; revised .

Control of Linear-Threshold Brain Networks via Reservoir Computing

Michael McCreesh **Jorge Cortés**

ABSTRACT Learning is a key function in the brain to be able to achieve the activity patterns required to perform various activities. While specific behaviors are determined by activity in localized regions, the interconnections throughout the entire brain play a key role in enabling its ability to exhibit desired activity. To mimic this setup, this paper examines the use of reservoir computing to control a linear-threshold network brain model to a desired trajectory. We first formally design open- and closed-loop controllers that achieve reference tracking under suitable conditions on the synaptic connectivity. Given the impracticality of evaluating closed-form control signals, particularly with growing network complexity, we provide a framework where a reservoir of a larger size than the network is trained to drive the activity to the desired pattern. We illustrate the versatility of this setup in two applications: selective recruitment and inhibition of neuronal populations for goal-driven selective attention, and network intervention for the prevention of epileptic seizures.

I. Introduction

When a function is performed by any part of the human body, it is associated with patterns of activity in the brain. Examples include actions such as eye movements, where gaze direction is associated with particular cognitive processes [1], recollection of memories, where stored memories are associated with specific sequences of activity [2], and motion planning and movement [3]. In dealing with network models for brain dynamics, it is therefore important to ascertain their ability to exhibit a given activity pattern at a certain time. For mathematical models of neuronal activity, such as Wilson-Cowan [4], Hodgkin-Huxley [5], integrate and fire [6], sigmoidal or threshold-based firing-rate models [7], among others which vary significantly in scale and properties [8], [9], this ability translates into the problem of reference tracking.

The classical approach to solving the reference tracking problem for a controlled dynamical system is the computation of an explicit input signal (that might be state-dependent) such that the desired activity pattern is exhibited by the network model. In the present context, this is undesirable for various reasons. First, the brain functions by processing the constant stream of information in the form of electrical signals it receives to achieve the desired activity pattern rather than a priori computing explicit control inputs. Second, the actual computation of these explicit expressions requires precise knowledge of the strength of synaptic interconnections in the brain, which is both impractical and challenging. Furthermore, the required conditions on network structure for these inputs to work become increasingly difficult to check with the scale of the model.

An alternative approach to tackle both of these problems is using learning methods. Neural networks in various forms have increasingly been used to model the way that the brain learns from, and adapts to, its surroundings and stimuli [10]–[13]. Machine learning techniques allow for the online determination of controls that result in the desired activity based on data, rather than computing explicit expressions, matching the functioning of the brain by constantly processing information, and also removes the requirement for exact knowledge of the synaptic connections. These observations motivate our study here of two brain functions that can be modeled with reference tracking using learning methods: selective inhibition/recruitment and seizure rejection.

Literature Review: We consider brain networks governed by a linear-threshold rate dynamics. In the context of the brain, linear-threshold network (LTN) dynamics form a mesoscale model with each node in the network representing the average firing rate of a population of neurons, whose size could range from individual neurons to millions. The dynamics have been used extensively for modeling neurons for network models in the brain [7], [14]–[16]. In a controls context, this model has also been used to study a variety of behaviors such as memory [17], goal-driven selective attention [18]–[20], and epilepsy [21], [22]. The properties of LTN dynamics have been studied extensively, including equilibria [17], [23], stability [18], [24], [25], and oscillatory and chaotic behavior [22], [26]–[28]. Additionally, properties of controlling networks governed by the linear-threshold dynamics have been studied using both model-based control [20] and data-based control [29]. Due to the typical lack of full knowledge of the network structure, we are interested in using data-driven control methods. One of

the major benefits of LTNs compared to other commonly used brain models is that the dynamics is piecewise linear, allowing for analysis of the model as a piecewise affine system [30]. However, much of the analysis for these dynamics is for networks with either constant inputs or discrete-time models that do not directly reflect continuous-time models. For the problem of reference tracking considered here it is important to consider time-varying control inputs, particularly in networks composed of multiple subnetworks, where the interconnections are inherently time-varying.

Multiple forms of the reference tracking problem exist, dependent on the available information and the problem goal, including when the reference signal is known [31], [32] versus when it is unknown [33], [34]. This work belongs to the first class, where we wish for the network to exhibit the behavior specified by a known reference signal. This can be motivated by a variety of problems in the brain, including memory recollection, where a memory is saved as a given activity pattern and it is recalled by the brain network exhibiting that pattern [2].

To learn controls in an online fashion that solve the reference tracking problem for a brain network with a known reference, we employ machine learning techniques, which allow to consider large-scale complex networks, such as those seen in the brain [35]–[37]. In particular, we use reservoir computing [38], [39], which is based on training an output vector (the *readout*) from a Recurrent Neural Network (RNN), rather than its internal weights.

Reservoir computing has a variety of advantages over classical gradient-descent based RNN training, the main one being that the training process is faster and computationally inexpensive compared to standard RNN training [40]. Further, reservoir computing displays certain parallels with biological activity. First, it gives an interpretation as to how arbitrary cortical circuits without supervised adaptation are able to perform purposeful computations [41]. Second, the ability of a single reservoir to perform multiple tasks by training multiple output vectors corresponding with each task mimics neural circuits' ability to have multiple purposes [40]. Due to these parallels, reservoir computing has been used to study a variety of phenomena, including connectivity and memory tasks [42], cortical dynamics in monkeys [43] and seizure detection [44]. Reservoir computing has also been used as a controller for dynamical systems, in which the reservoir output is used to determine an input such that the system converges to a desired trajectory [45]. This has led to the framework of next-generation reservoir computing (NG-RC), proposed in [46], [47] for both prediction and control. In Section IV we provide an overview of the RC and NG-RC frameworks, but for more details we refer the reader to [40], [46], [48], [49] and the references therein.

Statement of Contributions: We study brain networks with behavior governed by linear-threshold dynamics with time-varying inputs. Our first contribution relates to the ability to achieve reference tracking for the LTN dynamics. We begin by providing conditions for existence and uniqueness of points such that the error dynamics vanish. Using this fact we illustrate that under certain conditions on the reference signal and the network structure, reference tracking can be achieved for LTN networks and provide examples of controls that result in tracking. Our second contribution relates to achieving reference tracking using the data-driven machine learning frameworks. Considering the problem of reference tracking in the context of selective inhibition and recruitment and seizure rejection, we use examples to show that the reservoir computing and next-generation reservoir computing frameworks can be used as controllers to achieve reference tracking. Our contributions expand on state of the art results by considering convergence to reference trajectories rather than constant equilibrium points and applying a machine learning approach.

Preliminaries: We let $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, denote the reals and nonnegative reals, resp. Vectors and matrices are identified by boldfaced letters. For vectors (matrices) $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ (resp. $\mathbb{R}^{n\times m}$), $\mathbf{x} \leq \mathbf{y}$ is the component-wise comparison. For two vectors $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$, $\mathbf{x} \oplus \mathbf{y} \in \mathbb{R}^{n+m}$ is the concatenation of vectors $\mathbf{x}$ and $\mathbf{y}$. The identity matrix of dimension n is $\mathbf{I}_n$. $\mathbf{0}_n$ and $\mathbf{0}_{n\times m}$ denote the n-vector and $n\times m$ matrix of zeros, resp. For $\mathbf{W} \in \mathbb{R}^{n\times n}$, we denote its element-wise absolute value, spectral radius, and induced 2-norm by $|\mathbf{W}|, \rho(\mathbf{W})$, and $\|\mathbf{W}\|$, resp. Similarly, we let $\|\mathbf{x}\|$ denote the 2-norm of a vector $\mathbf{x} \in \mathbb{R}^n$.

For $x \in \mathbb{R}$ and $m \in \mathbb{R}_{>0}$, $[x]_0^m$ denotes the threshold operator, defined by $[x]_0^m = \min\{\max\{x, 0\}, m\}$. For $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{m} \in \mathbb{R}^n_{>0}$, this operation is done component-wise as $[\mathbf{x}]_{\mathbf{0}}^{\mathbf{m}} = [[x_1]_0^{m_1}, \ldots, [x_n]_0^{m_n}]$. A matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$ is

- a *$\mathcal{P}$-matrix* if all principal minors of $\mathbf{W}$ are positive (denoted $\mathbf{W} \in \mathcal{P}$);
- *totally Hurwitz* if all principal submatrices of $\mathbf{W}$ are Hurwitz (denoted $\mathbf{W} \in \mathcal{H}$);
- *totally $\mathcal{L}$-stable* if there exists $\mathbf{P} \succ 0$ such that $(-\mathbf{I} + \mathbf{W}^\top\Sigma)\mathbf{P} + \mathbf{P}(-\mathbf{I} + \Sigma\mathbf{W}) \prec 0$ for $\Sigma = \mathrm{diag}(\sigma)$ and all $\sigma = \{0, 1\}^n$ (denoted $\mathbf{W} \in \mathcal{L}$);
- *absolutely Schur stable* if $\rho(|\mathbf{W}|) < 1$.

These matrix classes satisfy [18, Lemma II.3] the following inclusions:

- if $\mathbf{W}$ is absolutely Schur stable then $-\mathbf{I} + \mathbf{W} \in \mathcal{H}$;
- it $\|\mathbf{W}\| < 1$ then $\mathbf{W} \in \mathcal{L}$;
- if $\mathbf{W} \in \mathcal{L}$ then $-\mathbf{I} + \mathbf{W} \in \mathcal{H}$;
- if $-\mathbf{I} + \mathbf{W} \in \mathcal{H}$ then $\mathbf{I} - \mathbf{W} \in \mathcal{P}$.

II. Problem Formulation

We start by providing details on linear-threshold brain modeling following [7], [18], then formalize the problem considered in this work. We consider a brain network composed of n nodes whose activity are determined by a linear-threshold rate dynamics. These dynamics represent a model of the average firing rate of neuron populations, rather than showing the action potentials of individual neurons. The

linear-threshold network (LTN) dynamics is given by

$$\tau\dot{\mathbf{x}} = -\mathbf{x} + [\mathbf{W}\mathbf{x} + \mathbf{c}]_{\mathbf{0}}^{\mathbf{m}}, \tag{1}$$

where each component of $\mathbf{x} \in \mathbb{R}^n$ represents the firing rate of a population of neurons, $\mathbf{W} \in \mathbb{R}^{n\times n}$ represents the synaptic connectivity between populations, and $\mathbf{c}$ captures any inputs to the network. These inputs could be internal, representing connections from neurons not in the model, or external, such as either invasive or non-invasive neurostimulation signals. The constant $\mathbf{m}$ provides a threshold on the firing rate of the neurons, and τ is a diagonal matrix defining the timescale of the dynamics for each population.

The LTN dynamics can be represented as a state-dependent switched affine system, which has switching regions defined by the threshold term $[\mathbf{W}\mathbf{x} + \mathbf{c}]_{\mathbf{0}}^{\mathbf{m}}$. The dynamics has 3^N switching regions, each defined by a switching variable $\sigma \in \{0, \ell, s\}^N$, and are defined as follows

$$\begin{aligned}\Omega_\sigma = \{\mathbf{x} \mid\ & (\mathbf{W}\mathbf{x}+\mathbf{c})_i \in (-\infty, 0]\forall i \text{ s.t. } \sigma_i = 0,\\ & (\mathbf{W}\mathbf{x}+\mathbf{c})_i \in (0, \mathbf{m}_i)\forall i \text{ s.t. } \sigma_i = \ell, \text{ and}\\ & (\mathbf{W}\mathbf{x}+\mathbf{c})_i \in [\mathbf{m}_i, \infty]\forall i \text{ s.t. } \sigma_i = s\}.\end{aligned} \tag{2}$$

The threshold term can then be expressed over each of these regions using diagonal matrices Σ^ℓ and Σ^s. These are defined, for $q \in \{\ell, s\}$, as follows: $\Sigma^q_{ii} = 1$ if $\sigma_i = q$ and $\Sigma^q_{ii} = 0$ otherwise. This leads to the piecewise-affine form of the dynamics (1) being defined as

$$\tau\dot{\mathbf{x}} = (-\mathbf{I} + \Sigma^\ell\mathbf{W})\mathbf{x} + \Sigma^\ell\mathbf{c} + \Sigma^s\mathbf{m}, \quad \mathbf{x} \in \Omega_\sigma. \tag{3}$$

We make the following assumption regarding the synaptic weight matrix.

Assumption II.1. *Assume the weight matrix* $\mathbf{W}$ *satisfies*

1) $\det(\mathbf{W}) \neq 0$;
2) $\det(-\mathbf{I} + \Sigma^\ell\mathbf{W}) \neq 0$ *for each* Σ^ℓ *corresponding to a switching region* Ω_σ.

Note that this assumption does not come at the cost of biological plausibility, since the set of matrices which do not satisfy it has measure zero and is therefore not restrictive. Assumption II.1 ensures that the right-hand side of the piecewise-affine dynamics has unique solutions, allowing for the existence of well-defined equilibria.

Leveraging the piecewise-affine form of the LTN dynamics, one can identify conditions on the synaptic weight matrix $\mathbf{W}$ that ensure existence and uniqueness of equilibrium (EUE), and stability properties of the dynamics. For technical details on these results and other properties of the LTN dynamics we refer the reader to [18] for individual dynamics and to [19], [20] for hierarchical dynamics. Of particular interest to us are the following:

1) if $\mathbf{W}$ satisfies $\mathbf{I} - \mathbf{W} \in \mathcal{P}$, then for each constant $\mathbf{c}$ the dynamics (1) has a unique equilibrium;
2) if $\mathbf{W} \in \mathcal{L}$, then for all constants $\mathbf{c}$, the dynamics is globally exponentially stable to a unique equilibrium.

We note that the condition $\mathbf{W} \in \mathcal{L}$ is sufficient for global exponential stability (GES). In [18], it is hypothesized that $-\mathbf{I} + \mathbf{W} \in \mathcal{H}$ is a necessary and sufficient for GES, which is less conservative, however it is not proven. In such cases, the piecewise-affine nature of the dynamics implies that the equilibrium, as a function of the network input $\mathbf{c}$, can be described by a piecewise-affine map. These maps are particularly useful, when considering the properties of large interconnected linear-threshold networks, to express the interdependencies of the equilibria of specific subnetworks in terms of other subnetworks.

Beyond having the network converge to stable equilibria, for many applications it is instead desirable to have the network track a particular reference trajectory. This requires non-constant inputs $\mathbf{c}(t)$, designed in a way that makes the network activity converge to the desired trajectory. Due to the complexity of brain networks and the fact that their structure is not precisely known, designing such a controller analytically is challenging. This motivates the use of a data-driven learning techniques to determining appropriate control signals to achieve trajectory tracking. Here, we consider reservoir computing, which we discuss in detail in Section IV. We formalize the problem as follows.

Problem II.2. *Consider a network defined by the LTN dynamics* (1) *with state* $\mathbf{x}(t) \in \mathbb{R}^n$ *and reference signal* $\mathbf{r} : \mathbb{R} \to \mathbb{R}^n$. *Determine a control signal such that the network converges to the reference signal, i.e.,*

$$\lim_{t\to\infty} \|\mathbf{x}(t) - \mathbf{r}(t)\| = 0.$$

Remark II.3. (Trajectory Tracking for Selective Inhibition): Our previous work [19], [20] has studied a version of trajectory tracking for the LTN dynamics (1) in the context of selective inhibition and recruitment of hierarchical multi-layered networks. Selective inhibition means that a control is defined such that the network converges to a trajectory in which the state of a subset of nodes is driven to zero. In such interconnected networks, the equilibrium map for each layer of the network is described by a complex recursive combination of the equilibrium maps of the other network layers, which significantly complicates the precise determination of the control for tracking a specific trajectory. The data-driven approach taken here addresses this by determining the controls based only on the system output and desired reference trajectory, requiring no knowledge of the input-dependent equilibrium maps. •

III. Linear Threshold Networks and Reference Tracking

In this section we show that, given a reference trajectory, there exists a control that makes the LTN dynamics track it asymptotically. This result sets the basis for our forthcoming use of reservoir computing techniques to synthesize the controller.

Given a reference signal $\mathbf{r}$, consider the error with the system state, $\mathbf{e}(t) = \mathbf{x}(t) - \mathbf{r}(t)$. The corresponding LTN error dynamics is given by

$$\tau\dot{\mathbf{e}} = -\mathbf{e} + [\mathbf{W}\mathbf{e} + \mathbf{W}\mathbf{r}(t) + \mathbf{c}]_{\mathbf{0}}^{\mathbf{m}} - \mathbf{r}(t) - \tau\dot{\mathbf{r}}(t). \tag{4}$$

To show that reference tracking is achieved, our strategy proceeds by establishing that the origin is the only equilibrium of the error dynamics (4) and that it is GES. To reason about this, we first note that the error dynamics (4) is time-dependent, and therefore the set of points where the vector field vanishes changes with time. The following result provides conditions such that at each time, there is only one point where this happens. To ease the exposition, given $\mathbf{c} \in \mathbb{R}^n$, let

$$f(\mathbf{e}, t) = -\mathbf{e} + [\mathbf{W}\mathbf{e} + \mathbf{W}\mathbf{r}(t) + \mathbf{c}]_{\mathbf{0}}^{\mathbf{m}} - \mathbf{r}(t) - \tau\dot{\mathbf{r}}(t),$$

denote the time-dependent vector field defining the error dynamics.

Theorem III.1. (Existence and Uniqueness of Points where LTN Error Dynamics Vanishes): *Given $\mathbf{c} \in \mathbb{R}^n$, if $\mathbf{I} - \mathbf{W} \in \mathcal{P}$, then for each $\bar{t} \in \mathbb{R}_{\geq 0}$, there exists a unique point $\mathbf{e}^*(\bar{t})$ such that $f(\mathbf{e}^*(\bar{t}), \bar{t}) = \mathbf{0}$.*

Proof:
Let $\bar{t} \in \mathbb{R}_{\geq 0}$ be a fixed time and consider the associated time-invariant dynamics $\tau\dot{\mathbf{e}} = f(\mathbf{e}, \bar{t})$. Note that a point where this vector field vanishes corresponds to an equilibrium point of the time-invariant dynamics. Let us therefore show that for each time $\bar{t} \in \mathbb{R}_{\geq 0}$, a unique equilibrium point exists. For fixed $\bar{t} \in \mathbb{R}_{\geq 0}$, let $\mathbf{d}_1 = \mathbf{W}\mathbf{r}(\bar{t}) + \mathbf{c}$ and $\mathbf{d}_2 = \mathbf{r}(\bar{t}) + \tau\dot{\mathbf{r}}(\bar{t})$. This leaves us with the dynamics

$$\tau\dot{\mathbf{e}} = -\mathbf{e} + [\mathbf{W}\mathbf{e} + \mathbf{d}_1]_{\mathbf{0}}^{\mathbf{m}} - \mathbf{d}_2. \tag{5}$$

To show existence and uniqueness of an equilibrium of (5), we note that this dynamics can be written in a similar piecewise-affine form to the original LTN dynamics (3),

$$\tau\dot{\mathbf{e}} = (-\mathbf{I} + \Sigma^{\ell}\mathbf{W})\mathbf{e} + \Sigma^{\ell}\mathbf{d}_1 + \Sigma^{s}\mathbf{m} - \mathbf{d}_2,$$

defined over the same switching regions given in (2) and with the same system matrix $(-\mathbf{I} + \Sigma^{\ell}\mathbf{W})$. Then, by following the proof of [18, Theorem IV.1], we have that this dynamics has a unique equilibrium for any values of $\mathbf{d}_1, \mathbf{d}_2$ if and only if $\mathbf{I} - \mathbf{W} \in \mathcal{P}$. ∎

Theorem III.1 implies that the curve $t \mapsto \mathbf{e}^*(t)$, which we term *nullcline curve*, is well-defined. Note that a constant nullcline curve in fact corresponds to an equilibrium of the original time-dependent dynamics (4).

To achieve lossless reference tracking, $\mathbf{r}(t)$ must lie in $[\mathbf{0}, \mathbf{m}]$ for all t due to the bounding of the dynamics. We begin by considering open-loop tracking before later providing a closed-loop control. Consider the control $\mathbf{c}_{\text{ol}}(t) = -\mathbf{W}\mathbf{r}(t) + \mathbf{r}(t) + \tau\dot{\mathbf{r}}(t)$. With this control, the error model (4) becomes

$$\tau\dot{\mathbf{e}} = -\mathbf{e} + [\mathbf{W}\mathbf{e} + \mathbf{r}(t) + \tau\dot{\mathbf{r}}(t)]_{\mathbf{0}}^{\mathbf{m}} - \mathbf{r}(t) - \tau\dot{\mathbf{r}}(t). \tag{6}$$

The next result identifies conditions such that the dynamics (1) converges exponentially to the reference trajectory $\mathbf{r}(t)$ by showing that the nullcline curve is constant at the origin, $\mathbf{e}^*(t) = 0$ for all $t \in \mathbb{R}_{\geq 0}$, and that the origin is GES.

Theorem III.2. (Reference Tracking for LTN Dynamics with Open-Loop Control): *Consider the open-loop LTN error dynamics* (6). *If $\mathbf{W} \in \mathcal{L}$ and $\mathbf{r}(t) + \tau\dot{\mathbf{r}}(t) \in [\mathbf{0}, \mathbf{m}]$ for all $t \in \mathbb{R}_{\geq 0}$, the origin is GES under the dynamics* (6).

Proof:
Since $\mathbf{r}(t) + \tau\dot{\mathbf{r}}(t) \in [\mathbf{0}, \mathbf{m}]$, it is immediate that the origin satisfies $f(\mathbf{0}, t) = \mathbf{0}$ for all $t \in \mathbb{R}_{\geq 0}$. Then, since $\mathbf{W} \in \mathcal{L}$ implies $\mathbf{I} - \mathbf{W} \in \mathcal{P}$, by Theorem III.1, the nullcline curve is unique and therefore constant. To conclude the proof, we show that the origin is GES under this dynamics, following a similar argument to [50, Theorem IV.8].

Let $\mathbf{W}_i$ denote the i'th row of $\mathbf{W}$. After some manipulations, we can rewrite the error dynamics as

$$\tau\dot{\mathbf{e}}(t) = (-\mathbf{I} + \mathbf{M}(t)\mathbf{W})\mathbf{e}(t),$$

where $\mathbf{M}(t)$ is a diagonal matrix given by

$$\mathbf{M}_{ii}(t) = \begin{cases} \frac{[\mathbf{W}_i\mathbf{e} + (\mathbf{r} + \tau\dot{\mathbf{r}})_i]_{\mathbf{0}}^{\mathbf{m}_i} - [(\mathbf{r} + \tau\dot{\mathbf{r}})_i]_{\mathbf{0}}^{\mathbf{m}_i}}{\mathbf{W}_i\mathbf{e}} & \text{if } \mathbf{W}_i\mathbf{e} \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Since the threshold operator is monotonically increasing and Lipschitz with constant one, we have $\mathbf{M}_{ii}(t) \in [0, 1]$. Therefore, $\mathbf{M}(t)$ lies in the convex hull of $\{\Sigma^{\ell}(\sigma)\}_{\sigma \in \{0,\ell\}^n}$ for all t. Thus, there exists a time-varying convex combination $(\alpha_\sigma(t))_{\sigma \in \{0,\ell\}^n}$ such that

$$\mathbf{M}(t) = \sum_{\sigma \in \{0,\ell\}^n} \alpha_\sigma(t)\Sigma^{\ell}, \quad t \geq 0.$$

Now, since $\mathbf{W} \in \mathcal{L}$, there exists a matrix $\mathbf{P} \succ 0$ and a scalar $\lambda > 0$ such that

$$(-\mathbf{I} + \mathbf{W}^{\top}\Sigma^{\ell})\mathbf{P} + \mathbf{P}(-\mathbf{I} + \Sigma^{\ell}\mathbf{W}) \leq -\lambda\mathbf{I}, \quad \forall \sigma \in \{0,\ell\}^n.$$

Consider then the candidate quadratic Lyapunov function $V(\mathbf{e}) = \mathbf{e}^{\top}\mathbf{P}\mathbf{e}$. Its derivative along the dynamics satisfies

$$\begin{aligned} \tau\dot{V}(\mathbf{e}(t)) &= \mathbf{e}^{\top}[(-\mathbf{I} + \mathbf{W}^{\top}\mathbf{M}(t))\mathbf{P} + \mathbf{P}(-\mathbf{I} + \mathbf{M}(t)\mathbf{W})]\mathbf{e} \\ &= \mathbf{e}^{\top}\Big[\sum_{\sigma \in \{0,\ell\}^n} \alpha_\sigma(t)\big[(-\mathbf{I} + \mathbf{W}^{\top}\Sigma^{\ell})\mathbf{P} + \mathbf{P}(-\mathbf{I} + \Sigma^{\ell}\mathbf{W})\big]\Big]\mathbf{e} \\ &\leq -\lambda\|\mathbf{e}\|^2 \leq -\frac{\lambda}{\rho(\mathbf{P})}V(\mathbf{e}(t)), \end{aligned}$$

which ensures the origin is GES under (4), see e.g., [51, Theorem 4.10]. ∎

Remark III.3. (Lack of Uniqueness of Input Map for Reference Tracking): We note that, while the input $\mathbf{c}_{\text{ol}}(t)$ used for Theorem III.2 is sufficient for ensuring GES of the origin for the error dynamics, and hence reference tracking, it is not unique. For example, due to the thresholding of the LTN dynamics, any (constant) reference trajectory that lies on the lower or upper threshold can be converged to using a higher magnitude control (either positive or negative, depending on the network and which threshold) than the one defined in $\mathbf{c}_{\text{ol}}(t)$ in order to drive the signal to the boundary. •

The open-loop control used for Theorem III.2 requires the network to satisfy $\mathbf{W} \in \mathcal{L}$ to achieve reference tracking. However, this condition is not always met, as there exist many brain networks that are not stable on their own, see

e.g., [18], [52]. Our next result shows that tracking can still be achieved with unstable synaptic weight matrices using feedback control of the form $\mathbf{c}(t) = \mathbf{K}\mathbf{x}(t) + \mathbf{d}(t)$, where $\mathbf{K} \in \mathbb{R}^{n\times n}$ is the feedback gain. Note that $\mathbf{K}$ may have rows entirely composed of zeros, corresponding to nodes in the network that cannot be directly impacted by feedback. We consider the control $\mathbf{c}_{\text{cl}}(t) = \mathbf{K}\mathbf{x}(t) - (\mathbf{W}+\mathbf{K})\mathbf{r}(t) + \mathbf{r}(t) - \tau\dot{\mathbf{r}}(t)$, which gives the closed-loop error dynamics

$$\tau\dot{\mathbf{e}} = -\mathbf{e} + [(\mathbf{W}+\mathbf{K})\mathbf{e} + \mathbf{r}(t) - \tau\dot{\mathbf{r}}(t)]_{\mathbf{0}}^{\mathbf{m}} - \mathbf{r}(t) - \tau\dot{\mathbf{r}}(t). \tag{7}$$

The next result provides conditions for the dynamics (1) to converge exponentially to the reference trajectory $\mathbf{r}(t)$ using the closed-loop control $\mathbf{c}_{\text{cl}}$ by by showing that the nullcline curve is constant at the origin, $\mathbf{e}^*(t) = 0$ for all $t \in \mathbb{R}_{\geq 0}$, and that the origin is GES.

Theorem III.4. (Reference Tracking for LTN Dynamics with Closed-Loop Control): *Consider the closed-loop error dynamics* (7)*. If* $\mathbf{W}+\mathbf{K} \in \mathcal{L}$ *and* $\mathbf{r}(t) + \tau\dot{\mathbf{r}}(t) \in [\mathbf{0}, \mathbf{m}]$ *for all* $t \in \mathbb{R}_{\geq 0}$*, the origin is GES under the dynamics* (7)*.*

Proof:
The dynamics (7) corresponds to the open-loop error dynamics (6) with the synaptic weight matrix $\mathbf{W}+\mathbf{K} = \tilde{\mathbf{W}}$. By Theorem III.2, if $\tilde{\mathbf{W}} \in \mathcal{L}$, the origin is GES under the dynamics (6). Therefore, with the chosen feedback control, the origin is GES under the closed-loop dynamics (7). ■

Remark III.5. (Reference Tracking in Under-actuated LTN Systems): In the results above, we have assumed that the network is fully actuated for the purpose of determining a control such that reference tracking can be achieved. For systems that are not fully actuated, i.e., the control is of the form $\mathbf{B}\mathbf{c}(t)$, where $\mathbf{B}$ is a diagonal matrix with zeros indicating components of the network that cannot be actuated, it is generally not possible to achieve reference tracking in all components. If $\mathbf{W} \in \mathcal{L}$, then the non-actuated component follows a trajectory that depends on the interconnection with the other network components rather than the desired reference signal. The only case when the under-actuated component i will converge to the reference trajectory is if $[\mathbf{W}_i\mathbf{r}(t)]_{\mathbf{0}}^{\mathbf{m}} = \mathbf{r}(t) + \tau\dot{\mathbf{r}}(t)$ for all $t \in \mathbb{R}_{\geq 0}$. We note that the actuated components of the network will still converge to their desired reference under the controls of Theorems III.2 or III.4. •

Having established reference tracking under the LTN dynamics with both open-loop (cf. Theorem III.2) and closed-loop (cf. Theorem III.4) controls, we note that the implementation of the corresponding inputs, either as inputs from other neuronal populations or through neurostimulation, is challenging, and becomes increasingly unrealistic with larger network dimensions, characteristic of brain modeling. First, evaluating the controls provided is difficult because it requires full knowledge of the synaptic weight matrix $\mathbf{W}$. Due to both the complexity of the brain and the difficulty in measuring the individual impact of individual neuron populations, particularly those deep in the brain [53], this is unrealistic. Beyond knowledge of its structure, the results require that $\mathbf{W}$ (or $\mathbf{W}+\mathbf{K}$) are in $\mathcal{L}$. This becomes computationally difficult to check as the network scales up, and the known sufficient conditions to ensure being in $\mathcal{L}$ become increasingly conservative. As such, for large networks, it is preferable to divide the network into layers based on the timescales, where each layer has an individual constant τ_i. In this case, dependent on the timescales, the results of Theorems III.2 and III.4 are no longer directly applicable due to the activity of the interconnection terms affecting the dynamics of each layer. In such a case, one can use a singular perturbation argument [19], but the recursively-defined equilibrium maps involved make the determination of the exact control signals particularly challenging.

Second, explicitly computing a direct control signal is not a realistic representation of the way the brain is believed to operate. However, implementing a data-driven approach, where all that is needed is the desired reference signal and the control is determined through internal dynamics, seems more realistic and not as computationally expensive. In the following section we provide an explanation of the reservoir computing and next-generation reservoir computing frameworks. We demonstrate both can be used to achieve reference tracking in linear-threshold networks later in Sections V and VI.

IV. Reservoir Computing

Here we provide an overview of the mathematical basis of the reservoir computing framework, which we later apply to control synthesis for reference tracking in LTN networks. We begin by discussing reservoir computing for predicting system outputs and for control before considering the same problems using next-generation reservoir computing. We finish by discussing parameter selection and comparing the two frameworks.

A. Reservoir Computing for Prediction

We first overview the basic structure of a reservoir computer for predicting the outputs of an unknown system. Assume we have a dynamical system defined by

$$\mathbf{y}(t) = f(\mathbf{c}(t)) \tag{8}$$

where $\mathbf{y} \in \mathbb{R}^m$ is the system output, $\mathbf{c} \in C \subseteq \mathbb{R}^n$ is the input and the driving function $f : \mathbb{R}^n \to \mathbb{R}^m$ for the system is unknown. We define a reservoir as a dynamical system

$$\dot{\mathbf{x}}(t) = F(\mathbf{J}\mathbf{x}(t) + \mathbf{J}_{\text{in}}\mathbf{c}(t)), \tag{9}$$

where $\mathbf{x} \in X \subseteq \mathbb{R}^N$ is the internal state of the reservoir, $\mathbf{J} \in \mathbb{R}^{N\times N}$ is a matrix that provides the reservoir structure, and $\mathbf{J}_{\text{in}} \in \mathbb{R}^{N\times n}$ in an input matrix. We assume that $N \gg n$ and $F : X \times C \to X$ is a nonlinear activation function. The goal of the reservoir computing framework is to use the activity of the high-dimensional reservoir dynamical system (9) to estimate the outputs of the unknown system (8).

The output estimate is defined

$$\tilde{\mathbf{y}}(t) = \mathbf{J}_{\text{out}}\mathbf{x}(t),$$

where $\mathbf{J}_{\text{out}} \in \mathbb{R}^{m\times N}$ is an output vector trained to achieve an accurate estimate of the unknown system.

The key component for the reservoir computing framework to be able to provide a successful estimate is the *echo state property* [40]. This property is dependent on the activation function F and the reservoir matrix $\mathbf{J}$. For the activation function, we require [54] that $F : X \times C \to X$ is defined on compact sets X and C. The compactness of the state space X is given for the most commonly used activation functions in machine learning, such as $\tanh$ or the logistic sigmoid function, and importantly, also for the linear-threshold function $[\cdot]_{\mathbf{0}}^{\mathbf{m}}$ considered in this paper. We assume the compactness of the input set C, which is realistic in most applications.

To define the echo state property, we introduce the following notations. We will denote the reservoir dynamical system (9), which combines the activation function, F, and the reservoir matrix, $\mathbf{J}$, by $\mathbf{F}(\mathbf{x}(t), \mathbf{c}(t)) = F(\mathbf{J}\mathbf{x}(t) + \mathbf{J}_{\text{in}}\mathbf{c}(t))$. Let $X^{+\infty} \coloneqq \{\mathbf{x}^{+\infty} = \{\mathbf{x}(t)\}_{t=0}^{\infty} \mid \mathbf{x}(t) \in X,\ \forall t \geq 0\}$ and $C^{+\infty} \coloneqq \{\mathbf{c}^{+\infty} = \{\mathbf{c}(t)\}_{t=0}^{\infty} \mid \mathbf{c}(t) \in C,\ \forall t \geq 0\}$ denote sets of right infinite state and input sequences. A right infinite state sequence $\mathbf{x}^{+\infty}$ is *compatible* with input state sequence $\mathbf{c}^{+\infty}$ when $\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}(t), \mathbf{c}(t))$ for all $t \geq 0$.

Definition IV.1. (Echo State Property [54]): *A reservoir $\mathbf{F} : X \times C \to X$ defined on compact sets X and C satisfies the echo state property with respect to C if and only if for any right infinite input sequence $\mathbf{c}^{+\infty} \in C^{+\infty}$ and any two right infinite state vector sequences $\mathbf{x}_1^{+\infty}, \mathbf{x}_2^{+\infty} \in X^{+\infty}$ compatible with $\mathbf{c}^{+\infty}$, there exists a sequence $\{\delta_t\}_{t=0}^{\infty}$ such that $\|\mathbf{x}_1(t) - \mathbf{x}_2(t)\| \leq \delta_t$, where $\lim_{t\to\infty} \delta_t = 0$.*

The echo state property relates to the asymptotic convergence of the state of the reservoir, which is influenced by a driving input. It can be thought of as the concept of fading memory, in that trajectories of the reservoir should converge to the same point given the same input, regardless of the prior history of the reservoir.

Necessary and sufficient conditions for the echo state property to hold are dependent on both the reservoir matrix, $\mathbf{J}$, and the activation function, F. Regardless of the activation function, guaranteeing the echo state property depends on the stability of the reservoir matrix[1].

What remains is to train the reservoir computing framework to get an accurate estimate $\tilde{\mathbf{y}}$ of the output $\mathbf{y}$. The defining feature of reservoir computing is that we only train the output vector $\mathbf{J}_{\text{out}}$, rather than the internal weights of the reservoir matrix, $\mathbf{J}$. In order to train the reservoir output matrix, we take a training input $\{\mathbf{c}_T(t)\}$ and drive the system (8) using this input, getting a driven training output timeseries $\{\mathbf{y}_T(t)\}$. Then, using the training signal $\{\mathbf{c}_T(t)\}$, we drive the reservoir (9) to get a timeseries of driven reservoir states $\{\mathbf{x}_T(t)\}$. Compiling the set of training outputs and driven reservoir states into matrices $\mathbf{Y}_T$ and $\mathbf{X}_T$, we compute the output matrix $\mathbf{J}_{\text{out}}$ using a Tikhonov regularization,

$$\text{argmin}_{\mathbf{J}_{\text{out}}} \|\mathbf{Y}_T - \mathbf{J}_{\text{out}}\mathbf{X}_T\|^2 + \|\beta\mathbf{J}_{\text{out}}\|^2, \tag{10}$$

where $\beta > 0$ is a regularization parameter. The performance of the reservoir computer for predicting outputs can then be measured by using the trained system to provide a prediction of a second set of arbitrary inputs and comparing the accuracy with the true outputs for this set. The main parameter for how well the reservoir computer works is the size of the reservoir, N. As N increases, the higher-dimensional reservoir can then exhibit an increasingly large number of possible behaviors, especially relatively to the size of the system being predicted. Then, in the process of training the system, we are able to relate these behaviors to the system trajectories through the linear output operator $\mathbf{J}_{\text{out}}$. Based on this, with a randomly defined reservoir, by increasing N sufficiently, the reservoir computer can achieve good prediction of the system outputs. Parameters such as the reservoir weight matrix $\mathbf{J}$, the input matrix $\mathbf{J}_{\text{in}}$, the training input, and the regularization parameter β can also impact the quality of the output prediction.

B. Reservoir Computing as a Controller

We are interested in applying a reservoir computer as a controller for a dynamical system, with the goal of controlling the system to a given reference trajectory [45]. Consider a dynamical system

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{c}(t)) \qquad \mathbf{y}(t) = g(\mathbf{x}(t)), \tag{11}$$

where $\mathbf{x}(t), \mathbf{c}(t) \in \mathbb{R}^n$ are the state variables and inputs, respectively, and $\mathbf{y}(t) \in \mathbb{R}^m$ is the output of the system. The functions f and g define the system evolution and measurable outputs, and potentially are unknown. Let $\mathbf{r}(t) \in \mathbb{R}^m$ denote a desired reference trajectory for the output. The reservoir dynamics is defined as

$$\dot{\mathbf{u}}(t) = F(\mathbf{J}\mathbf{u}(t) + \mathbf{J}_{\text{in}}\mathbf{y}(t) + \mathbf{J}_{\text{ref}}\mathbf{r}(t+\delta)), \tag{12}$$

where $\mathbf{u}(t) \in \mathbb{R}^N$ is the reservoir state, $\mathbf{J} \in \mathbb{R}^{N\times N}$ defines the internal reservoir connections, $\mathbf{J}_{\text{in}} \in \mathbb{R}^{N\times m}$ scales the system output $\mathbf{y}(t)$ into the reservoir, $\mathbf{J}_{\text{ref}} \in \mathbb{R}^{N\times m}$ scales the reference signal, and $\delta > 0$ dictates how far ahead we provide a desired reference value. The function F is the activation function, and we assume that F and $\mathbf{J}$ are chosen such that the reservoir has the echo state property. We then connect the output of the reservoir dynamics (12) with the input of the system (11) by defining

$$\mathbf{c}(t) = \mathbf{J}_{\text{out}}\mathbf{u}(t).$$

Figure 1 illustrates this setup.

To train the reservoir computer, we use an open-loop version of the schematic shown in Figure 1 with a training input and a delay on the output. Here, instead of inputting the reference trajectory into the reservoir, we use the current system output, $\mathbf{y}(t)$, along with the future output, $\mathbf{y}(t+\delta)$, which is directly computable from the system and training

[1] Much of the reservoir computing literature is in discrete time, which also impacts sufficient conditions for the echo state property. In the discrete-time case with $F(\cdot) = \tanh(\cdot)$, a sufficient condition for the echo state property is that $\mathbf{J}$ is diagonally Schur stable.

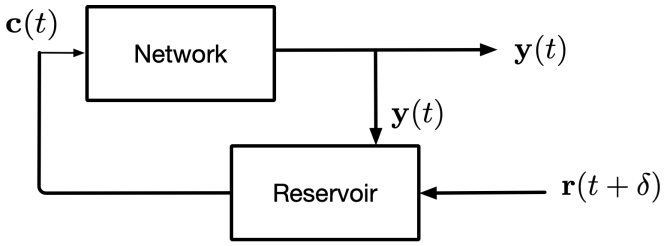

FIGURE 1: A reservoir computer steers the dynamical system (11) to a reference signal $\mathbf{r}(t)$.

input. The future output acts in the role of the reference signal, with the goal of determining how the training signal moves the system from $\mathbf{y}(t)$ to $\mathbf{y}(t+\delta)$. The reservoir training dynamics is then

$$\dot{\mathbf{u}}(t) = F(\mathbf{J}\mathbf{u}(t) + \mathbf{J}_{\text{in}}\mathbf{y}(t) + \mathbf{J}_{\text{ref}}\mathbf{y}(t+\delta)).$$

Figure 2 illustrates this training setup. The output vector

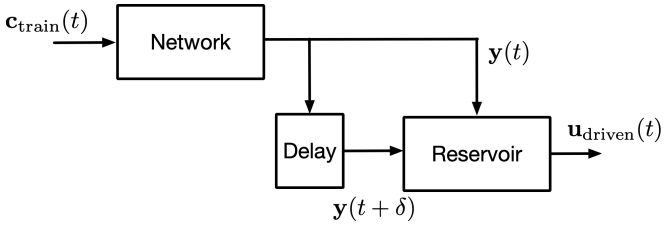

FIGURE 2: Schematic for training a reservoir computer to steer the dynamical system (11) to a desired reference trajectory.

for the reservoir computer, $\mathbf{J}_{\text{out}}$, is then trained using the Tikhonov regularization (10) to minimize the difference between $\mathbf{c}_{\text{train}}$ and the reservoir prediction, $\mathbf{J}_{\text{out}}\mathbf{u}_{\text{driven}}$. This setup can be extended by providing different information to the reservoir, either about the reference signal $\mathbf{r}(t)$, or the system output, $\mathbf{y}(t)$. A common choice is providing the reservoir with $\dot{\mathbf{y}}(t)$. However, this is typically discretely estimated from $\mathbf{y}(t)$ and as such it is just a transform from providing $\mathbf{y}(t)$ [45]. Instead, in our treatment, we provide additional information regarding the reference signal, in particular $\dot{\mathbf{r}}(t)$, which we can compute since it is known in advance. This leads to the final schematic for the reservoir controller in Figure 3.

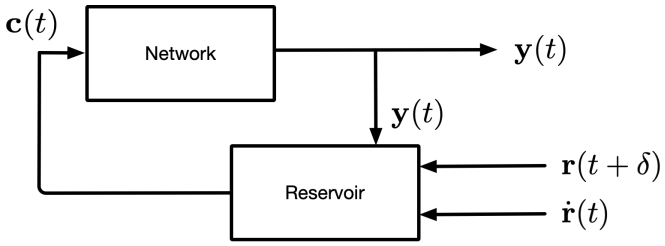

FIGURE 3: Schematic for using a reservoir computer as a controller incorporating additional information on the reference signal $\mathbf{r}(t)$.

C. Next-Generation Reservoir Computing

The observation, cf. [55], that the underlying equations of the reservoir computing framework have similarities with nonlinear vector autoregression (NVAR) and dynamic mode decomposition led to the construction of the next-generation reservoir computing (NG-RC) framework [46] based on NVAR. We explain this framework below, both for prediction and control.

1) NG-RC for Prediction

Similarly to the classical RC, we consider the dynamical system (8). Instead of defining a reservoir, we define two feature vectors, $\mathcal{O}_{\text{lin},t}$ and $\mathcal{O}_{\text{nl},t}$, defined as follows: $\mathcal{O}_{\text{lin},t} = \mathbf{c}(t) \oplus \mathbf{c}(t-i_1) \oplus \cdots \oplus \mathbf{c}(t-i_p)$ is the linear feature vector defined for p discretely sampled prior points of the system inputs. $\mathcal{O}_{\text{nl},t}$ is a nonlinear feature vector that is an arbitrary nonlinear function of the vector $\mathcal{O}_{\text{lin},t}$. Any nonlinear function can be chosen, albeit it is common [46] to employ the monomials up to some order k. These feature vectors are then used to predict the outputs of the unknown system (8) in place of the activity of the reservoir (11).

The two feature vectors are concatenated, commonly with an additional constant $d \in \mathbb{R}$, to give $\mathcal{O}_{\text{total},i} = d \oplus \mathcal{O}_{\text{lin},i} \oplus \mathcal{O}_{\text{nl},i}$ and the output of (8) is predicted by $\tilde{\mathbf{y}}(t) = \mathbf{J}_{\text{out}}\mathcal{O}_{\text{total},t}$. To achieve an accurate prediction the output vector $\mathbf{J}_{\text{out}}$ is predicted with a Tikhonov regularization as

$$\text{argmin}_{\mathbf{J}_{\text{out}}} \|\mathbf{y}(t) - \mathbf{J}_{\text{out}}\mathcal{O}_{\text{total},t}\|^2 + \|\beta\mathbf{J}_{\text{out}}\|^2, \qquad (13)$$

which requires running the system with a minimum of i_p+1 inputs in order to fully determine the vector $\mathcal{O}_{\text{total},t}$. This setup, with one prior time step and quadratic monomials, is shown in Figure 4.

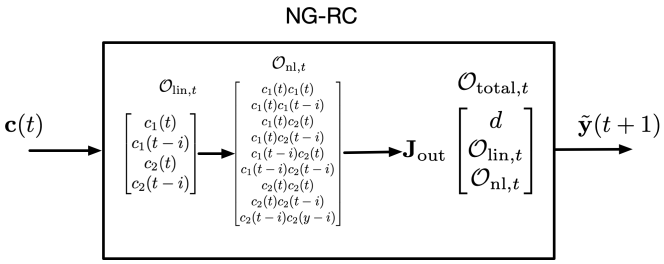

FIGURE 4: Schematic for using next-generation reservoir computing to predict system outputs with quadratic monomials used for the nonlinearity. The matrix $\mathbf{J}_{\text{out}}$ is trained with a known input/output sequence $\{(\mathbf{c}(t), \mathbf{y}(t))\}$ with a minimum length of two points, before being able to predict future outputs based on the input.

2) NG-RC for Control

Similarly to the RC, we are interested in applying the NG-RC as a controller to bring a dynamical system to a desired reference trajectory. To do so, we use an open-loop training period followed by a closed-loop control period. To construct the NG-RC controller we follow the approach in [47] and consider an extension to employ multiple NG-RCs, which we leverage in our simulations.

We consider the control system given by (11). We define linear and nonlinear feature vectors $\mathcal{O}_{\text{lin},t}$ and $\mathcal{O}_{\text{nl},i}$. Unlike NG-RC for prediction, $\mathcal{O}_{\text{lin},t}$ is not dependent on prior time steps and instead $\mathcal{O}_{\text{lin},t} = \mathbf{y}(t)$, the observable outputs. The

nonlinear feature vector is then a function of the outputs, where again we typically use polynomial expressions.

What differs between the prediction and control setups is the definition of the total feature vector, $\mathcal{O}_{\text{total},t}$. Rather than just combining the linear and nonlinear feature vectors with a constant, we also include the system input, giving $\mathcal{O}_{\text{total},t} = \mathbf{c}(t) \oplus d \oplus \mathcal{O}_{\text{lin},t} \oplus \mathcal{O}_{\text{nl},t}$. With this feature vector, Tikhonov regularization (13) is used to learn the output vector $\mathbf{J}_{\text{out}}$ in an open-loop fashion such that the prediction is given by $\hat{\mathbf{y}}(t) = \mathbf{J}_{\text{out}}\mathcal{O}_{\text{total},t}$, as in the top schematic of Figure 5.

Then, for the purpose of defining a control, we split the output and feature vector in the following way.

$$\mathbf{J}_{\text{out}}\mathcal{O}_{\text{total},t} = \mathbf{J}^X_{\text{out}}\begin{bmatrix} d \\ \mathcal{O}_{\text{lin},t} \\ \mathcal{O}_{\text{nl},t} \end{bmatrix} + \mathbf{J}^{\mathbf{c}}_{\text{out}}\mathbf{c}(t) = \mathbf{J}^X_{\text{out}}\mathcal{O}_{X,t} + \mathbf{J}^{\mathbf{c}}_{\text{out}}\mathbf{c}(t).$$

From here, a closed-loop system is created where the matrices $\mathbf{J}^X_{\text{out}}$ and $\mathbf{J}^{\mathbf{c}}_{\text{out}}$, along with the reference signal $\mathbf{r}(t)$, are used to define the control term. With the error term $\mathbf{e}(t) = \mathbf{x}(t) - \mathbf{r}(t)$, the control is defined by

$$\mathbf{c}(t) = (\mathbf{J}^{\mathbf{c}}_{\text{out}})^{-1}[\mathbf{r}(t+\delta) - \mathbf{J}^X_{\text{out}}\mathcal{O}_{X,t} + \mathbf{K}\mathbf{e}(t)], \tag{14}$$

where $\mathbf{K}$ is a proportional control matrix determined by trial-and-error to optimize performance. The closed-loop control period is shown in the bottom schematic of Figure 5. This control is derived in [47] for a discrete-time system with the control input entering linearly, i.e., of the form $\mathbf{y}(t+1) = \mathbf{F}(X(t)) + \mathbf{B}\mathbf{c}(t)$. While the LTN dynamics is not of this form, our simulations in Sections V and VI show that this control still achieves satisfactory performance.

We can also extend this approach to construct a controller out of multiple NG-RCs rather than just one. In this approach each successive NG-RC is used to minimize the error remaining from the control determined by the prior NG-RC. The additional NG-RC layers are added by training output vectors $\mathbf{J}_{\text{out},i}$ for each layer with the feature vector $\mathcal{O}_{\text{total},t}$ such that the output prediction is

$$\tilde{\mathbf{y}}(t) = \sum_{i=1}^{M} \mathbf{J}_{\text{out},i}\mathcal{O}_{\text{total},t},$$

where M is the number of layers. The layers are trained successively, with each layer being dependent on the output of the layers before. Each successive layer is trained using Tikhonov regularization on the error between the true output and the predicted output by the sum of the layers before. Formally, if we denote $\tilde{\mathbf{y}}_{1:i}(t)$ to be the predicted output from the first i layers, the output vector for layer $i+1$ is determined by the following

$$\mathbf{J}^{i+1}_{\text{out}} = \operatorname{argmin} \left\| (\tilde{\mathbf{y}}_{1:i}(t) - \mathbf{y}(t)) - \mathbf{J}^{i+1}_{\text{out}}\mathcal{O}_{\text{total},t} \right\|^2 + \left\| \beta \mathbf{J}^{i+1}_{\text{out}} \right\|^2.$$

The control signal is then given by

$$\mathbf{c}(t) = (\sum_{i=1}^{m} \mathbf{J}^c_{\text{out},i})^{-1}(\mathbf{r}(t+\delta) + \sum_{j=1}^{m}(\mathbf{J}^X_{\text{out},j}\mathcal{O}_{X,t}) + \mathbf{K}\mathbf{e}(t)).$$

We note that this process could be modified to use different feature vectors across the multiple NG-RC's. The process for computing the output vectors would be the same, with some differences appearing in the determination of the final control signal.

Remark IV.2. (Comparison of Parameter Selection in RC and NG-RC): The selection of parameters is of paramount importance to the performance of the RC and NG-RC algorithms. One of the benefits of RC compared to traditional machine learning algorithms is that there are significantly fewer parameters to optimize [48]. This is even more so for NG-RC, cf. [46]. In comparing RC and NG-RC, one of the biggest differences is in the number of parameters that need to be selected, and the difficulty in their selection. In RC, the parameters to be selected are the reservoir matrix, $\mathbf{J}$, the activation function, $\mathbf{F}$, the input vector $\mathbf{J}_{\text{in}}$, the regularization parameter, β and the training signal, $\mathbf{c}_{\text{train}}$. Of these, the most important is the selection of the reservoir matrix, which is typically done randomly, and its relation with the activation function to guarantee the echo state property. Despite significant research into choices of the reservoir [48], [49], [56], [57] an optimal choice is not known. In NG-RC for control, the parameters to be tuned are the nonlinear feature vector, $\mathcal{O}_{\text{nl},t}$, the proportional control vector $\mathbf{K}$, and the training signal, $\mathbf{c}_{\text{train}}$. In NG-RC for prediction, one needs to additionally choose the linear feature vector, $\mathcal{O}_{\text{lin},t}$. •

Remark IV.3. (System Output for Reference Tracking): We note that in in this section we have defined the reservoir computing framework for an arbitrary system output $\mathbf{y}(t)$, while in Problem II.2 we aim for reference tracking of the system state. As such, in the following applications we consider the specific case of $\mathbf{y}(t) = \mathbf{x}(t)$, which matches the problem description.

V. Application to Selective Inhibition and Recruitment

In this section, we provide our first illustration of the use of RC and NG-RC controllers to achieve reference tracking for linear-threshold networks. We consider the problem of selective inhibition and recruitment, and illustrate the recruitment of a subset of a network to a chosen reference signal for a network structure motivated by the hierarchical nature of the brain [58] and studies of selective attention [59]. We do this with both the RC and NG-RC controllers and include a comparison of their performance.

Selective inhibition and recruitment is the problem of reacting to stimuli by inhibiting task-irrelevant neuron populations to zero, while recruiting the remaining task-relevant neuron populations to a particular activity pattern. Due to the hierarchical nature of the brain [58], this problem has been studied in networks composed of subnetworks operating at different timescales, and leads to the following formalization of the network structure. Consider a network composed of N subnetworks, each of composed of n_i nodes and with corresponding timescale τ_i. We construct a hierarchy by organizing the subnetworks such that $\tau_N \leq \tau_{N-1} \leq \cdots \leq \tau_1$ and each subnetwork is connected only to the subnetworks directly above and below it in the hierarchy, cf. Figure 6. Subnetworks at the bottom of the network,

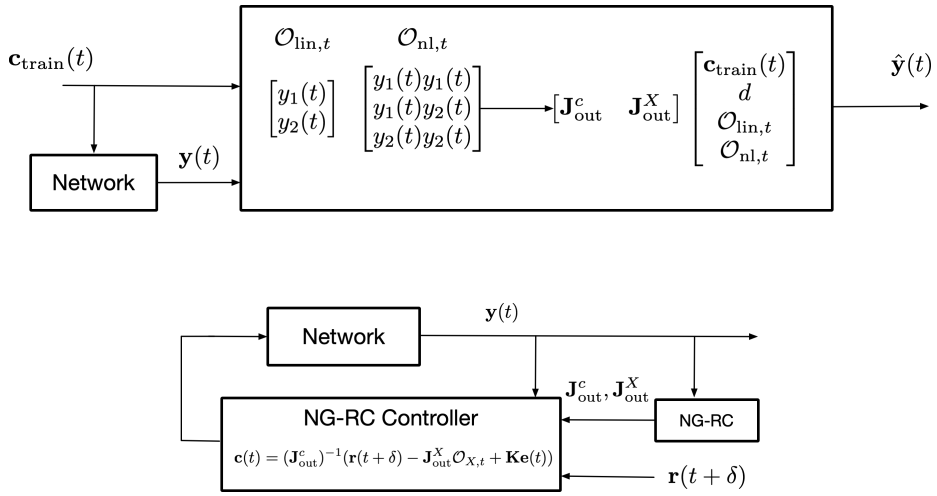

FIGURE 5: The top schematic shows the open-loop learning component of using the NG-RC as a controller. The closed-loop control component is shown in the second schematic, with the control given as in (14).

with fast timescales, represent regions in the brain that operate quickly, such as sensory areas, while the top of the network represents regions such as the prefrontal cortex, which operate relatively slowly.

The problem of selective inhibition and recruitment is then formalized as follows. For each subnetwork let $\mathbf{x}_i^1 \in \mathbb{R}^{r_i}$ denote the set of task-relevant nodes and $\mathbf{x}_i^0 \in \mathbb{R}^{n_i - r_i}$ denote the set of task-irrelevant nodes. Then, determine a control $\mathbf{c}_i^* : \mathbb{R}_{\geq 0} \to \mathbb{R}^{n_i}$ such that the task-relevant nodes are recruited to a non-zero reference trajectory $\mathbf{r}_i^* : \mathbb{R}_{\geq 0} \to \mathbb{R}^{n_i}$, i.e., $\mathbf{x}_i^1 = \mathbf{r}_i^*(t)$ and the task-irrelevant nodes are inhibited to zero, i.e., $\mathbf{x}_i^0(t) = \mathbf{0}$.

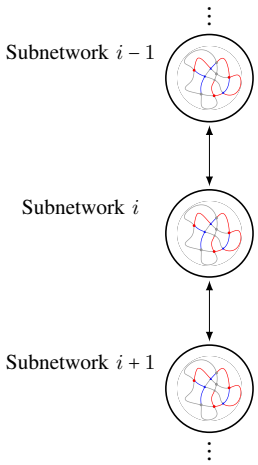

FIGURE 6: Hierarchical brain network considered for selective inhibition and recruitment. Nodes are divided between excitatory (red), inhibitory (blue), and those to be inhibited (gray). Edge colors match the direction of the node from which they originate, indicating that they provide either an excitatory (red) or inhibitory (blue) connection, while gray labels indicate that their node of origin has been inhibited and the connection provides no activity.

This problem has been addressed in [19], [20] for constant, recursively-defined reference trajectories $\mathbf{r}^*(t) = \mathbf{r}$. However, this involves the explicit computation of a control with complexity that increases significantly with scale based on the size and hierarchical nature of the network. Here, instead, we show how the problem can be solved in a data-driven way using the RC and NG-RC frameworks. Once the readout has been learned, the reservoir computer determines a control that produces the desired reference tracking behavior.

A. Setup

We consider a network composed of three subnetworks, each one being an excitatory-inhibitory pair. We then aim to recruit one node in each network, while inhibiting the other to zero using reservoir controllers. In this example, the controllers are representing the impact of other neuronal populations, outside the explicitly modeled ones, that impact behavior. The subnetworks considered are defined by the following randomly generated synaptic weight matrices:

$$\mathbf{W}_1 = \begin{bmatrix} 0.0112 & -0.9903 \\ 0.4101 & -0.5115 \end{bmatrix}, \quad \mathbf{W}_2 = \begin{bmatrix} 0.4614 & -0.7342 \\ 0.0950 & -0.5115 \end{bmatrix},$$
$$\mathbf{W}_3 = \begin{bmatrix} 0.1136 & -0.2110 \\ 0.7732 & -0.0800 \end{bmatrix}.$$

These networks are combined with the timescales $\tau_1 = 4$, $\tau_2 = 1$ and $\tau_3 = 1/3$ to create a hierarchy. The interconnections between the networks are also randomly generated, cf. Section C.

When tuning the RC and NG-RC controllers, as the size of the network increases, the number of parameters and difficulty in tuning them all concurrently increases drastically. As such, instead of training an RC or NG-RC to control the entire network at once, we begin by training each subnetwork individually. Following the determination of a controller for each individual layer, we train a second RC on the error remaining from using the first controller on the interconnected network.

We use a randomly generated 100-node reservoir, and provide inputs related to both the reference signal and its derivative. For the NG-RC, we use a nonlinear feature vector composed of the unique quadratic monomials, and

the constant term in the feature vector $\mathcal{O}_{\text{total},t}$ is equal to 0.5. The training signals for the networks are created by sampling a $\mathcal{N}(0, 0.1)$ distribution.

In the following sections we illustrate that the with this reservoir and NG-RC the proposed controllers provide successful convergence to provided reference signals. The plots in Figures 7-9 show this visually, with details and error values in the text.

B. Individual Layers

We begin by training each excitatory-inhibitory pair individually to track a given reference signal without the interconnections in the network. While training both RC and NG-RC, we tune the regularization parameter β (RC and NG-RC) and feedback vector $\mathbf{K}$ (NG-RC) for each layer. Figure 7 shows plots of both the RC and NG-RC converging to the desired reference signal. In each plot, the system evolves without control until $t = 25$ when the controller is turned on. Control parameters are provided in Table 1, along with the root-mean-square error (RMSE) between the reference and actual signal, calculated from the point the controller is turned on. Further parameters related to the training and control of the systems are discussed in Section D.

For the top layer, we recruit the excitatory node to the reference signal $\mathbf{r}^*(t) = \sin(\frac{\pi t}{100}) + 2$, while the inhibitory node is inhibited to zero, cf. Figure 7(a). Here we see that both the NG-RC and RC controllers track the reference signal successfully, though the RC works slightly better in terms of tracking, with a RMSE of 0.0293 versus a RMSE of 0.0401 for the NG-RC.

NG-RC				RC		
Layer	β	**K**	**RMSE**	**Layer**	β	**RMSE**
Top	0.5	-5	0.0401	Top	0.3	0.0293
Middle	1.2	-1	0.0805	Middle	0.5	0.0255
Bottom	0.7	-0.1	0.0752	Bottom	0.1	0.0835

TABLE 1: The parameters and errors for the NG-RC (left) and RC (right) controllers for each network layer. The NG-RC parameters are the regularization parameter β and the proportional control $\mathbf{K}$. For the RC we consider only the regularization parameter. For both controllers, the RMSE is between the reference signal and the network state, including both the excitatory and inhibitory nodes.

For the middle layer, we again recruit the excitatory node, this time to the reference signal $\mathbf{r}^*(t) = \sin(\frac{2\pi t}{100}) + 2$, cf. Figure 7(b). Again both the NG-RC and RC controllers result in the tracking of the reference signal, though for this network and timescale, the NG-RC fluctuates rapidly around the desired values rather than following it exactly. This is reflected in the RMSE, where the NG-RC takes a value of 0.0805 versus a value of 0.0255 for the RC controller.

For the bottom layer, we recruit the inhibitory node to the triangle wave with frequency $\frac{1}{100}$ Hz and amplitude 1, centered at 2, while inhibiting the excitatory node to zero, cf. Figure 7(c). Both the NG-RC and RC controllers result in effective recruitment to the desired signal, with the NG-RC (resp. RC) controlled system lying slightly above (resp. below) the reference signal. Here the RMSE for the NG-RC is slightly lower than for the RC, at 0.0752 versus 0.0835.

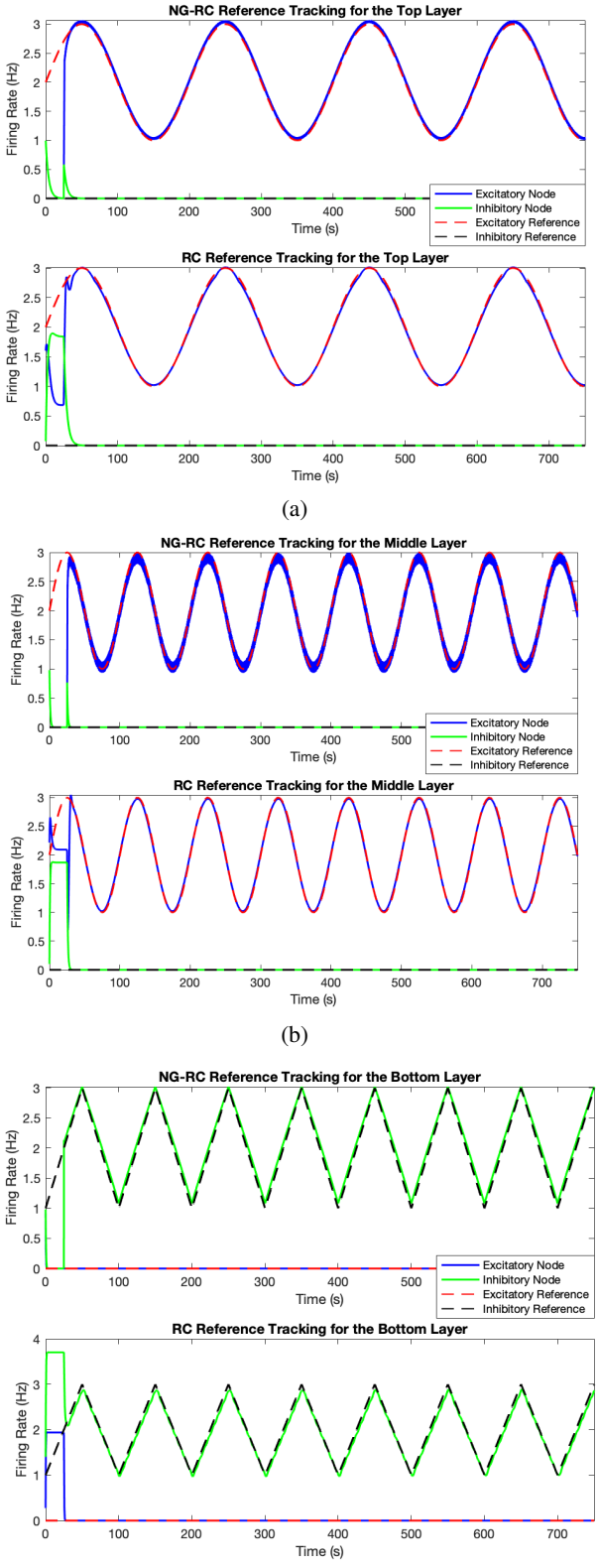

FIGURE 7: Selective inhibition and recruitment without considering interconnections between the layers using NG-RC and RC controllers for the subnetwork in the (a) top, (b) middle, and (c) bottom layers. Control parameters and the RMSE between the system and reference signal are given in Table 1.

C. Interconnected Network

We now consider an interconnected system defined by the subnetworks $\mathbf{W}_1, \mathbf{W}_2$, and $\mathbf{W}_3$. Due to the difficulty in tuning the parameters as the network size increases, we do this by using a multi-layer approach, as described for the NG-RC in Section IV. We use the reservoir controller as determined for the individual layers before training a second layer to cover the error introduced by the interconnection. Here we show that as the magnitude of the interconnection weight increases it becomes more difficult to control the overall network, and also illustrate that the NG-RC controller is more robust to changes in the weights of the network interconnections than the RC controller.

We consider a network $\mathbf{W}$ defined by $\mathbf{W}_{\text{layers}} + \gamma \mathbf{W}_{\text{connections}}$, with

$$\mathbf{W}_{\text{layers}} = \begin{bmatrix} \mathbf{W}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{W}_3 \end{bmatrix},$$

$$\mathbf{W}_{\text{connections}} = \begin{bmatrix} \mathbf{0} & \mathbf{W}_{12} & \mathbf{0} \\ \mathbf{W}_{21} & \mathbf{0} & \mathbf{W}_{23} \\ \mathbf{0} & \mathbf{W}_{32} & \mathbf{0} \end{bmatrix},$$

where $\mathbf{W}_{12}, \mathbf{W}_{21}, \mathbf{W}_{23}$ and $\mathbf{W}_{32}$ are interconnection matrices between the layers, while $\gamma \in \mathbb{R}_{\geq 0}$ weights the connection strength. The interconnection matrices are randomly generated and scaled such that $\|\mathbf{W}_{\text{connections}}\| \approx 0.01$. In this way, small γ values give a network that is close to having no connections.

For a interconnection weight $\gamma = 20$, Figure 8 shows that the two-layer NG-RC controller tracks the desired reference signals for each layer. This occurs after re-tuning weights, in particular the regularization parameter for the deep layer and the feedback parameter $\mathbf{K}$. For the same interconnection weight, Figure 9 shows the performance of the two-layer RC controller, exhibiting general recruitment of the network to the reference signal, but with worse performance than both the individual-layer recruitment and the NG-RC controller. In particular, we note that recruitment is not achieved as well for the middle layer, with the network moving both above and below the reference at different points.

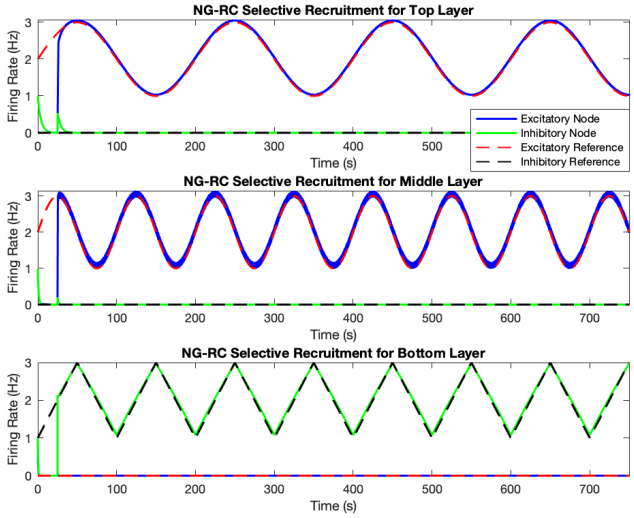

FIGURE 8: Selective inhibition and recruitment for all three layers in the interconnected network with $\gamma = 20$ using the two-layer NG-RC controller. With this level of interconnection, the controller provides performance similar to the single-layer recruitment, with RMSE errors for each layer being $0.0433, 0.0819$, and 0.0572 for the top, middle, and bottom layers, resp.

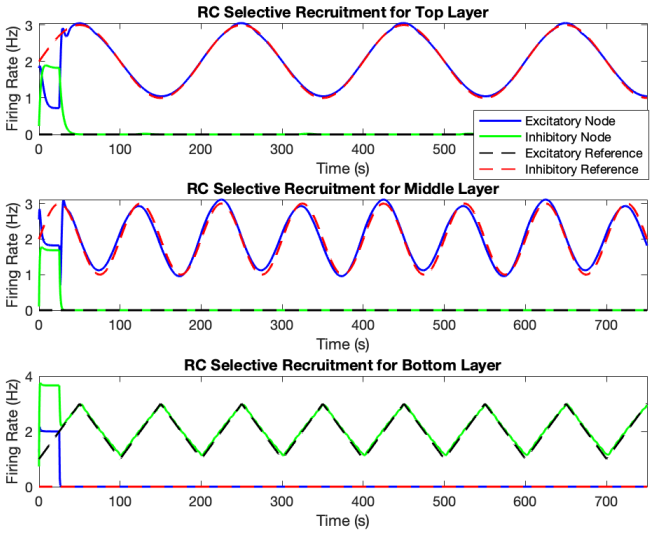

FIGURE 9: Selective inhibition and recruitment for all three layers in the interconnected network with $\gamma = 20$ using the two-layer RC controller. With this level of interconnection, the controller provides similar levels of performance to the individual networks for the top and bottom layers, while some additional error is introduced in the middle layer. The errors for the top, middle, and bottom layers are $0.0308, 0.1222$ and 0.0716, resp.

To directly compare the performance of the NG-RC and RC controllers, Figure 10 plots the RMSE of the signals with the references for both controllers as the interconnection weight γ increases. The NG-RC controller is significantly more robust to increasing interconnection weights compared to the RC controller, which quickly moves away from satisfactory recruitment of the network to the reference trajectory. One explanation for this improved robustness by the NG-RC controller is in the determination of the control input. For the RC controller, the control input is determined strictly from the output of the reservoir. Meanwhile, the NG-RC controller additionally reconsiders the error between the reference signal and the system state, and modifies the control accordingly using the parameter $\mathbf{K}$. Therefore, despite the increasing magnitude of interconnections adding additional error to the attempted tracking, the NG-RC controller is able to control these errors for longer due to its proportional control term.

D. Comparison between the RC and NG-RC Frameworks

From the results and plots above, we see that the RC and NG-RC frameworks both have scenarios where they more successfully selectively recruit the system to a reference. In particular, for smaller networks, the RC framework produces a similar, or slightly improved, quality recruitment. Meanwhile, the NG-RC controller is more robust to increasing

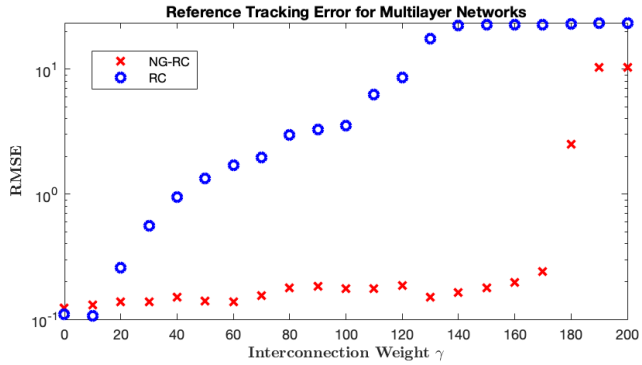

FIGURE 10: Comparison of recruitment error between the NG-RC and RC controllers as the magnitude of interconnections in the multilayer network increases. The NG-RC controller maintains a smaller error for significantly larger interconnections than the RC controller, before both become unsuccessful with large interconnections.

the magnitude of interconnections between layers. However, these comparisons are made only on the error between the reference signal and predicted signal. Depending on the situation, further metrics may be important in comparing the two controllers, such as training signal length, training time, and control signal and magnitude. In Table 2, we compare the training parameters and times, listed for the single-layer NG-RC controller, the single-layer RC controller, the multilayer NG-RC, and multilayer RC controller[2].

Network	Signal Length	Training Time	Control Time
NG-RC Single	500	0.0093	1.1543
NG-RC Multi	500	0.0655	1.6158
RC Single	80000	4.9564	6.0179
RC Multi	80000	26.18	18.17

TABLE 2: Comparison of training parameters and times.

Table 2 shows that the NG-RC framework allows for a much shorter training signal, and results in much faster times, both for the learning and controlling portion of the simulation. This aligns with the discussion of the frameworks in Section IV and illustrates that, if training time is important when using these frameworks, NG-RC performs significantly better. We note that the training signal lengths for both frameworks was determined after experimentation, with the lengths chosen to be the minimum lengths ensuring that performance in terms of recruitment error were acceptable.

Figure 11 shows the control signal being generated by the RC and NG-RC controllers for the bottom layer without any interconnections, and compares it with the one obtained in Theorem III.2. It is clear that while the two frameworks use similar controls to achieve selective recruitment, for selective inhibition they use significantly different controls in terms of magnitude (this is consistent with our observation in Remark III.3). In particular, both frameworks apply a control to the node being inhibited that is significantly higher

[2]Simulations were all computed in MATLAB r2023a on a 2019 MacBook Pro with a 2.8GHz Quad-Core Intel Core i7 processor.

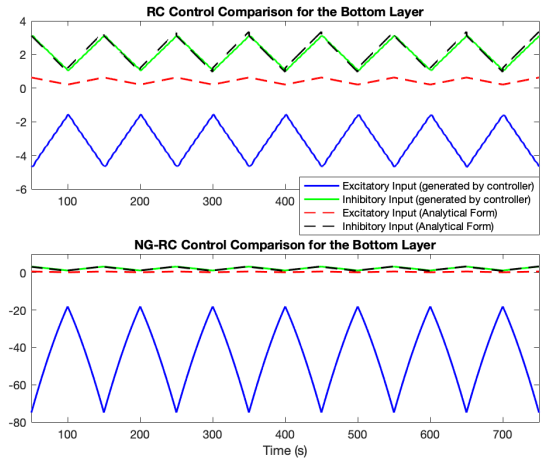

FIGURE 11: The control signals generated by the RC and NG-RC controllers when recruiting the bottom layer of the network individually. Both the RC and NG-RC controllers generate control inputs that match the analytically determined control in Theorem III.2 for the recruited node, in this case the inhibitory node. Meanwhile, the excitatory node is inhibited to zero using excessive inhibition, with significantly more inhibition being used by the NG-RC controller. Both result in high-performance tracking.

magnitude (more inhibitory) than the analytically determined control from Theorem III.2. Further, the NG-RC control is significantly higher magnitude than the RC control. Due to the observed difference in generated controls, depending on physical system constraints (such as those on control magnitude), the RC controller could be preferred, despite the longer training time.

VI. Application to Seizure Rejection

In our second application of reference tracking for LTN dynamics using reservoir computing, we consider the problem of epileptic seizure rejection. Epilepsy is a disease which impacts 50 million people worldwide and up to 30 percent of those have drug-resistant epilepsy [60], which instead can be treated with neuromodulation [61]. Data-driven methods have been been used in seizure detection [62] to predict seizure activity based on electroencephalogram (EEG) data. During epileptic seizures, brain activity becomes highly synchronized in a pathological manner, which results in the seizure symptoms. A key problem in epilepsy research is the detection of seizures in the early stages before symptoms begin to appear. This is desirable as then an interjection could be made in order to prevent the remainder of a seizure, and ideally, prevent most or all of the associated symptoms. Here we wish to look at how the reservoir computer controller network design, representing an external neurostimulation device, can be used to apply the control action to prevent seizure behavior upon detection through having the network track a desired 'safe' signal.

We first train a LTN model of a brain network with input and output restrictions to track EEG data that includes seizures. Then, we use the reservoir controller so that, when

a seizure is predicted, the network activity is brought to a desired pattern that does not exhibit seizure symptoms. In this work we utilize an arbitrary 'safe' signal that does not predict seizure activity using a synchrony-based approach. However, in clinical application the desired reference trajectory during intervention would be carefully determined by physicians to avoid any other possible pathological behaviors. Following the intervention we then allow the brain network to return to normal function without additional input from the controller.

A. Overview of the Approach

We consider EEG data taken from the "CHB-MIT Scalp EEG Database" [63], [64]. In the data, the seizure locations are noted and are also predicted in [65]. The seizure prediction method in [65] is based on the synchrony measure weighted phase lag index (WPLI) [66], which we also use here. Other seizure prediction approaches are discussed in [67].

WPLI is defined to measure synchrony between two signals, and is used in particular to compare electrophysiological signals. In [65], this is used comparing channels of an EEG for the purpose of seizure prediction. While their prediction technique involves further analysis on top of computing the WPLI, what is important to note is that the metric attains a high value (excessive synchrony) before a seizure. In our seizure rejection approach, we aim to have the network track a signal that reduces the WPLI between the brain regions as determined by the EEG data, in order to move away from seizure activity. In particular, whenever the WPLI is computed and determined to be above threshold $m_{\text{intervene}}$, we will intervene with a control term, computed using a RC, for a predetermined amount of time $t_{\text{intervene}}$, that drives the network to a safe activity pattern, r_{safe}. After this, the intervention will be stopped and only begin again if the next computation of the WPLI is above the threshold. This procedure is summarized in Algorithm 1.

B. Weighted Phase Lag Index and Data Processing

The WPLI measures synchrony between two signals based on the instantaneous phase of the two signals over a given time window. We compute the instantaneous phase through the Hilbert transform of a signal [68]. In particular, the instantaneous phase of a signal $\mathbf{x}(t)$ with Hilbert transform $\hat{\mathbf{x}}(t)$ is given by

$$\phi(t) = \arctan\left(\frac{\hat{\mathbf{x}}(t)}{\mathbf{x}(t)}\right).$$

For a time window Δ_t containing N points, the WPLI between signals $\mathbf{x}_1(t)$ and $\mathbf{x}_2(t)$ is then defined as:

$$\text{WPLI}_{\Delta_t} = \frac{\left|\frac{1}{N}\sum_{p=1}^{N}\sin(\phi_1(t)-\phi_2(t))\right|}{\frac{1}{N}\sum_{p=1}^{N}\left|\sin(\phi_1(t)-\phi_2(t))\right|}. \tag{15}$$

The WPLI takes values in the interval $[0,1]$, with low values corresponding to no coupling between the signals or a phase difference equal to $0 \pmod{\pi}$, while stronger phase locking gives higher values for the WPLI. If the signal is phase-locked with a non-zero difference, then the WPLI is equal to 1.

Algorithm 1 Seizure Rejection with RC Controller

Input: $T, m_{\text{intervene}}, t_0, t_{\text{intervene}}, r_{\text{safe}}$

```
Train RC on brain network model
Initialize control v(t_0) = 0
Initialize counter k = 1
while t ≥ t_0 do
    if t = kT + t_0 for k ∈ ℤ and not intervening then
        Compute and update WPLI
        k = k + 1
    end if
    if WPLI > m_intervene and t < (k − 1)T + t_intervene then
        Compute control v(t) with RC to drive system to r_safe
        Propagate brain network model with control v(t)
    else
        Do not intervene and set v(t) = 0
        Propagate brain network model with control v(t)
    end if
end while
```

Using the WPLI to compare to EEG signals on raw data fails to capture any trends due to the artifacts and noise characteristic of EEG. We therefore take two steps to process the raw signal and reduce noise. First, the data is filtered using a bandpass filter on a specified, typically small, band. These are typically experimentally determined and we use, following [65], the range of $8-13$ Hz. Second, the signal is differentiated with respect to time and the absolute value is computed. This is done to flatten the basic noise and emphasize the peaks of the signal [69].

C. Reproducing the EEG Data

As seizure activity is based on synchrony between brain regions, and the WPLI is computed between two signals, we consider a network with two outputs, each representing a channel of the EEG. In particular, we consider an 6-node brain network governed by linear-threshold dynamics, with two outputs. In addition, to match neurostimulation constraints due to implanting electrodes, we also limit the inputs to two of the nodes. Here, we consider the second seizure of patient 3 in the MIT EEG database, which is clinically found to occur at time $t = 730$ seconds within the EEG file and lasts 65 seconds. Following [65], we let the network outputs represent channels F4-C4 and T8-P8 of the EEG. We use a NG-RC setup as in Figure 5 to learn a control signal, $\mathbf{c}: \mathbb{R} \to \mathbb{R}^2$, such that each network mimics the EEG data of channels F4-C4 and T8-P8.

The synaptic weight matrix of the randomly generated network is given by

$$\mathbf{W} = \begin{bmatrix} 0.3711 & 0.0642 & 0.3530 & -0.2614 & -0.2079 & -0.0668 \\ 0.1369 & 0.03837 & 0.1442 & -0.0067 & -0.1887 & -0.3239 \\ 0.1001 & 0.1440 & 0.2700 & -0.1189 & -0.1174 & -0.0091 \\ 0.2363 & 0.0310 & 0.0015 & -0.2568 & -0.1436 & -0.1791 \\ 0.0542 & 0.0760 & 0.1080 & -0.557 & -0.0440 & -0.3020 \\ 0.2213 & 0.1887 & 0.1996 & -0.2418 & -0.3127 & -0.0345 \end{bmatrix}, \tag{16}$$

with inputs in nodes 1 and 3. The outputs are defined by

$$\begin{bmatrix} \mathbf{y}_{F4-C4} \\ \mathbf{y}_{T8-P8} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \mathbf{x}, \tag{17}$$

and actuation is limited to the first and third nodes. The replication of the EEG data is done using an NG-RC controller with a 500-point training signal sampled from a $\mathcal{N}(0, 0.1)$ distribution. Figure 12 illustrates the replication of the EEG data, along with the original data, shown for a 9-second time period preceding the seizure by tracking the original data with the NG-RC controller. For the purposes of computing the WPLI and seizure rejection, we consider the data from $t = 490$ seconds to $t = 855$ seconds, which corresponds to four minutes before the seizure and one minute after the seizure. On this timescale, it is more difficult to visually observe the success of the replication of the EEG data using the NG-RC. Figure 13 shows the EEG replication over the extended time period along with the WPLI over this time. The WPLI computed from the original data is also included to illustrate accuracy of the replication. For both the replicated and original data, the WPLI is computed with a time window of 6 seconds and overlap of one second as in [65]. It is evident that as the seizure approaches the WPLI is increasing, reaching the maximum value of 1 prior to the seizure (which begins at $t = 730$), before dropping to a low level during and after the pathological behavior.

D. Rejecting Seizure Behavior

Medical interventions will seek to prevent the seizure behavior displayed in Figure 13. In this section, we apply the seizure rejection method described in Algorithm 1 to accomplish this by interjecting in the model to keep the WPLI between the channels F4-C4 and T8-P8 below a threshold. To do so, we run the system with the base control $\mathbf{c}$ learned in the prior section using the NG-RC, so that the network exhibits the same behavior as the EEG data. Then, we interject the system with an additional control $\mathbf{v} : \mathbb{R} \to \mathbb{R}^2$ each time the computed WPLI reaches or exceeds a threshold of 0.8, so that the WPLI is reduced by utilizing the NG-RC controller to have the network track a pair of signals that do not exhibit excessive synchrony.

In our simulations we use a control (determined by the NG-RC controller) that modulates the network to a predetermined safe signal for a period of 60 seconds, a duration based on the average length of a generalized tonic-clonic seizure [70]. We define our safe signal based on two sinusoids with different frequencies to guarantee a lack of synchrony and add white noise to match the noisy nature of EEG measurements. Figure 14 illustrates the application of Algorithm 1 to the EEG with a first rejection at $t = 625$, when the WPLI first meets the threshold of 0.8, and a second intervention at $t = 700$. We see from here on that, with the intervention, the WPLI stops increasing towards the upper limit of 1 (which is the expected marker for leadup to seizure behavior [65]).

VII. Conclusions and Future Work

We have tackled the problem of control design for reference tracking in linear-threshold firing rate network models through the reservoir computing framework. We first formally designed explicit open- and closed-loop controllers that achieve reference tracking under suitable conditions on the synaptic connectivity. To overcome the difficulty of determining precisely the strength of interconnections in the brain, required by these controllers, and the fact that the identified conditions become increasingly difficult to check with network size, together with considerations of biological implausibility, we have proposed the use of reservoir computing to synthesize the control signals. We have shown how the RC and NG-RC frameworks can be used as controllers for the problem of selective recruitment and inhibition of LTN networks, allowing for an arbitrarily chosen reference trajectory. We have also used an NG-RC controller to replicate EEG data as well as reject epileptic seizure activity. Future work will apply the reservoir computing control framework to larger networks, with the limiting factor being the determination of optimal parameters for learning. Another direction for future work is the study of how the composition of the network and reservoir, and symmetries between them due to their structure as linear-threshold networks, could be exploited to achieve improved control performance. Finally, we will investigate how different reservoir or feature vector structures can improve performance and explore reference tracking with limited sensing.

REFERENCES

[1] M. Marconi, N. D. C. Blanco, C. Zimmer, and A. Guyon, "Eye movements in response to different cognitive activities measured by eyetracking: a prospective study on some of the neurolinguistics programming theories," *Journal of Eye Movement Research*, vol. 16, no. 2, 2023.

[2] A. P. Vaz, J. H. Wittig Jr., S. K. Inati, and K. A. Zaghloul, "Replay of cortical spiking sequences during human memory retrieval," *Science*, vol. 367, no. 6482, pp. 1131–1134, 2020.

[3] G. Ariani, J. A. Pruszynski, and J. Diedrichsen, "Motor planning brings human primary somatosensory cortex into action-specific preparatory states," *eLife*, vol. 11, p. e69517, 2022.

[4] H. R. Wilson and J. D. Cowan, "Excitatory and inhibitory interactions in localized populations of model neurons," *Biophysical Journal*, vol. 12, no. 1, pp. 1–24, 1972.

[5] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[6] A. N. Burkitt, "A review of the integrate-and-fire neuron model: I. homogeneous synaptic input," *Biological Cybernetics*, vol. 95, pp. 1–19, 2006.

[7] P. Dayan and L. F. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. Computational Neuroscience, Cambridge, MA: MIT Press, 2001.

[8] E. M. Izhikevich, *Dynamical Systems in Neuroscience*. Cambridge, MA: MIT Press, 2007.

[9] E. Nozari, M. A. Bertolero, J. Stiso, L. Caciagli, E. J. Cornblath, X. He, A. S. Mahadevan, G. J. Pappas, and D. S. Bassett, "Macroscopic resting-state brain dynamics are best described by linear models," *Nature Biomedical Engineering*, vol. 8, no. 1, pp. 68–84, 2024.

[10] M. Khosla, G. H. Ngo, K. Jamison, A. Kuceyeski, and M. R. Sabuncu, "Cortical response to naturalistic stimuli is largely predictable with deep neural networks," *Science Advances*, vol. 7, no. 22, p. eabe7547, 2021.

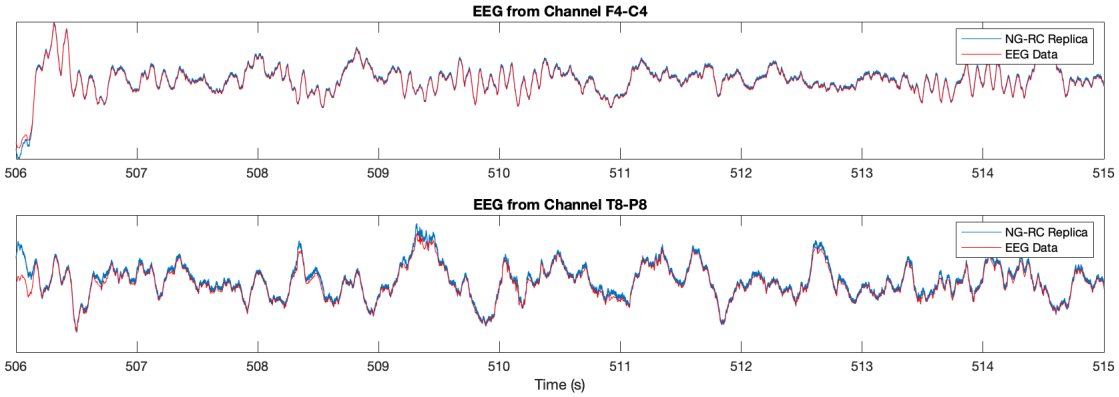

FIGURE 12: Illustration of the replication of the EEG data from the second seizure of patient 3 over the timeframe $t = (506, 515)$ seconds for channels F4-C4 and T8-P8 using a NG-RC with network matrix (16) and outputs (17). This precedes the beginning of the seizure at $t = 730$. The replication achieves a root-mean-square error (RMSE) of 0.0310 for the F4-C4 channel and 0.0316 for the T8P8 channel on this time interval.

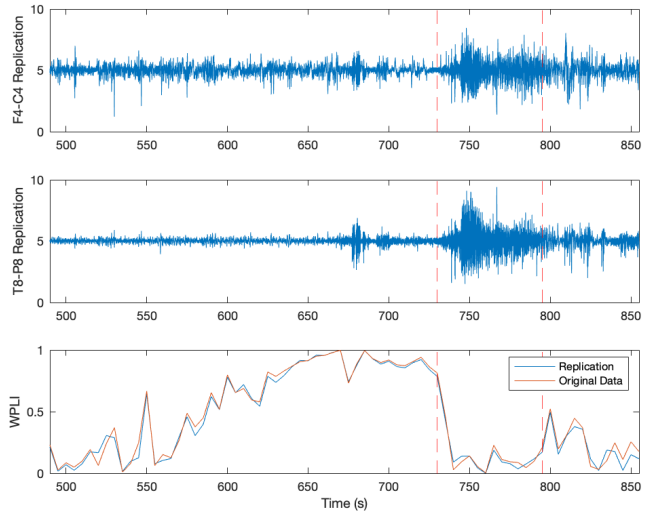

FIGURE 13: The replicated EEG from four minutes prior to and one minute after the second seizure of patient 3 ($t \in (490, 855)$). The first panel illustrates the activity of the F4-C4 electrode while the second panel is the T8-P8 electrode. Over this time frame the RMSE on the replication for the F4-C4 electrode is 0.0737, while for the T8-P8 electrode it is 0.0896. The third panel shows the WPLI for both the replicated EEG and the original data, which has a RMSE of 0.0474. We note that the WPLI increases and peaks prior to the seizure, the start and end of which are indicated by the red lines.

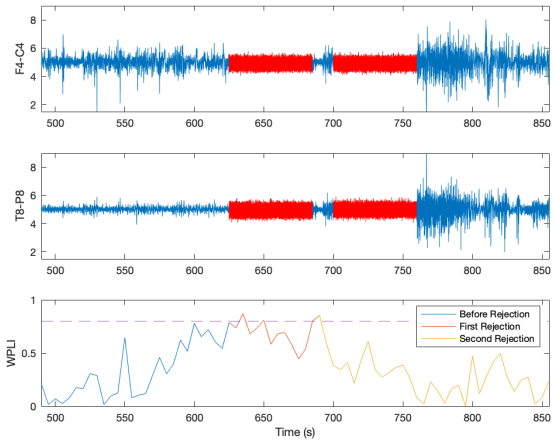

FIGURE 14: The EEG in channels F4-C4 and T8-P8 from $t = 490$ to $t = 855$, with two interventions, at $t = 625$ and at $t = 700$, when the WPLI reaches the threshold of 0.8. The controlled portions of the EEG are shown in red. We see that after interventions the WPLI moves below the threshold of 0.8, and does not exhibit the drastic drop in WPLI at the beginning of the seizure. During normal periods, a low-magnitude control (of average amplitude 100) is applied to track the base EEG activity, before an additional high-magnitude control (of average amplitude 4000) is applied during the seizure rejection periods to modulate to a safe signal. These controls are of high amplitude due to the high-frequency EEG activity requiring fast changes in value.

[11] F. Shao and Z. Shen, "How can artificial neural networks approximate the brain?," *Frontiers in Psychology*, vol. 13, p. 970214, 2023.

[12] N. Kriegeskorte, "Deep neural networks: a new framework for modeling biological vision and brain information processing," *Annual Review of Vision Science*, vol. 1, pp. 417–446, 2015.

[13] Y. Li, H. Yang, and S. Gu, "Upgrading voxel-wise encoding model via integrated integration over features and brain networks," *BioRxiv*, 2022.

[14] P. Blomquist, A. Devor, U. G. Indahl, I. Ulbert, G. T. Einevoll, and A. M. Dale, "Estimation of thalamocortical and intracortical network models from joint thalamic single-electrode and cortical laminar-electrode recordings in the rat barrel system," *PLoS Computational Biology*, vol. 5, no. 3, p. e1000328, 2009.

[15] T. Heiberg, B. Kriener, T. Tetzlaff, G. T. Einevoll, and H. E. Plesser, "Firing-rate models for neurons with a broad repertoire of spiking behaviors," *Journal of Computational Neuroscience*, vol. 45, pp. 103–132, 2018.

[16] F. Ratliff and H. K. Hartline, *Studies on Excitation and Inhibition in the Retina*. Rockefeller University Press, 1974.

[17] C. Curto, J. Geneson, and K. Morrison, "Stable fixed points of combinatorial threshold-linear networks," *Advances in Applied Mathematics*, vol. 154, p. 102652, 2024.

[18] E. Nozari and J. Cortés, "Hierarchical selective recruitment in linear-threshold brain networks. Part I: Intra-layer dynamics and selective inhibition," *IEEE Transactions on Automatic Control*, vol. 66, no. 3, pp. 949–964, 2021.

[19] E. Nozari and J. Cortés, "Hierarchical selective recruitment in linear-threshold brain networks. Part II: Inter-layer dynamics and top-down recruitment," *IEEE Transactions on Automatic Control*, vol. 66, no. 3, pp. 965–980, 2021.

[20] M. McCreesh and J. Cortés, "Selective inhibition and recruitment in linear-threshold thalamocortical networks," *IEEE Transactions on Control of Network Systems*, vol. 11, no. 1, pp. 375–388, 2024.

[21] A. Allibhoy, F. Celi, F. Pasqualetti, and J. Cortés, "Optimal network interventions to control the spreading of oscillations," *IEEE Open Journal of Control Systems*, vol. 1, pp. 141–151, 2022.

[22] F. Celi, A. Allibhoy, F. Pasqualetti, and J. Cortés, "Linear-threshold dynamics for the study of epileptic events," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1405–1410, 2021.

[23] K. P. Hadeler and D. Kuhn, "Stationary states of the Hartline-Ratliff model," *Biological Cybernetics*, vol. 56, no. 5-6, pp. 411–417, 1987.

[24] J. Feng and K. P. Hadeler, "Qualitative behaviour of some simple networks," *Journal of Physics A: Mathematical and General*, vol. 29, no. 16, pp. 5019–5033, 1996.

[25] Z. Yi and K. K. Tan, "Multistability of discrete-time recurrent neural networks with unsaturating piecewise linear activation functions," *IEEE Transactions on Neural Networks*, vol. 15, no. 2, pp. 329–336, 2004.

[26] M. McCreesh, T. Menara, and J. Cortés, "Sufficient conditions for oscillations in competitive linear-threshold brain networks," *IEEE Control Systems Letters*, vol. 7, pp. 2886–2891, 2023.

[27] E. Nozari, R. Planas, and J. Cortés, "Structural characterization of oscillations in brain networks with rate dynamics," *Automatica*, vol. 146, p. 110653, 2022.

[28] K. Morrison, A. Degeratu, V. Itskov, and C. Curto, "Diversity of emergent dynamics in competitive threshold-linear networks," *SIAM Journal on Applied Dynamical Systems*, vol. 23, no. 1, pp. 855–884, 2024.

[29] X. Wang and J. Cortés, "Data-driven control of linear-threshold network dynamics," in *American Control Conference*, (Atlanta, Georgia), pp. 114–119, June 2022.

[30] D. Liberzon, *Switching in Systems and Control*. Systems & Control: Foundations & Applications, Birkhäuser, 2003.

[31] J.-P. Laumond, *Robot motion planning and control*, vol. 229. Springer, 1998.

[32] J.-C. Latombe, *Robot motion planning*, vol. 124. Springer Science & Business Media, 2012.

[33] A. Tsiamis, A. Karapetyan, Y. Li, E. C. Balta, and J. Lygeros, "Predictive linear online tracking for unknown targets," *arXiv preprint arXiv:2402.10036*, 2024.

[34] Y. Xu, Z. Wu, W. Che, and D. Meng, "Reinforcement learning-based unknown reference tracking control of HMASs with nonidentical communication delays," *Science China Information Sciences*, vol. 66, no. 7, p. 170203, 2023.

[35] G. Baggio, D. S. Bassett, and F. Pasqualetti, "Data-driven control of complex networks," *Nature Communications*, vol. 12, no. 1, pp. 1–13, 2021.

[36] Y. Qin, T. Menara, S. Oymak, S. Ching, and F. Pasqualetti, "Representation learning for context-dependent decision-making," in *American Control Conference*, (Atlanta, GA), pp. 2130–2135, 2022.

[37] V. Narayanan, J. T. Ritt, J. Li, and S. Ching, "A learning framework for controlling spiking neural networks," in *American Control Conference*, pp. 211–216, IEEE, 2019.

[38] J. Z. Kim and D. S. Bassett, "A neural machine code and programming framework for the reservoir computer," *Nature Machine Intelligence*, vol. 5, no. 6, pp. 622–630, 2023.

[39] L. M. Smith, J. Z. Kim, Z. Lu, and D. S. Bassett, "Learning continuous chaotic attractors with a reservoir computer," *Chaos*, vol. 32, no. 1, 2022.

[40] M. Lukoševičius, H. Jaeger, and B. Schrauwen, "Reservoir computing trends," *KI-Künstliche Intelligenz*, vol. 26, pp. 365–371, 2012.

[41] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[42] F. Damicelli, C. C. Hilgetag, and A. Goulas, "Brain connectivity meets reservoir computing," *PLoS Computational Biology*, vol. 18, no. 11, p. e1010639, 2022.

[43] P. Enel, E. Procyk, R. Quilodran, and P. F. Dominey, "Reservoir computing properties of neural dynamics in prefrontal cortex," *PLoS Computational Biology*, vol. 12, no. 6, p. e1004967, 2016.

[44] C. Merkel, Q. Saleh, C. Donahue, and D. Kudithipudi, "Memristive reservoir computing architecture for epileptic seizure detection," *Procedia Computer Science*, vol. 41, pp. 249–254, 2014.

[45] D. Canaday, A. Pomerance, and D. J. Gauthier, "Model-free control of dynamical systems with deep reservoir computing," *Journal of Physics: Complexity*, vol. 2, no. 3, p. 035025, 2021.

[46] D. J. Gauthier, E. Bollt, A. Griffith, and W. A. S. Barbosa, "Next generation reservoir computing," *Nature Communications*, vol. 12, no. 1, p. 5564, 2021.

[47] R. Kent, W. A. S. Barbosa, and D. J. Gauthier, "Controlling chaotic maps using next-generation reservoir computing," *Chaos*, vol. 34, no. 2, 2024.

[48] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009.

[49] 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*, vol. 115, pp. 100–123, 2019.

[50] E. Nozari and J. Cortés, "Hierarchical selective recruitment in linear-threshold brain networks. Part I: Intra-layer dynamics and selective inhibition," *arXiv preprint arXiv:1809.01674v2*, 2019.

[51] H. Khalil, *Nonlinear Systems, 3rd ed.* Englewood Cliffs, NJ: Prentice Hall, 2002.

[52] S. Song, P. J. Sjöström, M. Reigl, S. Nelson, and D. B. Chklovskii, "Highly nonrandom features of synaptic connectivity in local cortical circuits," *PLOS Biology*, vol. 3, no. 3, p. e68, 2005.

[53] U. S. Bhalla, "How to record a million synaptic weights in a hippocampal slice," *PLoS Computational Biology*, vol. 4, no. 6, p. e1000098, 2008.

[54] I. B. Yildiz, H. Jaeger, and S. J. Kiebel, "Re-visiting the echo state property," *Neural Networks*, vol. 35, pp. 1–9, 2012.

[55] E. Bollt, "On explaining the surprising success of reservoir computing forecaster of chaos? The universal machine learning dynamical system with contrast to VAR and DMD," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, no. 1, 2021.

[56] S. Haeusler and W. Maass, "A statistical analysis of information-processing properties of lamina-specific cortical microcircuit models," *Cerebral Cortex*, vol. 17, no. 1, pp. 149–162, 2007.

[57] Y. Xue, L. Yang, and S. Haykin, "Decoupled echo state networks with lateral inhibition," *Neural Networks*, vol. 20, no. 3, pp. 365–376, 2007.

[58] J. T. Serences and S. Kastner, "A multi-level account of selective attention," *The Oxford Handbook of Attention*, p. 76, 2014.

[59] R. Desimone and J. Duncan, "Neural mechanisms of selective visual attention," *Annual Review of Neuroscience*, vol. 18, no. 1, pp. 193–222, 1995.

[60] J. W. Sander and S. D. Shorvon, "Epidemiology of the epilepsies," *Journal of Neurology, Neurosurgery, and Psychiatry*, vol. 61, no. 5, p. 433, 1996.

[61] M. Abouelleil, N. Deshpande, and R. Ali, "Emerging trends in neuromodulation for treatment of drug-resistant epilepsy," *Frontiers in Pain Research*, vol. 3, p. 839463, 2022.

[62] P. Buteneers, D. Verstraeten, P. van Mierlo, T. Wyckhuys, D. Stroobandt, R. Raedt, H. Hallez, and B. Schrauwen, "Automatic detection of epileptic seizures on the intra-cranial electroencephalogram of rats using reservoir computing," *Artificial Intelligence in Medicine*, vol. 53, no. 3, pp. 215–223, 2011.

[63] A. H. Shoeb, *Application of Machine Learning to Epileptic Seizure Onset Detection and Treatment*. PhD thesis, Massachusetts Institute of Technology, 2009.

[64] A. Goldberger, L. Amaral, L. Glass, J. Hausdorff, P. C. Ivanov, R. Mark, and H. E. Stanley, "Physionet: components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[65] P. Detti, G. Z. M. de Lara, R. Bruni, M. Pranzo, F. Sarnari, and G. Vatti, "A patient-specific approach for short-term epileptic seizures prediction through the analysis of EEG synchronization," *IEEE Transactions on Biomedical Engineering*, vol. 66, no. 6, pp. 1494–1504, 2018.

[66] M. Vinck, R. Oostenveld, M. van Wingerden, F. Battaglia, and C. M. A. Pennartz, "An improved index of phase-synchronization for electrophysiological data in the presence of volume-conduction, noise and sample-size bias," *Neuroimage*, vol. 55, no. 4, pp. 1548–1565, 2011.

[67] R. Cherian and E. G. Kanaga, "Theoretical and methodological analysis of EEG based seizure detection and prediction: An exhaustive review," *Journal of Neuroscience Methods*, vol. 369, p. 109483, 2022.

[68] A. Zygmund, *Trigonometric Series*, vol. 1. Cambridge University Press, 2002.
[69] K. K. Majumdar and P. Vardhan, "Automatic seizure detection in ECoG by differential operator and windowed variance," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 19, no. 4, pp. 356–365, 2011.
[70] P. Meritam Larsen, S. Wüstenhagen, D. Terney, E. Gardella, H. Aurlien, and S. Beniczky, "Duration of epileptic seizure types: A data-driven approach," *Epilepsia*, vol. 64, no. 2, pp. 469–478, 2023.

Michael McCreesh received his B.A.Sc and M.A.Sc degrees in Mathematics and Engineering from Queen's University, Kingston, Canada in 2017 and 2019, resp. He received his Ph.D. degree in Mechanical Engineering from the University of California San Diego in 2024. His current research interests include control theory and its application to theoretical neuroscience, in particular the application of dynamical systems to model brain networks.

Jorge Cortés (M'02, SM'06, F'14) received the Licenciatura degree in mathematics from Universidad de Zaragoza, Zaragoza, Spain, in 1997, and the Ph.D. degree in engineering mathematics from Universidad Carlos III de Madrid, Madrid, Spain, in 2001. He held postdoctoral positions with the University of Twente, Twente, The Netherlands, and the University of Illinois at Urbana-Champaign, Urbana, IL, USA. He was an Assistant Professor with the Department of Applied Mathematics and Statistics, University of California, Santa Cruz, CA, USA, from 2004 to 2007. He is currently a Professor in the Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA, USA. He is a Fellow of IEEE, SIAM, and IFAC. His current research interests include distributed control and optimization, network science, nonsmooth analysis, reasoning and decision making under uncertainty, network neuroscience, and multi-agent coordination in robotic, power, and transportation networks.