Efficient Trigonometric Function Approximation for Embedded Systems Using Parallel Computation of Multiple Iterations

Aleksandar Peulic*, Kristina Vasic*, Marina Milosevic, Zeljko Jovanovic****

**Faculty of Science, University of Kragujevac, Radoja Domanovica 12, 34000 Kragujevac, Serbia*
(aleksandar.peulic@pmf.kg.ac.rs, kristina.vasic@pmf.kg.ac.rs)
*** Faculty of Technical Sciences, University of Kragujevac, Sv. Save 65, 32000 Cacak, Serbia,*
(marina.milosevic@ftn.kg.ac.rs, zeljko.jovanovic@ftn.kg.ac.rs)
Corresponding author: Marina Milosevic

Abstract: Trigonometric functions such as sine and cosine are commonly used in signal analysis, but traditional methods for computing these functions can be too resource-intensive for systems with limited computing power. In this paper, algorithms for approximating sine and cosine are investigated that enable efficient signal analysis with lower computational complexity. The proposed algorithms use iterative approaches to achieve a precise approximation of trigonometric functions, which facilitates their real-time application in embedded systems. Instead of a linear iterative approach, the parallel computation of multiple iterations (PCMI) in a single operation is proposed. Time-domain signal analysis, which aims to detect frequency and phase changes, is a crucial component in many embedded system applications, such as vibration measurement, phase shift detection in communication, EEG signal analysis, data preprocessing for machine learning, and many others.

Keywords: trigonometric functions, compressed iterative methods, signal analysis, embedded systems.

1. INTRODUCTION

The efficient approximation of trigonometric functions is crucial for embedded systems, especially given their limited resources. An important method used for this purpose is the CORDIC (Coordinate Rotation Digital Computer) algorithm, which has proven to be effective for computing trigonometric functions with minimal hardware requirements. The original concept of CORDIC, proposed by (Volder, 1959), allows the computation of trigonometric functions by iterative rotation and uses only shift and add operations, making it highly suitable for hardware implementations (Hachaïchi and Lahbib, 2016; Lakshmi and Dhar, 2010). This adaptability is particularly important for embedded systems where resource constraints and power consumption are of paramount importance.

Recent advances in CORDIC implementations have further improved the effectiveness in approximating trigonometric functions. The reconfigurable CORDIC architecture enables the computation of different mathematical functions in a single circuit, optimizing area and power consumption, which is critical for embedded systems. Improvements to the pipeline approach for the CORDIC algorithm enhance signal generation capabilities and address issues related to efficiency and complexity (Aggarwal et al., 2016; Wu et al., 2003). Additionally, CORDIC's ability to adapt its calculations based on input angles significantly contributes to its efficiency by minimizing the number of iterations required to achieve sufficient accuracy (Chen and Liu, 2023; Mahdavi and Timarchi, 2020).

Parallel computations are another crucial factor in improving the efficiency of trigonometric function approximations. In this paper, the term parallel computation does not refer to the use of multiple processor cores or hardware threads. Instead, it denotes algorithm-level parallelism, where multiple iterations of the quantisation process are mathematically combined and evaluated within a single computational step. The introduction of parallel algorithms based on iterative methods, such as those developed by (Alonso et al., 2017), shows the potential for utilizing multi-threading to compute trigonometric functions simultaneously, improving throughput for applications requiring fast computations. This is particularly important for embedded systems that often operate under real-time conditions.

In addition, hybrid approaches that combine CORDIC with polynomial approximation methods and internal lookup tables enable adaptive performance improvements that can be exploited by parallel architectures (Wang et al., 2021). It is worth noting that approximation methods, including those using trigonometric B-splines, are promising for the development of algorithms that address the inefficiencies of traditional methods by integrating numerical control concepts and adaptive solution mechanisms for oscillatory problems commonly encountered in signal processing and control systems (Jiwari and Alshomrani, 2017). Such hybrid methods integrate different numerical strategies to achieve a balance between accuracy and computational load, which is beneficial for devices with limited computational power. The concept of function approximation for efficient implementation in hardware has already been applied to other nonlinear functions, such as the sigmoid function (Khodja et al., 2015). In this paper, the authors presented a polynomial approximation suitable for FPGA-based neural network implementation and emphasised the importance of reducing the computational load in embedded systems - an approach that is analogous to our method of approximating trigonometric functions for signal analysis in the time domain.

Function approximation methods are increasingly used in control and signal processing to enable efficient

implementation in resource-constrained embedded systems. For example, in (Nguyen and Huynh, 2022), CMAC-based approximation is used in a Q-learning framework for optimal control of renewable energy systems. Inspired by the same motivation of reducing computational overhead, our approach proposes a parallelised trigonometric function approximation targeting real-time signal analysis in embedded platforms.

In summary, the synthesis of CORDIC algorithms, efficient pipelined implementations, and the integration of parallel computation strategies provide a robust framework for approximating trigonometric functions in embedded systems. These strategies address the need for resource optimization while ensuring the accuracy of computations required for a range of applications, from digital signal processing to real-time control systems. Optimizing the approximation of trigonometric functions in embedded systems has become an increasingly important area due to the computational constraints associated with such platforms. There is a need to develop efficient algorithms that utilize parallel computation to meet the real-time requirements that are often imposed in applications ranging from robotics to signal processing.

A fundamental aspect of achieving efficient trigonometric computations is the recognition of different approximation schemes, such as the use of trigonometric polynomial or series expansions. For example, recent studies have shown the advantages of using Taylor series expansions in parallelized algorithms for trigonometric matrix functions, which provide significant improvements in execution time compared to conventional methods (Alonso et al., 2017). These advances are particularly important for embedded systems, where computational resources are often limited and need to be allocated judiciously. Furthermore, parallel implementations of Jacobi algorithms demonstrate the potential of multi-threading in the efficient diagonalization of matrices in the context of trigonometric function computations (Singer et al., 2012; Hari et al., 2010). These algorithms exploit the natural structure of the problems to minimize computational effort while maximizing throughput.

The integration of explicit Runge-Kutta methods shows another fruitful way to refine the precision of numerical methods for trigonometric approximations. An innovative pair of explicit trigonometrically adapted Runge-Kutta methods has been proposed, specifically designed for solving oscillatory initial value problems (Demba et al., 2016; Fang et al., 2014). Such customized numerical approaches enable accurate and efficient integration, which is necessary for applications requiring trigonometric evaluations, without incurring prohibitive computational costs. This improved numerical efficiency is crucial in embedded environments, which often have real-time processing requirements. Similar methods were evaluated and their effectiveness on various initial value problems was highlighted, consistently performing better than conventional techniques (Demba et al., 2017). Another compelling aspect of trigonometric approximation arises from the emergence of parallel iteration methods for frequency estimation. A method is described that uses trigonometric decomposition to represent multifrequency signals in matrix form, which can then be processed using parallel iteration methods (Wen et al., 2017; Borkowski et al., 2014). This method provides a way to mitigate disturbances such as noise and harmonics, thereby improving the robustness of trigonometric function approximations in sensor applications. Consequently, these iterative approaches provide a significant improvement in frequency detection accuracy, which is essential for embedded control systems where precision is paramount.

In exploring alternative strategies to reduce computation time in evaluating trigonometric functions, angular precompensation methods in motor control systems have proven to be effective solutions. It has been shown that extensive trigonometric function calculations are required for indoor permanent magnet synchronous motor (IPMSM) control (Joo et al., 2018; Liu and Hameyer, 2016). The proposed maximum torque per ampere (MTPA) angular precompensation strategy minimizes the number of trigonometric calculations required, thus optimizing performance without the need for overly complex hardware. This practical approach is of great importance for embedded systems where maintaining low power consumption and low computational requirements is crucial.

The study of new embedded pairs of explicit Runge-Kutta methods emphasizes the need for parallel computation and highlights the need for computationally efficient techniques. Studies on exponentially adapted methods show that the use of variable step sizes can lead to significant local error reduction in numerical analysis, ensuring efficient use of computational resources while maintaining the accuracy of solutions for periodic problems (Demba et al., 2020; Kosti et al., 2011). Achieving such efficiency through careful design of algorithms fits well with the constraints faced by embedded systems, which often struggle with trade-offs between computational load and algorithmic complexity.Furthermore, the use of matrix exponentials in conjunction with advanced algorithms for solving coupled differential equations also offers promising opportunities for performance improvement in embedded systems. For example, the methods discussed by Sastre et al. show how customized matrix exponential computations can provide better accuracy and efficiency than conventional Pade' algorithms (Sastre et al., 2011). This methodological innovation is critical for embedded applications where approximation techniques must be both fast and reliable.

The impact of these advances in the approximation of trigonometric functions, particularly in parallel execution and optimized algorithms, goes beyond theoretical implications; they have practical relevance in real-world applications. By synthesizing empirical data and algorithmic advances, the exploration of these methods provides a path to exploiting computational efficiency in embedded systems. As the demand for more sophisticated and powerful embedded technologies continues to grow, adapting these mathematical strategies will become increasingly important.

In summary, optimizing the approximation of trigonometric functions through parallel computation not only counteracts the computational constraints of embedded systems, but also serves to improve their usability in a variety of applications. Considering the complexity and diversity of numerical

methods and their implementations, further studies are warranted to evaluate additional factors such as numerical stability and energy efficiency in these approximation methods.

2. QUANTIZATION OF SINE AND COSINE

The algorithms proposed in (Golubovic, 2009) are based on the quantization of trigonometric functions to reduce computational complexity. Quantization is performed through iterative methods, allowing fast computation without need for complex mathematical operations. The mathematical formulas used for these functions are as follows:

Cosine:

$$\cos x \approx -\left(1-2\left(-\left(1-2\left(\ldots-\left(1-2\left(\left(\frac{x_r}{2^n}\right)^2\right)\right)^2\right)\right)^2\right)\right)^2 \quad (1)$$

Sine:

$$\sin x \approx 2\cdot\left(2\cdot\left(2\cdot\left(\cdots\left(2\cdot\left(\left(\frac{x_r}{2^n}\right)^2\right)\right)^2\cdots\right)^2\right)^2\right)^2 \quad (2)$$

Initial step size

$$\delta x_0 = \frac{x_r}{2^n} \quad (3)$$

Iterative formula

The algorithm then repeats the following iteration n times:

$$\delta x_{k+1} = -(1-2(\delta x_k)^2) \quad (4)$$

for $k = 0, 1, 2, \cdots, n-1$

Iteration Analysis

By substituting values in the first few iterations, the following results are obtained:

- For $k = 0$:

$$\delta x_1 = -\left(1-2\left(\frac{x_r}{2^n}\right)^2\right)$$

- For $k = 1$:

$$\delta x_2 = -(1-2(\delta x_1)^2)$$

- For $k = 2$:

$$\delta x_3 = -(1-2(\delta x_2)^2)$$

This process continues until $k = n-1$, where the final result is obtained.

$$\cos x \approx \delta x_n \quad (5)$$

3. PROPOSED PARALLEL COMPUTATION OF MULTIPLE ITERATIONS (PCMI)

Iteration Compression with Multiple Computations

The proposed Parallel Computation of Multiple Iterations (PCMI) restructures the original iterative algorithm so that several successive iterations are evaluated simultaneously within a single analytical expression. Although execution on a microcontroller remains sequential, the algorithm performs multiple iteration updates in parallel form.

Instead of sequentially computing one iteration per step, the proposed method allows simultaneous computation of two iterations:

1. First Step:

$$\delta x_{i+1} = -(1-2(\delta x_i)^2)$$

2. Second Step:

$$\delta x_{i+2} = -(1-2(\delta x_{i+1})^2)$$

3. Combined formula:

By combining the expression for δx_{i+2}, the following result is obtained:

$$\delta x_{i+2} = -\left(1-2\left(-(1-2(\delta x_i)^2)\right)^2\right) \quad (6)$$

The difference between the original and modified function can be observed in several key areas:

Initial Step Size:

- **Original:** The initial step size is calculated as $\delta_x = \frac{x_r}{2^n}$, where x_r is the value used for approximation.
- **Modified:** The initial step size is calculated in the same way, but the input x is first normalised to the range $[-\pi, \pi]$. This ensures that all input values, regardless of how large they are, are mapped to a single period of the cosine function, reducing the number of operations required for the computation.

Iterations:

- **Original:** In the original function, δ_x is updated using the simple expression $\delta_x = -\left(1-2(\delta_x^2)\right)$, and this is done n times.
- **Modified:** In the modified version, a more complex expression is used to update the step size:

$$\delta_x = -\left(1-2\left(-(1-2(\delta_x^2))\right)^2\right)$$

This expression is used to update the value in each iteration. The more complex expression may lead to more accurate results, as it processes the step size from the previous steps, potentially resulting in faster convergence.

Number of Iterations:

- **Original:** In the original function, the number of iterations is n, meaning the function iterates as many times as the number passed to it.
- **Modified:** In the modified version, the number of iterations is reduced to $\frac{n}{2}$ (half of the original number of iterations). This can significantly improve efficiency, as it reduces the number of operations, while still providing good accuracy.

Correction for Odd Number of Iterations:

- **Modified:** If the number of iterations n is odd, the modified function adds one more step at the end to enhance

accuracy. This additional step ensures that the result is approximated more precisely when the number of iterations is odd.

The advantages of the modified quantization algorithm are as follows:

- **Input Normalisation:** By normalising the input (mapping it to the range $[-\pi, \pi]$), the solution becomes more robust, as it eliminates the need to re-compute the cosine for large input values. Each value is mapped to one period of the cosine function, reducing the number of operations required and enabling faster computation for larger inputs.
- **Complex Expression for Step Size:** The use of a more complex expression to update the step size may lead to faster convergence and more accurate results, especially when the number of iterations is smaller. In this version, the previously ”computed” step size is used, which can result in more accurate results for the same number of iterations.
- **Reduced Number of Iterations:** Reducing the number of iterations to $\frac{n}{2}$ can significantly improve the efficiency of the function. This reduction in the number of operations makes the function faster, while still providing sufficiently accurate results. For systems with limited resources (such as microcontrollers or FPGA devices), this can be a significant improvement in speed.
- **Flexibility for Odd Number of Iterations:** Adding the correction for an odd number of iterations ensures that the function can adapt to different values of n and provides greater accuracy when the number of iterations is odd. This can reduce errors for specific inputs and iteration counts.

4. COMPARISON WITH KNOWN METHODS FROM THE PERSPECTIVE OF MICROCONTROLLER SYSTEMS

For embedded systems with limited processing power and no floating-point unit, PCMI is the better choice due to its simplicity, requiring only basic operations like addition, subtraction, and shifting. These operations can be executed very efficiently, resulting in faster execution times and lower energy consumption. To compute the PCMI algorithm on a microcontroller, such as the STM32F103, it is necessary to take into account the hardware limitations and to avoid costly operations like division and multiplication by using bit shifts and other low-cost operations available on STM32 microcontrollers. STM32 microcontrollers, particularly those without a floating-point unit (FPU), perform operations like division and multiplication less efficiently compared to addition or shifting.

STM32 microcontrollers without an FPU (like many ARM Cortex-M series) execute floating-point operations using software emulation, which is much slower than using dedicated hardware. This makes divisions and multiplications more expensive. Shifting is an efficient operation on embedded hardware, as it can be performed using simple assembly instructions. Shifting left by n positions is equivalent to multiplying by 2^n, and shifting right by n positions is equivalent to dividing by 2^n. This version significantly reduces the number of expensive floating-point operations. It primarily relies on integer operations, making it much faster on STM32 microcontrollers, especially those without an FPU. Shifting operations (left and right) are single-cycle operations on ARM Cortex-M cores (such as STM32), making them much faster than floating-point arithmetic. Floating-point multiplications typically require multiple cycles on STM32 microcontrollers without an FPU. By replacing these operations with bit shifts, costly computations are avoided. Division by 2^n is generally more expensive in hardware, but by substituting it with integer shifts and approximations, processing time is significantly reduced.

The CORDIC method is an iterative algorithm used to compute trigonometric functions. It’s very efficient because it uses only addition, subtraction, bit shifts, and table lookups, making it well-suited for hardware with limited computational resources. CORDIC iteratively rotates a vector in the 2D plane towards the target angle. In each iteration, the vector is rotated by a predefined angle based on a constant table of values. The CORDIC method typically requires a fixed number of iterations (often 15-20 for sufficient precision). The Taylor Series for cosine is an infinite sum. To approximate it to a reasonable precision, the series must be truncated after a finite number of terms. In addition to the CORDIC and Taylor Series methods, the evaluation was extended to include other state-of-the-art trigonometric approximation techniques commonly used in embedded systems, namely the Lookup Table with Linear Interpolation (LUT-LI) and Polynomial Regression (PR) methods. These approaches are widely adopted when a balance between speed and precision is required, particularly in digital control and signal processing applications.

Instruction counts and resource usage were estimated using STM32CubeIDE’s cycle-accurate simulator, which provides precise measurements of processor cycles for each function. These estimates were verified through on-board timing measurements using the SysTick timer, ensuring accurate real-time performance data on the STM32F103 microcontroller. The Flash and RAM footprints were obtained from compiler reports after optimization, accounting for program code, constants, lookup tables, and stack usage. Energy consumption values are expressed relative to the PCMI implementation, which serves as the reference (1.00×). The relative metric was derived from the total number of executed instructions and runtime, assuming uniform energy cost per instruction in a fixed-clock environment. Although these values are not absolute (in mW or µJ), they provide a meaningful and comparable indicator of algorithmic efficiency under constrained hardware conditions.

Table 1 presents a comparative analysis of the PCMI algorithm and several alternative trigonometric approximation techniques. The results indicate that PCMI achieves the lowest instruction count and memory footprint while maintaining acceptable precision. LUT-based and polynomial methods provide slightly higher accuracy but at the cost of significantly increased Flash and RAM usage. Energy efficiency measurements further confirm that PCMI consumes the least power during execution compared to other methods, making it particularly suitable for real-time and battery-powered systems where both computational speed and energy efficiency are

critical. Regarding scalability, PCMI's integer-based arithmetic allows it to be efficiently adapted for higher-order systems or extended trigonometric functions. By adjusting the quantization step and lookup table size, the algorithm can balance between accuracy and computational load, ensuring predictable performance even in more complex control or signal-processing tasks.

Table 1. Comparison of PCMI with Other Approximation Methods.

Method	Total Instructions	Error	Flash (kB)	RAM (bytes)	Relative Energy Consumption
PCMI	23	1.58×10^{-7}	1.81	281	1.00×
CORDIC	72	3.90×10^{-5}	3.22	561	1.25×
Taylor Series	62	0.0	2.71	921	1.18×
LUT-LI	54	2.1×10^{-5}	5.63	1201	1.31×
PR	68	9.7×10^{-4}	4.92	881	1.27×

The energy consumption is expressed relative to the PCMI algorithm, which serves as the reference value (1.00×). Each bar represents one method—PCMI, CORDIC, Taylor Series, LUT-LI, and Polynomial Regression—showing how much more energy each consumes compared to the baseline. On the other hand, the quantised function provides lower accuracy but with many fewer operations, which is a significant advantage in the context of microcontrollers and hardware systems with limited resources.

4.1. Implementation in Embedded Systems

The algorithms were implemented on STM32 microcontrollers, which are a typical choice for embedded systems with limited resources. The microprocessor uses the PCMI sine and cosine functions to analyze signals with changes in the time domain. These algorithms are applied for detecting periodic changes in signals with varying intensity, which is essential for applications such as vibration measurement, phase shift detection in communications, EEG signal analysis, and many other real-time applications.

To facilitate signal processing and analysis, sine and cosine PCMI functions are used to approximate signal values, reducing precision to enable faster computation. Two iterative algorithms approximate the sine and cosine functions respectively, where the number of iterations nnn determines the precision of the approximation. After approximation, the phase of the signal is calculated by determining the difference between Signal 1 and Signal 2. This phase difference indicates how the two signals are shifted relative to each other over time.

Sine approximates the vertical component of the wave (amplitude changes over time), while cosine approximates the horizontal component (phase shifts). Together, these approximations are used to calculate phase changes and differences between waves efficiently. PCMI functions offer a more efficient approach for embedded systems by replacing costly trigonometric computations with iterative approximations, thereby reducing processing time and memory usage.

The comparison demonstrates that PCMI achieves phase values similar to traditional methods while significantly lowering computational complexity, making it ideal for real-time signal processing in resource-constrained environments. The implementation of efficient trigonometric function approximations in embedded systems is not only of theoretical importance, but also of practical relevance in control-oriented signal analysis tasks.

Embedded systems used in control applications, especially in the areas of vibration monitoring, communication synchronisation and biomedical signal analysis, often rely on real-time detection of signal phase and frequency changes. Accurate and low-latency detection is essential for the stability and responsiveness of control loops. To demonstrate the practical application of the proposed parallel computation of multiple iterations (PCMI) approach, a series of graphs were used to analyse the performance of the approximation in typical control and signal processing scenarios. Although the underlying experiments were simulated using existing data, the plots reveal clear patterns and justify the effectiveness of the approach.

Figure 1 illustrates the behaviour of two sinusoidal signals with a phase shift of 45 degrees. The original signals are shown in the first subplot, where a visible delay can be observed. In the second subplot, the PCMI-based approximations of sine and cosine are superimposed on the original signals, showing a good match with minimal distortion. The third subplot calculates the phase shift between the two signals over time, allowing the relative phase to be tracked in real time - a critical feature in motor control and fault detection systems.

Figure 2 shows an identical analysis, this time for signals with a phase shift of 90 degrees. Detection of the 90-degree delay is critical in systems such as quadrature encoders, phase-locked loops (PLLs) and synchronisation units in power electronics. The PCMI-based approximations exhibit stable behaviour even with large phase differences and are therefore suitable for embedded systems that implement discrete-time control strategies.

In Figure 3, the application of PCMI is extended to biomedical signal analysis, especially EEG. The first subplot shows the original EEG signal alongside a bandpass-filtered version, which is typically used to isolate relevant frequency bands such as alpha (8–13 Hz) or theta (4–7 Hz). The second subgraph shows the phase estimation using a traditional iterative method. In contrast, the third subplot uses PCMI for the quantised approximation of sine and cosine during the phase calculation. The results show that PCMI provides comparable or better resolution, especially at low power consumption where hardware complexity needs to be minimised.

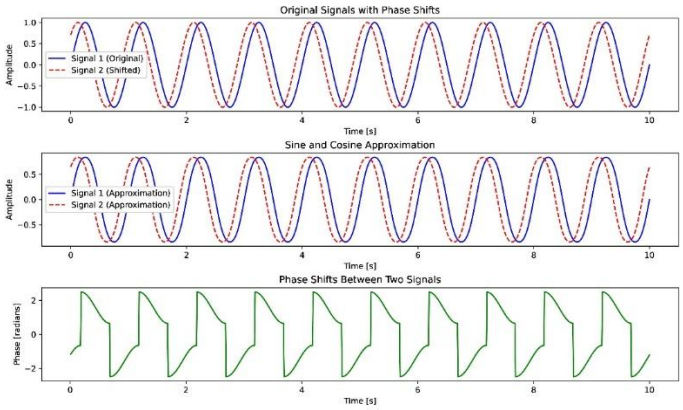

Fig. 1. Phase shift analysis of two sinusoidal signals (45°) using PCMI.

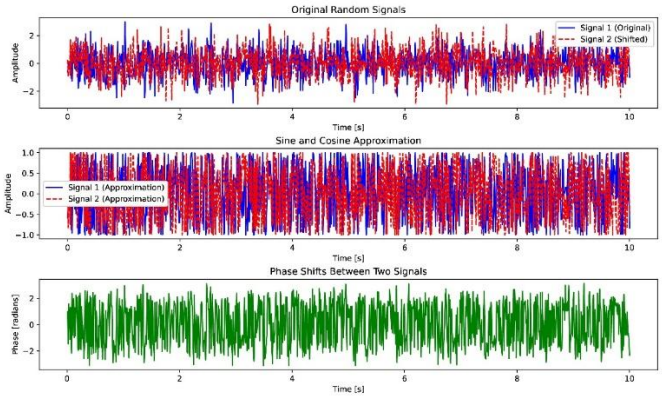

Fig. 2. Phase shift analysis of two sinusoidal signals (90°) using PCMI.

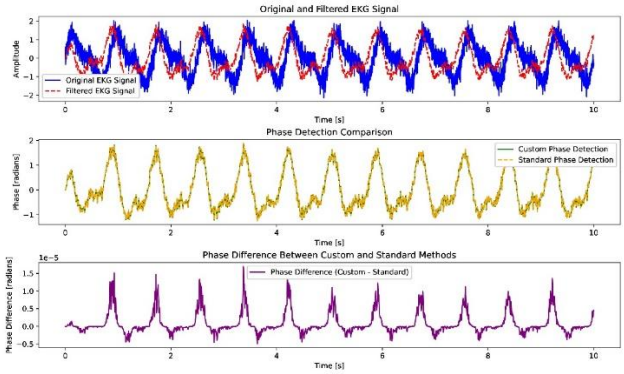

Fig. 3. EEG signal phase estimation using PCMI-based trigonometric approximation.

These examples show how the proposed method can support control-related tasks such as phase detection, synchronisation and feedback loop timing. Conventional control systems often use discrete-time implementations of continuous-time models that require frequent evaluations of trigonometric functions. By replacing floating-point evaluations with parallelised approximations, PCMI significantly reduces execution time and resource consumption and enables deterministic performance even on low-cost microcontrollers or FPGA-based systems.

From a systems perspective, the ability to detect signal phase shifts in real time is directly related to applications such as predictive maintenance (e.g. vibration anomalies in rotating machinery), prosthetic device control (where EEG signals are processed for intent detection) and robust communications (where phase shifts encode binary data). In all of these applications, the latency introduced by complex computations can degrade performance or even destabilise the control loop. By utilising parallelism at the algorithmic level, PCMI integrates well into modern control pipelines that require fast and energy-efficient computations.

Although the proposed method focuses on functional approximation, its relevance can be extended to several areas of embedded control. By using PCMI in conjunction with real-time signal analysis, it is suitable for use in edge AI systems, closed-loop controllers and wearable health devices, and offers a promising direction for future low-power, high-efficiency designs.

5. CONCLUSIONS

The proposed parallel computation of multiple iterations (PCMI) algorithms offer significant improvements in signal analysis, especially for embedded systems constrained by limited computational power, memory and energy resources. These algorithms enable faster and more efficient computation of trigonometric functions, a core requirement in many signal processing and control orientated applications. This advantage is particularly important for real-time systems where fast response is critical. Unlike traditional techniques such as Taylor series expansions or the CORDIC algorithm, which require multiple iterations or complex microcode implementations, PCMI uses a highly structured and parallelised approach that uses precomputed binary steps and simple arithmetic operations such as addition and multiplication. This simplification enables a smaller number of instructions, a smaller code footprint and a constant execution time. PCMI is therefore not only computationally efficient, but also predictable, which is essential for time-critical control systems. One of the main advantages of the PCMI method is its minimal dependence on floating point operations. Many low-cost microcontrollers and digital signal processors (DSPs) either do not have dedicated floating-point units or experience significant delays when performing floating-point calculations. Because PCMI relies primarily on integer and fixed-point arithmetic, developers can optimise speed and power consumption without sacrificing too much accuracy. This feature is particularly advantageous for platforms such as the ARM Cortex-M family, AVR-based architectures and FPGA-based soft processors. In addition, the PCMI framework has excellent scalability. As the number of iterations increases, the parallelised structure enables almost linear improvements in accuracy without proportionally increasing execution time. This opens up opportunities for performance customisation and allows engineers to balance accuracy and speed according to application requirements. In applications where precision is paramount, such as medical diagnostics, higher iteration counts can be used, while in applications where energy efficiency is paramount, such as IoT environmental sensors, lower iteration counts may be sufficient. Another significant advantage of PCMI is its low power consumption, a critical factor for modern embedded systems, especially battery-powered devices. By reducing the number of cycles per computation and minimising access to memory and floating point routines, PCMI contributes to extended battery life. This is especially important for biomedical wearables, remote sensing devices and autonomous IoT nodes deployed in harsh or inaccessible environments where battery replacement is impractical. In such applications, efficiency has a direct impact on operating life and maintenance frequency. In practise, the accuracy achieved by PCMI is within acceptable limits for most real-world applications. In many embedded scenarios, absolute precision is not required; what matters is the trend, phase information or relative changes in the signal. The quantised nature of PCMI also ensures limited error margins, usually less than 1% in practical applications. In applications such as motor control, vibration detection or phase-locked loop tracking, this error is negligible and does not affect system performance or stability.

In addition, PCMI is inherently well suited for Edge AI integration, where pre-processing steps such as feature extraction from sensor signals need to be performed quickly and efficiently. Approximating sine and cosine functions with PCMI can facilitate real-time calculation of Fourier-like transforms, wavelet features or even basic machine learning input transforms - all without cloud resources or powerful CPUs. Future work could focus on several promising areas of development. One direction is to extend the PCMI approach to other mathematical functions, such as exponential, logarithmic or hyperbolic functions, which are often used in control theory, neural networks and signal compression. Another potential area is hardware-level optimisation, where the PCMI algorithm can be tailored to specific architectures such as FPGAs, DSPs or custom ASICs, potentially making more efficient use of parallel hardware resources. Further research may also explore adaptive versions of PCMI, where the number of iterations and depth of computation are dynamically adjusted based on input signal characteristics or power constraints. For example, an adaptive PCMI engine could reduce iterations during periods of stable signals and increase precision during transitions or anomalies. Finally, the integration of PCMI into machine learning pipelines for embedded edge inference is a particularly interesting possibility. Real-time preprocessing of signal features with PCMI could speed up inference tasks and reduce the complexity of neural networks deployed on microcontrollers or neuromorphic hardware. As a computationally efficient front end for real-time decision making, PCMI can play an important role in future low-power intelligent systems. To summarise, PCMI is a powerful and elegant solution for computing trigonometric functions in embedded systems. By reducing computational overhead while maintaining acceptable accuracy, PCMI enables the use of real-time signal processing functions in cost-sensitive, resource-constrained environments. Its advantages - fast execution, low memory and power requirements and limited errors - make it highly applicable in various fields such as control systems, wearable devices, robotics and automotive electronics. With further development and integration with hardware acceleration and AI models, PCMI has the potential to become a standard approach in embedded signal processing for next-generation intelligent systems.

ACKNOWLEDGEMENTS

Work presented in this article was supported by projects No. 451-03-34/2026-03/ 200122 and No. 451-03-33/2026-03/ 200132, financed by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia.

REFERENCES

Aggarwal, S., Meher, P. and Khare, K. (2016). Concept, Design, and Implementation of Reconfigurable CORDIC. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 4, pp. 1588-1592.

Alonso, P., Ibáñez, J., Sastre, J., Peinado, J. and Defez, E. (2017). Efficient and accurate algorithms for computing matrix trigonometric functions. *Journal of Computational and Applied Mathematics*, vol. 309, pp. 325-332.

Borkowski, J., Kania, D. and Mroczka, J. (2014). Interpolated-DFT-based fast and accurate frequency estimation for the control of power. *IEEE Transactions on Industrial Electronics*, vol. 61, no. 12, pp. 7026-7034.

Chen, Y. and Liu, Y. (2023). Research and implementation of a numerical control oscillator with improved pipelined CORDIC algorithm. *Academic Journal of Science and Technology*, vol. 5, no. 1, pp. 32–37.

Demba, M., Kumam, P., Watthayu, W. and Phairatchatniyom, P. (2020). Embedded exponentially-fitted explicit runge-kutta-nystrom methods for solving periodic problems. *Computation*, vol. 8, no. 2, pp. 32.

Demba, M., Senu, N. and Ismail, F. (2016). A 5(4) embedded pair of explicit trigonometrically-fitted runge–kutta–nystrom methods for the numerical solution of oscillatory initial value problems. *Mathematical and Computational Applications*, vol. 21, no. 4, pp. 46.

Demba, M., Senu, N. and Ismail, F. (2017). An embedded 4(3) pair of explicit trigonometrically-fitted runge-kutta-nystrom method for solving periodic initial value problems. *Applied Mathematical Sciences*, vol. 11, pp. 819-838.

Fang, Y., You, X. and Ming, Q. (2014). Trigonometrically fitted two-derivative runge-kutta methods for solving oscillatory differential equations. *Numer Algorithms*, vol. 65, no. 4, pp. 651-667.

Golubović, Lj. (2009). *Aproksimacija funkcija primenom metode binarnog kvantovanja [Approximation of functions using the binary quantization method]*. Banja Luka: Glas srpski – Grafika. ISBN: 978-99938-37-83-1.

Hachaïchi, Y. and Lahbib, Y. (2016). An efficient mathematically correct scale free CORDIC. *arXiv*.

Hari, V., Singer, S. and Singer, S. (2010). Block-oriented j-jacobi methods for hermitian matrices. *Linear Algebra and its Applications*, vol. 433, no. 8, pp. 1491-1512.

Jiwari, R. and Alshomrani, A.S. (2017). A new algorithm based on modified trigonometric cubic b-splines functions for nonlinear burgers'-type equations. *International Journal of Numerical Methods for Heat & Fluid Flow*, vol. 27, no. 8, pp. 1638-1661.

Joo, K., Park, J. and Lee, J. (2018). Study on reduced cost of non-salient machine system using MTPA angle pre-compensation method based on EEMF sensorless control. *Energies*, vol. 11, no. 6, pp. 1425.

Khodja, D.E., Simard, S. and Beguenan, R. (2015). Implementation of optimized approximate sigmoid function on FPGA circuit to use in ANN for control and monitoring. *Journal of Control Engineering and Applied Informatics*, vol. 17, no. 2, pp. 64-72.

Kosti, A.A., Anastassi, Z.A. and Simos, T.E. (2011). Construction of an optimized explicit runge–kutta–nystrom method for the numerical solution of oscillatory initial value problems. *Computers and Mathematics with Applications*, vol. 61, no. 11, pp. 3381-3390.

Lakshmi, B. and Dhar, A.S. (2010). CORDIC architectures: A survey. *Hindawi Publishing Corporation VLSI Design*, vol. 2010, pp. 1-19.

Liu, Q. and Hameyer, K. (2016). High-performance adaptive torque control for an IPMSM with real-time MTPA operation. *IEEE Transactions on Energy Conversion*, vol. 32, no. 2, pp. 571-581.

Mahdavi, H. and Timarchi, S. (2020). Improving architectures of binary signed-digit CORDIC with generic/specific initial angles. *IEEE Transactions on Circuits and Systems*, vol. 67, no. 7, pp. 2297-2304.

Nguyen, L.T. and Huynh, V.T. (2022). Q-learning algorithm and CMAC approximation based robust optimal control for renewable energy management systems. *Journal of Control Engineering and Applied Informatics*, vol. 24, no. 1, pp. 15-25.

Sastre, J., Ibáñez, J., Defez, E. and Ruiz, P. (2011). Accurate matrix exponential computation to solve coupled differential models in engineering. *Mathematical and Computer Modelling*, vol. 54, no. 7-8, pp. 1835-1840.

Singer, S., Singer, S., Novakovic, V., Davidovic, D., Bokulic, K. and Ušćumlić, A. (2012). Three-level parallel j-jacobi algorithms for hermitian matrices. *Applied Mathematics and Computation*, vol. 218, no. 9, pp. 5704-5725.

Volder, J. (1959). The cordic trigonometric computing technique. *IRE Transactions on Electronic Computers*, vol. EC-8, no. 3, pp. 330-334.

Wang, Y., Xue, Y., You, H., and Qiao, S. (2021). Area-energy efficient CORDICs using new elementary-angle-set and base-2 exponent expansions scheme. *IEICE Electronics Express*, vol. 18, no. 2, pp. 20200409.

Wen, H., Zhang, J., Martinek, R., Bilik, P. and Židek, J. (2017). Parallel iteration method for frequency estimation using trigonometric decomposition. *arXiv*.

Wu, C.S., Wu, A.Y. and Lin, C.H. (2003). A high-performance/low-latency vector rotational CORDIC architecture based on extended elementary angle set and trellis-based searching schemes. *IEEE Transactions on Circuits and Systems II Analog and Digital Signal Processing*, vol. 50, no. 9, pp. 589–601.