ReRAM-Based Process-In-Memory Accelerator for Iterative Solvers: A Systematic Survey

Boyu Geng[1], Mingjia Fan[1], Zhou Jin[2], Weifeng Liu[1]
1. SSSLab, Dept. of CST, China University of Petroleum-Beijing, China
2. College of Integrated Circuits, Zhejiang University, Hangzhou, China
Email: jinzhou@cup.edu.cn

2025 IEEE International Symposium on Circuits and Systems (ISCAS) | 979-8-3503-5683-0/25/$31.00 ©2025 IEEE | DOI: 10.1109/ISCAS56072.2025.11043455

***Abstract*—Iterative solvers are fundamental in scientific computing, particularly for solving large-scale linear equations, which are central to a variety of applications such as simulations and data analysis. Traditional optimization strategies for iterative solvers, however, are predominantly designed around von Neumann architectures, which suffer from significant data movement costs and the "memory wall" problem, limiting overall computational performance. In this context, processing-in-memory (PIM) architectures, especially those utilizing resistive random-access memory (ReRAM), offer a promising alternative by enabling in-situ computing, thereby reducing data movement and overcoming the storage bottleneck. These architectures have already shown substantial potential in accelerating tasks like neural network training and graph computations, and they provide new opportunities for optimizing iterative solvers. This paper systematically surveys ReRAM-based iterative solver accelerators, categorizing key contributions into four main areas: mixed-precision techniques, feedback circuit theory, floating-point computation support, and leveraging content-addressable memory (CAM) to address irregularity and sparsity. We also discuss four future research directions aimed at further improving iterative solver performance.**

***Index Terms*—ReRAM, sparse iterative solver, process-in-memory, accelerator, hardware-software codesign.**

I. Introduction

Solving a series of sparse linear systems is fundamental to scientific computing, underpinning a wide range of applications from simulations in quantum mechanics to optimization in machine learning [1]–[4]. Traditional iterative solvers, while effective, often face limitations due to their reliance on conventional von Neumann architectures [5]. This dependence leads to significant data movement overhead between processors and main memories, which can severely impact computational performance and power consumption.

In response to these challenges, processing-in-memory (PIM) architectures have emerged as a transformative approach, enabling computation to occur closer to where data is stored [6]–[10]. Among these, resistive random-access memory (ReRAM) stands out for its potential to facilitate in situ computing, thereby alleviating data transfer bottlenecks and improving overall system efficiency. ReRAM, an advanced non-volatile memory, offers high density and excellent CMOS integration. ReRAM's crossbar array can perform matrix multiplication at O(1) complexity.

The PIM architecture based on ReRAM has achieved significant success in neural network and graph computation [11]–[16]. However, its application in scientific computing, particularly in iterative solvers, remains limited. This is primarily due to several challenges, e.g., the presence of complex and irregular operators that are difficult to map onto a crossbar structure, sparse matrix computations, and the need for high floating-point precision in iterative solvers, etc.

This paper reviews the existing research on accelerated iterative solvers using ReRAM, categorizing the efforts into four main directions: (1) Mixed-precision strategy, (2) Feedback circuit, (3) Enabling floating-point computations, (4) Addressing irregularity and sparsity with content-addressable memory (CAM). Table I shows the summary Based on our analysis of current work, we identify four key challenges and future directions: (1) Implementing complex operators that cannot be simply mapped to Multiply Accumulate (MAC), (2) Supporting general and irregular sparse matrix structures, (3) Ensuring floating point precision, and (4) Overcoming the inherent non-ideality of ReRAM and other devices. Future research should focus on hardware-software co-design to enable high-performance, energy-efficient, and scalable iterative solvers on PIM architectures, unlocking their potential for a broader range of scientific computing applications.

II. Background

A. Iterative Solvers

Iterative solvers are algorithms designed to approximate the solution of a linear system $Ax = b$ by iteratively refining an initial guess. They are especially efficient for large, sparse matrices, where direct solvers, such as LU or Cholesky decomposition, are often impractical due to their high computational complexity and memory requirements [30]–[35]. The advantages of iterative solvers include their ability to efficiently handle large-scale problems, their potential for parallelization, and their suitability for situations where the matrix changes incrementally or an approximate solution is sufficient.

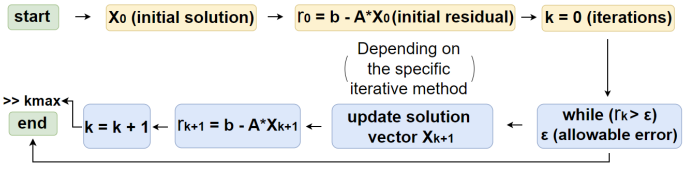

Fig. 1: An example of computational flow in iterative solvers.

TABLE I: Summary of ReRAM based PIM accelerators for iterative solvers.

Category	Citation	Solver	Baseline	Speedup	Energy Efficiency	Iteration	Sparsity	Year
Mixed-precision strategy	[17]	CG	NVIDIA Tesla K10 GPU	1500×	8.5×	✔	✔	2015
	[18]	GMRES	IBM POWER8 CPU, NVIDIA P100 GPU	6.3−17.5× (CPU), 3.6−7.8× (GPU)	6.8 − 24×	✔	✔	2018
	[19]	Block-Jacobi preconditioned flexible GMRES	2.3 GHz 8-Core Intel i9 machine equipped with 64 GB of system memory	FGMRES required 2× to 4× fewer FLOPS than PGMRES + ILU	——	✔	✖	2023
Based on the feedback circuit theory	[20]	——	——	O(1)	——	✖	✖	2019
	[21]	——	——	O(logN) or O(1) (model covariance), O($(1/\lambda_{min})$)	——	✖	✔	2020
	[22]	Eigenvector	——	O(1)	——	✖	✖	2020
	[23]	Jacobi iterative method	NVIDIA Tesla P40 GPU	100×	1000×	✖	✔	2021
	[24]	Least-Squares Fitting	NVIDIA K40m GPU	132 − 3282×	8201 − 96738×	✖	✖	2022
	[25]	——	original AMC	——	1.6 − 1.67×	✖	✖	2024
Enabling floating-point computations	[26]	CG, BiCG	NVIDIA Tesla P100 GPU	10.3×	10.9×	✔	✔	2018
	[27]	CG, BiCGSTAB	NVIDIA Tesla V100 GPU, PIM accelerator [26]	12.59× (CG GPU), 12.94× (CG PIM), 13.34× (BiCGSTAB GPU), 15.98× (BiCGSTAB PIM)	——	✔	✔	2023
Dealing with irregularity and sparsity with CAM	[28]	AMG	AMD 2nd EPYC 7702 CPU, NVIDIA Tesla A100 GPU	10× (CPU), 100× (GPU)	100× (CPU), 1000× (GPU)	✔	✖	2023
	[29]	JPCG	AMD 2nd EPYC 7702 CPU, NVIDIA Tesla A100 GPU, Xilinx Alveo U280 FPGA	1000× (CPU), 10× (GPU), 10× (FPGA)	100× (CPU), 100× (GPU), 10× (FPGA)	✔	✔	2024

B. ReRAM-Based Process-in-Memory Architecture

ReRAM is an emerging non-volatile storage technology based on a metal-insulator-metal (MIM) structure [28]. Its resistance-switching mechanism involves adjusting the resistance of a memory cell between a high resistance state (HRS) and a low resistance state (LRS), with information retained even after power loss. In terms of applications, ReRAM enables efficient MAC operations, which are essential for matrix computations, particularly in deep learning and scientific computing. Additionally, ReRAM's analog CAM design significantly enhances data density, reduces operational energy consumption [29], and minimizes area by matching analog inputs to stored values. Furthermore, ReRAM-based designs can support functionalities similar to Ternary CAM (TCAM), eliminating the need for analog-to-digital conversion and further reducing power consumption.

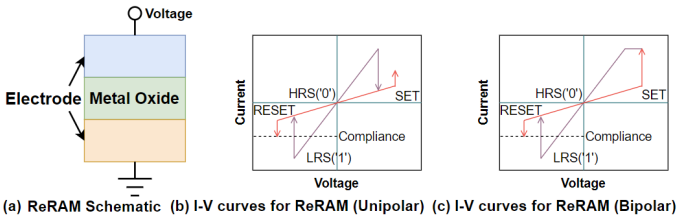

Fig. 2: The basic of ReRAM.

III. ReRAM-based PIM for Iterative Solvers

A. Mixed-Precision Strategy

Mixed-precision strategies, which combine low-precision and high-precision computing, have long been explored to accelerate iterative solvers while maintaining solution accuracy. This approach is particularly well-suited for ReRAM-based architectures, which excel at providing low-precision initial solutions. ReRAM's ability to perform matrix-vector multiplications with low precision allows these initial estimates to be generated quickly, significantly reducing the number of iterations needed to solve large-scale problems. These computationally intensive tasks benefit from ReRAM's fast analog processing capabilities, while high-precision refinements are handled by the CPU in digital domain to ensure accuracy.

This hybrid strategy is an effective way to accelerate iterative solvers while maintaining both performance and solution quality. Several studies, including those by Richter et al. [17] and Kalantzis et al. [19], have demonstrated the effectiveness of this approach using memristor-based arrays such as ReRAM and PCM [18]. However, the hybrid nature of this approach necessitates collaboration with conventional CPU architectures, as critical steps (e.g., residual calculations and orthogonalization) still rely on digital precision to ensure algorithmic robustness. This limitation highlights a trade-off between analog efficiency and digital reliability, suggesting that full utilization of PIM architectures for end-to-end acceleration remains challenging.

B. Based on the Feedback Circuit Theory

Feedback-based circuit architectures have emerged as a paradigm-shifting methodology for equation solving, offering an alternative to traditional iterative algorithms by enabling direct, single-step solutions through physical circuit dynamics. The foundational work by Sun et al. [20] pioneered this approach, demonstrating that ReRAM crossbar arrays integrated with feedback circuits could solve linear systems and compute eigenvectors in situ. By leveraging the intrinsic interplay of Ohm's law and Kirchhoff's circuit laws, this framework translated iterative mathematical operations into analog physical processes, achieving solutions in a single computational step without the need for iterative refinement.

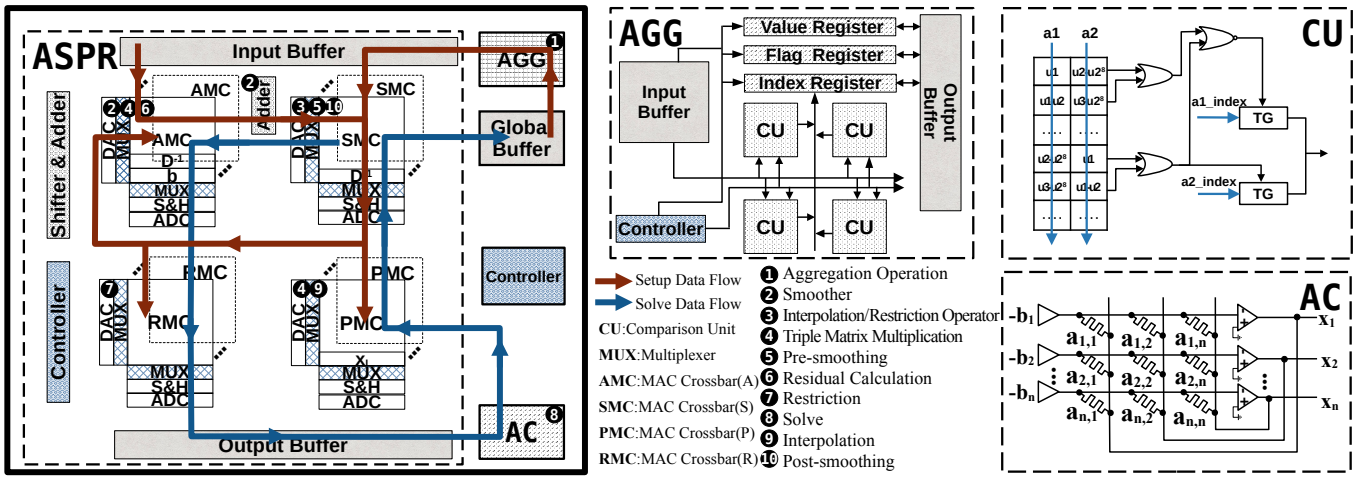

Fig. 3: The architecture of AmgR [28].

Subsequent studies further expanded the theoretical and practical scope of this paradigm. Sun et al. [21] quantified its advantages, revealing ultra-low time complexity: O(1) for covariance matrices and O(log N) for more general structured systems. Song et al. [23] demonstrated that classical iterative solvers (e.g., Jacobi and Gauss-Seidel methods) could be re-engineered on ReRAM crossbars with iteration-free convergence. Their closed-loop feedback design bypassed traditional convergence conditions, enabling instant solutions through hardware-driven equilibrium. These approaches can also be extended to solve eigenvalue problems [22] and least-squares fitting problems [24], broadening their application to other important matrix-related tasks with high efficiency.

Despite these advances, scalability remains a critical challenge due to the inherent limitations of analog ReRAM hardware, including manufacturing variability, device non-idealities (e.g., conductance drift), and prohibitive costs for large-scale integration. To address these bottlenecks, Pan et al. [25] proposed BlockAMC, a scalable architecture that partitions large matrices into sub-blocks distributed across multiple ReRAM arrays. By localizing analog computations within smaller, more manageable blocks, BlockAMC mitigates error propagation while preserving the O(1) time complexity advantage, thereby bridging the gap between analog efficiency and practical scalability. While this paradigm shows immense potential for real-time, energy-efficient computing, significant challenges remain in scaling to large matrices and achieving high-precision accuracy, particularly in the presence of analog hardware imperfections.

C. Enabling Floating-Point Computations

For scientific computing, solving equations often requires high precision, necessitating the use of floating-point arithmetic. However, traditional ReRAM arrays perform fixed-point computations, posing a challenge for efficiently realizing floating-point operations. Feinberg et al. proposed several strategies to enable floating-point scientific computing on memristive accelerators [26], including exploiting exponent range locality, early termination of fixed-point computation, and static operation scheduling. The design of this work aims to reduce the number of bits used for the exponent by truncating several higher-order bits. While this truncation, along with mantissa padding, reduces the bit requirements, it could potentially compromise the convergence of iterative solvers.

To overcome these limitations, Song et al. introduced ReFloat [27], a data format and accelerator architecture tailored for floating-point processing in ReRAM. By leveraging exponent value locality, ReFloat significantly reduced the number of bits needed to represent floating-point numbers and optimized the exponent base. This mitigated convergence issues, lowered hardware costs, minimized crossbars and processing cycles. As a result, ReFloat offered a low-cost, high-performance solution for floating-point processing in ReRAM, greatly improving upon previous work and meeting the precision demands of scientific computing.

Scientific computing imposes significantly higher precision requirements compared to applications like neural networks and graph processing, which often tolerate lower numerical accuracy. While ReRAM architectures excel in supporting fixed-point and low-precision floating-point computations, making them ideal for neural networks and graph-based tasks, they currently fall short of meeting the stringent precision demands of scientific computing. Although recent advancements have made strides in developing cost-effective high-precision solutions, further research is essential to optimize ReRAM-based accelerators for the extreme accuracy and reliability required in scientific computing applications.

D. Dealing with Irregularity and Sparsity with CAM

To address the irregular operations and discontinuous data access patterns inherent in sparse matrix computations, CAM has emerged as a transformative technology. By leveraging CAM's parallel search capabilities, novel methodologies now efficiently resolve the scattered and non-deterministic memory access challenges of sparse matrices, significantly improving computational efficiency, reducing overhead, and enabling high-performance numerical processing.

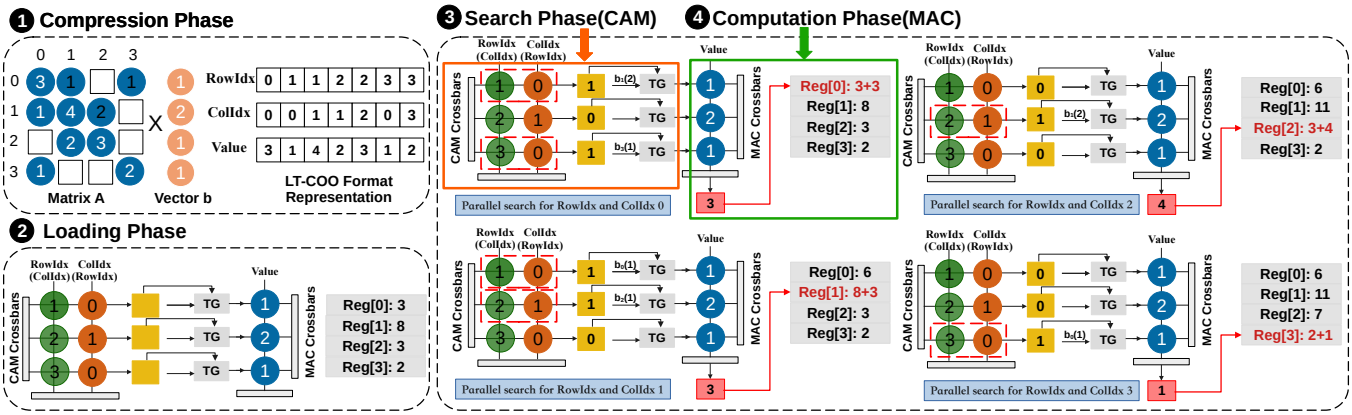

Fig. 4: Implement SpMV on ReRAM, including Compression, Loading, Search, and Computation Phases [29].

The AmgR framework [28] exemplifies this innovation. Designed as a ReRAM-based PIM architecture, AmgR accelerated Algebraic Multigrid (AMG) solvers by classifying AMG kernels into three reusable hardware modules: aggregation operation (handled by the AGG unit), multiplication of the inverse matrix by a vector (solving coarse grid equations, managed by the AC unit), and arithmetic between vectors and matrices (executed by the ASPR unit), as shown in Fig. 3. A dedicated AGG unit using analog CAM addressed the irregularity of aggregation operations. Specifically, the weight of each undirected edge and the index of its corresponding node were input into the CU in the AGG. By comparing the weights of the undirected edges, the CU selected the top k largest edges, and marks the index of the corresponding node with registers. In each iteration, the nodes corresponding to the top k largest edges were gathered into a coarse node. Repeat the above process until the entire coarsening process was complete. Furthermore, to mitigate ReRAM's poor write endurance, AmgR introduced a new mapping strategy that reduced data handling and writes, prolonging ReRAM's endurance. Experimental results demonstrated AmgR's effectiveness, achieving significant performance improvements and energy consumption reductions compared to HYPRE on CPU and AmgX on GPU.

Building on this, a ReCG architecture [29] was proposed by Fan et al., which is tailored to accelerate sparse Conjugate Gradient (CG) solvers. ReCG addressed two critical challenges: mapping sparse irregular operations onto regular ReRAM crossbars, and optimizing dataflows to alleviate ReRAM write durability concerns. For SpMV operation, the most time-consuming phase in CG solver, ReCG introduced a sparse matrix compressed storage format and a four-phase workflow (as shown in Fig. 4). During the compression phase, ReCG stored only the lower triangular portion of the sparse matrix, significantly reducing data storage and transmission costs. The loading phase involved writing non-diagonal element indices to CAM crossbars and their corresponding values to MAC crossbars. The search phase executed concurrent searches for row and column indexes using CAM, while the computation phase leveraged MAC arrays to perform calculations based on the search outcomes. This approach achieves significant performance gains and energy savings over CPU, GPU, and FPGA accelerators.

Unlike neural networks, which primarily rely on structured or block-sparse matrices, scientific computing must handle arbitrarily unstructured sparse matrices generated by physical simulations. These matrices exhibit irregular non-zero distributions, complicating optimization. Furthermore, scientific operators (e.g., multigrid solvers) demand far greater algorithmic complexity than the matrix-vector multiplications common in machine learning. The integration of CAM into ReRAM-based PIM architectures, as realized in AmgR and ReCG, marks a paradigm shift in sparse matrix computation. By co-designing hardware and algorithms to address irregular data access and operational patterns, these frameworks achieve unprecedented efficiency and energy savings, setting a new benchmark for high-performance scientific computing.

IV. CONCLUSION

In this paper, we provide a comprehensive discussion on the latest advancements in iterative solver accelerators leveraging ReRAM technology. Key aspects covered include mixed-precision strategies, feedback circuits, enabling floating-point computations, and dealing with irregularity and sparsity with CAM. Building upon existing research findings, we identify four pivotal challenges that necessitate resolution in future investigations: **implementing complex operators that defy straightforward mapping to MAC operations; accommodating general and irregular sparse matrix structures; achieving high-precision floating-point arithmetic; and mitigating the inherent non-idealities of ReRAM devices.** Addressing these challenges will furnish invaluable guidance for the design and deployment of high-performance iterative solvers, thereby establishing a robust foundation for the advancement of scientific computing in the foreseeable future.

ACKNOWLEDGMENT

This work was supported by NSFC (Grant No. U23A20301, 62204265, 62234010), the State Key Laboratory of Computer Architecture (ICT, CAS) (No. CARCHA202115). Zhou Jin is the corresponding author.

REFERENCES

[1] Y. Lu, L. Zeng, T. Wang, X. Fu, W. Li, H. Cheng, D. Yang, Z. Jin, M. Casas, and W. Liu, "Amgt: Algebraic multigrid solver on tensor cores," in *SC*, 2024.

[2] Y. Bai, X. Yang, Y. Lu, D. Niu, C. Zhuo, Z. Jin, and W. Liu, "Efficient spectral-aware power supply noise analysis for low-power design verification," in *DATE*, 2024.

[3] D. Yang, Y. Zhao, Y. Niu, W. Jia, E. Shao, W. Liu, G. Tan, and Z. Jin, "Mille-feuille: A tile-grained mixed precision single-kernel conjugate gradient solver on gpus," in *SC*, 2024.

[4] Y. Zhao, X. Yang, Y. Bai, L. Zeng, D. Niu, W. Liu, and Z. Jin, "Csp: Comprehensively-sparsified preconditioner for efficient nonlinear circuit simulation," in *ICCAD*, 2024.

[5] Y. Lu and W. Liu, "Dasp: Specific dense matrix multiply-accumulate units accelerated general sparse matrix-vector multiplication," in *SC*, 2023.

[6] E. Yi, Y. Duan, Y. Bai, K. Zhao, Z. Jin, and W. Liu, "Cuper: Customized dataflow and perceptual decoding for sparse matrix-vector multiplication on hbm-equipped fpgas," in *DATE*, 2024.

[7] Y.-J. Hsiao, C.-F. Nien, and H.-Y. Cheng, "Respar: Reordering algorithm for reram-based sparse matrix-vector multiplication accelerator," in *ICCD*, 2021.

[8] E. Ipek, "Memristive accelerators for dense and sparse linear algebra: from machine learning to high-performance scientific computing," 2019.

[9] X. Zhang, Z. Li, R. Liu, X. Chen, and Y. Han, "Gas: General-purpose in-memory-computing accelerator for sparse matrix multiplication," *IEEE Transactions on Computers*, 2024.

[10] X. Yang, B. Taylor, A. Wu, Y. Chen, and L. O. Chua, "Research progress on memristor: From synapses to computing systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2022.

[11] C.-Y. Wang, Y.-W. Chang, and Y.-H. Chang, "Sgirr: Sparse graph index remapping for reram crossbar operation unit and power optimization," in *ICCAD*, 2022.

[12] H. Jin, C. Liu, H. Liu, R. Luo, J. Xu, F. Mao, and X. Liao, "Rehy: A reram-based digital/analog hybrid pim architecture for accelerating cnn training," *IEEE Transactions on Parallel and Distributed Systems*, 2021.

[13] T. Yang, D. Li, Y. Han, Y. Zhao, F. Liu, X. Liang, Z. He, and L. Jiang, "Pimgcn: A reram-based pim design for graph convolutional network acceleration," in *DAC*, 2021.

[14] S. A. Ghasemi, B. Jahannia, and H. Farbeh, "Grapha: An efficient reram-based architecture to accelerate large scale graph processing," *Journal of Systems Architecture*, 2022.

[15] P.-Y. Chen, F.-Y. Gu, Y.-H. Huang, and C. Lin, "Wrap: Weight remapping and processing in rram-based neural network accelerators considering thermal effect," in *DATE*, 2022.

[16] C. Liu, H. Liu, H. Jin, X. Liao, Y. Zhang, Z. Duan, J. Xu, and H. Li, "Regnn: a reram-based heterogeneous architecture for general graph neural networks," in *DAC*, 2022.

[17] I. Richter, K. Pas, X. Guo, R. Patel, J. Liu, E. Ipek, and E. G. Friedman, "Memristive accelerator for extreme scale linear solvers," in *GOMACTech*, 2015.

[18] M. Le Gallo, A. Sebastian, R. Mathis, M. Manica, H. Giefers, T. Tuma, C. Bekas, A. Curioni, and E. Eleftheriou, "Mixed-precision in-memory computing," *Nature Electronics*, 2018.

[19] V. Kalantzis, M. S. Squillante, C. W. Wu, A. Gupta, S. Ubaru, T. Gokmen, and L. Horesh, "Solving sparse linear systems via flexible gmres with in-memory analog preconditioning," in *HPEC*, 2023.

[20] Z. Sun, G. Pedretti, E. Ambrosi, A. Bricalli, W. Wang, and D. Ielmini, "Solving matrix equations in one step with cross-point resistive arrays," *Proceedings of the National Academy of Sciences*, 2019.

[21] Z. Sun, G. Pedretti, P. Mannocci, E. Ambrosi, A. Bricalli, and D. Ielmini, "Time complexity of in-memory solution of linear systems," *IEEE Transactions on Electron Devices*, 2020.

[22] Z. Sun, G. Pedretti, E. Ambrosi, A. Bricalli, and D. Ielmini, "In-memory eigenvector computation in time o (1)," *Advanced Intelligent Systems*, 2020.

[23] T. Song, X. Chen, and Y. Han, "Eliminating iterations of iterative methods: Solving large-scale sparse linear system in o(1) with rram-based in-memory accelerator," in *GLSVLSI*, 2021.

[24] X. Chen and Y. Han, "Solving least-squares fitting in $o(1)$ using rram-based computing-in-memory technique," in *ASP-DAC*, 2022.

[25] L. Pan, P. Zuo, Y. Luo, Z. Sun, and R. Huang, "Blockamc: Scalable in-memory analog matrix computing for solving linear systems," in *DATE*, 2024.

[26] B. Feinberg, U. K. R. Vengalam, N. Whitehair, S. Wang, and E. Ipek, "Enabling scientific computing on memristive accelerators," in *ISCA*, 2018.

[27] L. Song, F. Chen, H. Li, and Y. Chen, "Refloat: Low-cost floating-point processing in reram for accelerating iterative linear solvers," in *SC*, 2023.

[28] M. Fan, X. Tian, Y. He, J. Li, Y. Duan, X. Hu, Y. Wang, Z. Jin, and W. Liu, "Amgr: Algebraic multigrid accelerated on reram," in *DAC*, 2023.

[29] M. Fan, X. Chen, D. Yang, Z. Jin, and W. Liu, "Recg: Reram-accelerated sparse conjugate gradient," in *DAC*, 2024.

[30] Y. Saad, *Iterative methods for sparse linear systems*. SIAM, 2003.

[31] D. Niu, Y. Tao, Z. Jin, Y. Dong, C. Wang, and C. Sun, "Islu: Indexing-efficient sparse lu factorization for circuit simulation on gpus," in *ICCAD*, 2024.

[32] G. Feng, H. Wang, Z. Guo, M. Li, T. Zhao, Z. Jin, W. Jia, G. Tan, and N. Sun, "Accelerating large-scale sparse lu factorization for rf circuit simulation," in *Euro-Par*, 2024.

[33] X. Fu, B. Zhang, T. Wang, W. Li, Y. Lu, E. Yi, J. Zhao, X. Geng, F. Li, J. Zhang *et al.*, "Pangulu: A scalable regular two-dimensional block-cyclic sparse direct solver on distributed heterogeneous systems," in *SC*, 2023.

[34] Z. Jin, W. Li, Y. Bai, T. Wang, Y. Lu, and W. Liu, "Machine learning and gpu accelerated sparse linear solvers for transistor-level circuit simulation: A perspective survey," in *ASP-DAC*, 2024.

[35] J. Zhao, Y. Wen, Y. Luo, Z. Jin, W. Liu, and Z. Zhou, "Sflu: Synchronization-free sparse lu factorization for fast circuit simulation on gpus," in *DAC*, 2021.

[36] T. Wang, W. Li, H. Pei, Y. Sun, Z. Jin, and W. Liu, "Accelerating sparse lu factorization with density-aware adaptive matrix multiplication for circuit simulation," in *ICCAD*, 2023.

[37] M. A. Zidan, Y. Jeong, J. Lee, B. Chen, S. Huang, M. J. Kushner, and W. D. Lu, "A general memristor-based partial differential equation solver," *Nature Electronics*, 2018.