Nonlinear synaptic interaction as a computational resource in the Neural Engineering Framework

Andreas Stöckel, Aaron R. Voelker, Chris Eliasmith | {astoecke, arvoelke, celiasmith}@uwaterloo.ca
Centre for Theoretical Neuroscience, University of Waterloo | http://ctn.uwaterloo.ca/

Motivation ①

Approximate **arbitrary, nonlinear, multivariate functions** directly in the **dendritic tree** by exploiting nonlinearities exposed by conductance-based synapses

Exploiting synaptic nonlinearity ④

1. Decouple synaptic nonlinearity H from somatic nonlinearity G by applying its inverse G^{-1}

$$\not{G}[J] = \not{G}[H(g_E^1(x) + g_E^2(y), g_I^1(x) + g_I^2(y))]$$

2. Given the target function $\phi(x, y)$ and the current mapping from the NEF ② we know

$$J[\phi(x,y)] = H[\vec{w}_E^1\,\vec{a}^1(x) + \vec{w}_E^2\,\vec{a}^2(y), \vec{w}_I^1\,\vec{a}^1(x) + \vec{w}_I^2\,\vec{a}^2(y)]$$

Input current; *x-activities*; *y-activities*; *non-negative excitatory/inhibitory synaptic weights*

3. Synaptic weights encoding $\phi(x, y)$ can be found by solving the above equation for $\vec{w}_E, \vec{w}_I$

- For two-compartment, conductance-based LIF, H is a rational function [2]

$$J = H(g_E, g_I) = \frac{a_0 + a_1 g_E + a_2 g_I}{b_0 + b_1 g_E + b_2 g_I}$$

- Finding non-negative $\vec{w}$ is a convex quadratic programming problem; unique, optimal solution guaranteed and can be computed quickly

Neural Engineering Framework (NEF) [1] ②

- **Representation:** Populations represent $\vec{x}$

$$a(\vec{x}) = G[J(\vec{x})] = G[\alpha\langle\vec{e}, \vec{x}\rangle + J_0]$$

Neuron activity; *Somatic input current*; *Gain*; *Encoder*; *Bias current*

- **Transformation:** Connections between neuron populations compute functions $f(\vec{x})$

$$f(\vec{x}) \approx D^f\,\vec{a}(\vec{x})$$

Decoding matrix (synaptic weights); *Population activity*

Results and Conclusion ⑤

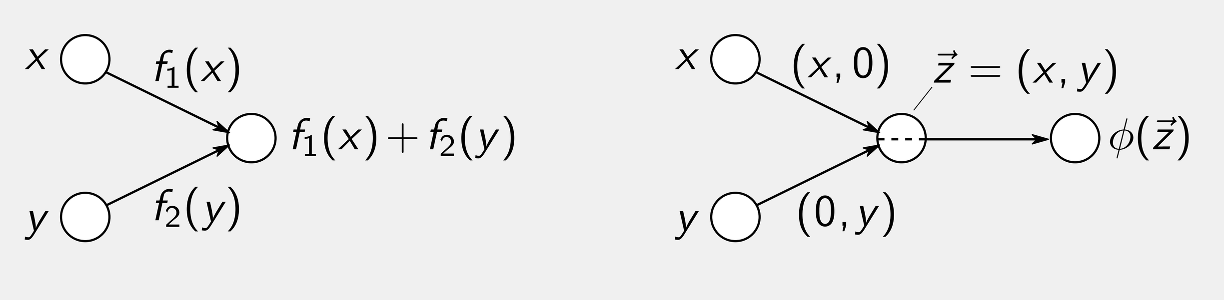
Table 1 Accuracies in spiking simulation for the network setups in ③
Input domain$(x, y) \in [0, 1]^2$. Normalized RMSE between target and decoded value. Values in percent, smaller is better.

Function ϕ	No intermediate, current-based **(a)**	Intermediate, current-based **(b)**	No intermediate, cond.-based **(c)**
$x + y$	**1.44**	2.36	3.08
$x \cdot y$	17.18	**5.93**	7.52
$\Vert(x, y)\Vert$	4.30	**2.78**	3.02
$x / (1 + 10\,y)$	24.20	15.52	**12.12**
$\mathrm{atan}(x, y)$	8.30	9.98	**5.63**
$x \cdot (x > y)$	23.69	21.81	**21.44**

- Conductance-based synapses can replace intermediate populations when approximating multivariate functions

Multivariate functions in feed-forward neural networks ③

Approximating nonlinear, multivariate functions in neural networks requires a middle-layer representing the input variables



(a) Multiple inputs to one population are linearly superimposed
(b) Without nonlinear synapses, an intermediate population is required
(c) Here: exploit nonlinear interaction between excitatory and inhibitory channels

Population; Exc. connection; Linear connection; Inh. connection

References

[1] Eliasmith and Anderson. *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems.* MIT press, 2003.

[2] Koch. *Biophysics of Computation: Information Processing in Single Neurons.* Oxford University Press, 1999.

[3] Vu et al. "The mechanism of tonic inhibition of crayfish escape behavior: distal inhibition and its functional significance". In: *Journal of Neuroscience* 13.10 (1993).

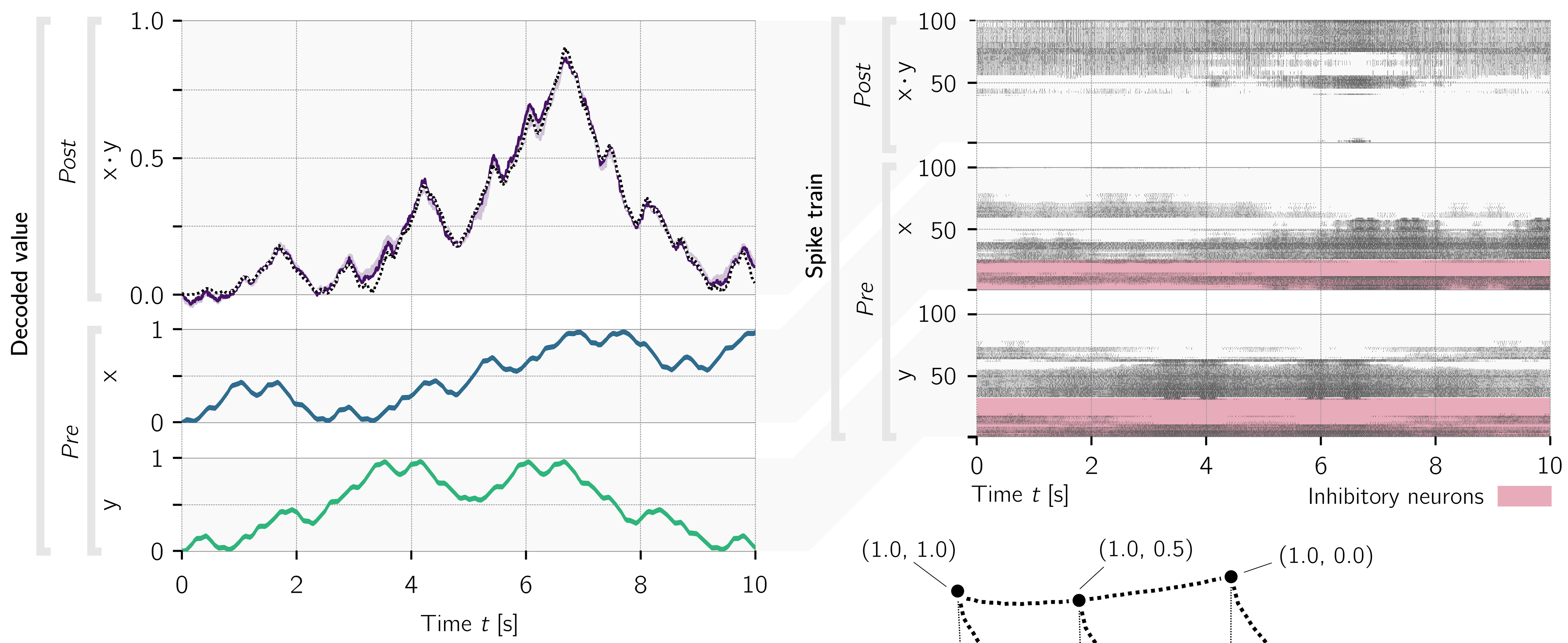
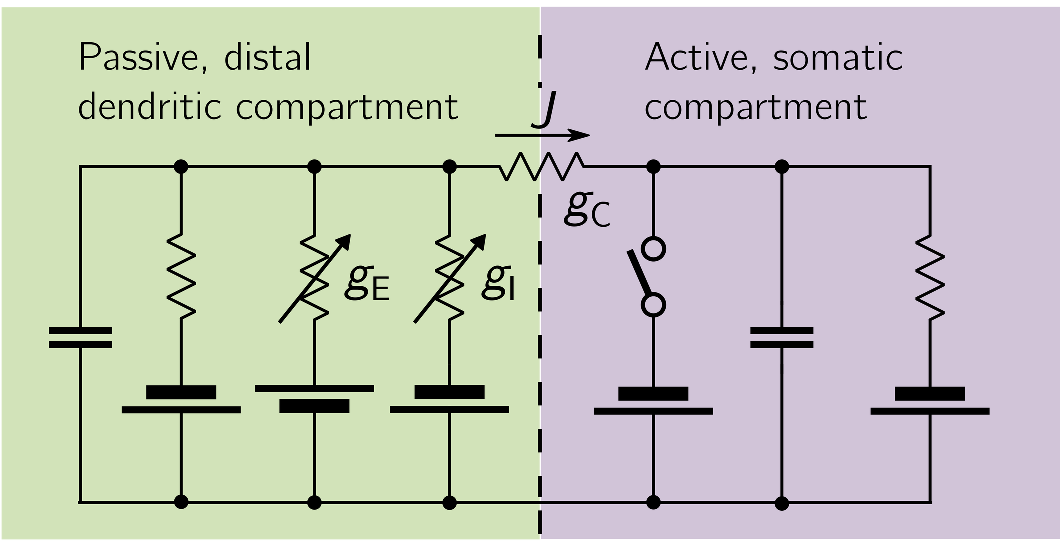


Figure 1 ▲
Synaptic computation of $x \cdot y$
Data over 50 trials

25/75% percentile; Min/max; Target value



◀ **Figure 2**
Equivalent circuit diagram of the two-compartment LIF neuron used in the experiments (adapted from [2, 3])

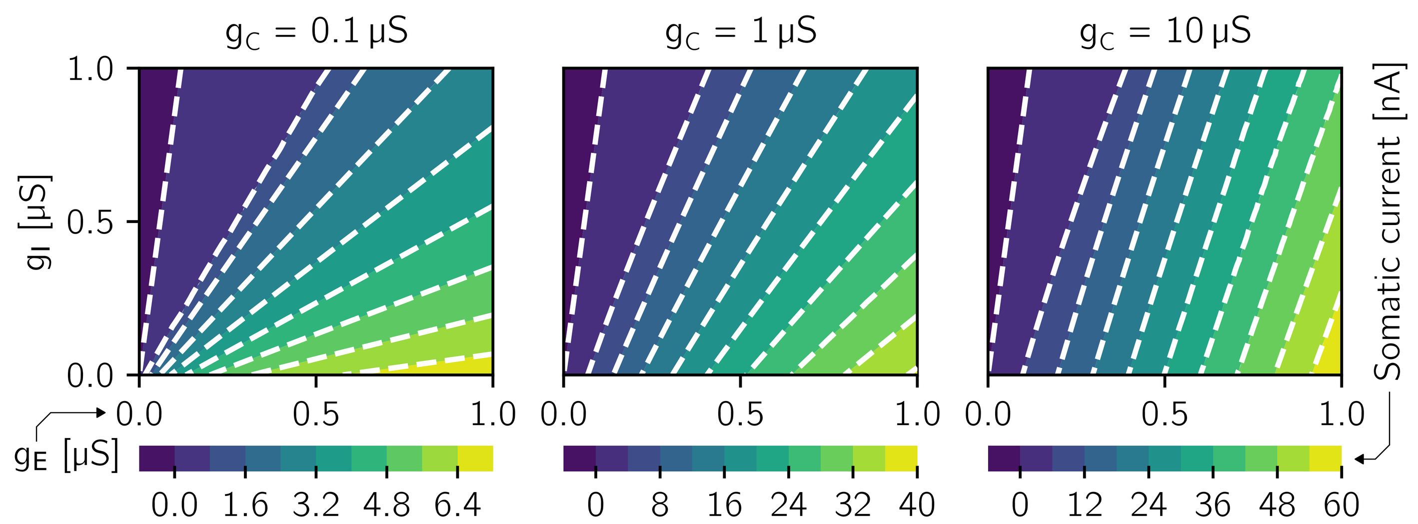
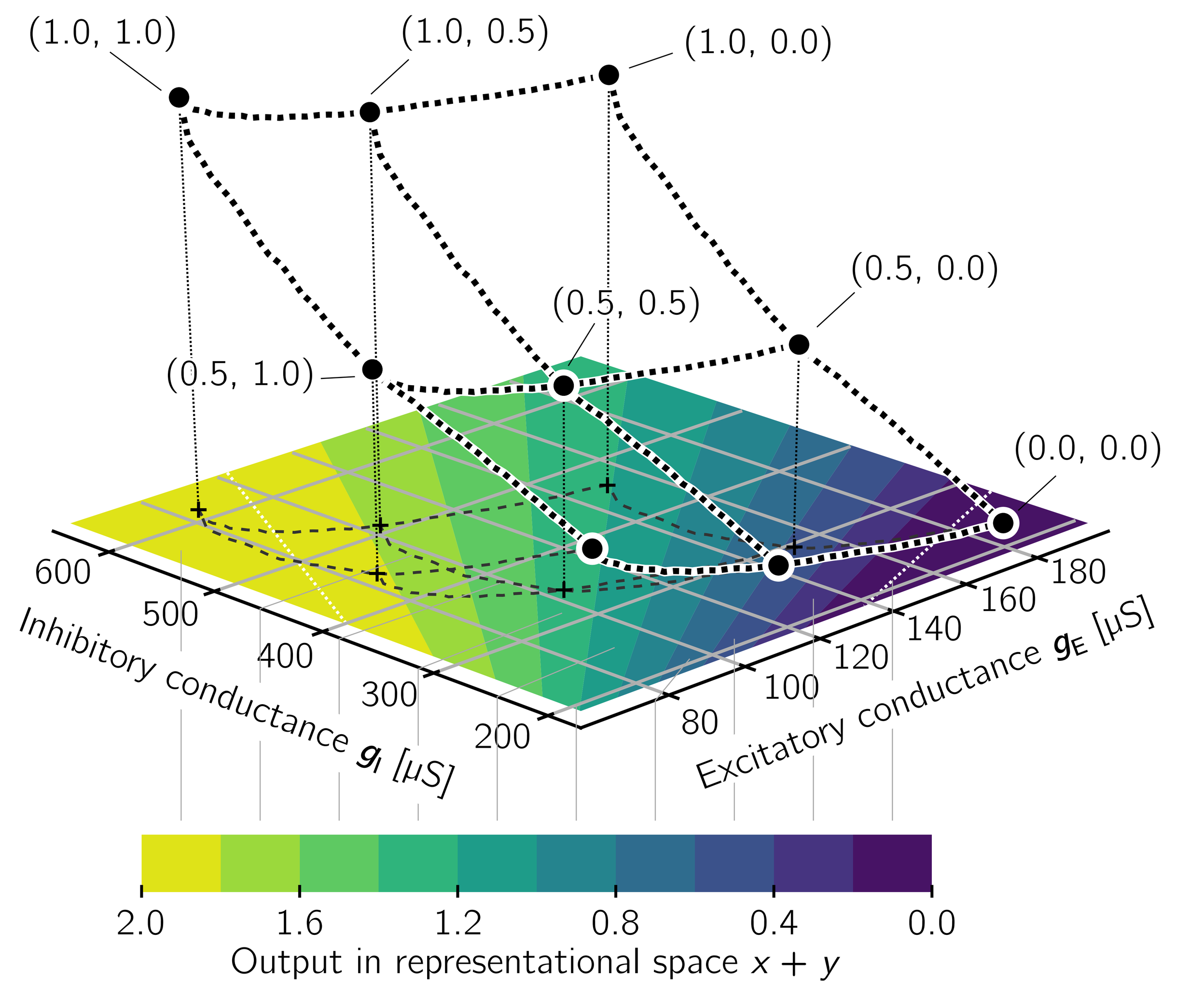


Figure 3 ▶
Influence of g_C on the synaptic nonlinearity



Input in representational space (x, y) ●
Decoded conductances (g_E, g_I) +

Figure 4 ▶
Approximation of the multivariate function $x + y$