Perspective

Memristors for the curious outsiders

Francesco Caravelli[1,†] and **Juan Pablo Carbajal** [2,]

[1] Theoretical Division and Center for Nonlinear Studies, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, USA; caravelli@lanl.gov

[2] HSR University of Applied Sciences Rapperswil, Institute for Energy Technology, 8640 Rapperswil, Switzerland; juan.pablo.carbajal@hsr.ch

† These authors contributed equally to this work.

Version September 28, 2018 submitted to Technologies

Abstract: We present both an overview and a perspective of recent experimental advances and proposed new approaches to performing computation using memristors. A memristor is a 2-port passive resistive component and in which its resistance evolves dynamically and depends on an internal parameter. This review is meant to guide nonpractitioners in the field of memristive circuits and their connection to machine learning and neural computation.

Keywords: memristors; neuromorphic computing; analog computation; machine learning

---

Contents

1. Introduction

The present review aims at providing a structured view over the many ares in which memristor technology is becoming popular. As many other hyped topics, there is a risk that most of the activity we see today will disipate into smoke in the coming years. Hence we have carefully slected a set of topics in which we have experience and we believe will remina relevant when memristors move out of the spotlight. After a general overview on memristors, we provde a historical overview of the topic. Next, we present an intuitive and a mathematical view on the topic, which we bleieve is needed to understand why anybody would consider alternative forms of computation. Thus, experts in the fields might find this article slow-paced.

A memristor is a resistive component in which the resistance depends on the history of the applied inputs: voltage or current. Different curves of the applied input elicit a different dnyamic response and final resistance of the memristor. In addition, if we remove the input after certain time, thne leave the component alone, and come back to use it, the device will resume its operation from a resistance very simoilar to the one in which we left it; that is, they act as non-volatiel memories. The interplay between the response behavior and the non-volatility of the device defines its usability either as a storage device or for more involved purposes such as neuromorphic computing. The article will gravitate around the fact that memristors are electronic components which behave similarly to human neuronal cells, and are an alternative building block for neuromorphic chips. Memristors, unlike other proposed components with neural behavior, can perform computation without requiring CMOS if not for readout reasons.

After the experimental realization of a memristor by Strukov *et al.*, which brought them to their current popularity, Leon Chua wrote a article titled "If it's pinched it's a memristor" [2]. The title refers to the fact that the voltage drop across the device is proportional to the current flowing through it, but that shows an hysteresis loop when controlled with a sinusoidal voltage. That is, the proportionality forces the hysteresis loop to be pinched when the current is zero. The claim in Chua is that a device which satisfies these two properties (plus a third one described in section 3), then the device must necessarily be a memristor. Chua also proved that such a device cannot be obtained from the combination of nonlinear capacitors, inductors and resistors [2–4]. It has been shown, however, that this property is a modeling deficiency for redox-based resistive switches [5].

Resistive switching was of interest even before the 2008 article. For instance the review of Waser and Aono shows that Titanium Dioxide had been in the radar for non-volatile memories for decades. Nevertheless one can identify a clear explosion of interest after the 2008 publication. In Table 1 we mentioned a list of established and new companies working to develop memristors technology using a variety of compounds. The table also contains two recent companies working on Resistive memory (ReRAM, RRAM) and Phase-Change-Material (PCM) type of materials.

Table 1. A non-exhaustive list of companies using oxide materials to produce resistive switches.

HP, Hynix	TiO_x
IBM	$SrTiO_x$
IMEC, Fujitsu, Samsung	NiO_x
SMIC	$CuSiO_x$
Sharp, TSMC	$TiOn$
NEC, Panasonic	TaO_x
Macromix	WO_x
Crossbar Inc.	ReRam
Qimonda, Ovonyx, Samsung, IBM, Intel, Hynix, KnowM	PCM type

Going into the many physical mechanisms that make a memristive device work does require a deep knowledge of material properties. For example, the histeretic behavior can be cause by Joule heating, as shown in an example taken from macroscopic granular materials [7]. Also, hysteresis is a phenomenon which is common in nanoscale devices and it can be derived using Kubo response theory [8]. Kubo response theory allows calculating the correction to the resistance induced by a time dependent perturbation: this is the formal framework to address resistive switching due to either electrical, thermal or mechanical stresses in the material. We can classify physical mechanisms that lead to memristive behavior in electronic components in four main types:

- Structural changes in the material (PCM like): in these material the current or the applied voltage trigger a phase transition between two different resistive states;
- Resistance changes due to thermal or electric excitation of electrons in the conduction bands (anionic): in these devices the resistive switching is due to either thermally or electrically induced hopping of the charge carriers in the conducting band;
- Electrochemical filament growth mechanism: in these materials the applied voltage induces filament growth from the anode to the catode of the device, thur reducing or increasing the resistance;
- Spin-torque: the resistance change is induced via the giant magnetoresistance switching due to a change in alignment of the spins at the interface between two differently polarized magnetic materials.

These mechanisms above are truly different in nature, and whilst not the only ones considered in the literature, are the most common ones. We provide a technical introduction to these mechanisms in Appendix A for completeness. We also suggest the reviews of the subject of resistive switching given in [9–12].

The primary application of memristors, as we will shown in this article, is towards neuromorphic computing. The word neuromorphic was coined by Carver Mead [13] to describe analog circuits which can mimic the behavior of biological neurons. In the past years the field has experienced an explosive development in terms of manufacturing neuromorphic novel chip architectures, which can reproduce the behavior of certain parts of the brain circuitry. Among the components used in neuromorphic circuits we consider the memristor whose behavior resembles the one of a certain type of neurons. The analogies between biological neuronal systems and electronic circuits are manifold: conservation of charge, thresholding behavior, integration to mention jus a few. For instance, diffusion of calcium across the membrane can be mapped to diffusion in electronic components, and a computational role associated to the circuit design. Also, it is known that the brain is power efficient: roughly twenty percent of an individual's energy is spent on brain activity, and this is roughly around 10 watts. Besides their role in biological neural models (section 6), we also discuss theoretical approaches to memristive circuits and their connection to machine learning (section 4 and 5).

The connection to machine learning is complemented with an overview on analog computation (section 5.1). Historically, the very first (known) computer was analog. It dates (approximately) 2100 years old, and it has been found in a shipwreck off the cost of the island of Antikythera at the beginning of the past century [14] (only recently it has been understood as a model of planet motion). Despite our roots in analog computation, our era is dominated by digital computers. Digital computers have been extremely useful at performing several important tasks in computation. We foresee that future computers will likely be a combination of analog (or quantum analog) and digital (or quantum digital) computing chips. At the classical level, several analog computing systems have been proposed. Insofar, most of the proposed architectures are based on biological systems, whose integration with CMOS can be challenging, and this is the reason why analog electronic components are seen as promising alternatives [15,16].

Digital computation has been dominated by the von Neumann architecture in combination with CMOS technology. Within this architecture we find two types of memory: Random Access Memory (RAM), a volatile and quick memory in which computation is performed, and non-volatile Hard-Disk (HD) for storage of long term information. The neat separation between memory (RAM and HD) and computing (the processors or CPU) is a key features of computation using von Neumann architecture. It requieres that the data is split into packets, transfer to the CPU where computation is performed, and then repeat until the full computation task has been completed. As far as our understading goes, this separation is not present in the brain, in which memeory and processing happen within the same units. Several proposed architectures that mimic this have been proposed, most of them based on memristors [17]. This type of computation is referred to as *memcomputing*.

This article is organized as follows. We first provide a historical introduction to memristors, as it is understood by the authors (sec. 2). We then provide the key ideas behind the technology with simple mathematical models (sec. 3). Albeit separated from the main text, we have provided an introduction to the main technology and physical principles underlying memristors in the Appendices A. We then focus our gaze on the description and use of memristors both for data storage (sec. 4) and data processing (sec. 5): the former is the current target for marketing the technology, the second is believed to be the main application of memristors in the long run. The similarity between memristors and neurons allows the implementation of machine learning on chip via Memristor/FPGA interfaces using crossbar arrays, we cover this topic in section 4.1. We have dedicated section 5.1 to overview the fundamental topic of analog computation, followed by a brief recapitulation of machine learning techniques that can be seamlessly used with memeristors (sec. 5.2- 5.3). We close the article with some remarks that should help in the discussion of the topics covered.

2. Brief history of memristors

The earliest known occurrence of a “memristive" behavior in circuit components is in the study, by Sir Humprey Davy’s, of arc (carbon) lamps [18], dated to the late 19th century. Another example, which was key to the discovery of the radio, is the coherer [19,20]. The coherer was invented by Branly [21] after the studies of Calzecchi-Onesti, and inspired Marconi’s radio receiver [22]. Branly’s coherer serves as perfect homemade memristor [7], as it simply requires either a (fine) metallic filling or some metallic beads, and it falls within the Physics discipline of electrical properties of granular media. Coherers are very sensitive to magnetic fields, and thus can act as a radio receiver and as we describe in the Appendix A, they do posses a typical hysteric behavior which is associated with memristive components. At the simplest level of abstraction, a memristor is a very peculiar type of nonlinear resistance with an hysteresis (e.g. they possess an internal memory). Currently, the discussion is focused not only on the application of this technology to novel computation and memory storage, but also on the fundamental role of memristors in circuit theory [23]. While this discussion is important, let us see first discuss why memristive behavior is not uncommon in analog computation. We thus use the analog of hydraulic computers.

Hydraulic computers were built in Russia during the 1930s before valves and semiconductor were invented. The hydraulic computers designed by the Russian scientist Vladimir Lukyanov were used to solve differential equations (Fig. 1); other models like MONIAC [24], would be later employed by the Bank of England to perform economic forecasts.

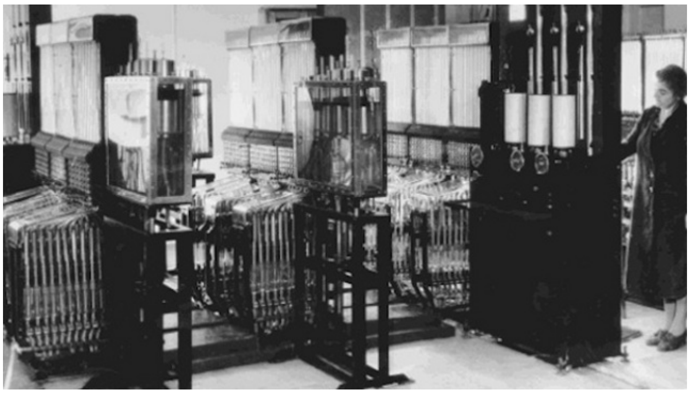


Figure 1. Lukyanov’s hydraulic computer or water integrator. Picture from Moscow Polytechnical Museum.

The idea behind these computers was to use hyrdraulics to solve differential equations. As Kirchhoff laws can be stated for any conservative field, we have that the pressure drop in a loop is equal to the actions of pumps in that loop. The conservation of mass is equivalent to conservation of charge, and to Kirchhoff's second law: in the steady state, the the mass of water entering a node must be equal to the mass flowing out. Ohm's law is equivalent to Poiseuille's law in the pipes: a porous material in the pipes, or a constriction, is equivalent to a resistance. A material that expands when wet, like a sponge, will increase the resistance to the flow as it absorbs the water. This is equivalent to a higher resistance which depends on the amount of water that flowed thought the system, i.e. memristor-like. The sponge, however, does not have a polarity, while memristor do depending on the physical mechanism which induces the switching. Other memristor-like systems can be built with other mechanical analogs [25,26], plants and potato tubes [27] or slime moulds [28,29] just to name a few.

The modern history of memristors is tied to the work of Leon Chua. The first time the word memristor (as an abbreviation for memory resistor) appeared was in the now celebrated Chua 1971 article [3]. During the 1960's, Leon Chua worked extensively on the mathematical foundations of nonlinear circuits. When he moved to Berkeley, where he currently is a Professor, he had already won several awards such as the IEEE Kirchhoff Award. The definition of memristor was made clearer in a second paper with his student at the time, Sung Mo Kang [4]. The second work was an important generalization of the notion of memristor and is the one we used in the present paper. Chua and Kang introduced the notion of 'memristive device': a resistance which depends on a state variable (or variables), which is sufficient to describe the physical state (resistance) of the device at any time. The component defined by Chua, and then Chua and Kang, is a resistance whose value depends on some internal parameter, which in turn has to evolve dynamically according either to current and voltage. Implicitly, one needs to also define the relationship between the resistance and the state variable, which characterizes the device resistance. In the analogy above with hydraulic computers, the state variable represents by the density of holes in the sponge as a function of time, while the resistance is the amount of traversable area. The 1976 and the 1971 papers were mostly mathematical and formal, without any connection to the physical properties of a real device. The 1976 paper also introduced the fact that if one controls the device with a sinusoidal voltage, then one should observe Lissajous figures in the current-voltage (I-V) diagram of the device. It also established that any electronic component that displayed a pinched hysteresis in the I-V diagram has to be a memristor. The eager reader will find more details on the devices in the rest of the paper.

Over 30 years after the work by Chua and Kang, the field of memristors was basically non-existent. Strukov, Williams and collaborators, working at Hewlett-Packard Labs were studying on oxide materials and resistive switching. The physics of resistive switching was known since the early '70s, with the introduction of Phase-Change Materials. The physical origin of resistive switching was well studied, albeit not fully understood [6,30,31]. The idea that memristors could be experimentally realized became popular with the article of Strukov *et al.* at HP Labs. Before, the word memristor was confined to the theoretical predictions put forward by Chua, and by Chua and Sung Mo Kang, and the subject remained basically a mathematical exercise for almost 40 years, until the HP group realized that Titanium Dioxide components could fit the definition of memristors.

3. Mathematical models of memristors

In this section, we provide a general mathematical description of memristive systems followed by the details of the memristor model with linear memroy dynamics proposed by Strukov *et al.*. The latter is one of the simples memristors models that captures the behaviors relevant for technological applications of memristive systems.

The fundamental 2-port passive electric components are the resistors, capacitors and inductors. These components couple the voltage difference v applied between their ports with the current i flowing through, with the following differential relationships:

$$\begin{aligned} \text{Resistor:} \quad & \mathrm{d}v = R\,\mathrm{d}i; \\ \text{Capacitor:} \quad & \mathrm{d}q = C\,\mathrm{d}v, \quad \mathrm{d}q = i\,\mathrm{d}t; \\ \text{Inductor:} \quad & \mathrm{d}\Phi = L\,\mathrm{d}v, \quad \mathrm{d}\Phi = v\,\mathrm{d}t. \end{aligned} \tag{1}$$

These relations introduce the resistance R, the capacitance C, and the inductance L. The memristor was initially and abstractly formulated as the relationship:

$$\mathrm{d}\Phi = M\,\mathrm{d}q, \tag{2}$$

which, in order to be invertible, must have M defined either in terms of Φ, or q, or it must be a constant. The latter case coincides with the resistor, while if M is defined in terms of Φ, then we have a voltage-controlled memristor; if it is defined in terms of q it is a current-controlled memristor. The units of the coupling M are Ohms, as can be seen by the units of the quantities it relates. This latter fact and its definition, justify the qualifier of *nonlinear resistor*. However, accoridng to Chua [2], only nonlinear resistor fullfiling three specific requirements are classified as memristors. The three requirements are stated as properties of the current-voltage ($I - V$) diagram that is observed experimentally when the device is controlled with a sinuisoidal voltage at a certain frequency. As mentioned in the introduction, the first is a pinched hysteris loop, i.e. $V = 0 \to I = 0$. The second is that as the frequency of the driving signal increases then correlation coefficient of the $I - V$ curve decreases. The third requirement is the recovery of pure resistance (no hysteresis) for high frequencies. If we plot $I(t)$ vs $V(t)$, the diagram is the celebrated pinched hysteresis loop which only memristor satisfy. An example is shown in Fig. 2.

Many systems, not necessarily electric ones, can fulfill all these properties. Physical systems with memristor-like behavior are denominated memristive systems, and are described by the following nonautonomous (input-driven) dynamical system,

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} &= F(\boldsymbol{x}, u), \\ y &= H(\boldsymbol{x}, u)u, \end{aligned} \tag{3}$$

where $\boldsymbol{x}$ is a vector of internal states, y a measured scalar quantity and u a scalar magnitude controlling the system. The pair (y, u) is usually voltage-current or current-voltage, hence the second equation is simply Ohm's law for the voltage-current relationship. In the first case (current controlled system) the function $H(\boldsymbol{x}, u)$ is the called memristance, and it is called memductance in the second case (voltage controlled). The function $H(\boldsymbol{x}, u)$ is assumed to be continuous, bounded and of constant sign, leading to the zero-crossing property or pinched hysteresis: $u = 0 \Rightarrow y = 0$. The states $\boldsymbol{x}$ can be many physical states other than charge, e.g. the internal temperature of a granular material. The minimum number of internal states on which $H(\boldsymbol{x}, u)$ depends is called the *order* of the memristive system.

Among the class of memristive systems the model proposed by Strukov *et al.* is appealing due to its simplicity. This model, shown in eqns. (4)-(5) captures the basic properties of memristors that are relevant for understanding of the device and for its technological applications. Eqns. (4)-(5) model the behavior of a current-controlled Titanium Dioxide memristor [1], also known as the HP-memristor:

$$\frac{\mathrm{d}w(t)}{\mathrm{d}t} = \alpha w(t) - \frac{1}{\beta} I(t), \tag{4}$$

$$\frac{V(t)}{I(t)} = R\left(w(t)\right) := R_{\mathrm{on}}\left[1 - w(t)\right] + R_{\mathrm{off}} w(t) \tag{5}$$

$$w(0) = w_0 \to R(w_0) = R_0, \quad 0 \leq w(t) \leq 1, \tag{6}$$

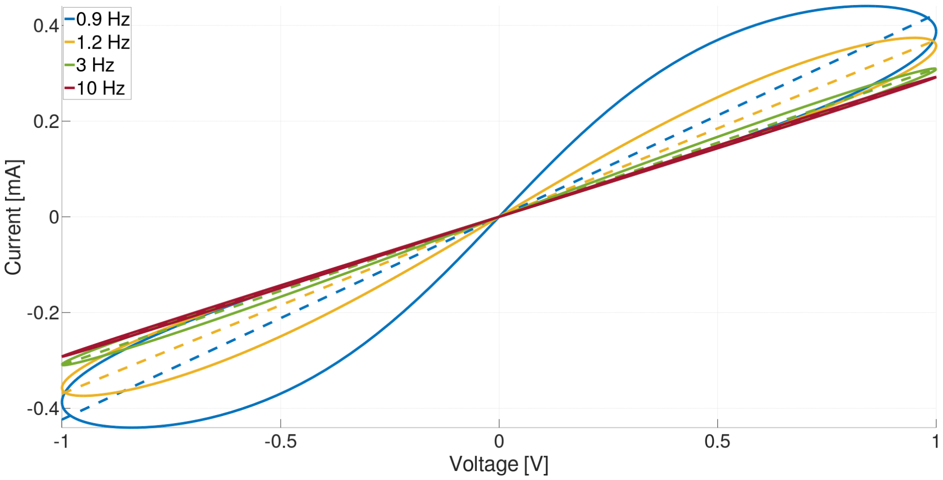


Figure 2. Pinched hysteresis loop of a HP-memristor. The parameters of the model are $\alpha = 0$, $\beta = 0.3\,\text{mA}$, $R_{\text{on}} = 1\,\text{k}\Omega$ and $R_{\text{off}} = 6\,\text{k}\Omega$. The driving voltage is $V(t) = \sin(2\pi f t)$ with f taking several values. We see that for increasing frequencies the hysteresis is reduced, eventually converging to a line (resistor). The loop slope (dashed lines) decreases with increasing frequency.

where $I(t)$ is the current flowing in the device at time t. The quantities $R_{\text{on}} \leq R_{\text{off}}$ in the parametrization of the resistance R in terms of the adimensional variable $w(t)$ (called *memory*), define the two limiting resistances. The parameters α and β, adimensional and dimensional respectively, define the dynamic properties of the memristor. The parameter $\alpha \geq 0$, sometimes called *volatility*, indicates how fast the resistance drifts towards R_{off} in the absence of currents. Since the resistance function depends on a single state w the model is of first order. Albeit this model has several drawbacks when it comes to its connection to the physical behavior of ion migration in the conductor [8,23,25,32–35], it will be the reference for most derivations and discussions in this article.

We begin with the case with $\alpha = 0$, i.e. a non-volatile memristor. We solve the system of equations above using a driving voltage $V(t)$ which is such that for all times $0 < w(t) < 1$, i.e. the memristor should not saturate at any time, e.g. with a zero mean small amplitude $V(t)$ [36]:

$$R\left(w(t)\right)\frac{d}{dt}w(t) = \frac{d}{dt}\left((R_{\text{off}} - R_{\text{on}})\frac{1}{2}w^2(t) + R_{\text{on}}w(t)\right) = -V(t), \tag{7}$$

from which we derive, if we define $\xi = \frac{R_{\text{off}}-R_{\text{on}}}{R_{\text{on}}}$ and integrate over time:

$$\left(\frac{\xi}{2}w^2(t) + w(t)\right) = \left(\frac{\xi}{2}w^2(t_0) + w(t_0)\right) + \int_0^t V(\tau)\mathrm{d}\tau,, \tag{8}$$

which solution is given by

$$w(t) = \frac{\sqrt{2\left(\frac{\xi}{2}w^2(t_0) + w(t_0) \ + \ \int_0^t V(\tau)\mathrm{d}\tau\right)\xi + 1} - 1}{\xi}. \tag{9}$$

This equation fulfills the three properties defining a memristor: it has the zero crossing property, the loop tilts to the right when we increase the frequency of the driving signal, and the loop becomes a line for high frequencies. To see the latter consider $V(t) = V_0 \cos(\omega t)$, as the frequency $\omega \to \infty$. For this voltage the integral in eq. (8) goes to 0, and $w(t) \to w(t_0)$. This implies a *constant* resistance, i.e. the memristor becomes a resistor for high frequencies.

In this model, we have that $w(t) \sim q(t)$, where $q(t)$ is the charge in the conductor. For a titanium dioxide thin film, it was shown in [1] that

$$\beta^{-1} \sim \frac{\mu_e R_{\text{on}}}{D}, \tag{10}$$

and thus

$$R(q) \approx R_{\text{off}} \left(1 - \frac{\mu_e R_{\text{on}}}{D^2} q\right). \tag{11}$$

Where μ_e is the electron mobility in the film, R_{off} is the resistance of the undoped material (for instance titanium oxides, $\sim 100\,\Omega$), and D is a characteristic length of the film. From the micrometer to the nanometer the value of β grows by a factor of 10^6, which is why the memristance in this material is a nanoscale effect.

Memristors are also referred to as resistive switches, because if $\frac{R_{\text{off}}}{R_{\text{on}}} \gg 1$, and β is small enough, the memristor can be quickly driven from one state to the other via a current inversion. The operation is often referred to a SET or $RESET$ in the technical literature, depending on the operation one is interested in, and it allows the use of memristors as bits.

In the volatile case, i.e. $\alpha > 0$, the memristor does not retain its memory state. When a volatile memristor is not activated via an external forcing, the memristor drifts to its insulating phase, $R(t = \infty) = R_{\text{off}}$. That is, if $I(t) = 0$, then the internal memory is simply given by $w(t) = w_0 e^{\alpha t}$. This behavior was first experimentally observed in [37]. Volatility is discussed again in Section 5.4.

The numerical model of the memristor allows us to study their theoretical capabilities as well as simulating large networks of these devices. From the point of view of numerical simulations, the dynamics at the boundaries of w need to be stable, or alternatively, w be constrained in $[0, 1]$. In general the latter does not scale well in terms of runtime, and for this reason it is often useful to bound the internal states of the memristor model via the introduction of window functions [1,38–43]. Since we are not aware of systematic validations of windowing functions with real devices, we believe they should be considered tricks useful for large simulations. We mention here common windowing techniques based on polynomials, for a extensive review we refer the reader to [41]. Polynomial windowing functions are inspired by non-linear dopants drift can be introduced via the so-called Jogeklar window function $F(x)$ [39,43], such that $F(1) = F(0) = 0$, which generalized the evolution of $w(t)$ as

$$\frac{\mathrm{d}w(t)}{\mathrm{d}t} = \alpha w(t) - F(w)\frac{1}{\beta}I(t), \tag{12}$$

where the window function can take the form $F(w) = 1 - (2w - 1)^{2p}$ with p positive integer. Depending on the type of memristor model different window functions might provide a better approximation of the nonlinear effects near the boundaries of the memory. For an overview of the various mechanisms that can be involved we refer the reader to the Appendix A.

4. Memristors for storage

A memristor can be used as a digital memory of at least one bit. The simplest way to achieve this, is to use the memristor as a switch. If the memristor is non-volatile we can set its memristance to the value R_{on} or R_{off} using a voltage or current pulse, and associate these resistances with the state of a bit. We stress that non-volatility is essential for memory applications, because a volatile memristor, i.e. one that drifts back to a high resistance state autonomously, would require a permanent source of current/voltage to keep it in the low resistance state. Volatility will, in general, render memristive memory worthless in terms of energy consumption. This is one of the reasons why volatility is engineered out of the devices meant for storage applications, e.g. ReRAM.

In order to illustrate the mechanism of flipping a bit, consider the non-volatile memristor model described by eqn. (4) (i.e. $\alpha = 0$) connected to a constant voltage source V_{write}. Solving the differential equation gives:

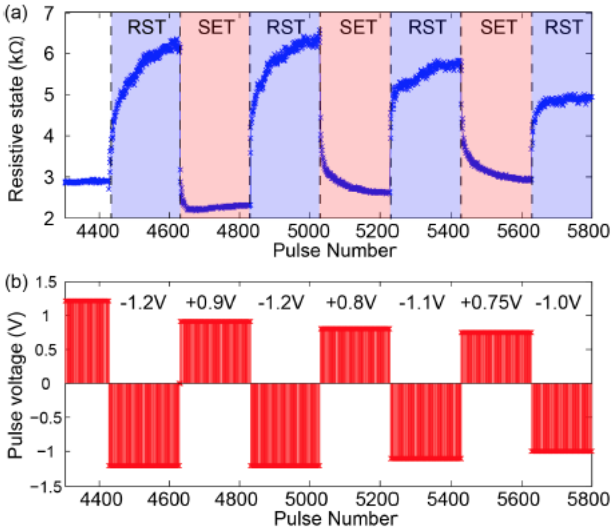


Figure 3. Reset and set for TiO_2, from [44], with pulse within the hundred of microseconds.

$$w(t) = \sqrt{a + bV_{\text{write}} t}, \tag{13}$$

where the coefficients a, b depend on the parameters, but not on the input voltage. Thus, by controlling the sign of the input voltage V_{write} we can switch the resistance from R_{off} to R_{on} and viceversa (the flip of a bit). The switching happens within a characteristic time τ:

$$\sqrt{a + bV_{\text{write}}\tau} = 1 \to \tau \leq \frac{1}{bV_{\text{write}}}. \tag{14}$$

Hence, to flip the bit we need to apply V_{write} for at least this amount of time. To read the bit, we apply a voltage $V_{\text{read}} \ll V_{\text{write}}$ (and optionally for period of time shorter than τ) and compute the resistance from current measurements.

Eqn. (13) applies only to the idealized memristor described by the model in eqns. (4)-(5). This model might not be valid for real devices which will show a different dynamic response to input voltages or currents. However, the idea of controlling and reading a memristor bit with pulsed inputs remains the same. Fig. 3 shows the response of a real TiO_2 memristor to write voltages (SET and RESET).

The fact that we can write and read the state via signal pulses allows for advantageous scaling of power consumption and bit density [see 45, for details]. As we discuss below, it is possible to use crossbar arrays with memristors to increase the density of components. The density of components in crossbar arrays scales as $\frac{1}{4\ell^2}$, where ℓ is set by the length scale of optical litography (a few nanometers). The reported state of the art as we write this article is roughly 7 GB/mm^2, and with writing currents of 0.1 nA [46].

Another challenge for storage based applications, besides volatility, is device variability, e.g. the variation of properties like the switching time τ for devices built under similar conditions. Current research in oxides focuses on these variability aspects, how to standardize the production of memristors, and the optimization of properties like the switching and retention time, and the durability of each singular device. The status of the technology for memory storage for different type of devices is presented in Tab. 2.

Table 2. Characteristics of various storage devices (for details see [47]). The table compares to memristor technology to competitors like PCM and more standard devices based on STT, DRam, Flash and HD.

	Memristor	PCM	STT-RAM	DRAM	Flash	HD
Chip area per bit (F^2)	4	10	14-64	6-8	4-8	n/a
Energy per bit (pJ)	0.1-3	10^{1-2}	0.1-1	10^0	10^3	10^4
Read time (ns)	<10	20-70	10-30	10-50	10^{4-5}	10^4
Write time (ns)	20-30	10^{1-2}	10^{1-2}	10^1	10^5	10^6
Retention (yrs)	10	10	10^{-1}	10^{-5}	10	10
Cycles endurance	10^{12}	10^{7-8}	10^{15}	10^{17}	$105 - 8$	10^{15}
3D capability	yes	no	no	no	yes	n/a

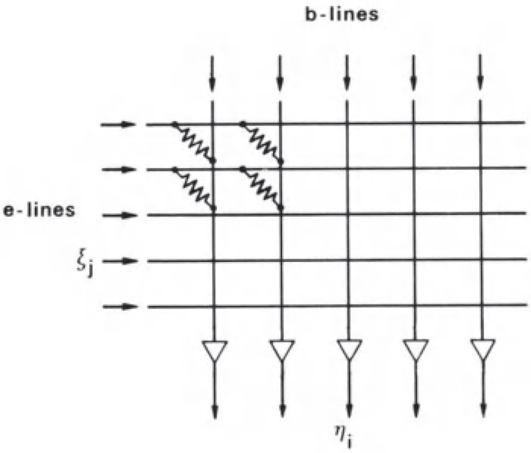


Figure 4. Learning matrix, introduced by Steinbuch [48] and reproduced from [49].

4.1. Crossbar arrays

In this section we briefly review the crossbar array architecture used in memristor based storage and its application in artificial neural networks.

Crossbar arrays are based on the architecture depicted in Fig. 4. The figure shows an array composed of horizontal (*e-lines*) and vertical (*b-lines*) lines that are initially electrically isolated from each other. A 2-port component, e.g. a memristor, is connected across each pair of vertical and horizontal lines.

To use crossbar arrays, a voltage ξ_j is applied to the j-th e-line, and an another voltage η_i to the i-th b-line. A memristance $w_{j,i}$, placed across j-th e-line and the the i-th b-line, is controlled by the voltage $\xi_j - \eta_i$. This arrangement allows for simple indexing of the memristances, and is the mechanism behind a Content-Addressable-Memory (CAM) which is used in crossbar arrays. The idea of this construction dates back to Steinbuch's "Die Lernmatrix" [48,50].

Crossbar arrays can be used for matrix-vector multiplication using the voltages $\{\xi_j\}$ as inputs and the voltages $\{\eta_i\}$ as outputs. For a resistance independent of the input voltage or current, the relation between them is given by $\vec{\eta} = A\vec{\xi}$, where the elements of the matrix A are:

$$A_{ij} = \frac{M_{ij}^{-1}}{\tilde{R}_i^{-1} + \sum_{s=1} M_{is}^{-1}}. \tag{15}$$

$\tilde{R}_i$ are the resistances on the output b-lines $\vec{\eta}$, and M_{ij} is the memristance of the memristor between the i-th b-line and j-th e-line. An algorithm for setting the (mem)resistances given a matrix can be found in [51]. To apply this method with memristors, the voltages differences need to be small/short,

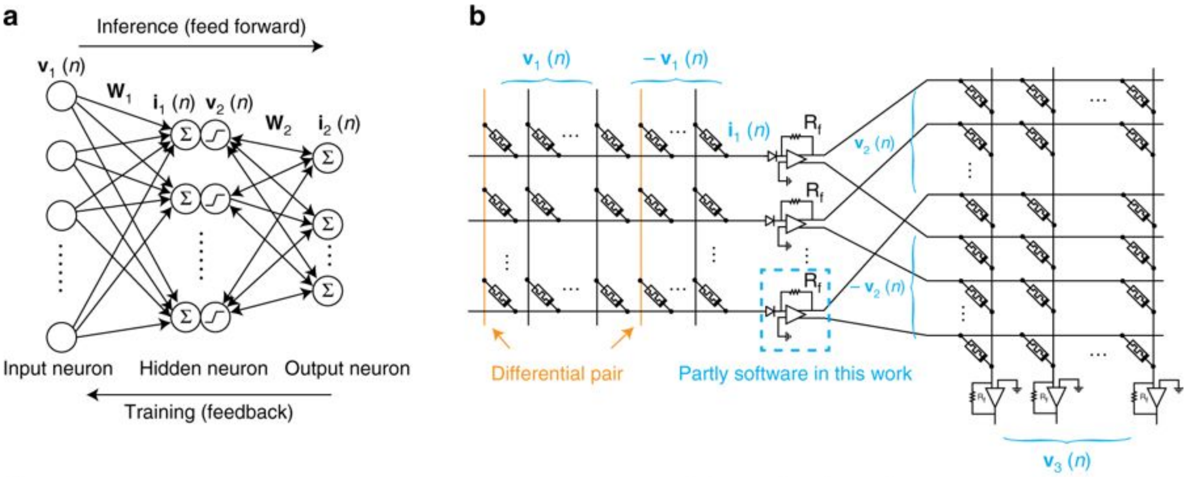


Figure 5. Memristors used for synaptic weights in ANNs. Reproduced from Fig. 2 of [54]

to avoid changing their memristance, or all memristors need to be in one of their limiting states R_{on} or R_{off}.

The component density of crossbar arrays can be increased by stacking [52,53]. The stacking of L crossbar layers on top of each other scales the density of components by the factor L, i.e. a theoretical scaling of $\frac{L}{4\ell^2}$. In multilayered arrays memristors are controlled using the corresponding horizontal and vertical lines of each layer.

Memristive crossbar arrays can be used to encode the synaptic weights of feed-forward or recurrent artificial neural networks (ANNs) [54]. In ANNs the input to neurons in a given layer (post-synaptic) is computed as the multiplication of the outputs of neurons in the previous layer (pre-synaptic) and the matrix of synaptic weights connecting the two layers (Fig. 5). Then, the multiplication is carried over using the multiplication algorithm previously described. The output of pre-synaptic neurons is encoded in the voltages of e-lines of the crossbar array, and the input to post-synaptic neurons is decoded from the currents on the b-lines. Applications go beyond this direct implementation of the multiplication algorithm. For example in [54] the synaptic weights are encoded as the difference of the conductance between two memristors. Similar ideas have been exploited to design other computational models based on stateful logic [55] and differential pair synapses (called “kT-RAM") [56].

The potential benefits of utilizing memristor based ANNs are speed and energy efficiency. The computation and storage use the same location in the network, and analog inputs are directly feed to the neurons. This minimizes the reading-writing costs incurred by the conventional von Neumann architecture, and the energy losses of analog-to-digital conversion.

4.2. Synaptic plasticity

Synaptic plasticity can be broadly defined as the modification of the synaptic conductance as a function of the activity of the neurons connected to it. This definition rules out autonomous plasticity, which is the slow synaptic conductance decay of inactive synapses (volatility). Autonomous plasticity is fundamental for data processing applications and will be considered in section 5.

Non-autonomous synaptic plasticity can be classified in several types, e.g. spike rate-dependent plasticity, spike timing-dependent plasticity, short-term plasticity, long-term plasticity, etc. In the study of synaptic plasticity of the human cortex, Hebbian or Anti-Hebbian (related to the simultaneous firing of two neurons) [49,57,58] are often underpinning the learning mechanism. Our aim here is not to describe the types of plasticity, their biological underpinnings, or how difficult is to isolate each class in biological and experimental systems, but to illustrate how memristors, used as memories, can implement the change in weights based on pre- and/or post-synaptic activity.

For an overview of the different types of synaptic plasticity and references to further reading, we refer the reader to the book chapter by La Barbera and Alibart [59].

Synaptic plasticity is modeled by choosing the plasticity inducing variables $x \in X$ (e.g. relative arrival times of spikes, relative neural activity, etc.) and a mapping from these to the change of the synaptic weight

$$f_X : X \to W_\Delta \subset \mathbb{R}. \tag{16}$$

This mapping is based on biological models, or simplified adaptation mechanisms. For example in Spike Timing-Dependent Plasticity (STDP), the inducing variable is the relative timing of two or more activity spikes in the connected neurons. Plasticity is then defined with a mapping $f_{\text{STDP}} : T^{n-1} \to W_\Delta$, taking the relative timing of n spikes $\in T^{n-1}$ (typically 2 or 3) to a weight change $\in W_\Delta$ (usually represented as relative change). In Spike Rate-Dependent Plasticity, we replace the timing domain with a relative rate domain.

To implement plasticity in memristors as synaptic weights, we need a writing voltage that represents the synaptic change. That is, we need a further mapping

$$f_V : W_\Delta \to V, \tag{17}$$

where V is the set of valid writing voltages. The composed mapping $f_V \circ f_X : X \to V$ gives the final implementation of synaptic plasticity. The mapping f_V depends on all the characteristics of the technology used, e.g. the physical mechanisms of memristance (see Appendix A), the neural network architecture (e.g. crossbar array), the controlling electronics, etc. Obtaining the function f_V in the mapping above is the main challenge in synaptic plasticity applications, and thus requires considerable effort. A survey of complete and partial implementations of synaptic plasticity in nanoscale devices is summarized in [59, sec. 4.1]. For example [60] uses STDP to implement unsupervised learning with ReRAM synapses.

STDP receives a lot of attention in the neuromorphic field as exemplified by the latter reference and the review of Serrano-Gotarredona *et al.* in which we base the following setences. The reader interested in hardware impelentations of STDP shold consult that resource. STDP is among the most developed memristors implementation of in-silico plasticity, it can be implemented in very large and very dense arrays of memristors without global synchronization, and learning occurs on-line in a single integrated phase (as opposed to off-line learning). The impact of the dynamical model of the memristor has been studied in the implementation of STDP, and the learning rules can be adapted to the different behaviors. As in most application of memristors as non-volatile storage of (synaptic) weights, it suffers from the intrinsic varibality of the units, which more general neuroimorphic circuits are able to exploit [62].

5. Memristors for data processing

Device variability (sec. 4) and volatility (sec. 4.2) were mentioned as current challenges for most applications based on memristive memories. This is in contrast with biological systems, which are not built in clean rooms and it is hard to think of evolution exploiting ideal systems as a reference for design. Biological systems perform despite noise, nonlinearity, variability, lack of robustness, and volatility. Whether these ingredients hinder the performance of biological systems or are actually a building block for it, it is still unknown. Sometimes they are avoided, not because it would have an undesired effect in practice, but simply because its effect cannot be easily modeled or studied (e.g. we have but a few tools to deal with nonlinear systems intrinsically, beyond the iteration of linear methods). Hence, there is no reasons to believe that eliminating naturally occurring properties is the path to success in achieving artificial systems that perform comparably to biological ones.

This section overviews some applications that embrace device volatility [63], nonlinear transients, and variability [64, sec. 7.5], to implement learning methods with memristors. To put these methods within an unifiying framwework, we first review the concept of analog computation, and discuss its relation to the physical substrate on which it is implemented.

5.1. Analog computation

To pin down the concept of analog computation we compare analog and digital computers, assuming that the latter is familiar to most readers. The principal distinction between analog and digital computers is that digital operates on discrete representations in discrete steps, while analog operates on continuous representations, i.e. discrete vs. continuous computation [refer to 65, for a complete discussion and historical overview] [66,67].

In all kinds of computation the abstract mathematical structure of the problem and the algorithm are instantiated in the states and processes of the physical system used as computer [68]. For example, in the current digital computer, computation is carried on using strings of symbols that bare no physical relationship to the quantities of interest; while in analog computers the latter are proportional to the physical quantities used in the computation. Figure 6 illustrates the relation between representation of the problem in the designers mind and instantiation in the computers states.

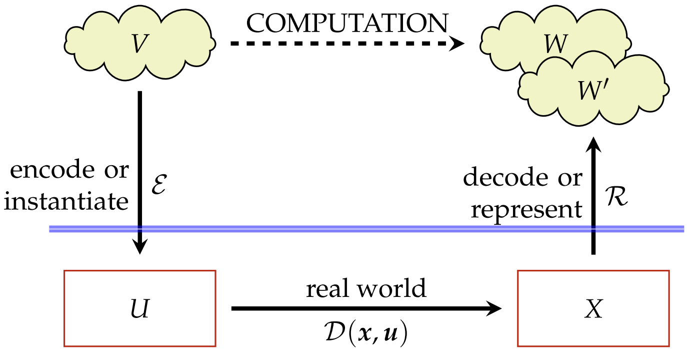


Figure 6. Computing with dynamical systems adapted from [68]. Conceptual depiction of the sets and transformations involved in a typical computation using a dynamical system $\mathcal{D}$. In this case, the computation is defined by its action on the input-output set V, W. The inputs $\boldsymbol{u}(t) \in U$ to $\mathcal{D}$ are encoded or instantiated versions of V through the transformation $\mathcal{E}$. The output of the dynamical system $\boldsymbol{x}(t) \in X$ is decoded or represented back into the set W' via the readout transformation $\mathcal{R}$. When W and W' are similar, the composed transformation $\mathcal{R} \circ \mathcal{D} \circ \mathcal{E}$ is a proxy for the sought computation.

We can decompose computation in three stages: i) encoding, ii) processing, iii) decoding. To program a computer, we need to design a suitable process to encode the input and to decode the result of the computation. The three stages require an advanced understanding of the behavior of the physical substrate, how it reacts to inputs and how it transforms its states. This is true if the computer is meant to implement an universal model of computation, or if it is specialized hardware optimized to solve a particular subclass of problems.

The encoding and decoding maps relate the states of the computer to our understanding of the problem, and their definition is a recurring challenge in the design of computers. We mentioned this difficulty when building memristive synaptic arrays with plasticity (sec. 4.2); data representation in biological systems is still an open research field, and natural systems tend to smear out our pristine categorizations of encodings. There is yet another difficulty to attend when defining encoding and decoding maps. To avoid confounding, the class of maps needs to be restricted, because ill-defined maps (e.g. encryption) will complicate computation, while too sophisticated maps could render the contribution of the computing device negligible. The former will deteriorate the computing performance, while the latter is just bad design. In other words, the problem needs to be specified using the "language" of the computing device. Using the wrong language increases the difficulty of the problem, and consequently decreases performance. To understand the language of the device we need the equivalent of Shannon's analysis of the differential analyzer [69]. The encoding-decoding pair is also linked to the "natural basis of computation" [70] of a device, which refers to the description of the device behavior suited for the computation purposes.

The success of digital computers is in part given by the efficiency to instantiate and process an universal model of computation able to solve all kinds of computation problems, as conjectured by the Church-Turing(-Deutch) thesis. This is achieved by a precise control on each step of computation and the way the computer transforms its states. Another advantage of current digital computers over analog prototypes is the very high precision they provide for the instantiation and solution of a problem's quantities. This high precision, however, is unnecessary in many engineering applications, in which the input data are known to only a few digits, the equations may be approximate or derived from experiments, and the results are not sensitive to round-off errors [see 71, for an overview]. Therefore, research into specialized hardware (analog of digital) is a worthy activity.

Since analog computers escape the frame of relevance of the Church-Turing thesis, it has been argued that they can be more powerful than digital computers[65, sec. "Analog Computation and the Turing Limit"]. Besides this hypothetical benefit, it is worth exploring the efficiency of analog computers to solve subclasses of problems, i.e. specialized analog hardware, and to understand their pervasiveness in natural systems, perhaps linked to the precision attained by systems built from many imprecise cheap modules. This is not a simple task, since many of these analog computers outsource some of the process control present in digital computers to the natural dynamics of the physical substrate, elevating the bar on the level of understanding required to build and program them.

As an example of an analog computer let us consider an hydrostatic polynomial root finder [72]. Consider the following polynomial equation:

$$\sum_{i=1}^{n} x^i c_i + c_0 = 0, \tag{18}$$

The aim is to find a real value x for which the equality holds, i.e. we want one root of the polynomial $p(x) = \sum_{i=1}^{n} x^i c_i + c_0$. For the sake of this illustration, lets consider the case $n = 2$. In Fig. 7 we show the design of an hydrostatic root solver. In one side of a pivoting lever, a shape is set to represent each term in the derivative of the polynomial (e.g. flat for the derivative of $c_1 x$, linear for the derivative of $c_2 x^2$). In the other side of the lever we set a weight for the constant term. When the device is submerged into water, the depth at the device equilibrates is a solution of the polynomial equation.

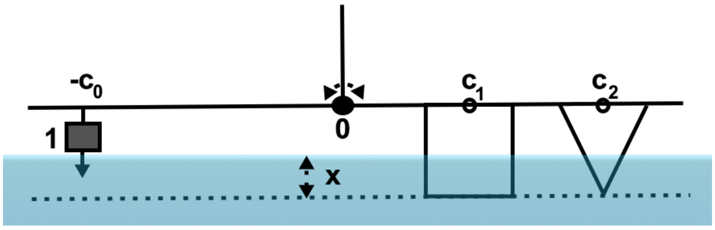


Figure 7. The hydrostatic root solver. Shapes encode the polynomial coefficients. A weight encodes the constant term. The depth at which torque is zero, is a solution of the polynomial equation.

The example above illustrate the major role played by the understanding of the physical device, and how it allows us to encode and solve the specific problem we are interested in. As mentioned before this is common to all sort of computers; we proceed to mention a few. The electronic digital computer exploits the behavior of transistors to encode the symbols of the computational model (essentially Boolean logic) it uses for computation. Quantum annealers, considered for efficiently solving Mixed Integer quadratic Programming [73, sec. 2.7], use the spins of a physical system and the computation of a quantum Ising model to solve the optimization [74]. DNA computing exploits the self-assembly and preferential attachment of DNA strands to encode tiling systems with Wang tiles [75,76]. Slime molds [29] computation exploits the behavior of these protists to implement distributed optimization, e.g. shortest path, using the position of nutrients on the Petri dish to encode problems, and chemotatic behavior of the mold to solve them; the solution is read out from the distirbution of mold cells in the Petri dish. Ant colony optimization [77] is inspired by the behavior of their natural counterpart and uses digital models of pheromone dynamics for computation (see also stigmergy [78]).

In the subsequent sections we describe in some technical details algorithms meant to realize computers using memristive systems.

5.2. Generalized linear regression, Extreme Learning Machines, and Reservoir computing

Arguably the most used method to relate a set of values X (inputs) to another set of values $\boldsymbol{y}$ (outputs) via an algebraic relation is linear regression: $\boldsymbol{y} = X^\top \boldsymbol{\beta}$. However, in many realistic situations we believe that the relation between these sets of values is unlikely to be linear. Hence a nonlinear counterpart is needed. The simplest way to generalized a scalar linear regression model between inputs and outputs, is to apply a nonlinear transform to the inputs to obtain a new linear model $\boldsymbol{y} = G(X)^\top \boldsymbol{\gamma}$, where only the coefficient vector $\boldsymbol{\gamma}$ (or matrix when the output is not scalar) is learned from the data. This nonlinear method is linear regression in a space generated by nonlinear transformations of the input (see kernel methods [79,80]). In neuromorphic computation the transform is commonly given a particular structure by choosing a set of N_g nonlinear functions and apply it to each input vector:

$$G(X)^\top_{ij} = g_j(\boldsymbol{x}(i)), \quad i = 1, \ldots, N \; j = 1, \ldots, N_g. \tag{19}$$

This method is known as Extreme Learning Machines (ELM) [81] and has been implemented in hardware exploiting the variability of fabricated devices to generate the nonlinear transformations [82,83].

This formulation resembles two other mathematical methods that are worth mentioning: generalized Fourier series and generalized linear methods. The relation with generalized Fourier series is made evident when the samples have a natural ordering (e.g. not i.i.d. time samples $i \leftrightarrow t_i$), then we can write the regression model as

$$y(t) = \sum_{k=1}^{N_g} \gamma_k g_k(\boldsymbol{x}(t)), \tag{20}$$

which has the structure of a truncated generalized Fourier series (but not all the ingredients).

The resemblance with generalized linear models is made evident by considering the function $g(\boldsymbol{x}) = \ell^{-1}(\boldsymbol{x}^\top \boldsymbol{\eta})$ (ℓ is the link function) and $N_g = 1$; we obtain:

$$y_i = \gamma_1 \ell^{-1}(\boldsymbol{x}(i)^\top \boldsymbol{\eta}). \tag{21}$$

However generalized linear models require that we learn the vector of coefficients $\boldsymbol{\eta}$ from the data, rendering the problem nonlinear. This breaks the analogy with ELM in which the $\boldsymbol{\eta}$ vector should be fixed a priori. The analogy is somehow rescued if we are given a distribution $p(\boldsymbol{\eta})$ for the $\boldsymbol{\eta}$ vector encoding prior knowledge or beliefs about the solution to the generalized linear model. In this case we can build an ELM with $N_g \gg 1$

$$y_i = \sum_{k=1}^{N_g} \gamma_k \ell^{-1}(\boldsymbol{x}(i)^\top \boldsymbol{\eta}_k) \approx \int_H \ell^{-1}(\boldsymbol{x}(i)^\top \boldsymbol{\eta}) p(\boldsymbol{\eta}) \mathrm{d}\boldsymbol{\eta}, \tag{22}$$

where the set of $\{\boldsymbol{\eta}_k\}$ coefficient vectors are sampled from the prior distribution, expecting that model averaging [84] will approximate the generalized linear model.

Summarizing, ELM uses a linear combination of a predefined dictionary of functions to approximate input-output relations. The next step of generalization is Reservoir Computing

(RC) [85–87], in which the N_g functions are the solution of a differential or difference equations using the data as inputs, e.g.

$$\begin{aligned} \boldsymbol{u}(t) &= \mathcal{E}(\boldsymbol{x})(t) \ \in \mathbb{R}^{\dim \boldsymbol{u}}, & (23)\\ \mathcal{D}_{\boldsymbol{\lambda}}\boldsymbol{q} &= B\boldsymbol{u}(t),\ B \in \mathbb{R}^{\dim \boldsymbol{q} \times \dim \boldsymbol{u}}, & (24)\\ \boldsymbol{g} &= H\boldsymbol{q},\ H \in \mathbb{R}^{N_g \times \dim \boldsymbol{q}}, & (25)\\ y(t) &= \boldsymbol{\gamma}^\top \boldsymbol{g} = \sum_{k=1}^{N_g} \gamma_k g_k(t, \boldsymbol{u}(t), \boldsymbol{\lambda}) & (26) \end{aligned}$$

where the differential or difference equations are denoted with $\mathcal{D}_{\boldsymbol{\lambda}}$ (operator notation) with $\boldsymbol{\lambda}$ a vector of parameters that includes physical properties and the boundary (initial) conditions, and $\dim \boldsymbol{q} \geq N_g$. The connectivity matrices B and H are typically random, the latter mixes the $\dim \boldsymbol{q}$ states to obtain N_g signals (these could also be nonlinear mappings). As explained in section 5.1, the input data is encoded by the transformation $\mathcal{E}$ (or $B \circ \mathcal{E}$) to properly drive the system. This encoding, the operator $\mathcal{D}_{\boldsymbol{\lambda}}$, and the connectivity matrices are defined a priori, and only the coefficients $\boldsymbol{\gamma}$ combining the $\boldsymbol{g}$ functions are learned from the data, as in ELM. These coefficient (or $\boldsymbol{\gamma}^\top H$) define the readout transformation $\mathcal{R}$ (see Fig. 6).

The generalization proposed by RC is made obvious with the choice of arguments for the component functions $\{g_k\}$ in eq. (26): they can have i) an intrinsic dependence on time, e.g. autonomous behavior of the dynamical system; ii) they depend on the inputs, and iii) they depend on the properties of the dynamical system. Property ii) says that these functions are not fixed as in ELM, they are shaped by the data signal. Stated like this, the problem of implementing computation with reservoirs is strongly related to a nonlinear control problem.

RC allows for machine learning applications using natural or random dynamical systems, as opposed to carefully engineered ones. The only strong requirement is that we are able to stimulate states of the system independently with signals encoding the input data, a classical example is the perceptron in a water bucket [88]. Hence, RC implementations using memristive networks has received considerable attention (software [see 63, and references therein] and hardware [89]).

The case of memristor based RC using the HP-memristor, eqns. (4)-(5), is fairly well understood: a differential equation to simulate the propagation of signals across the circuit and the interaction between memristors has been derived in [90] (see also eqn. (34)). In those equations the role of the circuit topology is extremely important in the collective dynamics of the circuit and in processing the input information. When working with RC and memristors, it is important to prevent the saturation of all devices, since a saturated memristor becomes a linear resistor that only scales the input.

5.3. Neural engineering framework

The Neural engineering Framework [64] exploits our current understanding of neural data processing to implement desired computations. It confines linear models to a particular class of basis functions, inspired by biologically plausible neuron models but not restricted to them. Here we describe this framework in the case of function representation, the structure of the framework is analogous to other representation instances (scalar, vector, etc.). The reader is referred to the original work [64] for a comprehensive description.

The framework entails the characterization of an admissible set of functions that can be represented by a population of N neurons. In particular, the functions and their domain need to be bounded: $f : (x_{\min}, x_{\max}) \to (f_{\min}, f_{\max})$. These functions are then encoded by a population of neurons with a predefined set of encoders of the form:

$$a_i(f(x)) = G_i\left(I_i(f(x))\right), \tag{27}$$

$$I_i(f(x)) = \alpha_i \langle f(x)\tilde{\phi}_i(x)\rangle + I_i^{\text{bias}}, \tag{28}$$

where I_i represents the total input current to the i-th neuron soma. The functions a_i and G_i are the tuning curves observed by neurophysiologists and the response function of the i-th neuron, respectively. G_i is a biologically inspired model of the firing rate, e.g. integrate-and-fire (LIF) neuron. The encoding generators $\{\tilde{\phi}_i\}$ (analogous to preferred directions) are defined a priori, and $\langle f(x)\tilde{\phi}_i(x)\rangle$ is a functional defined on the space of functions to be represented, e.g. in the original description this is the mean over x. These encoders convert the function $f(x)$ into firing rates (or actual spike counts for the case of temporal encoding).

The encoding is matched with a corresponding decoding procedure that brings firing rates (spike counts) back to the function space. The decoder takes the generic form:

$$\hat{f}(x) = \sum_i^N a_i(f(x))\phi_i(x), \tag{29}$$

where $\phi(x)$ are the unknowns of the framework. That is, given some input x and neural population's firing rates (spike counts) we can build functions of the input using the decoders $\{\phi_i\}$. In the original formulation the decoders are obtained via minimization of least square errors (with regularization in the case of noisy encodings), but other methods could be used, e.g. optimal L_2 dictionaries [91].

The framework defined in eqns. (27)-(29) can realize arithmetic on functions of the input as well as nonlinear transformations. It has also been use to represent linear time-independent systems with neuromorphic hardware [92].

NEF, RC, and ELM use the same form of decoding: the output is the scalar product of a input-independent vector with a input-dependent one. The input-dependent vector is given by intrinsic properties of the computing device, it is the results of internal mechanisms. The decoders are learned from data, and different decoders implement different computations (on the same input-dependent vectors). However, RC and ELM learn a finite dimensional vector $\gamma \in \mathbb{R}^{N_g}$ while NEF, in the functional form shown here, needs to learn N infinite dimensional decoders. The latter is mildly relaxed if the input domain is discretized with N_x points, rendering the functional decoders N_x-dimensional vectors. In General it is expected that $N_g \ll N_x N$, that is the degrees of freedom of NEF, is higher that the one of RC and ELM. Hence NEF has the risk to transfer all the computation to the decoders making the contribution of the neural population marginal (or even an obstacle). Extra regularity assumption on the decoders $\{\phi_i(x)\}$ (eq. (29)) are needed to match decoders effective capacity to the capacity of the dynamic responses $\{g_i(x)\}$ (eq. (26)).

Memristor networks can be used to implement NEF, by implementing the response functions G_i of neural models. The G_i functions corresponding to LIF neuron model is

$$G[I(z)] = \begin{cases} \frac{1}{\tau_0 - \tau_{\text{RC}}\log\left(1 - \frac{I_F}{I(z)}\right)} & I(z) > I_F, \\ 0 & \text{otherwise} \end{cases} \tag{30}$$

where $I(z)$ is given by eqn. (28). And a similar functional form can be achieved by a non-volatile memritor with parasitic capacitance, as shown in eq. (33) (see next section). However other response functions, which might be easier to implement with memristors, can be used with NEF. Other aspects of neural networks, such as critical behavior [93–95] can be observed in networks of memristors [96], although a theoretical understanding of their collective behavior is still poor for both systems [97–99].

5.4. Volatility: autonomous plasticity

Volatility is a key feature when processing information with memristors (in contrast to memory applications). RC needs volatility to avoid trivial linear input-output mappings and NEF requires it to model the forgetting behavior of neurons. There are many physical processes that can lead to a memristive volatile device, hence the source of volatility should be discussed in the context of a given technology. In what follows we show how volatility, can be linked with a capacitance in parallel (parasitic) to a non-volatile device. Consider an ideal series memristor-capacitor circuit [43] feed with a controlled current. The memristor is modelled with eqns. (4), with $\alpha = 0$, Kirchhoff's voltage law for this circuit gives:

$$R(w)\overbrace{\frac{\mathrm{d}q}{\mathrm{d}t}}^{I(t)} = -\frac{1}{C}\overbrace{q(t)}^{\int I(t)\mathrm{d}t} \to \frac{\mathrm{d}q}{\mathrm{d}t}(t) = -\frac{q(t)}{R(q(t))C}, \tag{31}$$

$$R(q(t)) = R_{\text{on}}\left(1 + \frac{q(t)}{\beta}\right) + R_{\text{off}}, \tag{32}$$

from which we obtain a limiting solution for $t \gg 1$

$$q(t) = \frac{\beta}{\xi} W\left(\frac{\xi}{\beta} e^{-\frac{t}{R_{\text{on}}C}} e^{-\frac{c_1}{\beta R_{\text{on}}}}\right), \tag{33}$$

where W is the product-log (Lambert) function [100], c_1 is an integration constant, and as before $\xi = \frac{R_{\text{off}} - R_{\text{on}}}{R_{\text{on}}}$. Eqn. (33) shows that for large times the system has a typical exponential RC decay, which is shown in Fig. 8 (left). This behavior is observed in experiments [101] where after an external stimuli an exponential-like decay is observed (see Fig. 8 (right)).

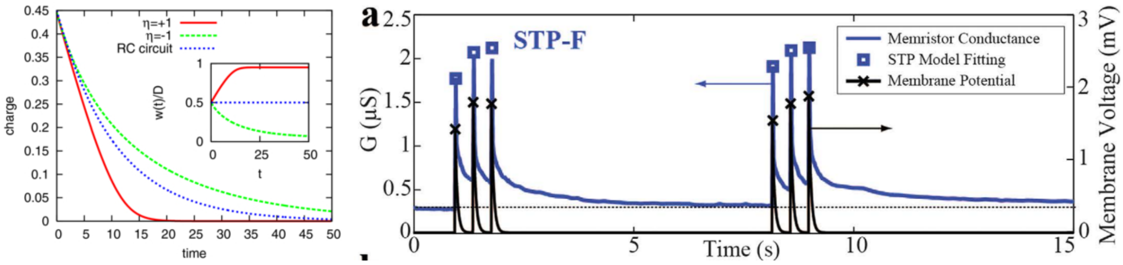


Figure 8. Left: theoretical profile of eqn. (33), taken from [43]. Right: Experimental profile of short term plasticity with TiO_2, taken from [101].

The limit of a normal RC circuit can be obtained from the above in the limit $\beta \to \infty$, in which $\lim_{\beta\to\infty} \beta W(\beta^{-1}G(t)) = G(t)$. We thus see that the response of a memristor-capacitor and a RC circuit differs, with the first obtaining a longer retention than the first. Also, the Lambert W has several properties of the logarithm function, and thus the response is similar to the one suggested in eqn. (33). In the case $\alpha \neq 0$ for the model we described here, one has that $w(t) = e^{t\alpha}\left(c_0 + \frac{1}{\beta}\int_0^t d\tau e^{-t\tau} I(t)\right)$. Thus, the correspondence between the parameter w and the charge is not exact. In addition, the parameter α constant is only an approximation. The conductance of memristive devices decays when there is no input, and the rate of decay depends on the state of the memristor. This is compatible with a state-dependent parameter α, rather than a constant [see Fig. 1 of 37]. A survey of recent hardware designs for temporal memory is provided in [102].

5.5. Basis of computation

As mentioned before, to design computers with memristors we need to understand and harness their natural computational power. This is no simple task in general, but for a networks of memristors linear in an internal parameter and current controlled, we can write down the differential equation describing the evolution of the memory states and obeying Kirchhoff voltage and current laws [90]:

$$\frac{d\vec{w}}{dt}(t) = \alpha\vec{w}(t) - \frac{1}{\beta}(I + \frac{R_{\text{off}} - R_{\text{on}}}{R_{\text{on}}}\Omega W)^{-1}\Omega\vec{S}(t), \tag{34}$$

where α and β are the parameters in eqns. (4)-(5), Ω is a projector operator which depends on the circuit topology, $W_{ij}(t) = \delta_{ij}w_i(t)$ and $\vec{S}(t)$ is vector of applied voltages. For arbitrary memristor components the generalization of eqn. (34) is not known. In the approximation $R_{\text{off}} = pR_{\text{on}}$, with p of order one, the equation above can be recast in the form of a (constrained) gradient descent [103], which is reminiscent of the fact that the dynamics of a purely memristive circuit has an approximate Lyapunov function [104,105]. In the simplified setting of purely memristive circuit without any other components it can be shown that these circuits execute Quadratically Unconstrained Binary Optimization [106]. This idea is in general not recent, and it can be traced back to Hopfield [107–109] for continuous neurons.

6. Memristive galore!

6.1. Memristive computing

In this section we review some works developing computation algorithms networks of memristors and external contorl hardware based in crossbar arrays and FPGA.

In [110,111] it has been shown that memristive circuits can be used to solve mazes: connecting the entrance and exit of a maze, the memristive circuit as in Fig. 9 will re-organize (when controlled in DC) to allow the majority of the current to flow along the shortest path. Although this phenomenon already occurs with regular resistances, it is enhanced with memristors. Memristors outside the shortest path go to their OFF state (high resistance) and the current difference (the contrast) to the active shortest path is augmented. This example shows that the wiring between memristors and the asymptotic resistance value are deeply connected, and is reminiscent of the ant-colony optimization algorithms [77], molecular computation [112], and other cellular automata models [113].

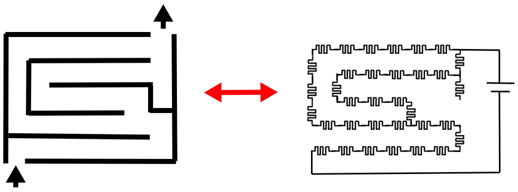


Figure 9. The maze-memristor mapping suggested in [110].

Ideas along these lines can be pushed further in order to explore memristors as complex adaptive systems [114] able to self-organize with the guidance of the circuit topology and the control of external voltages. Using eqn. (34) it is possible to obtain approximate solutions, for instance, of the combinatorial Markowitz problem [105]. Hybrid CMOS/Memristive circuits can, in principle, tackle harder problems via a combination of external control and self-organization [115].

In the literature several models of memristor-based architectures have been proposed. Several of these proposals are based on the attractor dynamics of volatile dissipative electronics and inspired by biological systems. For instance, a general theory of computing architecture based on memory components (memcapacitors, meminductors and memristors [116]) has been introduced recently in [117], called Universal Memory Machines (UMM), and shown to be Turing complete. Similarly, an architecture based on memristors which includes Anti-Hebbian and Hebbian learning (AHaH) has been proposed in [56,118] for the purpose of building logic gates and for machine learning. In both cases of UMM and AHaH the solutions of the problems under scrutiny are theoretically embedded in the attractors structure of the proposed dynamical systems, and have not been tested experimentally.

One way to show that a memristive system composed of memristors is a universal computing architecture is to break the system modularly into logic gates based on memristor, and show that the set of obtained gates is universal (which includes NOT and at least one of an AND or OR gates, as in DeMorgan's law [119]). Turing completeness follows from an infinite random access memory (the infinite tape). Experimentally, it has been shown that it is possible to build logic gates with memristors (we mention for instance [120]). An improvement upon this basic idea is to build input-output agnostic logic gates using memristors. Any port of an agnositc gate can be used as input or output, and the remaining states of the gate will converge to the states of a logic gate, regardless of whether the binary variable is at the output or at the input of the gate. For example, if the output of an agnostic AND gate is set to TRUE, the input variables will rearrange to be both TRUE; but if the output is FALSE, the inputs will re-arrange such as to contain at least one FALSE. These are called Self-Organizing Logic Gates (SOLG) and it is suggested to use these to solve the max-SAT problem [121,122] [see also 123, and references therein].

The two example above show that memristors can be used both for analog computation, as in the case of shortest path problems, or to reproduce and extend the properties of digital logic gates.

6.2. Natural memristive information processing systems: Squids, Plants, and Amoebae

In recent years, and with the participation of L. Chua, there have been several reports re-interpreting models of natural information processing systems (neural networks, chemical signaling, etc.) in terms of memristors units. Herein in we mention three examples: giant axon of squids [124], electrical networks of some plants [125], and Amoeba adaptation.

Giant squids are model organisms big enough that they can be analyzed in detail at the singular cell level, and for which we posses a mechanistic model of the dynamics of their axons: the Hudgkin-Huxley model. The model describes the voltage at the interface between synapses and dendrites, which is regulated by the flow of calcium and potassium. The electrical circuit associated with this model is shown in Fig. 10. It entails the introduction of a nonlinear variable resistor for the calcium channel, and a linear variable resistance for the potassium channel and a capacitance [126]. The equations of this model, when put in memristive form, are given by:

$$i_{\mathrm{K}} = \overbrace{g_{\mathrm{K}} w_1^4}^{R_{\mathrm{K}}^{-1}} V_{\mathrm{K}}, \tag{35}$$

$$i_{\mathrm{Na}} = \overbrace{g_{Na} w_2^3 w_3}^{R_{\mathrm{Na}}^{-1}} V_{\mathrm{Na}}, \tag{36}$$

$$\frac{\mathrm{d}w_1}{\mathrm{d}t} = (\mathrm{K}_1\, V_{\mathrm{K}} + \mathrm{K}_2) \left[e^{\mathrm{K}_1\, V_{\mathrm{K}} + \mathrm{K}_2} - 1\right]^{-1} (1 - w_1), \tag{37}$$

$$\frac{\mathrm{d}w_2}{\mathrm{d}t} = (\mathrm{Na}_1\, V_{\mathrm{Na}} + \mathrm{Na}_2) \left(e^{\mathrm{Na}_1\, V_{\mathrm{Na}} + \mathrm{Na}_2} - 1\right)^{-1} (1 - w_2) + \mathrm{Na}_3\, e^{\mathrm{Na}_4\, V_{\mathrm{Na}} + \mathrm{Na}_5}\, w_2, \tag{38}$$

$$\frac{\mathrm{d}w_3}{\mathrm{d}t} = \mathrm{Na}_6\, e^{\mathrm{Na}_7\, V_{\mathrm{Na}} + \mathrm{Na}_8}\, (1 - w_3) - \left(e^{\mathrm{Na}_1\, V_{\mathrm{Na}} + \mathrm{Na}_9} + 1\right)^{-1} w_3, \tag{39}$$

where we see that a first order (R_{K}) and second order (R_{Na}) memristors are involved. The parameters $\{\mathrm{K}_i\}$ and $\{\mathrm{Na}_i\}$ characterize the dynamics of the channels [see 124, for details].

The model above is a fit of the observed voltage data for the giant axon, and is useful in the analytical study of brain cell dynamics. The proposed model allows the interpretation of synapses as circuits composed of rather nonlinear and non-ideal memristors. That is, that memristors can be central in providing an alternative interpretation of a established model and further the understanding of biological neural information processing.

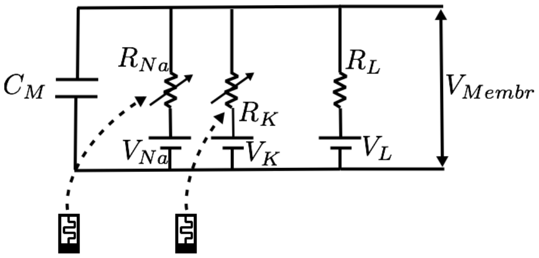


Figure 10. The Hodgkin-Huxley model, with the variable resistances of the Sodium and Potassium channels interpreted as memristors.

In [125] three types of memristors models are developed and compared with the responses of some plants to periodic electrical stimulation. The authors observed that in the studied plants the pinched hysteresis loop did not collapsed into a line for very high frequencies, as required for ideal memristors. To recover this non-ideal behavior a parasitic resistor-capacitor pair was added in parallel to the ideal memristor model. The general solution for their models is:

$$i_m(t) = \frac{e^{\beta t} V(t)}{\beta R_o \int_0^t h(V(x)) e^{\beta x} \mathrm{d}x + A}, \tag{40}$$

$$I = i_m + i_{RC}, \tag{41}$$

where β is a parameter related to the time constant of the memristor, $V(t)$ is the driving periodic voltage, and $R_o h(V)$ is the memristance of a voltage controlled memristor. Depending on the model of the memristor considered h and the constant A take different forms. The total current I is the observed magnitude, and i_{RC} is the current through a series resistor-capacitor circuit in parallel with the memristor. The study hints that memristive behavior is intrinsic to plants electrical signaling and that plant physiology could be better understood if memristors are considered as "essential model building blocks".

A memristive model of amoeba adaptation was introduced in the form of the simple circuit shown in Fig. 11 [127,128], and the model is simple enough that we can report it here. Albeit the original article points to the concept of amoeba learning, we believe it is more appropriate to be addressed as a model of amoeba adaptation. The memristor considered is a voltage controlled memristor introduced in [129]:

$$\frac{\mathrm{d}M}{\mathrm{d}t} = f(V_M)\left(\theta(V_M)\theta(M - R_1) + \theta(-V_M)\theta(R_2 - M)\right), \tag{42}$$

$$f(V) = \frac{\beta - \alpha}{2}\left(|V + V_T| - |V - V_T|\right) - \beta V, \tag{43}$$

where $\theta(\cdot)$ is the Heaviside step function.

Because the inductance and the resistor in the circuit are in series, the same current I flows through them. The capacitor and the memristor are in parallel, hance their voltage drop are equal: $V_C = V_M$. The conservation of voltage on the mesh implies $V_C + V_L + V_R = V(t)$. We have $V_R = RI$ and $V_L = LI$. The memristance $M(t)$ affects the voltage drop on the capacitor,

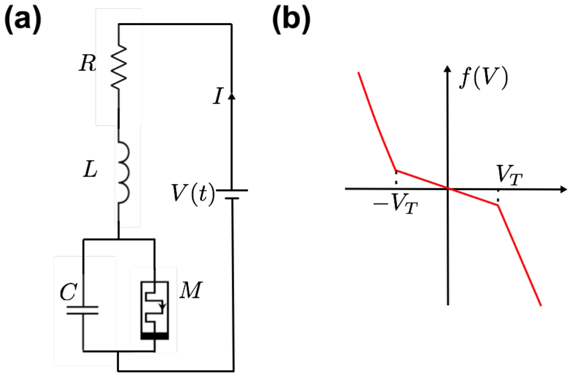


Figure 11. The amoeba memristive learning model of [127]. (a) The circuit with capacitance C and memristor M in parallel (with resistance $R(t)$), and in series to a resistance R and an inductance L. (b) The function $f(V)$ for the memristor response in voltage.

$$CV_C + \frac{V_C}{M(t)} = I. \tag{44}$$

We thus obtain the three coupled differential equations:

$$\frac{\mathrm{d}I}{\mathrm{d}t} = -\frac{R}{L}I + \frac{V - V_C}{L}, \tag{45}$$

$$\frac{\mathrm{d}V_C}{\mathrm{d}t} = -\frac{1}{MC}V_C + \frac{I}{C}, \tag{46}$$

$$\frac{\mathrm{d}M}{\mathrm{d}t} = f(V_M)\left(\theta(V_M)\theta(M - R_1) + \theta(-V_M)\theta(R_2 - R)\right). \tag{47}$$

The stationary state requires that all time derivatives are zero. The circuit is adaptive to new stimuli. For instance, in Fig. 12 we see the response of the system to new inputs, and new stationary states are obtained. Albeit amoeba's adaptation is not as developed as in higher mammals, more general models entailing Pavloviav "conditioning" have also been proposed in the literature using memristors [130, 131].

6.3. Self-organized critically in networks of memristors

We now consider the interaction between a high number of components with memory. A common feature of large systems of interacting units with thresholds or discontinuous dynamics is critical behavior. For example, self-organized criticality (SOC) is evinced when a dynamical system self-tunes into a state for which a qualitative change in the systems' behavior is imminent (e.g. a bifurcation). These critical states are characterized by power law cross-correlation functions. The current characterization of the sufficient ingredients for SOC, however, is phenomenological: a system of interacting particles or agents in which thresholds are present and whose dynamics is dominated by their mutual interaction. One of the main motivations of SOC is the explanation of power spectra of the functional form $P \sim \omega^\alpha$, with $-1 < \alpha < -3$, in physical systems and in nature [132]. The typical example is for instance the Gutenberg-Richter law of earthquakes, whose distribution of magnitude is Richter's law [133,134]. SOC has been suggested to be the underlying mechanism in the observed critical behavior of the brain [94]. In this case, neurons can be interpreted as thresholding functions (logical gates) and it is thus tempting to interpret the criticality of the brain as a SOC phenomenon.

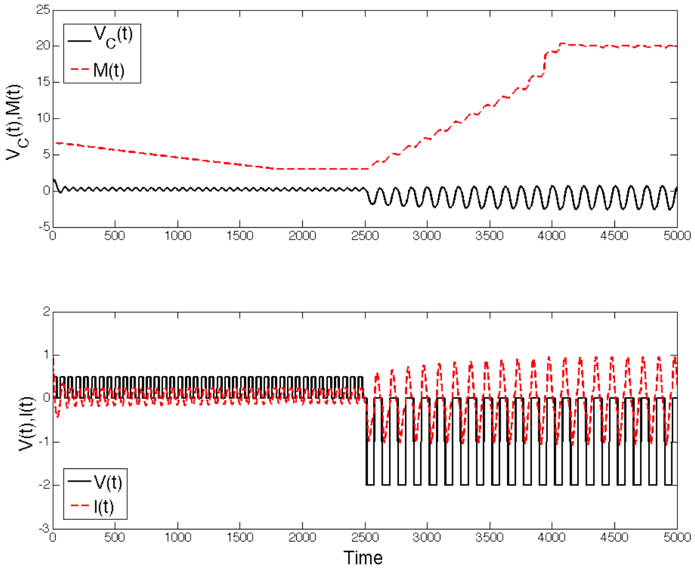


Figure 12. Simulation of eqn. (47) and adaptation of the circuit to different stimuli. We consider the parameters $\beta = 100$, $\alpha = 0.1$, $R_1 = 3$, $R_2 = 20$, $C = 1$, $R = 1$, $L = 2$, $V_t = 2.5$ as in [127]. We stimulate the circuit with a square input $V(t) = 0.5$ and frequency $\omega = 10$ and then with reduce by a half the frequency, with $V(t) = -2$. We see that the circuits "adapts" to the new stimulus after a transient. Initial conditions were $I_0 = 1$, $V_c^0 = 1, R(0) = 7$ and used an Euler integration scheme with step $dt = 0.1$.

SOC can be produced using large networks of memristors: atomic switch networks are dominated by the interaction due to Kirchhoff laws [96], and thus the observed criticality seems intuitively (power law distribution of the power spectra, for instance) to be connected to a SOC-type [99] phenomenon because of the rather nonlinear and threshold-like behavior of memristors. It is however easy to observe that power law distributions in power spectra can be obtained in a rather simple way as follows [90]. Consider a system of linearly interacting memristors, whose linearized dynamics close to a fixed point obtained with DC voltage stimulation (i.e. saturated memristors: resistors) is written as:

$$\frac{d}{dt}\vec{w}(t) = A\vec{w}(t). \tag{48}$$

The matrix A is a non-trivial combination of voltage sources and projectors on the subspace of the circuit's graph [103]. We divide the spectrum of A in positive and negative eigenvalues, and the distribution $\rho_+(A)$ and $\rho_-(A)$. Since $0 \leq w_i(t) \leq 1$, we look at the average relaxation $\langle w(t)\rangle = \sum_i \frac{w_i(t)}{N}$, which can be written as

$$\langle w(t)\rangle = \frac{1}{N}\sum_i \left(\sum_j (e^{At})_{ij} w_j^0\right) = \frac{1}{N}\text{trace}\left(e^{At}W_0\right) = \frac{1}{N}\text{trace}\left(e^{\lambda^+ t}\tilde{W}^0 + e^{\lambda^- t}\tilde{W}^0\right), \tag{49}$$

The positive part of the spectrum will push memristors to the $w = 1$ state, while the negative part to the $w = 0$ state. We can write the trace on each positive and negative state as:

$$\frac{1}{N}\text{trace}(e^{-\lambda_i^- t}\tilde{w}_i) = \int d\lambda \rho^-(\lambda) e^{-\lambda t}\langle w_i^0\rangle = \frac{1}{2}\int d\lambda \rho^-(\lambda) e^{-\lambda t}, \tag{50}$$

if the memristors are randomly initialized. As it turns out, if $\rho^-(\lambda)$ is power law distributed, then $\langle w(t)\rangle \approx t^\gamma$. From this, we observe that the power spectrum distribution is of the form $P(\omega) \approx \omega^{-(1-\gamma)}$, from which a "critical state" is obtained. It was shown in [90] that if the circuit is random enough, numerical simulations produce $\gamma \approx -1$. A similar argument can be obtained for $\rho^{+}(\gamma)$ with the transformation $w \to 1 - w$. This result, when using netowkrs of HP-memristors, is in line with what is experimentally observed in [96].

The mechanism above is not classified as SOC, and it is only required a certain matrix A to obtain it. This implies that unless thresholding is present, the criticality observed is not self-tuned, but it might be due to the complex interconnections. This is not the case if, however, the memristors themselves have voltage induced switching [99]. In this case, the criticality is due to the a SOC-like phenomenon which had been already observed in random fuse networks [135], which is described by a percolation transition.

6.4. Memristors and CMOS

The idea of using variable resistances in order to implement learning algorithms is not new. As we have seen, crossbar arrays were introduced already as early as 1961 [48]. The idea of using instead variables resistances precedes the paper of Steinbuch by one year, and was introduced by Widrow [136] in order to implement the Adaline algorithm explained below. The "memistor", a name extremely similar to the one of "memristor" introduced ten years later, was a variable resistance controlled in current. This implies that, differently from a memristor, a memistor is a 3-port device that is current-controlled externally [137]. This factor limits the ability to package a huge number of synapses. For (modern) machine learning applications, however, it is necessary to implement learning rules with a large number of neurons and synapses. As we have seen, memristors are the equivalent component for a synapse [138]. The neuron is instead the biological equivalent of a N-port logic gate (threshold function). For more general applications a crossbar-array like packing is desirable. There are many ways of using memristors for applications in neural networks. To get a sense of the type of circuits involved. For instance, the circuit proposed in Fig. 13 provides a simple circuit which has a linear output neuron controlled by a resistance R, without threshold. The threshold can be however easily introduce via a Zener diode. In order to understand why memristors do not have necessarily an advantage over digital implementations of neural networks, we follow the argument of [139] to understand the energy efficiency. Using Landauer theory, the energy of a digital gate is $E_{gate} \approx -2\log(p_{err})kT$, where p_{err} is the probability of an error. For the analog implementation above, the only dissipation is due to Johnson-Nyquist noise in the amplifier, which is of the order $4kTf$, with f the amplifier's bandwidth and N the number of synapses. Keeping track of error and number of bit precision L, one reaches the conclusion that

$$E_{dig} \approx 24\log(\frac{1}{p_{err}})\log_2^2(L)NkT, \tag{51}$$

$$E_{memr} \approx \frac{1}{24}\log(\frac{1}{p_{err}})L^2N^2kT, \tag{52}$$

which scales in the number of artificial neurons in favor of digital implementations. This surprising result thus confirms that the devil is in the detail, and that not necessarily analog implementations of analog systems are better.

The architecture of Fig. 13 however does not take advantage of the scalability of crossbar arrays which we have mentioned earlier, in terms of number of neurons, and other implementations might be more energy efficient. For instance, in [140] first experimental results of memristive technology for pattern classification on a 3x3 image matrix was studied using crossbars and TiO_2 memristors. In general, learning using crossbars follows a general weight update strategy. For regressions, current controlled memristors can be used with a simple serial architecture [63] while unsupervised learning can be performed by implementing the K-means algorithm [141–143].

Next, we focus on applications of memristor technology for Machine Learning (ML). Algorithms like backpropagation on conventional general-purpose digital hardware (i.e., von Neumann architecture) is highly inefficient: one reason for this is the physical separation between the memory

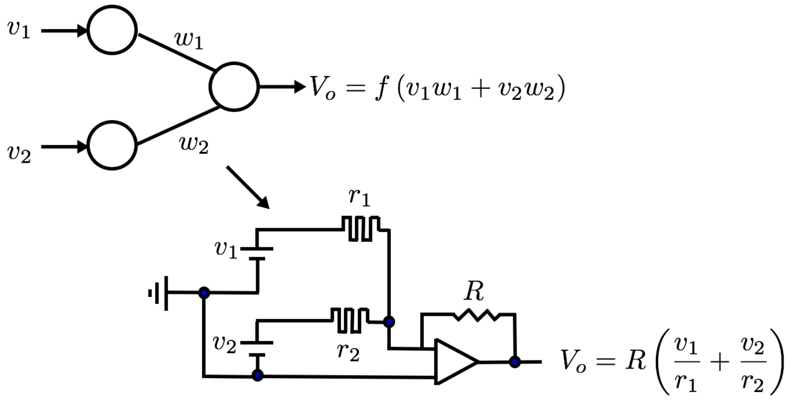

v_1
v_2
w_1
w_2
$V_o = f(v_1w_1 + v_2w_2)$
r_1
r_2
v_1
v_2
R
$V_o = R\left(\frac{v_1}{r_1} + \frac{v_2}{r_2}\right)$

Figure 13. Memristor equivalent of a neural network with three neurons and two synapses, with an output amplifier.

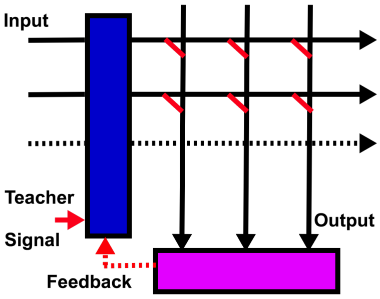


Figure 14. Feedback loop and learning in crossbar arrays.

storage (RAM) of synaptic weights and the arithmetic module (CPU), which is used to compute the update rules. The bus between CPU and RAM acts as a bottleneck, commonly called von Neumann bottleneck. The idea is thus to use memristive based technology to introduce computation and storage on the same platform. One way to introduce learning into crossbars is by introducing feedback into the system, as in Fig. 14.

Let us consider a discretized dynamics, and call W^k the weight matrix in the crossbar array at time step k. A general update is of the form

$$W^{k+1} = W^k + f(W^k, Q), \tag{53}$$

where $f(W^k)$ is a certain function of the weights and some ancillary variables Q (which could be training data, inputs, outputs, etc). For instance, in the case of neural networks training (gradient descent type), $f(W^k) = -\eta \nabla_{W^k} \|\vec{t} - \vec{o}\|^2$ where $\vec{t}$ is the output we aim to obtain (given the inputs), and $\vec{o}$ is the output. For resistive crossbars, we have seen that $\vec{o} = G(W^k)\vec{v}$ is linear in the inputs and G is the conductance, while η is a time scale parameter. If $f(W)_{mn} = \eta x_m^k x_n^k$, where x_n^k is the teacher inputs (patterns) indexed by the index $k = 1, \cdots, K$, then the Hebbian learning rule called adaline algorithm [57,144]. From the point of view of circuit theory, the feedback can be introduced via CMOS-Memristor integration. For instance, in [145] one way to perform online learning with an adaline algorithm using metal-oxide-semiconductor field-effect transistors (MOSFETs).

Models like the one just described can, for instance, be used for sparse coding [146]. Sparse coding can be mathematically formulated as a problem in linear algebra. Let us consider a vector which describes a certain quantity of interest $\vec{x}$ (for instance, an image), and a dictionary which is available as $\vec{\phi}^i, i = 1, \cdots, M$. Let us assume that ϕ^i and $\vec{x}$ belong to $\mathbb{R}^N$. If $M > N$ and the vectors are independent, we can always find the coefficients a_i such that $\sum_i a_i \vec{\phi}^i = \vec{x}$, and there is an infinite number of ways to do this expansion. The goal of sparse coding is solving the following optimization problem:

$$\vec{x} = \sum_{i=1}^{M} a_i \vec{\phi}^i, \tag{54}$$

$$\min_{\vec{a}} \|\vec{a}\|_0, \ \|\cdot\| \text{ 0-norm}, \tag{55}$$

and which is notoriously NP-hard. The problem above can be relaxed by replacing the 0-norm with the 1-norm, and a system of differential equations for continuous neurons can be implemented in crossbar arrays. In general, equations of the type

$$\frac{d}{dt}\vec{u}(t) = F(\vec{u}(t), \vec{q}(t)), \tag{56}$$

where $u_i(t)$ and $q(t)$ are some control functions and $F(\cdot)$ represents a generic continuous function. Some details are provided in the Appendix B. The variables $u_i(t)$ are intented as the memory elements in a memristor: in a crossbar array system, the equations above have been implemented by combining a field programmable gate array (FPGA) with a crossbar in [147], where using a threshold function $T_\lambda(x) = x$ if $x > \lambda$ and zero otherwise. Another update rule used in experiments is Sanger's update rule [148], defined as

$$W^{k+1} = W^k + 2\eta \vec{o}^t (\vec{x} - (2W^k - I)\vec{o}), \tag{57}$$

where $\vec{o}$ is the output of the crossbar array, and $\vec{x}$ the input, and used in [149] in order to perform PCA, again with the use of FPGA.

We have already discussed reservoir computation [63,89] as another way of using memristors within the framework of machine learning whilst taking advantage of their temporal dynamics. Reservoir computation is usually divided into two main parts: a network (called reservoir) in which the connectivity is fixed and connected to the input, and a second network (which is being trained) which is squeezed between the reservoir and the output. This framework was implemented with a trivial memristor reservoir and one layer output with 32x32 crossbar of memristors and trained using an external FPGA. The model was then used to classify the MNIST dataset (5x4 images) as a proof of principle application. The advantage of using reservoir computing is that it can be implemented both for online learning (for instance via logistic regressions) and for classification in a teacher-signal framework, and with relatively little computational effort.

Concluding, CMOS provides an advantage for controlling memristive circuits, but it is possible to use just the inner dynamics of memristors to perform learning [103].

The main question that remains unanswered is whether analog system have an advantage over digital ones at all. A strong argument is provided by Vergis *et al.*, where it is shown that analog devices can be simulated with polynomial resources on a digital machine with enough resources. We pointed out the importance of the collective properties of memristive circuits. The dynamics of a collection of memristors interacting on a circuit can, in principle, derived from the implementation of circuit voltage and current constraints, and strongly depends on the dynamics of a single unit. Understanding the interaction of memristors via Kirchhoff laws can in principle enable the application of memristors to a variety of computational tasks. Below, we make this more precise using a general mapping from digital to analog computation. We consider a differential equation of the form:

$$\frac{\mathrm{d}\vec{y}}{\mathrm{d}t}(t) = f(\vec{y}(t), t), \quad y(t_0) = y_0, \tag{58}$$

which describes a physical system. In [150], a constructive proof based on Euler integration method is provided, and it is shown that the amount of resources needed to simulate the system above on a digital machine is polynomial in the quantities R and ϵ:

$$R = \max_{t_0 \leq t \leq t_f} ||\frac{\mathrm{d}^2\vec{y}}{\mathrm{d}t^2}(t)||, \ \epsilon = ||\vec{y}(t_f) - \vec{y}^*||, \tag{59}$$

where $\vec{y}^*$ is the simulated system, and thus ϵ is our required precision. Since for quantum systems $R \propto 2^N r^*$, and r^* is a constant, classical computers require an exponential amount of resources to simulate a quantum physical systems. This does not mean that classical systems can always be simulated: if the second derivative is large, our system requires a lot of computational power to be simulated on a digital machine. A similar argument applies also to quantum computers, on which there has been a huge effort in the past decades. In the case of a quantum system with N qubits, the vector $\vec{y}(t)$ is 2^N dimensional according to the Schrödinger equation. In a typical (analog) electronic computer, $\frac{d^2}{dt}\vec{y}$ is the derivative of the voltage. Thus, if our circuit presents instability, it is generically hard to simulate the system. This is specially striking in the case of chaotic behavior, which is known to emerges when a dynamical system is connected to a hard optimization problem [151]. These are also arguments against the simulation of memristive system on a digital computers, which do not apply to the actual physical system performing the analog computation.

7. Closing remarks

In the present paper we have provided an introduction and overview of both the applications as memory, and the appealing features of memristors beyond the purpose of memory storage. We have discussed the history of memristors, with the purpose of helping the reader understand their role in modern electronic circuitry and why memory is a normal feature at the nanoscale. The perspective which we have tried to provide is that despite current applications focus on the implementation of standard machine learning algorithms on chips, memristors can be used to perform analog computation which goes beyond the standard framework of crossbars.

In magnetic materials which are commonly used for memory purposes, the interaction between the magnetic spins (which represent the bits) is purposely avoided in order to guarantees the spin to flip, and memory be lost. Via the interaction between the spins however, logic gates can be constructed. Similarly, memristors can be used as memories trying to be avoid their interaction in the circuit via Kirchhoff laws, or one can try to harness this interaction to perform computation. At the most basic level, memristors can be interpreted as synapses, and the introduction of hybrid CMOS-memristor technology can allow the implementation of supervised and unsupervised machine learning algorithms that make use these features. This resonates with the fact that memristors have been proposed to play an important role in biology, as for instance in the case of the amoeba adaptation and the Hudkin-Huxley model. In electronics, there is an ongoing debate which focuses on the fundamental nature of the memristor [23,25]: shall the memristor be considered as a purely passive and fundamental device, along resistance, capacitance and inductance? Independently from the answer to this question or not, memristive type switching is a realistic phenomenon observed experimentally and fully deserves the attention of scientists and technologists.

Building on these ideas, it make sense to pursue the complex dynamical features of memristors, interacting via Kirchhoff laws, for self-organizing computational devices. Before the dynamical features of memristors can be fully harnessed, it is imperative to be able to understand the single memristor physical principle in order to implement reliable memory units, in particular using the crossbar array framework we have described. This task requires a deep knowledge of the dynamics of a single memristor component, via models which accurately describes the relevant behavior of the device.

We have also tried to emphasize that there are several mechanisms which enable resistive switching, and different components will follow different mechanisms to change state. Despite their difference, the dynamics of components share a common feature: the competition between two phenomena. These phenomena can be cast as “forgetting" (decay to an off state when voltage is not applied) and “reinforcement" (tendency to state change which depends on the current). As discussed by Kohonen, this competition is an important feature among several analog computing paradigms. As examples we take ant-colony optimization [77] and experimental results with memristive devices [37,152].

We also pointed out the importance of the collective properties of memristive circuits for computation. In particular how we can harness the intrinsic variability of memristors to different computational problems. However, analog machines are good at specific computational tasks, while digital ones excel in their generality. Thus the integration of CMOS and analog systems in future computers is favorable in the long term. Memristors are one of many technologies which are able to encode computational tasks and simultaneously be used as memory; which can be easily integrated in modern computers. However, in certain instances the von Neumann architecture is not the best architecture for performing calculations, and we mentioned the case of quadratic optimization and generic combinatorial problems.

Due to our focus in memristives devices for computation, and the models of computation that are compatible with their properties, we have omitted reviewing the role that memristive system can have in biological cognitive systems, and their relation to stochastic resonance [153–160]. This decision was taken to keep the presentation focused and consistent.

In conclusion, we have provided an overview of the current research questions and applications of memristive technology. We have provided also a rather long, but far from exhaustive bibliography on the subject which might help the interested reader in learning the subject.

Acknowledgments: We thank Fabio L. Traversa, Giacomo Indiveri, Alex Nugent, and Miklos Csontos for their useful comments and observations. **FC** acknowledges the support of NNSA for the U.S. DoE at LANL under Contract No. DE-AC52-06NA25396. **JPC** received support from the discretionary funding scheme of The Swiss Federal Institute of Aquatic Science and Technology project EmuMore.

Author Contributions: FC and JPC contributed equally to this work.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A Physical mechanisms for resistive change materials

In the main text we have mentioned that Branly's coherer can be interpreted as a granular material induced memristor. Before we discuss some more modern memristors, we believe it makes sense to give a sense why granularity is important for nonlinear resistive behavior. Branly's coherer serves as perfect homemade memristor, as it simply requires either a (fine) metallic filling or some metallic beads, and it falls within the Physics discipline of electrical properties of granular media. The qualitative and qualitative behavior of the case of the metallic beads contained in (and constrained to) an insulating medium of PVC is presented below [7,20]. The metallic beads are assumed to be in mutual contact, and a force F applied at the two extremities enforces it. We assume that there are two temperatures in the system one at the microcontact between the beads at an equilibrium temperature T [20] and a room temperature T_0 which is the one of the beads. Without going into the details, it is possible to show a non-linear behavior in the resistivity of N beads via the study of the contact between the beads. The metallic beads contacts can be though of as Metal-Oxide-Oxide-Metal contact, reminishent of the memristor we will discuss below. Kohlrausch's equation establishes the voltage drop at the contact. Given the current I flowing in the beads, one has that

$$V = NL \int_{T_0}^{T_c} \lambda(T)\rho_{el}(T)dT, \tag{A1}$$

where $\lambda(T)$ is the thermal conductivity of the material and $\rho_{el}(T)$ the density of electrons, while T_0 is the beads room temperature and T_c the maximum temperature at the contact when the current is flowing. If R_0 is the resistance when the contact is cold, clearly we can rewrite (using Ohm's law) the equation above as the following effective equation:

$$IR_0 = V(T). \tag{A2}$$

where R_0 depends on the geometry of the contact. It can be shown however that the maximum temperature T_c depends on the voltage as $T_m^2 = T_0^2 + \frac{U^2}{4L}$ where L is the Lorentz constant, by noticing that via the Wiedemann-Franz that $\rho_{el}\lambda = LT$. Also Mathiesen's rule for the electron mobility shows that the electron mobility is linear in T, with a proportionality constant that is material dependent. Putting all these facts together, it is not hard to see the hysteretic behavior of the system. This effective model reproduces well the controlled experiments. Since the voltage drop is zero when the current is zero, this also implies a pinched hysteresis, or resistive behavior. These "mechanical" nonlinear resistors are prototypes for the more complicated case shown below [see 20, for more details].

Appendix A.1 Phase change materials

Phase change materials (PCM) are glassy materials: this implies that usually these materials can have different phases in which the material can be either ordered (crystalline-like) or disordered (amorphous, as in a liquid). In these two phases we associate two different resistances. If the structural change can be associated to the values of an applied voltage, then when the transition occurs one has a rather quick transition from one resistive state to another [161–163] due to an electrical instability. These materials are considered memristors by some, but not by all researchers, and were discovered as early as the late '60s [164] in amorphous chalcogenides. This is the most mature of the emerging memory technologies. Since we have used various analogies befores, the reader in need of a visual way to understand these type of materials might find some ideas on the shelves of a pharmacy. Phase-change materials are being used for instant freeze packages normally used in case of injuries, and are also called "gel packs". In order to initiate the cooling, users typically need either to mix two materials, or the quickly apply a certain force on the package. The material will use the "kick" to initiate a phase transition, absorb the heat and thus lower the temperature of the package. PCM memory devices work in a similar manner, but on a much lower scale (~ 20 nm), and the "kick" is provided by the electric field. These materials were not commercial for years due to the rapid advancement of silicon-based

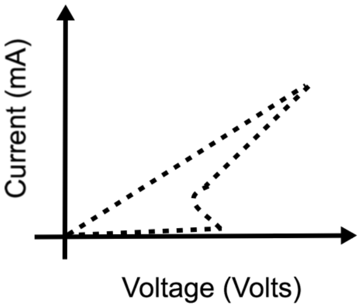


Figure A1. Current-Voltage diagram typical of a PCM memory storage. The sharp transition at low currents and at a typical "Critical voltage" is clearly visible.

technology. The typical I-V diagram is shown in Fig. A1. One generically has two type of materials squeezed between two electrodes: on one side one has an insulating material with a small conducting channel, directly connected to the phase change material. As the channel heats up, the phase change material locally changes phase starting from point of contact at the conduction channel, until it reaches the other electrode.

In some chalcogenides-based memristors (called Self-Directed channel memristors, or SDC), ions are constrained to follow certain channels, and their operation is similar in many ways to both PCM-components and atomic switches [165].

Appendix A.2 Oxide based materials

The oxide and anionic materials based on transition metal work instead differently. In order to understand why a memristor might be different from a normal conductor it is useful to understand what happens when two materials with different properties are "merged" together: those of charge donor (excess of electrons) and charge receiver (excess of electron "holes"), which are also called doped and undoped materials. In oxide materials the carrier of the charge is typically the oxygen. It should be mentioned that whilst various mechanisms have been suggested, very likely all of these coexist in a typical oxide material, including filament formation [166]. In general, whether the resistive switching is thermally or electrically driven, the typical understanding is that a chemical transition occurs in the material, and that the hysteresis is due to vacancy movements in the materials. The transition is either directly driven by the direct application of the electric field, or as a byproduct of the heating of the material due to the current. In semiconductors, the key quantity of interest is the energy gap E_g between the valence and the conduction channels. If the gap is of the order of $E_p \approx kT$, then thermal effects which might let a charge carrier jump into the conduction channel become important. If the gap is too large, electric effects are the dominant ones. One example is provided by the bipolar and non-volatile switching, described by Fig. A2.

The shape of I-V curve generically shows what type of mechanism is underlying the switching. As the field effects become more prominent (for instance, the effect of Schottky barriers at the junctions), the I-V diagram becomes non-linear. There can be other non-volatile and non-volatile switching in which we are not discussing here, and due to thermal excitations only. A simplified model of vacancy-charge movement in the dielectric has been proposed in [167]. When non-linearities are present in non-volatile materials, it means that the switching is dominated by the electrical switching. In many ways, some of the physical phenomena happening in memristors can be intuitively understood in terms of the simplest semiconductor: the diode, represented in Fig. A3 (a).

The diode can be thought of as a dramatically nonlinear resistance, made by merging two material, a doped (filled with defects) and undoped one, with a thin interstitial when charges exchange. We can characterize the diode by the two voltages $V_{break} < 0 < V_{crit}$. For voltages above V_{crit}, the diode

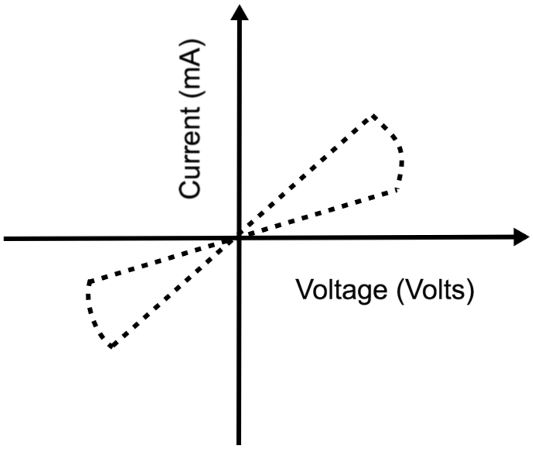


Figure A2. Bipolar and non-volatile switching pinched hysteresis.

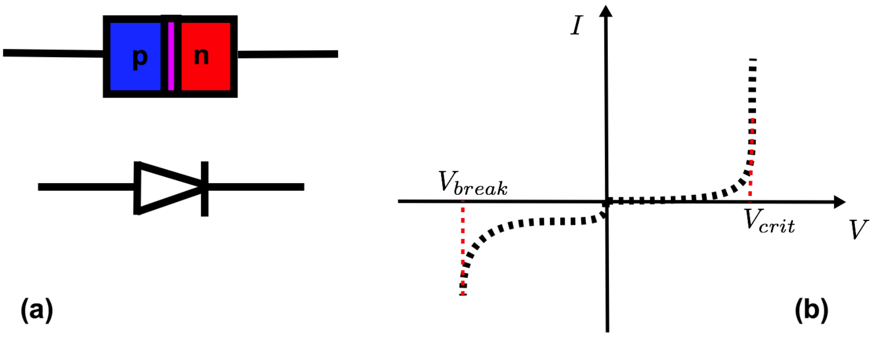
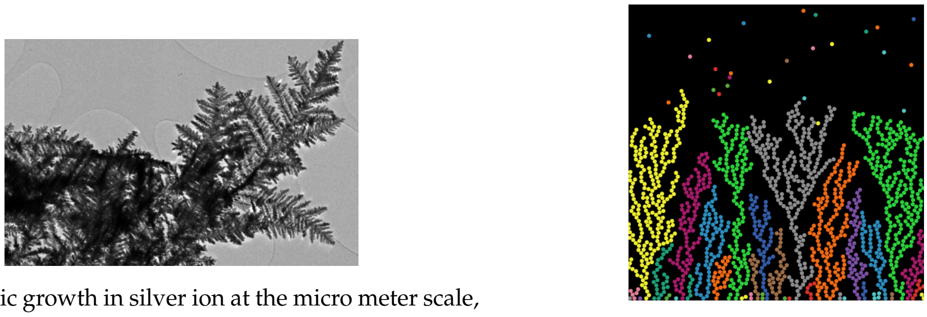


Figure A3. The pn-junction. (a) The pink region is the charge exchange region for the doped and undoped region. (b) A stylized response in voltage of a diode. In many ways, the shape resembles the nonlinearity which occur in nonlinear memristors in which field effects become dominant.

will have a very low resistance, and for voltages lower than V_{break}, the diode will breakdown. In many ways, the shape of Fig. A3 (b) is what is usually seen in memristive oxide components, with the difference that the memristor will exhibit a hysteresis. At the interfaces (pink zone in Fig. A3 (a)), the charge carriers will have to overcome a barrier (Schottky barrier) (characterized by V_{crit}) to continue their flow into the material. When this barrier is overcome, the flow is almost free. On the other hand, when the flow is inverted, the undoped material will act as a very high barrier to the flow into the dielectric region. However, the charge carriers can still penetrate the material and damage it in an irreversible manner. These are usually called Zener cascades, and are often observed in memristors. Albeit cartoonish, this picture can help understand some of the nonlinear phenomena happening in memristors, and not captured by the linear resistance model we discussed above. As a final comment, endurance in oxide materials is one of the key problems in the technological competitiveness of memristors [168]. When the number of cycles becomes comparable with the lifetime of the component, also some exotic phenomena as multiple pinchpoints (tri- or four-lobes hysteresis loops) in the V-I diagram can occur. These can be modelled via the introduction of fractal derivatives [169].

Appendix A.3 Atomic switches

The third mechanism described in this paper (albeit not the last!), and maybe the most visually appealing, is the filament growth memristor. The materials, often called *atomic switch networks* in the literature, work in a slightly different manner than the previously described memristors [96,170–173]. The idea behind these components is that the two electrodes work, when the electrical field is applied, as an electrode and a catod. The applied electric field induces an electrochemical reaction which triggers the growth of filaments. From a highly resistive state, these filaments reduce the resistivity by introducing new channels for the charge carriers to flow. The closest physical growth model which can describe the growth of the filaments is provided by Diffusion-Limited-Aggregation (as in Fig. A4b). Each colored filament represents one possible channel. The typical charge carriers are either silver ions or some silver compounds. There are not many experiments focused instead on the collective behavior of memristive networks. It is worth mentioning however what are the observed features for Ag^+, or Atomic Switch Networks, whose collective dynamics is interesting for the emergence of seemingly critical states [174]. In fact, whilst the dynamic of a single filament is simpler to describe, the system exhibits collective power law power spectrum with an exponent close to 2. Albeit this exponent can be explained via the superposition of wide range of relaxation timescales for each memristor [90], the critical behavior is the accepted one because of their intrinsic nonlinearity [96].



(a) Dendritic growth in silver ion at the micro meter scale, from [175].

(b) Diffusion-Limited-Aggregation simulated using NetLogo.

Figure A4. Comparison between dendritic growth in silver ion materials and Diffusion-Limited-Aggregation simulated using NetLogo.

In order to see the variety of memristive behavior, ere we consider two models suggested in the literature which are different than the simple TiO_2 linear model memristor.

The first, suggested in [176] as a phenomenological switching model between two off and on states in TiO_2, is of the form:

$$R(w) = R_{\text{off}}(1-w) + R_{\text{on}}w, \tag{A3}$$

$$\frac{dw}{dt} = \begin{cases} f_{\text{off}} \sinh(\frac{i}{i_{\text{off}}}) e^{-e^{\frac{w-a_{\text{off}}}{w_c} - \frac{|i|}{b}} - \frac{w}{w_c}} & i > 0, \\ f_{\text{on}} \sinh(\frac{i}{i_{\text{on}}}) e^{-e^{-\frac{w-a_{\text{on}}}{w_c} - \frac{|i|}{b}} - \frac{w}{w_c}} & i < 0, \end{cases} \tag{A4}$$

where the parameters $f_{\text{on/off}}$, $i_{\text{on/off}}$ and $a_{\text{on/off}}$ are state dependent, while w_c and b are not. Also, the model above shows that the energy depends exponentially on the current.

Another metal oxide of interest is WO_x, studied in [177]. The model equations for a single component are given by:

$$I = \alpha(1-w)\left(1 - e^{\beta V}\right) + w\gamma \sinh(\delta V), \tag{A5}$$

$$\frac{\mathrm{d}w}{\mathrm{d}t} = \lambda\left(e^{\eta_1 V} - e^{-\eta_2 V}\right). \tag{A6}$$

The model above, it is interesting to note, is controlled in voltage rather than current, and the parameters $\alpha, \beta, \gamma, \delta$ and η_1 and η_2 are positive. For $V = 0$, $I = 0$, which implies a pinched hysteresis (however, the hysteresis is different from TiO_2 devices). The parameter w is physically interpreted as the portion of the device in which oxygen charges tunnel through the device. For $w = 1$ one has tunneling dominated device, while at $w = 0$ one has a Schottky-dominated conduction.

One question which might be relevant to mention at this point is: what is the advantage of using memristors rather than other memory devices? The perpective of the present paper is that there are two different reasons why these devices can turn useful [178]. The first advantages is the density compared to standard memory. For instance, compared to DRAM and SRAM, memristors (or PCM) retain memory for years rather than less than seconds. Compared to SRAM however, whose read-write time is less than a nanosecond, memristors with current technology are one order of magnite slower. Also, in terms of read-write cycles, the technology of memristors and PCM is between $3-5$ order of magnitude less, but still much more durable than HDD. The picture is that memristors and PCM are not uniquely better than the current standard in computing.

Appendix A.4 Spin torque

Spin-torque memory materials have an advantage in terms of durability Grollier *et al.* over other materials such as transition metal oxides. These are often considered as second-order memristive devices but a simplified model of spin-torque induced resistance is provided in [180]. The starting point is the Landau-Ginzburg-Gilbert (LGG) equation with a spin-torque interaction [181]. We consider two magnetic layers perpendicular to the flow of the current, and in which one is fully polarized: its magnetic orientation is fixed in a direction perpendicular to the current, while the second layer is free. Via the LGG equation with rotational symmetry, the dynamics of the angle between the pinned and free layer is given by

$$\frac{d}{dt}\theta(t) = \alpha\gamma H_k \sin(\theta(t))\,(p - \cos(\theta(t)))\,, \tag{A7}$$

where γ is called gyromagnetic ratio, α is the damping parameter, H_k is the perpendicular anysotropy in th free layer and $p = f\hbar I$ is a current dependent which represents the effect of the current-polarization interaction. We have emphasized the presence of $\hbar$ to imply that this is a purely quantum correction.

In order to see how this device is a memristor, we see that in the simpler case with full rotational symmetry one has an induced magneto-resistance $R(\theta)$, which depends on the angle θ in the case of full rotational symmetry as:

$$R\left(\theta(t)\right) = \frac{1}{a + b\cos\left(\theta(t)\right)}, \tag{A8}$$

where a and b are constants which depend on the resistance in the free layer and on the ration between the highest and lowest achievable resistances. We thus see that in the case with full rotational symmetry, spin-torque materials are first order memristors.

Appendix B Sparse coding example

Sparse coding is the solution of the following optimization problem:

$$\vec{x} = \sum_{i=1}^{M} a_i \vec{\phi}^i, \tag{A9}$$

$$\min_{\vec{a}} \|\vec{a}\|_0, \ \|\cdot\| \text{ 0-norm}, \tag{A10}$$

and which is notoriously NP-hard. Given some technical conditions which we do not discuss, the problem above can be approximated in certain situations by replacing the 0-norm with the 1-norm, and the minimization replaced with

$$\min_{\vec{a}} \|\vec{x} - \sum_{i=1}^{M} a_i \vec{\phi}^i\|_2^2 + \lambda \|\vec{a}\|_1, \tag{A11}$$

where $\|\cdot\|_2$ is the two norm and λ a Lagrange multiplier. The problem above can be encoded in a neural system via a locally competitive algorithm (LCA), which is formulated as follows. Given the coefficients $a_i(t)$, consider a "neuron" variable $u_i(t)$ such that $a_i(t) = T_\lambda\left(a_i(t)\right)$ and where $T_\lambda(\cdot)$ is a threshold function. We consider an energy function of the form

$$E = \frac{1}{2}\|\vec{x} - \hat{x}\| + \lambda \sum_m C(a_m), \tag{A12}$$

with $\vec{x}$ defined as above and $\hat{x} = \sum_{i=1}^{M} a_i \vec{\phi}^i$, and $C(\cdot)$ a certain unspecified cost function. One then looks at a dynamics for the neuron state $u_i(t)$ of the form

$$\dot{u}_i(t) = \frac{1}{\tau}\frac{\delta}{\delta a_i} E, \tag{A13}$$

for a certain relaxation constant τ, which it can be easily seen to be defined (given $\vec{b}(t) = \Phi\vec{x}(t)$, $\Phi = [\vec{\phi}^1, \cdots, \vec{\phi}^M]$, as

$$\dot{u}_m(t) = \frac{1}{\tau}\left(b_m(t) - u_m(t) - \sum_{n \neq m} G_{mn} a_n(t)\right), \tag{A14}$$

with $G_{mn} = \vec{\phi}^m \cdot \vec{\phi}^n$. The correspondence between the threshold function and the cost function is given by the equation:

$$\lambda \frac{d}{da_m} C(a_m) = u_m - a_m = \left(u_m - T_\lambda(u_m)\right) \tag{A15}$$

The equations above are suitable to be implemented on a memristive circuit, as besides a forcing term and a leaky integration term, there is a nonlinear integration term and are implentable via Hopfield continuous networks [182,183]. The equations above would be linear if the thresholding function T_λ were trivial.

1. Strukov, D.B.; Snider, G.S.; Stewart, D.R.; Williams, R.S. The missing memristor found. *Nature* **2008**, *453*, 80–83.
2. Chua, L. If it's pinched it's a memristor. *Semiconductor Science and Technology* **2014**, *29*, 104001.
3. Chua, L. Memristor-The missing circuit element. *IEEE Transactions on Circuit Theory* **1971**, *18*, 507–519.
4. Chua, L.; Sung Mo Kang. Memristive devices and systems. *Proceedings of the IEEE* **1976**, *64*, 209–223.
5. Valov, I.; Linn, E.; Tappertzhofen, S.; Schmelzer, S.; van den Hurk, J.; Lentz, F.; Waser, R. Nanobatteries in redox-based resistive switches require extension of memristor theory. *Nature Communications* **2013**, *4*, 1771.
6. Waser, R.; Aono, M. Nanoionics-based resistive switching memories. *Nature Materials* **2007**, *6*, 833–840.
7. Béquin, P.; Tournat, V. Electrical conduction and Joule effect in one-dimensional chains of metallic beads: hysteresis under cycling DC currents and influence of electromagnetic pulses. *Granular Matter* **2010**, *12*, 375–385.
8. Di Ventra, M.; Pershin, Y.V. On the physical properties of memristive, memcapacitive and meminductive systems. *Nanotechnology* **2013**, *24*, 255201.
9. Di Ventra, M.; Pershin, Y.V. Memory materials: a unifying description. *Materials Today* **2011**, *14*, 584–591.
10. Kuzum, D.; Yu, S.; Wong, H.S.P. Synaptic electronics: materials, devices and applications. *Nanotechnology* **2013**, *24*, 382001.
11. Yang, J.J.; Strukov, D.B.; Stewart, D.R. Memristive devices for computing. *Nature Nanotechnology* **2013**, *8*, 13–24.
12. Mikhaylov, A.N.; Gryaznov, E.G.; Belov, A.I.; Korolev, D.S.; Sharapov, A.N.; Guseinov, D.V.; Tetelbaum, D.I.; Tikhov, S.V.; Malekhonova, N.V.; Bobrov, A.I.; Pavlov, D.A.; Gerasimova, S.A.; Kazantsev, V.B.; Agudov, N.V.; Dubkov, A.A.; Rosário, C.M.M.; Sobolev, N.A.; Spagnolo, B. Field- and irradiation-induced phenomena in memristive nanomaterials. *physica status solidi (c)* **2016**, *13*, 870–881.
13. Mead, C. Neuromorphic electronic systems. *Proceedings of the IEEE* **1990**, *78*, 1629–1636.
14. Freeth, T.; Bitsakis, Y.; Moussas, X.; Seiradakis, J.H.; Tselikas, A.; Mangou, H.; Zafeiropoulou, M.; Hadland, R.; Bate, D.; Ramsey, A.; Allen, M.; Crawley, A.; Hockley, P.; Malzbender, T.; Gelb, D.; Ambrisco, W.; Edmunds, M.G. Decoding the ancient Greek astronomical calculator known as the Antikythera Mechanism. *Nature*, *444*, 587–591.
15. Adamatzky, A., Ed. *Advances in Physarum Machines*; Vol. 21, *Emergence, Complexity and Computation*, Springer International Publishing: Cham, 2016.
16. Dalchau, N.; Szép, G.; Hernansaiz-Ballesteros, R.; Barnes, C.P.; Cardelli, L.; Phillips, A.; Csikász-Nagy, A. Computing with biological switches and clocks. *Natural Computing*.
17. Di Ventra, M.; Pershin, Y.V. The parallel approach. *Nature Physics* **2013**, *9*, 200–202.
18. Davy, H. Nicholson's Journal of Natural Philosophy. *Chemistry, and the Arts* **1801**, *4*, 326.
19. Falcon, E.; Castaing, B.; Creyssels, M. Nonlinear electrical conductivity in a 1D granular medium. *The European Physical Journal B* **2004**, *38*, 475–483.
20. Falcon, E.; Castaing, B. Electrical conductivity in granular media and Branly's coherer: A simple experiment. *American Journal of Physics* **2005**, *73*, 302–307.
21. Branly, E. Variations de conductibilite sous diverse influences electriques. *R. Acad. Sci* **1890**, *111*, 785–787.
22. Marconi, G. Wireless Telegraphic Communication: Nobel Lecture 11 December 1909, Nobel Lectures. In *Physics*; Elsevier Publishing Company 196–222. p. 198: Amsterdam, 1967; pp. 1901–1921.
23. Abraham, I. The case for rejecting the memristor as a fundamental circuit element. *Scientific Reports* **2018**, *8*, 10972.
24. Strogatz, S. Like Water for Money. https://opinionator.blogs.nytimes.com/2009/06/02/guest-column-like-water-for-money/, 2009. Last accessed on 2018-08-06.
25. Vongehr, S.; Meng, X. The Missing Memristor has Not been Found. *Scientific Reports* **2015**, *5*, 11657.
26. Vongehr, S. Purely Mechanical Memristors: Perfect Massless Memory Resistors, the Missing Perfect Mass-Involving Memristor, and Massive Memristive Systems, [1504.00300].
27. Volkov, V.; et al. Memristors in plants. *Plant Signal Behav.* **2014**, *9*.
28. Gale, E.; Adamatzky, A.; de Lacy Costello, B. Slime Mould Memristors. *BioNanoScience* **2015**, *5*, 1–8.
29. Gale, E.; Adamatzky, A.; de Lacy Costello, B. Erratum to: Slime Mould Memristors. *BioNanoScience* **2015**, *5*, 9–9.

30. Szot, K.; Dittmann, R.; Speier, W.; Waser, R. Nanoscale resistive switching. *Phys. Status Solidi* **2007**, *1*, 10076–10092.
31. Tsuruoka, T.; K., T.; Hasegawa, T.; Aono, M. Forming and switching mechanisms of a cation-migration-based oxide resistive memory. *Nanotechnology* **2010**, *21*.
32. Abraham, I. Quasi-Linear Vacancy Dynamics Modeling and Circuit Analysis of the Bipolar Memristor. *PLoS ONE* **2014**, *9*, e111607.
33. Abraham, I. An Advection-Diffusion Model for the Vacancy Migration Memristor. *IEEE Access* **2016**, *4*, 7747–7757.
34. Abraham, I. Memristor - The fictional circuit element. *Scientific Reports* **2018**, *8*, 10972, [1808.05982].
35. Tang, S.; Tesler, F.; Marlasca, F.G.; Levy, P.; Dobrosavljević, V.; Rozenberg, M. Shock Waves and Commutation Speed of Memristors. *Physical Review X* **2016**, *6*, 011028.
36. Wang, F. Memristor for Introductory Physics, [0808.0286].
37. Ohno, T.; Hasegawa, T.; Nayak, A.; Tsuruoka, T.; Gimzewski, J.K.; Aono, M. Sensory and short-term memory formations observed in a Ag2S gap-type atomic switch. *Appl. Phys. Lett.* **2011**, *203108*, 1–3.
38. Pershin, Y.V.; Ventra, M.D. Spice model of memristive devices with threshold. *Radioengineering* **2013**, pp. 485–489.
39. Biolek, Z.; Biolek, D.; Biolková, V. Spice Model of Memristor With Nonlinear Dopant Drift. *Radioengineering* **2009**, pp. 210–214.
40. Biolek, D.; Di Ventra, M.; Pershin, Y.V. Reliable SPICE Simulations of Memristors, Memcapacitors and Meminductors. *Radioengineering* **2013**, *22*.
41. Biolek, D.; Biolek, Z.; Biolkova, V.; Kolka, Z. Reliable modeling of ideal generic memristors via state-space transformation. *Radioengineering* **2015**, *24*, 393–407.
42. Nedaaee Oskoee, E.; Sahimi, M. Electric currents in networks of interconnected memristors. *Physical Review E* **2011**, *83*, 031105.
43. Joglekar, Y.N.; Wolf, S.J. The elusive memristor: properties of basic electrical circuits. *European Journal of Physics* **2009**, *30*, 661–675.
44. Mostafa, H.; Khiat, A.; Serb, A.; Mayr, C.G.; Indiveri, G.; Prodromakis, T. Implementation of a spike-based perceptron learning rule using TiO2-x memristors. *Frontiers in Neuroscience* **2015**, *9*.
45. Linn, E.; Di Ventra, M.; Pershin, Y.V. ReRAM Cells in the Framework of Two-Terminal Devices. In *Resistive Switching*; Wiley-VCH Verlag GmbH & Co. KGaA: Weinheim, Germany, 2016; pp. 31–48.
46. Pi, S.; Li, C.; Jiang, H.; Xia, W.; Xin, H.; Yang, J.J.; Xia, Q. Memristor Crossbars with 4.5 Terabits-per-Inch-Square Density and Two Nanometer Dimension, [1804.09848].
47. Meena, J.; Sze, S.; Chand, U.; Tseng, T.Y. Overview of emerging nonvolatile memory technologies. *Nanoscale Research Letters* **2014**, *9*, 526.
48. Steinbuch, K. Die Lernmatrix. *Kybernetik* **1961**, *1*, 36–45.
49. Kohonen, T. *Self-Organization and Associative Memory*; Vol. 8, *Springer Series in Information Sciences*, Springer Berlin Heidelberg: Berlin, Heidelberg, 1989.
50. Steinbuch, K. Adaptive networks using learning matrices. *Kybernetik* **1964**, *2*.
51. Xia, L.; Gu, P.; Li, B.; Tang, T.; Yin, X.; Huangfu, W.; Yu, S.; Cao, Y.; Wang, Y.; Yang, H. Technological Exploration of RRAM Crossbar Array for Matrix-Vector Multiplication. *Journal of Computer Science and Technology* **2016**, *31*, 3–19.
52. Strukov, D.B.; Williams, R.S. Four-dimensional address topology for circuits with stacked multilayer crossbar arrays. *Proceedings of the National Academy of Sciences* **2009**, *106*, 20155–20158.
53. Li, C.; Han, L.; Jiang, H.; Jang, M.H.; Lin, P.; Wu, Q.; Barnell, M.; Yang, J.J.; Xin, H.L.; Xia, Q. Three-dimensional crossbar arrays of self-rectifying Si/SiO2/Si memristors. *Nature Communications* **2017**, *8*, 15666.
54. Li, C.; Belkin, D.; Li, Y.; Yan, P.; Hu, M.; Ge, N.; Jiang, H.; Montgomery, E.; Lin, P.; Wang, Z.; Song, W.; Strachan, J.P.; Barnell, M.; Wu, Q.; Williams, R.S.; Yang, J.J.; Xia, Q. Efficient and self-adaptive in-situ learning in multilayer memristor neural networks. *Nature Communications* **2018**, *9*, 2385.
55. Itoh, M.; Chua, L. Memristor Cellular Automata and Memristor Discrete-Time Cellular Neural Networks. In *Memristor Networks*; Springer International Publishing: Cham, 2014; pp. 649–713.
56. Nugent, M.A.; Molter, T.W. Thermodynamic-RAM technology stack. *International Journal of Parallel, Emergent and Distributed Systems* **2018**, *33*, 430–444, [https://doi.org/10.1080/17445760.2017.1314472].

57. Hebb, D. *The Organization of Behavior*; Wiley & Sons, 1949.
58. Koch, G.; Ponzo, V.; Di Lorenzo, F.; Caltagirone, C.; Veniero, D. Hebbian and Anti-Hebbian Spike-Timing-Dependent Plasticity of Human Cortico-Cortical Connections. *Journal of Neuroscience* **2013**, *33*, 9725–9733, [http://www.jneurosci.org/content/33/23/9725.full.pdf].
59. La Barbera, S.; Alibart, F. Synaptic Plasticity with Memristive Nanodevices. In *Advances in Neuromorphic Hardware Exploiting Emerging Nanoscale Devices*; Suri, M., Ed.; Springer: New Delhi, 2017; chapter 2, pp. 17–43.
60. Ielmini, D. Brain-inspired computing with resistive switching memory (RRAM): Devices, synapses and neural networks. *Microelectronic Engineering* **2018**, *190*, 44–53.
61. Serrano-Gotarredona, T.; Masquelier, T.; Prodromakis, T.; Indiveri, G.; Linares-Barranco, B. STDP and STDP variations with memristors for spiking neuromorphic learning systems. *Frontiers in Neuroscience* **2013**, *7*.
62. Payvand, M.; Nair, M.; Müller, L.; Indiveri, G. A neuromorphic systems approach to in-memory computing with non-ideal memristive devices: From mitigation to exploitation. *Faraday Discussions* **2018**.
63. Carbajal, J.P.; Dambre, J.; Hermans, M.; Schrauwen, B. Memristor models for machine learning. *Neural Computation* **2015**, *27*, [1406.2210].
64. Eliasmith, C.; Anderson, C.H. *Neural Engineering: Computational, Representation, and Dynamics in Neurobiological Systems*; MIT Press: Cambridge, MA, USA, 2002.
65. MacLennan, B.J. Analog computation. In *Computational Complexity: Theory, Techniques, and Applications*; Meyers, R.A., Ed.; Springer: New York, 2012; pp. 161–184.
66. MacLennan, B. The promise of analog computation. *International Journal of General Systems* **2014**, *43*, 682–696.
67. MacLennan, B.J. Physical and Formal Aspects of Computation: Exploiting Physics for Computation and Exploiting Computation for Physical Purposes. In *Advances in Unconventional Computing*; A., A., Ed.; Springer: Cham, 2017; pp. 117–140.
68. Horsman, C.; Stepney, S.; Wagner, R.C.; Kendon, V. When does a physical system compute? *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **2014**, *470*, 20140182–20140182.
69. Shannon, C.E. Mathematical Theory of the Differential Analyzer. *Journal of Mathematics and Physics* **1941**, *20*, 337–354.
70. Borghetti, J.; Snider, G.S.; Kuekes, P.J.; Yang, J.J.; Stewart, D.R.; Williams, R.S. 'Memristive' switches enable 'stateful' logic operations via material implication. *Nature* **2010**, *464*, 873–876.
71. Moreau, T.; San Miguel, J.; Wyse, M.; Bornholt, J.; Alaghi, A.; Ceze, L.; Enright Jerger, N.; Sampson, A. A Taxonomy of General Purpose Approximate Computing Techniques. *IEEE Embedded Systems Letters* **2018**, *10*, 2–5.
72. Levi, M. A Water-based Solution of Polynomial Equations. https://sinews.siam.org/Details-Page/a-water-based-solution-of-polynomial-equations-2, 2017. Last accessed on 2018-06-20.
73. Axehill, D. Integer quadratic programming for control and communication. PhD thesis, Institute of Technology. Department of Electrical Engineering and Automatic Control. Linköping University, 2008.
74. Venegas-Andraca, S.E.; Cruz-Santos, W.; McGeoch, C.; Lanzagorta, M. A cross-disciplinary introduction to quantum annealing-based algorithms. *Contemporary Physics* **2018**, *59*, 174–197.
75. Rothemund, P.W.K.; Papadakis, N.; Winfree, E. Algorithmic Self-Assembly of DNA Sierpinski Triangles. *PLoS Biology* **2004**, *2*, e424.
76. Qian, L.; Winfree, E.; Bruck, J. Neural network computation with DNA strand displacement cascades. *Nature* **2011**, *475*, 368–372.
77. Dorigo, M.; Gambardella, L.M. Ant colonies for the travelling salesman problem. *Biosystems* **1997**, *43*, 73–81.
78. Bonabeau, E. Editor's Introduction: Stigmergy. *Artificial Life* **1999**, *5*, 95–96.
79. Muller, K.R.; Mika, S.; Ratsch, G.; Tsuda, K.; Scholkopf, B. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks* **2001**, *12*, 181–201.
80. Hofmann, T.; Schölkopf, B.; Smola, A.J. Kernel methods in machine learning. *The Annals of Statistics* **2008**, *36*, 1171–1220.
81. Huang, G.B.; Zhu, Q.Y.; Siew, C.K. Extreme learning machine: a new learning scheme of feedforward neural networks. Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on. IEEE, 2004, Vol. 2, pp. 985–990.

82. Patil, A.; Shen, S.; Yao, E.; Basu, A. Hardware architecture for large parallel array of Random Feature Extractors applied to image recognition. *Neurocomputing* **2017**, *261*, 193–203.
83. Parmar, V.; Suri, M. Exploiting Variability in Resistive Memory Devices for Cognitive Systems. In *Advances in Neuromorphic Hardware Exploiting Emerging Nanoscale Devices*; M., S., Ed.; Springer: New Delhi, 2017; chapter 9, pp. 175–195.
84. Hoeting, J.A.; Madigan, D.; Raftery, A.E.; Volinsky, C.T. Bayesian model averaging: A tutorial. *Statistical Science* **1999**, *14*, 382–401.
85. Jaeger, H. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. Technical report, German National Research Institute for Computer Science, Bonn, Germany, 2001.
86. Maass, W.; Natschläger, T.; Markram, H. Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural computation* **2002**, *14*, 2531–60.
87. Verstraeten, D. Reservoir Computing: computation with dynamical systems. Phd dissertation, Ghent University, 2009.
88. Fernando, C.; Sojakka, S. Pattern Recognition in a Bucket. In *Advances in Artificial Life*; Banzhaf, W.; Ziegler, J.; Christaller, T.; Dittrich, P.; Kim, J.T., Eds.; Springer Berlin Heidelberg, 2003; pp. 588–597.
89. Du, C.; Cai, F.; Zidan, M.A.; Ma, W.; Lee, S.H.; Lu, W.D. Reservoir computing using dynamic memristors for temporal information processing. *Nature Communications* **2017**, *8*, 2204.
90. Caravelli, F.; Traversa, F.L.; Di Ventra, M. Complex dynamics of memristive circuits: Analytical results and universal slow relaxation. *Physical Review E* **2017**, *95*, 022140.
91. Sheriff, M.R.; Chatterjee, D. Optimal Dictionary for Least Squares Representation. *Journal of Machine Learning Research* **2017**, *18*, 1–28, [1603.02074].
92. Corradi, F.; Eliasmith, C.; Indiveri, G. Mapping arbitrary mathematical functions and dynamical systems to neuromorphic VLSI circuits for spike-based neural computation. 2014 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2014, pp. 269–272.
93. Benna, M.K.; Fusi, S. Computational principles of synaptic memory consolidation. *Nature Neuroscience* **2016**, *19*, 1697–1706.
94. Chialvo, D. Are our senses critical? *Nat. Phys.* **2006**, *2*, 301–302.
95. Hesse, J.; Gross, T. Self-organized criticality as a fundamental property of neural systems. *Front. Syst. Neurosci.* **2014**, *23*.
96. Avizienis, A.; et al.. Neuromorphic Atomic Switch Networks. *PLoS ONE* **2012**, *7*.
97. Caravelli, F.; Hamma, A.; Di Ventra, M. Scale-free networks as an epiphenomenon of memory. *EPL (Europhysics Letters)* **2015**, *109*, 28006.
98. Caravelli, F. Trajectories Entropy in Dynamical Graphs with Memory. *Frontiers in Robotics and AI* **2016**, *3*.
99. Sheldon, F.C.; Di Ventra, M. Conducting-insulating transition in adiabatic memristive networks. *Physical Review E* **2017**, *95*, 012305.
100. Veberic, D. Having Fun with Lambert W(x) Function, [1003.1628].
101. Berdan, R.; Vasilaki, E.; Khiat, A.; Indiveri, G.; Serb, A.; Prodromakis, T. Emulating short-term synaptic dynamics with memristive devices. *Scientific Reports* **2016**, *6*, 18639.
102. Krestinskaya, O.; Dolzhikova, I.; James, A.P. Hierarchical Temporal Memory using Memristor Networks: a survey, [1805.0292].
103. Caravelli, F. The mise en scéne of memristive networks: effective memory, dynamics and learning. *International Journal of Parallel, Emergent and Distributed Systems* **2018**, *33*, 350–366.
104. Caravelli, F.; Barucca, P. A mean-field model of memristive circuit interaction. *EPL (Europhysics Letters)* **2018**, *122*, 40008.
105. Caravelli, F. Asymptotic behavior of memristive circuits and combinatorial optimization, [1706.03001].
106. Boros, E.; Hammer, P.L.; Tavares, G. Local search heuristics for Quadratic Unconstrained Binary Optimization (QUBO). *Journal of Heuristics* **2007**, *13*, 99–132.
107. Hu, S.; et al.. Associative memory realized by a reconfigurable memristive Hopfield neural network. *Nat. Comm.* **2015**, *6*.
108. Kumar, S.; et al.. Chaotic dynamics in nanoscale NbO2 Mott memristors for analogue computing. *Nature* **2017**, *548*, 318–321.
109. Tarkov, M. Hopfield Network with Interneuronal Connections Based on Memristor Bridges. *Advances in Neural Networks* **2016**, pp. 196–203.

110. Pershin, Y.V.; Di Ventra, M. Solving mazes with memristors: A massively parallel approach. *Physical Review E* **2011**, *84*, 046703.
111. Pershin, Y.V.; Di Ventra, M. Self-organization and solution of shortest-path optimization problems with memristive networks. *Physical Review E* **2013**, *88*, 013305.
112. Adlerman, L.M. Molecular Computation of Solutions to Combinatorial Problems. *Science* **1994**, *266*, 1021–1024.
113. Adamatzky, A. Computation of shortest path in cellular automata. *Mathematical and Computer Modelling* **1996**, *23*, 105–113.
114. Prokopenko, M. Guided self-organization. *HFSP J.* **2009**, *3*, 287–289.
115. Williams, S.; et al.. A hybrid nanomemristor/transistor logic circuit capable of self-programming. *Proc. Nat. Acad. Sci.* **2009**, *106*, 1699–1703.
116. Di Ventra, M.; Pershin, Y.V.; Chua, L.O. Circuit Elements With Memory: Memristors, Memcapacitors, and Meminductors. *Proceedings of the IEEE* **2009**, *97*.
117. Traversa, F.L.; Ventra, M.D. Universal Memcomputing Machines. *IEEE Transactions on Neural Networks and Learning Systems* **2015**, *26*, 2702–2715.
118. Nugent MA, M.T. AHaH Computing–From Metastable Switches to Attractors to Machine Learning. *PLoS ONE* **2014**, *9*.
119. Hurley, P. *A Concise Introduction to Logic*, 12 ed.; Cengage Learning: Cambridge, UK, 2015.
120. Gale, E.; de Lacy Costello, B.; Adamatzky, A. Boolean Logic Gates from a Single Memristor via Low-Level Sequential Logic. Unconventional Computation and Natural Computation; Mauri, G.; Dennunzio, A.; Manzoni, L.; Porreca, A.E., Eds.; Springer Berlin Heidelberg: Berlin, Heidelberg, 2013; pp. 79–89.
121. Traversa, F.L.; Di Ventra, M. Polynomial-time solution of prime factorization and NP-complete problems with digital memcomputing machines. *Chaos* **2017**, *27*.
122. Traversa, F.L.; et al.. Evidence of an exponential speed-up in the solution of hard optimization problems, [1710.09278].
123. Traversa, F.; Di Ventra, T. Memcomputing: Leveraging memory and physics to compute efficiently. *Jour. App. Phys.* **2018**, *123*.
124. Sah, M.P.; Hyongsuk Kim.; Chua, L.O. Brains Are Made of Memristors. *IEEE Circuits and Systems Magazine* **2014**, *14*, 12–36.
125. Markin, V.S.; Volkov, A.G.; Chua, L. An analytical model of memristors in plants. *Plant Signaling & Behavior* **2014**, *9*, e972887.
126. Hodgkin, A.L.; Huxley, A.F. Currents carried by sodium and potassium ions through the membrane of the giant axon of Loligo. *The Journal of Physiology* **1952**, *116*, 449–72.
127. Pershin, Y.V.; La Fontaine, S.; Di Ventra, M. Memristive model of amoeba's learning. *Phys. Rev. E* **2010**, *80*.
128. Saigusa, T.; et al.. Amoebae Anticipate Periodic Events. *Phys.Rev. Lett.* **2008**, *100*.
129. Yang, J.J.; et al.. Memristive switching mechanism for metal/oxide/metal nanodevices. *Nat. Nano.* **2008**, *3*.
130. Pershin, Y.V.; Di Ventra, M. Experimental demonstration of associative memory with memristive neural networks. *Neural Networks* **2010**, *23*, 881–886.
131. Tan, Z.H.; Yin, X.B.; Yang, R.; Mi, S.B.; Jia, C.L.; Guo, X. Pavlovian conditioning demonstrated with neuromorphic memristive devices. *Scientific Reports* **2017**, *7*, 713.
132. Turcotte, D.L. Self-organized criticality. *Reports on Progress in Physics* **1999**, *62*, 1377–1429.
133. Jensen, H.J. *Self-Organized Criticality*; Cambridge University Press: Cambridge, 1998.
134. Marković, D.; Gros, C. Power laws and self-organized criticality in theory and nature. *Physics Reports* **2014**, *536*, 41–74, [1310.5527].
135. Alava, M.J.; Nukala, P.K.V.V.; Zapperi, S. Statistical models of fracture. *Advances in Physics* **2006**, *55*, 349–476.
136. Widrow, B. An Adaptive 'Adaline' Neuron Using Chemical 'Memistors'. Technical Report 1553-2, Stanford Electronics Laboratories, 1960.
137. Adhikari, S.P.; Kim, H. Why Are Memristor and Memistor Different Devices? In *Memristor Networks*; Adamatzky, A.; Chua, L., Eds.; Springer International Publishing: Cham, 2014; pp. 95–112.
138. Cai, W.; Tetzlaff, R. Why are memristor and memristor different devices? In *Memristor Networks*; Adamatzky, A, C.L., Ed.; Springer, 2014; pp. 113–128.
139. DeBenedictis, E.P. Computational Complexity and New Computing Approaches. *Computer* **2016**, *49*, 76–79.

140. Alibart, F.; Zamanidoost, E.; Strukov, D.B. Pattern classification by memristive crossbar circuits using ex situ and in situ training. *Nature Communications* **2013**, *4*, 2072.
141. Tissari, J.; et al.. K-means clustering in a memristive logic array. *Proc. IEEE 15th (IEEE-NANO)* **2015**.
142. Merkel, C.; Kudithipudi, D. Unsupervised Learning in Neuromemristive Systems, [1601.07482].
143. Jeong, Y.; Lee, J.; Moon, J.; Shin, J.H.; Lu, W.D. K-means Data Clustering with Memristor Networks. *Nano Letters* **2018**, *18*, 4447–4453.
144. Widrow, B.; Lehr, M. 30 years of adaptive neural networks: perceptron, Madaline, and backpropagation. *Proceedings of the IEEE* **1990**, *78*, 1415–1442.
145. Soudry, D.; Di Castro, D.; Gal, A.; Kolodny, A.; Kvatinsky, S. Memristor-Based Multilayer Neural Networks With Online Gradient Descent Training. *IEEE Transactions on Neural Networks and Learning Systems* **2015**, *26*, 2408–2421.
146. Rozell, C.J.; Johnson, D.H.; Baraniuk, R.G.; Olshausen, B.A. Sparse Coding via Thresholding and Local Competition in Neural Circuits. *Neural Computation* **2008**, *20*, 2526–2563.
147. Sheridan, P.M.; Cai, F.; Du, C.; Ma, W.; Zhang, Z.; Lu, W.D. Sparse coding with memristor networks. *Nature Nanotechnology* **2017**, *12*, 784–789.
148. Sanger, T.D. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks* **1989**, *2*, 459–473.
149. Choi, S.; Sheridan, P.; Lu, W.D. Data Clustering using Memristor Networks. *Scientific Reports* **2015**, *5*, 10492.
150. Vergis, A.; Steiglitz, K.; Dickinson, B. The complexity of analog computation. *Mathematics and Computers in Simulation* **1986**, *28*, 91–113.
151. Ercsey-Ravasz, M.; Toroczkai, Z. Optimization hardness as transient chaos in an analog approach to constraint satisfaction. *Nature Physics* **2011**, *7*, 966–970.
152. Wang, Z.; et al.. Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing. *Nat. Mat.* **2017**, *16*, 101–108.
153. Moss, F. Stochastic resonance and sensory information processing: a tutorial and review of application. *Clinical Neurophysiology* **2004**, *115*, 267–281.
154. McDonnell, M.D.; Abbott, D. What is stochastic resonance? Definitions, misconceptions, debates, and its relevance to biology. *PLoS computational biology* **2009**, *5*, e1000348.
155. McDonnell, M.D.; Ward, L.M. The benefits of noise in neural systems: bridging theory and experiment. *Nature reviews. Neuroscience* **2011**, *12*, 415–26.
156. Stotland, A.; Di Ventra, M. Stochastic memory: Memory enhancement due to noise. *Physical Review E* **2012**, *85*, 011116.
157. Slipko, V.A.; Pershin, Y.V.; Di Ventra, M. Changing the state of a memristive system with white noise. *Physical Review E* **2013**, *87*, 042103.
158. Patterson, G.A.; Fierens, P.I.; Grosz, D.F. Resistive Switching Assisted by Noise. In *Understanding Complex Systems*; Springer, 2014; pp. 305–311.
159. Fu, Y.X.; Kang, Y.M.; Xie, Y. Subcritical Hopf Bifurcation and Stochastic Resonance of Electrical Activities in Neuron under Electromagnetic Induction. *Frontiers in Computational Neuroscience* **2018**, *12*.
160. Feali, M.S.; Ahmadi, A. Realistic Hodgkin–Huxley Axons Using Stochastic Behavior of Memristors. *Neural Processing Letters* **2017**, *45*, 1–14.
161. Ovshinsky, S.R. Reversible Electrical Switching Phenomena in Disordered Structures. *Physical Review Letters* **1968**, *21*, 1450–1453.
162. Neale, R.G.; Nelson, D.L.; Moore, G.E. Nonvolatile and reprogrammable, the read-mostly memory is here. *Electronic* **1970**.
163. Buckley, W.; Holmberg, S. Electrical characteristics and threshold switching in amorphous semiconductors. *Solid-State Electronics* **1975**, *18*, 127–147.
164. Ielmini, D.; Lacaita, A.L. Phase change materials in non-volatile storage. *Materials Today* **2011**, *14*, 600–607.
165. Campbell, K.A. Self-directed channel memristor for high temperature operation. *Microelectronics Journal* **2017**, *59*, 10–14.
166. Hoskins, B.D.; Adam, G.C.; Strelcov, E.; Zhitenev, N.; Kolmakov, A.; Strukov, D.B.; McClelland, J.J. Stateful characterization of resistive switching TiO2 with electron beam induced currents. *Nature Communications* **2017**, *8*, 1972.

167. Chernov, A.A.; Islamov, D.R.; Pik'nik, A.A. Non-linear memristor switching model. *Journal of Physics: Conference Series* **2016**, *754*, 102001.
168. Balatti, S.; et al.. Voltage-Controlled Cycling Endurance of HfOx -Based Resistive-Switching Memory. *IEEE Trans. Elect. Dev.* **2015**, *62*.
169. Hamed, E.M.; Fouda, M.E.; Radwan, A.G. Conditions and Emulation of Double Pinch-off Points in Fractional-order Memristor. 2018 IEEE International Symposium on Circuits and Systems (ISCAS), 2018, pp. 1–5.
170. Stieg, A.Z.; Avizienis, A.V.; Sillin, H.O.; Aguilera, R.; Shieh, H.H.; Martin-Olmos, C.; Sandouk, E.J.; Aono, M.; Gimzewski, J.K. Self-organization and Emergence of Dynamical Structures in Neuromorphic Atomic Switch Networks. In *Memristor Networks*; Adamatzky, A.; Chua, L., Eds.; Springer International Publishing: Cham, 2014; pp. 173–209.
171. Sillin, H.O.; Aguilera, R.; Shieh, H.H.; Avizienis, A.V.; Aono, M.; Stieg, A.Z.; Gimzewski, J.K. A theoretical and experimental study of neuromorphic atomic switch networks for reservoir computing. *Nanotechnology* **2013**, *24*, 384004.
172. Stieg, A.Z.; Avizienis, A.V.; et al.. Emergent Criticality in Complex Turing B-Type Atomic Switch Networks. *Adv. Mater.* **2012**, *24*, 286–293.
173. Wang, X.; et al. Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing. *Nat. Mat.* **2017**, *16*.
174. Scharnhorst, K.S.; Carbajal, J.P.; Aguilera, R.C.; Sandouk, E.J.; Aono, M.; Stieg, A.Z.; Gimzewski, J.K. Atomic switch networks as complex adaptive systems. *Japanese Journal of Applied Physics* **2018**, *57*, 03ED02.
175. Wen, X.; Xie, Y.T.; Mak, M.W.C.; Cheung, K.Y.; Li, X.Y.; Renneberg, R.; Yang, S. Dendritic nanostructures of silver: Facile synthesis, structural characterizations, and sensing applications. *Langmuir* **2006**, *22*, 4836–4842.
176. Pickett, M.D.; Strukov, D.B.; Borghetti, J.L.; Yang, J.J.; Snider, G.S.; Stewart, D.R.; Williams, R.S. Switching dynamics in titanium dioxide memristive devices. *Journal of Applied Physics* **2009**, *106*, 074508.
177. Chang, T.; Jo, S.H.; Kim, K.H.; Sheridan, P.; Gaba, S.; Lu, W. Synaptic behaviors and modeling of a metal oxide memristive device. *Applied Physics A* **2011**, *102*, 857–863.
178. International Technology Roadmap for Semiconductors. https://www.semiconductors.org/clientuploads/Research_Technology/ITRS/2011/2011ExecSum.pdf. Last accessed on 2018-06-30.
179. Grollier, J.; Querlioz, D.; Stiles, M.D. Spintronic Nanodevices for Bioinspired Computing. *Proceedings of the IEEE* **2016**, *104*, 2024–2039.
180. Xiaobin Wang.; Yiran Chen.; Haiwen Xi.; Hai Li.; Dimitrov, D. Spintronic Memristor Through Spin-Torque-Induced Magnetization Motion. *IEEE Electron Device Letters* **2009**, *30*, 294–297.
181. Sun, J.Z. Spin-current interaction with a monodomain magnetic body: A model study. *Physical Review B* **2000**, *62*, 570–578.
182. Hopfield, J.J. Neural networks and physical systems with emergent collective computational abilities. *Proc. of the National Academy of Sciences* **1982**, *79*, 2554–2558.
183. Hopfield, J.J.; Tank, D.W. Computing with neural circuits: a model. *Science* **1986**, *233*, 625–633.

© 2018 by the authors. Submitted to *Technologies* for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).