Letters

Memristor Crossbar Array Simulation for Deep Learning Applications

Elvis Díaz Machado, Jose Lopez Vicario, Enrique Miranda, *Senior Member, IEEE*, and Antoni Morell

***Abstract*—Hardware neural networks (HNNs) based on crossbar arrays are expected to be energy-efficient computing architectures for solving complex tasks due to their small feature sizes. Although there exist software libraries able to deal with circuit simulation of memristor networks, they still exceed the memory available of any consumer grade GPU's VRAM for large scale crossbar arrays while having a significant computational complexity. This work discusses an iterative method to implement a fast simulation of the corresponding memristor crossbar array with much more limited memory use.**

***Index Terms*—Memristor, crossbar array, circuit simulation, neural network hardware.**

I. Introduction

MEMRISTOR Crossbar Arrays (MCA) have attracted much attention in recent yearsin both neuromorphic computing and Deep Leaning (DL) applications [1], [2], as Multiply-Accumulate (MAC) operations are naturally computed by their structure. Practical implementation is not straightforward and issues such as nonlinearity, stochasticity, varying maxima, or the sneak path problem among others [3], [4], [5], [6], [7] have to be considered. In the context of DL solutions, the sneak path problem is particularly relevant since some authors point to the limitations of smaller MCAs [8]. We can obtain currents and potentials in the MCA by either using circuit analysis tools such as SPICE or by directly modelling and solving the voltage-current equations involved. Both approaches are generally slow in terms of computation time. In this regard, some efforts have been done in finding tailored solutions. In [9], the MCA is modelled as a linear system of equations and solved using a Generalized Minimal RESidual (GMRES) based method [10]. The authors propose a preconditioner to accelerate system solving. However, this approach experiences slow convergence rates when realistic values for the memristor conductances are used.

Manuscript received 18 April 2024; accepted 10 June 2024. Date of publication 17 June 2024; date of current version 24 June 2024. This work was supported in part by ERDF funds and MCIN/AEI/10.13039 /501100011033 project TED2021-129134B-I00 and in part by the European Union "NextGenerationEU"/PRTR funds. The review of this letter was arranged by Associate Editor M. Ottavi. *(Corresponding author: Elvis Díaz Machado.)*

The authors are with the Engineering School, Universitat Autonoma de Barcelona, 08193 Barcelona, Spain (e-mail: elvis.diaz@uab.cat).

Digital Object Identifier 10.1109/TNANO.2024.3415382

Additionally, the amount of Random-Access Memory (RAM) required should be controlled, which turns in a limiting factor for large scale applications such as in [11].

To overcome the referred problems, we propose a tailored solution to solve the linear system of equations representing the MCA. Instead of using GMRES, our approach considers Stationary Iterative Methods (SIMs) [12]. In those methods the key point consists in a clever decomposition of the matrix representing the investigated system.

The main contributions reported in this letter are:

- We propose an iterative method that ensures asymptotically faster convergence than previously reported approaches.
- The properties of the system and the proposed decomposition are analytically validated.
- It is demonstrated that our proposal has a significant smaller memory footprint (~ 20 times) as well as better scalability.

In what follows, Section II describes our system model, Section III details the proposed method, and Section IV provides numerical results reporting also the conclusion of this work.

II. System Model

MCAs can do matrix-vector product operations naturally by adding currents at the output of the array (see Fig. 1), where the amount of current at the these points depends on the input voltages and memristor conductances. In an idealized set-up with all resistors at $0\,\Omega$ except for the memristors, the currents at the output of the MCA, $\mathbf{i}^{\mathtt{S}} = [i_1^{\mathtt{S}}, \ldots, i_n^{\mathtt{S}}]^\top$, are $\mathbf{i}^{\mathtt{S}} = \mathbf{G}\mathbf{v}^{\mathtt{W}}$, where the item at the i-th file, j-th column in $\mathbf{G}$ corresponds to $g_{i,j}$ and $\mathbf{v}^{\mathtt{W}} = [v_1^{\mathtt{W}}, \ldots, v_m^{\mathtt{W}}]^\top$.

If we consider series resistances and the large arrays that are needed in DL architectures, we need to apply Kirchhoff's Current Law (KCL) at every junction as in [13]. Let us consider additional currents and voltages from the resistors at the top layer, with superindex $\mathtt{tl}$, and bottom layer, with superindex $\mathtt{bl}$ (see Fig. 2). The following set of equations results:

$$\begin{aligned}
&i^{\mathtt{tl}}_{i,j-1} = i^{\mathtt{tl}}_{i,j} + i_{i,j} \;\rightarrow && i,j \in \mathbb{N}\\
&g^{\mathtt{tl}}_{i,j-1}\left(v^{\mathtt{tl}}_{i,j-1} - v^{\mathtt{tl}}_{i,j}\right) = && 1 \le i < m\\
&g^{\mathtt{tl}}_{i,j}\left(v^{\mathtt{tl}}_{i,j} - v^{\mathtt{tl}}_{i,j+1}\right) + g_{i,j}\left(v^{\mathtt{tl}}_{i,j} - v^{\mathtt{bl}}_{i,j}\right) && 1 \le j < n \qquad (1)\\
&i^{\mathtt{bl}}_{i-1,j} + i_{i,j} = i^{\mathtt{bl}}_{i,j} \;\rightarrow && i,j \in \mathbb{N}
\end{aligned}$$

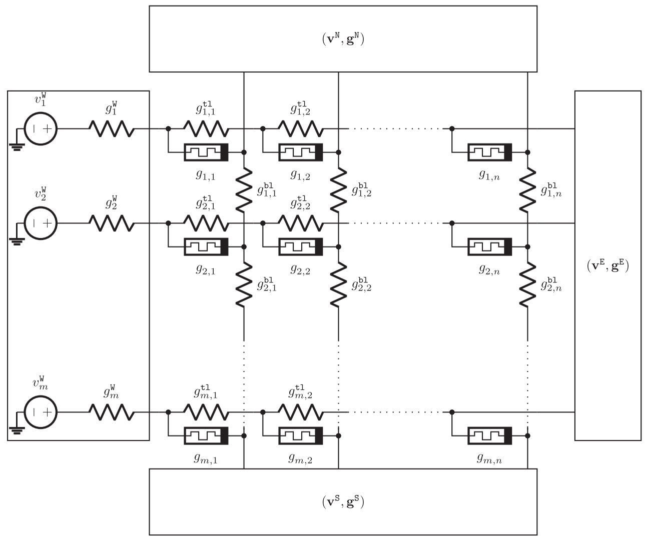

Fig. 1. Memristor Crossbar Array.

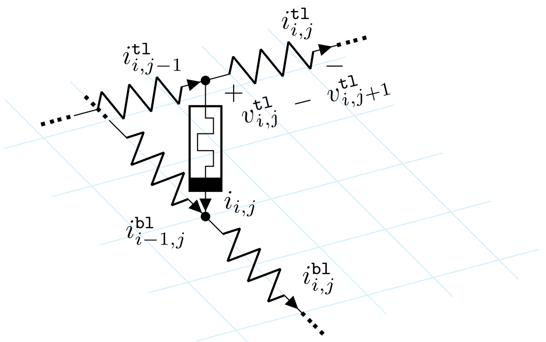

Fig. 2. Detail of a memristor in the MCA.

$$\begin{aligned} & g^{\text{bl}}_{i,j}\left(v^{\text{bl}}_{i,j} - v^{\text{bl}}_{i+1,j}\right) = && 1 \le i < m \\ & g^{\text{bl}}_{i-1,j}\left(v^{\text{bl}}_{i-1,j} - v^{\text{bl}}_{i,j}\right) + g_{i,j}\left(v^{\text{tl}}_{i,j} - v^{\text{bl}}_{i,j}\right) && 1 \le j < n \end{aligned} \tag{2}$$

Note that the elements with sub-indexes 0, n, and m appear due to voltage sources and resistances at the edges, for example $i^{\text{tl}}_{i,0} = v^{\text{W}}_i / r^{\text{W}}_i$ or $i^{\text{bl}}_{m,j} = v^{\text{S}}_j / r^{\text{S}}_j$ (see Fig. 1).

By rearranging and stacking all the equations in (1) and (2) we ultimately get the system we need to solve:

$$\mathbf{G}_{ABCD}\mathbf{v}^{\text{ext}} - \mathbf{i}^{\text{ext}} = \mathbf{0}, \tag{3}$$

where $\mathbf{0}$ is the all-zeros column vector, $\mathbf{v}^{\text{ext}}$ contains the voltages present at each junction of the circuit, and $\mathbf{i}^{\text{ext}}$ represents the currents at the boundaries of the circuit, for example $i^{\text{tl}}_{i,0}$ or $i^{\text{bl}}_{m,j}$ (see Fig. 1). More specifically, $\mathbf{v}^{\text{ext}} = \left[\texttt{vec}(\mathbf{V}^{\text{tl}})^\top, \texttt{vec}(\mathbf{V}^{\text{bl}})^\top\right]^\top$, where matrices $\mathbf{V}^{\text{tl}}$ and $\mathbf{V}^{\text{bl}}$ have $v^{\text{tl}}_{i,j}$ and $v^{\text{bl}}_{i,j}$ at their i,j-th entry, respectively. The operation $\texttt{vec}(\mathbf{X})$ vectorizes as $[x_{1,1}, x_{1,2}, \ldots, x_{1,n}, x_{2,1}, \ldots, x_{m,n}]^T$. Additionally, $\mathbf{i}^{\text{ext}} = \left[\texttt{vec}(\mathbf{I}^{\text{tl}})^\top, \texttt{vec}(\mathbf{I}^{\text{bl}})^\top\right]^\top$, where $\mathbf{I}^{\text{tl}}$ and $\mathbf{I}^{\text{bl}}$ have $i^{\text{tl}}_{i,j}$ and $i^{\text{bl}}_{i,j}$ at their i,j-th entry, respectively.

$\mathbf{G}_{ABCD}$ can be partitioned as $\mathbf{G}_{ABCD} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}$, with

$$\begin{aligned} \mathbf{A} &= \texttt{Diag}(\texttt{vec}(\mathbf{G})) \\ &\quad + \mathbf{T}'_m(-1, 2, -1, \mathbf{S}) \otimes \texttt{diag}\left(\mathbf{G}^{\text{tl}}\right) \end{aligned} \tag{4}$$

$$\begin{aligned} \mathbf{D} &= \texttt{Diag}(\texttt{vec}(\mathbf{G})) \\ &\quad + \texttt{diag}\left(\mathbf{G}^{\text{bl}}\right) \otimes \mathbf{T}'_n(-1, 2, -1, \mathbf{S}) \end{aligned} \tag{5}$$

$$\mathbf{B} = \mathbf{C}^\top = -\texttt{Diag}(\texttt{vec}(\mathbf{G})) \tag{6}$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D} \in \mathbb{R}^{mn \times mn}$, $\texttt{Diag}(\mathbf{x})$ is the diagonal matrix with elements $\mathbf{x}$, $\texttt{diag}(\mathbf{X})$ is the $\mathbf{X}$ matrix with the off-diagonal element equal to 0, and $\mathbf{T}'_m(c, b, a, \mathbf{S})$ is a quasi Toeplitz matrix of the form:

$$\mathbf{T}'_m(c, b, a, \mathbf{S}) =$$

$$\begin{bmatrix} b + \sqrt{a}\sqrt{c} + s_{1,1} & a & & & \\ c & b & a & & \\ & & \ddots & & \\ & & c & b & a \\ & & & c & b + \sqrt{a}\sqrt{c} + s_{m,m} \end{bmatrix}$$

Equations (4) and (5) assume that $\mathbf{g}^{\text{tl}}_i$ are constant along the rows, and $\mathbf{g}^{\text{bl}}_j$ along the columns, so they can be expressed as Kronecker products ($\otimes$).

III. Proposed Solving Method

Matrix $\mathbf{G}_{ABCD}$ has the following two properties: i) it is weakly chained diagonally dominant [14] and ii) it is an L-Matrix, i.e., all values at the diagonal are positive and all off-diagonals are non-positive. The combination of i) and ii) guarantees that $\mathbf{G}_{ABCD}$ is a non-singular M-Matrix and also that is semi-positive definite [15]. Thanks to these properties, we can resort to stationary methods [12] for solving the system. SIMs decompose the system matrix as $\mathbf{G}_{ABCD} = \mathbf{M} - \mathbf{N}$ and the differences among methods rely on the specific definition of $\mathbf{M}$ and, consequently, $\mathbf{N}$. It is mandatory that $\mathbf{M}^{-1}$ and $\mathbf{N}$ have non-negative entries. These conditions guarantee the convergence of the iterative method [12]. This is equivalent to the condition $\rho(\mathbf{M}^{-1}\mathbf{N}) < 1$, where $\rho(\mathbf{X})$ represents the spectral radius of the matrix $\mathbf{X}$, that is, its maximum eigenvalue. Additionally, the lower the spectral radius, the faster the method converges to the solution [12].

SIMs start with an initial guess of $\mathbf{v}^{\text{ext}}$, say $\mathbf{v}^{\text{ext}}_{(0)}$, and then iteratively find $\mathbf{v}^{\text{ext}}_*$ with the recursion $\mathbf{v}^{\text{ext}}_{(k+1)} = \mathbf{M}^{-1}(\mathbf{N}\mathbf{v}^{\text{ext}}_{(k)} + \mathbf{i}^{\text{ext}})$.

In our implementation, we set $\mathbf{v}^{\text{ext}}_{(0)} = \mathbf{M}^{-1}\mathbf{i}^{\text{ext}}$ and we consider the decomposition $\mathbf{G}_{ABCD} = \mathbf{M} - \mathbf{N}$ with

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} + (\alpha\mathcal{I} + \mathbf{B}) & \mathbf{0} \\ \mathbf{0} & \mathbf{D} + (\alpha\mathcal{I} + \mathbf{C}) \end{bmatrix} \tag{7}$$

$$\mathbf{N} = \begin{bmatrix} \alpha\mathcal{I} + \mathbf{B} & -\mathbf{B} \\ -\mathbf{C} & \alpha\mathcal{I} + \mathbf{C} \end{bmatrix} \tag{8}$$

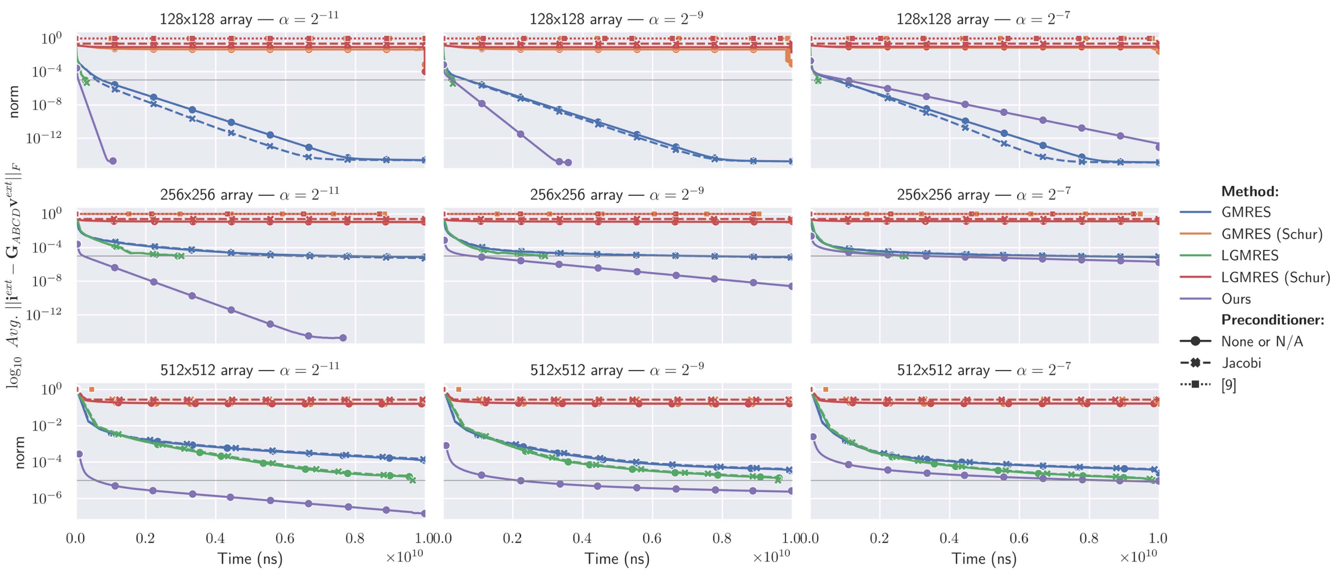

Fig. 3. Average residual vs. computation time (ns) of the proposed approach w.r.t state-of-the-art solutions. Colors in the legend represent the different methods, whereas line styles indicate the preconditioner used. As α gets closer to 1 (line resistance) our convergence slows, but remains linear.

where $\alpha = \mathtt{max}_{i,j}|b_{i,j}| = \mathtt{max}_{i,j}|c_{i,j}|$ and $\mathcal{I}$ is the identity matrix. This selection of α guarantees that all entries of $\mathbf{M}^{-1}$ and $\mathbf{N}$ are non-negative, and also minimizes the number of iterations.

In the event of addressing nonlinearities of the memristors or resistive switching devices, our method could be incorporated into an outer Newton-Raphson iteration [16], where $\mathbf{M}$ could remain a constant, whereas $\mathbf{N}$ would include the non-linear behavior.

IV. Numerical Results and Discussion

We test our method against well known iterative methods and preconditioners. It is worth noting that [9] uses GMRES(k) (with default $k = 20$) to solve the Schur complement of the matrix $\mathbf{G}_{ABCD}$. This reduces the size of the system to solve but increases the condition number for small α (which slows convergence). Therefore, we also compare against the baseline GMRES and LGMRES, without preconditioning, and with Jacobi preconditioner, which have sublinear convergence [17] for symmetric (normal) matrices.

In our experiments[1], we generated 5 different $\mathbf{G}$ matrices with entries that follow a normal distribution (we consider $g_i^{\mathtt{W}} = g_j^{\mathtt{S}} = g^{\mathtt{tl}} = g^{\mathtt{bl}} = 1\Omega^{-1}$ and $g_i^{\mathtt{N}} = g_j^{\mathtt{E}} = 10^{-6}\Omega^{-1}$).

Subsequiently, we scale the matrices to test for different values of α, which is the maximum conductance of the memristors in $\mathbf{G}$. Finally, we solved the system for 10 different input vectors $\mathbf{v}^{\mathtt{W}}$.

In Fig. 3 we present the average residual norm obtained for the different realizations $(\sum_x^5 \sum_y^{10} ||\mathbf{i}_{(y)}^{\mathtt{ext}} - \mathbf{G}_{ABCD}^{(x)}\mathbf{v}_{(y)}^{\mathtt{ext}}||_F/50)$. It is shown how the considered methods reduce the norm of the residuals as the the computation progresses. All simulations run for a maximum of 10 seconds or until they reach a halt condition (less than 10^{-6} relative improvement per iteration). The time is in ns as reported by `time.perf_counter_ns()` from `python`'s standard library. We used time as a metric because the alternative methods use non-equivalent iterations, resulting in different times per iteration. We forced the methods to run single threaded using environment variables.

The figure also shows how the different methods behave as a function of the size of the matrix $\mathbf{G}$. From these results the following conclusions can be drawn: not all the investigated methods are able to achieve a reasonable accuracy level (see 10^{-5} residual limit in the figures). Our method provides the higher slope in most of the cases. In addition, our proposed approach provides the best results only when the line resistances are two orders of magnitude below the memristors' least resistive state, which is the case in practice. However as the ratio between the wire and memristor resistances increases ($\alpha = g^{\mathtt{wl}}/\max(g_{ij})$), the spectral radius gets closer to 1, and this reduces the efficiency of our method.

Importantly, the required memory in our method is significantly lesser than in GMRES implementations. In our case, the memory footprint is approximately the vector input size whereas in GMRES(k)-based implementations the required memory is approximately k times the size of the input vector, being the default value $k = 20$.

In summary, we have demonstrated that a more efficient method to solve MCA circuit simulations is achievable. The proposed method scales better with larger structures as needed for the implementation of DL tasks.

[1]The script to run and visualize our results can be found at our github repository (https://github.com/Wireless-Information-Networking/mca_solver).

References

[1] E. Miranda and J. Suñé, "Memristors for neuromorphic circuits and artificial intelligence applications," *Materials*, vol. 13, no. 4, Feb. 2020, Art. no. 938. [Online]. Available: https://doi.org/10.3390/ma13040938

[2] S. Jain et al., "Neural network accelerator design with resistive crossbars: Opportunities and challenges," *IBM J. Res. Dev.*, vol. 63, no. 6, pp. 10:1–10:13, Nov./Dec. 2019. [Online]. Available: https://doi.org/10.1147/jrd.2019.2947011

[3] L. Gao, Q. Ren, J. Sun, S.-T. Han, and Y. Zhou, "Memristor modeling: Challenges in theories, simulations, and device variability," *J. Mater. Chem. C*, vol. 9, no. 47, pp. 16859–16884, 2021. [Online]. Available: https://doi.org/10.1039/d1tc04201g

[4] M. E. Fouda, S. Lee, J. Lee, A. Eltawil, and F. Kurdahi, "Mask technique for fast and efficient training of binary resistive crossbar arrays," *IEEE Trans. Nanotechnol.*, vol. 18, pp. 704–716, 2019. [Online]. Available: https://doi.org/10.1109/tnano.2019.2927493

[5] M. Pedro, J. Martin-Martinez, R. Rodriguez, M. Gonzalez, F. Campabadal, and M. Nafria, "A flexible characterization methodology of RRAM: Application to the modeling of the conductivity changes as synaptic weight updates," *Solid-State Electron.*, vol. 159, pp. 57–62, Sep. 2019. [Online]. Available: https://doi.org/10.1016/j.sse.2019.03.035

[6] R. Vadivel, M. S. Ali, and Y. H. Joo, "Robust h∞ performance for discrete time T-S fuzzy switched memristive stochasticneural networks with mixed time-varying delays," *J. Exp. Theor. Artif. Intell.*, vol. 33, no. 1, pp. 79–107, Feb. 2020. [Online]. Available: https://doi.org/10.1080/0952813x.2020.1725649

[7] D. Veksler et al., "Random telegraph noise (RTN) in scaled RRAM devices," in *Proc. IEEE Int. Rel. Phys. Symp.*, Apr. 2013, pp. MY.10.1–MY.10.4. [Online]. Available: https://doi.org/10.1109/irps.2013.6532101

[8] F. Aguirre et al., "Hardware implementation of memristor-based artificial neural networks," *Nature Commun.*, vol. 15, no. 1, Mar. 2024, Art. no. 1974. [Online]. Available: http://dx.doi.org/10.1038/s41467-024-45670-9

[9] R. Xie, M. Song, J. Zhou, J. Mei, and Q. Chen, "A fast method for steady-state memristor crossbar array circuit simulation," in *Proc. IEEE Int. Conf. Integr. Circuits, Technol. Appl.*, 2021, pp. 62–64. [Online]. Available: https://doi.org/10.1109/icta53157.2021.9661817

[10] Y. Saad and M. H. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM J. Sci. Stat. Comput.*, vol. 7, no. 3, pp. 856–869, Jul. 1986. [Online]. Available: https://doi.org/10.1137/0907058

[11] C. Lammie and M. R. Azghadi, "MemTorch: A simulation framework for deep memristive cross-bar architectures," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2020, pp. 1–5. [Online]. Available: https://doi.org/10.1109/iscas45731.2020.9180810

[12] Y. Saad, "Iterative methods for sparse linear systems," 2nd ed., Philadelphia, PA, USA: Soc. Ind. Appl. Math., Jan. 2003. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9780898718003

[13] A. Chen, "A comprehensive crossbar array model with solutions for line resistance and nonlinear device characteristics," *IEEE Trans. Electron Devices*, vol. 60, no. 4, pp. 1318–1326, Apr. 2013. [Online]. Available: https://doi.org/10.1109/ted.2013.2246791

[14] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge Univ. Press, Dec. 1985. [Online]. Available: http://dx.doi.org/10.1017/CBO9780511810817

[15] R. Plemmons, "M-matrix characterizations. I–nonsingular M-matrices," *Linear Algebra Appl.*, vol. 18, no. 2, pp. 175–188, 1977. [Online]. Available: https://doi.org/10.1016/0024-3795(77)90073-8

[16] E. Ngoya, J. Rousset, and J. Obregon, "Newton-Raphson iteration speed-up algorithm for the solution of nonlinear circuit equations in general-purpose CAD programs," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 16, no. 6, pp. 638–644, Jun. 1997. [Online]. Available: http://dx.doi.org/10.1109/43.640621

[17] E. Vecharynski and J. Langou, "The cycle-convergence of restarted GMRES for normal matrices is sublinear," *SIAM J. Sci. Comput.*, vol. 32, no. 1, pp. 186–196, Jan. 2010. [Online]. Available: http://dx.doi.org/10.1137/080727403