LA-UR-20-26991
Accepted Manuscript

The Computational Capacity of Mem-LRC Reservoirs

Sheldon, Forrest Crawford
Caravelli, Francesco
Kolchinsky, Artemy

Provided by the author(s) and the Los Alamos National Laboratory (2023-08-07).

To be published in:

DOI to publisher's version: 10.1145/3381755.3381779

Permalink to record:
https://permalink.lanl.gov/object/view?what=info:lanl-repo/lareport/LA-UR-20-26991

The Computational Capacity of Mem-LRC Reservoirs

Forrest Sheldon*
Francesco Caravelli*
fsheldon@lanl.gov
caravelli@lanl.gov
Los Alamos National Lab, T-4/CNLS
LA-UR-19-31374
Los Alamos, New Mexico

CCS CONCEPTS

• **Hardware → Neural systems**; *Emerging architectures.*

KEYWORDS

reservoir computing, memristors, neuromorphic computing

1 SUMMARY

Reservoir computing has a emerged as a powerful tool in data-driven time series analysis. The possibility of utilizing hardware reservoirs as specialized co-processors has generated interest in the properties of electronic reservoirs, especially those based on memristors as the nonlinearity of these devices should translate to an improved nonlinear computational capacity of the reservoir. However, designing these reservoirs requires a detailed understanding of how memristive networks process information which has thusfar been lacking. In this work, we derive an equation for general memristor-inductor-resistor-capacitor (MEM-LRC) reservoirs that includes all network and dynamical constraints explicitly. Utilizing this we undertake a study of the computational capacity of these reservoirs. We demonstrate that hardware reservoirs may be constructed with extensive memory capacity and that the presence of memristors enacts a tradeoff between memory capacity and nonlinear computational capacity. Using these principles we design reservoirs to tackle problems in signal processing, paving the way for applying hardware reservoirs to high-dimensional spatiotemporal systems.

2 BACKGROUND

Reservoir computing is a machine learning technique for constructing a mapping between two ordered sequences [7, 11]. An input sequence is used to drive a dynamical system, or *reservoir*, and the output sequence is approximated by a linear regression on the reservoir state. The reservoir dynamics are such that its current state can contain information concerning both the previous history of the driving input and nonlinear transformations thereof. The linearity of the output, combined with the nonlinearity of the reservoir allows us to interpret reservoir computing as approximating a nonlinear filter between the two sequences. Indeed, reservoir computers can be considered filter approximators in much the same way that feedforward neural networks may be considered function approximators [11]. The low computational cost of the linear regression and lack of reliance on backpropagation allows for very large and recurrent reservoirs to be utilized. This paradigm has proven effective in a wide variety of signal processing tasks including pattern generation and classification [1], time series forecasting [7], adaptive filtering and is currently the state of the art method in the prediction of chaotic systems [9].

The linear readout of reservoir computers implies that their computational power is intrinsic to the particular dynamical system employed, and designing these reservoirs requires an understanding of how the physical system in question stores and processes information. A framework for characterizing the computational capacity of a reservoir was developed in [3] by identifying the trajectories of the reservoir as living in a Hilbert space of fading memory functions (functions acting on some finite history of the driving sequence). The dimension of the space spanned by the system trajectories formed an invariant measure of the computational capacity of the system, naturally encompassing previous measures of memory capacity [8, 14] and revealing a fundamental trade-off between memory/linear functions and nonlinear functions of the input. The capacity for a reservoir X to reconstruct an approximation $\hat{z}$ of a function $z[u]$ on an interval T may then be interpreted as the projection of $z[u]$ onto the trajectory space,

$$C_X[z] = 1 - \frac{\min_W MSE_T[z, \hat{z}]}{\langle z^2 \rangle_T}. \quad (1)$$

where W are the readout weights of the network and $\langle \cdot \rangle_T$ is the time average. This is bounded on $0 \leq C_X[z] \leq 1$ and encompasses previous measures of memory, $m(\tau) = C_X[z[u](t) = u(t-\tau)]$ [3] while also generalizing to nonlinear functions of the input, *e.g.* $z[u](t) = u(t-\tau)^2$.

The performance of a reservoir relies on tuning the memory and nonlinear fitting capacity of the reservoir to the requirements of the specific problem. The ability to achieve this for arbitrary problems places emphasis on the scaling properties of the computational capacity with the reservoir size. In particular, useful reservoirs should display extensive or near extensive computational capacity

*Both authors contributed equally to this research.

(*i.e.*, computational capacity scaling approximately linearly with the reservoir size) [14].

The successes of reservoir computing have galvanized interest in hardware implementations [13] which could be used as specialized co-processors and in which very large reservoir sizes could be achieved. The construction of useful hardware reservoirs thus requires an understanding of how their computational capacity scales with system size. Memristor-based reservoirs have generated substantial interest, both for their resemblance to a synapse *in silica* and as they are perhaps the simplest example of a non-linear electrical element. However, studies of memristor-based reservoirs have relied on primarily anecdotal evidence of their efficacy, giving evidence from simulations that memristors *could* form effective reservoirs [4, 10, 12]. None, to our knowledge, have studied memristive reservoirs analytically or have given a systematic account of the linear and non-linear fitting capacity of these devices and how these capabilities scale with system size.

In this work we study the performance of memristor-LRC (Mem-LRC) networks as reservoirs, consisting of memristor, inductor, capacitor and resistor motifs assembled in a driven electrical network. The inclusion of linear elements allows us to assess the role of memristors in introducing nonlinear computational capacity to the reservoir in a systematic way and to achieve a tune-able trade-off between memory and nonlinearity, similar to the mixture reservoirs proposed in [6]. Achieving large computational capacities with MEM-LRC reservoirs requires a careful choice of components, local connectivity and global network structure.

3 THE COMPUTATIONAL PROBLEM

We are interested in the approximation of a nonlinear filter of the form,

$$z[u](t) = \int_0^\infty d\tau_1\, K_1(\tau_1)u(t-\tau_1) + \int_0^\infty d\tau_1 \int_0^\infty d\tau_2\, K_2(\tau_1,\tau_2)u(t-\tau_1)u(t-\tau_2) + \cdots \quad (2)$$

by means of a reservoir driven by an input signal $u(t)$. This is a linear combination of functions on the input of the form $z_{\tau_1}(t) = u(t-\tau_1)$, $z_{\tau_1,\tau_2}(t) = u(t-\tau_1)u(t-\tau_2), \ldots$ By the linearity of the reservoir fitting procedure, the ability to reconstruct these functions within a delay of $\tau_i < T$ implies the ability to reconstruct an arbitrary filter of the form 2 where the kernels K_i have support on $\tau_i < T$. Thus, by examining a reservoir's ability to reconstruct the functions above we capture a measure of it's universal computational capacity. In what follows, we will restrict ourselves to examining how reservoir design affects it's ability to construct the functions z_1 and z_2 which we will refer to as the linear capacity, $C_X[z_1](\tau_1)$ and quadratic capacity, $C_X[z_2](\tau_1,\tau_2)$ and study as functions of the delay $\tau_{1/2}$.

For input signal $u(t)$ we construct a zero-mean unit variance noise process with an autocorrelation time of 1 by generating uncorrelated gaussian noise on a fast timescale ($\delta t = 10^{-3}$) and smoothing with a double exponential window $e^{-|t|}$. Capacities were calculated over approximately 1000 autocorrelation times of the input signal.

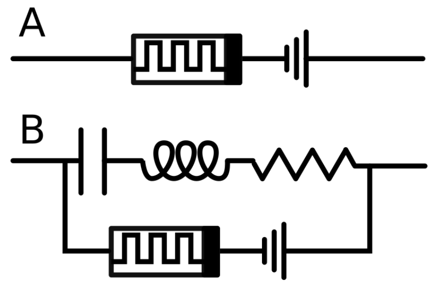

Figure 1: The motifs used to construct a reservoir from an underlying graph. Each edge in the graph is replaced with a one-port circuit as above to produce a memristive reservoir (A) or a MEM-LRC reservoir (B). In linear/LRC networks the memristor in B is replaced with a resistor.

4 CIRCUIT STRUCTURE

For the reservoir, we begin with a graph G of N edges, in which each edge will be replaced by a motif consisting of a 1-port circuit of memristors, resistors, capacitors, inductors and voltage/current generators. We focus on two motifs shown in Fig. 1, a purely memristive network (A) and a MEM-LRC network (B). In a linear LRC network, the memristor in (B) is replaced with a resistor.

Following [2], we begin by deriving a set of equations governing the behavior of the circuit which incorporate all dynamical and network constraints explicitly. For motif A we assume a linear memristive model,

$$R(w) = R_{off}(1-\chi w), \quad \chi = \frac{R_{off}-R_{on}}{R_{off}} \quad (3)$$

$$\dot{w} = -\alpha w + \frac{R_{off}}{\beta} I, \quad 0 \le w \le 1. \quad (4)$$

where α is a decay rate, and β is a combined voltage/time-scale. This leads to the network equation,

$$\dot{\vec{w}} = -\alpha\vec{w} + \frac{1}{\beta}(I - \chi\Omega_A W)^{-1}\Omega_A \vec{v} u(t). \quad (5)$$

Here $W = diag(\vec{w})$ is a diagonal matrix, Ω_A is a projector onto the space of voltage configurations and $\vec{v}$ is a random vector of input weights.

For motif B we organize the inductance, capacitance and resistances into diagonal matrices, L, C, and $R(w)/R_C$ respectively ($R(w)$ is the memristor resistance while R_C is the resistance in series with the capacitor branch). In matrix form, the equations governing the charge across the capacitor $\vec{a} = \vec{q}$ and the current into it $\vec{b} = \dot{\vec{q}}$,

$$\begin{pmatrix}\dot{\vec{a}}\\ \dot{\vec{b}}\end{pmatrix} = \begin{pmatrix} 0 & \mathbb{I} \\ -(LC)^{-1} & -L^{-1}(\Omega_{B/R^{-1}}R(w)+R_c)\end{pmatrix}\begin{pmatrix}\vec{a}\\ \vec{b}\end{pmatrix} + \begin{pmatrix} 0 \\ -L^{-1}\Omega_{B/R^{-1}}\vec{v}u(t)\end{pmatrix}. \quad (6)$$

The matrix $\Omega_{B/R^{-1}} = B^T(BR^{-1}(w)B^T)^{-1}BR^{-1}(w)$ is a non-orthogonal projector which implements Kirchoff's current law, $B\vec{i} = 0$ (for details see [2]). For the memristive degrees of freedom the same model

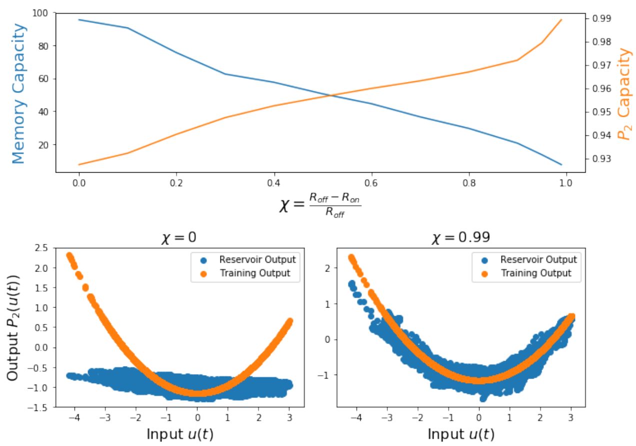

Figure 2: The computational capacity of MEM-LRC reservoirs as a function of $\chi = \frac{R_{off}-R_{on}}{R_{off}}$. In the top panel left axis, we see that the memory capacity of the network is decreasing function χ while on the right axis we see that the nonlinear capacity of the network is an increasing function, in agreement with the memory-nonlinearity tradeoff. We use the networks ability to construct $P_2(u) = \frac{1}{2}(3u^2 - 1)$ as a proxy. The lower panels show the reconstruction for no nonlinearity ($\chi = 0$, left) and large nonlinearity ($\chi = 0.99$, right).

as above in terms of the network variables above is,

$$\dot{\vec{w}} = -\alpha\vec{w} - \frac{R_{off}}{\beta} R^{-1}(w)\Omega_{B/R^{-1}(w)} R(w)\vec{b} + \tag{7}$$

$$\frac{R_{off}}{\beta} R^{-1}(w)(I - \Omega_{B/R^{-1}(w)})\vec{v}u(t) \tag{8}$$

The analysis of these equations allows us to prove that these reservoirs satisfy the fading memory and state separation properties [7] necessary for functioning as a reservoir and derive bounds on various element parameters. In simulations, the underlying network graph was chosen as a planar triangular lattice and unless otherwise noted, simulations were carried out in a 5x5 vertex lattice corresponding to 56 edges/motifs.

5 MEMRISTIVE RESERVOIR

Beginning with the purely memristive reservoir (Fig. 1 panel A, Eqn. 5) we observe that the nonlinearity of the circuit is controlled by χ and thus expect that the quadratic capacity of the network will be an increasing function of χ. (This is demonstrated in the context of Mem-LRC reservoirs in Fig. 2 where reconstruction of a quadratic function of the input was examined as a function of χ.) In simulations we choose $\chi = 0.99$ as a physically plausible value, $\beta = 1$ and α exponentially distributed with a scale of 8 Hz.

In Fig. 4 panel A, the linear and quadratic capacities of the reservoir are displayed. The large χ translates into a large quadratic capacity but only for short times $\tau_{1/2}$. The linear capacity of the network is quite poor, falling over the order of an autocorrelation length.

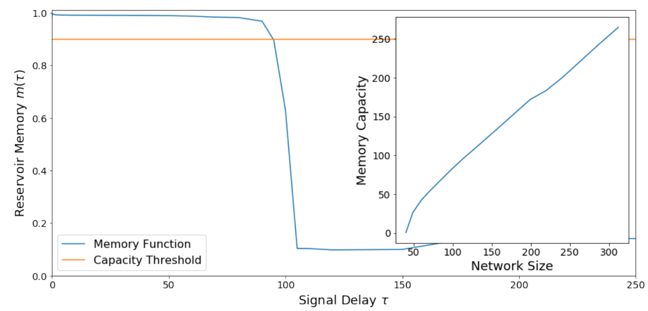

Figure 3: The memory function and 0.1−memory capacity scaling of LRC reservoirs. On the left we have displayed a typical memory function $m(\tau)$ of an 8×8 LRC lattice network which gives the capacity to reconstruct the input a time τ in the past. The memory capacity is measured as the first time this function falls below the threshold 0.9. In the inset to the right, as the network size is increased, we see that the memory capacity increases nearly linearly allowing us to achieve very long memories in hardware reservoirs. The memory lengths displayed here correspond to *hundreds* of autocorrelation times of the input sequence.

6 LRC RESERVOIRS

In order to achieve a large linear capacity, we employ a linear network (i.e. replacing the memristor in motif B with a resistor, leaving an LRC network). Using Eqn. ?? we use a scaling argument on the reservoir eigenvalues to achieve a distribution similar to that shown in [5, 14] in which the eigenvalues have constant negative real part, and are approximately equally spaced along the imaginary axis between $\pm N\Delta\omega$ where $\Delta\omega$ is a frequency resolution. To achieve this, all resistors and inductors are set to constant values r and l and capacitors are set such that,

$$c_n = \frac{1}{l\sqrt{(n\Delta\omega)^2 + \gamma^2}}, \qquad n = 1 \dots N \tag{9}$$

where $\gamma = \frac{r}{2l}$. The resulting network displays a high linear computational capacity for many autocorrelation lengths of the input. More importantly, the scaling of the memory capacity with system size is very nearly extensive, scaling as $\approx N/\ln N$ (here we define the ϵ−memory capacity $\tau_\epsilon = \min_t m(t) < 1 - \epsilon$ in order to examine it's scaling with system size) as seen in Fig. 3. An examination of several similar networks reveals that this is not a common feature and appears to hold only for this particular network motif and planar network structure. Properly designed networks are capable of recalling hundreds of autocorrelation times of the input sequence in simulation, even at relatively modest reservoir sizes. However, the linearity of these reservoirs lead to very low quadratic computational capacities. In the lower left panel of Fig. 2 we observe that quadratic reconstructions are limited to linear approximations of the desired output.

7 HYBRID RESERVOIRS

We can thus improve the memristive reservoir by running an auxiliary LRC reservoir in parallel, achieving both the linear capacity of the LRC network and the quadratic capacity of the memristor network. However, we can further leverage the memory of the LRC reservoir to increase the nonlinear network capacity as well.

First, we examine the result of using a memristive reservoir to drive an auxiliary LRC reservoir. Voltages across memristors were used to drive the LRC reservoir through a sparse connection matrix. While the LRC reservoir no longer provides linear computational capacity, we expect that by introducing memory of the nonlinear capacity calculated in the memristive reservoir, we can increase the quadratic capacity of the reservoir for longer delays. In Fig. 4 panel B we see precisely this behavior. While linear capacity remains low, the quadratic capacity has been greatly extended, particularly for diagonal ($\tau_1 = \tau_2$) reconstructions.

Lastly, we examine the behavior of Mem-LRC reservoirs composed of a planar triangular lattice with each edge replaced with a motif of the form shown in Fig. 1 B. The resulting networks appear to have the best features of LRC and memristor networks, showing large linear and quadratic capacities simultaneously. Additionally the quadratic capacity displays higher off-diagonal values than those shown by the layered memristor LRC network.

8 CONCLUSION

The problem of designing electronic reservoirs requires a detailed understanding of how these systems process and retain information about their inputs. By designing linear reservoirs with extensive linear computational capacity we are able to introduce nonlinearity through memristors in a controlled way and display large improvements in the computational capacity of the resulting reservoir. This results in more flexible systems for computational tasks which is a key requirement for useful hardware reservoirs.

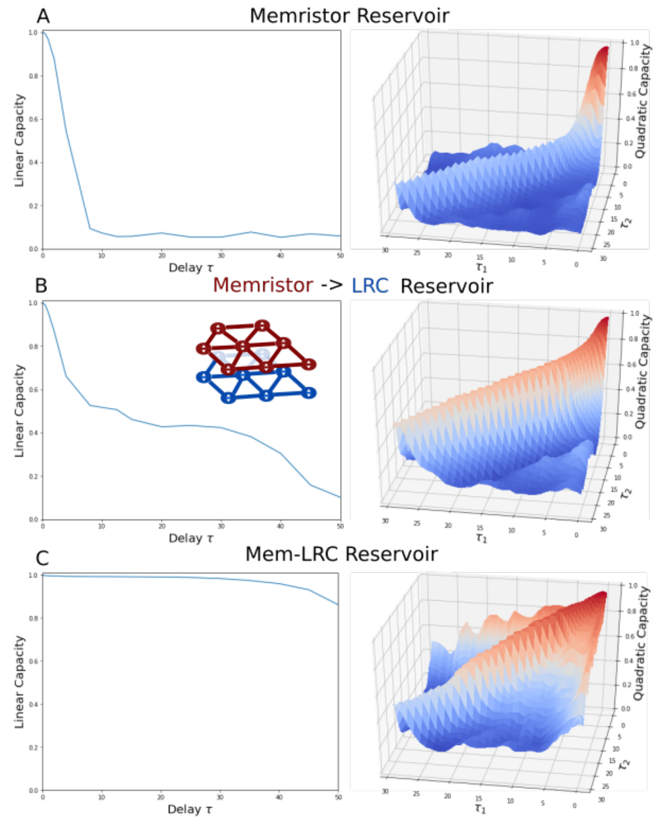

Figure 4: The linear ($C_X[z_1](\tau_1)$) and quadratic ($C_X[z_2](\tau_1, \tau_2)$) capacities for memristive (A), layered memristor-LRC (B), and combined Mem-LRC (C) reservoirs. Pure memristive reservoirs display large quadratic capacities only for short times and have little linear capacity (A). By driving an LRC network with a memristive reservoir, the memory properties of the LRC reservoir may be used to enhance the quadratic capacity (B). A combined Mem-LRC reservoir displays both large linear and quadratic capacities and the best off diagonal quadratic capacity of any reservoir (C).

REFERENCES

[1] Nils Bertschinger and Thomas Natschläger. 2004. Real-time computation at the edge of chaos in recurrent neural networks. *Neural computation* 16, 7 (2004), 1413–1436.

[2] Francesco Caravelli, Fabio Lorenzo Traversa, and Massimiliano Di Ventra. 2017. Complex dynamics of memristive circuits: Analytical results and universal slow relaxation. *Physical Review E* 95, 2 (2017), 022140.

[3] Joni Dambre, David Verstraeten, Benjamin Schrauwen, and Serge Massar. 2012. Information processing capacity of dynamical systems. *Scientific reports* 2 (2012), 514.

[4] Chao Du, Fuxi Cai, Mohammed A Zidan, Wen Ma, Seung Hwan Lee, and Wei D Lu. 2017. Reservoir computing using dynamic memristors for temporal information processing. *Nature communications* 8, 1 (2017), 2204.

[5] Michiel Hermans and Benjamin Schrauwen. 2010. Memory in linear recurrent neural networks in continuous time. *Neural Networks* 23, 3 (2010), 341–355.

[6] Masanobu Inubushi and Kazuyuki Yoshimura. 2017. Reservoir computing beyond memory-nonlinearity trade-off. *Scientific reports* 7, 1 (2017), 10199.

[7] Herbert Jaeger. 2001. *The "echo state" approach to analysing and training recurrent neural networks-with an erratum note.* Technical Report 148.34. Bonn, Germany: German National Research Center for Information Technology GMD.

[8] Herbert Jaeger. 2002. Short term memory in echo state networks. GMD-Report 152. In *GMD-German National Research Institute for Computer Science (2002), http://www.faculty.jacobs-university.de/hjaeger/pubs/STMEchoStatesTechRep.pdf.* Citeseer.

[9] Herbert Jaeger and Harald Haas. 2004. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science* 304, 5667 (2004), 78–80.

[10] Manjari S Kulkarni and Christof Teuscher. 2012. Memristor-based reservoir computing. In *2012 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*. IEEE, 226–232.

[11] Wolfgang Maass, Thomas Natschläger, and Henry Markram. 2002. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation* 14, 11 (2002), 2531–2560.

[12] Matthew J Marinella and Sapan Agarwal. 2019. Efficient reservoir computing with memristors. *Nature Electronics* (2019), 1–2.

[13] Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. 2019. Recent advances in physical reservoir computing: A review. *Neural Networks* 115 (2019), 100–123. https://doi.org/10.1016/j.neunet.2019.03.005

[14] Olivia L White, Daniel D Lee, and Haim Sompolinsky. 2004. Short-term memory in orthogonal neural networks. *Physical review letters* 92, 14 (2004), 148102.