 10.1109/ISIE.2011.5984330

3 authors:

Abhisek Ukil
University of Auckland
254 PUBLICATIONS **4,939** CITATIONS
SEE PROFILE

Vishal H Shah
ABB India limited
6 PUBLICATIONS **215** CITATIONS
SEE PROFILE

Bernhard Deck
Hitachi Power Grids
13 PUBLICATIONS **226** CITATIONS
SEE PROFILE

Fast Computation of *arctangent* Functions for Embedded Applications: A Comparative Analysis

Abhisek Ukil*, *Senior Member, IEEE*, Vishal H Shah†, Bernhard Deck‡
*ABB Corporate Research, Baden-Daettwil, Switzerland, Email: abhisek.ukil@ch.abb.com
†ABB Corporate Research, Bangalore, India, Email: vishal.h.shah@in.abb.com
‡ABB Sécheron S.A., MV, Baden, Switzerland, Email: bernhard.deck@ch.abb.com

***Abstract*— Inverse tangent or arctangent function has many applications, for example, in estimating the phase angle of complex number utilized in electrical ac circuit analysis, power systems analysis, etc. Therefore, fast and accurate computation of the arctangent function is of great importance. Implementation of the $arctan$ or $atan2$ using series expansions is accurate when computational cost is not an issue. However, for embedded applications such implementations cannot be used. As alternative, different approximations are proposed. In this paper, we present an efficient implementation using look-up table with 101-points. The accuracy is then increased by linear interpolation. The arctangent computation is extended to four quadrant operation using particular properties of $\arctan$. The proposed scheme is implemented in a 60MHz microcontroller platform, typical for embedded applications. Comparative evaluations show that the accuracy of the proposed method is better than the reported approximation techniques. The time consumption of the proposed method is significantly less than that of the library function of the same microcontroller platform.**

I. Introduction

Inverse tangent or arctangent or $arctan(x)$ is the multi-valued function $tan^{-1}(x)$ [1], defined for all real numbers. Fig. 1 shows the $tan^{-1}(x)$ along the x-axis, taking a single argument. The two-argument function $atan2$ is a variation of the arctangent function, which computes the principal value of the argument function applied to the complex number of form $x+iy$ [2]. The complex argument of a complex number $z = x + iy$ is often written as

$$\theta = tan^{-1}\left(\frac{y}{x}\right), \tag{1}$$

where, θ (sometimes also denoted as ϕ), corresponds to the counterclockwise angle from the positive real axis. The complex arguments are extensively used in ac circuit analysis, electrical power systems, etc. Therefore, for field devices like power systems relay, etc, it is often required to compute the phase angles by utilizing the inverse tangent functions. The inverse tangent functions are usually available in the standard libraries of C/C++ [3], MATLAB [4], Mathematica [5], Java [6], MS-Excel [7], etc. However, these library function implementations are not optimized for usage in the low-end field devices with limited computational resources like microcontrollers, DSPs of smaller bit-size, memory, clock-speed, etc. This paper describes an efficient implementation of the $arctan$ and $atan2$ functions for devices with limited computational power.

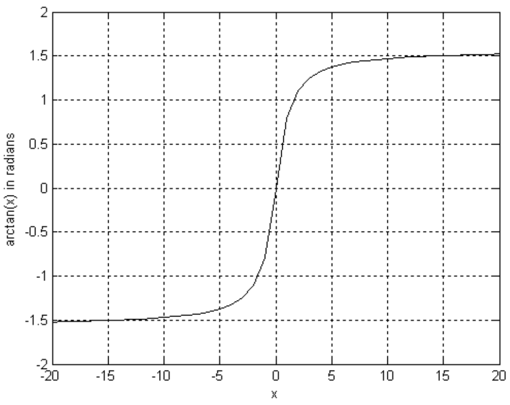


Fig. 1. Inverse tangent function.

II. Background Information

A. $arctan$, $atan2$

$arctan(x)$ has the following Maclaurin series expansion [1]

$$\begin{cases} \arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots \\ = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{2n+1}; |x| < 1, x \neq i, -i. \end{cases} \tag{2}$$

An efficient series expression for $arctan(x)$ is given by Euler [2]

$$\arctan(x) = \frac{x}{1+x^2} \sum_{n=0}^{\infty} \prod_{k=1}^{n} \frac{2kx^2}{(2k+1)(1+x^2)}. \tag{3}$$

$arctan(x)$ has the following interesting functional properties.

$$\arctan(-x) = -\arctan(x). \tag{4}$$

$$\arctan(x) + \arctan\left(\frac{1}{x}\right) = \frac{\pi}{2}, \qquad \text{if } x > 0. \tag{5}$$

$$\arctan(x) + \arctan\left(\frac{1}{x}\right) = -\frac{\pi}{2}, \qquad \text{if } x < 0. \tag{6}$$

The inverse tangent function also satisfies the addition formula

$$\arctan(x) + \arctan(y) = \arctan\left(\frac{x+y}{1-xy}\right). \tag{7}$$

The single-argument $arctan$ function does not distinguish between diametrically opposite directions. For any real arguments x and y (both not equal to zero), $atan2(y, x)$ is the angle in rad (radian) between the positive x-axis of a plane and the point given by the coordinates (x, y) on it. The angle is positive for counter-clockwise angles (upper half-plane, $y > 0$), and negative for clockwise angles (lower half-plane, $y < 0$).

The $atan2$ function was first introduced in computer programming languages, but now it is also common in other fields of science and engineering. For example, the anticlockwise angle from the x-axis to the vector $(1, 1)$, calculated in the usual way as $arctan(1/1)$, is $\pi/4$ rad, or 45 deg. However, the angle between the x-axis and the vector $(-1, -1)$ appears, by the same method, to be $arctan(1/1)$, again $\pi/4$ rad, even though the answer should be $-3\pi/4$ rad or -135 deg [1]. The $atan2$ function takes into account signs of both vector components. The definition of the $atan2$ is as follows

$$atan2(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & \text{if } x > 0 \\ \pi + \arctan\left(\frac{y}{x}\right) & \text{if } y \geq 0, x < 0 \\ -\pi + \arctan\left(\frac{y}{x}\right) & \text{if } y < 0, x < 0 \\ \frac{\pi}{2} & \text{if } y > 0, x = 0 \\ -\frac{\pi}{2} & \text{if } y < 0, x = 0 \\ \text{undefined} & \text{if } y = 0, x = 0. \end{cases} \quad (8)$$

This produces results in the range $(-\pi, \pi]$, which can be mapped to $[0, 2\pi)$ by adding 2π to the negative values. Traditionally, $atan2(0, 0)$ is undefined.

B. Utilization of the Functions

Electrical ac circuit analysis is a major domain, where inverse tangent functions are used for phase angle estimations. In the analysis of eletrical power systems, complex phasors [8] are used. Computation of the electrical phasors, often involve utilization of the discrete Fourier transform (DFT) method [9]. DFT of a sequence $x(n)$ (voltage, current, etc) has the form

$$X(\omega) = X_{real}(\omega) + jX_{imag}(\omega) = X_{mag}(\omega)\angle(X_{angle}(\omega)), \quad (9)$$

where, the rectangular form has the real and imaginary parts X_{real} and X_{imag}. In the polar form, the magnitude part is $X_{mag}(\omega) = \sqrt{(X_{real}(\omega)^2 + X_{imag}(\omega)^2}$, and the angle part is $X_{angle}(\omega) = tan^{-1}\left(\frac{X_{imag}(\omega)}{X_{real}(\omega)}\right)$. The angle part of the polar form often utilizes $atan2$ function. Phase angle estimation is a critical step in many applications.

C. Existing Implementations

Functional implementation of the inverse tangent functions is often based on the series expansion. Optimized BBP (Bailey-Borwein-Plouffe) type formulae can be referenced in [1]. $atan2$ uses function definition as in (8). These implementations can be referred in C/C++ [3], MATLAB [4], Mathematica [5], Java [6], Microsoft Excel [7], etc.

For systems without a hardware multiplier, the function $atan2$ can be implemented in a numerically reliable manner by the CORDIC (COordinate Rotation DIgital Computer) method [10],[11]. CORDIC is a simple and efficient algorithm to calculate hyperbolic and trigonometric functions, first described in 1959 by Jack E. Volder [10]. It is commonly used when no hardware multiplier is available (e.g., simple microcontrollers and FPGAs) as the only operations it requires are addition, subtraction, bit-shift and table lookup. CORDIC type implementations can be referred to in [12],[13],[14]. However, the CORDIC type algorithms are in general sequential in nature, making them rather slow. Therefore, when faster computation is required, e.g., in power system protection where we talk in the range of 10ms, these type of algorithms are not suited. The work in [15] reports increase in computation speed by adding more hardware.

For applications in embedded systems, arctangent could be implemented in look-up table (LUT), by including values in a read-only memory. Alternatively, series expansion (by Taylor series or Chebyshev polynomial) could be used to approximate the angle. Lyons [16] proposed a second-order approximation function

$$\arctan(x) \approx \frac{x}{1 + 0.28125x^2}, \qquad -1 \leq x \leq 1. \quad (10)$$

Some of the other efficient approximations discussed by Rajan and colleagues [17] are

$$\arctan(x) \approx \frac{\pi}{4}x + 0.273x(1 - |x|), \qquad -1 \leq x \leq 1. \quad (11)$$

$$\begin{cases} \arctan(x) \approx \frac{\pi}{4}x - x(|x| - 1)(0.2447 + 0.0663|x|), \\ -1 \leq x \leq 1. \end{cases} \quad (12)$$

III. Proposed Method

A. Algorithm

In the proposed implementation, we use a LUT-based approach. In the LUT, the function values (not all points) are precomputed and kept as entries in the table. Sometimes it is argued that this requires a lot of memory [16],[17]. To address that, we propose to keep realistic number of entries in the table, and increase the accuracy by linear interpolation.

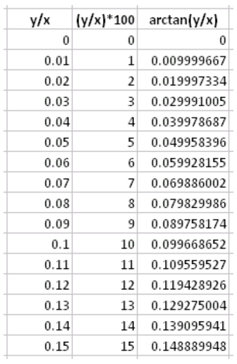
For the four quadrant implementation, we used the $atan2$ definition (see (8)). For the $arctan$ computation in the $atan2$ (see (8)), we used the properties in (4),(5) and (6). This would lead us to extend the computation of the $arctan$ function from the range 0 to 1 (i.e. 0 to $\frac{\pi}{4}$ rad), to the four quadrants.

Our function, like standard interface, has two arguments, $atan2(y, x)$. We created a table with 101 entries for the $arctan$ value for the ratio ($=\frac{y}{x}$) in the range 0 to 1, (i.e., in steps of 0.01). First 16 entries are shown in Table I. It is to be noted that only the third column of the Table I is stored as LUT for the computation. This will be explained below. The values stored in Table I are of type double as per IEEE format. Double type has a bit width of 64 bits.

In Table I, in the second column we keep the $\frac{y}{x}$ multiplied by 100, to represent them as positive integers from 0 to 100. Fig. 2 shows the variation of the $arctan$ function for the ratio ($\frac{y}{x}$). The x-axis corresponds to the second column in Table

TABLE I

FIRST 16 ENTRIES OF THE LOOK-UP TABLE FOR ARCTANGENT COMPUTATION



y/x	(y/x)*100	arctan(y/x)
0	0	0
0.01	1	0.009999667
0.02	2	0.019997334
0.03	3	0.029991005
0.04	4	0.039978687
0.05	5	0.049958396
0.06	6	0.059928155
0.07	7	0.069886002
0.08	8	0.079829986
0.09	9	0.089758174
0.1	10	0.099668652
0.11	11	0.109559527
0.12	12	0.119428926
0.13	13	0.129275004
0.14	14	0.139095941
0.15	15	0.148889948

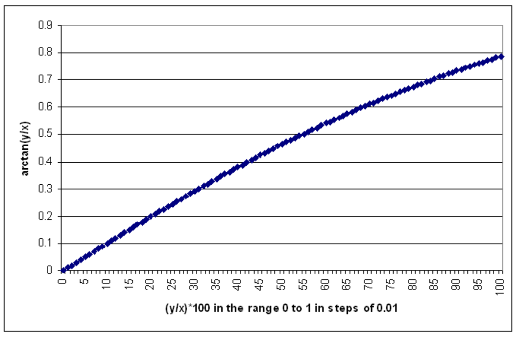


Fig. 2. $arctan(y/x)$ vs $100(y/x)$.

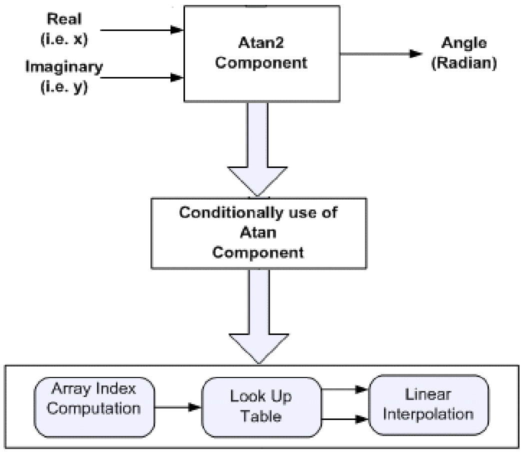


Fig. 3. Signal processing flow for atan2 function.

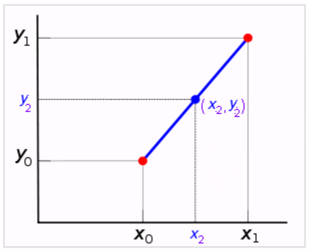


Fig. 4. Linear interpolation between two points.

I. From Fig. 2, we can see the function as a monotonically increasing one.

The computation algorithm is described in the signal processing flowchart shown in Fig. 3. For the computation of the $arctan$ of a real number, the three sequential blocks shown at the bottom of the Fig. 3 are used. The operations would be explained below.

For a real number($=\frac{y}{x}$) input in the range $< 0, 1 >$, we first multiply it by 100 to make it in the range $< 0, 100 >$, and then retrieve the nearest integer by rounding it. This integer added by one would give the index for the $arctan$ value in the LUT (third column in Table I). To care for the rounding and increase the accuracy, linear interpolation would be used, which will be explained below.

B. Linear Interpolation

The designed LUT is having resolution of 0.01 for y/x ratio. For better accuracy, the gaps in the LUT can be filled by utilizing the linear interpolation. It is a method of curve fitting using linear polynomials. From Fig. 2, as the curve is a monotonically increasing one, linear interpolation will be suitable, assuming straight line between two adjacent points. Mathematically, for two known points given by the co-ordinates (x_0, y_0) and (x_1, y_1), the linear interpolate is the straight line between the two points, described by (13), shown in Fig. 4.

$$\frac{y - y_0}{x - x_0} = \frac{y_1 - y_0}{x_1 - x_0}. \tag{13}$$

Using (13), for a value x_2 in the interval (x_0, x_1), the value y_2 along the straight line is given by

$$y_2 = y_0 + (x_2 - x_0)\frac{(y_1 - y_0)}{(x_1 - x_0)}. \tag{14}$$

In our case, in (14), $(x_1 - x_0)$ is the difference between two adjacent indices, i.e., 1. x_2 and x_0 are the real number input and the nearest integer respectively. y_0 is the LUT value at index x_0 (x_0+1 if we reference the table entries starting from 1 instead of 0). y_1 is the next value to y_0 in the LUT. We have to care for the case when y_0 is the last entry in the LUT, then $y_1 - y_0 = 0$. So, for $arctan$ computation, we need

$$\begin{cases} \widehat{input} = round(input \times 100), \\ index = \widehat{input} + 1, \\ angle_{new} = LUT(index) + \\ (input \times 100 - \widehat{input})(LUT(index + 1) - LUT(index)). \end{cases} \tag{15}$$

IV. RESULTS

A. Example Computation: arctan

First we show an example of computation using the proposed method. Let's say, we want to compute $arctan(0.025)$.

Using (15), the computation is as follows.
$\widehat{input} = round(0.025 \times 100) = 2$.
$index = \widehat{input} + 1 = 3$.
The angle value from the third column in Table I for this $\widehat{input} = 2$ (see second column in Table I) would be from row number 3. As Table I starts from 0, the indices would be forwarded by 1 from the $\widehat{input}$ value, given by $index$. The exception would be when $\widehat{input} = 100$, which will then be the last entry in the table. Therefore, for $\widehat{input} = 2$ (nearest integer part), $LUT(index) = LUT(3) = 0.019997334$ (see third column in Table I). Please note, this index forwarding by 1 is for better explanation. In 'C' implementation, as the array indexing starts from 0 (being the first entry), we would not need $index = \widehat{input} + 1$. Instead, the array index for the angle value corresponding to the $\widehat{input}$ would be given by itself. That's why, it is flexible to design a 101-point table (one-dimensional array in 'C') of angle values, where the indices correspond to the rounded input multiplied by 100, covering uniformly the range 0 to 1.

For integer 3, the angle value would come from row number 4 of the third column in Table I, i.e., $LUT(index + 1) = 0.029991005$. Please note $0.025 \times 100 = 2.5$ lies between 2 and 3. Applying (15), we get

$angle_{new} = 0.019997334 + (0.025 \times 100 - 2)(0.029991005 - 0.019997334) = 0.0249941695$ rad.

Using MATLAB [4], $atan(0.025) = 0.0249947936$ rad.

So, the error=6.24×10^{-7} rad.

The errors using different approximations in (10)-(12), with respect to the MATLAB value are mentioned below.

Using (10),
$angle = 0.0249956062$ rad,
error=-8.13×10^{-7} rad.

Using (11),
$angle = 0.0262893291$ rad,
error=-12.94×10^{-4} rad.

Using (12),
$angle = 0.0256399181$ rad,
error=-6.45×10^{-4} rad.

B. Example Computation: atan2

Here, we show an example of computation of $atan2$ using the proposed method. Let's say, we want to compute $atan2(40, -1)$. This means, $(y/x) = -40$.

As $(\frac{y}{x} < 0)$, using the property of $arctan$ as in (6),

$arctan(-40) = -\frac{\pi}{2} - arctan(-1/40) = -\frac{\pi}{2} - arctan(-0.025)$.

Using the property in (4),

$arctan(-40) = -\frac{\pi}{2} - arctan(-0.025) = -\frac{\pi}{2} + arctan(0.025)$.

Computation of $arctan(0.025) = 0.0249941695$ rad (using the LUT-based approach) is demonstrated in the previous section. So,

$arctan(-40) = -\frac{\pi}{2} + 0.0249941695 = -1.54580215729490$ rad.

Now, as $y = 40, x = -1$, using the $atan2$ definition in (8),

$atan2(40, -1) = \pi + arctan(-40) = \pi - 1.54580215729490 = 1.59579049629490$ rad.

Using MATLAB [4], $atan2(40, -1) = 1.59579112041382$ rad.
So, the error=6.24×10^{-7} rad.

C. Accuracy

The example computations in the previous sections show better accuracy using the proposed method over the other approximations [16],[17]. The analysis also shows that the accuracy of the $atan2$ computation is determined by the accuracy of the $arctan$ computation. Moreover, using the properties of $arctan$ shown in (4) - (6), and the $atan2$ definition in (8), it is sufficient to compute the values in the range $< 0, 1 >$, which would then cover all possible values. It is also interesting to note that the accuracy of the $arctan$ computation in the range $< -1, 0 >$ would be of inverse nature to the accuracy in the range $< 0, 1 >$.

The algorithm was implemented in the Keil MCB2130 evaluation board [18], with ARM7TDMI processor, having clock-speed of 60MHz [19]. The microcontroller development environment has μVision IDE, debugger and simulation environment, ARM C,C++ compiler from ARM [20].

The angle estimation error of the $arctan$ over the range $< -1, 1 >$ is shown in Fig. 5. The error is computed by comparing the estimated values using the proposed algorithm in the evaluation board, against values computed using MATLAB in a PC. It is already discussed that accuracy of $atan2$ computation follows that of $arctan$, and it suffices to compute the accuracy in the range $< 0, 1 >$ to cover all range of values. Nevertheless, the accuracy is shown over the range $< -1, 1 >$ to make a fair comparison with the previously reported results [16],[17].

The error function in Fig. 5 is antisymmetric with respect to the x-axis, evident from the $arctan$ property in (4). The maximum error for the proposed method, as shown in Fig. 5, is about 2.42×10^{-5} rad, for the input $x = -0.625$. In comparison, for similar input range, the maximum error for the approach in (10) is 4.5×10^{-3} rad [16]. The maximum error for the approach in (11) is 3.8×10^{-3} rad [17], and for the approach in (12) is 1.5×10^{-3} rad [17]. Therefore, the proposed method, using the 101-point LUT, is significantly more accurate than previously reported methods.

If we select every second entry in Table I, to make a 51-point LUT, the maximum angle error increases to 22×10^{-3} rad, as shown in Fig. 6.

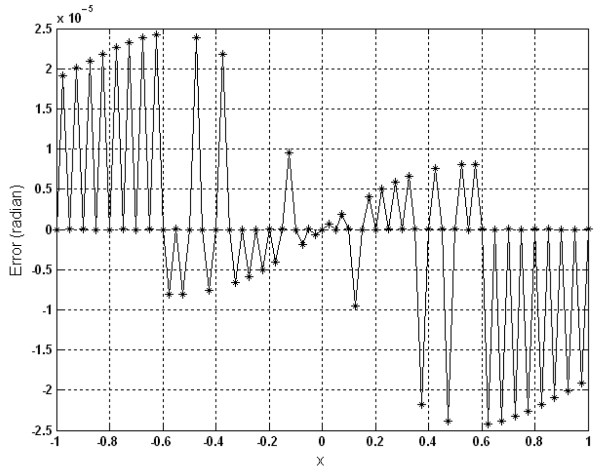


Fig. 5. Error of angle estimation for 101-point LUT.

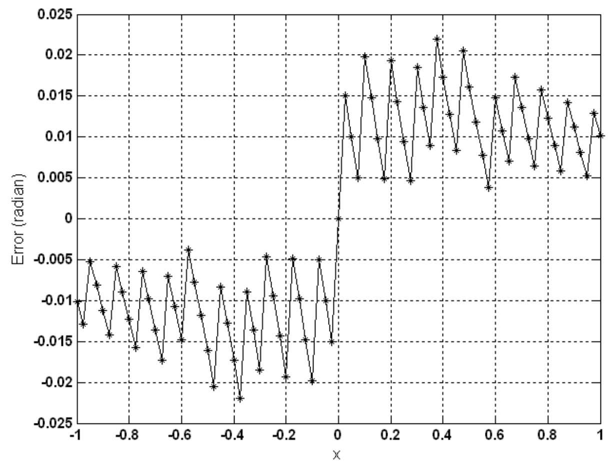


Fig. 6. Error of angle estimation for 51-point LUT.

D. Computation Time

The time consumption of the proposed method, using the 101-point LUT, (extended to four quadrant operation) was judged against the "math.h" library $atan2$ function for the four quadrant operation in the MDK-ARM IDE [20]. A LeCroy oscilloscope was used to estimate the function time consumption running in the aforesaid evaluation board. Fig. 7 shows that the $atan2$ library function in the the MDK-ARM IDE [20], which takes about 75μs. In comparison, Fig. 8 shows that the proposed LUT-based implementation of the $atan2$ function consumes about 35μs. The proposed implementation thus reduces the time consumption by 53%, against a library function implementation, expectedly highly optimized. The processor task load was same for both timing measurements.

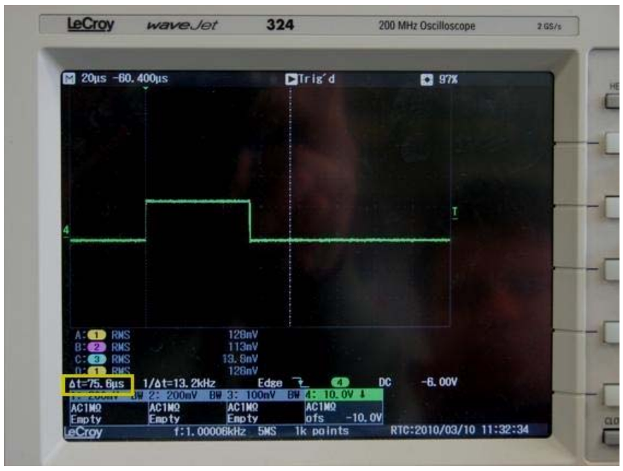


Fig. 7. Time consumption of the $atan2$ library function: 75μs.

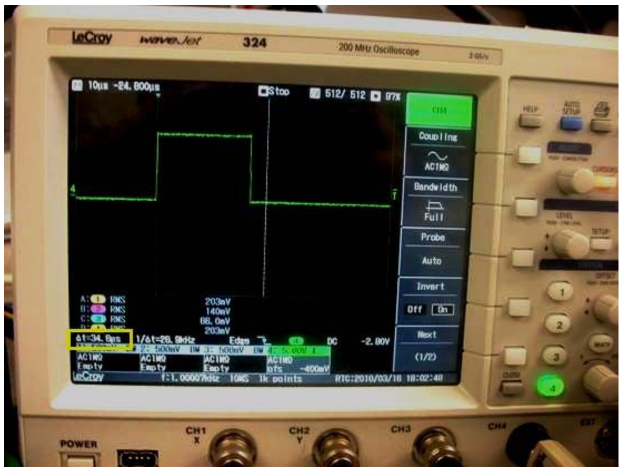


Fig. 8. Time consumption of the proposed implementation of the $atan2$ function: 35μs.

V. Discussion of Results

1) The time-consumptions (75μs and 35μs) are specific to the Keil MCB2130 evaluation board [18]. It might change for a different platform. However, the intention of such comparison is to demonstrate that the LUT-based implementation would expectedly run faster than even optimized library functions of typical microcontrollers for embedded applications.
2) The reduction of the time-consumption are particularly important for real-time applications. For example, in typical power systems protection [9] relays, the sampling frequency is about 2-3kHz, resulting in about 500-333μs between arrival of two samples. Usually, all the protection function computations have to be performed before the next sample arrives. And for three-phase

operations, often there might be multiple calls for the arctangent function. Thus, the reported saving of the computation time would be very beneficial.

3) In Table I, we show three columns for better understanding purpose. In reality, as the row-indeces are ordered sequentially (as integers from 1 to 101), it is sufficient to store only the third column in the LUT.
4) In (15), we have a term $(LUT(index + 1) - LUT(index))$. This is the difference between an entry and the next one in Table I. This would be a static quantity. Therefore, we could pre-compute this difference and keep in the LUT as a second column. This would eliminate the difference operation in (15), reducing the time-consumption further. However, that will come at an expense of additional column in the LUT, consuming memory. This should be checked for particular embedded platform, applications, etc.
5) As argued above, and in section IV.A, to facilitate an easier linear index referencing, the table was designed with 101-points to cover the range of 0 to 1, in steps of roughly 0.01. If we design a table with number of elements in power of two (which could ease the hardware division), e.g., 128 instead of 100, we would lose the property that $round(\frac{y}{x} \times 100)$ gives the array index of the angle value. Then, we would need a second column to keep track of $\frac{y}{x}$, which would increase the memory burden.
6) The impacts of the computations of $\frac{y}{x}$ and the 'round' operation on latency are insignificant for the computation of $arctan$. The time consumption reduction already includes those.
7) In (15), we have a term $(LUT(index + 1) - LUT(index))$. This is the difference between an entry and the next one in Table I. This would be a static quantity. Therefore, we could pre-compute this difference and keep in the LUT as a second column. This would eliminate the difference operation in (15), reducing the time-consumption further. However, that will come at an expense of additional column in the LUT, consuming memory. This should be checked for particular embedded platform, applications, etc.

VI. Conclusions

Inverse tangent or arctangent function has many applications, e.g., in estimating the phase angle of complex number utilized in electrical ac circuit analysis, power systems analysis, etc. Series expansion-based implementation of the arctangent, typically found in PC-based applications cannot be used for embedded applications. As alternative, different approximations [16],[17] are proposed . In this paper, we have presented an efficient LUT-based implementation. Sometimes it is argued that LUT-based approach requires a lot of memory [16],[17]. This paper presents quantitative comparisons of the different approaches in terms of accuracy, computation time, size of LUT (indicator of memory consumption), implemented in a rather low-end platform. Even though LUT-based technique is generally known, it is hard to find quantitative comparisons for particular functions. This work aims to fill gaps in the literature concerning choice of arctangent function implementation for embedded applications.

The proposed implementation uses a LUT with 101 points. The accuracy is increased by linear interpolation. This particular length of LUT is chosen to facilitate an easy array indexing. The arctangent computation is extended to four quadrant operation using particular properties of $\arctan$. The proposed scheme is implemented in a 60MHz Keil MCB2130 evaluation board [18]. Comparative evaluations show that the accuracy of the proposed method (maximum angle error of 2.42×10^{-5} rad) is better than the reported approximation techniques (maximum angle error 1.5×10^{-3} rad [17]). The time consumption of the proposed method (35μs) is significantly less than that of the library function (75μs) of the MCB2130 board, using the Keil MDK-ARM IDE [20].

Acknowledgment

This work was supported by ABB Corporate Research.

References

[1] M. Abramowitz, I. A. Stegun, eds., *Handbook of Mathematical Functions with Formulas, Graphs, Mathematical Tables*, New York: Dover, 1972.
[2] I. N. Bronshtein, K. A. Semendyayev, *Handbook of Mathematics*, 3rd ed. New York: Springer-Verlag, 1997.
[3] cplusplus, "C++, C library: cmath (math.h): atan2." Available: http://www.cplusplus.com/reference/clibrary/cmath/atan2/
[4] The MathWorks, "MATLAB documentation," 2010. Available: http://www.mathworks.com/access/helpdesk/help/techdoc/ref/atan2.html
[5] Wolfram Research, "Mathematica documentation - ArcTan." Available: http://reference.wolfram.com/mathematica/ref/ArcTan.html
[6] Sun Microsystems, "java.lang Class Math documentation." Available: http://java.sun.com/javase/6/docs/api/java/lang/Math.html
[7] Microsoft, "MSDN documentation: WorksheetFunction.Atan2 Method." Available: http://msdn.microsoft.com/en-us/library/bb238940.aspx
[8] A. G. Phadke, J. S. Thorp, *Synchronized Phasor Measurements and Their Applications*, Springer: New York, 2008.
[9] A. Ukil, *Intelligent Systems and Signal Processing in Power Engineering*, Springer: Heidelberg, 2007.
[10] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.*, vol. EC-8, no. 3, pp. 330–334, 1959.
[11] J. E. Volder, "The Birth of CORDIC," *J. VLSI Signal Processing*, vol. 25, no. 2, pp. 101–105, 2000.
[12] S. J. Bellis, W. P. Marnane, "A CORDIC Arctangent FPGA Implementation for a High-Speed 3D-Camera System," *Lect. Notes. Comp. Sc.*, vol. 1896/2000, pp. 485–494, 2000.
[13] K. Maharatna et al., "A CORDIC like processor for computation of arctangent and absolute magnitude of a vector," In Proc. 59th Int. Symp. Circuits Systems, 2004.
[14] D. D. Hwang, D. Fu, A. N. Willson, "A 400-MHz processor for the conversion of rectangular to polar coordinates in 0.25-μm CMOS," *IEEE J. Solid-State Circuits*, vol. 38, no. 10, pp. 1771–1775, 2003.
[15] I. Koren, O. Zinaty, "Evaluating elementary functions in a numerical coprocessor based on rational approximations," *IEEE Trans. Comput.*, vol. 39, no. 8, pp. 1030–1037, 1990.
[16] R. Lyons, "Another contender in the arctangent race," *IEEE Signal Processing Magazine*, vol. 20, no. 1, pp. 109–111, 2004.
[17] S. Rajan, S. Wang, R. Inkol, A. Joyal, "Efficient approximations for the arctangent function," *IEEE Signal Processing Magazine*, vol. 23, no. 3, pp. 108–111, 2006.
[18] Keil, "MCB2130 evaluation board." Available: http://www.keil.com/mcb2130/
[19] Keil, "MCB2130 evaluation board: technical specifications." Available: http://www.keil.com/mcb2130/specs.asp
[20] Keil, "MDK-ARM microcontroller development kit." Available: http://www.keil.com/arm/