SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

February 2023, Vol. 66 122402:1–122402:11
https://doi.org/10.1007/s11432-021-3374-x

A memristive neural network based matrix equation solver with high versatility and high energy efficiency

Jiancong LI†, Houji ZHOU†, Yi LI* & Xiangshui MIAO

[1]*School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan* 430074*, China;*
[2]*School of Integrated Circuits, Huazhong University of Science and Technology, Wuhan* 430074*, China;*
[3]*Wuhan National Laboratory for Optoelectronics, Wuhan* 430074*, China*

Received 30 August 2021/Revised 9 October 2021/Accepted 16 November 2021/Published online 4 November 2022

Abstract As the main topic in modern scientific computing and machine learning tasks, matrix equation solving is suffering high computational latency and tremendous power consumption due to the frequent data movement in traditional von Neumann computers. Although the in-memory computing paradigms have shown the potential to accelerate the execution of solving matrix equations, the existing memristive matrix equation solvers are still limited by the low system versatility and low computation precision of the memristor arrays. In this work, we demonstrate a hybrid architecture for accurate, as well as efficient, matrix equation solving problems, where the memristive crossbar arrays are used for the parallel vector-matrix multiplication and the digital computer for accuracy. The linear neural-network solving (NNS) method is adopted here and its versatility for various types of matrix equations is proved. The weight-slice computation method is developed to perform the analog matrix multiplication with high efficiency and high robustness in the array. The solution results confirmed that typical matrix equations can be solved by this memristive matrix equation solver with high accuracy. Further performance benchmarking demonstrates that the generalized memristive matrix equation solver has low solving time-complexity while outperforming the state-of-the-art CMOS and in-memory processors.

Keywords matrix equation solving, memristor, linear neural network, matrix-multiplication, analog computing

Citation Li J C, Zhou H J, Li Y, et al. A memristive neural network based matrix equation solver with high versatility and high energy efficiency. Sci China Inf Sci, 2023, 66(2): 122402, https://doi.org/10.1007/s11432-021-3374-x

1 Introduction

Matrix equation solving has become a fundamental problem in the mathematical field and plays an essential role in various modern data-intensive computation applications including machine learning, scientific computing [1]. Matrix equations in these real-world workloads involve two types according to the shape of the coefficient matrix A $(m \times n)$. For example, the coefficient matrices in the scientific computing problems are usually squared and non-singular ones $(m = n)$ while they are non-squared ones $(m \neq n)$ in other tasks like machine learning and compress sensing. The shape of the coefficient matrix affects the solution process greatly, thus various mathematical solving algorithms have been developed. Typically, these matrix equations solving methods rely on performing matrix factorizations or iterative matrix multiplications in traditional von Neumann computers [2,3]. However, the performance of the von-Neumann computer is challenged by the high-power consumption and high latency of data communication between the memory and computing unit. This makes the solutions of matrix equations computationally expensive in energy efficiency as the matrix dimension increases dramatically.

In-memory computing with emerging memristors has shown a remarkable potential to break through the communication bottleneck in von Neumann architecture. In the crossbar architecture, a memristor array can naturally execute a one-step multiple-and-accumulate (MAC) operation with high parallelism

* Corresponding author (email: liyi@hust.edu.cn)
† Authors Li J C and Zhou H J have the same contribution to this work.

based on Ohm's law and Kirchhoff's law [4–6]. Previous studies have revealed the great acceleration for matrix equation solving in memristor crossbar arrays, including the mixed-precision solver [7], the generalized partial differential equation solver [8], and the one-step analog solver [9]. However, these memristive matrix equation solvers are still facing the limitation of low system versatility for diverse scenes as they are specifically designed for the matrix equations with non-singular coefficient matrices and the situation of non-squared coefficient matrices is less considered. Besides, solving matrix equations typically requires high MAC precision (at least 7-bit) [7,8]. The memristor array, despite that, can only perform low-precision MAC (typically < 5 bit) [6,7] due to the existence of device non-ideal factors, and the existing precision improvement methods always lead to a significant reduction in energy efficiency [7,8]. Thus, the design of a generalized matrix equation solving method which provides high precision and energy efficiency is in demand.

In this work, we proposed a generalized memristive matrix equation solver (GMMS), which utilizes the linear neural network solving (NNS) method to efficiently obtain the solution for the typical types of matrix equations. A hybrid system architecture is designed based on the mixed-precision strategy and the quantization techniques to get the acceleration of NNS. Moreover, a weight-slicing computation (WSC) is developed to perform the analog matrix-multiplication with high accuracy and high robustness. It is confirmed in simulation that the solution of typical matrix equations (including the non-singular matrix equation, rank-deficient problem, overdetermined problem, and underdetermined problem) can approach the analytical results automatically in our GMMS. Further evaluations exhibit low computation complexity and high matrix-multiplication efficiency of the solver, which indicates that the GMMS can be effectively used to process matrix equation intensive workloads in hardware.

2 Solving method and system architecture

A generalized matrix equation can be described as follows:

$$A \cdot X = B, \tag{1}$$

where A is an $m \times n$ coefficient matrix, and B is the known result matrix with the shape of $m \times k$. The X is an $n \times k$ matrix to be solved. The solution of the matrix equation is determined by the size of the coefficient matrix A: (1) $m = n$, if A is a non-singular matrix, this matrix has a unique solution. On the contrary, if A is a rank-deficient matrix, we need to find the optimal solution for the matrix equation. (2) $m \neq n$, we need to find the optimal solution for both overdetermined ($m > n$) and underdetermined ($m < n$) problems. Different matrix equations generally have their special mathematical methods to obtain better solutions, for instance, the iteration method like the Jacobi method is typically used for non-singular problems, while the pseudo-inverse method is employed for the overdetermined ones. These methods ensure the effectiveness of understanding but reduce the solving versatility in the hardware system. Generalized matrix equation solvers are eagerly required to handle various applications. In this work, an NNS method reported in [10] is discussed in detail to investigate its general ability for various types of matrix equations. The main theory of NNS is that an approximate solution of the matrix equations can be obtained by optimizing the residual error $\varepsilon_0 = \|AX - B\|_{\mathrm{F}}$, where $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius-norm of the matrix [3]. The neural network is considered to be a well-performing optimizer and the optimization of the residual error can be performed by the training process of the neural network where A is the training input and B is the target (Figure 1(a)). To match the pattern of the training, Eq. (1) is rewritten as

$$[A_1, A_2, \ldots, A_m]^{\mathrm{T}} X = [B_1, B_2, \ldots, B_m]^{\mathrm{T}}, \tag{2}$$

where A_i and B_i indicate the row vectors of matrix A and B, respectively. A_i $(i = 1, 2, \ldots, n)$ will be put in the network continuously before the solution being updated. The updating rule is determined by the gradient descent method and shown as follows:

$$X_{n+1} = X_n + lr \cdot A^{\mathrm{T}} \cdot (B - a \cdot X_n), \tag{3}$$

where lr is the pre-set update rate. Notably, matrix solving only involves the linear transformation such that the nonlinear activation function is not required.

Furthermore, to accelerate the solving process, a hybrid system is developed for the hardware implementation of NNS which combines the high accuracy digital computer and the parallel memristor

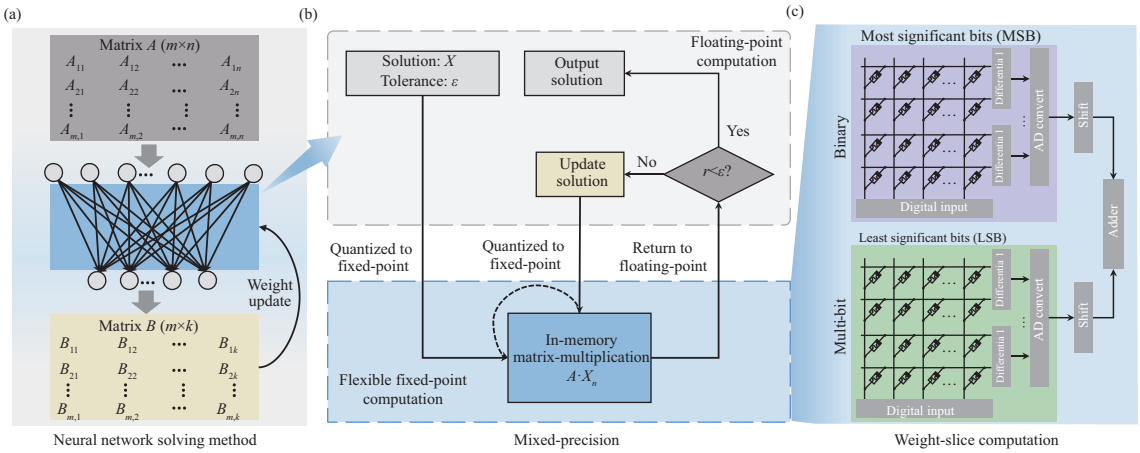

Figure 1 (Color online) Generalized memristive matrix equation solver (GMMS). (a) The solver utilizes the neural network to provide the approximate solution of matrix equations; (b) the designed hybrid architecture to implement the great acceleration of the NNS; (c) the weight-slice computation method for MAC with high accuracy and high efficiency.

arrays [11]. The computational intensive VMM operations are performed efficiently in the fixed-point precision crossbar arrays in parallel. The other operations, which require high precision like solution update, are achieved by the floating-point digital processor. This hybrid system, as well as the solving method, is named GMMS. Thus, this hybrid system gets compromise between high accuracy and high efficiency to solve the typical matrix equations. To solve the matrix equation $AX = B$ in the GMMS, the coefficient matrix A is quantized and transferred to the MAC unit, and the matrix multiplication result $\widehat{B}_n$ will be feedback to the digital computer. The digital computer determines whether the iteration should be terminated by pre-set tolerance ε. Then the updating matrix is calculated by the digital computer and transferred to the memristor array. The device conductance will be reprogrammed by the voltage pulses using the fast programming method [12]. When the iteration ends, the solution matrix X is obtained and in situ stored in the array (Figure 1(b)).

In order to obtain more accurate results in the crossbar array, the bit-slice method is adopted and developed here. Multiple states storage in the memristor array brings higher computation efficiency while reducing the accuracy of the stored data due to the non-ideal factors of the memristor. It is noticeable that the partial error of the most significant bit (MSB) data will lead to a more serious overall error during the computing when compared with the least significant bit (LSB) one. Taking the accuracy and efficiency into consideration at the same time, a WSC method is proposed to improve the MAC precision with high computation efficiency. The p-bit signed number is mapped to m-bit memristors ($p > m$) by the given MSB of the $(p-1)$-bit effective data using the binary code, as the binary code is commonly considered can provide higher accuracy. The LSB data ($(p-2)$-bit) are mapped by giving each memristor an l-bit data ($l < m$, l can divide $(p-1)$) to promise the computing efficiency (Figure 1(c)). Every two columns of memristors are connected and take the difference between two elements by $G = G_+ - G_-$ to map the sign-bit. Notably, the effective-bit of input A_n is mapped to voltage in the same way and the signed-bit is presented by the voltage polarity. For MAC operation, the input data will be converted to the voltage values by a digital-to-analog converter (DAC) array and the analog computation results will be converted to the digital values by the analog-to-digital converter (ADC) array. The final MAC results will be provided by the shift-and-add operation based on the binary arithmetic rule.

3 Matrix equation solving with squared coefficient matrix

Squared coefficient matrices, which have the highest solution requirements, are employed to estimate the performance of GMMS. The squared coefficient matrices ($m = n$) have two types according to their ranks: (1) full rank squared coefficient matrices and their solution can be derived by $X = A^{-1} \cdot B$, where A^{-1} is the inverse matrix of A; (2) rank-deficient matrices, which do not have unique solutions, and need to find the optimal solution that meets the conditions in the solution space.

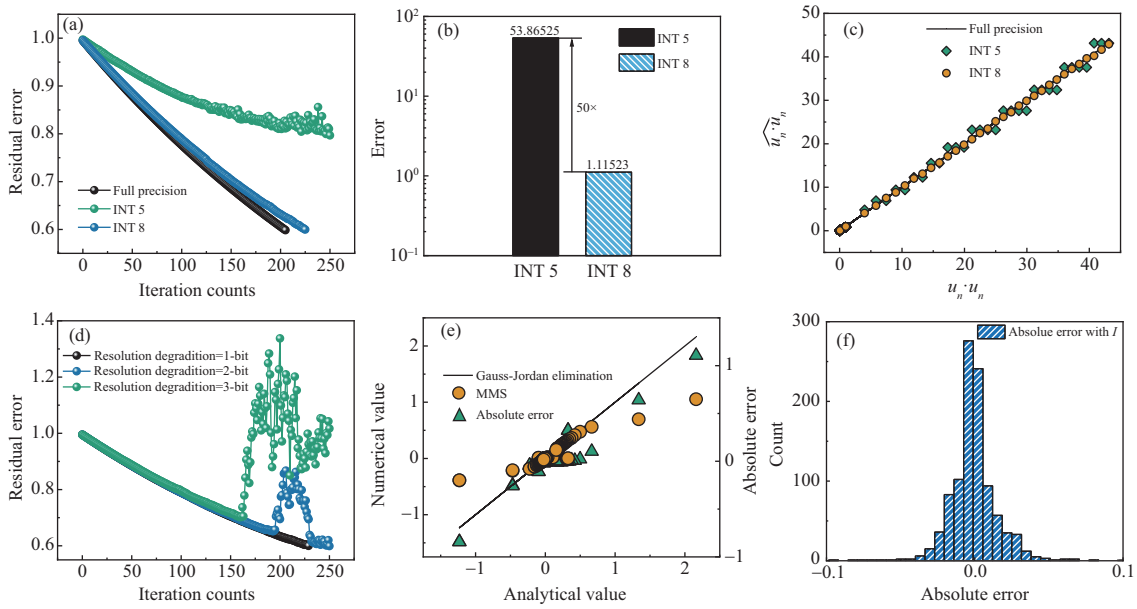

Figure 2 (Color online) (a) Recorded iteration operations under different MAC precision for solving approximate inverse of the covariance problem; (b) comparison of the computing error to calculate the inner dot product of the template vector under different MAC precision; (c) the result of element-by-element multiplication $\widehat{u_n \cdot u_n}$ against the exact value $u_n \cdot u_n$; (d) impact of reducing the ADC resolution for the LSB chip; (e) solution of the covariance matrix inverse problem, also with the absolute error to the analytical value; (f) absolute error distribution of the predicated identity matrix to the analytical identity matrix.

3.1 Non-singular matrix equation

The performance of the GMMS to solve the matrix equation with a non-singular coefficient matrix is evaluated by solving the inverse matrix equation $AX = I$. Here A is an $m \times m$ non-singular matrix and I is an identity matrix with the same shape as A. The solution matrix X is equivalent to the inverse matrix A^{-1}, which is unique for a certain non-singular matrix A. Here, we considered a 32×32 second-order model covariance matrix as the matrix A to present the real-world problems [13], where

$$A_{i,j} = \begin{cases} \dfrac{1}{|i-j|^2}, & \text{if } i \neq j, \\ 1 + \sqrt{i}, & \text{if } i = j. \end{cases} \tag{4}$$

The main influential factors which will affect the solutions of GMMS, including the WSC precision, the write noise of the data storage, and the conditional number of matrix A, are discussed in detail.

3.1.1 *WSC precision*

We first estimate the solver performance under different fixed-point MAC precision. In our simulation, the computation precision of the LSB array is set to 3-bit, and the fixed-point computation precision is extended to INT 5 and INT 8 by the cooperation of the MSB array with one or two LSB arrays, respectively. Here, the resolution of the DAC and the device conductance-states are both 3-bit which is commonly seen in memristive analog computing units [6,14]. The memristive device on/off ratio was set to 103 to guarantee the binary computation accuracy [15]. Figure 2(a) shows that the solution under INT 8 can achieve lower residual error compared with the solution under INT 5. Moreover, the convergence of the solution under INT 8 MAC can be approximated to the situation under full precision MAC. Figure 2(b) presents the computation error to calculate $\|A\|_{\mathrm{F}}$ with different MAC precision when compared with the full-precision MAC. It turns out that the MAC error for INT 5 is 50 times higher than INT 8. Here, the error is defined as Error $= \|A\|_{\mathrm{F}} - \widehat{\|A\|}_{\mathrm{F}}$ where $\|A\|_{\mathrm{F}}$ is the full-precision result and the $\widehat{\|A\|}_{\mathrm{F}}$ is the WSC result. Figure 2(c) shows the result of element-by-element multiplication $\widehat{u_n \cdot u_n}$ plotted against the exact value $u_n \cdot u_n$, where u_n is the matrix element of A . These results illustrate that when quantization accuracy cannot fit the data effectively, the MAC error will increase significantly, and makes it difficult for GMMS to converge. Moreover, the required ADC resolution for analog-to-digital conversion was also

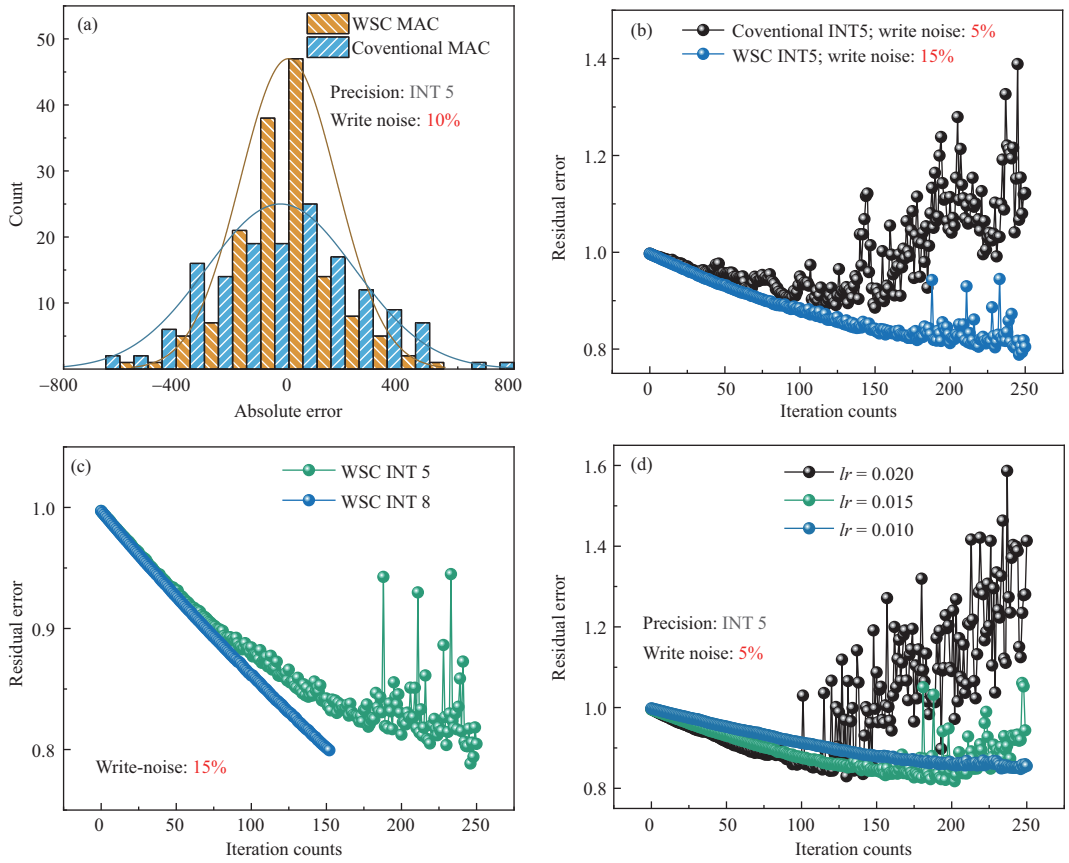

Figure 3 (Color online) (a) Matrix multiplication accuracy comparison between the proposed WSC and the conventional MAC method; (b) comparison of the proposed WSC method with the conventional MAC method to solve the matrix equation; (c) accuracy comparison of the INT 5 WSC method with the INT 8 WSC to solve the matrix equation; (d) accuracy comparison of INT 5 WSC to solve the matrix equation under different lr.

estimated. The simulation result shows that the required ADC resolution should not be 2-bit lower than the full-precision conversion resolution for the LSB array (Figure 2(d)) and should be equivalent to the full-precision conversion resolution for the MSB array. The full-precision analog-to-digital conversion of a memristor-based MAC unit is [8]

$$b_{\mathrm{ADC}} = b_{\mathrm{DAC}} + b_M + \log_2 j, \tag{5}$$

where b_{DAC} is the bit-resolution of the DAC, b_M is the bit-resolution of the memristor (device conductance states), and j is the column number of the device array. Thus, in our simulation, the ADC resolution should not be lower than 10-bit. Figure 2(e) shows the solution $\widehat{A}^{-1}$ obtained by GMMS and the absolute error to the analytical A^{-1} value of the model covariance inverse problem. The absolute error between the predicated $\widehat{I} = \widehat{A}^{-1}A$ and the analytical I is shown in Figure 2(f).

3.1.2 *Write-noise*

Next, the impact of the conductance write noise, which is considered to be a major issue to reduce the MAC accuracy is discussed. Here, we used the WSC and the conventional MAC method [6, 14, 16] to calculate the $\|A\|_{\mathrm{F}}$ under INT 5 precision. The absolute error distribution under 10% write noise within 100 testing cycles demonstrates that the impact of the write-noise has been strongly reduced by the WSC method (Figure 3(a)). The system robustness has been estimated in the covariance matrix inverse solving test, and the simulation result indicates that introducing WSC enables the GMMS to better converge under larger write-noise (Figure 3(b)). Further simulations show that the WSC method can also be effective when applied to higher computation precision (Figure 3(c)). Moreover, we provide two methods to further reduce the impact of the conductance write-noise: (1) using the binary-code to map the front q-bit ($q < p - 2$) of the LSB data is capable to improve the GMMS robustness, but it will reduce the

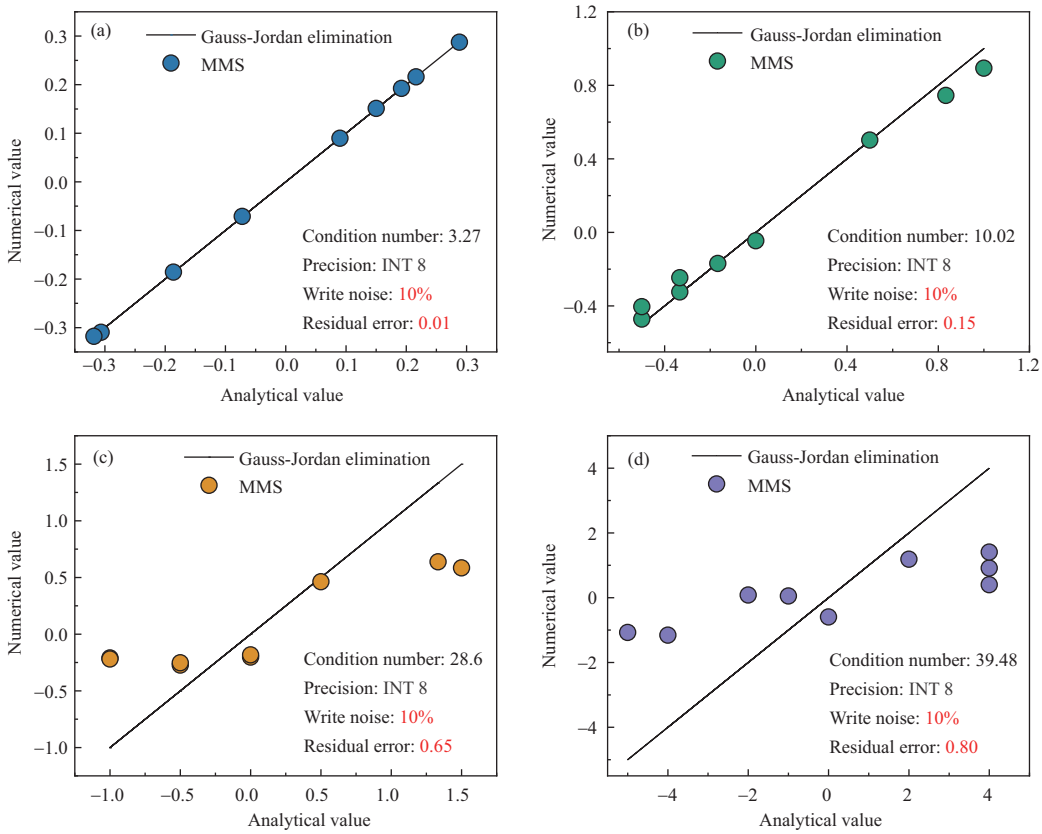

Figure 4 (Color online) The impact of the matrix condition number for solving matrix inverse. For 3×3 matrices composed of discrete values with different condition numbers of (a) $k = 3.27$, (b) $k = 10.03$, (c) $k = 28.60$, and (d) $k = 39.48$.

MAC efficiency; (2) the MAC error can be described by $e = A \cdot X_n - \widehat{A \cdot X_n}$, where $\widehat{A \cdot X_n}$ is the MAC result of the crossbar array. Thus, appropriately reducing the learning rate can make the system achieve better convergence since the solution update error $lr \cdot e$ will be smaller during the iterations. Notably, reducing the lr will increase the iteration cost.

3.1.3 *Matrix condition number*

Moreover, the matrix condition which is regarded as a critical issue that affects the solution accuracy was also discussed [17]. Suppose that the INT 8 WSC is used to guarantee the MAC precision, while the conductance write noise is set to 10%. It must be noted that the same settings of WSC precision and write noise will be used in the subsequent solving tests. Here, the maximum iteration numbers are set to 250. Figure 4 shows the results for solving matrix inverse using the GMMS with different condition numbers (see Appendix B). The solution error significantly increased and the simulated digital value was crucially offset from the analytical value obtained by the Gauss-Jordan elimination. The simulation result points out that a matrix with a large condition number substantially reduces system robustness against the device write noise. For a matrix with an extremely high condition number (known as ill-conditioned matrix), additional strategies such as increasing the iteration cycles, using higher-precision WSC, and utilizing the iterative refinement method should be introduced [7], which can help GMMS to be used as the approximate matrix inverse preconditioner [18].

3.2 Rank-deficient matrix equation

Typically, it is difficult to solve rank-deficient problems which have an infinitude of solutions. Mathematically, the minimum-norm least-square (MNLS) solution of the problem is commonly required [19]. To solve the optimal solution of the rank-deficient problems, not only the optimization of the residual error is required, but also the minimum norm solution stands in need. Here, a matrix equation contains a 3×3 rank-deficient matrix is used as a benchmark (see Appendix B). Simulation results indicate that the residual error will be optimized by the GMMS in the iterative process and eventually converge (Figure 5(a)).

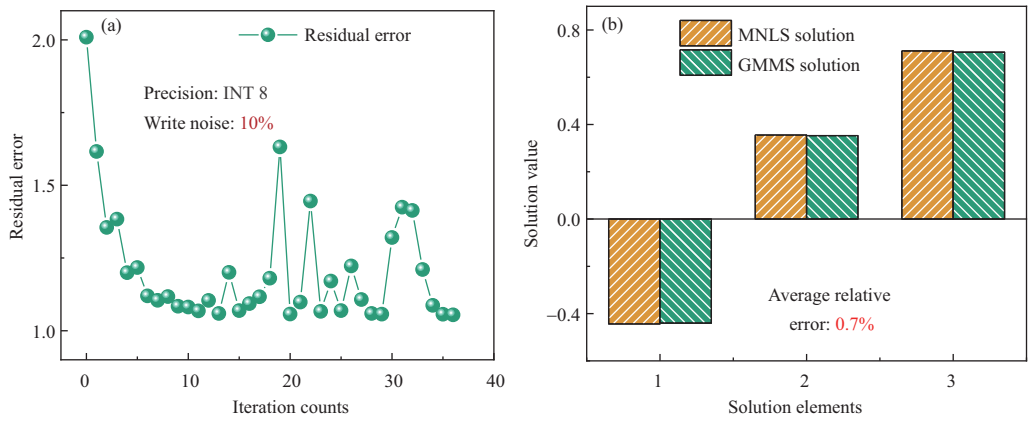

Figure 5 (Color online) Solving rank-deficient problems. (a) Recorded convergence of the residual error for solving the proposed rank-deficient problem; (b) comparison of the GMMS solution with the MNLS solution of the proposed rank-deficient problem.

The average relative error between the analytical MNLS solution and the GMMS solution is 0.7%, which demonstrates that the GMMS can also optimize the matrix norm during the iterations and provide a reliable approximate MNLS solution (Figure 5(b)).

4 Matrix equation solving with non-squared coefficient matrix

The previous simulation has analyzed the effectiveness of using GMMS to solve the matrix equations with square coefficient matrices. Here, we prefer to discuss the effectiveness of GMMS for the matrix equations with non-squared coefficient matrices. According to the shape of matrix A, the matrix equations can be divided as (1) the overdetermined problem for $m > n$, and (2) the underdetermined problem for $m < n$.

4.1 Overdetermined problem

For the overdetermined problems, it is often required to find the least-square (LSQ) solution of the problem, which can be derived by $X = A^{\mathrm{T}} \cdot A^{-1} \cdot A^{\mathrm{T}} \cdot B$ [20]. Here, an overdetermined matrix equation obtained from a polynomial curve fitting problem is used for a toy test. The polynomial function can be illustrated as

$$y(c, w) = w_0 + w_1 X + \cdots + w_M X^M = \sum_{j=0}^{M} w_j X^j, \tag{6}$$

where M is the order of the polynomial, and x^j denotes x raised to the power of j. Note that, although (6) is a nonlinear model, the polynomial coefficients $W = (w_0, \ldots, w_M)^{\mathrm{T}}$ can be obtained by solving linear matrix equations with nonlinear factors. A 5-point dataset (see Appendix B) is used for a 3-order polynomial curve fitting demonstration. Figure 6(a) shows that the average absolute error between the analytical parameter and the GMMS based one is 0.0017, and the residual error has been optimized to 0.015 within 500 cycles (Figure 6(b)). This demonstration represents the capability of the GMMS to perform higher-precision solutions of the overdetermined matrix equation applications. Thus, the GMMS can further be used to implement the overdetermined matrix equation-based machine learning workloads including the linear-regression, classification, and even to training the output layers of a neural network [21] since these tasks commonly have a higher tolerance for solution errors.

4.2 Underdetermined problem

The underdetermined problems require finding the MNLS solution while the equation is a consistent linear system. Here, a matrix equation $AX = B$ with the 4×2 coefficient matrix is used for solving test (see Appendix B). Figure 6(c) shows the iterations of GMMS to solve these underdetermined problems, where the residual error has been reduced to less than 0.5. The solution obtained by GMMS has approached the analytical MNLS value and the average absolute error has been reduced to 0.055 (Figure 6(d)). The simulation results show that, for the consistent underdetermined problems, the GMMS solution can only approach the analytical MNLS solution when the residual error approximates zero, this is the reason to

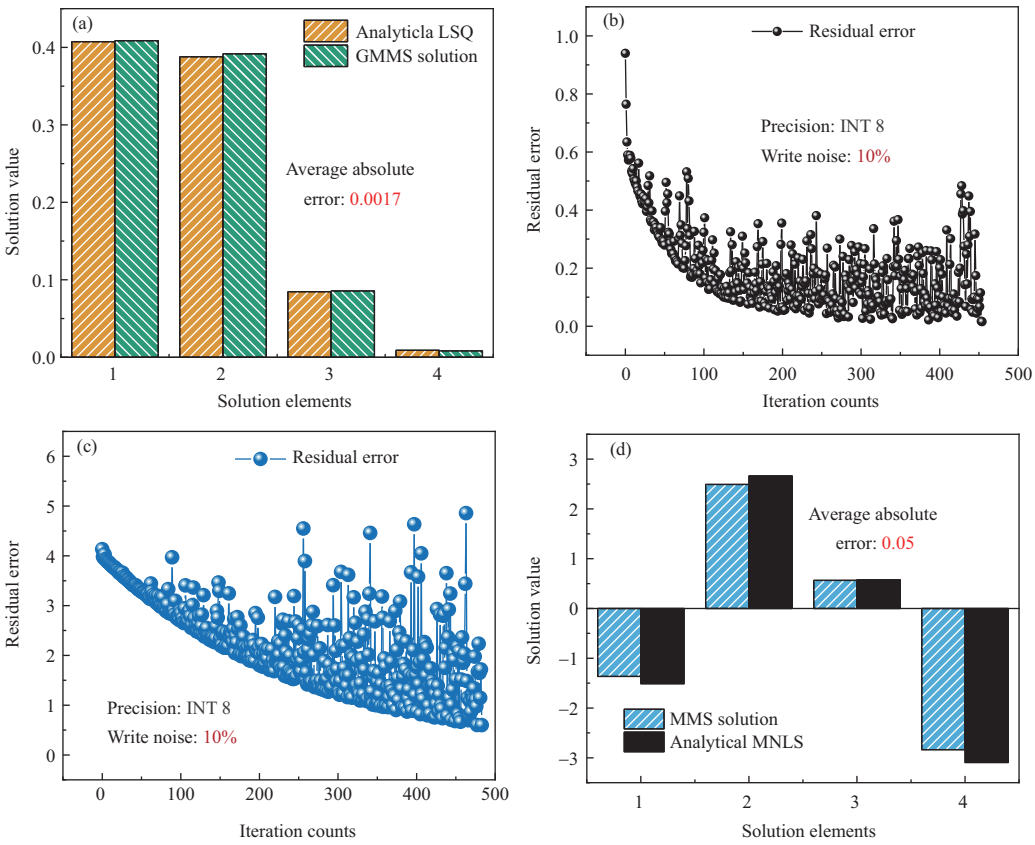

Figure 6 (Color online) Solving overdetermined matrix and underdetermined matrix equations. (a) Comparison between the analytical solution and the GMMS solution for the polynomial curve fitting tasks; (b) decrease of the residual error during the iterations for the polynomial curve fitting tasks; (c) recorded convergence of the residual error for solving the proposed underdetermined problem; (d) comparison of the GMMS solution with the MNLS solution of the proposed underdetermined problem.

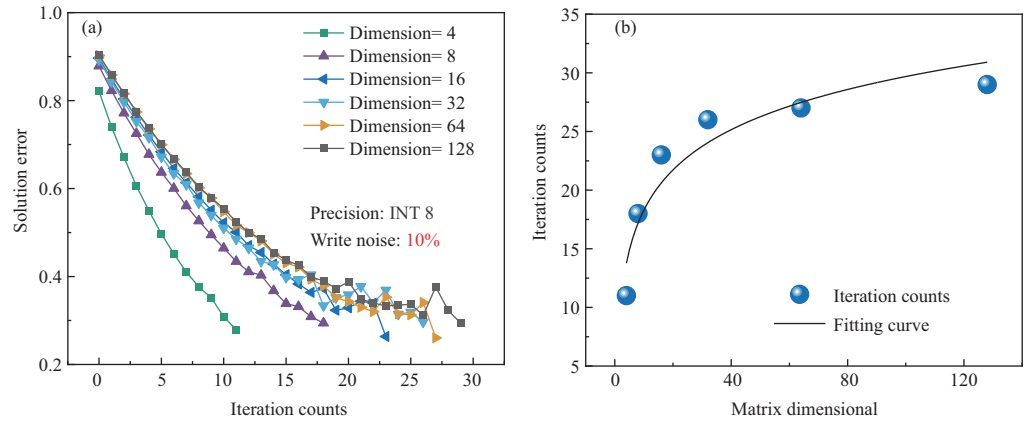

Figure 7 (Color online) Solution time-complexity estimation. (a) Recorded convergence of the residual error for solving matrix inverse under different matrix dimensions; (b) the fitting curve for the number of iteration steps y and the matrix dimension x.

require higher iterations for solving underdetermined problems. Moreover, the conductance write noise has been shown to influence the accuracy of the underdetermined problems. Thus, according to the previous analysis, we suggest using higher precision WSC to reduce the influence of conductance write noise. It has to be mentioned that, although GMMS is capable to process the real-world underdetermined workloads such as the combinational optimization [22]. For some underdetermined matrix equation workloads, such as compress sensing, it is required to optimize the $l1$-norm of the solution matrix to find the sparse solution of the equation [23]. The usage of GMMS was limited in these applications, as GMMS optimized the Frobenius-norm during the solution process.

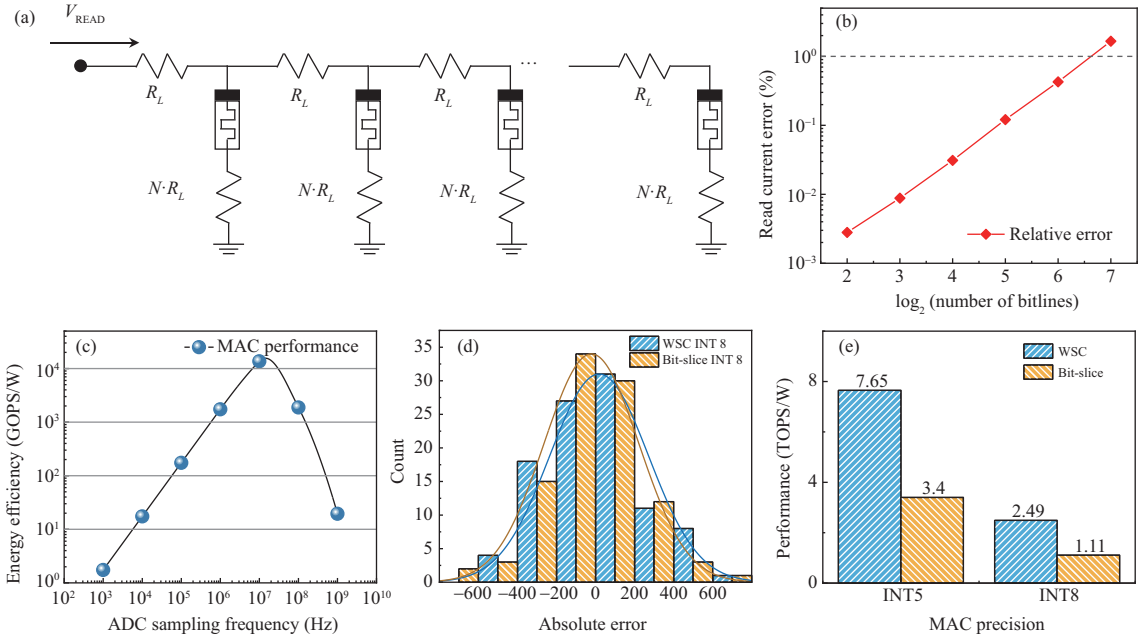

Figure 8 (Color online) Performance benchmarking. (a) Illustration of the SPICE simulation circuit utilized to simulate the read current errors. The line resistance is set to 0.2 Ω (considered a prospected implementation in 65 nm CMOS technology and a device pitch of 600 nm) and N is the crossbar size [28]; (b) the relationship between the read current error with the crossbar size; (c) the performance of the single memristor array under different ADC sampling frequencies; (d) robustness comparison of the WSC method and 2-based bit-slice method, the conductance variation is 10%; (e) performance comparison.

5 Performance benchmark and discussion

Here, the overall performance of GMMS is further estimated. Figure 7(a) shows the convergence of the residual error when solving the matrix inverse with different dimensions. The testing matrices are obtained from a one-dimensional heat equation (see Appendix B) [24]. Figure 6(b) presents the fitting curve for the number of iteration steps y and the matrix dimension x which obeys $y \approx 4.95 \ln x + 6.90$. Time complexity to execute the matrix-multiplication is approximate $O(n)$ in the crossbar array [25]. Thus, the time-complexity for solving matrix inverse by GMMS is approximate to $O(n \ln n)$. For comparison, the time-complexity for solving matrix inverse in a digital computer is approximate to $O(n^3)$. This result indicates the great acceleration of GMMS for solving matrix equations. Notably, the iteration cost corresponds to the ε, lr, WSC precision, and the matrix condition number. Thus, for situations requiring a large number of iterations, device endurance might be a critical issue for the practical use of GMMS. Further simulation for the solution updating during the solving process indicates that introducing the mixed-precision strategy can strongly reduce the frequency of the reprogrammed operation for devices, which relieves the pressure of the solution operation on the endurance of the device (see Appendix A). Moreover, according to the mixed-precision techniques [26], we suggest the MAC precision of WSC is INT 8, which is capable to guaranteed both the solution accuracy and computation efficiency for typical equations. For the matrix equations that are hard to converge such as the ill-conditioned problems and the overdetermined problems, we suggest the solution should be performed under INT 16 MAC precision. Besides that, to take the advantages of parallel computing, we suggest using GMMS for the equations $AX = B$ while B are matrices.

Furthermore, the performance of the WSC MAC unit was estimated. The supposed memristors model is from [27]. We use the SPICE simulation to estimate the impact of the line resistance to the read-out current error. Figure 8(a) shows the SPICE circuit utilized to simulate the read current errors. The simulation result indicates that the relative error of the read current will be higher than 1%, while the crossbar size larger than 64×64 (Figure 8(b)). Thus, we supposed the crossbar size is 64×64. Therefore, the power consumption of the memristive devices is 0.052 mW in an operations cycle (considering the influence of the line resistance and the line parasitic capacitance [28]). 64 of 22 nm DACs are needed to activate an array and the power consumption of DACs is 0.76 pJ [8]. The utilized transimpedance amplifier (TIA) has $f_{\mathrm{GBW}} = 28.6$ MHz and consumes 36 μW [28]. To make a single crossbar array

unit achieve the best performance, the sampling frequency of the ADCs is chosen to be 10 MHz, where the overall energy efficiency of a single MAC core is 13.6 TOPS/W (Figure 8(c)). The performance of WSC to calculate the MAC with INT 5 and INT 8 precision is evaluated to be 7.65 and 2.43 TOPS/W. For comparison, the energy efficiency of the most advanced CMOS processor (Google TPU) to perform 8-bit computation is 2.3 TOPS/W [29]. Moreover, the evaluation points out the 2.25× performance improvement of WSC over bit-slice in-memory MAC method [8] to execute the same precision MAC, while the systems robustness to conductance write noise is similar (Figure 8(d) and (e)). Notably, we used a stand-alone ADC at each column of the memristor array in our estimation, using the shared ADCs or array ADCs [30, 31] can further improve the performance of the memristor-based system and reduce the area consumption.

6 Conclusion

In summary, a GMMS using analog memristors is demonstrated in simulation to provide an efficient solution for the typical matrix-equation-based workloads. The mixed-precision method and a robustness-motivated fixed-point computation method are introduced to overcome the precision-limitation and the computation variability of the analog MAC unit. Our results elucidate that the solver can provide an accurate solution with high versatility, high robustness, low time-complexity, and high matrix-multiplication efficiency for solving matrix equations. We believe this method is promising for memristor-based analog computing to accelerate real-world matrix equation-intensive tasks.

Acknowledgements This work was partly supported by National Key R&D Program of China (Grant No. 2019YFB2205100), National Natural Science Foundation of China (Grant Nos. 61874164, 61841404), Hubei Key Laboratory of Advanced Memories, and Hubei Engineering Research Center on Microelectronics.

Supporting information Appendixes A and B. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Ran A C M, Reurings M C B. A fixed point theorem in partially ordered sets and some applications to matrix equations. Proc Amer Math Soc, 2004, 132: 1435–1443
2 Tan L, Kothapalli S, Chen L X, et al. A survey of power and energy efficient techniques for high performance numerical linear algebra operations. Parall Comput, 2014, 40: 559–573
3 Golub G H, van Loan C F. Matrix Computations. 4th ed. Baltimore: Johns Hopkins University Press, 2013
4 Sebastian A, Le Gallo M, Khaddam-Aljameh R, et al. Memory devices and applications for in-memory computing. Nat Nanotechnol, 2020, 15: 529–544
5 Chen J, Li J C, Li Y, et al. Multiply accumulate operations in memristor crossbar arrays for analog computing. J Semicond, 2021, 42: 013104
6 Hu M, Graves C E, Li C, et al. Memristor-based analog computation and neural network classification with a dot product engine. Adv Mater, 2018, 30: 1705914
7 Le Gallo M, Sebastian A, Mathis R, et al. Mixed-precision in-memory computing. Nat Electron, 2018, 1: 246–253
8 Zidan M A, Jeong Y J, Lee J, et al. A general memristor-based partial differential equation solver. Nat Electron, 2018, 1: 411–420
9 Sun Z, Pedretti G, Ambrosi E, et al. Solving matrix equations in one step with cross-point resistive arrays. Proc Natl Acad Sci USA, 2019, 116: 4123–4128
10 Li J C, Zhou H J, Li Y, et al. Self-adaptive matrix equation solving in analog memory array 1. In: Proceedings of the 5th IEEE Electron Devices Technology & Manufacturing Conference (EDTM), 2021
11 Micikevicius P, Narang S, Alben J, et al. Mixed precision training. 2017. ArXiv:1710.03740
12 Li C, Belkin D, Li Y N, et al. Efficient and self-adaptive in-situ learning in multilayer memristor neural networks. Nat Commun, 2018, 9: 2385
13 Sun Z, Pedretti G, Mannocci P, et al. Time complexity of in-memory solution of linear systems. IEEE Trans Electron Dev, 2020, 67: 2945–2951
14 Cai F X, Correll J M, Lee S H, et al. A fully integrated reprogrammable memristor-CMOS system for efficient multiply-accumulate operations. Nat Electron, 2019, 2: 290–299
15 Cheng L, Li J C, Zheng H X, et al. In-memory hamming weight calculation in a 1T1R memristive array. Adv Electron Mater, 2020, 6: 2000457
16 Sheridan P M, Cai F X, Du C, et al. Sparse coding with memristor networks. Nat Nanotech, 2017, 12: 784–789
17 Higham N J. Accuracy and Stability of Numerical Algorithms. Philadelphia: Society for Industrial and Applied Mathematics, 2002
18 Benzi M, Meyer C D, Tuma M. A sparse approximate inverse preconditioner for the conjugate gradient method. SIAM J Sci Comput, 1996, 17: 1135–1149
19 Sheng X P, Su Y F, Chen G L. An iterative method for the minimum norm-least square solution of the matrix equation $A^TXB + B^TX^TA = D$ (in Chinese). Numer Math J Chinese Univ, 2008, 4: 301–307
20 Weisberg S. Applied Linear Regression. Hoboken: John Wiley & Sons, 2005
21 Sun Z, Pedretti G, Bricalli A, et al. One-step regression and classification with cross-point resistive memory arrays. Sci Adv, 2020, 6: 2378

22 Meng X R, Saunders M A, Mahoney M W. LSRN: a parallel iterative solver for strongly over-or underdetermined systems. SIAM J Sci Comput, 2014, 36: 95–118
23 Mamaghanian H, Khaled N, Atienza D, et al. Compressed sensing for real-time energy-efficient ECG compression on wireless body sensor nodes. IEEE Trans Biomed Eng, 2011, 58: 2456–2466
24 Cannon J R. The One-dimensional Heat Equation. Cambridge: Cambridge University Press, 1984
25 Sun Z, Huang R. Time complexity of in-memory matrix-vector multiplication. IEEE Trans Circ Syst II, 2021, 68: 2785–2789
26 Wang K, Liu Z J, Lin Y J, et al. HAQ: hardware-aware automated quantization with mixed precision. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019. 8612–8620
27 Huang X D, Li Y, Li H Y, et al. Forming-free, fast, uniform, and high endurance resistive switching from cryogenic to high temperatures in W/AlO_x/Al_2O_3/Pt bilayer memristor. IEEE Electron Dev Lett, 2020, 41: 549–552
28 Berdan R, Marukame T, Ota K, et al. Low-power linear computation using nonlinear ferroelectric tunnel junction memristors. Nat Electron, 2020, 3: 259–266
29 Jouppi N P, Young C, Patil N, et al. In-datacenter performance analysis of a tensor processing unit. In: Proceedings of the 44th Annual International Symposium on Computer Architecture, 2017
30 Xu R Y, Liu B, Yuan J. Digitally calibrated 768-kS/s 10-b minimum-size SAR ADC array with dithering. IEEE J Solid-State Circ, 2012, 47: 2129–2140
31 Li T L, Sakai S, Kawada S, et al. A column-parallel hybrid analog-to-digital converter using successive-approximation-register and single-slope architectures with error correction for complementary metal oxide silicon image sensors. Jpn J Appl Phys, 2013, 52: 1–7