http://www.moderntechno.de/index.php/meit/article/view/meit41-01-023
DOI: 10.30890/2567-5273.2025-41-01-023

UDC 621.391

ANALYTICALLY CONTINUOUS FUNCTIONS FOR COMPUTING THE ARGUMENT OF A COMPLEX NUMBER

Lukin K.A.
Dr. of Phys.-Math. Sciences, Prof., IEEE Fellow
ORCID: 0000-0001-9998-9207
Konovalov V.M.
Leading Engineer
ORCID: 0009-0004-1932-4627
A. Ya. Usikov Institute of Radiophysics & Electronics,
National Academy of Sciences of Ukraine,
Kharkiv, Proskury 12, 61085

Abstract. *Computing the argument of a complex number is fundamental to signal processing, navigation, and computational geometry. The standard atan2 function, while numerically efficient, produces cumbersome piecewise expressions in symbolic computation systems, complicating automatic differentiation, integration, and code generation. We present a family of analytically continuous functions – Atan4, Asin4, and Acos4 – that provide closed-form alternatives to classical inverse trigonometric functions. These functions combine analytical expressiveness with numerical efficiency, being expressed entirely through elementary algebraic operations without explicit piecewise definitions. We rigorously prove the equivalence between Atan4 and* $\mathrm{atan2}(\sin x, \cos x)$*, analyze its analytical properties, and present optimized implementations including branchless scalar and vectorized (SIMD) algorithms. Benchmarks demonstrate that optimized scalar Atan4 achieves 0.94× the performance of atan2, while vectorized implementations provide more than threefold speedup in mass data processing. Application to phase analysis of frequency-modulated signals demonstrates practical advantages in symbolic manipulation and automatic code generation. The systematic approach extends naturally to Asin4 and Acos4, making this function family a promising addition to mathematical libraries for specialized tasks requiring both analytical rigor and computational efficiency.*

Key words: *argument computation, atan2, Atan4, inverse trigonometric functions, symbolic computation, phase analysis, branchless algorithms, SIMD vectorization.*

Introduction

Computing the argument of a complex number is a fundamental operation in engineering and scientific applications, including signal processing, radar systems, navigation, quantum mechanics, and computational geometry. The standard function for this purpose is $\mathrm{atan2}(y, x)$, developed in the 1960s-1970s and now included in virtually all programming languages and mathematical libraries [1], [2]. Its advantages are well established: correct operation in all four quadrants, numerical stability when $x \approx 0$, and high performance through hardware and library-level optimizations.

However, atan2 has a piecewise-defined structure that makes it inconvenient for symbolic computation systems. When used in automatic differentiation, integration, or

expression simplification, atan2 typically generates complex conditional expressions requiring additional processing. Classical sources such as Numerical Recipes [2], along with comprehensive treatments by Higham [3] and foundational work by Knuth [8], discuss the numerical efficiency of atan2 but acknowledge challenges in symbolic contexts. Modern computer algebra systems such as SymPy [4] support atan2 but represent it using piecewise-defined expressions, complicating automatic simplification. In signal processing applications – particularly FM signal phase analysis as detailed by Oppenheim and Schafer [5] and Haykin [6] – atan2 is widely used for instantaneous frequency estimation, yet its piecewise nature hinders analytical derivations.

To address this limitation, we propose a family of functions – Atan4, Asin4, and Acos4 – that: (1) preserve the numerical stability and correctness of atan2; (2) possess closed-form analytical representations; (3) enable efficient symbolic transformations and automatic code generation; (4) achieve performance comparable to or exceeding atan2 in vectorized computations. This paper focuses primarily on Atan4 as the most practically significant member of the family, while demonstrating the systematic nature of the approach through its natural extension to Asin4 and Acos4.

The remainder of this paper is organized as follows. Section II defines the Atan4 function and Section III proves its equivalence to atan2. Section IV analyzes analytical properties. Sections V-VI present implementation strategies and regularization techniques. Section VII presents performance benchmarks, Section VIII demonstrates application to FM signal analysis, and Sections IX-X provide discussion and conclusions.

II. Definition of the Atan4 function

We introduce the function Atan4(x) as a closed-form analytical alternative to atan2:

$$\mathrm{Atan4}(x) = \arctan\left(\frac{\sin x}{\cos x}\right) + \pi\left[1 - \frac{\mathrm{sign}(\sin x)}{2}(1 + \mathrm{sign}(\cos x))\right]$$

where

$$sign(t) = \begin{cases} \frac{t}{|t|}, & t \neq 0 \\ 0, & t = 0 \end{cases}$$

is the standard sign function. The function is expressed entirely through elementary operations (arctan, sin, cos, sign), providing a compact analytical representation.

The range of Atan4 is $[0, 2\pi)$ for $x \in \mathbb{R}$. To convert to the standard principal value range $(-\pi, \pi]$, we define:

$$\mathrm{Atan4}_{\mathrm{pr}}(x) = \begin{cases} \mathrm{Atan4}(x), & \text{if } \mathrm{Atan4}(x) \le \pi \\ \mathrm{Atan4}(x) - 2\pi, & \text{if } \mathrm{Atan4}(x) > \pi \end{cases}$$

This mapping ensures full consistency with the classical definition of the argument of a complex number [5], [6].

Alternative form for symbolic computation. The sign function can be expressed directly through trigonometric functions:

$$\mathrm{Atan4}(x) = \arctan\left(\frac{\sin x}{\cos x}\right) + \pi\left[1 - \frac{1}{2}\frac{\sin x}{|\sin x|}\left(1 + \frac{\cos x}{|\cos x|}\right)\right]$$

This representation preserves algebraic structure in computer algebra systems, as $|t|$ is processed as $\sqrt{t^2}$ rather than expanding into piecewise definitions.

Extension to Asin4 and Acos4. The approach extends systematically to other inverse trigonometric functions:

$$\mathrm{Asin4}(x) = \mathrm{sign}_\varepsilon(\cos x) \cdot \arcsin(\sin x) + \frac{\pi}{2}[1 - \mathrm{sign}_\varepsilon(\cos x)] \cdot \mathrm{sign}_\varepsilon(\sin x)$$

$$\mathrm{Acos4}(x) = \mathrm{sign}_\varepsilon(\sin x) \cdot \arccos(\cos x) + \frac{\pi}{2}[1 - |\mathrm{sign}_\varepsilon(\sin x)|][1 - \mathrm{sign}_\varepsilon(\cos x)]$$

All three functions are mathematically equivalent: $\mathrm{Atan4}(x) \equiv \mathrm{Asin4}(x) \equiv \mathrm{Acos4}(x) \equiv x \ (mod\ 2\pi)$. From the standpoint of numerical stability and performance, Atan4 is preferable in most cases, as arctangent is numerically stable, has an unrestricted domain, and is highly optimized in mathematical libraries.

III. Equivalence of Atan4 and atan2

Theorem 1. For any real x, the following equality holds:

$$\mathrm{Atan4}_{\mathrm{pr}}(x) = \mathrm{atan2}(\sin x, \cos x)$$

Proof. We verify equivalence by examining each quadrant separately.

Case 1: First quadrant ($\cos x > 0$, $\sin x \ge 0$)

In this region,

$\arctan(\sin x/\cos x) = \arctan(\tan x) = x$ (modulo 2π for x reduced to $[0, \pi/$

2]).

The correction term evaluates to:

$$\pi[1-\frac{1}{2}(1)(1+1)]=\pi[1-1]=0$$

Thus, $\mathrm{Atan4}(x)=x$, which coincides with $\mathrm{atan2}(\sin x,\cos x)$.

Case 2: Second quadrant $(\cos x<0,\sin x>0)$

Here, $\arctan(\sin x/\cos x)$ yields $x-\pi$ (the principal value). The correction term evaluates to:

$$\pi\left[1-\frac{1}{2}(1)(1-1)\right]=\pi$$

Thus, $\mathrm{Atan4}(x)=(x-\pi)+\pi=x$, matching atan2.

Case 3: Third quadrant $(\cos x<0,\sin x<0)$

The principal value is again $x-\pi$. The correction term evaluates to:

$$\pi\left[1-\frac{1}{2}(-1)(1-1)\right]=\pi$$

Result: $\mathrm{Atan4}(x)=x$. After conversion to $(-\pi,\pi]$, this equals $\mathrm{atan2}(\sin x,\cos x)$.

Case 4: Fourth quadrant $(\cos x>0,\sin x<0)$

Principal value: $\arctan(\sin x/\cos x)=x$. The correction term evaluates to:

$$\pi\left[1-\frac{1}{2}(-1)(1+1)\right]=2\pi$$

Result: $\mathrm{Atan4}(x)=x+2\pi$. After mapping to $(-\pi,\pi]$, this yields x, as required.

Boundary points $(x=0,\ \pi/2,\ \pi,\ 3\pi/2)$: Values at these points, obtained through one-sided limits, coincide with atan2 by continuity.

Unlike the standard atan2 function, the proposed Atan4 is not defined piecewise but expressed in closed-form, facilitating symbolic manipulation.

IV. Analytical properties

1. Motivating example: Symbolic differentiation

To illustrate the advantage of closed-form representation, consider differentiating the phase function in SymPy. For $\mathrm{atan2}(\sin(x),\cos(x))$, this produces a piecewise expression requiring simplification to reduce to unity. In contrast, differentiation of

Atan4(x) using the regularized form yields directly: 1. This demonstrates the practical advantage of Atan4 for symbolic computation.

2. Continuity and differentiability

The Atan4 function is continuous over $\mathbb{R}$ except at points where $\text{sign}(0)$ is undefined ($\sin x = 0$ or $\cos x = 0$). These discontinuities are eliminated through regularization (Section VI). Within each quadrant, Atan4(x) locally coincides with the linear function $x + C$, where C depends on the quadrant. Hence:

$$\frac{d}{dx}\text{Atan4}(x) = 1$$

for all x except at quadrant boundaries. At boundaries, one-sided limits coincide, making the function piecewise smooth for practical purposes.

3. Periodicity and stability

Atan4 is 2π-periodic, consistent with atan2 behavior and the classical definition of argument. For large $|x|$, reduce the argument modulo 2π (using fmod or equivalent) to prevent numerical error accumulation.

V. Numerical implementation

1. Branchless implementation

To eliminate conditional jumps, compute sign arithmetically. The branchless computation paradigm, extensively analyzed in the literature [7], enables elimination of conditional branches and improved efficiency of SIMD-based implementations:

$$\text{sign}(t) = (t > 0) - (t < 0)$$

This enables branchless evaluation:

$$\text{Atan4}(x) = \arctan\left(\frac{\sin x}{\cos x}\right) + \pi\left[1 - \frac{\text{sign}(\sin x)}{2}(1 + \text{sign}(\cos x))\right]$$

where sign is computed without branching using comparison operators.

2. Vectorized (SIMD) implementation

Using AVX2 or AVX-512 enables processing 4-8 elements simultaneously: load vector x, compute $\sin(x)$ and $\cos(x)$ vectorially, evaluate arctan vectorially, compute sign using vectorized comparisons, and add correction term. This approach provides approximately $3 - 8\times$ acceleration compared to element-wise atan2, depending on hardware and compiler optimizations. Performance gains are architecture-specific and

depend on data alignment, compiler optimization level, and availability of vectorized math libraries.

VI. Regularization of the sign function

The function $\text{sign}(t) = t/|t|$ is undefined at $t = 0$. In practical computations, this discontinuity can cause numerical instability when processing noisy data, performing numerical integration, or applying gradient-based optimization methods where continuity and differentiability are essential.

Regularization principle. To eliminate the discontinuity at zero, we introduce a regularized sign function:

$$\text{sign}_\varepsilon(t) = \frac{t}{\sqrt{t^2 + \varepsilon^2}}, \qquad \varepsilon > 0$$

As $\varepsilon \to 0$, this function converges pointwise to the classical $\text{sign}(t)$. Unlike the discontinuous classical function, $\text{sign}\_\varepsilon(t)$ provides a smooth transition between -1 and $+1$ near zero. The corresponding substitutions for trigonometric expressions are:

$$\frac{\sin(x)}{|\sin(x)|} \Rightarrow \frac{\sin(x)}{\sqrt{\sin^2 x + \varepsilon^2}}, \qquad \frac{\cos(x)}{|\cos(x)|} \Rightarrow \frac{\cos(x)}{\sqrt{\cos^2 x + \varepsilon^2}}$$

Properties. For any $\varepsilon > 0$, $\text{sign}\_\varepsilon(t)$ is continuous and infinitely differentiable over its entire domain. The parameter ε controls the transition width. The regularized function has a continuous derivative:

$$\frac{d}{dt}\text{sign}_\varepsilon(t) = \frac{\varepsilon^2}{(t^2 + \varepsilon^2)^{3/2}}$$

with maximum derivative at $t = 0$ equal to $1/\varepsilon$ and decay rate $\sim 1/t^2$ for large $|t|$. This approach ensures numerical stability against rounding and overflow errors, as discussed in the numerical analysis literature [3], [9], [10].

Application in Atan4. The fully regularized Atan4 function is:

$$\text{Atan4}_\varepsilon(x) = \arctan\left(\frac{\sin x}{\cos x}\right) + \pi\left\{1 - \frac{1}{2}\text{sign}_\varepsilon(\sin x)[1 + \text{sign}_\varepsilon(\cos x)]\right\}$$

This formulation is globally continuous on $\mathbb{R}$, infinitely differentiable on $\mathbb{R}$, and converges to Atan4 as $\varepsilon \to 0$.

Practical recommendations. For double precision (float64), use $\varepsilon \sim 10^{-12}$ to 10^{-8}; for single precision (float32), use $\varepsilon \sim 10^{-6}$ to 10^{-4}. In symbolic computation, use

non-regularized form and apply regularization as a limiting operation ($\varepsilon \to 0$) in final analytical results. In numerical optimization, always use regularized form to ensure continuous gradients.

VII. Performance benchmarks

Experimental setup: Intel Core i7-10700K, GCC 12 with $-O3$ optimization, 10^6 values uniformly distributed in $(-10\pi, 10\pi)$, average over 100 runs.

Table 1 - Performance comparison

Method	Time (ms)	Relative Speed
atan2(sin,cos) (libm)	12.3	1.0×
Atan4 (naive)	47.8	0.26×
Atan4 (branchless)	15.2	0.81×
Atan4 (branchless, inlined)	13.1	0.94×
Atan4 (AVX2 vectorized)	3.8	3.2×

Authors

Results are specific to the test configuration. Performance may vary with different compilers, processor architectures, and data alignment patterns. Accuracy: Atan4 matches atan2 to machine epsilon ($\sim 10^{-16}$ for double precision) in all tested cases. Key findings: naive implementation is $\sim 4\times$ slower than atan2; optimized branchless scalar version achieves $0.94\times$ performance; vectorized implementation provides $> 3\times$ speedup, demonstrating advantage for mass data processing.

VIII. Application: Phase analysis of frequency-modulated signals

We demonstrate the practical advantages of Atan4 through phase analysis of frequency-modulated (FM) signals.

Problem formulation. Consider a frequency-modulated signal:

$$s(t) = A\cos\varphi(t), \qquad \varphi(t) = \omega_0 t + \beta\sin(\Omega t)$$

where A is amplitude, ω_0 is carrier frequency, β is modulation index, and Ω is modulation frequency. Task: compute the instantaneous frequency $\omega(t) = d\varphi/dt$.

Traditional approach using atan2. The standard method constructs the analytic signal using the Hilbert transform, extracts phase $\hat{\varphi}(t) = \text{atan2}(\text{Im}[\text{s\_analytic}(t)], \text{Re}[\text{s\_analytic}(t)])$, and differentiates to obtain instantaneous frequency. When attempting symbolic differentiation in a CAS, atan2 produces complex piecewise

expressions that require extensive simplification, complicate analytical manipulation, hinder automatic code generation, and obscure the underlying mathematical structure.

Improved approach using Atan4. Direct phase computation: $\hat{\varphi}(t) = \mathrm{Atan4}(\varphi(t))$. Symbolic differentiation yields:

$$\omega(t) = \frac{d}{dt}\varphi(t) = \omega_0 + \beta\Omega\cos(\Omega t)$$

This result matches the exact theoretical derivative, contains no piecewise definitions, and is immediately usable for further analysis.

Practical benefits. The analytical form enables direct Fourier analysis using Bessel functions of the first kind, allows calculation of optimal filter bandwidth and time-varying filter parameters, enables sensitivity analysis computed symbolically without numerical approximation, and supports direct export to C/CUDA/VHDL for hardware implementation via automatic code generation tools.

Comparison of symbolic processing efficiency. We compare symbolic processing in SymPy for test expressions where $f(t) = t^2 + \sin(3t)/t$ (representative complex function):

Table 2 - SymPy processing comparison

Operation	Original Expression	Simplification Time (ms)	Size of Simplified Expression (nodes)
diff(atan2(sin(f(t)), cos(f(t))), t)	Piecewise(...)	15.2	45
diff(Atan4(f(t)), t)	Derivative(f(t), t)	2.1	3

Authors

Key findings: 7.2 × faster symbolic processing, 15 × smaller expression trees, and direct analytical form vs. complex piecewise structure.

Extension to real signals with noise. For signals containing noise, using regularized Atan4:

$$\omega(t) \approx \frac{d}{dt} Atan4_{\varepsilon}(\hat{\varphi}(t))$$

The regularization parameter ε provides natural noise robustness, smoothing small noise fluctuations near $\sin x = 0$ or $\cos x = 0$, avoiding spurious discontinuities in derivative estimation, and enabling stable gradient-based tracking algorithms.

IX. Discussion

Systematic nature of the approach. The proposed family – Atan4, Asin4, and Acos4 – demonstrates a unified methodology for constructing closed-form analytical representations of complete (four-quadrant) inverse trigonometric functions. All three are built according to a consistent principle: use the principal value as the base function, add a correction term constructed from ratios of trigonometric functions to their absolute values, and express the result using elementary algebraic operations without explicit conditional logic. This systematicity indicates generality: the approach is a transferable method applicable to other multivalued functions requiring unwrapping or quadrant disambiguation.

Advantages and optimal use cases. Key strengths include purely algebraic representation (unlike formulations with explicit sign functions which CAS expand into piecewise definitions, Atan4 uses only trigonometry, absolute values, and division), strict mathematical equivalence to atan2, symbolic computation efficiency (demonstrated $7\times$ faster processing and $15\times$ smaller expression trees), vectorization advantages (branchless structure enables efficient SIMD implementations), and gradient continuity (regularized form provides smooth derivatives for optimization algorithms).

Optimal application domains: computer algebra systems (primary use case; eliminates piecewise expansion), automatic code generation (clean symbolic forms translate directly to efficient code), SIMD-vectorized mass computations (branchless structure crucial for performance), gradient-based optimization (smooth derivatives required for convergence), and analytical derivations (compact forms simplify manual calculation).

Limitations. Modern CPUs have heavily optimized atan2 implementations in libm/glibc, often using hardware-accelerated instructions, lookup tables with polynomial refinement, and guaranteed IEEE 754 compliance for edge cases. Atan4 cannot leverage these hardware-specific optimizations directly, potentially limiting scalar performance on some architectures. While theoretically equivalent, Atan4 requires careful attention to division by zero when $\cos x = 0$, numerical precision near

$\sin x = 0$ or $\cos x = 0$, and behavior with non-finite inputs. The regularization parameter ε introduces approximation error $O(\varepsilon^2)$ and requires parameter tuning.

Positioning relative to atan2. Atan4 is not intended as a universal replacement for atan2 but rather as a specialized tool complementing atan2. Prefer atan2 for conventional numerical computations where performance and standardization are paramount. Choose Atan4 when symbolic manipulation, automatic differentiation, or vectorization are critical.

X. Conclusion

We have presented a systematic approach to constructing closed-form analytical representations of inverse trigonometric functions for computing the argument of a complex number. The proposed function family – Atan4, Asin4, and Acos4 – combines analytical rigor with numerical efficiency, addressing long-standing limitations of traditional piecewise-defined functions.

Principal contributions include: analytical formulation using only elementary algebraic operations without explicit piecewise definitions, rigorous proof of mathematical equivalence between Atan4 and $\text{atan2}(\sin x, \cos x)$, characterization of continuity, differentiability, periodicity, and development of regularization techniques ensuring global smoothness, multiple implementation strategies (scalar branchless, regularized for optimization, vectorized SIMD), performance validation showing optimized scalar Atan4 achieves 94% of atan2 performance while vectorized implementations provide $> 3\times$ speedup, and application demonstration showing $7\times$ faster symbolic processing and seamless integration into automatic code generation workflows.

The Atan4/Asin4/Acos4 family is best viewed as a specialized tool for domains where analytical expressiveness is paramount: symbolic mathematics systems, automatic differentiation frameworks, code generation tools, and vectorized signal processing. We acknowledge that Atan4 cannot match the extensive hardware optimization of established atan2 implementations in standard scalar computations, and is best viewed as complementing rather than replacing atan2.

Future research directions include extension to complex arguments, rigorous error

analysis for finite-precision arithmetic, hardware acceleration through FPGA and GPU kernels, standardization proposals for ISO C/C++ standards, and broader applications in quantum computing phase estimation, computer vision, and robotics. The development of Atan4 demonstrates that computational mathematics continues to benefit from revisiting classical problems with modern tools and requirements. As scientific computing becomes increasingly integrated with symbolic systems and machine learning pipelines, the demand for functions that are simultaneously analytically elegant and numerically efficient will only grow.

References

1. *IEEE Standard for Floating-Point Arithmetic*, IEEE Std 754-2019, IEEE, 2019, doi:10.1109/IEEESTD.2019.8766229

2. W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed., Cambridge Univ. Press, 2007, doi:10.1017/CBO9780511813890

3. N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, 2002, doi:10.1137/1.9780898718027

4. A. Meurer et al., "SymPy: symbolic computing in Python," *PeerJ Comput. Sci.*, vol. 3, e103, 2017, doi:10.7717/peerj-cs.103

5. Oppenheim, A. V., & Schafer, R. W. 2010. *Discrete-Time Signal Processing.* 3rd ed. Pearson. ISBN: 9780131988422

6. Haykin, S., & Moher, M. 2022. *Communication Systems.* 6th ed. Wiley. ISBN: 978-1-119-82826-6

7. J. L. Bentley, "Programming Pearls: Eliminating Branches from a Binary Search," *Communications of the ACM*, vol. 32, no. 12, pp. 1492-1498, 1989, doi:10.1145/76380.76381

8. D. E. Knuth, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, Addison-Wesley, 1998, doi:10.5555/280635

9. Burden, R. L., Faires, J. D., & Burden, A. M. 2024. *Numerical Analysis.* 11th ed. Pearson. ISBN: 978-0137343888

10. J. Stoer и R. Bulirsch, *Introduction to Numerical Analysis*, 3rd ed., Springer, 2002, doi:10.1007/978-0-387-21738-3