Equations solver network: an analog approach

D. PORTO[1], M. BRANCIFORTE[1], G. ADRAGNA[1], G. SICURELLA[2]
[1]System Research and Application
STMicroelectronics S.r.l
Stradale Primosole 50, Catania
ITALY

[2]Smartpower TR&D
STMicroelectronics S.r.l
Stradale Primosole 50, Catania
ITALY

Abstract: - The proposed work, starting from the re-formulation of algebraic equations solution in terms of matrix products, aims to describe how some relatively complex calculus, such as algebraic as long as differential equation solutions, can be performed in a fast and accurate way through simple analog resistive circuits, without iteration algorithms and therefore without involving digital processors, providing finally a huge reduction of computation time and consumption. Integration of the HW structures here discussed may open the way to a new generation of cheap problem solvers, useful for real time applications such as robotics or analog co-processors.

Key-Words: - In Memory Computing, Phase Changing Memory, Polynomial Equations, Matrix Algebra, Analogue Networks, Motor Control

Received: May 9, 2026. Revised: July 3, 2026. Accepted: September 6, 2026. Published: November 2, 2026.

1 Background and basic operations

Analog matrix computation is a topic that has been recently investigated for performing out-of-processor operations proper of PCM-based In Memory Computing (IMC)[1][10-11], which aims to perform some operations still inside the memory without involving the processor and limiting the continuous crossing of the "memory wall". In fact, the data movement among processors and memory is responsible, according to studies in [2-3], of about 95% of energy consumption, affecting also the computation time. Actually, Phase Changing Memory (PCM) technology [12] is just starting to deal with the high dimension arrays needed for memories implementation (each PCM memory cell needs to be written and cleared many times), providing also a limited resolution of analog values, whose improvements are expected only in next years. However, the basic principle which implements matrix products is simple and can be today adopted if very large arrays and continuous rewriting are not needed. Considering a matrix product

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ A_{31} & A_{32} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 \\ A_{21}x_1 + A_{22}x_2 \\ A_{21}x_1 + A_{22}x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad (1)$$

This can be shortly written as **AX=B,** meaning that **B** is the vector resulting from the product. If we consider X as a voltage vector and A as a conductance matrix, the resulting vector B is a current vector displaying Ohm's law **GV=I** that can be implemented by the structure in Figure 1.

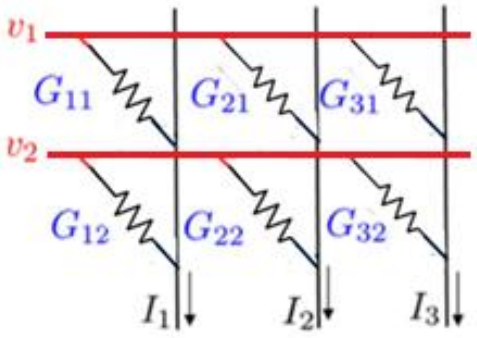


Figure 1. Elementary resistive structure able to perform a matrix product, according to those used for In Memory Computing

Result is in this case given, according to Kirchhoff laws, by:

$$I_1= G_{11}v_1+ G_{12}v_2 \quad I_2= G_{21}v_1 + G_{22}v_2 \quad I_3= G_{31}v_1 + G_{32}v_2 \tag{2a}$$

This structure proves that matrix products can be successfully and quickly performed by a resistive network, regardless of the cell technology adopted. In general, this means that the conductance matrix G allows the calculation of multiple products and sums, if voltages are taken as inputs and currents as outputs. On the other hand, also divisions can be implemented, if inputs and outputs are inverted and G is diagonal. For 3x3 matrix, it can be easily demonstrated that.

$$v_1=I_1/G_{11} \ , v_2= I_2/ G_{22} \ , \ v_3=I_3/G_{33} \tag{2b}$$

Simple structures like that in figure 2 can simultaneously perform many operations at the same time and are still able to solve some nontrivial problems usually formulated in terms of matrix. One of this is the solution of linear algebraic systems:

$$\boldsymbol{AX=B} \quad \rightarrow \quad \boldsymbol{X=A^{-1}B} \tag{3}$$

The evaluation of inverse matrix A^{-1} can be reduced to a routine performing few matrix sums and products and one scalar divisions, e.g. by using the Cayley-Hamilton formula:

$\boldsymbol{A^n} + a_1\boldsymbol{A^{n-1}}+....+ a_{n-1}\boldsymbol{A} + a_n \boldsymbol{I} = \boldsymbol{0}$, which gives

$$\boldsymbol{A^{-1}}= (-a_{n-1} \boldsymbol{I}- a_{n-2}\boldsymbol{A} -.....-a_1\boldsymbol{A^{n-2}}-\boldsymbol{A^{n-1}})/ a_n \tag{4}$$

(n = dim(**A**))

The values $a_1, a_2, ...a_n$ are the coefficient of characteristic polynomial of **A**, easily derivable with well known sums and products of its coefficients (e.g. $a_1 = -Tr(\boldsymbol{A})$, $a_n = det(\boldsymbol{A})$...).

For 3x3 matrix, whose inversion is fundamental in solving third order differential equations, it means:

$$a_1 = -a_{11}-a_{22}-a_{33}$$

$$a_2 = a_{11}a_{22}+a_{22}a_{33} + a_{11}a_{33}-a_{13}a_{31}-a_{23}a_{32}-a_{12}a_{21}$$

$$a_3 = a_{11}a_{22}a_{33} + a_{13}a_{21}a_{32} + a_{12}\, a_{23}a_{31} -$$

$$-a_{13}a_{22}a_{31} - a_{11}a_{23}a_{32} - a_{12}\, a_{21}a_{33} \tag{5}$$

In next sections it will be shown that also some relevant problems that seem not to involve matrix products (like solutions of equations) can be partially or totally expressed through the above, and which could be the related networks able to implement these tasks.

2 Polynomial equations via matrix products

Polynomial Equations are fundamental in a wide variety of problems, being also the basis of differential equations solutions and very often it is a nontrivial task for processors due to the several iterations that are needed in some cases when required solutions are more than two. If we focus our attention to polynomial equation whose degree is less than four, these ones, according to Galois experience, can be solved directly through radicals without trial-and-error routines and could be good candidates to be implemented via matrix computing. However, radicals seem to be a huge task to perform if dealing only with sums and products**.**

Using methods explained in [4] and [5], in [9] it is described how to express the solution of a second grade without square root and with low error. This fact is reflected in an easy formulation of the solutions through few simple matrix products. In fact, considering the equation in its complete form:

$$\lambda^2/b +\lambda+a =0 \tag{6}$$

Assuming $\sqrt{\Delta} \cong \Delta/(2b-4a)+b/2-a$, real solutions are:

$$\begin{bmatrix} -1/2 & -1/4 & -1 \\ -1/2 & 1/4 & 1 \end{bmatrix} \begin{bmatrix} b \\ b-2a \\ \frac{b^2-4ab}{4b-8a} \end{bmatrix} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} \tag{7}$$

In [9] it is also shown how to use approximations coming from classical literature [6] to obtain one solution of a third-grade equation (real case) in two matrix steps.

$$\lambda^3/b +\lambda^2 +a\,\lambda +e =0$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & b & 1 \end{bmatrix} \begin{bmatrix} -3b & 9 & 0 \\ 2b & -9b & 0 \\ 0 & 0 & 27 \end{bmatrix} \begin{bmatrix} b \\ a \\ e \end{bmatrix} = \begin{bmatrix} p \\ q \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -\frac{q}{3p} & 0 & 0 \\ 0 & \frac{13q}{36p} & 0 \\ 0 & 0 & \frac{3q}{64p} \\ -\frac{b}{3} & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ q^2/p^3 \\ q^4/p^6 \end{bmatrix} = \lambda_1 \tag{8}$$

Finding one of the solutions reduces the problem to the second-grade case. All these methods will be used in the next important section.

3 The fourth-grade solution

Let now us consider the equation:

$$\lambda^4 + b\lambda^3 + c\lambda^2 + d\lambda + e = 0 \quad (9)$$

This is the highest degree equation that is solvable by formulas and is very common in a wide variety of applications, such as the observer problem in Field Oriented Control of motors and kinematic description of robotic arm motion. The solution method can be summarized in five steps, using partially the previously described routines.

Step 1: Matrix calculation of auxiliary variables p, q, r. A compact formulation for them can be the following:

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & b & 2 \end{bmatrix} \begin{bmatrix} -3b & 8 & 0 \\ 8b & -32 & 0 \\ 0 & 0 & 32 \end{bmatrix} \begin{bmatrix} b \\ c \\ d \end{bmatrix} = \begin{bmatrix} p \\ q \end{bmatrix} \quad (10)$$

$$\left(\begin{bmatrix} b & 0 & 0 \\ 0 & 1 & \\ & 0 & 1 \\ 0 & & 0 \end{bmatrix} \begin{bmatrix} -3b & 0 & 0 \\ 0 & 16b & \\ & 0 & -64 \\ & & 0 \end{bmatrix} \begin{bmatrix} b \\ b \\ b \\ 1 \end{bmatrix} \right)^T \begin{bmatrix} b \\ c \\ d \\ e \end{bmatrix} = r$$

Step 2: Solve the equation $s^3+2ps^2+(p^2-4r)s-q^2=0$ with method described in previous section.

Step 3: Using p, q and the positive solution s, calculate the following auxiliary values:

$$\alpha=\sqrt{s}\ , \quad \beta = \frac{s}{2} + \frac{p}{2} - \frac{q}{2\alpha}, \quad \gamma = \frac{s}{2} + \frac{p}{2} + \frac{q}{2\alpha} \quad (11)$$

(note: α can be evaluated using (6) with suitable values)

Step 4: Solve the two second order equations $x^2+\alpha x+\beta=0$ and $x^2-\alpha x+\gamma=0$ with method described in section 3. The solutions are (x_1, x_2) and (x_3, x_4) respectively.

Step 4: Final solutions are given by the linear relation:

$$\lambda_i=(x_i-b)/4 \quad (12)$$

This can be summarized by the scheme in Fig. 2.

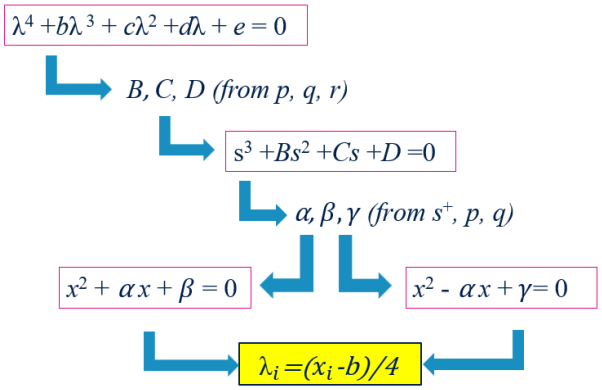


Figure 2. Solution flow of a fourth grade equation by using third and second grade routines

4 Simulation results

Despite this algorithm involves approximations and linearization of determined quantities, the results are accurate, resulting in most cases, in a very low relative error. In figure 3, an average error plot of roots of equation (9) obtained keeping constant b and d and randomizing c and e within a reasonable application range is presented. It can be shown that in the selected window the error is globally below 2%.

A single root analysis has been also performed randomizing *all* parameters on 100 cases ($b,d \in$[-5,5], $c\in$[-20,20] $e\in$ [-100,100]). Mean relative error on each root is reported in table I.

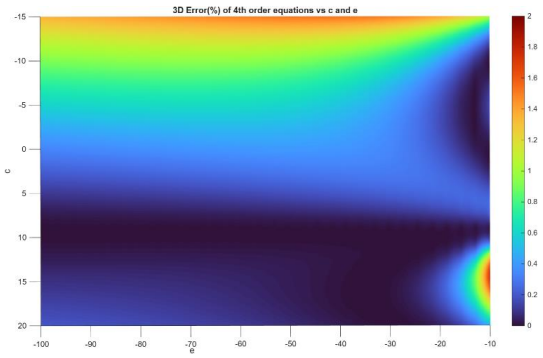


Figure 3. Average error (%) on roots in a randomized window of parameters c and e of equation $\lambda^4 +b\lambda^3 +c\lambda^2 +d\lambda +e = 0$ (with $b=d=5$)

Table I Mean error over 100 random test cases

Solutions	% mean error
λ_1	0,10%
λ_2	1,34%
λ_3	1,34%
λ_4	0,75%

5 Hardware implementation

The algorithm as described in the previous sections, allows to reduce the complexity of the problem into atomic operations such as sums and products. This approach, if correctly exploited, can lead to fast and efficient HW implementations.

However, analyzing the numerical behavior of the system, it appears that even for small inputs, internal subproducts exhibit a wide dynamic range. From a hardware implementation perspective, these observations highlight the necessity of managing wide dynamic ranges effectively to maintain result accuracy. Since analog implementation is desired, this issue becomes even more critical, if left unaddressed, the unconstrained internal quantities could cause saturation in the components involved, thereby amplifying and propagating numerical errors. A possible solution to address this problem has been found by the implementation of a Dynamic Scaling system. This approach can be applied to all the Analog mathematical operators involved in the circuit, effectively maintaining the internal signal inside the desired dynamic range during the flow of operations, avoiding saturation. The Simulink block diagram is shown in Figure. 4. (-log block in input and -antilog block in output is omitted). Threshold trespassing conditions apply once (or twice) a scaling factor of 2.

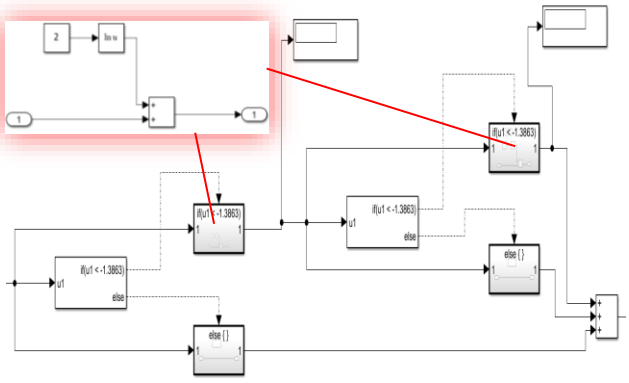


Figure 4. Dynamic gain scaling Architecture

A possible hardware implementation of such systems has been studied, the ORCAD diagram is shown in Figure 5:

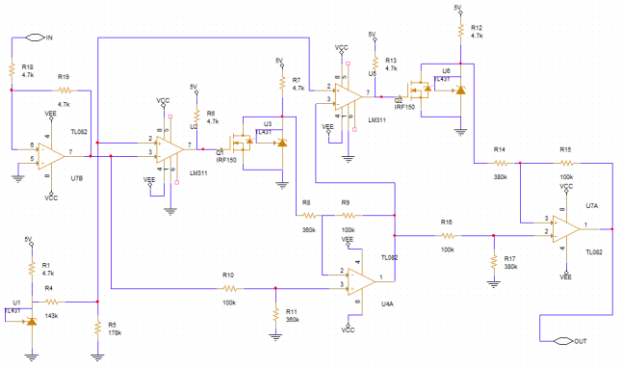


Figure 5. Schematic of auto-scaling circuit in an analog multiplier.

This structure consists of an auto-scaling circuit, in an analog multiplier, with two divide-by-two stages, for a maximum division by 4, activated only when the input voltage exceeds the opamp saturation threshold.

6 A further evolution: Differential Equations

All the previously explained methods open the way to solve in a fast and accurate way the more complex problem of differential equations. In fact, the solution of the polynomial equation is the basis for solving a differential equation of identical order. If we consider a system of four differential equations evolving from its initial condition q(0) (unforced case)

$$\dot{\boldsymbol{q}}(\boldsymbol{t}) = \boldsymbol{A}\boldsymbol{q}(\boldsymbol{t}) \tag{13}$$

with $\boldsymbol{A^{4x4}}$, the solution is given by the formula

$$\boldsymbol{q}(\boldsymbol{t}) = \boldsymbol{e^{At}}\boldsymbol{q}(\boldsymbol{0}) = \sum_{\boldsymbol{i=0}}^{\boldsymbol{3}} \boldsymbol{\alpha_i}(\boldsymbol{t})\boldsymbol{A^i}\boldsymbol{q}(\boldsymbol{0}) \tag{14}$$

where

$$\begin{bmatrix}\alpha_0(t)\\ \alpha_1(t)\\ \alpha_2(t)\\ \alpha_3(t)\end{bmatrix} = \begin{bmatrix}1 & \lambda_1 & \lambda_1^2 & \lambda_1^3\\ 1 & \lambda_2 & \lambda_2^2 & \lambda_2^3\\ 1 & \lambda_3 & \lambda_3^2 & \lambda_3^3\\ 1 & \lambda_4 & \lambda_4^2 & \lambda_4^3\end{bmatrix}^{-1} \begin{bmatrix}e^{\lambda_1 t}\\ e^{\lambda_2 t}\\ e^{\lambda_3 t}\\ e^{\lambda_4 t}\end{bmatrix} \tag{15}$$

and λ_1 , λ_2 , λ_3 , λ_4 are the eigenvalues of $\boldsymbol{A}$, solutions of a fourth-grade equation. It must be noted that all the operations needed to solve the system of differential equations in (13), such as polynomial equation solving, inverse matrix calculation and vector products (see Fig. 6), can be described with the chain matrix products explained

in previous chapters. It is a very remarkable result, because only by adding a suitable dedicated module with a similar architecture of the main one can provide the solution immediately, without iterations, transients, or digital processing. An example of circuit implementing the coefficients of the solution $y(t)=c_1e^{\lambda_1 t}+c_2e^{\lambda_2 t}+c_3e^{\lambda_3 t}$ of a third order differential equation is also depicted (here for simplicity $y''(0)=y'(0)=0, y(0)\neq 0$, real roots case).

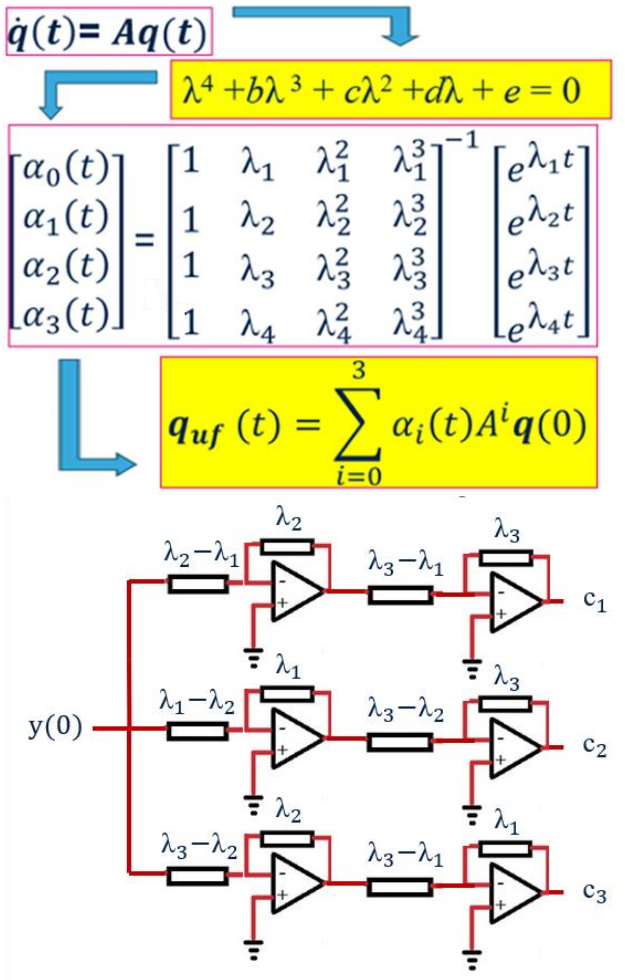


Figure 6. Solution flow of a fourth order system of differential equations and an exaple of circuit implementing the coefficients of the solution of a third-order differential equation $y'''+by''+cy'+dy=0$ with $y''(0)=y'(0)=0$

7 Integration advantages

The hardware presented in the previous chapters is well-suited for mixed-signal integrated implementation, where digital and analog electronic architectures coexist within the same silicon device. In particular, to minimize computational errors in the analog quantities under consideration, established integrated design techniques such as "matching", "offset compensation", and "trimming" can be employed. A natural progression of the hardware model introduced in this work is its realization within a single integrated device. This device can act as an analog coprocessor supporting the main computing unit, which, relieved from the most computationally intensive tasks, can be more efficiently dimensioned. This approach not only optimizes system performance but also contributes to a reduction in the overall application cost. An example of such architecture, specifically in the motor control domain [7-8][13], is illustrated in Fig. 7.

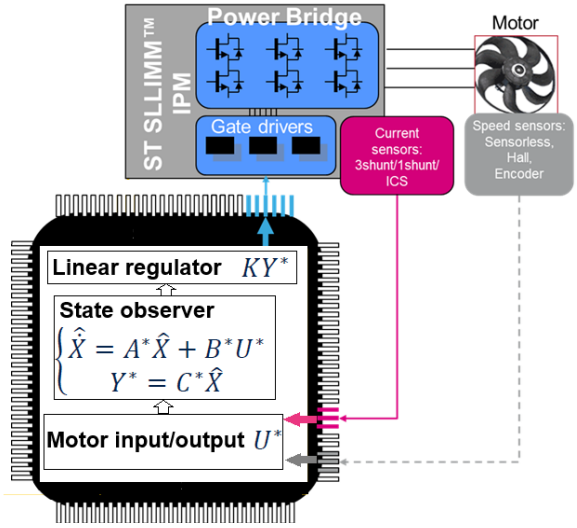


Figure 7. Motor-control system architecture with analog co-processor

8 Conclusions

In this paper it has been described how relatively simple analog structures inspired to those used for In Memory Computing can be able to solve equations in a closed form, immediately and without iterations. A self-adjusting method capable of keeping the operations results always inside a selected voltage range has also been presented. The advantage of avoiding some nontrivial computations to the processor may offer fast and low-consumption solutions to some real-time tasks, for example those dealing with complex control problems and in general in multi-robotic environment cooperating for a target task.

A possible extension of the method to differential equations has been finally presented, opening the way to be adopted in a huge number of other fields of application

Finally, it has been shown that integration can provide a quantum leap to the presented method, improving precision,

portability and cost effectiveness, enlarging also the field of application, giving to the resulting

device the flexibility of a stand-alone processor as well as the possibility to be used as an analog coprocessor to which address some dedicated complex operations

References:

[1] M. Horowitz, 1.1 Computing's Energy Problem (and what we can do about it), *2014 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, 2014, pp. 10-14.

[2] Y.H. Chen, J. Emer, V. Sze, Using Dataflow to Optimize Energy Efficiency of Deep Neural Network Accelerators, *IEEE Micro*, Vol. 37, No. 3, 2017, pp. 12-21.

[3] A. Sebastian, M. Le Gallo, E. Eleftheriou, Computational phase-change memory: beyond von Neumann computing, *Journal of Physics D: Applied Physics*, Vol. 52, No. 44, 2019, pp.1-12

[4] A.I. Hussen, "Approximating Square Roots and Cube Roots", *At Right Angles*, Vol. 4, No. 1, 2015

[5] C. M. Taisbak, Cube roots of integers. A conjecture about Heron's method in Metrika III.20 , *Historia Math.* , Vol 41, 2014, pp. 103–104

[6] L. Leithold, *The Calculus with Analytic Geometry, (fifth edition)*, Harper & Row Publishers, Inc, 1986.

[7] E. Tse, M. Athans, Observer theory for continuous-time linear systems, *Information and Control*, Vol. 22 No. 5, 1973, pp. 405-434

[8] B.H. Mouna, S. Lassaad, Speed sensorless indirect stator field oriented control of induction motor based on Luenberger observer, 2006 *IEEE International Symposium on Industrial Electronics*, Vol. 3. IEEE, 2006, pp. 2473-2478

[9] D. Porto, "Fast processing through resistive networks: the matrix computing", *WSEAS International Journal of Applied Mathematics, Computational Science and Systems Engineering*, vol. 10, June 2024, pp. 112-118

[10] Y. -B. Lee et al., "A Fast and Energy-Efficient Nanoelectro- mechanical Non-Volatile Memory for In-Memory Computing," *2023 IEEE 36th International Conference on Micro Electro Mechanical Systems (MEMS)*, Munich, Germany, 2023, pp. 5-8,

[11] N. Srivastava, A. K. Rajput, M. Pattanaik and G. Kaushal, "An Energy-Efficient and Robust 10T SRAM Based in-Memory Computing Architecture," 2023 *36th International Conference on VLSI Design and 2023 22nd International Conference on Embedded Systems (VLSID)*, IEEE, 2023, pp. 133-138

[12] A. Antolini, A. Lico, E. F. Scarselli, M. Carissimi and M. Pasotti, "Phase-change memory cells characterization in an analog in-memory computing perspective," *2022 17th Conference on Ph.D Research in Microelectronics and Electronics (PRIME)*, IEEE, 2022, pp. 233-236

[13] M. Nicola and C. -I. Nicola, "Improvement Performances of Sensorless Control for PMSM Based on FOC Strategy Using Luenberger Observer, Sine Cosine Algorithm, and RL-TD3 Agent," *2023 International Conference on Electromechanical and Energy Systems (SIELMEN)*, IEEE, 2023, pp. 1-7