

# Chapter 6

## One Step in-Memory Solution of Inverse Algebraic Problems



Giacomo Pedretti

### 6.1 Introduction

Linear algebra problems, such as solving a linear system of equations, are the backbone of modern scientific computing and data-intensive tasks. Among these, machine learning is currently the discipline with most effort of study from scientists and engineers, to unleash the full power of computational algorithms with applications to any aspect of our life. These powerful algorithms, such as linear and logistic regression, are usually executed in conventional digital hardware by combining sequences of boolean functions on binary data. Thus, computing complicated operations requires a large memory and many computing steps. These problems are encoded in matrix form and executed by iteratively performing matrix-vector multiplications [7, 36], resulting in a polynomial computational time complexity, for example  $O(N^3)$  where  $N$  is the size of the problem. In this chapter, novel analog circuits for the solution of matrix equations in one step will be presented. Thanks to the in-memory computing framework, the problem implementation does not require data transfer between memory and processing unit, resulting in unprecedented speed. With nanoscale crosspoint resistive memories, the novel circuit requires also less area compared to traditional technology. The results pave the way for the development of memory computing unit to overcome the limitation of current accelerators.

---

G. Pedretti (✉)

Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, piazza Leonardo da Vinci, 32, Milano, Italy  
e-mail: [giacomo.pedretti@polimi.it](mailto:giacomo.pedretti@polimi.it)

© The Author(s) 2021

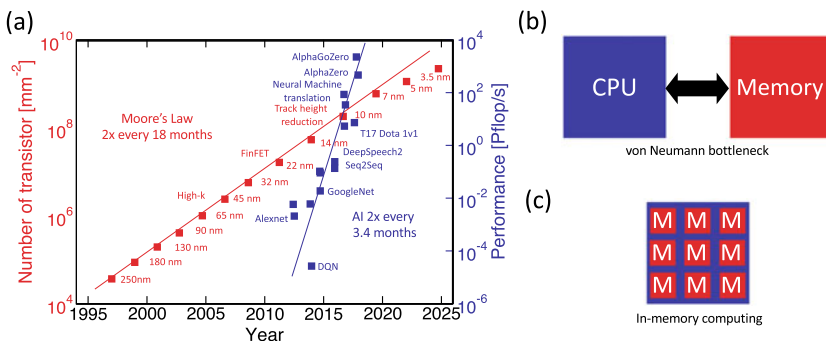
A. Geraci (ed.), *Special Topics in Information Technology*,

PoliMI SpringerBriefs, [https://doi.org/10.1007/978-3-030-62476-7\\_6](https://doi.org/10.1007/978-3-030-62476-7_6)

## 6.2 In Memory Computing

Technology scaling has been driven by Moore’s law [21] in the last decades, predicting that the number of transistors per  $mm^2$  of an integrated circuit doubles every 18 months. Figure 6.1a depicts in red the exponential growth fitted from real data of different technology node of the last 25 years and the confirmed prediction from the future releases [28]. It is already possible to see a deviation from the ideal exponential scaling, as the some technologies will see the market with significant delay. Moore’s law is in fact slowing down, due to physical limits of devices scaling and increased cost manufacturing [28]. It has also been observed [13] that the energy dissipated by transistors has decreased exponentially with the technology node until the late 80’s. However, by now this energy should have reached the thermal fluctuation  $kT$ , known as Landauer limit [13], which is impossible with modern Complementary-Metal-Oxide-Semiconductor (CMOS) technologies. Even future predictions are far away from the Landauer limit. It is thus evident that new computing technologies need to be developed to keep the pace with Moore’s Law and reduce the energy dissipation. Among this, novel memory devices have attracted research interest also from the computing community [39], in fact they have been demonstrated able of performing traditional computing tasks such as boolean function [31].

However Moore’s law speed is not enough. Figure 6.1a shows a comparison of Moore’s law (in red) with the performance required for executing state of the art algorithms (in blue) developed across the last years [3]. It is possible to see that with an exponential growth of the number of floating point operations per second (FLOP/s) that doubles every 3.4 months, the required resources scaling outperforms Moore’s Law. This suggest that not only new materials and devices are needed to fulfill Moore’s law requirements, but a shift of paradigm in architecture is needed to outperform traditional computing systems. Figure 6.1b shows the conventional



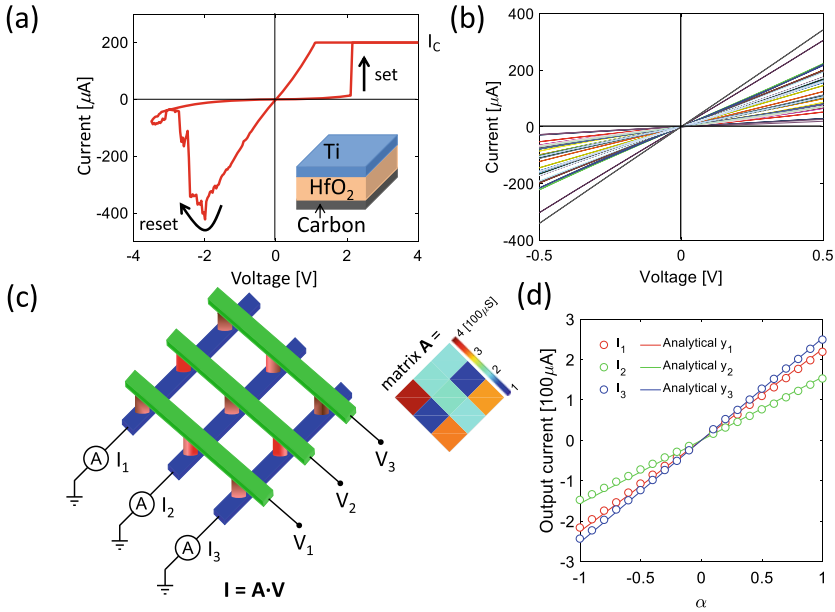
**Fig. 6.1** **a** Comparison between the exponential growth of Moore’s Law (red) and the required performance for executing modern artificial intelligence (AI) algorithms (blue), **b** conventional von-Neumann architecture suffering from a bottleneck when transferring data between memory and processing unit and **c** in-memory computing concept

von-Neumann architecture [22], where the processing unit (blue) is responsible only for executing operations whereas the memory unit (red) is responsible for storing them. Most of nowadays computers are based on this architecture, where one or multiple types of memory store the data with the central processing units (CPU) or graphic processing units (GPU) performing computation. When a lot of data need to be analyzed this architecture exposes a bottleneck in computation, known as von-Neumann bottleneck [19], due to the time and energy spent for handling data travelling from memory to processor and back. A new computing architecture that avoids the bottleneck is then desired.

In-memory computing with novel resistive memories [10, 11] has been proposed as a solution to overcome the limitation of both Moore's law speed and von Neumann bottleneck. The idea is to harness intrinsic materials properties of such memories to create new computing paradigms based on physical laws and known as physical computing [11, 44]. By organizing memories in crosspoint arrays it is possible then to have a compact accelerator known as memory processing unit (MPU) [44], which does not require data transfer and can perform computations within the memory. Figure 6.1c shows a conceptual representation of a MPU architecture, with many memory cores interconnected with each other. The novel computing unit have been shown to have unprecedented speed up compared with traditional and specific circuit for acceleration [25].

### 6.3 In-Memory Matrix-Vector-Multiplication Accelerator

Emerging memory devices, commonly referred as memristors, have recently attracted interest for their application both as memory and computing elements. Among these, resistive random access memories (RRAM) are a promising candidate for computing, due to their low energy operation, high endurance, small area and cost-effective fabrication [9]. Figure 6.2a shows a typical current-voltage characteristic of a RRAM device which is depicted in the inset and made of a Ti top electrode (TE) deposited on a  $\text{HfO}_x$  layer and a C bottom electrode (BE) [2]. After a forming process it is possible to change and modulate the conductance of the device. A positive pulse applied from the TE to the BE will result in a filament growth from TE to BE, or set transition, bringing the device into a low resistance state (LRS). By fixing the maximum current flowing to the RRAM during the set transition, namely compliance current ( $I_C$ ), it is possible to avoid hard breaks of the device oxide and modulate the LRS conductance.  $I_C$  can be fixed by an external circuit, such a Source Measurement Unit (SMU), or with a transistor connected with the drain at the BE, that can also be used as selector device in an array configuration. By applying a negative pulse the RRAM undergoes a reset, resulting in the filament rupture and a gap formation in the conductive path, thus an high resistance state (HRS). The gap width can be controlled by the maximum applied negative voltage during reset and can be used as well to modulate the conductance. Figure 6.2b shows different measured conductance states demonstrating the possibility of analog programming of the RRAM device. Given



**Fig. 6.2** **a** Resistive Random Access Memory (RRAM) I-V characteristics. By applying a positive voltage it is possible to set the memory device into a low resistance state (LRS) whose conductance is controlled by the maximum compliance current  $I_C$  flowing during the set operation, while by applying a negative voltage it is possible to reset the device into a high resistance state (HRS) whose depth is controlled by the maximum applied negative voltage. Inset shows the fabricated Ti-HfO<sub>x</sub>-C RRAM device. **b** Different conductance achieved by modulating  $I_C$  during the set operation. **c** Crosspoint memory architecture for multiply-accumulate operation. RRAM devices are organized in an array representing an analog matrix  $A$ , by applying a voltage vector  $V$  on the columns, the current vector  $I$  at the rows is the matrix-vector multiplication result  $I = GV$ . Inset represent a measured programmed matrix  $A$ . **d** Measured (circles) and calculated (lines) currents  $I$  as function of the parameter  $\alpha$  controlling the applied voltage vector  $V = \alpha[0.2, 0.3, 0.4]$  with  $-1 \leq \alpha \leq 1$ . Adapted from [33]

the possibility of representing in principle any given number, applications in analog operation acceleration with RRAM devices have rapidly arisen. Different architectures have been presented to accelerate analog problems such as crosspoint arrays [40] and content addressable memories [17]. Figure 6.2c shows a crosspoint array implementation where memristive devices are arranged in a matrix form to directly write an algebraic matrix of real positive numbers  $G$  into the RRAM conductance. By applying an input voltage vector  $V$  on the crosspoint columns, the current flowing through the crosspoint rows is  $I = GV$  or the dot product of matrix  $G$  by vector  $V$ . In this way, it is possible to accelerate dot product, also referred as matrix-vector-multiplication (MVM), in one step [10, 11]. Memristive crosspoint has been shown able to accelerate different problems based on MVM, such as the training [16, 27, 38] and inference [20, 41] of neural networks, image processing [18], sparse coding [29], optimization problems [6, 24] and the solution of linear equations through

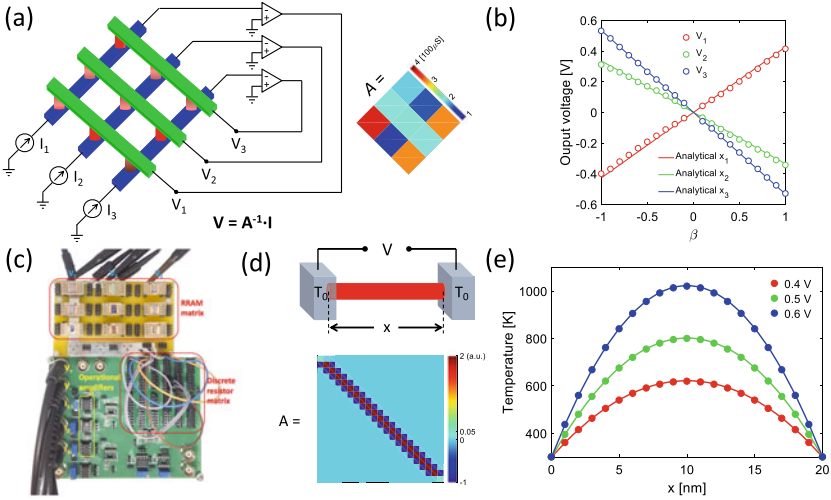
iterative numerical approaches [14, 43]. Integrated circuits comprising memristive arrays and the circuitry need to generate the input, such as digital-analog-converters (DAC), sense and read the outputs, such as transimpedance amplifiers (TIA) and analog-digital-converters (ADC), and cell selecting and routing, able to accelerate MVM have been proposed [5, 37, 42], outperforming modern processor both in throughput and energy saving [42].

## 6.4 One Step in-Memory Solution of Inverse Algebraic Problems

Crosspoint arrays offer the analog capability of writing arbitrary positive real matrixes coefficients, however iterative operations are usually performed in conventional digital hardware [6, 43]. To harness the full potential of the analog approach, iterations can be performed in the analog domain through feedback connected operational amplifiers [23, 33, 34]. By properly programming the conductance matrix and connecting the feedback amplifiers, one can solve different inverse problems such as linear systems [33], eigenvectors calculation and pageranking[32], linear and logistic regressions [34].

### 6.4.1 In-Memory Solution of Linear Systems in One-Step

Operational amplifiers in negative feedback configuration offer analog implementation of loops. Solving a system of linear equation is the equivalent matrix operation of performing a division between two scalars. This is the role of a TIA, an operational amplifier with a feedback resistance  $R$  connected between the negative input and the output. Grounding the positive input and injecting a current  $I$  on the negative input, the output voltage will adjust on  $V = IR$  or  $V = I/G$  with  $G = 1/R$  conductance of the resistance  $R$ . This is due to the negative feedback effect and the nature of the operational amplifier that has a very large input impedance. By considering a matrix version of this circuit, it is then possible to calculate the solution of a linear system encoded in a matrix of conductance  $G$ , which is connected in feedback with operational amplifiers [33]. Figure 6.3a shows the circuit schematic for a 3 equations linear system. The system coefficients are encoded in the conductance matrix  $A$  (Fig. 6.3a inset) measured on 9 HfO<sub>x</sub> RRAM devices arranged in crosspoint configuration. The crosspoint rows are connected to the negative input of the operational amplifiers, the columns to the output of the operational amplifiers while the positive input of the operational amplifiers is kept connected to ground. By injecting a current  $I$  on the rows representing the known vector of the linear system the output voltage vector will be the solution of the linear system  $V = A^{-1}I$ , which is computed in one step without digital iteration [33]. Figure 6.3b demonstrate the concept showing



**Fig. 6.3** **a** Circuit for solving a linear system in one step comprising a cross-point array of RRAM devices (red cylinders) programmed with the conductance matrix  $A$  (inset), connected with the rows (blue bars) at the negative input of operational amplifiers. By injecting a current  $I$  through the rows, the columns (green lines) connected to the operational amplifier outputs will stabilize to a voltage vector  $V = A^{-1}I$  which is the solution of a linear system. **b** Measurements of output voltages (circles) and analytical (lines) solution of the linear systems  $AV = I$  with  $I = \beta[0.2; 1; 1]$  and  $-1 \leq \beta \leq 1$  as function of the controlling parameter  $\beta$ . The comparison of the measured voltages with the analytical solution support the accuracy of the system. **c** Hardware implementation of the circuit for solving linear systems with  $\text{HfO}_x$  devices arranged on a printed circuit board with commercial operational amplifiers controlled with an external arbitrary waveform generator. **d** 1-dimensional Fourier equation encoded with 21 points in a  $21 \times 21$  conductance matrix. **e** Output voltages  $V$  (circle) simulated with SPICE for a larger circuit showing the solution of the 1-D Fourier heat equation confirming a good agreement with the analytical result (lines). Adapted from [33]

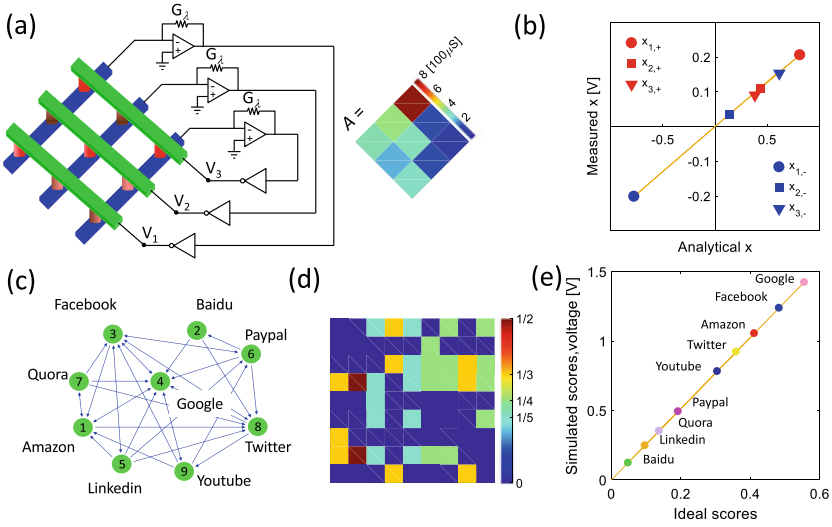
the measured voltage  $V$  and the analytical solution of the linear system  $AV = I$  with  $I = \beta[0.2; 1; 1]$  as function of the controlling parameter  $\beta$ , indicating a good agreement between electrical measurements and analytical result. The measurements were performed on a printed circuit board (PCB) with  $\text{Ti}/\text{HfO}_x/\text{C}$  RRAM devices [2] arranged on a crosspoint configuration and connected in feedback with commercial operational amplifiers (Fig. 6.3c). The known vector is given as voltage with an arbitrary waveform generator and then converted to current with input resistance connected to the negative input of the operational amplifiers. The output voltage is monitored with an external oscilloscope. To represent both positive and negative coefficients of the linear system, it is possible to use two separated crosspoint that represent the matrixes  $B$  and  $C$ , with  $A = B - C$ . By connecting matrix  $B$  to the circuit of Fig. 6.3a, the output voltage to the matrix  $C$  through negative buffers and feeding both matrix  $B$  and matrix  $C$  with the same input current representing the known vector, one can solve an arbitrary linear system  $(B - C)V = AV = I$  where  $A$  has both positive, negative and zero coefficients [33]. As an example, this circuit

can be used to solve differential equations such as the Fourier heat equation [33]. Figure 6.3d shows a 1-D Fourier heat equation encoded in its  $21 \times 21$  discretized matrix form, that can directly be mapped in a crosspoint array and be solved in one step. Figure 6.3e shows the output voltage vector  $V$  simulated with in SPICE representing the solution of the problem in Fig. 6.3d for different starting temperature compared with the analytical results and as function of the distance.

The results shows a good match supporting the use of the circuit for solving large systems of equations. In fact, interestingly the solution time does not depends on the matrix, thus linear system, dimension [35]. One can think about the operational amplifier in negative feedback configuration, where the bandwidth is limited by the loop gain and equal to  $f_{max} = GBWP \cdot \frac{R_{in}}{R_{in} + R_f}$  where  $GBWP$  is the gain bandwidth product of the operational amplifiers,  $R_{in}$  the input resistance and  $R_f$  the feedback resistance. By considering a feedback matrix, it is possible to demonstrate that the settling time, thus the bandwidth, is solely limited by the minimal eigenvalue of the matrix [35] and not by its size making the time complexity  $O(1)$ . This is an unprecedented speedup compared with conventional conjugate gradient solvers [30], where time complexity is  $O(N)$  at its best and quantum computing [8], where the best time complexity is  $O(\log(N))$  where  $N$  is the size (i.e. the number of equations) in the linear system. The result supports the use of the circuit for solving systems of linear equations in one step, outperforming digital and quantum computers.

## 6.4.2 In-Memory Eigenvectors Calculation in One-Step

Many scientific and machine learning problems, such as the solution of differential equations, require not the simple solution of a linear system, but the eigenvector computation. Mathematically speaking, this means to solve the equation  $Ax = \lambda x$ , which can be arranged such as  $(A - \lambda I)x = 0$ . It is possible to observe that by encoding on a crosspoint the matrix  $A$  and on a second crosspoint the diagonal matrix  $\lambda I$ , with the mixed matrix configuration it is possible to compute the eigenvectors with the feedback circuit of Sect. 6.4.1 [32, 33]. Figure 6.4a shows a compact circuit schematic for calculating the eigenvector solution where the diagonal matrix is represented with feedback conductance  $G_\lambda$ . To guarantee the stability of the circuit, only the eigenvectors corresponding to highest positive and lower negative eigenvalue can be computed. In fact the circuit works at the boundary of stability with a loop gain  $G_{Loop} = 1$ . Without any input current the opamp corresponding to the maximum value of the eigenvector saturates while the others adjust resulting in an output voltage vector  $V$ , which normalized by the supply voltage, it is equal to the normalized eigenvector  $x$  corresponding to the non-trivial solution of  $(A - \lambda I)x = 0$ . Figure 6.4a-inset shows a programmed conductance matrix  $A$  and Fig. 6.4b the measured eigenvectors calculation corresponding to the highest positive (red) and lowest negative (blue) eigenvalue as function of the analytical solution, showing a good agreement. It has to be noted that to compute the eigenvector corresponding to the negative eigenvalue



**Fig. 6.4** **a** Circuit for solving the eigenvector equation  $Ax = \lambda x$ , where  $x$  is the eigenvector and  $\lambda$  the maximum positive eigenvalue. With an input current  $I = 0$ , the operational amplifier corresponding to the maximum value of the eigenvector  $x$  saturates. By normalizing the output voltages, the eigenvector is found. Inset shows a  $3 \times 3$  matrix encoded in RRAM conductance. **b** Experimental solution of the eigenvector corresponding to the highest positive (red) and lowest negative (blue) eigenvalue, as function of the analytical solution. The eigenvalues are encoded in the feedback conductance  $G_\lambda$ . **c** Illustration of Pagerank algorithm, web pages are represented by green circles and the corresponding citation with blue arrows. **d** Stochastic link matrix corresponding to the problem in (c), which is calculated by normalizing the boolean link matrix by the sum over each column. **e** Simulation (circles) results of Pagerank problem in (c) as function of the ideal ranking. Adapted from [33]

the analog inverter of Fig. 6.4 should be removed with the output voltages of the operational amplifiers directly connected to the crosspoint array  $A$ . Unfortunately, in any case the highest eigenvalue  $\lambda_1$  must be known. To do that it is possible to apply iterative solution such as power iteration, or a sweep the conductance  $G_\lambda$  until one of the operational amplifier saturates. However, for some applications such as Pagerank the algorithm at the backbone of Google search engine [4], the maximum eigenvalue is always known a priori. Figure 6.4c shows an illustration of a web pages network with pages represented with green circles and citation represented by blue arrows. Goal of pageranking is to give a score to every webpage corresponding to its authority, namely how many citation receives from other pages with high authority. To do that it is possible to compute the eigenvector corresponding to the maximum eigenvalue of a stochastic matrix, namely the boolean link matrix between webpages normalized by the sum over each column [32, 33]. Interestingly, the maximum eigenvalue of such matrix is always known and  $\lambda_1 = 1$ , making the system highly feasible for giving such solution. Figure 6.4d shows the stochastic matrix corresponding to the network in Fig. 6.4c, whose SPICE simulated eigenvector solution is plotted in



Fig. 6.4e as function of the ideal solution showing good agreement. The circuit was also simulated with measured RRAM conductance tuned with a program and verify algorithm showing good agreement with the Harvard500 dataset results [32]. As the circuit in Sect. 6.4.1, the circuit for eigenvector computation shows a constant time complexity  $O(1)$  [32], making it aggressively interesting for machine learning and scientific applications compared with other computing technologies.

### 6.4.3 In-Memory Regression and Classification in One-Step

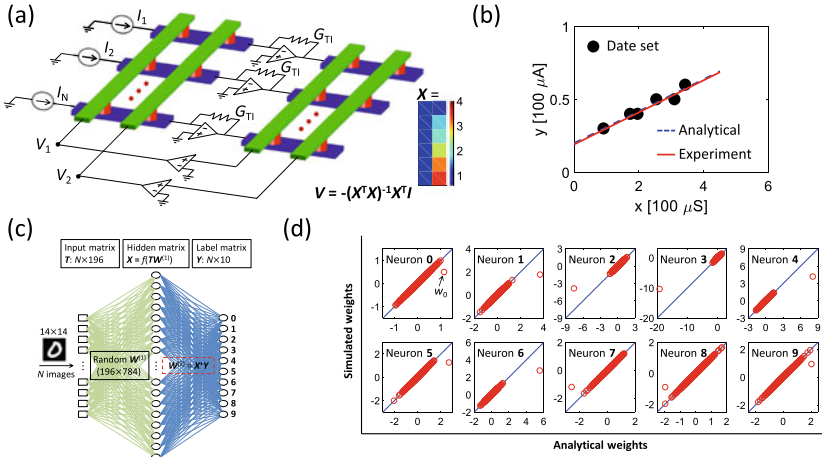
Many computing problems have more unknowns than equations or more equations than unknowns. The latter is the case of regression problem, which is a fundamental machine learning model for predicting a certain data behavior or classify its class. Linear and logistic regression are among the most used ML algorithms [1]. A linear regression problem can be described with the overdetermined linear system  $Xw = y$ , where  $X$  is a  $N \times M$  matrix with  $N > M$ ,  $y$  is known vector of size  $N \times 1$  and  $w$  is the unknown weight vector of size  $M \times 1$ . There is no exact solution to this problem, but the best solution can be calculated with the least squares error approach, that minimizes  $\|\epsilon\| = \|Xw - y\|_2$  which is the euclidean norm of the error. This can be done through the Moore-Penrose pseudoinverse [26] solving the equation

$$w = (X^T X)^{-1} X^T y. \quad (6.1)$$

To calculate  $w$  in one step, it is possible to cascade multiple analog stages representing all the parts of the equation. Figure 6.5a shows a schematic of the realized circuit for calculating linear regression weights in one step [34]. The conductance matrix  $X$  encodes the explanatory variables while the dependent variables are injected as current  $I$ . The output voltage of the rows amplifier will then adjust on  $V_{row} = (VX + I)/G_{TI}$ , thanks to the transimpedance configuration. Being the columns of the right crosspoint array connected to the input of the columns operational amplifier the current should be equal to zero, such as

$$\frac{(VX + I)}{G_{TI}} X^T = 0. \quad (6.2)$$

By rearranging equation (6.2), it is possible to observe that the weights of equation (6.1) are obtained in one step, without iterations as voltage  $V$  [34]. The inset of Fig. 6.5a shows a programmed conductance matrix on  $\text{HfO}_x$  arranged in a double array configuration and representing the linear regression problem of Fig. 6.5b, which shows a comparison of the experimental linear regression and the analytical one, evidencing a good agreement. Interestingly, with the same circuit is also possible to compute logistic regression in one step, thus classify data. By encoding in the conductance matrix the explanatory variables and injecting the class as input current, indeed it is possible to obtain the weights corresponding to a binary classification of



**Fig. 6.5** **a** Circuit for calculating regressions operation trough the Moore-Penrose pseudo inverse. Inset shows a programmed linear regression problem. **b** Experimental results and analytical calculation of the linear regression of 6 data points. **c** Neural network topology implemented for the weights optimization in one step. **d** Simulated weights as function of the analytical weights for the training of the neural network classification layer. Adapted from [34]

data. To illustrate such concept it is possible for example to train an output layer of a neural network in one step. Figure 6.5c shows a neural network topology, namely an extreme learning machine (ELM) used as example for neural network training. The network is made of 196 input neurons (corresponding to the pixels of an input image from the MNIST dataset reshaped on a  $14 \times 14$  size), 784 hidden neurons on a single hidden layer and 10 output neurons corresponding to the numbers from 0 to 9 of the MNIST dataset [15]. The first layer weights are randomized with a uniform distribution between 1 and  $-1$  and the output last layer is trained with logistic regression. By encoding in the conductance matrix the dataset evaluated on the hidden layer it is possible to use the circuit for calculating the weights of the second layer corresponding to a single output neuron in one step [34]. Figure 6.5d shows a comparison between the analytical weights and the simulated weights with a SPICE circuit simulation, showing little differences. The accuracy of the network trained with the circuit in recognizing the MNIST dataset is 92% which is equivalent to the ideal result for such network.

To evaluate the performance of the circuit it is possible to consider the number of computing steps required for training such neural network on a von Neumann architecture. With conventional computing approach, the complexity for calculating the logistic regression weights of equation (6.1) is composed by  $O(M^2N)$  to multiply  $X^T$  by  $X$ ,  $O(MN)$  to multiply  $X^T$  by  $y$  and  $O(M^3)$  to compute the LU factorization of  $XX^T$  and use it to calculate  $(XX^T)^{-1}$ . Thus,  $M^2N + MN + M^3$  floating points operations are required. In the case of the training of the neural network classification layer of Fig. 6.5c,  $2.335 \times 10^9$  operations are required. Given that

the simulated weight training with the in-memory closed loop crosspoint circuit required 145  $\mu\text{s}$  [34], the circuit has an equivalent throughput of 16.1 TOPS. The overall power consumption of the simulated circuit is calculated to be 355.6 mW [34] per training operation assuming a conductance unit of  $10\mu\text{S}$  in the circuit. As a result the efficiency of the circuit is calculated to be 45.3 TOPS/W. As an approximate comparison the energy efficiency of Google TPU is 2.3 TOPS/W [12] while the energy efficiency of an in-memory open loop circuit is 7.02 TOPS/W [29], evidencing that the in-memory closed loop solution is 19.7 and 6.5 times more efficient, respectively. The results show the appealing feasibility of the in-memory computing circuit for solving machine learning tasks, such as training a neural network with unprecedented throughput.

## 6.5 Conclusions

In this chapter in-memory circuit accelerators for inverse algebra problems have been presented. Compared to previous results, thanks to operational amplifiers connected in feedback configuration, it is possible to solve such problems in just one step. First the open loop crosspoint circuit for matrix vector multiplication is illustrated. Then, the novel crosspoint closed loop circuits are demonstrated able of solving linear systems and computing eigenvectors, in one step without iterations. Finally, the concept is extended to machine learning tasks such as linear regression and neural networks training in one step. These results supports in-memory computing as a future computing paradigm to obtain size independent time complexity solution of algebraic problems in a compact and low energy platform.

**Acknowledgements** The author would like to thank P. Mannocci for his critical reading of the manuscript.

## References

1. The State of Data Science and Machine Learning (2017). <https://www.kaggle.com/surveys/2017>
2. Ambrosi E, Bricalli A, Laudato M, Ielmini D (2019) Impact of oxide and electrode materials on the switching characteristics of oxide ReRAM devices. *Faraday Discuss* 213:87–98. <https://doi.org/10.1039/C8FD00106E>
3. Amodei D, Hernandez D. AI and compute. <https://openai.com/blog/ai-and-compute/>
4. Bryan K, Leise T (2006) The \$25,000,000,000 eigenvector: the linear algebra behind google. *SIAM Rev* 48(3):569–581. <https://doi.org/10.1137/050623280>
5. Cai F, Correll JM, Lee SH, Lim Y, Bothra V, Zhang Z, Flynn MP, Lu WD (2019) A fully integrated reprogrammable memristor-CMOS system for efficient multiply-accumulate operations. *Nat Electron* 2(7):290–299. <https://doi.org/10.1038/s41928-019-0270-x>
6. Cai F, Kumar S, Vaerenbergh TV, Liu R, Li C, Yu S, Xia Q, Yang JJ, Beausoleil R, Lu W, Strachan JP (2019) Harnessing intrinsic noise in memristor hopfield neural networks for combinatorial optimization. <https://arxiv.org/1903.11194>

7. Golub GH, Van Loan CF (2013) Matrix computations, 4th edn. Johns Hopkins studies in the mathematical sciences. The Johns Hopkins University Press, Baltimore. OCLC: ocn824733531
8. Harrow AW, Hassidim A, Lloyd S (2009) Quantum algorithm for linear systems of equations. *Phys Rev Lett* 103(15):150502. <https://doi.org/10.1103/PhysRevLett.103.150502>
9. Ielmini D (2016) Resistive switching memories based on metal oxides: mechanisms, reliability and scaling. *Semicond Sci Technol* 31(6):063002. <https://doi.org/10.1088/0268-1242/31/6/063002>
10. Ielmini D, Pedretti G (2020) Device and circuit architectures for in-memory computing. *Adv Intell Syst*, p 2000040. <https://doi.org/10.1002/aisy.202000040>
11. Ielmini D, Wong HSP (2018) In-memory computing with resistive switching devices. *Nat Electron* 1(6):333–343. <https://doi.org/10.1038/s41928-018-0092-2>
12. Jouppi NP, Borchers A, Boyle R, Cantin PL, Chao C, Clark C, Coriell J, Daley M, Dau M, Dean J, Gelb B, Young C, Ghaemmahami TV, Gottipati R, Gulland W, Hagmann R, Ho CR, Hogberg D, Hu J, Hundt R, Hurt D, Ibarz J, Patil N, Jaffey A, Jaworski A, Kaplan A, Khaitan H, Killebrew D, Koch A, Kumar N, Lacy S, Laudon J, Law J, Patterson D, Le D, Leary C, Liu Z, Lucke K, Lundin A, MacKean G, Maggiore A, Mahony M, Miller K, Nagarajan R, Agrawal G, Narayanaswami R, Ni R, Nix K, Norrie T, Omernick M, Penukonda N, Phelps A, Ross J, Ross M, Salek A, Bajwa R, Samadiani E, Severn C, Sizikov G, Snellman M, Souter J, Steinberg D, Swing A, Tan M, Thorson G, Tian B, Bates S, Toma H, Tuttle E, Vasudevan V, Walter R, Wang W, Wilcox E, Yoon DH, Bhatia S, Boden N (2017) In-datacenter performance analysis of a tensor processing unit. In: Proceedings of the 44th annual international symposium on computer architecture - ISCA '17, pp 1–12. ACM Press, Toronto, ON, Canada. <https://doi.org/10.1145/3079856.3080246>
13. Landauer R (1988) Dissipation and noise immunity in computation and communication. *Nature* 335(27):779–784
14. Le Gallo M, Sebastian A, Mathis R, Manica M, Giefers H, Tuma T, Bekas C, Curioni A, Eleftheriou E (2018) Mixed-precision in-memory computing. *Nat Electron* 1(4):246–253. <https://doi.org/10.1038/s41928-018-0054-8>
15. Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. In: Proceedings of the IEEE 86(11):2278–2324. <https://doi.org/10.1109/5.726791>
16. Li C, Belkin D, Li Y, Yan P, Hu M, Ge N, Jiang H, Montgomery E, Lin P, Wang Z, Song W, Strachan JP, Barnell M, Wu Q, Williams RS, Yang JJ, Xia Q (2018) Efficient and self-adaptive in-situ learning in multilayer memristor neural networks. *Nat Commun* 9(1):2385. <https://doi.org/10.1038/s41467-018-04484-2>
17. Li C, Graves CE, Sheng X, Miller D, Foltin M, Pedretti G, Strachan JP (2020) Analog content-addressable memories with memristors. *Nat Commun* 11(1):1638. <https://doi.org/10.1038/s41467-020-15254-4>
18. Li C, Hu M, Li Y, Jiang H, Ge N, Montgomery E, Zhang J, Song W, Davila N, Graves CE, Li Z, Strachan JP, Lin P, Wang Z, Barnell M, Wu Q, Williams RS, Yang JJ, Xia Q (2018) Analogue signal and image processing with large memristor crossbars. *Nat Electron* 1(1):52–59. <https://doi.org/10.1038/s41928-017-0002-z>
19. Merolla PA, Arthur JV, Alvarez-Icaza R, Cassidy AS, Sawada J, Akopyan F, Jackson BL, Imam N, Guo C, Nakamura Y, Brezzo B, Vo I, Esser SK, Appuswamy R, Taba B, Amir A, Flickner MD, Risk WP, Manohar R, Modha DS (2014) A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* (6197):668–673. <https://doi.org/10.1126/science.1254642>
20. Milo V, Zambelli C, Olivo P, Perez E, Mahadevaiah MK, Ossorio OG, Wenger C, Ielmini D (2019) Multilevel HfO<sub>2</sub>-based RRAM devices for low-power neuromorphic networks. *APL Mater* 7(8):081120. <https://doi.org/10.1063/1.5108650>
21. Moore GE (2006) Cramming more components onto integrated circuits, Reprinted from *Electronics*, volume 38, number 8, April 19, 1965, pp.114 ff. *IEEE Solid-State Circuits Soc Newsl* 11(3): 33–35. <https://doi.org/10.1109/N-SSC.2006.4785860>
22. von Neumann J (1945) First draft of a report on the EDVAC. <https://doi.org/10.5555/1102046>

23. Pedretti G (2020) In-memory computing with memristive devices. Ph.D. thesis, Politecnico di Milano
24. Pedretti G, Mannocci P, Hashemkhani S, Milo V, Melnic O, Chicca E, Ielmini D (2020) A spiking recurrent neural network with phase change memory neurons and synapses for the accelerated solution of constraint satisfaction problems. *IEEE J Explor Solid-State Comput Devices Circuits*, pp 1–1. <https://doi.org/10.1109/JXCDC.2020.2992691>. <https://ieeexplore.ieee.org/document/9086758/>
25. Peng X, Kim M, Sun X, Yin S, Rakshit T, Hatcher RM, Kittl JA, Seo JS, Yu S (2019) Inference engine benchmarking across technological platforms from CMOS to RRAM. In: Proceedings of the international symposium on memory systems - MEMSYS '19, pp 471–479. ACM Press, Washington, District of Columbia. <https://doi.org/10.1145/3357526.3357566>
26. Penrose R (1955) A generalized inverse for matrices. *Math Proc Camb Philos Soc* 51(3):406–413. <https://doi.org/10.1017/S0305004100030401>
27. Prezioso M, Merrih-Bayat F, Hoskins BD, Adam GC, Likharev KK, Strukov DB (2015) Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* 521(7550):61–64. <https://doi.org/10.1038/nature14441>
28. Salahuddin S, Ni K, Datta S (2018) The era of hyper-scaling in electronics. *Nat Electron* 1(8):442–450. <https://doi.org/10.1038/s41928-018-0117-x>
29. Sheridan PM, Cai F, Du C, Ma W, Zhang Z, Lu WD (2017) Sparse coding with memristor networks. *Nat Nanotechnol* 12(8):784–789. <https://doi.org/10.1038/nnano.2017.83>
30. Shewchuk JR (1994) An introduction to the conjugate gradient method without the agonizing pain. Technical report CMU-CS-94-125, School of Computer Science, Carnegie Mellon University, Pittsburgh
31. Sun Z, Ambrosi E, Bricalli A, Ielmini D (2018) logic computing with stateful neural networks of resistive switches. *Adv Mater* 30(38):1802554. <https://doi.org/10.1002/adma.201802554>
32. Sun Z, Ambrosi E, Pedretti G, Bricalli A, Ielmini D (2020) In-memory pagerank accelerator with a cross-point array of resistive memories. *IEEE Trans Electron Devices* 67(4):1466–1470. <https://doi.org/10.1109/TED.2020.2966908>. <https://ieeexplore.ieee.org/document/8982173/>
33. Sun Z, Pedretti G, Ambrosi E, Bricalli A, Wang W, Ielmini D (2019) Solving matrix equations in one step with cross-point resistive arrays. *Proc Natl Acad Sci* 116(10):4123–4128. <https://doi.org/10.1073/pnas.1815682116>
34. Sun Z, Pedretti G, Bricalli A, Ielmini D (2020) One-step regression and classification with cross-point resistive memory arrays. *Sci Adv* 6(5):eaay2378. <https://doi.org/10.1126/sciadv.aay2378>
35. Sun Z, Pedretti G, Mannocci P, Ambrosi E, Bricalli A, Ielmini D (2020) Time complexity of in-memory solution of linear systems. *IEEE Trans Electron Devices*, pp 1–7. <https://doi.org/10.1109/TED.2020.2992435>. <https://ieeexplore.ieee.org/document/9095220/>
36. Tan L, Kothapalli S, Chen L, Hussaini O, Bissiri R, Chen Z (2014) A survey of power and energy efficient techniques for high performance numerical linear algebra operations. *Parallel Comput* 40(10):559–573. <https://doi.org/10.1016/j.parco.2014.09.001>. <https://linkinghub.elsevier.com/retrieve/pii/S0167819114001112>
37. Wan W, Kubendran R, Eryilmaz SB, Zhang W, Liao Y, Wu D, Deiss S, Gao B, Raina P, Joshi S, Wu H, Cauwenberghs G, Wong HSP (2020) 33.1 A 74 TMACS/W CMOS-RRAM neurosynaptic core with dynamically reconfigurable dataflow and in-situ transposable weights for probabilistic graphical models. In: 2020 IEEE international solid-state circuits conference - (ISSCC), pp 498–500. IEEE, San Francisco, CA, USA. <https://doi.org/10.1109/ISSCC19947.2020.9062979>
38. Wang Z, Li C, Lin P, Rao M, Nie Y, Song W, Qiu Q, Li Y, Yan P, Strachan JP, Ge N, McDonald N, Wu Q, Hu M, Wu H, Williams RS, Xia Q, Yang JJ (2019) In situ training of feed-forward and recurrent convolutional memristor networks. *Nat Mach Intell* 1(9):434–442. <https://doi.org/10.1038/s42256-019-0089-1>
39. Wang Z, Wu H, Burr GW, Hwang CS, Wang KL, Xia Q, Yang JJ (2020) Resistive switching materials for information processing. *Nat Rev Mater*. <https://doi.org/10.1038/s41578-019-0159-3>

40. Yang JJ, Strukov DB, Stewart DR (2013) Memristive devices for computing. *Nat Nanotechnol* 8(1):13–24. <https://doi.org/10.1038/nnano.2012.240>
41. Yao P, Wu H, Gao B, Eryilmaz SB, Huang X, Zhang W, Zhang Q, Deng N, Shi L, Wong HSP, Qian H (2017) Face classification using electronic synapses. *Nat Commun* 8(1):15199. <https://doi.org/10.1038/ncomms15199>
42. Yao P, Wu H, Gao B, Tang J, Zhang Q, Zhang W, Yang JJ, Qian H (2020) Fully hardware-implemented memristor convolutional neural network. *Nature* 577(7792):641–646 (2020). <https://doi.org/10.1038/s41586-020-1942-4>
43. Zidan MA, Jeong Y, Lee J, Chen B, Huang S, Kushner MJ, Lu WD (2018) A general memristor-based partial differential equation solver. *Nat Electron* 1(7):411–420. <https://doi.org/10.1038/s41928-018-0100-6>
44. Zidan MA, Strachan JP, Lu WD (2018) The future of electronics based on memristive systems. *Nat Electron* 1(1):22–29. <https://doi.org/10.1038/s41928-017-0006-8>

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

