technologies

MDPI

Perspective

Memristors for the Curious Outsiders

Francesco Caravelli [1,*] and Juan Pablo Carbajal [2,3]

[1] Theoretical Division and Center for Nonlinear Studies, Los Alamos National Laboratory, Los Alamos, NM 87545, USA
[2] Institute for Energy Technology, HSR (Hochschule für Technik Rapperswil) or University of Applied Sciences Rapperswil, 8640 Rapperswil, Switzerland; juan.pablo.carbajal@hsr.ch
[3] Swiss Federal Institute of Aquatic Science and Technology, Eawag, Überlandstrasse 133, 8600 Dübendorf, Switzerland
* Correspondence: caravelli@lanl.gov

Received: 28 September 2018; Accepted: 3 December 2018; Published: 9 December 2018

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-terminal passive component with a dynamic resistance depending on an internal parameter. We provide an brief historical introduction, as well as an overview over the physical mechanism that lead to memristive behavior. 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

1. Introduction

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

The memristor was introduced as a device that "behaves somewhat like a nonlinear resistor with memory" [1], the resistance of the component depends on the history of the applied inputs: voltage or current. Different curves of the applied input elicit a different dynamic response and final resistance of the memristor. In addition, if we remove the input after certain time, leave the component alone, and come back to use it, the device will resume its operation from a resistance very similar to the one in which we left it; that is, they act as non-volatile memories. Furthermore, memristors also react to the direction of the current, i.e., they have polarity. This polarity is shown explicitly on the symbol of the memristor used in electronic diagrams: a square signal line (as opposed to a zigzag line for the resistor), inside an asymmetrically filled rectangle, as in Figure 1.

Figure 1. Electronic symbol of the memristor.

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 present article gravitates 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 complementary metal–oxide–semiconductor (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 an article titled "If it's pinched it's a memristor" [2]. The title refers to the fact that the voltage drop across the device shows a hysteresis loop pinched to the zero of the voltage–current axes when controlled with a sinusoidal voltage or current. The claim by Chua is that a device which satisfies this property must necessarily be a memristor. Chua also proved that such a device cannot be obtained from the combination of nonlinear capacitors, inductors and resistors [1–3]. It has been shown, however, that this property is a modeling deficiency for redox-based resistive switches [4].

Resistive switching was of interest even before the 2008 article. For instance, the review of 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. Next, we give a list of established and new companies developing memristors technology using a variety of compounds. The list is given without a particular order as these companies work on various types of compounds and type of memristors (for instance both resistive random-access memory (ReRam) and phase-change material (PCM)). Specifically, we are aware of: HP, SK Hynix, IBM, IMEC, Fujitsu, Samsung, SMIC, Sharp, TSMC, NEC, Panasonic, Macromix, Crossbar Inc., Qimonda, Ovonyx, Samsung, Intel, KnowM, 4DS Memories Ltd., Global Foundaries, Western Digital (before called SanDisk), Toshiba, Macronix, Nanya, NEC, Rambus, ST Microelectronics, Winbond, Adesto Technologies Corporation, HRL Laboratories LLC, and Elpida.

Going into the many physical mechanisms that make a memristive device work does require a deep knowledge of material properties. For example, the hysteretic behavior can be caused by Joule heating, as shown in an example taken from macroscopic granular materials [5]. In addition, hysteresis is a phenomenon that is common in nanoscale devices and it can be derived using Kubo response theory [6]. Kubo response theory is used to calculate 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 materials, the current or the applied voltage triggers 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. For instance, in Mott memristors, the resistive switching is due to the quantum phenomenon known as Mott insulating-conducting transition in metals, which changes the density of free electrons in the material.
- Electrochemical filament growth mechanism: in these materials, the applied voltage induces filament growth from the anode to the cathode of the device, thus reducing or increasing the resistance;
- Spin-torque: the quantum phenomenon of resistance change 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.

The primary application of memristors, as we will show in this article, is towards neuromorphic computing. The word neuromorphic was coined by Carver Mead [7] to describe analog circuits which can mimic the behavior of biological neurons. In the past several 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 memristors 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, and integration to mention just 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. In addition, 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 W. Besides their role in biological neural models (Section 6), we also discuss theoretical approaches to memristive circuits and their connection to machine learning (Sections 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 is (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 [8] (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 [9,10].

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 long-term storage of information. The neat separation between memory (RAM and HD) and computing (the processors or central processing unit (CPU)) is a key feature of computation using von Neumann architecture. It requires that the data is split into packets, transferred to the CPU where computation is performed, and then repeated until the full computation task has been completed. As far as our understanding goes, this separation is not present in the brain, in which memory and processing happen within the same units. Several proposed architectures that mimic this property have been proposed, most of them based on memristors [11]. 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 (Section 2). We then provide the key ideas behind the technology with simple mathematical models (Section 3). Albeit separated from the main text, we have provided an introduction to the main technology and physical principles underlying memristors in Appendix A. We then focus our gaze on the description and use of memristors both for data storage (Section 4) and data processing (Section 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 (field-programmable gate array) 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 memristors (Sections 5.2–5.3). We discuss in Section 6 computation as an epiphenomenon of memristor dynamics and in connection with CMOS. 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 [12], dated to the late 19th century. Another example, which was key to the discovery of the radio, is the coherer [13,14]. The coherer was invented by Branly [15] after the studies of Calzecchi-Onesti, and inspired Marconi's radio receiver [16]. Branly's coherer serves as perfect homemade memristor [5], 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 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 [17]. While this discussion is important, we 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; other models like the Monetary National Income Analogue Computer (or MONIAC) [18], would be later employed by the Bank of England to perform economic forecasts.

The idea behind these computers was to use hydraulics 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 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 memristors do, depending on the physical mechanism which induces the switching. Other memristor-like systems can be built with other mechanical analogs [19,20], plants and potato tubes [21] or slime moulds [22,23] 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 [1]. During the 1960s, 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 (Institute of Electrical and Electronics Engineers) Kirchhoff Award. The definition of memristor was made clearer in a second paper with his student at the time, Sung Mo Kang [3]. The second work was an important generalization of the notion of memristor and is the one we used in the present paper. Chua and Sung Mo 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 by Chua and Sung Mo Kang, is a resistance whose value depends on some internal parameter, which in turn has to evolve dynamically according either to current or voltage. Implicitly, one needs to also define the relationship between the resistance and the state variable, which characterizes the device resistance. In the analogy 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.

For the 30 years after the work by Chua and Sung Mo Kang, the field of memristors was basically non-existent. During the mid 2000s, Strukov, Williams and collaborators, working at Hewlett-Packard Labs (Palo Alto, CA, USA) were studying oxide materials and resistive switching. The physics of resistive switching has been known since the early 1970s, with the introduction of Phase-Change Materials. The physical origin of resistive switching was well studied, albeit not fully understood [24–26]. The idea that memristors could be experimentally realized became popular with the article of Strukov et al. at HP Labs.

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 memory dynamics proposed by Strukov et al. This is one of the simplest memristors models that captures the behaviors relevant for technological applications of memristive systems.

The fundamental 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:

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

These relations introduce the resistance R, the capacitance C, and the inductance L. The ideal memristor was initially and abstractly formulated as the flux–charge 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. For the sake of simplicity we do not make a difference between current-controlled and charge-controlled; this is an abuse of the language. The units of the coupling M are Ohms, as can be seen by the units of the quantities it relates, which justifies the notion of memristor at least as a nonlinear resistor. However, according to Chua, only nonlinear resistors showing a pinched hysteresis loop, i.e., $V = 0 \rightarrow I = 0$, are classified as memristors. An example is shown in Figure 2. Further classifications (ideal, extended, generic, see [2]) can be added to a device depending on other properties of the current–voltage (I–V) diagram that are observed experimentally when the device is controlled with a sinusoidal voltage at a certain frequency. For example, if for high frequencies the I–V becomes a straight line (recovery of pure linear resistance, no hysteresis), the device is a generic memristor, under Chua's nomenclature. General memristors might be locally-active, showing regions where the resistance changes sign [27].

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

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

where 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 $H(x,u)$ is the called memristance, and it is called memductance in the second case (voltage-controlled). The function $H(x,u)$ is assumed to be continuous, bounded, and of constant sign. In order to rule out locally active devices, it is necessary to further require that $H(x,u)$ is monotonic in u. This leads to the zero-crossing property or pinched hysteresis: $u = 0 \Rightarrow y = 0$. The states 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 $H(x,u)$ depends is called the *order* of the memristive system.

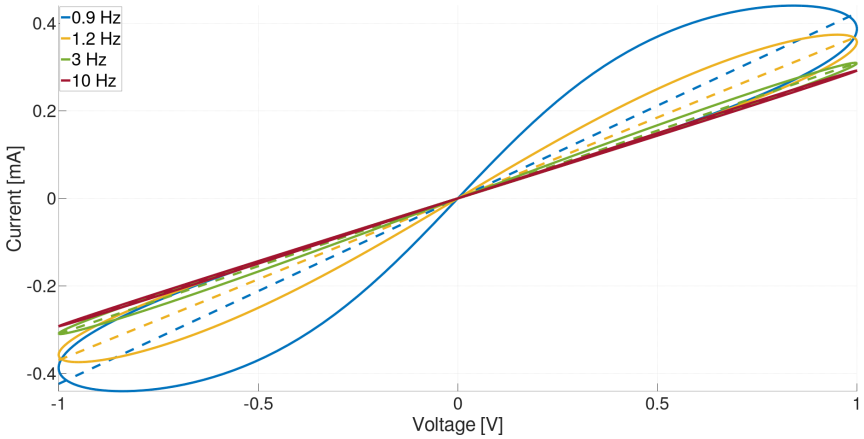


Figure 2. Pinched hysteresis loop of an ideal memristor. The parameters of the model are $\alpha = 0$, $\beta = 0.3$ mA, $R_{\text{on}} = 1$ kΩ and $R_{\text{off}} = 6$ kΩ. The driving voltage is $V(t) = \sin(2\pi f t)$ with f taking several values. We see that, for increasing frequencies, the hysteresis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

The reader should not overlook the difference between an ideal memristor (a mathematical model), memristive systems (also a mathematical model), and a memristor (a physical device). Ideal memristors are models of devices in which the internal parameter controlling the resistance is the charge, which is linked to the flux. Memristive systems generalize this behavior, and the internal parameters can be other physical quantities, or combination of physical quantities, even if they do not include flux or charged particles. The latter is better attuned to many memristors, i.e., physical realizations of memristive behavior.

Among the (large) class of memristive systems, the model proposed by Strukov et al. is appealing due to its simplicity. This model, shown in Equations (4) and (5), captures the basic properties of memristors that are relevant for the understanding of the device and for its technological applications.

Equations (4) and (5) model the behavior of a current-controlled Titanium Dioxide memristor [28], also known as the HP-memristor:

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

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

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

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 (with units s^{-1}) and β (with units $\mathrm{s}^{-1}\mathrm{A}^{-1}$), 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 [29]. Since the resistance function depends on a single state w, this is a first order memristive system. Although this model has several drawbacks when

it comes to its connection to the physical behavior of ion migration in the conductor [6,17,19,30–32], it will be the reference for most derivations and discussions in this article, as it captures several key properties of a generic memristor.

We begin with the case with $\alpha = 0$, i.e., a non-volatile memristor. We solve the system of equations above using a zero mean small amplitude 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 [33]:

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

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

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

whose 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 fundamental property of a memristor: it has the zero crossing property (pinched hysteresis loop). Furthermore, it shows properties of generic memristors: the loop tilts to the right for driving signals with increasing frequencies, becoming a straight line for sufficiently 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 Equation (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 for $\alpha = 0$, 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 [28] that

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

and thus

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

where μ_e is the electron mobility in the film, R_{off} is the resistance of the undoped material (for instance titanium oxides, ~100 Ω), 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 when $\frac{R_{\mathrm{off}}}{R_{\mathrm{on}}} \gg 1$ and β is small enough, the memristor can be quickly driven from one state to the other via a current (or voltage) 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 non-volatile 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 state, $R(t = \infty) = R_{\mathrm{off}}$. That is, if $I(t) = 0$, then the internal memory is simply given by $w(t) = w_0 e^{\alpha t}$, where w_0 is the initial condition. This behavior was first experimentally observed in [34]. Volatility is discussed again in Section 5.4.

Following the distinction between memristive and memristors we have discussed, when $\alpha \neq 0$, the parameter w cannot be uniquely identified with the charge, hence the model does not correspond to a bona fide memristor. In fact, for a single component the solution of:

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

is given by

$$w(t) = \frac{e^{\alpha t}}{\beta} \int_1^t e^{-\alpha \tau} \frac{d}{d\tau} q(\tau)\, d\tau + w_0 e^{\alpha t}, \tag{13}$$

which is a composite function that includes the charge.

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 [28,35–40]. Since we are not aware of any systematic validations of windowing functions with real devices, we believe they should be considered tricks useful for large simulations. It is common to use windowing techniques based on polynomials; for an extensive review, we refer the reader to [38]. Polynomial windowing functions that reproduce nonlinear dopants drift can be introduced via the so-called Joglekar window function $F(x)$ [36,40], such that $F(1) = F(0) = 0$, which generalizes the evolution of $w(t)$ as

$$\frac{\mathrm{d}w(t)}{\mathrm{d}t} = \alpha w(t) - F(w) \frac{1}{\beta} I(t), \tag{14}$$

where the window function can take the form $F(w) = 1 - (2w - 1)^{2p}$, with p a 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 physical mechanisms that can be involved, we refer the reader to Appendix A.

Numerical simulations of memristive networks can be achieved with different methods. Efficient ones include the SPICE methodology [41,42] and the more recent Flux-Charge method [43]. Furthermore, general solvers for Differential-algebraic system of equations (e.g., SUNDIALS solvers [44]) can also be used for networks with a few hundred components, without major loss of performance. For some networks (without capacitance or inductance), it is possible to derive a monolithic system of ordinary differential equations for the evolution of the network states, which already include the nonlocal algebraic constraints imposed by Kirchhoff's laws [45,46]. The choice of the method is largely based on the questions to be addressed by the simulations, which often results in a trade-off between generality of the solver (e.g., flexible memristor model) and performance.

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.

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

$$w(t) = \sqrt{a + bV_{\mathrm{write}} t}, \tag{15}$$

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 vice versa (the flip of a bit). The switching happens within a characteristic time τ:

$$\sqrt{a + bV_{\text{write}}\tau} = 1 \rightarrow \tau \leq \frac{1}{bV_{\text{write}}}. \tag{16}$$

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 via the measurement of the resulting current.

Equation (15) applies only to the idealized memristor described by the model in Equations (4) and (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. Figure 3 shows the response of a real TiO_2 memristor to write voltages (SET and RESET).

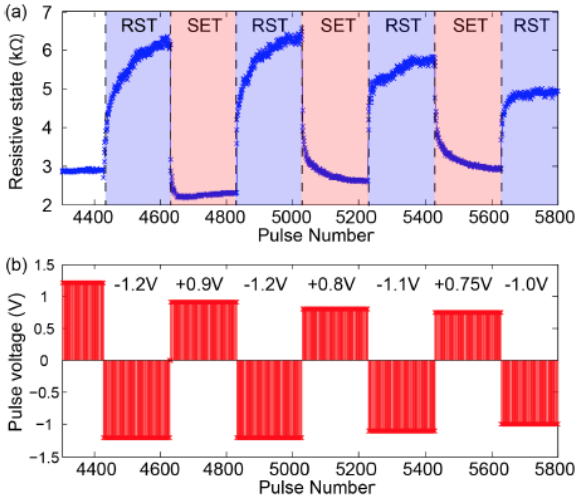


Figure 3. *RESET* and *SET* for TiO_2 with pulse within the hundred of microseconds, reproduced with permission from [47]. The subfigure (**a**) is the evolution of resistance as a function of the voltage, which is shown in subfigure (**b**).

The fact that it is possible to write and read the state via signal pulses allows for advantageous scaling of power consumption and bit density (see [48] 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 lithography (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 [49].

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 Table 1.

Table 1. Characteristics of various storage devices (for details see [50]). The table compares to memristor technology to competitors like Phase-Change Materials (PCM) and more standard devices based on Spin-Transfer Torque (STT), dynamic random-access memory (DRam), Flash and hard disk (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 (years)	10	10	10^{-1}	10^{-5}	10	10
Cycles endurance	10^{12}	10^{7-8}	10^{15}	10^{17}	10^{5-8}	10^{15}
3D capability	yes	no	no	no	yes	n/a

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 Figure 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-terminal component, e.g., a memristor, is connected across each pair of vertical and horizontal lines.

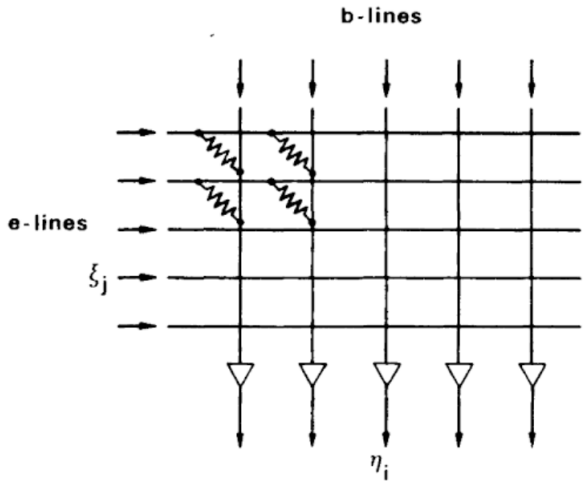


Figure 4. Learning matrix, introduced by Steinbuch [51], reproduced with permission from [52].

In order to operate 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 $M_{j,i}$, placed across j-th e-line and 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" (the learning matrix) [51,53].

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:

$$A_{ij} = \frac{M_{ij}^{-1}}{\tilde{R}_i^{-1} + \sum_{s=1} M_{is}^{-1}}. \tag{17}$$

$\tilde{R}_i$ 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 [54]. To apply this method with memristors, the voltages differences need to be small/short,

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 various layers of crossbar arrays on top of each other [55,56]. The stacking of L crossbar layers scales the density of components by a 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) [57]. 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 (Figure 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 [57], 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 [58] and differential pair synapses (called "kT-RAM") [59].

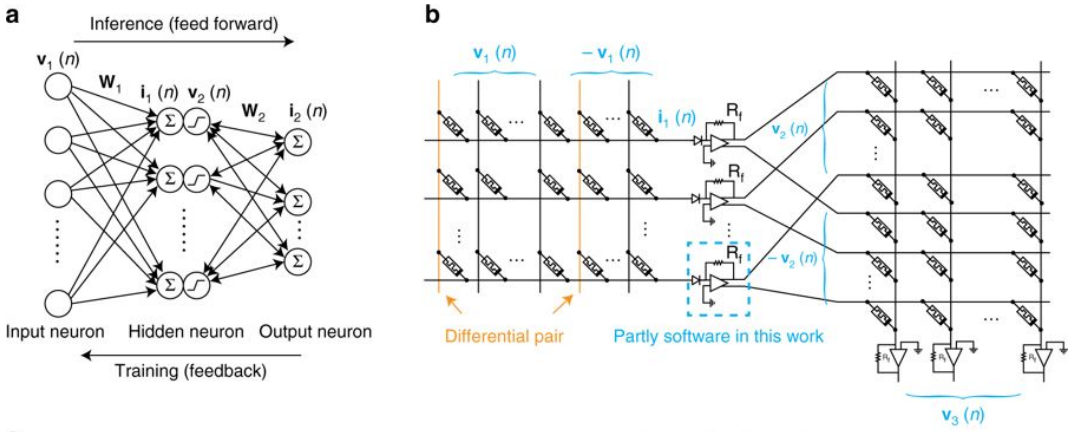


Figure 5. Memristors used for synaptic weights in artificial neural networks (ANNs), reproduced with permission from Figure 2 of [57]. Subfigure (**a**): memristive neural network. In Subfigure (**b**) we introduce the crossbar implementation of the memory using memristors.

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 fed to the neurons. This architecture minimizes the reading-writing costs incurred by the conventional von Neumann architecture, and potentially 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, spike-timing-dependent, short-term, long-term, etc. In the study of synaptic plasticity of the human cortex, Hebbian or Anti-Hebbian (related to the simultaneous firing of two neurons) [52,60,61] are often the underpinning learning mechanisms. Our aim here is not to describe the types of plasticity, their biological origin, or how difficult it is to isolate each class in biological and experimental systems. Our intention is but to illustrate how memristors, used as memories, can be used to 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 [62].

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

$$f_X : X \to W_\Delta \subset \mathbb{R}. \tag{18}$$

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.

In order 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

$$f_V : W_\Delta \to V, \tag{19}$$

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 [62] (Section 4.1). For example, Reference [63] uses STDP to implement unsupervised learning with resistive 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., on which we base the following sentences. The reader interested in hardware implementations of STDP should 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 online in a single integrated phase (as opposed to offline 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 variability of the units, which more general neuromorphic circuits are able to exploit [64].

5. Memristors for Data Processing

Device variability (Section 4) and volatility (Section 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 believe that evolution exploits ideal systems in reference to its own architecture 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 are 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 [65], nonlinear transients, and variability [66] (Section 7.5), to implement learning methods with memristors. To put these methods within an unifying framework, 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

In order 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 [67] for a complete discussion and historical overview) [68,69].

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 [70]. For example, in the current digital computer, computation is carried out 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.

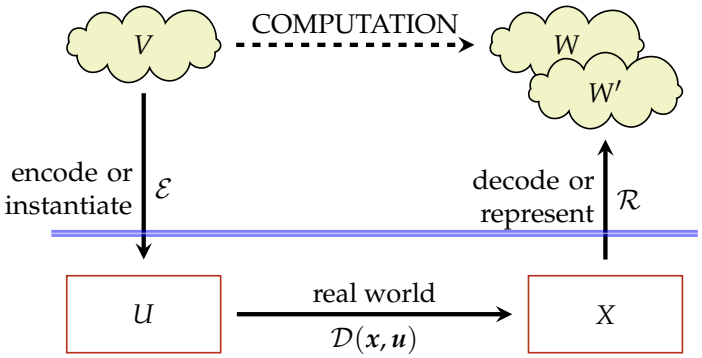


Figure 6. Computing with dynamical systems adapted from [70]. Conceptual depiction of the sets and transformations involved in a typical computation using a dynamical system $\mathcal{D}$. In this case, the computation is defined by its action on the input-output set V, W. The inputs $\boldsymbol{u}(t) \in U$ to $\mathcal{D}$ are encoded or instantiated versions of V through the transformation $\mathcal{E}$. The output of the dynamical system $x(t) \in X$ 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. Programming a computer implies first encoding our input to the processing unit and then decoding the output to obtain the result of the computation. These 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 (Section 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 to 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 [71]. The encoding-decoding pair is also linked to the "natural basis of computation" [72] 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 a universal model of computation able to solve all kinds of computation problems, as conjectured by the Church–Turing(–Deutch) thesis [73]. This is achieved by a precise control of each step of the computation and the way the computer transforms (or evolves) its states. Another advantage of modern 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 [74] 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 [67] (Section "Analog Computation and the Turing Limit"). Besides this 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. It is possible to achieve a significant speedup in computational time, usually at the expense of other quantities; for instance [75] reported an speedup for nondeterministic polynomial time (NP)-Complete problems at an exponential cost in energy.

As an example of an analog computer, let us consider a hydrostatic polynomial root finder proposed by Levi [76]. Consider the following polynomial equation:

$$\sum_{i=1}^{n} x^i c_i + c_0 = 0. \tag{20}$$

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, let us consider the case $n = 2$. In Figure 7, we show the design of a hydrostatic root solver. On one side of a pivoting lever, a certain 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 $c_2 x^2$). On the other side of the lever, we set a weight for the constant term. When the device is submerged into the water holding the pivoting point, the depth at which the device equilibrates (i.e., becomes horizontal) is a solution of the polynomial equation.

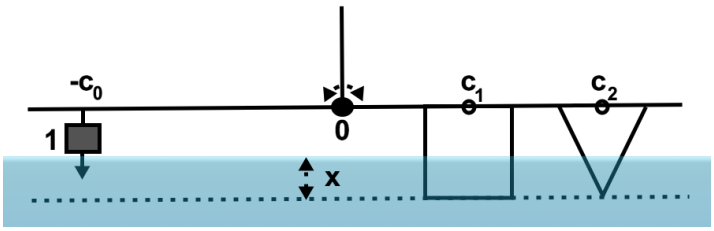


Figure 7. The hydrostatic root solver. Shapes encode the polynomial coefficients. A weight encodes the constant term. The depth at which torque is zero is a solution of the polynomial equation.

The example above illustrates 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 with mentioning 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 [77] (Section 2.7), use the spins of a physical system and the computation of a quantum Ising model to solve the optimization [78]. Deoxyribonucleic acid (DNA) computing exploits the self-assembly and preferential attachment of DNA strands to encode tiling systems with Wang tiles [79,80]. Slime mold [23] 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 distribution of mold cells in the Petri dish. Ant colony optimization [81] is inspired by the behavior of their natural counterpart and uses digital models of pheromone dynamics for computation (see also stigmergy [82]).

In the subsequent sections, we describe in some technical details the 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 y (outputs) via an algebraic relation is linear regression: $y = X^\top \boldsymbol{\beta}$. 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 transformation to the inputs to obtain a new linear model $y = G(X)^\top \boldsymbol{\gamma}$, where only the coefficient vector $\boldsymbol{\gamma}$ (or matrix when the output is not scalar) is learned from the data. This nonlinear method is a linear regression in a space generated by nonlinear transformations of the input (see kernel methods [83,84]). In neuromorphic computation, the transformation is commonly given a particular structure by choosing a set of N_g nonlinear functions and applying it to each input vector:

$$G(X)^\top_{ij} = g_j(x(i)), \quad i = 1, \ldots, N \; j = 1, \ldots, N_g. \tag{21}$$

This method is known as Extreme Learning Machines (ELM) [85] and has been implemented in hardware exploiting the variability of fabricated devices to generate the nonlinear transformations [86,87]. Briefly explained, the differences among the devices generate different outputs for the same input, which are then used as the dictionary $G(X)^\top_{ij}$.

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

$$y(t) = \sum_{k=1}^{N_g} \gamma_k g_k(x(t)), \tag{22}$$

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 $g(x) = \ell^{-1}(x^\top \boldsymbol{\eta})$ (ℓ is the link function) and $N_g = 1$; we obtain:

$$y_i = \gamma_1 \ell^{-1}(x(i)^\top \boldsymbol{\eta}). \tag{23}$$

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 $N_g \gg 1$

$$y_i = \sum_{k=1}^{N_g} \gamma_k \ell^{-1}(x(i)^\top \boldsymbol{\eta}_k) \approx \int_H \ell^{-1}(x(i)^\top \boldsymbol{\eta}) p(\boldsymbol{\eta}) \mathrm{d}\boldsymbol{\eta}, \tag{24}$$

where the set of $\{\boldsymbol{\eta}_k\}$ coefficient vectors are sampled from the prior distribution, expecting that model averaging [88] 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) [89–91], in which the N_g functions are the solution of a differential or difference equations using the data as inputs, e.g.,

$$\boldsymbol{u}(t) = \mathcal{E}(x)(t) \ \in \mathbb{R}^{\dim \boldsymbol{u}}, \tag{25}$$

$$\mathcal{D}_{\boldsymbol{\lambda}} \boldsymbol{q} = B\boldsymbol{u}(t), \ B \in \mathbb{R}^{\dim \boldsymbol{q} \times \dim \boldsymbol{u}}, \tag{26}$$

$$\boldsymbol{g} = H\boldsymbol{q}, \ H \in \mathbb{R}^{N_g \times \dim \boldsymbol{q}}, \tag{27}$$

$$y(t) = \boldsymbol{\gamma}^\top \boldsymbol{g} = \sum_{k=1}^{N_g} \gamma_k g_k(t, \boldsymbol{u}(t), \boldsymbol{\lambda}), \tag{28}$$

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 Figure 6).

The generalization proposed by RC is made obvious with the choice of arguments for the component functions $\{g_k\}$ in Equation (28): 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.

Reservoir computing (RC) allows for machine learning applications using natural or random dynamical systems, as opposed to carefully engineered ones. Its capacity to exploit wildly different physical substrates for computation has been recently highlighted [92]. 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 [93]. Hence, RC implementations using memristive networks has received considerable attention (software [see [65] and references therein] and hardware [94]). In these applications, the memristive system (single device or network) plays the role of the dynamical system $\mathcal{D}_{\boldsymbol{\lambda}}$ (cf. Figure 6), the decoding map is a linear readout of several system states (memories, currents, voltage drops, etc.), and the encoding is usually tailored for the given application at hand.

The case of memristor based RC using the HP-memristor, Equations (4) and (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 [45] (see also Equation (36)). 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. That is, RC heavily relies in the nonlinear (and volatile) behavior of the dynamical system.

5.3. Neural Engineering Framework

The Neural engineering framework (NEF) [66] 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 [66] 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:

$$a_i(f(x)) = G_i\left(I_i(f(x))\right), \tag{29}$$

$$I_i(f(x)) = \alpha_i \langle f(x)\tilde{\phi}_i(x)\rangle + I_i^{\text{bias}}, \tag{30}$$

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:

$$\hat{f}(x) = \sum_i^N a_i(f(x))\phi_i(x), \tag{31}$$

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 [95].

The framework defined in Equations (29)–(31) can realize arithmetic on functions of the input as well as nonlinear transformations. It has also been used to represent linear time-independent systems with neuromorphic hardware [96].

Neural Engineering Framework (NEF), Reservoir Computing (RC), and Extreme Learning Machines (ELM) use the same form of decoding: the output is the scalar product of an input-independent vector with an input-dependent one. The input-dependent vector is given by intrinsic properties of the computing device, it is the result 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 $N_g \ll N_x N$, which are 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 $\{\phi_i(x)\}$ (Equation (31)) are needed to match decoders effective capacity to the capacity of the dynamic responses $\{g_i(x)\}$ (Equation (28)).

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

$$G[I(z)] = \begin{cases} \frac{1}{\tau_0 - \tau_{RC} \log\left(1 - \frac{I_F}{I(z)}\right)}, & I(z) > I_F, \\ 0, & \text{otherwise}, \end{cases} \tag{32}$$

where $I(z)$ is given by Equation (30). A similar functional form can be achieved by a non-volatile memristor with parasitic capacitance, as shown in Equation (35) (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 [97–99] can be observed in networks of memristors [100], although a theoretical understanding of their collective behavior is still poor for both systems [101–103].

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 [40] feed with a controlled current. The memristor is modeled with Equations (4), with $\alpha = 0$, Kirchhoff's voltage law for this circuit gives:

$$R(w)\overbrace{\frac{\mathrm{d}q}{\mathrm{d}t}}^{I(t)} = -\frac{1}{C}\overbrace{q(t)}^{\int I(t)\mathrm{d}t} \rightarrow \frac{\mathrm{d}q}{\mathrm{d}t}(t) = -\frac{q(t)}{R(q(t))C}, \tag{33}$$

$$R(q(t)) = R_{on}\left(1 + \frac{q(t)}{\beta}\right) + R_{off}, \tag{34}$$

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

$$q(t) = \frac{\beta}{\xi} W\left(\frac{\xi}{\beta} e^{-\frac{t}{R_{on}C}} e^{-\frac{c_1}{\beta R_{on}}}\right), \tag{35}$$

where W is the product-log (Lambert) function [104], c_1 is an integration constant, and as before $\xi = \frac{R_{off} - R_{on}}{R_{on}}$. Equation (35) shows that, for large times, the system has a typical exponential RC circuit decay, which is shown in Figure 8 (left). This behavior is observed in experiments [105] where after an external stimuli an exponential-like decay is observed (see Figure 8 (right)).

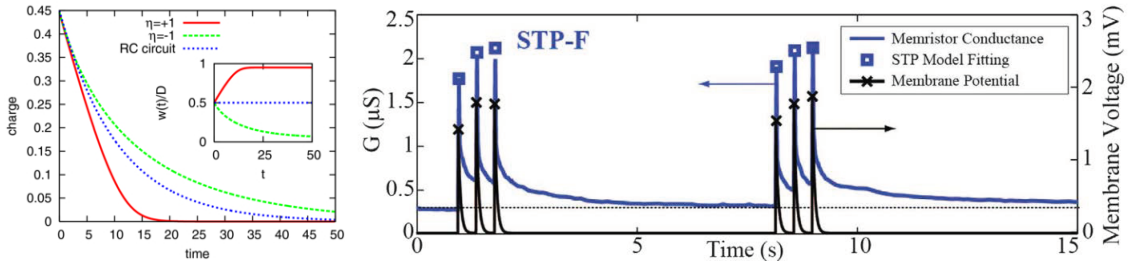


Figure 8. (**Left**): theoretical profile of Equation (35). (**Right**): experimental profile of short-term plasticity with TiO_2. Both figures are reproduced with permission from [40] and [105], respectively.

The limit of a normal RC circuit can be obtained from the above in the limit $\beta \rightarrow \infty$, in which $\lim_{\beta\rightarrow\infty} \beta W(\beta^{-1}G(t)) = G(t)$. The response of a memristor-capacitor and an RC circuit differs,

with the first obtaining a longer retention than the first. In addition, the Lambert function W has several properties of the logarithm function, and thus the response is similar to the one suggested in Equation (35). 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 $\alpha \equiv$ 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 Figure 1 of [34]). A survey of recent hardware designs for temporal memory is provided in [106].

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 networks of current-controlled memristors linear in an internal parameter, we can write down the differential equation describing the evolution of the memory states and obeying Kirchhoff voltage and current laws [45]:

$$\frac{\mathrm{d}\vec{w}}{\mathrm{d}t}(t) = \alpha\vec{w}(t) - \frac{1}{\beta}(I + \frac{R_{\mathrm{off}} - R_{\mathrm{on}}}{R_{\mathrm{on}}}\Omega W)^{-1}\Omega\vec{S}(t), \tag{36}$$

where α and β are the parameters in Equations (4) and (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 Equation (36) is not known. In the approximation $R_{\mathrm{off}} = pR_{\mathrm{on}}$, with p of order one, the equation above can be recast in the form of a (constrained) gradient descent [107], which is reminiscent of the fact that the dynamics of a purely memristive circuit has an approximate Lyapunov function [108,109]. In the simplified setting of purely memristive circuit without any other components, it can be shown that these circuits execute Quadratically Unconstrained Binary Optimization [110]. This idea is in general not recent, and it can be traced back to Hopfield [111–113] for continuous neurons. It is also not the only alternative approach to Hopfield networks using memristors [114–117], as we will also discuss it later.

6. Memristive Galore!

6.1. Memristive Computing

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

In [118,119], 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 Figure 9 will re-organize (when controlled in direct current (DC)) to allow the majority of the current to flow along the shortest path. Although this phenomenon already occurs with regular resistances (with a diode in parallel), 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 values are deeply connected; this fact is reminiscent of the ant-colony optimization algorithms [81], molecular computation [120], and other cellular automata models [121].

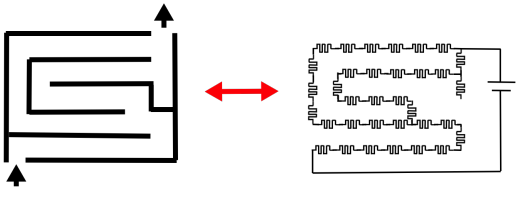


Figure 9. The maze-memristor mapping suggested in [118].

Ideas along these lines can be pushed further in order to explore memristors as complex adaptive systems [122] able to self-organize with the guidance of the circuit topology and the control of external voltages. Using Equation (36), it is possible to obtain approximate solutions, for instance, of the combinatorial Markowitz problem [109]. Hybrid CMOS/Memristive circuits can, in principle, tackle harder problems via a combination of external control and self-organization [123].

In the literature, several models of memristor-based architectures have been proposed. Many 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 [124]) has been introduced recently in [125], 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 [59,126] 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 is a universal computing architecture is to break the system modularly into logic gates based on memristors, 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 [127]). 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 [128,129]). An improvement upon this basic idea is to build input-output agnostic logic gates using memristors. Any port of an agnostic 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 be re-arranged 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 [130,131] (see also [132] and references therein). Similar ideas were recently proposed using nanoscale magnetic materials rather than memristors [133].

The two examples 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 Chua, there have been several reports re-interpreting models of natural information processing systems (neural networks, chemical signaling, etc.) in terms of memristors units. Here, we mention three examples: giant axon of squids [134], electrical networks of some plants [135], and Amoeba adaptation [136].

Squids. 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 (HH) model. The HH 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 Figure 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 [137]. The equations of this model, when put in memristive form, are given by:

$$i_{\mathrm{K}} = \overbrace{g_{\mathrm{K}} w_1^4}^{R_{\mathrm{K}}^{-1}} V_{\mathrm{K}}, \tag{37}$$

$$i_{\mathrm{Na}} = \overbrace{g_{Na} w_2^3 w_3}^{R_{\mathrm{Na}}^{-1}} V_{\mathrm{Na}}, \tag{38}$$

$$\frac{\mathrm{d}w_1}{\mathrm{d}t} = (\mathrm{K}_1 V_{\mathrm{K}} + \mathrm{K}_2) \left[e^{\mathrm{K}_1 V_{\mathrm{K}} + \mathrm{K}_2} - 1 \right]^{-1} (1 - w_1), \tag{39}$$

$$\frac{\mathrm{d}w_2}{\mathrm{d}t} = (\mathrm{Na}_1 V_{\mathrm{Na}} + \mathrm{Na}_2) \left(e^{\mathrm{Na}_1 V_{\mathrm{Na}} + \mathrm{Na}_2} - 1 \right)^{-1} (1 - w_2) + \mathrm{Na}_3 e^{\mathrm{Na}_4 V_{\mathrm{Na}} + \mathrm{Na}_5} w_2, \tag{40}$$

$$\frac{\mathrm{d}w_3}{\mathrm{d}t} = \mathrm{Na}_6 e^{\mathrm{Na}_7 V_{\mathrm{Na}} + \mathrm{Na}_8} (1 - w_3) - \left(e^{\mathrm{Na}_1 V_{\mathrm{Na}} + \mathrm{Na}_9} + 1 \right)^{-1} w_3, \tag{41}$$

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 [134] for details).

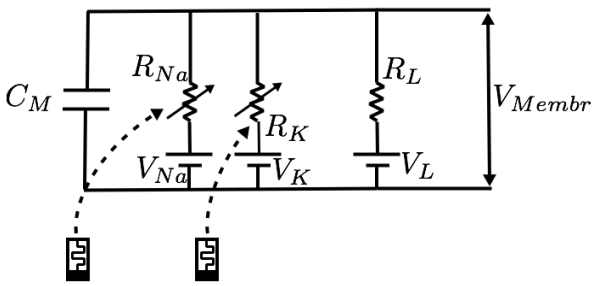


Figure 10. The Hodgkin–Huxley model, with the variable resistances of the Sodium and Potassium channels interpreted as memristors.

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.

Plants. In [135], three types of memristors models were 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 collapse 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:

$$i_m(t) = \frac{e^{\beta t} V(t)}{\beta R_o \int_0^t h(V(x)) e^{\beta x} \mathrm{d}x + A}, \tag{42}$$

$$I = i_m + i_{RC}, \tag{43}$$

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".

Amoeba. A memristive model of amoeba adaptation was introduced in the form of the simple circuit shown in Figure 11 [136,138], 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 [139]:

$$\frac{\mathrm{d}M}{\mathrm{d}t} = f(V_M)\left(\theta(V_M)\theta(M-R_1) + \theta(-V_M)\theta(R_2-M)\right), \tag{44}$$

$$f(V) = \frac{\beta-\alpha}{2}\left(|V+V_T| - |V-V_T|\right) - \beta V, \tag{45}$$

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

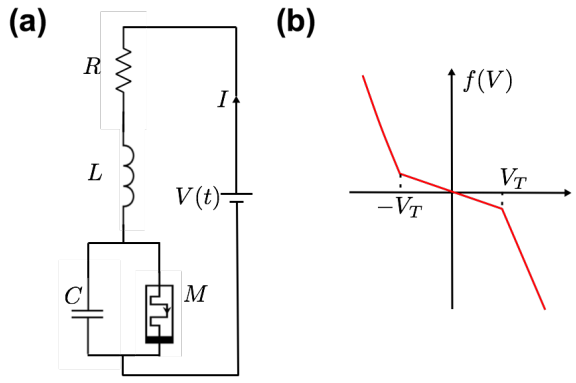


Figure 11. The amoeba memristive learning model of [138]. (**a**) the circuit with capacitance C and memristor M in parallel (with resistance $R(t)$), and in series to a resistance R and an inductance L; (**b**) the function $f(V)$ for the memristor response in voltage.

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, hence 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,

$$CV_C + \frac{V_C}{M(t)} = I. \tag{46}$$

We thus obtain the three coupled differential equations:

$$\frac{\mathrm{d}I}{\mathrm{d}t} = -\frac{R}{L}I + \frac{V-V_C}{L}, \tag{47}$$

$$\frac{\mathrm{d}V_C}{\mathrm{d}t} = -\frac{1}{MC}V_C + \frac{I}{C}, \tag{48}$$

$$\frac{\mathrm{d}M}{\mathrm{d}t} = f(V_M)\left(\theta(V_M)\theta(M-R_1) + \theta(-V_M)\theta(R_2-R)\right). \tag{49}$$

The stationary state requires that all time derivatives are zero. In this state, the circuit is sensitive to new stimuli, i.e., it adapts. For instance, in Figure 12, we see the response of the system to new inputs, and new stationary states are obtained. Albeit amoeba's adaptation are not as developed as in higher mammals, more general models entailing Pavloviav "conditioning" have also been proposed in the literature using memristors [140,141].

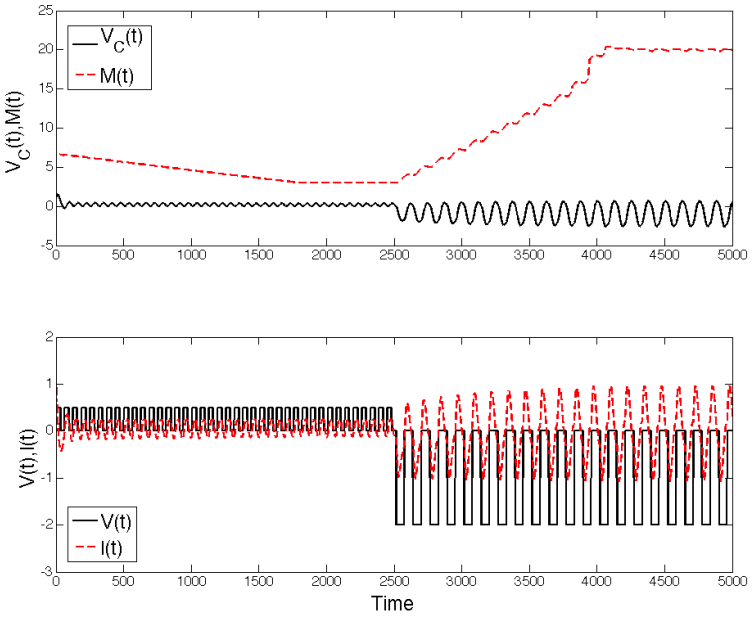


Figure 12. Simulation of Equation (49) and adaptation of the circuit to different stimuli. We consider the parameters $\beta = 100$, $\alpha = 0.1$, $R_1 = 3$, $R_2 = 20$, $C = 1$, $R = 1$, $L = 2$, $V_t = 2.5$ as in [138]. We stimulate the circuit with a square input $V(t) = 0.5$ and frequency $\omega = 10$ 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 $dt = 0.1$.

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 configuration for which a qualitative change in the systems' behavior is imminent (e.g., a bifurcation). These critical configurations are characterized by states with power law cross-correlation functions. 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 [142]. The typical example is, for instance, the Gutenberg–Richter law of earthquakes, whose distribution of magnitude is Richter's law [143,144]. 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. SOC has been suggested to be the underlying mechanism in the observed critical behavior of the brain [98]. 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.

SOC can be produced using large networks of memristors: atomic switch networks are dominated by the interaction due to Kirchhoff laws [100], and thus the observed criticality seems intuitively (power law distribution of the power spectra, for instance) to be connected to a SOC-type [103] 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 [45]. Consider a system of interacting memristors, whose linearized dynamics close to a fixed point obtained with DC voltage stimulation (i.e., saturated memristors: resistors) is written as:

$$\frac{\mathrm{d}\vec{w}(t)}{\mathrm{d}t} = A\vec{w}(t). \tag{50}$$

The matrix A is a non-trivial combination of voltage sources and projectors on the subspace of the circuit's graph [107]. 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

$$\langle w(t) \rangle = \frac{1}{N} \sum_i \left(\sum_j (e^{At})_{ij} w_j^0 \right) = \frac{1}{N} \text{trace} \left(e^{At} W_0 \right) = \frac{1}{N} \text{trace} \left(e^{\lambda^+ t} \tilde{W}^0 + e^{\lambda^- t} \tilde{W}^0 \right). \quad (51)$$

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:

$$\frac{1}{N} \text{trace}(e^{-\lambda_i^- t} \tilde{w}_i) = \int d\lambda \rho^-(\lambda) e^{-\lambda t} \langle w_i^0 \rangle = \frac{1}{2} \int d\lambda \rho^-(\lambda) e^{-\lambda t}, \quad (52)$$

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 [45] 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 networks of HP-memristors, is in line with what is experimentally observed in [100].

The phenomenon above is not classified as SOC, since only a certain matrix A is required 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 [103]. In this case, the criticality is due to the a SOC-like phenomenon which had been already observed in random fuse networks, which is described by a percolation transition [145].

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 [51]. The idea of using instead variables resistances precedes the paper of Steinbuch by one year, and was introduced by Widrow [146] in order to implement the Adaline algorithm explained below. The "**memistor**", a name extremely similar to the one of "**memristor**" introduced ten years later by Chua, was a current-controlled variable resistance [147]. This limited the ability to package a huge number of memistor 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 [148]. 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 [149]. For instance, the circuit proposed in Figure 13 provides a simple circuit which has a linear output neuron controlled by a resistance R, without a 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 [150] to understand the energy efficiency. Using Landauer theory, the energy dissipated by 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

$$E_{dig} \approx 24 \log(\frac{1}{p_{err}}) \log_2^2(L) NkT, \quad (53)$$

$$E_{memr} \approx \frac{1}{24} \log(\frac{1}{p_{err}}) L^2 N^2 kT, \quad (54)$$

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

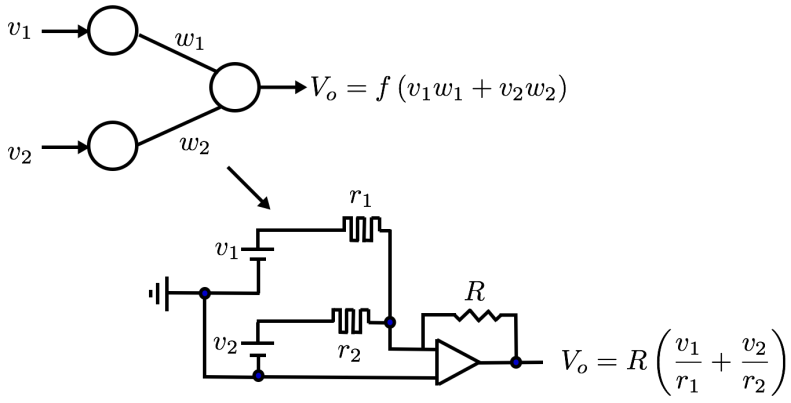


Figure 13. Memristor equivalent of a neural network with three neurons and two synapses, with an output amplifier.

The architecture of Figure 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 [151], first experimental results of memristive technology for pattern classification on a 3×3 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 [65] while unsupervised learning can be performed by implementing the K-means algorithm [152–154].

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) are highly inefficient: one reason for this is the physical separation between the memory 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. As mentioned before, 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 Figure 14.

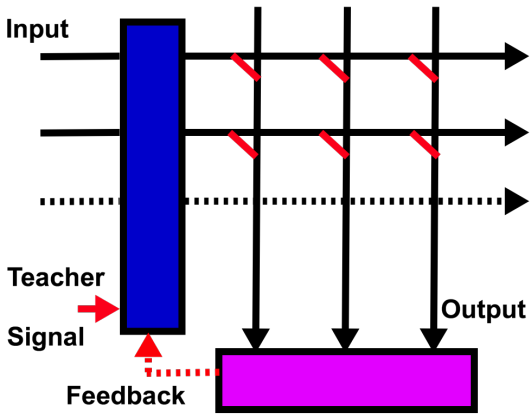


Figure 14. Feedback loop and learning in crossbar arrays.

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

$$W^{k+1} = W^k + f(W^k, Q), \tag{55}$$

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 is called adaline algorithm [60,155]. From the point of view of circuit theory, the feedback can be introduced via CMOS-Memristor integration. For instance, in [156], one way to perform online learning with an adaline algorithm using metal-oxide-semiconductor field-effect transistors (MOSFETs) has been provided.

Models like the one just described can, for instance, be used for sparse coding [157]. Sparse coding can be mathematically formulated as a problem in linear algebra. Consider a vector which describes a certain quantity of interest $\vec{x}$ (for instance, an image), and a dictionary which is available as $\vec{\phi}^i$, $i = 1, \cdots, M$. Furthermore, 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:

$$\vec{x} = \sum_{i=1}^{M} a_i \vec{\phi}^i, \tag{56}$$

$$\min_{\vec{a}} \|\vec{a}\|_0, \ \| \cdot \| \text{ 0-norm}, \tag{57}$$

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, these are equations of the type

$$\frac{\mathrm{d}\vec{u}(t)}{\mathrm{d}t} = F(\vec{u}(t), \vec{q}(t)), \tag{58}$$

where $u_i(t)$ and $q(t)$ are some control functions and $F(\cdot)$ represents a generic continuous function. Some details are provided in Appendix B. The variables $u_i(t)$ are intended 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 [158], 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 [159], defined as

$$W^{k+1} = W^k + 2\eta \vec{o}^t (\vec{x} - (2W^k - I)\vec{o}), \tag{59}$$

where $\vec{o}$ is the output of the crossbar array, and $\vec{x}$ the input, which is used in [160] in order to perform PCA, again with the use of FPGA.

Recent advances in Deep Neural Network implementations using memristor technology (PCM) show remarkable efficiency in terms of energy consumption for a given constant accuracy, with gains up to two order of magnitudes when compared to graphics processing unit (GPU) implementations [116]. The architecture used therein is not dissimilar from the one described in this section using a combination of CMOS and crossbar arrays. This is another example of how crossbar arrays are amenable to efficient matrix multiplication, which renders them competitive to other GPU implementations [111,112,117].

We have already discussed reservoir computing (RC) [65,94] as another way of using memristors within the framework of machine learning whilst taking advantage of their temporal dynamics (see

Section 5.2). In [94], RC was implemented with a memristor reservoir and one layer output with 32×32 crossbar of memristors and trained using an external FPGA. The model was then used successfully to classify the MNIST dataset (5×4 images) as a proof of principle application. The advantage of using RC is that it can be implemented both for online learning and for classification in a teacher–signal framework, and with relatively little computational effort.

In conclusions, CMOS provides an advantage for controlling memristive circuits, but it is possible to use just the inner dynamics of memristors to perform learning [107].

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:

$$\frac{\mathrm{d}\vec{y}}{\mathrm{d}t}(t) = f(\vec{y}(t), t), \quad y(t_0) = y_0, \tag{60}$$

which describes a physical system. In [161], a constructive proof based on a 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:

$$R = \max_{t_0 \leq t \leq t_f} ||\frac{\mathrm{d}^2\vec{y}}{\mathrm{d}t^2}(t)||, \; \epsilon = ||\vec{y}(t_f) - \vec{y}^*||, \tag{61}$$

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, $\frac{\mathrm{d}^2\vec{y}}{\mathrm{d}t^2}$ 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 [162]. 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. Conclusions

In the present article, we have provided an introduction and overview of memristors, 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 screened to avoid the spins to flip, and the memory be lost. Via the interaction between the spins however, logic gates can be constructed [133]. Similarly, memristors can be used as memories, thus trying to avoid their interaction in the circuit via

Kirchhoff laws, or one can try to harness their 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 of 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 plant signaling, amoeba adaptation, and the Hudgkin–Huxley model of squids neurons.

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 though 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 describe 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 the internal state. Despite their differences, the dynamics of memristive components share a common feature, which is the competition between two internal 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 [81] and experimental results with memristive devices [34,163].

We have 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 for their generality. Thus, the integration of CMOS and analog systems in future computers is favorable in the long term. The memristor is 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 memristive 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 [164–171]. 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.

Author Contributions: F.C. and J.P.C. contributed equally to this work.

Funding: F.C. acknowledges the support of NNSA for the U.S. Department of Energy at Los Alamos National Laboratory under Contract No. DE-AC52-06NA25396. J.P.C. received support from the discretionary funding scheme of the Swiss Federal Institute of Aquatic Science and Technology project EmuMore.

Acknowledgments: We thank Fabio L. Traversa, F. Sheldon, Magdalena E. Dale, Giacomo Indiveri, Alex Nugent, Miklos Csontos, Themis Prodromakis, Yogesh Joglekar, for their useful comments and observations, and Shihe Yang, Teuvo Kohonen, Themis Prodromakis, Yogesh Joglekar and Qiangfei Xei for the permission of using figures from their work for this review. Also, we thank in particular Magdalena E. Dale for helping us with the list of companies.

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 quantitative behavior of the case of the metallic beads contained in (and constrained to) an insulating medium of polyvinyl chloride (PVC) is presented below [5,14]. 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 [14] and a room temperature T_0, which is the one of the beads. Without going into the details, it is possible to show a nonlinear 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, reminiscent 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 $\rho_{el}\lambda = LT$. In addition, 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 [14] for more details). A posteriori, many phenomena, both quantum and classical, have been reinterpreted as "memristors" (for instance, Josephson junctions [172]). Below, we provide a non-exhaustive list of mechanisms that have been discussed in the literature. For more detailed information, we suggest the reviews on the subject of resistive switching given in [173–176].

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 [177–179] 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 1960s [180] in amorphous chalcogenides. This is the most mature of the emerging memory technologies. Since we have used various analogies before, 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 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 technology. The typical I–V diagram is shown in Figure A1. One generically has two types 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 [181].

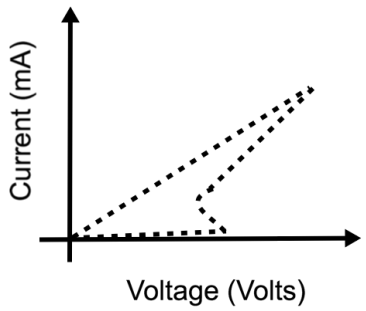


Figure A1. Current–Voltage diagram typical of a phase change materials (PCM) memory storage. The sharp transition at low currents and at a typical "Critical voltage" is clearly visible.

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 [182]. 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 Figure A2.

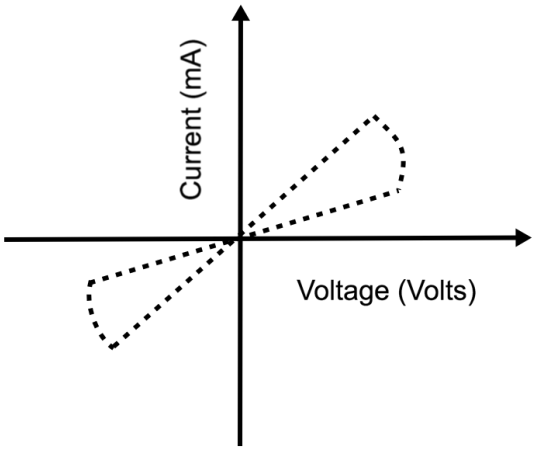


Figure A2. Bipolar and non-volatile switching pinched hysteresis.

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 nonlinear. There can be other non-volatile and non-volatile switching in which we are not discussing here, and due to thermal excitation only. A simplified model of vacancy-charge movement in the dielectric has been proposed in [183]. When nonlinearities 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 Figure A3a.

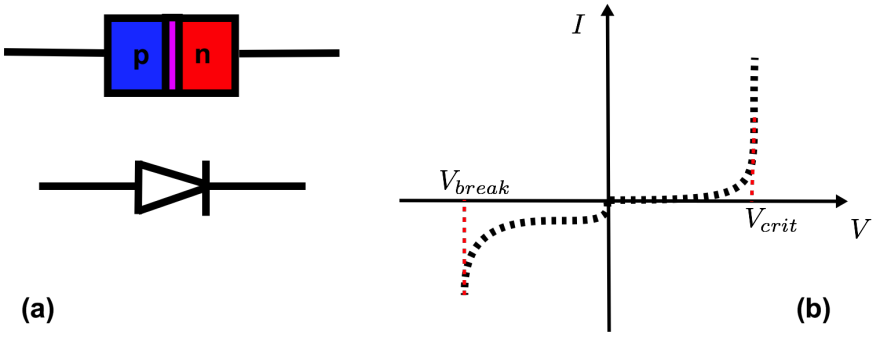


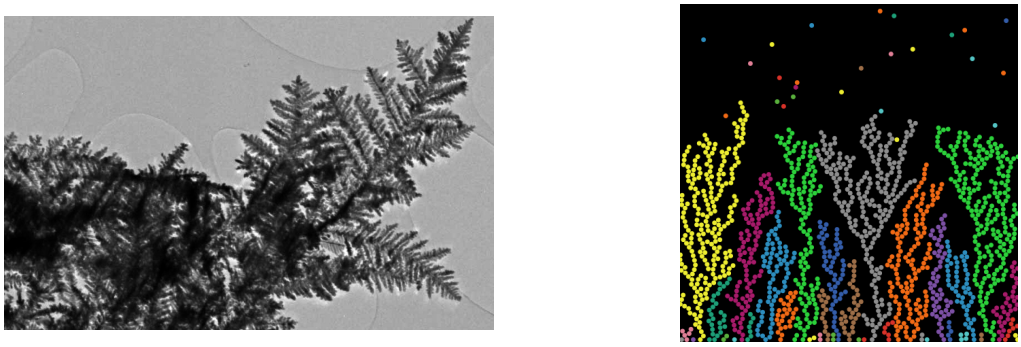
Figure A3. The pn-junction. (**a**) the pink region is the charge exchange region for the doped and undoped region; (**b**) a stylized response in voltage of a diode. In many ways, the shape resembles the nonlinearity which occur in nonlinear memristors in which field effects become dominant.

The diode can be thought of as a dramatically nonlinear resistance, made by merging two materials, 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 will have a very low resistance, and, for voltages lower than V_{break}, the diode will breakdown. In many ways, the shape of Figure A3b 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 Figure A3a), 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 [184]. When the number of cycles becomes comparable with the lifetime of the component, some exotic phenomena as multiple pinchpoints (tri- or four-lobes hysteresis loops) in the V–I diagram can also occur. These can be modeled via the introduction of fractal derivatives [185].

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 [100,163,186–188]. The idea behind these components is that the two electrodes work, when the electrical field is applied, as an electrode and a cathode. 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 Figure 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 [189]. 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 [45], the critical behavior is the accepted one because of their intrinsic nonlinearity [100].



(**a**) (**b**)

Figure A4. Comparison between dendritic growth in silver ion materials and Diffusion-Limited-Aggregation simulated using NetLogo. (**a**) dendritic growth in silver ion at the micro meter scale, reproduced with permission from [190]; (**b**) diffusion-limited-aggregation simulated using NetLogo.

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

The first, suggested in [191] as a phenomenological switching model between two off and on states in TiO_2, is of the form:

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

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

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. In addition, the model above shows that the energy depends exponentially on the current.

Another metal oxide of interest is WO_x, studied in [192]. 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 a 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 perspectives of the present paper is that there are two different reasons why these devices can turn useful [193]. The first advantages is the density compared to standard memory. For instance, compared to DRAM and static random-access memory (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 magnitude slower. In addition, in terms of read–write cycles, the technology of memristors and PCM is between 3–5 orders of magnitude less, but still much more durable than hard disk drive (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 [194] 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 [195]. The starting point is the Landau–Ginzburg–Gilbert (LGG) equation with a spin-torque interaction [196]. 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

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

where γ is called gyromagnetic ratio, α is the damping parameter, H_k is the perpendicular anysotropy in the 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 of full rotational symmetry, spin-torque materials are first order memristors.

Appendix A.5. Mott Memristors

In this section, we discuss resistive switching due to the Mott between insulating and conducting phase in metals [112,197]. In the Mott transition, the important quantity of interest is the electron density, which acts as a control parameter for the transition. This implies that, in general, the transition can be induced via pressure change or doping. The transition is usually studied in the context of weakly correlated electron liquids, in which the Coulomb interactions are screened and thus can be considered weak. In general, there are various known mechanisms that enable an insulator transition. At the single electron (charge carriers in general), there can be band insulators, which we have discussed before in the context of oxide based materials—Peierls insulators [198], which are due to lattice deformations that distort the band gaps; or Anderson insulators [199], an effect due to the interaction of a particle with the disorder in the materials in which, at the quantum level, the charge carrier becomes localized.

The Mott transition occurs due an electron–electron interaction, and leads to the formation of a gap between the ground state of the system and the excited state: this implies that, in order to kick a charge into the conduction band, the system requires overcoming a barrier. The transition occurs because of the repulsion between the electrons, which impedes the free flow of these into the material. Mott memristors take advantage of the fact that certain materials (such as niobium dioxide [200]) have a current induced Mott transition. This is due to the fact that, as the current is enough to locally heat the material above a certain threshold value, the material undergoes phase transition.

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:

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

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 implementable via Hopfield continuous networks [201,202]. The equations above would be linear if the thresholding function T_λ were trivial.

References

1. Chua, L. Memristor-The missing circuit element. *IEEE Trans. Circuit Theory* **1971**, *18*, 507–519. [CrossRef]
2. Chua, L. If it's pinched it's a memristor. *Semicond. Sci. Technol.* **2014**, *29*, 104001. [CrossRef]
3. Chua, L.; Kang, S.M. Memristive devices and systems. *Proc. IEEE* **1976**, *64*, 209–223. [CrossRef]
4. 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. *Nat. Commun.* **2013**, *4*, 1771. [CrossRef]
5. 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. *Granul. Matter* **2010**, *12*, 375–385. [CrossRef]
6. Di Ventra, M.; Pershin, Y.V. On the physical properties of memristive, memcapacitive and meminductive systems. *Nanotechnology* **2013**, *24*, 255201. [CrossRef]
7. Mead, C. Neuromorphic electronic systems. *Proc. IEEE* **1990**, *78*, 1629–1636. [CrossRef]
8. Freeth, T.; Bitsakis, Y.; Moussas, X.; Seiradakis, J.H.; Tselikas, A.; Mangou, H.; Zafeiropoulou, M.; Hadland, R.; Bate, D.; Ramsey, A.; et al. Decoding the ancient Greek astronomical calculator known as the Antikythera Mechanism. *Nature* **2006**, *444*, 587–591. [CrossRef]
9. Adamatzky, A. (Ed.) *Advances in Physarum Machines*; *Emergence, Complexity and Computation*; Springer International Publishing: Cham, Switzerland, 2016; Volume 21.
10. Dalchau, N.; Szép, G.; Hernansaiz-Ballesteros, R.; Barnes, C.P.; Cardelli, L.; Phillips, A.; Csikász-Nagy, A. Computing with biological switches and clocks. *Nat. Comput.* **2018**, *17*, 761–779. [CrossRef]
11. Di Ventra, M.; Pershin, Y.V. The parallel approach. *Nat. Phys.* **2013**, *9*, 200–202. [CrossRef]
12. Davy, H. Additional experiments on Galvanic electricity. *J. Nat. Philos. Chem. Arts* **1801**, *4*, 326.
13. Falcon, E.; Castaing, B.; Creyssels, M. Nonlinear electrical conductivity in a 1D granular medium. *Eur. Phys. J. B* **2004**, *38*, 475–483. [CrossRef]
14. Falcon, E.; Castaing, B. Electrical conductivity in granular media and Branly's coherer: A simple experiment. *Am. J. Phys.* **2005**, *73*, 302–307. [CrossRef]
15. Branly, E. Variations de conductibilite sous diverse influences electriques. *R. Acad. Sci* **1890**, *111*, 785–787.
16. Marconi, G. Wireless telegraphic communication: Nobel Lecture 11 December 1909, Nobel Lectures. In *Physics*; Elsevier Publishing Company: Amsterdam, The Netherlands, 1967; pp. 196–222, 198, 1901–1921.
17. Abraham, I. The case for rejecting the memristor as a fundamental circuit element. *Sci. Rep.* **2018**, *8*, 10972. [CrossRef]
18. Strogatz, S. Like Water for Money. Available online: https://opinionator.blogs.nytimes.com/2009/06/02/guest-column-like-water-for-money/ (accessed on 6 August 2018).
19. Vongehr, S.; Meng, X. The missing memristor has not been found. *Sci. Rep.* **2015**, *5*, 11657. [CrossRef]
20. Vongehr, S. Purely mechanical memristors: Perfect massless memory resistors, the missing perfect mass-involving memristor, and massive memristive systems. *arXiv* **2015**, arXiv:1504.00300.
21. Volkov, A.G.; Tucket, C.; Reedus, J.; Volkova, M.I.; Markin, V.S.; Chua, L. Memristors in plants. *Plant Signal Behav.* **2014**, *9*, e28152. [CrossRef]
22. Gale, E.; Adamatzky, A.; de Lacy Costello, B. Slime mould memristors. *BioNanoScience* **2015**, *5*, 1–8. [CrossRef]
23. Gale, E.; Adamatzky, A.; de Lacy Costello, B. Erratum to: Slime mould memristors. *BioNanoScience* **2015**, *5*, 9. [CrossRef]

24. Szot, K.; Dittmann, R.; Speier, W.; Waser, R. Nanoscale resistive switching in SrTiO3 thin films. *Phys. Status Solidi* **2007**, *1*, R86–R88. [CrossRef]
25. Waser, R.; Aono, M. Nanoionics-based resistive switching memories. *Nat. Mater.* **2007**, *6*, 833–840. [CrossRef]
26. Tsuruoka, T.; Terabe, K.; Hasegawa, T.; Aono, M. Forming and switching mechanisms of a cation-migration-based oxide resistive memory. *Nanotechnology* **2010**, *21*, 425205. [CrossRef]
27. Chua, L. Everything you wish to know about memristors but are afraid to ask. *Radioengineering* **2015**, *24*, 319. [CrossRef]
28. Strukov, D.B.; Snider, G.S.; Stewart, D.R.; Williams, R.S. The missing memristor found. *Nature* **2008**, *453*, 80–83. [CrossRef]
29. Gupta, I.; Serb, A.; Berdan, R.; Khiat, A.; Prodromakis, T. Volatility characterization for RRAM devices. *IEEE Electron Device Lett.* **2017**, *38*. [CrossRef]
30. Abraham, I. Quasi-linear vacancy dynamics modeling and circuit analysis of the bipolar memristor. *PLoS ONE* **2014**, *9*, e111607. [CrossRef]
31. Abraham, I. An advection-diffusion model for the vacancy migration memristor. *IEEE Access* **2016**, *4*, 7747–7757. [CrossRef]
32. Tang, S.; Tesler, F.; Marlasca, F.G.; Levy, P.; Dobrosavljević, V.; Rozenberg, M. Shock waves and commutation speed of memristors. *Phys. Rev. X* **2016**, *6*, 011028. [CrossRef]
33. Wang, F. Memristor for introductory physics. *arXiv* **2008**, arXiv:0808.0286.
34. 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. [CrossRef]
35. Pershin, Y.V.; Ventra, M.D. Spice model of memristive devices with threshold. *Radioengineering* **2013**, *22*, 485–489.
36. Biolek, Z.; Biolek, D.; Biolková, V. Spice model of memristor with nonlinear dopant drift. *Radioengineering* **2009**, *18*, 210–214.
37. Biolek, D.; Di Ventra, M.; Pershin, Y.V. Reliable SPICE simulations of memristors, memcapacitors and meminductors. *Radioengineering* **2013**, *22*, 945.
38. Biolek, D.; Biolek, Z.; Biolkova, V.; Kolka, Z. Reliable modeling of ideal generic memristors via state-space transformation. *Radioengineering* **2015**, *24*, 393–407. [CrossRef]
39. Nedaaee Oskoee, E.; Sahimi, M. Electric currents in networks of interconnected memristors. *Phys. Rev. E* **2011**, *83*, 031105. [CrossRef]
40. Joglekar, Y.N.; Wolf, S.J. The elusive memristor: properties of basic electrical circuits. *Eur. J. Phys.* **2009**, *30*, 661–675. [CrossRef]
41. Abdalla, A.; Pickett, M.D. SPICE modeling of memristors. In Proceedings of the IEEE International Symposium of Circuits and Systems (ISCAS), Rio de Janeiro, Brazil, 15–18 May 2011. [CrossRef]
42. Li, Q.; Serb, A.; Prodromakis, T.; Xu, H. A memristor SPICE model accounting for synaptic activity dependence. *PLoS ONE* **2015**, *10*. [CrossRef]
43. Corinto, F.; Forti, M. Memristor circuits: Flux—Charge analysis method. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2016**, *63*, 1997–2009. [CrossRef]
44. Hindmarsh, A.C.; Brown, P.N.; Grant, K.E.; Lee, S.L.; Serban, R.; Shumaker, D.E.; Woodward, C.S. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Trans. Math. Softw. (TOMS)* **2005**, *31*, 363–396. [CrossRef]
45. Caravelli, F.; Traversa, F.L.; Di Ventra, M. Complex dynamics of memristive circuits: Analytical results and universal slow relaxation. *Phys. Rev. E* **2017**, *95*, 022140. [CrossRef] [PubMed]
46. Caravelli, F. Locality of interactions for planar memristive circuits. *Phys. Rev. E* **2017**, *96*. [CrossRef] [PubMed]
47. 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. *Front. Neurosci.* **2015**, *9*. [CrossRef] [PubMed]
48. 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. [CrossRef]
49. 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. *arXiv* **2018**, arXiv:1804.09848.
50. Meena, J.; Sze, S.; Chand, U.; Tseng, T.Y. Overview of emerging nonvolatile memory technologies. *Nanoscale Res. Lett.* **2014**, *9*, 526. [CrossRef] [PubMed]

51. Steinbuch, K. Die Lernmatrix. *Kybernetik* **1961**, *1*, 36–45. [CrossRef]
52. Kohonen, T. Self-organization and associative memory. In *Springer Series in Information Sciences*; Springer: Berlin/Heidelberg, Germany, 1989; Volume 8.
53. Steinbuch, K. Adaptive networks using learning matrices. *Kybernetik* **1964**, *2*. [CrossRef]
54. 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. *J. Comput. Sci. Technol.* **2016**, *31*, 3–19. [CrossRef]
55. Strukov, D.B.; Williams, R.S. Four-dimensional address topology for circuits with stacked multilayer crossbar arrays. *Proc. Natl. Acad. Sci. USA* **2009**, *106*, 20155–20158. [CrossRef]
56. 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/SiO_2/Si memristors. *Nat. Commun.* **2017**, *8*, 15666. [CrossRef]
57. Li, C.; Belkin, D.; Li, Y.; Yan, P.; Hu, M.; Ge, N.; Jiang, H.; Montgomery, E.; Lin, P.; Wang, Z.; et al. Efficient and self-adaptive in-situ learning in multilayer memristor neural networks. *Nat. Commun.* **2018**, *9*, 2385. [CrossRef]
58. Itoh, M.; Chua, L. Memristor cellular automata and memristor discrete-time cellular neural networks. In *Memristor Networks*; Springer International Publishing: Cham, Switzerland, 2014; pp. 649–713.
59. Nugent, M.A.; Molter, T.W. Thermodynamic-RAM technology stack. *Int. J. Parall. Emerg. Distrib. Syst.* **2018**, *33*, 430–444. [CrossRef]
60. Hebb, D. *The Organization of Behavior*; Wiley & Sons: Hoboken, NJ, USA, 1949.
61. Koch, G.; Ponzo, V.; Di Lorenzo, F.; Caltagirone, C.; Veniero, D. Hebbian and anti-Hebbian spike-timing-dependent plasticity of human cortico-cortical connections. *J. Neurosci.* **2013**, *33*, 9725–9733. [CrossRef] [PubMed]
62. 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, India, 2017; Chapter 2, pp. 17–43. [CrossRef]
63. Ielmini, D. Brain-inspired computing with resistive switching memory (RRAM): Devices, synapses and neural networks. *Microelectron. Eng.* **2018**, *190*, 44–53. [CrossRef]
64. 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 Discuss.* **2018**. [CrossRef] [PubMed]
65. Carbajal, J.P.; Dambre, J.; Hermans, M.; Schrauwen, B. Memristor models for machine learning. *Neural Comput.* **2015**, *27*, [CrossRef] [PubMed]
66. Eliasmith, C.; Anderson, C.H. *Neural Engineering: Computational, Representation, and Dynamics in Neurobiological Systems*; MIT Press: Cambridge, MA, USA, 2002.
67. MacLennan, B.J. Analog computation. In *Computational Complexity: Theory, Techniques, and Applications*; Meyers, R.A., Ed.; Springer: New York, NY, USA, 2012; pp. 161–184. [CrossRef]
68. MacLennan, B. The promise of analog computation. *Int. J. Gener. Syst.* **2014**, *43*, 682–696. [CrossRef]
69. MacLennan, B.J. Physical and formal aspects of computation: Exploiting physics for computation and exploiting computation for physical purposes. In *Advances in Unconventional Computing*; Andrew, A., Ed.; Springer: Cham, Switzerland, 2017; pp. 117–140. [CrossRef]
70. Horsman, C.; Stepney, S.; Wagner, R.C.; Kendon, V. When does a physical system compute? *Proc. R. Soc. A Math. Phys. Eng. Sci.* **2014**, *470*, 20140182. [CrossRef]
71. Shannon, C.E. Mathematical theory of the differential analyzer. *J. Math. Phys.* **1941**, *20*, 337–354. [CrossRef]
72. 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. [CrossRef]
73. Deutsch, D. Quantum theory, the Church–Turing principle and the universal quantum computer. *Proc. R. Soc.* **1985**, *400*. [CrossRef]
74. 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 Embed. Syst. Lett.* **2018**, *10*, 2–5. [CrossRef]
75. Traversa, F.L.; Ramella, C.; Bonani, F.; Di Ventra, M. Memcomputing NP-complete problems in polynomial time using polynomial resources and collective states. *Sci. Adv.* **2015**, *1*, e1500031. [CrossRef] [PubMed]

76. Levi, M. A Water-Based Solution of Polynomial Equations. Available online: https://sinews.siam.org/Details-Page/a-water-based-solution-of-polynomial-equations-2 (accessed on 20 June 2018).
77. Axehill, D. Integer Quadratic Programming for Control and Communication. Ph.D. Thesis, Institute of Technology, Department of Electrical Engineering and Automatic Control, Link?ping University, Link?ping, Sweden, 2008.
78. Venegas-Andraca, S.E.; Cruz-Santos, W.; McGeoch, C.; Lanzagorta, M. A cross-disciplinary introduction to quantum annealing-based algorithms. *Contemp. Phys.* **2018**, *59*, 174–197. [CrossRef]
79. Rothemund, P.W.K.; Papadakis, N.; Winfree, E. Algorithmic self-assembly of DNA sierpinski triangles. *PLoS Biol.* **2004**, *2*, e424. [CrossRef]
80. Qian, L.; Winfree, E.; Bruck, J. Neural network computation with DNA strand displacement cascades. *Nature* **2011**, *475*, 368–372. [CrossRef] [PubMed]
81. Dorigo, M.; Gambardella, L.M. Ant colonies for the travelling salesman problem. *Biosystems* **1997**, *43*, 73–81. [CrossRef]
82. Bonabeau, E. Editor's introduction: Stigmergy. *Artif. Life* **1999**, *5*, 95–96. [CrossRef]
83. Muller, K.R.; Mika, S.; Ratsch, G.; Tsuda, K.; Scholkopf, B. An introduction to kernel-based learning algorithms. *IEEE Trans. Neural Netw.* **2001**, *12*, 181–201. [CrossRef]
84. Hofmann, T.; Schölkopf, B.; Smola, A.J. Kernel methods in machine learning. *Ann. Stat.* **2008**, *36*, 1171–1220. [CrossRef]
85. Huang, G.B.; Zhu, Q.Y.; Siew, C.K. Extreme learning machine: A new learning scheme of feedforward neural networks. In Proceedigns of the 2004 IEEE International Joint Conference on the Neural Networks, Budapest, Hungary, 25–29 July 2004; Volume 2, pp. 985–990.
86. 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. [CrossRef]
87. Parmar, V.; Suri, M. Exploiting variability in resistive memory devices for cognitive systems. In *Advances in Neuromorphic Hardware Exploiting Emerging Nanoscale Devices*; Manan, S., Ed.; Springer: New Delhi, India, 2017; Chapter 9, pp. 175–195. [CrossRef]
88. Hoeting, J.A.; Madigan, D.; Raftery, A.E.; Volinsky, C.T. Bayesian model averaging: A tutorial. *Stat. Sci.* **1999**, *14*, 382–401.
89. 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.
90. Maass, W.; Natschläger, T.; Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput.* **2002**, *14*, 2531–2560. [CrossRef] [PubMed]
91. Verstraeten, D. Reservoir Computing: Computation with Dynamical Systems. Ph.D. Thesis, Ghent University, Ghent, Belgium, 2009.
92. Dale, M.; Miller, J.F.; Stepney, S.; Trefzer, M.A. A substrate-independent framework to characterise reservoir computers. *arXiv* **2018**, arXiv:1810.07135.
93. 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, Germany, 2003; pp. 588–597.
94. Du, C.; Cai, F.; Zidan, M.A.; Ma, W.; Lee, S.H.; Lu, W.D. Reservoir computing using dynamic memristors for temporal information processing. *Nat. Commun.* **2017**, *8*, 2204. [CrossRef] [PubMed]
95. Sheriff, M.R.; Chatterjee, D. Optimal dictionary for least squares representation. *J. Mach. Learn. Res.* **2017**, *18*, 1–28.
96. Corradi, F.; Eliasmith, C.; Indiveri, G. Mapping arbitrary mathematical functions and dynamical systems to neuromorphic VLSI circuits for spike-based neural computation. In Proceedings of the 2014 IEEE International Symposium on Circuits and Systems (ISCAS), Melbourne VIC, Australia, 1–5 June 2014; pp. 269–272. [CrossRef]
97. Benna, M.K.; Fusi, S. Computational principles of synaptic memory consolidation. *Nat. Neurosci.* **2016**, *19*, 1697–1706. [CrossRef] [PubMed]
98. Chialvo, D. Are our senses critical? *Nat. Phys.* **2006**, *2*, 301–302. [CrossRef]
99. Hesse, J.; Gross, T. Self-organized criticality as a fundamental property of neural systems. *Front. Syst. Neurosci.* **2014**, *23*. [CrossRef]
100. Avizienis, A.V.; Sillin, H.O.; Martin-Olmos, C.; Shieh, H.H.; Aono, M.; Stieg, A.Z.; Gimzewski, J.K. Neuromorphic atomic switch networks. *PLoS ONE* **2012**, *7*. [CrossRef]

101. Caravelli, F.; Hamma, A.; Di Ventra, M. Scale-free networks as an epiphenomenon of memory. *EPL (Europhys. Lett.)* **2015**, *109*, 28006. [CrossRef]
102. Caravelli, F. Trajectories entropy in dynamical graphs with memory. *Front. Robot. AI* **2016**, *3*. [CrossRef]
103. Sheldon, F.C.; Di Ventra, M. Conducting-insulating transition in adiabatic memristive networks. *Phys. Rev. E* **2017**, *95*, 012305. [CrossRef] [PubMed]
104. Veberic, D. Having fun with Lambert W(x) function. *arXiv* **2010**, arXiv:1003.1628
105. Berdan, R.; Vasilaki, E.; Khiat, A.; Indiveri, G.; Serb, A.; Prodromakis, T. Emulating short-term synaptic dynamics with memristive devices. *Sci. Rep.* **2016**, *6*, 18639. [CrossRef] [PubMed]
106. Krestinskaya, O.; Dolzhikova, I.; James, A.P. Hierarchical temporal memory using memristor networks: A survey. *arXiv* **2018**, arXiv:1805.02921.
107. Caravelli, F. The mise en scéne of memristive networks: effective memory, dynamics and learning. *Int. J. Parall. Emerg. Distrib. Syst.* **2018**, *33*, 350–366. [CrossRef]
108. Caravelli, F.; Barucca, P. A mean-field model of memristive circuit interaction. *EPL (Europhys. Lett.)* **2018**, *122*, 40008. [CrossRef]
109. Caravelli, F. Asymptotic behavior of memristive circuits and combinatorial optimization. *arXiv* **2017**, arXiv:1712.07046.
110. Boros, E.; Hammer, P.L.; Tavares, G. Local search heuristics for Quadratic Unconstrained Binary Optimization (QUBO). *J. Heuristics* **2007**, *13*, 99–132. [CrossRef]
111. Hu, S.G.; Liu, Y.; Liu, Z.; Chen, T.P.; Wang, J.J.; Yu, Q.; Deng, L.J.; Yin, Y.; Hosaka, S. Associative memory realized by a reconfigurable memristive Hopfield neural network. *Nat. Commun.* **2015**, *6*. [CrossRef]
112. Kumar, S.; Strachan, J.P.; Williams, R.S. Chaotic dynamics in nanoscale NbO_2 Mott memristors for analogue computing. *Nature* **2017**, *548*, 318–321. [CrossRef]
113. Tarkov, M. Hopfield network with interneuronal connections based on memristor bridges. *Adv. Neural Netw.* **2016**, 196–203. [CrossRef]
114. Sebastian, A.; Tuma, T.; Papandreou, N.; Le Gallo, M.; Kull, L.; Parnell, T.; Eleftheriou, E. Temporal correlation detection using computational phase-change memory. *Nat. Commun.* **2017**, *8*, 1115. [CrossRef] [PubMed]
115. Parihar, A.; Shukla, N.; Jerry, M.; Datta, S.; Raychowdhury, A. Vertex coloring of graphs via phase dynamics of coupled oscillatory networks. *Sci. Rep.* **2017**, *7*, 911. [CrossRef] [PubMed]
116. Ambrogio, S.; Narayanan, P.; Tsai, H.; Shelby, R.M.; Boybat, I.; di Nolfo, C.; Sidler, S.; Giordano, M.; Bodini, M.; Farinha, N.C.P.; et al. Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature* **2018**, *558*, 60–67. [CrossRef]
117. Hu, S.G.; Liu, Y.; Liu, Z.; Chen, T.P.; Wang, J.J.; Yu, Q.; Deng, L.J.; Yin, Y.; Hosaka, S. A memristive Hopfield network for associative memory. *Nat. Commun.* **2015**, *6*, 7522. [CrossRef]
118. Pershin, Y.V.; Di Ventra, M. Solving mazes with memristors: A massively parallel approach. *Phys. Rev. E* **2011**, *84*, 046703. [CrossRef] [PubMed]
119. Pershin, Y.V.; Di Ventra, M. Self-organization and solution of shortest-path optimization problems with memristive networks. *Phys. Rev. E* **2013**, *88*, 013305. [CrossRef]
120. Adlerman, L.M. Molecular computation of solutions to combinatorial problems. *Science* **1994**, *266*, 1021–1024. [CrossRef]
121. Adamatzky, A. Computation of shortest path in cellular automata. *Math. Comput. Model.* **1996**, *23*, 105–113. [CrossRef]
122. Prokopenko, M. Guided self-organization. *HFSP J.* **2009**, *3*, 287–289. [CrossRef]
123. Borghetti, J.; Li, Z.; Straznicky, J.; Li, X.; Ohlberg, D.A.; Wu, W.; Stewart, D.R;. Williams, R.S. A hybrid nanomemristor/transistor logic circuit capable of self-programming. *Proc. Nat. Acad. Sci. USA* **2009**, *106*, 1699–1703. [CrossRef] [PubMed]
124. Di Ventra, M.; Pershin, Y.V.; Chua, L.O. Circuit elements with memory: Memristors, memcapacitors, and meminductors. *Proc. IEEE* **2009**, *97*. [CrossRef]
125. Traversa, F.L.; Ventra, M.D. Universal memcomputing machines. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 2702–2715. [CrossRef] [PubMed]
126. Nugent, M.A.; Molter, T.W. AHaH computing–from metastable switches to attractors to machine learning. *PLoS ONE* **2014**, *9*. [CrossRef] [PubMed]
127. Hurley, P. *A Concise Introduction to Logic*, 12th ed.; Cengage Learning: Cambridge, UK, 2015.

128. Gale, E.; de Lacy Costello, B.; Adamatzky, A. Boolean logic gates from a single memristor via low-level sequential logic. In *Unconventional Computation and Natural Computation*; Mauri, G., Dennunzio, A., Manzoni, L., Porreca, A.E., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 79–89.
129. Papandroulidakis, G.; Khiat, A.; Serb, A.; Stathopoulos, S.; Michalas, L.; Prodromakis, T. Metal oxide-enabled reconfigurable memristive threshold logic gates. In Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 27–30 May 2018.
130. Traversa, F.L.; Di Ventra, M. Polynomial-time solution of prime factorization and NP-complete problems with digital memcomputing machines. *Chaos* **2017**, *27*. [CrossRef] [PubMed]
131. Traversa, F.L.; Cicotti, P.; Sheldon, F.; Di Ventra, M. Evidence of an exponential speed-up in the solution of hard optimization problems. *Complexity* **2018**, *2018*, 7982851. [CrossRef]
132. Traversa, F.; Di Ventra, T. Memcomputing: Leveraging memory and physics to compute efficiently. *J. Appl. Phys.* **2018**, *123*. [CrossRef]
133. Caravelli, F.; Nisoli, C. Computation via interacting magnetic memory bites: Integration of boolean gates. *arXiv* **2018**, arXiv:1810.09190.
134. Sah, M.P.; Hyongsuk, K.; Chua, L.O. Brains are made of memristors. *IEEE Circuits Syst. Mag.* **2014**, *14*, 12–36. [CrossRef]
135. Markin, V.S.; Volkov, A.G.; Chua, L. An analytical model of memristors in plants. *Plant Signal. Behav.* **2014**, *9*, e972887. [CrossRef]
136. Saigusa, T.; Tero, A.; Nakagaki, T.; Kuramoto, Y. Amoebae anticipate periodic events. *Phys. Rev. Lett.* **2008**, *100*, 018101. [CrossRef] [PubMed]
137. Hodgkin, A.L.; Huxley, A.F. Currents carried by sodium and potassium ions through the membrane of the giant axon of Loligo. *J. Physiol.* **1952**, *116*, 449–472. [CrossRef] [PubMed]
138. Pershin, Y.V.; La Fontaine, S.; Di Ventra, M. Memristive model of amoeba's learning. *Phys. Rev. E* **2010**, *80*, 021926. [CrossRef] [PubMed]
139. Yang, J.J.; Pickett, M.D.; Li, X.; Ohlberg, D.A.; Stewart, D.R.; Williams, R.S. Memristive switching mechanism for metal/oxide/metal nanodevices. *Nat. Nano* **2008**, *3*, 429. [CrossRef] [PubMed]
140. Pershin, Y.V.; Di Ventra, M. Experimental demonstration of associative memory with memristive neural networks. *Neural Netw.* **2010**, *23*, 881–886. [CrossRef] [PubMed]
141. Tan, Z.H.; Yin, X.B.; Yang, R.; Mi, S.B.; Jia, C.L.; Guo, X. Pavlovian conditioning demonstrated with neuromorphic memristive devices. *Sci. Rep.* **2017**, *7*, 713. [CrossRef] [PubMed]
142. Turcotte, D.L. Self-organized criticality. *Rep. Prog. Phys.* **1999**, *62*, 1377–1429. [CrossRef]
143. Jensen, H.J. *Self-Organized Criticality*; Cambridge University Press: Cambridge, UK, 1998. [CrossRef]
144. Marković, D.; Gros, C. Power laws and self-organized criticality in theory and nature. *Phys. Rep.* **2014**, *536*, 41–74, [CrossRef]
145. Alava, M.J.; Nukala, P.K.V.V.; Zapperi, S. Statistical models of fracture. *Adv. Phys.* **2006**, *55*, 349–476. [CrossRef]
146. Widrow, B. *An Adaptive 'Adaline' Neuron Using Chemical 'Memistors'*; Technical Report 1553-2; Stanford Electronics Laboratories: Stanford, CA, USA, 1960.
147. Adhikari, S.P.; Kim, H. Why are memristor and memistor different devices? In *Memristor Networks*; Adamatzky, A., Chua, L., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 95–112. [CrossRef]
148. Cai, W.; Tetzlaff, R. Why are memristor and memristor different devices? In *Memristor Networks*; Adamatzky, A., Chua, L., Eds.; Springer: Berlin, Germany, 2014; pp. 113–128, [CrossRef]
149. Johnsen, K.G. An introduction to the memristor—A valuable circuit element in bioelectricity and bioimpedance. *J. Electr. Bioimpedance* **2012**, *3*, 20–28. [CrossRef]
150. DeBenedictis, E.P. Computational complexity and new computing approaches. *Computer* **2016**, *49*, 76–79. [CrossRef]
151. Alibart, F.; Zamanidoost, E.; Strukov, D.B. Pattern classification by memristive crossbar circuits using ex situ and in situ training. *Nat. Commun.* **2013**, *4*, 2072. [CrossRef] [PubMed]
152. Tissari, J.; Poikonen, J.H.; Lehtonen, E.; Laiho, M.; Koskinen, L. K-means clustering in a memristive logic array. In Proceedings of the IEEE 15th International Conference on Nanotechnology (IEEE-NANO), Rome, Italy, 27–30 July 2015.

153. Merkel, C.; Kudithipudi, D. Unsupervised learning in neuromemristive systems. In Proceedings of the 2015 National Aerospace and Electronics Conference (NAECON), Dayton, OH, USA, 15–19 June 2015.
154. Jeong, Y.; Lee, J.; Moon, J.; Shin, J.H.; Lu, W.D. K-means data clustering with memristor networks. *Nano Lett.* **2018**, *18*, 4447–4453. [CrossRef] [PubMed]
155. Widrow, B.; Lehr, M. 30 years of adaptive neural networks: Perceptron, Madaline, and backpropagation. *Proc. IEEE* **1990**, *78*, 1415–1442. [CrossRef]
156. Soudry, D.; Di Castro, D.; Gal, A.; Kolodny, A.; Kvatinsky, S. Memristor-based multilayer neural networks with online gradient descent training. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 2408–2421. [CrossRef]
157. Rozell, C.J.; Johnson, D.H.; Baraniuk, R.G.; Olshausen, B.A. Sparse coding via thresholding and local competition in neural circuits. *Neural Comput.* **2008**, *20*, 2526–2563. [CrossRef]
158. Sheridan, P.M.; Cai, F.; Du, C.; Ma, W.; Zhang, Z.; Lu, W.D. Sparse coding with memristor networks. *Nat. Nanotechnol.* **2017**, *12*, 784–789. [CrossRef]
159. Sanger, T.D. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Netw.* **1989**, *2*, 459–473. [CrossRef]
160. Choi, S.; Sheridan, P.; Lu, W.D. Data clustering using memristor networks. *Sci. Rep.* **2015**, *5*, 10492. [CrossRef]
161. Vergis, A.; Steiglitz, K.; Dickinson, B. The complexity of analog computation. *Math. Comput. Simul.* **1986**, *28*, 91–113. [CrossRef]
162. Ercsey-Ravasz, M.; Toroczkai, Z. Optimization hardness as transient chaos in an analog approach to constraint satisfaction. *Nat. Phys.* **2011**, *7*, 966–970. [CrossRef]
163. Wang, Z.; Joshi, S.; Savel'ev, S.E.; Jiang, H.; Midya, R.; Lin, P.; Hu, M.; Ge, N.; Strachan, J.P.; Li, Z.; et al. Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing. *Nat. Mater.* **2017**, *16*, 101–108. [CrossRef] [PubMed]
164. Moss, F. Stochastic resonance and sensory information processing: A tutorial and review of application. *Clin. Neurophysiol.* **2004**, *115*, 267–281. [CrossRef] [PubMed]
165. McDonnell, M.D.; Abbott, D. What is stochastic resonance? Definitions, misconceptions, debates, and its relevance to biology. *PLoS Comput. Boil.* **2009**, *5*, e1000348. [CrossRef] [PubMed]
166. McDonnell, M.D.; Ward, L.M. The benefits of noise in neural systems: Bridging theory and experiment. *Nat. Rev. Neurosci.* **2011**, *12*, 415–426. [CrossRef]
167. Stotland, A.; Di Ventra, M. Stochastic memory: Memory enhancement due to noise. *Phys. Rev. E* **2012**, *85*, 011116. [CrossRef] [PubMed]
168. Slipko, V.A.; Pershin, Y.V.; Di Ventra, M. Changing the state of a memristive system with white noise. *Phys. Rev. E* **2013**, *87*, 042103. [CrossRef]
169. Patterson, G.A.; Fierens, P.I.; Grosz, D.F. Resistive switching assisted by noise. In *Understanding Complex Systems*; Springer: Berlin, Germany, 2014; pp. 305–311.
170. Fu, Y.X.; Kang, Y.M.; Xie, Y. Subcritical hopf bifurcation and stochastic resonance of electrical activities in neuron under electromagnetic induction. *Front. Comput. Neurosci.* **2018**, *12*. [CrossRef]
171. Feali, M.S.; Ahmadi, A. Realistic Hodgkin–Huxley axons using stochastic behavior of memristors. *Neural Process. Lett.* **2017**, *45*, 1–14. [CrossRef]
172. Peotta, S.; Di Ventra, M. Superconducting memristors. *Phys. Rev. Appl.* **2014**, *2*. [CrossRef]
173. Di Ventra, M.; Pershin, Y.V. Memory materials: A unifying description. *Mater. Today* **2011**, *14*, 584–591. [CrossRef]
174. Kuzum, D.; Yu, S.; Wong, H.S.P. Synaptic electronics: Materials, devices and applications. *Nanotechnology* **2013**, *24*, 382001. [CrossRef] [PubMed]
175. Yang, J.J.; Strukov, D.B.; Stewart, D.R. Memristive devices for computing. *Nat. Nanotechnol.* **2013**, *8*, 13–24. [CrossRef] [PubMed]
176. 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.; et al. Field- and irradiation-induced phenomena in memristive nanomaterials. *Phys. Status Solidi (c)* **2016**, *13*, 870–881. [CrossRef]
177. Ovshinsky, S.R. Reversible electrical switching phenomena in disordered structures. *Phys. Rev. Lett.* **1968**, *21*, 1450–1453. [CrossRef]
178. Neale, R.G.; Nelson, D.L.; Moore, G.E. Nonvolatile and reprogrammable, the read-mostly memory is here. *Electronic* **1970**, *43*, 56–60.

179. Buckley, W.; Holmberg, S. Electrical characteristics and threshold switching in amorphous semiconductors. *Solid-State Electron.* **1975**, *18*, 127–147. [CrossRef]
180. Ielmini, D.; Lacaita, A.L. Phase change materials in non-volatile storage. *Mater. Today* **2011**, *14*, 600–607. [CrossRef]
181. Campbell, K.A. Self-directed channel memristor for high temperature operation. *Microelectron. J.* **2017**, *59*, 10–14. [CrossRef]
182. 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. *Nat. Commun.* **2017**, *8*, 1972. [CrossRef]
183. Chernov, A.A.; Islamov, D.R.; Pik'nik, A.A. Non-linear memristor switching model. *J. Phys. Conf. Ser.* **2016**, *754*, 102001. [CrossRef]
184. Balatti, S.; Ambrogio, S.; Wang, Z.; Sills, S.; Calderoni, A.; Ramaswamy, N.; Ielmini, D. Voltage-controlled cycling endurance of HfO_x-based resistive-switching memory. *IEEE Trans. Electron Devices* **2015**, *62*. [CrossRef]
185. Hamed, E.M.; Fouda, M.E.; Radwan, A.G. Conditions and emulation of double pinch-off points in fractional-order memristor. In Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 27–30 May 2018; pp. 1–5. [CrossRef]
186. 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, Switzerland, 2014; pp. 173–209.
187. 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. [CrossRef] [PubMed]
188. Stieg, A.Z.; Avizienis, A.V.; Sillin, H.O.; Martin-Olmos, C.; Aono, M.; Gimzewski, J.K. Emergent criticality in complex turing b-type atomic switch networks. *Adv. Mater.* **2012**, *24*, 286–293. [CrossRef] [PubMed]
189. 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. *Jpn. J. Appl. Phys.* **2018**, *57*, 03ED02. [CrossRef]
190. 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. [CrossRef] [PubMed]
191. 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. *J. Appl. Phys.* **2009**, *106*, 074508. [CrossRef]
192. Chang, T.; Jo, S.H.; Kim, K.H.; Sheridan, P.; Gaba, S.; Lu, W. Synaptic behaviors and modeling of a metal oxide memristive device. *Appl. Phys. A* **2011**, *102*, 857–863. [CrossRef]
193. International Technology Roadmap for Semiconductors. Available online: http://www.itrs2.net/ (accessed on 8 December 2018).
194. Ralph, D.C.; Stiles, M.D. Spin transfer torques. *J. Magn. Magn. Mater.* **2008**, *320*, 1190–1216. [CrossRef]
195. Wang, X.; Chen, Y.; Xi, H.; Li, H.; Dimitrov, D. Spintronic memristor through spin-torque-induced magnetization motion. *IEEE Electron Device Lett.* **2009**, *30*, 294–297. [CrossRef]
196. Sun, J.Z. Spin-current interaction with a monodomain magnetic body: A model study. *Phys. Rev. B* **2000**, *62*, 570–578. [CrossRef]
197. Pickett, D.M.; Medeiros-Riberi, G.; Williams, R.S. A scalable neuristor built with Mott memristors. *Nat. Mater.* **2013**, *12*. [CrossRef] [PubMed]
198. Kagoshima, S. Peierls phase transition. *Jpn. J. Appl. Phys.* **1981**, *20*. [CrossRef]
199. Evers, F.; Mirlin, A. Anderson transitions. *Rev. Mod. Phys.* **2008**, *80*. [CrossRef]
200. Chopra, K.L. Current-controlled negative resistance in thin niobium oxide films. *Proc. IEEE* **1963**, *51*, 941–942. [CrossRef]
201. Hopfield, J.J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA* **1982**, *79*, 2554–2558. [CrossRef] [PubMed]
202. Hopfield, J.J.; Tank, D.W. Computing with neural circuits: A model. *Science* **1986**, *233*, 625–633. [CrossRef]