Doctoral Dissertation

Doctoral Program in Electrical, Electronics and Communications Engineering
(34^{th}cycle)

Memristor-based hardware accelerators: from device modeling to AI applications

Francesco Marrone

******

Supervisor:
Prof. Fernando Corinto

Politecnico di Torino
2022

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Francesco Marrone
2022

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

Acknowledgements

Exactly ten years ago I entered Politecnico di Torino as a freshman studying Electronics Engineering. Now, ten years later, at the time of defending the four and half years of my research work which are condensed in this manuscript, I cannot be anything but grateful to this institution which has been like a second home to me for quite a long time. I have met lots of amazing people in this school and would like to thank as many as I can fit on these acknowledgements pages.

I consider myself extremely lucky to have had Prof. Fernando Corinto as the advisor of my PhD. He brought me into the Neuromorphic Computing research community during my Master's thesis and has been since then not only my scientific point of reference but also an inspiration for his human kindness. He has always supported me with a mix of good words, freedom and concrete help during all the many crisis along the PhD journey. I will never thank him enough for the polishing work that he has done on my professional attitude by smoothing the perfectionism and fueling the curiosity.

It is very difficult for me to imagine this adventure without Dr. Gianluca Zoppo. He has been not only my PhD buddy but also a true friend. I do thank him from my hearth for his invaluable support through difficult times and for pushing me to do my best while stuck in research discouragement. He will always be a model to me for his work ethic and, with deep gratitude, I wish him good luck for his future career.

Last but not least, I want to thank Dr. Jacopo Secco. I will never forget all the coffee breaks with good conversations we have had during the years. Thanks to him I have had the opportunity to enrich my professional experience with corporate R&D project where to apply the latest advancements in

Machine Learning. I have learnt a lot from him and consider him a model for his spirit of enterprise and ambition.

A big thanks is also due to Dr. Abu Sebastian and his team, in particular to Benedikt Kersting, for the opportunity of studying and experiment with their Phase Change Memory technology at IBM Research Zurich. With them I also need to thank Prof. Leon O. Chua for coming up with the idea for this research project and for all the stimulating conversations we had while he was visiting professor in our research group.

A special thanks goes to the group of Prof. R. Stanley Williams at Texas A&M for letting me join their research collaboration with my group during the outbreak of COVID-19 in 2020. I consider the work done during that year the most scientifically productive of my PhD journey. I have learnt a lot from them, in particular from the discussions I had with Anil Korkmaz, and the feedbacks given by Prof. Williams and Prof. Samuel Palermo have taught me a lot on the way to impact as a researcher.

Thanks also to Dr. Alon Ascoli and Prof. Mauro Di Marco for reading and reviewing my thesis. I think your suggestions have substantially improved this manuscript.

A big greeting goes to Dr. Andrea Napolitano, Dr. Riccardo Gervasi and Ada Palamà. I think we have had some of the best lunch breaks together. I have to say that it was a pleasure to meet them in my doctoral years.

A big thanks goes to my family which has always supported me and provided unconditional love. A huge thanks also goes to Gaia who has supported, encouraged, and helped me during this tough year.

I would also need to thank many close friends which have been fundamental during these years and in my life in general but they are probably too many so I ask in advance their pardon if they are not explicitly named here...you all have impacted who I am now.

Summary

In recent times, the interest in artificial intelligence algorithms has skyrocketed well out of academia marking a substantial shift in most, if not all, the industrial sectors. This revolution started in the early 2010s and was made possible by the combined availability of very powerful parallel processing units (e.g. GPUs) and more and more abundant "big data". Although the GPU has served well as platform to kick-start this revolution, currently this conventional digital hardware accelerator struggles to satisfy the increasingly high requirements of machine learning algorithms within a reasonable power consumption range. Most of the energy inefficiency which characterizes conventional digital hardware accelerators (e.g. GPU, TPU, FPGA etc) is to blame on the traditional von Neumann architecture they implement which involves data being sent back and forth between memory and the processor. In this context, neuromorphic computing has emerged as an alternative to contemporary processing units' architectural design choices which may rise to the challenges posed by modern AI training and inference tasks. Neuromorphic computing takes inspiration from the biology of the animal brain by putting the storage of information and its processing in very close spatial proximity if not encoded and performed by the same processing element. It is this the case of memristive technologies which have attracted the interest of lots of researchers and companies as good candidates to implement the synaptic function in neuromorphic hardware accelerators. The overall goal of this doctoral thesis is the investigation of memristive based computing architecture exploiting an holistic approach ranging from the modeling of memristive devices to the mapping of advanced machine learning algorithm on crossbar arrays. The first part of the thesis is devoted to memristive technologies modeling techniques while its second part deals with the design and study of meristor-based architectures for machine learning tasks. In particular the thesis contains novel results on the use of DRMs for Phase Change Memory devices, the study of Spiking Neural Networks and the design of advanced memristive crossbar based accelerators for linear algebra problems (Pagerank).

Contents

List of Figures

List of Tables

Introduction

Motivations

In the last decade the emergence of big data problems and real world machine learning applications has put harder and harder challenges in front of computer engineers. Sought-after tasks such as speech recognition and image detection, which take fractions of milliwatt for the human brain to accomplish, require many orders of magnitude more power to achieve on traditional Von Neumann digital processors. In the new era of computing, the high inefficiency of the conventional paradigms calls for novel computing approaches. Very promising among those are the physics-based paradigms (e.g. Neuromorphic Computing, Quantum Computing etc.), where the fundamental laws of nature are exploited to carry out the computation. Whereas Quantum Computing exploits the laws of Quantum Mechanics, Neuromorphic Computing exploits the laws of Classical Electrodynamics to build bio-inspired analog computers which are much more energy-efficient and compact than their traditional digital counterparts.
On the neuromorphic computing side, large interest about memristive technologies has grown during the past decade. Memristors, theorized by Chua[3, 4] in 1971 as the 4^{th} fundamental circuit elements, are $1-$port systems whose electric conductance depends on the past evolution of the port variables (e.g. current and/or voltage)[5]. These fundamental elements take their name by the union of the two words "memory" and "resistor", as such circuit components behave as resistors with memory properties. Although conceived half a century ago, the memristor has been identified with a physical implementation only in 2008 when the research group lead by Stanley Williams at HP Labs recognized the hallmark of memristive behavior[6] in

their Titanium Oxide nanodevices [7]. Since then, the numerous and heterogeneous technologies that fall into the memristive class have been widely exploited in analog and digital systems for a broad scope of applications, including amplifiers, filters, oscillators, logic gates and pseudo-random number generators [8–10]. At the time of writing, given memristors' multilevel analogue memory capabilities, mass storage and neuromorphic computing seem to be the most promising among those applications with each technology being best suited for certain problems[11]. In terms of voltage-current characteristic a two terminal memristor device is described by a memductance, which may depend on a set of first and second order state variables, commonly linked to the internal geometric parameters and to the internal temperature.

Fundamental to the development of a memristor-based neuromorphic computing system is the complexity of the physical modeling to be employed in higher-level circuit simulations. Thus, high-level identification techniques, circuit modeling methods and technology specific compact models are needed to analyze analogue computing circuits made of memristors. Although many decades long efforts have been put towards the development of very accurate mathematical models for each technology that can explain the fine grain dynamical behavior of these devices, still the very complex physics based models found in literature do not serve the purpose of providing good prediction on the global behavior of large number of those devices in a short amount of time. Since this is a surging problem for the simulation of large scale neuromorphic systems which may contain at least thousands of those devices, the research and development of novel modeling techniques and technology specific compact models has attracted the interest of many researchers both from the academia and the industry.

Furthermore devices of this kind also pose many challenges when operated in large arrays to perform computations. Each technology has it peculiarity and the device's non-idealities when compared to the ideal memristor theorized by Chua are many, to name a few:

- Nonlinearity of the Ohm's Law
- Drift of the memorized resistance over time

- Cycle-to-cycle variation of the programmed value
- Device-to-device variations

Although most of this nonideal effects have to be taken into account when designing a neuromorphic system, there are also approaches that try to exploit some of them as an advantage for the computation.

Thesis organization

This doctoral thesis is organized as follows. Chapter 1 introduces the Phase Change Memory (PCM) technology and proposes a physics-based compact model that is employed to compute and validate on experimental data the PCM Dynamic Route Maps. Chapter 2 introduces the Resistive Random Access Memory (ReRAM) nonvolatile memory technology and proposes a compact model suitable for memristor-based Spiking Neural Networks (SNN). Chapter 3 brings examples on the modeling of both Spiking/Artificial Neural Networks and more conventional linear algebra accelerators. Chapter 5 concludes the manuscript and gives an overlook on possible future developments of the work.

Main Contributions

The main contributions of this manuscript may be summarized with the following points:

- A novel physic-based current controlled model for Phase Change Memory devices.
- Computation and experimental validation of the PCM technology Dynamic Route Maps.
- Derivation of an almost analytical simplified model of the ReRAM memristor that involves two variables, the memductance, and the temperature, which can be related one-to-one with the synaptic efficacy and the calcium concentration in biological synapses

- Characterization of memristor-based spiking neural networks as discrete time nonlinear dynamical systems whose state variables are the mem-conductances and whose inputs and outputs are pre and postsynaptic spikes respectively.
- A case example of the design procedure for a memristor-based hardware accelerator.
- An in-depth study of the impact of non-idealities on memristor-based hardware accelerators for linear algebra problems solving with a particular focus on trade-offs and comparisons with the state of the art digital alternatives.

All the results presented in this manuscript are based on articles published by the author:

- Marrone, F., Secco, J., Kersting, B. et al. Experimental validation of state equations and dynamic route maps for phase change memristive devices. Sci Rep 12, 6488 (2022). doi: 10.1038/s41598-022-09948-6
- F. Marrone, G. Zoppo, F. Corinto and M. Gilli, "A Dynamic System Approach to Spiking Second Order Memristor Networks," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 69, no. 4, pp. 1641-1654, April 2022, doi: 10.1109/TCSI.2021.3137713.
- Zoppo Gianluca, Marrone Francesco, Corinto Fernando, Equilibrium Propagation for Memristor-Based Recurrent Neural Networks, Frontiers in Neuroscience, vol. 14, March 2020, doi: 10.3389/fnins.2020.00240
- G. Zoppo, A. Korkmaz, F. Marrone, S. Palermo, F. Corinto and R. S. Williams, "Analog Solutions of Discrete Markov Chains via Memristor Crossbars," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 68, no. 12, pp. 4910-4923, Dec. 2021, doi: 10.1109/TCSI.2021.3126477.

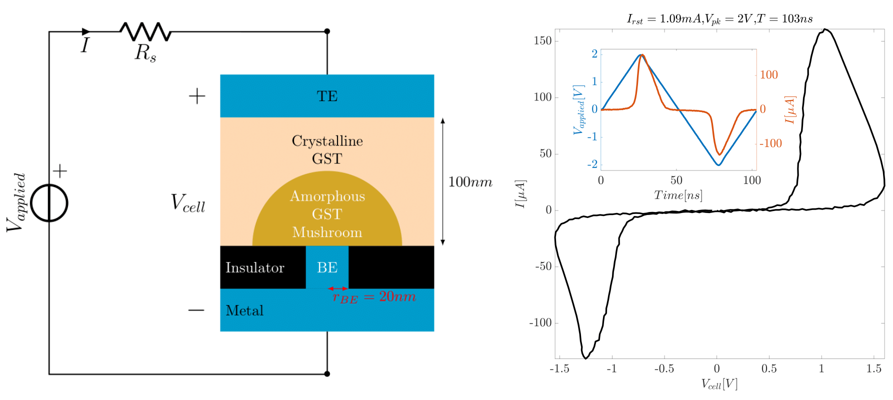
Chapter 1

Memristor Device Modeling: PCM

1.1 Introduction

Phase Change Memories (PCM) store data as the phase configuration of a layer of material between two metallic electrodes. The typical used materials include many different compounds of Ge, Te and Sb which all show high electrical resistivity in the amorphous phase and far lower resistivity in the crystalline phase. The memristance of a PCM device is tunable by changing the ratio between the amorphous and the crystalline fraction within the device volume usually by means of suitable electrical pulses, thus achieving a continuum of memristance (or memductance) states.

PCM devices show many fascinating nonlinear dynamical behaviors which all arise from the complex interaction between electrical, thermal and structural dynamics inside each device[12]. When a PCM cell is in mostly amorphous state (*OFF-state*), as the applied electrical fields is increased from relatively very low to higher values the conduction in the sandwiched material goes through ohmic, exponential and super-exponential regimes[13]. In the high-field *ON-state*, conduction through the amorphous phase becomes metal-like and the global flow of electrons in the PCM cell becomes dominated by the amorphous-crystalline Schottky barrier[14]. The event switching PCM devices from OFF to ON state is known as threshold switching which involves a feedback–driven thermo–electrical mechanism[15]. It starts when an externally applied field lowers the barrier between two close



(a) (b)

Fig. 1.1 **(a)** Schematic representation of a mushroom-type PCM device operated with a current compliance series resistor R_s. In the RESET state, the amorphous phase blocks the bottom electrode, and the device is in a high-resistance state. The effective thickness of the amorphous region is denoted by $u_a(t)$. $V_{\text{applied}}(t)$ is the externally applied input voltage, $I(t)$ is the current flowing through the PCM device and $V_{\text{cell}} = V_{applied} - R_s I$ is the intrinsic voltage drop on the device.**(b)** Measured pinched hysteresis loop for a mushroom PCM device stimulated by a 103 ns period triangular V_{applied} pulse of $2V$ peak voltage V_{pk}.

coulombic centers enough to induce an increase in the global conductivity of the device. This allows more current to flow and the temperature inside the PCM cell to rise. For high enough temperature the activation energy is strongly reduced and this in turn allows more electrons to flow through the material. This, in a very short time, leads to the threshold-switching event. Threshold switching plays a crucial role in the operation of PCM devices as it enables the fast *WRITE* operation by means of an abrupt temperature increase. Within the appropriate temperature range the *WRITE* operation can take the form of either a *SET* operation which increases the low-field conductivity by crystallizing the amorphous fraction or a *RESET* operation which does the opposite via the *melt-quenching* process which increases the internal PCM temperature up to the *GST* melting temperature $T_{MELT} \approx 877K$[16].

Accurate physical models have exploited integro-differential equations to describe the distinctive characteristic of each operating condition in PCM devices and thus numerical analysis is crucial to precisely capture the experimental observations [15]. In this Chapter the prototypical mushroom-type PCM device[17], depicted in Fig. 1.1a, is modeled as a memristive system as introduced in 1976 by S. Kang and L. O. Chua [4]. The PCM device studied in this Chapter shows the typical memristive pinched loop under bipolar periodic input, as exemplified in Figure 1.1b, and thus can be casted in the memristive systems class[3]. The aim is to represent PCM devices via a state–dependent Ohm's Law $v = R(\mathbf{x}, i)i$ linking the current i through the PCM cell and voltage v across its two terminals and a state equation $\dot{\mathbf{x}} = f(\mathbf{x}, i)$ which describes the dynamics of internal state variables $\mathbf{x}$. By meas of the memristor state–dependent Ohm's law it is possible to unfold complex dynamic behaviors in PCM devices via the concepts of Dynamic Route Map (DRM)[18]. If the state variable is a scalar $x \in \mathbf{R}$, the DRM parameterized by the input i consists in the plot of $f(x, i)$ in the plane $(\dot{x}, x)$. By varying the parameter i it is possible to picture the entire family of curves spanning the whole plane $(\dot{x}, x)$. Additionally in the last years, DRMs have also been found to be a convenient modeling tool for similar devices. Ascoli *et al.* have proven that, in principle, DRMs can help to investigate all the cases of switching dynamics in memristor devices[19], [20]. Afterwards others works have demonstrated how DRMs are a powerful tool for modeling of devices such as ReRAMs, or of complex systems such as *Cellular Nonlinear Networks* [21, 22].

The PCM devices modeled in this Chapter are the ones of mushroom-type kindly made available by Abu Sebastian's group at IBM Research in Zurich, Switzerland. They take their type acronym from the mushroom-like amorphous dome which is created during the melt–quenching process that brings the device in its OFF state. The phase-change layer consists in doped $Ge_2Sb_2Te_5$. The bottom metallic electrode (BE) is ≈ 20 nm in radius and ≈ 65 nm in length. The phase change material is ≈ 100 nm thick and extends up to the top electrode (TE) which is ≈ 100 nm in radius. In this Chapter it is shown how the modeling of this kind of devices can be successfully conducted via DRMs.

This Chapter is organized as follows: Section II goes deep into the physics of the device with an overview of the most accurate models for its dynamics and electric conduction, Section III introduces the novel proposed simplifications which enable the compact modeling of the device, Section IV reports the methodology to compute the Dynamic Route Maps of the PCM's first order dynamics, Section V concludes the Chapter and gives some future

directions for expand on this work.
The content of this Chapter is a re-elaborated version of a research paper published by the author [23].

1.2 Complete second order model

In Figure 1.2 is reported an effective block representation of the intricate interconnection between the various phenomena involved in PCM devices′ dynamics. It is composed, on first approximation, of two state variables T_{int} and u_a which are the temperature at the Amorphous/Crystalline Interface (ACI) and the amorphous dome effective radius respectively. In addition, the output current I (and the internally dissipated power $P = V_{cell}I$) are expressed via an algebraic relation which relate the input cell voltage drop V_{cell}, the ambient temperature T_{AMB} and the state variables.

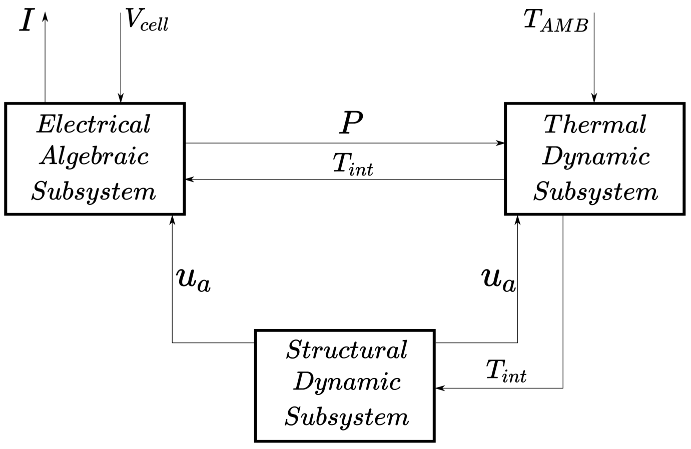

I
V_{cell}
T_{AMB}
Electrical Algebraic Subsystem
P
T_{int}
Thermal Dynamic Subsystem
u_a
u_a
Structural Dynamic Subsystem
T_{int}

Fig. 1.2 Simplified block representation of the PCM device subsystems. The system has two inputs respectively $V_{cell}(t)$ and T_{AMB} and one output $I(t)$. $V_{cell}(t)$ is the voltage drop on the PCM device. $I(t)$ is the current flowing through the device. T_{AMB} is the ambient temperature the PCM device is operated at. $P(t)$ is the algebraic electrical power dissipated by the device and computed as the product current $I(t)$ times voltage $V_{cell}(t)$. $u_a(t)$ is the amorphous mushroom thickness and its evolution is determined by the crystallization dynamics. $T_{int}(t)$ is the temperature at the amorphous-crystalline interface and its evolution is determined by the heat balance equation.

1.2.1 Complete Ohm's Law

Regarding the conduction mechanism in the amorphous-phase *GST*, more and more precise models have been proposed over the years to capture how it is influenced by the two state variables $(u_{\mathrm{a}}, T_{\mathrm{int}})$ and the applied voltage V_{cell}. Most of these models result accurate in specific electric-field domains.

In order to give an overview of the most common electric-field domain specific conduction model it should be introduced, for convenience, F_{T} as the transition field:

$$F_{\mathrm{T}} = \frac{q_{\mathrm{e}}}{\pi \epsilon_{\mathrm{r}} \epsilon_0 s^2}$$

where q_{e} is the fundamental electron charge, $\epsilon_{\mathrm{r}} \approx 10$ is the relative permittivity of the amorphous phase, ϵ_0 is the vacuum permittivity and s is the average distance between defects in the amorphous phase.

On the one hand for low to moderate electric field regimes (i.e. $F = \frac{V_{cell}}{u_a} << F_T$), the simple Poole [24] results accurate in describing electrical conduction. On the other hand in the high field domain ($F >> F_T$), the Pool-Frenkel [25][26][27] model gives the best approximation.

Below threshold switching (i.e. the device is in OFF state), the 3D Poole-Frenkel [28][29] emission of carriers from a two-center Coulomb potential well was experimentally demonstrated to best describe in an unified manner the conduction through *GST* material for both low and, moderate and high fields F.

3D Poole-Frenkel conduction formulates the density of free carriers under an applied field $F = \frac{V_{cell}}{u_a}$ at an interface temperature T_{int} via the integral expression:

$$n(F, T_{int}) = \frac{K}{2} \int_0^{\pi} exp(-\frac{E_a(T_{int}) - E_{PF}(F, \theta)}{k_B T_{int}}) sin(\theta) d\theta$$

where θ is the angle between the electric field and a possible direction of escape from the potential well and $E_a(T_{int}) = E_{a,0} - \frac{aT_{int}^2}{b+T_{int}}$ is the interface temperature dependent activation energy and $E_{PF}(F, \theta) = -max_r \Phi(r, \theta, F)$ is the Poole-Frenkel field-induced energy barrier lowering between two adjacent potential wells. These are described in this model by the electric potential profile $\Phi(r,\theta,F) = -qFrcos(\theta) - \frac{\beta^2}{4q}(\frac{1}{r} + \frac{1}{s-r}) + \frac{\beta^2}{qs}$ with $\beta = \frac{q^2}{\sqrt{q\pi\epsilon_r\epsilon_0}}$.

The amorphous phase conductivity per unit of area can be then computed as

$$\sigma(F,T_{int}) = q\mu(F)n(F,T_{int})$$

where $\mu(F) = \frac{\mu_0}{\sqrt{1+(\mu_0 F/v_{sat})^2}}$ is the field dependent mobility of carriers. The density of current $j(F,T_{int})$ can be readily obtained as $j(F,T_{int}) = \sigma(F,T_{int})F$ and the current flowing through the entire cell is found by multiplying the current density $j(F,T_{int})$ times the effective bottom electrode contact area A. This should be computed via the effective bottom electrode radius and not the physical radius of the heater.

Although 3D Poole-Frenkel emission model captures the conduction in PCM materials up to the threshold switching event, nonetheless it does not hold accurate above threshold switching, i.e. after the amorphous phase is switched to ON state.

Phase Change Memory cells can be *reset* to high memristance states (OFF states) via relatively high current pulses that rises the ACI temperature up to the melting temperature T_{MELT}. Given that most of the power is dissipated at the BE, higher reset currents flowing through the device imply higher power dissipation which cause T_{MELT} to be reached further away from the BE. The resulting amorphous mushroom that is created has an effective amorphous thickness u_a which is proportional to the reset current I_{RESET}. This is shown in Figure 1.7a where the $(i - v)$ characteristics under a ramp input are plotted for different reset conditions reported in the figure legend.

1.2.2 Complete PCM Dynamics

The thermal dynamics of PCM devices follows the typical Newton's law of cooling. The state variable of interest for this devices is T_{int} which is the temperature at the ACI and drives the amorphous dome crystallization process. It is algebraically obtained as $T_{int} = T_{AMB} + T_{SH}$ where T_{SH} is the intrinsically dynamic part of T_{int} and physically represents the cell Joule self-heating temperature increment while T_{AMB} is the constant ambient temperature. The dynamics of T_{SH} obeys the Ordinary Differential Equation (ODE):

$$\frac{d}{dt}T_{SH}(t) = \frac{1}{\tau_{th}}(R_{th}(u_a(t))P - T_{SH}(t)) \tag{1.1}$$

where $\tau_{th} = R_{th}(u_a(t))C_{th}(u_a(t))$ is the thermal time constant and $R_{th}(u_a(t))$, shown in Figure 1.3a, is the amorphous-thickness-dependent thermal resistance of the amorphous-phase dome that was estimated via FEM simulations in COMSOL [30] as the thermal resistance between the ACI and the external environment. Under the assumption of the BE electrode being being entirely blocked by the amorphous mushroom, an accurate linear approximation of $R_{th}(u_a)$ can be given, as

$$R_{th}(u_a(t)) = -\kappa u_a(t) + R_{th,0} \tag{1.2}$$

where, $\kappa \approx -0.024\frac{K}{\mu W nm}$ and $R_{th,0} \approx 1.908\frac{K}{\mu W}$.

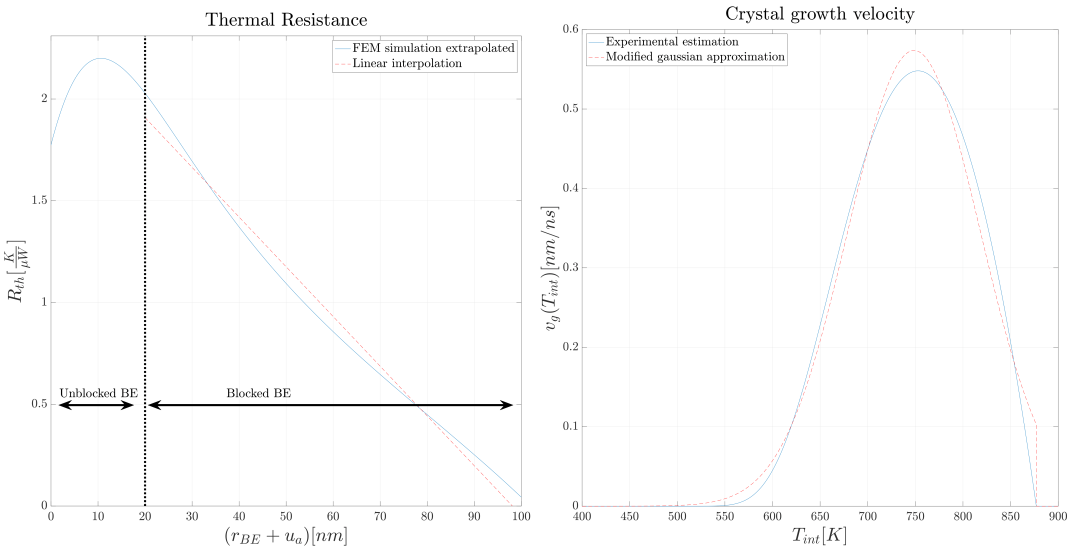
Given the linearly decreasing approximation of $R_{th}(u_a(t))$, as the thermal capacitance $C_{th}(u_a(t))$ is directly proportional to the amorphous dome volume, it is legitimate to assume $\tau_{th} \approx const$. This was found to be on the order of tens of nanoseconds[31].

The complete thermal dynamics of mushroom–type PCM cells, described by $T_{int} = T_{AMB} + T_{SH}$ and (1.1), can be given a circuital interpretation as reported in Fig. 1.4.

The temperature at the ACI T_{int} drives the crystallization process, i.e. the reduction in volume of the amorphous dome by the rearrangement of disordered atoms at the interface into a regular lattice. The structural dynamics in PCM cells governs the evolution of the effective amorphous thickness $u_a(t)$. It should be pointed out that this differs from the real amorphous thickness which is the distance of the ACI from the center of the amorphous mushroom (see [32]). The evolution of $u_a(t)$ is described by the ODE:

$$\frac{d}{dt}u_a(t) = -v_g(T_{int}(t)) \tag{1.3}$$

where $u_a \in [0, 80]nm$. The function $v_g(\cdot)$ is the interfacial-temperature-dependent crystal growth velocity which was estimated in [30] and is reported in Figure



(a) (b)

Fig. 1.3 **(a)** in continuous blue the effective thermal resistance, extrapolated from FEM simulations, as a function of the amorphous thickness $u_a(t)$. In dashed red the linear approximation of the effective thermal resistance in the blocked BE condition. **(b)** in continuous blue the experimentally estimated GST crystal growth velocity $v_g(T_{int})$ as a function of the interface temperature T_{int}. In dashed red its Gaussian approximation.

1.3b. A Gaussian approximation of $v_g(T_{int})$ has also recently been proposed in [33] as

$$v_g(T_{int}) = A_{v_g} exp(-(\frac{T_{int}-T_0}{\sigma_T})^2) \tag{1.4}$$

$A_{v_g} \approx 0.57\frac{nm}{ns}$, $T_0 \approx 749\,\mathrm{K}$ and $\sigma_T \approx 98K$. Higher order effects such as structural relaxation dynamics of the amorphous phase GST are not taken into account in this model and their effect can be considered negligible given the narrow observation time window used in collecting the experimental data.

1.3 Simplified first order model derivation

In this Section a simplified first order model of the PCM dynamics is derived which is employed, later in this Chapter, to compute the Dynamic Route Maps of the Mushroom type phase change memory device under study.

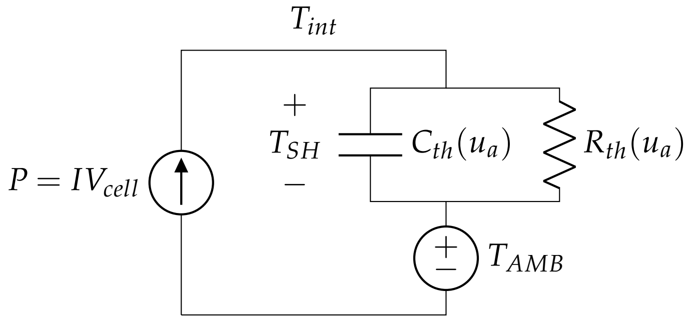


Fig. 1.4 First order dynamical thermal circuit that takes into account the presence of $C_{th}(u_a)$. Its dynamics can be described through Equation (1.1).

1.3.1 Simplified Ohm's Law

The read operation in PCM devices refers to the retrieval of the phase-configuration information by means of low amplitude current pulses. By assuming the BE to be fully blocked (i.e. $u_a > 0$) and that the applied field is very low (i.e. $F = \frac{V_{cell}}{u_a} \to 0$), then the voltage drop between the device electrodes can be well approximated by a linear relation $V_{cell} = R_m(u_a, T_{int})I$, as shown from experimental measurements in Fig. 1.7b, where the memristance, $R_m(u_a, T_{int})$ is given by

$$R_{\mathrm{m}}(u_{\mathrm{a}}, T_{\mathrm{int}}) = K^{'} u_{\mathrm{a}} exp(\frac{E_{\mathrm{a}}(T_{\mathrm{int}})}{k_{\mathrm{B}} T_{\mathrm{int}}}) \tag{1.5}$$

where k_{B} is the Boltzmann constant, $E_{\mathrm{a}}(T_{\mathrm{int}})$ is the temperature-dependent activation energy and $K^{'} = \frac{1}{\pi r_{\mathrm{BE}}^2 q_{\mathrm{e}} K \mu_0}$. This formulation of the memristance $R_{\mathrm{m}}(u_{\mathrm{a}}, T_{\mathrm{int}})$ can be readily derived from the first order expansion of the Poole conduction model for $F \to 0$.

The write operation in PCM devices involves the fast dissipation of power caused by current pulses which induces an amorphous-to-crystalline transition at the ACI. This happens via the threshold switching event, that puts the phase-change material sandwiched between the metal electrodes into a very low resistance state independent of the long–term memory state variable u_a. The voltage drop V_{cell}, after a short transient has passed since the threshold switching, is confined to a narrow voltage band around $V_{cell,ON} \approx 0.8V$ as shown in Fig. (1.7c) and Fig. (1.6). This can be explained as the very-low after-switching resistance state of the amorphous phase makes the conduction dominated by the barrier at the metal-semiconductor contact junction and by the resistance of the metallic electrodes. A simplified write model can thus be introduced as a reduction of the complex conduction mechanism to a simple nonlinear current-controlled resistor whose behavior, for high enough currents, is fairly well described by an ideal voltage generator of value $V_{cell,ON}$.

The two-domain approximation introduced above and illustrated in Fig. (1.7d can be summarized in the following state-dependent Ohm's Law

$$V_{cell} \approx \begin{cases} R_m(u_a, T_{int})I & I \ll I_{TH} \\ 0.8V & I \gg I_{TH} \end{cases} \tag{1.6}$$

where the well-below-threshold-switching model ($I \ll I_{\mathrm{TH}}$), represented with circuital symbolism in Fig. (1.7e), serves the purpose of recovering the phase configuration and the well-above-threshold-switching model ($I \gg I_{\mathrm{TH}}$), illustrated with circuital symbolism in Fig. (1.7f), is meant to accurately describe the cell power dissipation during the write phase and thus give good predictions about the temperature rise T_{SH} at the ACI. This is crucial in order to predict the internal state variables $(u_{\mathrm{a}}, T_{\mathrm{int}})$ evolution during the information storing phase and therefore it comes very handy to compute and plot the Dynamic Route Maps.

1.3.2 Simplified temperature dynamics

In order to compute the conventional Dynamic Route Maps it is necessary to reduce the second order model to first order. By considering that the input variables (i.e. V_{cell} and T_{AMB}) vary slowly with respect to the PCM cell thermal time constant $\tau_{th} \approx 10ns$, then a first order model can be obtained by reducing (1.1) to the algebraic equation:

$$T_{SH}(t) = R_{th}(u_a(t))P \tag{1.7}$$

This simplification lets the interface temperature to be computed as $T_{int} = T_{AMB} + T_{SH} = T_{AMB} + R_{th}(u_a)P$ where the input power $P = IV_{cell}$ as shown by the circuital representation in Fig. (1.5).

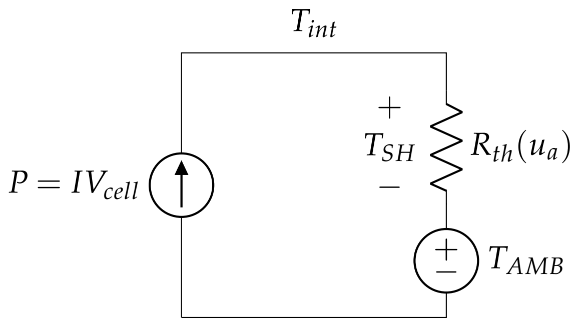


Fig. 1.5 For the sake of the presented model the input signals to the PCM device are considered to be at least twice longer in duration than τ_{th}, for this reason it is taken into account **(b)** described by Equation (1.7).

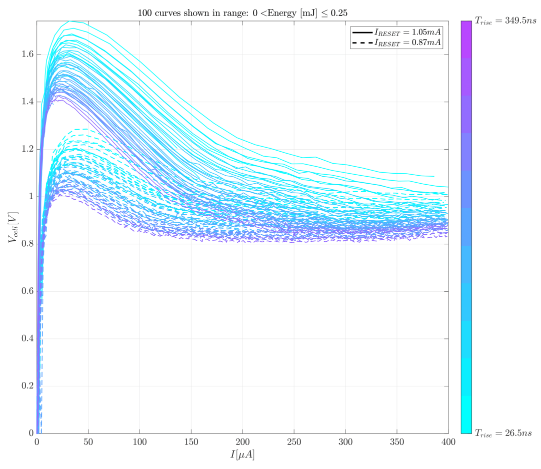
1.4 Dynamic Route Maps

In this Section the Dynamic Route Maps of the PCM mushroom cell's first order dynamics are computed by employing the previously developed simplified first order model in conjunction with the simlified Ohm's Law.

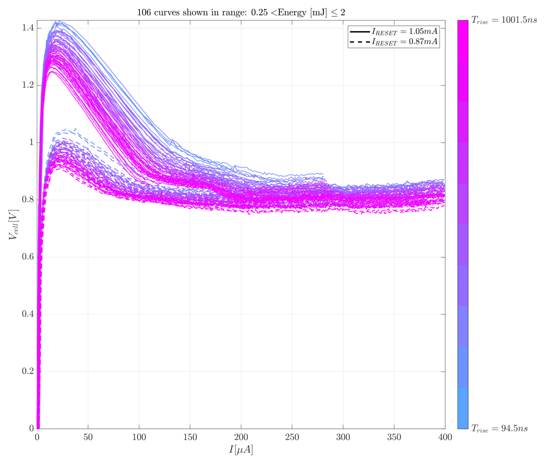
1.4.1 Numerically computed DRMs

Using the above introduced current–controlled model of PCM cells a set of simulations were performed to compute and plot the theoretical DRMs of a PCM cell. In order to show how DRMs fill the $(\dot{u}_a, u_a)$ plane it was performed an input current I sweep ranging from I =150 μA to 650 μ A. The computation was carried out combining Eqs. 1.3 and 1.4 and the results were plotted in Fig. 1.8. In all the reported case the initial u_a =40 nm and each DRM corresponds to a different constant current flowing through the cell for an ideally infinite time.

It is possible to observe that the DRMs calculation considered a slight numerical correction inside the relation $R_\mathrm{th}(u_\mathrm{a}(t))$ described in 1.2 and shown



(a)



(b)

Fig. 1.6 DSO measured $I - V_{cell}$ characteristics of a GST PCM cell, reset with two different reset currents I_{RESET}, driven by a voltage ramp V supplied by an Arbitrary Waveform Generator through a series resistor $R_S \approx 5.7k\Omega$. V_{cell} responses computed as $V_{cell} = V - R_S I$. Rise times T_{rise} spanning in the range $[26.5, 1001.5]ns$ and supplied voltages V in the range $[1,4]V$. Energy estimated as the trapezoidal approximation of the integral of the dissipated power $P = V_{cell} I$ along the curve from $(I, V_{cell}) = (0,0)$ up to the threshold switching peak. In **(a)** the I, V_{cell} characteristics for the PCM cell when not sufficiently heated. The final V_{cell} plateau in **(a)** varies substantially. In **(b)** the $I - V_{cell}$ characteristics for the PCM cell when sufficiently heated. The final V_{cell} plateau in **(b)** results to be narrowly centered around $V_{cell,ON} \approx 0.8V$.

in Fig. 1.3a. The added correction is aimed at compensating the numerical approximations generated by FEM simulations used to compute the $R_{\text{th}}(u_{\text{a}}(t))$ relation. The magnitude of the correction was empirically estimated on the experimental measurements from the PCM cell, as it will be later shown in this Section.

The simulation results show that for a given initial condition of u_{a}, the resulting final effective thickness of the GST mushroom depends on the current flowing through the device. This is due to the fact that T_{int} increases with the dissipated power P as made clear by Eq. 1.1. In fact, different dynamics of the effective thickness u_a can be observed in the cell for increasing values of current. In order to exemplify this, it is worth considering that for $I = 650\,\mu$A the effective thickness of the amorphous mushroom cannot reach $0\,nm$ but it stops its regression around 5.5 nm, while for instance, a current $I = 450\,\mu$A makes the amorphous region completely vanish. This behavior can be easily explained by resorting to Fig. 1.3b. As aforementioned, a higher input cur-

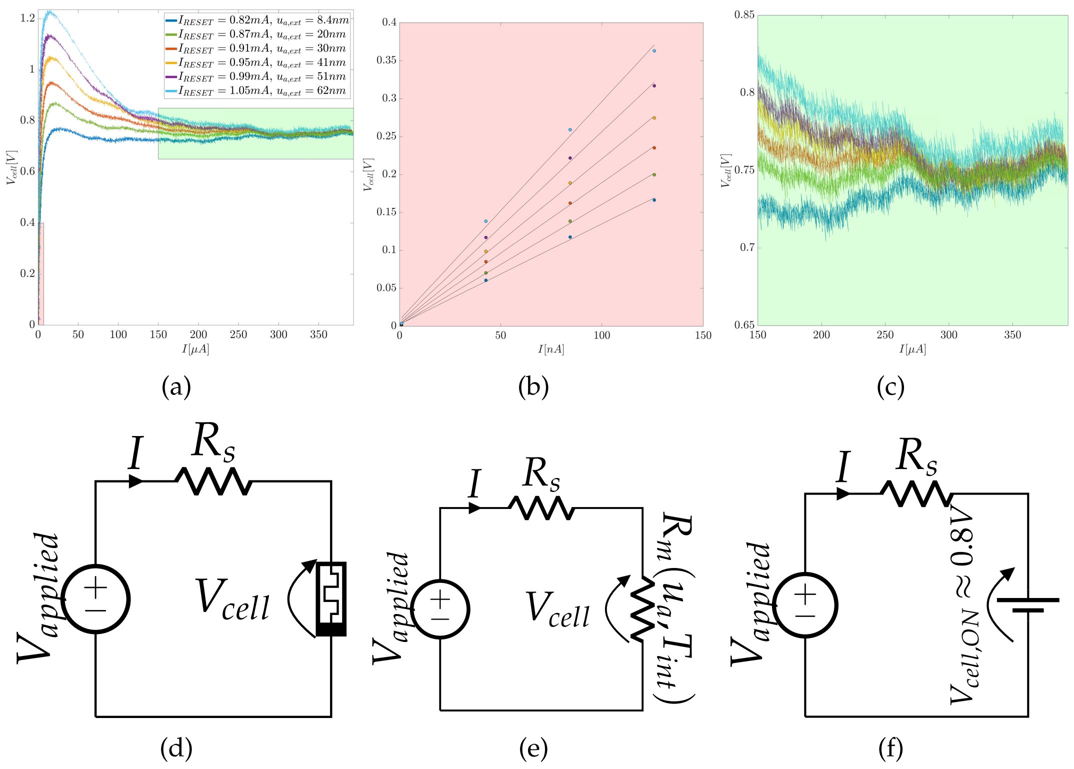


Fig. 1.7 **(a)** [Global (i,v) curves] DSO measured $I - V_{cell}$ characteristics of a GST PCM cell, reset with different reset currents I_{RESET}, showing the same plateau V_{cell} value for $I \geq I_{th} \approxeq 100\mu A$. Driving voltage $V_{applied}$ supplied by an by an Arbitrary Waveform Generator through a series resistor $R_S \approx 5.7k\Omega$. V_{cell} responses computed as $V_{cell} = V - R_S I$. Ramp rise time $T_{rise} = 3\mu s$. **(b)** [READ operation (i,v) curves] SMU measured low-field $I - V_{cell}$ characteristics of the same PCM cell, reset with the same currents I_{RESET} as those on the main axes, showing an almost ideal linear behavior for very low current/voltage value. **(c)** [WRITE set operation (i,v) curves] Zoom-in of the write region $I - V_{cell}$ characteristics, it is noticeable how V_{cell} is narrowly concentrated around $0.8V$.**(d)** [Global model] Global model of a PCM device as a nonlinear time-variant dynamical system **(e)** [READ operation model] READ domain model of a PCM device as a state-dependent linear resistor **(f)** [WRITE operation model] WRITE domain model of a PCM device as an ideal voltage generator $V_{cell,ON}$.

rent leads to a higher T_{int}, which may lead to blocking the GST crystallization dynamics when $T_{int} \approx T_{MELT}$ which is captured as a nonzero equilibrium of u_a. In the proposed model, this physically means that a fraction of the GST material close to the BE is molten again at 650 μA given that v_g drops to zero only for $T_{\rm int} = T_{\rm MELT}$ as shown in Figure (1.3b).

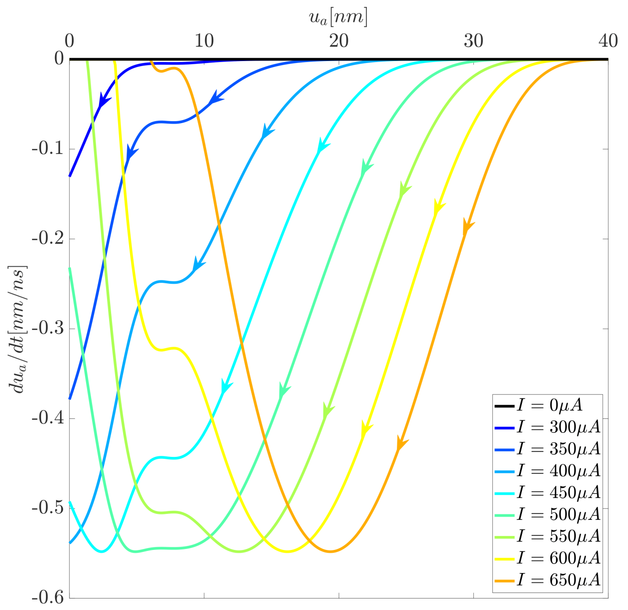


Fig. 1.8 DRM computed, for a fully switched PCM Amorphous Mushroom cell in the blocked-BE condition, by means of the proposed write model parameterized on various levels of injected current I.

1.4.2 Experimental data acquisition details

The Agilent 81150A Pulse Function Arbitrary Generator was used to apply the input signals while the Tektronix TDS3054B oscilloscope served to measure both AC voltages and currents (i.e. the applied voltage V_{applied} and the flowing current I). The oscilloscope sampled those currents and voltages with a frequency $f_s = 2.5\,\text{GHz}$. V_{applied} was sampled by the oscilloscope coming directly from the generator. The current flowing through the device was measured putting the oscilloscope in series operating as an ammeter. An isolated integrated linear resistor, R_s in close proximity to the device was also measured to be $\approx$ 5.7 $k\Omega$ and was used to compute the actual cell voltage drop V_{cell} during the post processing of the data. Given the involved frequencies of operation and the length of the wires (measured $\approx 60\,\text{cm}$), the phase shift between the voltage V_{applied} signal and the flowing current had to be taken into account. This delay was measured to be $\approx 6.4\,\text{ns}$.

All the measurements were preceded by a square reset pulse 950 μA in amplitude, lasting 1 μs and sharply edged both on the rising and falling fronts (7.5 ns each). The reset created an amorphous dome of $u_a \approx 40\,\text{nm}$.

This effective mushroom radius was estimated by fitting the resulting $I - V$ characteristics to a transport model proposed by Ielmini and Zhang[34]. A sequence of alternating write and read pulses were applied 20 μs after each reset pulse. All the write pulses with 7.5 ns leading and trailing edges were $\approx$ 2.93 V in amplitude and lasted for variable intervals of time ranging between 25 ns and 121 ns).After 25 ns, a read ramp voltage pulse with amplitude of 0.5 V and 50 ns duration followed the each writing pulse. After 25 ns break from the reading voltage ramp trailing edge (7.5 ns in duration) a new writing pulse was applied. This process was repeated for 10 write/read pulse couples

1.4.3 Comparison between numerical and experimental results

Following the same methodology applied to calculate the theoretical DRMs in Figure 1.8, the dynamic route maps were also obtained from experimental data collected on a GST PCM mushroom cell in the laboratory of IBM facility in Zurich. Fig. 1.10 report a representative set of three taken from the performed measurements. They differ from each other by the rectangular current pulse duration W and the number of write pulses applied: three pulses of $W = 121ns$ in Fig. 1.10(a), six pulses of $W = 42ns$ in Fig. 1.10(b) and seven pulses of $W = 25ns$ in Fig. 1.10(c). The first column of Fig. 1.10 reports the measured voltage drops on the cell (left axis), which can be found to settle at $V_{cell} \approx 0.8V$ (solid blue line in the graphs) for each given pulse after a very short transient, consistently with the data shown in Figure 1.6. In the same graphs in the first column (traced by a solid red line) is also reported the dissipated power P (right axis), which stays constant during each WRITE pulse. The reduction of u_a is measured, after each write pulse, via the READ operation as a change in $R_m(u_a, Tint)$ which is show by I-V curves displayed in the middle column of Fig. 1.10. The experimentally measured power waveforms reported in the left column of Fig. 1.10 were used to compute the respective transient DRM shown in the right column of Figure 1.10.

Once the dynamic routes have been calculated from experimentally collected power waveforms in a physical device, the same experiments performed on the PCM in the laboratory have been reproduced in a numerical simulation using the above developed domain specific model and considering the injected current to be constant and whose value was estimated from the collected current waveforms as the average during the middle of the write pulse. The results of the simulation are shown in Figure 1.11. Figures 1.11(a), 1.11(b) and 1.11(c) are referred to the measurements shown in Figures 1.10(a), 1.10(b) and 1.10(c), respectively. The comparison between the experimental measurements and the results obtained from the simulation (superimposed black lines) is shown on the I-V curves plotted in the fist column.

The correction applied to the thermal resistance $R_{\mathrm{th}}(u_{\mathrm{a}}(t))$ described in 1.2, was numerically obtained comparing the results. The numerical correction, from a physical point of view, can be considered as a refinement of the estimation of the time constant τ_{th}, introduced in the dynamic thermal circuit of the PCM cell shown in Fig. 1.4.

The dynamic route maps were computed using the simplified write model with the value of the input power constant and equal to the average delivered power measured during the write pulses plateau as reported in Fig. 1.10). The resulting DRMs are reported in the figures of the second column of Fig. 1.11 as solid red lines (DC dynamic routes). It can be noted that these dynamic routes "envelope" the TRANSIENT dynamic routes (i.e. the ones calculated from the experimental data shown in Fig. 1.10, solid blue line). This is due since the TRANSIENT dynamic routes take into consideration also the peaks of the dissipated power P. Each of the DC dynamic routes, was calculated considering the initial condition of $u_{\mathrm{a}}(t)$ at the beginning of each pulse train. Each route was traced considering an initial condition $u_{\mathrm{a}}(t=0) \approx 40\,\mathrm{nm}$ in order to generate the same effect of the corresponding reset pulse.

From this calculation, it can be observed that DRMs deserve to be considered a powerful tool for predicting the behavior of the PCM devices for system design. This is clear when looking at Fig. 1.9, from a high memristance (OFF state) associated with $u_a(t_0) \approx 50\,\mathrm{nm}$, the designer can choose to reach three different low resistance states by tuning the SET programming current. The understanding of how the first order dynamics evolves together

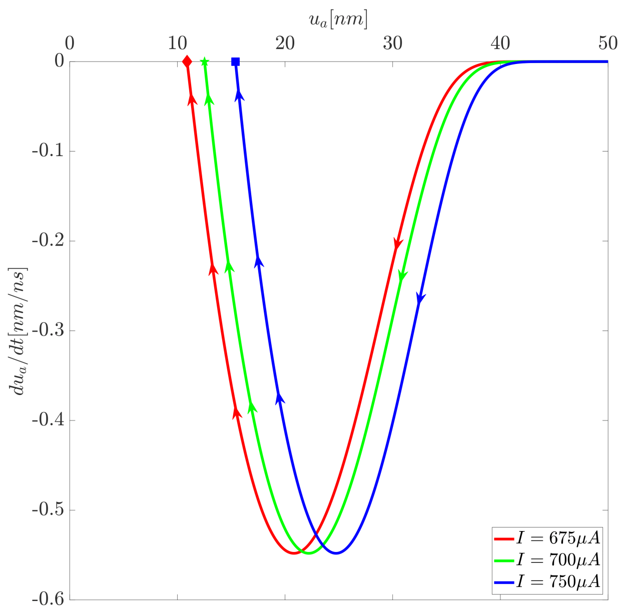


Fig. 1.9 Three dynamic route maps parameterized on different SET current values. Each route, when travelled, leads to a distinct nonzero equilibrium of the first order dynamic state variable u_a.

with the proof of existence of nonzero equilibria of the state variable u_a can be exploited to design programmable circuits such as programmable amplifiers and tunable filters as found in [23].

1.5 Conclusion

Phase Change Memories are good candidates for replacing CMOS technology in the next decades. They have already found application as non-volatile memory elements, basic computing elements for in-memory and neuromorphic computing and as components of reconfigurable electronic circuits as well. In this Chapter a current-controlled memristor model of PCM devices was developed based on

1. a state–dependent Ohm's law (1.6) which corresponds to the PCM conduction model during READ/WRITE operations

2. the dynamic route maps incorporating the dynamics of the long-term-memory state variable u_a under different input current pulses applied to the PCM.

The fundamental insight drawn from the DRMs for PCM is possibility of designing input current pulses to drive the thickness of the amorphous mushroom ($u_{\mathrm{a}}(t)$) and thus the corresponding low-field resistance. Thanks to the knowledge of the temporal evolution of $u_a(t)$ (i.e. Fig. 1.8), suitable current pulses can be chosen to modulate dissipation and manage the crystallization process. As aforementioned, exploiting DRMs, it is possible to derive the characteristic and model complex systems based through the interaction of memristive elements, as also shown with different technologies in the work by Ascoli *et al* [19]. The work here presented is thus to be considered as a significant step towards the inclusion of memristor–based PCM model in automatic design tools for programmable analog circuits and also for tunable synaptic elements in neuromorphic circuitry, overcoming the limitations given by the complex description of the dynamics of these elements.

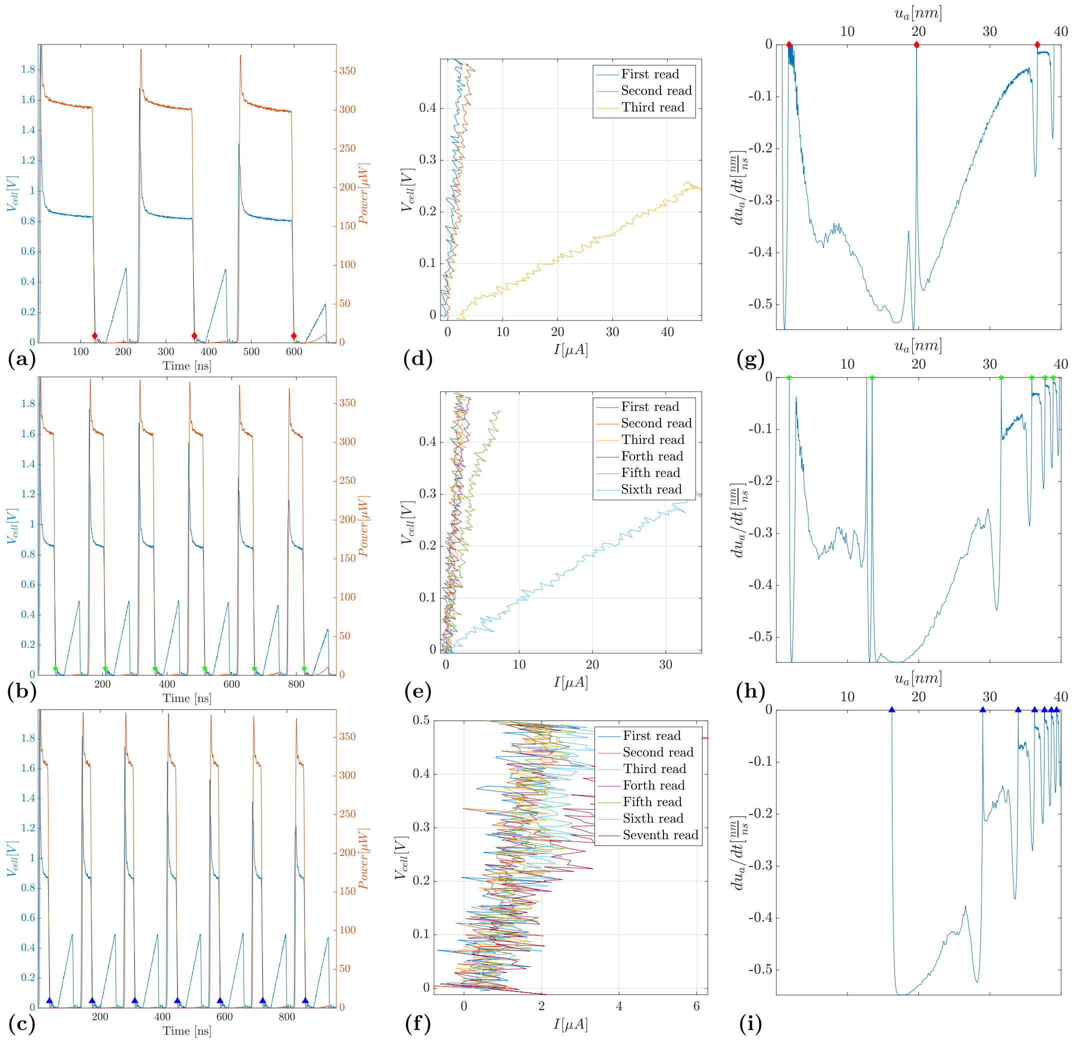


Fig. 1.10 (Left Column) Collection of experimentally measured cell voltage drops V_{cell} and the corresponding absorbed electrical *Power* for varying write pulse duration W. Each square high–current write pulse is interleaved with a below threshold triangular read pulse. (a) Three $W = 121ns$ duration pulses. (b) Six $W = 42ns$ duration pulses. (c) Seven $W = 25ns$ duration pulses. (Center Column) Collection of measured read pulses represented on the current–voltage plane for varying pulse duration W. The stored memory state u_a is here directly proportional to the slope of each curve being this a measure of the resistance state. (Right Column) Dynamic Routes, for varying pulse duration W, computed by means of the proposed model using as input the measured power signal on the left column.

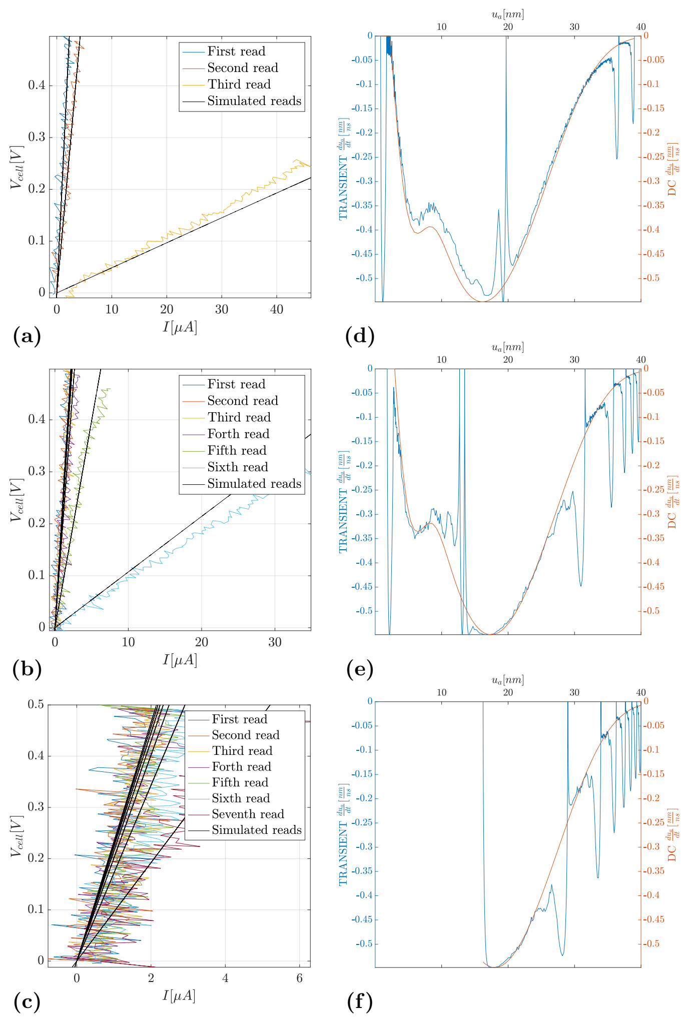


Fig. 1.11 (Left column) Matching between the measured read pulsed on the voltage–current plane, as reported in the central column of Figure 1.10, and the expected voltage–current curves as obtained by the same dynamical simulation generating the dynamic routes in the right column of Figure 1.10. (a) Three $W = 121ns$ duration pulses. (b) Six $W = 42ns$ duration pulses. (c) Seven $W = 25ns$ duration pulses.(Right column) Superposition of the computed dynamic routes, as reported in Figure 1.10, and the dynamic routes computed for a constant input power. The former are shown on the left axis as the TRANSIENT dynamic routes while the latter are on the right axis as the DC dynamic routes. The value of the constant input power used to compute the DC dynamic routes is the average measured delivered power during the write pulses plateau as reported in Figure 1.10.

Chapter 2

Memristor Device Modeling: ReRAM

2.1 Introduction

Resistive Random Access Memory (ReRAM) has been considered so far one of the most promising candidates for nonvolatile memory technology that could substitute NAND flash in the next decades. Furthermore, ReRAM devices arranged in crossbar arrays show low energy consumption and excellent scalability which make this technology a contender to become the standard in future brain-inspired processors.

In particular, tantalum oxide has become the material that during the years has caught most of ReRAM researchers' attention due to the demonstrated good memory performance metrics in many key aspects including device size (down to $28nm$), switching speed ($< 1\,ns$), endurance (up to 10^{12}) and multilevel switching capability (up to $3\,bit$)[35][36][37][38].

The ReRAM devices modeled in this Chapter are manufactured as a multilayer Pd/Ta_2O_{5-x}/TaO_y/Pd structure, as depicted in Fig. 2.1(a), where the functional layer comprises two layers of tantalum oxide with different stoichiometrics: an oxygen-rich Ta_2O_{5-x} layer at the top and an oxygen-deficient TaO_y layer at the bottom. Because of the different oxygen concentration in the two layers, oxygen vacancies (V_O) can be transferred between the top layer and the bottom layer by means of an externally applied electric field, leading to controllable resistive switching [37][39].

The ReRAM devices referred in this Chapter are the ones from [40] which are $1\mu m \times 1\mu m$ in size and were fabricated in a crossbar structure on a SiO_2 ($100\,nm$)/Si substrate with electrodes patterned by traditional photolithography. Initially, 35-nm bottom Pd electrodes were deposited by photolithography, e-beam evaporation, and lift-off processes. Next, a 35-nm TaO_y base layer was deposited via DC reactive sputtering with a Ta metal target in an Ar/O_2 gas mixture at 400 °C. The pressure of Ar/O_2 was $\approx 666\,Pa$, and the oxygen partial pressure in the mixture was $\approx$ 3%. A $5\,nm$ Ta_2O_{5-x} switching layer was then deposited via RF sputtering with a Ta_2O_5 ceramic target in a pure Ar atmosphere a $\approx 666\,Pa$. A $30\,nm$ top Pd electrode was then deposited by photolithography, e-beam evaporation, and lift-off processes. Eventually, a reactive ion etching process with SF_6/Ar mixture was performed in order to expose the bottom contacts.

A common IV loop curve of the $Pd/Ta_2O_{5-x}/TaO_y/Pd$ device is shown in Fig. 2.1(b). The device shows the typical memristive pinched loop with low threshold switching voltages. For voltages $\approx -0.9V$ the device is set from high-resistance (OFF) state to low resistance (ON) state and for voltages $\approx 1.1V$ it is reset from ON-state to OFF-state.

The resistive switching phenomenon can be explained with the V_O transfer between the Ta_2O_{5-x} and TaO_y layers[37]. As the two layers are in series, the total resistance of the device is primarily determined by the oxygen-rich Ta_2O_{5-x} layer. Hence a negative voltage lets V_O vacancies migrate to the Ta_2O_{5-x} layer and lowers the device resistance, while a positive voltage pushes those V_O vacancies to migrate back from the Ta_2O_{5-x} layer with a strong increase of the device resistance [37][39].

This Chapter is organized as follows: Section II goes deep into the physics of the device with an overview of the most accurate models for its dynamics and electric conduction, Section III introduces the novel simplified second order model of the device, Section IV employs the introduced model to compute the memristor response to various synaptic protocols, Section V proposes a study which demonstrates the model robustness to parameter variations, Section VI concludes the Chapter and gives some future directions

for expand on this work.
The content of this Chapter is a re-elaborated version of a research paper published by the author [1].

2.2 Complete second order model

In this Section the Pd/Ta_2O_{5-x}/TaO_y/Pd memristor physics is introduced in detail with a focus on the conduction model and accurate second order dynamics from [40].

2.2.1 Complete Ohm's Law

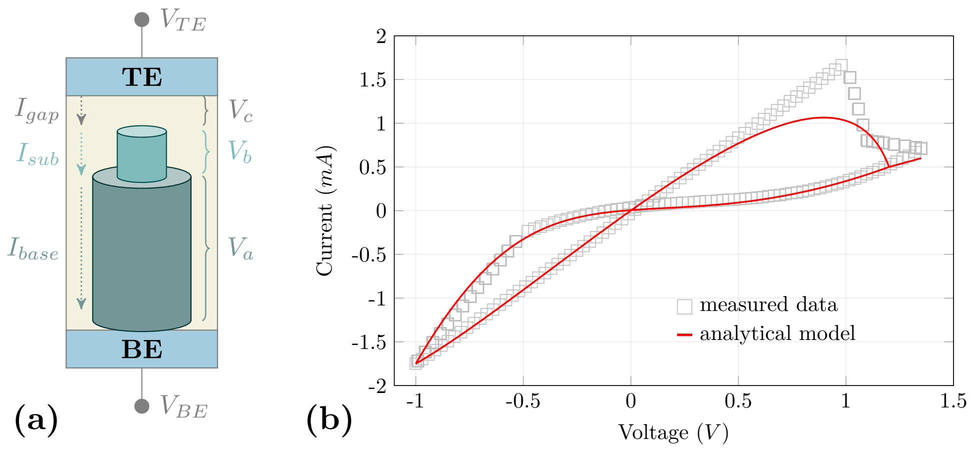


Fig. 2.1 (a) Schematic illustration of the device showing the filament structure. (b) Typical IV loop of a Pd/Ta_2O_{5-x}/TaO_y/Pd memristor.

The Pd/Ta_2O_{5-x}/TaO_y/Pd memristor considered in this derivation is schematically depicted in Fig. 2.1(a) and its Ohm's Law can be readily obtained considering its three conductive regions in series: the base-Conductive Filament (CF), the sub-CF and the depleted gap. The current flowing through those regions depends from the drop on each region and, in accordance with *SI Appendix* of [40], obeys 2.1.

$$\begin{aligned}
\mathbf{I}_{base} &= \frac{\pi r_0^2}{\rho \mathcal{L}_0} \mathbf{V}_a \\
\mathbf{I}_{sub} &= \frac{\pi \mathbf{r}^2}{\rho (L - \mathcal{L}_0 - \mathbf{g})} \mathbf{V}_b \\
\mathbf{I}_{gap} &= I_0 \exp\left(\frac{\mathbf{g}}{g_m}\right) \sinh\left(\frac{\mathbf{V}_c}{V_0}\right)
\end{aligned} \tag{2.1}$$

By combining 2.1 with the charge conservation law in 2.2 and the Kirchhoff Voltage Law in 2.3

$$\mathbf{I}_{base} = \mathbf{I}_{sub} = \mathbf{I}_{gap} = \mathbf{i}(t) \tag{2.2}$$

$$\mathbf{V}_a + \mathbf{V}_b + \mathbf{V}_c = \mathbf{V}_{TE} - \mathbf{V}_{BE} = \mathbf{v}(t) \tag{2.3}$$

it is possible to analytically describe the Pd/Ta_2O_{5-x}/TaO_y/Pd memristor as a two terminal circuit element, with the following relation between the voltage $\mathbf{v}(t)$ and the current $\mathbf{i}(t)$:

$$\begin{aligned}
\mathbf{v}(t) &= \frac{\rho \mathcal{L}_0}{\pi r_0^2} I_0 \left[1 + \frac{L - \mathcal{L}_0 - \mathbf{g}}{\mathcal{L}_0} \left(\frac{r_0}{\mathbf{r}}\right)^2\right] \frac{\mathbf{i}(t)}{I_0} \\
&\quad + V_0 \sinh^{-1}\left[\frac{\mathbf{i}(t)}{I_0} \exp\left(\frac{\mathbf{g}}{g_m}\right)\right]
\end{aligned} \tag{2.4}$$

where, L is the layer thickness, ρ is the CF regions' resistivity, r_0 and $\mathcal{L}_0$ are the constant radius and length of the base-CF region respectively, I_0, V_0, g_m are fitting parameters while $\mathbf{r}$ is the modulated radius of the sub-CF region and $\mathbf{g}$ is the gap length. The values of both physical and fitting parameters are taken from [40] and are here reported for convenience in Table 2.1.

Physical parameters		
E_a	0.85eV	ion migration energy barrier
a	0.1nm	hopping distance
f	10^{12}Hz	escape-attempt frequency
L	5.0nm	layer thickness
$\mathcal{L}_0$	2.5nm	base filament thickness
r_0	2.5nm	base filament radius
ρ	$2.2 \cdot 10^{-6}\ \Omega/\mathrm{m}$	resistivity of the CF
Fitting parameters		
I_0	15mA	Parameters for current
V_0	0.2V	
g_m	0.2nm	
r_m	0.8nm	
α	$3 \cdot 10^4$	dn/dy
β	$8 \cdot 10^3$	dn/dx
C_{pl}	9.1μJ/K	Heat capacitance
χ_1	$0.28\mu\mathrm{JK}^{-1}\mathrm{s}^{-1}$	Thermal conductance
H	$5.5 \cdot 10^7$	Temporal heat factor
δ	540	Heat dissipation factor

Table 2.1 Parameters used for modeling the memristor dynamics.

2.2.2 Complete ReRAM Dynamics

The sub-CF radius **r** and the gap length **g**, as shown in Fig. 2.1(a), are two geometric state variables and their time evolution is described by the following system of ODEs:

$$\frac{d\mathbf{g}}{dt} = \begin{cases} -\frac{1}{2}\exp\left(-\frac{E_a}{k_b\mathbf{T}}\right)\imath(t) & (\mathbf{v}\geq 0) \\ \\ -\frac{1}{2}\exp\left(-\frac{E_a}{k_b\mathbf{T}}\right)\left(\frac{r_0}{\mathbf{r}}\right)^2\imath(t) & (\mathbf{v}<0) \end{cases} \tag{2.5}$$

$$\imath(t) = \frac{\alpha a^2 f}{\mathcal{L}_0-\mathbf{g}} - 2af\sinh\left(\frac{qa\mathbf{v}}{\mathbf{g}k_b\mathbf{T}}\right) \tag{2.6}$$

$$\frac{d\mathbf{r}}{dt} = \begin{cases} -\frac{1}{2}\exp\left(-\frac{E_a}{k_b\mathbf{T}}\right)\left[\frac{\beta a^2 f}{\mathbf{r}-r_m}\right] & (\mathbf{v}\geq 0) \\ \\ +\frac{1}{2}\exp\left(-\frac{E_a}{k_b\mathbf{T}}\right)\left(\frac{r_0}{\mathbf{r}}\right)^2\left[\frac{\beta a^2 f}{\mathbf{r}-r_m}\right] & (\mathbf{v}<0) \end{cases} \tag{2.7}$$

where k_b is the Boltzmann constant, q is the electron charge, E_a is the ion migration energy barrier, f is the escape-attempt frequency, a is the hopping distance, and r_m, α, β are model parameters. The values of all those parameters were estimated in [40] and are reported in Table 2.1 for convenience.

The internal temperature dynamics is modeled via the two following coupled ODEs which describe the time evolution of the internal temperature

T and the bulk temperature of the outer region $\mathbf{T_b}$ respectively:

$$C_{p1}\,\frac{d\mathbf{T}}{dt} = \mathbf{v}(t)\,\mathbf{i}(t) - k_{th1}\,(\mathbf{T} - \mathbf{T_b}) \tag{2.8}$$

$$C_{p2}\,\frac{d\mathbf{T}_b}{dt} = \mathbf{v}(t)\,\mathbf{i}(t) - k_{th2}\,(\mathbf{T}_b - 300) \tag{2.9}$$

where $Cp1$ and C_{p2} are the thermal capacitances while k_{th1} and k_{th2} are the thermal conductances. The values of those parameters were estimated in [40] and are reported in Table 2.1 for convenience.

Given the tabled values of the thermal fitting parameters, the values for the temperature and the bulk temperature time constants τ_T and τ_b are estimated as:

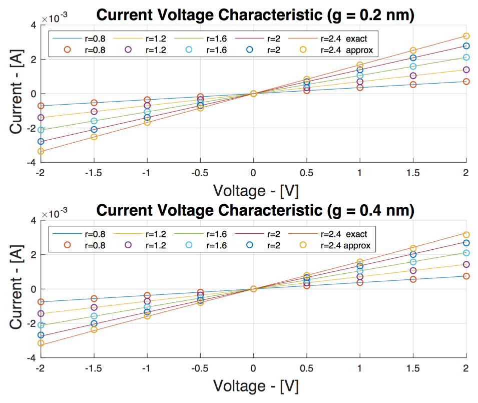
$$\tau_T = \frac{1}{\lambda_T} = \frac{C_{p1}}{k_{th1}} \approx 0.325\,10^{-9}\,s \tag{2.10}$$

$$\tau_b = \frac{1}{\lambda_b} = \frac{C_{p2}}{k_{th2}} = \frac{1}{5.4\,10^{6}} \approx 0.185\,10^{-6}\,s \tag{2.11}$$

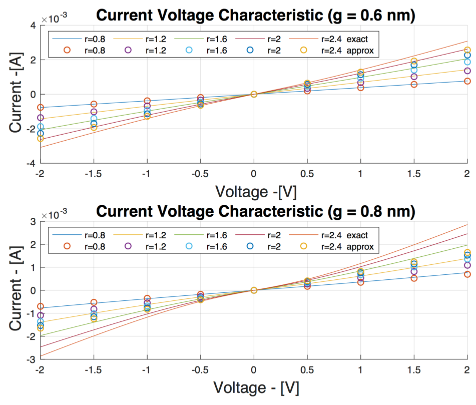
It should be noted that the internal temperature time constant τ_T is orders of magnitude smaller than the bulk temperature time constant τ_b. This fact is fundamental in order to unveil the role played by the temperature in second order memristors.

2.3 Simplified second order model derivation

The main difference between the calcium based model presented in [41] and the above described complete second order model derived in [40] is the fact that the discussed memristor model lacks a direct expression of the time-evolution of the memductance which represents the synaptic efficacy variable. A second major difference concerns the correspondent roles of the internal temperature **T** and the calcium concentration in equations (2.5)-(2.7) and in Eq. (1) of [41] respectively.



(a) $\mathbf{g} \in \{0.2, 0.4\}\,nm$



(b) $\mathbf{g} \in \{0.6, 0.8\}\,nm$

Fig. 2.2 Current-voltage characteristic, for different values of the radius $\mathbf{r}$ (expressed in nm), and of the length gap ($\mathbf{g} \in \{0.2, 0.4, 0.6, 0.8\}\,nm$). Solid lines represent the exact characteristic, obtained by numerically inverting the $\mathbf{v}$-$\mathbf{i}$ relation (2.4); the current values derived from the approximate expression of the mem-conductance given in (2.12) are represented by circles. Taken from [1] ©2022 IEEE.

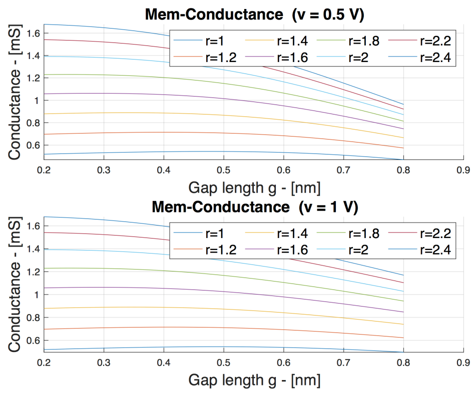
In this Section, in order to compare the memristor model developed in [40] with the calcium based biophysical model in [41] and unveil the main neuromorphic functions of $Pd/Ta_2O_{5-x}/TaO_y/Pd$ memristors in relation with biological synapses, a simplified model is derived which involves only two equations and two variables, the memductance $\hat{\mathbf{G}}$ and the internal temperature $\mathbf{T}$, which directly relate with synaptic efficacy and calcium concentration used in biophysical models to reproduce heterogeneous Spike-Timing-Dependent-Plasticity (STDP) curves.

2.3.1 Simplified Ohm's Law

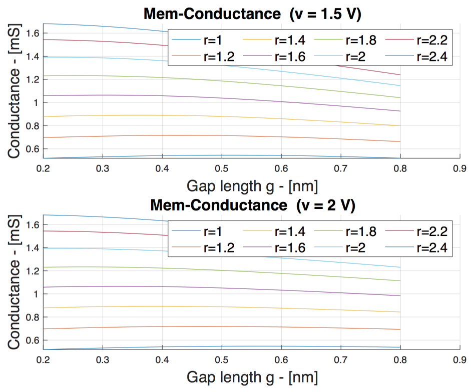
Even if Eq. (2.4) cannot be inverted in a closed form, the direct relationship between the current $\mathbf{i}(t)$ and the voltage $\mathbf{v}(t)$ can still be numerically computed. The memductance results to depend on the radius $\mathbf{r}$, on the gap length $\mathbf{g}$ and on the voltage $\mathbf{v}(t)$. The first way to derive an explicit form of the memductance is to use the first term of the Maclaurin expansion of $\sinh^{-1}$ function with respect to its argument $\frac{\mathbf{i}(t)}{I_0} \exp\left(\frac{\mathbf{g}}{g_m}\right)$, which produces the following formulation:

$$\tilde{\mathbf{G}}\left(\mathbf{g},\mathbf{r}\right) = \frac{1}{\frac{\rho\mathcal{L}_0}{\pi r_0^2}\left[1+\frac{L-\mathcal{L}_0-\mathbf{g}}{\mathcal{L}_0}\left(\frac{r_0}{\mathbf{r}}\right)^2\right]+\frac{V_0}{I_0}\exp\left(\frac{\mathbf{g}}{g_m}\right)} \tag{2.12}$$

As graphically shown in Fig. 2.2, the above expression seems to well match the exact relationship between the current $\mathbf{i}(t)$ and the voltage $\mathbf{v}(t)$ within a relatively large range of values taken by the state variables $\mathbf{g}$ and $\mathbf{r}$. Although some numerical inaccuracies occur for high values of $\mathbf{g}$, as expected, for lower values of the gap length $\mathbf{g}$ and a given fixed radius $\mathbf{r}$ a reasonable range of input voltages exists where the characteristic is close to linear and the matching with the approximate expression (2.12) is almost perfect. From the analysis of the stationary points of equation (2.5), it can be shown that the hypothesis of a relatively small gap length $\mathbf{g}$ is plausible for all the reasonable ranges of operational temperature and voltage.



(a) $\mathbf{v} \in \{0.5, 1.0\}\,V$



(b) $\mathbf{v} \in \{1.5, 2.0\}\,V$

Fig. 2.3 Memductance, as a function of the gap length $\mathbf{g}$, for different values of the radius $\mathbf{r}$ (expressed in nm), and different voltages $\mathbf{v} \in \{0.5, 1.0, 1.5, 2.0\}\,V$. Taken from [1] ©2022 IEEE.

The second way to simplify the $\mathbf{i}$-$\mathbf{v}$ relationship is to empirically observe, as in Fig. 2.3, that the memductance only weakly depends on the gap length $\mathbf{g}$ while most of its variation is accountable to the radius $\mathbf{r}$. In particular, for voltage higher than 1 V the memductance is almost independent from the gap length $\mathbf{g}$. For lower voltages it can be demonstrated, from the analysis of the stationary points of equation (2.5), that only a shallow range of the gap length can be explored, e.g. $\mathbf{g} < 0.4$ nm, and thus the memductance can still be considered varying with the radius $\mathbf{r}$ only.

It should be noted that, in spite of the strong approximation, simulations clearly show that the obtained formulation of the conductance is fairly accurate and can unveil all the major plasticity properties of second order memristors. This holds true for any admissible activation energy E_a and the whole voltage range of interest as will be motivated in the Appendix A.

By noting that $\mathcal{L}_0 = 0.5\,L$ and by considering the conduction through the base-CF and sub-CF regions only, from Eq. (2.4) it is obtained:

$$\hat{\mathbf{G}}(\mathbf{r}) = \frac{1}{R_s\left[1+\left(\frac{r_0}{\mathbf{r}}\right)^2\right]}, \quad \left(R_s = \frac{\rho\mathcal{L}_0}{\pi r_0^2}\right) \tag{2.13}$$

2.3.2 Memductance dynamics derivation

Starting from the simplified formulation of memductance $\hat{\mathbf{G}}$ in Eq. (2.13), the radius $\mathbf{r}$ can be expressed as:

$$\mathbf{r} = r_0\sqrt{\frac{R_s\hat{\mathbf{G}}}{1-R_s\hat{\mathbf{G}}}} \tag{2.14}$$

The following formulation of the time derivative of the memductance $\hat{\mathbf{G}}$ is easily derived from (2.13) and (2.7):

$$\begin{aligned}\frac{1}{\hat{\mathbf{G}}}\frac{d\hat{\mathbf{G}}}{dt} = \frac{1}{\hat{\mathbf{G}}}\frac{d\hat{\mathbf{G}}}{d\mathbf{r}}\frac{d\mathbf{r}}{dt} &= 2\,\frac{1}{\mathbf{r}}\,(1-R_s\hat{\mathbf{G}})\,\frac{d\mathbf{r}}{dt} \\ &= \exp\left(-\frac{E_a}{k_b\mathbf{T}}\right)\eta(\hat{\mathbf{G}})\end{aligned} \tag{2.15}$$

$$\eta(\hat{\mathbf{G}}) = \frac{\sqrt{\frac{(1-R_s\hat{\mathbf{G}})^3}{R_s\hat{\mathbf{G}}}}}{\sqrt{\frac{R_s\hat{\mathbf{G}}}{1-R_s\hat{\mathbf{G}}} - \frac{r_m}{r_0}}} \left(\frac{a}{r_0}\right)^2 \beta f \begin{cases} -1 & (\mathbf{v} \geq 0) \\ \\ \frac{1-R_s\hat{\mathbf{G}}}{R_s\hat{\mathbf{G}}} & (\mathbf{v} < 0) \end{cases} \tag{2.16}$$

The obtained expression is directly comparable with Eq. (1) of [41] as it exclusively depends on $\hat{\mathbf{G}}$ and $\mathbf{T}$, which emulate the synaptic efficacy and the calcium concentration respectively.

2.3.3 Simplified temperature dynamics

In regard of the temperature dynamic equations (2.8)-(2.9), it is considered the fact that the voltage spikes applied to the two terminals for short time periods can be modeled as constant sources. In fact, given that the conductance dynamics is substantially slower than the temperature dynamics which has the time constants (2.10)-(2.11), the conductance can be considered almost constant during the time intervals of both the programming ($\approx 20\,ns$) and heating ($\approx 1\,\mu s$) voltage pulses, according to [40]). In a time interval $[t_0, t]$ for the temperatures $\mathbf{T}$ and $\mathbf{T}_b$ it is obtained that:

$$\begin{aligned} \mathbf{T_b}(t) &= \exp[-\lambda_b(t-t_0)]\,(T_b(t_0) - T_b^\infty) + T_b^\infty \\ T_b^\infty &= 300 + \frac{1}{k_{th2}}\overline{\mathbf{G}}\mathbf{V}^2 \end{aligned} \tag{2.17}$$

$$
\begin{aligned}
\mathbf{T}(t) \;&=\; \exp[-\lambda_T(t-t_0)]\left(T(t_0)-\frac{1}{k_{th1}}\overline{\mathbf{G}}\mathbf{V}^2-T_b^{\infty}\right)\\
&+\; \frac{1}{k_{th1}}\overline{\mathbf{G}}\mathbf{V}^2+T_b^{\infty}\\
&+\; \left\{\frac{\lambda_T}{\lambda_T-\lambda_b}\left\{\exp[-\lambda_b(t-t_0)]-\exp[-\lambda_T(t-t_0)]\right\}\right\}\\
&\cdot\left\{T_b(t_0)-T_b^{\infty}\right\} \qquad (2.18)
\end{aligned}
$$

where $\overline{\mathbf{G}}$ is the value of the conductance at the beginning of the pulse and by $\mathbf{V}$ the amplitude of the constant voltage source modeling the pulse.

By considering that $\tau_b >> \tau_T$, as in (2.10)-(2.11), and that $t - t_0 \approx 15ns$ the contribution of the function $\exp[-\lambda_T(t-t_0)]$ is negligible, and thus the following simplified expression of the internal temperature $\mathbf{T}(t)$ is obtained:

$$
\begin{aligned}
\mathbf{T}(t) \;&\approx\; exp[-\lambda_b(t-t_0)]\,(T_b(t_0)-T_b^{\infty})+T_b^{\infty}+\frac{1}{k_{th1}}\overline{\mathbf{G}}\mathbf{V}^2\\
&\approx\; \mathbf{T_b}(t)+\frac{1}{k_{th1}}\overline{\mathbf{G}}\mathbf{V}^2 \qquad (2.19)
\end{aligned}
$$

Although in standard STDP models a single pulse makes the spike, accordingly to [40], memristors are typically excited by pre(post)synaptic spikes consisting of a programming and a heating pulse in sequence with time duration t_s and t_H respectively. The involved modeling is indeed more complex as the programming pulse of the second post(pre)synaptic spike may occur either before or after the end of the heating pulse of the first pre(post)synaptic pulse as illustrated in Fig. 2.4 for pre/post spike pair.

By denoting with γt_H the time shift between the beginning of the programming pulse of the second spike and the beginning of the heating pulse of the first spike, from the formulation of $\mathbf{T}(t)$ as in (2.19), an accurate explicit expression of the internal temperature $\mathbf{T}(\gamma)$ associated to the programming pulse is obtained:

$$\mathbf{T}(\gamma) \approx 300 + \hat{\mathbf{G}} \left\{ \frac{V_P^2}{k_{th1}} + \frac{V_P^2}{k_{th2}} \left[1 - \exp\left(-\frac{t_s}{\tau_b} \right) \right] + \Gamma \frac{V_H^2}{k_{th2}} \right\}$$

$$\Gamma = \begin{cases} \exp\left(-\frac{t_s}{\tau_b}\right)\left[1 - \exp\left(-\frac{\gamma t_H}{\tau_b}\right)\right] & \gamma < 1 \\ \\ \exp\left(-\frac{t_s + (\gamma-1)t_H}{\tau_b}\right)\left[1 - \exp\left(-\frac{t_H}{\tau_b}\right)\right] & \gamma \geq 1 \end{cases} \tag{2.20}$$

where the parameter γ is > 1 when the programming pulse starts after the end of the heating pulse and < 1 otherwise, V_H and V_P are the heating and programming voltage pulse amplitudes respectively while $\hat{\mathbf{G}}$ denotes the memductance which remains constant on first approximation.

2.4 Memristor response to synaptic protocols

The previously introduced analytic approach to $Pd/Ta_2O_{5-x}/TaO_y/Pd$ memristors modeling opens the door not only to the investigation of the synaptic properties of isolated second order memristors but also to the analysis of the neuromorphic properties of large memristor networks.

Firstly, under the hypothesis that the programming pulse of the second spike follows after the end of the heating pulse of the first spike namely $\gamma \geq 1$, by replacing the formulation of $\mathbf{T}$ (2.20) into (2.15) the STDP function corresponding to the memductance variation can be readily obtained in the form of a function of the time interval $\Delta_t = (\gamma - 1)t_H$ which elapses between two subsequent spikes.

As shown in Fig. 2.5, the memductance variation caused by a single pre/post spike pairs ($\Delta_t > 0$) is positive and thus we have potentiation of the synaptic connection while for a single post/pre spike pairs ($\Delta_t < 0$) the variation is negative hence we have depression of the synaptic connection. It should also be noted that the normalized variation of the memductance for

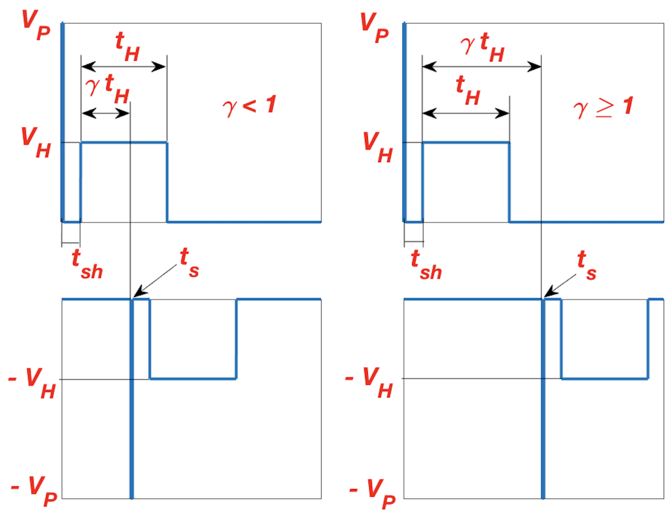


Fig. 2.4 Input voltage corresponding to a pre/post spike pair. Each spike is represented by the sequence of a programming pulse of duration t_s and a heating pulse of duration t_H, whereas t_{sh} denotes the time shift between the programming and the heating pulse. Left part: the programming pulse of the postsynaptic spike occurs before the end of the heating pulse of the first presynaptic pulse. Right part: the programming pulse of the postsynaptic spike occurs after the end of the heating pulse of the presynaptic pulse. The time interval between two spikes is denoted by $\Delta_t = (\gamma - 1)t_H$, with γ greater than 1 in the first case and less than 1 in the second one. Taken from [1] ©2022 IEEE.

second order memristors depends on the initial value of the conductance $\hat{\mathbf{G}}$ via the internal temperature $\mathbf{T}$ as made clear in Eq. (2.20).

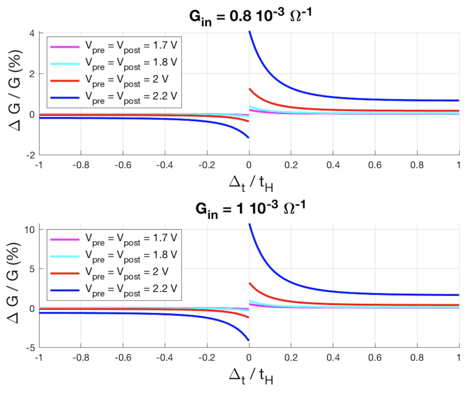
Even if spike pairs are a useful neuromorphic paradigm, they cannot clarify more complex synaptic protocols such as the effect of repetition frequency on the synaptic change [42]. All these complex protocols, including experiments involving triplets and quadruplets, can be explained and reproduced by means of Eqs. (2.15) and (2.20). This allows to apply on memristive synapses the approach presented in [41] and already used in experiments on hippocampal cultures.

The memductance variation as a function of the spike delay Δ_t is reported in Figs. 2.6-2.7-2.8, for 30 cycles at different frequencies of pre-post pairs, post-pre-post triplets and post-pre-pre-post quadruplets respectively.

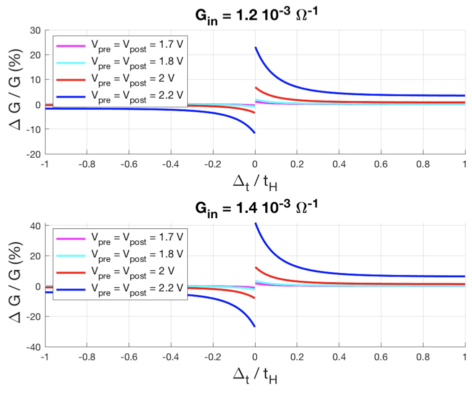
It should be noted that:

- Long Term Potentiation (LTP) is observed in all cases and in particular for spike triplets the conductance saturates to the maximum value of (2.13) $\hat{\mathbf{G}} = \frac{1}{2R_s}$;
- For spike pairs the potentiation is reduced when repetition frequency increases (i.e. when the interleave time interval t_f between two pairs decreases), which is in disagreement with most experimental protocols and represents a major drawback of classical STDP models [43];
- In order to observe the correct frequency response, spike triplets and/or quadruplets should be exploited, as reported in [43] and [41];

Additional results are reported in Figs. 2.9-2.10 which show the memductance variation as a function of the spike interval Δ_t for 30 cycles of pre-post-pre triplets and pre-post-post-pre quadruplets at different frequencies. As expected from the data in [41], those configurations provoke insignificant potentiation.



(a) $G_{in} \in \{0.8, 1.0\}\, 10^{-3}\Omega^{-1}$



(b) $G_{in} \in \{1.2, 1.4\}\, 10^{-3}\Omega^{-1}$

Fig. 2.5 STDP function corresponding to the mem-conductance variation, generated by a pre/post (post/pre) spike pair, as a function of the time interval $\frac{\Delta_t}{t_H}$, which separates two subsequent spikes. It is assumed $V_H = 0.8$ V, $\frac{t_H}{\tau_b} = 5.4$, and $\frac{t_s}{\tau_b} = 0.108$; G_{in} denote the initial value of the mem-conductance. Taken from [1] ©2022 IEEE.

2.5 Model robustness study

In the first Subsection the accuracy within the voltage range of interest of the proposed memductance expressions is proved for any admissible activation

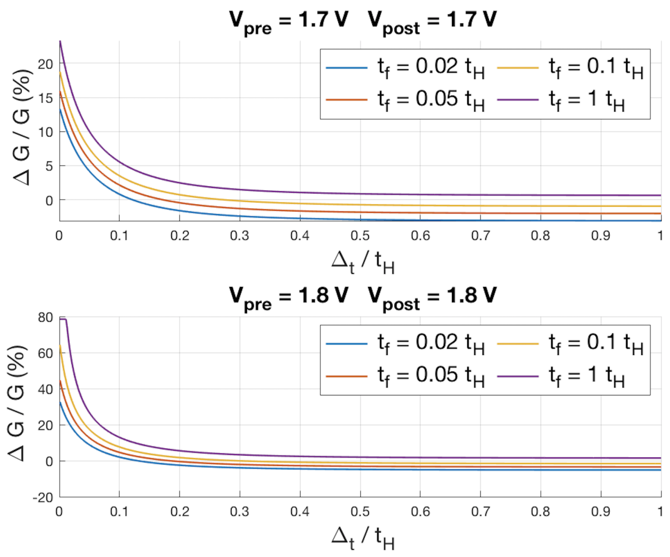


Fig. 2.6 Mem-conductance change versus $\frac{\Delta_t}{t_H}$, for 30 cycles of pre-post pairs at different frequencies, and different programming voltages. It is assumed $V_H = 0.8$ V, $\frac{t_H}{\tau_b} = 5.4$, and $\frac{t_s}{\tau_b} = 0.108$; t_f denotes the time interval between two pairs; the initial mem-conductance is $\hat{\mathbf{G}} = 1$ mS. Taken from [1] ©2022 IEEE.

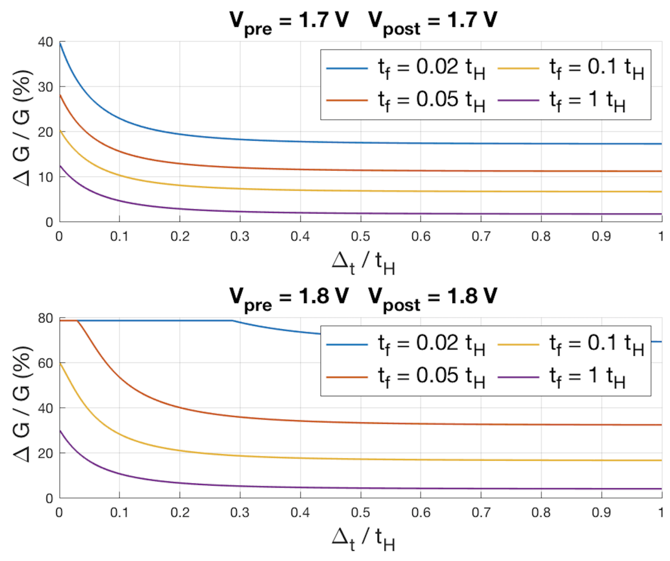


Fig. 2.7 Mem-conductance change versus $\frac{\Delta_t}{t_H}$, for 30 cycles of post-pre-post triplets at different frequencies, and different programming voltages. It is assumed $V_H = 0.8$ V, $\frac{t_H}{\tau_b} = 5.4$, and $\frac{t_s}{\tau_b} = 0.108$; t_f denotes the time interval between two triplets; the initial mem-conductance is $\hat{\mathbf{G}} = 1$ mS. Taken from [1] ©2022 IEEE.

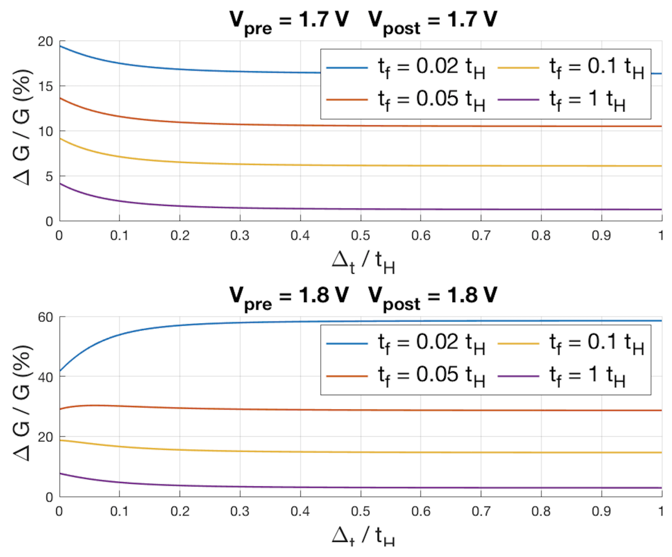


Fig. 2.8 Mem-conductance change versus $\frac{\Delta_t}{t_H}$, for 30 cycles of post-pre-pre-post quadruplets at different frequencies, and different programming voltages. It is assumed $V_H = 0.8$ V, $\frac{t_H}{\tau_b} = 5.4$, and $\frac{t_s}{\tau_b} = 0.108$; t_f denotes the time interval between two quadruplets; the initial mem-conductance is $\hat{\mathbf{G}} = 1$ mS. Taken from [1] ©2022 IEEE.

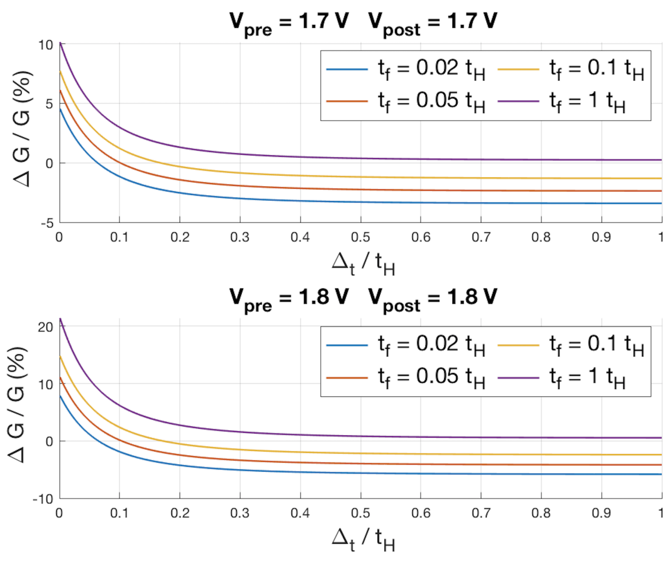


Fig. 2.9 Mem-conductance change versus $\frac{\Delta_t}{t_H}$, for 30 cycles of pre-post-pre triplets at different frequencies, and different programming voltages. It is assumed $V_H = 0.8$ V, $\frac{t_H}{\tau_b} = 5.4$, and $\frac{t_s}{\tau_b} = 0.108$; t_f denotes the time interval between two triplets; the initial mem-conductance is $\hat{\mathbf{G}} = 1$ mS. Taken from [1] ©2022 IEEE.

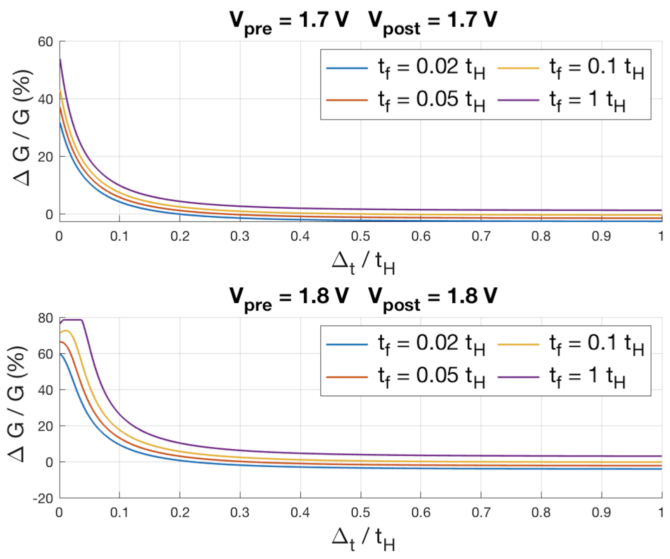


Fig. 2.10 Mem-conductance change versus $\frac{\Delta_t}{t_H}$, for 30 cycles of pre-post-post-pre quadruplets at different frequencies, and different programming voltages. It is assumed $V_H = 0.8$ V, $\frac{t_H}{\tau_b} = 5.4$, and $\frac{t_s}{\tau_b} = 0.108$; t_f denotes the time interval between two quadruplets; the initial mem-conductance is $\hat{\mathbf{G}} = 1$ mS. Taken from [1] ©2022 IEEE.

energy E_a. In the second Subsection influence of parameters variation on the proposed second order memristor model is examined.

2.5.1 Model accuracy when changing voltages and activation energies

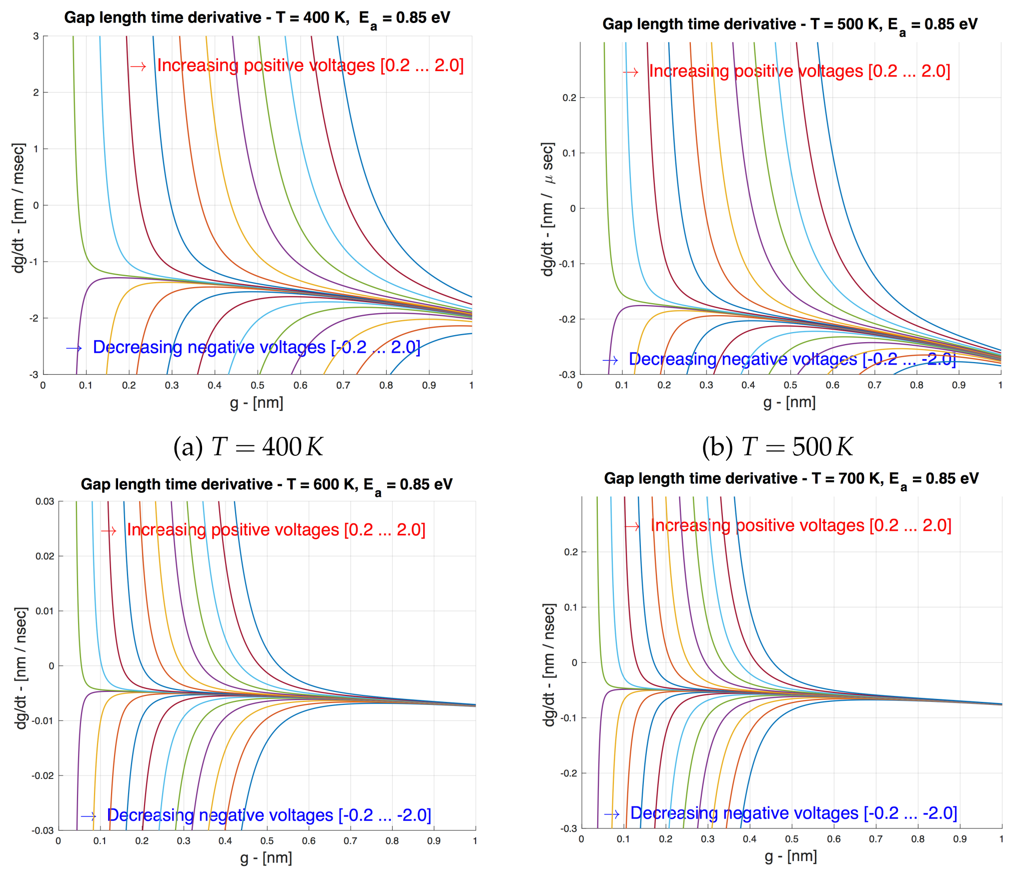
In this Subsection, the second order memristor model derived in Section 2.3 and in particular the memductance expressions (2.13) - (2.12) are shown to be accurate within the voltage range of interest and for any admissible activation energy E_a.

In fact, even if the exact current-voltage characteristic (2.4) does only depend on the state variables $\mathbf{g}$ and $\mathbf{r}$, it must be considered that the ODEs (2.5)-(2.6) and (2.7) governing the evolution of those two geometric parameters do explicitly depend on the applied voltage and the activation energy E_a.

It is made clear in Fig. 2.3 that the memductance exhibits a weak dependence on the gap length g for $0 \leq \mathbf{g} \leq 0.8\,nm$ and $0 \leq \mathbf{v} \leq 2\,V$. Since both the exact (2.13) and approximate (2.12) expressions entirely include the dependence on $\mathbf{r}$, in order to prove their accuracy, it is sufficient to show that the variation range of $\mathbf{g}$ lies within the interval $[0, 0.8nm]$.

From equations (2.5) - (2.6) it follows that, for constant voltages, the equilibria of $\mathbf{g}$ are given by the solutions of eq. (2.6), which is independent from the activation energy E_a:

$$\frac{\alpha\, a^2 f}{\mathcal{L}_0 - \mathbf{g}} - 2\, a\, f \sinh\left(\frac{q\, a\, \mathbf{v}}{\mathbf{g}\, k_b\, \mathbf{T}}\right) = 0 \tag{2.21}$$



(a) $T = 400\,K$

(b) $T = 500\,K$

(c) $T = 600\,K$

(d) $T = 700\,K$

Fig. 2.11 Gap length time derivative $d\mathbf{g}/dt$ as a function of $\mathbf{g}$, for $T \in \{400, 500, 600, 700\}\,K$, $\mathbf{r}/r_0 = 1$, $E_a = 0.85\,eV$ and different input voltages ranging from $-2\,V$ to $2\,V$. Taken from [1] ©2022 IEEE.

The plots of $d\mathbf{g}/dt$ as a function of $\mathbf{g}$, also known as Dynamic Route Maps (DRMs), are shown in Fig. 2.11, for $E_a = 0.85\,eV$, parameterized on the different values of the temperature reachable during the heating phase and constant voltages ranging from $-2\,V$ to $2\,V$.

When positive voltages are applied to the memristor, each DRM features a single zero $\hat{\mathbf{g}}(\mathbf{v},\mathbf{T})$ that is function of the voltage $\mathbf{v}$ and the temperature T. Because all those DRMs cross $d\mathbf{g}/dt = 0$ with a negative slope, the equilibria $\hat{\mathbf{g}}(\mathbf{v},\mathbf{T})$ are stable and, for given voltages and temperatures, they represent the maximum value $\mathbf{g}(t)$ can reach, i.e. $\mathbf{g}(t) \in [0, \hat{\mathbf{g}}(\mathbf{v},\mathbf{T})]$.

When negative voltages are applied to the memristor, the time derivative $d\mathbf{g}/dt$ is negative for all values of $\mathbf{g}(t)$ and thus the gap length shall remain below the values reachable via positive voltages.

From the plots in Fig. 2.11 it can be noticed that:

- When the input voltage is such that $0 \leq \mathbf{v} \leq 2\,V$ equilibria $\hat{\mathbf{g}}(\mathbf{v},\mathbf{T})$ decrease as the temperature $\mathbf{T}$ increases.
- When the temperature is such that $400\,K \leq T \leq 700\,K$ the equilibria $\hat{g}(\mathbf{v},\mathbf{T})$ increase as the voltage $\mathbf{v}$ increases.

Within the temperature range $\mathbf{T} \in [400,700]\,K$ and for voltages not higher than $2\,V$, the maximum value of $\hat{\mathbf{g}}(\mathbf{v},\mathbf{T})$, as shown by the equilibria in Fig. 2.11, is less than $0.8\,nm$. In accordance with Fig. 2.3 it is clear that the memductance only weakly depends on the gap length and thus its approximation is accurate for $E_a = 0.85\,eV$. Even for voltages higher than $2\,V$, the equilibria are only slightly pushed beyond $0.8nm$, and concurrently the range of $\mathbf{g}$ where the memductance depends on radius $\mathbf{r}$ only also enlarges.

Even if different activation energy values E_a were considered, the equilibria of the gap length $\hat{\mathbf{g}}(\mathbf{v},\mathbf{T})$ would not be altered, and the graph of the gap length time derivative $d\mathbf{g}/dt$ versus $\mathbf{g}$ of Fig. 2.11 would only be scaled along the vertical axis. Since the equilibria $\hat{g}(\mathbf{v},\mathbf{T})$ are not modified, the memductance approximate formulation is valid for the considered input voltages and for any value of activation energy E_a.

2.5.2 Model sensitivity to parameters variation

In this Subsection, the influence of parameters variation on the proposed second order memristor model is examined. The memductance formulation (2.13) fully encapsulates the dependence on the radius r and the associated parameters r_0 and r_m. In order to analyze the consequence of intrinsic parameters variation, the dependence of the STDP function on r_0 and r_m parameters should be investigated.

The STDP function is obtained from the derivative of the memductance with respect to time (2.15). This takes the form of a product of a first term depending on the temperature and a second term $\eta(\hat{\mathbf{G}})$ which depends on r_0 and r_m.

The memductance time derivative sensitivity to the parameters variation is then equivalent to $\eta(\hat{\mathbf{G}})$ sensitivity only, which can be easily derived from (2.16):

$$\frac{r_0}{\eta(\hat{\mathbf{G}},r_0)}\frac{\partial\eta(\hat{\mathbf{G}},r_0)}{\partial r_0} = \frac{\frac{r_m}{r_0} - 2\sqrt{\frac{R_s\hat{\mathbf{G}}}{1-R_s\hat{\mathbf{G}}}}}{\sqrt{\frac{R_s\hat{\mathbf{G}}}{1-R_s\hat{\mathbf{G}}}} - \frac{r_m}{r_0}} \tag{2.22}$$

$$\frac{r_m}{\eta(\hat{\mathbf{G}},r_m)}\frac{\partial\eta(\hat{\mathbf{G}},r_m)}{\partial r_m} = \frac{\frac{r_m}{r_0}}{\sqrt{\frac{R_s\hat{\mathbf{G}}}{1-R_s\hat{\mathbf{G}}}} - \frac{r_m}{r_0}} \tag{2.23}$$

Eqs. (2.22) and (2.23) show that the sensitivity depends on the memductance $\hat{\mathbf{G}}$ and the parameters r_0 and r_m. Therefore, in case of slight perturbation of the parameters, the STDP variation is only modified by the memductance value and not by the time shift Δ_t between pre/post (or post/pre) pairs.

As exemplified in Figs. 2.12-2.13, STDP curves are only slightly perturbed by small variations ($\approx 5\%$) of the parameters r_0 and r_m. Large-scale simulations have corroborated the robustness of the proposed simplified second order memristor model to intrinsic device parameters variation when applied for describing the synaptic functionality of the device.

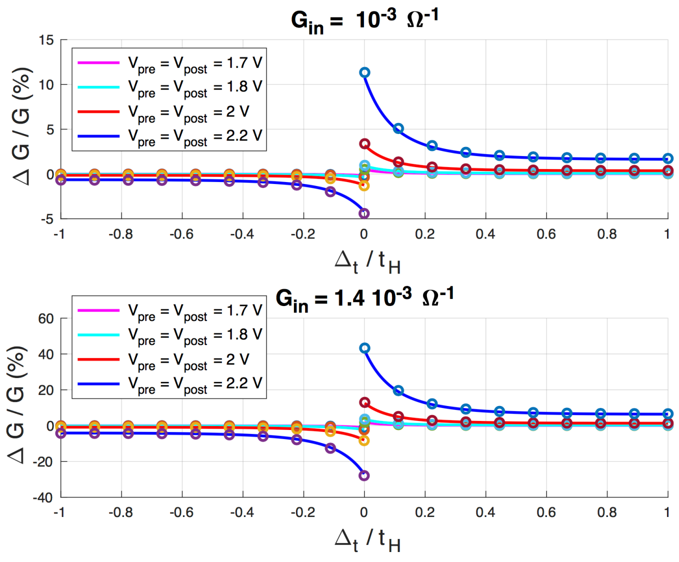


Fig. 2.12 Impact on STDP functions of a 5 % variation of the parameter r_0. A continuous line represents nominal values, circles represent perturbed values. Taken from [1] ©2022 IEEE.

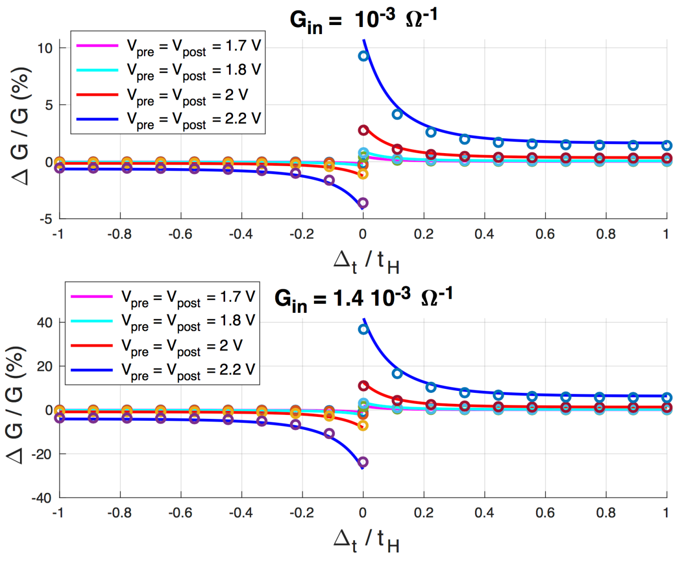


Fig. 2.13 Impact on STDP functions of a 5 % variation of the parameter r_m. A continuous line represents nominal values, circles represent perturbed values. Taken from [1] ©2022 IEEE.

2.6 Conclusion

In this Chapter a simplified analytical model of second order memristors was developed. This consists of only two state variables, the internal temperature and the memductance, which can be directly related to the quantities used in advanced biophysical models, precisely the calcium concentration and the synaptic efficacy.

The response to some relevant stimulation protocols were thoroughly studied, in particular cycles of spike pairs, triplets, and quadruplets at different frequencies. Thanks to this approach, it has been shown that:

- The majority of relevant synaptic properties of second-order memristors can be readily investigated and predicted.
- Synaptic behaviors which cannot be captured by classical spike pair based STDP models are easily reproduced.

Supposedly by using the proposed second-order memristor model together with advanced theoretical and numerical nonlinear dynamic techniques, the behavior of memristor networks stimulated by arbitrary presynaptic inputs can be predicted and the underlying learning mechanisms effectively investigated.

Chapter 3

Memristor-based Neural Networks

3.1 Introduction

Multiply and Accumulate (MAC) is a fundamental operation for all sorts of algorithms that require the computation of the weighted sum of the components of a vector.

When digitally performed on ASICs, the MAC operation applied to a n-dimensional vector requires either a digital adder, a digital multiplier and an accumulator register used for n clock cycles or, when the time constraint is more stringent, a multiple-input digital adder, lots of digital multipliers and an accumulator. Even if many optimizations are possible, the trade-off between circuit area / power consumption and computation latency is unavoidable. This becomes an even-harder design challenge when the MAC operations are two or more to be performed in parallel as required in case of the Matrix Vector Multiplication (MVM) between a $n \times m$ matrix and a n-dimensional vector. The computational complexity of MVM performed on the simpler and slower sequential single-adder-single-multiplier digital circuit is of the order $O(nm)$. Different MVM accelerators have been commercialized during the last three decades, the most famous being the Graphics Processing Unit (GPU) and latest being the Tensor Processing Unit (TPU). All those digital implementations speed up MVM computation by parallelizing products and additions at the expense of wafer area and power consumption.

If the coefficients of the weighted sum are constant, an analog implementation of the single MAC operation is quite straightforward as it only requires an Operational Amplifier Voltage Adder (OAVA). This was the most widespread solution in the early days of electronics and is still in use nowadays in certain application-specific analog circuits. The problem with the OAVA rises as in the majority of algorithms the coefficients of the MAC weighted sum change from time to time. In almost the totality of applications, physically changing the conductances which encode the weights in an OAVA is unfeasible. Even if analog multipliers can be used as a replacement for conductances, this extreme solution makes the OAVA complex to design, power/area hungry, affected by nonlinearity effects and thermal noise at the point that for most cases digital implementations have been preferred.

As the interest in memristive technologies has quickly risen in the last decade, the number of research works and market-ready products where memristors as programmable weights are used in OAVAs to implement MAC operation has risen too. These novel analog accelerators exploit the continuum of resistance states displayed by memristors in order to implement the MAC operation with a computation complexity on the order of $O(1)$. Most notably, m memristor-based OAVAs which take in input the same n-dimensional vector implement also the MVM with time complexity $O(1)$. This is very convenient in terms of speed with respect to the sequential single-adder-single-multiplier digital implementation which requires $O(nm)$ operations, and it is also, at least on paper, a big step forward in terms of power dissipation and occupied chip area with respect to both digital accelerators (e.g. GPUs or TPUs) and analog multiplier based solutions.

The circuit that implements the multiplication of a n-dimensional vector times a $n \times m$ matrix by means of a two dimensional memristor array is known as memristor crossbar. It consists in m OAVAs as depicted in Fig. 3.1. The physical implementation of such structure is made of n horizontal metallic input lines and m vertical metallic output lines with a memristor at each intersection between a vertical and a horizontal line. It is crucial to emphasize that differently from ASICs that continuously move data from memory to local registers, the crossbar structure performs each multiplication locally as each coefficient of the matrix is encoded as a conductance at the intersection between a horizontal and a vertical line. Therefore memristor crossbars can perform MVM in a massively parallel fashion with amazingly high power efficiency compared to traditional digital accelerators.

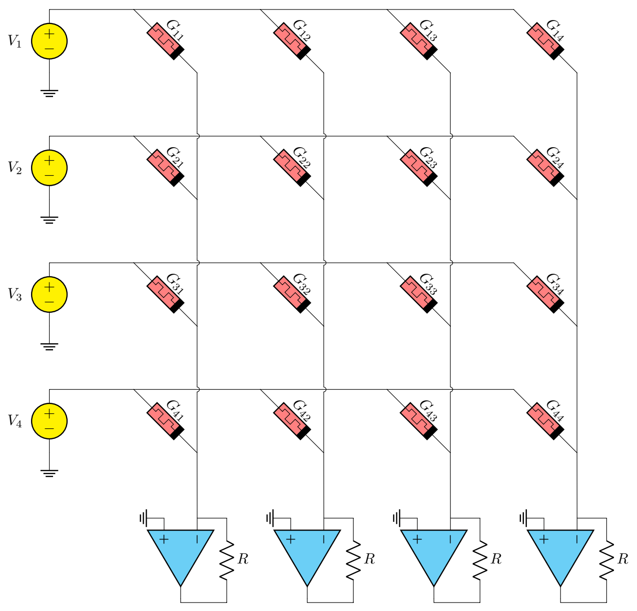


Fig. 3.1 Example of a 4×4 memristor crossbar array.

The fact that Moore's Law is approaching its end along with the well known von Neumann architecture inefficiency have fueled the interest in analog computing [44]. Architectures based on memristor crossbars can be faster and less power hungry in comparison to state-of-the-art digital computing systems still offering an acceptable accuracy for specific applications [45–50].

Section II is organized in two Subsections which illustrate the possible applications of the memristor crossbar as key fundamental building block to compute MAC operations both in Artificial and Spiking neural networks. Section III concludes the Chapter and gives some future directions for expand on this work. The content of this Chapter is a re-elaborated version of two research papers published by the author [51], [1].

3.2 Neuromorphic Applications

In recent times the rise of machine learning and artificial intelligence has created the demand for innovative computing platforms that could offer new, ultra–low power processing methods. Neuromorphic computing tries to go beyond the state-of-the-art in digital processing for certain specific applications (i.e. neural networks, probabilistic models etc.) by mimicking at various degrees the working principles of the brain. In this context the rising interest in the development of novel nonvolatile memory technologies (e.g. memristors) which show complex dynamics and nonlinear phenomena has perfectly matched the need for a compact device implementing the synaptic function ([3, 7]).

The hallmark of memristors is their synaptic plasticity effect which is also observed in biological neural systems. As conductance levels can be modified by regulating the firing activity of pre- and post-synaptic neurons, memristive neural networks enable both the emulation of neurobiological phenomena and the correspondent underlying learning process.

Spiking Neural Networks (SNNs) have attracted the interest of many researchers both in the fields of computational neuroscience and neuromorphic engineering as good candidates for real-time applications if they are implemented in hardware platforms, like memristor crossbars [52, 53]. SNNs have also demonstrated the same computational capability of traditional Artificial Neural Networks (ANNs) [54]. Given the connection between Hebbian learning and spiking [53], SNNs have been studied in the context of supervised, unsupervised and reinforcement learning [55, 56]. Nonetheless, these innovative kind of bioplausible networks have not reached the same accuracy of ANNs yet, primarily because of the lack of suitable and efficient training algorithms.

These novel computing paradigms are not only non-digital in nature, but may also enhance computing speed and power efficiency for multiple sensory streams. This can be obtained by means of the combination of memristor-based crossbars and advanced machine learning algorithms used to train neural networks. Within the realm of supervised learning, the most widespread method for training feedforward ANNs is the backpropagation algorithm. Even though it is recognized as a very powerful technique, its computational complexity is high and is unanimously considered as biologically implausible. Neurological studies have also revealed that the neural coding of information is highly dynamic, thus Recurrent Neural Networks (RNNs) seem a good option to model such a behavior and have been applied to study how neural circuits efficiently complete challenging tasks.

In the first part of this Section, two possible variations of the backpropagation algorithm for training RNNs, namely recurrent backpropagation and equilibrium propagation, are studied and compared on an ideal RNN which exploits the programmable memristor-based crossbar to implement a neural connectome.

In the second part of this Section, it is demonstrated a possible analytical modeling as discrete nonlinear dynamic systems of a memristor-based crossbar implementing a SNN which exploits the ReRAM device model derived in Chapter 2.

3.2.1 Artificial Neural Networks

The first generalization of the backpropagation algorithm to continuous-time recurrent networks came from [57] and [58] who separately obtained the same results. Recurrent backpropagation (RBP) iteratively modify the network's connectome so that, given the initial state and a fixed input, the system converges to a desired attractor state. This is an optimization problem where a Certain loss function associated to the system parameters has to be minimized. The novelty of RBP consists in the introduction of an associated differential equation which backpropagates the error signal. This avoids the direct computation of the gradient thus yielding a reduced number of required multiplications. In spite of being less computationally expensive, the presence of a side network that propagates the error derivatives makes RBP still far from bio-plausible.

Equilibrium Propagation (EP), a novel training method for energy-based models, was recently proposed in [59] as an alternative to the use of a side network for the backwards propagation of error signals. This two-phases learning technique is advantageous as it requires just one type of neural computation to train the network. During the first phase, called free-phase, the inputs are fixed and the network evolves towards a fixed point corresponding to an energy function local minimum. During the second phase, called nudged-phase, a small teaching signal is added to the input which nudges the network state towards a new but close-by fixed point that now corresponds to a lower cost value.

Although the two training methods are quite different, it can be observed how both share the same objective, finding low-energy configurations that have low cost values.

Memristor–based Recurrent Neural Network

In this part of the Subsection about ANN, the two aforementioned variants of the backpropagation algorithm for recurrent neural network are derived. The implementation of these class of networks that exploit the memristive crossbar array to perform MVMs can exploit any generic nonvolatile memory technology (e.g. Phase Change Memory, Resistive RAM, Spin Transfer Torque Memristors etc) as long as the single device is embedded in a suitable synaptic cell circuitry that enables its conductance tuning. The most widespread of these synaptic circuitries is the 1T–1M (one transistor–one memristor) topology which has the memristor connected between the bitline and the drain of a transistor (typically a n-MOS) while the transistor's gate is connected to the wordline. This very efficient IC topology enables the application of fast series of discrete programming pulses during the programming phase of the crossbar and the permanent connection of the desired Subsection of the crossbar to the bitline during inference. The programming can take place either during an online training phase when the crossbar is actively used to perform each step of training or during an offline one when the weight are written on it only at the end of a software performed training phase. In either cases the programming is performed by modulating the number of voltage (or current) pulses applied to the 1T-1M cell and/or the single pulse amplitude/duration (see [60] and [61]).

For the sake of clarity, in this part of the manuscript where the final goal is to introduce RBP and EP and compare them on a crossbar based neural network accelerator no particular technology is chosen and to perform the online training of the network a generic memristor model is used. In particular, let each synaptic weight in the 1T-1M cell be described by a generic memristor (see also [62], [5]) that is described by the following ODE and Ohm's Law:

$$\begin{cases} i = G(\mathbf{x})v \\ \frac{d\mathbf{x}}{dt} = f(\mathbf{x}, v) \end{cases} \tag{3.1}$$

where $\mathbf{x}$ is the memristor internal state vector, $G(\cdot)$ is the memductance, v is the voltage drop between the memristor terminals and i is the current flowing through the memristor. Be $G(\mathbf{x}) = w$ the memristor synaptic weight, in order to give a formal description of the network training via RBP and EP, its state vector can be defined by the two following dynamics:

$$\begin{cases} i = wv \\ \frac{dw}{dt} = f_1(w, v, y) \\ \frac{dy}{dt} = f_2(w, v, y) \end{cases} \qquad \begin{cases} i = wv \\ \frac{dw}{dt} = g(w, v) \end{cases} \tag{3.2}$$

Biologically-plausible learning algorithms

Even though in the last 70 years a lot of progress has been made investigating the rules that govern the learning process in the brain, yet neuroscientist are still far from formulating a complete theory that unveils the mystery of animal intelligence. Albeit in the last decade the success of deep learning in solving complex tasks ([63]) has made clear the role played by high cardinality of the set of neurons involved in neural computation, still the commonly used learning rules (e.g. Backpropagation) substantially differ from how the learning process takes form in the brain which to neuroscientists' best understanding should be local and strictly feedforward. For this reason, researchers in both the machine learning and computational neuroscience communities are currently very interested in designing and studying bio-inspired architectures with local learning rules that approximate the powerful backpropagation training process.

Among those neuron-like approaches other than equilibrium propagation ([59]), it is also worth citing membrane potential based backpropagation algorithm ([64]) and feedback alignment ([65]), target propagation algorithms ([66]). See [67] for an extensive review. Training of recurrent neural networks is the chosen task to evaluate those novel bio-plausible learning techniques. In fact, according to neurological studies, the neural representation is highly dynamic in nature and thus RNNs seem are the best candidates to capture such behavior while solving various computational problems. Even though EP shows a more suitable affinity for VLSI implementations ([59]), RBP still represents the very first attempt in approaching energy-based models in a supervised manner and thus is worth being reported and used as for a comparison. In the following of this Subsection, a brief introduction to the derivation of both algorithms is provided.

Recurrent backpropagation In the following a Recurrent Neural Network is described by its state vector $\mathbf{v}$ whose evolution is governed by:

$$\frac{dv_i}{dt} = -v_i + g_i \left(\sum_{j=1}^{N} w_{ij} v_j + I_i \right), \quad i = 1, \ldots, N \tag{3.3}$$

where I_i is i-th component of the external input vector applied to the i-th neuron and N is the total number of neurons in the network.

Any activation function g_i can be chosen as long as it is differentiable and monotone ([58]).

With the highest degree of generality, neurons in the network can be classified either as input, output or hidden units depending on the particular task to perform.

The aim of the learning procedure is to gradually modify the weights in the connectome w_{ij} so that, given a vector of input $\mathbf{I}$ and an initial condition $\mathbf{v}^0 = \mathbf{v}(t_0)$, the RNN vector state $\mathbf{v}(t)$, under the guidance of the field in (3.3), evolves towards a desired fixed point $\mathbf{v}^\infty = \mathbf{v}(t_\infty)$.

This is achieved by minimizing a loss function E which measures the euclidean distance between the actual fixed point and the desired fixed point:

$$E = \frac{1}{2}\sum_{i=1}^{N} J_i^2 = \frac{1}{2}\sum_{i=1}^{N}(T_i - v_i^\infty)^2 \tag{3.4}$$

where T_i is the i-th component of the desired output vector state $\mathbf{T}$ and J_i is the i-th component of the difference between the current fixed point $\mathbf{v}^\infty$ and $\mathbf{T}$.

Pay attention that E dependence on the weight matrix $\mathbf{W}$ is via the fixed point $\mathbf{v}^\infty(\mathbf{W},\mathbf{I})$.

Thus, a possible way to have the RNN state converge to a desirable attractor is to let it evolve in the weight parameter space along trajectories with opposite direction with respect to the gradient of E:

$$\frac{dw_{ij}}{dt} = -\eta\frac{\partial E}{\partial w_{ij}} = \eta\sum_{k=1}^{N} J_k\frac{\partial v_k^\infty}{\partial w_{ij}}, \quad \eta > 0 \tag{3.5}$$

where η is the learning rate. The first order derivative of v_k^∞ with respect to w_{ij} is computed by observing that the fixed points of (3.3) must satisfy the nonlinear equation:

$$v_k^\infty = g_k\left(\sum_{s=1}^{N} w_{ks}v_s^\infty + I_k\right). \tag{3.6}$$

By differentiating (3.6) with respect to w_{ij}:

$$\frac{\partial v_k^\infty}{\partial w_{ij}} = (\delta_{ki} - g_k'(\hat{I}_k^\infty)w_{ki})^{-1}g_i'(\hat{I}_i^\infty)v_j^\infty \tag{3.7}$$

where δ_{ki} is the Kronecker delta. It is unfortunate that (3.7) involves the calculation of a reciprocal in order to compute the weights' update and because of that in [58] the problem is bypassed by considering

$$y_i = g_i'(\hat{I}_i^\infty)\sum_{k=1}^{N} J_k(\delta_{ki} - g_k'(\hat{I}_k^\infty)w_{ki})^{-1} \tag{3.8}$$

which can be considered as the steady state of the side network:

$$\frac{dy_k}{dt} = -y_k + g'_k(\hat{I}_k^\infty)\left(\sum_{i=1}^{N} w_{ik}y_i + J_k\right). \tag{3.9}$$

Lastly, the weights' update rule is defined as:

$$\frac{dw_{ij}}{dt} = \eta y_i^\infty v_j^\infty \tag{3.10}$$

which consequently depends on the corresponding fixed points of the dynamical systems (3.3) and (3.9).

The complete learning algorithm can summarized as the sequence of phases that follows:

1) Let the dynamical system (3.3) evolve from a random initial state towards the corresponding fixed point $\mathbf{v}^\infty$;

2) Let the dynamical system (3.9) evolve from a random initial state towards the corresponding fixed point $\mathbf{y}^\infty$;

3) Update the weight matrix $\mathbf{W}$ according to:

$$hyperparameters\Delta w_{ij} = \eta y_i^\infty v_j^\infty, \quad \eta > 0. \tag{3.11}$$

Equilibrium propagation Let now energy function E be:

$$E(\mathbf{v}) = \sum_{i=1}^{N} \frac{v_i^2}{2} - \frac{1}{2}\sum_{i,j=1}^{N} w_{ij}g_i(v_i)g_j(v_j) - \sum_{i=1}^{N} I_i g_i(u_i) \tag{3.12}$$

where I_i is i-th component of the external input vector $\mathbf{I}$ applied to the network and neuron N is the number of neurons in the network. As for RBP, also EP does not require a particular choice of the activation functions $g_i(\cdot)\ \forall i = 1, \ldots, N$ as long as the chosen one is monotone and differentiable. Now suppose that the following gradient dynamics governs the time evolution of the state variable **v**:

$$\frac{dv_i}{dt} = -\frac{\partial E}{\partial v_i} = -v_i + g_i^{'}(v_i)\left(\sum_{j=1}^{N} w_{ij} g_i(v_j) + I_i\right), \quad i = 1, \dots, N \tag{3.13}$$

Pay attention on the fact that the network is recurrently connected with symmetric connections (i.e. $w_{ij} = w_{ji}$)). Usually in supervised learning, the output neurons try to recreate the targets **T**. The measure of deviation of the output values of the network, which are the fixed points $\mathbf{v}^{\infty}$, from the targets **T** is the quadratic loss function:

$$C = \frac{1}{2}\sum_{i=1}^{N}(T_i - v_i)^2 \tag{3.14}$$

It is worth noting that, for any state of **v**, the function $C(\mathbf{T}, \mathbf{v})$ is defined. The fundamental idea of EP is to introduce the augmented energy function:

$$F(\mathbf{v}, \mathbf{W}, \mathbf{T}) = E(\mathbf{v}, \mathbf{W}) + \beta C(\mathbf{v}, \mathbf{W}, \mathbf{T}) \quad \mathbf{v}, \mathbf{T} \in \mathbb{R}^N, \mathbf{W} \in \mathbb{R}^{N \times N}, \beta \geq 0 \tag{3.15}$$

and substitute the free dynamics with the augmented dynamics:

$$\frac{dv_i}{dt} = -\frac{\partial F}{\partial v_i} \tag{3.16}$$

The resulting second term $-\beta\frac{\partial C}{\partial v_i}$ gradually nudges **v** towards configurations with a lower cost value. This is achieved, as for RBP, by gradually modifying **W** in order to minimize the cost value of the fixed point $\mathbf{v}^{\infty}$. As a final step required to derive EP learning rule, consider the following objective function

$$J(\mathbf{W}) = C(\mathbf{v}^{\infty}, \mathbf{W}, \mathbf{T}) \quad \mathbf{v}, \mathbf{T} \in \mathbb{R}^N, \mathbf{W} \in \mathbb{R}^{N \times N} \tag{3.17}$$

Note that $J(\mathbf{W})$ is the cost at the fixed point. Equilibrium Propagation approximates the gradient $\frac{\partial J}{\partial \mathbf{W}}$ from measures at the fixed points of the free and the augmented dynamics. The two respective fixed convergence points are referred to $\mathbf{v}^{\infty}$, for free dynamics, and $\mathbf{v}_{\beta}^{\infty}$ for the augmented one. In [59] the authors proved the following equation:

$$\frac{\partial J}{\partial \mathbf{W}} = \lim_{\beta \to 0} \frac{\frac{\partial F}{\partial \mathbf{W}}(\mathbf{v}_{\beta}^{\infty}) - \frac{\partial F}{\partial \mathbf{W}}(\mathbf{v}^{\infty})}{\beta} \tag{3.18}$$

providing an alternative for the estimation of the objective function gradient. Consequently, the network implements in sequence the following dynamics during the training phase:

1) Free dynamics: **T** is clamped in input and the network state vector evolves under (3.13) relaxing to the free fixed point $\mathbf{v}^{\infty}$ where $\frac{\partial F}{\partial \mathbf{W}}(\mathbf{v}^{\infty})$ is first measured;

2) Weakly clamped phase: the influence parameter is injected as input to the network which relaxes to a new and nearby fixed point $\mathbf{v}_{\beta}^{\infty}$ where $\frac{\partial F}{\partial \mathbf{W}}(\mathbf{v}_{\beta}^{\infty})$ is measured.

3) Weights dynamics: according to (3.18) the interconnection matrix **W** is modified in a discrete step as follows:

$$\Delta w_{ij} \propto -\eta [g_i(v_i^{\beta,\infty}) g_j(v_j^{\beta,\infty}) - g_i(v_i^{\infty}) g_j(v_j^{\infty})], \quad \eta > 0. \tag{3.19}$$

Implementation and experimental results

In the two following paragraphs experimental evidence of the two models' efficiency is provided. In the first paragraph both EP and RBP are proven on a pattern reconstruction task. In the second paragraph the classification of a subset of the MNIST dataset is reported using Equilibrium Propagation as learning rule.

Pattern reconstruction In this paragraph, a comparison among the two aforementioned training techniques for pattern's reconstruction task is presented. For this task, the network is designed to have input and output units coincident with the same neurons and no hidden units are considered. Furthermore, symmetric weights are chosen for both methods as the construction of the gradient dynamics (3.13) implies. This condition is also sufficient to ensure the convergence of the model (3.3). Each pattern in Figure 3.2, during the training phase, is repeatedly fed to the RNN via a constant input **I** until it is memorized by the network. If multiple patterns are to be stored, the previous steps are iterated for each single image of the dataset for many epochs. In this manuscript, for the reported set of numerical experiments, the patterns were fed to the network in the same order for each epoch. This experiment design choice was found to be not restrictive as similar performances were also obtained by feeding the patterns in a random fashion.

During training, the following hyperparameters and initial conditions were set:

- Random initialization of the state vector $\mathbf{v}$;
- For RBP, whenever the first state vector $\mathbf{v}$ converges , the second state vector is reset to $\mathbf{y}(0) = (0.5, \ldots, 0.5)^T \in \mathbb{R}^N$;
- The interconnection matrix **W** is initialized with uniform random values between $[-0.1; 0.1]$;
- The used activation functions $g_i \; \forall i = 1, \ldots, N$ are hyperbolic tangent functions;
- The learning rate $\eta = 0.01$;
- The maximum number of epochs is 300.
- Time spans for simulating the dynamics of the systems are chosen long enough to guarantee the convergence of the state vector(s).

Additionally, a brief analysis on the importance of local-global interconnections in the RNN's connectome is performed in order to assess the implementability of this solution in an actual IC. The reader interested in the relation between the network topology and the computational performance of attractor neural networks may refer to [68], [69] or [70], [71] for further insights on current approaches for improving energy efficiency of hardware-implemented neural networks by means of sparse and less costly number of interconnections. In this work, for the sake of lower complexity, a naive investigation was performed by increasingly cutting out global connections from a full matrix by simply setting to zero all the entries that were located outside a band about the main diagonal. The cut out of K outer diagonals from the matrix decreases the number of synapses from N^2 to $N^2 - K(K+1)$.

With the aim of testing the network, corrupted patterns were randomly generated by flipping each pixel of the image either from white to black or the other way around with a uniform probability p. In the proposed analysis, a noisy pattern is considered reconstructed if the least square error with respect to the uncorrupted images is null. The network was validated by testing its recovery capabilities on 5000 corrupted patterns for each one of the classes shown in Figure 3.2. The results measured on the RNN trained by means of both the aforeintroduced learning algorithms are shown in Figure 3.3 parameterized on different levels of applied noise (e.g. $p = 0.10$, $p = 0.15$, $p = 0.20$ and $p = 0.25$). It can be observed that both methods reach promising results in the case of full RNN's connectome. Nonetheless, EP can get better results even with a highly pruned connectome. This evidence, together with the cut of the side network, motivates us to investigate EP as a solution worthy of consideration for a VLSI implementation. The report improvement could be induced by the noisy estimator of the gradient given by (3.19) which lets the network efficiently explore the parameter space by escaping from local minima. Additionally, this may also be noticed in Figure 3.4 where high accuracy values are obtained by Equilibrium Propagation already in the first 50 epochs of training whereas Recurrent Backpropagation needs 300 epochs at least.

As a last analysis to show the efficiency of the two novel methods, a comparison is performed with two classical learning rules for training RNNs as associative memories. It is commonly recognized that a Hopfield neural network trained on uncorrelated patterns by means of the Hebbian rule can memorize $\approx 0.14N$ (where N is the number of neurons in the network) [72]. However, this storage capacity substantially lowers when patterns are correlated. This problem has been overcome by a novel learning algorithm introduced by [73] which significantly improved performances over the older Hebbian learning rule for the Hopfield model, both with correlated and uncorrelated data.

Learning Rule	Accuracy
Hebbian	0.1792
Storkey	0.2663
Recurrent BackProp	0.9968
Equilibrium Propagation	0.9971

Table 3.1 Accuracy for each single learning rule over 1000 corrupted images, with probability 0.1, for each of the 16 classes.

However, as Table 3.1 and the examples of Figure 3.5 show, the evidence is that both RBP and EP learning rules can teach the RNN to reconstruct even in the presence of correlated patterns.

Pattern classification As a second application of the learning algorithm proposed by [59] a RNN is tested on a pattern classification task. The model has symmetric connections and includes only 1 hidden layer and 1 output layer. No skip-layer and no lateral connections are included. The RNN is trained using EP to classify patterns from a subset made of 5 classes from the MNIST dataset, 600 patterns per class. The hard sigmoid was selected, following [59], as activation function.

Training the network consisted in the iteration of the following ordered steps:

1) Set a constant input pattern to the network;

2) Let the network freely evolve until the hidden and the outputs units converge to a stable equilibrium and store $g(v_i^{\infty})g(v_j^{\infty})$;

3) Apply the nudging signals, let the network evolve towards a different but closeby equilibrium point and store $g(v_i^{\beta,\infty})g(v_j^{\beta,\infty})$;

4) Update the connectome according to (3.19).

With the aim of performing the training process, (3.16) was discretized into short time steps of duration ϵ as follows:

$$v_{t+1} = v_t - \epsilon \frac{\partial F}{\partial v_i} \tag{3.20}$$

Nonetheless, as advised by [59], the state variable should be confined in the range $[0\,,1]$ and thus a slightly variation of the update rule was applied:

$$v_{t+1} = g\left(v_t - \epsilon \frac{\partial F}{\partial v_i}\right) \tag{3.21}$$

with $g(\cdot)$ being the hard sigmoid function.

The identified class corresponds to the index of the output unit that took the maximum value among all the others.

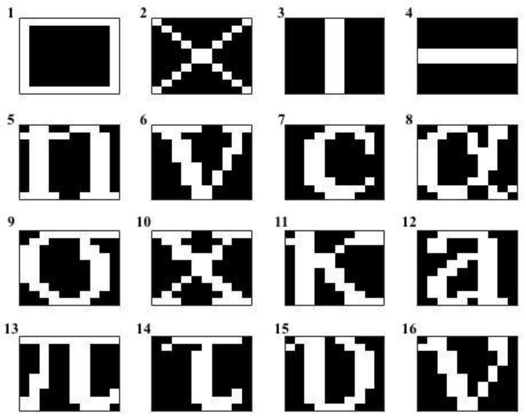
The hyperparameters were all chosen following the guidelines advised in [59]: the learning rate is set as $\epsilon = 0.5$ during the iterative inference, $\beta = 1$ is the the clamping factor value in the nudged phase, $\alpha_1 = 0.1, \alpha_2 = 0.05$ are the two learning rates for updating the weights in the first and second layer respectively.

Pay attention to the fact that in this work the learning rate η is not considered unique as in (3.19). On the other hand, instead of selecting a random sign for β during the second phase, the two learning rates α_1, α_2 are halved at the end of each epoch.

The obtained results are reported in Figure 3.6 and seem to be consistent with the data in [59].

3.2.2 Spiking Neural Networks

The study of biological neural networks revealed that the synaptic connection between two neurons is mostly strengthened depending on the coordinated activity of the pre-synaptic neuron and post-synaptic neuron it links rather than computations of all downstream neurons ([74], [75]).



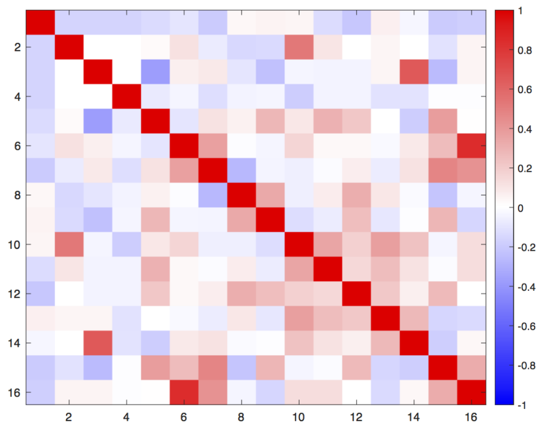
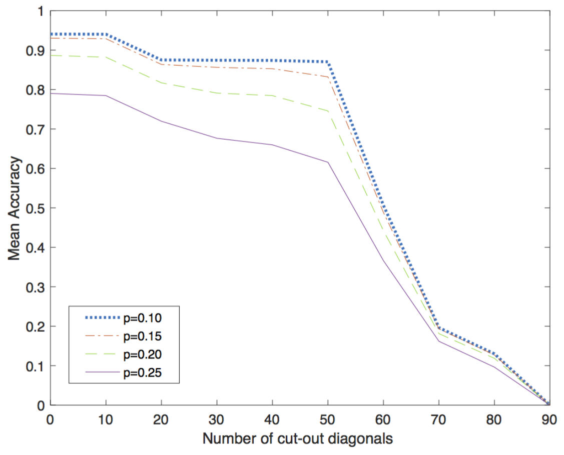


Fig. 3.2 In the top panel, the dataset of all the 16 patterns to be learnt. In the bottom panel, a graphical representation of the corresponding patterns' correlation matrix computed with the Pearson correlation coefficient. Taken from [51] ©2020 Frontiers.



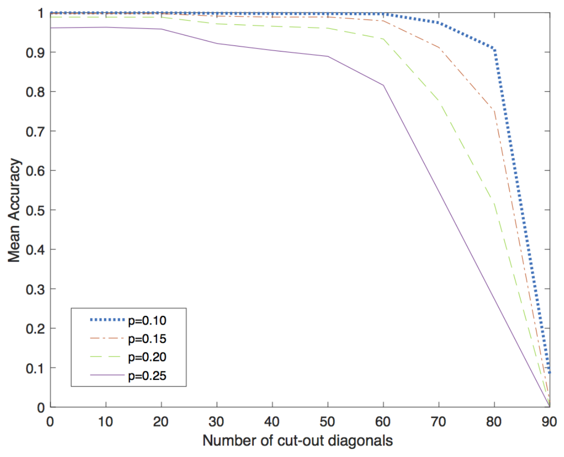


Fig. 3.3 Accuracy for different radius of connectivity. At the top, the results obtained by using recurrent backpropagation and at the bottom, the results obtained by using equilibrium propagation. p is the probability of flipping each pixel of the image from white to black and viceversa. Taken from [51] ©2020 Frontiers.

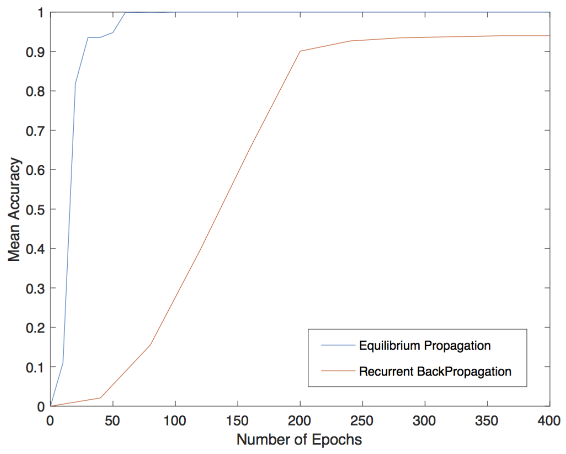


Fig. 3.4 Mean accuracy over 1000 reconstructed patterns for different number of epochs using Equilibrium Propagation (in blue) and Recurrent Backpropagation (in orange). Taken from [51] ©2020 Frontiers.

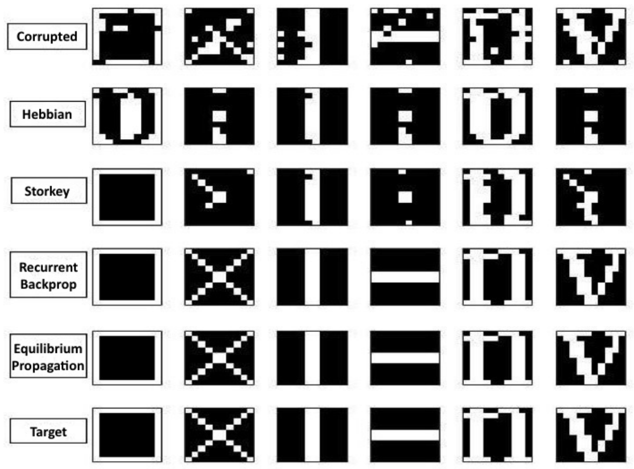


Fig. 3.5 From the top row: six corrupted patterns with probability $p = 0.1$, reconstructed pattern with Hebbian rule, Storkey rule, recurrent backpropagation rule, Equilibrium Propagation rule and in the last row the target patterns. Taken from [51] ©2020 Frontiers.

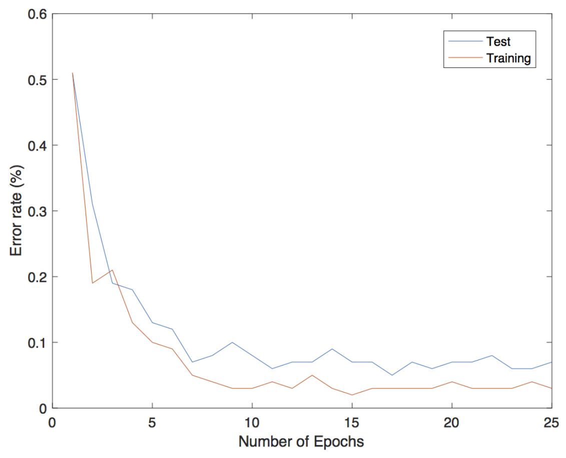


Fig. 3.6 Error rates of the trained neural network over 100 random patterns chosen among the training set (in orange) and 100 patterns from the test set (in blue) using Equilibrium Propagation learning rule. Taken from [51] ©2020 Frontiers.

Spiking Memristor networks

Spiking neural networks employing second order memristors as synaptic weights have exhibited the capability of classifying either dynamic or static data and of processing temporal events, by creating unique patterns, resulting from local synaptic potentiation and depression [76].

By using the simplified model introduce in Chapter 2, in this Subsection it will be shown how the pattern formation process in such networks can be studied and deeply investigated by treating them as discrete nonlinear dynamical systems.

Consider a structure made of N presynaptic neurons and M postsynaptic neurons, linked through a connectome of second-order memristors, which exhibit a conductance modeled by eqs. (2.15)-(2.16) and (2.20).

Input data might be encoded with either temporal or rate code, creating a set of presynaptic spikes for each neuron.

For completeness' sake, let's recall that all presynaptic/postsynaptic spikes can be modeled as a positive/negative programming pulse of magnitude V_{pre}/V_{post} lasting for t_s, which are followed by longer negative/positive heating pulses of amplitude V_H and duration t_H.

Be X_j and Y_i the sets of all the spikes of the j-th presynaptic neuron and i-th postsynaptic neuron respectively:

$$\begin{aligned} X_j &= \left\{ t^{pre}_{j,1}, t^{pre}_{j,2}, \ldots, t^{pre}_{j,K_j} \right\} \\ Y_i &= \left\{ t^{post}_{i,1}, t^{post}_{i,2}, \ldots, t^{post}_{i,H_i} \right\} \end{aligned} \tag{3.22}$$

with $t^{pre}_{j,k}$ $(1 \leq k \leq K_j)$ being the time when j-th neuron fires its k-th presynaptic spike and $t^{post}_{i,h}$ $(1 \leq h \leq H_i)$ being the time when i-th neuron fires its h-th postsynaptic spike.

For the chosen convention, all spikes are emitted at the beginning of the programming pulse of duration t_s and the i-th postsynaptic neuron voltage $u_i(t)$ drops to zero, immediately after the given threshold is reached, and thus it fires a postsynaptic spike $t^{post}_{i,h}$. In this formalism, the voltage $u_i(t)$ turns can be described by means of the following formula:

$$\begin{aligned} u_i(t) &= R \sum_{h=1}^{H_i+1} \sum_{j=1}^{N} \sum_{t^{pre}_{jk} \in (t^{post}_{i,h-1}, t^{post}_{i,h})} \hat{\mathbf{G}}_{i,j}(t^{pre}_{j,k})\, v^{pre}_{j,k}(t) \\ v^{pre}_{j,k}(t) &= V_H \left[\epsilon(t - t_{sh} - t^{pre}_{j,k}) - \epsilon(t - t_{sh} - t_H - t^{pre}_{j,k}) \right] \\ &+ V_P \left[\epsilon(t - t^{pre}_{j,k}) - \epsilon(t - t_s - t^{pre}_{j,k}) \right] \\ \epsilon(t) &= \Theta(t) \left[1 - \exp\left(-\frac{t}{\tau_m} \right) \right] \end{aligned} \tag{3.23}$$

with $[t^{post}_{i,0}\, t^{post}_{i,H_i+1}]$ representing the interval under consideration, R being a resistive constant, $\hat{\mathbf{G}}_{i,j}(t^{pre}_{j,k})$ indicating the value of the memductance at $t = t^{pre}_{j,k}$, τ_m being the postsynaptic neuron time constant, $\Theta(\cdot)$ being the Heaviside function, and t_{sh} denoting the time shift between the beginning of the programming pulse and the beginning of the heating pulse as shown in Fig. (3.7).

Be u^{th} the voltage threshold common to all the neurons in the network and $\hat{\mathbf{G}}^{h}_{i,j}$ the value taken by the memductance at $t = t^{post}_{i,h}$, during the h^{th}

postsynaptic spike of the i-th neuron, the memductance dynamics can correspondingly be modeled by the following set of $N \times M$ discrete-time state equations:

$$\begin{aligned}\hat{\mathbf{G}}_{i,j}^{h+1} &= \min\left\{G_{max}, \hat{\mathbf{G}}_{i,j}^{h+1/2} + \Delta_G^{pre/post}(\hat{\mathbf{G}}_{i,j}^{h+1/2}, \gamma^{pre/post})\right\} \\ \hat{\mathbf{G}}_{i,j}^{h+1/2} &= \max\left\{G_{min}, \hat{\mathbf{G}}_{i,j}^{h} + \Delta_G^{post/pre}(\hat{\mathbf{G}}_{i,j}^{h}, \gamma^{post/pre})\right\}\end{aligned} \tag{3.24}$$

$$\begin{aligned}\gamma^{post/pre} &= \frac{t_{j,k_F^h}^{pre} - t_{i,h}^{post} - t_{sh}}{t_H} \\ \gamma^{pre/post} &= \frac{t_{i,h+1}^{post} - t_{j,k_L^h}^{pre} - t_{sh}}{t_H}\end{aligned} \tag{3.25}$$

where

$$u_i(t_{i,h}^{post}) = u^{th} \text{ and } \forall t \neq t_{i,h}^{post} \rightarrow u_i(t) < u^{th} \ \ (1 \leq h \leq H_i) \tag{3.26}$$

In the above equations, k_F^h and k_L^h are the indexes of the first and of the last presynaptic spike of j-th neuron in the time interval $[t_{i,h}^{post}, t_{i,h+1}^{post})$, $\hat{\mathbf{G}}_{i,j}^{h+1/2}$ expresses the memductance value due to the first variation $\Delta_G^{post/pre}$ caused by the post/pre pair and $\hat{\mathbf{G}}_{i,j}^{h}$ expresses the final value of the memductance due to the second variation $\Delta_G^{pre/post}$ caused by the pre/post pair. G_{max} and G_{min} stand for the maximum and the minimum values, that, according to (2.13), each memductance can reach:

$$\begin{aligned}G_{max} &= \hat{\mathbf{G}}(r_0) = \frac{1}{2R_s} \\ G_{min} &= \hat{\mathbf{G}}(r_m) = \frac{1}{R_s\left[1 + \left(\frac{r_0}{r_m}\right)^2\right]}\end{aligned} \tag{3.27}$$

The memductance variations $\Delta_G^{post/pre}(\hat{\mathbf{G}}_{i,j}^{h}, \gamma^{post/pre})$ and $\Delta_G^{pre/post}(\hat{\mathbf{G}}_{i,j}^{h+1/2}, \gamma^{pre/post})$ may be directly calculated by substituting in (2.15)-(2.16) the memductance values and in (2.20) the opportune values of $\gamma^{post/pre}$, $\gamma^{pre/post}$ expressed in (3.25).

The ensemble of equations (3.24) - (3.25) - (3.26) together with (3.23) may be adequately employed to analyze the dynamics of a memristor spiking neural network, with any arbitrary presynaptic input spikes X_j.

Network Response to Presynaptic Periodic Input Spikes With the aim of demonstrating the power of the introduced analysis method, in this paragraph the network response to a sequence of periodic presynaptic spikes is used as a case study. By means of the formalism in (3.22), assuming $K_j = 1$ for all cells, the sequence of N presynaptic spikes may be pictured by the vector below when the time shift between two subsequent spikes is constant and equal to a fixed period T:

$$\mathcal{I} = \left\{ t_{1,1}^{pre}, t_{2,1}^{pre}, \cdots, t_{N,1}^{pre} \right\} \quad (t_{j+1,1}^{pre} - t_{j,1}^{pre} = T, \ 1 \leq j \leq N-1) \tag{3.28}$$

Given that memristor crossbars' columns are ideally all decoupled, without losing in generality, consider a generic i-th postsynaptic neuron connected to the memristors which inject current on its wordline only. In accordance with (3.23), the contribution of the programming pulses to the membrane voltage $u_i(t)$ is considered negligible due to their brief duration. Therefore, the threshold u^{th} can be reached and trigger a postsynaptic spike only in correspondence to a presynaptic neuron heating pulse. In this event, let's call $\alpha_j t_H$ ($\alpha_j > 0$) the time delay interleaving between the postsynaptic spike and the beginning of the generic j-th heating pulse at which the threshold u^{th} is crossed (check Fig. (3.7) for more details). The following Proposition holds:

Proposition 1. *Consider a network which includes N presynaptic neurons and one single postsynaptic neuron and assume N to be a multiple of an integer P and that the input $\mathcal{I}$, given by (3.28), is presented multiple times to the postsynaptic neuron. Consider the following $P+1$ equations set:*

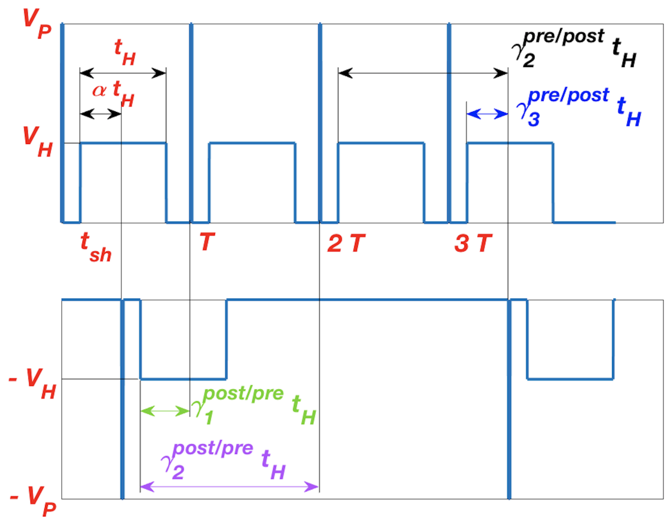

V_P
t_H
$\alpha\, t_H$
V_H
t_{sh}
T
$2\,T$
$3\,T$
$\gamma_2^{pre/post}\, t_H$
$\gamma_3^{pre/post}\, t_H$
$-V_H$
$\gamma_1^{post/pre}\, t_H$
$\gamma_2^{post/pre}\, t_H$
$-V_P$

Fig. 3.7 Example of a sequence of presynaptic input spikes of period T (upper figure), giving rise to postsynaptic output of period $3T$. Each pre/post synaptic spike is represented by a programming pulse of magnitude V_P and duration t_s, followed by a heating pulse of duration t_H; the time interval between the beginning of the programming pulse and the beginning of the heating pulse is denoted with t_{sh}. It is assumed that the postsynaptic spike occurs αt_H time units after the beginning of one presynaptic spike (with $0 \leq \alpha \leq 1$) and the following parameters, reported in (3.31), are shown for some mem-conductances: $\gamma_{1,2}^{post/pre}$ (related to the time shift between the beginning of the presynaptic programming pulses and the beginning of the postsynaptic heating pulse) and $\gamma_{2,3}^{pre/post}$ (related to the time shift between the beginning of the postsynaptic programming pulse and the beginning of the presynaptic heating pulses). Taken from [1] ©2022 IEEE.

$$\Delta^{pre/post/pre}(\hat{\mathbf{G}}_p^0, \gamma_p^{post/pre}, \gamma_p^{pre/post}) = \hat{\mathbf{G}}_p^1 - \hat{\mathbf{G}}_p^0 = 0 \qquad (1 \le p \le P) \tag{3.29}$$

$$\begin{aligned} \hat{\mathbf{G}}_p^1 &= \min\left\{G_{max}, \hat{\mathbf{G}}_p^{1/2} + \Delta_G^{pre/post}(\hat{\mathbf{G}}_p^{1/2}, \gamma_p^{pre/post})\right\} \\ \hat{\mathbf{G}}_p^{1/2} &= \max\left\{G_{min}, \hat{\mathbf{G}}_p^0 + \Delta_G^{post/pre}(\hat{\mathbf{G}}_p^0, \gamma_p^{post/pre})\right\} \end{aligned} \tag{3.30}$$

$$\begin{aligned} \gamma_p^{post/pre} &= \frac{pT - \alpha t_H - 2t_{sh}}{t_H} \\ \gamma_p^{pre/post} &= \frac{(P-p)T + \alpha t_H}{t_H} \end{aligned} \tag{3.31}$$

$$\begin{aligned} u^{th} = RV_H \Bigg\{ &\sum_{p=1}^{P-1} \hat{\mathbf{G}}_p^{1/2} \exp\left(-\frac{(P-p)T}{\tau_m}\right) \left[\exp\left(-\frac{(\alpha-1)t_H}{\tau_m}\right) \right. \\ &\left. - \exp\left(-\frac{\alpha t_H}{\tau_m}\right)\right] + \hat{\mathbf{G}}_P^{1/2}\left[1 - \exp\left(-\frac{\alpha t_H}{\tau_m}\right)\right]\Bigg\} \end{aligned} \tag{3.32}$$

and the following inequality:

$$RV_H \sum_{p=1}^{P-1} \hat{\mathbf{G}}_p^{1/2} \exp\left(-\frac{(P-p)T}{\tau_m}\right)\left[1 - \exp\left(-\frac{(t_H)}{\tau_m}\right)\right] < u^{th} \tag{3.33}$$

When the inequality (3.33) holds and there exist $\alpha > 0$ and P memductances $\hat{\mathbf{G}}_p^0$, $(1 \le p \le P)$ which satisfy the above $P+1$ equations set (3.29) - (3.32), then the discrete time system (3.24) - (3.26) shows a steady state solution, which has the two features: 1) memductance patterns display a spatial periodicity of order P; 2) postsynaptic spikes fire with a temporal periodicity of PT.

Proof. The considered periodic input $\mathcal{I}$ shows one presynaptic spike per presynaptic neuron, with period T, i.e. in accordance with (3.28) $t_{j+1}^{pre} -$

$t_j^{pre} = T$. Consider the time delay between two generic postsynaptic spikes $[t_h^{post}, t_{h+1}^{post})$.

The last presynaptic spike prior to the postsynaptic spike t_h^{post} is fired by the the presynaptic neuron of order J and P presynaptic spikes occur in the time interval $[t_h^{post}, t_{h+1}^{post})$. In accordance with (3.23), by not considering the programming pulse effect, the voltage $u(t)$, due to the input $\mathcal{I}$, in the generic interval $[t_h^{post}, t_{h+1}^{post})$ can be written as follows:

$$\begin{aligned} u(t) &= R \sum_{p=1}^{P} \hat{\mathbf{G}}_{J+p}(t_{J+p}^{pre})\, v_{J+p}^{pre}(t) \\ v_{J+p}^{pre}(t) &= V_H \left[\epsilon(t - t_{sh} - t_{j,k}^{pre}) - \epsilon(t - t_{sh} - t_H - t_{j,k}^{pre}) \right] \end{aligned} \tag{3.34}$$

As aforesaid, the threshold u^{th} can be crossed during a presynaptic heating pulse only. Without a loss of generality, as pictured in Fig. (3.7), it can be assumed that the threshold crossing, which fires the postsynaptic spike t_h^{post} occurs after a time αt_H since the beginning of the J^{th} presynaptic heating pulse.

If the voltage $u(t)$ drops to zero after the postsynaptic spike, then the expression (3.34) takes the form:

$$\begin{aligned} u(t) &= R V_H \sum_{p=1}^{P} \hat{\mathbf{G}}_{J+p}(t_{J+p}^{pre}) \Big[\epsilon(t - t_h^{post} - pT + \alpha t_H) \\ &\quad - \epsilon(t - t_h^{post} - pT + \alpha t_H - t_H) \Big] \end{aligned} \tag{3.35}$$

It should be noted that the index J, which expresses the presynaptic spike order, preceding the t_h^{post} postsynaptic spike, can be omitted. This turns to be very helpful in order to simplify (3.35). Furthermore, by denoting with $\hat{\mathbf{G}}_p^0$ the memductance values in correspondence of the t_h^{post} postsynaptic spike, then, following (3.24) and (3.30), $\hat{\mathbf{G}}_p(t_p^{pre})$ is given by $\hat{\mathbf{G}}_p^{1/2}$, i.e. it is obtained by $\hat{\mathbf{G}}_p^0$ through a post/pre variation.

By means of the just introduced simplified notation, the following formulation of the voltage $u(t)$ at $t = t_h^{post} + PT$ can be readily obtained from (3.35):

$$u(t_h^{post} + PT) = RV_H \left\{ \sum_{p=1}^{P-1} \hat{\mathbf{G}}_p^{1/2} \exp\left(-\frac{(P-p)T}{\tau_m}\right) \cdot \left[\exp\left(-\frac{(\alpha-1)t_H}{\tau_m}\right) - \exp\left(-\frac{\alpha t_H}{\tau_m}\right)\right] + \hat{\mathbf{G}}_P^{1/2}\left[1 - \exp\left(-\frac{\alpha t_H}{\tau_m}\right)\right]\right\} \tag{3.36}$$

Equivalently, the postsynaptic voltages at the end of the heating pulses of the first $P-1$ presynaptic spikes, can be obtained from (3.35) as $u_q = u[t_h^{post} + qT + (1-\alpha)t_H]$, $(1 \leq q \leq P-1)$:

$$u_q = u[t_h^{post} + qT + (1-\alpha)t_H] = RV_H \left\{ \sum_{p=1}^{q} \hat{\mathbf{G}}_p^{1/2} \cdot \exp\left(-\frac{(q-p)T}{\tau_m}\right)\left[1 - \exp\left(-\frac{t_H}{\tau_m}\right)\right]\right\} \tag{3.37}$$

The following considerations hold:

1. When the ensemble of P equations (3.30)-(3.31) is satisfied, all the memductances $\hat{\mathbf{G}}_p^0$, $(1 \leq p \leq P)$ turn out to be unchanged following a pre/post/pre triplet.

2. The right side of (3.36) is equal to the right side of (3.32), therefore if the supplementary $P+1$ equation (3.32) is also satisfied, then the voltage threshold u^{th} is crossed at $t = t_h^{post} + PT$. As a result, PT is the time delay between the two subsequent postsynaptic spikes under consideration t_h^{post} and t_{h+1}^{post}, i.e. $t_{h+1}^{post} - t_h^{post} = PT$.

3. It can be easily checked that the voltages u_q in (3.37) do satisfy the following property:

$$u_q \leq u_{q+1}, \ (1 \leq q \leq P-2) \tag{3.38}$$

and thus the maximum value of u_q is u_{P-1}. As the maximum of $u(t)$ is at the end of the heating pulse of a presynaptic spike, u_{P-1} also expresses the maximum value assumed by $u(t)$ resulting from the contribution of the first $P-1$ presynaptic spikes. Note that the formulation of u_{P-1}, i.e. (3.37) with $q = P-1$, is equal to the left side of the inequality in (3.33). Thus, when (3.33) holds, the voltage $u(t)$ stays lower than the threshold u^{th} until when the heating pulse of the $P-1^{th}$ presynaptic spike ends. This in turn implies that, in accordance with (3.36), the threshold is crossed for the first time at $t = t_{h+1}^{post} = t_h^{post} + PT$.

As a final step in the proof, consider a pattern made of a replica of P memductances $\hat{\mathbf{G}}_1^0, \hat{\mathbf{G}}_2^0, \cdots, \hat{\mathbf{G}}_P^0$ which satisfy (3.29) - (3.32), and (3.33). It can be easily verified that such a pattern is an equilibrium point of the discrete-time dynamical system modeled by (3.24) - (3.26), which demonstrates the existence of a steady state solution with spatial periodicity P. Furthermore, given that for such a pattern $t_{h+1}^{post} - t_h^{post} = PT$, it is also demonstrated the second part of the thesis, i.e. the periodicity of PT in the occurrence of postsynaptic spikes. ■

It should be noted that the thorough and meticulous study of the curves which represent $\Delta^{pre/post/pre}(\hat{\mathbf{G}}_p, \gamma_p^{post/pre}, \gamma_p^{pre/post})$ as a function of $\hat{\mathbf{G}}_p$, it follows that either they have no zeroes (which means that the memductance eventually saturates to G_{min} or G_{max}) or they show a negative slope at the zero, consequently ensuring the stability of the solution.

The results presented in this Subsection may have significant applications. A first example concerns the analysis of spiking neural networks dynamics, and the subsequent characterization of space and temporal periodic patterns. A second application may regard the study of unsupervised learning mechanisms which occurs in memristive neural networks. Numerical solutions of (3.29) - (3.32) demonstrate that all presynaptic frequencies are dynamically encoded onto a memductance patterns. Therefore such patterns are

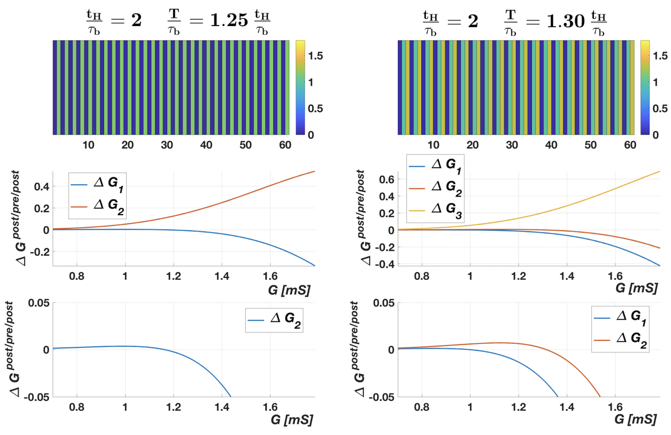


Fig. 3.8 Spiking network composed by 60 presynaptic neurons and one postsynaptic neuron. Upper part: mem-conductance pattern periodicity (2, for $\frac{T}{\tau_b} = 1.25\frac{t_H}{\tau_b}$, and 3, for $\frac{T}{\tau_b} = 1.3\frac{t_H}{\tau_b}$); lower parts: mem-conductance variations, due to a sequence of post/pre/post spikes. Taken from [1] ©2022 IEEE.

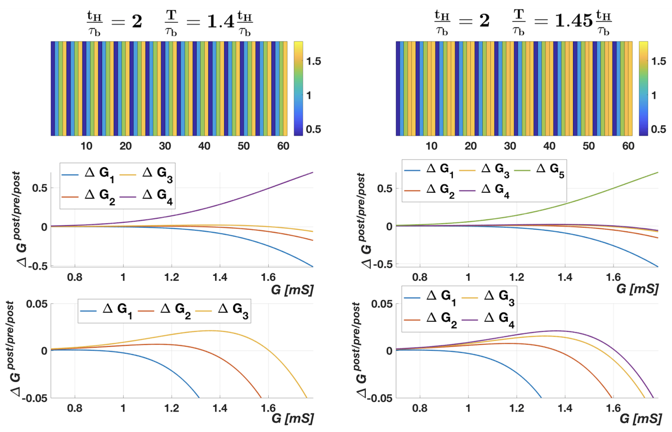


Fig. 3.9 Spiking network composed by 60 presynaptic neurons and one postsynaptic neuron. Upper part: mem-conductance pattern periodicity (4, for $\frac{T}{\tau_b} = 1.4\frac{t_H}{\tau_b}$, and 5, for $\frac{T}{\tau_b} = 1.45\frac{t_H}{\tau_b}$); lower parts: mem-conductance variations, due to a sequence of post/pre/post spikes. Taken from [1] ©2022 IEEE.

exploitable in order to classify different ensembles of presynaptic spikes. A last application example regards the possibility of engineering supervised learning methods with the aim of optimizing neural network performance. As reported in [55], this would require estimating the memductance pattern which maximizes the probability of a given postsynaptic output. A deeper elaboration of the proposed dynamic system-based approach could well serve the purpose of searching the required memductance pattern.

The results of the analysis of a SNN containing 60 presynaptic neurons a 1 postsynaptic by means of the proposed approach are reported in Fig. 3.8 and 3.9 for $t_H = 2\tau_b$ and increasing values of the input period. It should be noted that

- The memductance pattern and the postsynaptic spike periodicity increase proportionally to the presynaptic periodic input T, spanning from a spatial periodicity of 2 (with temporal period $2T$ and $\frac{T}{\tau_b} = 1.25\frac{t_H}{\tau_b}$) to a spatial periodicity of 5 (with temporal period $5T$ and $\frac{T}{\tau_b} = 1.45\frac{t_H}{\tau_b}$);
- The initial $P-1$ memductances evolve towards a stable value within the range (G_{min}, G_{max}), characterized by a zero of the post-pre-post curve, with a negative slope;
- The P^{th} memductance takes the value G_{max}, coherently with (3.30), due to the positivity of the post-pre-post variation for any $\hat{\mathbf{G}}$ within the interval (G_{min}, G_{max});
- The neural network simulation which involved the application of 240 iterations of each input sequence, can precisely reproduce the results theoretically predicted by (3.29) - (3.32).

3.3 Conclusion

In the first part of this Chapter the dynamics of memristor–based recurrent artificial neural networks has been studied. The analyzed network was trained by using two distinct generalizations of the backpropagation algorithm tailored for continuous-time energy-based models. Such local learning rules enable the gradual adjustment of the memristor–based artificial neural network connectome without computing the gradient of the loss function. Even if additional experimental work is needed to find the best memristive technology to implement such networks, the performed analysis shows two suitable learning rules applied to the synaptic weights' update. Those two learning algorithms can be implemented in a by a series of discrete programming pulses in a READ-WRITE scheme. The results obtained from simulations make clear that both rules largely outperform conventional approaches for training RNNs for pattern reconstruction tasks. Furthermore, equilibrium propagation has also proven to well perform on classification tasks.

In the second part of this Chapter memristor spiking neural networks have been characterized as discrete nonlinear dynamic systems, with memductances playing the role of state variables and presynaptic/postsynaptic spikes as inputs and outputs. By means of explicitly formulated state equations, given in Chapter 2, which govern the memductance evolution, it has been shown how the network response to periodic presynaptic inputs can be easily found by computing the system equilibria and analyzing their stability properties. Even if additional work is necessary, the proposed approach provides a theoretical framework for understanding temporal learning properties of second order memristors [52, 76]. Furthermore it has the potential, employed with advanced nonlinear dynamic techniques, of enabling the investigation of the response of memristor networks to arbitrary presynaptic inputs and the underlying learning mechanisms.

Chapter 4

Memristor-based Linear Algebra Accelerators

4.1 Introduction

As conventional memristor crossbars can compute the MVM in $O(1)$, it has also been shown that linear systems and constrained eigenvector equations can be solved by the same 2D array of memristors in $O(1)$ complexity [77] by slightly changing the input/output interconnections. For all those applications anyway the benefit of memristor crossbars is the merge of information storage and computation which avoids the von Neumann bottleneck. Notably, MVM time complexity using a memristor crossbar does not depend from the matrix size.

The aim of this Chapter is to provide a thorough study of a proposed memristor crossbar-based accelerator. The problem the system is designed to solve by means of the memristor crossbar is the acceleration of the Markov Chain (MC) inference process. Discrete MC models are a widespread class of statistical models used, to name a few, in medical decision making, complex networks' analysis, large-scale economics, biology, engineering, physics and one of the most famous real-world applications is Google's Pagerank algorithm [78–80]. These class of problems are typically solved using matrix methods. Two different accelerator architectures are proposed in the following Chapter each one with it strengths and complexities. On the one hand the first architecture, which is a common MVM accelerator, will be referred to as the open-loop configuration and will compute the successive updates to a discrete Markov process in a step-by-step fashion. On the other hand the second proposed architecture, originally conceived by Ielmini and co-workers, will be referred to as the feedback configuration and takes the solution of this linear algebraic problem one step further by requiring a single time step to directly compute the MC stationary distribution with very little hardware overhead. In pursuit of this aim of studying and comparing these two solutions, the mathematical tools need to evaluate a memristor crossbar-based accelerator are introduced in terms of noise contributions, systematic errors and trade-offs for energy efficiency. The obtained simulation results for the two architectures are compared with the state of the art in digital computations.

In Section II a brief introduction to the theoretical background of MCs is given. In Section III different mathematical approaches to solve discrete-time MCs are described. Section IV presents the implementation of MCs on memristor crossbars and the use of conversion algorithms. In Section V a detailed precision analysis of the proposed architectures is carried out. In Section VI the system accuracy is evaluated. Section VII benchmarks the system performance for two different applications with random and ill-conditioned matrices. In Section VIII the conclusions about this study are drawn. The content of this Chapter is a re-elaborated version of the research paper published by the author [2].

4.2 Theoretical background

Markov models appeared in the scientific literature of the early 1900s to forecast the evolution of a system that switches randomly between states. When the future behavior of the system depends on the recent past only, with very weak influence from the remote history, the system can be modeled by a Markov process. If the space of states is discrete, a Markov process is called a Markov Chain.

MC models are described by a state vector and a transition matrix that can be used to forecast the system future states. Diffuse attention has been payed

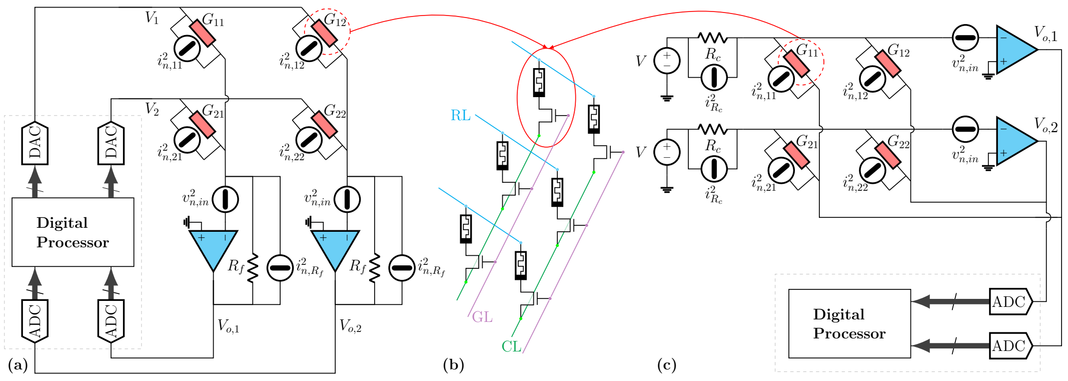


Fig. 4.1 (a) Illustration of a 2×2 open-loop crossbar used to perform matrix-vector multiplication. All thermal noise sources associated with TIAs (formed by an op-amp and a feedback resistor) and crossbar 1T1M cells are shown. (b) The one-transistor one-memristor (1T1M) cell at each junction in a crossbar. GL: Gate Line, RL: Row Line, CL: Column Line. (c) Illustration of a 2×2 feedback crossbar for solving linear equations. The noise sources are input resistors and TIAs (formed by an op-amp and crossbar cells). Taken from [2] ©2021 IEEE.

to the calculation of MCs' stationary distribution. In the years, numerous viable solutions to this mathematical problem have emerged [80]. To cite those that are applied in this study, the MC stationary distribution can be obtained either by an iterative process or by solving an eigenvector problem cast as a linear system solution problem.

Other approaches include, to name a few, LU decomposition and Gaussian Elimination which are the de facto standard algorithms for solving linear equations when the matrix is small and dense. However when this last requirement is not satisfied, iterative methods such as Gauss-Seidel and Power Method algorithms are preferred for determining the stationary distribution of discrete MCs. Nonetheless, time complexity for this category of algorithms is polynomial with respect to the matrix size [81].

This Section introduces discrete-time MCs by providing the required mathematical background needed to well understand the following of the Chapter. Curious readers that want to know more about Markov Chains can refer to [82].

A discrete random variable with finite state X is a measurable function $X : \Omega \to \Gamma$ where Ω is the set of admissible outcomes and $\Gamma = \{1, \ldots, m\}$. A discrete time stochastic process with finite state is a sequence $\{X_k : k \in \mathbb{N}\}$ of discrete random variables with finite state.

Definition 1. *A first-order MC with finite state is a discrete time stochastic process that for all $k \in \mathbb{N}$ satisfies the Markov property:*

$$p(X_{k+1} = x_{k+1} | X_k = x_k, \dots, X_0 = x_0) = p(X_{k+1} = x_{k+1} | X_k = x_k). \tag{4.1}$$

A first-order MC is (time) homogeneous if for all $k, h \in \mathbb{N}$:

$$p(X_{k+1} = x_{k+1} | X_k = x_k) = p(X_{h+1} = x_{h+1} | X_h = x_h). \tag{4.2}$$

Henceforward, the studied MCs are assumed to be homogeneous and move between a finite number of states. This guarantees that the transition matrix $\mathbf{M}$ can represent in a compact form all the transition probabilities:

$$M_{ij} = p(X_{k+1} = j | X_k = i). \tag{4.3}$$

where $\mathbf{M}$ is a $m \times m$ non-negative probability matrix whose column elements sum to 1. The matrix size clearly corresponds to the number of states m of the MC. To find the probability distribution $\mathbf{p}_k = [p(X_k = 1), \dots, p(X_k = m)]^T$ at time k from the distribution at previous time $k-1$, the following MVM has to be performed:

$$\mathbf{p}_k = \mathbf{M}\mathbf{p}_{k-1} \tag{4.4}$$

From Eq. 4.4, it can be readily found that:

$$\mathbf{p}_k = \mathbf{M}^k \mathbf{p}_0 \tag{4.5}$$

where $\mathbf{M}^k = \prod_1^k M$ and $\mathbf{p}_0$ is the initial distribution.

Definition 2. *A finite MC with transition matrix $\mathbf{M}$ is defined irreducible if for all pairs of states (i, j), at least one $r = r(i, j) \in \mathbb{N}$ exists such that $M_{ij}^r > 0$.*

Irreducibility ensures that all the states are reachable from any other state. MCs with transition matrix $\mathbf{M}$ are said to admit a stationary distribution $\mathbf{p}_\infty = [p_\infty^1, \dots, p_\infty^m]^T$ when the following condition is satisfied:

$$\mathbf{p}_\infty = \mathbf{M}\mathbf{p}_\infty, \quad \sum_{i=1}^{m} p_\infty^i = 1. \tag{4.6}$$

Eq. 4.6 can be thought as if the chain by chance starts in the stationary distribution, it indefinitely remains in that distribution. More than one stationary distribution may be admissible, all corresponding to the eigenvectors of the transition matrix eigenvalue 1. Nonetheless, if the chain is irreducible, the singularity of the stationary distribution $\mathbf{p}_\infty$ is ensured and for every state i, p^i_∞ expresses the mean return time to that particular state. Another way of looking at the component of the stationary distribution p^i_∞ is to think at component as the proportion of time spent by the system in a given state. Unless the MC is also regular, its convergence towards its stationary distribution is not guaranteed for all the initial distributions. Hence, the concept of MC regularity has to be introduced.

Definition 3. *A matrix $\boldsymbol{M}$ is defined primitive if one value $r \in \mathbb{N}$ exists such that $M^r_{ij} > 0$ for all pairs of states (i, j). A finite MC with a primitive transition matrix $\boldsymbol{M}$ is said to be regular.*

It is arguable that regular a MC is always irreducible, but actually the inverse assertion needs a further constraint:

Definition 4. *The period d_i of the state i is given by $d_i = gcd\{r \geq 1 : M^r_{ii} > 0\}$ with gcd denoting the greatest common divisor. A MC state i is said to be aperiodic if $d_i = 1$. The finite MC and its transition matrix $\boldsymbol{M}$ are called aperiodic when all states are aperiodic.*

Proposition 2. *a) Irreducible and aperiodic finite MC $\Leftrightarrow$ Regular MC;*

b) Irreducible finite MC and there exists $i \in \Gamma$ such that $M_{ii} > 0 \Rightarrow$ Regular MC;

c) Regular MC $\Rightarrow \lim_{n\to\infty} p(X_n = i | X_0 = j) = p^i_\infty \; \forall j$.

Reader interested in further details on the proofs can read [82]. In all the cases when a limiting distribution exists, the chain asymptotically evolves towards it independently of the initial condition. Each component i of the limiting probability distribution vector can be interpreted either as the long-run probability of finding the system in the state i or the proportion of time spent by the chain in a that state.

4.3 Solving Discrete-Time Markov Chains

Given an initial probability distribution $\mathbf{p}_0$, the evolution of a finite MC can computed by repeatedly iterating Eq. (4.4). The next most probable state of the chain can always be forecasted by computing the MVM between the transition matrix $\mathbf{M}$ and the current probability distribution vector $\mathbf{p}_k$. In order to make clear the process, in Figure 4.2 the evolution of the distribution vector is shown for a generic regular MC over different iterations. It should be noted how, after a certain number of transitions, the discrete state probability vector asymptotically converges to the stationary distribution.

From Eq. (4.6), the stationary distribution can be interpreted as the solution of either an eigenvector problem or of a linear system. As aforementioned there are two classes of algorithms commonly employed to find it: direct and iterative methods [80]. The first class mostly needs the entire coefficient matrix stored in memory and applications are usually constrained by memory capacity and computing power (e.g. Gaussian Elimination, LU decomposition, GTH-algorithm, etc). When the problem is large, the second class of algorithms is the only viable way (e.g. Gauss-Seidel, Power Method, etc). In both cases the time complexity is polynomial with respect to the matrix size. For additional details on space and time complexity the interested reader may refer to [81].

Historically, the MC stationary distribution search has been treated as an eigenvector problem which could be solved by iteratively applying the Power Method. This algorithm starts with an initial distribution vector $\mathbf{p}_0$, which might also be random. At each next iteration $k + 1$, it computes a MVM between the transition matrix $\mathbf{M}$ and the current distribution vector $\mathbf{p}_k$ until the stop condition is reached.

Very recently, a novel closed-loop memristor crossbar design was proposed which can find the dominant eigenvector of a matrix in a single time step [83]. On the one hand this novel approach avoids the numerous MVMs required by Power Method, on the other hand it makes impossible to compute the k-th step forecast which not rarely is required in certain MC applications.

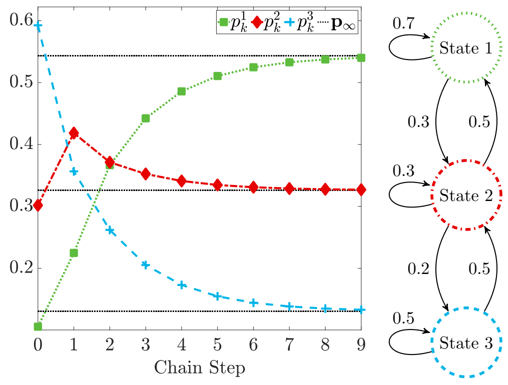


Fig. 4.2 Graphical representation of a generic regular three-state MC together with Power Method iterations. Black dashed lines correspond to the limiting distribution. Taken from [2] ©2021 IEEE.

The transition matrix irreducibility ensures that there is one stationary distribution only. Nonetheless, the convergence of both the proposed implementations is determined by the primitivity of $\mathbf{M}$. If the stochastic matrix $\mathbf{M}$ is not primitive, then it might have more than one eigenvalue on the unitary circle and this may arise convergence problems. In particular, it may happen that the far future probabilities of the chain greatly depend on the initial condition $\mathbf{p}_0$. Conversely, primitive matrices have only one eigenvalue on the unit circle with all the others strictly less than 1 in modulus. Thus, primitive matrices guarantee the convergence of the probability vector to the unique dominant eigenvector, e.g. starting from any probability vector, the Power Method is guaranteed to converge. When the matrix is not primitive, it is still possible to consider a modified transition matrix $\mathbf{Q} = \alpha\mathbb{I}_m + (1-\alpha)\mathbf{M}$ where $\mathbb{I}_m$ is the $m \times m$ identity matrix and $0 < \alpha < 1$. If $\mathbf{p}_\infty = \mathbf{M}\mathbf{p}_\infty$ then one can observe that $\mathbf{Q}\mathbf{p}_\infty = [\alpha\mathbb{I}_m + (1-\alpha)\mathbf{M}]\mathbf{p}_\infty = \mathbf{p}_\infty$. Hence, the two MCs whose transition matrices are $\mathbf{M}$ and $\mathbf{Q}$ respectively share the same stationary distribution. Still, the MC with $\mathbf{Q}$ as transition matrix is a regular MC (see Prop. 2(b)) and thus has a unique limiting distribution and the choice of α is the only factor to influence the results' accuracy. An alternative to the iterative method is the direct computation of the eigenvector $\mathbf{p}_\infty$ by solving an associated linear system as shown in the following proposition.

Proposition 3. *Let $\boldsymbol{M}$ be the transition matrix of a finite irreducible MC with stationary distribution $\boldsymbol{p}_\infty$.*

Let $\boldsymbol{A} = (\mathbb{I}_m - \boldsymbol{M} + \mathbf{1}_{m\times m})$ where $\mathbf{1}_{m\times m}$ is the matrix with all entries equal to 1 *and $\mathbf{1}_m$ is the column vector of ones. Then:*

a) $\boldsymbol{A}$ is non-singular and $A^{-1}_{ii} > 0\ \forall i = 1,\dots,m$;

b) $\boldsymbol{p}_\infty$ is the unique solution of the system

$$\boldsymbol{A}\boldsymbol{x} = \mathbf{1}_m. \tag{4.7}$$

Proof. a) For details see [84, 85].
b) The objective is to find $\mathbf{p}_\infty$ such that $\mathbf{M}\mathbf{p}_\infty = \mathbf{p}_\infty$, subject to $\sum_{i=1}^m p^i_\infty = 1$. This constraint can be written as

$$\mathbf{1}_{m\times m}\mathbf{p}_\infty = \mathbf{1}_m. \tag{4.8}$$

By summing $\mathbf{p}_\infty - \mathbf{M}\mathbf{p}_\infty = 0$ to (4.8) and by collecting $\mathbf{p}_\infty$, the claim is obtained. For a), the irreducibility assumption ensures the non-singularity of the matrix, thus the uniqueness of the solution. ■

To conclude, (4.6) and (4.7) have in common the same solution. Nevertheless, the direct calculation of the linear system in (4.7) guarantees no convergence issues which could occur with the iterative methods.

4.4 Crossbar Implementations

The transition matrix **M** can be mapped within a precision of choice onto a memristor crossbar array by programming each memductance G_{ij} inside the correspondent 1T1M cell as shown in Fig. 4.1(b). As explained in this Chapter introduction, in order to perform a MVM by means of a (mem)resistive crossbar array, a vector of voltages has to be applied to the rows of the open-loop crossbar as shown in Fig. 4.1(a) and then the multiplication happens thanks to the Ohm's Law at each intersection between a bit-line and a word-line where a memristor exists. All the resulting currents are summed at each column by means of the Kirchhoff's Current Law. Each current flowing out of its column in a single time step is physically measured using a transimpedance amplifier (TIA) and taken as the correspondent component of the vector resulting from the MVM operation. The iterative approach which performs a MVM operation at each time step enables both the *k*-th MC state prediction and if required even the recursive update up to convergence as in the Power Method. When the MC is regular eventually, after many iterations, the output probability distribution vector will settle to the MC stationary distribution. It should be mentioned that during the whole computation the transition matrix must not change. In order to ensure that, the voltage components of the input vector must be constrained within the operating voltage range that avoids conductance drift.

Differently from the open-loop crossbar solution, the feedback architecture shown in Fig. 4.1(c) can solve systems of linear equations [77] such as

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{4.9}$$

in which $\mathbf{x}$ is the unknown vector while $\mathbf{A}$ and $\mathbf{b}$ are the known matrix and the vector respectively. In this promising architecture, each TIA composed of an op-amp and the memristive crossbar itself as feedback serve two purposes. Exactly as in the open-loop architecture, the TIAs enable the conversion from current to voltage. Conversely from the former architecture, in the feedback solution the created virtual reference nodes serve the purpose of letting the input current $\mathbf{I}$ divide over the entire crossbar along feedback path. Due to Kirchhoff's Current Law, the following equation holds at the outputs of the TIAs':

$$\mathbf{I} + \mathbf{G}\mathbf{V}_o = 0 \tag{4.10}$$

with $\mathbf{G}$ being the memductance matrix of the crossbar.

As shown in Fig. 4.1(c), the resistors R_c serve the purpose of converting input voltages to currents which are provided as input vector $\mathbf{I} = -\mathbf{b}$. In a single time step, Eq. (4.9) is solved and the solution is the output voltage $\mathbf{V}_o$:

$$\mathbf{x} = \mathbf{V}_o = -\mathbf{G}^{-1}\mathbf{I} \tag{4.11}$$

The convenience of employing a crossbar to solve the linear equation $\mathbf{A}\mathbf{x} = \mathbf{b}$ is that the calculation of $\mathbf{A}^{-1}$ is not needed which drastically reduces the computational complexity.

Architecture	**Computation**	**Preliminary Transformation**	**Recovering Transformation**	**Number of Steps**
Open-Loop Crossbar	Next State Prediction $\mathbf{p}_k = \mathbf{M}\mathbf{p}_{k-1}$	$\mathbf{G}_{map} = \gamma\mathbf{M} + \delta\mathbf{1}_{m\times m}$	$\mathbf{y} = \frac{1}{\gamma}(\tilde{\mathbf{y}} - \delta\mathbf{1}_m)$	One Time-Step Solution
	Power Method $\mathbf{p}_\infty = \mathbf{M}\mathbf{p}_\infty$	$\mathbf{G}_{map} = \gamma\mathbf{M} + \delta\mathbf{1}_{m\times m}$	$\mathbf{y} = \frac{1}{\gamma}(\tilde{\mathbf{y}} - \delta\mathbf{1}_m)$	Iterations until Stopping Criteria
Feedback Crossbar	Linear System $\mathbf{p}_\infty = \mathbf{M}\mathbf{p}_\infty$	$\mathbf{A} = \mathbb{I}_m - \mathbf{M} + \mathbf{1}_{m\times m}$ $\mathbf{G}_{map} = \gamma\mathbf{A} + \delta\mathbf{1}_{m\times m}$	-	One Time-Step Solution

Fig. 4.3 Summary of the architectures and operations in this work. The open-loop crossbar enables the computation of both the next state distribution via an MVM operation and the stationary distribution of the chain by the sequential update provided by the Power Method. The requirements are linear transformations to set the conductance range and recover the correct result. When the evolution of the system is not required, the stationary distribution of the MC can be computed in a single step with the feedback crossbar configuration. After a preliminary linear transformation of the system, the routine does not need a recovering step. Taken from [2] ©2021 IEEE.

Equation (4.9) has the same form as (4.7), where $\mathbf{x} = \mathbf{p}_\infty$, $\mathbf{A} = (\mathbb{I}_m - \mathbf{M} + \mathbf{1}_{m\times m})$ and $\mathbf{b} = \mathbf{1}_m$. Hence, memristive crossbar arrays can perform the computation of the MCs' stationary distribution vector without iterating. Furthermore, a key feature of this system is that the vector $\mathbf{b}$ in (4.7) has all unitary entries, e.g. $\mathbf{1}_m$. Therefore, *the Markov Chain stationary distribution is obtainable by means of the same input current for each row in the crossbar.* This fact substantially lowers the complexity of the peripheral circuitry in the feedback crossbar architecture with respect to the open-loop counterpart by making Digital-to-Analog Converters (DACs) unnecessary, thus lowering power consumption [86, 87]. Matrix $\mathbf{A}$ in the form $(\mathbb{I}_m - \mathbf{M} + \mathbf{1}_{m\times m})$ is mappable onto the crossbar with each conductance within the technology imposed range of memductances. The resulting stationary distribution vector $\mathbf{p}_\infty$ can be measured as the TIAs' output.

Due to the always limited conductance range offered by any memristive technology, computing either a MVM or the solution of a linear system may be challenging with a crossbar array. The entries of the matrix $\mathbf{A}$ require to be mapped into this range using the following linear transformation [45, 47]:

$$\begin{aligned}\gamma &= \frac{G_{on} - G_{off}}{A_{max} - A_{min}} \\ \delta &= G_{on} - \gamma A_{max} \\ \mathbf{G}_{map} &= \gamma \mathbf{A} + \delta \mathbf{1}_{m\times m}\end{aligned} \tag{4.12}$$

where G_{on} and G_{off} correspond to the highest and the lowest memductances reachable, respectively. A_{max} and A_{min} are to the highest and lowest elements in the matrix $\mathbf{A}$. $\mathbf{G}_{map}$ is the memductance matrix programmed into the crossbar and henceforth, for writing convenience, $\mathbf{G} = \mathbf{G}_{map}$. The required linear transformation can be preliminary computed by means of a digital system.

After the output vector from the open-loop crossbar architecture depicted in Fig. 4.1(a) gets measured, the correct result can be obtained using the inverse operations that follows:

Proposition 4. *Let $\boldsymbol{M}$ be the transition matrix of a finite MC. If $\boldsymbol{I} = \boldsymbol{MV}$ and $\tilde{\boldsymbol{I}} = \boldsymbol{GV}$ then $\boldsymbol{I} = \frac{1}{\gamma}(\tilde{\boldsymbol{I}} - \delta \mathbf{1}_m)$.*

Proof. Consider the linear operation $\mathbf{I} = \mathbf{MV}$. By multiplying both sides by γ and then adding vector $\delta \mathbf{1}_{m\times m}\mathbf{V}$ one obtains:

$$\begin{aligned}\gamma \mathbf{I} + \delta \mathbf{1}_{m\times m}\mathbf{V} &= \gamma \mathbf{MV} + \delta \mathbf{1}_{m\times m}\mathbf{V} \\ \gamma \mathbf{I} + \delta \mathbf{1}_m &= \mathbf{GV}\end{aligned} \tag{4.13}$$

by using $\mathbf{1}_{m\times m}\mathbf{V} = \mathbf{1}_m$ and $\mathbf{A} = \mathbf{M}$ in (4.12). Let $\tilde{\mathbf{I}} = \gamma \mathbf{I} + \delta \mathbf{1}_m$ then $\mathbf{I} = \frac{1}{\gamma}(\tilde{\mathbf{I}} - \delta \mathbf{1}_m)$. ∎

Hence, once the analog to digital conversion is completed, the adimensional MVM result can be recovered on a digital system by means of Prop. 4. The closed form relation linking the measured and the adimensional outputs cannot be easily generalized to the linear systems case. Nonetheless, while computing the stationary distribution via the crossbar in Fig. 4.1(c), Markov Chains are demonstrated to meet the circuit requirements thanks to the following proposition:

Proposition 5. *Let $\boldsymbol{A} = (\mathbb{I}_m - \boldsymbol{M} + \mathbf{1}_{m\times m})$ where $\boldsymbol{M}$ is the transition matrix of an irreducible finite MC. The two following systems are equivalent:*

$$\boldsymbol{AV} = \mathbf{1}_m, \qquad \boldsymbol{GV} = (\gamma + \delta)\mathbf{1}_m.$$

Proof. Consider the linear system $\mathbf{AV} = \mathbf{1}_m$. By multiplying both sides by γ and then adding the vector $\delta\mathbf{1}_{m\times m}\mathbf{V}$, one obtains the equality:

$$\gamma\mathbf{AV} + \delta\mathbf{1}_{m\times m}\mathbf{V} = \gamma\mathbf{1}_m + \delta\mathbf{1}_{m\times m}\mathbf{V} \tag{4.14}$$

By collecting $\mathbf{V}$ and observing that $\mathbf{1}_{m\times m}\mathbf{V} = \mathbf{1}_m$, one gets $\mathbf{GV} = (\gamma + \delta)\mathbf{1}_m$. ∎

This solution bypasses the need of a recovering step to find the adimensional solution of the linear system (4.7). Furthermore, two different ways exist to compute the output:

- When $(\gamma + \delta)\mathbf{1}_m$ is in input, the architecture will output the normalized eigenvector;
- When $\mathbf{1}_m$ is in input, the architecture will output a vector which can be transformed into the solution by means of a normalization.

Depending on the selected of memristive technology and the required matrix **A**, there is a high risk of the $(\gamma + \delta)$ term in the first option being a number so small that the circuit could not be physically designed. On the other hand, the second option is free from that risk for any technology and matrix **A** and thus is the chosen approach in this dissertation. It is worth highlighting that both configurations need ADCs to convert the crossbar TIAs output and perform post-processing operations in the digital domain.

In summary, *the open-loop crossbar architecture in Fig. 4.1(a) is employable either to find the stationary distribution of a regular MC by a series of sequential updates via the Power Method or to forecast the distribution vector at the next time step via a simple MVM operation. In the particular case when the evolution of the system is not relevant, a MC's stationary distribution can be computed in a single step using the feedback crossbar architecture in Fig. 4.1(c).* In Figure 4.3 the two structures/approaches are compared for MC applications.

In physical ICs, each circuit component introduces random fluctuations and systematic errors:

- random noise limits the system precision,
- the systematic errors degrade the output accuracy.

Recently, different metrics have been adopted to estimate the output precision and accuracy of memristive crossbars, such as cosine similarity [83], relative error [88] and bit-precision [47]. The two following Sections provide a novel analysis of both random and systematic errors. Application-agnostic mathematical tools are introduced to systematically estimate precision and accuracy in the two architectures from Fig. 4.1(a) and (c).

4.5 System Precision Analysis

In this Section, the two architectures are studied with a focus on thermal and programming noise sources. A signal-to-noise ratio (SNR) metric will be employed to estimate the output precision in terms of bits and an accurate comparison between memristive crossbars and their analog and digital counterparts is provided. From this point forward, $\mathbf{G} \in \mathbb{R}^{m\times m}$ and $\mathbf{V}, \mathbf{I} \in \mathbb{R}^{m}$ such that

$$\mathbf{G}\mathbf{V} = \mathbf{I}. \tag{4.15}$$

4.5.1 Open-Loop Crossbar

The open-loop crossbar configuration in Fig. 4.1(a) is the first to be investigated. The noise sources affecting the output precision are two: programming and thermal noises. Memristors in the crossbar give a contribution to both noise sources, while feedback resistors and op-amps contribute to the thermal noise only. Flicker (1/f) noise generated by transistors and op-amps is neglected because the proposed system is operated in a wide bandwidth configuration and the contribution of the thermal noise overwhelms the (1/f) noise.

Thermal Noise

In Figure 4.1(a), the thermal noise sources in the open-loop architecture are shown. The thermal noise coming from transistors in the 1T1M structures can be neglected by means of proper sizing [89]. The mean square current noise $\overline{i^2_{nT,jk}}$ generated in a single memristor at the j-th row and k-th column is given by:

$$\overline{i^2_{nT,jk}} = 4kTBW_nG_{jk}, \tag{4.16}$$

with BW_n being the noise bandwidth. Thermal noise is modeled by means of a zero mean Gaussian distribution with variance σ_T^2, in which $\sigma_T^2 = \overline{i^2_{nT}}$ [90]. As all memristors' contributions are independent, the total current noise variance at the input of the k-th TIA is:

$$\overline{i^2_{nT,k}} = \sum_{j=1}^{m} \overline{i^2_{nT,jk}}. \tag{4.17}$$

Then, the total voltage noise variance at the output of the k-th TIA is:

$$\overline{v^2_{nT,k}} = \overline{i^2_{nT,k}}R_f^2 + \overline{v^2_{nT,R_f}} \tag{4.18}$$

with $\overline{v^2_{nT,R_f}}$ being the feedback resistor voltage noise variance. The thermal noise contribution generated by the op-amp is be discussed in the feedback crossbar Subsection.

Programming Noise

Programming noise is the other effect which degrades the overall system precision. Memristor are commonly embedded in the 1T1M structure (Fig. 4.1(b)) which enables their programming by dynamically driving the transistor gate with appropriate voltage levels which in turn control the amount of current flowing in the memristor [91]. By means of this structure it is also possible to be interdict unused cells in the crossbar [92]. For instance, a $m \times m$ crossbar can perform $k \times k$ operations ($m > k$) by interdicting the unused cells transistors. In this Chapter, tantalum-oxide memristors [93, 94] with a $100\mu S - 1000\mu S$ conductance range are chosen as a possible technology reference. In the selection of the appropriate conductance range, the trade-off to consider is between low and high conductance levels. It is known from previous studies[45, 83] that higher conductances enable precise programming, while, lower values offer improved power efficiency. Memristors can be programmed to an arbitrary conductance within a range $G_{range} = [G_{off}, G_{on}]$, while accepting a chosen tolerance [95, 96] which is function of an iterative read-write programming scheme. This procedure for writing a value into memristors enables improved precision at the expense of more iterations and in turn higher energy expenditure. With the aim of comparing the precision of the proposed analog accelerator to an equivalent digital implementations, G_{range} is subdivided into 2^{N_b} subranges (levels), with N_b being the equivalent number of bits. To the best of the author's knowledge, the highest experimentally measured memristor bit precision reported in literature at the time of writing is of 8 bits (256 memductance levels)[97].

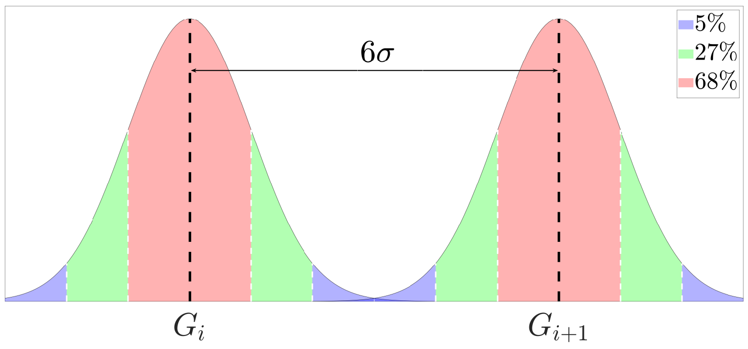


Fig. 4.4 Normally distributed programming noise model that illustrates the intended write distribution between two conductance levels separated by the pre-defined bit precision. Taken from [2] ©2021 IEEE.

It is well known that a same device programmed to a same target conductance level cannot be exactly set to that desired value. From programming cycle to programming cycle the actual written level fluctuates. These cycle-to-cycle fluctuations are statistically Gaussian distributed. When the application requires very frequent memristors reprogramming, this phenomenon affects precision and in the following of the text will be referred to as programming noise. However this is not always the case as many applications need the conductance matrix to stay constant and be repeatedly used for iterative calculations. In this latter case, after the first programming phase, the error remains constant until the next programming stage. In this case, the error becomes a systematic level shift in the mean conductance and thus only affects the output accuracy. This phenomenon affecting the computation accuracy will be called programming error and analyzed in the next Section.

It is common in literature for the programming noise to be modeled by a normal distribution with a standard deviation of σ [45, 47, 46, 83]. In this manuscript, in order to analyze the programming noise contribution on output precision, a normally distributed perturbation $\sim N(0,\sigma^2)$ is added to each ideal memristor conductance level. In order to have those levels distinguishable with high degree of confidence, as in Fig. 4.4, each interval of width $\Delta L = \frac{G_{range}}{(2^{N_b}-1)}$ ought to contain at least $\pm 3\sigma$. With this aim in mind, for the rest of this analysis it is set $\sigma = \Delta L/6$.

Consider a single MVM operation where **I** is the output vector while **G** and **V** are the known variables. Suppose to perturb the matrix **G** by adding a disturbance sampled from a Gaussian distribution with zero mean and variance σ^2:

$$\tilde{\mathbf{G}} = \mathbf{G} + \mathbf{Z}, \quad Z_{ij} \sim N(0,\sigma^2). \tag{4.19}$$

Then, system (4.15) becomes:

$$\tilde{\mathbf{I}} = \tilde{\mathbf{G}}\mathbf{V} = (\mathbf{G}+\mathbf{Z})\mathbf{V} = \mathbf{G}\mathbf{V} + \mathbf{Z}\mathbf{V} = \mathbf{I} + \mathbf{Z}\mathbf{V} \tag{4.20}$$

with $\tilde{\mathbf{I}}$ being the perturbed output. The k-th component of Eq. (4.20) is a linear combination of independent Gaussian random variables and therefore the following holds [98]:

$$\tilde{I}_k - I_k \sim N(0, \|\mathbf{V}\|_2^2\sigma^2) \quad \forall k \tag{4.21}$$

with $\|\cdot\|_2$ being the euclidean norm. In practice, if all the entries of a matrix are randomly perturbed by disturbances sampled from a Gaussian distribution $N(0,\sigma^2)$, then the k-th output current noise will be normally distributed $N(0,\sigma_p^2)$ where $\sigma_p^2 = \|\mathbf{V}\|_2^2\sigma^2$. From σ_p^2 output variance expressed in terms of current noise it is readily derived the output voltage noise by multiplying the former one times the square of the feedback resistance: $\overline{v_{np,k}^2} = \sigma_p^2 R_f^2$. This turns to be very helpful in the SNR analysis at the output in order to compare the contributions of programming noise and thermal noise.

4.5.2 Feedback Crossbar

Differently from the open-loop structure in Fig. 4.1(a) which uses voltage inputs, The feedback crossbar depicted in Fig. 4.1(c) uses current inputs and thus current noise variances are needed in order to find the output distribution. Furthermore, the random fluctuations' amplitude greatly changes with the conductance matrix itself. Consider the linear system $(\mathbf{G}+\mathbf{Z})(\mathbf{V}+\epsilon) = (\mathbf{I}+\eta)$ and suppose that $\|\mathbf{Z}\|\|\mathbf{G}^{-1}\| < 1$ for an arbitrary chosen induced matrix norm, it follows that:

$$\frac{\|\epsilon\|}{\|\mathbf{V}\|} \leq \frac{K(\mathbf{G})}{1-\|\mathbf{Z}\|\|\mathbf{G}^{-1}\|}\left(\frac{\|\mathbf{Z}\|}{\|\mathbf{G}\|}+\frac{\|\eta\|}{\|\mathbf{I}\|}\right) \tag{4.22}$$

with $K(\mathbf{G}) = \|\mathbf{G}\|\|\mathbf{G}^{-\mathbf{1}}\|$ being the condition number [81]. From Eq. (4.22) follows that when the conductance matrix $\mathbf{G}$ is not hill-conditioned (i.e. $K(\mathbf{G}) \approx 1$), small perturbations applied to $\mathbf{G}$ and $\mathbf{I}$ result in small variations of the solution $\mathbf{V}$. When instead $\mathbf{G}$ is ill-conditioned (i.e. $K(\mathbf{G}) \gg 1$), small perturbations applied to $\mathbf{G}$ and $\mathbf{I}$ might result in large variations of the solution $\mathbf{V}$. In conclusion, it can be stated that the system precision heavily depends on the nature of the application as it will be shown in the following.

Thermal Noise

Consider Eq. (4.15) where $\mathbf{V}$ is the unknown voltage vector while $\mathbf{I}$ and $\mathbf{G}$ are the input current vector and the conductance matrix respectively. By solving this linear system via the feedback crossbar, Fig. 4.1(c) makes clear how all the memristors in the feedback path and all input resistors generate their own mean square current noises $\overline{i_{nT}^2}$, and thus the total current noise variance generate by the j-th row of the feedback crossbar is:

$$\sigma_{T_j,F}^2 = 4kTBW_n\left(\sum_{k=1}^{m} G_{jk} + \frac{1}{R_c}\right) \tag{4.23}$$

with R_c being the input resistor. The feedback output noise analytical formulation can be obtained by summing to the input current vector $\mathbf{I}$ a disturbance vector $\boldsymbol{\jmath}$ whose components $\eta_j \sim N(0,\sigma^2_{T_j,F})\ \forall j$:

$$\mathbf{G}\tilde{\mathbf{V}} = \tilde{\mathbf{I}} = \mathbf{I} + \boldsymbol{\jmath} \Rightarrow \tilde{\mathbf{V}} = \mathbf{V} + \mathbf{G}^{-1}\boldsymbol{\jmath} \tag{4.24}$$

with $\tilde{\mathbf{I}}$, $\tilde{\mathbf{V}}$ being the perturbed input and output of the system in Eq. (4.15) respectively, and $\mathbf{V} = \mathbf{G}^{-1}\mathbf{I}$. The j-th component of Eq. (4.24) is a linear combination of independent Gaussian random variables resulting in [98]:

$$\tilde{V}_j - V_j \sim N\left(0, \sum_{k=1}^{m}(G^{-1}_{jk})^2\sigma^2_{T_k,F}\right) \quad \forall j. \tag{4.25}$$

Input resistors and memristors are accountable for the most of the voltage noise variance at the output of the j-th TIA which is:

$$\overline{v^2_{nT,j}} = \sum_{k=1}^{m}(G^{-1}_{jk})^2\sigma^2_{T_k,F}. \tag{4.26}$$

The remaining contribution to thermal noise in crossbars comes from the CMOS transistors which make up operational amplifiers. In order to accurately estimate this source of noise, the voltage noise variance of a transistor at the output of an amplification stage with gain A_o is:

$$\overline{v^2_{n,tr}} = \frac{4kT\Gamma BW_n A_o^2}{g_m}. \tag{4.27}$$

with g_m being the transconductance value of each stage. The dynamic range D, at each stage output is:

$$D = \frac{\overline{V_o^2}}{\overline{v^2_{n,tr}}} = \frac{(V_o^2)g_m}{4kT\Gamma BWA_o^2} \tag{4.28}$$

with V_o being the output voltage. Therefore, g_m for each stage can be estimated as:

$$g_m = \frac{4kT\Gamma BW_n A_o^2 D}{(V_o^2)}. \tag{4.29}$$

Note that the g_m values from Eq. (4.29) are a conservative estimation. The total output voltage noise variance of a two-stage operational amplifier can be computed using these transconductance values as [99]:

$$\overline{v_{n,o}^2} = 4kT\Gamma BW_n R_L^2 g_{m_2}(g_{m_1} g_{m_2} R_{o_1}^2 + 1) \tag{4.30}$$

with g_{m_1} and g_{m_2} being the transconductance of the first and second amplification stages respectively, R_{o_1} being the output resistance of the first stage and R_L the load resistance. This latter coincides with the parallel resistances of all the memristors along the feedback path. The first and second terms in Eq. 4.30 correspond to the first and second amplification stages voltage noise variances respectively. It can be noted how a lower memristor conductance yields a higher output voltage noise.

Negative feedback can be employed in order to improve the noise performance with a degree proportional to the closed-loop gain of the system. The input-referred noise of the operational amplifier, $\overline{v_{n,in}^2} = \frac{\overline{v_{n,o}^2}}{A_o^2}$, can be multiplied by the noise gain in order to compute the total output voltage noise variance in a feedback crossbar system:

$$\overline{v_{n,amp}^2} = \frac{1}{\beta^2}\overline{v_{n,in}^2}. \tag{4.31}$$

Programming Noise

Conductance matrix $\mathbf{G}$ and input current vector $\mathbf{I}$ are the known in the linear system of Eq. (4.15) while $\mathbf{V}$ is the unknown vector. By perturbing the linear system (4.15) as in Eq. (4.19) the following conclusions can be drawn. Given a matrix norm of choice, in case that the perturbation is sufficiently small $\|\mathbf{G}^{-1}\mathbf{Z}\| < 1$, if follows that $(\mathbf{G}+\mathbf{Z})^{-1}$ is non-singular. Furthermore, the Neumann expansion's linear part of $(\mathbf{G}+\mathbf{Z})^{-1}$ is $(\mathbf{G}+\mathbf{Z})^{-1} \approx \mathbf{G}^{-1} - \mathbf{G}^{-1}\mathbf{Z}\mathbf{G}^{-1}$. Thanks to the introduced approximation, the analytical computation is made practicable:

$$\tilde{\mathbf{V}} = (\mathbf{G}+\mathbf{Z})^{-1}\mathbf{I} \approx \mathbf{V} - \mathbf{G}^{-1}\mathbf{Z}\mathbf{V} \tag{4.32}$$

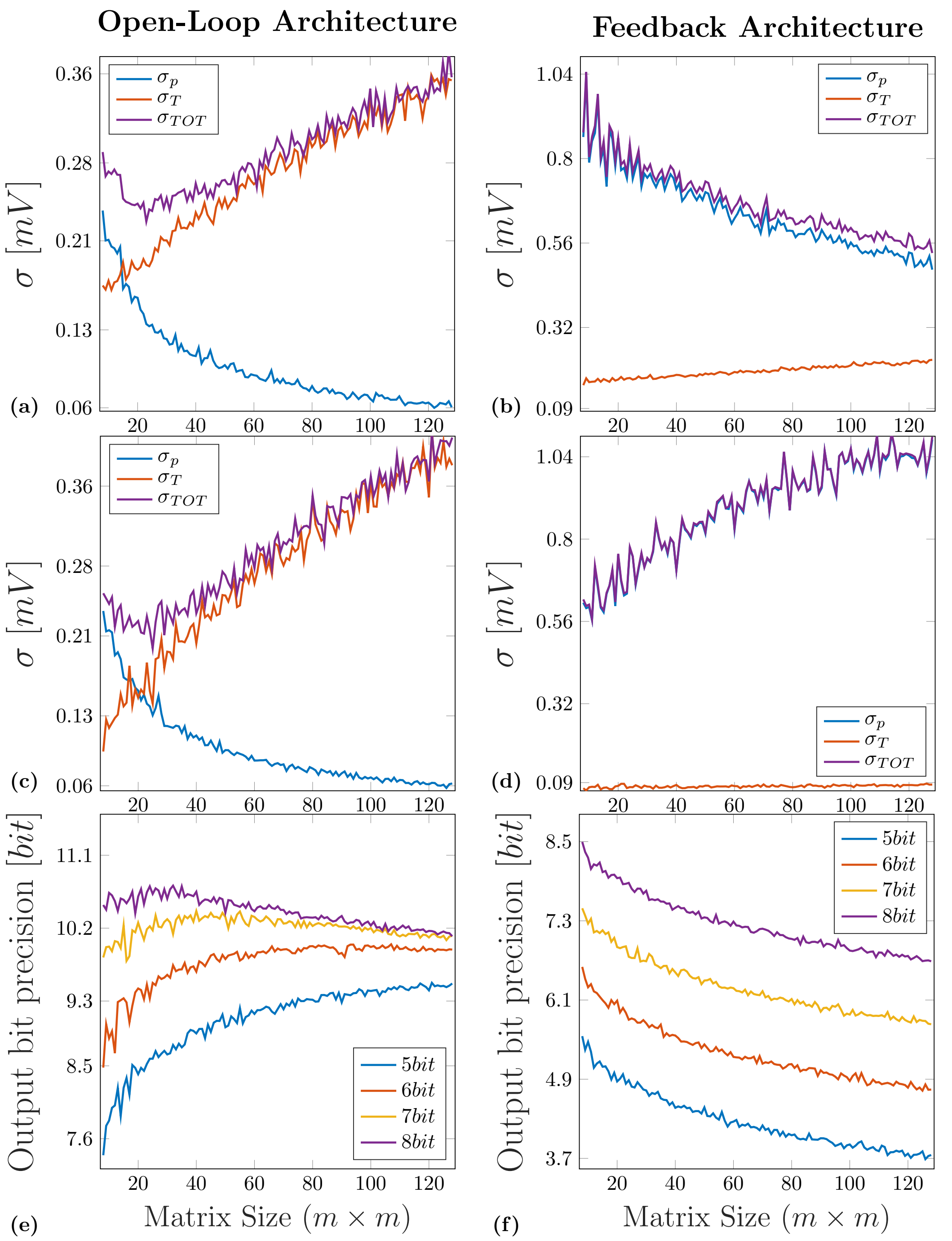


Fig. 4.5 Simulated programming σ_p, thermal σ_T and total σ_{TOT} noise of a generic output component for different crossbar sizes using both (a) open-loop and (b) feedback architectures and 8 bits precision memristors with a conductance range of $[100\mu S, 1000\mu S]$. (c) and (d) replicate the first row plots with a conductance range of $[10\mu S, 1000\mu S]$. (e) and (f) are the equivalent output bit precision for different matrix sizes using open-loop and feedback crossbars, respectively. The total noise contribution of different memristor precision was considered and compared using MATLAB. The σ_p estimates were computed by using 1000 perturbations of a random transition matrix. The same operation was repeated for σ_T by perturbing the input vector. The total noise σ_{TOT} was calculated by perturbing both transition matrix and input vector. These estimates were then used to evaluate the output bit precision. To obtain the relationship between precision and noise, systematic errors were excluded. The σ_T includes thermal noise contributions of memristors, resistors and op-amps. Taken from [2] ©2021 IEEE.

with $\mathbf{V} = \mathbf{G}^{-1}\mathbf{I}$. Due to the linearity of Gaussian random variables combination, each j-th component of Eq. (4.32) becomes [98]:

$$\tilde{V}_j - V_j \sim N\left(0, \|G_{j\cdot}^{-1}\|_2^2 \|\mathbf{V}\|_2^2 \sigma^2\right) \quad \forall j \tag{4.33}$$

with $\|G_{j\cdot}^{-1}\|_2$ being the euclidean norm of the j-th row of the inverse of $\mathbf{G}$. If all the entries of a matrix are perturbed by adding a component wise source of noise sampled from a Gaussian distribution $N(0,\sigma^2)$, it can be concluded that the j-th output voltage noise variance driving from the programming noise is $\overline{v_{np,j}^2} = \|G_{j\cdot}^{-1}\|_2^2 \|\mathbf{V}\|_2^2 \sigma^2$.

4.5.3 Output Precision

In the following Subsection, the global output precision gets evaluated in both structures. All the systematic errors coming from the actual circuit implementation are not considered here. In this way, the analysis remains focused on the stochastic error contributions only.

The formerly obtained equations for programming and thermal noises were experimentally verified and subsequently employed to present noise contributions in Fig. 4.5. Both the feedback and open-loop output programming noise are proportional to the norm of the ideal vector $\mathbf{V}$. For increasing matrix sizes, given that the sum of the components of the vector $\mathbf{V}$ is equal to 1, the corresponding norm gets smaller. The consequence of this is that the programming noise σ_p decreases with a trend proportional to $(1/\sqrt{m})$ as shown in Figs. 4.5(a) and (b). The total thermal noise for a single column, in the open-loop architecture, shows to be proportional to matrix size m as in Eq. (4.17). Analogously, in the feedback architecture, the product in Eq. (4.25) shows to be proportional to m. As a consequence, for both configurations σ_T increases as the matrix size grows. Different noise trends can be found for other types of matrices, but Eqs. (4.17) and (4.25) are valid independent of the application. Although matrices of other kinds may yield different noise trends, Eqs. (4.17) and (4.25) hold independently from the application. These findings make feasible the system SNR and output bit precision computations. The output bit precision enables a direct comparison between the proposed analog systems and their digital implementation counterparts

[100]. The thermal noise is usually simulated in addition to the programming noise, however it was found that the final result is unchanged by assuming the two noise sources independent. The overall SNR for each output is:

$$SNR = \frac{\overline{V_o^2}}{\overline{v_{nT,o}^2} + \overline{v_{np,o}^2} + \overline{v_{n,amp}^2}} \tag{4.34}$$

with $\overline{v_{nT,o}^2}$ and $\overline{v_{np,o}^2}$ being the thermal and programming noise variances at each output, respectively, and $\overline{v_{n,amp}^2}$ is the voltage noise variance generated in the op-amp while $\overline{V_o^2}$ is proportional to the output signal power. The noise generated by op-amps is considered negligible. The output bit precision is calculated as:

$$\text{Output Bit Precision} = \frac{10\log(SNR) - 1.76}{6.02} \tag{4.35}$$

As shown in Fig. 4.5(a) and (c), thermal noise overshadows the programming noise in the open-loop architecture for bigger matrices and is the limiting factor to the output bit precision. This means that, for larger matrices, the design can be optimized by choosing lower precision memristors which save time and energy during the crossbar programming phase. On the other hand, in the feedback architecture, the programming noise is the performance limiting factor, and thus employing higher precision memristors results in better SNR. A single additional bit at the memristor level can yield even more than one bit improved precision at the crossbar output. The fact that the output bit precision in Fig. 4.5(f) lowers in the feedback crossbar configuration may be imputed to lower levels of output signal power for larger matrices.

By increasing the levels of output signal power, both the SNR and output bit precision can be improved. The negative side of using the feedback crossbar architecture in Fig. 4.1(c) over the open-loop one in Fig. 4.1(a) is the presence of memristors along the crossbar feedback loop. This unfortunately puts a limit to the highest possible output voltage which keeps memristors in their linear operation region. Consequently, this fact in turn lowers the SNR and output signal power.

Regarding the relationship between the conductance range and SNR, it is found that the memristor technology G_{on}/G_{off} ratio only marginally influences the thermal noise contribution for fixed G_{on} and decreasing G_{off}.

The programming noise does not depend on G_{on}/G_{off} ratio in the open-loop configuration (see Eq. (4.21)). Nonetheless, in the feedback architecture the parallel equivalent resistance $R_m = 1/G_m$ increases and thus also programming noise increases. One way to mitigate this effect consists in changing the input vector in Eq. (4.33). Larger G_{on}/G_{off} ratios primarily influence the individual memristors' bit precision, as shown in Fig. 4.5. Given the demonstrated better programming noise performance in larger crossbar arrays (see Figs. 4.5 (b) and (d)), in the following, G_{on}/G_{off} ratio is set to 10. Keeping G_{on}/G_{off} ratio fixed while decreasing G_{on} (i.e. $0.1 - 1\mu S$) increases R_m thus resulting in higher thermal noise and decreased SNR. This a trade-off results in improved power efficiency. At the same time, R_m strongly affects the power consumption if compared to G_{on}/G_{off} ratio.

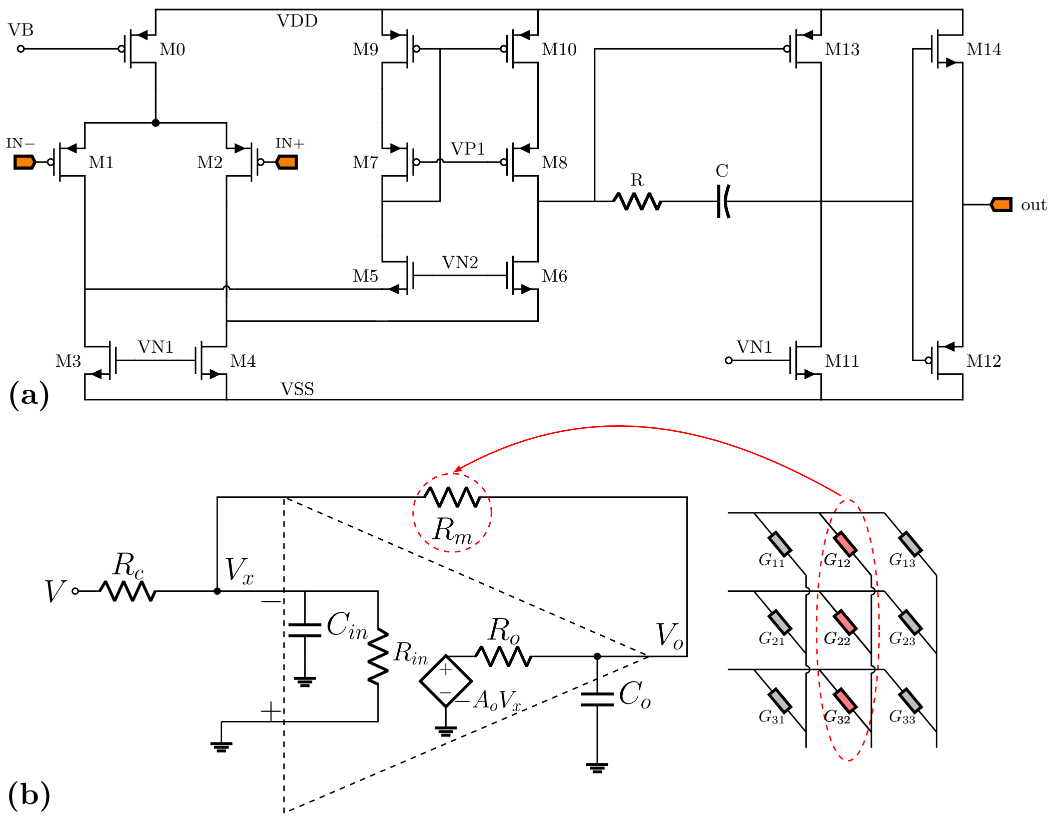


Fig. 4.6 a) Schematic of the transistor level op-amp. b) Schematic of a single TIA with the input resistor in a multi-loop feedback crossbar structure. R_m is the parallel combination of memristors resistances in the feedback path. Taken from [2] ©2021 IEEE.

4.6 System Accuracy Analysis

Precision metric can not entirely describe the performance of the system. In recent times, reducing the wire resistances effect by means of an estimation of linear scaling factors for each column output was proposed in [46]. Furthermore, other systematic errors affect the real world op-amp implementation such as finite gain, input and output resistance and capacitance, voltage offsets, non-linearity which all result in inaccuracy. As previously mentioned, also the stochastic fluctuations in the programmed memductance value, which are called programming errors, result in system inaccuracy. Moreover, the matrix' properties are very important for the feedback architecture overall accuracy. In the following, these additional problems are addressed and methods to mitigate the errors for both configurations are described.

4.6.1 Open-Loop Crossbar

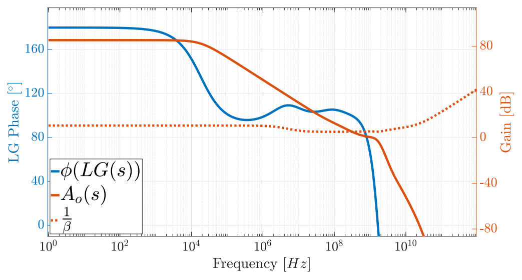


Fig. 4.7 Frequency domain analysis to determine stability of each loop for the feedback circuit of Fig. 1(c). The plots show the results for a single loop in a multi-loop system. The stability of the loop is determined by the intersection point of the open-loop gain curve of the op-amp ($A_o(s)$) and noise gain curve ($1/\beta(s)$). The phase margin is calculated on the LG phase curve ($\phi(LG(s))$) using this point. DC gain and GBW of the op-amp are $86dB$ and $1.1GHz$, respectively. Taken from [2] ©2021 IEEE.

On the one hand the main issue for the open-loop architecture is the voltage division due to the source impedance [101], while on the other hand, global system stability and op-amp gain are a lesser concern. Therefore, for current to voltage conversion, in place of a resistor, in the open-loop architecture a TIA is employed which provides the benefit of decreased input impedance seen from the crossbar which is:

$$R_i = \frac{R_f}{A_o + 1} \approx \frac{R_f}{A_o} \tag{4.36}$$

with R_i being the input resistance of the TIA while A_o the op-amp open-loop gain. During operation, all the memristors ought to be operated in their linear region [46] in order to avoid unwanted variation in the memductance levels. This means that the voltage between the memristors' terminals, V_m, ought to be kept much lower than the characteristic technology threshold switching voltage:

$$V_m < V_j - V_{ref} \tag{4.37}$$

with V_{ref} being the TIA virtual reference voltage while V_j the crossbar's j-th row input voltage.

As the gain of the op-amp is not infinite, each TIA's virtual reference node is not exactly equal to V_{ref}. This results in a systematic error introduced on each output voltage. An after-measurement compensation of the error which compensates this systematic error is possible. This error correction is derived by analyzing a non-ideal voltage adder. The k-th output voltage can be adjusted post-measurement by scaling it of a correction factor which is:

$$G_f + \frac{(\sum_{j=1}^{m} G_{jk} + G_{in} + G_f)(G_f + G_o)}{G_o A_o - G_f} \tag{4.38}$$

with G_{in} and G_o being the op-amp's input and output conductances respectively, $\sum_{j=1}^{m} G_{jk}$ the sum of the memristor conductances connected to the k-th TIA, and G_f the TIA feedback conductance. This multiplicative correction is operated all at once with the recovering step defined in Prop. 4.

4.6.2 Feedback Crossbar

The design of an accurate version of the feedback architecture in Fig. 4.1(c) poses more challenges than the open-loop configuration. This is due to the crossbar itself being located along the negative feedback path, which result in stricter requirements on the op-amp performance. In the feedback crossbar, each op-amp together with the crossbar resistance form a TIA which is responsible for the distribution of the input currents. A custom transistor-level op-amp is designed with 86 dB DC gain and 1.1 GHZ GBW in a 28 nm CMOS node with 0.9V supply voltage. Figure 6(b) reports the schematic of the designed op-amp which employs a two-stage folded cascode structure and a Class B output stage. While gain and global system stability are not a concern in the design of the open-loop crossbar, a careful analysis on these two is essential for the feedback crossbar. A sufficient condition that ensures the system operates in a stable negative feedback [77] is the positivity of the diagonal entries of the inverse matrix $(\mathbb{I}_m - \mathbf{M} + \mathbf{1}_{m\times m})^{-1}$. This condition is satisfied thanks to Prop. 3(a).

An a high open-loop gain operational amplifier is chosen in order to minimize the finite gain error. Unfortunately, this also results in high power consumption and therefore asks for a trade-off between power consumption and accuracy. The voltage difference between the TIA input (virtual reference) and the output nodes is limited to values lower than the memristor threshold switching voltage thus ensuring the memristors operating in their linear region. Apart from keeping the linearity which is fundamental in the operation of the crossbar array as a MVM accelerator, this also avoids memductance drifts due to slow unwanted reprogramming. The inverting terminal of the TIA must be at the virtual reference potential in order to guarantee the correct distribution of input current. Any potential value within the common mode range of the op-amp can be used as the virtual reference value. This can be compensated afterwards by subtracting this, possibly non-null, value from the TIA output voltage before the normalization step of the results.

The analysis of a single TIA loop gain in the feedback crossbar configuration is performed, using the circuit in Fig. 4.6(a), with the purpose of studying how non-idealities affect the output accuracy. Both op-amp input and output parasitic capacitances are neglected as they do not impact the gain error analysis. As in this special case all inputs are equal, approximate circuit could be studied where R_m is the equivalent parallel memristor resistances along the feedback path. The derived transfer function takes the

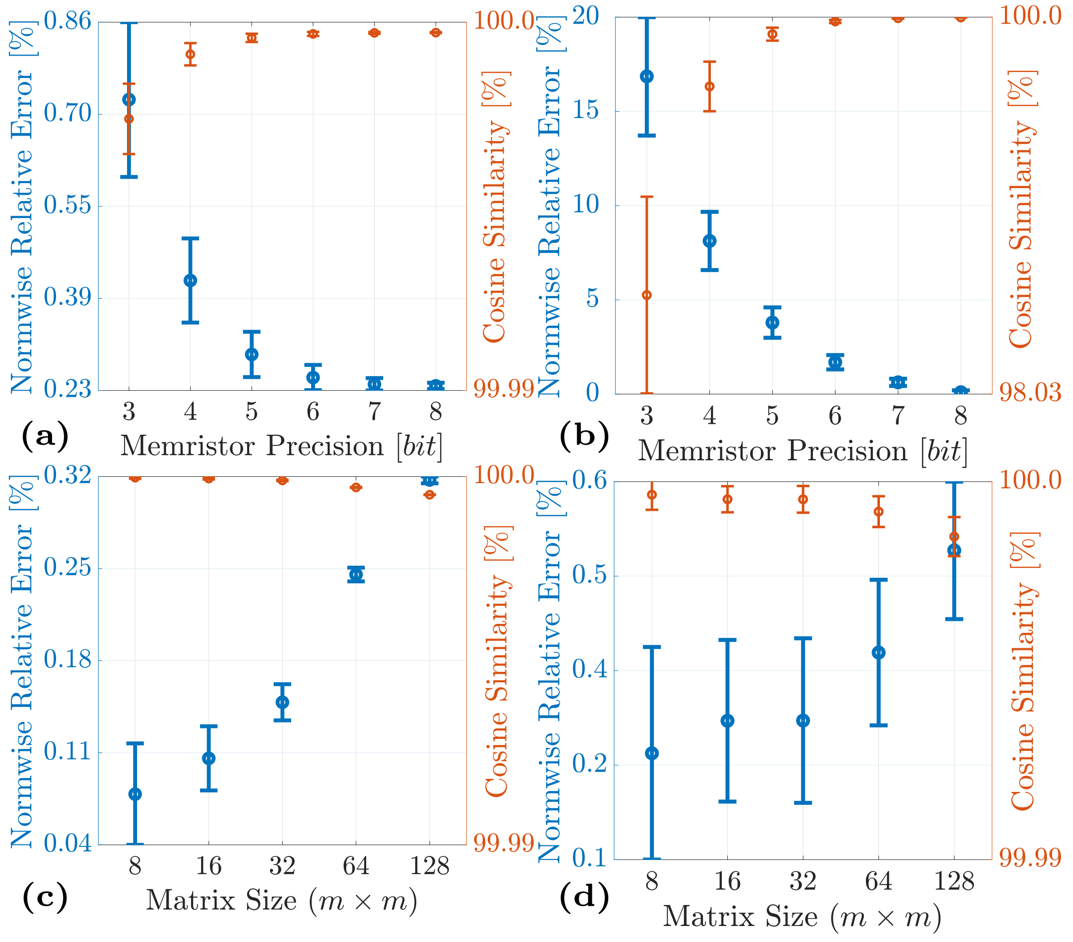


Fig. 4.8 Output accuracy in terms of normwise relative error and cosine similarity versus different memristor programming precisions for (a) open-loop and (b) feedback configurations for a 64 $\times$ 64 crossbar. Output accuracy versus different matrix sizes is given for (c) open-loop and (d) feedback configurations with 8 bits precision memristors and non-ideal op-amps. The expected accuracy is measured using 1000 perturbations of a random matrix. The error bars contain the distribution of each set of runs for 95% of the calculations. The normwise relative error is a more sensitive indicator than cosine similarity. The results are mainly affected by the errors arising from memristor precision rather than op-amp non-idealities. Taken from [2] ©2021 IEEE.

form:

$$\frac{V_o}{V} = -\frac{\frac{R_m}{R_c}}{1 + \frac{R_m + R_o}{A_o - \frac{R_o}{R_m}}\left(\frac{1}{R_c} + \frac{1}{R_m} + \frac{1}{R_{in}}\right)}. \tag{4.39}$$

The gain error E_g in Eq. (4.39) is:

$$E_g = \frac{R_m + R_o}{A_o - \frac{R_o}{R_m}}\left(\frac{1}{R_c} + \frac{1}{R_m} + \frac{1}{R_{in}}\right) \tag{4.40}$$

with R_{in} and R_o being the input and output resistances of the op-amp respectively, while A_o is the DC open-loop gain of the op-amp. It should be noted that the gain error can minimized either by increasing the open-loop gain, or by increasing the input resistance, or by reducing the op-amp output resistance. Furthermore, it should also be noted that a higher input resistance R_c and lower memristances in the crossbar reduce the gain error further.

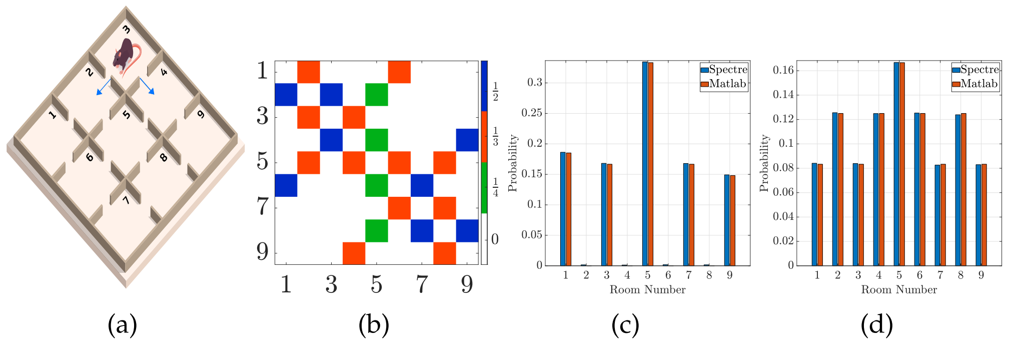


Fig. 4.9 (a) Example of a mouse trapped in a maze. (b) Transition matrix representing the probability that the mouse goes from one room to another. (c) Comparison between the 6-th step probability distribution using MATLAB and the proposed open-loop crossbar with 8–bits precision memristors. (d) Comparison between the stationary distribution using MATLAB and the proposed feedback crossbar with 8–bits precision memristors. Taken from [2] ©2021 IEEE.

Another factor to be meticulously taken into consideration is the relationships between input/output resistance of the op-amp, the input voltage and conductance range of the memristors in the crossbar. The output resistance R_o, in the feedback configuration (see Fig. 4.1(c)), is designed in order to avoid an incorrect virtual reference at node V_x and saturation as the op-amps are loaded with resistors. In turn, the value of the parallel combination of all the memristors along the feedback path R_m can be as low as 100Ω or even lower for very large crossbars. This requires the op-amps to capable of drive low resistance loads. In order to satisfy this requirement, the transistor-level op-amp design employs an efficient low output impedance stage which guarantees good linearity over the required output voltage swing range.

Also the stability analysis is carried out considering a single loop in the feedback crossbar architecture from Fig. 4.1(c). The effect on stability of the op-amp output capacitance is considered negligible as the pole that it forms in the transfer function of a single loop is placed at high frequencies thanks to the low value of the output resistance R_o. However this consideration holds as long as the value of the output capacitance does not substantially increases due to the ADC or op-amp output stage, in which case the pole moves to a lower frequency and should be taken into account. Conversely, the parasitic input capacitance of the op-amp C_{in}, is crucial to the stability of the crossbar configuration. This capacitor creates a pole in combination with R_m, R_c and R_o, which reduces the margin of phase and the system global stability. The frequency at which this pole lives is

$$f_p = \frac{R_c + R_m + R_o}{2\pi R_c C_{in}(R_m + R_o)}. \tag{4.41}$$

A possible way to achieve the maximum width of band available is to operate the circuit in Fig. 4.1(c) as an attenuator, e.g. $V_o < V$. Therefore, the system margin of phase is computed by considering the loop noise gain in Fig. 4.7. The op-amp input capacitance generates a zero in the transfer function which makes the noise gain curve increase by 20 dB/dec once passed the break point of f_p. The system margin of phase is reduced when f_p lives at a frequency lower than the loop unity-gain frequency. The effect of the input capacitance on the noise gain is:

$$\frac{1}{\beta} = 1 + \frac{Z_f}{Z_{in}} \tag{4.42}$$

with Z_f being the feedback impedance, β the feedback factor and $Z_{in} = R_c || \frac{1}{sC_{in}}$ for high R_{in}. The phase curve in Fig. 4.7 goes down rapidly because of the contribution given by the poles associated to the op-amp input capacitance which is located at f_p. The loop gain, expressed as $LG(s) = \beta A_o(s)$, has a margin of phase which is determined by means of the intersection point between the op-amp open-loop gain ($A_o(s)$) and the op-amp noise gain ($1/\beta$) curves.

The gain and stability need to be analyzed for all the loops in the circuit in order to guarantee the correct operation of the feedback crossbar architecture. Moreover, for different matrix sizes, trade-offs should be evaluated as R_m varies when the number of memristors changes in the crossbar.

In the same fashion of the precision analysis, also for the feedback configuration, properties of the programmed matrix are very relevant on accuracy and time to convergence [102]. Therefore, the same implementation of the feedback architecture may converge with different accuracy when changing the transition matrix.

4.6.3 Output Accuracy

The global output accuracies are calculated, taking into consideration the aforedescribed systematic errors, for both the feedback and open-loop architectures. The metrics employed are the cosine similarity and normwise relative error. Being $\mathbf{x}$ the high accuracy output obtained from a digital system (i.e. MATLAB), $\tilde{\mathbf{x}}$ the output vector measured on the crossbar architectures and $||\cdot||_2$ the euclidean vector norm, then the cosine similarity $sim(\theta)$ and the normwise relative error E_r are defined respectively as as:

$$sim(\theta) = \frac{\mathbf{x} \cdot \tilde{\mathbf{x}}}{||\mathbf{x}||_2 ||\tilde{\mathbf{x}}||_2}, \quad E_r = \frac{||\mathbf{x} - \tilde{\mathbf{x}}||_2}{||\mathbf{x}||_2}. \tag{4.43}$$

The variation of the programmed memductance for a same target value from cycle to cycle worsens the overall output precision as mentioned in Section 4.5. For the open-loop configuration, the accuracy evaluation depends on which of two following use cases are of interest. On the one hand if the next step distribution is the only computation of interest, then a unique programming error affects the open-loop configuration. This error is a particular sample drawn from the general distribution of programming noise. On the other hand if either the multi-step distribution or the stationary distribution of a MC are the computation of interest, then the programming error accumulates and degrades the output accuracy over iterations.

On the contrary, with the feedback architecture, if one is only interested in the calculation of the MC stationary distribution, each programming cycle generates a systematic error which affects the output accuracy.

The expected accuracy distribution computed by simulating the systems with both programming error and non-ideal op-amps is shown in Figure 4.8. How changing the memristor precision affects the output accuracy is shown in Figs. 4.8(a) and (b) for a 64×64 crossbar array. The impact of larger and larger random transition matrices, which range in size from 8×8 up to 128×128, is reported in Figs. 4.8(c) and (d) with a 8–bit precision memristors. The accuracy is calculated, for each plot, in terms of cosine similarity and normwise relative error. Each data point is obtained by averaging the results of 1000 simulations with a fixed random transition matrix and a new sample perturbation at each run. The reported error bars encompass the distribution of 95% of each set of simulations and the mean expected accuracy. It is clear how the output accuracy gets better as the memristor precision increases both in the open-loop and feedback architectures. Conversely, the global normwise relative error gets higher in both configurations when the matrix size grows. Other systematic sources of error do not depend on the cycle-to-cycle variation of programmed memductances. Cosine similarity has a trend opposite to the normwise relative error, as foreseeable from its definition (4.43). As cosine similarity saturates for the considered cases, it is not employed as a metric in the following of this Chapter.

4.7 Case Studies

In the following Section, two applications are proposed with the aim of demonstrating how the system performs on two quite different use cases. Both the feedback and open-loop crossbar configurations' performances are measured in terms of energy efficiency, accuracy and SNR.

4.7.1 Mouse in a Maze

One of the typical examples employed to explain of the concepts of a Markov Chain is the problem of the mouse in a maze (Fig. 4.9(a)). This toy example is perfect to illustrate the approach to MC problems with a random transition matrix. The maze is composed of nine rooms and each of them has gates to access adjacent rooms. At each following step, the mouse selects the room to visit next basing its choice on the probability columns from the transition matrix in Fig. 4.9(b). All the rooms are labeled with an integer value from 1 to 9. It is assumed that the mouse starts its exploration from room 3. Given the transition matrix, it is possible to forecast the position probability distribution of the mouse in the maze after an arbitrary number of time steps. This can be achieved by repeatedly applying Eq. (4.4); e.g. in Fig. 4.9(c) the position probability distribution after 6 steps is reported. The mouse is constrained to go only from an odd-numbered room to an even-numbered room or the other way around, no odd-to-odd or even-to-even movements are possible. Thus, the MC is periodic: for any starting position, even-numbered or odd-numbered states will alternate. Nonetheless, the mouse can enter all the rooms in the maze, which implies the MC is irreducible. This latter condition ensures that the stationary distribution is unique. By solving Eq. (4.7), one can analytically find $\mathbf{p}_\infty = \left[\frac{1}{12}, \frac{1}{8}, \frac{1}{12}, \frac{1}{8}, \frac{1}{6}, \frac{1}{8}, \frac{1}{12}, \frac{1}{8}, \frac{1}{12}\right]^T$, see Fig. 4.9(d). The result does not match the limiting distribution because the chain is not regular, still it provides the mean occupation times of each room.

Performance Evaluation

The calculation of the probability distribution after 6 steps is performed via the open-loop architecture from Fig. 4.1(a) with 8–bits precision memristors and simulated by the Cadence's Spectre simulator using the standard corner conditions, convergence settings, and a temperature of 27°C. MATLAB is employed to collect the raw output voltages from the circuit simulation, normalize the obtained values and iterate the process. In parallel MATLAB also computes the ideal (true) results used to measure the accuracy of the system. The obtained probability distribution is shown in Fig. 4.9(c). The normwise relative error is found to be $\approx 0.82\%$. The stationary distribution computation is performed via the feedback architecture in Fig. 4.1(c) and the resulting stationary distribution $\mathbf{p}_\infty$ is shown in Fig. 4.9(d). The normwise relative error is found to be $\approx 0.61\%$. Both architectures show encouraging performance in terms of accuracy, thanks to the modest size of the application transition matrix. These results can still be improved by reducing systematic errors via higher precision memristors and/or higher gain operational amplifiers. Both SNR and energy efficiency random matrix problem of this kind are here not considered and are evaluated for the next application which has a larger matrix size.

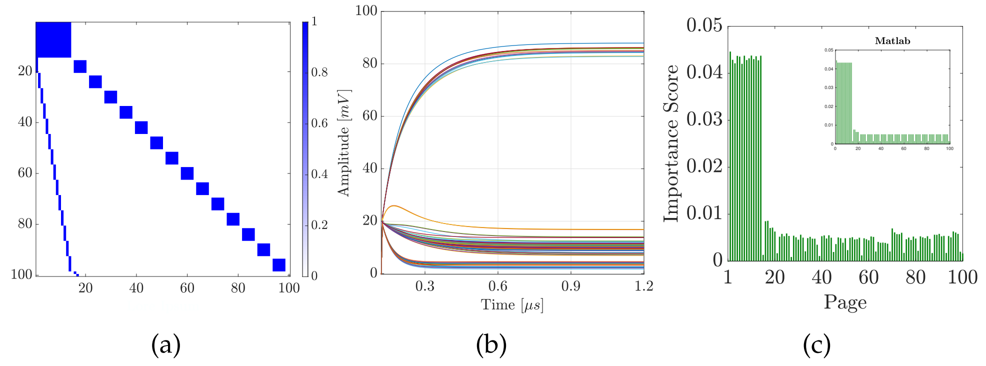


Fig. 4.10 (a) Adjacency matrix representing the links between the 100 pages of MathWorks dataset. (b) Transient behavior of the feedback system converging to the limiting distribution of the chain. (c) Simulated pageranks provided by the feedback crossbar. Taken from [2] ©2021 IEEE.

4.7.2 The Pagerank Algorithm

At the core of Google's search engine there is a Markov Chain problem [78]. This application and type of MC is quite different from the previous mouse-in-a-maze example and serves the purpose of exploring the applicability of the proposed architectures for real world use cases. Pagerank algorithm has at its core the solution of a MC problem. It considers a random surfing on the web graph [103]. Each page is assigned a score proportional to the probability of being visited after a large number of transitions which basically means computing a MC limiting distribution. Consider a web made of m pages, an adjacency matrix $\mathbf{A}$ can be created by the following process: set to 1 A_{ij} if node i is linked to node j, and 0 otherwise. At this point, the transition matrix **W** can be obtained from A by normalizing all the entries of the matrix so that each column sums to 1. All the columns which contain only zeros (i.e. dangling nodes) are substituted by uniform distribution vectors so that **W** is a well-defined stochastic matrix. This way of constructing the matrix does not ensure the irreducibility of the Markov Chain and thus the existence of an unique stationary distribution. This may happen if not all the pages can be reached from any given arbitrary node. A workaround for this issue which makes the transition matrix irreducible is to allow the web surfer to navigate from the current web-page to a new random one. Applying this solution means perturbing the transition matrix **W** with a damping factor α which uniformly spreads part of the rank. In formulas this can be expressed as $\mathbf{M} = \alpha\mathbf{W} + (1-\alpha)\mathbf{R}$, where **R** is the matrix with all entries equal to $\frac{1}{m}$. The obtained matrix **M** is usually called the Google Matrix and is computed with $\alpha \approx 0.85$. **M** is also primitive. This latter property result in two valid interpretations of the Pagerank: mean occupation times or long-run probabilities.

Performance Evaluation

The feedback architecture from Fig. 4.1(c) is employed to execute the aforeintroduced Pagerank algorithm with 8–bits precision memristors. With the aim of producing a realistic comparison with the recent literature, the MathWorks dataset [104] is used. Fig. 4.10(a) reports the computed adjacency matrix. Cadence/Spectre is employed to perform the circuit simulation. The resulting time transient simulation of the outputs is reported in Fig. 4.10(b). MATLAB is used to normalize the obtained results. In Fig. 4.10(c), the resulting pages' importance score is reported. From the simulations, the normwise relative error is estimated to be $\approx 4.2\%$.

The op-amp transistor level implementation, provided in Section 4.6, is employed to perform the simulations of this application with $\approx 86dB$ open-loop gain, $1.1GHz$ gain-bandwidth-product while consuming $163\mu W$ of power with $0.9V$ power supply. It can be noted from Fig. 4.10(b) that the op-amps converge in $552ns$ within 0.1% error of their final value. The

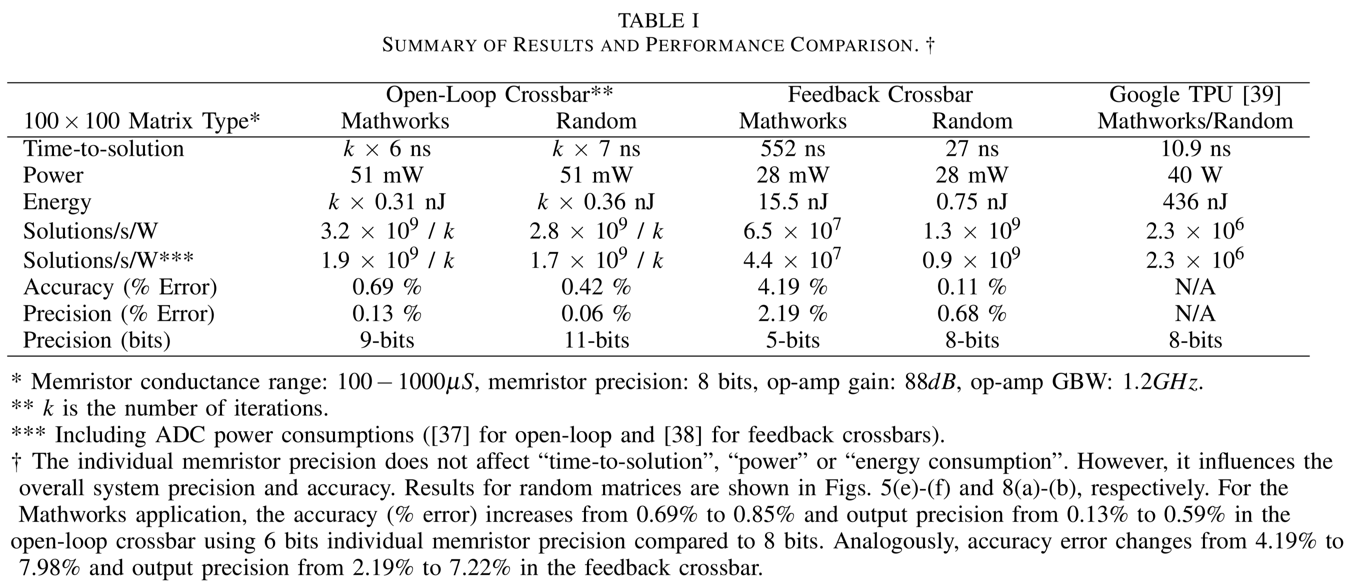


TABLE I
SUMMARY OF RESULTS AND PERFORMANCE COMPARISON. †

	Open-Loop Crossbar**		Feedback Crossbar		Google TPU [39]
100×100 Matrix Type*	Mathworks	Random	Mathworks	Random	Mathworks/Random
Time-to-solution	$k \times 6$ ns	$k \times 7$ ns	552 ns	27 ns	10.9 ns
Power	51 mW	51 mW	28 mW	28 mW	40 W
Energy	$k \times 0.31$ nJ	$k \times 0.36$ nJ	15.5 nJ	0.75 nJ	436 nJ
Solutions/s/W	3.2×10^9 / k	2.8×10^9 / k	6.5×10^7	1.3×10^9	2.3×10^6
Solutions/s/W***	1.9×10^9 / k	1.7×10^9 / k	4.4×10^7	0.9×10^9	2.3×10^6
Accuracy (% Error)	0.69 %	0.42 %	4.19 %	0.11 %	N/A
Precision (% Error)	0.13 %	0.06 %	2.19 %	0.68 %	N/A
Precision (bits)	9-bits	11-bits	5-bits	8-bits	8-bits

* Memristor conductance range: $100 - 1000\mu S$, memristor precision: 8 bits, op-amp gain: $88dB$, op-amp GBW: $1.2GHz$.
** k is the number of iterations.
*** Including ADC power consumptions ([37] for open-loop and [38] for feedback crossbars).
† The individual memristor precision does not affect "time-to-solution", "power" or "energy consumption". However, it influences the overall system precision and accuracy. Results for random matrices are shown in Figs. 5(e)-(f) and 8(a)-(b), respectively. For the Mathworks application, the accuracy (% error) increases from 0.69% to 0.85% and output precision from 0.13% to 0.59% in the open-loop crossbar using 6 bits individual memristor precision compared to 8 bits. Analogously, accuracy error changes from 4.19% to 7.98% and output precision from 2.19% to 7.22% in the feedback crossbar.

Fig. 4.11 Taken from [2] ©2021 IEEE.

amount of total power dissipated in the system is:

$$\sum_{i=1}^{100}(I_{in}^2 R_{m,i} + I_{in}^2 R_c + P_{amp}) \approx 28\ mW \tag{4.44}$$

with I_{in} being the input current in each row and $R_{m,i}$ the parallel of the memristances connected to the $i-$th row. The analysis shows that $>$ 90% of the total power is dissipated between the op-amps and the input resistors R_c. The op-amp′s power dissipation might be improved by employing a more advanced technology node. Nevertheless, the power efficiency of the feedback architecture may still be limited due to the requirements of high gain and GBW.

Equation (4.34) is employed to compute the system SNR for this Pagerank use case. The SNR mean over the 100 output components is computed to be $32dB$, which corresponds to an output bit precision of $\approx$ 5 *bits*.

The MathWorks′ transition matrix minimum eigenvalue is found to be $\lambda_{min} = 0.0028$ which is very low and thus it results in a longer convergence time [102]. Conversely, larger minimum eigenvalues significantly reduces the convergence time. To make it clear, when programmed with a random stochastic matrix the feedback circuit converges in just 27 *ns* which is 20 times less than the time needed for the MathWorks′ transition matrix. This proves that the feedback architecture performance is heavily dependent on the properties of the transition matrix. Furthermore, as shown in Table I, three additional output bit precision are obtained for a random transition matrix. The application matrix condition number strongly influences the system precision. Given a random transition matrix **M**, the condition number of the matrix $(\mathbb{I}_m - \mathbf{M} + \mathbf{1}_{m\times m})$ is on average 10 times smaller than the MathWorks′ one. For ill-conditioned matrices (e.g. the one employed to compute the Pagerank of MathWorks) small perturbations in both the input and the matrix result in larger errors at the output [81].

With the aim of comparing the feedback and the open-loop crossbar architectures on the Pagerank of MathWorks′ network, the Power method is used in the open-loop configuration. The same transistor level op-amp implementation from the feedback circuit is adopted also in the case of the open-loop configuration. In this case, energy efficiency can be improved because the open-loop crossbar op-amp has weaker constraints on the gain or GBW with respect to the feedback crossbar. The total open-loop crossbar power dissipation is:

$$\sum_{j=1,k=1}^{100} (I_{jk}^2 R_{jk} + I_f^2 R_f + P_{amp}) \approx 51\ mW \tag{4.45}$$

with R_{jk} being the resistance of a generic memristor in the crossbar, I_{jk} the current through this memristor and and I_f the total current through the feedback resistor. In contrast to the feedback circuit, the open-loop crossbar time to convergence does not depend on the matrix properties. Accuracy in terms of normwise relative error is estimated from simulations to be $\approx 0.69\%$. The open-loop configuration is $\approx 56dB$, which corresponds to $\tilde{9}$ *bits* output precision per iteration. The open-loop crossbar SNR analysis in the case of a random matrix is also reported in Table I.

Normwise relative error is employed in order to assess the relative systematic shift between the ideal value computed by a digital system and the one obtained from the proposed architecture. In order to fairly compare accuracy and precision, the systematic shift vector $\mathbf{x} - \tilde{\mathbf{x}}$ from Eq. (4.43) is substituted by a vector whose total output noise standard deviations are $\sigma_{TOT_i} = \sqrt{\sigma_{p_i}^2 + \sigma_{T_i}^2}\ \forall i = 1, \ldots, m$. The introduced metric measures the relative dispersion from the ideal output value $\mathbf{x}$. Table I reports the percent normwise relative errors which, for the open-loop crossbar, refer to a single iteration.

The Power Method algorithm implementable in the open-loop architecture might be executed in two very different ways:

- via an analog normalization and recovery circuitry.
- via Analog-to-Digital-Conversion (ADC) performing normalization and recovery steps on a digital system and, afterwards, feeding back the result in input for the following iteration via Digital-to-Analog-Converters (DACs).

Registers may be either analog sample and hold circuits or CMOS based digital implementations, are fundamental in order to apply previous step results as input to the next step. In both cases, these additional components will dissipate power and increase the computation latency. The usage at each iteration of a digital system introduces quantization noise hence further reducing the global output precision of the system. The required number of steps is a function of both the specific application and the target output accuracy. Table I reports the energy and time cost to compute k iterations via the open-loop crossbar configuration.

To sum up, the feedback configuration may enable higher energy efficiency thanks to its one-step operation even if this might be limited because of the time to convergence of the architecture. Another point in favor of the feedback crossbar solution is its input circuitry which requires a fixed vector of same magnitude currents and thus results much simpler and more efficient than the one required by the open-loop crossbar architecture. A drawback of the feedback crossbar is that it is convenient only for moderately high output precision computations (around 4 to 8 *bits*). When higher precision is required (e.g. $(> 9\ bits)$), the open-loop crossbar solution might return the desired results after a sufficient number of iterations. Another possibility would be to use the merge two solutions into a single architecture where a first estimation of the stationary distribution is computed via the feedback architecture and few high precision steps are performed by an open-loop configuration. Both the proposed solutions beat the competing digital systems on the same task [105] in terms of solutions/s/W as shown in Table I.

4.8 Conclusion

In this Chapter, the process of using memristor crossbars to solve MC problems was described. It was shown that the open-loop architecture can implement the iterative computations of the MC $k-$th distribution, while the feedback architecture entirely avoids possible convergence issues of iterative algorithms by computing the stationary distribution in one single step. The proposed circuit implementations were thoroughly analyzed and, in the meanwhile, various circuit design trade-offs were introduced. The impact of matrix size and memristor precision on the output accuracy and precision was studied using SNR, output bit precision and normwise relative error metrics. The MathWorks dataset was used to prove the effectiveness of the two proposed solutions even for ill-conditioned matrices. The obtained results show how memristor crossbars based analog accelerators might in the near future overcome conventional digital computing systems in domains where the energy efficiency is a key requirement while maintaining at the same time a competitive output precision.

A possible future extension of this study should try to generalize the presented results and take into consideration other important non-idealities such as interconnections and source driver's parasitic resistance along with other device parasitics.

Chapter 5

Conclusions

5.1 Summary

The first modeling effort in Chapter 1 of this thesis has been devoted to Phase Change Memories. PCMs are rising in prominence as CMOS technology alternatives which are starting being employed as fundamental elements for implementing the MVM operation in neuromorphic computing, as alternative to NAND flash for mass storage designs and even as tunable elements for reconfigurable circuits. In this manuscript, the development has been reported of a novel state-dependent Ohm's Law to describe the dynamical evolution of the PCM devices' geometrical state variable (amorphous thickness) during the WRITE operation. This stem from the observation of the intriguing complex dynamics of Phase Change Memories. A simulation model was developed starting from physics models available in the literature as well as newly collected laboratory measurements on nanometric PCM device of the mushroom kind. The Dynamic Route Maps were computed for the first order dynamics of the device during the write operation which gradually drives the device from the initial RESET state to a low resistance state. The DRM were also experimentally measured by meas of the successive application of write pulses interleaved with read pulses. This represents a noteworthy empirical verification and has supported the establishment of an accurate 1-st order model to describe PCM dynamics. One important information learnt from the DRMs for PCM is the meticulous design of SET pulses to govern the evolution of the amorphous dome thickness and thus the cell low-field resistance. From the knowledge gained in this study, suitable SET pulses can be selected to modulate dissipation in analog electronic circuits.

The second modeling effort in Chapter 2 of this thesis has been focused on Resistive Random Access Memories. Firstly a simplified analytical model of second order ReRAM memristors was derived. This is formed by only two variables (memductance and internal temperature) which are directly relatable to the state variables used in certain advanced biophysical models, in particular the synaptic efficacy and the calcium concentration. Thanks to its capability of forecasting the memductance evolution for multiple combinations of input spikes, this model was employed to investigate single memristors' synaptic properties and the overall dynamics of memristor spiking networks. The synaptic response of ReRAMs to stimulation was examined at different frequencies for different protocols: cycles of spike pairs, triplets, and quadruplets. The knowledge gained from this study shows that:

- The most relevant synaptic characteristics of ReRAMs (extensible to many second-order memristors) can be readily studied and forecasted.
- The proposed compact model can capture many synaptic behaviors why can not be correctly reproduced by standard spike pair based STDP models.

In Chapter 3 this ReRAM model has found application on the modeling of memristor-based Spiking Neural Networks. SNNs have been characterized as they were discrete nonlinear dynamic systems, with memductances selected as state variables and pre and postsynaptic spikes as inputs and outputs. The state equations governing the memductance evolution have been derived, and it has also been proven that the network response to periodic presynaptic inputs can be easily determined by computing the equilibria of the system and discussing their stability properties.

The other good example of neuromorphic computing platform presented in Chapter 3 is the memristor-based Recurrent Neural Network to be trained by meas of two different generalizations of the backpropagation algorithm.

The dynamical evolution of this memristor–based RNN has been analyzed in detail. In situ learning rules, such ash Equilibrium Propagation, enable the memristor–based artificial neural network to continuously adapt and adjust the synaptic weights. This all while skipping the loss function's gradient direct computation. The results of simulation put it clear that both methods strongly outperform conventional solution for corrupted-pattern reconstruction problems. Equilibrium Propagation showed to perform well also on classification tasks.

The last part of Chapter 3 focuses on a more conventional linear algebra hardware accelerator. Firstly, how to solve MC problems using memristor crossbars was clarified. From the study has been found that the open-loop crossbar architecture may implement iterative computations and also find the probability distribution after k steps in a sequence. On the opposite, the feedback crossbar architecture directly computes the stationary distribution problem thus avoiding all the possible convergence issues which may arise in iterative algorithms. The circuit was carefully analyzed and a various circuit trade-offs were discussed. SNR, output bit precision and normwise error were estimated in order to measure the impact of of memristor bit precision and matrix size on output accuracy and precision. The capabilities of the modeled architectures were proven on the MathWorks dataset which provides a real-world ill-conditioned matrix. It turned out that memristive crossbars are good candidates to take over conventional digital computing in terms of energy efficiency while retaining a competitive output precision for certain classes of specific applications.

5.2 Outlook

On the PCM modeling side, a future effort could focus on the integration of the proposed physics-based compact model into actual automatic design tools for programmable analog circuits. Another research direction would be to exploit the obtained Dynamic Route Maps for designing a broad class of input stimuli which may result more effective and convenient for programming PCM devices.
On the ReRAM modeling side, a promising research direction would be to

employ the proposed second-order model along with some theoretical and numerical advanced nonlinear dynamic techniques in order to thoroughly investigate the response of memristor-based spiking neural networks to arbitrary presynaptic inputs and the underlying learning mechanisms.
Concerning the implementation of artificial recurrent neural networks, additional work is needed in order to find the memristive technology that works for the online training phase via Equilibrium Propagation algorithm.
Concerning memristor-based hardware accelerators for linear algebra problems, in upcoming publications, the presented will be generalized and also other important non-idealities (wire and source driver resistances etc) will be incorporated into the modeling.

References

[1] Francesco Marrone, Gianluca Zoppo, Fernando Corinto, and Marco Gilli. A dynamic system approach to spiking second order memristor networks. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 69(4):1641–1654, 2022.

[2] Gianluca Zoppo, Anil Korkmaz, Francesco Marrone, Samuel Palermo, Fernando Corinto, and R. Stanley Williams. Analog solutions of discrete markov chains via memristor crossbars. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 68(12):4910–4923, 2021.

[3] L. Chua. Memristor-the missing circuit element. *IEEE Transactions on Circuit Theory*, 18(5):507–519, 1971.

[4] L. O. Chua and Sung Mo Kang. Memristive devices and systems. *Proceedings of the IEEE*, 64(2):209–223, 1976.

[5] Fernando Corinto, Pier Paolo Civalleri, and Leon O Chua. A theoretical approach to memristor devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 5(2):123–132, 2015.

[6] Leon Chua. If it's pinched it's a memristor. *Semiconductor Science and Technology*, 29(10):104001, sep 2014.

[7] D. Strukov, G. Snider, and D. Stewart R. S. Williams. The missing memristor found. *Nature*, 453:80–83, 2008.

[8] Sangho Shin, Kyungmin Kim, and Sung-Mo Kang. Memristor applications for programmable analog ics. *IEEE Transactions on Nanotechnology*, 10(2):266–274, 2011.

[9] Julien Borghetti, Gregory S. Snider, Philip J. Kuekes, J. Joshua Yang, Duncan R. Stewart, and R. Stanley Williams. 'memristive' switches enable 'stateful' logic operations via material implication. *Nature*, 464(7290):873–876, 4 2010.

[10] Han Bao, Zhongyun Hua, Houzhen Li, Mo Chen, and Bocheng Bao. Discrete memristor hyperchaotic maps. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 68(11):4534–4544, 2021.

[11] Mario Lanza, Abu Sebastian, Wei D. Lu, Manuel Le Gallo, Meng-Fan Chang, Deji Akinwande, Francesco M. Puglisi, Husam N. Alshareef, Ming Liu, and Juan B. Roldan. Memristive technologies for data storage, computation, encryption, and radio-frequency communication. *Science*, 376(6597):eabj9979, 2022.

[12] Manuel Le Gallo and Abu Sebastian. An overview of phase-change memory device physics. *Journal of Physics D: Applied Physics*, 53:213002, 2020.

[13] M. Nardone, M. Simon, I. V. Karpov, and V. G. Karpov. Electrical conduction in chalcogenide glasses of phase change memory. *Journal of Applied Physics*, 112(7):071101, 2012.

[14] H. J. Kroezen, G. Eising, G. ten Brink, G. Palasantzas, B. J. Kooi, and A. Pauza. Schottky barrier formation at amorphous-crystalline interfaces of gesb phase change materials. *Applied Physics Letters*, 100(9):094106, 2012.

[15] Manuel Le Gallo, Aravinthan Athmanathan, Daniel Krebs, and Abu Sebastian. Evidence for thermally assisted threshold switching behavior in nanoscale phase-change memory cells. *Journal of Applied Physics*, 119(2):025704, 2016.

[16] A. Faraclas, N. Williams, A. Gokirmak, and H. Silva. Modeling of set and reset operations of phase-change memory cells. *IEEE Electron Device Letters*, 32(12):1737–1739, 2011.

[17] Geoffrey W. Burr, Matthew J. Breitwisch, Michele Franceschini, Davide Garetto, Kailash Gopalakrishnan, Bryan Jackson, Bülent Kurdi, Chung Lam, Luis A. Lastras, Alvaro Padilla, Bipin Rajendran, Simone Raoux, and Rohit S. Shenoy. Phase change memory technology. *Journal of Vacuum Science & Technology B*, 28(2):223–262, 2010.

[18] L. Chua. Five non-volatile memristor enigmas solved. *Applied Physics A*, 124(8):563, 7 2018.

[19] Alon Ascoli, Ioannis Messaris, Ronald Tetzlaff, and Leon O Chua. Theoretical foundations of memristor cellular nonlinear networks: Stability analysis with dynamic memristors. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(4):1389–1401, 2019.

[20] Alon Ascoli, Stephan Menzel, Vikas Rana, Tim Kempen, Ioannis Messaris, Ahmet Samil Demirkol, Michael Schulten, Anne Siemon, and Ronald Tetzlaff. A deep study of resistance switching phenomena in taox reram cells: System-theoretic dynamic route map analysis and experimental verification. *Advanced Electronic Materials*, 8(8):2200182, 2022.

[21] Alon Ascoli, Ronald Tetzlaff, Sung-Mo Kang, and Leon O Chua. Theoretical foundations of memristor cellular nonlinear networks: a drm 2-based method to design memcomputers with dynamic memristors. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(8):2753–2766, 2020.

[22] David Maldonado, Mireia B Gonzalez, Francesca Campabadal, Francisco Jimenez-Molinos, M Moner Al Chawa, Stavros G Stavrinides, Juan B Roldan, Ronald Tetzlaff, Rodrigo Picos, and Leon O Chua. Experimental evaluation of the dynamic route map in the reset transition of memristive rerams. *Chaos, Solitons & Fractals*, 139:110288, 2020.

[23] Francesco Marrone, Jacopo Secco, Benedikt Kersting, Manuel Le Gallo, Fernando Corinto, Abu Sebastian, and Leon O. Chua. Experimental validation of state equations and dynamic route maps for phase change memristive devices. *Scientific Reports*, 12(1):6488, Apr 2022.

[24] R. Ongaro and A. Pillonnet. Synthetic theory of poole and poole-frenkel (pf) effects. *IEE Proceedings A (Science, Measurement and Technology)*, 138:127–137(10), 3 1991.

[25] J. Frenkel. On pre-breakdown phenomena in insulators and electronic semi-conductors. *Phys. Rev.*, 54:647–648, 10 1938.

[26] Y. H. Shih, M. H. Lee, M. Breitwisch, R. Cheek, J. Y. Wu, B. Rajendran, Y. Zhu, E. K. Lai, C. F. Chen, H. Y. Cheng, A. Schrott, E. Joseph, R. Dasaka, S. Raoux, H. L. Lung, and C. Lam. Understanding amorphous states of phase-change memory using frenkel-poole model. In *2009 IEEE International Electron Devices Meeting (IEDM)*, pages 1–4, 2009.

[27] Robert M. Hill. Poole-frenkel conduction in amorphous solids. *The Philosophical Magazine: A Journal of Theoretical Experimental and Applied Physics*, 23(181):59–86, 1971.

[28] J. L. Hartke. The three-dimensional poole-frenkel effect. *Journal of Applied Physics*, 39(10):4871–4873, 1968.

[29] Giovanni Betti Beneventi, Lucrezia Guarino, Massimo Ferro, and Paolo Fantini. Three-dimensional poole-frenkel analytical model for carrier transport in amorphous chalcogenides. *Journal of Applied Physics*, 113(4):044506, 2013.

[30] Abu Sebastian, Manuel Le Gallo, and Daniel Krebs. Crystal growth within a phase change memory cell. *Nature Communications*, 5(1):4314, 7 2014.

[31] Manuel Le Gallo. *Phase-Change Memory: Device Physics and Application to non-von Neumann Computing*. PhD thesis, ETH Zurich, 2017.

[32] Abu Sebastian, Nikolaos Papandreou, Angeliki Pantazi, Haralampos Pozidis, and Evangelos Eleftheriou. Non-resistance-based cell-state metric for phase-change memory. *Journal of Applied Physics*, 110(8):084505, 2011.

[33] J. Secco, F. Corinto, and A. Sebastian. Flux–charge memristor model for phase change memory. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 65(1):111–114, 2018.

[34] Daniele Ielmini and Yuegang Zhang. Analytical model for subthreshold conduction and threshold switching in chalcogenide-based memory devices. *Journal of Applied Physics*, 102(5):054517, 2007.

[35] Y. Hayakawa, A. Himeno, R. Yasuhara, W. Boullart, E. Vecchio, T. Vandeweyer, T. Witters, D. Crotti, M. Jurczak, S. Fujii, S. Ito, Y. Kawashima, Y. Ikeda, A. Kawahara, K. Kawai, Z. Wei, S. Muraoka, K. Shimakawa, T. Mikawa, and S. Yoneda. Highly reliable taox reram with centralized filament for 28-nm embedded application. In *2015 Symposium on VLSI Circuits (VLSI Circuits)*, pages T14–T15, 2015.

[36] Antonio C. Torrezan, John Paul Strachan, Gilberto Medeiros-Ribeiro, and R. Stanley Williams. Sub-nanosecond switching of a tantalum oxide memristor. *Nanotechnology*, 11 2011.

[37] Myoung-Jae Lee, Chang Bum Lee, Dongsoo Lee, Seung Ryul Lee, Man Chang, Ji Hyun Hur, Young-Bae Kim, Chang-Jung Kim, David H. Seo, Sunae Seo, U-In Chung, In-Kyeong Yoo, and Kinam Kim. A fast, high-endurance and scalable non-volatile memory device made from asymmetric ta2o5-x/tao2-x bilayer structures. *Nature Materials*, 10(8):625–630, Aug 2011.

[38] W. Kim, S. Menzel, D. J. Wouters, R. Waser, and V. Rana. 3-bit multilevel switching by deep reset phenomenon in pt/w/taox/pt-reram devices. *IEEE Electron Device Letters*, 37(5):554–567, 2016.

[39] Ji Hyun Hur, Myoung-Jae Lee, Chang Bum Lee, Young-Bae Kim, and Chang-Jung Kim. Modeling for bipolar resistive memory switching in transition-metal oxides. *Phys. Rev. B*, 82:155321, Oct 2010.

[40] Sungho Kim, Chao Du, Patrick Sheridan, Wen Ma, ShinHyun Choi, and Wei D. Lu. Experimental demonstration of a second-order memristor and its ability to biorealistically implement synaptic plasticity. *Nano Letters*, 15(3):2203–2211, 3 2015.

[41] Michael Graupner and Nicolas Brunel. Calcium-based plasticity model explains sensitivity of synaptic changes to spike pattern, rate, and dendritic location. *Proceedings of the National Academy of Sciences*, 109(10):3991–3996, 2012.

[42] Per Jesper Sjöström, Gina G. Turrigiano, and Sacha B. Nelson. Rate, timing, and cooperativity jointly determine cortical synaptic plasticity. *Neuron*, 32(6):1149–1164, 12 2001.

[43] Jean-Pascal Pfister and Wulfram Gerstner. Triplets of spikes in a model of spike timing-dependent plasticity. *Journal of Neuroscience*, 26(38):9673–9682, 2006.

[44] R. S. Williams. What's next? [the end of moore's law]. *Computing in Science Engineering*, 19(2):7–13, 2017.

[45] Can Li et al. Analogue signal and image processing with large memristor crossbars. *Nature Electronics*, 1(1):52, 2018.

[46] Miao Hu et al. Memristor-based analog computation and neural network classification with a dot product engine. *Advanced Materials*, 30(9):1705914, 2018.

[47] M. Hu et al. Dot-product engine for neuromorphic computing: Programming 1t1m crossbar to accelerate matrix-vector multiplication. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2016.

[48] M. Hu, J. P. Strachan, Zhiyong Li, R. Stanley, and Williams. Dot-product engine as computing memory to accelerate machine learning algorithms. In *2016 17th International Symposium on Quality Electronic Design (ISQED)*, pages 374–379, 2016.

[49] A. Shafiee et al. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 14–26, 2016.

[50] Aayush Ankit et al. PUMA: A programmable ultra-efficient memristor-based accelerator for machine learning inference. *CoRR*, abs/1901.10351, 2019.

[51] Gianluca Zoppo, Francesco Marrone, and Fernando Corinto. Equilibrium propagation for memristor-based recurrent neural networks. *Frontiers in Neuroscience*, 14, 2020.

[52] YeonJoo Jeong and Wei Lu. Neuromorphic computing using memristor crossbar networks: A focus on bio-inspired approaches. *IEEE Nanotechnology Magazine*, 12(3):6–18, 2018.

[53] Richard Kempter, Wulfram Gerstner, and J. Leo van Hemmen. Hebbian learning and spiking neurons. *Phys. Rev. E*, 59:4498–4514, 4 1999.

[54] Wolfgang Maass and Henry Markram. On the computational power of circuits of spiking neurons. *Journal of Computer and System Sciences*, 69(4):593–616, 2004.

[55] Jean-Pascal Pfister, Taro Toyoizumi, David Barber, and Wulfram Gerstner. Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning. *Neural Computation*, 18(6):1318–1348, 06 2006.

[56] Walter Senn and Jean-Pascal Pfister. *Spike-Timing-Dependent Plasticity, Learning Rules*, pages 1–10. Springer New York, New York, NY, 2013.

[57] Luis B Almeida. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *Proceedings, 1st First International Conference on Neural Networks*, volume 2, pages 609–618. IEEE, 1987.

[58] Fernando J Pineda. Generalization of back propagation to recurrent and higher order neural networks. In *Neural information processing systems*, pages 602–611, 1988.

[59] Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.

[60] Emmanuelle J Merced-Grafals, Noraica Dávila, Ning Ge, R Stanley Williams, and John Paul Strachan. Repeatable, accurate, and high speed multi-level programming of memristor 1t1r arrays for power efficient analog computing applications. *Nanotechnology*, 27(36):365202, 8 2016.

[61] R. Liu, D. Mahalanabis, H. J. Barnaby, and S. Yu. Investigation of single-bit and multiple-bit upsets in oxide rram-based 1t1r and crossbar memory arrays. *IEEE Transactions on Nuclear Science*, 62(5):2294–2301, 10 2015.

[62] CHUA Leon. Everything you wish to know about memristors but are afraid to ask. *Radioengineering*, 24(2):319, 2015.

[63] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[64] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience*, 10:508, 2016.

[65] Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7:13276, 2016.

[66] Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 498–515. Springer, 2015.

[67] James CR Whittington and Rafal Bogacz. Theories of error back-propagation in the brain. *Trends in cognitive sciences*, 2019.

[68] Patrick N McGraw and Michael Menzinger. Topology and computational performance of attractor neural networks. *Physical Review E*, 68(4):047102, 2003.

[69] Jennifer Hasler and Harry Bo Marr. Finding a roadmap to achieve large neuromorphic hardware systems. *Frontiers in neuroscience*, 7:118, 2013.

[70] Dietrich Stauffer, Amnon Aharony, Luciano da Fontoura Costa, and Joan Adler. Efficient hopfield pattern recognition on a scale-free neural network. *The European Physical Journal B-Condensed Matter and Complex Systems*, 32(3):395–399, 2003.

[71] Gouhei Tanaka, Ryosho Nakane, Tomoya Takeuchi, Toshiyuki Yamane, Daiju Nakano, Yasunao Katayama, and Akira Hirose. Spatially arranged sparse recurrent neural networks for energy efficient associative memory. *IEEE transactions on neural networks and learning systems*, 2019.

[72] ROBERTJ McEliece, Edwardc Posner, Eugener Rodemich, and Santoshs Venkatesh. The capacity of the hopfield associative memory. *IEEE transactions on Information Theory*, 33(4):461–482, 1987.

[73] Amos Storkey. Increasing the capacity of a hopfield network without sacrificing functionality. In *International Conference on Artificial Neural Networks*, pages 451–456. Springer, 1997.

[74] Donald Olding Hebb. *The organization of behavior: a neuropsychological theory*. Science Editions, 1962.

[75] Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[76] Mohammed A. Zidan, YeonJoo Jeong, and Wei D. Lu. Temporal learning using second-order memristors. *IEEE Transactions on Nanotechnology*, 16(4):721–723, 2017.

[77] Zhong Sun et al. Solving matrix equations in one step with cross-point resistive arrays. *Proceedings of the National Academy of Sciences*, 116(10):4123–4128, 2019.

[78] Amy N Langville and Carl D Meyer. *Google's PageRank and beyond: The science of search engine rankings*. Princeton university press, 2011.

[79] Philipp Von Hilgers and Amy N Langville. The five greatest applications of markov chains. In *Proceedings of the Markov Anniversary Meeting*, pages 155–158. Citeseer, 2006.

[80] Wai-Ki Ching and Michael K Ng. Markov chains. *Models, algorithms and applications*, 2006.

[81] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical mathematics*, volume 37. Springer Science & Business Media, 2010.

[82] Bruno Sericola. *Markov chains: theory and applications*. John Wiley & Sons, 2013.

[83] Z. Sun, E. Ambrosi, G. Pedretti, A. Bricalli, and D. Ielmini. In-memory pagerank accelerator with a cross-point array of resistive memories. *IEEE Transactions on Electron Devices*, 67(4):1466–1470, 2020.

[84] Jeffrey J Hunter. Generalized inverses of markovian kernels in terms of properties of the markov chain. *Linear Algebra and its Applications*, 447:38–55, 2014.

[85] Carl D Meyer, Jr. The role of the group generalized inverse in the theory of finite markov chains. *Siam Review*, 17(3):443–464, 1975.

[86] X. Liu et al. Reno: A high-efficient reconfigurable neuromorphic computing accelerator design. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2015.

[87] A. Ankit et al. Panther: A programmable architecture for neural network training harnessing energy-efficient reram. *IEEE Transactions on Computers*, 69(8):1128–1142, 2020.

[88] Fan Zhang and Miao Hu. Memristor-based deep convolution neural network: A case study. *CoRR*, abs/1810.02225, 2018.

[89] G. Vasilescu. *Electronic Noise and Interfering Signals: Principles and Applications*. Signals and Communication Technology. Springer Berlin Heidelberg, 2006.

[90] B. Razavi. *Design of Analog CMOS Integrated Circuits*. McGraw-Hill Education, 2000.

[91] L. Sun, N. Zheng, T. Zhang, and P. Mazumder. Fault modeling and parallel testing for 1t1m memory array. *IEEE Transactions on Nanotechnology*, 17(3):437–451, 2018.

[92] C. Xu et al. Overcoming the challenges of crossbar resistive memory architectures. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pages 476–488, 2015.

[93] J. P. Strachan, A. C. Torrezan, F. Miao, M. D. Pickett, J. J. Yang, W. Yi, G. Medeiros-Ribeiro, and R. S. Williams. State dynamics and modeling of tantalum oxide memristors. *IEEE Transactions on Electron Devices*, 60(7):2194–2202, 2013.

[94] C. E. Graves, N. Davila, E. J. Merced-Grafals, S. Lam, J. P. Strachan, and R. S. Williams. Temperature and field-dependent transport measurements in continuously tunable tantalum oxide memristors expose the dominant state variable. *Appl. Phys. Lett.*, 110(12), 2017.

[95] E. J. Merced-Grafals, N. Davila, N. Ge, J. P. Strachan, and R. S. Williams. Repeatable, accurate, and high speed multi-level programming of memristor 1t1r arrays for power efficient analog computing applications. *Nanotechnology*, 27(36), 2016.

[96] P. Yao, H. Wu, B. Gao, et al. Fully hardware-implemented memristor convolutional neural network. *Nature*, 577:641–646, 2020.

[97] F. Alibart, L. Gao, B. D. Hoskins, and D. B. Strukov. High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm. *Nanotechnology*, 23(7), 2012.

[98] George Casella and Roger L Berger. *Statistical inference*, volume 2. Duxbury Pacific Grove, CA, 2002.

[99] C. Svensson and J. J. Wikner. Power consumption of analog circuits: A tutorial. *Analog Integrated Circuits and Signal Processing*, 65(2):171–184, 2010.

[100] B. Razavi. *Principles of data conversion system design*. IEEE Press, 1995.

[101] Alexander Dozortsev, Israel Goldshtein, and Shahar Kvatinsky. Analysis of the row grounding technique in a memristor-based crossbar array. *International Journal of Circuit Theory and Applications*, 46(1):122–137, 2018.

[102] Z. Sun and R. Huang. Time complexity of in-memory matrix-vector multiplication. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 68(8):2785–2789, 2021.

[103] Tie-Yan Liu. *Learning to rank for information retrieval*. Springer Science & Business Media, 2011.

[104] C. Moler. *Experiments with MATLAB*. MATLAB Central File Exchange, 2020.

[105] N. P. Jouppi et al. In-datacenter performance analysis of a tensor processing unit. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pages 1–12, 2017.