Computational properties of multi-compartment LIF neurons with passive dendrites

Andreas Stöckel\textsuperscript{1,2,*} and Chris Eliasmith\textsuperscript{1,2}

\textsuperscript{1} Centre for Theoretical Neuroscience, University of Waterloo, Canada
\textsuperscript{2} Applied Brain Research, Waterloo, Canada
* Author to whom any correspondence should be addressed.

E-mail: andreas.stoeckel@appliedbrainresearch.com

Keywords: dendritic computation, neural engineering framework, spiking neural networks

Abstract
Mixed-signal neuromorphic computers often emulate some variant of the LIF neuron model. While, in theory, two-layer networks of these neurons are universal function approximators, single-layer networks consisting of slightly more complex neurons can, at the cost of universality, be more efficient. In this paper, we discuss a family of LIF neurons with passive dendrites. We provide rules that describe how input channels targeting different dendritic compartments interact, and test in how far these interactions can be harnessed in a spiking neural network context. We find that a single layer of two-compartment neurons approximates some functions at smaller errors than similarly sized hidden-layer networks. Single-layer networks with with three compartment neurons can approximate functions such as XOR and four-quadrant multiplication well; adding more compartments only offers small improvements in accuracy. From the perspective of mixed-signal neuromorphic systems, our results suggest that only small modifications to the neuron circuit are necessary to construct more computationally powerful and energy efficient systems that move more computation into the dendritic, analogue domain.

1. Introduction

Biological neurons possess intricately detailed dendrites. Both theoretical and empirical studies suggest that the distribution of synaptic sites along these dendrites has a significant impact on neural computation [1–3]. Still, there is no widely accepted high-level theory of dendritic computation, that, for example, is on a similarly high level of abstraction as the LIF neuron [4].

A better understanding of dendritic computation is not only of interest to neurobiologists. Being able to systematically harness nonlinear dendritic interaction for computation can inform neuromorphic hardware design and thereby increase the computational power of neuromorphic computers at relatively small costs. This is particularly true for mixed-signal neuromorphic systems—computers that use analogue model neurons in conjunction with a digital interconnect (e.g., [5–8]). As we demonstrate below, combing two or more model neurons into a multi-compartment neuron in the analogue domain can sometimes outperform multi-layer networks of the same size while reducing digital communication overhead.

Compared to the complex active dendritic mechanisms often discussed in the theoretical neuroscience literature (e.g., [4, 9, 10]), our notion of dendritic computation is decidedly simple. Specifically, we do not include active effects such as dendritic spikes in our model. Instead, we investigate to what degree passive nonlinear interaction between dendritic compartments provides a substantial computational advantage over standard LIF neurons. This results in a family of ’n-compartment LIF’, or ’n-LIF’ neurons. Forgoing active dendrites has the benefit of keeping neural dynamics mathematically tractable; however, our models do not possess the computational power ascribed to dendrites by the studies listed above. That is, an n-LIF neuron with fixed parameters is generally not as powerful as a multi-layer perceptron (MLP).

Our work is an extension of the neural engineering framework (NEF; [11]) and a direct continuation of our previous study on passive dendritic computation [12]. In a nutshell, the NEF is a method for constructing biologically constrained functional spiking neural networks. This allows researchers to ground high-level function
in low-level biological detail [13–15]. At the same time, by outlining a systematic way to map high-level mathematical functions onto spiking-neural networks, the NEF has been successfully employed as a programing model for neuromorphic hardware [8, 16, 17].

Compared to our previous study, we consider arbitrary configurations of $n$-LIF neurons, and not just two-compartment neurons. In particular, we show that three-compartment LIF neurons can solve the XOR problem and approximate four-quadrant multiplication. This is not possible with two-compartment LIF neurons.

Below, we first discuss our method for approximating functions in spiking neural networks, and suggest theoretical benefits of using neurons with multiple input channels (i.e., neurons with dendrites). We then describe and analyse the model for neuromorphic hardware [8, 16, 17].

2. Function approximation in spiking neural networks

Before we analyse the potential computational benefits of multi-compartment neurons, we describe our approach to constructing neural networks more carefully. Note that the ideas presented in this section have previously been described in greater detail elsewhere [12, 18].

2.1. Synaptic weight optimisation problem

We generally assume that our spiking neural networks are partitioned into neuron populations. Each population represents a vectorial quantity $x$ over some compact set $X \subseteq \mathbb{R}^d$. This representation is established by assigning a normative tuning constraint, or tuning curve, to each neuron $i$. The tuning curve defines the average activity $a_i(x)$ that we expect a neuron to exhibit if its population represents a specific value $x$:

$$a_i(x) = G[J_i(x)] = G[\alpha_i(e_i, x) + \beta_i]. \tag{1}$$

Here, $G[J]$ is a mapping from a somatic input current $J$ onto an average firing rate, and $J_i(x)$ maps from $x$ onto a current. Lastly, $\alpha_i$, $\beta_i$ are a gain and bias, and $e_i$ is a unit encoding vector.

2.1.1. Multi-channel neurons

In this paper, we are mostly interested in neuron models with $k > 1$ input channels. That is, we do not just have a response curve $G[J]$, but a response surface $\mathcal{G}[g]$ that maps some channel state $g = (g_1, \ldots, g_k)$ onto an average neural activity.

However, when solving for synaptic weights $W_i$, we typically do not use $\mathcal{G}$ directly. Instead, we decompose the multivariate response curve $\mathcal{G}$ into functions $G[J]$ and $H(g)$ such that $\mathcal{G}(g) = G[H(g)]$. Here, $G[J]$ maps a current onto an average rate as before, and $H(g)$ maps $g$ onto a current. This decomposition allows us to solve for weights in current-space.

Importantly, we always use equation (1) to assign a tuning curve to each neuron, even for multi-channel neurons. We merely harness the additional degrees of freedom provided by multi-channel neurons to—as we discuss next—realise this tuning and to compute some functions more accurately.

2.1.2. Current-space weight optimisation

Consider two neuron populations representing quantities $x \in X \subseteq \mathbb{R}^d$ and $y \in Y \subseteq \mathbb{R}^d$, respectively, where the population representing $x$ projects onto the population representing $y$. By adjusting the synaptic weights between these two populations, we can approximate functions $f : X \to Y$. In order to do this systematically, we need to find synaptic weights $W_i \in \mathbb{R}^{k \times m}$ for each post-synaptic neuron $i$. These weights must both realise the normative tuning constraint, and, at the same time, approximate the desired function. To simplify this process, we assume that the pre-synaptic neurons already possess the desired tuning. We can then minimise the following loss:

$$E(W_i) = \frac{1}{|X|} \int_X \mathcal{E} \left( H(W_i, a^{pre}(x)), J_i(f(x)) \right)^2 dx. \tag{2}$$

Here, $a^{pre} \in \mathbb{R}^m$ are the combined activities of the pre-populations according to equation (1), and $f$ is the function that we would like to compute in the connection from the pre- to the post-population. Furthermore, $\mathcal{E}$ is the superthreshold error function, defined as follows:
Figure 1. Computing functions \( f(x_1, x_2) \) using either a hidden-layer or by harnessing nonlinearities present in multi-channel neurons. (A) In cases where the variables \( x_1, x_2 \) are represented in separate neuron populations, input must first be fed into a ‘hidden’ neuron population representing both \( (x_1, x_2) \). We can then decode any function over \( (x_1, x_2) \) from this intermediate population. (B) Multi-channel neurons can be harnessed to compute some functions \( f(x_1, x_2) \) without the need for an intermediate population by exploiting the nonlinear interaction between input channels (in this case, excitatory and inhibitory channels, shown in blue and red, respectively).

\[
E(J_{\text{dec}}, J_{\text{tar}}) = \begin{cases} 
J_{\text{dec}} - J_{\text{tar}} & \text{if } J_{\text{tar}} > J_{\text{th}}, \\
J_{\text{dec}} - J_{\text{th}} & \text{if } J_{\text{tar}} \leq J_{\text{th}} \text{ and } J_{\text{dec}} > J_{\text{th}}, \\
0 & \text{if } J_{\text{tar}} \leq J_{\text{th}} \text{ and } J_{\text{dec}} \leq J_{\text{th}}.
\end{cases}
\]  

(3)

This function relaxes the optimisation problem in case the target current \( J_{\text{tar}} \) is below the neuron’s firing threshold \( J_{\text{th}} \). In this case, the magnitude of the decoded current \( J_{\text{dec}} \) does not matter—as long as it is below the firing threshold. We refer to including \( E \) in equation (2) as subthreshold relaxation; this can substantially reduce approximation errors [12].

Equation (2) can further be combined with constraints such as forcing certain entries in \( W_i \) to be zero, or requiring that \( W_i \) be nonnegative. This can be used to construct networks that, for example, adhere to Dale’s principle [19, 20] or spatial connectivity constraints [15].

2.2. Theoretical and practical considerations when using multi-channel neurons

Neuron populations that realise diverse tuning according to equation (1) are universal approximators [21]. By increasing the number of hidden neurons, any function \( f \) over \( \mathbb{X} \) can be linearly decoded from the neural activities with arbitrarily small errors.

Of course, this raises the question: why should we use multi-channel neurons at all? To answer this question, consider the case where two separate neuron populations represent variables \( x_1 \) and \( x_2 \) (figure 1(a)). To compute \( f \) with nonlinear interaction between \( x_1 \) and \( x_2 \) (i.e., functions other than \( f(x_1, x_2) = f_1(x_1) + f_2(x_2) \)), both \( x_1 \) and \( x_2 \) must be represented in a common population with encoders \( e_i \) that are not axis aligned. This can be accomplished by computing the linear function \( f(x_1, x_2) = (x_1, x_2) \) in the connection to an intermediate population, and then computing the desired function in the connection to a target population. However, relying on intermediate populations is not always desirable.

2.2.1. Multi-channel neurons can be more compatible with biological connectivity constraints

One reason to avoid intermediate populations are connectivity constraints in models of neurobiological systems. For example, attention circuits in cerebral cortex do not seem to contain intermediate populations that represent both the signal and the attentional gain [22]. Instead, one hypothesis for how attentional gain could be realised is that gain-modulation (i.e., \( f(x_1, x_2) = x_1 x_2 \)) arises from nonlinear interaction between excitatory and inhibitory input channels [23–25]. Such a ‘dendritic computation’ scheme is illustrated in figure 1(b).

2.2.2. Dendritic computation can support computing some functions more efficiently

A second reason for avoiding intermediate populations is efficiency. Being able to decode any function \( f \) from the pre-synaptic activities implies that \( a^\text{pre}(x) \) implicitly spans a function basis. Assume that a one-dimensional population with \( m \) neurons produces \( \ell \) decodable (within a certain error) basis functions \( \varphi_j(x) \).
A \(d\)-dimensional function basis should then produce \(\ell^d\) decodable basis functions, namely all product terms \(\varphi_j(x_1) \ldots \varphi_{jd}(x_d)\).

Empirically speaking, increasing the dimensionality of a neuron population only results in a moderate increase of decodable basis functions (figure 2). In theory, it takes exponentially more neurons to reach \(\ell^d\) basis functions. It can thus be more efficient (i.e., require fewer neurons and lead to a smaller error) to decode \(k\) one-dimensional functions and to instead rely on nonlinear interaction between input channels to produce the necessary product terms.

### 2.2.3. Dendritic computation is not universal

Unfortunately, such a dendritic computation scheme cannot be a universal function approximator. Specifically, if \(d > 1\), then for any fixed \(H(g_1, \ldots, g_k)\) in \(C^\infty\), \(\varepsilon > 0\), and arbitrary \(f^{\ell}(x_i)\) in \(C^\infty\), there always exist \(f(x_1, \ldots, x_d)\) in \(C^\infty\) such that the following holds almost everywhere on a compact set \(X \subset \mathbb{R}^d\):

\[
\left\| f(x_1, \ldots, x_d) - H \left( f^{\ell}_1(x_1) + \cdots + f^{\ell}_d(x_d), \ldots, f^{\ell}_1(x_1) + \cdots + f^{\ell}_d(x_d) \right) \right\| > \varepsilon.
\]

This follows from the limited number of independent product terms in \(H(\ldots)\) when performing a Taylor series expansion. As a consequence, dendritic computation can only approximate some functions well; yet, as we demonstrate below, this includes practically useful functions.

### 2.3. Example: XOR and four-quadrant multiplication

To illustrate our notion of ‘dendritic computation’, consider the XOR problem [26]. Here, the goal is to assign a label \(y \in \{-1, 1\}\) to the four corners of a square, where diagonal points belong to different classes (figure 3(a)). This can also be phrased as a product: the sign of \(\left[ x_1 x_2 - x_1 - x_2 \right]\) is exactly the XOR pattern (figure 3(b)). 'Four-quadrant multiplication' \(f(x_1, x_2) = x_1 x_2\) is thus a continuous version of the XOR problem; we use this function as a benchmark in our experiments below.

Now, in order to solve the XOR problem, consider the aforementioned case where we represent \(x_1, x_2\) in separate populations. For post-synaptic neurons with a monotonic response \(G[\xi]\), we can only decode additive functions of the form \(f(x_1) + f(x_2)\). Although such functions can be highly nonlinear (figure 3(c)), it is impossible to solve the XOR problem in this way. However, introducing an intermediate layer representing \((x_1, x_2)\) (i.e., constructing an MLP) allows us to decode arbitrary functions over these two variables, including XOR and four-quadrant multiplication (figure 3(d)). Yet, as we discussed above in section 2.2, establishing a powerful multivariate basis in the intermediate population may require a relatively large number of neurons.

The idea behind dendritic computation is to compute functions such as XOR by changing the response curve of the neuron. For example, if we had \(G[\xi] = G[\xi^2 - 1]\), then \(G[f(x_1) + f(x_2)]\) contains product-terms between \(x_1\) and \(x_2\) and we can solve the XOR problem (figure 3(e)). Similarly, for a two-channel neuron \(G[\xi_1, \xi_2] = \xi_1 \xi_2\), we could (trivially) compute four-quadrant multiplication. However, as stated above, we can always find functions that cannot be approximated in this way.

---

**Figure 2.** Number of decodable basis functions over the number of neurons. Each line is the mean number of \(d\)-dimensional Legendre polynomials that can be decoded with an error smaller than the indicated threshold from a ReLU neuron population of size \(m\) (mean is over 100 trials). The three columns correspond to a one-, two- and three-dimensional population. The total number of basis functions tested for each data point is \(2^d\), where \(d\) is the dimensionality of the population. The number of decodable basis functions is not drastically larger for higher-dimensional neuron populations.
3. Mathematical analysis of multi-compartment LIF neurons

As we discussed above, dendritic computation may offer an advantage in scenarios where neural resources are not sufficient to establish a good multivariate function basis. In the next section, we test whether we can observe such a computational advantage in spiking networks using neurons from a family of n-compartment LIF neurons, henceforth called ‘n-LIF’. However, before we perform these experiments, we first need to define n-LIF neurons more carefully and derive the nonlinearity $H$ required for our optimisation problem in equation (2). To gain a better understanding of the theoretical properties of n-LIF neurons, we furthermore provide simple rules that (on a theoretical level) characterise the interaction between input channels. We then explore in how far these predictions hold for biologically plausible n-LIF configurations.

Note that neuron models very similar to the n-LIF neurons discussed here have already been implemented on mixed-signal neuromorphic hardware. For example, the BrainScales 2 system contains analogue switches that can be used to connect the membranes of neighbouring neurons through an adjustable coupling conductance [27, 28].

3.1. Mathematical description of n-LIF neurons

An n-LIF neuron is a connected graph of resistively coupled capacitive compartments. Each neuron possesses exactly one active somatic compartment with LIF dynamics, while the other compartments represent passive dendrites. Compartments may hold any number of passive current- and conductance-based channels. Channels are either constant (e.g., leak channels), or receive external input (synapses). Compartments are connected through coupling conductances $C_{ij}$. Examples of such neurons are depicted in figure 4.

3.1.1. Superthreshold dynamics

In contrast to the standard LIF model, the active n-LIF compartment has an explicit spike model. This is important, as the spike potential causes a substantial current to backpropagate into the dendritic compartments [29].

More specifically, we model the superthreshold dynamics as follows. Whenever the membrane potential $v$ surpasses the threshold $v_{\text{th}}$, $v$ is clamped to a ‘spike potential’ $v_{\text{spike}}$ for a period $\tau_{\text{spike}}$ and subsequently forced to $v_{\text{reset}}$ for the refractory period $\tau_{\text{ref}}$. We use $\tau_{\text{spike}} = 1$ ms and $\tau_{\text{ref}} = 2$ ms in our models.

3.1.2. Subthreshold dynamics

The dynamics of an n-LIF neuron can be described in matrix form. Let $k$ be the number of non-static input channels, and $g \in \mathbb{R}^k$ a vector representing all channel states. We have

$$\frac{d}{dt} C_m \circ v(t) = A [g(t)] v(t) + b [g(t)]$$

$$= - [L + \text{diag} (a' + A' g(t))] v(t) + [b' + B' g(t)] .$$

Here, $C_m \in \mathbb{R}^n$ is a vector of membrane capacitances, ‘$\circ$’ denotes elementwise multiplication, $A[g(t)] \in \mathbb{R}^{n \times n}$ is the ‘voltage feedback matrix’, and $b[g(t)] \in \mathbb{R}^n$ describes the input to the system. We further decompose $A, b$ into input-independent and input-dependent terms: the graph Laplacian $L \in \mathbb{R}^{n \times n}$ and the vectors $a', b' \in \mathbb{R}^n$.
\( \mathbf{b}' \in \mathbb{R}^n \) describe the input-independent portions of the system, whereas the matrices \( \mathbf{A}' \in \mathbb{R}^{n \times k} \) and \( \mathbf{B}' \in \mathbb{R}^{n \times k} \) describe its input-dependent parts. The construction of these matrices is detailed in appendix A.

3.2. Deriving a surrogate model of the dendritic nonlinearity \( H \)

To solve our current-space weight optimisation problem in equation (2), we need to find a function \( H(\mathbf{g}) \) that maps a synaptic state \( \mathbf{g} = \mathbf{W}_a \mathbf{a}^{\text{pre}} \) onto an average somatic current \( \mathbf{J} \). Trivial cases aside, this \( H \) cannot be stated closed form. This is due to the nonlinear interaction between the membrane potential and conductance-based input channels, as well as the nonlinear superthreshold dynamics. Still, we can derive a parametrised ‘surrogate model’ that approximates \( H \) well and that can—as we discuss below—be fitted to empirical data.

For the purpose of building networks of spiking neurons, we are primarily interested in the superthreshold regime. We can thus exploit the fact that \( n \)-LIF neurons are tonically spiking. That is, the somatic compartment oscillates between the reset and threshold potential with a fixed frequency—the somatic membrane potential is effectively clamped to some value \( \bar{v} \) between reset and threshold potential.

To estimate the average current flowing into the somatic compartment, we replace the system from equation (4) with a reduced system described by matrices \( \hat{\mathbf{A}}[\mathbf{g}] \), \( \hat{\mathbf{b}}[\mathbf{g}] \). For constant \( \mathbf{g} \), we obtain an equilibrium state \( \tilde{\mathbf{v}}^{\text{eq}} \):

\[
\tilde{\mathbf{v}}(t) = \mathbf{v}^{\text{eq}} + \exp \left( \hat{\mathbf{A}}[\mathbf{g}] \text{diag}(\hat{\mathbf{C}}_m)^{-1} t \right) (\tilde{\mathbf{v}}(0) - \mathbf{v}^{\text{eq}}), \quad \text{where} \quad \mathbf{v}^{\text{eq}} = -\hat{\mathbf{A}}[\mathbf{g}]^{-1} \hat{\mathbf{b}}[\mathbf{g}].
\]  

(5)

As we describe in more detail in appendix A, the reduced system is specifically constructed such that \( \mathbf{v}^{\text{eq}} \) converges to the voltage that we would obtain if the somatic compartment were clamped to \( \bar{v} \). In the somatic compartment itself, the voltage difference \( \tilde{\mathbf{v}}^{\text{eq}} - \bar{v} \) is proportional to the current flowing into the compartment. Given a vector \( \tilde{\mathbf{c}} \) of somatic coupling conductances with \( \tilde{c}_1 = 1 \), the average current \( H(\mathbf{g}) \) flowing into the soma is modelled as \( H(\mathbf{g}) = (\tilde{\mathbf{c}}, \tilde{\mathbf{v}}^{\text{eq}}) \).

3.3. Theoretical properties of the \( n \)-LIF nonlinearity

As we discussed in section 2.2, there must be some nonlinear interaction between different input channels for dendritic computation to offer any advantage. Analysing the dendritic nonlinearity \( H(\mathbf{g}) \) sheds some light on how far this is the case for \( n \)-LIF neurons (see appendix B for a mathematical derivation). To summarise, and as is illustrated in figure 5, there are three types of interaction between inputs: affine, shunting, and multiplication.

3.3.1. Affine terms

The ‘computationally weakest’ \( n \)-LIF neurons are those where the somatic current is merely an affine function over the input channels. This is the case for \( n \)-LIF neurons that only possess current-based input channels, or any \( n \)-LIF neuron where input is solely fed into the somatic compartment (figures 4(a), (b) and 6(a), (b); [30]).
Figure 5. Effects of the relative location of input channels on the dendritic nonlinearity \( H \). (A) Dendritic tree used in our example. The second compartment is only connected through the soma to the third and fourth compartment. (B) Table illustrating how pairs of input channels in (A) interact. Grey rectangles correspond to branches with nonlinear interaction. The symbol ‘/’ indicates that \( H \) only contains linear combinations of terms containing each variable; ‘\( \div \)’ indicates that the variable in this column acts divisively on the other variable in the numerator; ‘•’ indicates that \( H \) contains product terms with these two variables.

Figure 6. Dendritic nonlinearity models \( H \) for the \( n \)-LIF neurons depicted in figure 4. Hatched regions correspond to subthreshold currents. Limits were chosen such that the spike onset is approximately on the diagonal of each plot. Shared parameters are given in table D1. (A) and (B) Single-compartment neurons with current- and conductance-based synapses. Apart from scaling, the two models are equivalent. (C) Two-compartment neuron with \( c_{12} = 30 \) nS. The contour lines are still straight lines but no longer parallel due to shunting. (D) Slice through the response curve of a three-compartment LIF neuron with \( c_{12} = 40 \) nS, \( c_{23} = 100 \) nS. The inputs \( g_{2E} = 95 \) nS and \( g_{1I} = 20 \) nS are kept constant. The contour-lines are curved.

3.3.2. Shunting
Nonlinear interaction between the input channels of an individual dendritic compartment is limited to shunting (cf [31], section 1.5). Specifically, conductance-based input channels act ‘divisively’ on the other input channels in that compartment; an example would be the two-compartment neuron in figure 4(c) where the excitatory and inhibitory conductances \( g_{E} \) and \( g_{I} \) both act divisively on \( H \) (cf figure 6(c)).

Shunting in a single compartment cannot be used to compute ‘interesting’ functions such as XOR [12]. Still, as is visible in our experiments below, and as is described in [12], a single layer of two-compartment neurons can compute a wide range of functions with a substantially smaller error than a single layer, or, in some cases, even two layers of standard LIF neurons.

3.3.3. Product terms
Input channels targeting different dendritic compartments on the same branch interact multiplicatively (figures 4(d), (e) and 6(d)). Notably, there is at most one current-based input channel per product term;
current-based inputs cannot interact non-linearly. Correspondingly, \( n \)-LIF neurons with three or more compartments could, at least in theory, compute four-quadrant multiplication.

However, exploiting the product terms of conductance-based inputs to compute multiplication over all four quadrants is difficult. Multiplicative terms are similar in magnitude to the linear terms. We must thus find synaptic weights that compensate for the linear terms, and, in addition, compute the desired function. Furthermore, there is a tradeoff between the coupling conductances \( c_i \) and the maximum somatic current. If \( c_i \) is small, it becomes more difficult for distal compartments to influence the soma. However, countering this by choosing larger \( c_i \) results in more linear \( H \), which makes these inputs less computationally useful.

### 3.4. Evaluating the model quality

Our theoretical observations are only of value if the neuron actually behaves as our model predicts. Recall that the somatic current model \( H \) (cf section 3.2) is the result of major simplifications. For example, we assumed that the somatic compartment is effectively clamped to a constant potential \( \bar{\ell} \), that \( g \) is constant, and that spike generation has no effect on the dendritic compartments. These assumptions are readily violated in practice.

Thus, equation (5) is better interpreted as a ‘template’ for the overall mathematical shape of the equilibrium potential \( \bar{v}^{eq}(g) \) and the non-linearity \( H(g) = \langle c, \bar{v}^{eq}(g) \rangle \). We thus declare non-zero entries in \( \mathbf{A}, \mathbf{B} \) to be free parameters and use gradient-based optimisation such as Adam [32], LBFGS-B, or a sequential quadratic programme (SQP) to calibrate these parameters. In our testing, all of these methods perform equally well. In the context of this paper, we use the SQP implemented in our software library libnlif, which in turn is based on the ‘OSQP’ solver library [33]3. In appendix C, we describe a conceptually very similar SQP that we use to solve for synaptic weights and that is also implemented in libnlif.

Figure 7 depicts empirical spike rate measurements together with our model predictions \( H(g) \) after fitting the model to the depicted ground truth for the two-, three-, and four-compartment models depicted in figure 4 (see table D1 for parameters). For two-compartment neurons (figure 7(a)), \( H \) can be made to fit empirical rates extremely well (with RMSEs below 2 s

### 3.5. Empirical evaluation of the \( n \)-LIF nonlinearity

According to our analysis from section 3.3, adding more compartments and input-channels to \( n \)-LIF neurons should increase their computational power. We now test in how far this holds empirically. Of course, we cannot test every possible \( n \)-LIF neuron configuration and parameter set; however, due to symmetries in \( n \)-LIF neurons with multiple branches, the sequential arrangements of compartments in figure 4 are, at least in theory and according to preliminary tests, the most computationally powerful \( n \)-LIF configurations. We manually tuned the coupling conductances to balance maximum somatic current and non-linearity (cf section 3.3.3).

#### 3.5.1. Methods

Consider a scenario where two variables \( x_1, x_2 \) are represented in two separate one-dimensional neuron populations (figure 1(b)). To assess the ‘computational power’ of different \( H \) in an unbiased manner, we generate random current functions \( J_i(x_1, x_2) \) with increasing frequency content \( \rho^{-1} \) (cf figure 8(a); \( \rho \) is the standard-deviation of a low-pass filter).

We sample 256 training samples \((x_1, x_2)\) from a 100 \(
\times \) 100 grid; the test error is computed over all 10000 points with normal-distributed noise (standard deviation of \( 10^{-2}d_{\text{max}} \)) added to the pre-activities to test generalisation. We assume \( J_{Ih} = 0 \) nA for subthreshold relaxation, and use an SQP implemented in libnlif to locally optimise the synaptic weights \( \mathbf{W} \) to equation (2) for the \( H \) corresponding to the different neuron models with the fitted parameters from above. Again, it is possible to use Adam or quasi-Newton methods such as L-BFGS-B to optimize the synaptic weights, but we did not observe a substantial difference in the results with respect to the approximation errors, and our SQP-based solver (appendix C) typically converges within thirty epochs.

#### 3.5.2. Results

Results are depicted in figure 8(b). Note that the performance of the three- and four-compartment neuron is statistically indistinguishable for \( \rho^{-1} > 0.4 \). Surprisingly, the difference between the two- and three-compartment neuron is relatively subtle, with the largest difference at \( \rho^{-1} \approx 1.5 \) and a median reduction in error of 38\%. Still, the three- and four-compartment neurons perform slightly better than a multiplicative baseline (i.e., \( H(g_1, g_2) = g_1g_2 \)), while the two-compartment neuron is slightly worse and features a much higher variance. As in our previous study [12], we find that two-compartment neurons substantially outperform one-compartment neurons.

3 libnlif is available online at https://github.com/astoeckel/libnlif.
3.5.3. Discussion

The computational advantage of three- and four-compartment neurons is not as clear-cut as is suggested by our theoretical analysis. While better performance compared to the multiplicative baseline indicates that three- and four-compartment neurons are able to compute multiplicative and not just merely divisive terms, this only results in a small average improvement in performance for random functions $J_{\rho}$.

This negligible increase in performance for four-compartment neurons likely stems from the complications with product terms listed in section 3.3.3. It might be possible to find more balanced configurations of the three- and four-compartment neurons that offer a larger computational benefit, but we were not able to do so in our experiments.

Another caveat with this experiment is that the optimisation problems for three- and four-compartment neurons are highly non-convex. The weight solver is neither guaranteed nor expected to converge to an optimal solution. Correspondingly, three- and four-compartment neurons may be more powerful than what is suggested here; however, we were not able to improve upon our results using a more exhaustive stochastic optimisation scheme such as Adam with random restarts. In contrast, equation (2) is rather docile in the case of the two-compartment neuron and our solutions tend to be close to the global optimum [18].

4. Spiking neural network experiment

Our theoretical analysis of the $n$-LIF neuron family in section 3.3 predicted that models with three or more compartments can compute products of the input channel state variables $g_1, \ldots, g_k$. Further evidence for this is that the three-compartment nonlinearity model $H$ slightly outperforms the multiplicative baseline $H(g_1,g_2) = g_1g_2$ (section 3.5). We furthermore saw in section 3.4 that our model $H$ can be used to predict the empirical neuron rate quite well—at least after fitting neuron parameters.

As such, it might be possible to solve the XOR problem, and to compute four-quadrant multiplication (cf section 2.3) with these neurons. The goal of this section is to put this theory to the test by placing the neurons...
Figure 8. Multi-compartment current decoding error for random target functions with increasing 'complexity' $\rho^{-1}$.
(A) Examples depicting some of the random current functions $J_\rho$ used in the experiment. Functions have zero mean and an RMS of 0.5 nA. The inverse low-pass filter coefficient $\rho^{-1}$ can be seen as corresponding to the 'complexity' of the functions; this value is varied on the horizontal axis in (B). The horizontal location of the diagrams roughly corresponds to their $\rho^{-1}$ value on the axis below. (B) Normalised superthreshold current error (cf equation (2)) using the neurons depicted in figures 4(a) and (c)–(e) over functions of increasing complexity (see above). Lines are the median over 500 trials; shaded areas correspond to the 25th and 75th percentiles. The multiplicative baseline is for $H(g_1, g_2) = g_1 g_2$. Results for the three- and four-compartment neurons are not significantly different for $\rho^{-1} > 0.4$.

Figure 9. Positive intercepts induce large current decoding errors when computing four-quadrant multiplication. (A) Target population tuning curves coloured by the relative static decoding error. (B) Same data, but in terms of the $x$-intercepts and the maximum firing rate. Positive intercepts induce the largest errors. (C) Visualisation of the current function for the tuning curves highlighted in (B). Coloured background corresponds to the target currents, dashed contour lines to the decoded somatic current, grey background corresponds to subthreshold currents. Errors $E$ are the NRMSE.

discussed above in a spiking neural network and solving for synaptic weights that compute some benchmark functions.

4.1. Solving for biologically plausible weights with Dale’s principle
We compare the performance of our ‘dendritic computation’ network (figure 1(b)) to a two-layer neural network (figure 1(a)) with a similar amount of neural resources in terms of the total number of compartments.
Figure 10. Computing four-quadrant multiplication in a spiking neural network using three-compartment LIF neurons. (A) Top two plots: inputs $x_1(t)$ and $x_2(t)$ represented by the pre-populations. Bottom: mathematical target $x_1(t)x_2(t)$, filtered target function, as well as the decoded target population output. This particular trial reaches a dynamic network error of $E_{\text{net}} = 16\%$. (B) Spike rasters depicting the activity of each population (only half of the neurons are depicted). All pre-neurons act both excitatorily and inhibitorily.

Specifically, two pre-populations represent $x_1$ and $x_2$, respectively, and we would like to represent $f(x_1, x_2)$ in the target population, where $f$ is the benchmark function. Benchmark functions include addition, multiplication, division, and the maximum. Where applicable, we approximate our benchmark functions $f$ over the domain $[-1, 1]^2$; for some functions, we provide results for $[0, 1]^2$.

In every scenario, the two pre-populations and the target population possess 100 neurons each. The intermediate population—if used—consists of 200 neurons with fixed random tuning curves as described in section 2.1; in other words, we perform layer-wise and not global optimisation of the network. Tuning curves are chosen such that all $x$-intercepts $\xi_0$ are in $[-0.95, 0.95]$. Maximum firing rates are sampled from 50 Hz to 100 Hz. All other parameters are as specified in table D1. We furthermore adhere to Dale’s principle and mark 30\% of all neurons as purely inhibitory.

The total number of compartments is $200 + 100n$ in our $n$-LIF experiments; the intermediate population experiment requires 500 compartments. The number of non-zero synaptic weights is about 20 000 for the one- and two-compartment experiments, 60 000 for the two-layer experiment, and $20 000(n − 1)$ for three- and four-compartment neurons.

Data are collected over a ten-second simulation during which the inputs $x_1(t)$, $x_2(t)$ follow a fourth order Hilbert curve. We report the NRMSE between the decoded network output over time and a reference signal passed through the same synaptic filters (see figure 10, which we refer to later, as an example). We use the SQP solver described in appendix C.

Results are listed in the upper portion of table 1 under ‘standard parameters’. Notably, dendritic computation with multi-compartment neurons outperforms the two-layer network for most benchmark functions. However, among our $n$-LIF neurons, baseline errors tend to increase with the number of compartments $n$. That is, errors for the three-compartment neuron are slightly higher than those for the two-compartment neuron, etc.

A notable exception to this is four-quadrant multiplication, i.e., $x_1 \times x_2$ over $[-1, 1]^2$. Here, the three- and four-compartment neurons clearly outperform the LIF and two-compartment LIF neurons. The one-compartment LIF neuron reaches an NRMSE of 100\% — the network outputs a constant zero, which indeed is the best linear approximation of multiplication over all four quadrants. Similarly, since the two-compartment LIF neuron can only approximate two out of the four quadrants, the error is at 79\%. In contrast, the three- and four-compartment neurons reach errors of about 36\%; the resulting network solves the XOR problem, however, the output does not properly match four-quadrant multiplication at all points. The two-layer network clearly outperforms all other setups with a 13\% error. However, keep in mind that this is the only benchmark function where the two-layer network has a smaller error compared to the dendritic computation schemes, while requiring more resources.
four-quadrant multiplication well. Apart from these deviations, three-compartment neurons seems to be able, qualitatively speaking, to compute multiplicative terms and should be capable of solving the XOR problem. We confirmed this in a spiking network context, where two-compartment neurons outperform larger two-layer networks in a wide range of benchmark tasks. Specifically, we found (cf section 4.1), that two-compartment neurons often reach 50% of the approximation error at about half of the neural resources in terms of neural compartments.

4.2. Improving four-quadrant multiplication performance

The experiment in section 3.5 indicated that our three- and four-compartment neuron models should outperform a multiplicative baseline nonlinearity. We would thus expect that we can approximate four-quadrant multiplication with errors much lower than observed 30%.

The reason why we observe much higher errors in a network context is illustrated in figure 9. The current functions $J_i(x_1,x_2) = \alpha_i (\xi_i - x_1 x_2) + \beta_i$ induced by target population tuning curves with negative $x$-intercepts $\xi_0$ (i.e., $\beta_i > 0$) can indeed be approximated well. In other words, we can compute four-quadrant multiplication extremely well for individual post-neurons. However, the weight solver does not find good solutions for the current functions resulting from tuning curves with positive intercepts.

Potential culprits are the increased steepness of the current function, our random sampling not being dense enough to cover the relevant portions of $J_i(x_1,x_2)$, and systematic biases in the weight solver. A pragmatic solution to this problem is to simply change the tuning curves of the target population. This affects the classes of functions that can be decoded from this population.

The bottom portion of table 1 labelled ‘adapted parameters’ lists the results of an experiment where we tune the target-population $x$-intercepts $\xi_0$ to be strictly negative. We furthermore do not account for Dale’s principle—they effectively increases the number of pre-neurons and doubles the number of synaptic weights.

With these changes, we are able to reach a mean dynamic network error $E_{net}$ of about 20%, albeit with high inter-trial standard deviations of $\pm 5\%$. Some trials reach errors below 15%. Figure 10 depicts an example trial. Errors are mostly visible at the extrema, that is, the target population does not quite represent 1 and $-1$. Apart from these deviations, three-compartment neurons seems to be able to, qualitatively speaking, compute four-quadrant multiplication well.

5. Discussion

We have explored dendritic computation as a method for function approximation in spiking neural networks. We saw that, from a theoretical perspective, networks utilizing dendritic computation can outperform multi-layer neural networks with fixed encoders. To put this to the test, we analysed a family of $n$-compartment LIF neurons with passive dendrites—one of the simplest possible neuron models that supports dendritic computation and that can be realised on mixed-signal neuromorphic hardware platforms by connecting multiple neuron circuits.

Our theoretical analysis of $n$-LIF neurons in section 2.2 suggested that three-compartment neurons can compute multiplicative terms and should be capable of solving the XOR problem. We confirmed this in a spiking neural network experiment, where two-compartment neurons outperform larger two-layer networks in a wide range of benchmark tasks. Specifically, we found (cf section 4.1), that two-compartment neurons often reach 50% of the approximation error at about half of the neural resources in terms of neural compartments.

Table 1. Function approximation errors $E_{net}$ for spiking networks using various $n$-LIF neurons. Results are the mean and standard deviation over 100 trials. The best result for a particular target function is set in bold; darker background colours indicate a worse ranking of the result. $\lambda$ is a regularisation factor, $\xi_0$ denotes the range of the tuning-characteristic $x$-intercepts (cf section 2.1).

| Neuron | Domain | Current-based LIF | Conductance-based $n$-LIF |
|--------|--------|-------------------|--------------------------|
|        | $x_1 + x_2$ | $[1, -1]$ | $[0.1, 1]$ |
|        | $x_1 (1 + x_2)$ | $[0, 1]$ | $[1, 0]$ |
|        | $\sqrt{x_1 + x_2}$ | $[0, 1]$ | $[1, 0]$ |
|        | $x_1 \times x_2$ | $[0, 1]$ | $[1, 0]$ |
|        | $x_1 \times x_2$ | $[-1, 1]$ | $[0, 1]$ |
|        | $[(x_1, x_2)]$ | $[-1, 1]$ | $[0, 1]$ |
|        | atan($x_1, x_2$) | $[-1, 1]$ | $[0, 1]$ |
|        | max($x_1, x_2$) | $[-1, 1]$ | $[0, 1]$ |
| Standard | $n = 2$ | $n = 3$ | $n = 4$ |
| $x_1 + x_2$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ |
| $x_1 (1 + x_2)$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ |
| $\sqrt{x_1 + x_2}$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ |
| $x_1 \times x_2$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ |
| $x_1 \times x_2$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ |
| $x_1 \times x_2$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ |
| $[(x_1, x_2)]$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ |
| atan($x_1, x_2$) | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ |
| max($x_1, x_2$) | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ |

Adapted parameters ($\lambda = 10^{-3}, \xi_0 \in [-0.95, 0.95];$ with Dale’s principle, $p_{inh} = 30\%$)

| Neuron | Domain | Current-based LIF | Conductance-based $n$-LIF |
|--------|--------|-------------------|--------------------------|
|        | $x_1 + x_2$ | $[1, -1]$ | $[0.1, 1]$ |
|        | $x_1 (1 + x_2)$ | $[0, 1]$ | $[1, 0]$ |
|        | $\sqrt{x_1 + x_2}$ | $[0, 1]$ | $[1, 0]$ |
|        | $x_1 \times x_2$ | $[0, 1]$ | $[1, 0]$ |
|        | $x_1 \times x_2$ | $[-1, 1]$ | $[0, 1]$ |
|        | $[(x_1, x_2)]$ | $[-1, 1]$ | $[0, 1]$ |
|        | atan($x_1, x_2$) | $[-1, 1]$ | $[0, 1]$ |
|        | max($x_1, x_2$) | $[-1, 1]$ | $[0, 1]$ |
| $n = 2$ | $n = 3$ | $n = 4$ |
| $x_1 + x_2$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ |
| $x_1 (1 + x_2)$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ |
| $\sqrt{x_1 + x_2}$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ |
| $x_1 \times x_2$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ |
| $x_1 \times x_2$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ |
| $x_1 \times x_2$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ |
| $[(x_1, x_2)]$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ |
| atan($x_1, x_2$) | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ |
| max($x_1, x_2$) | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ | $1.0 \pm 0.2\%$ |
Dendritic computation outperforming the two-layer network for various target functions is caused by the two-layer network establishing a two-dimensional basis with only 200 neurons. This results in a substantial static decoding error (cf [11], section 2.2). Dendritic computation in a two-compartment neuron can thus, especially in noisy spiking neural network contexts, be a viable alternative. Moreover, given that our neural compartments are merely passively coupled LIF neurons without the active circuitry, we would expect that the two-compartment network uses at most half as much power as the two-layer network. In practice, we expect the reduction in power usage to be even more pronounced, since joining LIF compartments eliminates the overhead for digital spike-based communication, which is a substantial portion of the power budget of neuromorphic systems [7].

With some manual tuning, a network of three-compartment neurons can competitively (compared to a two-layer network) compute four-quadrant multiplication, a continuous version of the XOR problem that cannot be solved by two-compartment neurons. However, in general, we observed larger function approximation errors for three- and four-compartment neurons compared to two-compartment neurons. While it may seem as if these neurons are thus less ‘powerful’, this is not the case. Smaller static decoding errors for these functions (i.e., the same experiment as in section 3.5, but for the specific I; data not shown) indicate that the increase in error stems from the somatic current model $H$ not predicting the neuron precisely enough (cf section 3.4). A more charitable interpretation is thus that three- and four-compartment neurons reach errors similar to the two-compartment neuron for most functions, despite the surrogate model $H$ being inaccurate. It should thus be possible to reach smaller function approximation errors if we were able to improve $H$.

To conclude, the methods presented here extend the NEF, an already widely used set of methods for programming neuromorphics [17], towards harnessing nonlinear synapses and dendrites. Our results offer guidance when designing new mixed-signal neuromorphic systems that can be programed using the NEF. Being able to combine up to three neuron compartments provides purely analogue computation as a resource for high-level computation, while, at the same time, equipping researchers modelling biological systems with the ability to better map real neural circuits onto the hardware. Researchers interested in simulating $n$-LIF neurons may find our open source library libnlif useful, which provides efficient simulation and weight optimisation routines.

Our observations are also interesting from a biological perspective. The fact that increasing the number of compartments beyond three does not substantially increase the computational power of these neurons—at least when using biologically plausible reversal potentials—suggests that passive dendrites alone can—under biological constraints—at most compute functions such as XOR. This may explain the prevalence of active dendrites, which have been hypothesised to be more powerful [4, 9, 10], in many brain regions.

Future work may include further exploring the power of larger $n$-LIF neurons, and improving our models of $H$ to better exploit the computational power of more compartments. As well, there may be some further potential when leaving biologically plausible parameter ranges behind (i.e., providing symmetric excitatory and inhibitory reversal potentials). Finally, it would be interesting to validate our results on real neuromorphic hardware that has the ability to combine multiple neuron circuits, such as BrainScales 2 [27].

Acknowledgments

This work was supported by the Canada Research Chairs programme, NSERC Discovery Grant 261453, and AFOSR Grant FA9550-17-1-002.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://osf.io/y64xu/.

Appendix A. Assembling the model matrices

As we outlined in equation (4), the subthreshold dynamics of an $n$-LIF neuron are fully described by a handful of matrices. The goal of this appendix is to outline how these matrices are constructed. We then describe how to transform these matrices into the reduced system matrices, which are used to compute the equilibrium potential $\tilde{\psi}$ and, indirectly, to model the somatic current $H(g)$. Furthermore, we provide some rules for conditioning the reduced system matrices, which is important before calibrating the model parameters or solving for synaptic weights.

A.1. Constructing the $n$-LIF system matrices

Recall that the subthreshold dynamics of the $n$-LIF neuron are given as
degree matrix and the adjacency matrix, and, in our case, is given as

coupling conductance between the compartment, all conductance-based channels are replaced by current-based channels weighted by the differ-

e between

A.2. Constructing the reduced system matrices

To obtain \( \tilde{L} \) and \( \tilde{A} \) from \( L \) and \( A \), we set the first column and row of \( L \) to zero and replace connections to the somatic compartment with static conductance-based channels. In the somatic compartment, all conductance-based channels are replaced by current-based channels weighted by the difference between \( \bar{\nu} \) and the channel reversal potential (cf [34]); furthermore, we let \( (\tilde{a})_1 = 1 \) and add \( \bar{\nu} \) to \( (\tilde{b})_1 \).

We provide examples of reduced systems in table A2.

A.3. Conditioning the n-LIF system

The reduced system described above is conceptually useful, but notoriously ill-conditioned. In fact, our weight and parameter optimisation methods do not work properly without preconditioning the reduced system. We suggest the following series of transformations as a preconditioning step. Apart from the final scaling step, this transformation does not alter the current predicted by \( H \) in any way.

### Table A1. Matrix representations of the first four neuron models in figure 4. See text for details.

| Model | \( a' \) | \( A' \) | \( b' \) | \( B' \) | \( L \) |
|-------|-------|-------|-------|-------|-------|
| (A)   | \( g_L \) | \([0 \ 0]\) | \(g_L E_L\) | \([1 \ -1]\) | \([0\ 0]\) |
| (B)   | \( g_L \) | \([1 \ 1]\) | \(g_L E_L\) | \([E_L \ E_L]\) | \([0\ 0]\) |
| (C)   | \( g_L \) | \([0 \ 0]\) | \(g_L E_L\) | \([0 \ 0]\) | \([-c_{12} \ -c_{12}\ 0\ -c_{12} \ -c_{12}\ -c_{12}\ 0]\) |
| (D)   | \( g_L \) | \([1 \ 0 \ 0]\) | \(g_L E_L\) | \([0 \ 0 \ 0\ -c_{12} \ -c_{12} \ -c_{12}\ 0]\) | \([-c_{12} \ -c_{12}\ -c_{12}\ -c_{12}\ -c_{12}\ -c_{12}\ -c_{12}\ 0]\) |

### Table A2. Reduced matrix representations of the first four multi-compartment neuron models in figure 4. The somatic compartment is disconnected from the remaining neuron model. Connections to the somatic compartment are replaced by a static conductance-based channel with reversal potential \( \bar{\nu} \). The voltage difference between \( \bar{\nu} \) and the equilibrium potential of the new model is proportional to the current flowing into the somatic compartment.

| Model | \( \tilde{a}' \) | \( \tilde{A}' \) | \( \tilde{b}' \) | \( \tilde{B}' \) | \( \tilde{L} \) | \( \tilde{\nu} \) |
|-------|-------|-------|-------|-------|-------|-------|
| (A)   | \([1 \ 0 \ 0]\) | \([g_L(E_L - \bar{\nu}) + \bar{\nu}]\) | \([1 \ -1]\) | \([0\ 0]\) | \([1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]\) |
| (B)   | \([1 \ 0 \ 0]\) | \([g_L(E_L - \bar{\nu}) + \bar{\nu}]\) | \([E_L - \bar{\nu} \ E_L - \bar{\nu}]\) | \([0\ 0]\) | \([1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]\) |
| (C)   | \([1 \ g_L + c_{12}]\) | \([0 \ 0 \ 0\ g_L(E_L - \bar{\nu}) + \bar{\nu}]\) | \([0 \ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]\) | \([1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]\) |
| (D)   | \([1 \ g_L + c_{12}]\) | \([0 \ 0 \ 0\ g_L(E_L - \bar{\nu}) + \bar{\nu}]\) | \([E_L \ E_L \ E_L \ E_L]\) | \([0 \ 0 \ 0\ 0\ 0\ 0\ 0\ 0\ 0]\) | \([1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]\) |

\[
\frac{d}{dt} C_m \circ \mathbf{v}(t) = \mathbf{A}[g(t)] \mathbf{v}(t) + \mathbf{b}[g(t)] = -\left[\tilde{L} + \text{diag} (\tilde{a}' + \tilde{A}'g(t))\right] \mathbf{v}(t) + [\tilde{b}' + \tilde{B}'g(t)].
\]
Step 1. Offset all voltages such that $\bar{v} = 0$. This is accomplished by updating $\bar{b}'$ and $\bar{A}'$:

$$\bar{b}' \leftarrow \bar{b}' - \left( \bar{L}' + \text{diag}(\bar{a}') \right) \bar{v}, \quad \bar{A}' \leftarrow \bar{A}' - \bar{A}' \bar{v}. $$

This rule follows from applying an offset $o$ to the equilibrium state of the n-LIF system:

$$-\bar{A}[g]^{-1}\bar{b}[g] + o = -\bar{A}[g]^{-1} \left( \bar{b}[g] - \bar{A}[g] o \right).$$

Expanding $\bar{A}[g]o$ with $o = -\bar{v}$ yields the above equations.

Step 2. Scale all voltages such that $\tilde{c}_i \in \{1, 0\}$ or $\tilde{a}_i = 1$. Together with $\bar{v} = 0$, this implies that the equilibrium state of each compartment connected to the soma directly expresses the current flowing into the soma. To this end, we first assemble a positive vector of scaling factors $\alpha \in \mathbb{R}^n$ and update $\bar{L}', \bar{a}', \bar{A}', \tilde{\alpha}$ as follows:

$$\alpha_i = \begin{cases} 
\tilde{c}_i & \text{if } \tilde{c}_i > 0, \\
\tilde{a}_i & \text{if } \tilde{a}_i > 0 \text{ and } \tilde{c}_i = 0, \\
1 & \text{otherwise},
\end{cases}$$

This update rule follows from scaling the equilibrium state of the n-LIF system:

$$-\text{diag}(\alpha)\tilde{A} \tilde{\bar{b}}[g] = -\left( \tilde{A} \tilde{\bar{g}} \right)^{-1}\tilde{b}[g].$$

Again, expanding $\tilde{A} \tilde{\bar{g}}$ yields the first three update equations. Scaling $\tilde{\alpha}$ is necessary to preserve the correct output current. In general, if $\bar{v} \neq 0$, then $\bar{v}$ must be multiplied by $\text{diag}(\alpha)$.

Step 3. Scale the system input and output. Inputs to the n-LIF system are typically in the microsiemens ($\mu$S) or nanoamperes (nA) range, while outputs are usually single-digit nanoamperes values. We hence suggest scaling all inputs by $\alpha_{in} = 10^6$ (before passing them into the system) and all outputs by $\alpha_{out} = 10^9$. Given that $\bar{v} = 0$, we can adapt the system as follows:

$$\bar{A}' \leftarrow \frac{1}{\alpha_{in}} \bar{A}', \quad \bar{b}' \leftarrow \alpha_{out} \bar{b}', \quad \bar{B}' \leftarrow \frac{1}{\alpha_{in}} \bar{B}'$$.

A.4. Impact of conditioning in practice

Consider the three-compartment neuron depicted in figure 4(d). Using the parameters in table D1 and coupling conductances $c_{12} = 50$ nS, $c_{23} = 200$ nS, we get the following system:

$$\tilde{a}' = \begin{bmatrix} 1 \\ 100 \times 10^{-9} \\ 50 \times 10^{-9} \end{bmatrix}, \quad \tilde{\alpha}' = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad \tilde{b}' = \begin{bmatrix} -57.5 \times 10^{-3} \\ -6.125 \times 10^{-9} \\ -3.25 \times 10^{-9} \end{bmatrix}, \quad \tilde{c}' = \begin{bmatrix} 1 \\ 50 \times 10^{-9} \\ 0 \end{bmatrix}.$$

$$\tilde{B}' = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 20 \times 10^{-3} \end{bmatrix}, \quad \tilde{L}' = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 200 \times 10^{-9} & -200 \times 10^{-9} \end{bmatrix}.$$

The condition number $\kappa$ of $\tilde{A}[g] = \tilde{L}$ is approximately $\kappa \approx 10^7$. As pointed out by Cheney et al [35, section 8.4, p. 406] this implies that we lose about seven digits of precision when computing $\tilde{A}[g]^{-1}\tilde{b}$. In contrast, the conditioned system has a condition number of $\kappa \approx 10^3$.

$$\tilde{a}' = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, \quad \tilde{\alpha}' = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 20 & 20 \\ 20 & 20 & 0 \end{bmatrix}, \quad \tilde{b}' = \begin{bmatrix} 0 \\ -0.375 \\ -0.375 \end{bmatrix}, \quad \tilde{c}' = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

$$\tilde{B}' = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 57.5 \\ 57.5 & -17.5 & 0 \end{bmatrix}, \quad \tilde{L}' = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & -4 \\ 0 & -4 & 4 \end{bmatrix}.$$

Appendix B. Mathematical derivation of the n-LIF properties

In section 3.3, we briefly described rules that govern the interaction of n-LIF input channels. To summarise, there is no nonlinear interaction between current-based inputs, inputs targeting the somatic compartment,
or between inputs targeting different ‘branches’ of the neuron. Furthermore, nonlinear interaction between
inputs targeting the same compartment is limited to shunting. Conductance-based channels between different
compartments of the same ‘branch’ of a neuron interact multiplicatively. We phrase this more formally in the
following theorem.

**Theorem 1.** Consider an \( n \)-LIF neuron with \( \ell \) branches, where each compartment is connected to \( k \) unique conductance- and \( k \) unique current-based input channels. We denote these inputs as \( g_i^s \) and \( j^s_i \), where \( i \) and \( j \) are the compartment and input channel indices, respectively. Furthermore, arrange the compartment indices such that \( i_{m-1} + 1, \ldots, i_m \) belong to the same branch, where \( m \) with \( 1 \leq m \leq \ell \) is the branch index. Then, the somatic current model \( H \) has the following form

\[
H_0(g_1^1, \ldots, g_k^1, j_1^1, \ldots, j_k^1) + \frac{H_1^s(g_1^s, \ldots, g_k^s, j_1^s, \ldots, j_k^s)}{H_1^s(g_1^s, \ldots, g_k^s)} + \cdots + \frac{H_m^s(g_{\ell-1}^{s+1}, \ldots, g_{\ell}^{s+1}, j_1^{s+1}, \ldots, j_k^{s+1})}{H_m^s(g_{\ell-1}^{s+1}, \ldots, g_{\ell}^{s+1})},
\]

where (A) \( H_0 \) is an affine function over all inputs injected into the somatic compartment, (B) the \( H_m^s \) are non-negative affine functions of product terms between conductance-based inputs belonging to different dendritic compartments, and (C) the \( H_m^s \) are affine functions of product terms between conductance-based inputs belonging to different dendritic compartments, and at most one dendritic current-based input per product term.

**Proof.** We first prove claim A, followed by both claims B and C. However, to do so, we need to lay out the problem at hand in more detail. First, note that all \( \ell \) branches of the neuron, including the soma itself, form block matrices \( \tilde{C}_m[g] \) within the reduced system matrix \( \tilde{A}[g] \). In other words, the reduced system matrix and its inverse are given as

\[
\tilde{A}[g] = \begin{bmatrix}
1 & 0 & \cdots & 0 \\
0 & \tilde{C}_1[g] & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & \tilde{C}_\ell[g]
\end{bmatrix} \quad \Leftrightarrow \quad \tilde{A}[g]^{-1} = \begin{bmatrix}
1 & 0 & \cdots & 0 \\
0 & \tilde{C}_1[g]^{-1} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & \tilde{C}_\ell[g]^{-1}
\end{bmatrix}, \tag{B.1}
\]

To simplify writing out the block matrix \( \tilde{C}_m[g] \), let \( r = i_{m-1} + 1 \) be the index of the first compartment in each branch, and \( s = i_m \) be the index of the last compartment in each branch. Each block matrix is of the form

\[
\tilde{C}_m[g] = \begin{bmatrix}
d_r[g] & -c_{r,r+1} & \cdots & -c_{r,s} \\
-c_{r+1,r} & \ddots & \ddots & \ddots \\
\vdots & \ddots & \ddots & \ddots \\
-c_{s,r} & \cdots & -c_{s-1,r} & d_s[g]
\end{bmatrix},
\]

where, by construction of \( \tilde{A}[g] \), each diagonal element \( d_i[g] \) is an affine function in the conductance-based input channels. At the same time, each entry in the input vector \( \tilde{b}_i[g] \) is an affine function in both the conductance- and current-based input channels targeting this channel. We have

\[
d_i[g] = \tilde{a}_i^0 + \sum_{j=1}^k \tilde{a}_i^j g_j^s, \quad (\tilde{b}_i[g])_i = \tilde{b}_i^0 + \sum_{j=1}^k \tilde{b}_i^j g_j^s + \sum_{k=1}^k \tilde{c}_j^k j^s,
\]

where \( \tilde{a}_i^j, \tilde{b}_i^j, \tilde{c}_j^k \) are the corresponding system matrix entries. Finally, recall equation (5), i.e., the mapping between the input \( g \) and the predicted somatic current:

\[
H(g) \approx H_0(g_{i_1}^1, \ldots, g_{i_k}^1, j_{i_1}^1, \ldots, j_{i_k}^1) \approx \sum_{i=1}^n \tilde{c}_i(\tilde{v}_i^{eq} - \tilde{v}) \quad \text{where} \quad \tilde{v}^{eq} = -\tilde{A}[g]^{-1}\tilde{b}[g]. \tag{B.2}
\]

**Claim A.** The first claim (i.e., the structure of the affine function \( H_0 \)) trivially follows from multiplying the inverse of the first ‘somatic block’ of the reduced system matrix \( \tilde{A}[g] \) (i.e., the identity) with the first entry of the reduced input vector \( \tilde{b}[g] \). This results in first entry of \( \tilde{v}^{eq} \); further combining this with equation (B.2) yields an affine function \( H_0(g_{i_1}^1, \ldots, g_{i_k}^1, j_{i_1}^1, \ldots, j_{i_k}^1) \).

**Claims B and C.** The last two claims, i.e., the somatic current model \( H \) being a sum of rational functions with the given structural constraints, can be obtained by systematically inverting the individual block matrices \( \tilde{C}_m[g] \).
In particular, note that the sum-of-rational-functions structure directly follows from plugging \( \tilde{\mathbf{A}}[\mathbf{g}]^{-1} \) from equation (B.1) into equation (B.2). As we will become apparent in a moment, the product between the inverted block matrix \( \tilde{\mathbf{C}}_m[\mathbf{g}] \) for branch \( m \) and the corresponding section of the input vector \( \tilde{\mathbf{b}}[\mathbf{g}] \) form the individual rational functions. By construction, this rational function can only depend on the input channels targeting the branch \( m \). That is, each sum term \( m \) can only depend on the variables \( g_1, \ldots, g_s, f_1, \ldots, f_k \), where \( r, s \) are as defined above.

To see that the product between the inverted block matrix and the input vector forms a rational function, remember that the inverse of a matrix can be written as its adjoint scaled by the inverse of its determinant [e.g., 36, theorem 1.9, p. 366]:

\[
\tilde{\mathbf{C}}_m[\mathbf{g}]^{-1} = \frac{1}{\det(\tilde{\mathbf{C}}_m[\mathbf{g}])} \text{adj}(\tilde{\mathbf{C}}_m[\mathbf{g}]).
\]

Crucially, the determinant fully determines the denominator \( H^\delta_m \) of each rational function. The division by the determinant is the only division in each product between the inverted block-matrix \( \tilde{\mathbf{C}}_m[\mathbf{g}]^{-1} \) and the corresponding portion of the input vector \( \tilde{\mathbf{b}}[\mathbf{g}] \).

In addition to the properties listed above, claim B states that \( H^\delta_m \) only depends on the conductance-based inputs, and that it only contains products between inputs targeting different compartments. The first statement follows from \( \tilde{\mathbf{A}}[\mathbf{g}] \) only containing conductance-based inputs by construction. The second statement follows from the permutation expansion of the determinant [e.g., 36, section 4.1.3, p. 337], we have

\[
H^\delta_m(g_1, \ldots, g_s) = \det(\tilde{\mathbf{C}}_m[\mathbf{g}]) = \sum_{\pi \in \mathbb{P}} \text{sgn}(\pi) \prod_{i=1}^{n} \left( \tilde{\mathbf{C}}_m[\mathbf{g}] \right)_{\pi(i)+1, i}, \text{ where } \mathbb{P} = S(1, \ldots, n-1).
\]

Here, \( S(X) \) denotes the set of permutations of the set \( X \), and \( \text{sgn}(\pi) \in \{-1, 1\} \) is the ‘signum’ of the permutation \( \pi \). Importantly, each product in the determinant only contains each diagonal element \( d_i[\mathbf{g}] \) exactly once and there are no product-terms between inputs targeting the same compartment \( i \).

Finally, claim C states that the numerators \( H^\delta_m \) only contains product terms between inputs targeting different compartments, and that each product term features at most one current-based input. The second statement is a result of \( \tilde{\mathbf{A}}[\mathbf{g}] \) only depending on conductance-based inputs, and all current-based inputs only influencing \( \tilde{\mathbf{b}}[\mathbf{g}] \). Multiplications with current-based inputs are a result of the final matrix-vector product between \( \tilde{\mathbf{A}}^{-1}[\mathbf{g}] \) and the input matrix \( \tilde{\mathbf{b}}[\mathbf{g}] \) and there can be only one current-based input per product term.

The first statement follows from the structure of adjoint. The numerator \( H^\delta_m \) is determined by the matrix product between the adjoint of \( \tilde{\mathbf{C}}_m[\mathbf{g}] \) and the corresponding portion of \( \tilde{\mathbf{b}}[\mathbf{g}] \). Notably, the adjoint of a matrix \( \mathbf{A} \) is a matrix of determinants of \( \mathbf{A} \) where the \( k \)th row and the \( \ell \)th column have been deleted:

\[
\left( \text{adj}(\tilde{\mathbf{C}}_m[\mathbf{g}]) \right)_{\ell k} = \sum_{\pi \in \mathbb{P}} \text{sgn}(\pi) \prod_{i=1}^{n} \left( \tilde{\mathbf{C}}_m[\mathbf{g}] \right)_{\pi(i)+1, i}, \text{ where } \mathbb{P} = S(1, \ldots, \ell - 1, \ell + 1, n-1).
\]

Hence, the entry \( (\text{adj}(\tilde{\mathbf{C}}[\mathbf{g}]))_{\ell k} \) neither contains the term \( d_i[\mathbf{g}] \), nor \( d_i[\mathbf{g}] \), and the product-terms formed by computing the matrix-vector product between \( \text{adj}(\tilde{\mathbf{C}}_m[\mathbf{g}]) \) and \( \tilde{\mathbf{b}}[\mathbf{g}] \) contain each input channel at most once. This concludes our proof. \( \square \)

**Appendix C. Solving for \( n \)-LIF synaptic weights**

In order to integrate \( n \)-LIF neurons into spiking neural networks, we must solve for synaptic weights. That is, given a set of pre-activities \( \mathbf{a} \), and a set of target currents \( \mathbf{f} \), we would like to find weights \( \mathbf{w} \) such that the somatic current predicted by our model is close to the desired target. The same approaches outlined here can analogously be applied to the model calibration (cf section 3.4).

Just as we discussed above, we assume that each of the \( k \) input channels is given as \( \mathbf{g} = \mathbf{W} \mathbf{a} \), where \( \mathbf{a} \in \mathbb{R}^m \) is the vector of all pre-activities, and \( \mathbf{W} \in \mathbb{R}^{k \times m} \) is a nonnegative sparse connection matrix. Entries in \( \mathbf{W} \) corresponding to invalid connections between pre- and post-neurons must be forced to zero when performing weight optimisation.

Let \( \mathcal{E} \) be the superthreshold error function (equation (3)), and \( \mathbf{L}, \tilde{\mathcal{C}}, \tilde{\mathcal{A}}, \tilde{\mathbf{b}}, \tilde{\mathbf{B}} \) describe the reduced and conditioned system matrices describing the neuron. Given \( N \) samples \( (\mathbf{a}_i, \mathbf{f}_i) \), we must minimise the following loss function with respect to the non-zero entries of \( \mathbf{W} \):

\[
\text{Minimise } \frac{1}{N} \sum_{i=1}^{N} \mathcal{E}(\tilde{\mathcal{C}}_i(\mathbf{W})\mathbf{a}_i) \text{ subject to } \mathbf{W} \text{ non-negative, sparse.}
\]
\[ E = \sum_{\ell=1}^{N} \mathcal{E}\left(\langle \check{v}^n W a_r, \check{c} \rangle, J_r \right)^2 + \lambda N \|W\|^2 \]

\[ = \sum_{\ell=1}^{N} \mathcal{E}\left(\left\langle \left( \bar{L} + \text{diag}(\alpha' + \check{A}' W a_r) \right)^{-1} \left( \check{b}' + \check{B}' W a_r \right), \check{c} \right\rangle, J_r \right)^2 + \lambda N \|W\|^2. \quad (C.1) \]

We explore two different approaches to solving this problem. First, using gradient-based methods, and second, by constructing a SQP. Both algorithms are implemented in our software library libnli.

C.1. Gradient descent

To determine the loss function gradient, we use the matrix inverse differential tensor (see [37, section 10.6, equation (1), p. 198]). Patiently applying the chain and product rule, the gradient of the loss function equation (C.1) with respect to \( W \) is (without the regularisation gradient \( 2\lambda N W \)):

\[ \frac{\partial E}{\partial (W)_{rs}} = -2 \sum_{\ell=1}^{N} \mathcal{E}(H(W a_r), J_r) \left( \sum_{j=1}^{n} \sum_{i=1}^{n} \left( \check{A}[W a_r]^{-1} \right)^{ij} \left( \check{b}' \right)_{ij} (a_r)_i (\check{c})_i \right) 
+ \sum_{k=1}^{n} \left( \check{A}[W a_r]^{-1} \right)^{ik} \left( \check{A}' \right)^{kj} (a_r)_k (\check{b}' W a_r)_{ij} (\check{c})_l \right). \quad (C.2) \]

Note that the superthreshold error function \( \mathcal{E}(I_{\text{dec}}, J_{\text{tar}}) \) is a piecewise function with three separate cases; luckily, the above equation handles all three at the same time. In case \( I_{\text{tar}} < I_{\text{th}} \) and \( I_{\text{dec}} < I_{\text{th}} \), the output of \( \mathcal{E} \) is zero, and thus the gradient is zero as well. In the other two cases, \( \mathcal{E} \) is equal to \( I_{\text{dec}} \) with some offset. This does not affect the gradient.

To enforce nonnegativity of \( W \), and thus nonsingularity of \( \check{A}[W a_r] \), weights should be clipped to zero after each gradient descent step. In the case of quasi-Newton methods, a bounded algorithm such as L-BFGS-B must be used.

C.2. Sequential quadratic programme

An alternative approach is to use a SQP. Let \( \Theta \) be a vector of the non-zero weights in \( W \). Introducing \( N \) auxiliary variables \( v_\ell \in \mathbb{R}^n \) we can split the optimisation problem as follows:

\[ E_1 = \sum_{\ell=1}^{N} \mathcal{E}(v_\ell, \check{c}), J_r \]
\[ E_2 = \sum_{\ell=1}^{N} \left\| \left( \bar{L} + \text{diag}(\alpha' + \check{A}' W \Theta a_r) \right) v_\ell - \left( \check{b}' + \check{B}' W \Theta a_r \right) \right\|^2. \]

Expanding \( E_2 \), we find that there is only one product term containing both \( \Theta \) and \( v_\ell \). Linearising this product term in \( E_2 \) at \( (\Theta_0, v_1, \ldots, v_{N,0}) \) using a Taylor expansion results in the following expression:

\[ E'_2 = \sum_{\ell=1}^{N} \left\| \bar{L} v_\ell - \check{b}' - \check{B}' W \Theta a_r + \check{a}' \circ v_\ell + \check{A}' W \Theta a_r \circ v_{\ell,0} + \check{A}' W \Theta a_r \circ v_{\ell,0} - \check{A}' W \Theta a_r \circ v_{\ell,0} \right\|^2. \]

This equation is in linear least-squares form over \( \Theta \) and \( v_1, \ldots, v_N \). Combining this with a soft trust-region term \( E_3 \), as well as regularisation factors \( \lambda_1, \lambda_2 \), the final loss function \( E \) is

\[ E = \alpha_1 E_1 + \alpha_2 E'_2 + \alpha_3 E_3 + \lambda_1 N \|\Theta\|^2 + \lambda_2 \sum_{\ell=1}^{N} \|v_\ell\|^2, E_3 = N \|\Theta - \Theta_0\|^2 + \sum_{\ell=1}^{N} \|v_\ell - v_{\ell,0}\|. \quad (C.3) \]

This loss function, including a nonnegativity constraint on \( \Theta \) and subthreshold relaxation can be expressed as a quadratic programme. We obtain a SQP by minimising \( E \) and using the resulting \( \Theta \) as \( \Theta_0 \) in a subsequent iteration.

Appendix D. Neuron parameters

See table D1.
Table D1. Neuron and synaptic parameters used unless explicitly stated otherwise.

| Parameter                      | Symbol | Value       | Parameter                      | Symbol | Value       |
|--------------------------------|--------|-------------|--------------------------------|--------|-------------|
| Superthreshold dynamics        |        |             | Reversal potentials            |        |             |
| Threshold potential            | $v_{th}$ | $-50 \text{ mV}$ | Exc. reversal potential       | $E_h$  | $20 \text{ mV}$ |
| Spike potential                | $v_{spike}$ | $20 \text{ mV}$ | Resting potential             | $E_l$ | $-65 \text{ mV}$ |
| Refractory period              | $\tau_{ref}$ | $2 \text{ ms}$ | Inh. reversal potential       | $E_l$ | $-75 \text{ mV}$ |
| Spike period                   | $\tau_{spike}$ | $1 \text{ ms}$ |                                |        |             |
| Reset potential                | $v_{reset}$  | $-65 \text{ mV}$ |                               |        |             |
| Relaxation target threshold    | $j_{th}$ | $0.56 \text{ nA}$ |                               |        |             |
| Synaptic time-constants        |        |             | Conductances                   |        |             |
| Exc. synapse time-constant     | $\tau_{syn,E}$ | $5 \text{ ms}$ | Coupling conductance          | $c_{12}$ | $100 \text{nS}$ |
| Inh. synapse time-constant     | $\tau_{syn,I}$ | $10 \text{ ms}$ | Coupling conductance          | $c_{23}$ | $200 \text{nS}$ |
| Membrane capacitances          | $C_{rs}$  | $1 \text{nF}$ | Coupling conductance          | $c_{34}$ | $500 \text{nS}$ |
|                                |         |             | Leak conductances             | $d_{41}$ | $50 \text{nS}$ |

**References**

[1] Mel B W 1994 Information processing in dendritic trees *Neural Comput.* **6** 1031–85
[2] Koch C, Mo C H and Softky W 2002 Single-cell models *The Handbook of Brain Theory and Neural Networks* 2nd edn ed M A Arbib (Cambridge, MA: MIT Press) pp 1044–9
[3] Polsky A, Mel B W and Schiller J 2004 Computational subunits in thin dendrites of pyramidial cells *Nat. Neurosci.* **7** 621
[4] London M and Häusser M 2005 Dendritic computation *Annu. Rev. Neurosci.* **28** 503–32
[5] Schemmel J, Brüderle D, Gründl A, Hock M, Meier K and Müller S 2010 A wafer-scale neuromorphic hardware system for large-scale neural modeling *Proc. 2010 IEEE Int. Symp. Circuits and Systems* pp 1947–50
[6] Pfeil T et al 2013 Six networks on a universal neuromorphic computing substrate *Front. Neurosci.* **7** 11
[7] Benjamini C, Hartlieb A and Eliasmith C H 2003 Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems (Cambridge, MA: MIT Press) p 380
[8] Boahen K 2017 A neuron’s prospectus *Comput. Sci. Eng.* **19** 14–28
[9] Poirazi P, Brannon T and Mel B W 2003 Pyramidal neuron as two-layer neural network *Nat. Rev. Neurosci.* **10** 2727–2739 e3
[10] Eliasmith C and Anderson C H 2003 Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems (Cambridge, MA: MIT Press) p 380
[11] Stöckel A and Eliasmith C 2021 Passive nonlinear dendritic interactions as a computational resource in spiking neural networks *Neural Comput.* **33** 96–128
[12] Eliasmith C, Stewart T C, Choo X, Bekolay T, DeWolf T, Tang Y and Rasmussen D 2012 A large-scale model of the functioning brain *Science* **338** 1202–5
[13] Eliasmith C 2013 *How to Build a Brain: A Neural Architecture for Biological Cognition* (Oxford Series on Cognitive Models and Architectures) (Oxford: Oxford University Press) p 456
[14] Stöckel A, Stewart T C and Eliasmith C 2021 Connecting biological detail with neural computation: application to the cerebellar granule–golgi microcircuit *Top. Cogn. Sci.* **13** 515–33
[15] Corradi F, Eliasmith C and Indiveri G 2014 Mapping arbitrary mathematical functions and dynamical systems to neuromorphic VLSI circuits for spike-based neural computation *IEEE Int. Symp. Circuits and Systems (ISCAS)* (Melbourne)
[16] Voelker A R and Eliasmith C 2021 Programming neuromorphics using the neural engineering framework *Handbook of Neuroengineering* ed N V Thakor (Berlin: Springer) pp 1–43
[17] Stöckel A 2022 Harnessing neural dynamics as a computational resource *PhD Thesis* University of Waterloo, Waterloo, ON
[18] Eccles J C 1986 Chemical transmission and Dale’s principle *Progress in Brain Research* vol 149 (Amsterdam: Elsevier) pp 3–13
[19] Strata P and Harvey R 1999 Dale’s principle *Brain Res. Bull.* **50** 349–50
[20] Hornik K, Stinchcombe M and White H 1989 Multilayer feedforward networks are universal approximators *Neural Netw.* **2** 359–66
[21] Bobier B, Stewart T C and Eliasmith C 2014 A unifying mechanistic model of selective attention in spiking neurons *PLoS Comput. Biol.* **10** e1003577
[22] Salinas E and Thier P 2000 Gain modulation *Neuron* **27** 15–21
[23] Chance F S, Abbott L F and Reyes A D 2002 Gain modulation from background synaptic input *Neuron* **35** 773–82
[24] Abbott L F and Chance F S 2005 Drivers and modulators from push–pull and balanced synaptic input *Cortical Function: A View from the Thalamus* (Progress in Brain Research vol 149) (Amsterdam: Elsevier) pp 147–55
[25] Minsky M and Papert S A 1969 *Perceptrons: An Introduction to Computational Geometry* 1st edn (Cambridge, MA: MIT Press) p 308
[26] Schemmel J, Kriener L, Müller P and Meier K 2017 An accelerated analog neuromorphic hardware system emulating NMDA- and calcium-based non-linear dendrites *2017 Int. Joint Conf. Neural Networks (IJCNN)* pp 2217–26
[27] Kaiser J, Billaudelle S, Müller E, Tetzlaff C, Schemmel J and Schmitt S 2021 Emulating dendritic computing paradigms on analog neuromorphic hardware *Neuroscience* **489** 290–300
[28] Capadacy C and Van Vreeswijk C 2006 Direct control of firing rate gain by dendritic shunting inhibition *J. Integr. Neurosci.* **65** 199–222

**ORCID iDs**

Andreas Stöckel [https://orcid.org/0000-0001-5965-4541](https://orcid.org/0000-0001-5965-4541)
[30] Kiselev M, Ivanov A and Ivanov D 2020 Approximating conductance-based synapses by current-based synapses Advances in Neural Computation, Machine Learning, and Cognitive Research IV ed B Kryzhanovsky, W Dunin-Barkowski, V Redko and Y Tiumentsev (Berlin: Springer) pp 394–402

[31] Koch C 1999 Biophysics of Computation: Information Processing in Single Neurons (Oxford: Oxford University Press) p 562

[32] Kingma D P and Ba J 2015 Adam: a method for stochastic optimization 3rd Int. Conf. Learning Representations, ICLR 2015, Conf. Track Proc. (San Diego, CA, USA 7–9 May 2015) ed Y Bengio and Y LeCun

[33] Stellato B, Banjac G, Goulart P, Bemporad A and Boyd S 2020 OSQP: an operator splitting solver for quadratic programs Math. Prog. Comp. 12 637–72

[34] Stöckel A, Voelker A R and Eliasmith C 2017 Point neurons with conductance-based synapses in the neural engineering framework (arXiv:1710.07659)

[35] Cheney W and Kincaid D 2012 Numerical Mathematics and Computing 7th edn (Boston, Massachusetts, United States: Brooks/Cole) p 677

[36] Hefferon J 2020 Linear Algebra 4th edn (Ann Arbor, MI: Orthogonal Publishing) p 525

[37] Lütkepohl H 1997 Handbook of Matrices (New York: Wiley) p 320