Memristor models for machine learning

Juan Pablo Carbajal, Joni Dambre, Michiel Hermans, and
Benjamin Schrauwen

Department of Electronics and Information Systems, Ghent
University, Belgium
{juanpablo.carbajal, michiel.hermans, benjamin.schrauwen,
joni.dambre}@ugent.be

July 15, 2014

Abstract

In the quest for alternatives to traditional CMOS, it is being suggested that digital computing efficiency and power can be improved by matching the precision to the application. Many applications do not need the high precision that is being used today. In particular, large gains in area- and power efficiency could be achieved by dedicated analog realizations of approximate computing engines. In this work, we explore the use of memristor networks for analog *approximate* computation, based on a machine learning framework called *reservoir computing*. Most experimental investigations on the dynamics of memristors focus on their nonvolatile behavior. Hence, the volatility that is present in the developed technologies is usually unwanted and it is not included in simulation models. In contrast, in reservoir computing, volatility is not only desirable but necessary. Therefore, in this work, we propose two different ways to incorporate it into memristor simulation models. The first is an extension of Strukov's model and the second is an equivalent Wiener model approximation. We analyze and compare the dynamical properties of these models and discuss their implications for the memory and the nonlinear processing capacity of memristor networks. Our results indicate that device variability, increasingly causing problems in traditional computer design, is an asset in the context of reservoir computing. We conclude that, although both models could lead to useful memristor based reservoir computing systems, their computational performance will differ. Therefore, experimental modeling research is required for the development of accurate volatile memristor models.

1 Introduction

As the scaling of the traditional MOSFET transistor is reaching its physical limits, alternative building blocks of future generation computers are being investigated. Most of this research is focused on producing controllable switch-like behavior, but over the last decade analog computation has also enjoyed a revival, partly due to the increasing success of neuromorphic devices. In particular,

1

arXiv:1406.2210v2 [cs.LG] 14 Jul 2014

memristor networks have been used for computational purposes, mostly using structured topologies, since their fabrication exploits technology developed for the assembly of cross-bar arrays of transistors [1; 2, section 5]. In most of these applications, memristors are used as programmable synaptic weights. However, at the nanoscale, even highly controlled processing flows cause relatively large variations in the device parameters. For this reason, computing systems built with such components must embrace this variability [1]. Increasingly, it is being suggested that efficiency and power gains can be achieved in digital computers by exploiting the fact that many applications do not need the high precision that is being used today. In this new research field of *approximate computing*, digital, low-precision neural network accelerators are currently being evaluated [3; 4].

Reservoir computing (RC) is a supervised learning framework rooted in recurrent neural network research (seminal work in [5; 6], recent developments in [7; 8]). It was originally applied to simulated neural networks in discrete time and can be implemented in open loop operation [9–13] or in feedback systems [14–16]. RC can also be used as a leverage to directly exploit the analog computation that naturally occurs in physical dynamical systems in a robust way, i.e. without the need to control all system parameters with high precision. Recently, this approach (*physical* RC or PRC) has been demonstrated (in simulation or experimentally) using several physical substrates. These include a water tank [17], tensegrity structures [18], opto-electronic and nano-photonic devices [15; 19–21] and resistive switch networks [13].

The computation in RC is based on the observed responses of a set of (interacting input-driven nonlinear) dynamical systems. At each point in time, the n observed responses are linearly combined, in order to optimally approximates a desired output. As is common in supervised machine learning approaches, the weights in this linear combination are optimized for a set of representative examples of the desired input-output behavior (i.e., the *task*). The system's responses to each of the input signals are sampled and the m samples recorded into a large $m \times n$ matrix called the *design matrix* $\boldsymbol{\Theta}$, which is usually used in a common linear regression setup:

$$\hat{\boldsymbol{y}} = \boldsymbol{\Theta}\boldsymbol{w} \quad \text{minimizing} \quad \text{dist}(y, \hat{y}), \tag{1}$$

where $\hat{y}$ is the (sampled) desired output. The readout vector $\boldsymbol{w}$ has n components, one for each observed state signal, and remains fixed after optimization (*training*). The aim is to optimize the readout weights in such a way that the system *generalizes* well to new data that has not been used for training. Note that a task is comprised of more than one desired output. The task can be described by comprehension, as in delay tasks or filtering tasks, in which case the desired outputs used to optimize (*train*) the readout weights are sampled from the set and the amount of training data is potentially unbounded (also known as synthetic data). However, as in many machine learning techniques, the task is such that no analytical or algorithmic solution is available. In this case, tasks is known only through a finite number of input-output examples.

In order to obtain useful results with PRC, the system's responses to the input signals must meet several requirements. First, reservoir computing strongly relies on a property of the input-output relation of the system generating the responses, namely *fading memory* (also known as the echo state property). It is

present when the output depends on previous inputs of the system in a decaying way, such that perturbations far in the past do not affect the current state. In this way, the functional relationship between the response of the system and the inputs is localized in time, providing a notion of "memory" [22]. Another role of fading memory is to avoid high sensitivity to initial conditions which, if present, could obscure the relation between inputs and outputs [see 23, and references therein].

Secondly, analog computation, and in particular RC, exploits the natural behavior of the device (the computer) to simulate a primary system. The class of computable primary systems is strictly related to the class of systems used as computer, e.g. both systems are modeled with the same class of differential equations [see "General-Purpose Analog Computation" in 24]. In RC this is reflected in the properties of the generated design matrix.

In general $m \gg n$ and the linear problem in Eq. (1) is rank-deficient or ill-posed [25]. The suitability of a dynamical system for reservoir computing depends on the extent to which input signals affect the system responses. In particular, the rank and range of the design matrix play a crucial role. Intuitively, the numerical rank r of Θ characterizes the power to reproduce features of the desired output. Indeed, in a geometrical interpretation, the responses in the design matrix span an r-dimensional subspace of all possible features [26]. Hence, for large ranks, one expects that the features required for a specific task are present within this subspace. The affinity between the task and the design matrix depends on the linear span of its singular vectors (the range of Θ). Highly complicated responses (e.g. characterized by randomness or by the chaotic dynamics of the underlying autonomous system), produce design matrices with high ranks but with extremely varied ranges. These matrices do not generalize well to unseen data, which leads to poor task performance. Moreover, if the responses do not significantly depend on the input history, they will not be very useful as features for obtaining a desired input-output relationship.

Whether or not a system can meet these requirements is for a large part determined by the system's inherent dynamics. Hence, to gauge what can be computed with a given system, it is crucial to understand its behavior. This allows us to propose encodings for the inputs and parameter values that optimize computation.

In this paper we revisit the implementation of RC using volatile resistive switches (for short we will call them memristors or memristive devices, despite the ongoing controversy [27]). This is a novel application of memristive devices as processing units rather than mere synaptic weights. Recent work has reported large random networks of memristors that can be fabricated with relative ease [28]. Their dynamical behavior indicates that they could be used as reservoirs [13; 29].

Most experimental investigations focus on the nonvolatile dynamics of memristors. Interestingly, volatility is present in the developed technologies [e.g. Fig. 1 of 30, see Fig. 1]. It can be observed that the conductance of the memristive device under study decays when there is no input, unless the conductive silver filament is fully formed. Additionally, the rate of decay depends on the stage of development of the filament: slow decay for incipient filaments and for those that are close to being fully formed. Fig. 1 sketches the state dependent decay rate. This behavior indicates a state dependent volatility with a maximum at some intermediate state of formation of the silver filament.
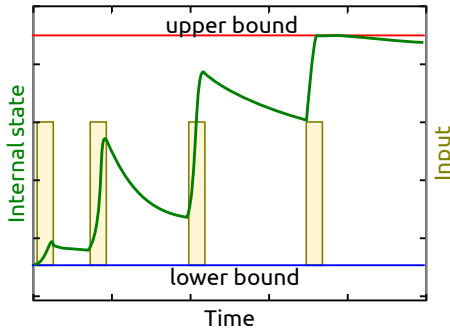
Figure 1: State dependent diffusion. Schematic reproduction of Fig. 1 of [30].

Due to the focus on ultra-dense storage (ReRAMs) applications, the current trend is to engineer volatility out of the devices. Consequently, reports of numerical models suitable for the simulation of volatile memristor networks are scarce [2, section 3.2]. The available mathematical models of memristors are adequate for the study of nonvolatile devices, i.e. having internal states without (or with negligible) autonomous dynamics.

To adequately study the potential use of memristor networks for RC, which requires fading memory, a deeper study of volatility is in order. For example, when working with memristors, volatility prevents saturation. This is key for RC, since a saturated memristor becomes a linear resistor that only scales the input. In this work we present a first study on models of memristors that focus on volatility and their dynamical response. These models are studied to provide a conceptual basis for the study of networks of these devices and their use in RC. The article begins with a short recapitulation of general models of memristive systems. Section 2.1 introduces the model proposed by Strukov [31] and its current driven solution, and proposes an equivalent nonvolatile Wiener model. Section 3 presents a simple modification of each model that introduces linear volatility. We focus on the production of harmonics and delays in the steady state response of these models, in the vein of the harmonic balance method. These aspects are important when designing the encoding to use for the inputs representing values in datasets. Section 4 describes RC using memristors connected in series and two simple application examples are given. Subsequent sections discuss the results and indicate future directions of research.

2 Memristive systems

Memristive systems are devices that can be modeled with a nonautonomous (input-driven) dynamical system of the form,

$$\dot{\boldsymbol{x}} = F(\boldsymbol{x}, u), \tag{2}$$

$$y = H(\boldsymbol{x}, u)u \tag{3}$$

where $\boldsymbol{x}$ is a vector of internal states, y a measured scalar quantity and u a scalar magnitude controlling the system. Due to its origins in electronics [32], the pair (y, u) is usually voltage-current or current-voltage. In the first case

("current controlled" system) the function $H(\boldsymbol{x}, u)$ is the called "memristance", and it is called "memductance" in the second case ("voltage controlled"). We use a general denomination for $H(\boldsymbol{x}, u)$ and call it *output function*. The output function is assumed to be continuous, bounded and of constant sign, leading to the zero-crossing property: $u = 0 \Rightarrow y = 0$. It has been shown that this property is a modeling deficiency for redox-based resistive switches [33], and an extended version of equation (3) was proposed, namely

$$y = H(\boldsymbol{x}, u)u + h(\boldsymbol{x}). \tag{4}$$

Where $h(\boldsymbol{x})$ is an autonomous output given by an internal battery in the system. This term, in general, removes the zero-crossing property making these systems incompatible with the original definition of memristor. In this work, we will not study these extended systems.

2.1 Memristor models

One of the most popular models of memristive behavior consists of a single internal state with linear dynamics and a piece-wise linear output function [31],

$$\dot{x} = \mu I(t), \tag{5}$$

$$H(x) = \begin{cases} R & \text{if } x < 0 \\ R - (R - r)x & \text{if } 0 \le x \le 1 \\ r & \text{if } x > 1 \end{cases}, \tag{6}$$

$$V(t) = H(x)I(t), \tag{7}$$

where I, V are the current and voltage, respectively and μ is a parameter related to the geometry and the physical properties of the material realizing the memristor. In particular it is proportional to the average ionic mobility. $H(x)$ plays the role of a state dependent resistance. Note that the value of x can grow unboundedly while $H(x)$ remains bounded. For this reason some authors consider the model valid only for $x \in [0, 1]$[31; 34, Eq. 31].

A modified version of this model applies a windowing function to the dynamics of the internal state, effectively bounding it [31; 35]

$$\begin{aligned} \dot{x} &= \mu x(1 - x)I(t), \\ H(x) &= R - \Delta r x, \\ V(t) &= H(x)I(t). \end{aligned} \tag{8}$$

The parameter $\Delta r = R - r$ depends on the bounds of the output function. The internal state is driven by a Bernoulli equation, hence we can obtain an explicit input-output relation for the current controlled memristor,

$$V(t) = \left[R - \Delta r \left(1 + \frac{1 - x_0}{x_0} e^{-\mu q(t)} \right)^{-1} \right] I(t), \tag{9}$$

where $q(t) = \int_0^t I(\tau)\, d\tau$.

Noteworthy, relation (9), corresponding to the nonlinear internal state dynamics (8) and linear output function, is indistinguishable from the relation corresponding to a system with linear internal state dynamics and sigmoidal output function, i.e. a Wiener model defined as

$$\dot{q} = \mu I(t),$$

$$H(q; x_0) = R - \Delta r \left(1 + \frac{1 - x_0}{x_0} e^{-q(t)} \right)^{-1}$$

$$= R - \frac{\Delta r}{2} \left[\tanh \left(\frac{q(t)}{2} + C(x_0) \right) + 1 \right]. \tag{10}$$

Note that this model is a smoothed version of the original Strukov model. The fact that the nonlinear model can be seen as a Wiener system imposes strong restrictions to the type of processing the system will be able to perform on the input signals. For example, its memory function will be restricted to exponential decays, as in linear systems. As will be discussed next, this equivalence is broken when an autonomous term is added to the dynamics. However, in Section 3.2 we will establish an approximated equivalence for small amplitude input signals, indicating that strong amplitudes will be needed to depart form a linear memory behavior.

3 Fading memory

We propose a modification of these models that includes an autonomous term,

$$\dot{x} = F(x, u) + D(x) \tag{11}$$

where for all x, we have that $F(x, 0) = 0$ and $D(x) < 0$. In the sections that follow we will restrict ourselves to the case when $D(x)$ is a linear function. Note the difference with Eq. (4). Since we are adding *diffusive* dynamics to the internal state, it does not change the zero-crossing property of the memristive system. This modification of the model in the context of resistance switching, arising from charge transport, implies that there is a flow of carriers that goes out of the electrical circuit of the device (e.g. additional sink) and it is not explicitly modeled. For example, in the hydraulic memristor studied in [34], the decay of the internal state is realized when the storage tank has a leak, that reduces the hydraulic resistance. Note that by doing this the methods to convert between current and voltage controlled devices described by Biolek [34] have to be extended, since these volatile models are not time invariant. As said before, the equivalence of the models (8) and (10) is lost.

3.1 Nonlinear dynamics

In the case of the nonlinear dynamics model (8) we write,

$$\dot{x} = \mu x(1 - x)I(t) - \lambda x = (\mu I(t) - \lambda)x - \mu I(t)x^2. \tag{12}$$

This equation can again be identified with the Bernoulli equation and the solution is (see Supplementary data for derivation)

$$x(t) = \left(1 + \frac{1 - x_0}{x_0} F(t)^{-1} + \lambda F(t)^{-1} \int_0^t F(z) \, \mathrm{d}z\right)^{-1} \tag{13}$$

$$F(t) = e^{\int_0^t \mu I(w) \, \mathrm{d}w - \lambda t}. \tag{14}$$

Comparing with Eq. (9), we see that λ introduces an exponentially decaying factor (linearly detrending the integral of the input) and adds the integral term. In the mathematical description that follows, we investigate the harmonics and delay generation in the steady response of the system. The presence of harmonics directly impacts the maximum rank that the RC design matrix can achieve. Similarly, as any delayed sinusoidal signal can be expressed as a superposition of the undelayed signal plus a cosine component of the same frequency, delays can also increase the maximum rank of the design matrix. We begin by expressing the convergence of the mean response (obtainable directly from the fixed point of Eq. (12)) and continue with the response generated by periodic input signals. We explicitly write the delay and harmonic generation in terms of interactions between parameters of the input signal and the system's parameters.

Using the first mean value theorem for integration when the input has mean value $\mu \bar{I} = m$ results in,

$$\bar{x}(t) = \left(1 + \frac{\lambda}{m - \lambda} + \frac{1 - x_0}{x_0} e^{(\lambda - m)t}\right)^{-1}, \tag{15}$$

which is nonzero in the long run only if $m > \lambda$. This average evolution has a pole at

$$t_c = \frac{1}{\lambda - m} \ln\left[\left(\frac{m}{\lambda - m}\right)\left(\frac{x_0}{1 - x_0}\right)\right]. \tag{16}$$

Taking $t = t_c + \tau$ the convergence to the steady state solution of Eq. (12) is given by

$$\bar{x}(t) = \frac{m - \lambda}{m}\left(1 - e^{(\lambda - m)\tau}\right)^{-1}, \tag{17}$$

which shows that all initial conditions converge to the same mean but their convergence time increases with decreasing initial conditions. In other words, a memristor with high resistance will take longer to converge to the steady state given by the mean value of the input. However, the timing properties of Eq. (14) do not depend on the initial condition. Therefore delays (and fading memory) depend only on the interaction between the input and the memristor parameters.

This interaction is crucial to determine how the system reacts to different inputs signals. We characterize two easily controllable properties of the inputs, namely, their mean value and their deviations from it. Deviations from the mean value can be seen as small amplitude oscillations. Therefore, to evince the interaction between system's parameters and the input signal, the latter is decomposed in its mean value and a zero-mean waveform:

$$\mu I(t) = m + \gamma(t), \tag{18}$$

$$\mu \int_0^t I(\tau) \, \mathrm{d}\tau = mt + \phi(t) + \phi_0, \tag{19}$$

where $\phi(t)$ is the primitive of $\gamma(t)$ and the initial values were all aggregated in the constant ϕ_0. We assume that both functions are bounded and proceed to calculate the harmonics generation and delay between input and output. For this family of inputs we get,

$$x(t)^{-1} = 1 + e^{-\phi_0} e^{-\phi(t)} \frac{1 - x_0}{x_0} e^{(\lambda - m)t} + \lambda \int_0^t e^{\phi(z) - \phi(t)} e^{(m-\lambda)(z-t)} \, \mathrm{d}z. \quad (20)$$

Assuming that $m = \lambda + \epsilon$ with $\epsilon > 0$, the second term vanishes in the long run. Additionally, if we assume small bounds for $\phi(t)$, i.e. $|\phi(z) - \phi(t)|$ small, we can expand the integrand obtaining,

$$\int_0^t e^{\phi(z) - \phi(t)} e^{-\epsilon(z-t)} \, \mathrm{d}z = \frac{1 - e^{-\epsilon t}}{\epsilon} + \sum_{n=1}^{\infty} \frac{1}{n!} \int_0^t (\phi(z) - \phi(t))^n \, e^{\epsilon(z-t)} \, \mathrm{d}z. \quad (21)$$

This expansion makes evident that the response of the system can be seen as the infinite superposition of the response of a linear system to all the homogeneous monomials in $\{\phi(z), \phi(t)\}$. Inserting this expansion back into Eq. (20), keeping terms of first order and removing vanishing terms we obtain,

$$x(t)^{-1} = \frac{\epsilon + \lambda}{\epsilon} + \lambda \int_0^t (\phi(z) - \phi(t)) \, e^{\epsilon(z-t)} \, \mathrm{d}z. \quad (22)$$

As a prototype of zero mean waveforms we take $\gamma(t) = \alpha \sin(\omega t)$ and proceed to specify the previous equations on inputs of this kind (this will provide the harmonic response of the memristor). Introducing this input into the previous equation we get,

$$x_\omega(t) = \left(\frac{\epsilon + \lambda}{\epsilon} - \frac{\lambda}{\epsilon} \frac{\alpha}{\sqrt{\omega^2 + \epsilon^2}} \sin(\omega t - \varphi) \right)^{-1} \quad (23)$$

$$= \frac{\epsilon}{m} \left(1 - \frac{\lambda}{m\omega} \alpha \sin(\varphi) \sin(\omega t - \varphi) \right)^{-1}, \quad (24)$$

where vanishing terms were removed (note $m = \epsilon + \lambda$) and the delay is given by

$$\sin(\varphi) = \frac{\omega}{\sqrt{\omega^2 + \epsilon^2}}. \quad (25)$$

Note that, as long as

$$\alpha < \frac{m\sqrt{\omega^2 + \epsilon^2}}{\lambda}, \quad (26)$$

we can further expand Eq. (24):

$$x_\omega(t) = \frac{\epsilon}{m} \left[1 + \sum_{n=1}^{\infty} \left(\frac{\lambda}{m\omega} \right)^n \alpha^n \big(\sin(\varphi) \sin(\omega t - \varphi) \big)^n \right]. \quad (27)$$

This result is a good approximation for small amplitudes of the inputs. For larger amplitudes, higher order terms in Eq. (21) are not negligible and harmonics of the input frequency increase their contribution.

The voltage drop across the memristor is obtained using Eq. (8) by multiplication with the input current and has three terms. The first term is the voltage drop across an effective resistor, and has the same frequency as the input:

$$\frac{rm + \Delta r\lambda}{\mu}\left(1 + \frac{\alpha}{m}\sin(\omega t)\right).$$ (28)

The second term is obtained by multiplication of the oscillatory part of $x_\omega(t)$ with the mean value of the input:

$$-\frac{\Delta r(m - \lambda)}{\mu}\sum_{n=1}^{\infty}\left(\frac{\lambda}{m\omega}\right)^n \alpha^n \big(\sin(\varphi)\sin(\omega t - \varphi)\big)^n.$$ (29)

The third term is formed by the product of the oscillatory parts of the input and $x_\omega(t)$,

$$-\frac{\Delta r(m - \lambda)}{m\mu}\sum_{n=1}^{\infty}\left(\frac{\lambda}{m\omega}\right)^n \alpha^{n+1}\big(\sin(\varphi)\sin(\omega t - \varphi)\big)^n \sin(\omega t).$$ (30)

The voltage drop is the sum of all these terms. Taking $n = 1$ and using trigonometric identities, the coefficients of the sine (a_ω) and cosine (b_ω) are:

$$a_\omega = \left(\frac{\alpha}{m}\right)\left(\frac{\Delta r\lambda}{\mu}\sin^2(\varphi) + r\right) = \left(\frac{\alpha}{m}\right)\left(\frac{\Delta r\lambda}{\mu}\frac{\omega^2}{\omega^2 + (m - \lambda)^2} + r\right),$$ (31)

$$b_\omega = \left(\frac{\alpha}{m}\right)\left(\frac{\Delta r\lambda}{\mu}\cos(\varphi)\sin(\varphi)\right) = \left(\frac{\alpha}{m}\right)\left(\frac{\Delta r\lambda}{\mu}\frac{\omega(m - \lambda)}{\omega^2 + (m - \lambda)^2}\right).$$ (32)

For the first harmonic 2ω we obtain:

$$a_{2\omega} = \frac{1}{2}\left(\frac{\alpha}{m}\right)^2\left(\frac{\Delta r\lambda}{\mu}\cos(\varphi)\sin(\varphi)\right) = \frac{1}{2}\left(\frac{\alpha}{m}\right)b_\omega,$$ (33)

$$b_{2\omega} = \frac{1}{2}\left(\frac{\alpha}{m}\right)^2\left(\frac{\Delta r\lambda}{\mu}\cos^2(\varphi)\right) = \frac{1}{2}\left(\frac{\alpha}{m}\right)^2\left(\frac{\Delta r\lambda}{\mu}\frac{(m - \lambda)^2}{\omega^2 + (m - \lambda)^2}\right).$$ (34)

The amplitude of the cosine contribution at the input frequency (and, as a consequence, the delay) is modulated by the ratio between the oscillation amplitudes and the mean value. This contribution is shown in the top-right panel of Figure 2. Additionally it depends on the frequency of the input signal and the difference $m - \lambda$. As was anticipated, the signal parameters play a mayor role in the response of the system and this can be used to propose "meaningful" encodings.

3.2   Wiener model

In the case of the Wiener model (10) we write,

$$\dot{z} = \mu I(t) - \lambda_\ell(z - z_s), \quad z_s < 0.$$ (35)

Where z_s is chosen such that the output function saturates to R when there are no inputs. Note that when $z_s \neq 0$, the diffusive term is not strictly negative as

requested in (11), the role of z_s is to prevent numerical overflows. The solution is readily obtained by integration:

$$z(t) = \mu \int_0^t I(\tau)e^{\lambda_\ell(\tau-t)} \, \mathrm{d}\tau + (z_0 - z_s)e^{-\lambda_\ell t} + z_s \tag{36}$$

and the timing properties are independent of the input characteristics. This makes the calculation of the harmonic response of the system trivial, so we use inputs of the form,

$$\mu I(t) = m + \sum_{i=1}^{N} \alpha_i \sin(\omega_i t). \tag{37}$$

Replacing in the corresponding equations we obtain:

$$z(t) = \overbrace{\frac{m}{\lambda_\ell}}^{\epsilon_\ell} + z_s + \overbrace{\sum_{i=1}^{N} \frac{\alpha_i}{\sqrt{\omega_i^2 + \lambda_\ell^2}} \sin(\omega_i t - \varphi_i)}^{q(t)}, \tag{38}$$

$$\sin(\varphi_i) = \frac{\omega_i}{\sqrt{\omega_i^2 + \lambda_\ell^2}}. \tag{39}$$

Since this is the response of a linear system, it generates no harmonics. Higher frequencies are introduced when the response becomes the argument of the sigmoid function in Eq. (10).

$$x(t)^{-1} = 1 + \frac{1 - x_0}{x_0} e^{-\epsilon_\ell} e^{-q(t)} =$$

$$1 + \frac{1 - x_0}{x_0} e^{-\epsilon_\ell} \left(1 + \sum_{n=1}^{\infty} \frac{(-1)^n}{n!} q(t)^n \right) \tag{40}$$

The first order gives,

$$x(t)^{-1} = 1 + \frac{1 - x_0}{x_0} e^{-\epsilon_\ell} \left[1 - \sum_{i=1}^{N} \frac{\alpha_i}{\omega_i} \sin(\varphi_i) \sin(\omega_i t - \varphi_i) \right]. \tag{41}$$

The case with a single frequency can be compared with Eq. (23). The equivalence between the first order models requires that (see Supplementary data for comparison of responses),

$$\begin{aligned} \lambda_\ell &= \epsilon \quad \text{(delay)} \\ z_s &= -\ln\left(\frac{x_0}{1 - x_0} \frac{\lambda}{\epsilon} \right) - \frac{m}{\lambda_\ell} \quad \text{(mean value).} \end{aligned} \tag{42}$$

This equalities introduce properties of the input into the system parameters. Therefore, this Wiener model is input signal dependent.

As mentioned before, the fact that we can establish an equivalence (approximated) between the nonlinear model and the Wiener model indicates that the

memory of the nonvolatile memristors will resemble that of a linear system. To observe memories that differ from exponential decays, high amplitude oscillations will be required. Note however, that the decay of conductance observed in experiments is not linear (see Fig. 1), possibly allowing for more complicated memory functions in a wider range of amplitudes.

4 Reservoir computing with memristors in series

In the RC context, the response of a set of memristors is used to assemble a design matrix which is used to linearly regress the desired output. A simple but restrictive approach is to ask that the responses are linearly combined instantaneously,

$$\hat{y}_k = \boldsymbol{\Theta}(t_k)^T \boldsymbol{w}. \tag{43}$$

Where t_k are the sampling timestamps, $\hat{y}_k$ is the approximation to the desired output y_k and $\boldsymbol{\Theta}(t_k)$ is a vector with the responses of the set. The mixing vector $\boldsymbol{w}$ is fixed for all t_k and has as many components as there are memristors in the system. For all timestamps we have

$$\hat{\boldsymbol{y}} = \boldsymbol{\Theta}\boldsymbol{w}, \tag{44}$$

$\boldsymbol{\Theta} \in \mathbb{R}^{T \times n}$ is the design matrix having the n responses of the set as columns. This approach will be used in section 4.1 to generate delayed versions of the input.

According to Eqs. (31)- (32) the voltage drop across n different memristors (different values of λ and μ) to a single frequency current of small amplitude (first order in I_ω) can provide at most a design matrix of rank 2 given by the sine and cosine components of the responses. For an input with N frequencies it is theoretically possible to obtain a maximum rank of $2N$, but probably this is unrealistic. The use of larger amplitudes can increase the rank due to the generation of higher harmonics. In this case, incommensurate frequencies in the input are the best option to maximize the rank, since they reduce the overlap of harmonics. Interestingly, the rank benefits from memristor variability, making it a desired feature of the building process instead of a nuisance [1, and reference 98 therein].

Figure 2 (left panel) shows a mockup setup of memristors connected in series. The input current is a zero mean signal $\gamma(t)$ and each memristor is fed a small local current I_{0i}. The case $I_{0i} = I_0$ for all i corresponds to a signal with mean value and no local current. The right panel of the figure shows the distribution of cosine components for independent mean values (top) and for a unique mean value (bottom). For simplicity of the presentation, the bottom panel assumes that all memristors have the same μ parameter. In general the cosine contribution will be controlled by $\mu_i I_0 - \lambda_i$, wider variety will, in general, improve the rank of the design matrix. In terms of control of the design matrix properties, the most versatile situation is the one with independent local inputs m_i. When this is not possible, variability of the memristor parameters is the key for successful RC. A wider variety responses to the same signal, i.e. parameter variability, would increase the rank of the design matrix.
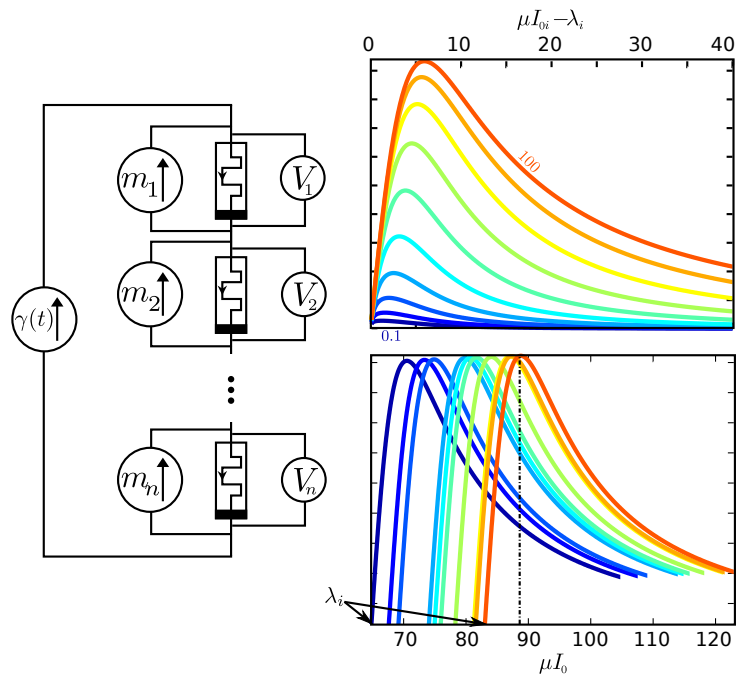
Figure 2: Memristors in series. (left) Set of memristors for simple signal processing. Each memristors is fed a constant current $m_i = \mu_i I_{0i}$. The right panel shows the cosine component (Eq. (32), arbitrary scale) for independent m_i (top) and for equal m_i (bottom). Labels on the curves indicate the value of λ_i. Independent m_i allow to tune $m_i - \lambda_i$ and the response of each memristor. When a single mean value is present (dotted line in bottom panel) the response depends on the natural distribution of the parameters.

Herein we assume that the diffusion parameter λ and the effective ionic mobility μ are independent parameters. The veracity of this assumption depends on the physical diffusive mechanism. If diffusion is coupled with carrier mobility, then λ will be proportional to μ. However, as in the case of the leaking hydraulic memristor this is not necessarily true in general.

4.1 Delayer

The delay task requires that the linear mixture of output voltages is able to recover a delayed version of the input signal. For the example presented here we have sampled 500 input signals defined by,

$$\gamma(t) = \alpha \sum_{n=1}^{12} \frac{\xi_n}{\pi n} \sin(\pi n t) - C. \tag{45}$$

With ξ_n independent Gaussian variables with zero mean and unit variance. The constant C removes the mean value of the signal for $t \in [0, 2]$. The memristor bank consisted of 50 units with $m_i - \lambda_i \in [0.1, 100]$ logarithmically spaced. The response of the system was calculated for 36 periods of the input. The last 12 periods were used for the training of the readout weights and a column filled with ones was added to the design matrix. These weights were obtained using ridge regression with 10-fold cross-validation.

The training performance of the system is shown in Fig. 3, top-left panel. The correlation coefficient interval is $[0.9, 1]$ indicating that the lower frequencies are more easily delayed (as indicated by Eq. (25)).

The trained weights were tested using a previously unseen input signal shown in the top-right panel. The deterioration of higher frequencies with increasing delay is also visible in these plots.

The panel at the bottom shows the distribution of the weights for each memristor (labeled with its ϵ_i parameter) and their variation with increasing delays. Higher delays recruit memristors with smaller ϵ_i. The integrals in Eq. (21) allows to interpret this in terms of memory, with lower ϵ_i corresponding to filters with longer time constants, i.e. longer memory. Note that volatility cannot be cancel out with a fixed mean value, since even if $\epsilon_i = 0$, diffusion is still driven by the mean value of the input with a quadratic dependency on the state (Eq. (12)). The smooth variation of the readout weights with the delay can be interpolated reducing the number of delays that need to be trained explicitly.

The numerical rank of the obtained design matrix in these experiments was approximately 5, with a clear gap between the singular values beyond that rank (discrete rank-deficient). Therefore, the size of memristor bank could be reduced. The size of the bank in this example was chosen to show the smooth variation of the readout weights as a function of ϵ_i. Similar performances as the one shown here were obtained with banks consisting of 10 memristors.

4.2 Binary operators

In this example we use data from the set $\mathbb{Z}^2/3 = \{0, 1, 2\}^2$ and we try to learn all mappings $\mathbb{Z}^2/3 \to \mathbb{Z}/3$, i.e. binary operators in $\mathbb{Z}/3$. Note that in the absence of noise, a design matrix with rank 9 would solve any of these tasks exactly. To
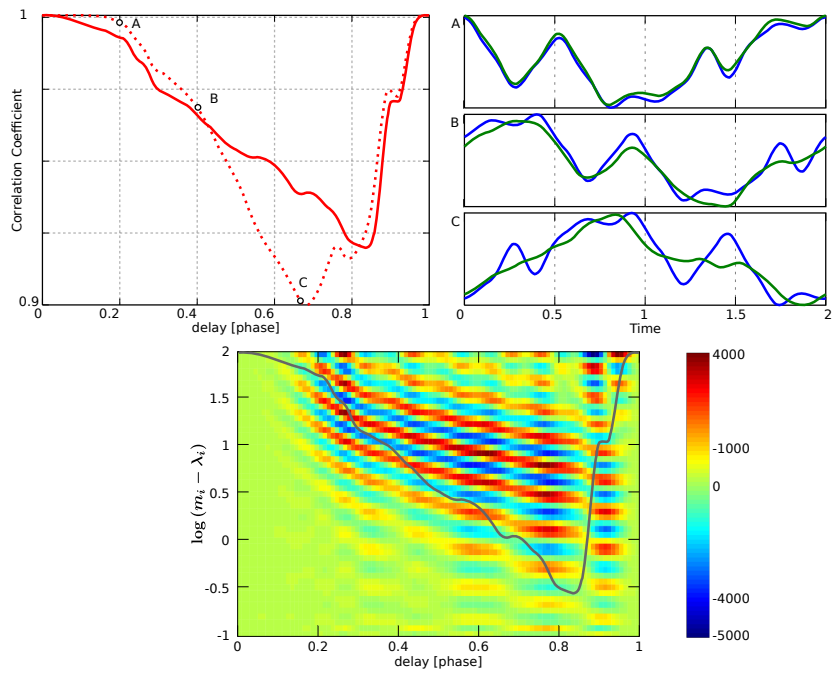
Figure 3: Signal delayer. Three panels showing the performance of the memristor bank for the delay task. (top-left) Correlation coefficient between the estimated and desired output, for different delays. The solid line corresponds to training values, the dotted line to a single test value. (top-right) Test example for three delays (A,B,C marked in the panel) high-frequency features of the signal are lost with higher delays. (bottom) Readout weights for each delay and ϵ_i parameter. The correlation coefficient is overlaid to guide the eye.

14
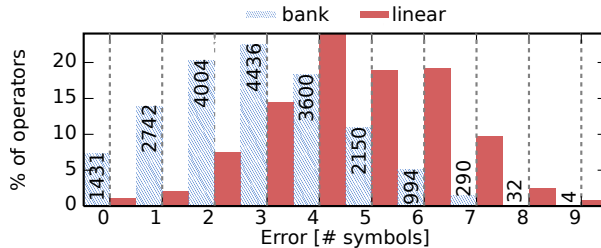
Figure 4: Solving all tasks in $\mathbb{Z}/3$. 10-fold cross validation error for all possible tasks in $\mathbb{Z}/3$ for a single 10 memristors bank. Each bar shows the tasks count and the y-axis shows the corresponding percentage. Pattern filled bars corresponds to the memristor bank, solid (red) bars are the results obtained using the inputs directly.

use the memristor bank with discrete datasets, we need an encoding from the domain of the dataset to the continuous input signal space. Each input pair $(s_1, s_2) \in \mathbb{Z}^2/3$ is mapped to an input signal via the formula,

$$u(t, s_1, s_2) = \frac{s_1 + 1}{3} \sin(\omega_1 t) + \frac{s_2 + 1}{3} \cos(\omega_2 t), \tag{46}$$

where $\omega_1 = \pi^2/\sqrt{2}$ and $\omega_2 = 3\pi\sqrt{3}$. These signals combine two (or more if desired) frequencies. The values s_i equalize the driving signal by defining frequency components within $u(t, s_1, s_2)$. Since we will be working with the long term response of the memristor bank, the ordering of the input pairs is irrelevant, i.e. the response of the system for symbols i is independent of the response for symbols $i - 1$. This also means that we are not exploiting the memory of the dynamical system. To do this, we would use encodings that present each element of the input pair sequentially. The encoding (46) allows us to use the bank as a static map between inputs and outputs similar to ELMs [36]. The same operators can be implemented in nonvolatile memristors, granted that all encoded inputs have zero mean.

For this application, the readout map is a linear combination of buffered output signals: for each input pair, we integrate the system for a period of time $T = {}^{20\pi}/\omega_1$ (sampling frequency $f = 150$ Hz) and sample the output voltages at timestamps $T - [0.95, 0.7, 0.5] \times {}^{2\pi}/\omega_1$. With this data we construct a single row of the design matrix. Therefore, the total number of readout weights is $3N$, with $N = 10$ the number of memristors in the bank.

Figure 4 shows the performance of the memristor bank for all possible binary operators in $\mathbb{Z}/3$. About 64% of all 3^{3^2} possible tasks are solved with 3 or less errors. For comparison, the plot also shows the performance of a design matrix constructed in the same way but using directly the input signals instead of the system's responses.

If we use random matrices as proxies for systems generating responses with very narrow autocorrelation functions, the performance of the memristor bank (64% with 3 or less errors) is obtained with random matrices of rank 7 and above (9×30 matrices with Gaussian entries and then SVD truncated to the desired rank). However the design matrix generated by the memristor rank has a numerical rank of 3 (1×10^{-2} tolerance). This indicates that the singular vectors are better suited for the task than the ones from random matrices.

5 Discussion

We have studied two models of volatile memristors that are reasonable modifications of popular models used nowadays for the study of nonvolatile devices. Model (8) has been used for the simulation of networks [37], while model (10) is an original contribution of the authors. Interestingly the nonlinear model of memristors can be thought of as a static nonlinearity applied on the trajectories of a linear dynamics model, i.e. a Wiener model. Though the equivalence between the models could be used for nonlinear system identification, its existence might hinder the usability of these systems in reservoir computing, since memory functions are restricted to decaying exponentials. A uniformly decaying memory function is unsuited for many machine learning problems, e.g. finding matching parentheses in a written sentence, which requires some kind of retrievable memory. However, the volatility observed in real devices [30] is more complex than the linear volatility studied herein. Nonlinear volatility might prevent an equivalent Wiener model. Additionally, memory can become intricate when the system is used in close loop.

The addition of linear volatility to resistance switching models based on charge transport, implies that there is a flow of carriers that goes out of the electrical circuit of the device. Here in this additional sink was not explicitly modeled. Improved models could include leak currents generated by diffusive processes and internal batteries, i.e. using Eqs. (4) and (11). Nevertheless, before moving towards more detailed models, we believe there is a need for thorough evaluation of the current models against the behavior of real devices.

When memristors are used in the RC framework, the natural variability of their parameters comes as a benefit contrary to what it is usually considered in other computational paradigms. The variability of the devices produces a gamma of responses (ideally, linearly independent responses) to the same input signal. This set can then be linearly combined to generate new functions. In this regard, RC is similar to Galerkin's method but the set of linearly independent functions used as generators are provided by the system itself rather than by design. This particularity makes the approach interesting for adaptive systems and has been used to generate motion control inputs for nonlinear systems [38, part II] and to draw connections to the synergy hypothesis in motor control [12].

Whether variations of the input can be afforded by the RC solution is highly dependent on the kind of system generating the responses and must be studied on a case by case basis. In general, systems showing fading memory for a family of driving signals are expected to be able to cope with input perturbations localized in time. This point highlights the role of the input encoding. The models studied here have a very well defined harmonic generation [35; 39] and encoding information in the spectrum (e.g. Eq. (46)) could exploit this natural behavior. However these memristor models are suited to produce only upshifts of frequency bands in the lower range of the spectrum. Another important feature of the model studied here is the coupling between the mean value of the inputs and the induced delay (and harmonic generation near the saturation). This indicates that low frequency variations are well suited for encodings. Note that this encoding would be much restricted in nonvolatile devices, since inputs with mean value would saturate the devices almost all the time.

The memristor bank presented in section 4 is the simplest topology that can be assembled with a set of memristors. It is a building block for the understand-

ing of larger topologies. It also offers a reference to evaluate the contribution of more complicated topologies. To study the complementary circuit, the parallel topology, we need to adapt the methods in [34] to include the time variant component of the dynamics. This will be done in subsequent works.

Regarding numerical simulations of large networks with the two models presented herein, there are several issues that are worth mentioning. For the case of the nonlinear state dynamics (Eq. (8)), $x = 0$ is unstable and any numerical error giving a slightly negative value will create a divergence of the simulations. To solve this, the value of x has to be forced to comply with $x > 0$ (slowing down the simulations) or one can build a nonlinear equation that is stable in both extrema of the interval $[0, 1]$. When $\lambda = 0$, once the internal state reaches $x = 0$ or $x = 1$ beyond machine precision, it cannot be driven out of those states since it becomes insensitive to the driving signal. This is an unavoidable problem when input signals have nonzero mean value. This is not the case for the model in Eq. (10). $H(x)$ can be driven out from saturation (in either extrema) using the appropriate input signal, even when numerical rounding errors are present. Note that model (8) provides a bounded internal state, while (10) does not. Since in the latter model saturation is present in the output function but not in the internal states, the integral of the input needed to "de-saturate" the internal state may grow with time and input signals with mean value could drive the internal states to overflow.

6   Conclusions

In this work we have studied a modified version of the nonlinear Strukov memristor model that includes a linear diffusion term. The long term behavior of the model and its response to periodic inputs have been presented in detail. The results evinced the role of the mean value of the inputs as a modulator of the memory of the memristors and the consequent generation of delays.

We have also presented a Wiener model that can approximate the behavior of the nonlinear model for inputs with small amplitude variations. This model can provide a starting point for nonlinear system identification of large networks. However, the existence of a Wiener model approximating the memristor behavior limits the complexity of its memory functions. Feedback loops would be required to overcome this drawback.

We have presented results on two academic tasks: delaying an input signal and binary operators in $\mathbb{Z}/3$. The delay task illustrated how RC recruits memristors with longer memories to generate higher delays. The binary operator task, showed that although the design matrices are rank-deficient their performance is comparable to high rank random matrices. The results also highlight that device variability is a desired feature for the implementation of RC in memristor networks.

Clearly, the different computational properties arising from the Strukov-based model and the Wiener model imply that a more in-depth study on the use of memristor networks for reservoir computing relies on the selection of a physically realistic model. Therefore, experimental modeling research is required for the development of accurate volatile memristor models.

Acknowledgements

References

[1] Indiveri G, Linares-Barranco B, Legenstein R, Deligeorgis G, Prodromakis T. Integration of nanoscale memristor synapses in neuromorphic computing architectures. Nanotechnology. 2013 Sep;24(38):384010.

[2] Kuzum D, Yu S, Wong HSP. Synaptic electronics: materials, devices and applications. Nanotechnology. 2013 Sep;24(38):382001.

[3] Esmaeilzadeh H, Sampson A, Ceze L, Burger D. Neural Acceleration for General-Purpose Approximate Programs. In: 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE; 2012. p. 449 – 460.

[4] Samadi M, Lee J, Jamshidi DA, Hormati A, Mahlke S. SAGE: self-tuning approximation for graphics engines. In: 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE; 2013. p. 13–24.

[5] Maass W, Natschläger T, Markram H. Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural computation. 2002 Nov;14(11):2531–60.

[6] Jaeger H, Haas H. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science (New York, NY). 2004 Apr;304(5667):78–80.

[7] Maass W. Liquid state machines: motivation, theory, and applications. In: Cooper SB, Sorbi A, editors. In Computability in Context: Computation and Logic in . . . . Imperial College Press; 2010. p. 275–296.

[8] Lukoševičius M, Jaeger H, Schrauwen B. Reservoir Computing Trends. KI - Künstliche Intelligenz. 2012 May;26(4):365–371.

[9] Triefenbach F, Jalalvand A, Schrauwen B, Martens JP. Phoneme recognition with large hierarchical reservoirs. In: Lafferty J, Williams CKI,

Shawe-Taylor J, Zemel RS, Culotta A, editors. Advances in Neural Information Processing Systems. vol. 23. Neural Information Processing System Foundation; 2010. p. 9.

[10] Buteneers P, Verstraeten D, Van Nieuwenhuyse B, Stroobandt D, Raedt R, Vonck K, et al. Real-time detection of epileptic seizures in animal models using reservoir computing. EPILEPSY RESEARCH. 2013;103(2-3):124–134.

[11] Ongenae F, Van Looy S, Verstraeten D, Verplancke T, Benoit D, De Turck F, et al. Time series classification for the prediction of dialysis in critically ill patients using echo state networks. ENGINEERING APPLICATIONS OF ARTIFICIAL INTELLIGENCE. 2013;26(3):984–996.

[12] Alessandro C, Carbajal JP, D'Avella A. A computational analysis of motor synergies by dynamic response decomposition. Front Comput Neurosci. 2013;.

[13] Sillin HO, Aguilera R, Shieh HH, Avizienis AV, Aono M, Stieg AZ, et al. A theoretical and experimental study of neuromorphic atomic switch networks for reservoir computing. Nanotechnology. 2013 Sep;24(38):384004.

[14] Reinhart RF, Jakob Steil J. Regularization and stability in reservoir networks with output feedback. Neurocomputing. 2012 Aug;90:96–105.

[15] Fiers M, Van Vaerenbergh T, wyffels F, Verstraeten D, Schrauwen B, Dambre J, et al. Nanophotonic reservoir computing with photonic crystal cavities to generate periodic patterns. IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS. 2014;25(2):344–355.

[16] wyffels F, Li J, Waegeman T, Schrauwen B, Jaeger H. Frequency modulation of large oscillatory neural networks. BIOLOGICAL CYBERNETICS. 2014;.

[17] Fernando C, Sojakka S. Pattern Recognition in a Bucket. In: Banzhaf W, Ziegler J, Christaller T, Dittrich P, Kim JT, editors. Advances in Artificial Life. Springer Berlin Heidelberg; 2003. p. 588–597.

[18] Caluwaerts K, D'Haene M, Verstraeten D, Schrauwen B. Locomotion without a brain: physical reservoir computing in tensegrity structures. ARTIFICIAL LIFE. 2013;19(1).

[19] Vandoorne K, Mechet P, Van Vaerenbergh T, Fiers M, Morthier G, Verstraeten D, et al. Experimental demonstration of reservoir computing on a silicon photonics chip. Nature Communications. 2014;5.

[20] Paquot Y, Duport F, Smerieri A, Dambre J, Schrauwen B, Haelterman M, et al. Optoelectronic reservoir computing. Scientific Reports. 2012;2:1–6.

[21] Larger L, Soriano MC, Brunner D, Appeltant L, Gutierrez JM, Pesquera L, et al. Photonic information processing beyond Turing: an optoelectronic implementation of reservoir computing. Optics Express. 2012;20(3):3241.

[22] Hermans M, Schrauwen B. Memory in linear recurrent neural networks in continuous time. Neural networks : the official journal of the International Neural Network Society. 2010 Apr;23(3):341–55.

[23] Manjunath G, Jaeger H. Echo state property linked to an input: exploring a fundamental characteristic of recurrent neural networks. Neural computation. 2013 Mar;25(3):671–96.

[24] MacLennan BJ. Analog Computation. In: Meyers RA, editor. Computational Complexity. New York: Springer; 2012. p. 161–184.

[25] Hansen PC. Rank-Deficient and Discrete Ill-Posed Problems. Society for Industrial and Applied Mathematics; 1998.

[26] Dambre J, Verstraeten D, Schrauwen B, Massar S. Information processing capacity of dynamical systems. Scientific reports. 2012 Jan;2:514.

[27] Vongehr S. Missing the Memristor. Advanced Science Letters. 2012 Oct;17(1):285–290.

[28] Avizienis AV, Sillin HO, Martin-Olmos C, Shieh HH, Aono M, Stieg AZ, et al. Neuromorphic atomic switch networks. PloS one. 2012 Jan;7(8):e42772.

[29] Stieg AZ, Avizienis AV, Sillin HO, Martin-olmos C, Aono M, Gimzewski JK. Emergent Criticality in Complex Turing B-Type Atomic Switch Networks. Advanced Materials. 2012;24(2):286–293.

[30] Ohno T, Hasegawa T, Nayak A, Tsuruoka T, Gimzewski JK, Aono M. Sensory and short-term memory formations observed in a Ag2S gap-type atomic switch. Applied Physics Letters. 2011;99(20):203108.

[31] Strukov DB, Snider GS, Stewart DR, Williams RS. The missing memristor found. Nature. 2008;453(7191):80–83.

[32] Chua LO. Memristor-the missing circuit element. IEEE Transactions on Circuit Theory. 1971;ct-18(5):507–519.

[33] Valov I, Linn E, Tappertzhofen S, Schmelzer S, van den Hurk J, Lentz F, et al. Nanobatteries in redox-based resistive switches require extension of memristor theory. Nature Communications. 2013 Apr;4:1771.

[34] Biolek Z, Biolek D, Biolkova V. Analytical Solution of Circuits Employing Voltage- and Current-Excited Memristors. IEEE Transactions on Circuits and Systems I: Regular Papers. 2012 Nov;59(11):2619–2628.

[35] Nedaaee Oskoee E, Sahimi M. Electric currents in networks of interconnected memristors. Physical Review E. 2011 Mar;83(3):031105.

[36] Huang GB, Zhu QY, Siew CK. Extreme learning machine: a new learning scheme of feedforward neural networks. In: Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on. vol. 2. IEEE; 2004. p. 985–990.

[37] Stieg AZ, Avizienis AV, Sillin HO, Aguilera R, Shieh Hh, Martin-olmos C, et al. Memristor Networks. Adamatzky A, Chua L, editors. Cham: Springer International Publishing; 2014.

[38] Carbajal JP. Harnessing Nonlinearities: Generating Behavior from Natural Dynamics [PhD]. University of Zürich; 2012.

[39] Georgiou PS, Barahona M, Yaliraki SN, Drakakis EM. Device Properties of Bernoulli Memristors. Proceedings of the IEEE. 2012 Jun;100(6):1938–1950.

[40] Octave community. GNU Octave 3.8.1; 2014. Available from: www.gnu.org/software/octave/.

[41] The Sage Development Team. Sage Mathematics Software (Version 6.1.1); 2014. Available from: http://www.sagemath.org.

[42] Inkscape community. Inkscape 0.48; 2014. Available from: http://www.inkscape.org/.