Analogue In-Memory Computing with Resistive Switching Memories

Giacomo Pedretti and Daniele Ielmini

Abstract In the era of pervasive artificial intelligence (AI) and internet of things (IoT), achieving a high energy efficiency is at the top of priority for computing systems. In this scenario, in-memory computing is gaining momentum as a new methodology to overcome the von Neumann architecture and the related memory bottleneck. One of the most promising device for in-memory computation is the resistive switching memory (RRAM), also known as memristor, thanks to controllable conductance, good scaling and relatively low energy consumption. However, to achieve the promised benefits of in-memory computing with RRAM in terms of performance and power consumption, it is necessary to address a number of open challenges at the device, architecture and algorithm levels. This chapter presents the status of in-memory computing with RRAM, including the device concept and characteristics, the computing architectures and the applications. The perspective of analogue computing is analyzed with reference to both matrix vector multiplication (MVM) and inverse MVM to accelerate linear algebra problems that are generally executed with iteration schemes, highlighting the advantages in terms of performance, energy consumption and computational complexity.

1 Introduction

The computing industry has been always driven by an urge for an exponential growth. The microprocessor performance has increased substantially in the last 50 years thanks to aggressive scaling in the transistor channel size which led to a doubling of the number of transistors per square mm every 18 month, as predicted by the Moore's Law [1]. However, this scaling trend has been slowing down due to technological, physical and process related issues [2]. On the other hand, a similar exponential law is emerging in the recent years, namely the performance (measured in FLOPS, or Floating Point Operations per Second), required by artificial intelligence (AI) and

G. Pedretti · D. Ielmini (✉)
Dipartimento di Elettronica, Informazione e Bioingegneria, and IU.NET, Politecnico di Milano, Piazza L. da Vinci 32–20133, Milano, Italy
e-mail: daniele.ielmini@polimi.it

R. Micheloni and C. Zambelli (eds.), *Machine Learning and Non-volatile Memories*,
https://doi.org/10.1007/978-3-031-03841-9_4

scientific computing which have been observed to double every 3.4 months [3]. It is then clear that, on the one hand, new devices are needed for continuing with the Moore's Law trend, while from the other one an architectural design effort is desired to keep the pace of the required performance of modern AI algorithms.

From the architectural standpoint, the von Neumann architecture [4], which constitutes the mainstream architecture of most digital computers, suffers from the memory bottleneck. In fact, the memory and the central processing unit (CPU) are physically separated, thus most of the time is spent for data movement between memory and processing chips [5–7]. On the opposite, in-memory computing aims at executing all operations within the memory chip without any need for data movement [6, 7]. A promising memory technology for in-memory computing is the class of the resistive memory devices [8], often dubbed memristors [9, 10], featuring low energy operation, high speed, high density and compatibility with the complementary-metal-oxide-semiconductor (CMOS) technology [8, 11]. In-memory computing concepts based on resistive memories have been demonstrated with several memory technologies, including the resistive switching random access memory (RRAM), the phase change memory (PCM), the ferroelectric random access memory (FERAM) [12] and the magnetic random access memory (MRAM) [8, 13]. Different computing concept can be executed inside the memory such as logic computing [7, 14], neuromorphic computing [15–19], stochastic operations [20] and analog computing [21]. In particular, analogue computing allows to accelerate several computing operations thanks to physical computation, where multiplication and summation are executed by physical laws in the analogue domain. Also, the unique architecture of the crosspoint memory array allows to parallelize computing, thus enabling a reduction of computational complexity with respect to the conventional digital computing. At the same time, in-memory computing is prone to errors and inaccuracies due to noise and device variations, which should be carefully taken into account for a fair comparison with the floating-point precision in digital circuits.

This chapter aims at reviewing the recent advances of analog in-memory computing with RRAM devices. First, we present the device properties and characteristics, in particular discussing the device requirements for analogue memory. Then we present the main analogue computing architectures with RRAM, focusing on matrix-vector-multiplication (MVM) for neural networks and optimization algorithms. Finally, we present the inverse-matrix-vector-multiplication (IMVM) architecture and illustrate the main applications and their advantages and drawbacks in terms of energy efficiency, time complexity and precision.

2 Resistive Switching Memories

Various types of nanoelectronic devices have been proposed in the latest years to replace or complement the conventional CMOS memory technologies at various levels of the hierarchy. Most of these memories rely on the concept of changing

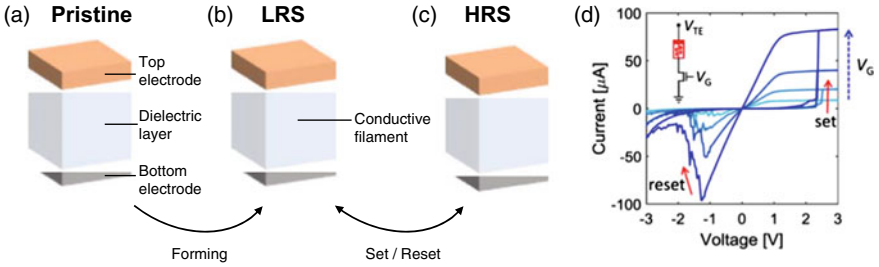

Fig. 1 RRAM devices. **a** RRAM are made by a dielectric material inserted between two metallic top electrode (TE) and bottom electrode (BE). In the pristine state they show a high resistance state (HRS). **b** By applying a positive forming voltage at the TE a filament grows from TE to BE resulting in low resistance state (LRS). **c** To retrieve the HRS it is possible to apply a reset voltage. **d** Typical I-V curve of a RRAM device in 1T1R configuration showing the ability of analog programming through different compliance current (or gate voltage V_G) and different reset voltages. Reprinted from [31] under Creative Commons License

the active material properties by the application of voltage or current programming pulses, thus storing the memory states as a specific material configuration. Several technologies for two terminal devices, such as resistive switching memory (RRAM) [11, 22], PCM [23, 24], FERAM [12, 25, 26] and MRAM [27], have been proposed. Among them, RRAM have attracted widespread research interest thanks to its low energy [28], high speed [29] and high density, combined with the ability of 3-dimensional integration [30]. Figure 1 shows the RRAM device structure, operation and switching characteristics. Typically, the device consists of a metal–insulator-metal (MIM) structure in its pristine state (Fig. 1a), which is a high resistance due the dielectric insulating layer. The device is generally initialized by the forming operation, consisting of the application of a relatively high voltage between the top electrode (TE) and bottom electrode (BE). During the forming process, the device undergoes a soft breakdown event with a local variation of the material composition, consisting of a low-resistivity filament which is responsible for the low resistance state (LRS, Fig. 1b). Then, it is possible to recover a high resistance state (HRS) by the application of a reset negative voltage between TE and BE which forms a depleted gap in the filament (Fig. 1c) that can be controlled by the maximum applied negative voltage. A set positive voltage pulse causes the device to switch back to the LRS with a continuous filament connecting the TE to the BE. The filament size is generally controlled by the maximum current flowing during the set operation, known as compliance current I_C and usually regulated by a series transistor. Figure 1d shows a typical current–voltage (I–V) curve of a RRAM device in a one-transistor-one-resistor (1T1R) structure with HfO_2 switching layer [31], highlighting the various states obtained by the modulation of the compliance current, which depends on the gate voltage V_G of the series transistor. Note that different resistive states are achieved at increasing I_C, thus demonstrating that RRAM can be used not only as a digital memory storing a '1' in the LRS and a '0' in the HRS, but also as a continuous analog memory with multiple states corresponding to different resistive values. The

first advantage is that multiple levels, hence multiple bits, can be stored in a single memory element, thus increasing the bit density in a memory array. Secondly, novel computing applications harnessing the analogue tuning of RRAM device can be unleashed to perform analogue in-memory computing [7, 21].

2.1 Memory Array Structures

RRAM devices can be arranged in various memory array structures for both as memory and computational unit, as shown in Fig. 2. The most straightforward configuration is passive crosspoint array where the RRAM device displays a simple one-resistor (1R) structure (Fig. 2a). In the crosspoint array each RRAM device is located at the intersection between a row line and a column line connecting the BE and TE of the device, respectively [32] By programming a conductance G_{ij} in the RRAM device connected between row i and column j of the crosspoint array and applying an input voltage vector $V = (V_1, V_2, \ldots, V_N)$ at the column terminals by keeping

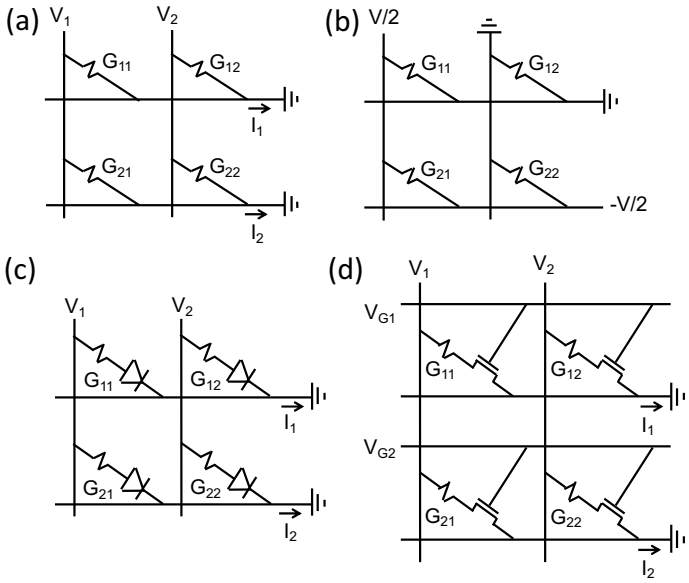

Fig. 2 Memory structures. **a** 1R crosspoint array where every device with conductance G_{ij} is connected between each row and column. **b** $V/2$ programming scheme in 1R crosspoint array, where only the selected cell (blue) receive the whole V voltage necessary for programming while the undesired selected cells (red) receive only $V/2$. **c** 1S1R crosspoint array, where a two terminal select device is inserted in series with every RRAM to mitigate the sneakpaths problem. **d** 1T1R array, where a transitory is used as select device and to regulate the compliance current during the set operation enabling analog programming. Reprinted with permission from [21] under Creative Commons License

the rows at ground, the resulting current is given by the matrix vector multiplication (MVM) formula:

$$I_i = \sum_{j=1}^{N} G_{ij} V_j \tag{1}$$

where N is the size of the crosspoint array. Note that the MVM operation naturally arises in the crosspoint array thanks to physical laws, namely the Ohm's law for the multiplication $I = GV$ and the Kirchhoff's law for the summation of individual currents incurring at the same row. Since MVM is a basic algebra operation, the crosspoint array structure is widely using in most analogue in-memory computing applications [7, 21, 33, 34]. The crosspoint array structure also takes advantage of a high memory density due to a memory cell area occupation of only $4F^2$, where F is the lithographical feature. Array organization in 3D arrays usually results in even smaller cell effective area, which is highly favorable for computing with large amounts of data [35]. However, the crosspoint array structure has the strong drawback of the difficult programmability and read disturb induced by sneak-path effect [36]. In fact, while selecting cell G_{ij} for set, reset or read operation, the cell row i and column j are biased, which results in unwanted current flows even if the unselected terminals are left floating, resulting in read disturb or possible set or reset operations at the unselected devices. Disturbs and sneak paths can be mitigated by suitable bias schemes, such as the $V/2$ biasing scheme in Fig. 2b [37, 38], where a voltage $V/2$ is applied to column j and $-V/2$ is applied to row i with all other rows/columns grounded. As a result, the voltage across all unselected cells is zero, except for the half-selected cells along row i and column j, where the voltage is reduced by a factor 2 thus minimizing the probability of undesired set or reset events. During the read operation, a voltage V_R is applied to the selected column while all the other columns and rows are connected to ground, which allows for reading all cells of the selected column in parallel [38].

However, due to the strong variation of set and reset voltages and to the limited set/reset resistance window, the passive crosspoint array with 1R structure can only be used with small array size, while becoming unpractical for the most typical array size for memory and computation.

2.1.1 1S1R Structure

To enable large crosspoint array size, a two terminal select device should be add, resulting in a one-selector/one-resistor structure (1S1R) [39–41] as shown in Fig. 2c. Selector devices should display a strong non-linear characteristic to prevent any current flowing in half selected devices with $V/2 < V_t$, where V_t is the selector device threshold voltage. Also, the select device should display large current at relatively large voltages to enable set and reset processes within the selected device.

The nonlinear characteristic should also be bidirectional, i.e., operable at both positive and negative voltages for set and reset processes, respectively. 1S1R crosspoint arrays are extremely promising for memory application, in particular for storage class memories to fill the performance/cost gap between DRAM and Flash memory. On the other hand, in-memory computing applications of 1S1R structures are yet to be unveiled, the main challenge being the contribution of the non-linear selector to the MVM operation.

2.1.2 1T1R Structure

The RRAM device can be connected in series with a transistor selector resulting in a one-transistor/one-resistor (1T1R) structure, as shown in Fig. 2d. To select a memory cell within an array for a set operation, the corresponding transistor gate line should be biased to turn on the transistor enough such that the applied TE voltage developed across the selected RRAM exceeds the set voltage. The transistor can be used for controlling I_C during the set operation, thus tuning the filament size hence the RRAM conductance, which makes the 1T1R structure ideal for analogue programming [42, 43]. During the reset or read operation, the selected gate line should be biased at a high voltage, such that all the TE voltage applied across the selected RRAM drops across the device, since the transistor resistance is negligible. Due to the excellent control of analogue state and lack of sneak paths, the 1T1R structure is by far the preferred configuration to demonstrate analogue-type in-memory computing functions [34, 44]. For the same reasons, we will restrict our focus on 1T1R structures in the following.

2.2 Requirements for Analogue Memory

The 1T1R structure allows to gradually control the conductance both during the set operation (via I_C) and the reset operation (via V_{stop}). In fact, the multilevel capability is a key requirement for analogue computing, making the memory able of representing multiple states in a single cell. In principle, if the available memory have only a few (or even two) possible resistance states, it is possible to memorize different slices of the analogue information in multiple devices [45], but the area occupation increases significantly. Another important requirement is the linearity of the I–V curve, so that by applying increasing input voltages, the cell current response increases linearly, thus satisfying Ohm's law for analogue multiplication. Note that, in most applications, this requirement can be circumvented by applying the input as a train of digital pulses with a simple unary or a more compact shift and add [45] encoding, and then reconstruct the analogue output by properly integrating the current in the time domain.

In general, partial HRS configurations obtained by the reset operation tend to show a non-linear characteristic, due the non-ohmic conduction in the depleted gap

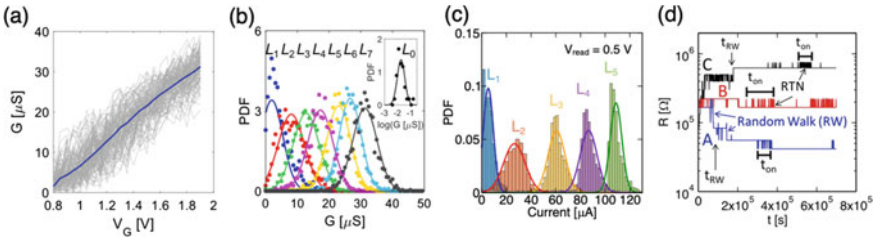

Fig. 3 Analog RRAM programming non-idealities. **a** Measured RRAM conductance as function of the gate voltage VG, showing an average linear dependence (blue), while single traces show large variations (grey). **b** Distribution of 7 levels of LRS obtained by modulating V_G and 1 level of HRS (inset) showing a conductance independent standard deviation. **c** Five analog levels programmed in a 4 kB RRAM array showing the both cell-to-cell and cycle-to-cycle variability. **d** Fluctuations of a programmed state on a RRAM device in HRS, showing different phenomena such as random walk and random telegraph noise. Reprinted with permission from [31, 43] under Creative Commons License. Reprinted with permission from [47]. Copyright 2015 IEEE

along the filament [11]. For this reason, it is most common to adopt I_C-controlled LRS configurations by set operation for preparing analogue states with variable conductance. Figure 3a shows the conductance as function of gate voltage in a 1T1R structure for 100 cycles and the median value [31]. At every cycle, the device was first prepared in the HRS, then a train of set pulses with fixed TE voltage $V_{\mathrm{TE}} = 3$ V above the threshold for set voltage and increasing gate voltage V_G was applied. As expected, the median value increases linearly with $V_G - V_T$, where $V_T = 0.7V$ is the transistor threshold voltage. However, one cannot solely rely on V_G (or equivalently I_C) for precisely controlling the device in a desired conductance, due to the cycle-to-cycle variations of the traces in Fig. 3a. These variations are generally attributed to the stochastic ionic migration during the set operation, which leads to variations in the shape and volume of the conductive filament [46, 47]. Figure 3b shows the resulting Gaussian distributions of conductance for 7 programmed LRS levels, indicating a standard deviation σ_G=3.8 μS. In addition to the cycle-to-cycle variability, a device-to-device variability arises as different devices in an array usually present different characteristics due to variation in the fabrication process causing specific geometry and material composition within the RRAM cell. Figure 3c shows distributions of currents for a read voltage $V_{\mathrm{read}} = 0.5V$ of 4 analogue levels programmed on a 4 kB RRAM array [43]. The observed variation includes contribution due both to cycle-to-cycle and device-to-device variability. To mitigate both cycle-to-cycle and device-to-device variability, it is possible to adopt program and verify algorithm. For instance, given a certain conductance G_{target} that should be approximately reached in the RRAM device, one can gradually increase the gate voltage as shown in Fig. 3b until the conductance G reaches a value between $G_{\mathrm{target}} - G_{\mathrm{tol}}$ and $G_{\mathrm{target}} + G_{\mathrm{tol}}$, where G_{tol} is the acceptable tolerance. If G exceeds $G_{\mathrm{target}} + G_{\mathrm{tol}}$, then a reset pulse can be applied to reduce G within the acceptable range. If G goes below $G_{\mathrm{target}} - G_{\mathrm{tol}}$ then another set pulse can be applied, until convergence into the [48, 49]. In principle, program-verify techniques allow to reach any desired conductance states within an

arbitrary tolerance range at the cost of increasing circuit complexity, programming energy and time. Also program-verify techniques tend to result in accelerated wear out of the memory device. In fact, after multiple set-reset operations the RRAM can be found in a non-ideal state, such as a stuck-off or stuck-on state [50]. One should carefully tune the maximum number of iterations during a program and verify routine to balance precision and device degradation.

After the RRAM device is programmed, the conductance state is also prone to time-dependent variations which may lead to conductance G to drift out of the tolerance range. In fact, RRAM suffers from resistance fluctuations over time, as shown in Fig. 3d for a HfO_x RRAM device [51]. The RRAM initially programmed at a given resistance might either increase or decrease its value, due to intermittent random telegraph noise (RTN) and random walk, which makes deterministic analog programming extremely challenging.

3 In-Memory Computing Architecture for Matrix-Vector Multiplication

The RRAM crosspoint array allows to execute the MVM operation simultaneously, in one step and in the analogue domain, thanks to physical Ohm's law multiplication and Kirchhoff's law summation of currents in (1) [33]. Figure 4a illustrates the basic MVM operation while Fig. 4b shows the correlation plot of the measured output currents as a function of the ideal MVM results obtained for a crosspoint array [48]. Motivated by the ubiquitous importance of MVM operations in data analytics and computing workloads, RRAM crosspoint for analogue MVM acceleration have been demonstrated for multiple applications [21, 32], such as neural networks acceleration [34, 52–55], image processing [44, 56], optimization algorithms [57–60], hardware

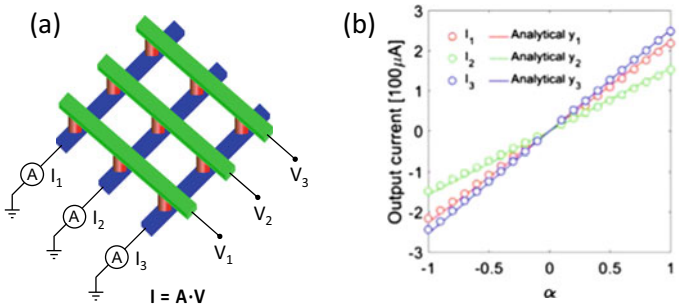

Fig. 4 MVM in crosspoint arrays. **a** A crosspoint array can be used for accelerating MVM. By programming a matrix A into the crosspoint conductance and applying a voltage vector V on the columns, the resulting current flowing into the rows tied to ground is $I = AV$. **b** MVM output current as function of α given an input voltage $V = \alpha V_0$. Reprinted with permission from [48], under Creative Commons License

security [38, 61, 62] and accelerated solution of differential equations [63]. Integrated circuit comprising CMOS mixed-signal circuits and RRAM arrays fabricated in the back end of the line have already been realized for several applications [64–67]. The most promising applications of MVM are probably neural network and optimization acceleration.

3.1 In-Memory Neural Network Accelerators

Analogue in-memory MVM has been widely used for feed-forward operation in neural network acceleration [21]. Figure 5a shows a conceptual illustration of a three-layer perceptron neural network [68]. Input data are applied on the left side, evaluated by the network layers from left to right, until reaching the output layer. At each layer in this forward transition, every neuron n_i emits a signal x_i which is multiplied by the synaptic weight w_{ij} before reaching the output neuron m_j. The evaluation of the output state y_j corresponding to neuron m_j consists of the summation of all the contributions from the previous layer, according to the formula $y_j = \sum_i x_i w_{ij}$, which is analogous to (1) where y_j is replaced by a physical output current, x_i is an input voltage and w_{ij} is the RRAM conductance. The forward operation of neural networks can thus be evaluated by an analogue MVM operation within a crosspoint array, with a throughput improvement up to 10^4 compared with multiply-accumulate (MAC) operations executed on a traditional digital computer [69]. Note that a RRAM device can only store positive weights whereas w_{ij} generally comprise both positive and negative values. Figure 5b–c show two possible techniques to enable mapping relative numbers as weights in a crosspoint neural network. In general, two RRAM devices can be used in parallel, each being biased with opposite polarity voltages to represent both negative and positive weights. For instance, a reference fixed conductance G_{ref}

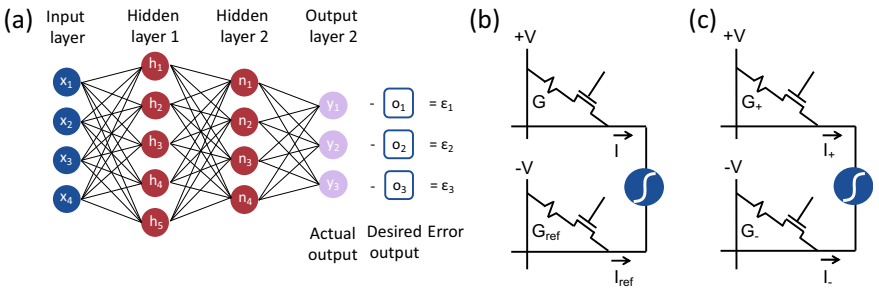

Fig. 5 Neural networks weights implementation. **a** Multilayer perceptron with an input layer, 2 hidden layers and an output layer. The output *y* is compared with the ground truth *o* and the calculated error is backpropagated for training the network. **b** representation of positive and negative weights with a single programmable RRAM and a fixed reference conductance. **c** a more flexible bipolar weight representation with two programmable RRAM devices. Reprinted with permission from [21] under Creative Commons License

with applied negative bias can be used in parallel to a programmable RRAM with conductance G_+ with applied positive bias such that the equivalent conductance is given by $G = G_+ - G_{\text{ref}}$, thus the modulation of G_+ allows to obtain both positive and negative weights, as shown in Fig. 5(b). A more flexible and granular approach is to use two programmable RRAM devices biased with opposite polarities with conductance G_+ and G_- to represent the overall weights as $G = G_+ - G_-$, as shown in Fig. 5(c). In this way, it is possible to program independently both conductance thus increasing the number of programmable weights in the 2-RRAM structure [70, 71].

In-memory computing can not only accelerate the feed-forward processing, also known as the inference phase, but also the training phase of neural networks [52, 53, 64]. In this case, after the forward evaluation of a single or multiple elements of a dataset, the weights are updated based on a learning rule [68]. A typical supervised training algorithm is backpropagation, where the output state of a neuron y_j is compared with its ideal result o_j and an error $\varepsilon_j = y_j - o_j$ is computed. This error is then back-propagated to the weights that are updated by an amount $\Delta w_{ij} = \eta x_i \varepsilon_j$, where η is the learning rate that controls the speed on which weights are updated and can be an important parameter for controlling convergence and overfitting. In the training operation, to compete with conventional digital hardware, the weight update should be both fast and precise [72], thus the requirements for computational memory are more aggressive. To enable both fast and precise training, an important feature is the linearity of the weight update [73].

The device characterization procedure to demonstrate the feasibility of online network training usually consists of the application of a train of programming pulses with a constant amplitude and shape for the increase and decrease of conductance. The ideal expected result is shown in Fig. 6a, where the weight value as a function of the number of programming pulse increases linearly under applied positive voltage pulses until reaching a maximum value (i.e., 1 on the relative axis of the figure), then returns to 0 under applied negative voltage pulses. The weight update ΔG should be independent of the starting conductance value, thus allowing the weight update even without reading the initial conductance thus speeding up the training process. As shown in Fig. 6b, RRAM devices generally show non-linear potentiation (increase of G) and depression (decrease of G), where the set/reset pulses have an abrupt effect of the conductance change followed by a saturation after a few pulses [71]. RRAM may also have an asymmetrical weight update, as shown in Fig. 6(c), due to different update rates in the potentiation and depression processes. Asymmetric update translates in a larger number of pulses needed for a positive update ΔG than a negative update $-\Delta G$, or vice versa. Usually, there is a characteristic conductance value G_{sym} where the positive and negative increments have the same amplitude [74]. In this case, it is possible to use the scheme of Fig. 5(b) with $G_{\text{ref}} = G_{\text{sym}}$, so that a symmetric potentiation/depression response is obtained [74, 75].

RRAM devices also usually display a limited conductance window spanning from $G_{\min} > 0$ to $G_{\max} < 1$ where the device can be programmed. This result is an offset from the ideal case as shown in Fig. 6d. In such a case, the weight configuration of Fig. 5b can help reaching the desired conductance by carefully tuning G_+ and G_-.

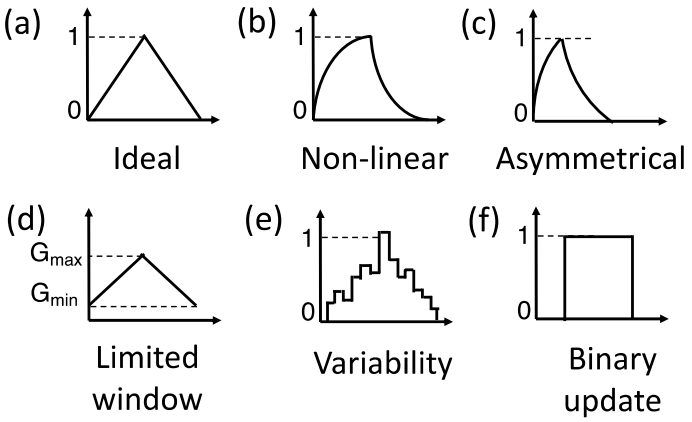

Fig. 6 Weight update characteristics. **a** Ideal weight update characteristics, with the conductance G linearly increasing and decreasing with positive and negative voltages pulses, respectively. **b** Non-linear weight update characteristic with G that after a steep increase saturates. **c** Asymmetrical weight update, with a different response to positive and negative pulses. **d** weight update characteristic with a limited conductance window. **e** Variability in weight update due to cycle-to-cycle variability. **f** Weight update with binary device. Reprinted with permission from [21] under Creative Commons License

In particular, programming the same conductance in G_+ and G_- devices allows to reproduce the case $w_{ij} = 0$, which is among the most probable values within the distribution of synapses in typical neural networks. As already discussed, RRAM also shows stochastic variations in the set and reset processes, which can lead to an unpredictable weight change during training as shown in Fig. 6e, thus affecting significantly the convergence operation. Stuck-on and -off states, where the conductance cannot be updated, further complicates the scenario.

In an extreme case RRAM devices could show only two conductance states (HRS and LRS) [76, 77] resulting in a binary weight update scheme as shown in Fig. 6f. This makes the weight encoding inherent digital which is suitable for the acceleration of a binary neural network (BNN). Training a BNN can be challenging due to the abrupt change of conductance. Figure 7a illustrates a stochastic approach for training BNNs using binary RRAM devices with an internal parameter controllable with the application of multiple programming pulses [76]. Two different weights value can be associated with the RRAM device, namely W_{int} and W_{ext} corresponding to a non-observable internal variable and the externally measured weight, respectively. W_{int} may correspond to the defect density and filament configuration within the device while W_{ext} corresponds to the measured conductance, as shown in Fig. 7a. The binary weight W_{ext} can only assume two values, 0 if the defects do not connect TE and BE, and 1 otherwise. By the application of multiple set and reset pulses it is possible to change the filament configuration, hence the continuous update of W_{int}, while W_{ext} only changes after a certain threshold is reached. This hybrid binary/analogue update can be adopted within a conventional backpropagation algorithm for training

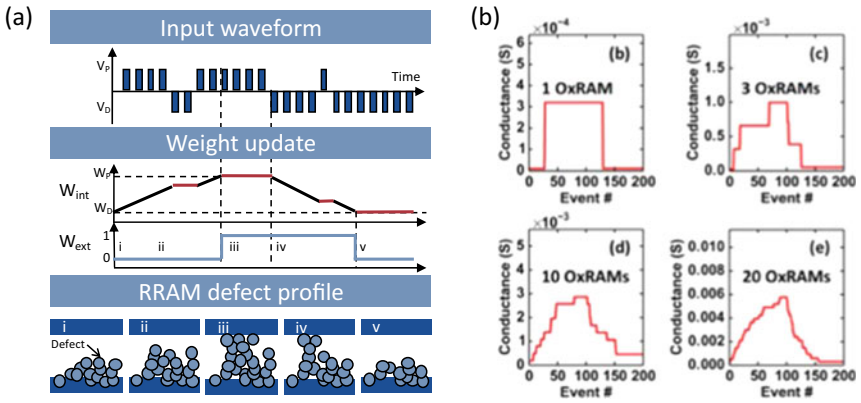

Fig. 7 Using binary devices for training neural networks. **a** Stochastic weight updated, where the state variable is represented by an internal weights which is updated by positive/negative pulses (top) and it is not measurable, and an external binary weight which is updated after multiple positive/negative pulses (center). The internal weight update represents a modification in the defect configuration while the external weight changes if a connection is created or broken from TE to BE (bottom). **b** Multiple binary devices used as synapse to represent an analog weight. Reprinted with permission from [76, 78]. Copyright 2017, 2015 IEEE

a BNN as if it was an analogue neural network. Similarly, W_{ext} can be calculated after measuring a W_{int} value, making it possible to use an analogue memory for training a BNN which usually shows high precision [77]. Another approach is illustrated in Fig. 7b, where multiple binary RRAM devices are used for training a conventional neural network [78]. In this procedure, multiple devices are connected in parallel to represent multiple weights. Every time an individual binary RRAM device increases/decreases its conductance, the overall weight experiences a corresponding incremental step. As the number of devices increases, the synaptic weight becomes increasingly analogue, thus making the characteristic similar to the ideal one. This comes at the cost of an increased area occupation. However, multiple devices in parallel can also be used to mitigate the effect of non-idealities and stochastic weight update [79] thus serving as a regularization of individual variations to achieve a more gradual weight update response in the presence of particularly unprecise devices.

RRAM devices have also been shown to be able of brain-inspired spike-based neural network implementation [18, 19, 80, 81]. In this case, information is usually encoded in spikes similarly to our brain, where learning takes place according to unsupervised weight update rules depending on the spike timing, such as the spike timing dependent plasticity (STDP) [18], the spike-rate dependent plasticity (SRDP) [80, 81] or semi-supervised training approaches implementing a teacher signal [19]. In most practical implementations, RRAM devices are used as artificial synapses, although fully-memristive architecture with RRAM-based synapses and neurons have also been presented [82].

3.2 In-Memory Optimization Accelerators

MVM is also at the core of many optimization algorithms, such as linear or quadratic programming techniques [83]. Specifically-designed neural networks can be implemented for searching the minimum of an energy landscape, usually relying on Hopfield neural networks (HNN) [84], namely recurrent neural networks where each neuron is connected to all the others with symmetric links ($w_{ij} = w_{ji}$) and no self-connection ($w_{ii} = 0$). This brain-inspired recurrent connectivity offers interesting cognitive functions, such as attractor learning/recall and associative memory [85], that have also been demonstrated with in-memory computing hardware [86–88].

HNNs also have shown the ability of solving constraint satisfaction problems (CSP) [89], which are ubiquitous in many different application fields [90]. In this case, every neuron has a highly nonlinear activation function and represents a state of the network, while connectivity between neurons define the constraints. By initializing a random input state, the network can gradually update its states and converge to an optimized final state by minimizing the energy landscape cost function given by:

$$E = -\frac{1}{2}\sum_{i,j}^{N} w_{ij} v_i v_j \tag{2}$$

where N is the total number of neurons and v_i represents the state of neuron i. The binary neuron state is updated depending on the evaluation of the input function $u_i = \sum_{i \neq j} w_{ij} v_j$ compared with a given threshold θ_i. Figure 8a shows an example of a convex energy cost function, which can be explored by the HNN to find the minimum, i.e. the optimization problem solution. Convex problems can be computed straightforwardly by conventional gradient descent techniques. However, when the the problem size and difficulty may quickly increase in typical CSP, which are known to become aggressively difficult as in the case of non-deterministic polynomial (NP) or NP-hard problems. This is due to the increase of the number of local minima in the energy landscape, where the the HNN state can be stuck as illustrated in Fig. 8b.

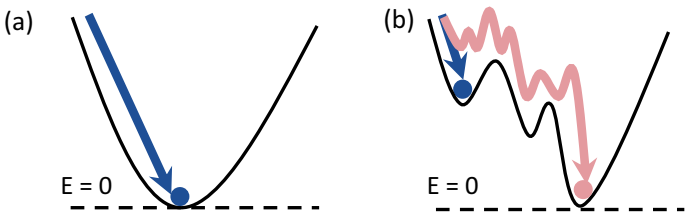

Fig. 8 Energy landscapes of optimization problems. **a** Search of the minimum of a convex energy landscape with a Hopfield neural network. The solution (blue) can reach it efficiently. **b** Search of the global minimum of a non-convex energy landscape with the deterministic solution (blue) being stuck in a local minimum. By adding noise (red) the solution can efficiently reach the global minimum

Non-convex CSPs can be solved by the simulated annealing technique [91]. By stimulating the HNN with random noise, it is possible to 'heat' the system, thus helping the state in escaping from the local minima and eventually reaching the correct solution in the global minimum of the energy function. Simulated annealing accelerators have been demonstrated by conventional CMOS circuits [92–95], quantum computing technologies [96, 97], optical computing technologies [98] and analogue in-memory computing [57–60, 99–102]. In the latter, memory devices can act both as MVM accelerators for the inference of the HNN and annealers by the generation of intrinsic random noise.

Figure 9a shows a conceptual circuit schematic of a HNN for accelerated annealing [59], which is based on an MVM current which is then fed back into the input neurons. The feedback is obtained by sampling the columns currents with an analogue-to-digital converter (ADC), post-processing it to obtain the neuron states and applying it to the crosspoint rows in either digital or analogue mode. The constraints are encoded in the weight matrix of the crosspoint conductance, while the intrinsic noise to stimulate the annealing can be generated by various techniques. For instance, one can leverage RRAM inherent stochasticity such as RTN [103] and 1/f noise [104], which can automatically stimulate the simulated annealing [102]. An additional crosspoint column can be used to program RRAM devices appropriately and then harvest noise

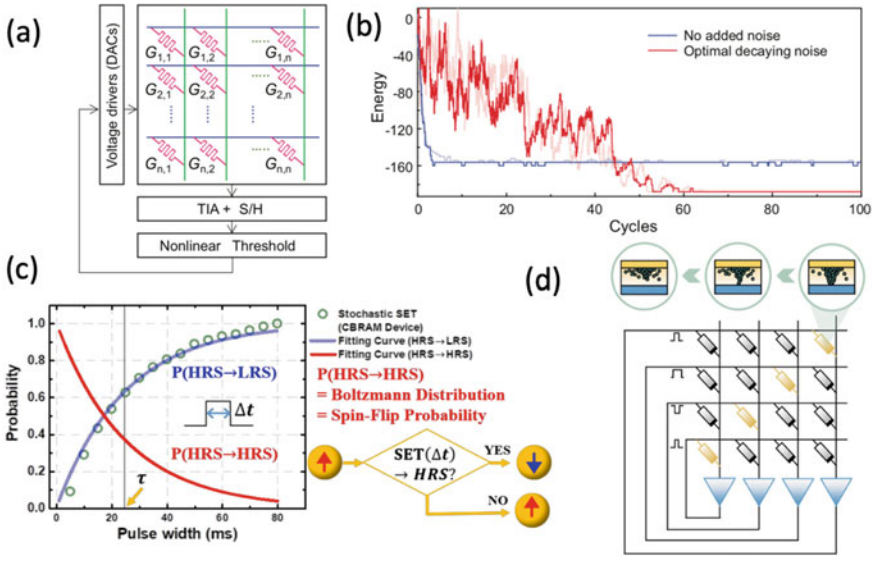

Fig. 9 In-memory simulated annealing techniques. **a** an extra column of a crosspoint array can be used to generate noise which is summed to the MVM response to perform simulated annealing. **b** Hopfield energy as function of iteration cycles for different noise levels. **c** stochastic switching of a RRAM device to generate random flip of a neuron state. The probability of switching can be finely tuned by regulating the set pulse width. **d** Hopfield network working in the chaotic regime by inserting diagonal connection that are gradually reduced in weight cooling the overall annealing procedure. Reprinted with permission from [58, 60, 66]. Copyright 2019, 2020 IEEE

with the desired figure to speed up optimization [59]. In fact, noise should be modulated based on the annealing scheme and should also change its characteristic during the annealing procedure, ideally reducing its magnitude while reaching the global minimum. Figure 9b shows the Hopfield energy to solve a 60-node Max-Cut problem as a function of iteration cycles for different noise levels [59]. A noise-free calculation results in a higher energy compared to noisy calculations, thus supporting the fundamental contribution of noise for the annealing. Also note that noise with large amplitude might be inefficient for reaching the energy minimum, since the state might also escape from the energy global minimum in this case. The tradeoff between exploratory and greedy strategies should be therefore carefully considered. A second approach is to use the RRAM stochasticity in its switching to generate a flip of one or more neurons with a given probability as shown in Fig. 9c [58]. In fact, the probability of setting a RRAM device can be controlled with the set pulse itself either by the voltage pulse amplitude [20] or the pulse width [58]. In this way, after a careful characterization of the set statistics of the RRAM device, it is possible to create naturally a stochastic, e.g., Gaussian, distribution from the device physics. A third approach is to operate the HNN in its chaotic behavior [105], by violating one of its definitions, namely $w_{ii} = 0$ by connecting each neuron in a self-feedback to itself as shown in Fig. 9d [57, 60]. By gradually resetting the self-feedback RRAM device the chaos can be reduced to let the system effectively reach the global minimum.

Optimizer circuits based on hardware HNN accelerated by RRAM crosspoint arrays have been shown to have better performance than traditional, optical and quantum computing [59], thanks to the low energy MVM operation and intrinsic, compact noise generators.

4 In-Memory Computing Architecture for Inverse MVM

MVM can be used to accelerate algebraic problems, such as the solution of linear systems and partial differential equations [63, 106]. However, this is usually done by iteratively performing the MVM operation, digitalize the currents with an ADC, post-process them and apply the correct output vector as input for the following MVM. While MVM can indeed accelerate the algebraic problem, solution compared with digital approaches, the number of iterations for reaching the convergence can be extremely large. To further accelerate the problem solution, the feedback operation can be obtained within the analog domain, by operational amplifiers connected between rows and columns of crosspoint arrays [31, 48, 107] as shown in Fig. 10a. Given a crosspoint array programmed with a conductance matrix G, by injecting a current vector I to its rows connected to the negative input of an operational amplifier (OA) with the positive input connected to ground, the columns connected to the OA outputs will adjust to a voltage V such that the overall current flowing within the OA is zero due to its high input impedance and negative feedback effect, namely $I + GV = 0$ [48]. This leads to:

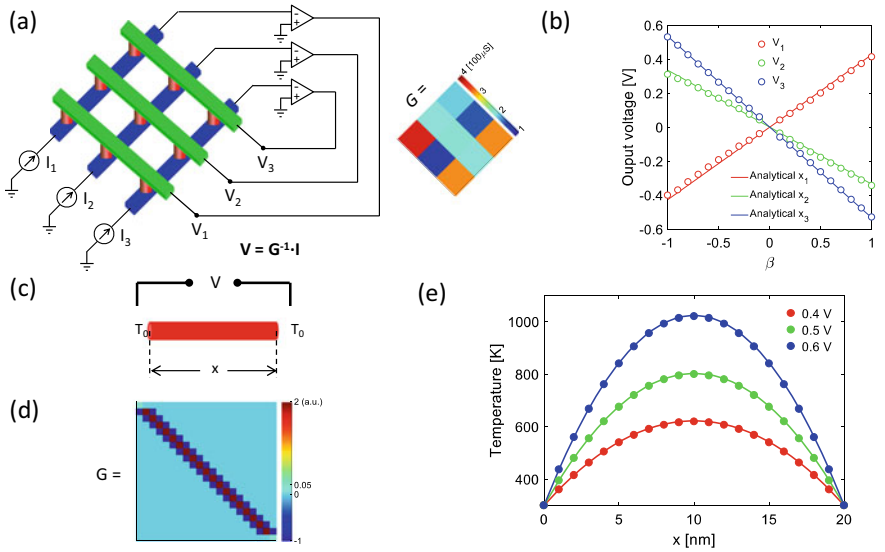

Fig. 10 In-memory solution of linear systems with IMVM. **a** IMVM circuit where a current is injected in a crosspoint programmed with conductance G rows which are connected to the virtual ground of operational amplifiers whose output is connected to the crosspoint columns. The result gives $V = G^{-1} I$. An example of measured 3×3 programmed matrix (inset). **b** Circuit output voltage as function of β with an input current $I = \beta I_0$. **c** representation of a 1-dimensional Fourier heat equation problem. **d** Fourier equation encoded in a crosspoint array. **e** Circuit simulation (circles) results compared with analytical solution showing good agreement. Reprinted with permission from [48] under Creative Commons License

$$V = -G^{-1} I \tag{3}$$

which corresponds to the solution of a linear system. In fact, by encoding in G a problem A and injecting a current $-I$ corresponding to a known term b, the resulting output voltage will be equal to $x = A^{-1} b$ as shown in Fig. 10a. This operation can also be referred as inverse MVM (IMVM), as the solution is the vector which must be multiplied to the given matrix to yield a certain output vector. Figure 10b shows an experimental demonstration of this circuit, where a 3×3 crosspoint array of RRAM devices was connected in feedback with OAs on a printed circuit board (PCB) [48]. By applying an input current vector with amplitude $I = \beta I_0$, where I_0 is a normalized vector and β represents the magnitude of the input vector, the measured voltage at the OA output displays a linear dependence on β as expected from the linearity of the system of equation, thus demonstrating the feasibility of the circuit in the solution of the linear system. Interestingly, the solution of a linear system is obtained in just one step by the circuit, without any iteration. Moreover, the time to solution does not depend on the matrix size, thus making the time complexity of the circuit constant, i.e., $O(1)$ complexity [108, 109]. This is extremely attractive in comparison with traditional algorithms such as conjugate gradient [110] or quantum

computing algorithms such as the Harrow-Hassidim-Lloyd (HHL) algorithm [111], which show a time complexity of $O(N)$ and $O(\log(N))$, respectively. Figure 10c shows a real word problem, the solution of a 1-dimensional steady-state Fourier equation for heat diffusion encoded in a crosspoint array and solved by the IMVM circuit. Matrix G in Fig. 10d represents the system of linear equations which describes the differential Fourier equation in the discrete domain by the finite difference method (FDM). Note that G displays both positive and negative coefficients, which can be encoded with a method similar to Fig. 5c, where the output voltage of the OAs are inverted and applied to a second crossbar representing the negative entries [48]. The known term encoded in the input currents correspond to the dissipated power in the one-dimensional structure and the output voltage represents the temperature profile along the 1-dimensional structure. Figure 11e shows the results obtained by a SPICE simulation of the IMVM circuit compared with the analytical solution for different voltage applied to the 1-dimensional structure, highlighting a accurate match between the ideal analytical solution of the equation and the circuit simulations.

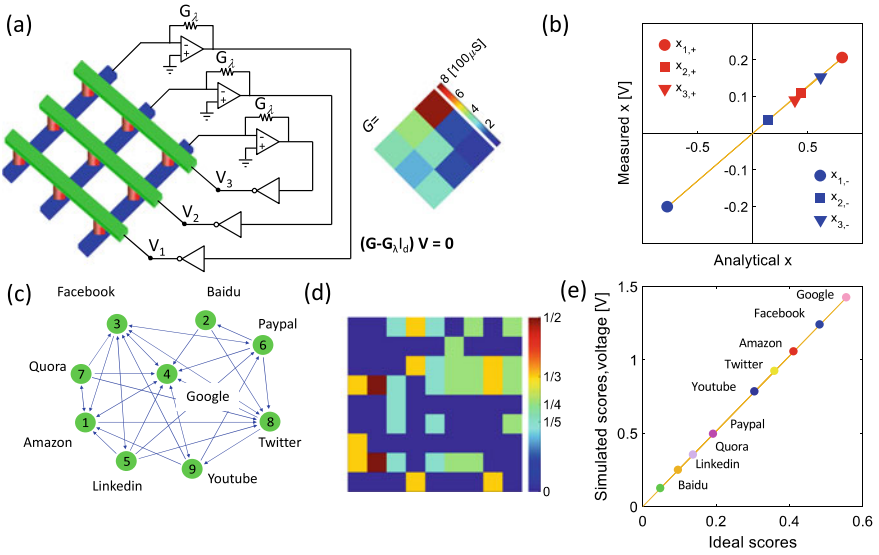

Fig. 11 In-memory eigenvectors calculation with IMVM. **a** IMVM circuit for eigenvector calculation, where no input is given and a conductance corresponding to the maximum eigenvalue G_λ is programmed in the TIA conductance. Inset shows a programmed measured matrix. **b** measured eigenvectors as function of the analytical calculation showing good agreement. **c** Graph of webpages used for Pagerank problem, where every circle is a webpage and the arrows represent citations. **d** Corresponding stochastic link matrix. **e** Simulated circuit result as function of the ideal scores showing good agreement. Reprinted with permission from [48] under Creative Commons License

4.1 In-Memory Eigenvector Calculation

By slightly modifying the topology of the analogue circuit in Fig. 10a, it is possible to calculate the principal eigenvector of the matrix programmed in the crosspoint [31, 48]. This is shown in Fig. 12a, where the matrix G is mapped in one crosspoint array and the principal eigenvalue λ of matrix G is mapped in the feedback resistor of the trans-impendence amplifier (TIA)s. A set of inverting OAs is then added in the feedback loop to compensate for the minus sign arising from the current–voltage conversion $V = -I/G_\lambda$ of the TIAs in Fig. 11a. The circuit is described by (3) with zero input current, thus leading to $(G - G_\lambda I)V = 0$, which corresponds to the non-trivial solution of the eigenvector problem for G. Figure 11b shows an experimental demonstration of the eigenvector circuit, showing the correlation plot of experimental components as a function of the ideal analytical values, for the eigenvectors corresponding to the maximum (principal) and the minimum eigenvalue [48].

The calculation of the principal eigenvector can be applied to relevant scientific computing tasks, such as the solution of the Schrödinger equation [48]. However,

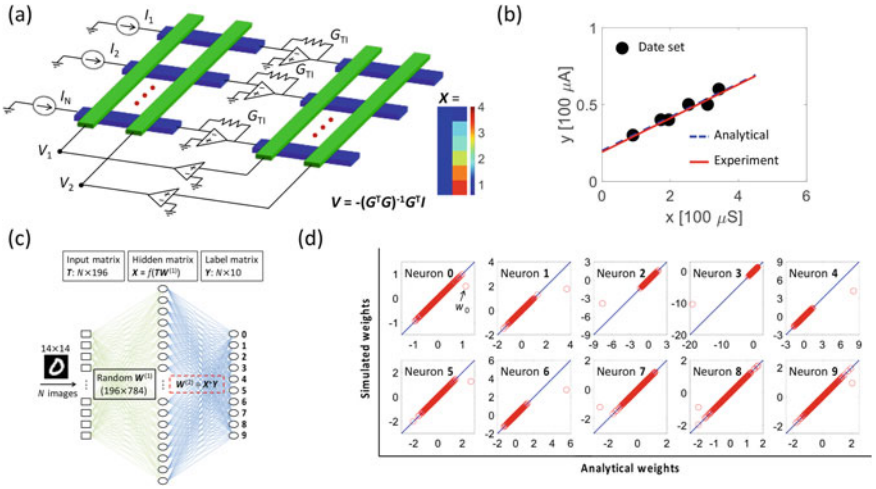

Fig. 12 In-memory regression calculation with IMVM. **a** IMVM circuit for Moore-Penrose pseudoinverse with a current I injected in a crosspoint array programmed with conductance G rows connected to a TIA whose output drives the rows of a second crosspoint array programmed with G^T. The column of the second crosspoint are connected to operational amplifiers whose outputs close the loop and are connected to the first crosspoint columns. The output voltage gives $V = -(G^{\mathrm{T}}G)^{-1}G^{\mathrm{T}}I$ which is the Moore Penrose pseudoinverse result. Inset shows a programmed linear regression problem. **b** measured fitting and analytical fitting of a programmed dataset showing good agreement. **c** ELM schematic for recognition of MNIST dataset with a random input layer and an output layer trained with logistic regression. **d** Circuit simulated weights as function of the analytical weights for the output layer showing a good agreement. Reprinted with permission from [107] under Creative Commons License

scientific computing requires high precision which is relatively difficult for in-memory computing due to imprecise RRAM programming. This problem can be circumvented by using IMVM for calculating a seed that is then refined with traditional computing technologies [106]. On the other hand, machine learning problems usually are less subject to noise and less sensitive to variations. For example, Pagerank [112], which is the algorithm that calculates webpage ranking on a search engine, requires the computation of the principal eigenvector of a link matrix corresponding to the adjacency between webpages as shown by the graph in Fig. 11c. Interestingly, the encoded matrix can be pre-processed to obtain a stochastic matrix (Fig. 11d) where the summation over all the columns is 1 and the principal eigenvalue is 1, thus making the IMVM circuit ideal for Pagerank calculation. Figure 11e shows a SPICE simulation of the Pagerank algorithm with IMVM compared with the analytical solution, highlighting the good agreement between the page scores [48]. The Pagerank problem particularly fits IMVM circuits also because the exact ranking is less important than the overall one, in fact users are usually interested in the first 10 webpages being displayed correctly in the Pagerank response, even if they are not listed in the correct order. For a more detailed assessment, the problem has been studied for a relatively large scale implementation with real conductance values programmed on HfO_x RRAM devices, showing a relatively low mismatch once a fine tuning of the conductance is performed [31]. The eigenvector calculation by IMVM has been shown to display a $O(1)$ time complexity, with an unprecedented speedup compared with other technologies [113].

4.2 Pseudoinverse and Regression Accelerators

Many machine learning problems can be written as the over-determined linear system $Xw = y$, where X is a rectangular $N \times M$ matrix with $N > M$ which encodes the explanatory variables, y is a $N \times 1$ known vector representing the dependent variables and w it the $M \times 1$ weight vector. Since this equation generally does not have a solution, its best approximation can be found by the linear regression, namely the least square error (LSE) algorithm that minimizes the Euclidean norm of the error, namely $\min \|Xw - y\|_2$. This minimization can be carried out by the pseudo-inverse, or Moore–Penrose inverse, namely matrix X^+ given by $X^+ = (X^T X)^{-1} X^T$, while the weights are given by $w = X^+ y$ [114]. Figure 12a shows an analogue IMVM circuit that can calculate the Moore–Penrose inverse matrix [107]. Two identical crosspoint arrays are used to map the matrix X in their conductance$G$. A vector of currents I representing the known term y is applied to the first crosspoint rows which are connected to the negative input of the TIAs with a feedback conductance G_{TI}. The first crosspoint columns are connected to the output terminals of a second stage of OAs with output voltage$V$. The first crosspoint execute the summation of input currents I and the MVM output GV, thus yielding an overall current $I + GV$ flowing into the TIAs. The latter develop a voltage across the second crosspoint array

given by $-(I + VG)G_{TI}^{-1}$. The columns of the second crosspoint are connected to the positive inputs of the second stage of OAs thus must be equal to zero, which translates in the equation $(I + VG)G_{TI}^{-1}G^T = 0$ or equivalently:

$$V = -\left(G^T G\right)^{-1} G^T I \tag{4}$$

where the Moore–Penrose inverse matrix G^+ is clearly identified [107]. The inset of Fig. 12a shows the 2×6 matrix G which was mapped in a HfO_x RRAM crosspoint array representing the explanatory variables of a linear regression problem with the experimental solution that gives the best linear fit plotted in Fig. 12b and compared with the analytical calculation showing a good agreement [107].

This concept can be extended to the logistic regression which is a powerful classification algorithm. In fact, by properly writing in different columns of matrix G the coordinates of the data and injecting a relative current 1 or 0 corresponding to the class, it is possible to obtain the straight line that best separates the input data. This is a powerful tool in machine learning as it can be used for training the classification layer of a neural network. Figure 12c shows the conceptual schematic of a fully-connected, 2-layer neural network according to the Extreme Learning Machine (ELM) model [107], where all synaptic weights in the first layer are assumed to be random weights, while the synaptic weights in the output layer are trained by logistic regression. SPICE simulations of the IMVM circuit for training the output layer of the ELM model for classifying handwritten digits of the MNIST dataset [68] are shown in Fig. 12d and compared with the analytical solution. The results indicate a good agreement with the ideal solution, thus supporting the feasibility of IMVM circuits for training neural networks [107].

5 Conclusions

This chapter presents an overview of analogue in-memory computing concepts with RRAM devices. RRAM displays ideal properties for computing, including high density, analogue storage and the ability for 3D integration. MVM in the analogue domain is perhaps the most promising type of in-memory computing function which is made possible by a RRAM array, typically with 1T1R structure of the individual memory cell. Experiments and simulations show an unprecedented speed up of MVM for neural networks acceleration and CSP optimization, while IMVM displays strong advantages in terms of computational complexity and energy efficiency for algebraic and machine learning problems. At the same time, the RRAM technology and its operations should be optimized to fulfil all requirements of multibit operation, fast switching, controllable noise and long retention time necessary for enabling this technology in a relevant environment in the edge or cloud. In particular, the RRAM technology development and computing architecture research should proceed with strong synergy to fully take advantage of the energy and performance benefits of in-memory computing.

References

1. G.E. Moore, 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 (2006). https://doi.org/10.1109/N-SSC.2006.4785860
2. S. Salahuddin, K. Ni, S. Datta, The era of hyper-scaling in electronics. Nat. Electron. **1**(8), 442–450 (2018). https://doi.org/10.1038/s41928-018-0117-x
3. D. Amodei, D. Hernandez, AI and compute, https://openai.com/blog/ai-and-compute/
4. J. von Neumann, First Draft of a Report on the EDVAC (1945). https://doi.org/10.5555/1102046
5. P.A. Merolla et al., A million spiking-neuron integrated circuit with a scalable communication network and interface. Science **345**(6197), 668–673 (2014). https://doi.org/10.1126/science.1254642
6. M.A. Zidan, J.P. Strachan, W.D. Lu, The future of electronics based on memristive systems. Nat. Electron. **1**(1), 22–29 (2018). https://doi.org/10.1038/s41928-017-0006-8
7. D. Ielmini, H.-S.P. Wong, In-memory computing with resistive switching devices. Nat. Electron. **1**(6), 333–343 (2018). https://doi.org/10.1038/s41928-018-0092-2
8. Z. Wang et al., Resistive switching materials for information processing. Nat. Rev. Mater. (2020). https://doi.org/10.1038/s41578-019-0159-3
9. L. Chua, Memristor-The missing circuit element. IEEE Trans. Circuit Theory **18**(5), 507–519 (1971). https://doi.org/10.1109/TCT.1971.1083337
10. D.B. Strukov, G.S. Snider, D.R. Stewart, R.S. Williams, The missing memristor found. Nature **453**(7191), 80–83 (2008). https://doi.org/10.1038/nature06932
11. D. Ielmini, Resistive switching memories based on metal oxides: mechanisms, reliability and scaling. Semicond. Sci. Technol. **31**(6), 063002 (2016). https://doi.org/10.1088/0268-1242/31/6/063002
12. T. Mikolajick et al., FeRAM technology for high density applications. Microelectron. Reliab. **41**(7), 947–950 (2001). https://doi.org/10.1016/S0026-2714(01)00049-X
13. J. Grollier, D. Querlioz, K.Y. Camsari, K. Everschor-Sitte, S. Fukami, M.D. Stiles, Neuromorphic spintronics. Nat. Electron. (2020). https://doi.org/10.1038/s41928-019-0360-9
14. Z. Sun, E. Ambrosi, A. Bricalli, D. Ielmini, Logic Computing with Stateful Neural Networks of Resistive Switches. Adv. Mater. **30**(38), 1802554 (2018). https://doi.org/10.1002/adma.201802554
15. G. Indiveri, B. Linares-Barranco, R. Legenstein, G. Deligeorgis, T. Prodromakis, Integration of nanoscale memristor synapses in neuromorphic computing architectures. Nanotechnology **24**(38), 384010 (2013). https://doi.org/10.1088/0957-4484/24/38/384010
16. C. Zamarreño-Ramos, L.A. Camuñas-Mesa, J.A. Pérez-Carrasco, T. Masquelier, T. Serrano-Gotarredona, B. Linares-Barranco, On Spike-Timing-Dependent-Plasticity, Memristive Devices, and Building a Self-Learning Visual Cortex. Front. Neurosci. **5** (2011). https://doi.org/10.3389/fnins.2011.00026
17. S.H. Jo, T. Chang, I. Ebong, B.B. Bhadviya, P. Mazumder, W. Lu, Nanoscale memristor device as synapse in neuromorphic systems. Nano Lett. **10**(4), 1297–1301 (2010). https://doi.org/10.1021/nl904092h
18. G. Pedretti et al., Memristive neural network for on-line learning and tracking with brain-inspired spike timing dependent plasticity. Sci. Rep. **7**(1), 5288 (2017). https://doi.org/10.1038/s41598-017-05480-0
19. W. Wang et al., Learning of spatiotemporal patterns in a spiking neural network with resistive switching synapses. Sci. Adv. **4**(9), eaat4752 (2018). https://doi.org/10.1126/sciadv.aat4752
20. R. Carboni, D. Ielmini, Stochastic memory devices for security and computing. Adv. Electron. Mater. **5**(9), 1900198 (2019). https://doi.org/10.1002/aelm.201900198
21. D. Ielmini, G. Pedretti, Device and Circuit architectures for in-memory computing. Adv. Intell. Syst. **2**(7), 2000040 (2020). https://doi.org/10.1002/aisy.202000040
22. H.-S.P. Wong et al., Metal–Oxide RRAM. Proc. IEEE **100**(6), 1951–1970 (2012). https://doi.org/10.1109/JPROC.2012.2190369

23. S. Raoux, W. Wełnic, D. Ielmini, Phase change materials and their application to nonvolatile memories. Chem. Rev. **110**(1), 240–267 (2010). https://doi.org/10.1021/cr900040x
24. G.W. Burr et al., Phase change memory technology. J. Vac. Sci. Technol., B: Nanotechnol. Microelectron.: Mater. Process. Meas. Phenom. **28**(2), 223–262 (2010). https://doi.org/10.1116/1.3301579
25. T. S. Boscke, J. Muller, D. Brauhaus, U. Schroder, U. Bottger, Ferroelectricity in hafnium oxide: CMOS compatible ferroelectric field effect transistors, in *2011 International Electron Devices Meeting* (Washington, DC, USA, 2011), pp. 24.5.1–24.5.4. https://doi.org/10.1109/IEDM.2011.6131606
26. H. Mulaosmanovic et al., Novel ferroelectric FET based synapse for neuromorphic systems, in *2017 Symposium on VLSI Technology* (Kyoto, Japan, 2017), pp. T176–T177. https://doi.org/10.23919/VLSIT.2017.7998165
27. C. Chappert, A. Fert, F.N. Van Dau, The emergence of spin electronics in data storage. Nat. Mater. **6**, 813–823 (2007)
28. B. Govoreanu et al., 10x10nm^2 Hf/HfOx crossbar resistive RAM with excellent performance, reliability and low-energy operation, in *2011 International Electron Devices Meeting* (Washington, DC, USA, 2011), pp. 31.6.1–31.6.4. https://doi.org/10.1109/IEDM.2011.6131652
29. A.C. Torrezan, J.P. Strachan, G. Medeiros-Ribeiro, R.S. Williams, Sub-nanosecond switching of a tantalum oxide memristor. Nanotechnology **22**(48), 485203 (2011). https://doi.org/10.1088/0957-4484/22/48/485203
30. S. Yu, H.-Y. Chen, B. Gao, J. Kang, H.-S.P. Wong, HfO_x-based vertical resistive switching random access memory suitable for bit-cost-effective three-dimensional cross-point architecture. ACS Nano **7**(3), 2320–2325 (2013). https://doi.org/10.1021/nn305510u
31. Z. Sun, E. Ambrosi, G. Pedretti, A. Bricalli, D. Ielmini, In-Memory PageRank Accelerator With a Cross-Point Array of Resistive Memories. IEEE Trans. Electron Devices **67**(4), 1466–1470 (2020). https://doi.org/10.1109/TED.2020.2966908
32. J.J. Yang, D.B. Strukov, D.R. Stewart, Memristive devices for computing. Nature Nanotech. **8**(1), 13–24 (2013). https://doi.org/10.1038/nnano.2012.240
33. S. N. Truong, K.-S. Min, New memristor-based crossbar array architecture with 50-% area reduction and 48-% power saving for matrix-vector multiplication of analog neuromorphic computing. JSTS: J. Semicond. Technol. Sci. **14**(3), 356–363 (2014). https://doi.org/10.5573/JSTS.2014.14.3.356
34. M. Hu et al., Memristor-based analog computation and neural network classification with a dot product engine. Adv. Mater. **30**(9), 1705914 (2018). https://doi.org/10.1002/adma.201705914
35. M.-C. Hsieh et al., Ultra high density 3D via RRAM in pure 28nm CMOS process, in *2013 IEEE International Electron Devices Meeting* (Washington, DC, USA, 2013), pp. 10.3.1–10.3.4. https://doi.org/10.1109/IEDM.2013.6724600
36. E. Linn, R. Rosezin, C. Kügeler, R. Waser, Complementary resistive switches for passive nanocrossbar memories. Nat. Mater. **9**(5), 403–406 (2010). https://doi.org/10.1038/nmat2748
37. D. Ielmini, Y. Zhang, Physics-based analytical model of chalcogenide-based memories for array simulation, in *2006 International Electron Devices Meeting* (San Francisco, CA, USA, 2006), pp. 1–4. https://doi.org/10.1109/IEDM.2006.346795
38. L. Gao, P.-Y. Chen, R. Liu, S. Yu, Physical unclonable function exploiting sneak paths in resistive cross-point array. IEEE Trans. Electron Devices **63**(8), 3109–3115 (2016). https://doi.org/10.1109/TED.2016.2578720
39. F. Li, X. Yang, A.T. Meeks, J.T. Shearer, K.Y. Le, Evaluation of SiO_2 antifuse in a 3D-OTP memory. IEEE Trans. Device Mater. Relib. **4**(3), 416–421 (2004). https://doi.org/10.1109/TDMR.2004.837118
40. Tz-Yi Liu et al., A 130.7mm^2 2-layer 32Gb ReRAM memory device in 24nm technology, in *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers* (San Francisco, CA, 2013), pp. 210–211. https://doi.org/10.1109/ISSCC.2013.6487703

41. G.W. Burr et al., Access devices for 3D crosspoint memory. J. Vac. Sci. Technol. B, Nanotechnolo. Microelectronics: Mater. Process. Meas. Phenom. **32**(4), 040802 (2014). https://doi.org/10.1116/1.4889999
42. D. Ielmini, Modeling the universal set/reset characteristics of bipolar rram by field- and temperature-driven filament growth. IEEE Trans. Electron Devices **58**(12), 4309–4317 (2011). https://doi.org/10.1109/TED.2011.2167513
43. V. Milo et al., Multilevel HfO_2-based RRAM devices for low-power neuromorphic networks. APL Mater. **7**(8), 081120 (2019). https://doi.org/10.1063/1.5108650
44. C. Li et al., Analogue signal and image processing with large memristor crossbars. Nat Electron **1**(1), 52–59 (2018). https://doi.org/10.1038/s41928-017-0002-z
45. A. Shafiee et al., ISAAC: a convolutional neural network accelerator with in-situ analog arithmetic in crossbars, in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)* (Seoul, South Korea, 2016), pp. 14–26. https://doi.org/10.1109/ISCA.2016.12
46. S. Balatti, S. Ambrogio, D.C. Gilmer, D. Ielmini, Set variability and failure induced by complementary switching in bipolar RRAM. IEEE Electron Device Lett. **34**(7), 861–863 (2013). https://doi.org/10.1109/LED.2013.2261451
47. S. Ambrogio, S. Balatti, A. Cubeta, A. Calderoni, N. Ramaswamy, D. Ielmini, Statistical fluctuations in HfO_x resistive-switching memory: Part I-set/reset variability. IEEE Trans. Electron Devices **61**(8), 2912–2919 (2014). https://doi.org/10.1109/TED.2014.2330200
48. Z. Sun, G. Pedretti, E. Ambrosi, A. Bricalli, W. Wang, D. Ielmini, Solving matrix equations in one step with cross-point resistive arrays. Proc Natl Acad Sci USA **116**(10), 4123–4128 (2019). https://doi.org/10.1073/pnas.1815682116
49. Y.-H. Lin et al., Performance impacts of analog ReRAM non-ideality on neuromorphic computing. IEEE Trans. Electron Devices **66**(3), 1289–1295 (2019). https://doi.org/10.1109/TED.2019.2894273
50. S. Balatti et al., Voltage-controlled cycling endurance of HfO_x-based resistive-switching memory. IEEE Trans. Electron Devices **62**(10), 3365–3372 (2015). https://doi.org/10.1109/TED.2015.2463104
51. S. Ambrogio, S. Balatti, V. McCaffrey, D.C. Wang, D. Ielmini, Noise-induced resistance broadening in resistive switching memory—Part II: array statistics. IEEE Trans. Electron Devices **62**(11), 3812–3819 (2015). https://doi.org/10.1109/TED.2015.2477135
52. C. Li et al., Efficient and self-adaptive in-situ learning in multilayer memristor neural networks. Nat. Commun. **9**(1), 2385 (2018). https://doi.org/10.1038/s41467-018-04484-2
53. P. Yao et al., Face classification using electronic synapses. Nat. Commun. **8**(1), 15199 (2017). https://doi.org/10.1038/ncomms15199
54. Z. Wang et al., Reinforcement learning with analogue memristor arrays. Nat. Electron. **2**(3), 115–124 (2019). https://doi.org/10.1038/s41928-019-0221-6
55. Z. Wang et al., In situ training of feed-forward and recurrent convolutional memristor networks. Nat. Mach. Intell. **1**(9), 434–442 (2019). https://doi.org/10.1038/s42256-019-0089-1
56. P.M. Sheridan, F. Cai, C. Du, W. Ma, Z. Zhang, W.D. Lu, Sparse coding with memristor networks. Nat. Nanotech. **12**(8), 784–789 (2017). https://doi.org/10.1038/nnano.2017.83
57. M.R. Mahmoodi, M. Prezioso, D.B. Strukov, Versatile stochastic dot product circuits based on nonvolatile memories for high performance neurocomputing and neurooptimization. Nat. Commun. **10**(1), 5113 (2019). https://doi.org/10.1038/s41467-019-13103-7
58. J.H. Shin, Y.J. Jeong, M.A. Zidan, Q. Wang, W.D. Lu, Hardware Acceleration of simulated annealing of spin glass by RRAM crossbar array, in *2018 IEEE International Electron Devices Meeting (IEDM)* (San Francisco, CA, 2018), pp. 3.3.1–3.3.4. https://doi.org/10.1109/IEDM.2018.8614698
59. F. Cai et al., Power-efficient combinatorial optimization using intrinsic noise in memristor Hopfield neural networks. Nat. Electron. (2020). https://doi.org/10.1038/s41928-020-0436-6
60. K. Yang, Q. Duan, Y. Wang, T. Zhang, Y. Yang, R. Huang, Transiently chaotic simulated annealing based on intrinsic nonlinearity of memristors for efficient solution of optimization problems. Sci. Adv. **6**(33), pp. eaba9901 (2020). https://doi.org/10.1126/sciadv.aba9901

61. M.R. Mahmoodi, D.B. Strukov, O. Kavehei, Experimental demonstrations of security primitives with nonvolatile memories. IEEE Trans. Electron Devices **66**(12), 5050–5059 (2019). https://doi.org/10.1109/TED.2019.2948950
62. H. Nili et al., Hardware-intrinsic security primitives enabled by analogue state and nonlinear conductance variations in integrated memristors. Nat. Electron. **1**(3), 197–202 (2018). https://doi.org/10.1038/s41928-018-0039-7
63. M.A. Zidan et al., A general memristor-based partial differential equation solver. Nat. Electron. **1**(7), 411–420 (2018). https://doi.org/10.1038/s41928-018-0100-6
64. P. Yao et al., Fully hardware-implemented memristor convolutional neural network. Nature **577**(7792), 641–646 (2020). https://doi.org/10.1038/s41586-020-1942-4
65. F. Cai et al., A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations. Nat. Electron. **2**(7), 290–299 (2019). https://doi.org/10.1038/s41928-019-0270-x
66. C. Li et al., CMOS-integrated nanoscale memristive crossbars for CNN and optimization acceleration, in *2020 IEEE International Memory Workshop (IMW)* (Dresden, Germany, 2020), pp. 1–4. https://doi.org/10.1109/IMW48823.2020.9108112
67. S. Yin, S. Yu, High-throughput in-memory computing for binary deep neural networks with monolithically integrated RRAM and 90-nm CMOS. IEEE Trans. Electron Devices **67**(10), 8 (2020)
68. Y. LeCun, Y. Bengio, G. Hinton, Deep learning. Nature **521**(7553), 436–444 (2015). https://doi.org/10.1038/nature14539
69. P. Chi et al., PRIME: a novel processing-in-memory architecture for neural network computation in ReRAM-Based main memory, in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)* (Seoul, South Korea, 2016), pp. 27–39. https://doi.org/10.1109/ISCA.2016.13
70. T. Gokmen, Y. Vlasov, Acceleration of Deep Neural Network Training with Resistive Cross-Point Devices: Design Considerations. Front. Neurosci. **10** (2016). https://doi.org/10.3389/fnins.2016.00333
71. S. Yu, Neuro-inspired computing with emerging nonvolatile memorys. Proc. IEEE **106**(2), 260–285 (2018). https://doi.org/10.1109/JPROC.2018.2790840
72. S. Ambrogio et al., Equivalent-accuracy accelerated neural-network training using analogue memory. Nature **558**(7708), 60–67 (2018). https://doi.org/10.1038/s41586-018-0180-5
73. G.W. Burr et al., Experimental demonstration and tolerancing of a large-scale neural network (165 000 Synapses) using phase-change memory as the synaptic weight element. IEEE Trans. Electron Devices **62**(11), 3498–3507 (2015). https://doi.org/10.1109/TED.2015.2439635
74. H. Kim et al., Zero-shifting technique for deep neural network training on resistive cross-point arrays. arXiv:1907.10228 [cs.ET] (2019)
75. S. Kim et al., Metal-oxide based, CMOS-compatible ECRAM for Deep Learning Accelerator, in *2019 IEEE International Electron Devices Meeting (IEDM)* (San Francisco, CA, USA, 2019), pp. 35.7.1–35.7.4. https://doi.org/10.1109/IEDM19573.2019.8993463
76. C.-C. Chang et al., Challenges and opportunities toward online training acceleration using rram-based hardware neural network," in *2017 IEEE International Electron Devices Meeting (IEDM)* (San Francisco, CA, USA, 2017), pp. 11.6.1–11.6.4
77. Z. Zhou et al., A new hardware implementation approach of BNNs based on nonlinear 2T2R synaptic cell, in *2018 IEEE International Electron Devices Meeting (IEDM)* (San Francisco, CA, 2018), pp. 20.7.1–20.7.4. https://doi.org/10.1109/IEDM.2018.8614642
78. D. Garbin et al., HfO_2-based OxRAM devices as synapses for convolutional neural networks. IEEE Trans. Electron Devices **62**(8), 2494–2501 (2015). https://doi.org/10.1109/TED.2015.2440102
79. I. Boybat et al., Neuromorphic computing with multi-memristive synapses. Nat. Commun. **9**(1), 2514 (2018). https://doi.org/10.1038/s41467-018-04933-y
80. V. Milo et al., A 4-Transistors/1-resistor hybrid synapse based on resistive switching memory (RRAM) capable of spike-rate-dependent plasticity (SRDP). IEEE Trans. VLSI Syst. **26**(12), 2806–2815 (2018). https://doi.org/10.1109/TVLSI.2018.2818978

81. Z. Wang et al., Toward a generalized Bienenstock-Cooper-Munro rule for spatiotemporal learning via triplet-STDP in memristive devices. Nat. Commun. **11**(1), 1510 (2020). https://doi.org/10.1038/s41467-020-15158-3
82. Z. Wang et al., Fully memristive neural networks for pattern classification with unsupervised learning. Nat. Electron. **1**(2), 137–145 (2018). https://doi.org/10.1038/s41928-018-0023-2
83. S. Liu, Y. Wang, M. Fardad, P.K. Varshney, A memristor-based optimization framework for artificial intelligence applications. IEEE Circuits Syst. Mag. **18**(1), 29–44 (2018). https://doi.org/10.1109/MCAS.2017.2785421
84. J. Hopfield, D. Tank, Computing with neural circuits: a model. Science **233**(4764), 625–633 (1986). https://doi.org/10.1126/science.3755256
85. J.J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons. Proc. Natl. Acad. Sci. **81**(10), 3088–3092 (1984). https://doi.org/10.1073/pnas.81.10.3088
86. S. B. Eryilmaz et al., Brain-like associative learning using a nanoscale non-volatile phase change synaptic device array. Front. Neurosci. **8** (2014). https://doi.org/10.3389/fnins.2014.00205
87. V. Milo, D. Ielmini, E. Chicca, Attractor networks and associative memories with STDP learning in RRAM synapses, in *2017 IEEE International Electron Devices Meeting (IEDM)* (San Francisco, CA, USA, 2017), pp. 11.2.1–11.2.4. https://doi.org/10.1109/IEDM.2017.8268369
88. G. Pedretti et al., A spiking recurrent neural network with phase change memory synapses for decision making, in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)* (Sevilla, 2020), pp. 1–5. https://doi.org/10.1109/ISCAS45731.2020.9180513
89. J.J. Hopfield, D.W. Tank, Neural computation of decisions in optimization problems. Biol. Cybern. **52**, 141–152 (1985)
90. A. Lucas, Ising formulations of many NP problems. Front. Phys. **2** (2014). https://doi.org/10.3389/fphy.2014.00005
91. S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by Simulated Annealing. Science **220**(4598), 671–680 (1983)
92. G.A. Fonseca Guerra, S.B. Furber, Using stochastic spiking neural networks on spinnaker to solve constraint satisfaction problems, Front. Neurosci. **11**, 714 (2017). https://doi.org/10.3389/fnins.2017.00714
93. H. Mostafa, L.K. Müller, G. Indiveri, An event-based architecture for solving constraint satisfaction problems. Nat Commun **6**(1), 8941 (2015). https://doi.org/10.1038/ncomms9941
94. T. Takemoto, M. Hayashi, C. Yoshimura, M. Yamaoka, 2.6 A 2 ×30k-Spin multichip scalable annealing processor based on a processing-in-memory approach for solving large-scale combinatorial optimization problems, in *2019 IEEE International Solid- State Circuits Conference—(ISSCC)* (San Francisco, CA, USA, 2019), pp. 52–54. https://doi.org/10.1109/ISSCC.2019.8662517
95. F.L. Traversa, C. Ramella, F. Bonani, M. Di Ventra, Memcomputing *NP*—complete problems in polynomial time using polynomial resources and collective states. Sci. Adv. **1**(6), e1500031 (2015). https://doi.org/10.1126/sciadv.1500031
96. V. S. Denchev et al., What is the computational value of finite-range tunneling?. Phys. Rev. X, **6**(3), 031015 (2016). https://doi.org/10.1103/PhysRevX.6.031015
97. S. Boixo et al., Evidence for quantum annealing with more than one hundred qubits. Nat. Phys. **10**(3), 218–224 (2014). https://doi.org/10.1038/nphys2900
98. P. L. McMahon et al., A fully programmable 100-spin coherent Ising machine with all-to-all connections, pp. 5
99. S. Kumar, J.P. Strachan, R.S. Williams, Chaotic dynamics in nanoscale NbO2 Mott memristors for analogue computing. Nature **548**(7667), 318–321 (2017). https://doi.org/10.1038/nature23307
100. G. Pedretti et al., 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 **6**(1), 89–97 (2020). https://doi.org/10.1109/JXCDC.2020.2992691

101. S. Kumar, R.S. Williams, Z. Wang, Third-order nanocircuit elements for neuromorphic engineering. Nature **585**(7826), 518–523 (2020). https://doi.org/10.1038/s41586-020-2735-5
102. M.R. Mahmoodi et al., An analog neuro-optimizer with adaptable annealing based on 64x64 0t1r crossbar circuit, in *2019 IEEE International Electron Devices Meeting (IEDM)* (San Francisco, CA, 2019), pp. 14.7.1–14.7.4. https://doi.org/10.1109/IEDM19573.2019.8993442
103. S. Ambrogio, S. Balatti, A. Cubeta, A. Calderoni, N. Ramaswamy, D. Ielmini, Statistical fluctuations in HfO_x resistive-switching memory: Part II—random telegraph noise. IEEE Trans. Electron Devices **61**(8), 2920–2927 (2014). https://doi.org/10.1109/TED.2014.2330202
104. S. Ambrogio, S. Balatti, V. McCaffrey, D.C. Wang, D. Ielmini, Noise-induced resistance broadening in resistive switching memory—Part I: intrinsic cell behavior. IEEE Trans. Electron Devices **62**(11), 3805–3811 (2015). https://doi.org/10.1109/TED.2015.2475598
105. L. Chen, K. Aihara, Chaotic simulated annealing by a neural network model with transient chaos. Neural Netw. **8**(6), 915–930 (1995). https://doi.org/10.1016/0893-6080(95)00033-V
106. M. Le Gallo et al., Mixed-precision in-memory computing. Nat Electron **1**(4), 246–253 (2018). https://doi.org/10.1038/s41928-018-0054-8
107. Z. Sun, G. Pedretti, A. Bricalli, D. Ielmini, One-step regression and classification with cross-point resistive memory arrays, Sci. Adv. **6**(5), eaay2378 (2020). https://doi.org/10.1126/sciadv.aay2378
108. Z. Sun, G. Pedretti, P. Mannocci, E. Ambrosi, A. Bricalli, D. Ielmini, Time complexity of in-memory solution of linear systems. IEEE Trans. Electron Devices **67**(7), 2945–2951 (2020). https://doi.org/10.1109/TED.2020.2992435
109. Z. Sun, G. Pedretti, D. Ielmini, Fast solution of linear systems with analog resistive switching memory (RRAM), in *2019 IEEE International Conference on Rebooting Computing (ICRC)* (San Mateo, CA, USA, 2019), pp. 1–5. https://doi.org/10.1109/ICRC.2019.8914709
110. J.R. Shewchuk, An Introduction to the Conjugate Gradient Method without the Agonizing Pain, School of Computer Science, Carnegie Mellon University, Pittsburgh, CMU-CS-94–125, (1994)
111. A. W. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for linear systems of equations. Phys. Rev. Lett.**103**(15), 150502 (2009). https://doi.org/10.1103/PhysRevLett.103.150502.
112. K. Bryan, T. Leise, The $25,000,000,000 eigenvector: the linear algebra behind google. SIAM Rev. **48**(3), 569–581 (2006). https://doi.org/10.1137/050623280
113. Z. Sun, G. Pedretti, E. Ambrosi, A. Bricalli, D. Ielmini, In-memory eigenvector computation in time O (1). Adv. Intell. Syst. 2000042 (2020). https://doi.org/10.1002/aisy.202000042
114. R. Penrose, A generalized inverse for matrices. Math. Proc. Camb. Phil. Soc. **51**(3), 406–413 (1955). https://doi.org/10.1017/S0305004100030401