Computational capacity of *LRC*, memristive, and hybrid reservoirs

Forrest C. Sheldon [1,2,*] Artemy Kolchinsky [3] and Francesco Caravelli [1]

[1]*Theoretical Division and Center for Nonlinear Studies, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, USA*

[2]*London Institute for Mathematical Sciences, 21 Albemarle St., London W1S 4BS United Kingdom*

[3]*Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, New Mexico 87501, USA*

(Received 3 March 2022; accepted 26 September 2022; published 27 October 2022)

Reservoir computing is a machine learning paradigm that uses a high-dimensional dynamical system, or *reservoir*, to approximate and predict time series data. The scale, speed, and power usage of reservoir computers could be enhanced by constructing reservoirs out of electronic circuits, and several experimental studies have demonstrated promise in this direction. However, designing quality reservoirs requires a precise understanding of how such circuits process and store information. We analyze the feasibility and optimal design of electronic reservoirs that include both linear elements (resistors, inductors, and capacitors) and nonlinear memory elements called memristors. We provide analytic results regarding the feasibility of these reservoirs and give a systematic characterization of their computational properties by examining the types of input-output relationships that they can approximate. This allows us to design reservoirs with optimal properties. By introducing measures of the total linear and nonlinear computational capacities of the reservoir, we are able to design electronic circuits whose total computational capacity scales extensively with the system size. Our electronic reservoirs can match or exceed the performance of conventional "echo state network" reservoirs in a form that may be directly implemented in hardware.

DOI: 10.1103/PhysRevE.106.045310

I. INTRODUCTION

Reservoir computing (RC) [1–3] is a model for performing computations on time series data, which combines a high-dimensional driven dynamical system, called a *reservoir*, with a simple learning algorithm. Reservoir computing has proven to be a powerful tool in a wide variety of signal processing tasks, including forecasting [1], pattern generation and classification [4], adaptive filtering, and prediction of chaotic systems [5]. Recently, extensions to spatiotemporal chaotic systems [6] have proven to be surprisingly effective.

Central to the success of reservoir computation is the use of large dynamical systems to generate nonlinear transformations and store memories of the driving signal. This has generated interest in developing nanoscale electronic reservoirs with large numbers of elements [7], incorporating both linear components (such as resistors, inductors, and capacitors) and nonlinear components such as *memristors*. Memristors, or "resistors with memory," are nanoscale devices whose resistance depends on the past history of the current. The currents flowing through these devices cause a rearrangement of ions, leading to a persistent (but also reversible) change in resistance. Memristors offer the possibility of harnessing both nonlinear behavior and memory in electronic circuits. For this reason, specialized circuits composed of large numbers of memristors promise a new generation of computational hardware operating orders of magnitude faster, and at far lower power, than traditional digital circuitry [8–13].

Recently, several experimental works examining memristor-based electronic reservoirs have shown remarkable promise [14–20]. However, there is still no fundamental understanding of when electronic circuits can be successfully employed as reservoirs, what type of functions they can compute, and how to design electronic reservoirs with specific computational properties. In this paper, we address this gap by providing a systematic and analytical study of the computational capacity of electronic reservoirs composed of traditional linear elements and memristors.

In order to be feasible as a reservoir, a driven dynamical system must satisfy certain properties which guarantee that its state encodes an informative function of the driving signal; we establish feasibility for models of linear *LRC* (inductor-resistor-capacitor) and memristor reservoirs. We also characterize the input-output relationships natural to electronic reservoirs, in the process showing how memristors may be viewed as a source of nonlinear computations. We then demonstrate how to combine linear and nonlinear elements to achieve a specific computational task (that of approximating a second-order filter). Lastly, the capacity of the reservoir to perform useful computations should increase as the size (i.e., dimensionality) of the reservoir is increased. In particular, the main motivation for electronic reservoirs is the potential to achieve very large reservoir sizes, but increases in size are useless if they do not lead to improved computational capacities of the reservoir. Optimally, the number of linearly independent inputs that the reservoir can reconstruct should scale linearly with the reservoir size—i.e., to borrow a term from statistical physics, their reconstruction capabilities should be extensive

*Author to whom correspondence should be addressed: fs@lims.ac.uk; Current address: London Institute for Mathematical Sciences, 21 Albemarle St., London W1S 4BS, United Kingdom.

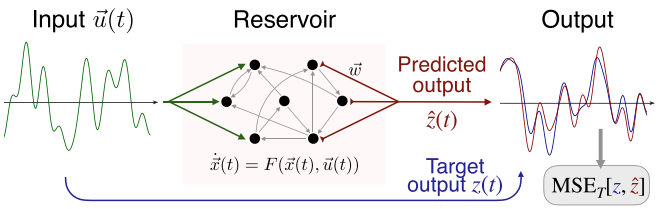

FIG. 1. Basics of reservoir computing. A time-dependent input $\vec{u}(t)$ is used to drive a reservoir, which is a (linear or nonlinear) dynamical system with internal state $\vec{x}(t)$. The predicted output is computed as a weighted linear combination of the internal state, $\hat{z}(t) = \vec{w}^T\vec{x}(t)$. The weights $\vec{w}$ are adjusted to minimize the time-averaged mean-squared error (MSE) between this predicted output and the target output $\vec{z}(t)$.

in reservoir size. For both *LRC* and memristor reservoirs, we consider measures of linear and nonlinear computational capacity, and show that they can be made to scale extensively. The approach we use to analyze the computational capacities of memristor and *LRC* reservoirs may be generalized to other reservoirs and computational tasks in analogy to the methods available to tune echo state networks (ESNs) by trading between memory storage and nonlinearity.

In the next section, we make these notions more precise. We provide an introduction to reservoir computing, electronic reservoirs, and memristors, as well as definitions of the measures of computational capacity that we will utilize. We then turn to the feasibility, tunability, and scalability of *LRC* and memristor-based reservoirs. Finally, we compare our results to more conventional ESN reservoirs [1], showing that electronic reservoirs are capable of matching or exceeding the performance of standard ESN reservoir implementations.

II. BACKGROUND

A. Reservoir computing

In what follows, we present a brief review of reservoir computing in continuous time. A schematic illustration is provided in Fig. 1. Readers familiar with RC and measures of capacity may skip to Sec. II C.

A *reservoir* is a multivariate driven dynamical system. At time t, the state of the dynamical system, which we indicate as $\vec{x}(t)$, is driven by an input $\vec{u}(t)$ and obeys the differential equation

$$\dot{\vec{x}}(t) = F(\vec{x}(t), \vec{u}(t)). \tag{1}$$

As a result of these dynamics, the state of the reservoir encodes information about $\vec{u}$ as transformations of its previous history. As an instructive example, a linear reservoir is governed by the equation

$$\dot{\vec{x}}(t) = A\vec{x}(t) + \vec{u}(t), \tag{2}$$

which has the long-time limit solution

$$\vec{x}(t) = \int_0^\infty d\tau e^{A\tau}\vec{u}(t-\tau), \tag{3}$$

showing that the state is a linear function of the past history of $\vec{u}$. We will consider reservoirs driven by a scalar input signal $u(t)$ as $\vec{u}(t) = \vec{v}u(t)$, where $\vec{v}$ is a vector of weights that defines how the input signal $u(t)$ enters into each element of the reservoir.

In order for a dynamical system to be considered *feasible* as a reservoir, its state must approach a nontrivial function of the input trajectory in the long-time limit. At a high level, we can state this requirement in terms of two conditions:

(i) *Fading memory*. If the system were to be started from two different initial conditions $\vec{x}_0 \neq \vec{x}'_0$ and driven with the same input trajectory u, the system's trajectories should eventually converge to the same state, $\vec{x}(t), \vec{x}'(t) \to \vec{x}[u](t)$ as $t \to \infty$. The statement above implies that the system has a finite temporal memory.

(ii) *State separation*. Different input sequences should drive the system into different trajectories, i.e., if the same initial condition were to be driven with two different input trajectories $u \neq u'$, the resulting reservoir trajectories must be sufficiently different.

The first condition requires that the state of the reservoir becomes a function of the input trajectory, while the second requires that this function carries information about the input trajectory.

Reservoirs that satisfy the fading memory property have a representation as a Wiener-Volterra series [21],

$$x_i[u](t) = \int_0^\infty d\tau_1\, h_{i1}(\tau_1)u(t-\tau_1) + \int_0^\infty d\tau_1 d\tau_2\, h_{i2}(\tau_1,\tau_2)u(t-\tau_1)u(t-\tau_2) + \cdots, \tag{4}$$

which decomposes each degree of freedom x_i into a series of linear and nonlinear components governed by kernel functions $h_{in}(\tau_1, \ldots \tau_n)$. This allows us to regard the reservoir as implementing a *filter* of the input trajectory. Reservoir computers can be thought of as approximating filters, in much the same way that feed-forward neural networks may be thought of as approximating functions [22,23]. This will be a useful characterization of both the reservoir and its possible outputs.

In addition to the input trajectory $u(t)$, we are also provided with a target output trajectory $z(t)$. The goal of reservoir computing is to learn to approximate the input-output mapping $u \mapsto z$ with an estimate $\hat{z}(t)$, which is a linear combination of the reservoir's variables, $\hat{z}(t) = \vec{w}^T\vec{x}(t)$. Here we will always assume that the $z(t)$ is a scalar. We will also assume that the target output is a function of u of the form in Eq. (4), denoted as $z[u]$. See Fig. 1 for details. Finally, as is often done, to the reservoir trajectories $\vec{x}$ we append a constant signal, $\vec{x}'(t) := [\vec{x}(t), \vec{1}(t)]$, to compensate for constant shifts in the output $z[u]$.

In the following, all quantities depend on the particular input signal $u = \{u(t) : 0 \leqslant t \leqslant T\}$ used to drive the reservoir and generate $\vec{x}[u](t)$, $\hat{z}[u](t)$, and $z[u](t)$. This dependence on u can be cumbersome to denote and so we have suppressed it in favor of indicating the dependence on the time interval by the subscript $\square_T$.

The interesting feature of RC is that only the output layer, given by the coefficients $\vec{w}$, is trained. Training is performed in the following manner: First, the reservoir is "initialized" by driving with the input signal on an interval $[-T', 0]$, until the fading memory property ensures that its state is independent

of the initial condition at $t = -T'$. Then, the reservoir is driven for an additional interval $[0, T]$. The coefficients $\vec{w}$ are learned via linear regression, by minimizing the time-averaged mean-squared error (MSE) between a reconstruction $\hat{z}(t) = \vec{w}^T \vec{x}'(t)$ and the target output trajectory $z(t)$ over the time interval $[0, T]$,

$$\mathrm{MSE}_T[z, \hat{z}] = \frac{1}{T}\int_0^T dt\,[z(t) - \hat{z}(t)]^2. \tag{5}$$

This optimization problem $\hat{\vec{w}} = \mathrm{argmin}_{\vec{w}}\,\mathrm{MSE}_T[z, \vec{w}^T\vec{x}']$ has a closed-form solution (note the use of the "hat" to denote the optimum). Defining the time average, $\langle f\rangle_T = \frac{1}{T}\int_0^T dt\, f(t)$, the solution is $\hat{\vec{w}} = \langle \vec{x}'^T\vec{x}'\rangle_T^{-1}\langle \vec{x}' z\rangle_T$, which we show in the Supplemental Material [24]. Regularization via ridge regression is also commonly employed in practice, which modifies the objective to $\mathrm{MSE}_T[z, \vec{w}^T\vec{x}'] + k||\vec{w}||^2$ [3].

In the space of functions on $[0, T]$, the optimal approximation $\hat{z}(t) = \hat{\vec{w}}^T\vec{x}'(t)$ can be understood as the projection of $z(t)$ onto the span of the reservoir trajectories $\vec{x}'(t)$. This is a consequence of the least-squares objective [25]. As the estimate $\hat{z}(t)$ is a projection of $z(t)$, we can evaluate the normalized mean-squared error of the reservoir reconstruction of z as

$$\mathrm{nMSE}_T[z, \hat{z}] = \frac{\mathrm{MSE}_T[z, \hat{z}]}{\langle z^2\rangle_T}. \tag{6}$$

It is normalized to $0 \leqslant \mathrm{nMSE}_T[z] \leqslant 1$ when calculated on the training interval $t \in [0, T]$. The nMSE is often a more useful measure than the MSE since it gives the *relative* error of the approximation to the variation in $z(t)$.

From now on, we will only consider the optimal approximation generated by the reservoir. We introduce the notation

$$\mathrm{nMSE}_T[z] \equiv \min_{\vec{w}} \mathrm{nMSE}_T[z, \vec{w}^T\vec{x}]. \tag{7}$$

$\mathrm{nMSE}_T[z]$ can be read as *the normalized mean-squared error for the reservoir's approximation of z on the time interval* $[0, T]$. Some authors [25] prefer to use a reversed scale, defining the *capacity* of the reservoir to approximate z as

$$C_T[z] = 1 - \mathrm{nMSE}_T[z], \tag{8}$$

where $\langle f\rangle_T = \frac{1}{T}\int_0^T dt\, f(t)$. The capacity is bounded $0 \leqslant C_T[z] \leqslant 1$, with 1 corresponding to a perfect reconstruction.

As mentioned above, the nMSE and capacity C_T can be given a geometric interpretation on the space of functions on $[0, T]$. For a function $z(t)$ normalized as $\langle z^2\rangle_T = 1$, the capacity $C_T[z]$ is the squared length of the projection of z onto the span of the reservoir trajectories $\vec{x}'$. The nMSE$[z]$ is the squared length of the component of $z(t)$ perpendicular to the span of the reservoir trajectories and so $C_T[z] + \mathrm{nMSE}[z] = 1$ can be seen as an expression of the Pythagorean Theorem.

B. Total memory and extensivity

In order to assess a reservoir, we require a more complete picture of its properties, that goes beyond its ability to approximate a single function. A main result of Dambre *et al.* [25] is that the capacities of a reservoir to approximate orthogonal functions give independent information about the reservoir (orthogonality is defined as $\lim_{T\to\infty}\langle zz'\rangle_T \to 0$; see Supplemental Material [24] for further details).

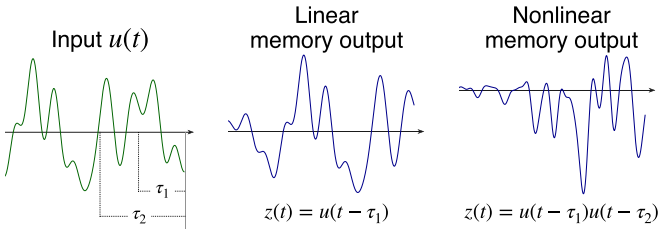

FIG. 2. The first two terms of the Wiener-Volterra series, given by Eq. (4), are linear and quadratic functions of the input. When taken as the target output z, these functions are used to define the linear and nonlinear memory of the reservoirs, as in Eqs. (10) and (11).

One consequence of this result is that in the long-time limit, capacities will converge to time-independent values that characterize the reservoir,

$$\lim_{T\to\infty} C_T[z] = C[z]. \tag{9}$$

While we cannot achieve this limit in practice, we can estimate it using techniques from finite-size scaling. (The capacities we report throughout this paper are obtained from these techniques, with detailed results of the analysis shown in the Supplemental Material [24].)

Another consequence of the result by Dambre is that rather than evaluating the reservoir's capacity to approximate only a single function z, we can evaluate its capacity over a family of orthogonal functions $\{z_i\}$. The resulting capacities allow us to characterize the reservoir's ability to approximate arbitrary functions of the form $\sum_i a_i z_i$. We relate the capacities of the reservoir on a family of functions, $C_T[z_i]$, to a linear combination of these functions, $C_T[\sum_i a_i z_i]$, in a precise way: *Consider a set of n orthonormal functions on $t \in [0, T]$, $\langle z_i z_j\rangle_T = \delta_{ij}$, and normalized linear combination $z = \sum_{i=1}^n a_i z_i$, $\langle z^2\rangle_T = 1$. Then, if $C_T[z_i] \geqslant 1 - \epsilon$ for all $i = 1 \ldots n$, then $C_T[z] \geqslant 1 - n\epsilon$.* This is proved in the Supplemental Material [24], where we also construct the function z^* that has maximal error. Our construction shows that when errors are uncorrelated or when the maximal error function lies within the basis, the capacity on the linear combination satisfies the tighter bound $C_T[z^*] \geqslant 1 - \epsilon$.

A natural family of functions for evaluating capacities are products of the delayed input $z(t) = u(t - \tau_1)\ldots u(t - \tau_n)$, which are the terms in the Wiener-Volterra series in Eq. (4). The first two such functions are shown schematically in Fig. 2. The first function, $z(t) = u(t - \tau_1)$, leads to the *linear memory function*,

$$m(\tau) = C[u(t - \tau)], \tag{10}$$

which reflects the ability of the reservoir to reconstruct a time-delayed version of the input at different lags [2,26,27]. We generalize this concept to nonlinear, nth-order memory functions,

$$m_n(\tau_1, \tau_2, \ldots, \tau_n) = C[u(t - \tau_1)u(t - \tau_2)\ldots u(t - \tau_n)], \tag{11}$$

which reflect the reservoir's ability to approximate the product of the input at various delays in the past. Achieving high nonlinear memory requires the reservoir to memorize past inputs as well as to be able to perform nonlinear operations on these inputs.

If the input function u is drawn from a family of random functions with mean zero $\langle u \rangle_T = 0$ and decaying autocorrelation $\langle u(t)u(t-\tau)\rangle_T \to 0$ as $\tau \to \infty$, then u and its time-delayed products will approach orthogonality as their delays differ. We generate input functions from smoothed Gaussian noise with correlations that decay exponentially with a unit timescale (see Supplemental Material [24]). The family of functions $\{u(t-\tau_1),\ u(t-\tau_1)u(t-\tau_2),\ u(t-\tau_1)u(t-\tau_2)u(t-\tau_3), \dots\}$ is approximately orthogonal for time delays satisfying $|\tau_i - \tau_j| > 1$. The memory functions m_n thus tell us independent information about the computational capacities of our reservoir over timescales greater than 1.

Finally, we summarize a reservoir's computational properties via the following quantity:

$$\tau_\epsilon = \int_0^\infty d\tau\ \Theta(m(\tau) > 1-\epsilon), \tag{12}$$

where Θ is the Heaviside step function. We refer to τ_ϵ as the *total linear memory* of the reservoir. This quantity captures the time delay up to which the history of the input is reconstructed with an error less than ϵ. This definition can be generalized to quantify the *total* nth*-order nonlinear memory* as

$$\tau_\epsilon^{(n)} = \int_0^\infty \cdots \int_0^\infty d\tau_1 \cdots d\tau_n\ \Theta[m_n(\tau_1, \dots, \tau_n) > 1-\epsilon]. \tag{13}$$

Previous studies have demonstrated linear reservoirs with extensive total memory, i.e., $\tau_\epsilon \propto N$ for a reservoir with N elements [26,27]. In this work, we present an electronic implementation of the linear reservoirs discussed in those papers. We then generalize this approach by designing an electronic reservoir which displays extensive scaling in its total quadratic memory, $\tau_\epsilon^{(2)} \propto N$.

C. Circuit elements and structure

We consider electronic circuits composed of traditional linear elements including inductors (L), capacitors (C), and resistors (R), active elements (voltage or current sources), as well as passive memory elements known as memristors (Mem) (see below). In all cases, the electronic reservoir will accept a vector input through a set of voltage sources $\vec{s}$. We can convert the scalar input $u(t)$ to a time-dependent voltage vector $\vec{s}(t)$ through a set of constant input weights $\vec{s}(t) = \vec{v}u(t)$. We will consider several circuit structures as shown in Figs. 3 and 4, including independent subcircuit reservoirs composed of *LRC*, single, and paired memristor circuits and memristive networks on a lattice.

The linear reservoirs that we consider will be composed of sets of *LRC* subcircuits as shown in Fig. 3 in which each *LRC* circuit, indexed by n, has component values l_n, r_n, c_n and is driven by a voltage generator $s_n = u(t)$ (taking $\vec{v} = \vec{1}$). Each circuit possesses two degrees of freedom $q_n(t)$, $\dot{q}_n(t)$ corresponding to the charge across and current entering the

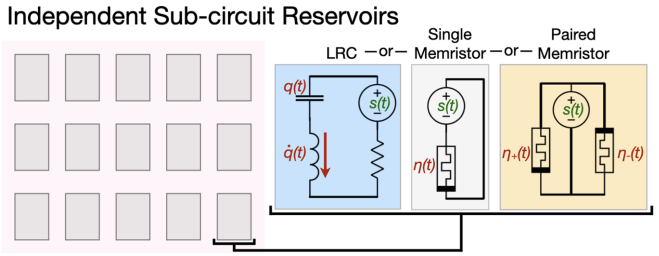

FIG. 3. A class of reservoirs that we consider are composed of independent subcircuits. Each circuit has an identical structure, but with parameters that vary to achieve independent trajectories when driven with the same input. In all subcircuits, the input is a voltage generator driven with $s(t)$ and shown in green. The output circuit variables are shown in red. The three subcircuit types are as follows: (i) *LRC* circuits with output variables $q(t)$, the charge on the capacitor and $\dot{q}(t)$, the current through the inductor, (ii) single memristors with output variable $\eta(t)$, the memristive state variable, and (iii) paired memristors biased in opposite directions with output variables $\eta_+(t)$ and $\eta_-(t)$. As the resistance is a linear function of η, the two are equivalent as output variables.

capacitor, which obey the following equations of motion:

$$\begin{bmatrix} \dot{q}_n(t) \\ \ddot{q}_n(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{1}{l_n c_n} & -\frac{r_n}{l_n} \end{bmatrix} \begin{bmatrix} q_n(t) \\ \dot{q}_n(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{s_n(t)}{l_n} \end{bmatrix}. \tag{14}$$

The output trajectories of an *LRC* subcircuit are the trajectories of the internal degrees of freedom, $\vec{x} = [\vec{q}(t), \dot{\vec{q}}(t)]$ (the vector notation covers the indexing over n). The dimension of an *LRC* reservoir of N subcircuits is thus $2N$. In the Supplemental Material [24], we prove that a network of *LRC* motifs is equivalent to a collection of separate subcircuits. There is thus no benefit to considering a network of these motifs and our treatment here is fully general.

In addition to linear elements, we consider reservoirs of nonlinear electronic components called memristors. Memristors are passive two-terminal devices characterized by the response relationship,

$$V(t) = R(\eta)I(t), \tag{15}$$

$$\dot{\eta}(t) = f(\eta(t), I(t)), \tag{16}$$

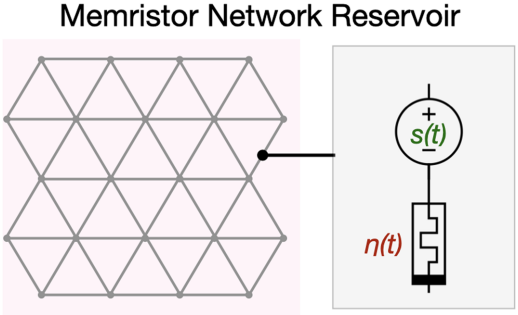

FIG. 4. The edges of the memristor networks that we consider consist of voltage-driven memristors which are arranged in a triangular lattice. The output variables are the memristive states in the networks $\vec{\eta}(t) = [\eta_1(t), \eta_2(t), \dots]$. The magnitude of the driving signal in each edge $s_e(t)$ varies on the interval $[S, -S]$ in equally spaced increments.

where $V(t)$ is the voltage drop across the memristor, $I(t)$ is current, $\eta(t)$ is the internal state of the memristor, and $R(\eta(t))$ is the state-dependent resistance. It can be seen that the resistance can depend on the past history of the current. Importantly, memristors are inherently nonlinear elements. As in the case of linear networks, the input to the circuits is through a set of voltage generators $\vec{s}(t) = \vec{v}u(t)$, where $\vec{v}$ is a constant vector of weights with units of voltage.

The internal states of a memristor circuit closely mimic the behavior of a neural network. We constrain ourselves to the linear current model similar to the one proposed in [11], along with a decay term [28],

$$R(\eta) = R_{\text{off}}(1-\eta) + R_{\text{on}}\eta, \tag{17}$$

$$R(\eta) = R_{\text{off}}(1-\chi\eta) \quad \left(\chi := \frac{R_{\text{off}} - R_{\text{on}}}{R_{\text{off}}}\right), \tag{18}$$

$$\dot{\eta}(t) = -\alpha\eta(t) + \frac{R_{\text{off}}}{\beta} I(t). \tag{19}$$

Here the constant $\alpha = 1/t^*$ is an inverse timescale, while β is an activation current per unit of time, which moderates the strength of the input signal. This model is the simplest approximation of a current-controlled memristor, as we show in the Supplemental Material [24]. The internal state η is limited to the interval $[0, 1]$ by hard barriers, so the resistance $R(\eta)$ varies between two limiting values $[R_{\text{on}}, R_{\text{off}}]$. We use η as the output variable of memristors, but as the resistance R is a linear function of η, the two are equivalent as components of a linear regression.

The term $-\alpha\eta(t)$ in Eq. (19) causes the memristor state η to decay to 0 in the absence of a current. This is called volatility in the context of memristors and is an important effect in many memory materials and their applications (see the review [29]). Here it plays a central role in providing the fading memory property for memristor circuits. However, not all memristors are volatile and commercial memristors are generally designed to be nonvolatile for memory applications. While nonvolatile memristors can still show fading memory (see [30,31]), nonvolatile fading memory is much more difficult to treat analytically and so we limit our study to the case of volatile memristors. This is not as great a limitation as it might seem, in the sense that any nonvolatile memristor can be turned into a volatile memristor by the addition of a voltage or current source that gives a small negative current in the absence of an applied voltage.

When considering networks of elements as in Fig. 4, it is convenient to introduce the cycle space projector Ω_A [28], which projects a vector that assigns a real number to each edge of a graph onto current configurations that satisfy Kirchhoff's current law. The projector Ω_A can be simply computed from the circuit graph $\mathcal{G}$, in which nodes of the graph represent electrical junctures and edges contain electrical elements. For a circuit in which the edges contain a voltage generator in series with a memristor with values $\vec{u}(t)$, as in Fig. 4, the equation of motion is given by [28]

$$\dot{\vec{\eta}}(t) = -\alpha\vec{\eta}(t) + \frac{1}{\beta}[I - \chi\Omega_A H(t)]^{-1}\Omega_A\vec{u}(t), \tag{20}$$

where we have used the convention $H(t) = \text{diag}\,\vec{\eta}(t)$. Interactions between memristors thus occur through the inverse of $I - \chi\Omega_A H(t)$ and are mediated both by χ and the Kirchhoff laws imposed by Ω_A. A memristor circuit generates reservoir trajectories $\vec{x} = \vec{\eta}(t)$.

III. RESULTS

A. Linear electronic reservoirs

In this section, we illustrate the ideas of memory and scaling introduced above in the familiar realm of linear circuits. We show that electronic reservoirs with extensive total memory can be constructed from *LRC* circuits and that these circuits can be understood as calculating a time-windowed version of the Fourier transform, akin to a spectrogram. Moreover, in the Supplemental Material [24], we prove the following result:

Feasibility of LRC circuits. Reservoirs of *LRC* circuit motifs satisfy the fading memory and state separation properties.

This justifies our use of these systems as reservoirs and the application of the memory measures defined above.

From the solution in Eq. (3), we observe that linear reservoirs generate linear functions of the input signal, and that their properties will depend on the eigenvalues of A. We thus examine the scaling of the total linear memory τ_ϵ using spectral techniques. In [26,27], it was noted that an extensive total memory was obtained for reservoirs with eigenvalues lying on a vertical line in the negative half plane, i.e., eigenvalues of the form $\lambda_{n,\pm} = -\gamma \pm in\Delta\omega$ for $n \in \{0 \ldots N\}$. As shown in the Supplemental Material [24], the solution to Eq. (14) depends on linear combinations of the integrals $\int_0^\infty d\tau\, e^{\lambda_{n,\pm}\tau} u(t-\tau)$. For the values of $\lambda_{n,\pm}$ above, this can be interpreted as calculating a local Fourier transform of the input. The frequencies $\text{Im}(\lambda_{n,\pm}) = \pm n\Delta\omega$ are evenly spaced with largest frequency $N\Delta\omega$ and smallest frequency $\Delta\omega$. The exponential window $e^{-\gamma\tau}$ applies a cutoff in time on the interval $0 \leqslant \tau \leqslant 1/\gamma$. This means that in order to not experience interference with previous time windows, we should set the lowest frequency $\Delta\omega$ to be on the same order as γ. Further details of this correspondence are given in the Supplemental Material [24] (Sec. VIII, Solution of *LRC* circuits).

The *LRC* subcircuit in Fig. 3 and governed by Eq. (14) has eigenvalues $\lambda_\pm = -\frac{r}{2l} \pm i\sqrt{\frac{1}{lc} - \frac{r^2}{4l^2}} = -\gamma \pm i\omega$ and a corresponding pair of trajectories, $q_n(t)$, $\dot{q}_n(t)$, corresponding to the charge and current entering the capacitor (see Supplemental Material [24]). As a consequence, *any eigenvalue spectrum in the negative half plane and symmetric in the upper and lower half planes can be achieved by a collection of LRC circuits with a particular choice of the component values* l_n, r_n, c_n. Given a γ and $\Delta\omega$, we choose $l_n = 1$, $r_n = 2\gamma$ for all n and

$$c_n = \frac{1}{n^2\Delta\omega^2 + \gamma^2}, \tag{21}$$

in which case the resulting *LRC* circuits have eigenvalues $\lambda_{n,\pm} = -\gamma \pm in\Delta\omega$.

Reconstructing the input at a given time lag, $z(t) = u(t - \tau_1)$, is equivalent to constructing an approximate representation of the δ function as the linear term of Eq. (4), $h_1(\tau) = \delta(\tau_1 - \tau)$. After fitting, the learned weights w_{q_n}, $w_{\dot{q}_n}$, and w_c

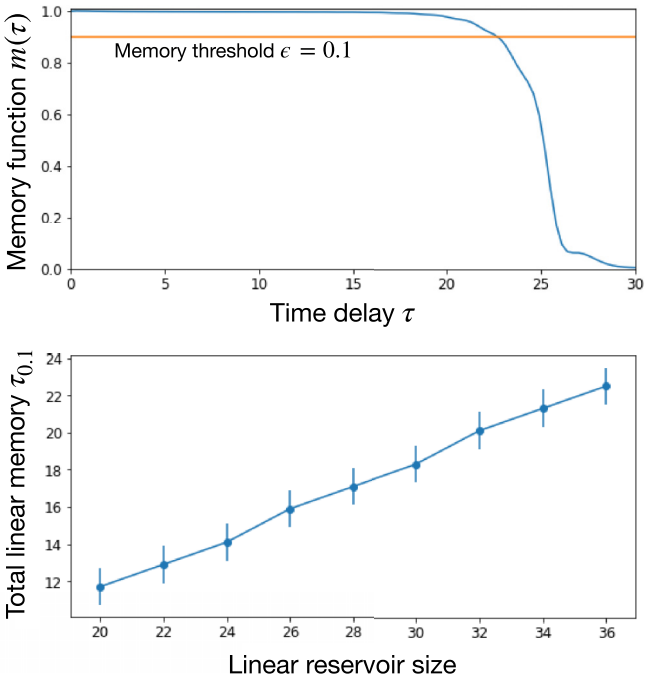

FIG. 5. The top panel displays the memory function for an *LRC* reservoir of 18 subcircuits (36 trajectories) with $\Delta\omega = \gamma = \frac{4}{18}$. The memory function $m(\tau)$ displays the capacity of the reservoir to reconstruct the delayed input $u(t-\tau)$. The reservoir retains an accurate memory of the input before falling sharply. In the bottom panel, we display the $\epsilon = 0.1$ total linear memory of *LRC* networks of various sizes. This corresponds to the signal delay at which the memory function falls below 0.9, displayed in the top panel as a horizontal line. As discussed in the main text, the total linear memory scales extensively with the system size.

may be used to construct the kernel $h_1(\tau)$ of the reservoir. An example of this is given in the Supplemental Material [24], which makes it clear that the training procedure is in fact constructing a δ function approximation.

To construct an *LRC* network of N circuits, we identify a maximum frequency $\omega_{\max}$ associated with our input signal and define $\Delta\omega = \omega_{\max}/N$ as the lowest frequency and resolution. This lowest frequency defines a timescale $t' \sim 1/\Delta\omega$ over which a signal could be represented by the Fourier series. We then choose $\gamma = \Delta\omega$ to suppress the signal for times longer than t'. For the smoothed Gaussian noise input signal used here (see Supplemental Material [24]), we take $\omega_{\max} = 4$.

We expect that if $\omega_{\max}$ is chosen to be sufficiently large so as to accurately represent the signal, the system's total linear memory τ_ϵ will scale as N. This is confirmed in Fig. 5, where we show the memory function $m(\tau) = C[u(t-\tau)]$ and the total linear memory τ_ϵ for $\epsilon = 0.1$ across a range of reservoir sizes. As expected, reservoirs of this type show an extensive total linear memory. Note that the precise value of ϵ is not particularly important since the memory function for these networks maintains a value near 1 before falling sharply. This implies that an *LRC* network will be able to approximate any function $z(t) = \int_0^{T^*} d\tau\, h_1(\tau)u(t-\tau)$ for any kernel h_1, as long as the network is large enough that its total linear memory obeys $\tau_\epsilon > T^*$.

B. Memristor reservoirs

We now turn to establishing analogous results for memristor reservoirs. To begin, in the Supplemental Material [24], we prove the following result:

Feasibility of memristor networks. Reservoirs of networked memristors satisfy local fading memory and state separation properties for sufficiently weak input signal.

Specifically, the proof requires that the driving signal satisfies $||\vec{u}(t)||_2 \leqslant \frac{(1-\chi)^2\alpha\beta}{\chi}$. This result was a surprise given the passive nature of the memristor model, but simulations have demonstrated the importance of weak driving to the fading memory property. The dependence on χ illustrates the constraints put on the driving signal by the nonlinearity of the memristors. The condition above illustrates a tradeoff between nonlinearity and volatility, governed by the decay constant α in maintaining the fading memory property. We also note that nonvolatile memristors have demonstrated a form of fading memory [30,31], though it is not clear what is at the root of this effect. Certainly, saturation effects at the boundary of the devices can also lead to fading memory for a strong signal and this will be in the opposite regime to the bound above. It is certainly possible that different memristors show fading memory over a wider range of driving signals and for varying reasons, but our investigations have made it clear that the linear model will only satisfy the local fading memory property when weakly driven. In the Supplemental Material [24], we also show that as long at the dissipative term is bounded below by a linear function of $\vec{\eta}$, a very similar bound will hold.

Further work has shown that when strongly driven, memristor networks can enter a transiently chaotic state in which trajectories will diverge in time [32], in apparent violation of the local fading memory property. However, after this transiently chaotic phase of the dynamics, the system will approach a steady state under constant driving. Presently it is not clear whether systems that display transient chaos can form useful reservoirs when the driving signal is time varying, as chaotic dynamics may recur repeatedly. Our simulations have indicated that maintaining the fading memory property is important, so all memristor networks that we employ are in the weakly driven regime.

The trajectory of a single memristor governed by Eq. (19) has the Volterra series expansion,

$$\eta(t) = \frac{1}{\beta}\int_0^\infty d\tau_1 e^{-\alpha\tau_1}u(t-\tau_1) + \frac{\chi}{\beta^2}\int_0^\infty d\tau_1 \int_{\tau_1}^\infty d\tau_2\, e^{-\alpha\tau_2}u(t-\tau_1)u(t-\tau_2) + O(\chi^2), \tag{22}$$

(see Supplemental Material [24]), where we have neglected boundary effects and where higher-order kernels of the input are suppressed by higher powers of χ. We immediately note that the kernel functions are exponentially decaying at a rate α. This is consistent with the fading memory property, as any perturbation in u will have an exponentially decreasing effect on η as time goes on. The expansion makes it clear that memristor trajectories depend on higher powers of the driving

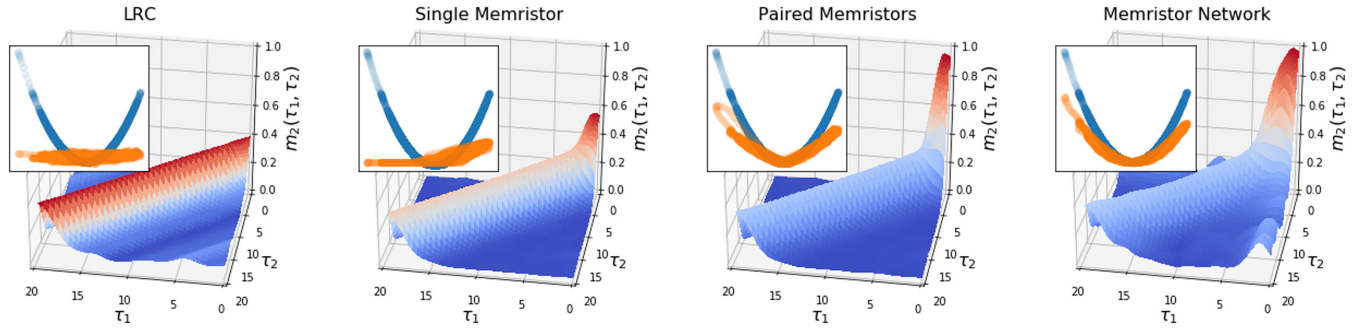

FIG. 6. The quadratic memory function $m_2(\tau_1, \tau_2)$ for a single *LRC* subcircuit (left), a single memristor subcircuit (middle left), a paired memristors subcircuit (middle right), and a triangular lattice of memristors (right) as shown in Figs. 3 and 4. This measures the reservoir's ability to approximate the function $z(t) = u(t - \tau_1)u(t - \tau_2)$. The colors of the surfaces are purely for visualization and values should be read from the z axes which all extend from 0 to 1. We define $\tau^* := \text{argmax}_\tau m_2(\tau, \tau)$, the optimal delay for an equal-time reconstruction; the insets show the corresponding reconstruction $\hat{z}$ (orange) and target output $z(t) = [u(t - \tau^*)]^2$ (blue) as a function of the input signal $u(t - \tau^*)$. As expected, the *LRC* circuit is only capable of generating linear approximations of the output. A single memristor reservoir is unable to isolate its quadratic component and misses negative parts of the reconstruction due to boundary effects (middle left inset). The addition of another memristor with opposite bias significantly increases the ability to reconstruct z. The memristor network shows an enhanced ability to reconstruct z and clear nonlinearity (right inset). The nonlinear memory function value of $m_2(\tau^*, \tau^*)$ for each network was 0.390 (*LRC*), 0.558 (single memristor), 0.960 (paired memristors), and 0.995 (memristor network).

signal $u(t)$. In this sense, memristors may be considered as a source of nonlinearity in electronic reservoirs.

As a generalization of the linear approximation task, we consider the approximation of an arbitrary second-order filter which depends on the input signal for a time T^* into the past. Specifically, we wish to approximate a function of the input with the form

$$z(t) = \int_0^{T^*} d\tau_1 \, F_1(\tau_1) u(t - \tau_1) + \int_0^{T^*} d\tau_1 \int_0^{T^*} d\tau_2 \, F_2(\tau_1, \tau_2) u(t - \tau_1) u(t - \tau_2). \quad (23)$$

Approximating this function with memristors requires isolating terms related to the second-order contribution to the trajectory in Eq. (22). To do this, we note that a reservoir of two memristors $\eta_+(t)$ and $\eta_-(t)$, each driven by $u(t)$ and $-u(t)$, respectively, allows us to construct their sum [away from the boundaries and up to terms of the order of $O(\chi^2)$] as

$$\eta_+(t) + \eta_-(t) \approx \frac{2\chi}{\beta^2} \int_0^\infty d\tau_1 \int_{\tau_1}^\infty d\tau_2 \, e^{-\alpha\tau_2} u(t - \tau_1) u(t - \tau_2),$$

which cancels all odd terms in $u(t)$ in Eq. (22). Including memristors in pairs thus allows the training procedure to isolate these quadratic components and learn their weights so as to approximate Eq. (23).

The $e^{-\alpha\tau}$ dependence of terms in the Wiener-Volterra series only allows a dependence on u on timescales of the order of $\frac{1}{\alpha}$. A lower value of α will integrate a longer window of the previous history into the current state, but will also obscure the value of the input signal at any single time. In reservoir computing, the parameters of the circuit are randomized to generate linearly independent trajectories, which allows the training to isolate different components of the input signal. In memristor networks, this may be accomplished by varying α and β (the timescales of decay and excitation for memristors), by varying the amplitude of the driving, or by introducing disorder into the structure of the circuit Ω_A. We employ what we view as the most practical option, which is varying the amplitude of the driving signal. In networks, memristors are driven with a proximal voltage generator that varies in amplitude from $+S$ to $-S$ in equally spaced increments, where S is a constant that may be tuned.

In Fig. 6, we show the quadratic memory function $m_2(\tau_1, \tau_2) = C[u(t - \tau_1)u(t - \tau_2)]$ for reservoirs composed of *LRC*, single memristors, paired memristors, and memristor networks, as shown in Figs. 3 and 4. As expected, while the *LRC* reservoir produces excellent linear reconstructions, it shows very poor ability to reconstruct quadratic functions. Among memristors, while single memristors show clear nonlinearity, paired memristors are markedly better and memristor networks show only a limited advantage over separate paired memristor subcircuits.

While memristor reservoirs give us the ability to calculate quadratic functions of the input with high accuracy for short times, the total quadratic memory $\tau_\epsilon^{(2)}$ does not scale extensively as the size of the reservoir is increased. This can be seen from the fact that the memristor network reservoir, which is 28 times bigger than the "paired memristors" reservoir, has a similar total quadratic memory. In the next section, we consider hybrid reservoirs of memristor and *LRC* components, which do demonstrate extensive scaling.

C. Hybrid deep reservoirs

The properties of *LRC* and memristor reservoirs may be combined to achieve improved scaling of the nonlinear memory. The reservoir structure we will examine uses the trajectories of a "surface layer" reservoir $\vec{x}_s(t)$, where voltage generators are driven by the input, $\vec{s}_s = \vec{v}u(t)$, to drive another "deep layer" reservoir $\vec{x}_d$ [33]. The deep layer voltage generators are driven by the surface layer trajectories as $\vec{s}_d = C\vec{x}_s$, where C is a matrix of coupling coefficients whose structure is discussed below and shown schematically in Fig. 7. As *LRC* and memristor components are kept in separate layers,

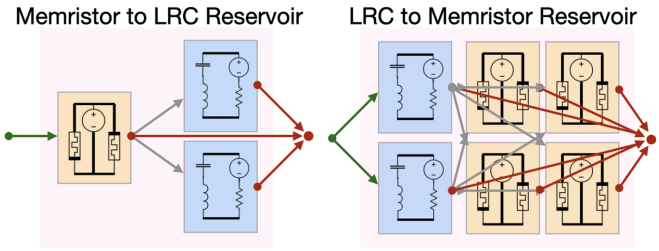

FIG. 7. We consider two deep reservoir structures in which the layers are independent subcircuit reservoirs of *LRC* or memristor subcircuits. Connections and layers are shown schematically to make the figures legible. Inputs to the reservoirs are shown in green, internal connections between layers are shown in gray, and outputs are shown in red. In the memristor to *LRC* reservoirs (left), the surface layer is a paired memristor circuit. Each of the output trajectories $\eta_{\pm}(t)$ is used to drive a deep layer reservoir of 10 *LRC* subcircuits. In the *LRC* to memristor reservoirs (right), each pair of surface layer *LRC* circuits drives 12 paired memristor circuits (not all shown here) such that all sums and differences of the four *LRC* trajectories drive a separate paired memristor circuit.

these deep reservoirs inherit the feasibility properties of their subcomponents. The training procedure uses all trajectories $\vec{x} = [\vec{x}_s, \vec{x}_d]$ in the regression.

As seen in Fig. 6, the ability of memristors to calculate quadratic functions of the input occurs only over very short delays. On the other hand, *LRC* networks show excellent linear memory approximations, but cannot reconstruct nonlinear functions. This would suggest that using a surface layer memristor network to generate nonlinear transformations of the input, and then using these to drive a deep layer *LRC* reservoir that would remember them, would give both good quadratic reconstructions and long memory of these reconstructions. This structure is shown in the left panel of Fig. 7. In Fig. 8, the top panel shows the result of using a pair of memristors configured as in the section above to drive an *LRC* reservoir. Each of the two memristor trajectories in $\vec{x}_s = [\eta_+(t),\ \eta_-(t)]$ is used as a source signal for a small *LRC* reservoir of 10 circuits, which we index by $n\pm$ with $n = 1 \ldots 10$. Each *LRC* circuit produces two trajectories $q_{n\pm}, \dot{q}_{n\pm}$, giving a total of $2 + 2 \times 10 \times 2 = 42$ output trajectories. The *LRC* trajectories are calculated by Eq. (14) with $s_{n\pm} = \eta_{\pm}$. The index n determines the parameters of the *LRC* elements given $\gamma = 0.4$ and $\Delta\omega = 0.4$ and Eq. (21).

The resulting reservoir trajectories are used to evaluate the quadratic memory function in the top panel of Fig. 8, with the results showing a substantial increase in the reservoir's computational capacity for "equal-time" quadratic reconstructions [defined via $m_2(\tau, \tau)$]. As the deep *LRC* reservoir is used to recall the equal-time products computed by the memristor reservoir, we expect that measures of their total quadratic memory $\tau_\epsilon^{(2)}$ will also scale extensively. However, increasing the reservoir size will not improve the reconstruction of unequal-time products where $\tau_1 \neq \tau_2$.

We next consider using a surface layer *LRC* reservoir to drive a deep layer memristor reservoir as shown in the right panel of Fig. 7. As guiding intuition, if we consider the *LRC* reservoir as computing the Fourier transform of the signal, the deep memristor layer will calculate products of this transform,

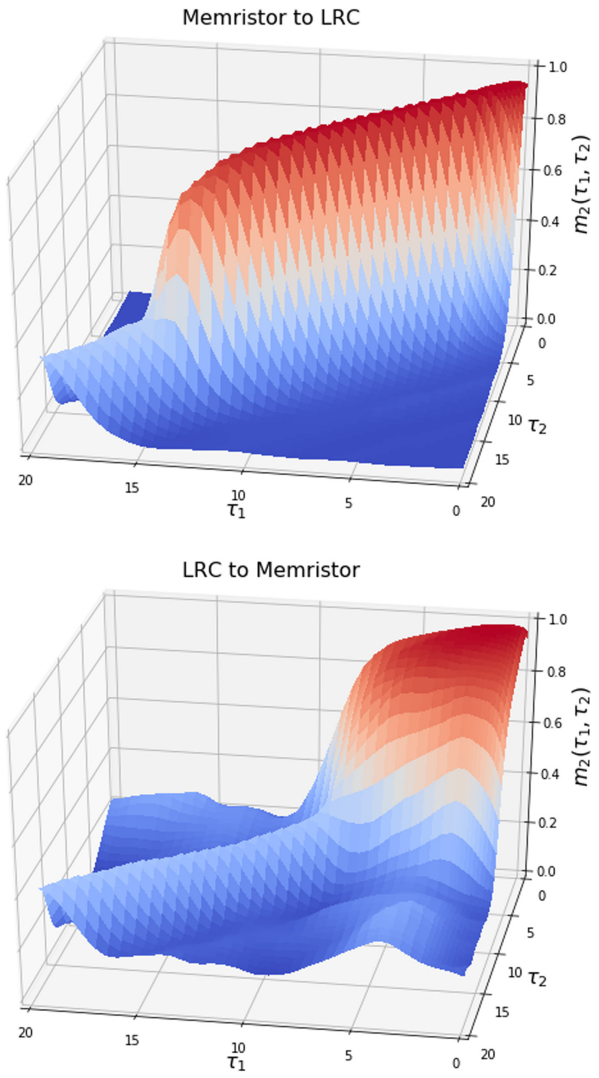

FIG. 8. The quadratic memory function $m_2(\tau_1, \tau_2)$ for the hybrid memristor and *LRC* reservoirs shown in Fig. 7. The top panel shows the result of using a driven pair of memristors to drive an *LRC* reservoir. The *LRC* reservoir stores memory of the nonlinear computation in the memristor network, leading to large equal-time quadratic capacities. In the lower panel, the result of using an *LRC* reservoir to drive a set of memristor pairs is shown. The memristor pairs compute products of the trajectories generated in the *LRC* network, approximately implementing a two-dimensional Fourier transform. This extends the quadratic memory function to longer delays compared with the paired memristor reservoir in Fig. 6. The *LRC* circuits were arranged with $\gamma = 0.4$, $\Delta\omega = 0.4$ corresponding to a cutoff frequency of 4 Hz.

akin to a two-dimensional Fourier transform in τ_1 and τ_2. We predict that the resulting network will display an improved unequal-time quadratic memory function. To test this, we implemented the same 10-circuit *LRC* reservoir as described above driven with the same input signal u. The resulting 20 trajectories, $q_n, \dot{q}_n$, $n = 1 \ldots 10$, are used to drive a set of memristor pairs such that the sum and difference of every pair of the 20 *LRC* trajectories are used to drive an independent pair of memristors. This means that for a particular pair η_{m+}, η_{m-}, the driving signal may be $q_n \pm q_{n'}$, $q_n \pm \dot{q}_{n'}$,

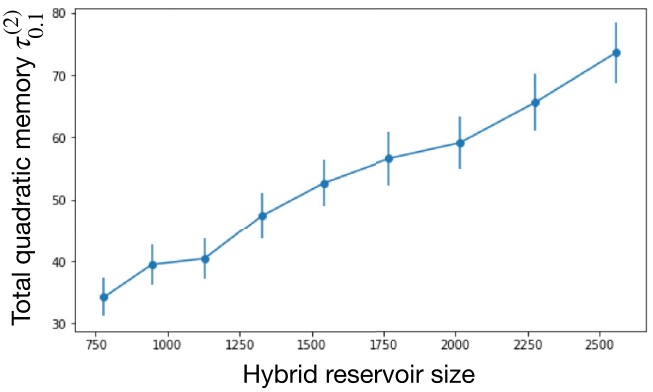

FIG. 9. The scaling of the total quadratic memory with reservoir size in the hybrid *LRC* to memristor reservoir. *LRC* reservoirs ranging from 10 to 18 subcircuits were used to drive a memristor reservoir as described in the main text, resulting in reservoirs ranging from 946 to 2556 internal degrees of freedom. The total quadratic memory $\tau^{(2)}_{0.1}$ (and error bars) was estimated from a finite-size scaling analysis, as detailed in the Supplemental Material [24].

or $\dot{q}_n \pm \dot{q}_{n'}$ such that all pairs n, n' and $q, \dot{q}$ are used. The resulting $2 \times 20 \times 19 + 20 = 780$ trajectories are used to train the reservoir. In the lower panel of Fig. 8, we calculate the quadratic memory function for this architecture. We observe a substantial improvement in the reservoir's ability to construct unequal-time products of the input signal. Although this requires a significant increase in the size of the reservoir, such an increase is expected. The number of unequal-time products with $\tau_1, \tau_2 < T^*$ scales quadratically in the maximum delay T^* and so the reservoir size must scale similarly.

In Fig. 9, we show the scaling of the total quadratic memory $\tau^{(2)}_{0.1}$ with the size of the reservoir. The total quadratic memory indeed scales extensively with the size of the reservoir, indicating that arbitrary unequal-time products may be reconstructed by a sufficiently large reservoir. Estimating these quantities accurately turns out to be quite subtle, as estimated values will display strong bias when calculated on a finite interval. In the Supplemental Material [24], we show how finite-size scaling can be used to obtain reliable estimates. We emphasize that it is the total nonlinear memory $\tau^{(2)}_{\epsilon}$ that can scale extensively with the system size; increasing the maximum delay T^* under which we can reconstruct products of the input will require that the reservoir size scales as T^{*2}.

Another natural architecture to consider would use memristor reservoirs as both surface and deep layers, as has been considered in works based on the simulation of these devices. Given the discussion above, we expect the primary benefit of this architecture would be to enhance higher-order nonlinear capacities, which is precisely what we observe in simulation. However, as this is outside the scope of the computational task we set out to achieve, we do not include results from such networks here.

D. Comparison with echo state networks

To show that these design considerations lead to improved performance, we construct a fitting task in which we must approximate a known function of the input signal. We construct the following target output:

$$z(t) = \int_0^{10} d\tau_1 \, K_1(\tau_1) u(t-\tau_1) + \int_0^{10} d\tau_1 \int_0^{10} d\tau_2 K_2(\tau_1, \tau_2) u(t-\tau_1) u(t-\tau_2), \tag{24}$$

where the kernels K_1 and K_2 are defined as

$$K_1(\tau_1) = e^{-0.5\tau_1} \cos(2\tau_1), \tag{25}$$

$$K_2(\tau_1, \tau_2) = -e^{-0.3(\tau_1+\tau_2)} \cos[2(\tau_1 - \tau_2)]. \tag{26}$$

Accurate approximation of this output requires a mixture of memory and nonlinearity.

TABLE I. Comparison of reservoir performance on the quadratic filtering task described in the main text.

Reservoir	dim($\vec{x}$)	nMSE	Gen. nMSE
Pure memristor network	800	0.1	2.0
ESN	780	0.02	0.03
LRC → memristor	780	0.01	0.01

In addition to the hybrid reservoir discussed above, we also apply an implementation of continuous-time echo state networks (ESNs) [1,3] for comparison. An ESN is a dynamical reservoir in which the internal states evolve according to

$$\dot{\vec{x}} = -\alpha \vec{x}(t) + \tanh[M\vec{x}(t) + \vec{v}u(t)], \tag{27}$$

where α is a decay term, $\tanh(\cdot)$ applies to every neuron, M is a matrix that in order to satisfy the fading property must have maximum eigenvalue less than one, and $\vec{v}$ scales the magnitude of the input $u(t)$ which drives each neuron.

We compare the results of a suitably tuned ESN to a pure memristor network as in Fig. 4, and a hybrid surface *LRC* to deep memristor reservoir. The memristor network is a 17×17 triangular lattice with 800 edges, each containing a memristor ($\alpha = 3$, $\beta = 1$, $\chi = 0.8$) and voltage generator. The elements of $\vec{v}$ were uniformly distributed on the interval $[-1, -0.1] \cup [0.1, 1]$. The *LRC* → memristor reservoir was configured identically to that presented in the previous section and included a total of 780 trajectories. Finally, the ESN consisting of 780 elements was run and tuned following the recommendations in [3]. As far as possible, each reservoir was configured to produce the same number of trajectories (the lattice structure of the naive memristor network imposes some constraints). Further details on the implementation of each reservoir can be found in the Supplemental Material [24]. The *LRC* → memristor reservoir was configured identically to that presented in the previous section and included a total of 780 trajectories. Finally, the ESN consisting of 780 elements was run and tuned following the recommendations in [3].

Each reservoir was first initialized on the interval $[0, 100]$ corresponding to approximately 100 autocorrelation times of the input driving signal. They were then trained on the interval $[100, 4000]$ and the $\text{nMSE}_{[100,4000]}$ is reported in Table I. Lastly, a generalization error is reported by calculating the

nMSE[z] on the interval [4000, 5000] with the weights that were trained on [100, 4000]. The generalization error is no longer normalized to [0, 1] as the weights are calculated on a different interval. However, it is the most important measure of the reservoir's ability to approximate the function $u \mapsto z$, rather than to simply fit this function on a single training interval.

The results of the training are shown in Table I. The hybrid *LRC* $\rightarrow$ memristor reservoir demonstrates a 10-fold improvement over the pure memristor network and performs on par with the ESN implementation in training as well as a twofold improvement in generalization error. We attribute this to the specialized structure of the *LRC* to memristor reservoir, which gives it an advantage in reliably calculating quadratic functions of the input. We conclude that suitably crafted analog reservoirs are thus capable of matching and even surpassing the performance of standard reservoirs. The generalization error of the pure memristor network exceeded 1 and shows that the MSE on this interval was twice the variation in z. This indicates no ability to generalize the fit from the training interval and highlights the importance of reporting generalization error, rather than training error, in work on electronic reservoirs.

IV. DISCUSSION

Despite wide interest in utilizing electronic circuits with memory for hardware reservoirs, an understanding of how these systems process and store information has been lacking. For echo state networks, the balance between memory and nonlinearity is controlled primarily by the spectral radius of the coupling matrix. However, no similar conditions have been explored for electronic networks.

In this work, we have shown that linear electronic reservoirs of *LRC* circuits can be constructed with optimal memory properties, having an eigenvalue spectrum known to correspond to an extensive memory (e.g., that scales proportionally to the number of components). This may be interpreted as performing a Fourier transform of the driving signal in hardware, where the required eigenvalue spectrum can be designed appropriately for a given problem.

In memristor reservoirs, we have shown that while the system contains contributions from terms of very high order, these are moderated in strength by powers of χ. It is essential that the reservoir be able to isolate the desired terms to make use of them in the training process. We have shown that using paired memristors of opposite polarity gives a substantial increase in the reservoir's ability to isolate their quadratic kernels.

Combining *LRC* and memristor networks into deep reservoirs allows the utilization of the memory capabilities of *LRC* reservoirs and the nonlinear capacities of memristor reservoirs in order to achieve specific computational goals. Utilizing an *LRC* network as a deep layer allows nonlinear computations performed in the surface memristor network to be stored for long times. Similarly, using an *LRC* reservoir as the surface layer to drive a deep layer memristor network will calculate products of Fourier modes and give enhanced unequal-time quadratic capacities. Most importantly, this leads to a total quadratic memory which scales extensively in the system size, such that arbitrary products of the input can be constructed by a sufficiently large reservoir.

This analysis can have substantial impacts on performance, as we show in our comparison to ESN reservoirs. The hybrid reservoirs we present give a 10-fold improvement over the naive memristor network implementation and perform on par with the ESN implementation. Properly constructed electronic reservoirs should thus be capable of matching the performance of standard reservoirs, but also allow the use of larger reservoirs and faster computation times.

Our approach to the analysis of the computational capacities of memristor and *LRC* reservoirs can be generalized to higher-order kernels of the network and to other nonlinear elements. In this sense, we present a general approach to the understanding of physical reservoirs, in analogy to the methods available to tune ESNs by trading between memory storage and nonlinearity [3].

ACKNOWLEDGMENTS

The work of F.C. and F.C.S. was carried out under the auspices of the NNSA of the U.S. Department of Energy at Los Alamos National Laboratory under Contract No. DE-AC52-06NA25396. F.C. was also financed via DOE-ER Grant No. PRD20190195, and F.C.S. by a CNLS Fellowship and Grant No. 20190195ER. Revision of this manuscript by F.C.S. was performed while employed at the London Institute for Mathematical Sciences. A.K. was supported by Grant No. FQXi-RFP-IPW-1912 from the Foundational Questions Institute and Fetzer Franklin Fund, a donor advised fund of the Silicon Valley Community Foundation. A.K. thanks the Santa Fe Institute for helping to support this research.

[1] H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks—with an erratum note, German National Research Center for Information Technology, GMD Tech. Rep. No. 148, 2001, https://www.ai.rug.nl/minds/uploads/EchoStatesTechRep.pdf (unpublished).

[2] H. Jaeger, Short term memory in echo state networks, Fraunhofer Institute for Autonomous Intelligent Systems, GMD Tech. Rep. No. 152 2002, https://www.ai.rug.nl/minds/uploads/STMEchoStatesTechRep.pdf (unpublished).

[3] M. Lukoševičius, A practical guide to applying echo state networks, in *Neural Networks: Tricks of the Trade*, edited by G. Montavon, G. B. Orr, and K.-R. Müller (Springer, New York, 2012), pp. 659–686.

[4] N. Bertschinger, and T. Natschläger, Real-time computation at the edge of chaos in recurrent neural networks, Neural Comput. **16**, 1413 (2004).

[5] H. Jaeger and H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science **304**, 78 (2004).

[6] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach, Phys. Rev. Lett. **120**, 024102 (2018).

[7] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, Recent advances in physical reservoir computing: A review, Neural Networks **115**, 100 (2019).

[8] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, Information processing using a single dynamical node as complex system, Nat. Commun. **2**, 468 (2011).

[9] F. Caravelli and J. P. Carbajal, Memristors for the curious outsiders, Technologies **6**, 118 (2018).

[10] L. Chua, Memristor-the missing circuit element, IEEE Trans. Circuit Theor. **18**, 507 (1971).

[11] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, The missing memristor found, Nature (London) **453**, 80 (2008).

[12] J. Torrejon, M. Riou, F. A. Araujo, S. Tsunegi, G. Khalsa, D. Querlioz, P. Bortolotti, V. Cros, K. Yakushiji, A. Fukushima *et al.*, Neuromorphic computing with nanoscale spintronic oscillators, Nature (London) **547**, 428 (2017).

[13] A. Zegarac and F. Caravelli, Memristive networks: From graph theory to statistical physics, Europhys. Lett. **125**, 10001 (2019).

[14] V. Athanasiou and Z. Konkoli, On improving the computing capacity of dynamical systems, Sci. Rep. **10**, 9191 (2020).

[15] J. P. Carbajal, J. Dambre, M. Hermans, and B. Schrauwen, Memristor models for machine learning, Neural Comput. **27**, 725 (2015).

[16] C. Du, F. Cai, M. A. Zidan, W. Ma, S. H. Lee, and W. D. Lu, Reservoir computing using dynamic memristors for temporal information processing, Nat. Commun. **8**, 2204 (2017).

[17] M. Inubushi and K. Yoshimura, Reservoir computing beyond memory-nonlinearity trade-off, Sci. Rep. **7**, 10199 (2017).

[18] M. S. Kulkarni and C. Teuscher, Memristor-based reservoir computing, in *2012 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)* (IEEE, Amsterdam, The Netherlands, 2012), pp. 226–232.

[19] M. J. Marinella and S. Agarwal, Efficient reservoir computing with memristors, Nat. Electron. **2**, 437 (2019).

[20] J. Moon, W. Ma, J. H. Shin, F. Cai, C. Du, S. H. Lee, and W. D. Lu, Temporal data classification and forecasting using a memristor-based reservoir computing system, Nat. Electron. **2**, 480 (2019).

[21] S. Boyd and L. Chua, Fading memory and the problem of approximating nonlinear operators with Volterra series, IEEE Trans. Circuit Syst. **32**, 1150 (1985).

[22] W. Maass, P. Joshi, and E. D. Sontag, Computational aspects of feedback in neural circuits, PLoS Comput. Biol. **3**, e165 (2007).

[23] W. Maass, T. Natschläger, and H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations, Neural Comput. **14**, 2531 (2002).

[24] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevE.106.045310 for further discussions of background material, details of calculations and simulations and proofs of statements in the text.

[25] J. Dambre, D. Verstraeten, B. Schrauwen, and S. Massar, Information processing capacity of dynamical systems, Sci. Rep. **2**, 514 (2012).

[26] M. Hermans and B. Schrauwen, Memory in linear recurrent neural networks in continuous time, Neural Networks **23**, 341 (2010).

[27] O. L. White, D. D. Lee, and H. Sompolinsky, Short-Term Memory in Orthogonal Neural Networks, Phys. Rev. Lett. **92**, 148102 (2004).

[28] F. Caravelli, F. L. Traversa, and M. Di Ventra, Complex dynamics of memristive circuits: Analytical results and universal slow relaxation, Phys. Rev. E **95**, 022140 (2017).

[29] R. Wang, J.-Q. Yang, J.-Y. Mao, Z.-P. Wang, S. Wu, M. Zhou, T. Chen, Y. Zhou, and S.-T. Han, Recent advances of volatile memristors: Devices, mechanisms, and applications, Advanc. Intell. Syst. **2**, 2000055 (2020).

[30] A. Ascoli, R. Tetzlaff, L. O. Chua, J. P. Strachan, and R. S. Williams, History erase effect in a nonvolatile memristor, IEEE Trans. Circuits Syst. I: Reg. Papers **63**, 389 (2016).

[31] S. Menzel, R. Waser, A. Siemon, C. La Torre, M. Schulten, A. Ascoli, and R. Tetzlaff, On the origin of the fading memory effect in rerams, in *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)* (IEEE, Thessaloniki, Greece, 2017), pp. 1–5.

[32] F. Caravelli, F. C. Sheldon, and F. L. Traversa, Global minimization via classical tunneling assisted by collective force field formation, Sci. Adv. **7**, 52 (2021).

[33] C. Gallicchio, A. Micheli, and L. Pedrelli, Deep reservoir computing: A critical experimental analysis, Neurocomputing **268**, 87 (2017).