# Improving Spiking Dynamical Networks: Accurate Delays, Higher-order Synapses, and Time Cells

**Aaron R. Voelker**[1, 2]**, Chris Eliasmith**[1]

[1]Centre for Theoretical Neuroscience, University of Waterloo, Waterloo, ON N2L 3G1, Canada.

[2]David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada.

**Keywords:** Neural Engineering Framework, Nengo, dynamical systems, spiking neural networks, higher-order synapses, delay computation, temporal coding, time cells

## Abstract

Researchers building spiking neural networks face the challenge of improving the biological plausibility of their model networks while maintaining the ability to quantitatively characterize network behavior. In this work, we extend the theory behind the Neural Engineering Framework (NEF), a method of building spiking dynamical networks, to permit the use of a broad class of synapse models while maintaining prescribed dynamics up to a given order. This theory improves our understanding of how low-level synaptic properties alter the accuracy of high-level computations in spiking dynamical networks. For completeness, we provide characterizations for both continuous-time (i.e., analog) and discrete-time (i.e., digital) simulations. We demonstrate the utility of these extensions by mapping an optimal delay line onto various spiking dynamical networks using higher-order models of the synapse. We show that these networks nonlinearly encode rolling windows of input history, using a scale-invariant representation, with accuracy depending on the frequency content of the input signal. Finally, we reveal that these methods provide a novel explanation of time cell responses during a delay task, which have been observed throughout hippocampus, striatum, and cortex.

# 1 Introduction

One of the central challenges in computational neuroscience is understanding how dynamic stimuli can be processed by neural mechanisms to drive behavior. Recurrent connections, cellular responses, and synaptic responses are ubiquitous sources of dynamics throughout the mammalian brain that must work in concert to support dynamic information processing (Kandel et al., 2000). How these low-level mechanisms interact in order to encode information about the history of a stimulus, across time, is the subject of considerable study. One approach to better understanding these mechanisms is to construct models that capture central features of neural dynamics, while implementing higher-level information processing.

The Neural Engineering Framework (NEF; Eliasmith & Anderson, 1999, 2003) proposes a method to model such dynamical systems in networks of spiking neurons (see Abbott et al., 2016; Denève & Machens, 2016, for reviews of other methods). The NEF has been used to construct a wide variety of neural models, including a 2.3 million neuron functioning model of the human brain, capable of performing perceptual, motor, and cognitive tasks (Eliasmith et al., 2012). This model incorporates many kinds of observed neural dynamics, including oscillations, sustained activity, and point attractor dynamics. The flexibility of the NEF has lead to it being deployed on mixed-analog-digital neuromorphic chips (Choudhary et al., 2012; Corradi et al., 2014; Voelker, Benjamin, et al., 2017; Voelker & Eliasmith, 2017) and digital architectures (Bekolay et al., 2013; Wang et al., 2014; Mundy et al., 2015; Berzish et al., 2016). Consequently, the NEF provides a practical method for programming neuromorphics, thus helping the field realize its promise of a low-energy computing platform that emulates core principles of

the nervous system (Boahen, 2017).

However, the NEF typically assumes an exponential model of the postsynaptic current (PSC) evoked by an action potential, which has a biologically implausible, instantaneous rise-time. This model is also known as a first-order lowpass filter. In contrast, the synapse models used in mixed-analog-digital neuromorphic chips are nonideal, featuring higher-order dynamics due to parasitic capacitances (Voelker, Benjamin, et al., 2017). Similarly, the synapse models commonly used in biological models incorporate distinct rise- and fall-times due to separate time scales of transmitter binding and unbinding, as well as axonal transmission delays due to the finite-velocity propagation of action potentials (Roth & van Rossum, 2009). To widen the scope of the NEF, we characterize the network-level effects of these higher-order synapse models, and harness them to implement certain classes of dynamical systems with improved accuracy.

A particularly important dynamical system that has not been implemented using the NEF is the pure continuous-time delay line. This system must represent a rolling window of input history. We provide a novel derivation of an optimal low-dimensional linear approximation to a continuous-time delay, and prove that the resulting *delay network* nonlinearly encodes its input across the delay interval. This network uses a scale-invariant representation, with a level of accuracy that depends on the input frequency, chosen dimensionality (i.e., the order of the approximation), and particular synapse model. Low-dimensional representations (e.g., $\leq 27$) of low-frequency signals (e.g., $\leq 50\,\text{Hz}$) are pervasive in biological systems (Cunningham & Byron, 2014; Waernberg & Kumar, 2017; Pulvermüller et al., 1997; Singer, 1999). To our knowledge, this work is the first to demonstrate that such a temporal code may be accurately

implemented using a spiking dynamical network.

Reservoir Computing approaches, such as Liquid State Machines (Maass et al., 2002) and Echo State Networks (Jaeger, 2001), may be used to approximate a delay line. However, since these networks use randomly chosen feedback weights, it is likely that they do not *efficiently* produce the dynamics of a pure delay. Rather, these networks represent a random variety of nonlinear memory traces (Lukoševičius, 2012). Discrete approaches to short-term memory, such as those taken by White et al. (2004) and Ganguli et al. (2008), while optimal in an information-theoretic sense, rely fundamentally on single time-step delays between rate-based neurons. In contrast, the method that we propose here works independently of the simulation time-step, and is optimal assuming the population of spiking neurons—coupled with some model of the synapse—accurately represents a low-dimensional, low-frequency, vector space. Furthermore, our framework is extended to account for arbitrary linear synapses, which improves our understanding of the relationship between synapse models and network-level computation. A detailed comparison of our method to Reservoir Computing remains the subject of future work.

A distinguishing feature of this work is that we begin the problem with a mathematical description of the ideal dynamics for the delay line, and then proceed by mapping this description onto a spiking neural network (using the NEF and its extensions). This is in contrast to methods that use online learning or backpropagation through time with rate-based neurons (De Vries & Principe, 1992; Sussillo & Abbott, 2009) or spiking neurons (Nicola & Clopath, 2016; Huh & Sejnowski, 2017; Gilra & Gerstner, 2017; Alemi et al., 2017). That is, we do not require any sophisticated training regimes in-

volving online learning or backpropagation; our delay network is trained offline using convex optimization (i.e., regularized least-squares), which yields a more complete understanding of its employed representation and dynamics.

The remainder of this paper is structured as follows. In section 2, we introduce the NEF, placing an emphasis on the mapping of continuous-time linear systems onto the canonical lowpass model of the synapse. In section 3.1, we use the NEF to derive a spiking dynamical network that delays its input signal by some fixed length of time, and then analyze this network in section 3.2 to characterize how it nonlinearly encodes the stimulus across a rolling window. In section 4, we extend the NEF to characterize the effects of higher-order synapse models, and to account for both continuous-time (i.e., for analog hardware) and discrete-time (i.e., for digital hardware) simulations. In section 5, we exploit the methods from section 4 to demonstrate their utility in computing delays and to provide insights into possible neural mechanisms. Specifically, we harness a small delay in the synapse model to improve the accuracy of a larger network-level delay. And, we illustrate the relevance of our delay network to biological modeling through qualitative and quantitative comparisons to the responses of time cells (Eichenbaum, 2014), suggesting a new characterization of how temporal representations may arise in the brain.

# 2 Neural Engineering Framework

The Neural Engineering Framework (NEF; Eliasmith & Anderson, 1999, 2003) consists of three mathematical principles used to describe neural computation. The NEF is

most commonly applied to building dynamic (i.e., recurrent) spiking neural networks, but also applies to non-spiking and feed-forward networks. Its primary strength lies in providing a well-understood and efficient means of engineering spiking neural models, and programming neuromorphic hardware, to perform mathematically described computations (Eliasmith, 2013; Boahen, 2017). In this section, we provide an overview of these methods applied to training both feed-forward and recurrent connection weights, in order to implement linear dynamical systems – although these methods extend to nonlinear dynamical systems as well (Voelker, Benjamin, et al., 2017; Voelker & Eliasmith, 2017). We present this framework in a manner that is consistent with the Nengo 2.4.0 simulator (Bekolay et al., 2013), which implements the NEF among other approaches.

## 2.1 Principle 1 – Representation

Let $\mathbf{x}(t) \in \mathbb{R}^q$ denote a $q$-dimensional time-varying signal, that is to be represented by a population of $n$ spiking neurons. To describe this representation, we define a nonlinear *encoding* and a linear *decoding* that together determine how neural activity relates to the represented vector.

First, we choose encoders $E = [\mathbf{e}_1, \ldots, \mathbf{e}_n]^\mathsf{T} \in \mathbb{R}^{n \times q}$, gains $\alpha_i > 0$, and biases $\beta_i$, $i = 1 \ldots n$, as parameters for the encoding, which map $\mathbf{x}(t)$ to neural activities. These parameters are fit from neuroanatomical data (e.g., tuning curves, preferred directions, firing rates, sparsity, etc.; see Friedl et al. (2016) for a recent example), or randomly sampled from distributions constrained by the domain of $\mathbf{x}(t)$ and the dynamic range of

the neuron models. In either case, the encoding is defined by:

$$a_i^{\mathbf{x}}(t) = G_i[\alpha_i \langle \mathbf{e}_i, \mathbf{x}(t) \rangle + \beta_i], \quad i = 1 \dots n, \tag{1}$$

where $a_i^{\mathbf{x}}(t)$ is the neural activity generated by the $i^{\text{th}}$ neuron encoding the vector $\mathbf{x}(t)$ at time $t$, $\langle \cdot, \cdot \rangle$ denotes a dot-product, and $G_i[\cdot]$ is the nonlinear dynamical system for a single neuron (e.g., a leaky integrate-and-fire (LIF) neuron, a conductance-based neuron, etc.). Then $a_i^{\mathbf{x}}(t) = \sum_m \delta(t - t_{i,m})$, where $\delta(\cdot)$ is the Dirac delta, and $\{t_{i,m}\}$ is the sequence of spike-times generated by equation 1.

Having defined an encoding, we introduce a postsynaptic filter $h(t)$, which acts as the *synapse model* by capturing the dynamics of a receiving neuron's synapse. In particular, this filter models the postsynaptic current (PSC) triggered by action potentials arriving at the synaptic cleft. For now, we fix $h(t) = \frac{1}{\tau} e^{-\frac{t}{\tau}}$ to be an exponentially decaying PSC with time-constant $\tau$, which is equivalent (in the Laplace domain) to the canonical first-order lowpass filter (also known as a "leaky integrator"). This is the conventional choice of synapse in the NEF, since it strikes a convenient balance between mathematical simplicity and biological plausibility (Eliasmith & Anderson, 2003). In section 4, we return to this point by considering more general synapse models that are capable of capturing a much broader variety of PSCs.

We can now characterize the decoding of the neural response, which determines the information extracted from the neural activities encoding $\mathbf{x}(t)$. Let $D = [\mathbf{d}_1, \dots, \mathbf{d}_n]^{\mathsf{T}} \in \mathbb{R}^{n \times q}$ be the decoding matrix that decodes $\mathbf{x}(t)$ from the population's activities $(a_i^{\mathbf{x}}(t))$ at time $t$. This linear decoding is described by:

$$(\mathbf{x} * h)(t) \approx \sum_{i=1}^{n} (a_i^{\mathbf{x}} * h)(t) \mathbf{d}_i, \tag{2}$$

where $*$ is the convolution operator that is used to apply the synaptic filter. Equation 2 takes a linear combination of the filtered activities, in order to recover a filtered version of the encoded signal.[1] To complete the characterization of neural representation, we solve for the optimal linear decoders $D$. This optimization is identical for Principles 1 and 2, as discussed below.

## 2.2 Principle 2 – Transformation

The second principle of the NEF addresses the issue of computing transformations of the represented signal. The encoding remains defined by equation 1. However, we now decode some desired function of $\mathbf{x}(t)$, $\mathbf{f} : \mathbb{R}^q \to \mathbb{R}^q$,[2] by applying an alternate matrix of decoders $D^{\mathbf{f}} = \left[\mathbf{d}_1^{\mathbf{f}}, \ldots, \mathbf{d}_n^{\mathbf{f}}\right]^{\mathsf{T}} \in \mathbb{R}^{n \times q}$ to the same activities:

$$(\mathbf{f}(\mathbf{x}) * h)(t) \approx \sum_{i=1}^{n} (a_i^{\mathbf{x}} * h)(t) \mathbf{d}_i^{\mathbf{f}}. \tag{3}$$

For both Principles 1 and 2, we optimize for $D^{\mathbf{f}}$ over the domain of the signal, $S = \{\mathbf{x}(t) : t \geq 0\}$, which is typically the unit $q$-ball $\{\mathbf{v} \in \mathbb{R}^q : \|\mathbf{v}\|_2 \leq 1\}$ or the unit $q$-cube $[-1, 1]^q$. To determine these decoders, we first let $r_i(\mathbf{v})$ be the limiting average firing rate of the $i^{\text{th}}$ neuron under the constant input $\mathbf{v} \in S$:

$$r_i(\mathbf{v}) = \lim_{t \to \infty} \frac{1}{t} \int_0^t a_i^{\mathbf{v}}(t')\, dt'. \tag{4}$$

For non-adaptive neuron models, equation 4 reduces to encoding $\mathbf{v}$ using a rate model. For adaptive neuron models, other definitions for $r_i(\mathbf{v})$ may be considered, but we limit

[1] It is more accurate to apply the filter to both sides, since in general the (time-invariant) decoders alone cannot compensate for the filter applied to the activities (this instead becomes the role of Principle 3).

[2] We may also consider transformations where the range has a different dimension, but the described framework will suffice for our purposes.

our discussion here to the (non-adaptive) spiking LIF model. To account for the variance introduced by neural spiking, and other sources of uncertainty, we introduce a white noise term $\eta \sim \mathcal{N}(0, \sigma^2)$. The optimality criteria for $D^{\mathbf{f}}$ is therefore:

$$D^{\mathbf{f}} = \underset{D \in \mathbb{R}^{n \times q}}{\arg\min} \int_S \left\| \mathbf{f}(\mathbf{v}) - \sum_{i=1}^{n} (r_i(\mathbf{v}) + \eta)\mathbf{d}_i \right\|^2 d^q\mathbf{v}. \tag{5}$$

Note that this optimization only depends on $r_i(\mathbf{v})$ for $\mathbf{v} \in S$, as opposed to depending on the signal $\mathbf{x}(t)$. In other words, the optimization is determined strictly by the distribution of the signal, and not according to its particular dynamics. Furthermore, this is a convex optimization problem, which may be solved by uniformly sampling $S$ (Voelker, Gosmann, & Stewart, 2017) and applying a standard regularized least-squares solver to the sampled data (Bekolay et al., 2013). Monte Carlo sampling introduces $\mathcal{O}\left(\frac{1}{\sqrt{m}}\right)$ error into the integral from equation 5, where $m$ is the number of samples, but this can be improved to $\widetilde{\mathcal{O}}\left(\frac{1}{m}\right)$—effectively squaring $m$—by the use of quasi-Monte Carlo methods (Fang & Wang, 1994; Knight et al., 2016). Nengo also supports alternative decoder solvers that optimize variations of equation 5 (e.g., Friedl et al., 2016; Kauderer-Abrams et al., 2017), but we do not use them here.

The accuracy of this approach relies on $r_i(\mathbf{v})$ being a suitable proxy for $a_i^{\mathbf{x}}(t)$ whenever $\mathbf{x}(t) = \mathbf{v}$. This zeroth-order approximation clearly holds in the steady-state for constant $\mathbf{x}(t)$, and turns out to be ideal in practice for low-frequency $\mathbf{x}(t)$ (Eliasmith & Anderson, 2003, appendix F.1), and likewise for $h(t)$ that filter out high frequencies (i.e., when the synaptic time-constant $\tau$ is large).

Equations 1 and 3 may then be used to derive a connection weight matrix between layers to implicitly compute the desired function $\mathbf{f}(\mathbf{x})$ within the *latent* vector space $\mathbb{R}^q$. Specifically, the weight matrix $W = [\omega_{ij}] \in \mathbb{R}^{n \times n}$, which maps activities from the

$j^{\text{th}}$ presynaptic neuron to the $i^{\text{th}}$ postsynaptic neuron (disregarding the gain, bias, and synapse), is given by:

$$\omega_{ij} = \left\langle \mathbf{e}_i, \mathbf{d}_j^{\mathbf{f}} \right\rangle. \tag{6}$$

Consequently, the matrices $E$ and $D^{\mathbf{f}}$ are a low-rank factorization of $W$. In other words, the process of decoding (equation 3) and then encoding (equation 1) is equivalent to taking the dot-product of the neural activities with the full-rank weight matrix $W$.

This factorization has important consequences for the computational efficiency of neural simulations. The crucial difference between the factorized and non-factorized forms is that it takes $\mathcal{O}(qn)$ operations per simulation time-step to implement this dot-product in the factored form of equation 6, as opposed to $\mathcal{O}(n^2)$ operations for a full-rank weight matrix. Since $q$ is typically held constant, this yields a factor $\mathcal{O}(n)$ improvement in simulation time. Similarly, this factorizations yields an $\mathcal{O}(n)$ reduction in memory, which significantly improves the scaling of neuromorphics (Mundy et al., 2015). In essence, this factorization provides a way to describe the network's latent state-vector $\mathbf{x}(t)$. This, in turn, permits us to perform useful computations by transforming the state-vector with $\mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$ error in the presence of spike noise (Eliasmith & Anderson, 2003, p. 47).

## 2.3 Principle 3 – Dynamics

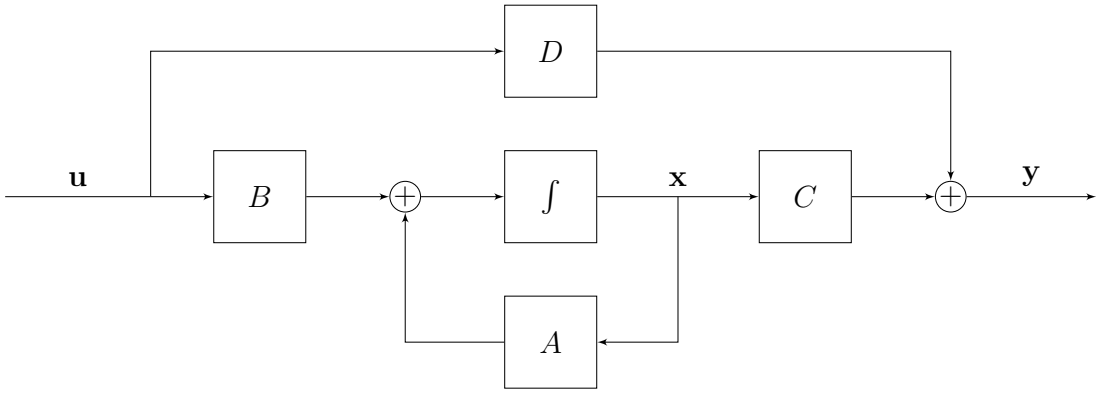


Figure 1: Block diagram for a LTI system. The integrator is driven by the signal $\dot{\mathbf{x}}(t)$.

Principle 3 is a method of harnessing the dynamics of the synapse model for network-level information processing. We begin by focusing our discussion of NEF dynamics on the neural implementation of continuous, linear time-invariant (LTI) systems:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + B\mathbf{u}(t), \\ \mathbf{y}(t) &= C\mathbf{x}(t) + D\mathbf{u}(t), \end{aligned} \tag{7}$$

where the time-varying signal $\mathbf{x}(t)$ represents the system state, $\dot{\mathbf{x}}(t)$ its time-derivative, $\mathbf{y}(t)$ the output, $\mathbf{u}(t)$ the input, and the time-invariant matrices $(A, B, C, D)$ fully describe the system (Brogan, 1991). This form of a LTI system is commonly referred to as the *state-space model*, but there are many other equivalent forms, which we will refer to later.

For LTI systems, the *dynamical primitive*—that is, the source of the dynamics—is the integrator (see Figure 1). However, the dynamical primitive at our disposal is the *leaky* integrator, given by the canonical first-order lowpass filter modeling the synapse:

$$h(t) = \frac{1}{\tau} e^{-\frac{t}{\tau}} = \mathcal{L}^{-1}\left\{\frac{1}{\tau s + 1}\right\}, \tag{8}$$

where $\mathcal{L}^{-1}\{\cdot\}$ denotes the inverse Laplace transform.[3] To be more precise, our approach is to represent the state-vector $\mathbf{x}(t)$ in a population of spiking neurons (Principle 1; equation 1), such that this vector is obtained by filtering some linearly decoded spike-trains with a leaky integrator (Principle 2; equation 3). Thus, the goal of Principle 3 is to determine the transformations required to implement equation 7, given that $\mathbf{x}(t)$ is obtained by some convolution with a leaky integrator, rather than the perfect integrator depicted in Figure 1.

Principle 3 states that in order to implement equation 7 in a population of neurons that represents $\mathbf{x}(t)$, we must compensate for the effect of "replacing" the integrator with a leaky integrator (compare Figures 1 and 2), that is, by driving the synapse with $\tau\dot{\mathbf{x}}(t)+\mathbf{x}(t)$ instead of only $\dot{\mathbf{x}}(t)$. This compensation is achieved as follows: implement the recurrent transformation $(\tau A + I)\mathbf{x}(t)$, and the input transformation $\tau B\mathbf{u}(t)$, but use the same output transformation $C\mathbf{x}(t)$, and the same passthrough transformation $D\mathbf{u}(t)$ (Eliasmith & Anderson, 2003, pp. 221–225). Specifically, this may be implemented in a spiking dynamical network by representing $\mathbf{x}(t)$ via Principle 1 and then using Principle 2 to decode the needed transformations. The resulting model is summarized in Figure 2.

[3] For comparison, the Laplace transform of the integrator is $\mathcal{L}\{1\} = \frac{1}{s}$.

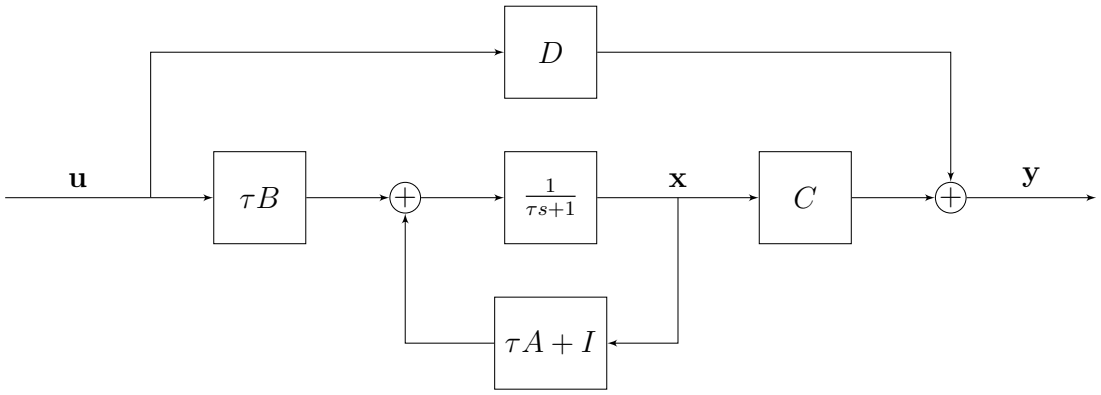


Figure 2: Block diagram for a LTI system, equivalent to Figure 1, with the integrator replaced by a first-order lowpass filter. The lowpass is driven by the signal $\tau\dot{\mathbf{x}}(t) + \mathbf{x}(t)$ to ensure that it implements the same system as in Figure 1.

The correctness of this "mapping" procedure relies on three assumptions: (1) the synapse model is equation 8, (2) the network is simulated in continuous time (or the discrete time-step is sufficiently small), and (3) the necessary representations and transformations are sufficiently accurate, such that the approximation error $\mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$ from equation 3 is negligible. In other words, assuming $n$ is sufficiently large, the architecture of Figure 2 is equivalent to Figure 1, but using the leaky integrator instead of an integrator as the dynamical primitive (Eliasmith & Anderson, 2003). Consequently, both systems compute the exact same signals $\mathbf{x}(t)$ and $\mathbf{y}(t)$. In section 4.1, we provide a novel proof of this equivalence. In sections 4.2–4.4, we extend Principle 3 to remove the first and second assumptions.

Principle 3 is useful for accurately implementing a wide class of dynamical systems (e.g., integrators, oscillators, attractor networks, etc.) to solve specific problems that frequently arise in neural modeling (e.g., Eliasmith & Anderson, 2000; Singh & Eliasmith, 2004; Eliasmith, 2005; Singh & Eliasmith, 2006). Furthermore, the class of state-space

models is isomorphic to the class of all finite-dimensional causal linear filters, or equivalently all rational (finite-order) proper transfer functions, which is a large and useful class of dynamical systems employed widely in control applications (Brogan, 1991). Given the ability of Principle 2 to compute nonlinear functions (i.e., equation 3), Principle 3 also naturally generalizes to nonlinear dynamical systems, but this is outside the scope of this work; we refer the reader to Voelker, Benjamin, et al. (2017) and Voelker & Eliasmith (2017) for recent nonlinear extensions.

# 3 Spiking Delay Networks

A fundamental dynamical system that has not yet been discussed within the NEF literature is the continuous-time delay line of $\theta$ seconds,[4] expressed as:

$$y(t) = (u * \delta_{-\theta})(t) = u(t - \theta), \quad \theta > 0, \tag{9}$$

where $\delta_{-\theta}$ denotes a Dirac delta function shifted backwards in time by $\theta$. This system takes a time-varying scalar signal, $u(t)$, and outputs a purely delayed version, $u(t - \theta)$. The task of computing this function both accurately and efficiently in a biologically plausible, spiking, dynamical network, is a significant theoretical challenge, that, to our knowledge, has previously remained unsolved.

The continuous-time delay is worthy of detailed consideration for several reasons. First, it is non-trivial to implement using continuous-time spiking dynamical primitives. Specifically, equation 9 requires that we maintain a *rolling window* of length $\theta$ (i.e., the history of $u(t)$, going $\theta$ seconds back in time). Thus computing a delay of $\theta$ seconds is

[4] Voelker (2015) proposed a NEF model, but did not include detailed analysis or extensions.

just as hard as computing every delay of length $\theta'$, for all $0 \leq \theta' \leq \theta$. Since any finite interval of $\mathbb{R}$ contains an uncountably-infinite number of points, an exact solution for arbitrary $u(t)$ requires that we maintain an uncountably-infinite amount of information in memory. Second, the delay provides us with a window of input history from which to compute arbitrary nonlinear functions across time. For instance, the spectrogram of a signal may be computed by a nonlinear combination of delays, as may any finite impulse response (FIR) filter. Third, delays introduce a rich set of interesting dynamics into large-scale neural models, including: oscillatory bumps, traveling waves, lurching waves, standing waves, aperiodic regimes, and regimes of multistability (Roxin et al., 2005). Fourth, a delay line can be coupled with a single nonlinearity to construct a network displaying many of the same benefits as Reservoir Computing (Appeltant et al., 2011). Finally, examining the specific case of the continuous-time delay introduces several methods and concepts we employ in generally extending the NEF (section 4).

## 3.1 Implementation

As it is impossible in practice (i.e., given finite-order continuous-time resources) to implement an arbitrary delay, we will instead approximate $u(t)$ as a low-frequency signal, or, equivalently, approximate equation 9 as a low-dimensional system expanded about the zeroth frequency in the *Laplace domain*. We begin by transforming equation 9 into the Laplace domain, $\mathcal{L}\{y(t)\} = \mathcal{L}\{u(t)\}\mathcal{L}\{\delta_{-\theta}(t)\}$, and then using the fact that $\mathcal{L}\{\delta_{-\theta}(t)\} = e^{-\theta s}$ to obtain:

$$F(s) := \frac{\mathcal{L}\{y(t)\}}{\mathcal{L}\{u(t)\}} = e^{-\theta s}, \tag{10}$$

where $F(s)$ is known as the *transfer function* of the system, defined as the ratio of the Laplace transform of the output to the Laplace transform of the input. Equation 10 should be understood as an equivalent way of expressing equation 9 in the Laplace domain, where the variable $s$ denotes a complex frequency. Thus far, we have only described the transfer function that we would like the network to implement.

The state-space model discussed in section 2.3 (equation 7) is related to its transfer function by the following:

$$F(s) = C(sI - A)^{-1}B + D. \tag{11}$$

Conversely, a transfer function can be converted into a state-space model satisfying equation 11 *if and only if* it can be written as a proper ratio of finite polynomials in $s$ (Brogan, 1991). The ratio is proper when the degree of the numerator does not exceed that of the denominator. In such a case, the output will not depend on future input, and so the system is *causal*. The degree of the denominator corresponds to the dimensionality of the state-vector, and therefore must be finite. These two conditions align with physically realistic constraints where time may only progress forward, and neural resources are limited.

However, the pure delay (equation 10) has infinite order when expressed as a ratio of polynomials in $s$, and so the system is irrational, or infinitely-dimensional. Consequently, no finite state-space model will exist for $F(s)$, which formalizes our previous intuition that an exact solution is impossible for finite, continuous-time systems. To overcome this, we must approximate the irrational transfer function $e^{-\theta s}$ as a proper ratio of finite-order polynomials. We do so using its *Padé approximants*—the coefficients of a Taylor series extended to the ratio of two polynomials—expanded about

$s = 0$ (Padé, 1892; Vajta, 2000):

$$[p/q]e^{-\theta s} = \frac{\mathcal{B}_p(-\theta s)}{\mathcal{B}_q(\theta s)},$$
$$\mathcal{B}_m(s) := \sum_{i=0}^{m} \binom{m}{i} \frac{(p+q-i)!}{(p+q)!} s^i. \tag{12}$$

This provides the transfer function of order $p$ in the numerator and order $q$ in the denominator that optimally approximates equation 10 for low-frequency inputs (i.e., up to order $p + q$).

After choosing $0 \leq p \leq q$, we may numerically find a state-space model $(A, B, C, D)$ that satisfies equation 11 using standard methods,[5] and then map this system onto the synapse using Principle 3. However, naïvely applying this conversion leads to numerical issues in the representation (i.e., dimensions that grow exponentially in magnitude), due in part to the factorials in equation 12.

To overcome this problem, we derive an equivalent yet normalized state-space model that we have not encountered elsewhere. We do so for the case of $p = q - 1$, since this provides the best approximation to the step-response (derived in appendix A.1):

$$A = \begin{pmatrix} -v_0 & -v_0 & \cdots & -v_0 \\ v_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ 0 & 0 & v_{q-1} & 0 \end{pmatrix}, \qquad \begin{aligned} B &= \begin{pmatrix} v_0 & 0 & \cdots & 0 \end{pmatrix}^{\mathsf{T}}, \\ C &= \begin{pmatrix} w_0 & w_1 & \cdots & w_{q-1} \end{pmatrix}, \\ D &= 0, \end{aligned} \tag{13}$$

where $v_i := \frac{(q+i)(q-i)}{i+1}\theta^{-1}$ and $w_i := (-1)^{q-1-i}\binom{i+1}{q}$, for $i = 0 \ldots q-1$. This model is now equivalent to equation 12, but without any factorials, and in the form of equation 7.[6] The choice of $q$ corresponds to the dimensionality of the latent state-vector $\mathbf{x}(t)$ that is to be represented by Principle 1 and transformed by Principle 2. Principle 3

[5] For instance, the function `tf2ss` in MATLAB or SciPy.

[6] In appendix A.3, we provide some additional manipulations of the state-space model.

may then be used to map equation 13 onto a spiking dynamical network to accurately implement an optimal low-frequency approximation of the delay.

To demonstrate, we implement a $1\,\mathrm{s}$ delay of $1\,\mathrm{Hz}$ band-limited white noise using $1{,}000$ recurrently connected spiking LIF neurons representing a $6$-dimensional vector space (see Figure 3). The connections between neurons are determined by applying Principle 3 (section 2.3) to the state-space model derived above (equation 13, $q = 6$) via the Padé approximants of the delay. The normalized root-mean-squared error (NRMSE) of the output signal is $0.048$ (normalized so that $1.0$ would correspond to random chance). This is achieved without appealing to the simulation time-step ($dt = 1\,\mathrm{ms}$); in fact, as shown in section 5.2, the network accuracy improves as $dt$ approaches zero due to the continuous-time assumption mentioned in section 2.3 (and resolved in section 4.2).

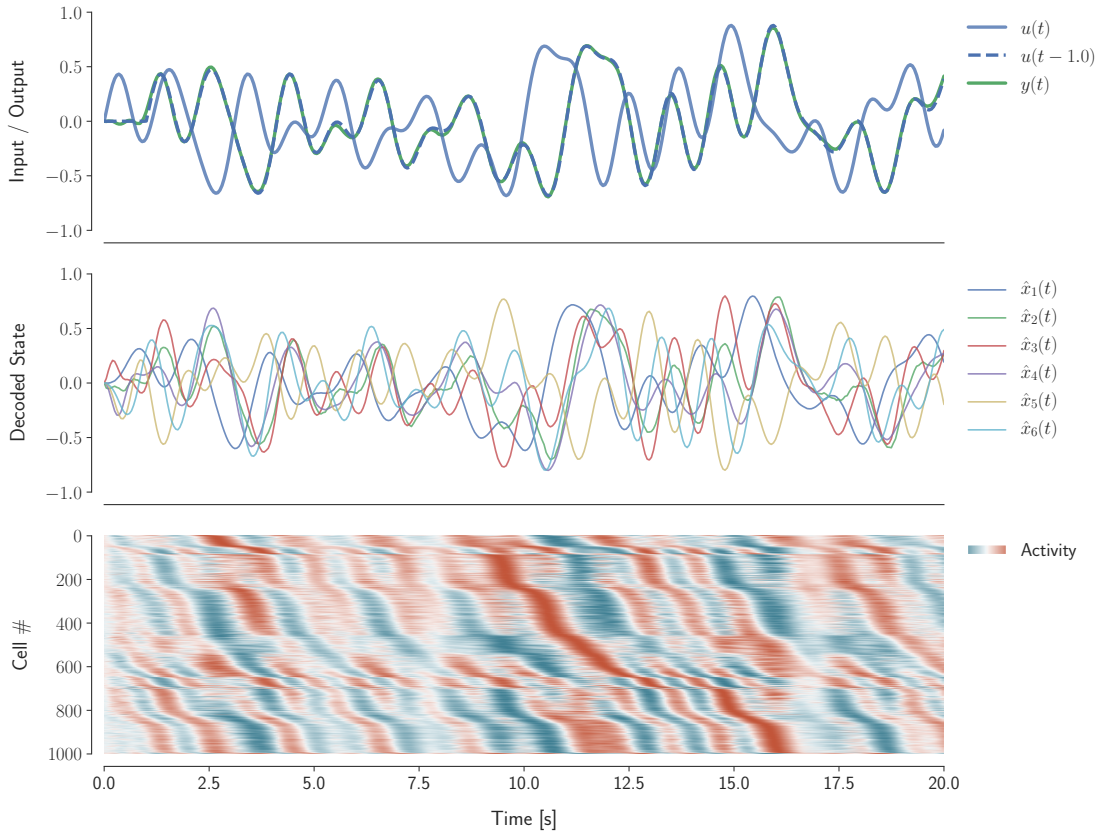


Figure 3: Delay of 1 s implemented by applying standard Principle 3 to equation 13 using $q = 6$, $dt = 1\,\text{ms}$, 1,000 spiking LIF neurons, and a lowpass synapse with $\tau = 0.1\,\text{s}$. The input signal is white noise with a cutoff frequency of 1 Hz. The plotted spikes are filtered with the same $\tau = 0.1\,\text{s}$, and encoded with respect to 1,000 encoders sampled uniformly from the surface of the hypersphere (sorted by time to peak activation).

## 3.2 Temporal Coding

The $q$-dimensional state-vector of the delay network represents a rolling window of length $\theta$. That is, a single delay network with some fixed $\theta > 0$ may be used to accurately decode any delay of length $\theta'$ ($0 \leq \theta' \leq \theta$). Different decodings require different

linear output transformations ($C$) for each $\theta'$, with the following coefficients (derived in appendix A.2):

$$w_{q-1-i} = \binom{q}{i}^{-1} \sum_{j=0}^{i} \binom{q}{j} \binom{2q-1-j}{i-j} \left(\frac{-\theta'}{\theta}\right)^{i-j}, \quad i = 0 \dots q-1. \tag{14}$$

The underlying dynamical state remains the same.

In Figure 4, we take different linear transformations of the same state-vector, by evaluating equation 14 at various delays between $0$ and $\theta$, to decode the rolling window of input from the state of the system.[7] This demonstrates that the delay network compresses the input's history (lasting $\theta$ seconds) into a low-dimensional state.

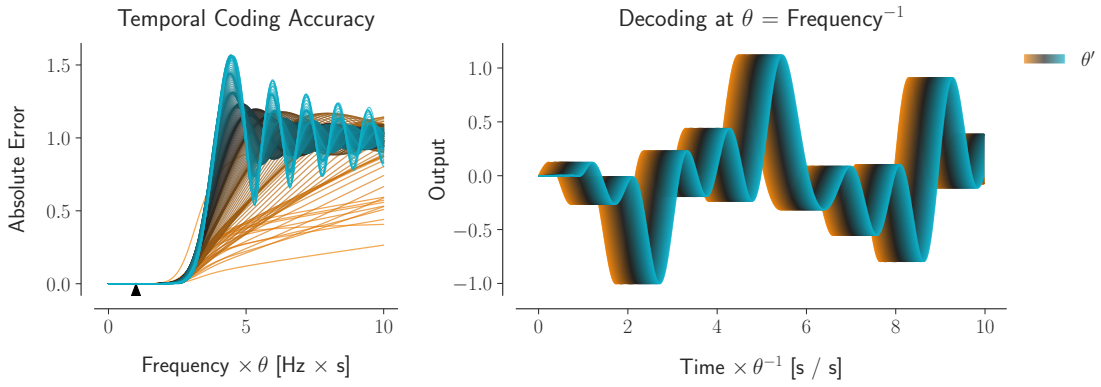


Figure 4: Decoding a rolling window of length $\theta$. Each line corresponds to a different delay, ranging from $0$ to $\theta$, decoded from a single delay network ($q = 12$). (Left) Error of each delay, as the input frequency is increased relative to $\theta$. Shorter delays are decoded more accurately than longer delays at higher frequencies. A triangle marks $\theta = \text{Frequency}^{-1}$. (Right) Example simulation decoding a rolling window of white noise with a cutoff frequency of $\theta^{-1}$ Hz.

In Figure 5, we sweep equation 14 across $\frac{\theta'}{\theta}$ to visualize the temporal "basis functi-

[7] The optimization problem from equation 5 need only be solved once to decode $\mathbf{x}(t)$ from the neural activity. The same decoders may then be transformed by each $C$ without loss in optimality (by linearity).

ons" of the delay network. This provides a way to understand the relationship between the chosen state-space representation (i.e., the $q$-dimensional $\mathbf{x}(t)$) and the underlying window representation (i.e., the infinite-dimensional $u(t)$). In particular, each basis function corresponds to the continuous window of history represented by a single dimension of the delay network. The instantaneous value of each dimension acts as a coefficient on its basis function, to contribute to the representation of the window at that point in time. Overall, the entire state-vector determines a linear combination of these $q$ basis functions to represent the window. This is analogous to the static function representation explored previously within the context of Principles 1 and 2 (Eliasmith & Anderson, 2003, pp. 63–72).

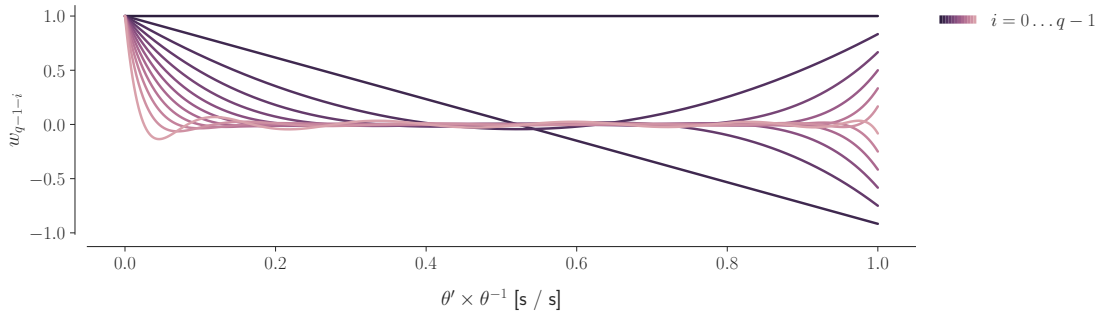


Figure 5: Temporal basis functions of the delay network ($q = 12$). Each line corresponds to the basis function of a single dimension ($i$) ranging from 0 (darkest) to $q - 1$ (lightest). The $i^{\text{th}}$ basis function is a polynomial over $\frac{\theta'}{\theta}$ with degree $i$ (see equation 14). The state-vector of the delay network takes a linear combination of these $q$ basis functions in order to represent a rolling window of length $\theta$.

The encoder of each neuron can also be understood directly in these terms as taking a linear combination of the basis functions (via equation 1). Each neuron nonlinearly encodes a projection of the rolling window onto some "preferred window" determined

by its own encoder. Since the state-vector is encoded by heterogeneous neural nonlinearities, the population's spiking activity supports the decoding of nonlinear functions across the entire window (i.e., functions that we can compute using Principles 1 and 2). Therefore, we may conceptualize the delay network as a *temporal coding* of the input stimulus, which constructs a low-dimensional state—representing an entire window of history—to encode the temporal structure of the stimulus into a nonlinear high-dimensional space of neural activities.

To more thoroughly characterize the delay dynamics, we analyze the behavior of the delay network as the dimensionality is increased (see Figure 6). Specifically, we perform a standard principal component analysis (PCA) on the state-vector for the impulse response, and vary the order from $q = 3$ to $q = 27$. This allows us to visualize a subset of the state-vector trajectories, via projection onto their first three principal components (see Figure 6-Top). The length of this trajectory over time distinguishes different values of $q$ (see Figure 6-Bottom). This length-curve is approximately logarithmic when $q = 6$, convex when $q \leq 12$, and sigmoidal when $q > 12$. To generate this figure we use a delay of $\theta = 10\,\text{s}$, but in fact this analysis is scale-invariant with time. This means that other delays will simply stretch or compress the impulse response linearly in time (not shown).

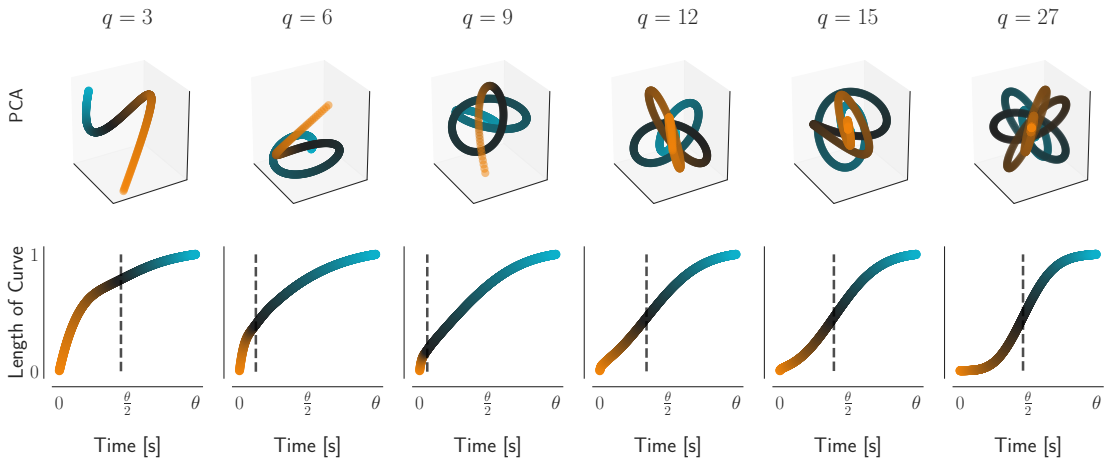


Figure 6: Impulse response of the delay network with various orders ($q$) of Padé approximants. (Top) The state-vector $\mathbf{x}(t)$ projected onto its first three principal components. (Bottom) The length of the curve $\mathbf{x}$ up to time $t$, computed using the integral $\int_0^t \|\dot{\mathbf{x}}(t')\|\, dt'$ (normalized to 1 at $t = \theta$). This corresponds to the distance travelled by the state-vector over time. The dashed line marks the last inflection point, indicating when $\mathbf{x}(t)$ begins to slow down.

We remark that the delay network is scale-invariant with the delay length over input frequency, that is, the accuracy for a chosen order is a function of $s \times \theta$ (see units in Figure 4 for instance). More specifically, for a fixed approximation error, the delay length scales as $\mathcal{O}\left(\frac{q}{f}\right)$, where $f$ is the input frequency. Then, the accuracy of the mapped delay is a function of the relative magnitude of the delay length to $\frac{q}{f}$, whose exact shape depends on the considered synapse model. To determine these functions for a wide class of synapses, we proceed by extending the NEF.

# 4 Extending the Neural Engineering Framework

With the example of building a continuous-time delay in hand, we now proceed to extend Principle 3 to arbitrary linear synapses. We focus on building a comprehensive theory for linear systems, but many of these same techniques also carry over to the case of nonlinear dynamical systems with heterogeneous synapses (Voelker, Benjamin, et al., 2017; Voelker & Eliasmith, 2017).

Let $F(s)$ be the transfer function for the linear dynamics that we wish to implement (equations 7 and 11), and let $H(s)$ be the transfer function for an arbitrary linear synapse model ($H(s) = \mathcal{L}\{h(t)\}$). As stated in section 2.3, introducing the synapse model means replacing the integrator ($s^{-1}$) with $H(s)$. This is equivalent to replacing $s$ with $H(s)^{-1}$. Notably, substituting $H(s)$ for the integrator results in the transfer function $F(H(s)^{-1})$, which no longer implements the original, desired dynamics $F(s)$. However, we would like to ensure that the new dynamics match the originally specified $F(s)$. The key insight is that we can determine a new function, $F^H(s)$, such that $F^H(H(s)^{-1}) = F(s)$. That is, we can solve for a function that provides the original dynamics when implemented using the transfer function $H(s)$ as the dynamical primitive. This is formalized by the following definition:

**Definition 4.1.** A function $F^H(s)$ "maps $F$ onto $H$" if and only if it satisfies:

$$F^H\left(\frac{1}{H(s)}\right) = F(s). \tag{15}$$

This definition compactly expresses the notion of a "change of dynamical primitive", in that $F(s)$ is mapped from the canonical primitive, $s^{-1}$, onto some new primitive, $H(s)$. Trivially, $F(s)$ maps itself onto $s^{-1}$. Non-trivial examples are given

throughout sections 4.1–4.4.

Once we identify a $F^H(s)$ that maps $F$ onto $H$, any state-space model $\left(A^H, B^H, C^H, D^H\right)$ that satisfies

$$F^H(s) = C^H\left(sI - A^H\right)^{-1}B^H + D^H \tag{16}$$

will implement the desired dynamics when using $H(s)$ as the dynamical primitive, by equations 11 and 15 (see Figure 7).

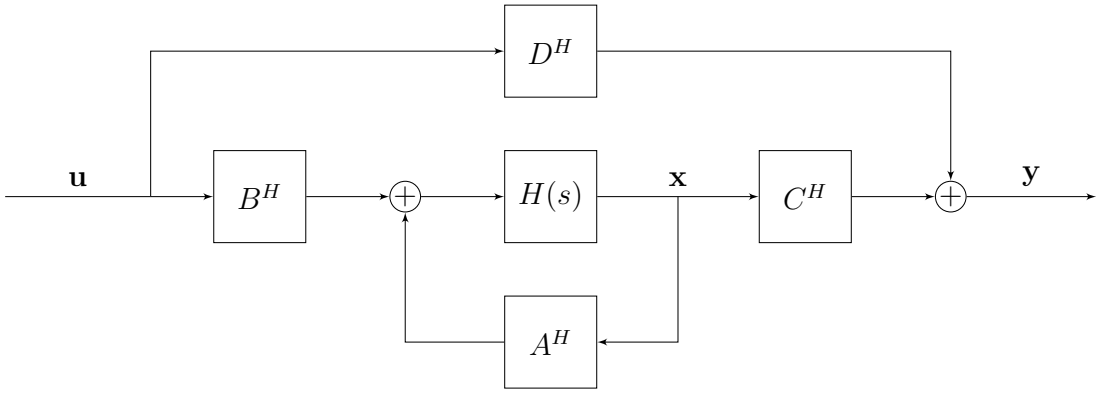


Figure 7: Block diagram for a LTI system, equivalent to Figure 1, with the integrator replaced by a more general linear filter $H(s)$. The state-space model $\left(A^H, B^H, C^H, D^H\right)$ is obtained from some transfer function $F^H(s)$ that maps $F$ onto $H$, as defined in the text. This generalizes Figure 2 to arbitrary linear synapse models.

Therefore, supposing $F^H(s)$ satisfies definition 4.1, and that it is convertible to a state-space model (equation 7), then Figure 7 is just another form of Figure 1, but with the integrator replaced by the synapse. Note that this construction, on its own, does not specify how to find a satisfying $F^H(s)$, nor whether such a function exists, nor whether it can be converted to a state-space model. We provide several examples leading to such a specification in section 4.4.

Before proceeding, we remark that the above theory directly carries over from the

continuous-time domain to the discrete-time domain. The discrete-time formulation of a LTI system is similar to equation 7, but increments time in steps of length $dt$:

$$\begin{aligned} \mathbf{x}[t + dt] &= \bar{A}\mathbf{x}[t] + \bar{B}\mathbf{u}[t], \\ \mathbf{y}[t] &= \bar{C}\mathbf{x}[t] + \bar{D}\mathbf{u}[t], \end{aligned} \tag{17}$$

where the discrete state-space model $\left(\bar{A}, \bar{B}, \bar{C}, \bar{D}\right)$ fully defines the system. The discrete-time equivalent to the Laplace transform is the *z-transform*, named for its use of the variable $z$ to denote the complex frequency domain. In this domain, $z^{-1}$ plays the role of $s^{-1}$, by performing a discrete shift forwards, one step in time (i.e., a delay of one time-step), instead of integration. A well-known result is that the transfer function of this discrete LTI system—defined as the ratio of the $z$-transform of the output to the $z$-transform of the input—is equal to $F(z) = \bar{C}(zI - \bar{A})^{-1}\bar{B} + \bar{D}$. Consequently, all of the previous discussion carries over to discrete LTI systems. In particular, for a discrete synapse expressed using the $z$-transform, $H(z)$, we have the analogous definition:

**Definition 4.2.** A function $F^H(z)$ "maps $F$ onto $H$" if and only if it satisfies:

$$F^H\left(\frac{1}{H(z)}\right) = F(z). \tag{18}$$

Given some $F^H(z)$ that maps $F$ onto $H$, any state-space model $\left(\bar{A}^H, \bar{B}^H, \bar{C}^H, \bar{D}^H\right)$ that satisfies

$$F^H(z) = \bar{C}^H\left(zI - \bar{A}^H\right)^{-1}\bar{B}^H + \bar{D}^H \tag{19}$$

will implement the desired dynamics $F(z)$ when using $H(z)$ as the dynamical primitive. Hence, the task of determining $F^H(\cdot)$ is identical for both continuous- and discrete-time domains—it is only $F(\cdot)$ and $H(\cdot)$ that differ.

## 4.1 Continuous Lowpass Synapse

The first example we consider demonstrates that our new theory recovers the standard form of Principle 3 from the NEF (see section 2.3). For the case of a continuous-time first-order lowpass filter (equation 8), $H(s) = \frac{1}{\tau s+1}$, let:

$$\begin{aligned} F^H(s) &:= C(sI - (\tau A + I))^{-1}(\tau B) + D \\ &= C\left(\left(\frac{s-1}{\tau}\right)I - A\right)^{-1} B + D. \end{aligned}$$

Then,

$$\begin{aligned} F^H(\tau s + 1) &= C\left(\left(\frac{(\tau s + 1) - 1}{\tau}\right)I - A\right)^{-1} B + D \\ &= F(s), \end{aligned}$$

which satisfies definition 4.1. Therefore, by equation 16,

$$\begin{aligned} A^H &= \tau A + I, \qquad & C^H &= C, \\ B^H &= \tau B, & D^H &= D. \end{aligned} \tag{20}$$

This completes our novel proof of Principle 3 from section 2.3.

## 4.2 Discrete Lowpass Synapse

When simulating any NEF network on a digital computer, we necessarily use time-steps of some length $dt > 0$ to advance the state of the network, updating at discrete moments in time (Bekolay et al., 2013). For instance, Nengo currently uses a default of $dt = 1$ ms, and implements a zero-order hold (ZOH) discretization of the synaptic filter. Implementing ZOH means that all continuous-time signals are held constant within

each time-step. This discretization of equation 8 gives:

$$H(z) = \frac{1-a}{z-a}, \quad a := e^{-\frac{dt}{\tau}}.$$

If our desired transfer function is expressed in continuous-time, $F(s)$, then we should also discretize it to $F(z) = \bar{C}(zI - \bar{A})^{-1}\bar{B} + \bar{D}$, with the same time-step, and again using ZOH discretization for consistency. Let,

$$\begin{aligned} F^H(z) &:= \bar{C}\left(zI - \frac{1}{1-a}\big(\bar{A} - aI\big)\right)^{-1}\left(\frac{1}{1-a}\bar{B}\right) + \bar{D} \\ &= \bar{C}\big((z(1-a)+a)I - \bar{A}\big)^{-1}\bar{B} + \bar{D}. \end{aligned}$$

Then,

$$\begin{aligned} F^H\left(\frac{z-a}{1-a}\right) &= \bar{C}\left(\left(\frac{z-a}{1-a}(1-a)+a\right)I - \bar{A}\right)^{-1}\bar{B} + \bar{D} \\ &= F(z), \end{aligned}$$

which satisfies definition 4.2. Therefore, by equation 19,

$$\begin{aligned} \bar{A}^H &= \frac{1}{1-a}\big(\bar{A} - aI\big), \qquad & \bar{C}^H &= \bar{C}, \\ \bar{B}^H &= \frac{1}{1-a}\bar{B}, & \bar{D}^H &= \bar{D}, \end{aligned} \tag{21}$$

provides an exact implementation of the desired system for digital architectures, regardless of the simulation time-step (assuming ZOH).

## 4.3 Delayed Continuous Lowpass Synapse

Next, we consider a continuous-time first-order lowpass filter with a time-delay of $\lambda$:

$$H(s) = \frac{e^{-\lambda s}}{\tau s + 1}. \tag{22}$$

This same model has been proposed by Roth & van Rossum (2009, equation 6.2) as a more realistic alternative to equation 8, that includes an axonal transmission delay of length $\lambda$ (on the order of $\tau$) to account for the finite-velocity propagation of action potentials. By commutativity of convolution, modeling the delay in the synapse (as in equation 22) is equivalent to modeling the delay in spike propagation. Equation 22 may also be used to account for feedback delays within some broader setting (e.g., when the feedback term is computed via some delayed system).

Letting $d := \frac{\lambda}{\tau} e^{\frac{\lambda}{\tau}}$, and $W_0(\cdot)$ denote the principal branch of the Lambert-$W$ function (Corless et al., 1996), we invert $y := H(s)^{-1}$ as follows:

$$\begin{aligned}
& & y &= (\tau s + 1) e^{\lambda s} \\
\iff & & \frac{\lambda}{\tau} e^{\frac{\lambda}{\tau}} y &= \left(\lambda s + \frac{\lambda}{\tau}\right) e^{\lambda s + \frac{\lambda}{\tau}} \\
\iff & & W_0(dy) &= \lambda s + \frac{\lambda}{\tau} \\
\iff & & \frac{1}{\lambda} W_0(dy) - \frac{1}{\tau} &= s,
\end{aligned}$$

where the second-last line assumes that $|\eta| < \pi$ and $\lambda \text{Re}[s] + \frac{\lambda}{\tau} > -\eta \cot \eta$, where $\eta := \lambda \text{Im}[s]$, in order for $\lambda s + \frac{\lambda}{\tau}$ to be within the principal branch (Corless et al., 1996, equation 4.4).[8] Therefore,

$$F^H(s) := F\left(\frac{1}{\lambda} W_0(ds) - \frac{1}{\tau}\right) \tag{23}$$

$$\implies \quad F^H(H(s)^{-1}) = F^H(y) = F(s).$$

As a demonstration of how we might use this mapping, suppose the desired transfer function for our system is a time-delay of $\theta$ seconds, $F(s) = e^{-\theta s}$ (equation 10). In

[8] A simpler (but only sufficient) condition is $\text{Re}[s] \geq -\frac{1}{\tau}$ and $|\text{Im}[s]| < \frac{\pi}{2\lambda}$. Thus, it suffices to consider input frequencies $< \frac{1}{4\lambda}$ Hz.

this setting, we are attempting to "amplify" a delay of $\lambda$ seconds in the synapse into a system delay of $\theta$ seconds at the network level. Letting $c := e^{\frac{\theta}{\tau}}$ and $r := \frac{\theta}{\lambda}$, and then substituting $F(s)$ into equation 23, provides the required function:

$$F^H(s) = \exp\left\{-\theta\left(\frac{1}{\lambda}W_0(ds) - \frac{1}{\tau}\right)\right\} = c\exp\{-rW_0(ds)\} = c\left(\frac{W_0(ds)}{ds}\right)^r.$$

This may be numerically converted into a state-space model, of arbitrary dimensionality $q$, via the $[q-1/q]$ Padé approximants of the following Maclaurin series:

$$F^H(s) = cr\sum_{i=0}^{\infty}\frac{(i+r)^{i-1}}{i!}(-ds)^i. \tag{24}$$

To our knowledge, there is no closed-form expression for the Padé approximants of equation 24, but there are methods to compute them accurately and in $\mathcal{O}(q^2)$ time (Sidi, 2003). Given these approximants, we may follow the same procedure from section 3.1 to obtain a LTI system in the form of equation 7. In section 5.2, we use this approach to improve the accuracy of the delay network demonstrated in section 3.1. We remark that each of $d$, $c$, and $r$ are dimensionless (i.e., unitless) constants that can be used to relate measurable properties of a biological system that may be governed by this description to the necessary network-level computations.

## 4.4 General Linear Synapse

Finally, we consider the general class of all linear synapse models of the form:

$$H(s) = \frac{1}{\sum_{i=0}^{k} c_i s^i}, \tag{25}$$

for some polynomial coefficients $(c_i)$ of arbitrary degree $k$. To the best of our knowledge, this class includes the majority of linear synapse models used in the literature.

For instance, this includes the first-order lowpass synapse that is standard in the NEF. It also includes the second-order alpha synapse, $\frac{1}{(\tau s+1)^2}$ (Rall, 1967)—the convolution of two exponentials with identical time-constants—which is commonly used in biological models (Koch & Segev, 1989; Destexhe et al., 1994b; Mainen & Sejnowski, 1995; Destexhe et al., 1998; Roth & van Rossum, 2009). The alpha synapse essentially filters the spike-trains twice, to produce PSCs with finite (non-instantaneous) rise-times. In addition, equation 25 includes a generalization of the alpha synapse, the double-exponential synapse, $\frac{1}{(\tau_1 s+1)(\tau_2 s+1)}$ (Wilson & Bower, 1989)—the convolution of two exponentials with time-constants $\tau_1$ and $\tau_2$—which has different rise- and fall-times to account for the separate time scales of rapid transmitter binding followed by slow unbinding (Destexhe et al., 1994b; Häusser & Roth, 1997; Roth & van Rossum, 2009). The double-exponential is also a suitable model to account for parasitic capacitances in neuromorphic hardware (Voelker, Benjamin, et al., 2017). Furthermore, equation 25 includes a higher-order generalization of the alpha synapse, the gamma kernel, $\frac{1}{(\mu^{-1} s+1)^k}$ (De Vries & Principe, 1992, equation 19). Finally, this class contains more exotic models, such as the $Q$-bandpass synapse model, $\frac{1}{\frac{1}{\omega^2}s^2+\frac{1}{\omega Q}s+1}$, where $\omega$ is the peak frequency in radians per second, and $Q$ is inversely proportional to the bandwidth. Such synapses have been used to model bandpass filtering from mechanoreceptors in the fingertip (Friedl et al., 2016) or from rods in the retina (Armstrong-Gold & Rieke, 2003).

As well, in appendix A.5, we demonstrate how to use equation 25 to model synapses with pure delays, and to prove a complete connection to ZOH discretization. Specifically, by the use of Padé approximants, we may rewrite any transfer function with a Taylor series expansion (even those that are irrational or improper) in the form of equa-

tion 25—albeit with some radius of convergence. This permits us to model, for instance, synapses with pulse-extended responses (Voelker, Benjamin, et al., 2017).[9] Similarly, the delayed lowpass (equation 22) may be expressed in the form of equation 25. Finally, any linear combinations of the aforementioned synapse models will also be a member of this class. Nonlinear synapse models, such as conductance-based synapses (e.g., Destexhe et al., 1994a, equation 6), are a current subject of study in the NEF (Stöckel, 2017).

To map $F$ onto equation 25, we begin by defining our solution to $F^H(s)$ in the form of its state-space model (see equation 16):

$$\begin{aligned} A^H &= \sum_{i=0}^{k} c_i A^i, & C^H &= C, \\ B^H &= \left( \sum_{j=0}^{k-1} s^j \sum_{i=j+1}^{k} c_i A^{i-j-1} \right) B, & D^H &= D, \end{aligned} \tag{26}$$

which we claim satisfies the required identity, $F^H(H(s)^{-1}) = F(s)$ (equation 15). To

[9] Padé approximants are computed "manually" by Voelker, Benjamin, et al. (2017, equations 9–11) to obtain a second-order approximation in the form of equation 25.

prove this claim, we first rearrange the following expression:

$$\begin{aligned}
\left(\sum_{j=0}^{k-1} s^j \sum_{i=j+1}^{k} c_i A^{i-j-1}\right)(sI - A) &= \sum_{j=0}^{k-1}\sum_{i=j+1}^{k} s^j c_i A^{i-j-1}(sI - A) \\
&= \sum_{i=0}^{k}\sum_{j=0}^{i-1} s^j c_i A^{i-j-1}(sI - A) \\
&= \sum_{i=0}^{k} c_i \left(\sum_{j=0}^{i-1} s^{j+1} A^{i-(j+1)} - s^j A^{i-j}\right) \\
&= \sum_{i=0}^{k} c_i \left(s^i A^{i-i} - s^0 A^{i-0}\right) \\
&= \left(\sum_{i=0}^{k} c_i s^i\right) I - \sum_{i=0}^{k} c_i A^i \\
&= H(s)^{-1} I - A^H.
\end{aligned}$$

We then complete the proof by substituting this result and the state-space model into our expression for the mapped transfer function from equation 16:

$$\begin{aligned}
F^H(H(s)^{-1}) &= C^H(H(s)^{-1} I - A^H)^{-1} B^H + D^H \\
&= C(H(s)^{-1} I - A^H)^{-1} \left(\sum_{j=0}^{k-1} s^j \sum_{i=j+1}^{k} c_i A^{i-j-1}\right) B + D \\
&= C(sI - A)^{-1} B + D \\
&= F(s).
\end{aligned}$$

An alternative derivation may also be found in (Voelker & Eliasmith, 2017, section 2.2). An interesting insight gained with this solution is that it is not, in general, the case that $B^H$ is time-invariant, since it depends on $s$. As a result, solutions will often not be a typical state-space model in the form of equation 7.

Nevertheless, such solutions can still be implemented, as is, in practice. Since $s^j$ is the $j^{\text{th}}$-order differential operator, this form of $B^H$ states that we must supply

the $j^{\text{th}}$-order input derivatives $\mathbf{u}^{(j)}$, for all $j = 1 \dots k-1$. When $k = 1$, as in section 4.1, this is trivial (no derivatives are needed). For $k > 1$, let us first define $B_j^H := \left(\sum_{i=j+1}^{k} c_i A^{i-j-1}\right) B$. Then equation 26 shows that the ideal state-space model must implement the input transformation as a linear combination of input derivatives, $\sum_{j=0}^{k-1} B_j^H \mathbf{u}^{(j)}$. If the derivatives of the input are available to the model, then the $F^H(s)$ we described may be used to precisely implement the desired $F(s)$.

However, if the required derivatives are not included in the neural representation, then it is natural to use a ZOH method by assuming $\mathbf{u}^{(j)} = 0$, for all $j = 1 \dots k-1$:

$$B^H = B_0^H = \left(\sum_{i=1}^{k} c_i A^{i-1}\right) B, \tag{27}$$

with $A^H$, $C^H$, and $D^H$ as before (see equation 26). This is now an equivalent model to equation 26 assuming ZOH, and in the form of the standard state-space model (equation 7). In appendix A.4, we characterize the dynamics of this new system in terms of $F$ and $H$ (independently of the chosen state-space model) for general inputs. We find that equation 27 adds $k-1$ new dimensions, for every dimension in $F$, to the state underlying the resulting dynamical system. In appendix A.5, we show that the our results yield a novel derivation of ZOH discretization for LTI systems.

To our knowledge, this specific state-space architecture has not been explored in theory or in practice. Given that equation 26 requires the input derivatives to accurately compute low-dimensional network-level dynamics, this strongly suggests that it is important to represent and compute derivatives in neural systems. Methods of computing derivatives have been explored by Tripp & Eliasmith (2010) within the NEF. As well, it has long been suggested that adaptive neural mechanisms (e.g., synaptic depression or spike-rate adaptation) may also play a fundamental role in computing such deriva-

tives (Abbott & Regehr, 2004; Lundstrom et al., 2008). However, there has typically been less emphasis placed on understanding temporal differentiation in comparison to other temporal operators, such as integration (Tripp & Eliasmith, 2010).

Finally, we again note that these results translate directly to the discrete-time domain. The required state-space matrices are the same as in equation 26, but with $(\bar{c}_i)$ corresponding to the coefficients of $H(z)$, bars affixed to each state-space matrix, and $z$ substituted for $s$ in $\bar{B}^H$. However, the $z^j$ operator is a shift backwards by $j$ time-steps (i.e., an acausal lookahead), and so the ZOH assumption is instead $\mathbf{u}[t + j(dt)] = \mathbf{u}[t]$ for all $j = 1 \dots k - 1$. Thus, the discrete analog to equation 27 is:

$$\bar{B}^H = \left( \sum_{j=0}^{k-1} \sum_{i=j+1}^{k} \bar{c}_i \bar{A}^{i-j-1} \right) \bar{B}. \tag{28}$$

# 5 Results

We are now in a position to bring together the two main themes of this work: implementing delays and extending the NEF to employ a wide variety of synapse models. To do so, we provide two example applications of our theory. First, we demonstrate that we can map the delay system onto a variety of synapse models used in practice. We show that including an axonal transmission delay in the synapse model significantly improves the ability of the network to compute continuous-time delays (and, by extension, any system that can be expressed in terms of delayed signals). Second, we explore intriguing similarities between our delay network responses and the recently discovered time cells (Eichenbaum, 2014; Tiganj et al., 2016), suggesting that this theory may provide a new understanding of these observed temporal responses in hippocampus,

stratium, medial prefrontal cortex (mPFC), and elsewhere in cortex (Mello et al., 2015; Luczak et al., 2015).

## 5.1 Methods

All networks were built and simulated using Nengo 2.4.0 (Bekolay et al., 2013)—a Python tool for simulating large-scale neural networks—with the nengolib 0.4.1 extension (Voelker, 2017; Voelker & Eliasmith, 2017). Simulations were run on a standard desktop computer[10] using Python 2.7.3, Numpy 1.11.3, and SciPy 0.18.0rc1, linked with the Intel Math Kernel Library (MKL).

All simulations used a $1$ ms time-step (with exception to Figure 9, which used a time-step of $0.01$ ms) and Nengo's default configurations for heterogeneous tuning curves and spiking LIF parameters. As detailed in appendix A.3, encoders were either sampled from the unit-axis vectors, or scattered uniformly along the surface of the hypersphere using quasi-Monte Carlo methods. We further normalized the delay systems using balanced realizations and Hankel singular values.

The architecture of each network follows the setup described by the NEF in section 2. A tutorial by Sharma et al. (2016) provides some additional details regarding the software application of NEF methods. Figures were created using Seaborn (Waskom et al., 2015), and are reproducible via `https://github.com/arvoelke/delay2017`.

[10] Intel® Core™ i7-4770 CPU @ 3.40 GHz.

## 5.2 Delay Networks with Higher-order Synapses

We begin by making the practical point that it is crucial to account for the effect of the simulation time-step in digital simulations, if the time-step is not sufficiently small relative to the time scale of the desired network-level dynamics. To demonstrate this, we simulate a $27$-dimensional delay network using $1{,}000$ spiking LIF neurons, implementing a $0.1$ s delay of $50$ Hz band-limited white noise. We vary the simulation time-step ($dt$) from $0.1$ ms to $2$ ms. The accuracy of our extension does not depend on $dt$ (see Figure 8-Left). When $dt = 1$ ms (the default in Nengo), the standard Principle 3 mapping (equation 20) obtains a NRMSE of $1.425$ ($43\%$ worse than random chance), versus $0.387$ for the discrete lowpass mapping which accounts for $dt$ (equation 21)—a $73\%$ reduction in error. As $dt$ approaches $0$ the two methods become equivalent.

More to the point, we can analyze the delay network's frequency response when using a delayed continuous lowpass synapse (equation 22) instead of the canonical lowpass (equation 8) as the dynamical primitive. This provides a direct measure of the possible improvement gains when using the extension. Figure 8-Right compares the use of Principle 3 (which accounts for $\tau$ but ignores $\lambda$), to our extension (which fully accounts for both; see section 4.3) when $\lambda = \tau$. The figure reveals that increasing the dimensionality improves the accuracy of our extension, while magnifying the error from Principle 3. In the worst case, the Principle 3 mapping has an absolute error of nearly $10^{15}$. In practice, saturation from the neuron model bounds this error by the maximum firing rates. Regardless, it is clearly crucial to account for axonal transmission delays to accurately characterize the network-level dynamics.

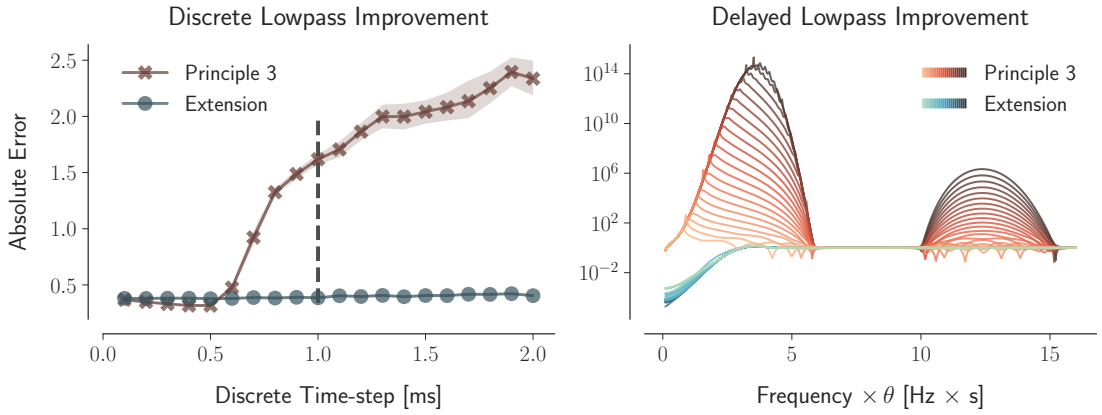


Figure 8: Comparing standard Principle 3 to our NEF extensions. (Left) Error from mapping a 27-dimensional 0.1 s delay onto 1,000 spiking LIF neurons, while varying the simulation time-step ($dt$). The input to the network is white noise with a cutoff frequency of 50 Hz. Unlike our extension, the standard form of Principle 3 does not account for $dt$. A dashed vertical line indicates the default time-step in Nengo. Error bars indicate a 95% confidence interval bootstrapped across 25 trials. (Right) Mapping the delay system onto a delayed continuous lowpass synapse (with parameters $\frac{\tau}{\theta} = 0.1$ and $\frac{\lambda}{\tau} = 1$). The order of the delay system ($q$) is varied from 6 (lightest) to 27 (darkest). Each line evaluates the error in the frequency response, $\left|e^{-\theta s} - F^H(H(s)^{-1})\right|$, where $F^H$ is determined by mapping the delay of order $q$ onto equation 22 using one of the two following methods. The method of our extension—which accounts for the axonal transmission delay—has monotonically increasing error that stabilizes at 1 (i.e., the high frequencies are filtered). The standard Principle 3—which accounts for $\tau$ but ignores $\lambda$—alternates between phases of instability and stability as the frequency is increased.

To more broadly validate our NEF extensions from section 4, we map the delay system onto: (1) a continuous lowpass synapse (see section 4.1); (2) a delayed continuous

lowpass synapse (see section 4.3); and (3) a continuous double-exponential synapse (see section 4.4). We apply each extension to construct delay networks of $2{,}000$ spiking LIF neurons. To compare the accuracy of each mapping, we make the time-step sufficiently small ($dt = 10\,\mu$s) to emulate a continuous-time setting. We use the Padé approximants of order $[5/6]$ for both equations 12 and 24. For the delayed lowpass, we again fix $\frac{\tau}{\theta} = 0.1$ and $\frac{\lambda}{\tau} = 1$. For the double-exponential, we fix $\tau_1 = \tau$ and $\frac{\tau_1}{\tau_2} = 5$. Expressing these parameters as dimensionless constants keeps our results scale-invariant with $\theta$.

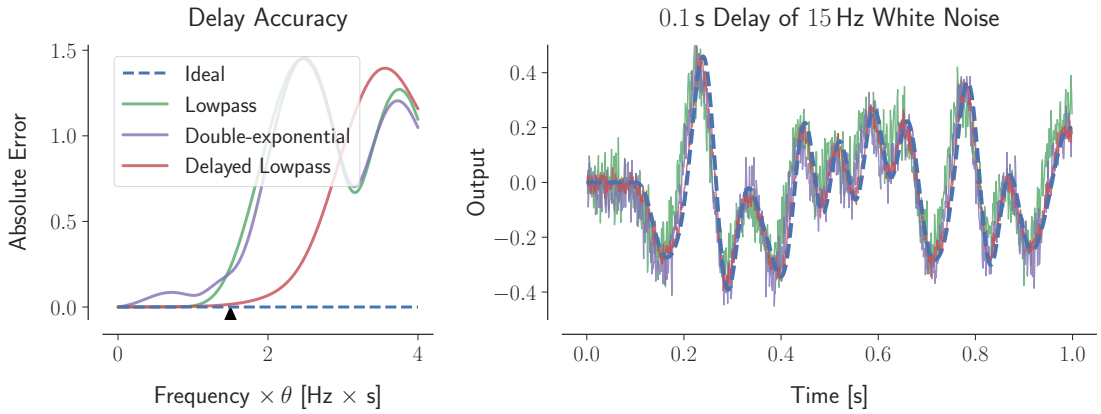


Figure 9: The pure delay mapped onto spiking networks with various synapse models (with parameters $q = 6$, $\frac{\tau}{\theta} = 0.1$, $\frac{\lambda}{\tau} = 1$, $\tau_1 = \tau$, and $\frac{\tau_1}{\tau_2} = 5$). (Left) Error of each mapping in the frequency domain. This subfigure is scale-invariant with $\theta$. (Right) Example simulation when $\theta = 0.1\,$s and the input signal is white noise with a cutoff frequency of $15\,$Hz, corresponding to the triangle (over $1.5$) from the left subfigure. We use a time-step of $0.01\,$ms ($10\,\mu$s) and $2{,}000$ spiking LIF neurons.

Figure 9 reveals that axonal delays may be effectively "amplified" $10$-fold while reducing the NRMSE by $71\%$ compared to the lowpass (see Figure 9-Right; NRMSE for

lowpass=$0.702$, delayed lowpass=$0.205$, and double-exponential=$0.541$). The double-exponential synapse outperforms the lowpass, despite the additional poles introduced by the ZOH assumption in equation 27 (see appendix A.4 for analysis). This is because the double-exponential filters the spike-noise twice. Likewise, by exploiting an axonal delay, the same level of performance (e.g., $5\%$ error) may be achieved at approximately $1.5$ times higher frequencies, or equivalently for $1.5$ times longer network delays, when compared to the lowpass synapse (see Figure 9-Left). In summary, accounting for higher-order synaptic properties allows us to harness the axonal transmission delay to more accurately approximate network-level delays in spiking dynamical networks.

Together, these results demonstrate that our extensions can significantly improve the accuracy of high-level network dynamics. Having demonstrated this for delays, in particular, suggests that the extension is useful for a wide variety of biologically relevant networks. To make this point more concrete, we now turn to a consideration of time cells.

## 5.3 Time Cells

We now describe a connection between the delay network from section 3 and recent neural evidence regarding time cells. Time cells were initially discovered in the hippocampus and proposed as temporal analogs of the more familiar place cells (Eichenbaum, 2014). Similar patterns of neural activity have since been found throughout striatum (Mello et al., 2015) and cortex (Luczak et al., 2015), and have been extensively studied in the rodent mPFC (Kim et al., 2013; Tiganj et al., 2016).

Interestingly, we find that our delay network produces qualitatively similar neural

responses to those observed in time cells. This is shown in Figure 10, by comparing neural recordings from mPFC (Tiganj et al., 2016, Figure 4 C,D) to the spiking activity from a network implementing a delay of the same length used in the original experiments. Specifically, in this network, a random population of $300$ spiking LIF neurons maps a $4.784\,\mathrm{s}$ delay onto an alpha synapse ($\tau = 0.1\,\mathrm{s}$) using our extension. The order of the approximation is $q = 6$ (see equation 13), and the input signal is a rectangular pulse beginning at $t = -1\,\mathrm{s}$ and ending at $t = 0\,\mathrm{s}$ (height $= 1.5$). The simulation is started at $t = -1\,\mathrm{s}$ and stopped at $t = 5\,\mathrm{s}$.

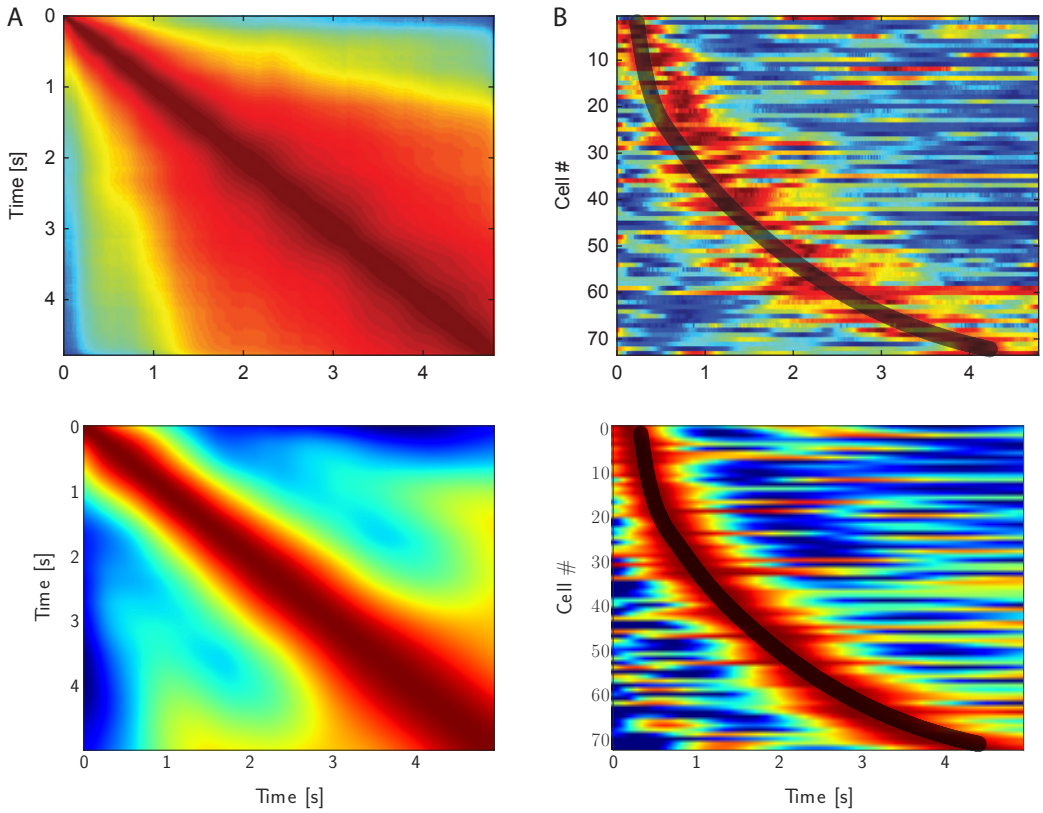


Figure 10: Comparison of time cells to a NEF delay network. (Top) Spiking activity from the rodent mPFC (reproduced from Tiganj et al., 2016, Figure 4 C,D). Neural recordings were taken during a maze task involving a delay period of $4.784\,\mathrm{s}$. (Bottom) Delay network implemented using the NEF (see text for details). $73$ time cells are selected by uniformly sampling encoders from the surface of the hypersphere. (A) Cosine similarity between the activity vectors for every pair of time-points. The diagonal is normalized to the warmest colour. The similarity spreads out over time. (B) Neural activity sorted by the time to peak activation. Each row is normalized between $0$ (cold) and $1$ (warm). We overlay the curve from Figure 6-Bottom ($q = 6$) to model the peak-response times.

We also note a qualitative fit between the length-curve for $q = 6$ in Figure 6 and

the peak response-times in Figure 10. Specifically, Figure 6-Bottom models the non-uniform distribution of the peak response-time of the cells as the length of the trajectory of $\mathbf{x}(t)$ through time. Implicit to this model are the simplifying assumptions that encoders are uniformly distributed, and that the L2-norm of the state-vector remains constant throughout the delay period. Nevertheless, this model produces a qualitatively similar curve when $q = 6$ to both peak response-times from Figure 10-Right (see overlay).

More quantitatively, we performed the same analysis on our simulated neural activity as Tiganj et al. (2016) performed on the biological data to capture the relationship between the peak and width of each time cell. Specifically, we fit the spiking activity of each neuron with a Gaussian to model the peak time ($\mu_t$) and the standard deviation ($\sigma_t$) of each cell's "time field".[11] This fit was repeated for each of the $250$ simulated spiking LIF neurons that remained after selecting only those that had at least $90\%$ of their spikes occur within the delay interval. The correlation between $\mu_t$ and $\sigma_t$ had a Pearson's coefficient of $0.68$ ($\rho < 10^{-34}$), compared to $0.52$ ($\rho < 10^{-5}$) for the biological time cells. An ordinary linear regression model linking $\mu_t$ (independent variable) with $\sigma_t$ (dependent variable) resulted in an intercept of $0.27 \pm 0.06$ (standard error) and a slope of $0.40 \pm 0.03$ for our simulated data, compared to $0.27 \pm 0.07$ and $0.18 \pm 0.04$ respectively for the time cell data. We note that we used the same bin size of $1\,\text{ms}$, modeled the same delay length, and did not perform any parameter fitting beyond the informal choices of $90\%$ cutoff, dimensionality ($q = 6$), area of the input signal ($1.5$), and synaptic time-constant ($\tau = 0.1\,\text{s}$).

[11] We set $a_1 = P = S = 0$ in equation 1 from Tiganj et al. (2016), since we have no external variables to control.

Neural mechanisms previously proposed to account for time cell responses have either been speculative (Tiganj et al., 2016), or rely on gradually changing firing rates from a bank of arbitrarily long, ideally spaced, lowpass filters (Shankar & Howard, 2012; Howard et al., 2014; Tiganj et al., 2015, 2017). It is unclear if such methods can be implemented accurately and scalably using heterogeneous spiking neurons. We suspect that robust implementation is unlikely given the high precision typically relied upon in these abstract models.

In contrast, our proposed spiking model has its network-level dynamics derived from first principles to optimally retain information throughout the delay interval, without relying on a particular synapse model or bank of filters. All of the neurons recurrently work together in a low-dimensional vector space to make efficient use of neural resources. By using the methods of the NEF, this solution is inherently robust to spiking noise and other sources of uncertainty. Furthermore, our explanation accounts for the nonlinear distribution of peak firing times as well as its linear correlation with the spread of time fields.

The observation of time cells across many cortical and subcortical areas suggests that the same neural mechanisms may be used in many circuits throughout the brain. As a result, the neural activity implicated in a variety of delay tasks may be the result of many networks optimizing a similar problem to that of delaying low-frequency signals recurrently along a low-dimensional manifold. Such networks would thus be participating in the temporal coding of a stimulus, by representing its history across a delay interval.

# Conclusion

We have discussed two main theoretical results. The first provides a method for accurately implementing continuous-time delays in recurrent spiking neural networks. This begins with a model description of the delay system, and ends with a finite-dimensional representation of the input's history that is mapped onto the dynamics of the synapse. The second provides a method for harnessing a broad class of synapse models in spiking neural networks, while improving the accuracy of such networks compared to standard NEF implementations. These extensions are validated in the context of the delay network.

Our extensions to the NEF significantly enhance the framework in two ways. First, it allows those deploying the NEF on neuromorphics to improve the accuracy of their systems given the higher-order dynamics of mixed-analog-digital synapses (Voelker, Benjamin, et al., 2017; Voelker & Eliasmith, 2017). Second, it advances our understanding of the effects of additional biological constraints, including finite rise-times and pure time-delays due to action potential propagation. Not only can these more sophisticated synapse models be accounted for, but they may be harnessed to directly improve the network-level performance of certain systems.

We exploited this extension to show that it can improve the accuracy of discrete-time simulations of continuous neural dynamics. We also demonstrated that it can provide accurate implementations of delay networks with a variety of synapse models, allowing systematic exploration of the relationship between synapse- and network-level dynamics. Finally we suggested that these methods provide new insights into the observed temporal properties of individual cell activity. Specifically we showed that time cell

responses during a delay task are well-approximated by a delay network constructed using these methods. This same delay network nonlinearly encodes the history of an input stimulus across the delay interval (i.e., a rolling window) by compressing it into a $q$-dimensional state, with length scaling as $\mathcal{O}\left(\frac{q}{f}\right)$, where $f$ is the input frequency.

While we have focused our attention on delay networks in particular, our framework applies to any linear time-invariant system. As well, though we have not shown it here, as with the original NEF formulation these methods also apply to nonlinear systems. As a result, these methods characterize a very broad class of combinations of synapse- and network-level spiking dynamical neural networks.

Many important questions still remain concerning the interactions between Principles 1, 2, and 3. While the error in our transformations scale as $\mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$ due to independent spiking, it has been shown that near-instantaneous feedback may be used to collaboratively distribute these spikes and scale the error as $\mathcal{O}\left(\frac{1}{n}\right)$ (Boerlin et al., 2013; Thalmeier et al., 2016). This reduction in error has potentially dramatic consequences for the efficiency and scalability of neuromorphics by reducing total spike traffic (Boahen, 2017). However, it is currently unclear whether this approach can be applied to a more biologically plausible setting (e.g., using neurons with refractory periods) while retaining this linear scaling property. Similarly, we wish to characterize the network-level effects of spike-rate adaptation, especially at higher input frequencies, in order to understand the computations that are most accurately supported by more detailed neuron models. This will likely involve extending our work to account for nonlinear dynamical primitives and subsequently harness their effects (e.g., bifurcations) to improve certain classes of computations.

# A Appendix

## A.1 Normalized state-space delay

In this appendix, we symbolically transform equation 12 into a normalized state-space model that avoids the need to compute any factorials. We first do so for the special case of $p = q - 1$, since this provides the best approximation to the step-response (Vajta, 2000). We begin by expanding equation 12:

$$\begin{aligned}
[q-1/q]e^{-\theta s} &= \frac{\sum_{i=0}^{q-1}\binom{q-1}{i}(2q-1-i)!(-1)^i\theta^i s^i}{\sum_{i=0}^{q}\binom{q}{i}(2q-1-i)!\theta^i s^i} \\
&= \frac{\frac{1}{\theta^q(q-1)!}\sum_{i=0}^{q-1}\frac{(q-1)!}{(q-1-i)!i!}(2q-1-i)!\theta^i s^i(-1)^i}{s^q+\frac{1}{\theta^q(q-1)!}\sum_{i=0}^{q-1}\frac{q!}{(q-i)!i!}(2q-1-i)!\theta^i s^i} \\
&= \frac{\sum_{i=0}^{q-1} c_i s^i}{s^q+\sum_{i=0}^{q-1} d_i s^i},
\end{aligned}$$

where $d_i := \frac{q(2q-1-i)!}{(q-i)!i!}\theta^{i-q}$ and $c_i := (-1)^i\left(\frac{q-i}{q}\right)d_i$.

This transfer function is readily converted into a state-space model in controllable canonical form:

$$A = \begin{pmatrix} -d_{q-1} & -d_{q-2} & \cdots & -d_0 \\ 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ 0 & 0 & 1 & 0 \end{pmatrix}, \qquad \begin{aligned} B &= \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix}^{\mathsf{T}}, \\ C &= \begin{pmatrix} c_{q-1} & c_{q-2} & \cdots & c_0 \end{pmatrix}, \\ D &= 0. \end{aligned}$$

To eliminate the factorials in $d_i$ and $c_i$, we scale the $i^{\text{th}}$ dimension of the state-vector by $d_{q-1-i}$, for all $i = 0 \ldots q-1$. This is achieved without changing the transfer function by scaling each $(B)_j$ by $d_{q-1-j}$, each $(C)_i$ by $1/d_{q-1-i}$, and each $(A)_{ij}$ by

$d_{q-1-i}/d_{q-1-j}$, which yields the equivalent state-space model:

$$A = \begin{pmatrix} -v_0 & -v_0 & \cdots & -v_0 \\ v_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ 0 & 0 & v_{q-1} & 0 \end{pmatrix}, \qquad \begin{aligned} B &= \begin{pmatrix} v_0 & 0 & \cdots & 0 \end{pmatrix}^{\mathsf{T}}, \\ C &= \begin{pmatrix} w_0 & w_1 & \cdots & w_{q-1} \end{pmatrix}, \\ D &= 0, \end{aligned}$$

where $v_i := \frac{(q+i)(q-i)}{i+1}\theta^{-1}$ and $w_i := (-1)^{q-1-i}\binom{i+1}{q}$, for $i = 0 \ldots q-1$. This follows from noting that $v_0 = d_{q-1}$ and $v_i := d_{q-1-i}/d_{q-i}$ for $i \geq 1$.

A similar derivation applies to the case where $p = q$, although it results in a pass-through ($D \neq 0$) which is suboptimal for step-responses. For brevity, we omit this derivation, and instead simply state the result:

$$A = \begin{pmatrix} -v_0 & -v_0 & \cdots & -v_0 \\ v_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ 0 & 0 & v_{q-1} & 0 \end{pmatrix}, \qquad \begin{aligned} B &= \begin{pmatrix} -v_0 & 0 & \cdots & 0 \end{pmatrix}^{\mathsf{T}}, \\ C &= \begin{pmatrix} 2(-1)^q & 0 & 2(-1)^q & 0 & \cdots & \cdots \end{pmatrix}, \\ D &= (-1)^q, \end{aligned}$$

where $v_i = \frac{(q+i+1)(q-i)}{i+1}\theta^{-1}$, for $i = 0 \ldots q-1$.

In either case, $A$ and $B$ depend on the delay length solely by the scalar factor $\theta^{-1}$. As a result, we may *control* the length of the delay by adjusting the gain on the input and feedback signals. The NEF can be used to build such controlled dynamical systems, without introducing multiplicative dendritic interactions or implausible on-the-fly connection weight scaling (Eliasmith & Anderson, 2000). The identification of this control factor is connected to a more general property of the Laplace transform, $F(a^{-1}s) = \mathcal{L}\{af(at)\}$ for all $a > 0$, that we can exploit to modulate the width of any filter on-the-fly (in this case affecting the amount of delay; results not shown).

## A.2 Decoding separate delays from the same network

Although the delay network has its dynamics optimized for a single delay $\theta > 0$, we can still accurately decode any delay $0 \leq \theta' \leq \theta$ from the same network. This means that the network is representing a rolling window (i.e., history) of length $\theta$. This window forms a temporal code of the input stimulus.

To compute these other delays, we would like to optimally approximate $e^{-\theta' s}$ with a transfer function $F_{\theta \to \theta'}(s) := \frac{\mathcal{C}(s;\, \theta, \theta')}{\mathcal{D}(s;\, \theta)}$ of order $[p/q]$, such that the denominator $\mathcal{D}(s;\, \theta)$ (which provides us with the recurrent transformation up to a change of basis) depends only on $\theta$, while the numerator $\mathcal{C}(s;\, \theta, \theta')$ (which provides us with the output transformation up to a change of basis) depends on some relationship between $\theta'$ and $\theta$.

From equation 12, we may write the denominator as:

$$\mathcal{D}(s;\, \theta) = \sum_{i=0}^{q} d_i(\theta) s^i, \quad d_i(\theta) := \binom{q}{i} \frac{(p+q-i)!}{(p+q)!} \theta^i.$$

We then solve for the numerator, as follows:

$$\begin{aligned} & [p/q] e^{-\theta' s} = \sum_{i=0}^{\infty} \frac{(-\theta' s)^i}{i!} = \frac{\mathcal{C}(s;\, \theta, \theta')}{\mathcal{D}(s;\, \theta)} \\ \iff \quad & \mathcal{C}(s;\, \theta, \theta') = \left( \sum_{i=0}^{\infty} \frac{(-\theta' s)^i}{i!} \right) \left( \sum_{j=0}^{q} d_j(\theta) s^j \right) + \mathcal{O}(s^{p+1}). \end{aligned}$$

By expanding this product and collecting like terms, the correct numerator up to order $p \leq q$ is:

$$\mathcal{C}(s;\, \theta, \theta') = \sum_{i=0}^{p} c_i(\theta, \theta') s^i, \quad c_i(\theta, \theta') := \sum_{j=0}^{i} \frac{(-\theta')^{i-j}}{(i-j)!} d_j(\theta).$$

Therefore, the optimal readout for a delay of length $\theta'$, given the dynamics for a delay of length $\theta$, is determined by the above linear transformation of the coefficients $(d_j(\theta))_{j=0}^{p}$.

We remark that $c_i(\theta,\theta) = \binom{p}{i}\frac{(p+q-i)!}{(p+q)!}(-\theta)^i$, since $F_{\theta\to\theta}(s) = [p/q]e^{-\theta s}$, by uniqueness of the Padé approximants, and by equation 12. As a corollary, we have proven that the following combinatorial identity holds for all $p, q \in \mathbb{N}$ and $i \in [0, \min\{p,q\}]$:

$$\binom{p}{i} = \sum_{j=0}^{i}(-1)^j\binom{q}{j}\binom{p+q-j}{i-j}.$$

For the case when $p = q-1$, we may also apply the same state-space transformation from appendix A.1 to obtain the normalized coefficients for the $C$ transformation (i.e., with $A$, $B$, and $D$ from equation 13):

$$\begin{aligned}
w_{q-1-i} &= \left(\sum_{j=0}^{i}\frac{(-\theta')^{i-j}}{(i-j)!}\binom{q}{j}\frac{(2q-1-j)!}{(2q-1)!}\theta^j\right)\left(\frac{(q-i)!i!(2q-1)!}{\theta^q(q-1)!q(2q-1-i)!}\theta^{q-i}\right)\\
&= \sum_{j=0}^{i}\binom{q}{j}\left(\frac{(2q-1-j)!}{(i-j)!(2q-1-i)!}\right)\left(\frac{(q-i)!i!}{q!}\right)(\theta^{j-i})(-\theta')^{i-j}\\
&= \binom{q}{i}^{-1}\sum_{j=0}^{i}\binom{q}{j}\binom{2q-1-j}{i-j}\left(\frac{-\theta'}{\theta}\right)^{i-j}, \quad i = 0\ldots q-1.
\end{aligned}$$

## A.3 Remarks on choice of state-space model

Although the transfer function is in fact a unique description of the input-output behavior of a LTI system, the state-space model (equation 7) is not. In general, one may consider any invertible matrix $T$ with the same shape as $A$, and observe that the state-space model $(TAT^{-1}, TB, CT^{-1}, D)$ has the same transfer function as $(A, B, C, D)$. Thus, the state-space model is only unique up to a change of basis. However, in the NEF, the basis $T$ may be "absorbed" into the representation of $\mathbf{x}(t)$ by using the encoders $ET$ in Principle 1, which, in turn, results in the decoders $D^{\mathbf{f}}(T^{-1})^{\mathsf{T}}$ from Principle 2. In other words, considering an alternative state-space model is equivalent to considering a change of basis for the representation.

In practice, when aiming to accurately represent $\mathbf{x}(t)$ using few neurons, it is important to balance the relative range of values within each dimension, such that a typical trajectory for $\mathbf{x}(t)$ stays within the space represented by the distribution of encoders, consistent with the samples of $S$ (see equation 5), and the dynamic range of each neuron. We balance the range of values by numerically computing the $T$ that results in a "balanced realization" of $(A, B, C, D)$ (Laub et al., 1987; Perev, 2011). We then set the encoders to be unit-length and axis-aligned, and optimize each dimension independently by using the methods from section 2.2. As mentioned in the main text, we occasionally include encoders that are scattered uniformly along the surface of the hypersphere using quasi-Monte Carlo sampling—specifically, using the inverse transform method applied to the Sobol sequence with a spherical coordinate transform (Fang & Wang, 1994; Knight et al., 2016)—to visualize a distributed representation. Finally, we occasionally scale each dimension by a diagonal transformation $T$ with the $i^{\text{th}}$ diagonal equaling $\max_t |x_i(t)|$ where $x_i(t)$ is obtained by simulating the desired system directly on a randomly sampled input. We also experimented with a diagonal transformation $T$ with the $i^{\text{th}}$ diagonal corresponding to the reciprocal of $2$ times the sum of the Hankel singular values (Glover & Partington, 1987) of the subsystem corresponding to $x_i(t)$. This has the effect of bounding the absolute value of each dimension above by $1$ in the worst case (Khaisongkram & Banjerdpongchai, 2007). We have found that these methods of normalizing state-space models typically improve the robustness of our networks across a wide range of parameters.

It is worth noting that the mappings from section 4—with the exception of equations 24 and 27—do not alter the representation of $\mathbf{x}(t)$. Disregarding these exceptions,

the same choice of basis is conveniently carried over to the implemented network. Yet, it is also the case that the dynamics of the system mapped by equation 27 do not depend on the chosen state-space model. This fact is proven implicitly in the following appendix, by characterizing the dynamics in terms of $F(s)$ and $H(s)$ alone.

## A.4 Analysis of poles resulting from equation 27

Consider the $F^H$ determined by equation 27, when the derivatives of the input signal are inaccessible. Let $\hat{F}(s) := F^H(H(s)^{-1})$ be the dynamics of the implemented system for this particular $F^H$. Due to the ZOH assumption, $\hat{F} \neq F$ in general, and so $F^H$ does not technically map $F$ onto $H$ (see definition 4.1). As discussed in section 4.4, the approximation only satisfies equation 15 when the input is held constant. To be clear, $\hat{F}(s) = F(s)$ for $s = 0$, but not necessarily for $s \neq 0$. Thus, it is important to characterize the difference between $\hat{F}$ and $F$ for general inputs.

To do so, we can examine the *poles* of the transfer function $F(s) = \frac{\mathcal{C}(s)}{\mathcal{D}(s)}$, which are defined as the complex roots of $\mathcal{D}(s)$. The poles of a system fully define the dynamics of its state (up to a change of basis). For instance, a system is exponentially stable if and only if $\text{Re}[s] < 0$ for all poles $s \in \mathbb{C}$. Furthermore, $s \in \text{sp}(A)$ if and only if $s$ is a pole, where $\text{sp}(A)$ denotes the eigenvalues of the state-space matrix $A$.[12] Therefore, we may characterize the poles of $\hat{F}$ in terms of the poles of $F$, in order to understand the behavior of the implemented system.

We begin by deriving the poles of $F^H$, recalling that $A^H = \sum_{i=0}^{k} c_i A^i$. Let $\mathbf{v} \neq \mathbf{0}$

[12] Note that $\text{sp}(A) = \text{sp}(TAT^{-1})$ for any invertible matrix $T$ with the same shape as $A$.

be an eigenvector of $A$ with eigenvalue $\lambda$, so that:

$$A\mathbf{v} = \lambda\mathbf{v} \quad \Longrightarrow \quad A^H\mathbf{v} = \sum_{i=0}^{k} c_i A^i\mathbf{v} = \left(\sum_{i=0}^{k} c_i\lambda^i\right)\mathbf{v}.$$

Hence, $\mathrm{sp}\big(A^H\big) = \left\{\sum_{i=0}^{k} c_i\lambda^i \,:\, \lambda \in \mathrm{sp}(A)\right\}$ is the full set of eigenvalues for $A^H$. This is also true for equation 26, since $A^H$ is identical, but we do not need this fact.

The denominator of $\hat{F}$ may now be written as $\prod_{\lambda\in\mathrm{sp}(A)}\Big(H(s)^{-1} - \sum_{i=0}^{k} c_i\lambda^i\Big)$. Therefore, the poles of $\hat{F}$ are the roots of the $q$ polynomials:

$$\mathcal{P}(\phi) := \sum_{i=0}^{k} c_i\big(\phi^i - \lambda^i\big), \quad \lambda \in \mathrm{sp}(A).$$

A trivial set of roots are $\phi = \lambda$, and thus each pole of $F$ is also a pole of $\hat{F}$, as desired. However, for a synapse of order $k$, there will also be $k-1$ additional poles for every pole of $F$. For this system to behave as $F$ given low-frequency inputs, we must have the old poles dominate the new poles. That is, we require $\mathrm{Re}[\lambda] \gg \mathrm{Re}[\phi]$ for all $\phi \neq \lambda$.

To provide a specific example, let us consider the double-exponential synapse, $H(s)^{-1} = (\tau_1 s + 1)(\tau_2 s + 1)$,

$$\begin{aligned}
\Longrightarrow \qquad & \mathcal{P}(\phi) = \tau_1\tau_2\phi^2 + (\tau_1+\tau_2)\phi - (\tau_1\tau_2\lambda^2 + (\tau_1+\tau_2)\lambda) = 0 \\
\Longleftrightarrow \qquad & \phi = \frac{-(\tau_1+\tau_2) \pm \sqrt{(\tau_1+\tau_2)^2 + 4\tau_1\tau_2(\tau_1\tau_2\lambda^2 + (\tau_1+\tau_2)\lambda)}}{2\tau_1\tau_2} \\
& \phantom{\phi} = \frac{-(\tau_1+\tau_2) \pm (2\tau_1\tau_2\lambda + (\tau_1+\tau_2))}{2\tau_1\tau_2}.
\end{aligned}$$

In this instance, the '$+$' case gives back the known poles, $\phi = \lambda$, and the '$-$' case provides the new poles, $\phi = \left(-\frac{\tau_1+\tau_2}{\tau_1\tau_2} - \lambda\right)$. Consequently, the poles of $\hat{F}$ are the poles of $F$, duplicated and reflected horizontally about the real line $-b$, where $b := \frac{\tau_1+\tau_2}{2\tau_1\tau_2}$ (and the imaginary components are unchanged).

Interestingly, $b$ may be rewritten as $\left(\frac{\tau_1+\tau_2}{2}\right)/\sqrt{\tau_1^2\tau_2^2}$, which is the ratio of the arithmetic mean to the geometric mean of $\{\tau_1, \tau_2\}$, which may in turn be interpreted as a cross-entropy expression (Woodhouse, 2001).

Regardless, $\hat{F}$ behaves like $F$ when $\text{Re}[\lambda] \gg -b$. For the delay system (equation 13), $\text{Re}[\lambda] \propto -\frac{q}{\theta}$ (in fact, the mean value across all $q$ poles achieves equality), and so we need $b \gg \frac{q}{\theta}$. This predicts that a delay system using the double-exponential synapse, without access to the input's derivative, must necessarily implement a delay that is proportionally longer than $\frac{q}{b}$. Otherwise, the second-order dynamics from the synapse will dominate the system-level dynamics. We note that $b^{-1}$ is longest for the case of the alpha synapse ($b^{-1} = \tau$), and shortest for the case of the lowpass synapse ($b^{-1} \to 0$ as $\tau_2 \to 0$).

## A.5 Relationship to discretization

As an aside, there is an interesting connection between definition 4.1 and the well-known problem of discretizing a linear dynamical system. A discrete LTI system (equation 17) is identical to a continuous LTI system (equation 7), with three adjustments: (1) the integrator $s^{-1}$ is replaced by a time-delay of $dt$ seconds, (2) the input signal is sampled every $dt$ seconds, and (3) the output signal is sampled every $dt$ seconds. Focusing on point 1, this is precisely the notion captured by definition 4.1 with respect to:

$$H(s) = e^{-(dt)s} = \frac{1}{e^{(dt)s}} = \frac{1}{\sum_{i=0}^{\infty} \frac{(dt)^i}{i!} s^i},$$

by the Maclaurin series of $e^x$. This is in the form of equation 25 with $c_i = \frac{(dt)^i}{i!}$ as $k \to \infty$. These coefficients are also the $[0/\infty]$ Padé approximants of $\sum_{i=0}^{\infty} \frac{(-dt)^i}{i!} s^i$.

If we make the ZOH assumption—that the input signal is held piecewise-constant over each continuous-time interval of length $dt$—then $\mathbf{u}^{(j)} = 0$ for all $j \geq 1$. Therefore, by equation 27, an equivalent state-space model is:

$$A^H = \sum_{i=0}^{k} c_i A^i = \sum_{i=0}^{k} \frac{(A(dt))^i}{i!} = e^{A(dt)}, \qquad C^H = C,$$
$$B^H = \left( \sum_{i=1}^{k} c_i A^{i-1} \right) B = A^{-1}\big(A^H - I\big)B, \qquad D^H = D,$$

which is precisely the discrete state-space model $(\bar{A}, \bar{B}, \bar{C}, \bar{D})$ obtained by ZOH discretization. This follows from the fact that points 2 and 3 coincide with the use of the model from equation 17.

This connection helps highlight the generality and consistency of our theory. In particular, the important procedure of discretizing linear state-space models may be viewed as an instance of accounting for changes in dynamical primitives. Furthermore, as one should hope, the ZOH assumption recovers the correct result, which is normally proven by integrating the linear differential equations over the interval $[0, dt]$.

# References


Abbott, L., DePasquale, B., & Memmesheimer, R.-M. (2016). Building functional networks of spiking model neurons. *Nature neuroscience*, *19*(3), 350–355.

Abbott, L., & Regehr, W. G. (2004). Synaptic computation. *Nature*, *431*(7010), 796–803.

Alemi, A., Machens, C., Denève, S., & Slotine, J.-J. (2017). Learning arbitrary dynamics in efficient, balanced spiking networks using local plasticity rules. *arXiv*

*preprint arXiv:1705.08026*.

Appeltant, L., Soriano, M. C., Van der Sande, G., Danckaert, J., Massar, S., Dambre, J., ... Fischer, I. (2011). Information processing using a single dynamical node as complex system. *Nature communications*, *2*, 468.

Armstrong-Gold, C. E., & Rieke, F. (2003). Bandpass filtering at the rod to second-order cell synapse in salamander (Ambystoma tigrinum) retina. *Journal of Neuroscience*, *23*(9), 3796–3806.

Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Stewart, T. C., Rasmussen, D., ... Eliasmith, C. (2013). Nengo: a Python tool for building large-scale functional brain models. *Frontiers in neuroinformatics*, *7*.

Berzish, M., Eliasmith, C., & Tripp, B. (2016). Real-time FPGA simulation of surrogate models of large spiking networks. In *International conference on artificial neural networks (ICANN)*.

Boahen, K. (2017). A neuromorph's prospectus. *Computing in Science & Engineering*, *19*(2), 14–28.

Boerlin, M., Machens, C. K., & Denève, S. (2013). Predictive coding of dynamical variables in balanced spiking networks. *PLoS Computational Biology*, *9*(11), e1003258.

Brogan, W. L. (1991). *Modern control theory (3rd edition)*. Prentice-Hall.

Choudhary, S., Sloan, S., Fok, S., Neckar, A., Trautmann, E., Gao, P., ... Boahen, K. (2012). Silicon neurons that compute. In *International conference on artificial neural networks* (Vol. 7552, pp. 121–128).

Corless, R. M., Gonnet, G. H., Hare, D. E., Jeffrey, D. J., & Knuth, D. E. (1996). On the Lambert W function. *Advances in Computational mathematics*, *5*(1), 329–359.

Corradi, F., Eliasmith, C., & Indiveri, G. (2014). Mapping arbitrary mathematical functions and dynamical systems to neuromorphic VLSI circuits for spike-based neural computation. In *IEEE international symposium on circuits and systems (ISCAS).* Melbourne.

Cunningham, J. P., & Byron, M. Y. (2014). Dimensionality reduction for large-scale neural recordings. *Nature neuroscience*, *17*(11), 1500–1509.

Denève, S., & Machens, C. K. (2016). Efficient codes and balanced networks. *Nature neuroscience*, *19*(3), 375–382.

Destexhe, A., Mainen, Z. F., & Sejnowski, T. J. (1994a). An efficient method for computing synaptic conductances based on a kinetic model of receptor binding. *Neural computation*, *6*(1), 14–18.

Destexhe, A., Mainen, Z. F., & Sejnowski, T. J. (1994b). Synthesis of models for excitable membranes, synaptic transmission and neuromodulation using a common kinetic formalism. *Journal of computational neuroscience*, *1*(3), 195–230.

Destexhe, A., Mainen, Z. F., & Sejnowski, T. J. (1998). Kinetic models of synaptic transmission. *Methods in neuronal modeling*, *2*, 1–25.

De Vries, B., & Principe, J. C. (1992). The gamma model – a new neural model for temporal processing. *Neural networks*, *5*(4), 565–576.

Eichenbaum, H. (2014). Time cells in the hippocampus: a new dimension for mapping memories. *Nature Reviews Neuroscience*, *15*(11), 732–744. doi: 10.1038/nrn3827

Eliasmith, C. (2005). A unified approach to building and controlling spiking attractor networks. *Neural computation*, *7*(6), 1276-1314.

Eliasmith, C. (2013). *How to build a brain: A neural architecture for biological cognition*. Oxford University Press.

Eliasmith, C., & Anderson, C. H. (1999). Developing and applying a toolkit from a general neurocomputational framework. *Neurocomputing*, *26*, 1013–1018.

Eliasmith, C., & Anderson, C. H. (2000). Rethinking central pattern generators: a general approach. *Neurocomputing*, *32–33*, 735-740.

Eliasmith, C., & Anderson, C. H. (2003). *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press.

Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., & Rasmussen, D. (2012). A large-scale model of the functioning brain. *science*, *338*(6111), 1202–1205.

Fang, K. T., & Wang, Y. (1994). *Number-theoretic methods in statistics*. Chapman & Hall.

Friedl, K. E., Voelker, A. R., Peer, A., & Eliasmith, C. (2016, 01). Human-inspired neurorobotic system for classifying surface textures by touch. *Robotics and Automation Letters*, *1*(1), 516-523. doi: 10.1109/LRA.2016.2517213

Ganguli, S., Huh, D., & Sompolinsky, H. (2008). Memory traces in dynamical systems. *Proceedings of the National Academy of Sciences*, *105*(48), 18970–18975.

Gilra, A., & Gerstner, W. (2017). Predicting non-linear dynamics: a stable local learning scheme for recurrent spiking neural networks. *arXiv preprint arXiv:1702.06463*.

Glover, K., & Partington, J. R. (1987). Bounds on the achievable accuracy in model reduction. In *Modelling, robustness and sensitivity reduction in control systems* (pp. 95–118). Springer.

Häusser, M., & Roth, A. (1997). Estimating the time course of the excitatory synaptic conductance in neocortical pyramidal cells using a novel voltage jump method. *Journal of Neuroscience*, *17*(20), 7606–7625.

Howard, M. W., MacDonald, C. J., Tiganj, Z., Shankar, K. H., Du, Q., Hasselmo, M. E., & Eichenbaum, H. (2014). A unified mathematical framework for coding time, space, and sequences in the hippocampal region. *Journal of Neuroscience*, *34*(13), 4692–4707.

Huh, D., & Sejnowski, T. J. (2017). Gradient descent for spiking neural networks. *arXiv preprint arXiv:1706.04698*.

Jaeger, H. (2001). The echo state approach to analysing and training recurrent neural networks. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, *148*, 34.

Kandel, E. R., Schwartz, J. H., Jessell, T. M., et al. (2000). *Principles of neural science* (Vol. 4). McGraw-Hill.

Kauderer-Abrams, E., Gilbert, A., Voelker, A. R., Benjamin, B. V., Stewart, T. C., & Boahen, K. (2017). A population-level approach to temperature robustness in neuromorphic systems. In *IEEE international symposium on circuits and systems (ISCAS)*. Baltimore, MD.

Khaisongkram, W., & Banjerdpongchai, D. (2007). On computing the worst-case norm of linear systems subject to inputs with magnitude bound and rate limit. *International Journal of Control*, *80*(2), 190–219.

Kim, J., Ghim, J.-W., Lee, J. H., & Jung, M. W. (2013). Neural correlates of interval timing in rodent prefrontal cortex. *Journal of Neuroscience*, *33*(34), 13834–13847.

Knight, J., Voelker, A. R., Mundy, A., Eliasmith, C., & Furber, S. (2016). Efficient SpiNNaker simulation of a heteroassociative memory using the Neural Engineering Framework. In *International joint conference on neural networks (IJCNN)*. Vancouver, BC.

Koch, C., & Segev, I. (1989). *Methods in neural modeling*. MIT Press Cambridge, MA.

Laub, A. J., Heath, M. T., Paige, C., & Ward, R. (1987). Computation of system balancing transformations and other applications of simultaneous diagonalization algorithms. *IEEE Transactions on Automatic Control*, *32*(2), 115–122.

Luczak, A., McNaughton, B. L., & Harris, K. D. (2015). Packet-based communication in the cortex. *Nature Reviews Neuroscience*.

Lukoševičius, M. (2012). *Reservoir computing and self-organized neural hierarchies* (Unpublished doctoral dissertation). Jacobs University Bremen.

Lundstrom, B. N., Higgs, M. H., Spain, W. J., & Fairhall, A. L. (2008). Fractional differentiation by neocortical pyramidal neurons. *Nature neuroscience*, *11*(11), 1335–1342.

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, *14*(11), 2531–2560.

Mainen, Z. F., & Sejnowski, T. J. (1995). Reliability of spike timing in neocortical neurons. *Science*, *268*(5216), 1503.

Mello, G. B., Soares, S., & Paton, J. J. (2015). A scalable population code for time in the striatum. *Current Biology*, *25*(9), 1113–1122.

Mundy, A., Knight, J., Stewart, T. C., & Furber, S. (2015). An efficient SpiNNaker implementation of the Neural Engineering Framework. In *International joint conference on neural networks (IJCNN).*

Nicola, W., & Clopath, C. (2016). Supervised learning in spiking neural networks with FORCE training. *arXiv preprint arXiv:1609.02545*.

Padé, H. (1892). Sur la représentation approchée d'une fonction par des fractions rationnelles. *Annales scientifiques de l'École Normale Suprieure*, *9*, 3-93.

Perev, K. (2011). Approximation of pure time delay elements by using hankel norm and balanced realizations. *Problems of Engineering Cybernetics and Robotics*, *64*, 24–37.

Pulvermüller, F., Birbaumer, N., Lutzenberger, W., & Mohr, B. (1997). High-frequency brain activity: its possible role in attention, perception and language processing. *Progress in neurobiology*, *52*(5), 427–445.

Rall, W. (1967). Distinguishing theoretical synaptic potentials computed for different soma-dendritic distributions of synaptic input. *Journal of Neurophysiology*, *30*(5), 1138–1168.

Roth, A., & van Rossum, M. C. (2009). Modeling synapses. *Computational modeling methods for neuroscientists*, *6*, 139–160.

Roxin, A., Brunel, N., & Hansel, D. (2005). Role of delays in shaping spatiotemporal dynamics of neuronal activity in large networks. *Physical review letters*, *94*(23), 238103.

Shankar, K. H., & Howard, M. W. (2012). A scale-invariant internal representation of time. *Neural Computation*, *24*(1), 134–193.

Sharma, S., Aubin, S., & Eliasmith, C. (2016). Large-scale cognitive model design using the Nengo neural simulator. *Biologically Inspired Cognitive Architectures*. doi: http://dx.doi.org/10.1016/j.bica.2016.05.001

Sidi, A. (2003). *Practical extrapolation methods: Theory and applications* (Vol. 10). Cambridge University Press.

Singer, W. (1999). Neuronal synchrony: A versatile code for the definition of relations? *Neuron*, *24*(1), 49–65.

Singh, R., & Eliasmith, C. (2004). *A dynamic model of working memory in the PFC during a somatosensory discrimination task.* Cold Spring Harbour, NY.

Singh, R., & Eliasmith, C. (2006). Higher-dimensional neurons explain the tuning and dynamics of working memory cells. *Journal of Neuroscience*, *26*, 3667-3678.

Stöckel, A. (2017). *Point neurons with conductance-based synapses in the Neural Engineering Framework* (Tech. Rep.). Waterloo, ON: Centre for Theoretical Neuroscience.

Sussillo, D., & Abbott, L. F. (2009). Generating coherent patterns of activity from chaotic neural networks. *Neuron*, *63*(4), 544–557.

Thalmeier, D., Uhlmann, M., Kappen, H. J., & Memmesheimer, R.-M. (2016). Learning universal computations with spikes. *PLoS Comput Biol*, *12*(6), e1004895.

Tiganj, Z., Hasselmo, M. E., & Howard, M. W. (2015). A simple biophysically plausible model for long time constants in single neurons. *Hippocampus*, *25*(1), 27–37.

Tiganj, Z., Jung, M. W., Kim, J., & Howard, M. W. (2016). Sequential firing codes for time in rodent medial prefrontal cortex. *Cerebral Cortex*.

Tiganj, Z., Shankar, K. H., & Howard, M. W. (2017). Neural scale-invariant time-frequency decomposition for detection of spatiotemporal features. *Submitted*.

Tripp, B., & Eliasmith, C. (2010). Population models of temporal differentiation. *Neural Computation*, *22*, 621-659.

Vajta, M. (2000). Some remarks on Padé-approximations. In *3rd TEMPUS-INTCOM symposium.* Veszprém, Hungary.

Voelker, A. R. (2015). Computing with temporal representations using recurrently connected populations of spiking neurons. In *Connecting network architecture and network computation.* Banff International Research Station for Mathematical Innovation and Discovery. doi: 10.14288/1.0304643

Voelker, A. R. (2017). *Nengolib – Additional extensions and tools for modelling dynamical systems in Nengo.* `https://github.com/arvoelke/nengolib/`. (Accessed: 2017-08-12)

Voelker, A. R., Benjamin, B. V., Stewart, T. C., Boahen, K., & Eliasmith, C. (2017). Extending the Neural Engineering Framework for nonideal silicon synapses. In *IEEE international symposium on circuits and systems (ISCAS).* Baltimore, MD.

Voelker, A. R., & Eliasmith, C. (2017). Methods for applying the Neural Engineering Framework to neuromorphic hardware. *arXiv submit/1989058.*

Voelker, A. R., Gosmann, J., & Stewart, T. C. (2017). *Efficiently sampling vectors and coordinates from the n-sphere and n-ball* (Tech. Rep.). Waterloo, ON: Centre for Theoretical Neuroscience. doi: 10.13140/RG.2.2.15829.01767/1

Waernberg, E., & Kumar, A. (2017). Low dimensional activity in spiking neuronal networks. *bioRxiv*. doi: 10.1101/109900

Wang, R., Hamilton, T. J., Tapson, J., & van Schaik, A. (2014). A compact neural core for digital implementation of the Neural Engineering Framework. In *Biomedical circuits and systems conference (BioCAS)* (pp. 548–551).

Waskom, M., Botvinnik, O., Hobson, P., Warmenhoven, J., Cole, J. B., Halchenko, Y., ... et al. (2015, June). *seaborn: v0.6.0.* Zenodo. doi: 10.5281/zenodo.19108

White, O. L., Lee, D. D., & Sompolinsky, H. (2004). Short-term memory in orthogonal neural networks. *Physical review letters*, *92*(14), 148102.

Wilson, M. A., & Bower, J. M. (1989). The simulation of large-scale neural networks. In *Methods in neuronal modeling* (pp. 291–333).

Woodhouse, I. H. (2001). The ratio of the arithmetic to the geometric mean: A cross-entropy interpretation. *IEEE transactions on geoscience and remote sensing*, *39*(1), 188–189.