Preamble

Figures and tables in this document are often accompanied by a link to the Jupyter Notebook that generated them. These notebooks contain more information than can be conveyed in just the caption. These links look like, e.g: 2021-12-06__local_HH_dV_shape. To visit these links, in the digital version of this document, just click them. In the paper version, convert them to an URL as follows: https://tfiers.github.io/phd/nb/2021-12-06__local_HH_dV_shape.html.

Chapter 1

Introduction

- Problem statement
- Opportunity of voltage imaging, core idea of inverting the causal 'presynaptic spike → postsynaptic PSP' relation
- Overview of thesis

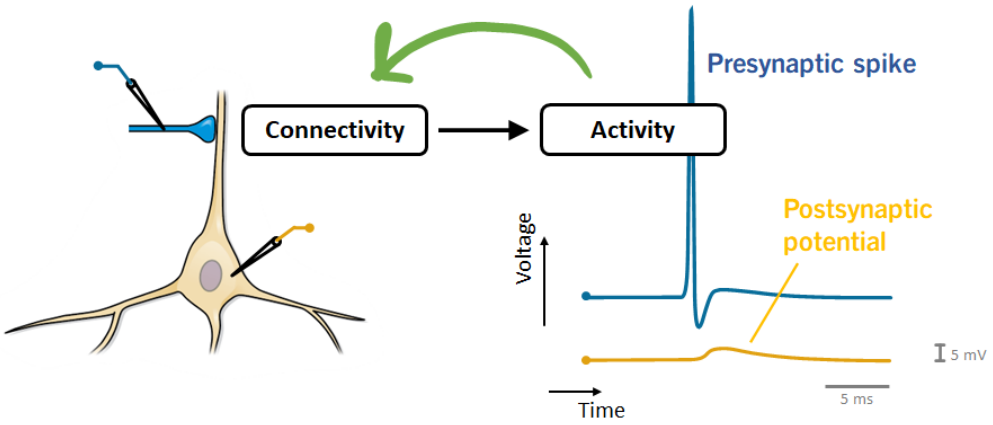

Figure 1: **The causal link between neural connectivity and activity.** On the left, a cartoon of a synaptic connection. The axon of presynaptic neuron M (in blue) impinges on postsynaptic neuron N (in brown). The electrode icons indicate that their membrane voltages are recorded (shown on the right). A succesful spike in neuron M will elicit a small but precisely-timed voltage bump in neuron N (the postsynaptic potential, PSP). There is thus a causal relationship between 1) the existence of a connection $M \to N$ and 2) both neurons' membrane voltages. This causal relationship (black arrow) is exploited to perform network inference from voltage recordings (green arrow).
Drawings adapted from Purves et al.'s "Neuroscience" textbook, 6th edition, 2018.

1.1 Connectomics

- Motivation
- Ground-truth connectomics: tracing of electron microscopy and fluoresent injection imaging
- Necessity of connection *inference*
- Mention 'invasive' connection testing (stimulate one cell, record possible neighbours)
- Limitations of 'connectomics', and of inferred vs 'actual' connectomics.
  Terminology, e.g. 'functional connectomics'

1.2 Voltage imaging

- Technologies (from dyes to GEVIs)
- Specs: cell yield, tissue depth, recording duration, SNR, species
- ..and growth of these over time, and comparison with calcium imaging.
  To extrapolate how these might advance in the future
- Comparison with other recording techniques: ephys, calcium imaging, (and briefly mention coarser methods)

1.3 Network inference

- Working with events/spikes only, versus working with continuous signals; or a hybrid as here.
- Overview of the spikes-only methods
- The connectomics competition, and the findings about the best methods
- Mention other application domains, like gene regulatory networks
- Conclusion: connection inference by spikes is poor (Ila Fiete etc), and Vm imaging offers unique advantage (causality)

Chapter 2

Simulation details

In this chapter, we decribe our experimental setup: the neuron model we simulate, its inputs, and how we simulate voltage imaging.

2.1 The AdEx neuron

We choose to simulate the 'AdEx' point neuron model, or the 'adaptive exponential integrate-and-fire' neuron[BG05], with conductance-based synaptic currents. The AdEx neuron is a leaky-integrate-and-fire (LIF) neuron model, with two additions. First, the full upstroke of each spike is simulated, as an exponential runoff. Second, an extra dynamic variable is added: the adaptation current. This current allows the simulation of many non-linear effects of real neurons, like spike-rate adaptation and post-inhibitory rebound.

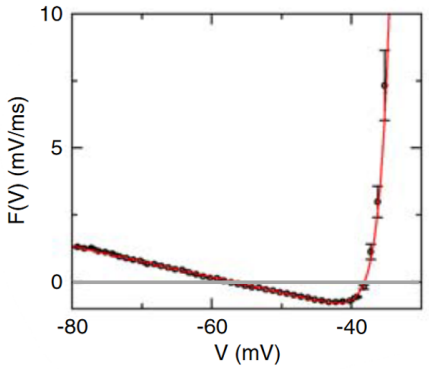

Figure 2: A linear-plus-exponential model (red) fit to data from a cortical pyramidal neuron (black), from [Bad+08]

The AdEx model consists of two differential equations (1 and 2), and a discontinuous update after a spike is generated (3). One equation simulates the membrane voltage V, and one the adaptation current w:

$$C\frac{\mathrm{d}V}{\mathrm{d}t} = -g_L(V - E_L) + g_L\Delta_T \exp\left(\frac{V - V_T}{\Delta_T}\right) - I_{\text{syn}} - w \quad (1)$$

$$\tau_w\frac{\mathrm{d}w}{\mathrm{d}t} = a(V - E_L) - w \quad (2)$$

Parameters are described in table 1.

The first term of equation (1) is the restorative force pulling the voltage back to the resting (or leak) potential E_L. The second, exponential term is what generates the spike upstrokes. It is practically zero over most of the sub-threshold regime, and only becomes large (and then very large) near the firing threshold. (This firing threshold is characterised further on).

I_{syn} is the synaptic current, explained in section 2.4. We use the sign convention of inter alia Dayan & Abbott[1] where membrane currents are defined as positive when positive charges flow *out* of the cell. I.e.

[1] [DA01], ch. 5.3, p. 162

a positive I_{syn} decreases the membrane voltage (itself defined as the electric potential inside minus outside the cell).

The adaptation current w decays exponentially to zero on its own ($-w$ in equation (2)), and is influenced by voltage deviations from equilibrium: for $a > 0$, w acts on V in the opposite direction of the deviation, and for $a < 0$, w acts in the same direction. Izhikevich calls the former a resonant current, and the latter an amplifying one.[2]

[2] [Izh07], section 5.2.4. Note the different notation used; see the translation in table 2.

Name	Description	Value
V	Membrane voltage	(in mV)
w	Adaptation current	(in pA)
C	Membrane capacitance	104 pF
g_L	Input / leak conductance	4.3 nS
E_L	Resting / leak potential	-65 mV
Δ_T	Threshold slope factor	0.8 mV
V_T	Location of minimum of $\frac{\mathrm{d}V}{\mathrm{d}t}$	-52 mV
τ_w	Time constant of adaptation current	88 ms
a	Sensitivity of adaptation current to V	-0.8 nS
θ	Spike definition threshold	40 mV
V_r	Reset voltage after spike	-53 mV
b	Adaptation current bump after spike	65 pA

Table 1: Quantities and parameters of the AdEx neuron, equations (1) to (3). Values are from a model fit to a cortical regular spiking (RS) neuron, from [Nau+08]. By defining the location of $\frac{\mathrm{d}V}{\mathrm{d}t}$'s minimum, V_T also co-determines the location of the firing threshold.

In this chapter, we will analyse $\frac{\mathrm{d}V}{\mathrm{d}t}$ as a function of V, i.e. analyse it as a dynamical system: will the voltage increase or decrease at the current voltage? For conciseness, we call this function $F(V)$. I.e. $F(V) = \frac{\mathrm{d}V}{\mathrm{d}t} =$ the right-hand-side of equation (1) here, scaled by $1/C$. We'll mostly analyse F in the absence of synaptic and adaptation currents, i.e. for I_{syn} and w both zero. Figure 2 shows the $F(V)$ curve for an AdEx neuron fit to a real neuron. Figure 5 compares the $F(V)$ curve of an AdEx neuron with that of another two-dimensional neuron model, the Izhikevich neuron.

In addition to the two differential equations, the AdEx model also consists of an instantaneous reset condition. When the membrane voltage V reaches a certain threshold θ, a spike is recorded, V is reset, and w is increased:

$$\text{if } V > \theta \text{ then: } \begin{aligned} V &\leftarrow V_r \\ w &\leftarrow w + b \end{aligned} \tag{3}$$

This bump of the adaptation current is what provides the spike rate adaptation: the more spikes the neuron has recently fired, the higher the adaptation current w and the more it drives down the voltage (equation (1)), away from the firing threshold.

For a full description of the different firing patterns that the AdEx neuron can exhibit, for different parameter values (i.e, a characterisa-

tion of its bifurcations), see [Nau+08], "Firing Patterns in the Adaptive Exponential Integrate-and-Fire Model". For an even fuller description, including a derivation of two-dimensional neuron models from higher-dimensional, Hodgkin-Huxley-like models, see Eugene Izhikevich's book, *Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting* [Izh07].

In terms of parameter choice, we choose to simulate a cortical regular spiking (RS) neuron, i.e. a 'standard' excitatory neuron that does not display e.g. bursting or fast-spiking behaviour. We take our parameter values from [Nau+08][3], where they fitted an AdEx model to recordings from real cortical RS neurons injected with different step currents. Parameter values are listed in table 1.

[3] Section 6, and Table 1, row 3

Examples of the signals V and w that this model generates are given later, in figure 10 and figure 13.

Analysis

Where are the fixed points of the dynamical system $\frac{\mathrm{d}V}{\mathrm{d}t} = F(V)$? I.e, where is $F = 0$? Like other neuron models, there are two fixed points: a stable one at the leak potential E_L, and an unstable one at the instantaneous firing threshold (which we'll call E_T).

We see in equation (1) (for $I_{\text{syn}} = 0$ and $w = 0$) that the leak potential E_L is indeed a fixed point – or rather lies very close to one: the exponential term is negligibly small at $V = E_L$. The second fixed point has no straightforward expression. The exact solutions for F's roots need the so called Lambert W or 'product logarithm' functions W_0 and W_{-1} (see figure 3). The roots are found at:

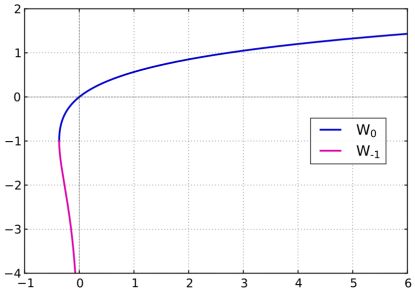

Figure 3: The Lambert W functions for real numbers.

$$V = E_L - \Delta_T W_k \left(-\exp\left(\frac{E_L - V_T}{\Delta_T} \right) \right) \tag{4}$$

..where $k = 0$ gives the resting potential, and $k = -1$ the instantaneous firing threshold.[4] For our cortical RS neuron, this gives us an instantaneous threshold of $E_T = -49.6$ mV.

[4] Here too we see that the true resting potential is almost identical to E_L: W_0 passes through zero, and its argument is ≈ 0, because the exponential's argument is negative. For W_{-1} however, figure 3 shows that the second term is not negligible around 0.

Also of interest – especially when comparing with the Izhikevich neuron later – is the slope of AdEx's $F(V)$, i.e. its derivative with respect to V:

$$\begin{aligned} \frac{\mathrm{d}F}{\mathrm{d}V} &= \frac{\mathrm{d}}{\mathrm{d}V}\left(-g_L(V - E_L) + g_L \Delta_T \exp\left(\frac{V - V_T}{\Delta_T} \right) \right) \\ &= -g_L + g_L \exp\left(\frac{V - V_T}{\Delta_T} \right) \end{aligned} \tag{5}$$

This derivative is zero at $V = V_T$. I.e, unlike what is suggested by the 'T' subscript, and the name 'effective threshold potential' given to it in the AdEx literature[BG05; Nau+08], V_T is not the instantaneous threshold potential (it is not a zero of F), but rather the minimum of F (it is the zero of $\frac{\mathrm{d}F}{\mathrm{d}V}$).

From equation (5), we also calculate the slope of F at its two roots. At $V = E_L$, the slope (also known as the leak or input conductance

here) is $-g_L$, plus a negligibly small exponential term. The negative sign shows that this is a stable fixed point: in a linearization of F around this point, at voltages below the resting potential, F (i.e. $\frac{\mathrm{d}V}{\mathrm{d}t}$) is positive and thus V will increase. At voltages above E_L, F is negative and V will decrease. Small deviations on either side of the resting potential will thus decay back to this resting potential.

At the firing threshold $V = E_T$ (i.e. the second solution to equation (4)), the slope is:

$$g_L \left(\exp\left(\frac{E_T - V_T}{\Delta_T} \right) - 1 \right) \tag{6}$$

..which is positive (as $E_T > V_T$), indicating that this is an unstable fixed point: a small deviation of the voltage above E_T will blow up to infinity (i.e, a spike is generated).

2.2 Alternative neuron models

Why did we choose the AdEx model to simulate neuron voltages? In short, because it strikes a good balance between realism and complexity. We briefly consider here two alternative neuron models: the simple leaky-integrate-and-fire (LIF) neuron, and the more complex Hodgkin-Huxley (HH) neuron. In the next section, we go into more depth on a third alternative, the very similar Izhikevich neuron.

A simpler model than AdEx would be the well-known LIF neuron:

$$C\frac{\mathrm{d}V}{\mathrm{d}t} = -g_L(V - E_L) - I_{\text{syn}}$$

$$\text{if } V > \theta, \text{ then: } V \leftarrow V_r$$

As is apparent from comparing this with equations (1) and (3), the AdEx model is an extension of the LIF model. The LIF neuron lacks a simulation of the upstroke of spikes (the exponential term in equation (1)), and the slower time-scale adaptation current (equation (2)), which allows the simulation of many qualitatively different real neuron types. It is especially this first addition, the full upstroke simulation, that seems relevant in generating realistic voltage traces.

Would this thesis have been very different had we used LIF neurons instead? Probably not, though it might depend on the mean voltage level of the simulated neuron: if it is well below the firing threshold, both LIF and AdEx are linear (the exponential term is negligible), and they behave quasi identically. When a spike is generated in the AdEx model, the exponential feedback makes the upstroke very fast, and thus not many timesteps in the simulation are spent on it, versus the linear regime.

On the other hand, when the neuron would continuously teeter just below its firing threshold, the LIF and AdEx models do not behave

similarly. LIF's $F(V)$ curve is still fully linear, while AdEx's is not, and AdEx will behave more like a real neuron – see figure 2.

Another well-known alternative neuron model is the class of Hodgin-Huxley (HH)-like neurons. These models simulate the full trajectory of a spike: both its upstroke and its downstroke. Unfortunately they also have many free parameters. They also take a bit longer to simulate, being higher dimensional (having more differential equations), and containing many more exponential terms, which take the brunt of the time when numerically evaluating a differential expression.

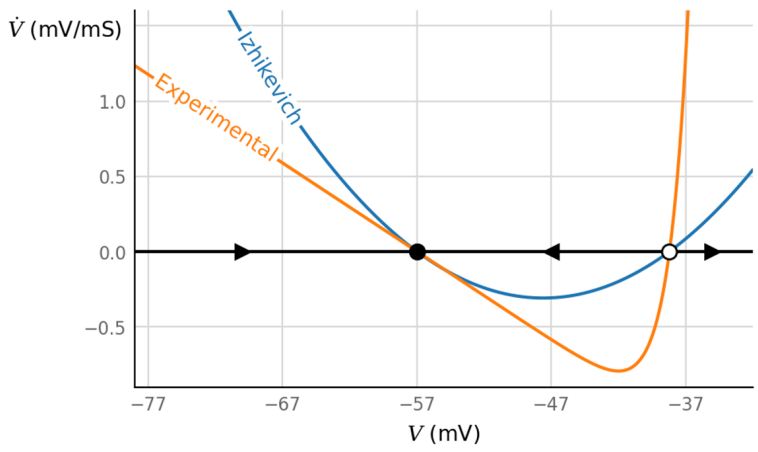

Figure 4: Neuron models as dynamical systems: a comparison of the $F(V)$ curves of the Izhikevich and AdEx neurons. 'Experimental' is an AdEx neuron fit to a cortical pyramidal neuron (using data from [Bad+08]). The Izhikevich neuron's parameters were chosen to match the fixed points and the leak conductance. Arrows indicate whether the voltage will increase or decrease (1D flow field). $\rightarrow \bullet \leftarrow$ is a stable fixed point (resting potential), $\leftarrow \circ \rightarrow$ is an unstable fixed point (spike threshold). $\dot{V} \equiv \frac{\mathrm{d}V}{\mathrm{d}t}$.

2.3 The Izhikevich neuron

Another alternative neuron model is the Izhikevich neuron, which is exceedingly similar to the AdEx neuron. These are the Izhikevich equations, using the same symbols as used before (in equations (1) to (3)):

$$C\frac{\mathrm{d}V}{\mathrm{d}t} = k(V - E_L)(V - E_T) - I_{\text{syn}} - w \tag{7}$$

$$\tau_w \frac{\mathrm{d}w}{\mathrm{d}t} = a(V - E_L) - w \tag{8}$$

$$\begin{aligned} &\text{if } V > \theta, \text{ then:} \\ &\quad V \leftarrow V_r \\ &\quad w \leftarrow w + \Delta w \end{aligned} \tag{9}$$

We have introduced two new parameters not present in the AdEx equations: the steepness of the parabola, k; and E_T, the instantaneous firing threshold (the firing threshold in the absence of any synaptic or adaptation currents).

The only difference with AdEx is in equation (7), where the $F(V)$ curve is not made by a linear plus exponential term, as in AdEx; but rather by a quadratic (a parabola). Its two zeros (the fixed points) are readily apparent, as E_L and E_T.

Correspondences with AdEx

In Izhikevich's book[5], different names are used for the same quanti-

[5] [Izh07], section 5.2.4, equations 5.7 & 5.8

ties:

$$C\frac{\mathrm{d}v}{\mathrm{d}t} = k(v - v_r)(v - v_t) - u + I \tag{10}$$

$$\frac{\mathrm{d}u}{\mathrm{d}t} = a(b(v - v_r) - u) \tag{11}$$

$$\begin{aligned} &\text{if } v > v_{\text{peak}}, \text{ then:} \\ &v \leftarrow c \\ &u \leftarrow u + d \end{aligned} \tag{12}$$

Table 2 compares both notation conventions.

AdEx	Izh	Description	Units
V	v	Membrane voltage	V
w	u	Adaptation current	A
τ_w	$1/a$	Time constant of adaptation current	s
E_L	v_r	Resting / leak potential	V
V_r	c	Reset voltage after spike	V
a	b	Sensitivity of adapt. current to V	S
b	d	Adaptation current bump after spike	A

Table 2: **Translating between Izhikevich and AdEx.** Different symbols used for the same quantities, in [BG05] and in most of this thesis ('AdEx') , and in [Izh07] ('Izh'). Membrane capacitance C (in farad) is the same in both notations.

Beside these straightforward correspondences, there are some parameters in either model that have no direct equivalent in the other: k and v_t in Izhikevich, and g_L, Δ_T, and V_T in AdEx. For those, we'll look at the shape of Izhikevich's $F(V)$, as we've done for the AdEx neuron before.

First, the AdEx parameter g_L. This is the input conductance, a.k.a. the leak conductance, and the slope of $F(V)$ around the leak potential. We can find this same conductance for the Izhikevich neuron by taking the derivative with respect to v of the right hand side of equation (10), at $w = 0$, $I_{\text{syn}} = 0$, and $v = v_r$. We find:

$$\frac{\mathrm{d}}{\mathrm{d}v}(k(v - v_r)(v - v_t))\Big|_{v=v_r} = k(v_r - v_t) \tag{13}$$

(this value is negative: the leak potential is a stable fixed point. This corresponds to equation (1), where we find '$-g_L$'). Thus, our first nontrivial correspondence:

$$g_L = k(v_t - v_r) \tag{14}$$

We've seen that V_T is the minimum of AdEx's F. The minimum of Izhikevich's F is easily found as the average of the parabola's two zeros. I.e, V_T corresponds to $(v_r + v_t)/2$.

Finally, Δ_T co-determines the slope of AdEx's F at the firing threshold (equation (6)). Given that Izhikevich's F is a parabola, with

slopes equal in magnitude at both roots, we already know the firing threshold slope: it is the same as the leak conductance, equation (14). Here, the AdEx model is more expressive than Izhikevich's: the slope of $\frac{\mathrm{d}V}{\mathrm{d}t}$ at the firing threshold can be independently tweaked from the leak conductance; in Izhikevich these two are clung together by the form of the quadratic equation.

Comparison with AdEx

The Izhikevich and AdEx models are very similar. Their phase spaces are topologically identical: the adaptive current equation is identical (up to a renaming of the variables); and the $F(V)$-graph has the same shape, with two fixed points: a stable fixed point at the resting potential, and an unstable one at the firing threshold (figure 4).

They differ in the exact shape: Izhikevich's $F(V)$ is a parabola, while AdEx is the more realistic 'linear subthreshold, and then transitioning to an exponential' (see figures 2 and 4). As a result, Izhikevich neurons have an unrealistically slow spike upstroke, examples of which can be seen in figure 5.

A second issue is Izhikevich's subthreshold nonlinearity. The effects of this can be seen in figure 6. Positive input currents produce stronger responses than equally large negative input currents. This is explained by the quadratic $\frac{\mathrm{d}V}{\mathrm{d}t}$ shape: positive deviations are attenuated less, and negative deviations more, than a linear neuron would. Real and AdEx neurons do not suffer this assymetry.

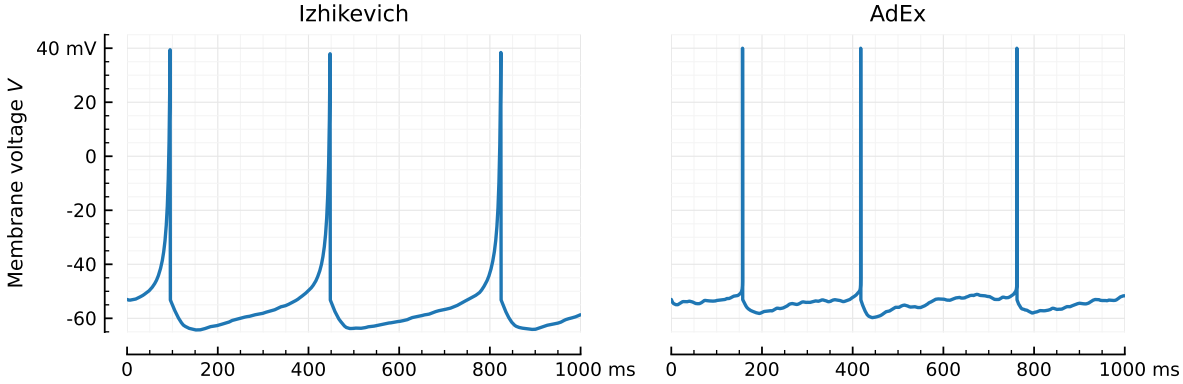

Figure 5: **Two neuron models behave differently for (near) identical parameters and input.**
The AdEx neuron's parameters are from [Nau+08], for a cortical regular spiking neuron. The Izhikevich neuron's parameters are copied from the AdEx neuron wherever they correspond directly. The other parameters are chosen so both models have the same resting and threshold potentials, and the same leak conductance, using the correspondences found earlier in this section. Both models receive EI-balanced synaptic input from 6500 Poisson spike trains with lognormal firing rates. The AdEx neuron was given stronger inputs ($\Delta g_{\text{exc}} = 12.2$ pS) than the Izhikevich neuron ($\Delta g_{\text{exc}} = 4$ pS), so as to obtain the same number of output spikes. (In both cases, $\Delta g_{\text{inh}} = 4\ \Delta g_{\text{exc}}$). For more details, see `2023-06-23__Vm_traces_AdEx_Izh__Brian`.

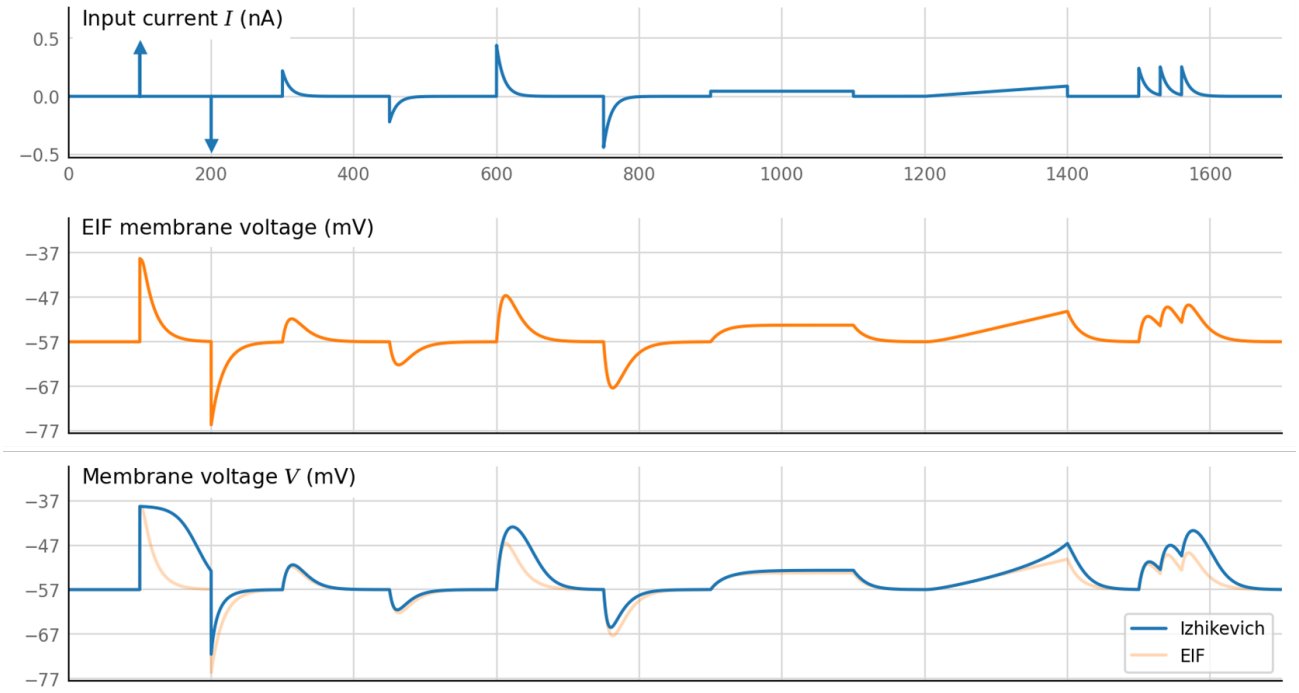

Figure 6: **The nonlinear response of Izhikevich neurons to subthreshold input currents.**
'EIF' stands for exponential integrate-and-fire; it has the same $\frac{\mathrm{d}V}{\mathrm{d}t}$ as an AdEx neuron. Adaptation currents are negligibly small for both models in this test scenario.

This nonlinearity is not visible for small voltage deviations, which is what the postsynaptic potentials we are interested in in this thesis tend to be. There is however an effect of the neuron's average voltage: if this voltage is constantly on the higher side, then inputs – both negative and positive – will cause larger responses than if the median voltage was lower.

2.4 Synapse model

One as of yet unexplained term in our neuron model, equation (1), is the synaptic current I_{syn}. This is the following sum over all input synapses i of the neuron:

$$I_{\text{syn}} = \sum_i g_i(V - E_i) \tag{15}$$

where V is the global membrane voltage of the neuron, E_i is the reversal potential of that synapse, and g_i is the local synaptic conductance, which is modulated by presynaptic spikes.

For an excitatory synapse, $V < E_i$, making $g_i(V - E_i)$ negative, increasing the membrane voltage according to the sign convention for I_{syn} in equation (1).

We simulate the synaptic conductances g_i as exponentially decaying signals (with time constant τ_g), and bump them up instantaneously on arrival of a presynaptic spike:

$$\frac{\mathrm{d}g_i}{\mathrm{d}t} = -g_i/\tau_g \tag{16}$$

On incoming presynaptic spike:

$$g_i \leftarrow g_i + \Delta g_i \tag{17}$$

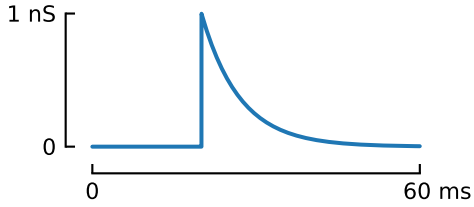

Figure 7: Example synaptic conductance trace $g_1(t)$, with a single incoming spike at $t = 20$ ms.

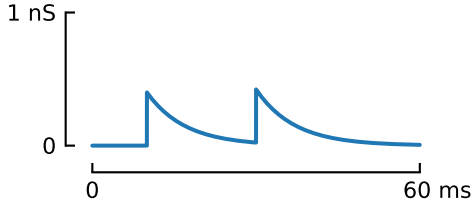

Figure 8: Another example trace $g_2(t)$, with spikes at $t = 10$ ms and 30 ms, and a smaller Δg.

Note that these are not the so called alpha-synapses. Those are two dimensional and also have an exponential rise, instead of just an exponential decay. (For an infinitely fast rise though, these models are of course the same). Simulating a full alpha synapse might increase the realism of our voltage traces, for a small simulation cost. We did not try this however. Foremost because alpha synapses fit to real data often have very fast rise times that are almost indistinguishable from instantaneous jumps.

For efficiency, we give all our excitatory synapses the same reversal potential, E_{exc}. Idem for the inhibitory synapses, with E_{inh}. This allows us to factor the synaptic current sum (equation (15)) as follows:

$$I_{\text{syn}} = (V - E_{\text{exc}}) \sum_{\text{exc } i} g_i \; + \; (V - E_{\text{inh}}) \sum_{\text{inh } i} g_i \tag{18}$$

The sums of conductance signals $g_i(t)$ can also be simplified. Say that the values of g_i at $t = 0$ are G_i. The solution to equation (16) (at least in the time until a new presynaptic spike arrives) is then

$$g_i(t) = G_i \, e^{-t/\tau_g} \tag{19}$$

With this, and when all synapses have the same time constant τ_g, the two sums in equation (18) can be factored as follows:[6]

$$\sum_i g_i(t) = \sum_i \left(G_i \, e^{-t/\tau_g} \right) = \left(\sum_i G_i \right) e^{-t/\tau_g} \tag{20}$$

[6] This is only valid in the time before any new spikes arrive. But the 'summability' still holds after a new spike. To see this, the given reasoning can be repeated, but simply with different values for the G_i (all decayed by an amount $e^{-t_{\text{spike}}/\tau_g}$, and one increased by a bump Δg_i), and then redefining t_{spike} to be $t = 0$.

This means that we need to only keep track of two conductance signals: g_{exc} and g_{inh}, each the sum of all excitatory or all inhibitory synaptic conductances.

Our synaptic current sum then becomes simply:

$$I_{\text{syn}} = (V - E_{\text{exc}})\, g_{\text{exc}} \; + \; (V - E_{\text{inh}})\, g_{\text{inh}}, \tag{21}$$

and we only need to simulate two differential equations, instead of one for every synapse:

$$\begin{aligned} \frac{\mathrm{d}g_{\text{exc}}}{\mathrm{d}t} &= -g_{\text{exc}}/\tau_g \\ \frac{\mathrm{d}g_{\text{inh}}}{\mathrm{d}t} &= -g_{\text{inh}}/\tau_g, \end{aligned} \tag{22}$$

where on arrival of a spike at synapse i either g_{exc} or g_{inh} is instantaneously increased by a value Δg_i, depending on whether that synapse is excitatory or inhibitory.

We choose an excitatory reversal potential of $E_{\text{exc}} = 0$ mV, an inhibitory one of $E_{\text{inh}} = -80$ mV, and a time constant for the synaptic conductance decay of $\tau_g = 7$ ms. These values are rather arbitrary, but in line with other simulation studies (e.g. [Bre+07]).

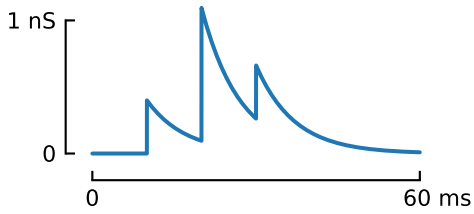

Figure 9: A third synaptic conductance trace $g_3(t)$, with three input spikes at the same times and strengths as in figures 7 and 8. This signal is simulated independently, but turns out to be equal to the sum of the two others: $g_3(t) \equiv g_1(t) + g_2(t)$.

2.5 Model summary

Combining equations (1) to (3) with equations (21) and (22), our complete AdEx point neuron with conductance-based synaptic current is:

$$C\frac{\mathrm{d}V}{\mathrm{d}t} = -g_L(V-E_L) + g_L\Delta_T \exp\left(\frac{V-V_T}{\Delta_T}\right) \underbrace{-g_{\text{exc}}(V-E_{\text{exc}}) - g_{\text{inh}}(V-E_{\text{inh}})}_{-I_{\text{syn}}} - w$$

$$\tau_w \frac{\mathrm{d}w}{\mathrm{d}t} = a(V-E_L) - w$$

$$\tau_g \frac{\mathrm{d}g_{\text{exc}}}{\mathrm{d}t} = -g_{\text{exc}}$$

$$\tau_g \frac{\mathrm{d}g_{\text{inh}}}{\mathrm{d}t} = -g_{\text{inh}}$$

When $V > \theta$: $V \leftarrow V_r$, $w \leftarrow w + b$

On input spike at..
exc. synapse i: $g_{\text{exc}} \leftarrow g_{\text{exc}} + \Delta g_i$
inh. synapse j: $g_{\text{inh}} \leftarrow g_{\text{inh}} + \Delta g_j$

We solve these equations numerically using first-order (Euler) integration, with a timestep Δt of 0.1 ms. This timestep is sufficiently small with respect to the different time constants in the model[7]. See section 2.9 below for more details on the numeric implementation.

Figure 10 shows the impulse response of this model, using a single spike, coming from either an excitatory or an inhibitory input neuron. The PSP bump is visible in the second panel from the bottom ('membrane voltage, V'). Note its tiny size, of about 0.04 mV. Our task will be to detect this tiny signal, in a sea of voltage imaging noise and PSP bumps of other input neurons.

Note also that, even though the inhibitory input is four times as strong as the excitatory one, its synaptic current and PSP bumps are not larger. (In fact, they are a smidge smaller). This is due to the fact that the neuron's resting potential (at -65 mV) lies closer to the in inhibitory reversal potential ($E_{\text{inh}} = -80$ mV) than the excitatory one ($E_{\text{exc}} = 0$ mV), making the the inhibitory term of the synaptic current I_{syn} weaker than the excitatory term (equation (21)).

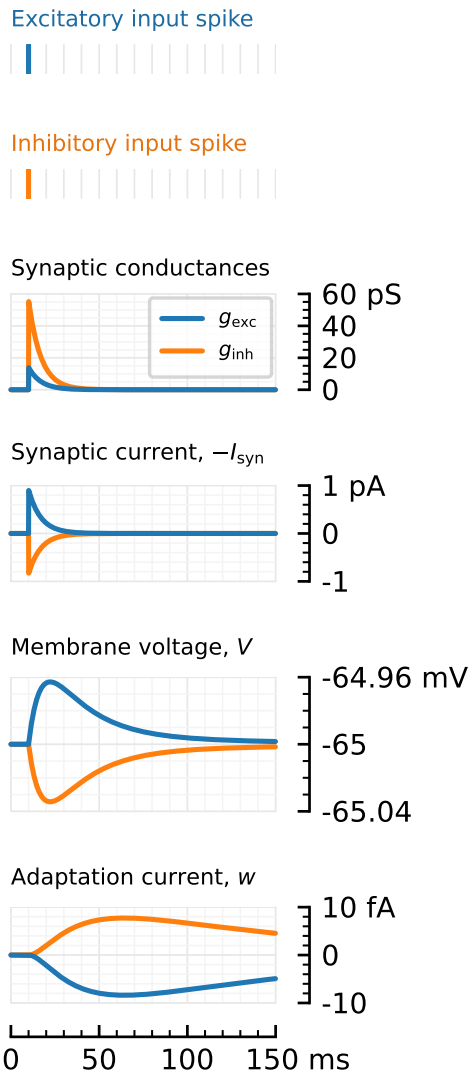

Figure 10: **Impulse response of the conductance-based AdEx neuron.** Overlay of two independent simulations, each using a single spike arriving at $t = 10$ ms, with synaptic weight of either $\Delta g_{\text{exc}} = 14$ pS, or $\Delta g_{\text{inh}} = 56$ pS. Parameters as in table 1, with $E_{\text{exc}} = 0$ mV, $E_{\text{inh}} = -80$ mV and $\tau_g = 7$ ms. For the inhibitory impulse response, $g_{\text{exc}}(t) \equiv 0$, and for the excitatory impulse response, $g_{\text{inh}}(t) \equiv 0$ (neither is shown). More details at `2023-09-05__Inhibitory_impulse_response_PSP`.

[7] $\tau_g = 7$ ms, $\tau_w = 88$ ms, $C/g_L = 24.2$ ms.

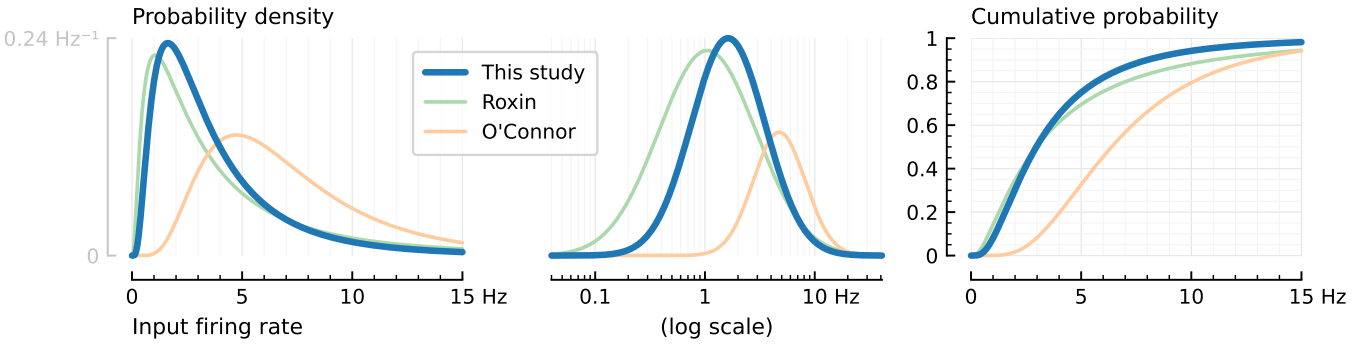

Figure 11: **Input spike trains are given log-normally distributed firing rates.**
The distribution used in our simulations is shown in bold. It has mean $\mu_x = 4$ Hz and variance of the underlying Gaussian $\sigma^2 = 0.6$. The two light distributions are from the literature. Note that the Roxin et al. distribution is slightly more heavy tailed than this study's: it has both more low firing and more high firing neurons.

2.6 Input in the N-to-1 setup

In our simplest experimental setup, we simulate just one AdEx neuron. Its input is provided by an array of N Poisson neurons, i.e. they each generate spike trains according to a Poisson process. We call this the 'N-to-1' setup.

Log-normal Poisson spiketrains

The firing rates of real neurons often follow a long-tailed distribution: most neurons do not fire much at all, while a few fire a lot. We recapitulate this in the firing rates of our Poisson input neurons, by drawing their firing rates from a log-normal distribution. We look to the literature for realistic parameters for this distribution, namely to the modelling paper of Roxin et al. [Rox+11], and the experimental sources it cites [HDZ08; OCo+10].

Log-normal distributions are usually parametrized with μ and σ: the location and scale of the underlying normal distribution, i.e. after log-transforming the input domain. In the above sources however, the mean μ_x of the data distribution itself is given. We can find μ, if σ is known, as $\mu = \ln(\mu_x) - \sigma^2/2$.

Roxin et al. use a mean rate μ_x of 5 Hz and a variance of the logarithm of the rate $\sigma^2 = 1.04$ in their figure 2. Hromádka et al. [HDZ08] recorded neurons in the auditory cortex of awake rats during acoustic stimulation. They find a mean firing rate μ_x of 6.2 Hz and a median of 2.4 Hz; no numeric variances are given. O'Connor et al. [OCo+10] recorded neurons in the barrel (whisker) cortex of behaving mice. Ensembled over all layers of the cortex they recorded from, they report $\mu_x = 7.4$ Hz, $\sigma_x = 12.6$ Hz, a median of 1.5 Hz, and an interquartile range of 9.5 Hz. This corresponds to a Gaussian variance of $\sigma^2 = \ln(1 + \sigma_x^2/\mu_x^2) = 0.30$. The Roxin and O'Connor

distributions are also shown in figure 11.

Given these data, we choose our parameters μ and σ so that the distribution lies roughly halfway O'Connor's and Roxin's, while making sure our firing rates are rather low than high: as our idea for connection inference rests on the number of spikes that can be used to calculate spike-triggered averages (see chapter 3), we don't want to overestimate the number of available input spikes and obtain overly optimistic results.

Our log-normal distribution has a median of 2.96 Hz, very close to Roxin's 2.97 Hz. O'Connor's has 6.4 Hz, which is markedly different from the median of 1.5 Hz they reported (hinting that their data distribution might not be log-normal).

Synaptic weights

We choose to simulate EI-balanced input, where we simulate four times as many excitatory as inhibitory input spiketrains; but with the inhibitory inputs four times as strong as the excitatory inputs.

I.e. if our total number of input spiketrains is $N = 6500$, we have $N_{\text{exc}} = 5200$ excitatory inputs, and $N_{\text{inh}} = 1300$ inhibitory inputs. In the N-to-1-setup, we give every input of the same type the same synaptic weight. I.e. in the model summary (section 2.5), $\forall i : \Delta g_i = \Delta g_{\text{exc}}$ and $\forall j : \Delta g_j = \Delta g_{\text{inh}}$. If we would thus choose $\Delta g_{\text{exc}} = 10$ pS, then, for 4:1 EI-balanced input, Δg_{inh} would be 40 pS.

To choose a value Δg_{exc} for the excitatory input strength, we look at the desired output firing rate of our one simulated neuron. We want it to spike at a realistic, average rate. Thus, we try for it to have the same output firing rate as the mean input firing rate: $\mu_x = 4$ Hz. This rate is achieved by testing a range of different input drives Δg_{exc}: see figure 12. For $N = 6500$ inputs, we find an average output firing rate of 4.0 Hz at $\Delta g_{\text{exc}} = 15$ pS (and thus $\Delta g_{\text{inh}} = 60$ pS).

Two asides on figure 12. First, from the bottom panel, we see that the AdEx model, with parameters for a cortical RS (regular spiking) neuron, is a so called 'Type I' neuron:[8] the firing rate can be made arbitrarily low, and there is no discontinuous jump from 0 Hz to some minimum firing rate.

Second, why does the output activity level increase for increasing excitatory input, if the inhibitory input becomes stronger by the same amount? The answer lies in the synaptic reversal potentials, which are $E_{\text{exc}} = 0$ mV and $E_{\text{inh}} = -80$ mV. The median membrane voltage (top panel in figure 12), at about -60 mV, lies closer to the inhibitory reversal potential than the excitatory potential, making the inhibitory term of the synaptic current I_{syn} weaker than the excitatory term (equation (21)).

The fact that this EI-balanced input leads to a net-excitatory effect can also be seen in figure 13: the signal $g_{\text{exc}} - g_{\text{inh}}$ hovers around zero,

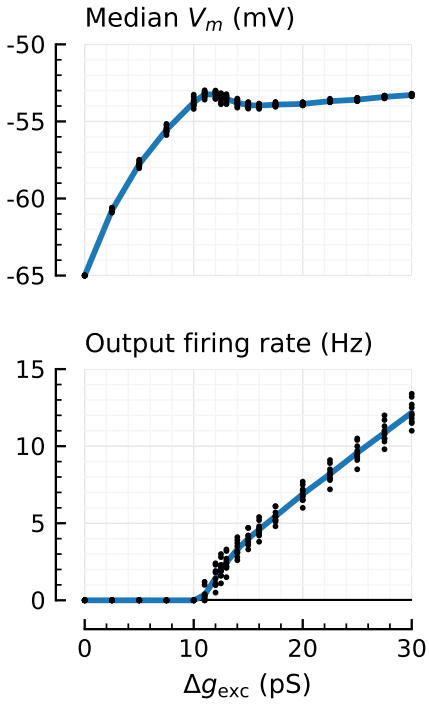

Figure 12: **Activity level of the output neuron in the N-to-1 setup, for increasing input strength Δg_{exc}.**
With $N = 6500$ EI-balanced inputs, and $\Delta g_{\text{inh}} = 4 \cdot \Delta g_{\text{exc}}$.
The blue line is the average over 10 different simulations (black dots), each with a different random seed for the input spike train generation.
Increasing input first heightens the voltage level without increasing the firing rate, and later heightens the firing rate without increasing the voltage level further.
V_m = membrane voltage V.
Source: `2023-08-05__AdEx_Nto1_we_sweep`.

[8] https://neuronaldynamics.epfl.ch/online/Ch4.S4.html

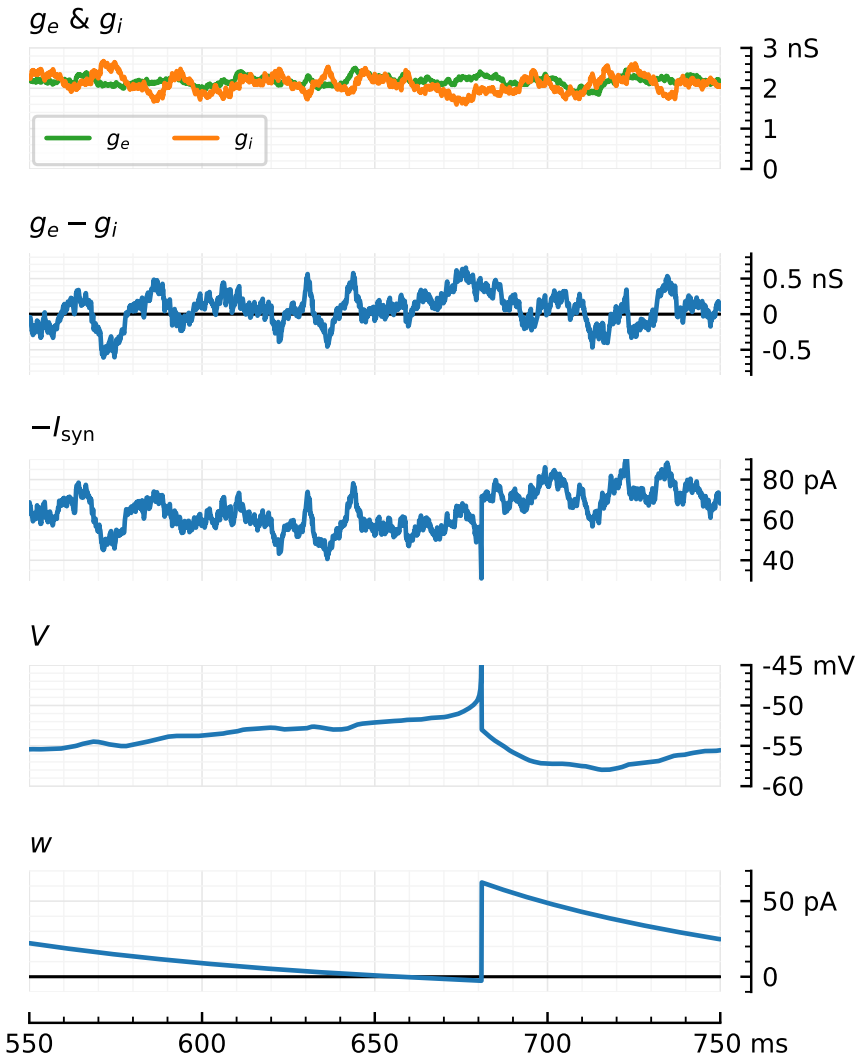

Figure 13: **Signals generated by the neuron model, for 6500 EI-balanced input spiketrains.**
Example extract from the signal traces of the conductance-based AdEx neuron. N-to-1 setup, with $\Delta g_{\text{exc}} = 15$ pS, and Poisson inputs with lognormally distributed firing rates. Neuron parameters for a cortical regular spiking neuron.
Note that the synaptic current signal $-I_{\text{syn}}$ is very similar to the difference of the excitatory and inhibitory synaptic conductances, as long as the membrane voltage V does not vary too much; on a spike (here at $t \approx 680$ ms), I_{syn} spikes as well.
Also note that the neuron spikes at a time when the inhibitory input randomly falls below the excitatory input, for a certain time. Spikes are 'fluctuation driven' in this input regime.
$g_e \equiv g_{\text{exc}}$ and $g_i \equiv g_{\text{inh}}$.
The V and $-I_{\text{syn}}$ signals are cut-off by the plot boundaries at the spike time.
Source: `2023-07-26__AdEx_Nto1_we_I_syn`.

but the signal I_{syn} — even though it has largely the same shape — is nowhere near zero.

Later, we will compare the performance of a connection-detection algorithm across different numbers of inputs N. We choose the approximately evenly log-spaced sequence

$$N \in [10,\ 20,\ 45,\ 100,\ 200,\ 400,\ 800,\ 1600,\ 3200,\ 6500].$$

For each of these N, we want the output firing rate to be the same, namely 4 Hz. To find the synaptic strengths Δg_{exc} that accomplish this, we perform an iterative search.[9] As an initial guess, it would make sense for the required input strength to scale inversely proportional to the number of inputs. We find however that there is a slight deviation from this linear expectation (figure 14): the less inputs, the less strong each input should be to reach the same output firing rate, compared to a linear extrapolation from our finding of $\Delta g_{\text{exc}} = 15$ pS for $N = 6500$. E.g. the linear expectation for $N = 10$ would be $\Delta g_{\text{exc}} = 9.75$ nS ($= 15$ pS $\cdot\, 6500/10$), but we find we need only $\Delta g_{\text{exc}} = 2.83$ nS to reach an output firing rate of 4 Hz.

[9] We use SciPy's `root_scalar` function, which uses 'Brent's method', which is like bisection but has faster convergence. We seed the algorithm with the bracket $[w_0/4, w_0 \cdot 4]$, where w_0 is the initial linear guess for Δg_{exc}. In about 8 iterations, we come within 0.01 Hz of the desired 4 Hz.

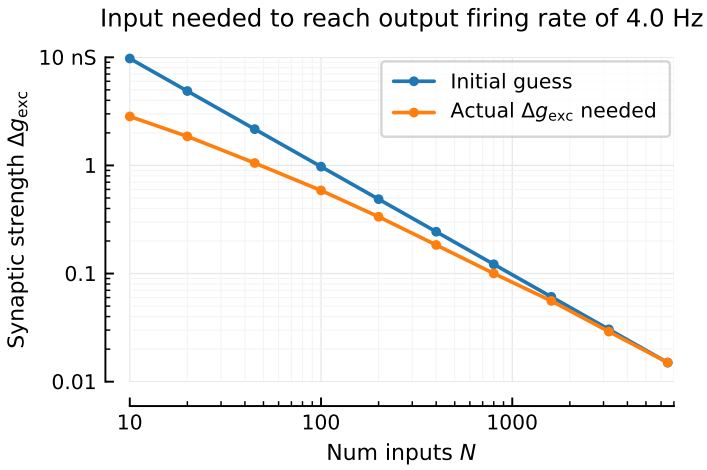

Figure 14: **Finding the right input strength in the N-to-1 setup, for different numbers of inputs N.** Δg_{exc} is the synaptic conductance increase per incoming spike from an excitatory input. For each N, the right Δg_{exc} value — namely the one for which the output neuron fires at 4 Hz – was found using an iterative search procedure. For every Δg_{exc} value evaluated, ten different 10-second simulations were run (each with a different random seed for the input spiketrain generation). The average output firing rate of these ten simulations was taken, and compared with the goal firing rate of 4 Hz. Δg_{exc} was then adjusted until the simulated firing rate was sufficiently close. In every case, Δg_{inh} was four times Δg_{exc}. Source: `2023-08-05__AdEx_Nto1_we_sweep`.

2.7 Spike ceiling

Our neuron model (section 2.5) consists of 1) a set of continuous differential equations, and 2) a discontinuous, instantaneous jump (namely whenever the membrane voltage V crosses the spike-definition threshold θ). Because we simulate our model using a discrete and finite timestep, this discontinuity introduces variability in the height of our spikes: the simulated voltage will never exactly equal the threshold θ in a given timestep, but will be either somewhere below it (where it will become the simulated spike height), or above it (in which case a spike is defined and the voltage is reset). This variability, or jitter, in the height of spikes can be seen in figure 15 (orange trace).

Real neurons generally do not show such variability: their spike heights are quite consistent (remarkably so). To make our simulated voltage traces look more realistic, we set the voltage to a fixed height at spike events. We call this modification "spike ceiling". It is illustrated in figure 15 (blue trace).

This spike ceiling is a common technique. In Izhikevich's book for example, the same technique is applied (there it is called "spike padding"): see listing 1. There is one difference however. Izhikevich's code pads the spikes during the simulation, and overwrites the spike height of the previous timestep, before the threshold crossing. We ceil our spikes after the simulation has fully run. We ceil the voltage one timestep later than Izhikevich: after the threshold

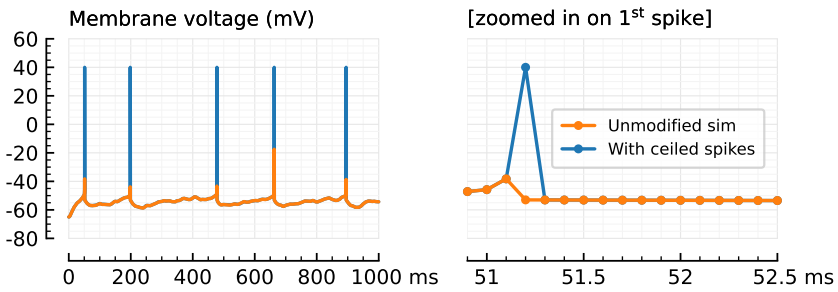

Figure 15: Example voltage trace, before (orange) and after (blue) spikes are ceiled.
Source: `2023-09-05__ceil_spikes`.

crossing; i.e. the timestep where the voltage was previously reset to V_r. This difference in approach is merely a personal aesthetic preference, and was not found to yield any difference in network inference performance in an informal test.

The act of spike ceiling in itself does have an effect on network inference performance. This is shown later, in section 3.1.

```
for i = 1:n-1              % forward Euler method
  v(i+1) = v(i)+tau*(k*(v(i)-vr)*(v(i)-vt)-u(i)+I(i))/C;
  u(i+1) = u(i)+tau*a*(b*(v(i)-vr)-u(i));
  if v(i+1) >= vpeak       % a spike is fired!
    v(i) = vpeak;          % padding the spike amplitude
    v(i+1) = c;            % membrane voltage reset
    u(i+1) = u(i+1)+d;     % recovery variable update
  end;
end;
```

Listing 1: Matlab code simulating the Izhikevich neuron. Lines pertinent to spike ceiling/padding are highlighted. Source: Eugene Izhikevich's 2007 book, *Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting*, section 8.4.1 ("Simple Model of Choice"), p. 274

2.8 Voltage imaging

The signals detected by a light microscope in a voltage imaging setup are not the same as the real membrane voltage traces of which they are a reflection.

We model this lossy transformation by simply adding Gaussian noise to our simulated membrane voltage. As in the voltage imaging literature, we quantify the amount of this noise by a 'spike-SNR' measure (spike signal-to-noise ratio). This is defined as the height of an average spike relative to the standard deviation of the noise:

$$\text{spike-SNR} = \frac{\text{spike height}}{\sigma_{\text{noise}}} \tag{23}$$

In other words, if we call our output signal y, and with V_i the voltage samples resulting from the numeric integration of equation (1), we have:

$$y_i = V_i + \varepsilon_i \tag{24}$$

$$\varepsilon_i \sim \mathcal{N}(0, \sigma_{\text{noise}})$$

A typical but conservative level of noise in voltage imaging recordings has a spike-SNR of 10. If we take as spike height the spike detection threshold minus the leak potential ($\theta - E_L$, see table 1), we obtain a σ_{noise} of 10.5 mV.

A more realistic model of the voltage imaging-transformation would also incorporate the exponential decay over time of the SNR (with a

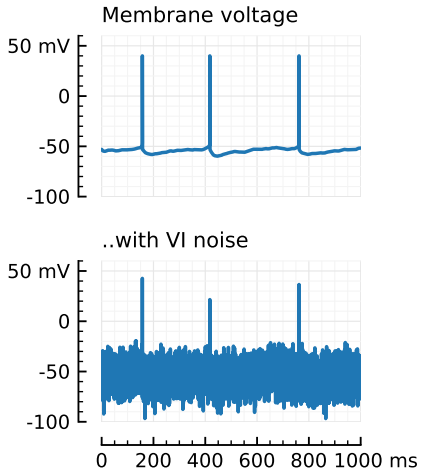

Figure 16: Simulated voltage trace (same as in figure 5, right), without and with 'voltage imaging' (VI) noise added ($\sigma_{\text{noise}} = 10.5$ mV).

time constant of about 10 minutes), and the short-term 'smearing in time' of voltage indicators. The latter could be done by passing the voltage signal through a linear filter with some non-instantaneous impulse response.

2.9 Software implementation

To numerically simulate the model described in this chapter, different software implementations have been used throughout my PhD.

Software	Language	Model	Time (seconds)	
			Compilation	Simulation
Brian2	Cython + Python	full	61	11
		merged	86	11
	C++	full	15	6.5
		merged	18	0.35
own	Julia	full	0.68	0.036

Table 3: **Time taken to simulate 10 seconds of the N-to-1 experiment** with N = 6500 Poisson inputs, and one conductance-based AdEx neuron. In the 'merged' models, only 200 Poisson spiketrains are independently simulated (instead of all 6500, as in the 'full' model). All other inputs are replaced by two single Poisson inputs (one excitatory and one inhibitory), with an equivalent effect. (That is, we use Brian's 'PoissonInput').
For details, see 2023-07-10__AdEx_Nto1_Brian_speedtest (Cython), 2023-08-02__speedtest_brian_standalone_AdEx_Nto1 (C++), and 2023_08_09__Poisson_cquantile_upperbound (Julia).

{to expand a bit}

Chapter 3

Spike-triggered averaging

As shown in figure 1, our idea for connection inference rests on the causal link 'presynaptic spike' → 'postsynaptic voltage bump'. I.e. we want to know for which neuron pairs a spike in one is reliably followed by such a bump in the other. The problem is that these bumps (the postsynaptic potentials or PSPs) are minute, and are easily drowned out by (1) other PSPs, (2) postsynaptic spikes, and (3) voltage imaging noise.

So, as is often done in neuroscience, we take the *average* over many instantiations, so as to hopefully find a signal in the noise. Specifically, we take spike-triggered averages, or **STA**s, of neurons' voltage traces. If there is a connection from a neuron 'M' to a neuron 'N', then an STA of neuron N's voltage imaging signal, based on neuron M's spikes, would hopefully show the PSP.

And indeed, when we construct a few such STAs, we do see something resembling a PSP bump: figure 17. We also find that the higher the firing rate of the presynaptic neuron, the cleaner the PSP-like shape is. This is of course because there are more presynaptic spikes and thus more windows to average over, which decreases the noise on the result. Finally, we see that inhibitory inputs result in downwards bumps in their STA, and excitatory inputs in upwards bumps.

Note that in this chapter – and the next one – we only look at the so called N-to-1 case (figure 18), where we simulate the voltage of one neuron, impinged on by N independent Poisson spiketrains. This is done for simplicity; it is only in the "Networks" chapter later on that we look at full networks, were inputs might be correlated with one another.

To use spike-triggered-averages as an actual connection test, we look specifically at the height of an STA, and compare it to a distribution of STA heights that we'd expect were the two neurons not connected. This is illustrated and explained in more detail in figure 19. This so called 'shuffle' test yields the proportion p of how many shuffled (random) spiketrains yield an STA with a larger height than the real STA. In a following section (Performance quantification), we'll use

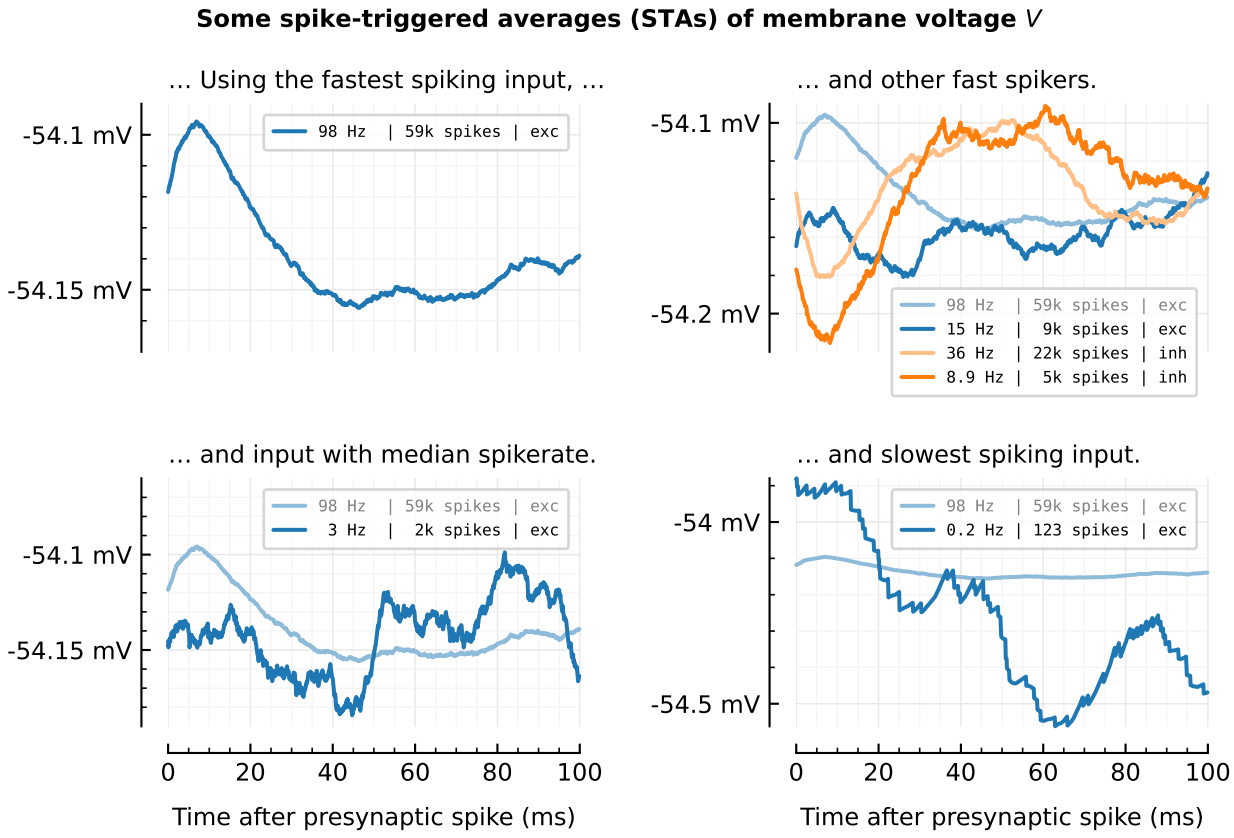

Figure 17: **Example STAs in the 10' simulation with 6500 inputs.** Note that every panel has a different y-axis (voltage) scale. The STA of the most active input is repeated in every panel (in faded blue), to allow a visual scale comparison nonetheless.
The inset legends indicate with how many presynaptic spikes the STA was calculated, and whether the input was an excitatory or inhibitory one.
The top right panel shows STAs of the 1st and 100th fastest spiking inputs, both within the excitatory inputs (blue shades), and within the inhibitory inputs (orange shades).
Source: `2023-09-13__Clippin_and_Ceilin`.

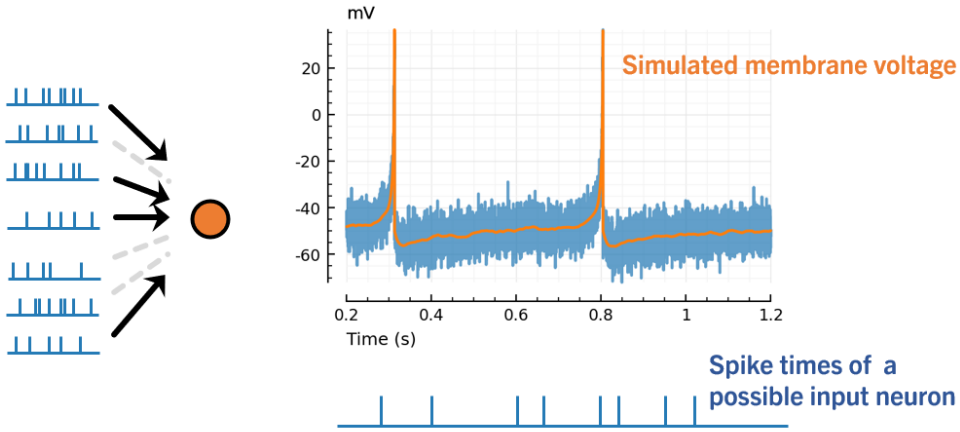

Figure 18: **The 'N-to-1' problem.**
Left: A neuron N (orange circle), and the spike trains of other neurons in the network (blue). Some of these other neurons impinge directly on N (black arrows), while others are not (directly) connected (dashed gray lines). Given only neuron N's voltage signal and the other neurons' spiketrains, we want to detect the direct inputs, while rejecting the not-directly-connected spiketrains.
Right: The simulated membrane voltage of the impinged-upon neuron (orange), and the same signal with Gaussian noise added, to simulate a voltage imaging signal (blue). Underneath the plot, one of the possible input spiketrains, time-aligned to the voltage signal. This alignment is used later to extract spike-triggered windows from the voltage signal.

this number (as $t = 1 - p$) to make predictions and compare them to the ground truth.

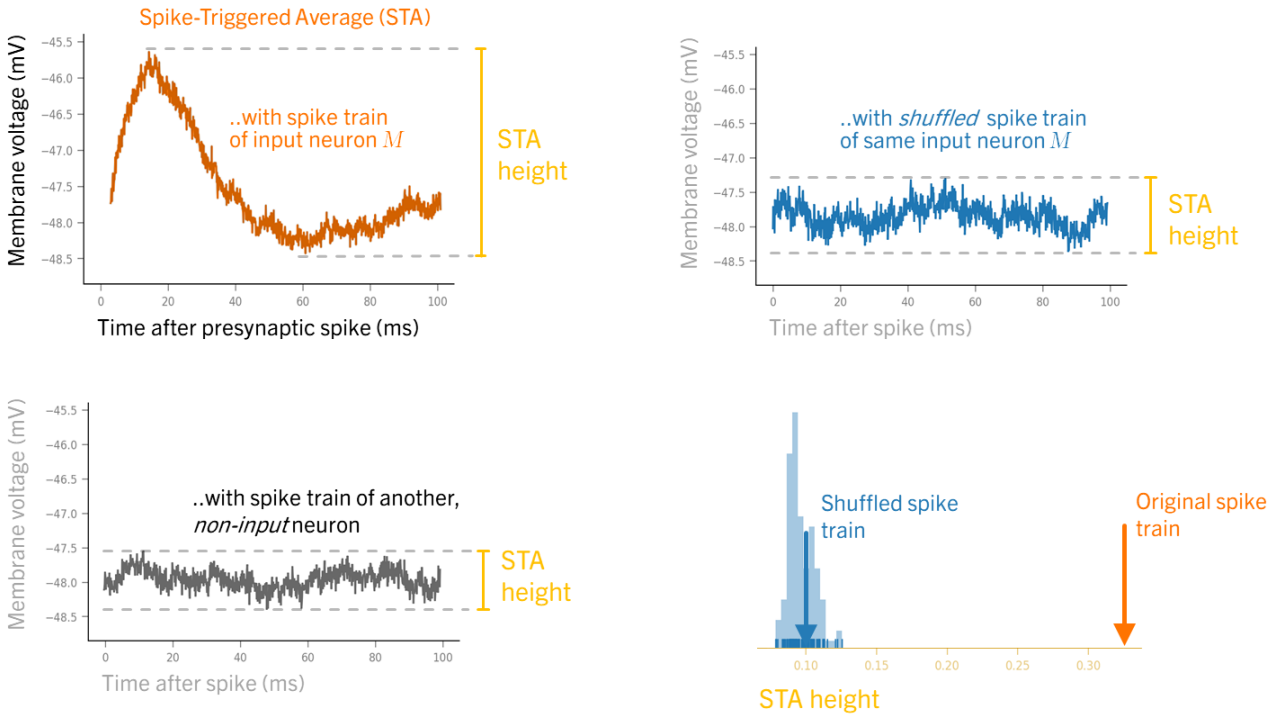

Figure 19: **A simple connection test: STA height with shuffle control.**
The spikes of a possible input neuron are aligned to the voltage trace of the neuron of interest N, as in figure 18. For every such spike, a 100-ms long window is cut out of the voltage of N. The average of all these windows is called the spike-triggered average (STA).
Left: Two example STAs of neuron N's membrane voltage: one for an actually connected input neuron, M (top, orange); and one for a non-input neuron (below, gray). Given an STA signal x, we will use its height $h = \max(x) - \min(x)$ (also known as 'peak-to-peak' or 'ptp') to test whether two neurons are connected.
Right: An STA of N's membrane voltage using a shuffled version of M's spike times (which is made by randomly permuting the inter-spike-intervals of M). This 'shuffled STA height' provides a control for the STA height connection test statistic: "what do we expect the STA height to be if there is *no* connection $M \to N$". By calculating different such shuffles, we obtain a null-distribution for the STA height test statistic. And by comparing the real STA height to this distribution, we can calculate a p-value. Here, the real STA is larger than all shuffle controls, of which there are 100. So $p < 0.01$, and at $\alpha = 0.05$, we conclude there is indeed a connection $M \to N$.

3.1 Ceiling and clipping

As explained in section 2.7, we modify our simulated voltage trace so that spikes have a consistent height. This modification has an effect on STAs, as is illustrated in figure 20: the blue trace is the signal without spike ceiling, the orange one with spike ceiling. Their corresponding STAs are shown on the right. Note that the orange STA (made with ceiled spikes) is much noisier than the blue STA (from the unmodified voltage trace).

This suggests a relatively easy intervention to drasticaly de-noise STAs, and presumably increase their effectiveness for network inference: namely to remove the spikes from the signal.

We tried this 'spike clipping' and it indeed drastically denoised the STA; see the green signal and STA in figure 20. We show that this decreased noise in the STA does indeed lead to an increase in network inference performance, by running a connection detection test without and with this spike clipping. The results are shown in figure 21: detection performance increases from an AUC of 0.56 for the non-clipped voltage trace, to an AUC of 0.79 for the voltage trace with clipped spikes.

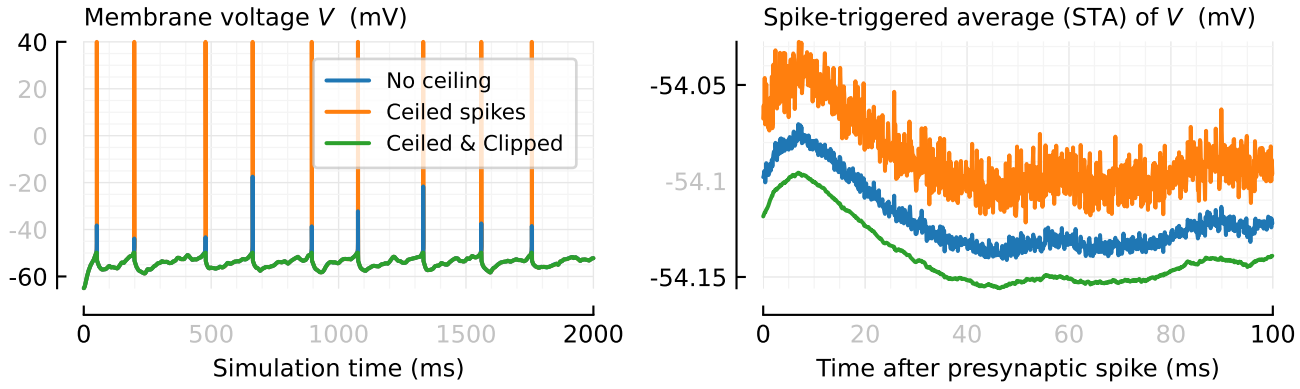

Figure 20: Example voltage traces and corresponding STAs, where the only difference is the height of spikes. In blue, the unmodified simulated voltage trace. In orange, the same, but with ceiled spikes (as in section 2.7). In green, the same as orange, but with the spikes clipped again after the ceiling (as explained in this section).
Source: `2023-09-13__Clippin_and_Ceilin`.

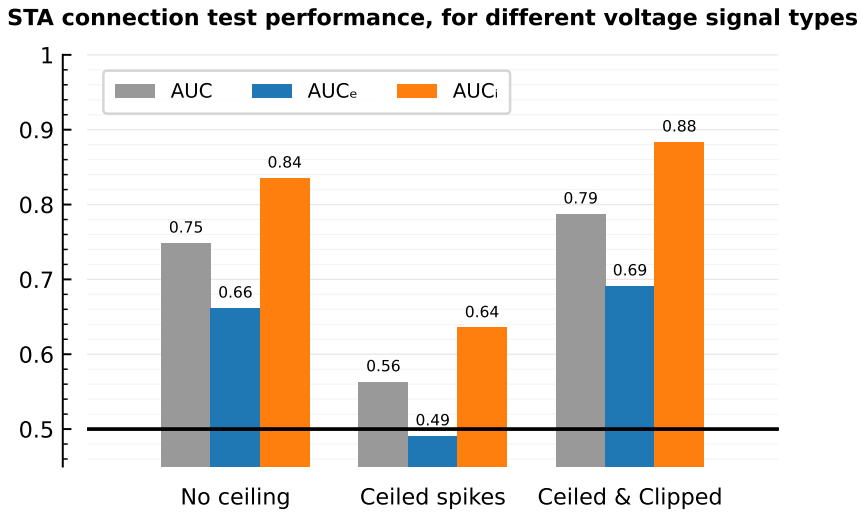

Figure 21: Connection detection performance for the three ways of handling spikes shown in figure 20: not modifying them; ceiling them; and clipping them. The area-under-the-curve or AUC measure is explained in the next section.
Source: 2023-09-13__Clippin_and_Ceilin.

3.2 Performance quantification

It is not easy to express in a single number how good a network inference algorithm is. Depending on what you find important as a user, different measures make more sense than others. This section looks at some measures to quantify the performance of our algorithms, and discusses the merits and disadvantages of each.

All the algorithms that we look at in this and the following chapter eventually output a single number per tested neuron pair (A, B): "How strongly do I believe there is a connection A→B?". (And: "Is that connection excitatory or inhibitory?": the sign of the number). We will call this connected-ness number "t".

To get actual predictions out of the algorithm, we must apply a threshold θ to these measures. If $|t| > \theta$, we classify the pair as connected, and as unconnected otherwise. For the detected pairs, we classify them as excitatory if t is positive, and inhibitory if it is negative.

Note that we use the same threshold for both excitatory and inhibitory connections. We could in fact use a different threshold – and we briefly look at this in figure 26 – but for simplicity, we apply the same threshold for both types of connection.

Each threshold chosen yields a different tradeoff between recall and precision (figure 22). At low thresholds, we can detect more connections ("true positives"); but we will also detect more non-inputs as being connected (false positives). This also lowers our precision.
Some definitions:

- True positive rate (TPR), aka recall, sensitivity, and power: out of all true connections tested, how many did we correctly classify?

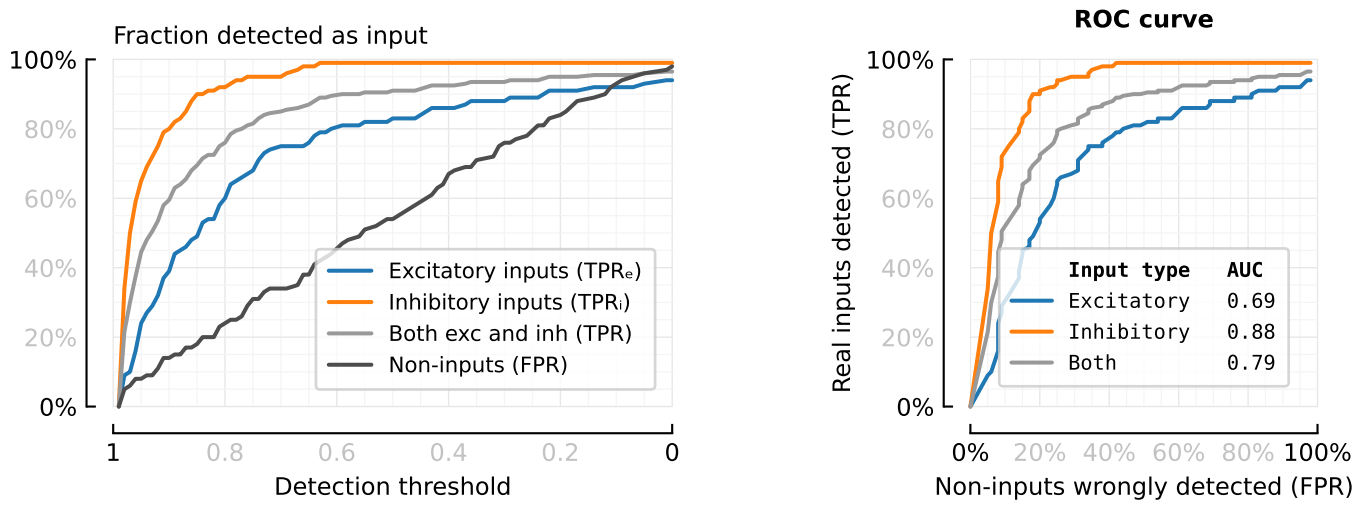

Figure 22: Lower detection thresholds increase both true and false positives. On the right, true positive rates are plotted against the false positive rate, to obtain the so called receiver operating characteric or ROC curve. As the FPR increases more or less linearly with the decrasing detection threshold, both graphs look very similar. Source: 2023-09-13__Clippin_and_Ceilin.

- False positive rate (FPR): out of all distractors we added to our test (randomly generated spiketrains), how many did we wrongly classify as an actual input?
- Precision, aka positive predictive value (PPV): out of all the neuron pairs that we classified as connected, how many are actually connected?

There are more measures that quantify the performance of a binary classifier at a given threshold than those three (such as negative predictive value, false discovery rate, false omission rate, ...).[1] But recall, FPR, and precision are commonly used ones.

Note that true positive rate (TPR) and precision are similar, in that they both count correct classifications. (They both have the number of true positives in their numerator). But recall (TPR) looks at the number of true positives from the point of view of the ground truth (how many did we find), and precision looks at it from the perspective of the experimenter (out of what this algorithm gives us, how much is correct?).

False positive rate and precision are also similar, in that they both measure the number of false positives. One advantage of using FPR over precision though, is that FPR does not depend on the number of distractors (unconnected spiketrains) that we add to our tests.[2] Whereas we can arbitrarily increase precision by including less unconnected trains in our test – up to the limit of 100% precision, when we do not add any distractors and all tested trains are actually connected.

[1] We do not actually perform binary classification: there are three classes (excitatory, inhibitory, unconnected). But it is not pure ternary (multiclass) classification either: we first classify as connected or not, and then (for the connected ones only), as excitatory or inhibitory. We could thus call it some kind of nested binary classification.

[2] Besides that, the more distractors we test, the more accurate our estimate of the FPR will be.

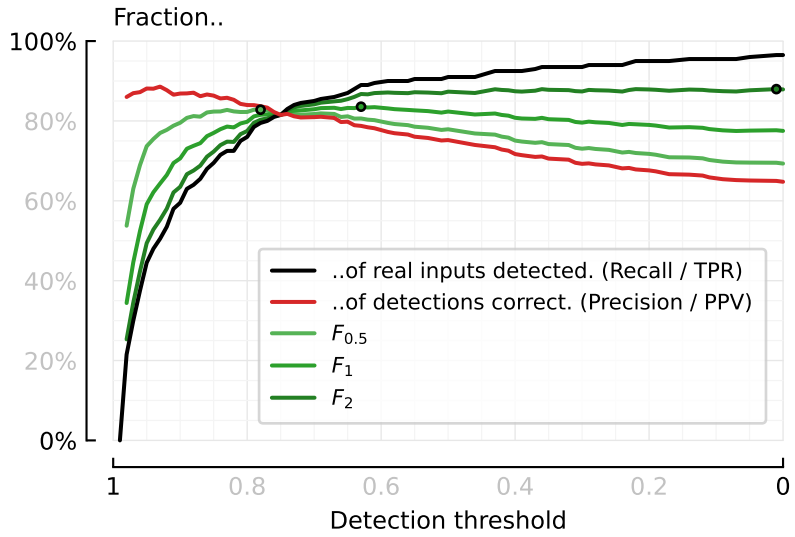

Figure 23: **Lower detection thresholds trade-off higher recall for lower precision.**
The F_β scores interpolate between the two measures. $F_{\to\infty}$ is recall, $F_{\to 0}$ is precision. The black circles on the F-curves indicate their maxima. Different trade-offs between precision and recall (different β-values) thus dictate different optimal detection thresholds.

In our tests, we choose the number of unconnected trains rather arbitrarily. (For example, when we test 100 excitatory and 100 inhibitory inputs, we also generate and test 100 unconnected spiketrains). A better way to choose this number of distractors might be to estimate what a realistic fraction of unconnected neurons would be in a typical voltage imaging experiment. Given some patch of brain tissue and one of the neurons in it, how many of the other recorded neurons in that patch will be connected to it? This is an interesting research question – and it is likely that answers can be found in the literature – but we do not explore it here.

In figure 22, we have looked at TPR and FPR, both as a function of the detection threshold and as a function of each other. In figures 23 to 25, we look at TPR (recall) and precision, again as a function of the detection threshold and as a function of each other.

Because recall and precision both increase for 'better' detectors, we might combine them into one measure. This is what the F-scores do: they are the harmonic mean of recall and precision, with recall and precision weighted differently depending on a parameter β. The F_β score attaches β times as much weight to precision P as to recall R:

$$F_\beta = \frac{(1+\beta^2)\cdot P \cdot R}{\beta^2 \cdot P + R} \tag{25}$$

For $\beta = 1$, precision and recall are weighted equally. The F_1-score is also the most widely used of the F-scores.

When we plot recall against precision, we get the so called PR-curves, shown in figure 24 for both excitatory and inhibitory inputs together,

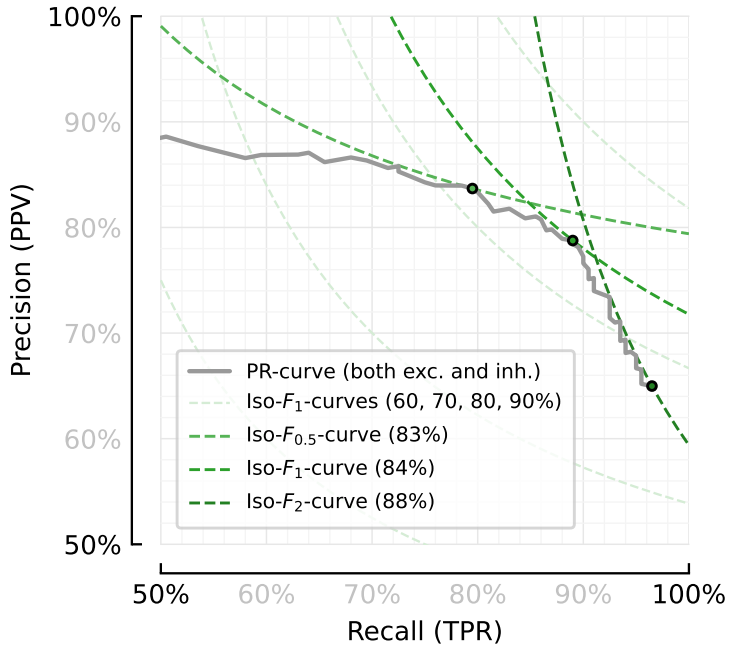

Figure 24: Precision plotted against recall for the STA-test in the N=6500 inputs, 10-minute-recording experiment. Black dots indicate where three different F_β-scores reach their respective maximum values.

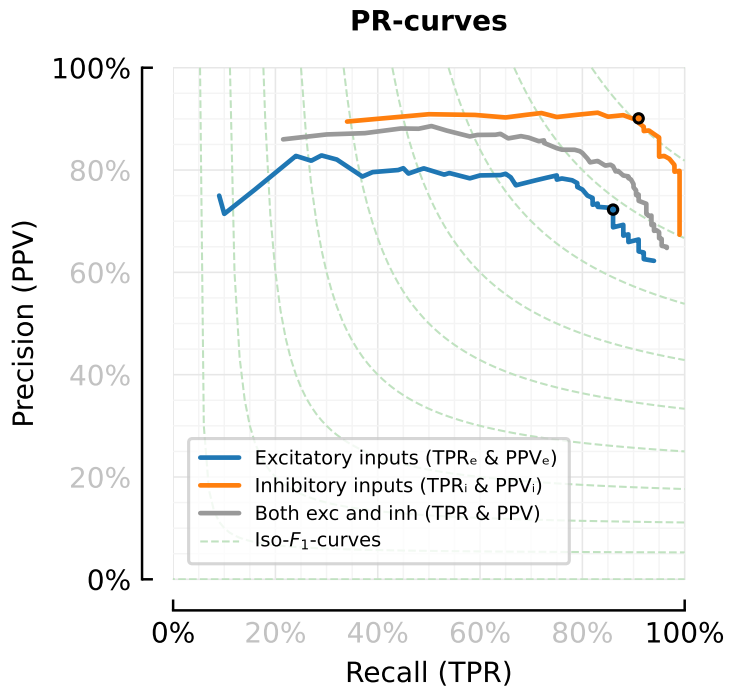

Figure 25: Same as in figure 24, but with excitatory and inhibitory inputs analysed separately. Note that we can detect more inhibitory inputs than excitatory inputs for the same precision value (or for the same false positive rate, as shown in figure 22).

and in figure 25 for both types separately.

Different thresholds yield different F_β-scores; but there is one threshold where your chosen F-score is maximal. This $\max$ F_1-score is a good candidate for the single "how good is this detector" measure we were looking for. We specifically choose F_1 as it weighs precision and recall equally (and we have no a-priori reason to prefer any one over the other), and because it is the most frequently used.

Another common single measure to quantify a classifier's perfor-

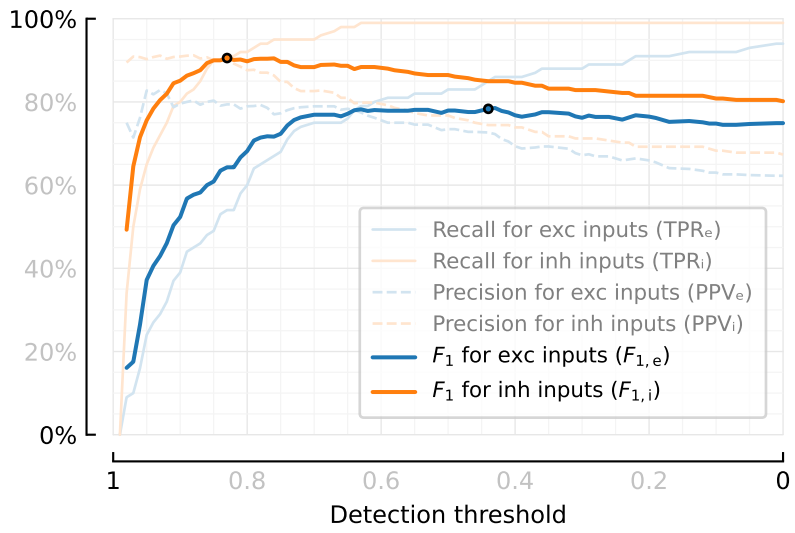

Figure 26: **Excitatory and inhibitory inputs reach $\max F_1$ at different thresholds.**
But for simplicity, whenever we use $\max F_1$ to evaluate a classifier, we will use only one threshold for both types of inputs, which will be a compromise between these two thresholds.

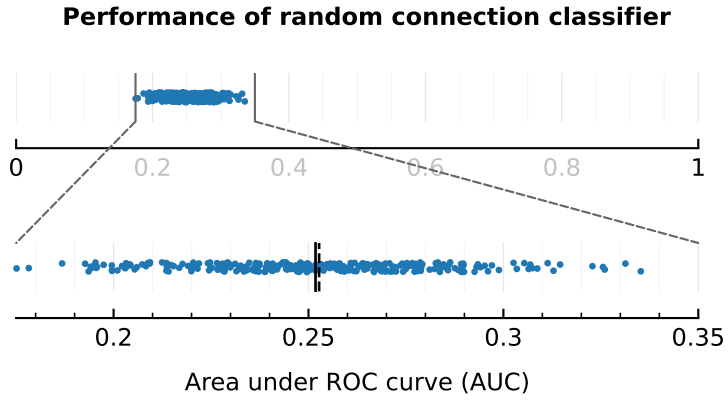

Figure 27: **Randomly classifying connections yields a chance level AUC < 0.5.**
300 random test results, and their performance as connection detector, quantified as area under their ROC curves. Solid black line is the mean, dashed line is the median.
In every of the 300 simulations, every connection (100 exc, 100 inh, and 100 unconnecteds) was assigned a random 't-value' uniformly between -1 and 1; and then the classification threshold was swept over these t-values.

mance is the area under its ROC-curve, or AUC, already shown in figure 22. A disadvantage of the AUC is that it is less interpretable as a number than the $\max F_1$ score (which immediately gives a rough idea of how many true connections you'll detect, and how many of your detections are correct). A disadvantage of the $\max F_1$ score is that it uses the precision, which, as discussed above, is rather arbitrary in our setup. AUC does not suffer this problem: the FPR is independent of how many distractors are added to the test.

To make the AUC scores somewhat more interpretable, we compare them to the AUC of a random classifier: a detector that classifies possible connections randomly. For simple binary classification, this results in an AUC of 0.5. But because we have three classes (unconnected, excitatory, and inhibitory), the random AUC will be lower. We simulated such random classifiers to find the chance-level AUC (figure 27), which turns out to be about 0.252.

3.3 Recording duration & noise

In this section we look at how longer or less noisy voltage imaging recordings improve connection inference. In figure 28, the signal-to-

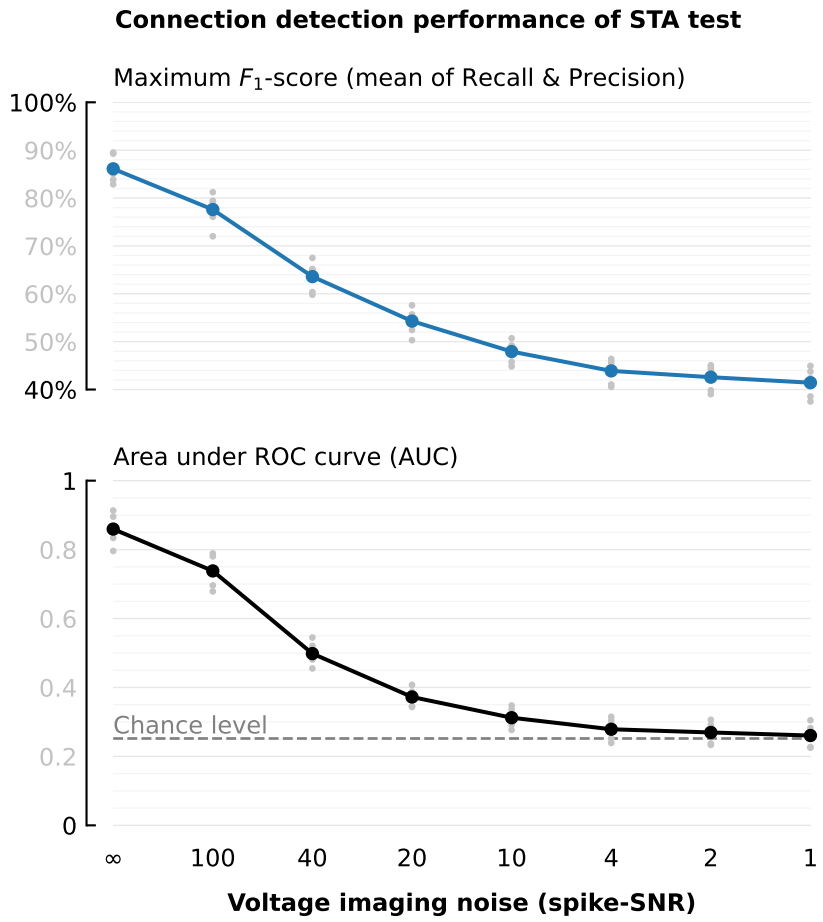

Figure 28: **Performance drops to chance level for noisier signals**. All simulations were 10 minutes long. Signal-to-noise (SNR) values on the x-axis are approximately (but not exactly) log-spaced. An SNR of '∞' corresponds to no noise (i.e. the voltage signal straight out of the simulation, without any noise added). For every SNR value, five different simulations were run (gray dots), each with a different RNG seed for input firing rate and spiketrain generation. The mean performances of these five simulations are plotted with larger dots and are line-connected. Only the 100 highest-firing excitatory and inhibitory inputs were tested. An additional 100 unconnected spiketrains were generated and tested, with similar firing rates as those 200 high-firing real inputs.
AUC chance level determined as in figure 27.
Source: `2023-09-20__STA_conntest_for_diff_recording_quality_n_durations`.

noise ratio (SNR) is varied, and in figure 29, we vary the recording duration.

As might be expected, noisier and/or shorter recordings decrease detection performance, down to chance level in the limit (namely: for noise levels almost as high as the spikes; and for recordings shorter than a minute). As to recording duration, interestingly, we do not yet see any flattening off of the detection performance curve for longer recording duratoins, up to the durations that we simulated (up to 1 hour).

For a concrete example of what a neuroscientist might expect from the STA-based connection test, we find that for a 10-minute recording with an SNR of 40 (which are more or less realistic for voltage imaging), the maximum F_1-score – for the 200-highest firing inputs, and an additionally tested 100 unconnected spiketrains – is about 65%.

I.e, in the N-to-1 setup with 6500 inputs, for the 200 highest-firing of those inputs, we detect approximately 130 of them as being connected (±65% recall). And of the spiketrains that we detect as inputs, about 65% are correctly classified (±65% precision). I.e. 35% of them are either excitatory connections classified as inhibitory and vice-versa; or they are random unconnected spiketrains classified as real inputs.

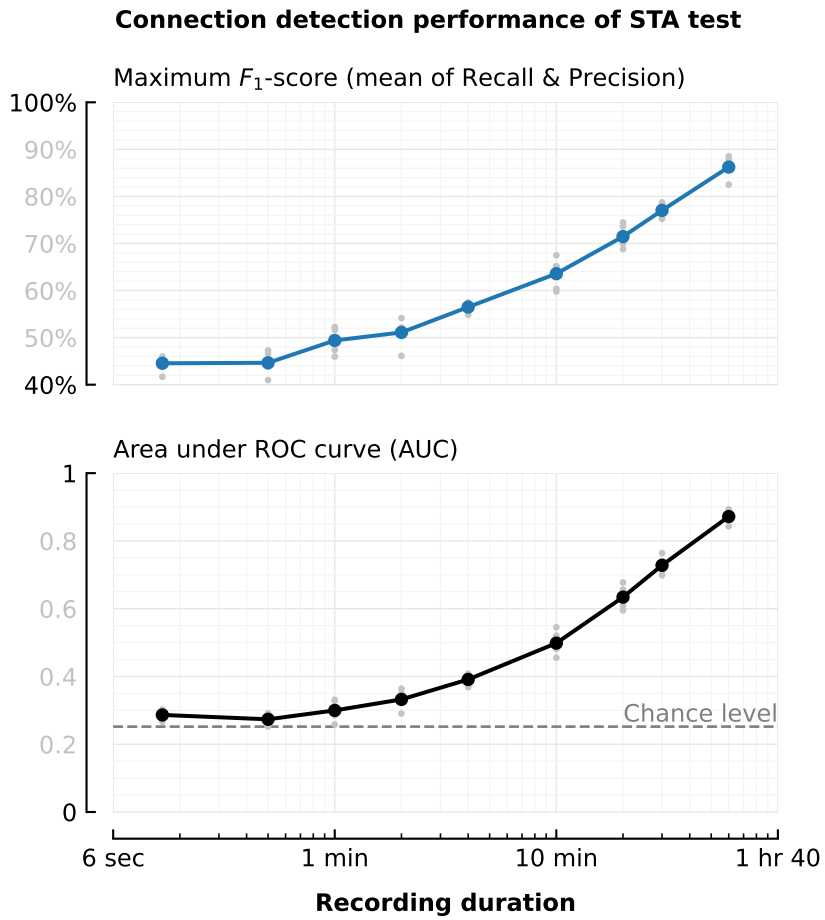

Figure 29: **Longer recordings allow more accurate connection inference.**
All simulations had a voltage imaging noise level (spike-SNR) of 40.
The simulation ('recording') durations are on a logarithmic axis. (The first two data points are at 10 and 30 seconds; the last one is at 1 hour).
For more, see figure 28's caption.

The area under the TPR/FPR-curve (AUC) for this recording duration and quality is about 0.50 (compared to the chance level of 0.25 – see figure 27).

3.4 Computational cost of STA test

The brunt of the time in using the STA as a connection test is in actually calculating these STAs – and the STAs of the shuffled versions of the tested spiketrains.

Table 4 lists different parameters that determine how long a connection test takes to run.

The chosen inputs to test (300 high firing trains) have a median firing rate of 16 Hz. I.e. at a simulation duration of 10 seconds, there are about 160 presynaptic spikes per tested connection. There are 101 times that many STAs to calculate per connection: once for the real spiketrain, and a 100 times for shuffles of it. For a 10 second simulation, there are thus about 16k STAs to calculate per connection. For 10 minutes: 967k STAs. For 1 hour: 5.8M STAs.

The computation time of the STA-based test thus scales linearly with the voltage imaging recording duration. We find that testing 300 possible spiketrain inputs to one neuron takes about one-fifth of the

Factor		Description	Value	Unit
N_{post}		Number of analyzed voltage signals (number of postsynaptic neurons)	1	voltage signals
N_{pre}		Number of tested spiketrains (possible presynaptic neurons) per 'post' neuron	300	spiketrains
N_{conn}	$= N_{\text{post}} \cdot N_{\text{pre}}$	Number of connections tested	300	connections
T		Duration of simulation or recording	10	minutes
λ		Firing rate of presynaptic neuron	16	spikes / second
N_w	$= T \cdot \lambda$	Number of windows per tested conn.	9600	windows
T_w		Window length	20	ms
Δt		Timestep of simulation or recording	0.1	ms
f_s	$= 1/\Delta t$	Sample rate	10	samples / ms
M	$= T_w \cdot f_s$	Number of samples per window	200	samples

Table 4: **Factors in how long a connection test takes to run.** The listed values are ones typically used in this thesis.

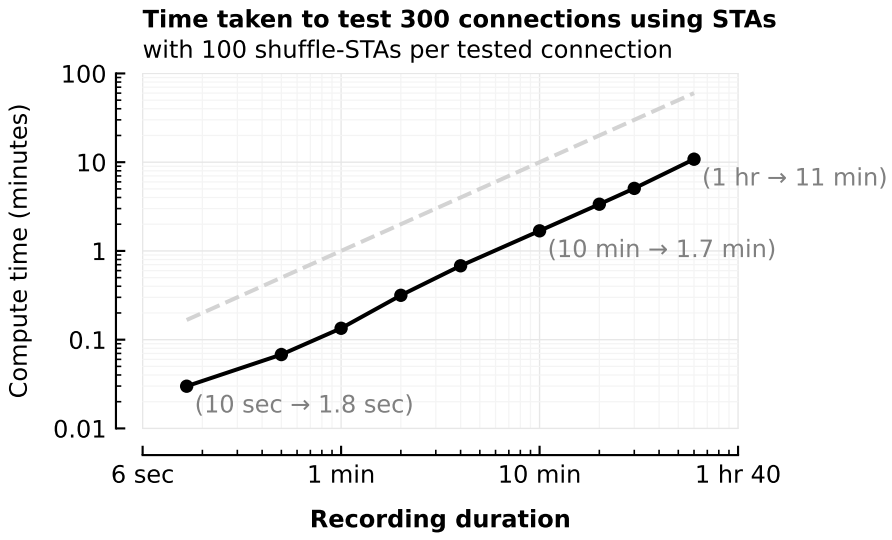

Figure 30: **Test time scales linearly with voltage signal duration**. Simulation timestep ('sample time') of 0.1 ms. STA length of 20 ms; i.e. 200 samples.
Black dots are the means over five simulations per duration. Compute times for individual simulations are plotted with gray dots; but the variation is so small that these gray dots are hidden behind the black means. Gray dashed line is the $y = x$ identity. Source: `2023-09-20_STA_conntest_for_diff_recording_quality_n_durations`.

time of the recording (figure 30): about 2 minutes of computation time for a 10-minute recording.

3.5 Conclusion

Using spike-triggered-averages of the postsynaptic voltage is an obvious way to look for post-synaptic potential bumps, and to thus detect neuron connections.

In this chapter we have found that this STA method indeed does significantly better than chance at detecting conections, under realistic conditions (6500 EI-balanced inputs to one neuron, 10 minute

recoding, voltage imagaing SNR of 40). Of the highest firing inputs (100 excitatory, 100 inhibitory), we detect about 65%, at a precision level where 65% of our detections are correct. The AUC is about 0.50.

This is significantly better than chance (AUC at chance level is 0.252), but there is *a lot* of room for improvement. Thus, we looked for new voltage-based connection detection methods. Our findings are described in the following chapter.

Chapter 4

New connection inference methods

4.1 Introduction

This chapter introduces three new methods to detect synaptic connections from voltage signals.

The first two methods are still based on the STA (spike-triggered average), as in the last chapter. And, also as before, they still use shuffled presynaptic spiketrains[1] to provide a null-distribution for the test statistic

The difference is in how the STA is used. In the previous chapter we used the height of the STA ('peak-to-peak', ptp) as a test statistic.
The first new method here instead *correlates* the STA with some template of what the STA of a true connection would look like, to calculate a test statistic.
The second method fits an idealized function to the STA, and uses the goodness-of-fit as a test statistic.

The third method steps away entirely from the STA, and instead uses the data of the different spike-triggered windows directly, without averaging them together in an STA.

[1] A note on terminology: when we write 'presynaptic' or 'postsynaptic' in this thesis, we often mean the neuron that we are testing as a *possible* presynaptic / postsynaptic connection.

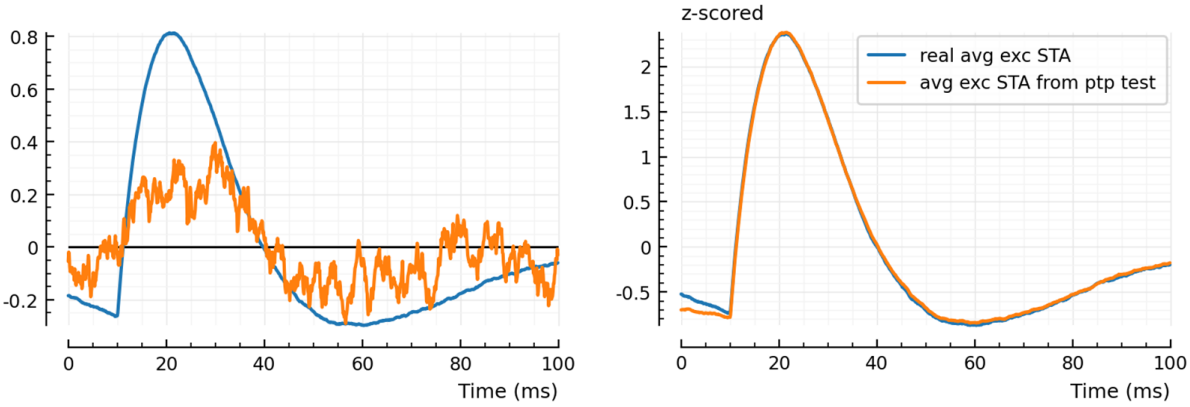

Figure 31: **Correlating the STA with a template.** *Left*: An example STA, and the template it will be correlated with to calculate the connection test statistic.
Right: Two possible STA templates: in blue, the average of the STAs of many true excitatory connections (namely of 50 neurons – part of a 1000-neuron recurrent network – that all had their voltage recorded; and all their excitatory inputs). In orange, the average STA of the excitatory connections detected with a strict 'peak-to-peak' test.

4.2 STA Template correlation

The idea behind this connection test is as follows: the more our actual STA looks like some idealized, 'clean' STA, the more likely the connection exists. We quantify this 'looking like' here by a simple Pearson correlation.

The question then remains what to correlate our real STA with. An ideal template to correlate with would be the average STA of all true connections in the network.[2] The knowledge of these connections is of course not available a-priori (it is what we are trying to infer). But it turns out we can obtain a signal that is very close to this ideal template with a two-step approach.

In the first step, we use the previous test statistic (peak-to-peak height of the STA); but with a stricter p-value threshold (a higher α). This gives us a sample of true connections with few false positives (but with many missed connections). The STAs of these found connections are then averaged, to obtain an estimate for the STA template. We find that this template matches the ideal template closely (see figure 31, right).

In the second step, we correlate this found template with the STAs of all possible connections (and with their shuffle-control counterparts). The correlation values values are then used as test statistic, with the original $\alpha = 0.05$ threshold.

We find that this correlation-based test statistic outperforms the simple peak-to-peak measure (figure 32).

[2] Specifically, either all excitatory connections, or all inhibitory connections. When correlating some connection's STA with an 'excitatory' (upwards) template, the sign of the correlation tells us whether the connection is excitatory ($+$) or inhibitory ($-$).

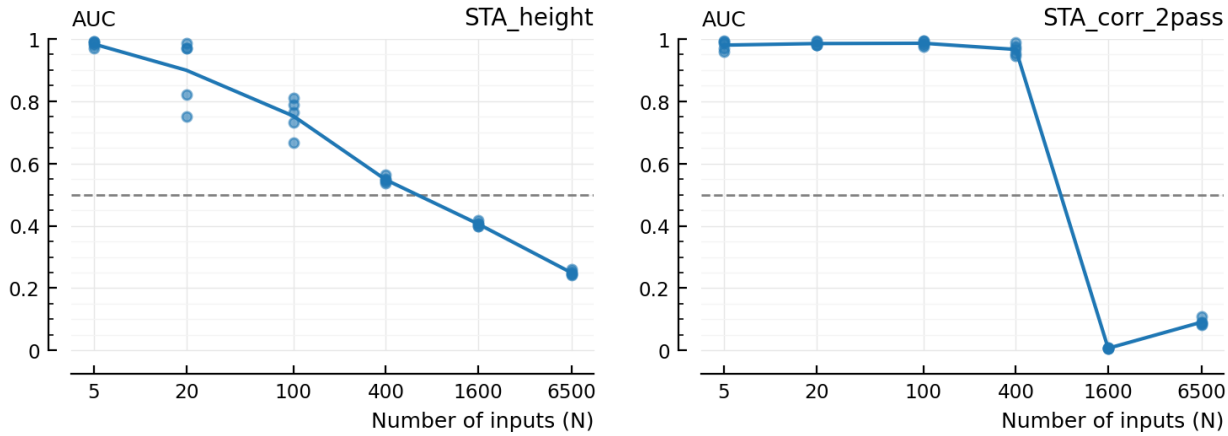

Figure 32: **Correlation outperforms STA height as connection test.** Compare also with figure 35.
AdEx neuron, 10-minute recording, no voltage imaging noise, 5 different seeds (blue dots). All inputs are tested (instead of just the highest firing ones), in addition to 100 random unconnected spiketrains. AUC chance level is at 0.252 (as in figure 27). Note that for the two highest numbers of inputs N, the AUC of the template correlation method falls below chance level. [This is suspicious and must be investigated].
Source: `2023-04-11__Nto1_AdEx_conntest_methods_comparison`.

4.3 Fitting a full STA model

Instead of a data-driven approach for the 'ideal' STA shape, for this method we manually design a function (a 7-parameter function that we'll call f), which we will fit to the actual STAs.

One part of this STA model function is shaped like a postsynaptic potential bump (PSP, figure 33). It is the impulse response of two linear integrators placed in series. Or, in other words, the convolution of two 'step-and-decay' functions.[3, 4↓] It is the postsynaptic potential in the simplest linear neuron model ($\frac{\mathrm{d}v}{\mathrm{d}t} \cdot \tau_\mathrm{m} = -v + I$) where the synaptic current I is also linear ($\frac{\mathrm{d}I}{\mathrm{d}t} \cdot \tau_\mathrm{s} = -I + s$).[5]

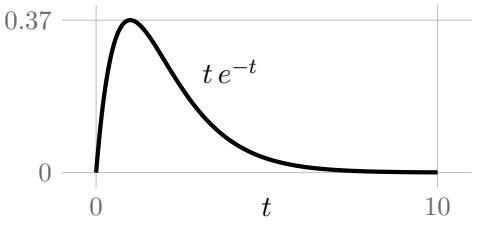

Figure 33: A simple PSP model.

Solving for v and with a single input spike at $t = 0$, we have, for $t \geq 0$:

$$\mathrm{PSP}(t) = \begin{cases} t\,e^{-t/\tau_\mathrm{m}} & \tau_\mathrm{m} = \tau_\mathrm{s} \\ \frac{\tau_\mathrm{m}\tau_\mathrm{s}}{\tau_\mathrm{m}-\tau_\mathrm{s}}\left(e^{-t/\tau_\mathrm{m}} - e^{-t/\tau_\mathrm{s}}\right) & \tau_\mathrm{m} \neq \tau_\mathrm{s} \end{cases} \tag{26}$$

We find that this function approximates the shape of the *simulated* postsynaptic potential in our actual neuron model well – even though our neuron model is not so linear.[6]

But even though equation (26) models our simulated *PSPs* well, it does not resemble our *STAs*[7]; see for example the average STAs in figure 31. First, the bump in the STA does not occur immediately after the presynaptic neuron's spike. This is due to the simulated axonal transmission delay, and it is easily replicated in the model by shifting the PSP function in time. But more interestingly, the STA shows a sort of 'dip', where it dives below baseline just after the PSP bump, and flares up again at the ends (a downwards slant in the initial delay period, and an upwards slant at the end of the STA).

These phenomena thus differentiate an STA from a PSP. We find empirically that they can be parsimonously modelled by subtracting a broad Gaussian curve from the PSP: see figure 34, right.[8]

Our model f is thus the difference of a delayed version of equation (26), and a Gaussian curve. The result is scaled by some factor α, to match the size of the specific STA that is being fit. α can be negative, to model inhibitory connections; f is then flipped upside-down, as in figure 34.

The model function thus has seven parameters: the PSP's delay, τ_m, and τ_s; the Gaussian's location, width, and relative height wrt. the PSP; and the final scaling α. All parameters, except for the Gaussian's, have a ready biophysical interpration.[9]

[3] 'Step-and-exponential-decay': $u(t) \cdot e^{-t/\tau}$, with $u(t)$ the Heaviside step function $\mathbf{1}_{t\geq 0}$.

[4] In computational neuroscience, this function is sometimes called the 'alpha function' or 'alpha synapse'. *Synapse*, because the double integrator model can be used to model synaptic conductance (g), with then a fast rise τ_1 (modelling neurotransmitter release), and a slower decay τ_2 (modelling its dissipation). The synaptic conductance model in our simulation is simpler: the 'rise' is instant, i.e. we only have exponential decay.

[5] τ_s and τ_m are the synaptic and membrane time constants, and $s(t) = \sum_i \delta(t - t_i)$ is the train of input spikes i.

[6] Our model for the membrane potential has a component (the adaptation current u) whose differential equation recurrently depends on v. Additionally, the synaptic currents I_j are *conductance-based*, meaning that they also recurrently depend on v: $I_j = g_j\,(v - E_j)$ (with E_j the reversal potential at synapse j, and the synaptic conductance g_j a linear integrator of the input spike train $s_j(t)$: $\frac{\mathrm{d}g_j}{\mathrm{d}t} \cdot \tau_{s,j} = -g_j + s_j$).

[7] This falsifies our initial implicit hypothesis that STAs are merely noisy reflections of PSPs.

[8] Note that in figure 34, we fit an *inhibitory* STA, and the model componentes are thus inverted: the delayed PSP bump (blue) is downwards, and the Gaussian 'dip' (orange) is upwards.

[9] Namely: α is proportional to the connection strength, and indicates excitation (+) or inhibition (−). The PSP's delay, τ_m, and τ_s are estimates of the axonal transmission delay and the synaptic and membrane time constants.

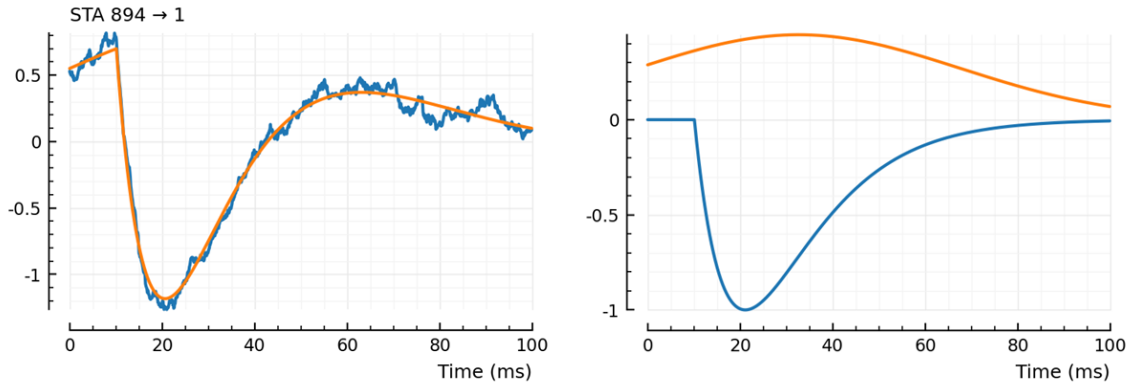

Figure 34: **Fitting a parametric model to the STA.** *Left*: the model f (orange) fit to the STA of an example inhibitory connection from a network simulation (blue). Both signals are z-scored. The components of the orange model function are shown on the *Right*: the delayed simple PSP model (blue) and the Gaussian 'dip' (orange).

We fit the model function to individual STAs using nonlinear least-squares optimization; more specifically using the Levenberg-Marquardt algorithm.[10] To make the fitted functions well-behaved and looking like actual STAs, we had to limit the creative freedom of the optimization algorithm, by enforcing box constraints on the parameters.

The final test statistic is the mean squared error (MSE) between z-scored versions of the fitted function and the STA. The normalization by z-scoring is necessary to be able to compare goodness of fit between STAs with different ranges[11]

In comparison with the previous two methods (STA height and STA template correlation), this model-fitting approach did not perform significantly better than STA height, and it was clearly worse than template correlation.

The model-fitting is also very computationally expensive: iteratively fitting a nonlinear function to an STA takes much longer than simply calculating a Pearson correlation between two STA signals.

For these two reasons, we have not performed an extensive performance evaluation of this method, as we have done for the previous and following methods (figures 32 and 35 respectively).

One advantage of this model-fitting method over the other methods though, is that most of the fitted parameters have a ready biophysical explanation: connection strength, transmission delay, and synaptic and membrane time constants.

[10] This is the standard method for nonlinear least-squares optimization. For example, in MATLAB's `lsqcurvefit` function, it is the default algorithm when the number of functions (in our case, 1) exceeds the number of datapoints (in our case, 1000: 100 ms of signal post-spike, at $\Delta t = 0.1$ ms).
We used the Levenberg-Marquardt implementation of the Julia package LsqFit.jl.

[11] Most importantly between a real STA and its shuffled versions. Because those shuffled STAs have a smaller range, they will have a lower MSE, even though the fit is visually clearly worse.

4.4 Linear regression of the upstroke

All previous methods are STA-based, i.e. different windows are cut from the postsynaptic voltage signal – one window for every presynaptic spike – and these are averaged into one signal, which is then used for further analysis. The method described in this section does not use STAs. We still construct the spike-triggered windows, but we do not average them. Instead, we use them to construct a gigantic data matrix for use in a linear regression.

Every single timepoint on the x-axis (in relative time after the presynaptic spike) will correspond to multiple voltage values on the y-axis; namely one for every window. Whereas for an STA, every timepoint on the x-axis corresponds to just a single y-value (the average voltage).

Why a linear regression? As we saw with the STAs in previous sections, they do not look like a line. But their first part, the "upstroke", might be approximated by one.

Let's number our spike-triggered windows $1, 2, .., N$ (for a presynaptic spiketrain with N spikes), and let's number the voltage values in window i as $[y_{i,1}, y_{i,2}, .., y_{i,M}]$ (for a window length of M samples long).[12] We will then perform the following linear regression:

[12] This window length M is an important parameter. The window must be approximately as long as the upstroke of the STA/PSP: not longer (so that the shape is too complex to be fit by a line), nor shorter (so that there would not be enough data for a proper fit).

$$\boldsymbol{y} = X \boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{27}$$

$$\begin{bmatrix} y_{1,1} \\ y_{1,2} \\ \vdots \\ y_{1,M} \\ y_{2,1} \\ y_{2,2} \\ \vdots \\ \vdots \\ y_{N,M} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ \vdots & \vdots \\ 1 & M \\ 1 & 1 \\ 1 & 2 \\ \vdots & \vdots \\ \vdots & \vdots \\ 1 & M \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \varepsilon_{1,1} \\ \varepsilon_{1,2} \\ \vdots \\ \varepsilon_{1,M} \\ \varepsilon_{2,1} \\ \varepsilon_{2,2} \\ \vdots \\ \vdots \\ \varepsilon_{N,M} \end{bmatrix} \tag{28}$$

I.e. we will regress voltage-after-spike against time-after-spike.

If we assume additive Gaussian noise, i.e.

$$\varepsilon_{i,j} \sim \mathcal{N}(0, \sigma^2) \tag{29}$$

then the maximum-likelihood estimate $\hat{\boldsymbol{\beta}}$ of the regression parameters $\boldsymbol{\beta}$ is obtained through minimizing the mean squared error (MSE)[13] between our observed voltage values $\boldsymbol{y}$ and the fitted line $\hat{\boldsymbol{y}} = X\hat{\boldsymbol{\beta}}$:

[13] https://statproofbook.github.io/P/slr-mle

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} ||\boldsymbol{y} - X\boldsymbol{\beta}||_2^2. \tag{30}$$

Our linear regression problem has become standard ordinary least squares (OLS) regression, for which a closed-form solution exists:

the so-called normal equations,

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \boldsymbol{y} \tag{31}$$

As is standard practice and for numerical stability ("don't explicitly invert a matrix if not needed"), we instead find the optimal intercept ($\hat{\beta}_0$) and slope ($\hat{\beta}_1$) of our linear fit through an implicit solution using QR-factorization, via Julia's left-division operator: $\hat{\boldsymbol{\beta}}$ `= X \ y`.

From this linear fit we thus obtain a slope $\hat{\beta}_1$. To use our regression as a connection test, we will perform a hypothesis test on this slope. The null hypothesis is that this slope is zero ($H_0 : \hat{\beta} = 0$): the voltage of the tested neuron does not react to spikes of the other tested neuron, and on average it stays flat.

If the slope β_1 really were 0, then we would expect the fitted slope $\hat{\beta}_1$ to be distributed normally around 0, as follows:[14]

$$\hat{\beta}_1 \sim \mathcal{N}(0, \sigma^2 [Q]_{2,2}) \tag{32}$$

where σ^2 is the variance of the sampling noise (equation (29)), and $[Q]_{2,2}$ is the second diagonal element of the 'cofactor matrix' Q,[15] which is the inverse of the Gram matrix $X^T X$:

$$Q = (X^T X)^{-1} \tag{33}$$

We do not know what the sampling noise σ^2 is, but we have a maximum-likelihood estimate $\hat{\sigma}^2$ for it,[16] namely the mean squared error (MSE) of the fit:

$$\hat{\sigma}^2 = \frac{1}{n} ||\boldsymbol{y} - X\hat{\boldsymbol{\beta}}||_2^2 \tag{34}$$

(where $n = M \cdot N$ is the number of datapoints).

Our final test statistic t to determine whether there is a connection from neuron A to neuron B is then the slope of the fit (of B's voltage after neuron A's spikes), normalized by how noisy the parameter fit is:

$$t = \hat{\beta}_1 / \hat{\sigma}_{\hat{\beta}_1} \tag{35}$$

where $\hat{\sigma}_{\hat{\beta}_1}$ is the standard error of the slope (equation (32) with equation (34)):

$$\hat{\sigma}_{\hat{\beta}_1} = \sqrt{\hat{\sigma}^2 [Q]_{2,2}} \tag{36}$$

Unlike in equation (32), t is no longer strictly normally distributed.[17] But when the number of datapoints is quite large, as is the case here, the t-distribution becomes practically normal, and our test statistic t will very closely follow the standard normal distribution under the null hypothesis ("the slope is zero").

We can (and do) use this t-value as-is in our connection test, i.e. as the test statistic over which the detection threshold is swept.

[14] https://gregorygundersen.com/blog/2021/09/09/ols-hypothesis-testing/

[15] The indices are off by one ($\hat{\beta}_1$ vs $[Q]_{2,2}$), because the intercept in linear regression is conventionally called β_0, and so we start numbering the elements of $\boldsymbol{\beta}$ from 0; but we number the elements of other vectors/-matrices from 1.

[16] https://statproofbook.github.io/P/slr-mle

[17] Instead, it follows a Student's t-distribution, with $n - p$ degrees of freedom, where $n = M \cdot N$ is the number of datapoints, and $p = 2$ is the number of parameters of the regression (β_0 and β_1).

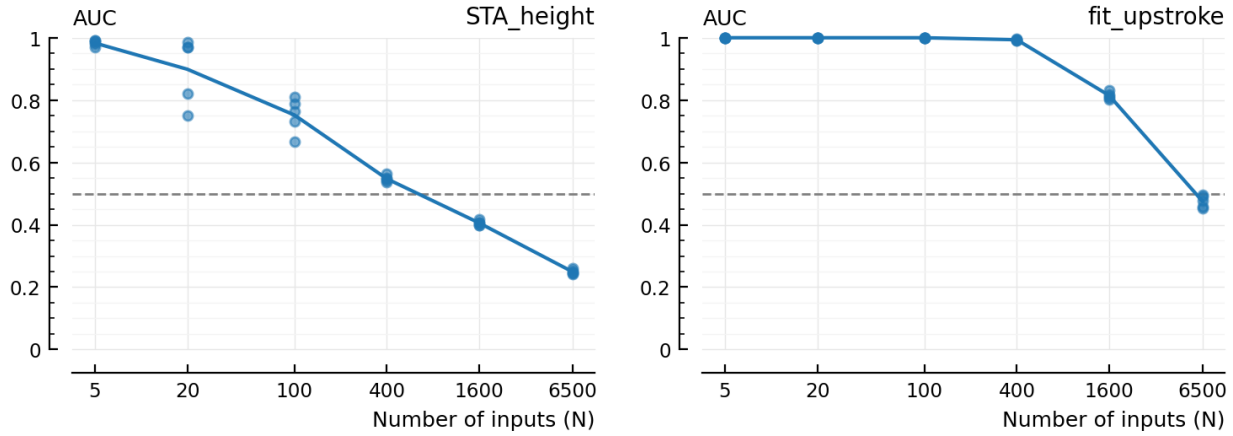

Figure 35: **Line fit outperforms STA height as connection test.** AdEx neuron, 10-minute recording, no voltage imaging noise, 5 different seeds (blue dots). All inputs are tested (instead of just the highest firing ones), in addition to 100 random unconnected spiketrains. AUC chance level is at 0.252 (as in figure 27).
Source: `2023-04-11__Nto1_AdEx_conntest_methods_comparison`.

Additionally, if our assumptions (linear data-generating process, additive Gaussian noise) would be correct,[18] we could also assign an actual probability (a p-value) to observing slopes as extreme as observed. Because t follows the standard normal distribution under the null-hypothesis, and with $\phi(x)$ the cumulative probability function of this distribution, we have: $p = 2\ \phi(-|t|)$: the probability that a sample smaller than $-|t|$ or larger than $|t|$ is drawn.

Figure 35 shows the performance of the described linear regression method, for different number of inputs N, and compared to the STA height method. We find a perfect performance (AUC of 1) for up to 400 inputs. After that, the AUC starts to drop, but it stays much higher than the STA height method. At the realistic number of inputs $N = 6500$, the linear fit's AUC is still at 0.5, while the STA height method performs at chance level (≈ 0.25)

[18] They are not. As one example, consider postsynaptic spikes as one of the sources of noise on the PSP: they are assymetrical (spikes are always upwards, never downwards).

Relationship to fitting the STA

For this last method, we emphasized that it does not operate on the STA, but instead uses the individual windows directly. But an analysis of the normal equations (equation (31)) reveals that fitting a line to all the individual windows pooled together is actually very similar to fitting a line to their average, the STA.

The linear regression on the STA has the following response vector $\overline{\boldsymbol{y}}$ and design matrix $\overline{X}$:

$$\overline{\boldsymbol{y}} \quad = \quad \overline{X} \quad \boldsymbol{\beta} \quad + \quad \overline{\boldsymbol{\varepsilon}} \tag{37}$$

$$\frac{1}{N}\begin{bmatrix} y_{1,1}+y_{2,1}+\ldots+y_{N,1} \\ y_{1,2}+y_{2,2}+\ldots+y_{N,2} \\ \vdots \\ y_{1,M}+y_{2,M}+\ldots+y_{N,M} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ \vdots & \vdots \\ 1 & M \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_M \end{bmatrix} \tag{38}$$

We see that the previous design matrix X consists of N stacked repetitions of the design matrix $\overline{X}$ here.

We can show that both regressions have the same solution for their normal equations, i.e. that

$$\begin{aligned} \hat{\boldsymbol{\beta}} &= (X^T X)^{-1} X^T \boldsymbol{y} \\ &= (\overline{X}^T \overline{X})^{-1} \overline{X}^T \overline{\boldsymbol{y}} \end{aligned} \tag{39}$$

As an example, working out the second element of $X^T\boldsymbol{y}$, we find:

$$\begin{aligned} [X^T\boldsymbol{y}]_2 &= [1\ 2\ \ldots M] \cdot \boldsymbol{y_1} + [1\ 2\ \ldots M] \cdot \boldsymbol{y_2} + \ldots + [1\ 2\ \ldots M] \cdot \boldsymbol{y_N} \\ &= [1\ 2\ \ldots M] \cdot (\boldsymbol{y_1} + \boldsymbol{y_2} + \ldots + \boldsymbol{y_N}) \\ &= [1\ 2\ \ldots M] \cdot N\overline{\boldsymbol{y}} \\ &= N \cdot [\overline{X}^T \overline{\boldsymbol{y}}]_2 \end{aligned} \tag{40}$$

(where $\cdot$ is the inner or dot product, and $\boldsymbol{y_i} = [y_{i,1}\ y_{i,2}\ \ldots\ y_{i,M}]$).

The same scaling holds for the first element of $X^T\boldsymbol{y}$, so that

$$X^T\boldsymbol{y} = N \cdot \overline{X}^T \overline{\boldsymbol{y}} \tag{41}$$

We also find the same scaling for each of the four elements of the Gram matrix, i.e. $X^T X = N \cdot \overline{X}^T \overline{X}$.
That makes the cofacor matrix:

$$(X^T X)^{-1} = \frac{1}{N} (\overline{X}^T \overline{X})^{-1} \tag{42}$$

Taking together equations (41) and (42), we indeed find the exact same solutions $\hat{\boldsymbol{\beta}}$ for the line fit (equation (39)), whether regressing all individual windows, or their average.

Even though the fitted slope $\hat{\beta}_1$ is the same, there is a difference in its standard error (equation (36)), and thus also in the test statistic t we'd use (equation (35)).
We can write the standard error on the slope in the first regression (on all the individual windows) as follows:[19]

[19] where c is shorthand for $\sqrt{[Q]_{2,2}/M}$.

$$\hat{\sigma}_{\hat{\beta}_1} = \frac{1}{N}\, c\, ||\boldsymbol{y} - X\hat{\beta}||_2 \tag{43}$$

$$= \frac{1}{\sqrt{N}}\, c \sqrt{\sum_{j=1}^{M} \frac{1}{N} \sum_{i=1}^{N} \left(y_{i,j} - (\hat{\beta}_0 + \hat{\beta}_1 \cdot j)\right)^2} \tag{44}$$

For the regression of the STA, we have:

$$\hat{\sigma}_{\hat{\overline{\beta}}_1} = c\, ||\overline{\boldsymbol{y}} - \overline{X}\hat{\beta}||_2 \tag{45}$$

$$= c \sqrt{\sum_{j=1}^{M} \left(\left(\frac{1}{N} \sum_{i=1}^{N} y_{i,j}\right) - (\hat{\beta}_0 + \hat{\beta}_1 \cdot j)\right)^2} \tag{46}$$

In other words, the standard error of the STA regression uses the squared error between the fitted line and the average measured voltage, whereas for the regression on all windows, we use the average of the squared errors between the fitted line and all of the individual measured voltages in one timepoint.

These last equations show the advantage of regression with individual windows, versus regressing the STA: the more windows are used (higher N) – i.e. the more presynaptic spikes this possible connection has – the lower the standard error on the fitted slope will be, and the higher the test statistic $t = \hat{\beta}_1/\hat{\sigma}_{\hat{\beta}_1}$. This is thanks to the scaling by $1/\sqrt{N}$ in equation (44), which is absent in equation (46).

This is a quantification of a heuristic that could go something like: "the more 'presynaptic' spikes there are, the more certain we can say there is or isn't a connection here". And when using just the STA, the knowledge of how many windows were used to calculate it is lost.

Two notes here: first, as the fitted parameters $\boldsymbol{\beta}$ are the same when regressing either all windows or the STA, we could just use the STA for fitting, and then manually add a correction factor $1/\sqrt{N}$ to the test statistic t. This is interesting computationally, as you then don't have to construct the very long design matrix X and response vector $\boldsymbol{y}$.
Second, one could argue the standard error of the STA regression *does* already account for the number of windows used: the more windows are averaged together, the less the STA will be noisy and the more it will look like a straight line (at its upstroke portion at least); and the lower the standard error will be.

4.5 Computational cost

- Timings of each method, extrapolation for larger number of tested connections

4.6 Summary

- Conclusions of the N-to-1 experiment
- Leadup to the network experiments: what we could not yet test (the problem of indirect connections, as e.g. identified in the connectomics challenge)

Chapter 5

Network model

Chapter 6

Discussion

6.1 Summary & conclusions

..

6.2 Future work

- Test on real data
- Direct comparison with spikes-only methods
- More complexity in the testing setup: different transmission delays and time constants per synapse / neuron, plus:
- Short term synaptic plasticity. Bursting. Oscillations.
- Simulate different brain areas (different cell types and connectivity patterns). Simulate the same area, but in different states (up vs down, e.g.)
- New connection test method to try: something deep learning-based (we have infinite training data, given our simulation)

References

[Bad+08] Laurent Badel, Sandrine Lefort, Thomas K. Berger, Carl C. H. Petersen, Wulfram Gerstner, and Magnus J. E. Richardson. "Extracting Non-Linear Integrate-and-Fire Models from Experimental Data Using Dynamic I−V Curves". In: *Biological Cybernetics* (Nov. 15, 2008). DOI: 10.1007/s00422-008-0259-4.

[BG05] Romain Brette and Wulfram Gerstner. "Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity". In: *Journal of Neurophysiology* (Nov. 2005). DOI: 10.1152/jn.00686.2005.

[Bre+07] Romain Brette et al. "Simulation of Networks of Spiking Neurons: A Review of Tools and Strategies". In: *Journal of Computational Neuroscience* (Dec. 1, 2007). DOI: 10.1007/s10827-007-0038-6.

[DA01] Peter Dayan and L. F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. Computational Neuroscience. The MIT Press, 2001. 460 pp. ISBN: 978-0-262-04199-7.

[HDZ08] Tomáš Hromádka, Michael R. DeWeese, and Anthony M. Zador. "Sparse Representation of Sounds in the Unanesthetized Auditory Cortex". In: *PLOS Biology* (Jan. 29, 2008). DOI: 10.1371/journal.pbio.0060016.

[Izh07] Eugene M. Izhikevich. *Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting*. Cambridge, Mass.: The MIT Press, 2007. 464 pp. ISBN: 978-0-262-51420-0.

[Nau+08] Richard Naud, Nicolas Marcille, Claudia Clopath, and Wulfram Gerstner. "Firing Patterns in the Adaptive Exponential Integrate-and-Fire Model". In: *Biological Cybernetics* (Nov. 15, 2008). DOI: 10.1007/s00422-008-0264-7.

[OCo+10] Daniel H. O'Connor, Simon P. Peron, Daniel Huber, and Karel Svoboda. "Neural Activity in Barrel Cortex Underlying Vibrissa-Based Object Localization in Mice". In: *Neuron* (Sept. 23, 2010). DOI: 10.1016/j.neuron.2010.08.026. pmid: 20869600.

[Rox+11] Alex Roxin, Nicolas Brunel, David Hansel, Gianluigi Mongillo, and Carl van Vreeswijk. "On the Distribution of Firing Rates in Networks of Cortical Neurons". In: *Journal of Neuroscience* (Nov. 9, 2011). DOI: 10.1523/JNEUROSCI.1677-11.2011. pmid: 22072673.