Passive nonlinear dendritic interactions as a general computational resource in functional spiking neural networks

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

\textsuperscript{1}Centre for Theoretical Neuroscience, University of Waterloo, Waterloo, Ontario, Canada

* astoecke@uwaterloo.ca

Abstract

Nonlinear interactions in the dendritic tree play a key role in neural computation. Nevertheless, modeling frameworks aimed at the construction of large-scale, functional spiking neural networks tend to assume linear, current-based superposition of post-synaptic currents. We extend the theory underlying the Neural Engineering Framework to systematically exploit nonlinear interactions between the local membrane potential and conductance-based synaptic channels as a computational resource. In particular, we demonstrate that even a single passive distal dendritic compartment with AMPA and GABA-A synapses connected to a leaky integrate-and-fire neuron supports the computation of a wide variety of multivariate, bandlimited functions, including the Euclidean norm, controlled shunting, and non-negative multiplication. Our results demonstrate that, for certain operations, the accuracy of dendritic computation is on a par with or even surpasses the accuracy of an additional layer of neurons in the network. These findings allow modelers to construct large-scale models of neurobiological systems that closer approximate network topologies and computational resources available in biology. Our results may inform neuromorphic hardware design and could lead to a better utilization of resources on existing neuromorphic hardware platforms.

Author summary

To generate mathematical models of biologically plausible “spiking” neural networks, theoretical neuroscientists have developed several modeling frameworks. However, these frameworks make simplifying assumptions about the way information is processed in individual neurons. In particular, they tend to assume that nonlinear transformation of signals mostly happens in a neuron’s cell body. In reality, signals already interact nonlinearly while traveling through the cell’s dendrites. In this paper, we augment one of the aforementioned modeling frameworks, the Neural Engineering Framework, to take some of these interactions into account. We demonstrate that dendritic nonlinearities can be systematically exploited to perform a variety of useful computations, while at the same time relying on a smaller number of neurons and a simpler network structure. As we continue to understand dendritic computation, we can improve our frameworks for characterizing neural computation, which in turn helps us to shed light on how brains produce the complex behaviors we observe in nature.
Introduction

One central challenge in theoretical neuroscience is to characterize the biophysical mechanisms that ultimately give rise to the complex behaviors we observe in animals [1]. Conversely, behavioral models used by cognitive scientists and psychologists tend to focus on the interaction between higher-level systems that are ultimately implemented in a neural substrate. These “top-down” models risk neglecting mechanistic constraints, such as neural connectivity, signal-to-noise ratios, and dynamics. To bridge the gap between biophysical mechanisms and high-level systems, theoretical neuroscientists have developed methods facilitating the description of behavioral models that satisfy mechanistic constraints of interest to a researcher. These methods include the Neural Engineering Framework (NEF) [2,3], Efficient, Balanced Spiking Networks (EBN) [4,5], and FORCE [6,7].

Generally speaking, these approaches describe how to translate dynamical systems—corresponding to some hypothesized brain function—into an idealized spiking neural network that adheres to the desired constraints [7,8]. This mechanistic grounding facilitates model validation by enabling a direct comparison of simulation results and empirical data [9,10].

The methods listed above primarily rely on two biophysical phenomena as computational resources: synaptic filtering and the nonlinear relationship between somatic input currents and the neural response, including spike generation and variations in membrane potential. Response models range from leaky integrate-and-fire (LIF) dynamics to the complex behaviors produced by Hodgkin-Huxley type conductance-based channels [11–13]. Crucially however, the approaches we have highlighted typically assume that the somatic input current is a linear superposition of filtered pre-synaptic events. While some research exists that explores nonlinear dendritic effects tailored towards specific tasks [14,15], nonlinearities are not systematically exploited to perform arbitrary computation. Yet, empirical and theoretical work suggests that active and passive nonlinear effects within the dendritic tree—and not only the soma—are at least partially responsible for the complex responses observed in some biological neurons, including cortical pyramidal cells [16–18]. In particular, London and Häusser [18] argue that in addition to voltage-gated ionic currents, fundamental passive dendritic effects such as shunting inhibition are worth being investigated as a computational resource.

Put differently, the above modeling frameworks only consider a subset of the computational resources available in individual neurons and thus underestimate their computational power [19]. This potentially forces modelers to construct networks that—connectivity-wise—are considerably more complex than what is actually required in biology to elicit a certain behavior. The goal of this paper is to alleviate this limitation by describing a mathematically tractable model of dendritic computation that is based on a central feature of neuronal biophysics—conductance-based synaptic channels. In doing so, and as demanded by London and Häusser [18], we demonstrate that even passive dendritic effects within a two-compartment neuron model provide significant computational advantages, both theoretically and in the context of a noisy spiking neural network.

From a machine learning perspective, the importance of dendritic computation can be highlighted by considering the XOR problem (Fig 1A to 1E), or equivalently, for continuous codomains, multiplication (Fig 1F). The XOR problem is defined as follows: find parameters for a two-dimensional classifier such that the points (0,0), (1,1) and (0,1), (1,0) are separated into two distinct classes. A classical result from machine learning theory is that such parameters do not exist for a Perceptron, a single-layer neural network [20]. For two input variables the Perceptron is defined as \( \sigma(u_x + v_y + b) \), where \( \sigma \) is a monotonic threshold function and \( u, v, b \) are the model...
parameters. Perhaps surprisingly, even when generalizing this expression to \( \sigma(f(x) + g(y)) \), where \( f, g \) are arbitrary univariate functions, it is not possible to solve the XOR problem (Fig 1A to 1C; see appendix S1 for a proof).

Fig 1. Visualization of the XOR decision problem for different types of classifiers. Markers correspond to the four data points that should be classified. The colored/hatched background corresponds to the output of one exemplary decision function. (A) The linear decision boundary of a single-layer Perceptron cannot solve the decision problem. (B, C) This still holds for the generalization \( \sigma(f(x) + g(y)) \). (D) A multi-layer Perceptron (MLP) of the form \( \sigma(\sum w_i \sigma(u_i x + v_i y + b_i)) \) can be optimized using gradient descent to solve the problem correctly. (E) An alternative solution using a non-monotonic nonlinearity \( \sigma'(\xi) = \sigma(\xi^2 - 1) \). (F) Multiplication of two real-valued variables \( x, y \) can be seen as a superset of the XOR problem.

One solution to this problem—corresponding to a more complex network topology—is to add a hidden layer to the network. This results in intermediate variables that code information about both \( x \) and \( y \), making the classification task linearly separable (Fig 1D). Crucially, a Multi Layer Perceptron (MLP) is a Universal Approximator. By adjusting the number of neurons in the hidden layer, the MLP can approximate any continuous function to an arbitrarily low margin of error [2].

Returning to our consideration of dendritic computation, we can notice that instead of using a fully parametrized intermediate nonlinearity, such as the hidden neuron layer, it is possible to exchange \( \sigma \) with a fixed non-monotonic nonlinearity \( \sigma' \). Specifically, \( \sigma' \) can be seen as implemented by dendritic computation. For example, there exists a parametrization for the nonlinearity \( \sigma'(\xi) = \sigma(\xi^2 - 1) \) that solves the XOR problem (Fig 1E). This is because the polynomial \((x + y)^2 = x^2 + 2xy + y^2\) encodes information about the product of \( x \) and \( y \). Yet, and in contrast to Universal Approximators, a fixed nonlinearity can only improve the accuracy for certain classes of functions. Nevertheless, as we demonstrate in our experiments below, even a moderately complex nonlinearity lowers the approximation error for many useful functions.
In this paper, we focus on extending the NEF to take a first step towards systematically exploiting nonlinear dendritic interactions as a computational resource. We lift one of the long-standing central assumptions of the NEF mentioned above, namely linear superposition of post-synaptic currents. We show that augmenting a leaky integrate-and-fire (LIF) neuron by adding a second, passive compartment featuring conductance-based excitatory and inhibitory input channels—as proposed by [22]—significantly increases its computational power. To determine all synaptic weights, we describe a layer-wise least-squares objective phrased in terms of a convex quadratic program that guarantees finding a globally optimal solution.

Materials and Methods

In this section, we review relevant portions of the Neural Engineering Framework (NEF)—namely the representation and transformation principles—followed by four extensions that improve its biological plausibility and ultimately enable us to take nonlinear dendritic effects into account. These extensions concern decoding in current space, nonnegative weights, equality relaxation for subthreshold currents, and the introduction of a dendritic nonlinearity model $H$, enabling us to systematically harness nonlinear dendritic effects for computation. We close this section with a description of the two-compartment LIF neuron model used in our experiments and a derivation of a suitable surrogate model for the dendritic nonlinearity $H$.

The Neural Engineering Framework (NEF)

As outlined above, the NEF is a method for the construction of recurrent spiking neural networks from high-level mathematical descriptions. The NEF has been applied to various research areas, including low-level modeling of neurobiological systems [14,23,24], and studying large-scale models of cognitive systems grounded in biology [3,12,25]. Furthermore, a software implementation of the NEF is part of the neural simulation package Nengo [26] and has been used as a neural compiler targeting analog and digital neuromorphic hardware [27,31].

At its core, the NEF describes three principles that govern the construction of neural networks. The first principle postulates that populations of spiking neurons represent values in a relatively low-dimensional manifold in their high-dimensional activity space via nonlinear encoding and linear decoding. According to the second principle, network connectivity defines transformations as mathematical functions of the represented values. Finally, the third principle (not covered here) states that represented values are control-theoretic variables and recurrent connections approximate arbitrary dynamical systems of these variables.

Representation

A fundamental assumption of the NEF is that populations of spiking neurons represent $d$-dimensional vectors $\vec{x} \in \mathbb{R}^d$ by means of nonlinear encoding and linear decoding. Encoding refers to the process of translating $\vec{x}$ to somatic currents $J$ that cause neurons in a population to spike, generating neural activity $\vec{a}(\vec{x}) \in \mathbb{R}^n$, where $n \gg d$. The inverse operation, decoding, estimates the represented $\vec{x}$ by linearly projecting from the high-dimensional population activity onto the lower-dimensional representation.

For the encoding process, each neuron $i$ in a population of size $n$ is assigned a tuning-curve that maps any represented value $\vec{x}$ onto a corresponding activity
The Neural Engineering Framework (NEF) is a computational framework for modeling the brain's information processing. It uses a combination of encoding and decoding operations to represent and transform information in the brain. The framework is based on the idea that the brain encodes information in the activity of neural populations, and that this information can be decoded to retrieve the original representation.

Encoding operations are used to map an input value to the activity of a neural population. This is done through the use of current translation functions, which are based on a first-order polynomial parametrized by gains and biases. The specific current translation function depends on modeling assumptions, but it is often defined as a first-order polynomial.

Decoding operations are used to reconstruct the represented value from the momentary population activity. This is done through the use of decoding matrices, which are determined through optimization. The optimization problem is solved using a Tikhonov regularized least-squares method.

The NEF is used to model a variety of neural processes, including sensory processing, motor control, and memory. It has been shown to be successful in simulating the behavior of real neural systems, and it is a useful tool for understanding the functioning of the brain.
\[ \min_D \sum_{k=1}^{N} \| D \vec{a}(\vec{x}_k) - \vec{x}_k \|_2^2 + \lambda \| D \|_F^2 = \min_D \| DA - X \|_F^2 + \lambda \| D \|_F^2, \quad (3) \]

where, \( \vec{x}_k \) is one of \( N \) training samples, \( A \in \mathbb{R}^{n \times N} \) is a matrix of population responses for each sample, \( X \in \mathbb{R}^{d \times N} \) is a matrix of training samples, and \( \lambda \) is a regularization term accounting for spike variability and other sources of error \[2\]. Applying the Moore-Penrose pseudo-inverse \[33\] yields

\[ D = X A^T (A A^T + \lambda I)^{-1}. \quad (4) \]

The decoders \( D \) are usually optimized offline under the assumption of the rate model defined in eq. \((2)\). As formulated, the same \( D \) can be used to decode represented values through time in spiking networks when defining activity \( \vec{a}(t) \) as low-pass filtered population spike trains. Linear low-pass filters are a common model for post-synaptic currents \[34\] and usually employed in spiking NEF networks \[2\]. Indeed, as mentioned, the above optimization can be done using activity matrices \( A \) defined by filtered simulated spiking activity, although at greater computational expense.

**Transformation**

Nonlinear encoding and linear decoding schemes related to the one described above have been used for a long time in the field of machine learning \[35\]. Neuron population tuning-curves \( \vec{a}(\vec{x}) \) span a function space with a set of non-orthogonal basis functions. If some relatively weak conditions are met, we can universally approximate any continuous function over the represented values by linearly combining these basis functions \[21\]. In the specific case of eq. \((3)\), the linear projection \( D \) approximates the identity function. By modifying the loss function and substituting \( X \) with a matrix \( X^f \) of target vectors \( f(\vec{x}_k) \) we can solve for decoders \( D^f \) that approximate arbitrary functions \( f \) (Fig 2D)

\[ \min_{D^f} \sum_{k=1}^{N} \frac{1}{2} \| D^f \vec{a}(\vec{x}_k) - f(\vec{x}_k) \|_2^2 + \lambda \| D^f \|_F^2 \Rightarrow D^f = X^f A^T (A A^T + \lambda I)^{-1}. \quad (5) \]

In order to construct neural networks, we need to find synaptic weight matrices \( W \in \mathbb{R}^{m \times n} \) that connect from a pre-population of \( n \) neurons to a post-population of \( m \) neurons. In particular, we would like to find a \( W \) that implicitly decodes \( f(\vec{x}) = \vec{y} \) from the pre-population and at the same time encodes \( \vec{y} \) in the post-population. If we assume that the current translation function \( J_i(x) \) is an intrinsic part of the neuron model, or, put differently, each neuron \( i \) is assigned its own response-curve \( G_i[\langle \vec{e}_i, \vec{x} \rangle] = G[\langle \vec{e}_i, J_i(\langle \vec{e}_i, \vec{x} \rangle) \rangle] \), both the encoding and decoding process are linear operators. With this simplification, \( W = E D^f \) fulfils the properties listed above, where \( E \in \mathbb{R}^{m \times d} \) is a matrix of post-population encoding vectors \( \vec{e}_i \), and \( D^f \) the desired function decoder for the pre-population \[2\].

This factorization of \( W \) into encoders and decoders has two consequences. First, from a technical perspective, and under the assumption that the represented dimensionality is significantly smaller than the number of neurons, factorizing the low-rank weight matrix facilitates efficient implementation of the corresponding spiking neural network in both hardware and software \[26, 29\]. Second, from a modeler’s perspective, networks constructed in this manner are additive (Fig 3A). Summing signals on the level of neural activity results in the represented values being summed as well. To see this, consider two pre-populations projecting onto a common
Fig 3. Multivariate computation in the NEF. (A) In case the current translation function $J_i$ is a part of the individual neuron response curve, NEF networks are additive: summation in activity space corresponds to addition in representational space. (B) Computing nonlinear multivariate functions $\varphi$ generally requires all variables to be represented in an intermediate population. (C) The dendritic computation scheme discussed in this paper. Two pre-populations project onto a post-population with separate excitatory and inhibitory input channels. The nonlinear interaction between these channels is exploited to compute $\varphi$.

post-population, where the first projection approximates a function $f_1$, and the second a function $f_2$. Let $i$ be the index of a neuron in the post-population. Then, it holds

$$G_i[\langle \vec{x}, a_{\text{pre}}(\vec{x}) \rangle + \langle \vec{y}, a_{\text{pre}}(\vec{y}) \rangle] = G_i[Df_1(\vec{x}) + Df_2(\vec{y})].$$

(6)

Crucially, when combining the input from multiple pre-populations, the post-population will always represent a linear combination of potentially nonlinear functions. As a consequence, and as discussed in the introduction, if we, for example, try to multiply two scalars $x$ and $y$, these values must be represented as a two-dimensional vector in a common pre-population (Fig 2C). In other words, when considering $(x, y) \in [-1, 1]^2$, there exists no real-valued function pair $f$, $g$ such that $xy = f(x) + g(y)$ (see Appendix S1).

Extending the Neural Engineering Framework

The NEF as described above makes several assumptions that are not biophysically plausible. This includes the presence of a bias current $\beta_i$ in each neuron, and excitatory and inhibitory connections not being treated separately.

In this section we first describe two extensions that lift the aforementioned assumptions. We then present an alternative synaptic weight solving procedure that takes subthreshold currents into account, followed by our proposed method for accounting for dendritic nonlinearities.

Decoding the current translation function

In the previous subsection we assumed that the current translation function $J_i(x)$ is an intrinsic part of the neuron model. Consequently, each neuron is not only assigned a neuron-specific tuning-curve $a_i(\vec{x})$, but also a neuron-specific response-curve $G_i[J]$. This comes at the cost of biological plausibility, since neurons in general do not possess a strong bias current source. Furthermore, mixing current-sources with conductance-based input channels is problematic, since negative currents interfere with conductance-based input channels by driving the membrane potential to extremely low values, which induces large currents in the conductance-based channels.
Trip and Eliasmith [36] demonstrate that it is possible to robustly solve for synaptic weight matrices that approximate arbitrary post-synaptic current functions. Correspondingly, instead of performing optimization directly with respect to represented values, we can solve for weights in terms of somatic currents and include the current translation in the target function. Since we assume current-based synapses, the post-synaptic current is linear in the pre-population activity. For each post-neuron \( i \) we must find a weight vector \( \vec{w}_i \) such that the following loss function is minimized

\[
\min_{\vec{w}_i} \sum_{k=1}^{N} \left( J_i(\langle \vec{e}_i, f(\vec{x}_k) \rangle) - \langle \vec{w}_i, \vec{a}_{pre}(\vec{x}_k) \rangle \right)^2 + \lambda \| \vec{w}_i \|^2_2 \Leftrightarrow \vec{w}_i = \vec{\alpha}_i A^T (A A^T + \lambda I)^{-1}.
\]

Interestingly, due to linearity, we can separate the weight matrix \( W \) resulting from the optimization problem in eq. (7) into the original factorized weight matrix \( W^f = D^f E \), cf. eq. (5), as well as a weight matrix \( W^{\text{bias}} \) responsible for the individual bias terms

\[
W = W^f + W^{\text{bias}} = \text{diag}(\vec{\alpha}) E D^f + \text{diag}(\vec{\alpha}) E D^{\text{bias}}.
\]

This implies, first, that we can solve for the weights decoding the bias term independently of the weights decoding the target function, and second, that networks built in this way are no longer additive (see above). Additivity can be restored by distributing the bias among all pre-populations ahead of time.

Nonnegative weights and Dale’s principle

So far we have assumed that synaptic weights are real-valued. This is problematic for two reasons. First, the least-squares optimization proposed above arbitrarily assigns algebraic signs to the synaptic weights; we cannot specify which connections are excitatory, and which inhibitory. Being able to do so is important, since cells in central nervous systems tend to follow Dale’s principle—neurons exclusively effect post-neurons in an excitatory or inhibitory manner [37]. Empirical data suggest that depending on the modeled brain region excitatory cells outnumber inhibitory cells by a factor between two and four [38, 39]. The optimized synaptic weights should reflect those numbers. Second, real-valued weights do not generalise to conductance-based synapses. The concept of negative conductances—in contrast to negative currents—is neither physically nor biologically plausible. Biological correlates of synaptic weights, such as the number of vesicles released from the pre-synapse or the channel density in the post-synapse, are inherently nonnegative quantities [34].

An extension to the NEF that takes these biological constraints into account has been proposed by Parisien [40]. The Parisien transform splits each projection into an excitatory and inhibitory path, where the latter mediates the signal from the excitatory pre-population through a population of inhibitory interneurons.

The solution we discuss here does not introduce interneurons, and as such does not affect the structure of the network. However, since inhibitory signaling in cortex predominantly relies on interneurons [41, 42], our method alone produces less biologically plausible network topologies. Modelers using the methods we describe below have to explicitly define interneuron populations according to the underlying biological system.

As the problem of choosing the correct network topology is orthogonal to dendritic computation, we assume in this paper that each population is arbitrarily split into a group of excitatory and inhibitory neurons. We write the somatic input current of post-neuron \( i \) in response to pre-synaptic activity as \( \langle \vec{a}^+_i, \vec{a}^+ \rangle - \langle \vec{a}^-_i, \vec{a}^- \rangle \), where, \( \vec{a}^\pm_i \) are the nonnegative excitatory and inhibitory weight vectors and \( \vec{a}^\pm \) the activities of
the excitatory and inhibitory neurons in the pre-population. Combining this current term with eq. (7) yields an optimization problem that allows us to solve for weights approximating \( f \) for each individual post-neuron \( i \)

\[
\min_{\vec{w}_i^+, \vec{w}_i^-} \frac{1}{2} \sum_{k=1}^{N} \left\| (\vec{w}_i^+, \vec{a}_k) - (\vec{w}_i^-, \vec{a}_k^-) - J \left( \langle \vec{e}_i, f(\vec{x}_k) \rangle \right) \right\|^2 + \lambda \left\| \vec{w}_i^+ \right\|^2 + \lambda \left\| \vec{w}_i^- \right\|^2
\]

\[
= \frac{1}{2} \left\| \vec{w}_i^- A' - J \right\|^2 + \lambda \left\| \vec{w}_i^- \right\|^2 \text{ where } \vec{w}_i^+ = (\vec{w}_i^+, \vec{w}_i^-), A' = (A^+, -A^-)^T, \text{ and } \vec{j}_k = J \left( \langle \vec{e}_i, f(\vec{x}_k) \rangle \right),
\]

subject to \( \vec{w}_i^+ \geq 0, \vec{w}_i^- \geq 0 \).

This problem is in canonical least-squares form and can be solved with a standard Tikhonov regularized nonnegative least-squares solver. Alternatively, eq. (8) can be phrased as a convex quadratic program (QP), a generalization of least-squares optimization [43]

\[
\min_{\vec{w}_i^+} \frac{1}{2} (\vec{w}_i^+)^T (A')^T A' \vec{w}_i^+ - (A')^T A' \vec{w}_i^+ + \lambda \left\| \vec{w}_i^+ \right\|^2, \quad \text{subject to } \vec{w}_i^+ \geq 0. \tag{9}
\]

The global minimum of a convex QP can be computed in polynomial time [14]. We propose a QP similar to eq. (9) to solve for conductance-based synaptic weights in the context of the two-compartment neuron model discussed below.

Equality relaxation for subthreshold currents

Injecting near-zero or negative currents into a neuron should not result in any output spikes—there exists a threshold current \( J_{\text{th}} \) below which the neuron does not spike. However, we do not take this into account when solving for somatic currents in eqs. (7) to (9). Instead, we optimize for synaptic weights that precisely evoke certain post-synaptic currents, despite the magnitude of currents smaller than \( J_{\text{th}} \) having no effect on the neural activity in a steady-state neural response model. Instead of optimizing for equality, i.e. \( J_{\text{dec}} = J_{\text{tar}} \), we could relax this condition to an inequality constraint \( J_{\text{dec}} \leq J_{\text{tar}} \) whenever \( J_{\text{tar}} < J_{\text{th}} \).

This is beneficial for two reasons. First, we ultimately care about momentary spike activity and not currents. Solely optimizing in relevant superthreshold regions of the current function leads to a smaller error with respect to the represented value of an ensemble. Second, neurons with narrow tuning curves may possess current translation functions that map onto extreme negative currents. Due to regularization, it is unlikely that decoders optimized with a relaxed loss function will inject large negative currents into the post-neurons, moderating the problem mentioned above regarding mixing current-sources and conductance channels.

We define a new optimization problem based on eq. (7) that treats target currents smaller than \( J_{\text{th}} \) as an inequality constraint. In particular,

\[
\min_{\vec{w}_i^+} \sum_{k=1}^{N} \frac{1}{2} E \left( J_i \langle \vec{e}_i, f(\vec{x}_k) \rangle, \langle \vec{w}_i, \vec{a}(\vec{x}_k) \rangle \right)^2 + \lambda \left\| \vec{w}_i \right\|^2,
\]

where \( E(J_{\text{tar}}, J_{\text{dec}}) = \begin{cases} 0 & \text{if } J_{\text{tar}} < J_{\text{th}} \text{ and } J_{\text{dec}} < J_{\text{th}} \text{,} \\ J_{\text{th}} - J_{\text{dec}} & \text{if } J_{\text{tar}} < J_{\text{th}} \text{ and } J_{\text{dec}} > J_{\text{th}} \text{,} \\ J_{\text{tar}} - J_{\text{dec}} & \text{otherwise,} \end{cases} \tag{10}\)

and \( N \) is the number of samples, \( J_i \) the current translation function for neuron \( i \), \( \vec{e}_i \) the encoding vector of the \( i \)th post-neuron, \( f \) the desired target function, \( \vec{a}(\vec{x}_k) \) is the
pre-population activity for the kth sample point $\bar{x}_k$, $\lambda$ is the regularization factor, and $J_{th}$ the aforementioned neuronal threshold current.

By splitting the matrix of pre-population activities $A$, and the vector of target currents $\bar{J}$, we can rewrite eq. (10) as a quadratic program. Let $A^+$ and $\bar{J}^+$ be the pre-population activities and target currents for all samples with a target current larger than $J_{th}$, and $A^-$ be the activities for those samples with a target current smaller than $J_{th}$. The QP is given as

$$
\min_{\bar{w}_i, \bar{s}_i} \frac{1}{2} \bar{w}_i^T ((A^+)^T A^+) \bar{w}_i - ((A^+)^T (\bar{J}^+)) \bar{w}_i + \|\bar{s}_i\|^2_2 + \lambda \|\bar{w}_i\|^2_2,
$$

subject to $A^- \bar{w}_i - \bar{s}_i \leq J_{th},$

where $\bar{s}_i$ is a vector of slack variables. Of course, if so desired, the nonnegativity constraint from eq. (9) can be incorporated into this quadratic program.

**Extension towards synaptic nonlinearity**

Up to this point we assumed current-based synapses. Their defining property is that the somatic current $J$ is linear in the synaptic weights $\bar{w}$ and the presynaptic activity $\bar{a}$. Now, consider a neuron model with $\ell$ nonlinear input channels. We write the corresponding response curve $\mathcal{G}$ for the $i$th neuron in a population as a multivariate function

$$
a_i = \mathcal{G} \left[ g_1^i, \ldots, g_\ell^i \right] = \mathcal{G} \left[ \langle \bar{a}_1^i, \bar{w}_1^i \rangle, \ldots, \langle \bar{a}_\ell^i, \bar{w}_\ell^i \rangle \right],
$$

where $g_1^i, \ldots, g_\ell^i$ describe to the synaptic state, such as the conductance of each synaptic channel in a neuron model with conductance-based synapses. As expressed in the above equation, we assume that on average each $g_k^i$ is linear in the pre-synaptic activity $\bar{a}_k^i$. However, crucially, we do not make any assumption regarding the effect of $g_k^i$ on the somatic current; more fundamentally, we do not assume that there exists an easily identifiable somatic current in the model at all.

The crucial idea is to mathematically reintroduce a “virtual” somatic current $J$ by decomposing $\mathcal{G}$ into a synaptic nonlinearity $H$ and a somatic nonlinearity $G$. We define $G$ and $H$ according to the following equivalence relations

$$
\mathcal{G} \left[ g_1^i, \ldots, g_\ell^i \right] = G \left[ H \left( g_1^i, \ldots, g_\ell^i \right) \right] \Leftrightarrow H \left( g_1^i, \ldots, g_\ell^i \right) = J \Leftrightarrow G \left[ J \right] = \mathcal{G} \left[ g_1^i, \ldots, g_\ell^i \right],
$$

where $H$ maps from the synaptic state onto an average somatic current and, as before, $G$ maps from a somatic current onto the output activity.

While this formalization does not constrain $G$ and $H$ beyond the above equivalence relationship, a sensible choice for $G$ and $H$ will facilitate solving for synaptic weights. For example, if the neuron model in question is an extension to the current-based LIF neuron model, it makes sense to select $G$ as the standard LIF response curve. Then, $H$ translates the synaptic state into an “LIF-equivalent somatic current”.

While, as we show in the next section, $H$, or at least a parametrized surrogate for $H$, can be derived in closed form, it is similarly possible to purely rely on empirical data for $H$ by sampling the neuron output rate over varying synaptic states. Assume that we can only observe $H$ indirectly by controlling the input channels of our model neuron and measuring the output activity $\mathcal{G}$. Depending on our choice of $G[J]$, we can
Fig 4. NEURAL RESPONSE CURVE DECOMPOSITION. (A) Illustration of the multivariate neuron response curve $G(g_E, g_I)$ for a two-compartment LIF neuron with excitatory and inhibitory conductance-based channels. (B,C) The chosen somatic nonlinearity $G$ and its inverse $G^{-1}$. (D) corresponding synaptic nonlinearity $H$. The neuron does not fire in the hatched regions, that is, $G^{-1}$ is ill-defined.

Fig 5. EXPLOITING SYNAPTIC NONLINEARITIES FOR COMPUTATION. Approximation of the (linear) function $x + y$ with a two-compartment LIF neuron. The optimization process needs to find synaptic weights decoding four functions $g_E^1(x)$, $g_E^2(y)$, $g_I^1(x)$, $g_I^2(y)$ such that $H(g_E^1(x) + g_E^2(y), g_I^1(x) + g_I^2(x))$ equals $J_i(x + y)$ defined by the neuron’s current translation function. The illustration depicts an $x$-$y$-lattice in representation space being warped onto conductance space, such that the current corresponding to the desired output in representational space is invoked.
apply an inverse mapping $G^{-1}$ to the recorded data to obtain $H$. See Fig 3 for an illustration.

When solving for weights that approximate a specific function, recall from the above review that the first NEF principle normatively assigns a somatic current $J_i(\vec{x})$ to any representational state $\vec{x}$. Correspondingly, given $H$, we can combine eq. (12) with the current-decoding problem discussed above, as well as the nonnegativity constraint and the equality relaxation, resulting in the following optimization problem for each post-neuron $i$

$$
\min_{\vec{w}_1^i, \ldots, \vec{w}_\ell^i} \sum_{k=1}^{N} E \left( J_i(\langle \vec{e}_i, f(\vec{x}_k) \rangle), H(\langle \vec{w}_1^i, \vec{a}_1^k \rangle, \ldots, \langle \vec{w}_\ell^i, \vec{a}_\ell^k \rangle) \right)^2 + \lambda \sum_{j=1}^{\ell} \| \vec{w}_j^i \|_2^2,
$$

subject to $\vec{w}_j^i \geq 0$,

where $E$ is as defined in eq. (10). See Fig 5 for a visualization of the optimization problem.

Two-compartment leaky integrate-and-fire neuron model

In the previous section we established the abstract notion of a dendritic nonlinearity model $H$. We now derive a concrete $H$ for a biologically plausible extension to the leaky integrate-and-fire (LIF) neuron with nonlinear conductance-based synapses. In particular, we consider a two-compartment version of the LIF neuron originally described in [22] and subsequently discussed by [17] and [45]. In addition to the active, somatic compartment, the two-compartment LIF model possesses a resistively coupled passive compartment that represents distal excitatory and inhibitory input into the dendritic tree.

We first review the neuron model itself, derive a parametrized surrogate for the synaptic nonlinearity $H$, and finally show that for this $H$ the synaptic weight optimization in eq. (14) can be phrased in terms of a convex quadratic program.

Model description

The sub-threshold dynamics of the conductance-based two-compartment LIF model can be expressed as a two-dimensional system of linear differential equations

$$
\begin{align*}
\frac{d}{dt} C_{m,1} v_1 &= g_C(v_2 - v_1) + g_{l,1}(E_L - v_1), \\
\frac{d}{dt} C_{m,2} v_2 &= g_C(v_1 - v_2) + g_{l,2}(E_L - v_2) + g_E(t)(E_E - v_2) + g_I(t)(E_I - v_2),
\end{align*}
$$

where state variables $v_1, v_2$ correspond to the membrane potential of the active somatic compartment and the passive dendritic compartment, respectively. $C_{m,1}, C_{m,2}$ are the compartment capacitances, $g_C$ is the inter-compartment coupling conductance, $g_{l,1}, g_{l,2}$ are the individual compartment leak conductances, $g_E(t), g_I(t)$ are the momentary excitatory and inhibitory conductances of the dendritic compartment as evoked by pre-synaptic spikes, and $E_L, E_E, E_I$ are the conductance-channel reversal potentials. An equivalent circuit diagram of the model is shown in Fig 6A.

In contrast to point-neuron models, and as pointed out by [14], a multi-compartment model mandates an explicit spike model. The strong depolarization in the somatic compartment during spike production propagates backwards into the dendritic compartment and has a significant effect on its membrane potential [Fig 6B]. The model accounts for this with a “spike-phase” occurring right before the LIF refractory period. The spike phase is implemented by clamping the
Fig 6. Two-compartment LIF neuron model. (A) Equivalent circuit diagram of the two-compartment LIF neuron model. The “somatic” compartment corresponds to a classical LIF neuron; the “dendritic” compartment is resistively coupled to the somatic compartment. The current flowing between the two compartments corresponds to the somatic current \( J = H(g_E, g_I) \). (B) Voltage trace for the two-compartment model state variables \( v_1 \) and \( v_2 \) during spike production. The artificial depolarization following spike events in the somatic compartment significantly affects \( v_2 \).

somatic compartment to a voltage \( v_{\text{spike}} \) over a time \( \tau_{\text{spike}} \) whenever the somatic membrane potential crosses the threshold \( v_{\text{th}} \). Subsequently, the membrane is clamped to \( v_{\text{reset}} \) for a period \( \tau_{\text{ref}} \).

**Somatic current surrogate model**

We assume that \( H(g_E, g_I) \) is equivalent to the current flowing from the dendritic into the somatic compartment (Fig 6A). Considering the definition in eq. (14), this implies that \( G \) must be the standard LIF response curve as defined in eq. (2). Yet, in practice, as pointed out in [46] and as we demonstrate in our experiments below, a rectified linear unit (ReLU) may be a sensible choice for \( G \) as well, especially when modeling noisy input.

Unfortunately, when considering both sub- and superthreshold dynamics, there exists no exact closed-form solution for the average somatic current given constant \( g_E, g_I \). Instead, our approach is to select a parametrized surrogate model for \( H \) and to fit the model parameters to results obtained from numerical simulations to account for inaccuracies in our derivation.

Assuming subthreshold dynamics described in eq. (15) are in equilibrium, \( g_E, g_I \) are constant, and applying the equality \( H(g_E, g_I) = g_C(v_2 - v_1) \), we get

\[
H(g_E, g_I) = g_C g_{L.2}(E_L - v_1) + g_E(E_E - v_1) + g_I(E_I - v_1) \over g_C + g_{L.2} + g_E + g_I.
\]

(16)

Since the somatic membrane potential is clamped during the spike and refractory phases, the current flowing into the soma during those times does not influence the overall firing rate (ignoring the feedback effect on \( v_2 \) discussed earlier). Furthermore, once the neuron is tonically firing, the somatic membrane potential oscillates between \( v_{\text{reset}} \) and \( v_{\text{th}} \). We can thus substitute \( v_1 \) with a constant average membrane potential \( \bar{v}_{\text{mem}} = \frac{1}{2}(v_{\text{reset}} + v_{\text{th}}) \) (Fig 7; see [47] for more detail). Note that a single-compartment model can be derived by taking the limit of eq. (16) for \( g_C \to \infty \). In this case, and as demonstrated in [47], \( H \) is an affine function and less interesting from a computational perspective.

Under the above assumptions, \( H(g_E, g_I) \) from eq. (16) can be written as a parametrized rational function

\[
H(g_E, g_I) = \frac{b_0 + b_1 g_E + b_2 g_I}{a_0 + a_1 g_E + a_2 g_I}, \quad \text{where } a_0, a_1, a_2, g_E, g_I \geq 0.
\]

(17)
Fig 7. **Average LIF membrane potential over firing rate.** Data corresponds to a standard LIF neuron (corresponding to the somatic compartment) over varying output rates. Data measured by capturing 1000 membrane potential traces for a ten second current sweep. Individual lines correspond to the effective input spike rates, where smaller input rates are equivalent to a larger amount of noise on the input current. Shaded areas correspond to the 25/75 percentile. (A) Average potential excluding the refractory period and spike production. Except for very low firing rates, the average potential remains relatively constant. Dotted line corresponds to \( v_{\text{som}} = \frac{1}{2} (v_{\text{reset}} + v_{\text{th}}) \). (B) Average potential including the refractory and spike production. Dotted line corresponds to a linear model that takes the relative amount of time spent in the subthreshold, spike, and refractory regime into account.

This model equation has one superfluous degree of freedom in the parameter space and one of the parameters can be set to an arbitrary non-zero constant value. One numerically stable normalisation is to set \( b_1 = 1 \).

These equations imply an absolute maximum and minimum somatic current; \( H(g_E, g_I) \) maps onto an open interval \( (J_{\text{max}}, J_{\text{min}}) \), where

\[
J_{\text{min}} = \lim_{g_I \rightarrow \infty} H(g_E, g_I) = -\frac{b_2}{a_2} = g_C(E_1 - \tau_{\text{som}}),
\]

\[
J_{\text{max}} = \lim_{g_E \rightarrow \infty} H(g_E, g_I) = \frac{b_1}{a_1} = g_C(E_E - \tau_{\text{som}}). \tag{18}
\]

In practice, the maximum attainable current for realistic conductance values is significantly smaller than \( J_{\text{max}} \), limiting the maximally attainable firing rate. This must be taken into account when selecting the neuron tuning curve.

Optimal model parameters with respect to a least squares loss can be found by solving the QP

\[
\min_{b_0, b_2, a_0, a_1, a_2} \sum_{J_i \gg J_{\text{th}}} \left( b_0 + b_2 g_i^2 - J_i a_0 - J_i^* g_E^i a_1 - J_i^* g_I^i a_2 - g_E^i \right)^2, \tag{19}
\]

subject to the nonnegativity constraints in eq. (17), where \( J_i = G^{-1} [\theta(g_E^i, g_I^i)] \). The conductances \( g_E^i, g_I^i \) should be sampled over the operating range of the neuron. See the notes regarding numerical stability in Appendix S2. The \( J_i \) denote the equivalent current that would evoke the measured output rate \( \theta(g_E^i, g_I^i) \) when injected into an ideal neuron with response curve \( G \). Samples with zero or very small output rates should be ignored in the optimization process for two reasons. First, the inverse of \( G \) is not defined for a zero output rate. Second, \( H \) was derived under the assumption of superthreshold dynamics and is thus not likely to predict subthreshold or near-subthreshold somatic currents correctly.
Optimal synaptic weights as a quadratic program

Given the nonlinearity model $H$ as defined in eq. $17$, our goal is to find weights $\bar{w}_i^E, \bar{w}_i^I$ such that a desired current $(j)_k$ flows into the soma for every sample $k$, cf. eq. $8$. Let $\div$ and $\circ$ denote elementwise division and multiplication, respectively. We can now write the desired relationship in vector notation as

$$\bar{j}_i = \frac{b_0 + b_1 A \bar{w}_i^E - b_2 A \bar{w}_i^I}{a_0 + a_1 A \bar{w}_i^E + a_2 A \bar{w}_i^I}, \quad \text{where } a_0, a_1, a_2, A, \bar{w}_i^E, \bar{w}_i^I \geq 0,$$

and $A$ is a matrix of pre-population activities. We can rearrange this equation into a canonical form

$$\bar{j}_i \circ (1 + a_1 A \bar{w}_i^E + 2 A \bar{w}_i^I) = b_0 + b_1 A \bar{w}_i^E - b_2 A \bar{w}_i^I,$$

$$\Leftrightarrow (a_1 J_i \circ A - b_1 A) \bar{w}_i^E + (a_2 J_i \circ A + b_2 A) \bar{w}_i^I = b_0 - a_0 \bar{j}_i, \quad \text{where } (J_i)_{mn} = (\bar{j}_i)_m,$$

$$\Leftrightarrow \bar{A}_i \bar{w}_i^E = \bar{b}_i,$$

where $\bar{A}_i = (a_1 J_i \circ A - b_1 A, a_2 J_i \circ A + b_2 A)^T$, $\bar{b}_i = (b_0 - a_0 \bar{j}_i)$, and $\bar{w}_i^E = (\bar{w}_i^E, \bar{w}_i^I)$. The elements in $\bar{A}_i$ can be interpreted as the maximum voltage differentials between the somatic and dendritic compartment that can be evoked by an excitatory or inhibitory pre-neuron for each sample. The goal of the weight optimization is to select synaptic conductances $\bar{w}_i^E$ that translate the voltage differentials into the desired currents $\bar{b}_i$. We account for the subthreshold equality relaxation, by splitting $\bar{A}_i$, $\bar{b}_i$ according to the samples invoking a zero firing rate (resulting in $\bar{A}_i^0$, $\bar{b}_i^0$) and those invoking a positive firing rate (resulting in $\bar{A}_i^+$, $\bar{b}_i^+$). Then, we can write the optimization problem outlined in eq. $14$ as a quadratic program

$$\min_{\bar{w}_i^E, \bar{s}_i} \frac{1}{2} (\bar{w}_i^E)^T ((\bar{A}_i^+)^T \bar{A}_i^+) \bar{w}_i^E - (\bar{b}_i^+)^T \bar{A}_i^+ \bar{w}_i^E + \lambda \|\bar{w}_i^E\|_2^2 + \|\bar{s}_i\|_2^2,$$

subject to $\bar{A}_i^0 \bar{w}_i^E - \bar{s}_i \geq \bar{b}_i^0$ and $\bar{w}_i^E \geq 0$. \hfill (21)

See Appendix S2 for a discussion on how to ensure that this problem is numerically stable.

Results

In this section we experimentally validate the methodology discussed above. Notably, we demonstrate that $H$ is indeed a suitable model of the two-compartment LIF dendritic nonlinearity. Furthermore, we study the computational properties of $H$ by approximating random bandlimited functions. Finally, we compare the performance of dendritic computation in a feed forward network with two-compartment LIF neurons to that of standard LIF neurons with an additional neuron layer. Unless explicitly specified, the neuron model parameters are chosen according to Table S3 in all experiments. In particular, we model fast excitatory synapses as an exponential low-pass with a time-constant of 5 ms as found in glutamatergic pyramidal neurons with AMPA receptor $[48]$. The inhibitory pathway is modeled with a 10 ms time-constant as found in inhibitory interneurons with GABA$_A$ receptors $[49]$. We use the cvxopt library $[50]$ as a QP solver. The source code of the computer programs used to conduct and evaluate the experiments can be found in the supplemental materials.
Fig 8. Somatic current model parameter fits.
Average spike rates $\mathcal{N}(g_E, g_I)$ measured in simulation are depicted as colored contour plots and solid lines in the cross-section. Dashed lines correspond to the model prediction $G[H(g_E, g_I)]$. Dotted lines indicate the location of the cross-section. $E$ denotes the RMS spike-rate error over the regions where either the measured or predicted spike rate is greater than 12.5. Columns correspond to different values of $g_C$. Top half shows the spike measured and modeled spike rates for constant conductance values $g_E, g_I$. The bottom half corresponds to conductance values with superimposed spike noise. The model fits the simulated response functions well after optimization.
Fitting the surrogate model to simulation data

Given excitatory and inhibitory conductances $g_E$ and $g_I$, the surrogate model $H(g_E, g_I)$ in eq. (17) predicts the current flowing into the somatic compartment of a two-compartment LIF neuron. $H$ is characterized by five parameters, for which a coarse theoretical estimate can be derived from eq. (16). Since, as discussed in the previous section, this theoretical estimate is based on several assumptions that are violated in practice, the parameters should be fit to empirical data, for example by solving the least squares problem suggested in eq. (19). Once a parameter set has been selected, $H$ can be used in conjunction with the current response curve $G[\cdot J]$ to predict the neural activity by computing $G[H(g_E, g_I)]$.

We perform two experiments that compare spike rates measured in numerical simulation to the spike rate prediction obtained before and after fitting the model parameters to empirical data. In the first experiment we measure empirical spike rates for constant conductances $g_E$, $g_I$. In the second experiment we superimpose artificial temporal spike noise on the conductance pair. Our results show that $H$ is a reasonably good predictor after fitting the parameters to empirical measurements.

**Constant conductances**

We consider three two-compartment LIF neurons with different coupling conductances $g_C \in \{50 \text{nS}, 100 \text{nS}, 200 \text{nS}\}$ and measure their output spike rate for conductances $g_E$, $g_I$ on an equally-spaced $100 \times 100$ grid. In contrast to the next experiment, we assume that the conductances stay constant throughout the one-second simulation period. The conductance range has been selected such that the maximum firing rate is $100 \text{s}^{-1}$, and the spike onset approximately coincides with the diagonal of the resulting $g_E$-$g_I$-rate contour plot. We measure the steady-state firing rate by taking the inverse of the median inter-spike-interval over the simulation period. We compare these data to the current predicted by $H$ according to the theoretical and optimized estimated parameter sets. Parameter optimization is based on a training-set of 200 conductance pairs sampled with uniform probability from the conductance-range. The final prediction error is computed over all $10^4$ grid points.

The results are depicted in Fig 8, individual parameters can be found in Table S4. When using the theoretical estimate, there is a significant discrepancy between the model prediction and the numerical simulation, especially for large $g_C$. This discrepancy is greatly reduced after fitting the model parameters. For output spike rates greater than $25 \text{s}^{-1}$, the fitted model prediction almost perfectly fits the empirical data. However, it fails to predict the spike onset correctly, placing it too early with respect to increasing $g_E$. Furthermore, the predicted slope at the spike onset is less steep than what is actually measured. Still, with an overall RMSE of about four spikes per second, the model provides a reasonable approximation of the empirical data.

**Conductances with artificial temporal spike noise**

In a network context, $g_E(t)$ and $g_I(t)$ are usually not constant, but instead modeled as the weighted sum of low-pass filtered spike trains, resulting in a considerable amount of “spike noise” being superimposed onto the signal. In this experiment we simulate artificial spike noise as two Poisson spike sources (one excitatory, one inhibitory) with rate $1/\lambda = 1800 \text{s}^{-1}$ for inhibitory synapses and a rate of $1/\lambda = 4500 \text{s}^{-1}$ for excitatory synapses. These rates were measured in the network experiment below. Spikes arriving at different synapses are simulated by uniformly sampling a random weight from the unit-interval for each spike event. The time-average conductance equals $g_E$, $g_I$, respectively. Apart from the simulation period being extended to one hundred seconds, the remaining experimental setup is unchanged from the last experiment.
As can be seen in the bottom cross-sections of the measured spike-rates in Fig 8, the steep spike-onsets predicted by the theoretical LIF response curve \( G[J] \) (eq. 2) are no longer present. Furthermore, the relationship between \( g_E \) and the rate appears to be roughly linear in each cross-section of the \( g_E-g_I \)-rate plot, which is not well captured by the standard LIF response curve. Hence, a “soft” version of the LIF response curve that takes noise and the stochastic nature of the spike production process into account would be a better choice [46, 51–54]. We instead take a pragmatic approach and define \( G[J] \) as a rectified linear unit (ReLU), that is \( G[J] = \max\{0, \alpha J + \beta\} \). Such a partially linear approximation is suitable since curvature in the measured rate is likely modeled by \( H \). The free parameters \( \alpha \) and \( \beta \) can be easily fitted to the empirically measured output rate and somatic current \( J \). However, in practice, setting \( \alpha = 1 \) and \( \beta = 0 \) is sufficient, since fitting the parameters for the dendritic nonlinearity \( H \) will implicitly determine the ReLU parameters as well. We separate this process into two steps to be able to show the before- and after model-fit comparison in Fig 8. Due to a relatively pronounced noise floor we only consider samples with a spike rate greater than 12.5 s\(^{-1}\) for both fitting the model parameters and calculating the RMSE.

Again, results are shown in the top half Fig 8. As before, there is a significant discrepancy between the predicted and measured spike rate when using the theoretical model parameters, although in general, the prediction error is smaller than in the previous experiment. Fitting the parameters results in excellent accuracy for spike rates greater than ten spikes per second (RMSE less than 2 s\(^{-1}\)). However, the fitted model does not capture the subtle sigmoid shape of the response curve near the spike onset that is particularly pronounced for larger \( g_C \). Generally speaking, the fit quality is even better than in the previous experiment, mostly because the added noise is smoothing the abrupt spike onset in the theoretical spike response curve model.

**Computational properties of the two-compartment LIF nonlinearity**

The dendritic nonlinearity model \( H \) for the two-compartment LIF neuron introduced in eq. (17) cannot solve the XOR problem—at least not when taking biophysical nonnegativity constraints into account. A proof of this fact is given in Appendix S1. As a direct consequence, \( H \) cannot be used to approximate multiplication over all four input quadrants. Nevertheless, as we demonstrate in this experiment, \( H \) approximates a larger class of functions with a significantly lower error than a purely current-based dendritic model.

On a conceptual level, a fine-grained characterization of the computational properties of a dendritic nonlinearity would be to measure how well \( H \) can be used to approximate functions of increasing complexity. A suitable proxy for the “complexity” of a function is its spatial frequency content, or, in other words, the number of Fourier coefficients significantly greater than zero. Consider the set of zero-mean and variance-normalized functions over \([-1, 1]^2\). Functions within this set with a very low spatial cut-off frequency will be approximately linear. In contrast, those with a cut-off frequency greater than one possess multiple maxima/minima over the interval, akin to multiplication and XOR. Correspondingly, since XOR cannot be solved by the two-compartment model, we would expect the conductance-based nonlinearity to mainly improve the approximation accuracy for functions with a spatial frequency smaller than one.

In our experiment we randomly generate bandlimited current functions \( J_\sigma(x, y) \) over the compact domain \((x, y) \in [-1, 1]^2\), where \( \sigma \) is a low-pass spatial filter coefficient that is inversely proportional to the cut-off frequency. We generate these functions by sampling from a normal distribution on a 16 × 16 grid and applying a
Fig 9. Median decoding error for random multivariate current functions. Normalized RMSE (normalization with respect to the target function RMS) between a random, zero mean two-dimensional function and the decoded approximation. The low-pass filter coefficient $\sigma$ is a proxy for the spatial frequency content in the target function. All points correspond to the median over 10000 trials. Dashed lines show results for not taking the subthreshold relaxation into account. The black lines show the results for a linear, current based dendritic model $H_{\text{cur}}$; colored lines show the results for the two-compartment conductance-based model with parameters given in Table S4 (without noise). Shaded area correspond to the 25/75 percentile for the current-based model and the conductance-based model with $g_C = 50\,\text{nS}$. Panels A and B show example target functions and approximations. Light grey/hatched regions correspond to subthreshold currents. Approximation errors $E$ are the normalized RMSE between the depicted target function and the decoding.

Gaussian low-pass spatial filter with standard deviation $\sigma$. The samples are whitened such that the mean is zero and the standard-deviation/RMS equals 1 nA. We compute synaptic weights that approximate this target function by solving the quadratic program specified in eq. (21). We compare the approximation error achieved with a conductance-based dendritic model $H_{\text{cond}}$ with the parameters derived in the last subsection, and a current-based model $H_{\text{cur}}(J_E, J_I) = J_E - J_I$. Furthermore, we repeat the same experiment both with and without the subthreshold relaxation.

The accuracy measurement is based on the network setup depicted in Fig 3C. Two independent pre-populations with 100 neurons each represent $x$ and $y$, respectively, and project onto the post-population. The population tuning curves are randomly generated with a maximum firing rate between 50 and 100 s$^{-1}$ per neuron. Since biological constraints are secondary for a mathematical analysis of $H$, all pre-neurons project both inhibitorily and excitorily onto each post-neuron. Furthermore, we solely measure the static decoding error, i.e., the difference between the target current function and the output of the surrogate model assuming ideal pre-population tuning curves. We consider dynamical simulation of a spiking network in the next subsection.

For the optimization we choose a regularization parameter of $\lambda = 10^{-4}$. With this $\lambda$, the current-based $H$ can decode a perfectly linear function with an error of less than $10^{-10}$, while the conductance-based $H$ achieves an error smaller than $10^{-4}$. 
Results are depicted in the bottom half of Fig 9. For a current-based neuron the error increases linearly on a log-log plot from a 1% error for low-frequency, almost linear, functions to an error of about 45% for functions with a spatial cut-off frequency greater than one. The 1% error is remarkable, since, as mentioned above, a perfectly linear function can be decoded with several orders of magnitude lower error using exactly the same methodology. Hence, even the slight curvature present in the low-frequency target functions is sufficient to drastically increase the error.

The conductance-based nonlinearity increases sub-linearly on a log-log plot, with an error of $10^{-4}$ for low-frequency functions. The error function converges to the results for the current-based model for spatial frequencies greater than one. For smaller spatial frequencies using conductance-based nonlinearity results in a one to two orders of magnitude smaller error compared to current-based neurons. The differences between the individual parameter sets derived in the last experiment are relatively small.

The subthreshold relaxation has a significant effect on both the accuracy of current-based and conductance-based decodings in the low-frequency region, resulting in a 50% error reduction in the current-based case, and a reduction of up to one order of magnitude in the conductance-based case.

**Dendritic computation of multivariate functions in spiking networks**

Our above experiments demonstrate that the nonlinearity $H$ defined in eq. (17) can be fit to the spiking response of a two-compartment LIF neuron with a relatively small error. Furthermore, the results suggest that the conductance-based nonlinearity can be used to decode a significantly larger class of current-functions from the pre-population than a linear, current-based model.

In our final experiment we test both properties simultaneously by measuring the decoding error in the context of a feed-forward spiking neural network. Taking the previous results into account, we would expect that the nonlinear interaction within the dendritic compartment of two-compartment LIF neurons is sufficient to approximate a wider range of multivariate, nonlinear functions with a small error than a population of standard LIF neurons.

Specifically, we analyze the network topologies depicted in Fig 3 with respect to the average decoding error when computing a set of multivariate functions $f(x, y)$ over the input domain $[0,1]^2$. The functions are defined in Table S5. On the input-side of the network, the scalars $x$ and $y$ are represented by populations of 100 neurons each. These populations project onto a target population representing $f(x, y)$. The target population either consists of standard current-based LIF neurons (Fig 3A) or conductance-based two-compartment neurons (Fig 3C). For the latter we use all parameter sets with and without the noise model summarized in Table S4. Alternatively, and corresponding to the two-layer topology in Fig 3B, we consider a current-based LIF network where the input is mediated via an additional layer of 200 neurons representing the vectorial quantity $(x, y)$.

For all neurons, we generate random tuning curves such that the maximum firing rate falls between 50 and $100 \text{s}^{-1}$ over the range of represented values. Neurons are randomly marked as either excitatory or inhibitory, where the probability of a neuron being inhibitory is 30%. Excitatory and inhibitory synapses are modeled as an exponential low-pass filter with time-constants of $\tau_E = 5 \text{ ms}$ and $\tau_I = 10 \text{ ms}$, respectively.

The network is simulated over 10s at a time-resolution of $10^{-4}$ s. Inputs $x$ and $y$ are sampled by moving through time along a fourth-order space-filling Hilbert curve.
The output of the target population is decoded and low-pass filtered at \( \tau = 100 \text{ ms} \). As a reference, we compute the desired target value \( f(x, y) \) from the original input and pass it through the same series of low-pass filters as the spiking signals. We use the average synaptic time-constant of \( 7.5 \text{ ms} \) to emulate the effect of the synaptic low-pass filters. Our final measure \( E_{\text{net}} \) is the normalized RMSE between the decoded output and the reference values over time; the normalization is relative to the RMS of the reference signal. See Fig 10 for an example input and output and the corresponding spike trains.

All synaptic weights are computed by solving the QP in eq. (21). We emulate current-based LIF neurons in the context of the conductance-based two-compartment LIF nonlinearity \( H \) by setting the model parameters to \( b_0 = a_1 = a_2 = 0, b_1 = 1, b_2 = -1 \). The regularization term \( \lambda \) has been chosen independently for each neuron type and model parameter set such that the network error \( E_{\text{net}} \) is minimized when computing multiplication. Refer to Fig S6A for more information on the choice of the regularization factor.

A summary of the results over 256 trials is given in Table 1. More detailed results can be found in Table S6A. For all but one target function (squared multiplication), the conductance-based two-compartment model with a coupling conductance of \( g_C = 50 \text{ ns} \) achieves the smallest error \( E_{\text{net}} \). Using the surrogate model parameters derived under noise is beneficial when computing multiplicative functions. For these target functions the synaptic connection matrix tends to be sparser, increasing the input noise. Apparently, this increase in noise matches the environment the neuron parameters have been optimized for. Interestingly, when taking the subthreshold relaxation into account, a purely current-based, single-layer network is competitive for all non-multiplicative target functions. Additional noise introduced by the intermediate population prevents the two-layer network from achieving a lower approximation error in these cases. However, this network is clearly better than all other setups when computing squared multiplication. This function possesses pronounced spatial high-frequency components and can, as explored in the last experiment, not be fitted well by the two-compartment model.

An effect that may contribute to the superior performance of the two-compartment neuron model are low-pass filter dynamics of the dendritic compartment. These filter the high-frequency spike noise and thus may reduce the target error. We control for this effect in an experiment described in Appendix S6, where we add an optimal low-pass filter to each network setup. Results are shown in Table S6B. While the pre-filter significantly reduces the error of the current-based setups, the conductance-based two-compartment model still performs at least as well as the two-layer setup and consistently outperforms the single-layer current based network.

**Discussion**

We discussed and derived a mathematical model of dendritic computation and experimentally demonstrated that networks with fewer layers but biophysically plausible dendritic nonlinearities can compute a broad range of multivariate functions as well as or better than networks typically built by functional modeling frameworks. In particular, we proposed a mathematical model of dendritic computation \( H \) that captures nonlinear interaction in the dendritic tree. By mapping individual channel states onto an average somatic current \( J \) this model can be integrated into mathematical frameworks that classically rely on current-based input channels.

Specifically, we demonstrated how to incorporate the dendritic nonlinearity \( H \) into the Neural Engineering Framework (NEF). To this end, we discussed extensions to the NEF that allow us to optimize for nonnegative synaptic weights that invoke a desired
Fig 10. Single spiking network experiment showing computation of multiplication using a two-compartment LIF neuron. (A) Top two plots: inputs $x(t)$ and $y(t)$ as represented by the two pre-populations. The input is a fourth order 2D Hilbert curve. Bottom: mathematical target $f(x(t), y(t)) = x(t)y(t)$, filtered target function, as well as the decoded target population output. (B) Spike raster plots corresponding to the spiking activity of each of the population. Red shaded background corresponds to inhibitory neurons in the pre-populations, all other neurons are excitatory.

Table 1. Spiking neural network approximation errors for function approximations on $[0, 1]^2$. Error values correspond to the normalized RMSE (relative to the RMS/standard deviation of the target function) and are measured as the difference between the output decoded from the target population and the desired output for a ten second sweep across a 4th order 2D Hilbert curve over the input space. Results are the mean and standard deviation over 256 trials. The best result for a target function is set in bold; darker background colors indicate a worse ranking of the result in the corresponding row. Additional tables can be found in Appendix S6.
somatic current $J$, and relax the optimization problem by taking subthreshold currents into account. We combined these methods with a specific surrogate model for $H$ in the context of a two-compartment LIF neuron. Finally, we performed a series of spiking neural network simulations that show that our methods allow dendritic nonlinearities to be systematically exploited to efficiently approximate nonlinear multivariate functions.

While our approach is a step towards providing a general model of dendritic computation in top-down neurobiological modeling frameworks, it admittedly has several limitations. Most importantly, we treat the dendritic nonlinearity $H$ as time-independent. Correspondingly, we implicitly assume that synaptic time-constants typically dominate the overall neuronal dynamics. However, dendritic trees in biology—especially when considering active channels and dendritic spikes [17]—possess filter properties and adaptation processes that are not accounted for in our model. It would be interesting to incorporate the dynamical properties of dendritic trees into the NEF by employing the recent techniques presented in [55].

A further shortcoming of the derivation of the surrogate model of $H$ for the two-compartment neuron model is the assumption that the average somatic membrane potential is constant. While we are able to alleviate this assumption to some degree by fitting the model parameters to simulation data, the exact model parameters depend on the specific working-regime in which the neuron is used. Deviations from the modeled behavior are particularly apparent in situations with output firing rates smaller than ten spikes per second (cf. Figs 7A and 8). Correspondingly, the dendritic nonlinearity presented in this paper may not be a suitable model for brain areas featuring extremely low maximum firing rates. There are two potential ways to work around this limitation. First, it may be possible to include an input-dependent membrane potential term in the nonlinearity. Or second, one could directly use a sampled model for $H$. While these approaches are compatible with the concept of dendritic nonlinearity as introduced above, they both increase the mathematical complexity of the weight optimization problem to a point where strategies such as stochastic gradient descent are required. These techniques tend to have significantly weaker guarantees regarding finding an optimal solution compared to the convex quadratic programs employed in this paper.

In light of the above limitations, we would like to re-emphasize that, as stated in the introduction, our goal is not to provide a detailed mechanistic model of dendritic computation. Instead, we hope to provide a useful tool that captures essential aspects of dendritic computation—a nonlinear interaction between input channels—while being computationally cheap and mathematically tractable, but still grounded in biophysics. This helps to bridge the gap between purely abstract functional networks and more biophysically grounded mechanisms.

A potential application of our work outside of neurobiological modeling is programming neuromorphic hardware. Neuromorphic computers are inspired by neurobiological principles and promise to reduce the energy consumption of certain computational problems by several orders of magnitude compared to conventional computers [50]. Especially when considering mixed analogue-digital neuromorphic hardware systems, it should be possible to achieve a higher energy efficiency by implementing a more complex model neuron—such as the two-compartment LIF neuron discussed here—and performing local analog computation. Potential future work in this regard would be to validate our methods on a neuromorphic computing platform that implements dendritic trees, such as the BrainScales 2 system [57].

Another line of future work is to consider arbitrary configurations of passive dendritic trees beyond the two-compartment LIF model. By applying Kirchhoff's circuit laws, any passive dendritic tree configuration can be described as a linear
dynamical system. Correspondingly, it is possible to derive the dendritic nonlinearity $H$. It would be interesting to see whether it is still possible to relatively quickly solve for connection weights and in how far the number of compartments influences the computational power of the dendritic nonlinearity.

In conclusion, we believe that the methods proposed here provide a solid grounding for future work exploring both detailed biophysical mechanisms in the context of functional spiking networks, and improving neuromorphic methods for neural computation. We have shown how to cast the determination of connection weights in a functional network with conductance based synapses as an optimization problem with guaranteed convergence to the minimum. This optimization not only exploits known dendritic nonlinearities, but respects specifiable network topologies that conform to Dale’s Principle. The result are functional spiking networks with improved accuracy and biophysical plausibility using fewer neurons than competing approaches.

Acknowledgments

The authors would like to thank Aaron R. Voelker for his comments on earlier drafts of this paper, as well as his advise and constructive criticism with respect to the conducted experiments.

References

1. Churchland PS, Koch C, Sejnowski TJ. What Is Computational Neuroscience? In: Computational Neuroscience. Cambridge, Massachusetts: MIT Press; 1993. p. 46–56.

2. Eliasmith C, Anderson CH. Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems. Cambridge, Massachusetts: MIT press; 2003.

3. Eliasmith C. How to Build a Brain: A Neural Architecture for Biological Cognition. Oxford Series on Cognitive Models and Architectures. Oxford University Press; 2013.

4. Boerlin M, Denève S. Spike-Based Population Coding and Working Memory. PLOS Computational Biology. 2011;7(2):1–18. doi:10.1371/journal.pcbi.1001080.

5. Boerlin M, Machens CK, Denève S. Predictive Coding of Dynamical Variables in Balanced Spiking Networks. PLOS Computational Biology. 2013;9(11):1–16. doi:10.1371/journal.pcbi.1003258.

6. Sussillo D, Abbott LF. Generating Coherent Patterns of Activity from Chaotic Neural Networks. Neuron. 2009;63(4):544–557. doi:https://doi.org/10.1016/j.neuron.2009.07.018.

7. Nicola W, Clopath C. Supervised Learning in Spiking Neural Networks with FORCE Training. Nature Communications. 2017;8(1):2208. doi:10.1038/s41467-017-01827-3.

8. Komer B, Eliasmith C. A unified theoretical approach for biological cognition and learning. Current Opinion in Behavioral Sciences. 2016;11:14–20. doi:https://doi.org/10.1016/j.cobeha.2016.03.006.
9. Stewart TC, Bekolay T, Eliasmith C. Learning to Select Actions with Spiking Neurons in the Basal Ganglia. Frontiers in Decision Neuroscience. 2012;6. doi:10.3389/fnins.2012.00002.

10. Duggins P, Stewart TC, Choo X, Eliasmith C. Effects of Guanfacine and Phenylephrine on a Spiking Neuron Model of Working Memory. Topics in Cognitive Science. 2017;doi:10.1111/tops.12247.

11. Schwemmer MA, Fairhall AL, Denéve S, Shea-Brown ET. Constructing Precisely Computing Networks with Biophysical Spiking Neurons. Journal of Neuroscience. 2015;35(28):10112–10134. doi:10.1523/JNEUROSCI.4951-14.2015.

12. Eliasmith C, Gosmann J, Choo XF. BioSpaun: A Large-Scale Behaving Brain Model with Complex Neurons; 2016. Available from arXiv:1602.05220. Cited 21 March 2018.

13. Duggins P. Incorporating Biologically Realistic Neuron Models into the NEF [Masters thesis]; 2017.

14. Bobier B, Stewart TC, Eliasmith C. A unifying mechanistic model of selective attention in spiking neurons. PLoS Computational Biology. 2014;10(6):e1003577. doi:10.1371/journal.pcbi.1003577.

15. Alemi A, Machens CK, Deneve S, Slotine JJ. Learning Nonlinear Dynamics in Efficient, Balanced Spiking Networks Using Local Plasticity Rules. In: Thirty-Second AAAI Conference on Artificial Intelligence; 2018.

16. Mel BW. Information Processing in Dendritic Trees. Neural Computation. 1994;6(6):1031–1085.

17. Koch C. Biophysics of Computation: Information Processing in Single Neurons. Oxford University Press; 1999.

18. London M, Häusser M. Dendritic Computation. Annual Review of Neuroscience. 2005;28(1):503–532. doi:10.1146/annurev.neuro.28.061604.135703.

19. Polsky A, Mel BW, Schiller J. Computational Subunits in Thin Dendrites of Pyramidal Cells. Nature Neuroscience. 2004;7:621.

20. Minsky M, Papert SA. Perceptrons: An Introduction to Computational Geometry. Expanded ed. MIT Press; 1987.

21. Hornik K, Stinchcombe M, White H. Multilayer Feedforward Networks Are Universal Approximators. Neural Networks. 1989;2(5):359–366. doi:https://doi.org/10.1016/0893-6080(89)90020-8.

22. Vu ET, Lee SC, Krasne FB. The Mechanism of Tonic Inhibition of Crayfish Escape Behavior: Distal Inhibition and Its Functional Significance. Journal of Neuroscience. 1993;13(10):4379–4393.

23. Kuo PD, Eliasmith C. Integrating Behavioral and Neural Data in a Model of Zebrafish Network Interaction. Biological Cybernetics. 2005;93(3):178–187.

24. Tripp B. A Search For Principles of Basal Ganglia Function [PhD Thesis]. University of Waterloo; 2009. Available from: http://hdl.handle.net/10012/4179
25. Eliasmith C, Stewart TC, Choo X, Bekolay T, DeWolf T, Tang Y, et al. A Large-Scale Model of the Functioning Brain. Science. 2012;338:1202–1205. doi:10.1126/science.1225266.

26. Bekolay T, Bergstra J, Hunsberger E, DeWolf T, Stewart TC, Rasmussen D, et al. Nengo: A Python Tool for Building Large-Scale Functional Brain Models. Frontiers in Neuroinformatics. 2014;7(48). doi:10.3389/fninf.2013.00048.

27. Choudhary S, Sloan S, Fok S, Neckar A, Trautmann E, Gao P, et al. Silicon Neurons That Compute. In: Villa AEP, Duch W, Erdi P, Masulli F, Palm G, editors. Artificial Neural Networks and Machine Learning – ICANN 2012. Springer Berlin Heidelberg; 2012. p. 121–128.

28. Mundy A, Knight J, Stewart TC, Furber S. An Efficient SpiNNaker Implementation of the Neural Engineering Framework. In: IJCNN. Institute of Electrical and Electronics Engineers (IEEE); 2015.

29. Berzish M, Eliasmith C, Tripp B. Real-Time FPGA Simulation of Surrogate Models of Large Spiking Networks. In: Villa AEP, Masulli P, Pons Rivero AJ, editors. Artificial Neural Networks and Machine Learning – ICANN 2016. Springer International Publishing; 2016. p. 349–356.

30. Blouw P, Choo X, Hunsberger E, Eliasmith C. Benchmarking Keyword Spotting Efficiency on Neuromorphic Hardware; 2018. Available from arXiv:1812.01739. Cited 7 February 2019.

31. Neckar A, Fok S, Benjamin BV, Stewart TC, Oza NN, Voelker AR, et al. Braindrop: A Mixed-Signal Neuromorphic Architecture With a Dynamical Systems-Based Programming Model. Proceedings of the IEEE. 2019;107(1):144–164. doi:10.1109/JPROC.2018.2881432.

32. MacNeil D, Eliasmith C. Fine-Tuning and the Stability of Recurrent Neural Networks. PLoS ONE. 2011;6.

33. Penrose R. On Best Approximate Solutions of Linear Matrix Equations. Mathematical Proceedings of the Cambridge Philosophical Society. 1956;52(1):17–19. doi:10.1017/S0305004100030929.

34. Roth A, van Rossum MCW. Modeling Synapses. In: Schutter ED, editor. Computational Modeling Methods for Neuroscientists. The MIT Press; 2009. p. 139–159.

35. Broomhead DS, Lowe D. Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks; 1988.

36. Tripp B, Eliasmith C. Neural Populations can Induce Reliable Postsynaptic Currents without Observable Spike Rate Changes or Precise Spike Timing. Cerebral Cortex. 2007;17:1830–1840.

37. Strata P, Harvey R. Dale's Principle. Brain Research Bulletin. 1999;50(5):349–350. doi:https://doi.org/10.1016/S0361-9230(99)00100-8.

38. Hendry S, Jones EG. Sizes and Distributions of Intrinsic Neurons Incorporating Tritiated GABA in Monkey Sensory-Motor Cortex. Journal of Neuroscience. 1981;1(4):390–408. doi:10.1523/JNEUROSCI.01-04-00390.1981.
39. Gabbott PLA, Somogyi P. Quantitative Distribution of GABA-Immunoreactive Neurons in the Visual Cortex (Area 17) of the Cat. Experimental Brain Research. 1986;61(2):323–331. doi:10.1007/BF00239522.

40. Parisien C, Anderson CH, Eliasmith C. Solving the Problem of Negative Synaptic Weights in Cortical Models. Neural Computation. 2008;20:1473–1494.

41. Freund T, Kali S. Interneurons. Scholarpedia. 2008;3(9):4720. doi:10.4249/scholarpedia.4720.

42. DeFelipe J, Jones EG. Neocortical Microcircuits. In: Handbook of Brain Microcircuits. 1st ed. Oxford, United Kingdom: Oxford University Press; 2010. p. 5–14.

43. Wright S, Nocedal J. Numerical Optimization. Springer Science. 1999;35(67-68):438–484.

44. Kozlov MK, Tarasov SP, Khachiyan LG. The Polynomial Solvability of Convex Quadratic Programming. USSR Computational Mathematics and Mathematical Physics. 1980;20(5):223–228. doi:https://doi.org/10.1016/0041-5553(80)90098-1.

45. Capaday C, Van Vreeswijk C. Direct Control of Firing Rate Gain by Dendritic Shunting Inhibition. Journal of integrative neuroscience. 2006;5(02):199–222.

46. Hunsberger E, Scott M, Eliasmith C. The Competing Benefits of Noise and Heterogeneity in Neural Coding. Neural Computation. 2014;26(8). doi:10.1162/NECO_a_00621.

47. Stöckel A, Voelker AR, Eliasmith C. Point Neurons with Conductance-Based Synapses in the Neural Engineering Framework; 2017. Available from arXiv:1710.07659. Cited 20 Oct 2017.

48. Jonas P, Major G, Sakmann B. Quantal Components of Unitary EPSCs at the Mossy Fibre Synapse on CA3 Pyramidal Cells of Rat Hippocampus. The Journal of Physiology. 1993;472(1):615–663.

49. Gupta A, Wang Y, Markram H. Organizing Principles for a Diversity of GABAergic Interneurons and Synapses in the Neocortex. Science. 2000;287(5451):273–278. doi:10.1126/science.287.5451.273.

50. Vandenberghe L. The CVXOPT Linear and Quadratic Cone Program Solvers; 2010. Available from: http://cvxopt.org/documentation/coneprog.pdf.

51. Capocelli RM, Ricciardi LM. Diffusion Approximation and First Passage Time Problem for a Model Neuron. Kybernetik. 1971;8(6):214–223. doi:10.1007/BF00288750.

52. Kreutz-Delgado K. Mean Time-to-Fire for the Noisy LIF Neuron-A Detailed Derivation of the Siegert Formula; 2015. Available from arXiv:1501.04032. Cited 16 May 2017.

53. Hunsberger E, Eliasmith C. Spiking Deep Networks with LIF Neurons; 2015. Available from arXiv:1510.08829. Cited 21 March 2018. Available from: http://arxiv.org/abs/1510.08829.

54. Hunsberger E. Spiking Deep Neural Networks: Engineered and Biological Approaches to Object Recognition [PhD thesis]; 2018. Available from: http://hdl.handle.net/10012/12819.
Supporting Information

S1 Appendix. Proofs regarding the computability of XOR We show that an additive function of the form $f(x) + g(y)$ cannot be used to solve the XOR problem, and furthermore, that the two-compartment LIF neuron cannot solve the XOR problem.

S2 Appendix. Remarks regarding numerical stability Techniques used in our experiments to rescale the quadratic programs such that they can be solved in a numerically stable manner.

S3 Table. Neuron and synaptic model parameters Parameters for the model neurons and synaptic filters used in the experiments.

S4 Table. Current model parameters before and after fitting to empirical data Table showing the parameters derived for the surrogate dendritic current model.

S5 Table. List of multivariate functions analyzed in the network experiment Equations and contour plots detailing the multivariate functions that were analyzed in the network experiment.

S6 Appendix. Supplemental spiking network experiment results Plots detailing the choice of the regularisation parameters, as well as an unabridged version of Table 1. Furthermore, results and description of an experiment analyzing the effects of an additional low-pass filter in the network to control for the dynamics of the two-compartment neuron model.
S1 Appendix: Proofs regarding the computability of XOR

We discuss a weaker version of the XOR problem that defines an inequality relationship between the four corner points of an arbitrary rectangle. Not being able to solve “weak XOR” implies that a nonlinearity cannot solve the “hard”, equality-constrained XOR problem. The inverse is not necessarily true.

**Definition 1** (Weak XOR Problem). We define a function \( \phi(x, y) \) as being able to solve the weak XOR problem if there exist \( x_0, y_0, x_1, y_1 \) such that
\[
(\phi(x_0, y_0) < \phi(x_0, y_1)) \land (\phi(x_1, y_1) < \phi(x_1, y_0)) \land (\phi(x_0, y_0) < \phi(x_1, y_0)) \land (\phi(x_1, y_1) < \phi(x_0, y_1)) .
\]

**Theorem 1** (Sum of univariate functions cannot solve XOR). Let \( \phi(x, y) = \sigma(f(x) + g(y)) \), where \( \sigma, f, g \) all map from \( \mathbb{R} \) to \( \mathbb{R} \) and \( \sigma \) is a monotonic function, such as the signum or Heaviside function. The function \( \phi \) cannot solve the weak XOR problem.

**Proof.** Suppose \( \phi(x, y) \) could solve the weak XOR problem. Substitute \( x_0' = f(x_0), x_1' = f(x_1), y_0' = g(y_0), y_1' = g(y_1) \) and apply the above definition. We get:
\[
(\sigma(x_0' + y_0') < \sigma(x_0' + y_1')) \land (\sigma(x_1' + y_1') < \sigma(x_1' + y_0')) \\
\land (\sigma(x_0' + y_0') < \sigma(x_1' + y_0')) \land (\sigma(x_1' + y_1') < \sigma(x_0' + y_1')) .
\]

Assume without loss of generality that \( \sigma \) is monotonically increasing. Then, the first line implies \((y_0' < y_1') \land (y_0' > y_1') \) and the second line \((x_0' < x_1') \land (x_0' > x_1') \). This is a contradiction, thus the theorem holds. \( \square \)

**Corollary 1** (Perceptron cannot solve XOR). The Perceptron nonlinearity is given as \( \phi(x, y) = \sigma(wx + vy + b) \), where \( \sigma \) is a real-valued, monotonic function. This nonlinearity cannot solve the weak XOR problem.

**Proof.** This directly follows from Theorem 1 for \( f(x) = wx + b \) and \( g(x) = vy \). \( \square \)

**Theorem 2** (Two-compartment LIF cannot solve XOR). The two-compartment LIF nonlinearity \( H(g_E, g_I) \) from eq. (17) cannot solve the weak XOR problem for nonnegative inputs.

**Proof.** For \( b_0 \neq 0, H \) as given in eq. (17) can be reparametrized to \( H' \)
\[
H(g_E, g_I) = H'( \frac{b_1 g_E}{|b_0|}, \frac{b_2 g_I}{|b_0|} ) = H'(x, y) = \frac{\pm 1 + x - y}{c_0 + c_1 x + c_2 y} \quad \text{where} \quad c_0 > 0 \quad \text{and} \quad c_1, c_2, x, y \geq 0 .
\]

Assume that \( \phi(x, y) = H'(x, y) \) can solve the weak XOR problem. Since the denominator in the above nonlinearity is strictly positive, we can safely cross-multiply with the denominator across the inequalities and apply the above definition
\[
(0 < x_0 y_0 c_1 + x_0 y_0 c_2 - x_0 y_1 c_1 - x_0 y_1 c_2 + y_0 c_0 \pm y_0 c_2 - y_1 c_0 \mp y_1 c_2) \\
\land (0 < -x_1 y_0 c_1 - x_1 y_0 c_2 + x_1 y_1 c_1 + x_1 y_1 c_2 - y_0 c_0 \mp y_0 c_2 + y_1 c_0 \mp y_1 c_2) \\
\land (0 < -x_0 y_0 c_1 - x_0 y_0 c_2 + x_1 y_0 c_1 + x_1 y_0 c_2 - x_0 c_0 \pm x_0 c_1 + x_1 c_0 \mp x_1 c_1) \\
\land (0 < x_0 y_1 c_1 + x_0 y_1 c_2 - x_1 y_1 c_1 - x_1 y_1 c_2 + x_0 c_0 \mp x_0 c_1 - x_1 c_0 \pm x_1 c_1) .
\]

This can be simplified to
\[
(0 < (c_1 + c_2)x_0 \pm c_2 + c_0)(y_0 - y_1) \land (0 < -(c_1 + c_2)x_1 \pm c_2 + c_0)(y_0 - y_1) \\
\land (0 < -(c_1 + c_2)y_0 \mp c_1 + c_0)(x_0 - x_1) \land (0 < (c_1 + c_2)y_1 \mp c_1 + c_0)(x_0 - x_1) .
\]

Due to the nonnegativity constraints either the first line implies \((y_0 - y_1 > 0) \land (y_0 - y_1 < 0)\) (for the “+” branch of the “±”), or the second line implies \((x_0 - x_1 < 0) \land (x_0 - x_1 > 0)\) (for the “+” branch of the “±”), which is a contradiction. The argument for \( b_0 = 0 \) is similar. Thus, the theorem holds. In contrast to the previous proof no contradiction can be derived for both lines at the same time. In other words, there are valid parameters \( c_0, c_1, c_2 \) for which there exist \( x_0, y_0, x_1, y_1 \) such that two of the four inequalities hold. \( \square \)
S2 Appendix: Remarks regarding numerical stability

Adhering to physical SI units when solving the optimization problems above tends to lead to numerical instabilities. In this appendix we summarize the renormalizations we applied when performing our experiments.

Renormalization of the model parameter optimization problem

As noted above, the somatic current surrogate model in eq. (17) has one superfluous degree of freedom in the parameter space. When solving for parameters, one of the six parameters \( b_0, b_1, b_2, a_0, a_1, a_2 \) has to be held constant; in other words, there are six different ways of solving for model parameters. As mentioned in the main text, setting \( b_1 = 1 \), i.e., expressing the other parameters relative to effect of the excitatory channel on the input current, is the most numerically stable of these variants.

We apply a set of rescaling factors to the parameters such that all values in the resulting QP matrices have approximately the same order of magnitude. These factors are \( \gamma = 10^9 \) (rescaling currents to nA) and \( \delta = 10^6 \) (rescaling conductances to pS). Now, the optimization problem is

\[
\min_{\bar{b}_0, \bar{b}_2, a'_0, a'_1, a'_2, J_0, \neq J_0} \sum_i \left( b_0 + \delta b'_2 g_i - \gamma J_i a'_0 - \gamma \delta J_i g_k a'_1 - \gamma \delta J_i g_i a'_2 - \delta g_k^2 \right)^2 , \quad \text{where} \quad J_i = G^{-1}[g_k^i \cdot g_k^i].
\]

The final model parameters are then given as

\[
b_0 = \frac{1}{\delta} \bar{b}_0, \quad b_1 = 1, \quad b_2 = \frac{1}{\delta} \bar{b}_2, \quad a_0 = \frac{\gamma}{\delta} \bar{a}_0, \quad a_1 = \gamma \bar{a}_1, \quad a_2 = \gamma a_2.
\]

Renormalization of the synaptic weight optimization problem

When using biologically plausible parameters and units, such as those listed in Table S4 solving the quadratic program in eq. (21) may be numerically unstable, though the problem is not necessarily ill-conditioned. This is because unit-wise the matrix \( \mathbf{X}_i \) and the vector \( \mathbf{b}_i \) defined in eq. (20) describe voltage differentials in the millivolt range, and currents in the pico- to nanoampere range, respectively. This can be mitigated by rescaling the synaptic weights and renormalizing the somatic nonlinearity parameters.

Consider the parameterization of the two-compartment LIF nonlinearity in eq. (17). For arbitrary \( a_0 \) the vector \( \mathbf{b}_i \) is now given as \( \mathbf{b}_i = b_0 - a_0 \mathbf{J}_i \). As \( g_E \) and \( g_i \) are linear in the synaptic weights \( \mathbf{w}_E, \mathbf{w}_i \), multiplying \( b_1, b_2, a_1, a_2 \) by a common factor \( \delta \) will scale the resulting synaptic weights by the inverse \( \delta^{-1} \). Since \( H \) in the above parametrization has a superfluous degree of freedom, rescaling all parameters at the same time does not change the resulting function value. We can thus scale the parameters such that \( b_1 \) is one (instead of \( a_0 \)). Our new model parameters \( b'_0, b'_1, b'_2, a'_0, a'_1, a'_2 \) are given as

\[
b'_0 = \frac{b_0}{\delta b_1}, \quad b'_1 = \frac{\delta b_1}{\delta b_1} = 1, \quad b'_2 = \frac{\delta b_2}{\delta b_1} = \frac{b_2}{b_1}, \quad a'_0 = \frac{a_0}{\delta b_1}, \quad a'_1 = \frac{\delta a_1}{\delta b_1} = \frac{a_1}{b_1}, \quad a'_2 = \frac{\delta a_2}{\delta b_1} = \frac{a_2}{b_1}.
\]

Of course, when using these model parameters, the synaptic weights computed by the optimizer must be scaled by \( \delta \) to obtain biologically plausible values; we use \( \delta = 1 \times 10^{-9} \) in our experiments.

As can be seen when combining the updated parameters with eq. (20), this normalization is equivalent to rescaling \( \mathbf{X}_i \) and \( \mathbf{b}_i \), where

\[
\mathbf{X}'_i = \frac{1}{b_1} \mathbf{X}_i, \quad \mathbf{b}'_i = \frac{1}{b_1} \mathbf{b}_i.
\]

To ensure equivalence between the original and normalized problem, the regularization factor in eq. (21) must be divided by the square of the scaling factor common to \( \mathbf{X}'_i \) and \( \mathbf{b}'_i \). Correspondingly, the new regularization factor of the normalized optimization problem is \( \lambda' = \lambda \cdot (b_1)^{-2} \).

In our experiments.

\[ 9 \quad \text{9} \]

\[ \Rightarrow \quad \Rightarrow \]

\[ \Rightarrow \quad \Rightarrow \]

\[ \Rightarrow \quad \Rightarrow \]

\[ \Rightarrow \quad \Rightarrow \]

\[ \Rightarrow \quad \Rightarrow \]

\[ \Rightarrow \quad \Rightarrow \]
### S3 Table: Neuron and synaptic model parameters

Table S3. Neuron and synaptic model parameters. These are the neuron and synapse model parameters used in the experiments, unless explicitly indicated otherwise.

| Parameter                  | Symbol | Value   | Parameter                  | Symbol | Value   |
|----------------------------|--------|---------|----------------------------|--------|---------|
| **Superthreshold dynamics**|        |         | **Reversal potentials**    |        |         |
| Threshold potential        | $v_{th}$           | −50 mV  | Exc. reversal potential    | $E_R$  | 20 mV   |
| Spike potential            | $v_{spike}$        | 20 mV   | Resting potential          | $E_L$  | −65 mV  |
| Refractory period          | $\tau_{ref}$       | 2 ms    | Inh. reversal potential    | $E_I$  | −75 mV  |
| Spike period               | $\tau_{spike}$     | 1 ms    |                            |        |         |
| Reset potential            | $v_{reset}$        | −65 mV  |                            |        |         |
| **Synaptic time constants**|        |         | **Conductances**           |        |         |
| Exc. synapse time constant | $\tau_{syn,E}$     | 5 ms    | Coupling conductance       | $g_C$  | 100 nS  |
| Inh. synapse time constant | $\tau_{syn,I}$     | 10 ms   | Leak conductances          | $g_{L,1}, g_{L,2}$ | 50 nS |
| Membrane capacitances      | $C_{m,1}, C_{m,2}$ | 1 nF    |                            |        |         |

*Note: The table above is a simplified representation and does not include all possible parameters.*
### S4 Table: Current model parameters before and after fitting to empirical data

**Table S4. Current model parameters before and after fitting to empirical data.** The parameters $b_0$, $b_1$, $b_2$, $a_0$, $a_1$, $a_2$ define the dendritic current model $H$ in eq. (16). $J_{\text{max}}$ and $J_{\text{min}}$ are the predicted absolute maximum/minimum somatic current. The parameters $\alpha$, $\beta$ characterize the rectified linear unit used as neuron nonlinearity $G[J]$ in the spike noise experiment.

| Symbol | Unit | Before fitting parameters | After fitting model parameters |
|--------|------|----------------------------|--------------------------------|
|        |      | gc                         | gc                            |
|        |      | Theoretical estimate       | Constant conductances         |
|        |      | 50 nS 100 nS 200 nS        | 50 nS 100 nS 200 nS           |
|        |      |                             | With spike noise              |
|        |      | 50 nS 100 nS 200 nS        | gc                            |
| $b_0$  | µS   | $-4.8$ $-4.8$ $-4.8$       | $-19.5$ $-18.8$ $-17.1$      |
| $b_1$  | 1000.0 | $1000.0$ $1000.0$ $1000.0$ | $1000.0$ $1000.0$ $1000.0$ |
| $b_2$  | mV$^{-1}$ | $-225.8$ $-225.8$ $-225.8$ | $-425.5$ $-376.0$ $-352.3$ |
| $a_0$  | nA$^{-1}$ | $258.1$ $129.0$ $64.5$     | $296.4$ $193.0$ $113.3$      |
| $a_1$  | nA$^{-1}$ | $258.1$ $129.0$ $64.5$     | $322.2$ $41.6$ $18.6$        |
| $J_{\text{max}}$ | nA | $3.9$ $7.8$ $15.5$ | $3.4$ $5.2$ $8.8$ |
| $J_{\text{min}}$ | nA | $-0.9$ $-1.8$ $-3.5$ | $-3.2$ $-9.0$ $-18.9$ |
| $\alpha$ | s$^{-1}$ nA$^{-1}$ | / / / | / / / |
| $\beta$  | s$^{-1}$ | / / / | / / / | 51.3 51.5 51.3 |
S5 Table: List of multivariate functions analyzed in the network experiment

Table S5. List of multivariate functions analyzed in the network experiment. The functions \( f(x, y) \) are scaled such that they map onto the interval \([0, 1]\) for \((x, y) \in [0, 1]^2\). Contour plots and RMS values are over the input domain \([0, 1]^2\) (axis labels omitted due to space constraints); purple (dark) corresponds to a value of zero, yellow (light) to a value of one.

| Name                  | Function          | RMS          | Contour |
|-----------------------|-------------------|--------------|---------|
| Addition              | \( f(x, y) = \frac{1}{2}(x + y) \) | \( f_{\text{RMS}} = \frac{\sqrt{12}}{12} \approx 0.5401 \) | ![Contour Plot](image1.png) |
| Multiplication        | \( f(x, y) = xy \) | \( f_{\text{RMS}} = \frac{1}{3} \approx 0.3333 \) | ![Contour Plot](image2.png) |
| Multiplication with square root | \( f(x, y) = \sqrt{xy} \) | \( f_{\text{RMS}} = \frac{1}{2} = 0.5 \) | ![Contour Plot](image3.png) |
| Squared multiplication | \( f(x, y) = x^2y^2 \) | \( f_{\text{RMS}} = \frac{1}{5} = 0.2 \) | ![Contour Plot](image4.png) |
| Controlled shunting   | \( f(x, y) = \frac{1 + x}{2(1 + y)} \) | \( f_{\text{RMS}} = \frac{\sqrt{12}}{12} \approx 0.5401 \) | ![Contour Plot](image5.png) |
| Norm                  | \( f(x, y) = \sqrt{\frac{1}{2}(x^2 + y^2)} \) | \( f_{\text{RMS}} = \frac{1}{\sqrt{3}} \approx 0.5774 \) | ![Contour Plot](image6.png) |
| Arctan                | \( f(x, y) = \frac{2}{\pi} \arctan \left( \frac{y}{x} \right) \) | \( f_{\text{RMS}} \approx 0.5672 \) | ![Contour Plot](image7.png) |
| Maximum               | \( f(x, y) = \max\{x, y\} \) | \( f_{\text{RMS}} = \frac{1}{\sqrt{2}} \approx 0.7071 \) | ![Contour Plot](image8.png) |
S6 Appendix: Supplemental spiking network experiment results

Complete spiking neural network results

Table S6A shows an extended version of the spiking network experiment results from Table 1. This includes the average model error $E_{\text{model}}$, i.e., the approximation error under the assumption that the model equations perfectly describe the network without any noise. Furthermore, we include results for the two-compartment model with coupling conductance $g_c = 200 \, \text{nS}$. In our experiment, this particular model is only competitive when computing multiplication (i.e., the target function for which the regularization parameters have been optimized; see Fig S6A) or the square root of multiplication, and only when taking noise into account during model parameter optimization.

Pre-filter experiments

As mentioned in the main text, the dendritic compartment in the two-compartment LIF neuron model can be interpreted as an additional first order low-pass filter with time constant $C_m g_L^{-1}$. For the parameters used in the above experiments (cf. Table S3), the time constant of this filter is 20 ms. This low-pass filter significantly reduces the amplitude of high-frequency transients such as spike noise in the input signal and may thus significantly contribute to the reduction of the final approximation error $E_{\text{net}}$. However, due to the coupling between the somatic and dendritic compartment, noise originating from the neuron’s action potentials is fed back into the dendritic compartment, increasing the noise floor and thus countering some of the potentially beneficial low-pass filter effects.

Separating the effects of the dendritic nonlinearity and the dendritic dynamics is non-trivial. To explore the effects of low-pass filters in the network, we performed an additional experiment in which we introduced a first-order low-pass filter (the “pre-filter”) located at the input of the target population (for single-layer networks) or the input of the intermediate population (in the case of the two-layer network). Notably, this low-pass filter is not included in the target signal filter-chain. Hence, the chosen filter time-constant, must balance between suppressing noise in the input and not dampening high-frequency components in the target signal.

We tuned both the regularization parameter $\sigma$ and the pre-filter time constant $\tau$ such that the network error $E_{\text{net}}$ is minimized when approximating multiplication (cf. Fig S6A). For standard LIF neurons, the error is minimized when selecting $\tau = 36.5 \, \text{ms}$. For two-compartment LIF neurons, the error is minimized when selecting $\tau = 23.7 \, \text{ms}$, or $\tau = 11.6 \, \text{ms}$ (smaller for larger $g_c$) when assuming the presence of noise in the model parameters. Function approximation errors are given in (cf. Table S6B).

In most cases, the final network error is closer to the predicted approximation error $E_{\text{model}}$. In particular when computing multiplication, the standard LIF model network approximation error is reduced from $21.5 \pm 4.0\%$ to $14.7 \pm 4.4\%$, whereas the error for the best two-compartment model ($g_c = 50 \, \text{nS}$, with noise model) is reduced from $11.9 \pm 1.8\%$ to $10.9 \pm 2.1\%$. However, note that adding a low-pass filter in the case of the two-compartment neuron increases the error for other target functions, since $\tau$, $\sigma$ were not optimized for those.

To conclude, a portion of the reduction in approximation error when using the two-compartment neuron model can be attributed to low-pass filter effects in the dendritic compartment. However, and in particular for $g_c = 50 \, \text{nS}$, a significant computational benefit remains when using the two-compartment model with conductance-based synapses.
Table S6A. Spiking neural network function approximations. Error values correspond to the normalised RMSE (relative to the RMS/standard deviation of the target function). $E_{\text{model}}$ is the static (i.e. without spike noise) decoding error assuming the dendritic and somatic models are accurate. $E_{\text{net}}$ is the decoding error of the target population when feeding the input through the entire network.

| Experiment setup | Target Functions | x + y | x × y | $\sqrt{x \times y}$ | $(x \times y)^2$ | x/(1 + y) | $\| (x, y) \|$ | atan(x, y) | max(x, y) |
|------------------|------------------|------|-------|---------------------|----------------|----------|------------|----------|----------|
| **Standard LIF** | no relaxation    | $E_{\text{model}}$ | 1.2 ± 0.3% | 27.0 ± 0.1% | 15.0 ± 0.2% | 45.9 ± 0.2% | 4.3 ± 0.1% | 6.9 ± 0.1% | 12.5 ± 0.2% | 15.0 ± 0.3% |
|                  | $E_{\text{net}}$ | 5.1 ± 0.6% | 26.2 ± 0.4% | 14.1 ± 0.4% | 44.5 ± 0.6% | 6.0 ± 0.4% | 8.0 ± 0.4% | 10.3 ± 0.3% | 14.9 ± 0.3% |
|                  | standard         | $E_{\text{model}}$ | 1.4 ± 0.6% | 15.7 ± 5.4% | 12.7 ± 4.1% | 29.8 ± 7.2% | 2.2 ± 0.5% | 3.2 ± 1.2% | 9.6 ± 1.0% | 10.1 ± 1.5% |
|                  |                  | $E_{\text{net}}$ | 5.5 ± 1.1% | 21.5 ± 6.6% | 19.7 ± 6.1% | 33.0 ± 6.6% | 5.2 ± 0.7% | 5.7 ± 1.1% | 8.6 ± 1.0% | 10.0 ± 0.9% |
|                  | two-layer        | $E_{\text{model}}$ | 0.7 ± 0.3% | 2.8 ± 1.1% | 5.1 ± 1.6% | 5.4 ± 2.3% | 0.8 ± 0.3% | 0.9 ± 0.3% | 6.3 ± 0.7% | 2.0 ± 0.6% |
|                  |                  | $E_{\text{net}}$ | 11.0 ± 1.3% | 15.4 ± 4.0% | 16.3 ± 3.0% | 18.7 ± 6.7% | 9.5 ± 0.8% | 10.5 ± 1.0% | 13.4 ± 1.1% | 11.3 ± 1.4% |
| **Two comp. LIF gC = 50 nS** | standard | $E_{\text{model}}$ | 0.8 ± 0.3% | 11.4 ± 2.0% | 6.1 ± 1.1% | 28.1 ± 3.6% | 0.8 ± 0.3% | 1.0 ± 0.3% | 4.9 ± 0.9% | 4.7 ± 0.8% |
|                  |                  | $E_{\text{net}}$ | 3.2 ± 1.1% | 13.9 ± 2.9% | 9.7 ± 2.6% | 27.7 ± 4.1% | 3.4 ± 1.0% | 3.1 ± 1.3% | 5.8 ± 1.3% | 5.5 ± 0.9% |
|                  | noise model      | $E_{\text{model}}$ | 1.0 ± 0.2% | 12.7 ± 1.7% | 7.1 ± 0.8% | 28.2 ± 3.1% | 2.8 ± 0.9% | 1.8 ± 0.3% | 5.9 ± 0.6% | 5.9 ± 0.4% |
|                  |                  | $E_{\text{net}}$ | 9.1 ± 1.2% | 11.9 ± 1.8% | 7.1 ± 1.0% | 27.4 ± 4.1% | 10.0 ± 1.6% | 8.9 ± 1.2% | 8.4 ± 1.0% | 7.7 ± 0.9% |
| **Two comp. LIF gC = 100 nS** | standard | $E_{\text{model}}$ | 0.9 ± 0.3% | 15.8 ± 2.7% | 7.5 ± 1.5% | 34.6 ± 3.8% | 1.3 ± 0.4% | 1.6 ± 0.6% | 5.6 ± 1.1% | 6.0 ± 0.9% |
|                  |                  | $E_{\text{net}}$ | 5.1 ± 1.2% | 18.2 ± 4.0% | 13.3 ± 4.2% | 34.3 ± 5.3% | 5.3 ± 1.3% | 4.3 ± 1.8% | 7.0 ± 1.2% | 7.3 ± 0.9% |
|                  | noise model      | $E_{\text{model}}$ | 0.8 ± 0.2% | 13.4 ± 1.8% | 6.9 ± 0.8% | 30.2 ± 3.2% | 2.2 ± 0.7% | 1.7 ± 0.3% | 5.6 ± 0.6% | 6.0 ± 0.4% |
|                  |                  | $E_{\text{net}}$ | 11.5 ± 1.3% | 14.2 ± 2.1% | 8.9 ± 1.7% | 30.3 ± 4.3% | 14.0 ± 1.9% | 12.3 ± 1.8% | 12.7 ± 1.6% | 9.7 ± 1.0% |
| **Two comp. LIF gC = 200 nS** | standard | $E_{\text{model}}$ | 0.8 ± 0.4% | 15.7 ± 3.6% | 7.6 ± 1.9% | 32.7 ± 4.6% | 1.8 ± 0.5% | 2.2 ± 1.0% | 6.4 ± 1.3% | 7.4 ± 0.9% |
|                  |                  | $E_{\text{net}}$ | 7.7 ± 1.4% | 23.3 ± 6.8% | 19.5 ± 5.8% | 35.4 ± 8.0% | 7.4 ± 1.2% | 6.5 ± 1.2% | 9.2 ± 1.5% | 9.3 ± 0.8% |
|                  | noise model      | $E_{\text{model}}$ | 0.7 ± 0.2% | 14.5 ± 1.7% | 7.6 ± 0.8% | 31.1 ± 3.1% | 1.8 ± 0.5% | 1.6 ± 0.2% | 5.7 ± 0.6% | 6.2 ± 0.4% |
|                  |                  | $E_{\text{net}}$ | 12.1 ± 1.8% | 15.5 ± 2.1% | 10.2 ± 2.5% | 32.0 ± 3.9% | 16.6 ± 3.0% | 14.0 ± 2.6% | 16.9 ± 2.5% | 12.5 ± 1.2% |
Fig S6A. Regularization and pre-filter parameter sweep. Sweep over the regularization parameter $\lambda$ in terms of $\lambda = (a_{\text{max}} \sigma)^2 = (100 \sigma)^2$ and the pre-filter first-order lowpass filter time-constant $\tau$ for various network setups. Results are based on a $33 \times 32$ grid with an average of eight repetitions per point. Contour plots show the logarithmic network error $\log_{10}(E_{\text{net}})$ when computing multiplication $f(x, y) = xy$ over $(x, y) \in [0, 1]^2$. Points marked with $\text{A}$ correspond to the regularization factor $\sigma$ minimizing the error for a small or zero pre-filter time constant $\tau$. Points marked with $\text{B}$ correspond to the $\sigma, \tau$ pair minimizing the error when using a pre-filter.
Table S6B. Spiking neural network function approximations with pre-filter. Error values correspond to the normalised RMSE (relative to the RMS/standard deviation of the target function). $E_{\text{model}}$ is the static (i.e. without spike noise) decoding error assuming the dendritic and somatic models are accurate. $E_{\text{net}}$ is the decoding error of the target population when feeding the input through the entire network.

| Experiment setup | Target Functions | $x + y$ | $x \times y$ | $\sqrt{x \times y}$ | $(x \times y)^2$ | $x/(1 + y)$ | $\| (x, y) \|$ | atan($x, y$) | max($x, y$) |
|------------------|------------------|---------|-------------|-----------------|----------------|------------|-----------|-------------|-------------|
| **Standard LIF** | $E_{\text{model}}$ | $1.0 \pm 0.3\%$ | $26.9 \pm 0.1\%$ | $14.7 \pm 0.2\%$ | $45.9 \pm 0.2\%$ | $4.3 \pm 0.1\%$ | $6.9 \pm 0.1\%$ | $12.3 \pm 0.2\%$ | $15.0 \pm 0.3\%$ |
|                  | $E_{\text{net}}$  | $4.3 \pm 0.9\%$ | $26.3 \pm 0.6\%$ | $13.3 \pm 0.5\%$ | $45.1 \pm 1.1\%$ | $6.2 \pm 0.9\%$ | $7.6 \pm 0.7\%$ | $10.6 \pm 0.7\%$ | $14.7 \pm 0.6\%$ |
| **Two comp. LIF** | $E_{\text{model}}$ | $1.2 \pm 0.5\%$ | $15.5 \pm 5.6\%$ | $12.0 \pm 4.0\%$ | $29.8 \pm 7.5\%$ | $2.2 \pm 0.5\%$ | $3.1 \pm 1.2\%$ | $9.3 \pm 1.1\%$ | $10.1 \pm 1.5\%$ |
| **LIF** $g_C = 50\text{nS}$ | $E_{\text{net}}$  | $4.2 \pm 0.8\%$ | $14.7 \pm 4.4\%$ | $9.2 \pm 3.0\%$ | $28.5 \pm 5.9\%$ | $5.5 \pm 1.1\%$ | $4.9 \pm 1.1\%$ | $7.2 \pm 1.0\%$ | $9.7 \pm 1.3\%$ |
|                  | $E_{\text{model}}$ | $0.9 \pm 0.4\%$ | $3.7 \pm 1.5\%$ | $6.8 \pm 2.1\%$ | $6.4 \pm 2.9\%$ | $1.1 \pm 0.3\%$ | $1.3 \pm 0.4\%$ | $7.6 \pm 0.7\%$ | $2.6 \pm 0.6\%$ |
|                  | $E_{\text{net}}$  | $7.8 \pm 1.3\%$ | $9.5 \pm 2.2\%$ | $9.9 \pm 2.4\%$ | $12.9 \pm 3.9\%$ | $9.4 \pm 1.3\%$ | $8.1 \pm 1.1\%$ | $10.4 \pm 0.9\%$ | $8.9 \pm 0.5\%$ |
| **Two comp. LIF** | $E_{\text{model}}$ | $0.6 \pm 0.3\%$ | $6.4 \pm 1.9\%$ | $4.5 \pm 1.1\%$ | $14.4 \pm 4.9\%$ | $0.5 \pm 0.2\%$ | $0.7 \pm 0.3\%$ | $3.6 \pm 1.1\%$ | $3.9 \pm 0.7\%$ |
| **LIF** $g_C = 100\text{nS}$ | $E_{\text{net}}$  | $3.9 \pm 1.7\%$ | $11.2 \pm 4.1\%$ | $6.2 \pm 2.0\%$ | $21.2 \pm 6.9\%$ | $4.7 \pm 1.8\%$ | $3.9 \pm 2.2\%$ | $6.4 \pm 2.6\%$ | $4.8 \pm 1.5\%$ |
|                  | $E_{\text{model}}$ | $0.8 \pm 0.2\%$ | $8.6 \pm 1.9\%$ | $5.7 \pm 0.9\%$ | $18.9 \pm 4.1\%$ | $2.7 \pm 0.9\%$ | $1.4 \pm 0.3\%$ | $4.7 \pm 0.7\%$ | $5.0 \pm 0.3\%$ |
|                  | $E_{\text{net}}$  | $10.7 \pm 1.3\%$ | $10.9 \pm 2.1\%$ | $7.2 \pm 1.0\%$ | $19.7 \pm 5.1\%$ | $11.9 \pm 1.9\%$ | $10.4 \pm 1.5\%$ | $12.0 \pm 1.5\%$ | $8.7 \pm 1.2\%$ |
| **Two comp. LIF** | $E_{\text{model}}$ | $0.5 \pm 0.2\%$ | $8.1 \pm 2.2\%$ | $4.6 \pm 1.0\%$ | $20.5 \pm 4.9\%$ | $0.5 \pm 0.2\%$ | $0.7 \pm 0.2\%$ | $3.6 \pm 0.9\%$ | $4.1 \pm 0.7\%$ |
| **LIF** $g_C = 200\text{nS}$ | $E_{\text{net}}$  | $4.6 \pm 2.5\%$ | $14.6 \pm 4.3\%$ | $8.4 \pm 2.9\%$ | $28.3 \pm 7.4\%$ | $7.1 \pm 4.2\%$ | $5.8 \pm 4.7\%$ | $12.3 \pm 5.9\%$ | $5.8 \pm 2.5\%$ |
|                  | $E_{\text{model}}$ | $0.9 \pm 0.2\%$ | $10.9 \pm 2.0\%$ | $5.8 \pm 0.8\%$ | $23.3 \pm 4.2\%$ | $2.2 \pm 0.7\%$ | $1.4 \pm 0.3\%$ | $4.8 \pm 0.6\%$ | $5.3 \pm 0.3\%$ |
|                  | $E_{\text{net}}$  | $14.4 \pm 1.8\%$ | $14.3 \pm 2.7\%$ | $8.7 \pm 1.1\%$ | $27.5 \pm 6.3\%$ | $17.7 \pm 3.3\%$ | $15.3 \pm 2.6\%$ | $19.0 \pm 2.8\%$ | $12.2 \pm 1.4\%$ |
| **Two comp. LIF** | $E_{\text{model}}$ | $0.5 \pm 0.2\%$ | $10.1 \pm 2.4\%$ | $5.1 \pm 1.1\%$ | $24.9 \pm 4.1\%$ | $0.5 \pm 0.2\%$ | $0.8 \pm 0.2\%$ | $4.0 \pm 0.9\%$ | $4.4 \pm 0.8\%$ |
| **LIF** $g_C = 200\text{nS}$ | $E_{\text{net}}$  | $5.0 \pm 2.9\%$ | $16.5 \pm 3.7\%$ | $10.1 \pm 3.3\%$ | $32.0 \pm 6.1\%$ | $10.5 \pm 6.8\%$ | $7.4 \pm 6.6\%$ | $17.4 \pm 7.5\%$ | $7.3 \pm 3.0\%$ |
|                  | $E_{\text{model}}$ | $0.7 \pm 0.2\%$ | $13.5 \pm 1.8\%$ | $7.3 \pm 0.8\%$ | $29.4 \pm 3.3\%$ | $1.8 \pm 0.5\%$ | $1.5 \pm 0.2\%$ | $5.4 \pm 0.6\%$ | $5.9 \pm 0.4\%$ |
|                  | $E_{\text{net}}$  | $13.4 \pm 2.1\%$ | $15.5 \pm 2.4\%$ | $9.6 \pm 1.8\%$ | $31.8 \pm 5.1\%$ | $18.3 \pm 3.9\%$ | $15.3 \pm 3.0\%$ | $18.9 \pm 3.1\%$ | $13.9 \pm 1.3\%$ |