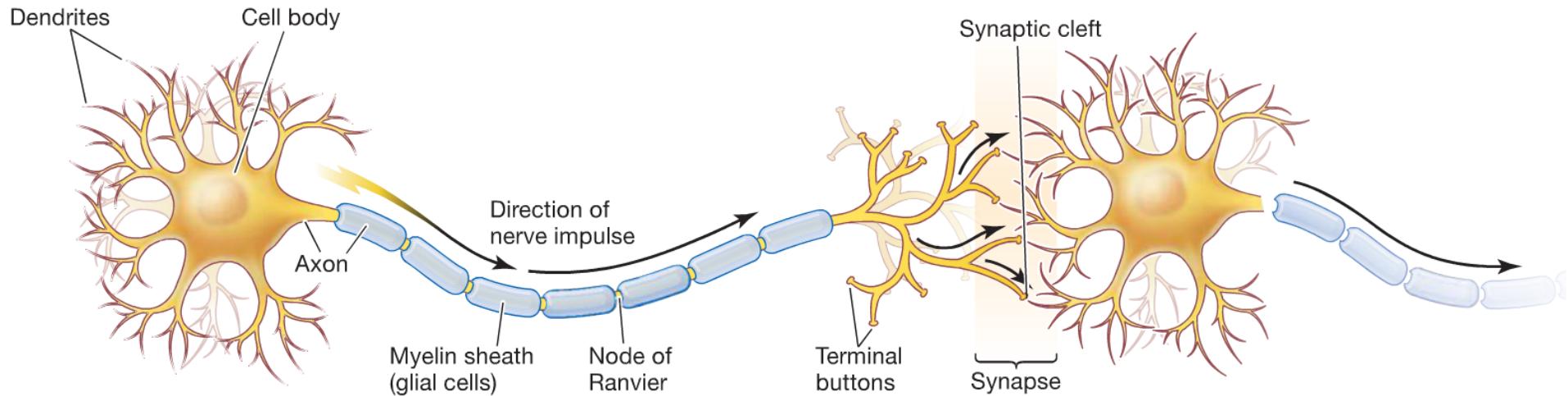


The computational building blocks of the brain



Main thesis - the neuron doctrine:

- The **soma** (site of signals integration), **dendrites** (synapses from source neurons) and **axon** (output pathway on target neurons)
- Neurons and synapses can be described by relatively compact, functional, phenomenological **mathematical models**.
- Neuronal communication can be summarized in binary, asynchronous messages (**spikes**).

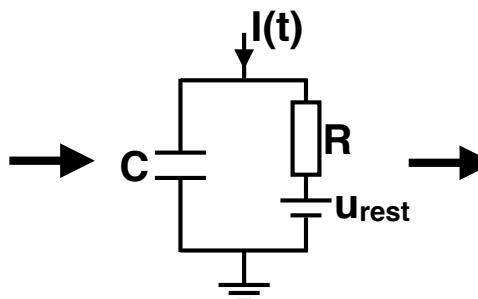
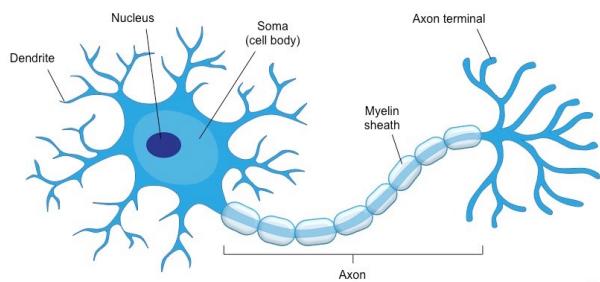
Models of Neuronal Dynamics

The passive membrane model

The simplest - most used electrical-mathematical model for neuronal dynamics

Capacitance for modeling passive membrane behavior - separation of ions. Membrane acts as a leaky capacitor whose voltage, in the absence of injected current, decays (or “leaks”) to a resting voltage level.

Resistance for modeling ion channels

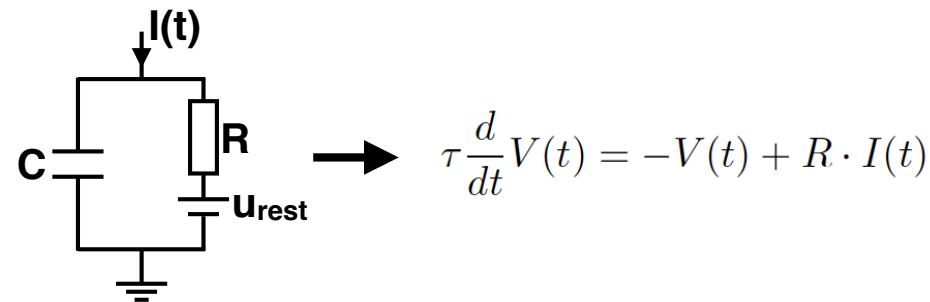


$$\tau \frac{d}{dt} V(t) = -V(t) + R \cdot I(t)$$

see recorded session

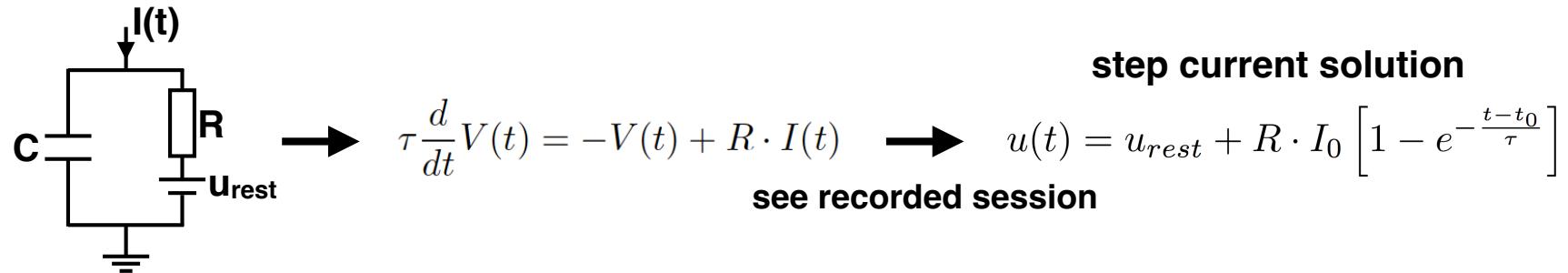
Models of Neuronal Dynamics

The passive membrane model



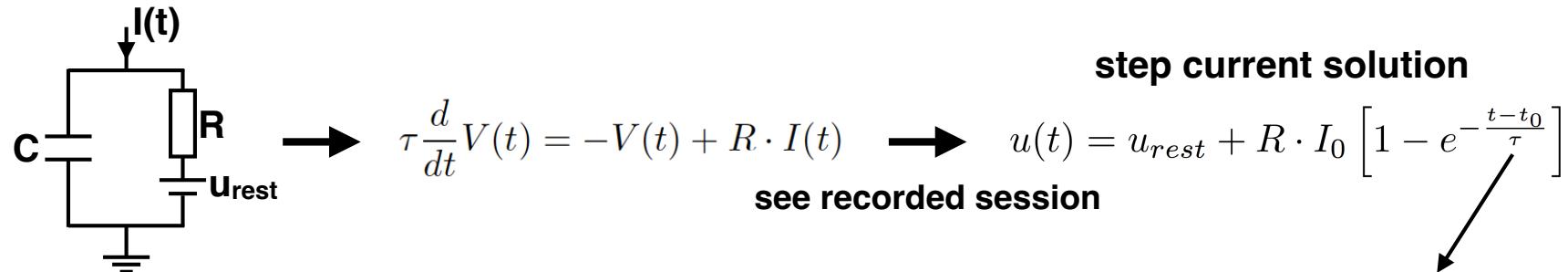
Models of Neuronal Dynamics

The passive membrane model

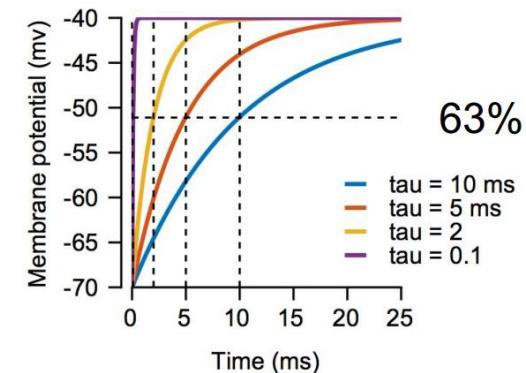


Models of Neuronal Dynamics

The passive membrane model

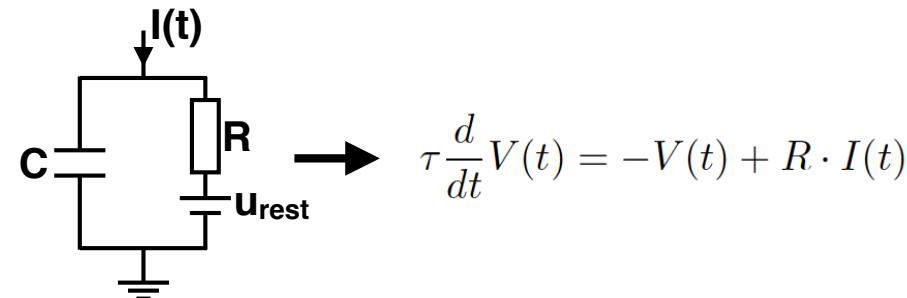


$=RC$, which is the time it took for the membrane potential to develop to 63% of its maximum value ($I(t) \cdot R$)



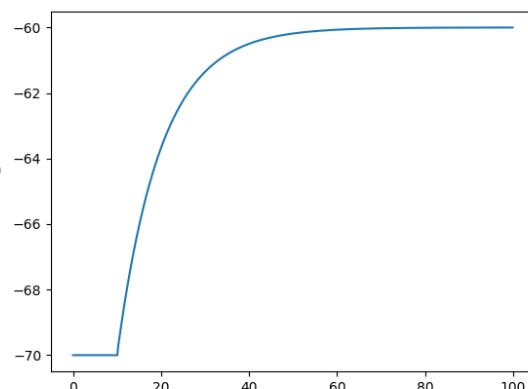
Models of Neuronal Dynamics

The passive membrane model

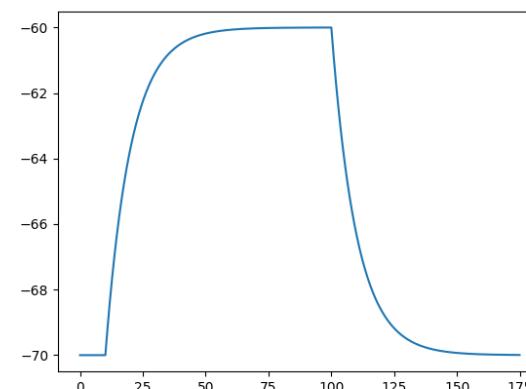


Solutions for **flat input, step current, pulse, short pulse** and **arbitrary input** are given in the lecture's handout

Membrane voltage reacts to a step current stimulation

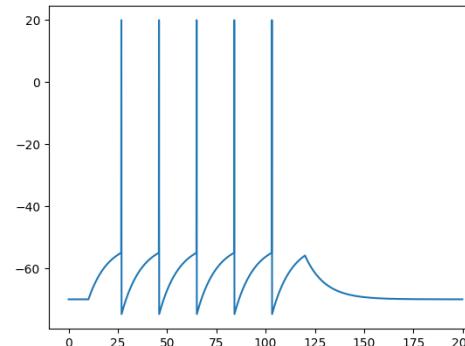
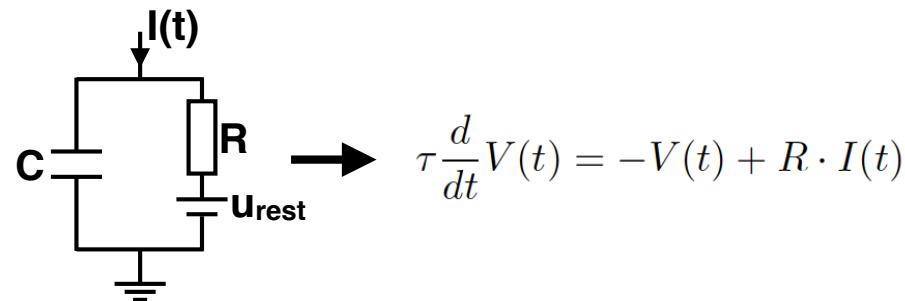


Membrane voltage reacts to a pulse current stimulation



Models of Neuronal Dynamics

The integrate and fire model



- Adding a **threshold behavior**, in which a neuron releases a **spike** of voltage and then **reseted** to a new u_r value, if the membrane voltage reaches a value θ .
- There is **no explicit modeling** of the ion channel kinetics responsible for this **spiking**. Since all action potentials sent down the axon are to a good approximation identical, the exact shape of the action potential is not important. The only informative feature of a neuron's spiking is their **timing**.
- For an input step function, the **largest** the current is, the **faster** the membrane voltage will reach θ . Therefore **firing frequency** is dependent upon the input current. This frequency-current relation is known as the '**f-I curve**'.

Level of abstraction in conceptualizing and modeling the brain

High levels of abstraction

The Cat is Out of the Bag: Cortical Simulations with 10^9 Neurons, 10^{13} Synapses

Rajagopal Ananthanarayanan¹, Steven K. Esser¹
Horst D. Simon², and Dharmendra S. Modha¹

¹IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120

²Lawrence Berkeley National Laboratory, One Cyclotron Road, Berkeley, CA 94720
{ananthr,ssesser}@us.ibm.com, hdsimon@lbl.gov, dmodha@us.ibm.com

In the quest for cognitive computing, we have built a **massively parallel cortical simulator**, C2, that incorporates a number of innovations in computation, memory, and communication. Using C2 on LLNL's Dawn Blue Gene/P supercomputer with **147,456 CPUs** and **144 TB** of main memory, we report two cortical simulations – at unprecedented scale – that effectively saturate the entire memory capacity and refresh it at least every simulated second. The simulation consists of **1.6 billion neurons** and **8.87 trillion synapses** with experimentally-measured gray matter thalamocortical connectivity. We demonstrate nearly perfect weak scaling and attractive strong scaling. The simulations, which incorporate phenomenological spiking neurons, individual learning synapses, axonal delays, and dynamic synaptic channels, exceed the scale of the cat cortex, marking the dawn of a new era in the scale of cortical simulations.



Neuroscience Expert Dr. Henry Markram on the IBM “Cat Brain” Simulation: “IBM’s claim is a HOAX”

Shortly after IBM announced their cat-scale brain simulation, [Henry Markram](#) of the [Blue Brain Project](#) published a very strong criticism of the claim. He called it ***“a mega public relations stunt - a clear case of scientific deception of the public”***. According to Markram these simulations do not come close to the complexity of an ant brain, let alone that of a cat brain.

Markram's first argument was that although the number of simulated neurons roughly equals that of a cat brain, the **model** used for each individual neuron was **trivially simple**. The neurons were modeled as single compartment "dots" completely lacking in biological realism. **Genuine simulation** of real neurons requires solving **millions of times more equations** than were used by IBM. Thus, **not even a millionth of a cat brain was simulated**.



Models of Neuronal Dynamics

The Izhikevich neuron model

- A quadratic integrate-and-fire type model with a recovery variable.
- It can replicate several characteristics of biological neurons while remaining computationally efficient.

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I$$
$$\frac{du}{dt} = a(b \cdot v - u)$$

- v represents the membrane potential of the neuron
- u represents a membrane recovery variable, which accounts for the activation of K⁺ and inactivation of Na⁺ ionic currents.
- In contrast to LIF, it was designed to fit biological observable behavior: the expression $0.04v^2 + 5v + 140$ was obtained by fitting the spike initiation dynamic of a cortical neuron



Models of Neuronal Dynamics

The Izhikevich neuron model

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I$$

$$\frac{du}{dt} = a(b \cdot v - u)$$

$$\text{if } v \geq 30 \text{ mV, then } \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases}$$

- Parameter a describes the time scale for membrane recovery.
- Parameter b describes the sensitivity of the membrane recovery to fluctuations in membrane potential, allowing achieving sub-threshold spiking.
- Parameter c commonly describes post-spike membrane potential ().
- Parameter d describes the membrane post-spike recovery.

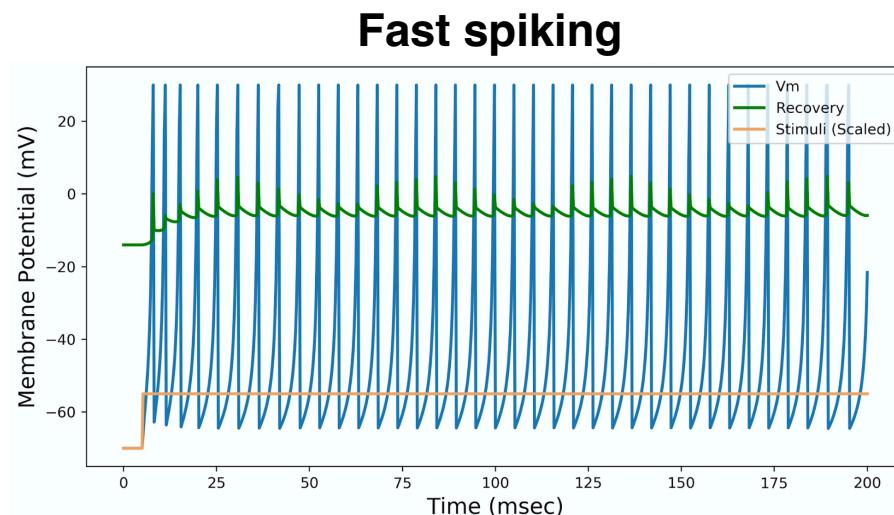
Why is it such a powerful model?



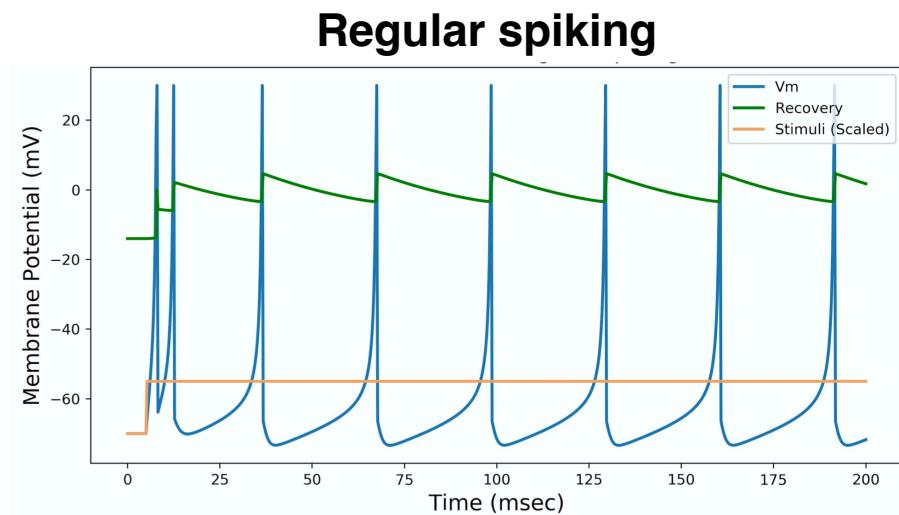
Models of Neuronal Dynamics

The Izhikevich neuron model

- Different value combination of these parameters account for different neuronal dynamic
- This model is frequently used in large scale simulations, such as the once performed by IBM.



$$a = 0.1, b = 0.2, c = -65, d = 2$$



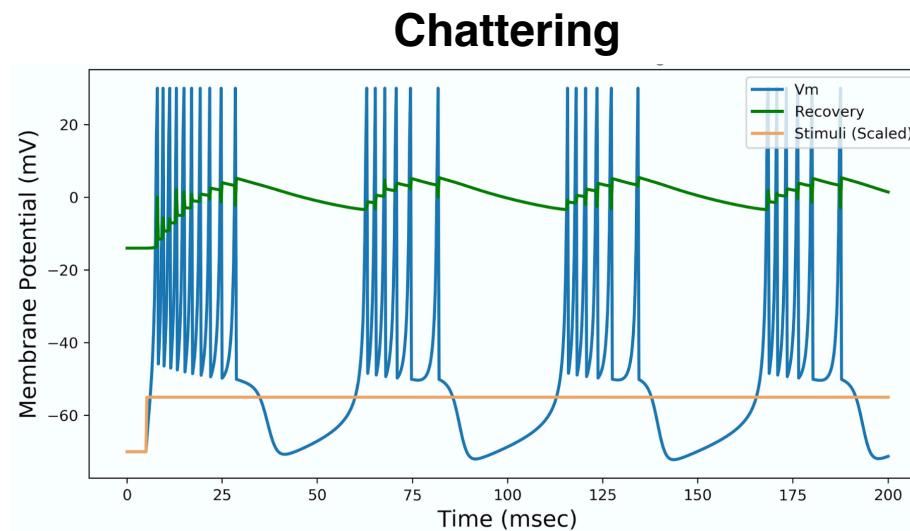
$$a = 0.02, b = 0.2, c = -65, d = 8$$



Models of Neuronal Dynamics

The Izhikevich neuron model

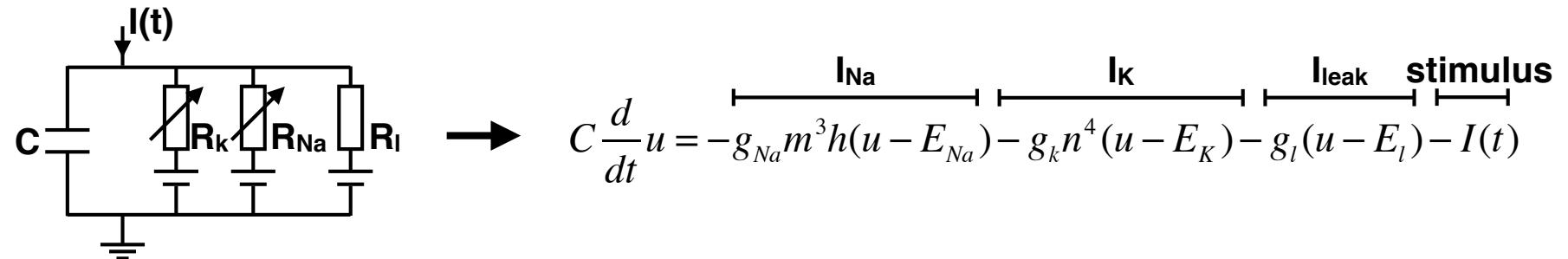
- Different value combination of these parameters account for different neuronal dynamic
- This model is frequently used in large scale simulations, such as the once performed by IBM.



$$a = 0.02, b = 0.2, c = -50, \text{ and } d = 2$$

Models of Neuronal Dynamics

The Hodgkin-Huxley model

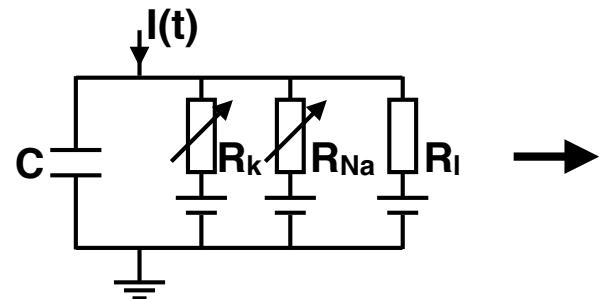


The most famous **biophysically based neuron model**: It models the voltage- and time-dependent conductances of real ion channels

Was originally used to describe the dynamics of the action potential in the squid giant axon but the description of the conductances in terms of activation and inactivation variables is now used ubiquitously to model neurons in a wide range of species and brain areas.

Models of Neuronal Dynamics

The Hodgkin-Huxley model



$$C \frac{du}{dt} = -g_{Na}m^3h(u - E_{Na}) - g_kn^4(u - E_K) - g_l(u - E_l) - I(t)$$

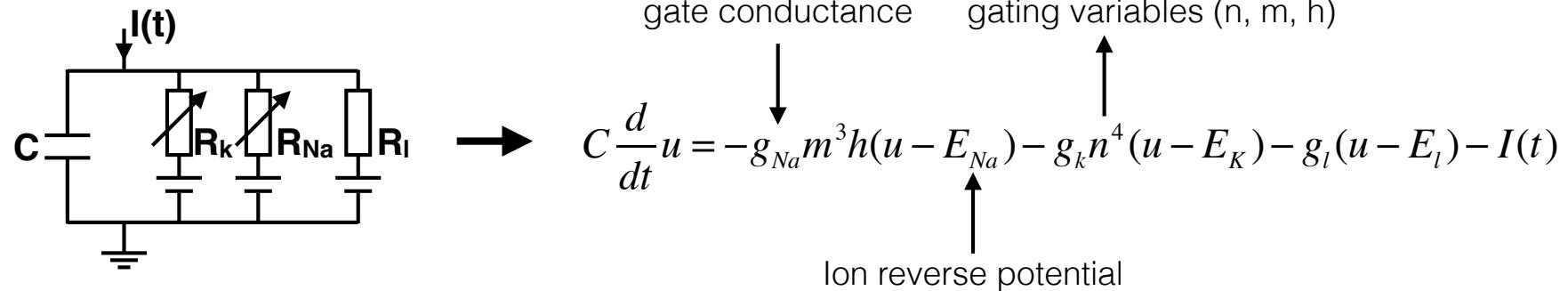
gate conductance gating variables (n, m, h)
Ion reverse potential

Gating variables:

$$\frac{dm}{dt} = -\frac{m - m_0(u)}{\tau_m(u)}$$

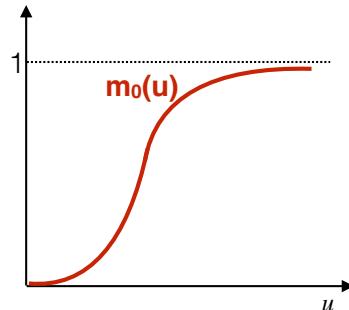
Models of Neuronal Dynamics

The Hodgkin-Huxley model



Gating variables:

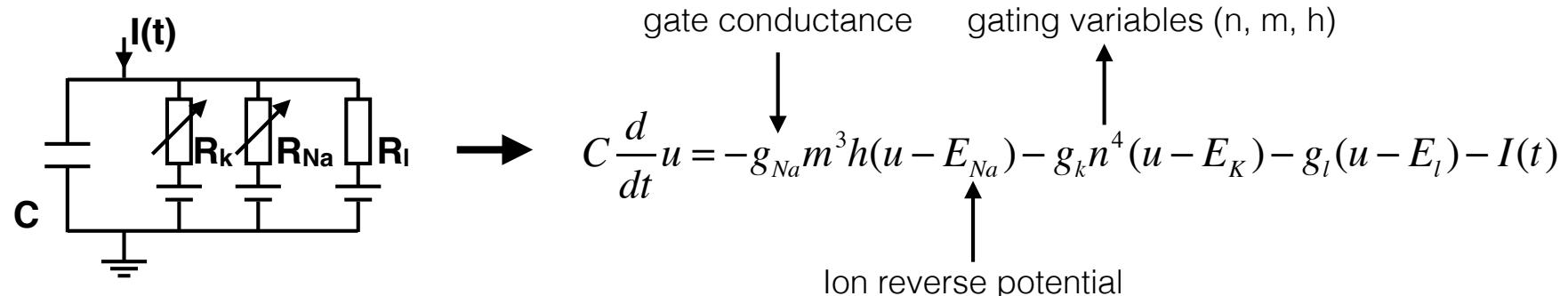
$$\frac{dm}{dt} = -\frac{m - m_0(u)}{\tau_m(u)} \rightarrow m(t) = m_0(u_1) + [m_0(u_0) - m_0(u_1)] \exp\left[\frac{-(t - t_0)}{\tau_m(u_1)}\right]$$



When the voltage is high, m is approaching $m_0=1$ and the gate opens. Values are derived experimentally.

Models of Neuronal Dynamics

The Hodgkin-Huxley model

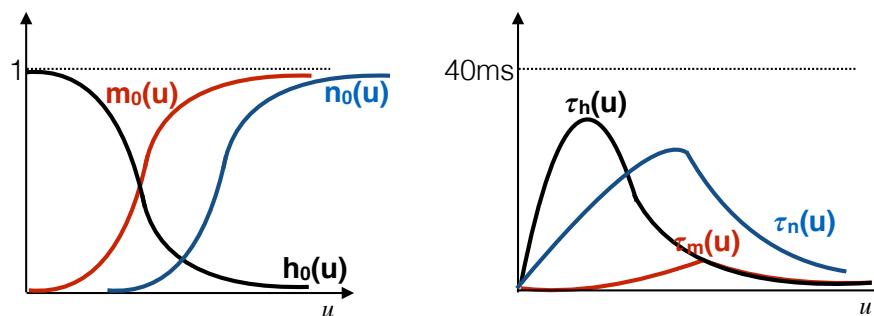


Gating variables:

$$\frac{dm}{dt} = -\frac{m - m_0(u)}{\tau_m(u)} \rightarrow m(t) = m_0(u_1) + [m_0(u_0) - m_0(u_1)] \exp\left[\frac{-(t - t_0)}{\tau_m(u_1)}\right]$$

$$\frac{dn}{dt} = -\frac{n - n_0(u)}{\tau_n(u)}$$

$$\frac{dh}{dt} = -\frac{h - h_0(u)}{\tau_n(u)}$$

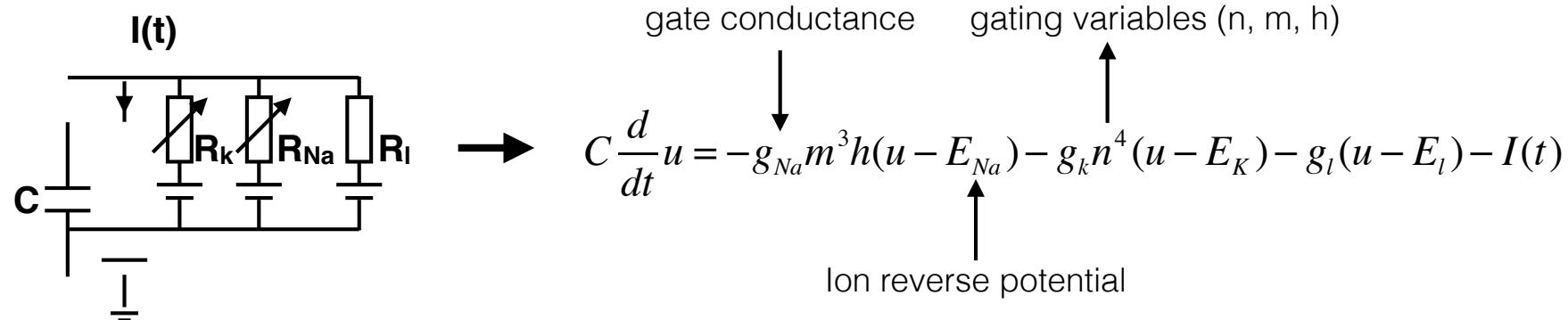


A general mathematical model for ion channels



Models of Neuronal Dynamics

The Hodgkin-Huxley model



Gating variables:

$$\frac{dm}{dt} = -\frac{m - m_0(u)}{\tau_m(u)} = a_m(u)(1 - m) - \beta_m m$$

$$\frac{dn}{dt} = -\frac{n - n_0(u)}{\tau_n(u)} = a_n(u)(1 - n) - \beta_n n$$

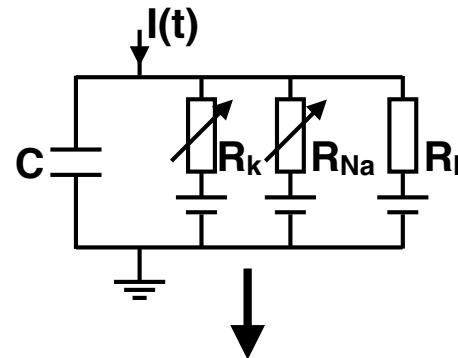
$$\frac{dh}{dt} = -\frac{h - h_0(u)}{\tau_h(u)} = a_h(u)(1 - h) - \beta_h h$$

x	E_x [mV]	g_x [mS/cm^2]
Na	55	40
K	-77	35
L	-65	0.3

x	$\alpha_x(u / \text{mV})$ [ms^{-1}]	$\beta_x(u / \text{mV})$ [ms^{-1}]
n	$0.02(u - 25) / [1 - e^{-(u-25)/9}]$	$-0.002(u - 25) / [1 - e^{(u-25)/9}]$
m	$0.182(u + 35) / [1 - e^{-(u+35)/9}]$	$-0.124(u + 35) / [1 - e^{(u+35)/9}]$
h	$0.25 e^{-(u+90)/12}$	$0.25 e^{(u+62)/6} / e^{(u+90)/12}$



The Hodgkin-Huxley model

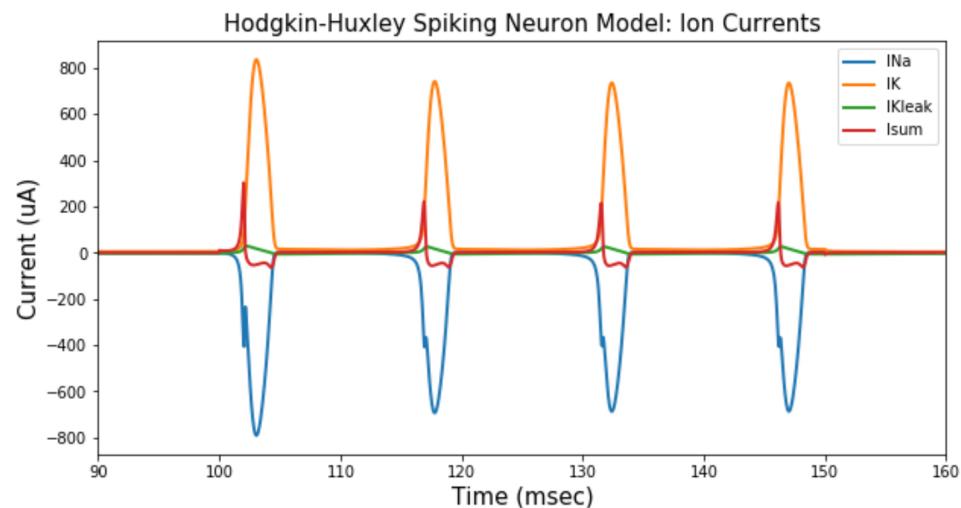
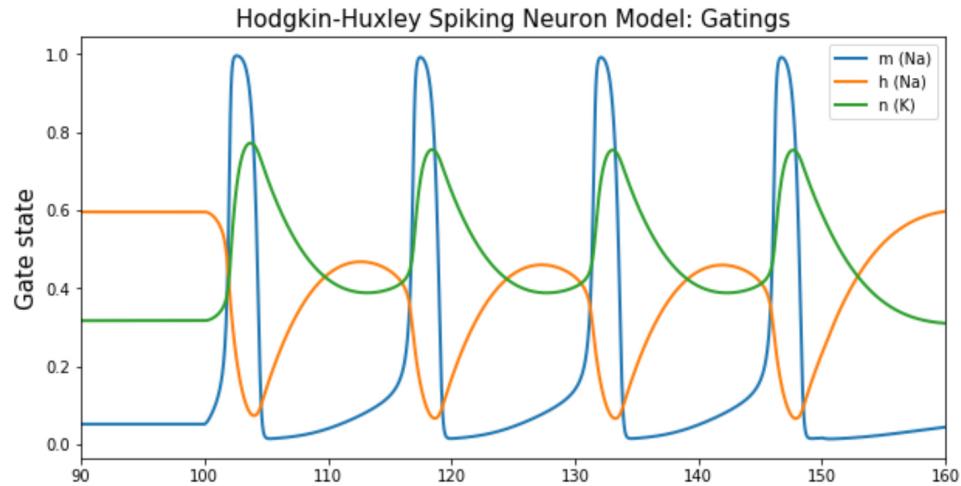
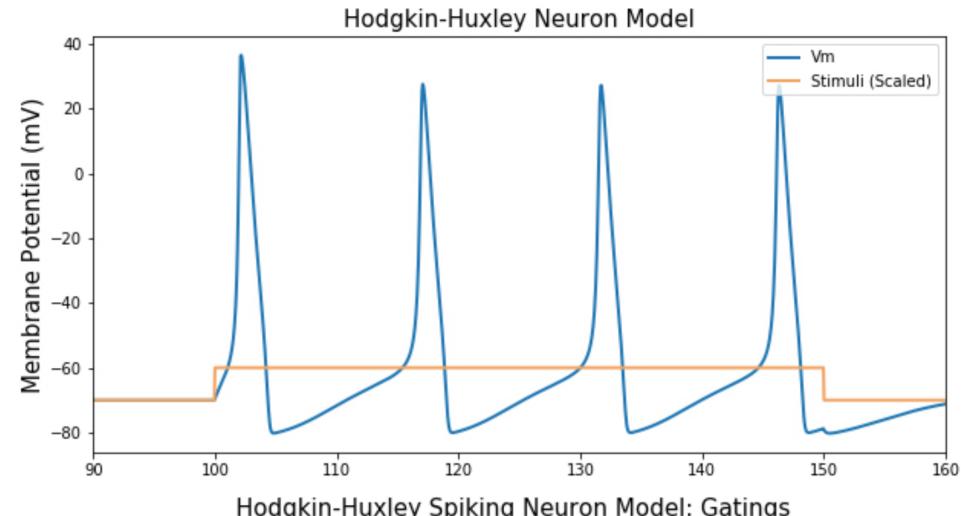


$$C \frac{du}{dt} = -g_{Na}m^3h(u - E_{Na}) - g_kn^4(u - E_K) - g_l(u - E_l) - I(t)$$

$$\frac{dm}{dt} = -\frac{m - m_0(u)}{\tau_m(u)}$$

$$\frac{dn}{dt} = -\frac{n - n_0(u)}{\tau_m(u)}$$

$$\frac{dh}{dt} = -\frac{h - h_0(u)}{\tau_m(u)}$$



Reconstruction and Simulation of Neocortical Microcircuitry

It is known that **different types of neurons** are connected through synapses with **different dynamics** and **strengths**, **strategically positioned** at different locations on the neurons' dendrites, somata, and axons, but the **functional significance** of this organization remains **unclear**.

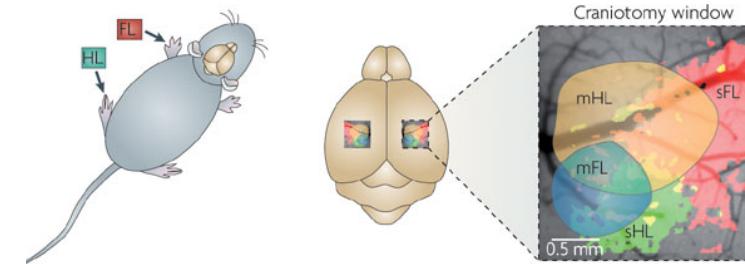
Computational approaches that **abstract away** this level of bio-logical detail have **not been able to explain** the **functional significance** of such intricate cellular and synaptic organization.

Although **future experimental research** will undoubtedly **advance our knowledge**, it is **debatable** whether experimental mapping **alone** can provide enough data to answer these questions.

Here, we present a **complementary algorithmic approach** that reconstructs neuronal microcircuitry across all layers using available sparse data and that **leverages biological principles** and **interdependencies** between datasets to **predict missing biological data**.

Reconstruction and Simulation of Neocortical Microcircuitry

We **digitally reconstructed** a small volume of tissue from layers 1 to 6 of the hind-limb somatosensory cortex of 2-week-old Wistar rat.



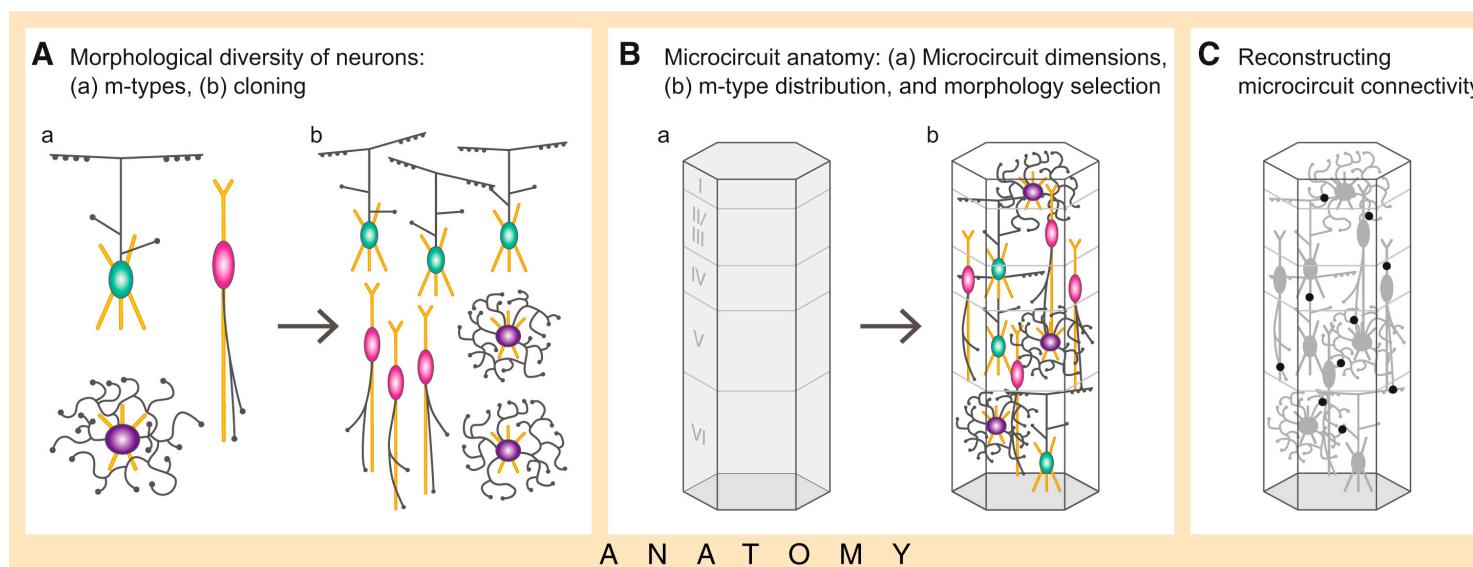
This model system was chosen not only because it is one of the most **comprehensively characterized** in the neocortex, but also because **experimental data** on its cellular and synaptic organization are **readily available** and **validation experiments** are relatively easy to perform.

Reconstruction of anatomical properties

We recorded and **digitally reconstructed neurons** from in vitro brain slices and **classified** the neurons in terms of established **morphological types**

We **positioned** the neurons in a digital volume of objectively defined dimensions according to **experimentally based estimates** of their **layer specific densities**

We reconstructed the connectivity between the neurons

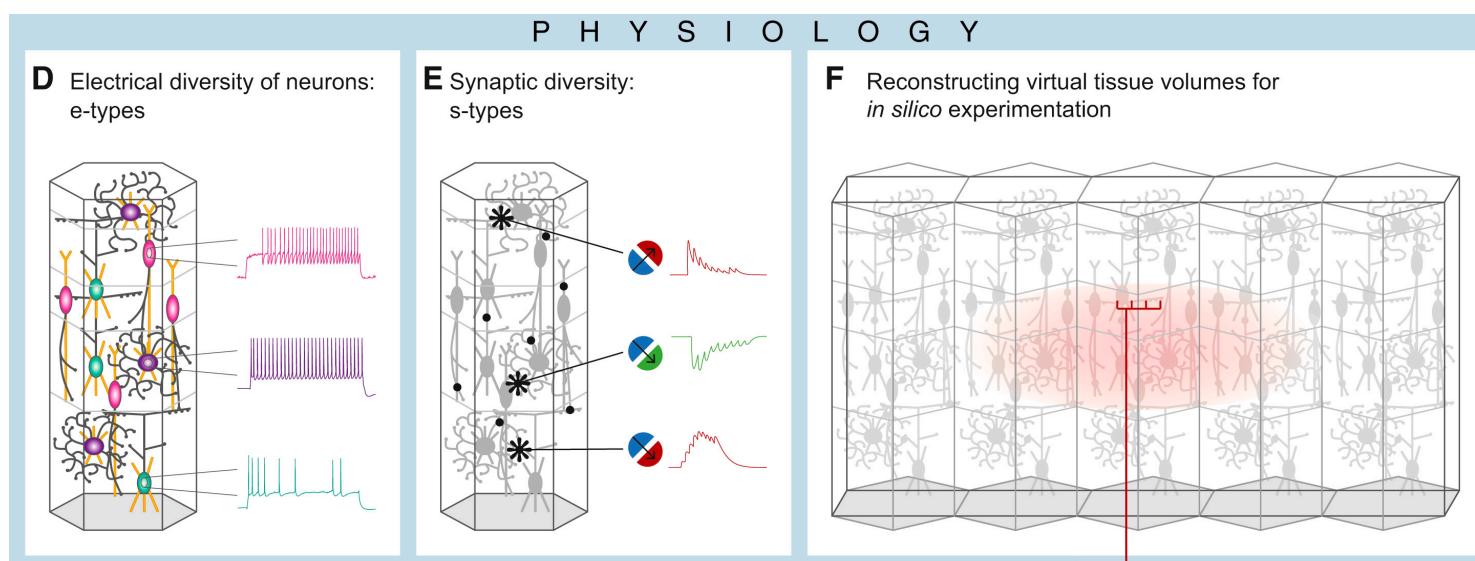


Reconstruction of physiological properties

Neurons were classified into **electrical types** and models were produced that captured the characteristic electrical behavior of each type.

Synapses were modeled to capture the characteristic **synaptic dynamics** of particular synapse types

We constructed a **virtual slice** and reconstructed **thalamic input** using experimental data



Reconstruction and Simulation of Neocortical Microcircuitry

This approach yielded a **first-draft digital reconstruction of the microcircuitry**, which was **validated** against a multitude of experimental datasets **not used** in the reconstruction.

The results suggest that **it is possible** to obtain dense maps of neural microcircuitry **without measuring every conceivable biological parameter** and point to minimal datasets required, i.e., **strategic data**.

Integrating complementary, albeit sparse, datasets also makes it possible to reconcile discrepancies in the literature, at least partially addressing the problem of data quality and reproducibility.

Reconstruction and Simulation of Neocortical Microcircuitry

To **simulate reconstructed microcircuits** at the level of detail described above, the **NEURON simulator was extended to run on supercomputers**

An algorithm was developed to **approximate input from the thalamus** to the central microcircuit in such a way as to satisfy experimental constraints.

Simulations exploring some of the **emergent behaviors of the reconstructed microcircuitry** reproduce a number of previous *in vitro* and *in vivo* findings and **provide insights** into the **design and functioning** of neocortical microcircuitry.

Model predictions include: a spectrum of network states ranging from synchronous to asynchronous activity; extracellular calcium regulates the network state through differential effects on synaptic dynamics; role of layers, neurons, and connection types in modulating network states;

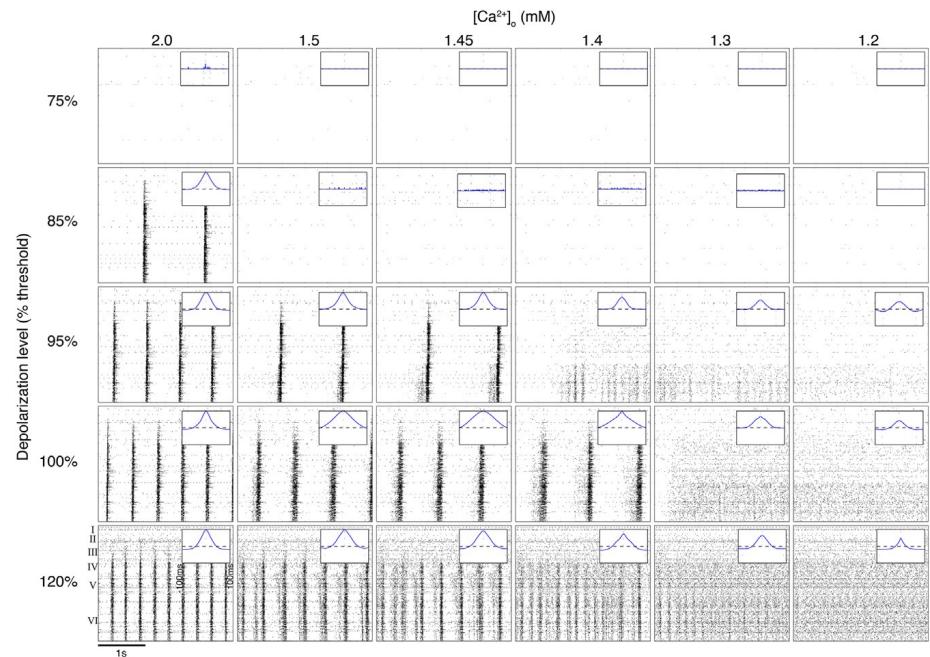
Reconstruction and Simulation of Neocortical Microcircuitry

The simulations suggest that cortical activity in vivo approaches a **critical transition** along the **synchronous asynchronous spectrum**.

Around this transition, spiking activity is highly correlated with fine temporal structure in synaptic input.

Maximal discrimination between spatially segregated inputs, the generation of fine temporal structures such as triplets, and soloist-like and chorister-like behavior all emerge close to the transition.

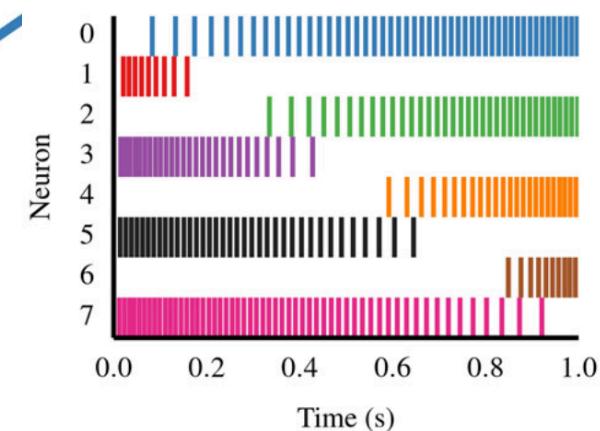
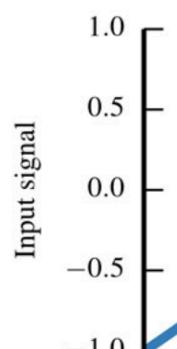
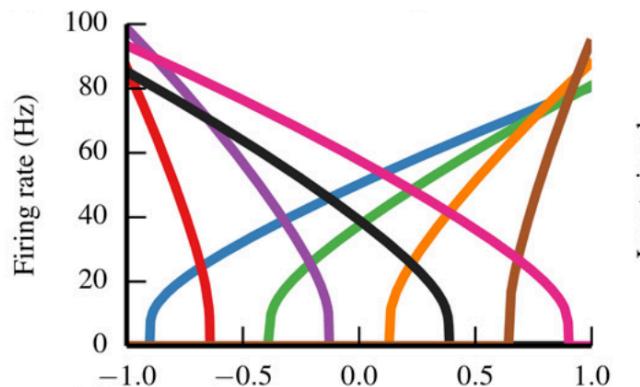
A recent study has experimentally characterized the plasticity mechanisms for maintaining the network close to this transition.



The Neural Engineering Framework (NEF)

Representation

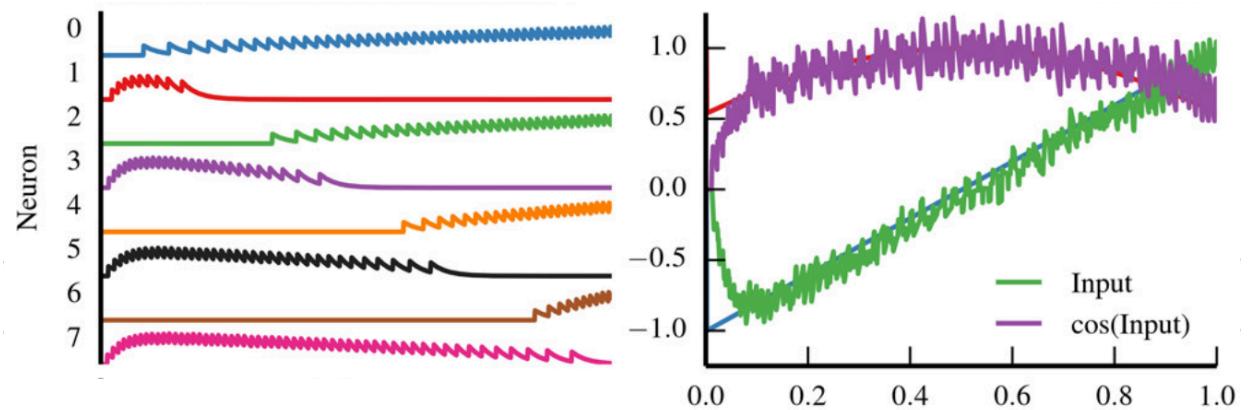
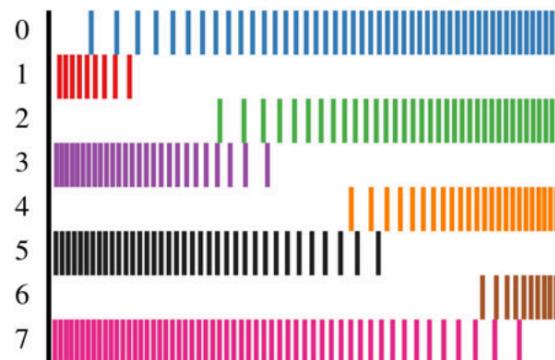
- NEF represents **neuronal information** with **time-varying vectors** of real numbers, which can be manipulated with conventional mathematics.
- In the encoding process, the input signal drives each neuron based on its **tuning curve**, which describes how likely that neuron is to respond to a given input signal (function of the **gain** of a neuron (how quickly the activity rises), the **bias** (the activity of a neuron given no signal), and the **encoding weight**). Tuning curves can be determined for any type of neuron.



The Neural Engineering Framework (NEF)

Decoding

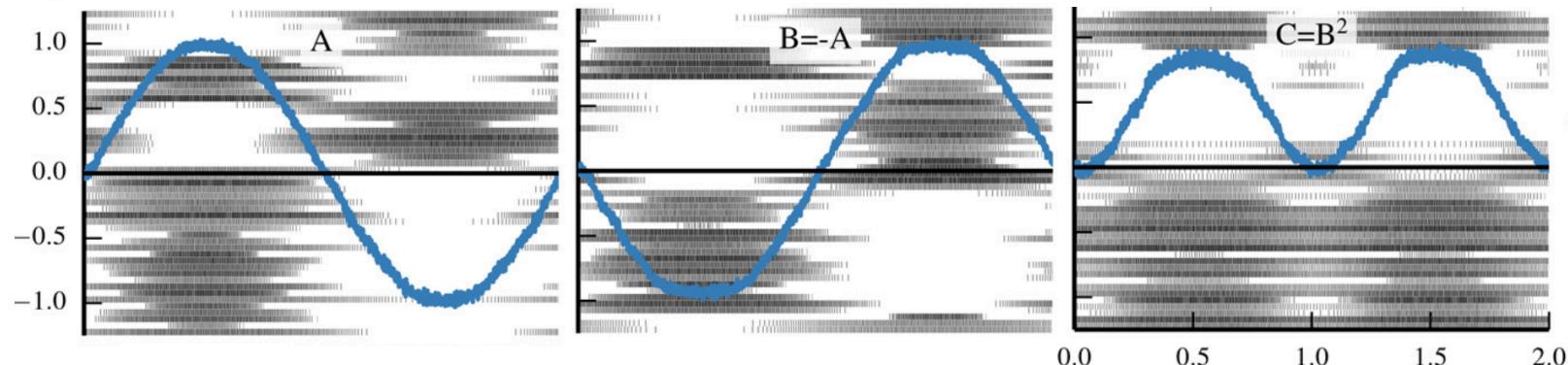
- In the decoding process, the spike trains are first **filtered** with an **exponentially decaying** filter accounting for the process of a spike generating a postsynaptic current. Those filtered spike trains are **summed together** with **weights** that are determined by solving a least-squares minimization problem.
- Decoding weights is typically performed using points sampled from the vector space that the population represents



The Neural Engineering Framework (NEF)

Transformation

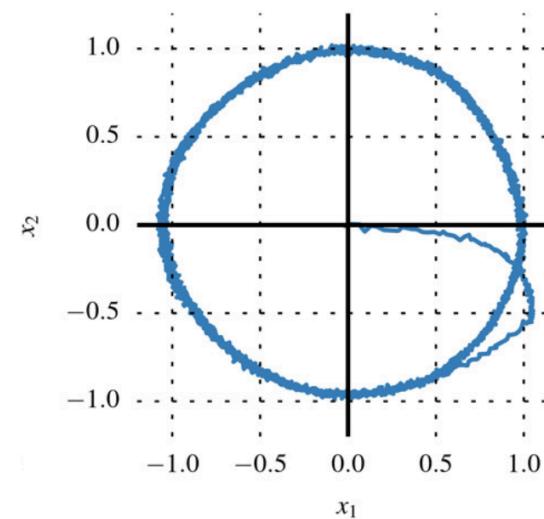
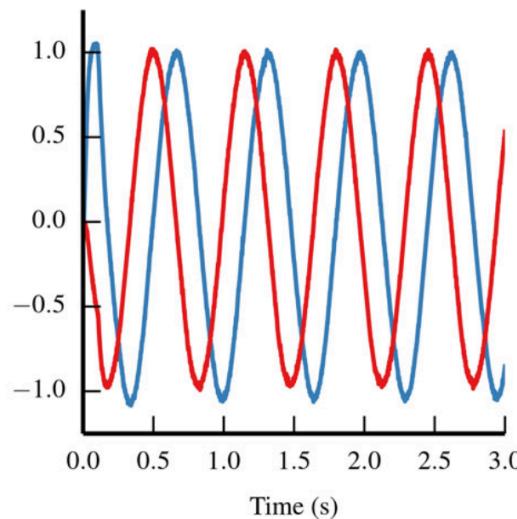
- Many factors affect synaptic strength. With NEF we summarize those factors in a scalar connection weight.
- In order to compute a function, we set the connection weights between two populations to be the product of the decoding weights for that function in the first population, the encoding weights for the downstream population, and any linear transform.



The Neural Engineering Framework (NEF)

Dynamics

- While feedforward vector transformations suffice to describe some neural systems, many require persistent activity through recurrent connections.
- When recurrent connections are introduced, the vectors represented by neural populations can be thought of as state variables in a dynamical system. The equations governing dynamics in such a system can be designed and analyzed using the methods of control theory, and translated into neural circuitry using the principles of representation and transformation.



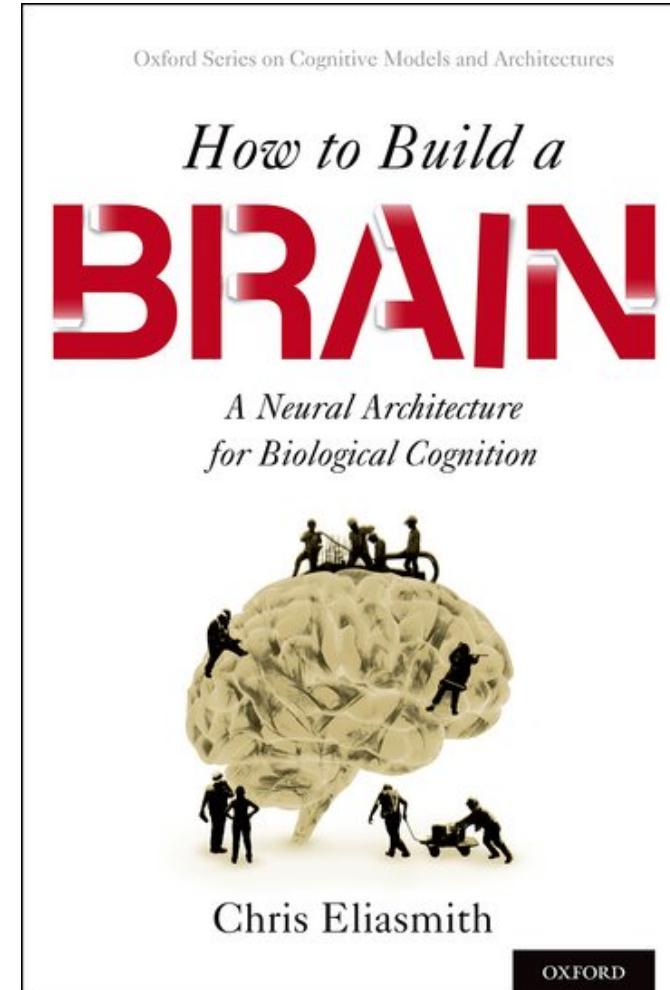
Nengo

- A software package for simulating large-scale neural systems.
- To use Nengo, you define groups of neurons in terms of what they represent, and then form connections between neural groups in terms of what computation should be performed on those representations.
- It uses the Neural Engineering Framework (NEF) to solve for the appropriate synaptic connection weights to achieve this desired computation.
- Supports various kinds of learning and make detailed spiking neuron models that implement complex high-level cognitive algorithms.
- Has been used to implement motor control, visual attention, serial recall, action selection, working memory, attractor networks, inductive reasoning, path integration, and planning.



Nengo

- Nengo 2.0 was recently implemented in Python. You should install Python from: <https://www.python.org/downloads/>.
Code was tested with Python's interpreter version 3.5.2
- Great choice for a python IDE is PyCharm Community: <https://www.jetbrains.com/pycharm/download/>
Windows version is preferred (stability)
- Installation instruction for Nengo: https://www.pythonhosted.org/nengo/getting_started.html



The Neural Engineering Framework (NEF)

Representation

Representation Formalism

- Value being represented: x
- Neural activity: a
- Neuron index: i

Encoding and decoding

- Have to define both to define a code
- Lossless code (e.g. Morse Code):
 - Encoding: $a = f(x)$
 - Decoding: $x = f^{-1}(a)$
- Lossy code:
 - Encoding: $a = f(x)$
 - Decoding: $x \sim g(a)$



The Neural Engineering Framework (NEF)

Representation

Distributed representation

- Not just one neuron per x value but rather many different a values for a single x
 - Encoding: $a_i = f_i(x)$
 - Decoding: $x \sim g(a_0, a_1, a_2, a_3, \dots)$
- *Linear decoding:*

$$\hat{x} = \sum_i a_i d_i$$

- *Elegant and simple*
- *Works fine with non-linear encoding (!)*



The Neural Engineering Framework (NEF)

Representation

Example: binary representation

- Encoding:

$$a_i = \begin{cases} 1 & \text{if } x \bmod 2^i > 2^{i-1} \\ 0 & \text{otherwise} \end{cases}$$

- Decoding:

$$\hat{x} = \sum_i a_i 2^{i-1}$$

- Suppose: $x = 13$
- Encoding: $a_1 = 1, a_2 = 0, a_3 = 1, a_4 = 1$
- Decoding: $x \sim 1 * 1 + 0 * 2 + 1 * 4 + 1 * 8$



The Neural Engineering Framework (NEF)

Representation

Neuron encoding

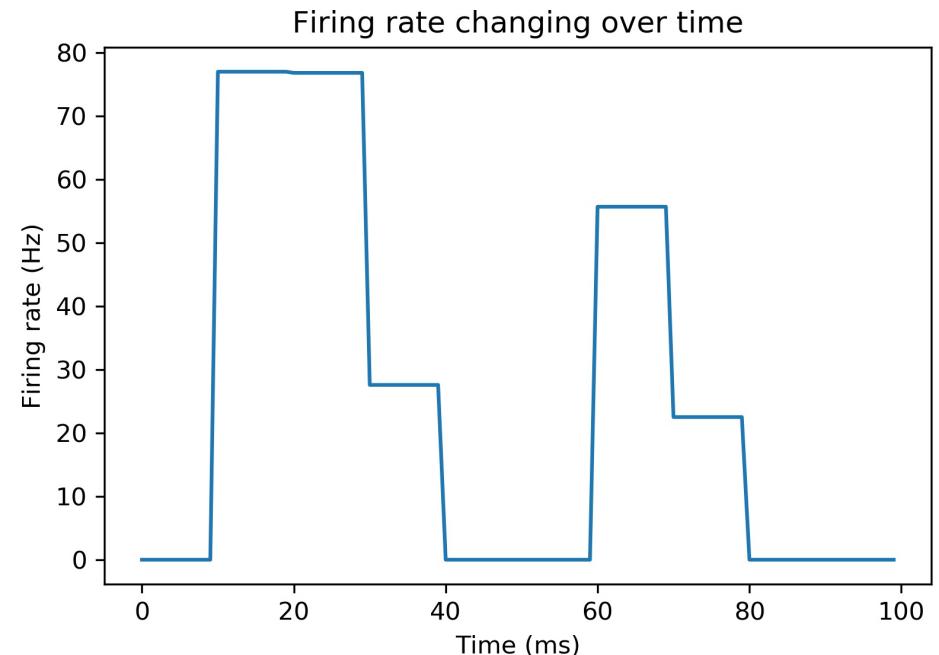
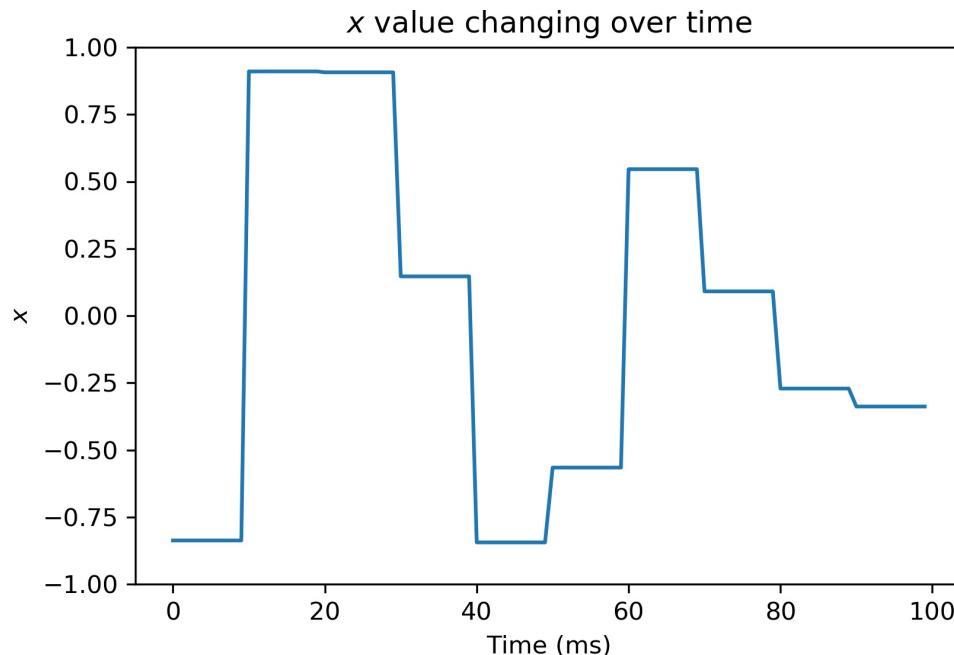
- Encoding: $a_i = f_i(x)$
- Firing rate goes up as total input current J goes up: $a_i = G_i(J(x))$
- What is G_i ?
 - Depends on how detailed a neuron model we want



The Neural Engineering Framework (NEF)

Temporal Representation

- Unfortunately, with a naive rate code, the formula for $G[J(t)]$ is an *average* firing rate over a long period of time (what does it mean to be firing at 10Hz for 10ms?)
- We need to model what's actually happening to that neuron over time

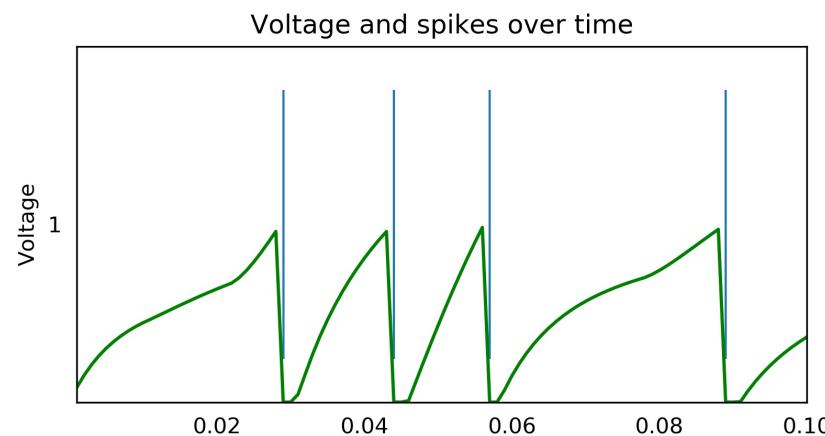
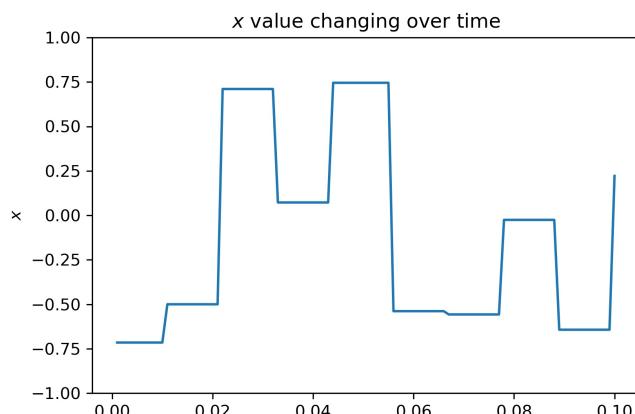


The Neural Engineering Framework (NEF)

Representation

From the Integrate and Fire Model...

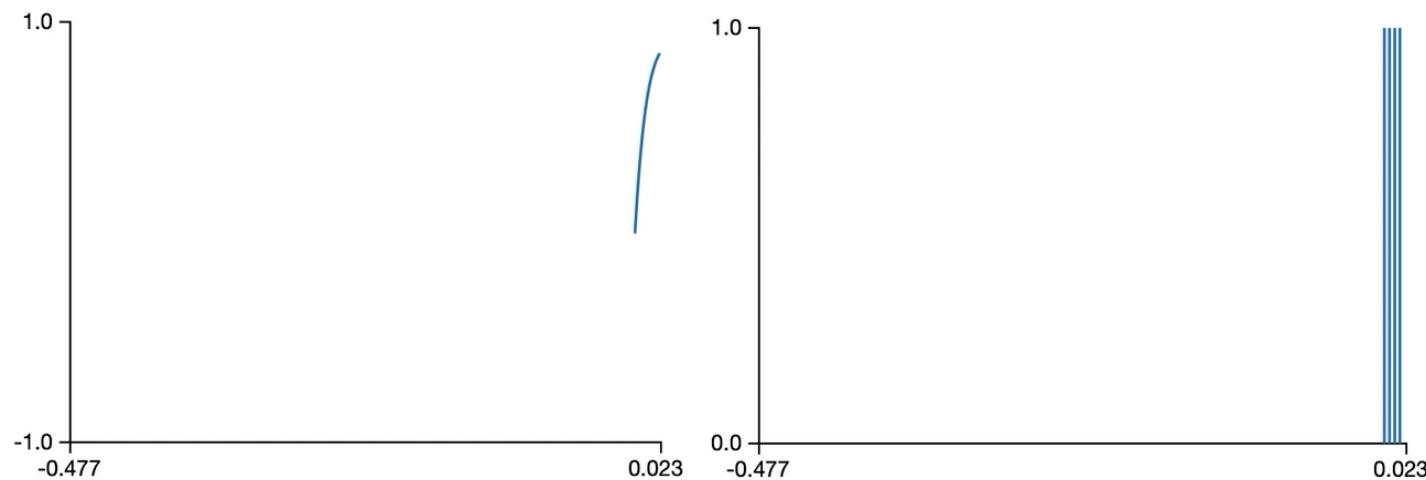
- Time per spike equals time to threshold + refractory period
- From $f = 1/t$ (frequency - time relationship): $a = \frac{1}{t_{th} + t^{ref}}$
- From the LIF model: $u(t) = u_{rest} + R \cdot I_0 \left[1 - e^{-\frac{t-t_0}{\tau}} \right]$
- By solving for t_{th} , we get: $a = \frac{1}{t^{ref} - \tau \ln(1 - \frac{V_{th}}{R \cdot I_0})}$



The Neural Engineering Framework (NEF)

Representation

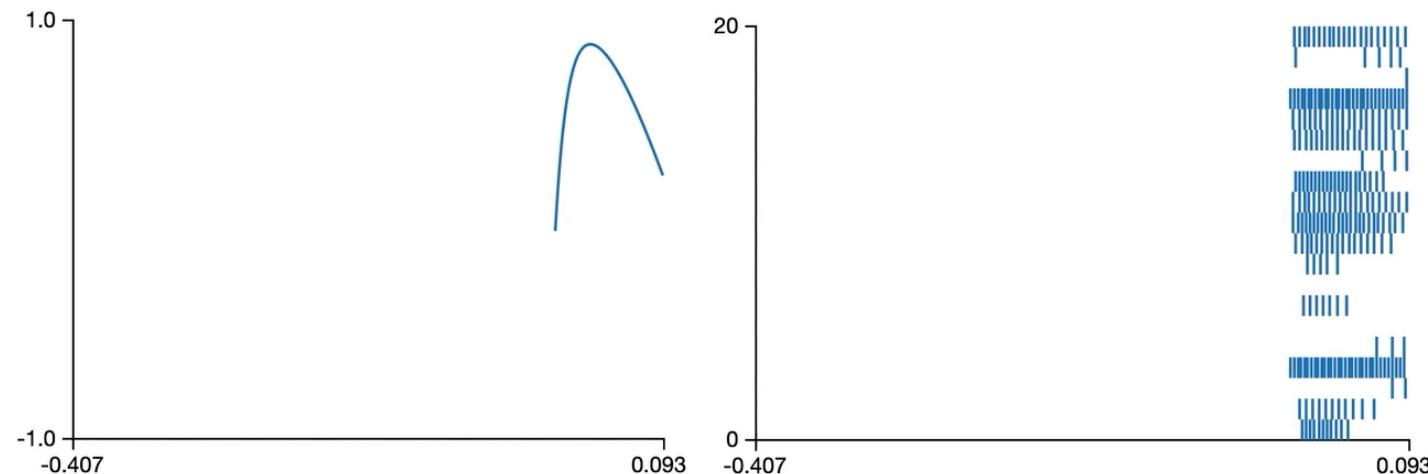
From the Integrate and Fire Model...



The Neural Engineering Framework (NEF)

Representation

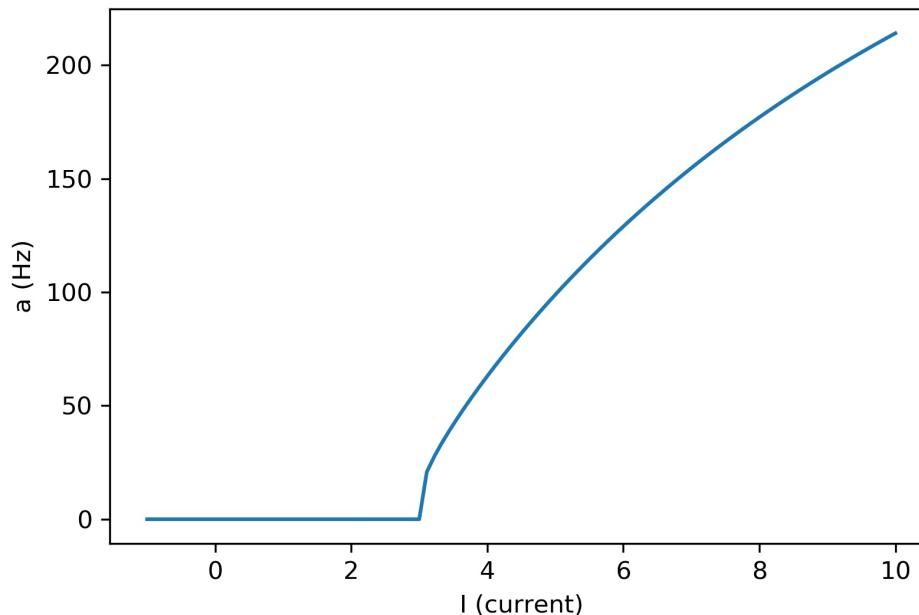
From the Integrate and Fire Model...



The Neural Engineering Framework (NEF)

Representation

- When $RC = 0.02$ and refractory is 0.002 the a - I curve will look like:



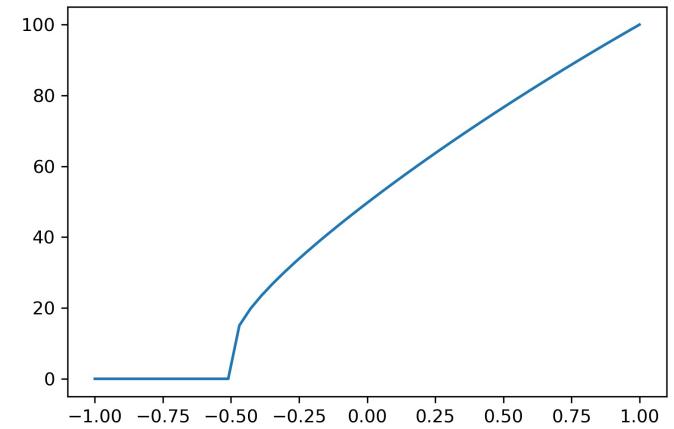
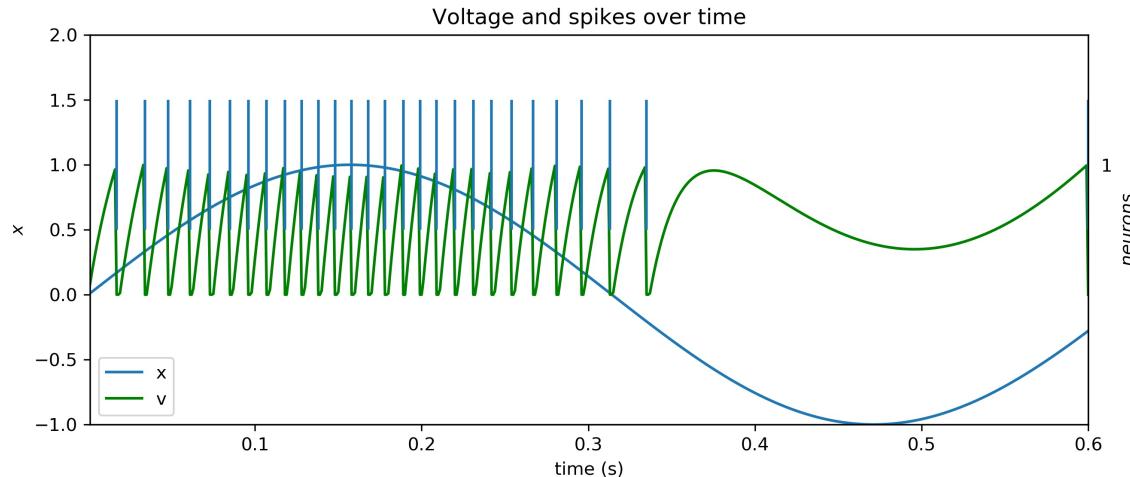
- These are called "response functions"
- How much neural firing changes with change in current
- Similar for many classes of cells (e.g. pyramidal cells - most of cortex)
- This is the G_i function in the NEF: it can be pretty much anything



The Neural Engineering Framework (NEF)

Representation

- For example (a sinus wave):



The Neural Engineering Framework (NEF)

Representation

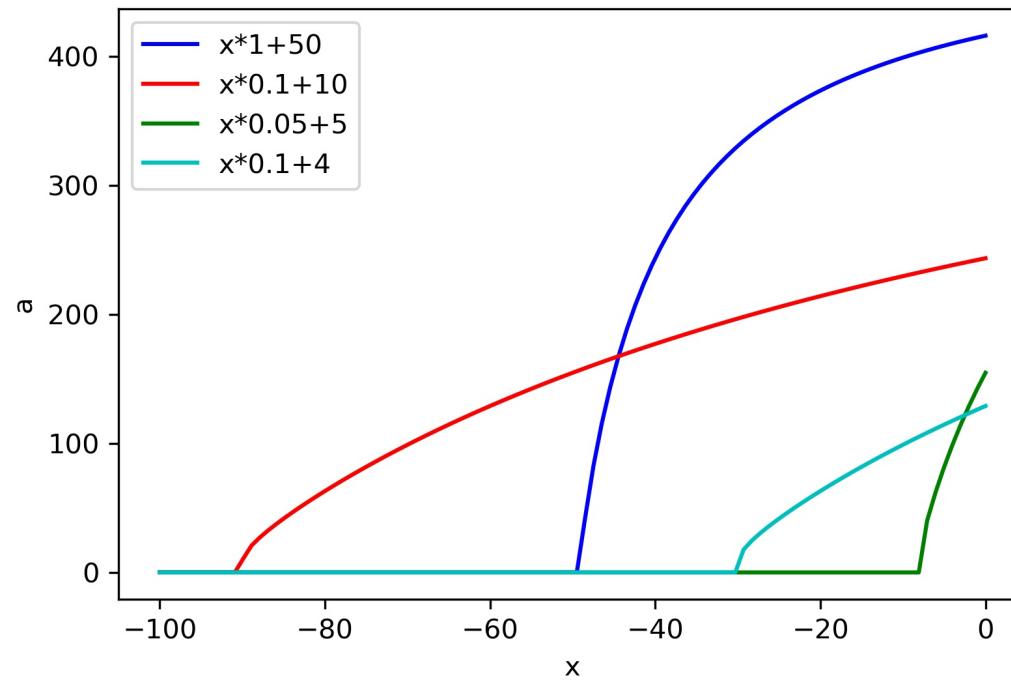
- Weaknesses:
 - These are "point" neurons (i.e. we're assuming the body of the neuron is just one infinitesimal point, rather than having lots and lots of compartments, each with its own RC circuit)
 - Assumes R is a constant
 - Assumes v_{th} is a constant
 - Assumes no adaptation (it's harder for a neuron to fire after it has recently fired)
 - Doesn't consider nonlinearities at the input



The Neural Engineering Framework (NEF)

Representation

- Neurons seem to be sensitive to particular values of x (receptive fields, visual stimulation...)
- How are neurons 'tuned' to a representation? or...
- What's the mapping between x and a ?
- Encoding: $a_i = G_i(J(x))$
- One simple mapping might be: $J = \alpha x + J^{bias}$



The Neural Engineering Framework (NEF)

Representation

- There's usually some x which gives a maximum firing rate ...and thus a maximum J ...
- Firing rate (and J) decrease as you get farther from the preferred x value
- So something like: $J = \alpha[\text{sim}(x, x_{\text{pref}})] + J^{\text{bias}}$
- What sort of similarity measure?



The Neural Engineering Framework (NEF)

Representation

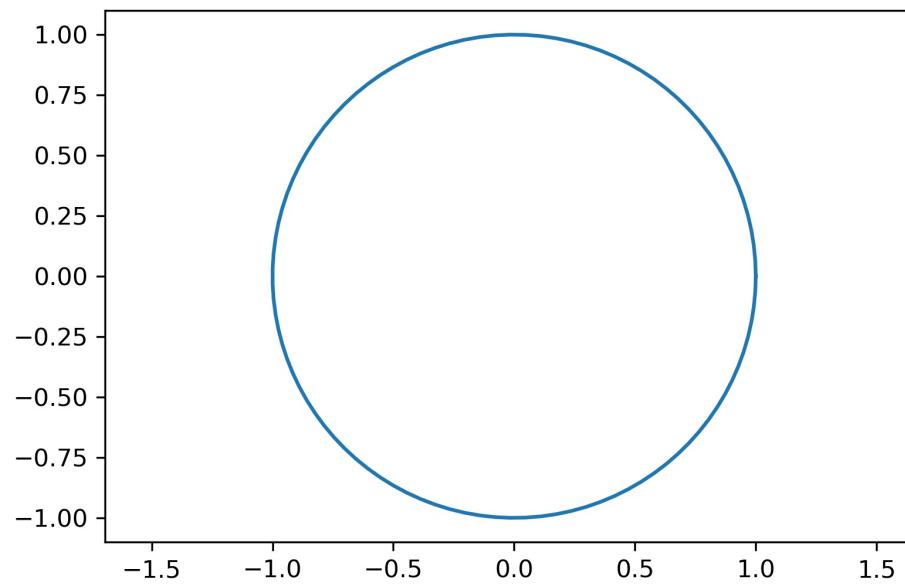
- A general form:

$$a_i = G_i [\alpha_i x \cdot e_i + J_i^{bias}]$$

A positive gain term

the encoder, or the
preferred direction vector
with a unit length

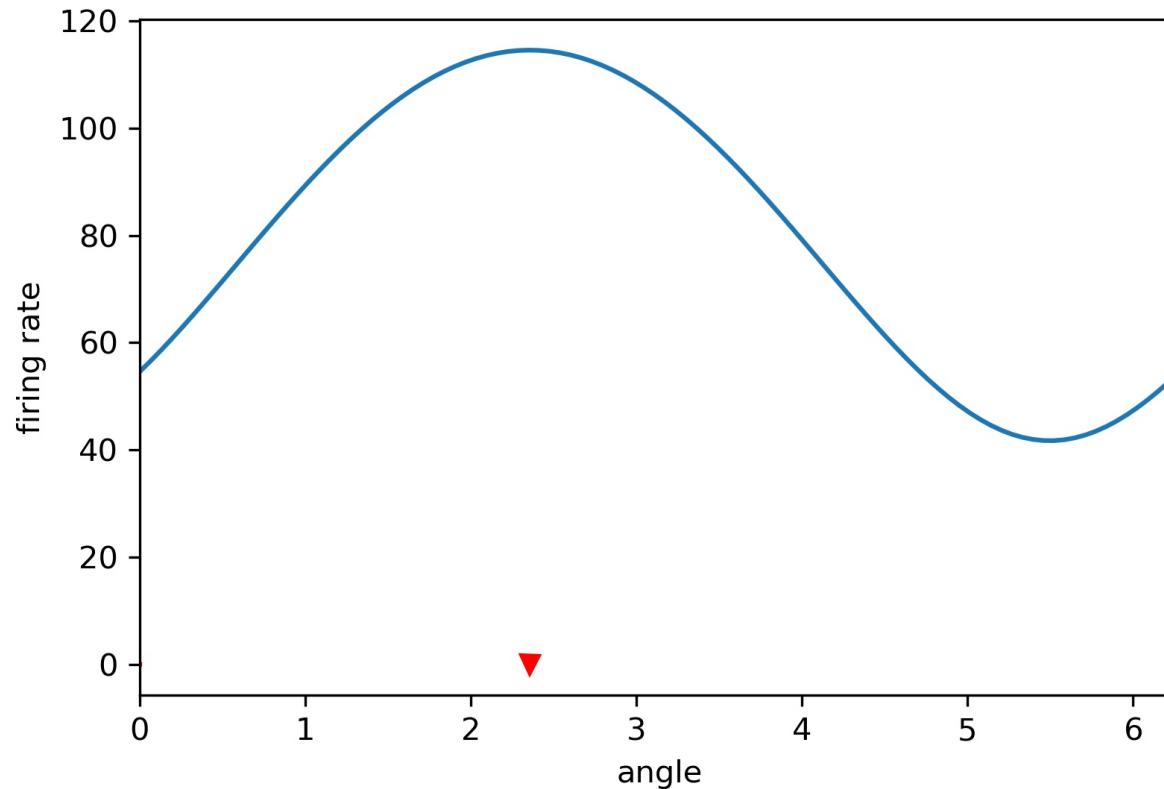
- In 1D e is either +1 or -1, in 2D:



The Neural Engineering Framework (NEF)

Representation

- If the preferred direction is: $[-1, 1]$ (indicated in red), the response curve would be:



The Neural Engineering Framework (NEF)

Representation

- Encoding: $a_i = G_i(J(x))$
- For decoding we want: $\hat{x} = \sum_i a_i d_i$
- But where do we get d_i from?
- By minimising the average error over all x : $E = \frac{1}{2} \int_{-1}^1 (x - \hat{x})^2 dx$
- Where: $E = \frac{1}{2} \int_{-1}^1 \left(x - \sum_i a_i d_i \right)^2 dx$
- Minimising the derivative with respect to d_i :

$$\frac{\partial E}{\partial d_i} = \frac{1}{2} \int_{-1}^1 2 \left[x - \sum_j a_j d_j \right] (-a_i) dx$$

$$\frac{\partial E}{\partial d_i} = - \int_{-1}^1 a_i x dx + \int_{-1}^1 \sum_j a_j d_j a_i dx$$



The Neural Engineering Framework (NEF)

Representation

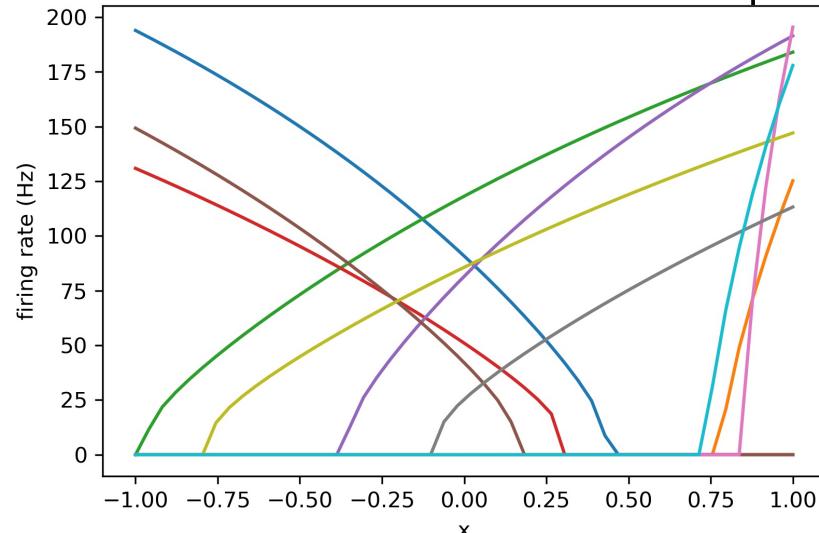
- Setting: $\frac{\partial E}{\partial d_i} = 0$
- Getting us to: $\int_{-1}^1 a_i x \, dx = \int_{-1}^1 \sum_j (a_j d_j a_i) \, dx$
$$\int_{-1}^1 a_i x \, dx = \sum_j \left(\int_{-1}^1 a_i a_j \, dx \right) d_j$$
- That's a system of N equations and N unknowns
- Approximate the integral by sampling over x :
$$\sum_x a_i x / S = \sum_j \left(\sum_x a_i a_j / S \right) d_j$$

 S is the number of x values to use (S for samples) $\Upsilon = \Gamma d$
- Solving the linear system:
$$d = \Gamma^{-1} \Upsilon$$
$$d_i = \sum_j \Gamma_{ij}^{-1} \Upsilon_j$$

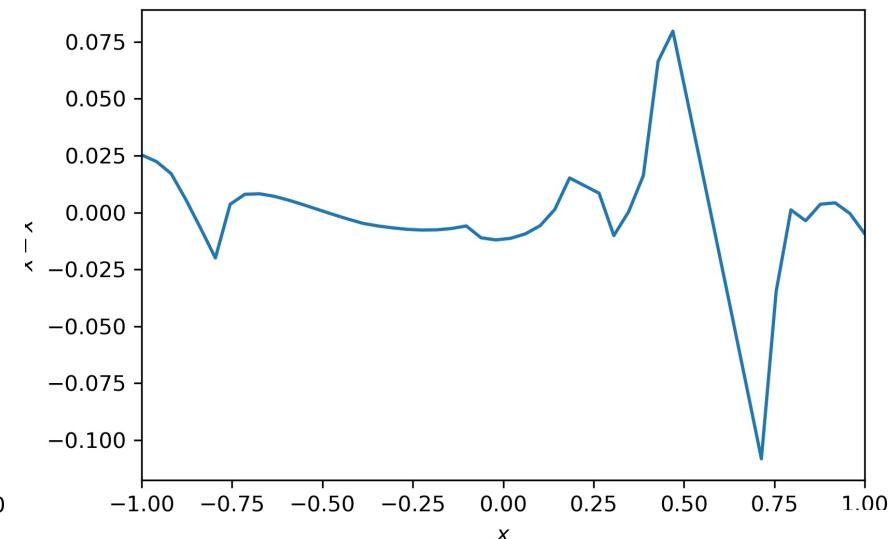
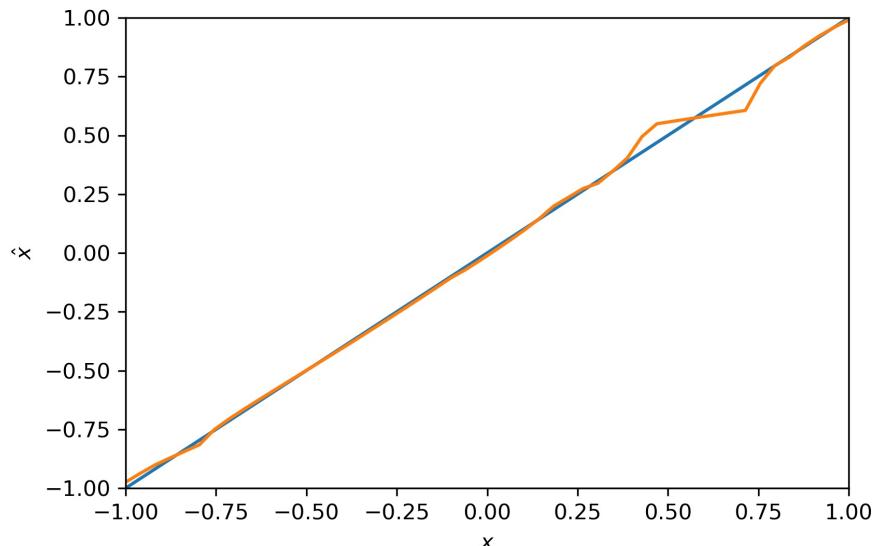


The Neural Engineering Framework (NEF)

Representation



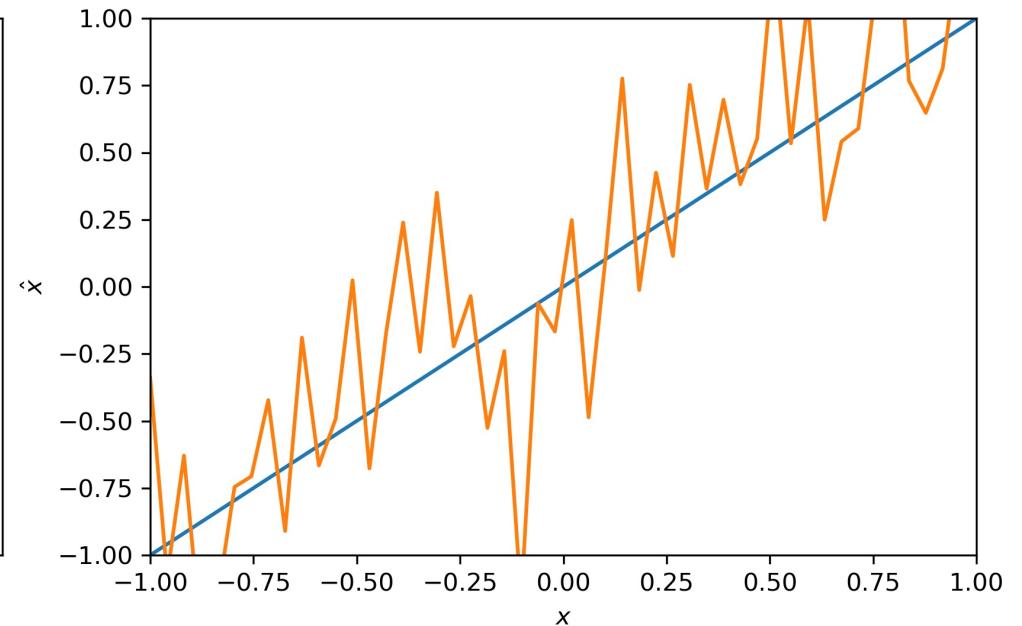
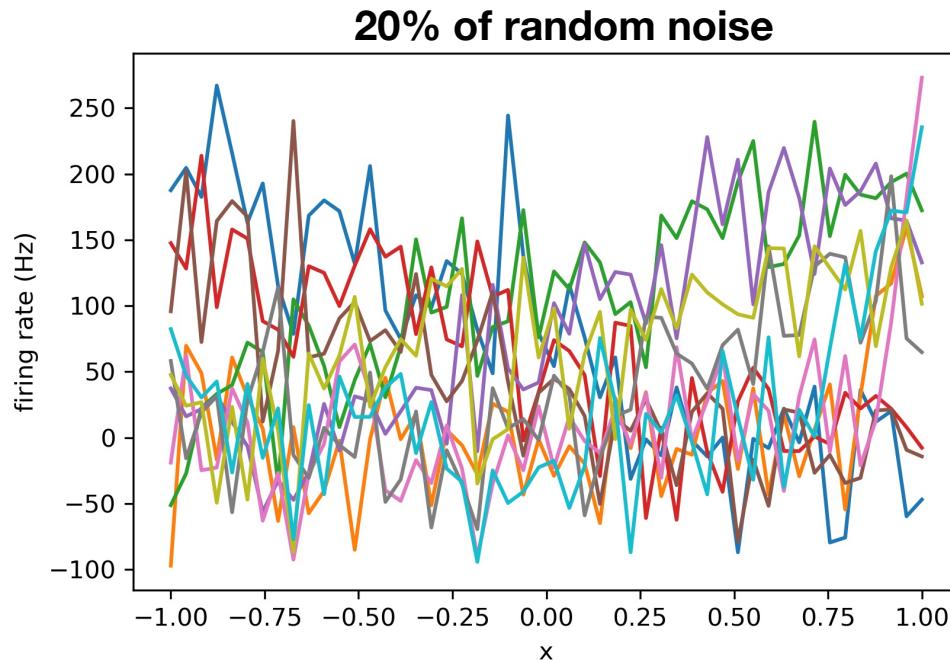
Defaults to LIF neurons, with random gains and biases for neurons between 100-200hz over -1,1



The Neural Engineering Framework (NEF)

Representation

- Neurons are not perfect: they are noisy entities
see: <https://icwww.epfl.ch/~gerstner/SPNM/node33.html>



The Neural Engineering Framework (NEF)

Representation

- Handling noise is one of the more efficient aspects in neuromorphic systems. Unlike engineered representation, biological representation explicitly address the effect of noise
- Taking noise into account (Introduce noise term η)

$$\hat{x} = \sum_i (a_i + \eta) d_i$$

$$E = \frac{1}{2} \int_{-1}^1 (x - \hat{x})^2 dx d\eta$$

$$= \frac{1}{2} \int_{-1}^1 \left(x - \sum_i (a_i + \eta) d_i \right)^2 dx d\eta$$

$$= \frac{1}{2} \int_{-1}^1 \left(x - \sum_i a_i d_i - \sum_i \eta d_i \right)^2 dx d\eta$$

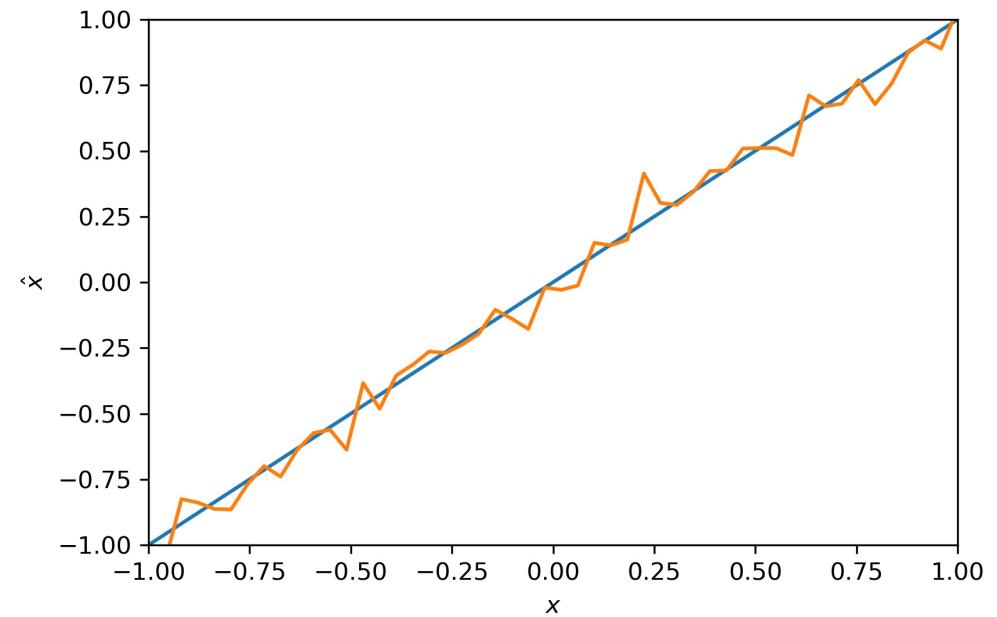
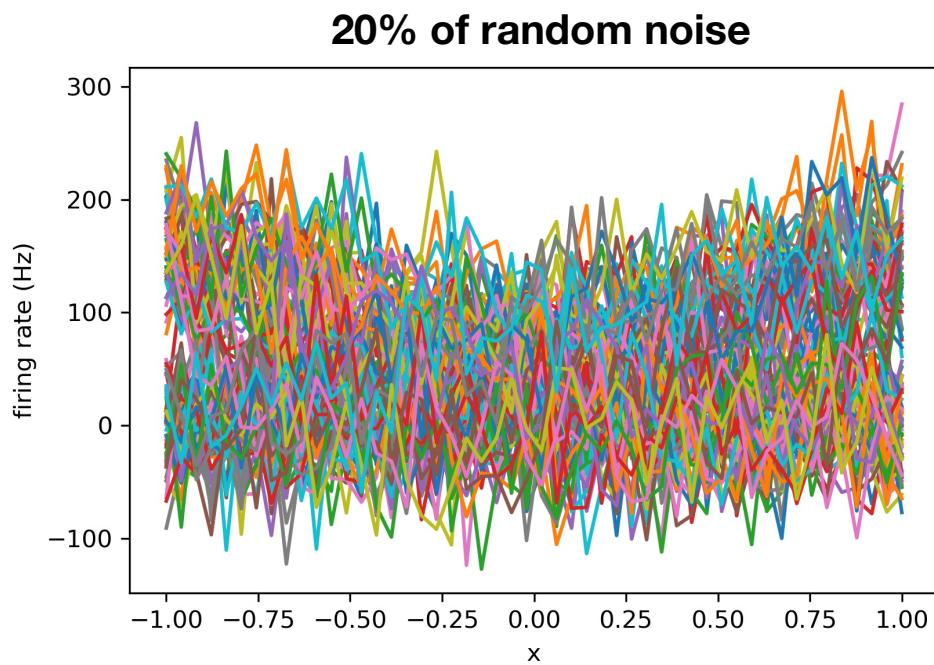
- Assume noise is gaussian, independent, mean zero, and has the same variance for each neuron

$$E = \frac{1}{2} \int_{-1}^1 \left(x - \sum_i a_i d_i \right)^2 dx + \sigma^2 \sum_i d_i^2$$



The Neural Engineering Framework (NEF)

Representation



The Neural Engineering Framework (NEF)

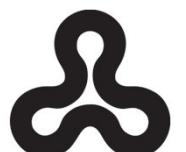
Representation

$$E = \frac{1}{2} \int_{-1}^1 \left(x - \sum_i a_i d_i \right)^2 dx + \sigma^2 \sum_i d_i^2$$

Error due to static distortion
(i.e. the error introduced by
the decoders themselves)

Error due to noise

- Noise error is proportional to $1/N$
- Distortion error is proportional to $1/N^2$
- When the number of neurons is greater than 100, the error is dominated by the noise term



The Neural Engineering Framework (NEF)

Temporal Representation

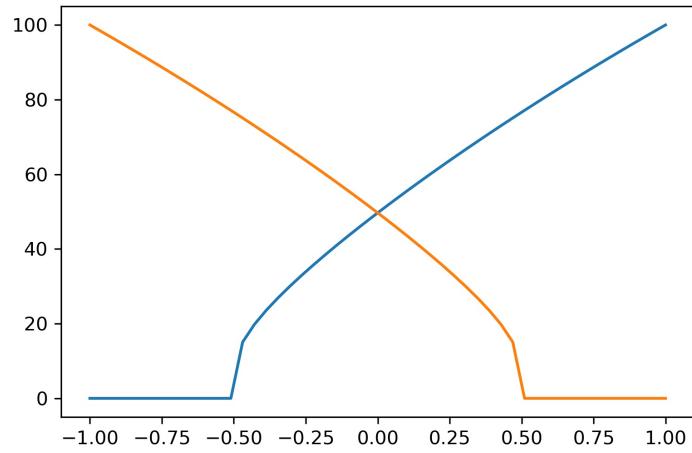
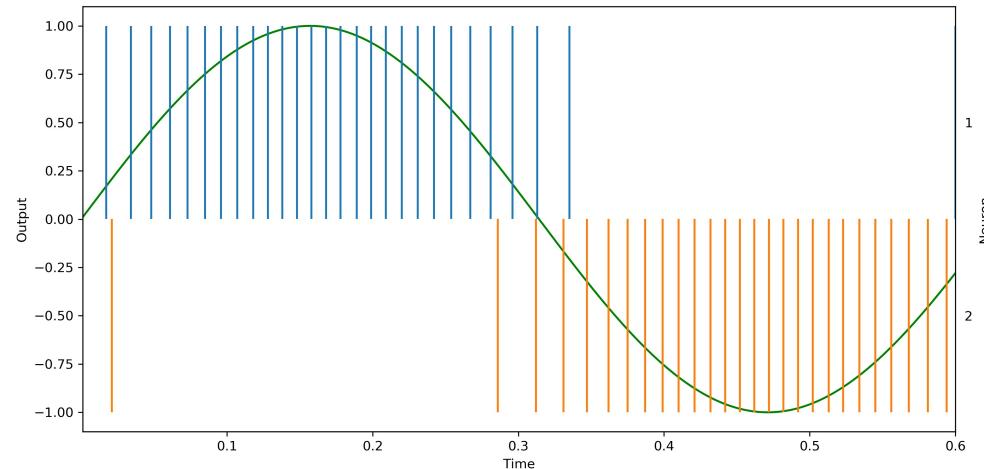
- We covered representation of a vector space by a population of neurons.
- We converted a vector x into neural activity values a_i , and then converted those back to the original value x
- What happens if x changes over time?
 - Maybe: $a = G[\alpha e \cdot x + J^{bias}]$, so...
$$a(t) = G[\alpha e \cdot x(t) + J^{bias}]$$
$$\text{where } G[J(t)] = \frac{1}{\tau_{ref} - \tau_{RC} \ln(1 - \frac{1}{J(t)})}$$
 - How do decode this? Can we stick with linear decoding?



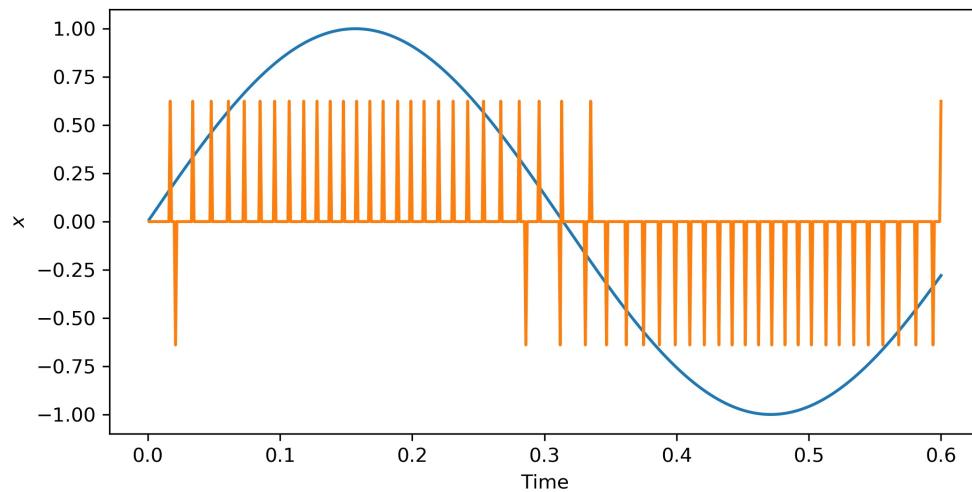
The Neural Engineering Framework (NEF)

Temporal Representation

- Consider:



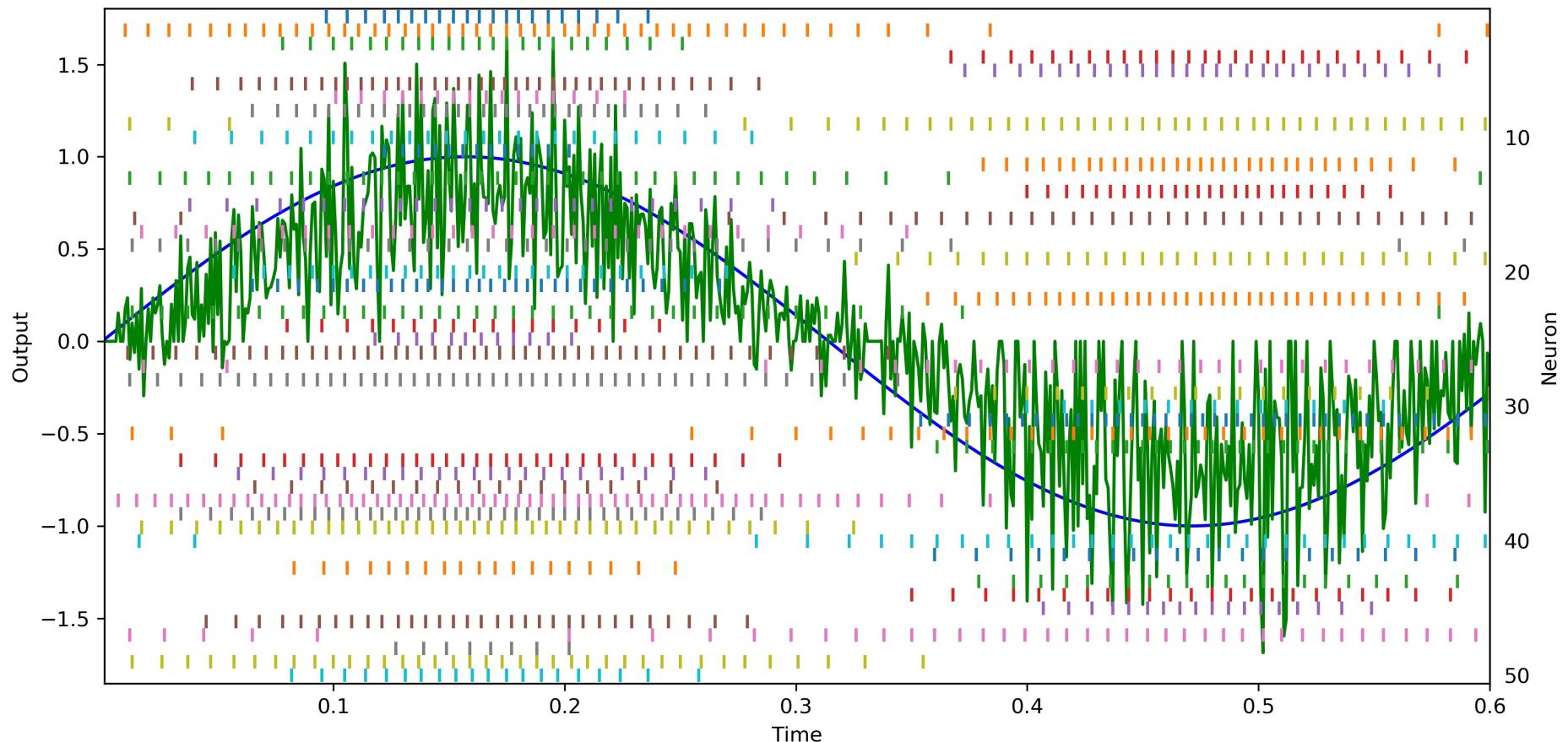
- Decoding x using a sum of activities with $\hat{x}(t) = \sum_i a_i(t)d_i$:



The Neural Engineering Framework (NEF)

Temporal Representation

- If we add more neurons:



The Neural Engineering Framework (NEF)

Temporal Representation

- What's happening here?
 - At a given instant in time t , only a very small number of neurons are firing.
 - If no neurons are firing at that instant, $x = 0$
 - There are usually only one or two neurons firing at any given time
 - As dt gets smaller, the problem gets even worse
 - What can we do?
 - We want a spike to contribute to other points in time
 - So we want to convert a train of spikes into some continuous function

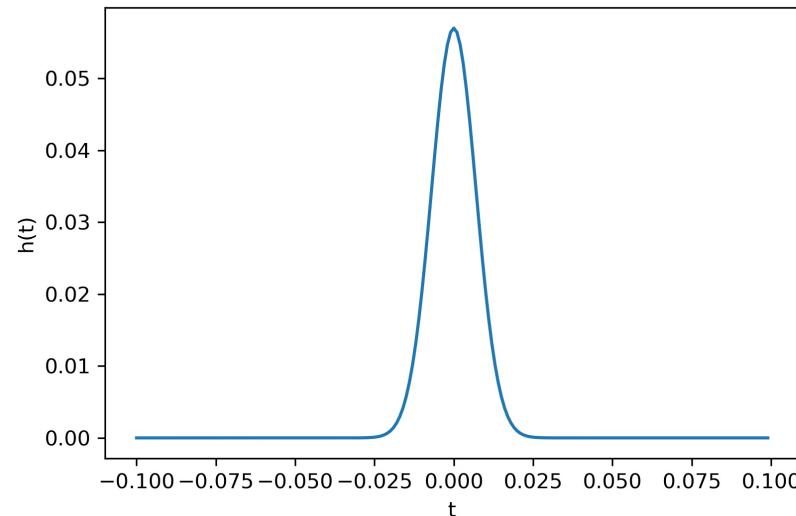


The Neural Engineering Framework (NEF)

Temporal Representation

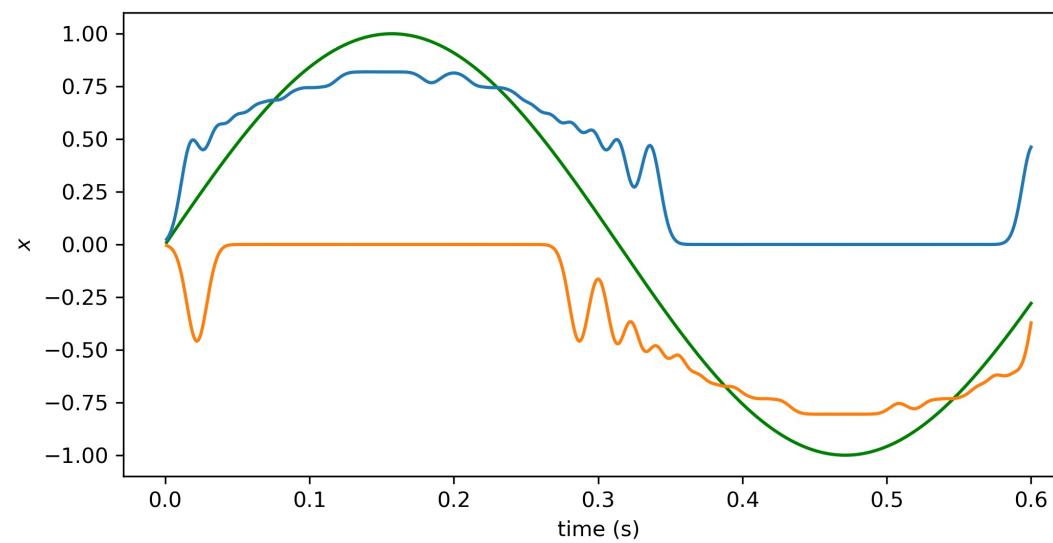
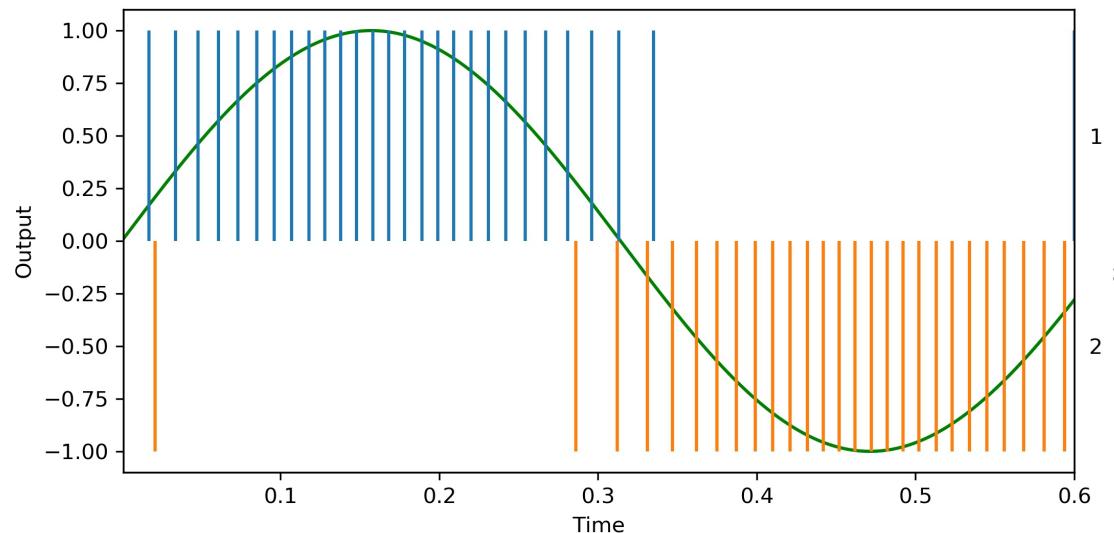
- Consider convolution: $\hat{x}(t) = \sum_i a_i(t) * h(t)d_i$

where $h(t)$ is a gaussian: $h(t) = e^{-t^2/2\sigma^2}$



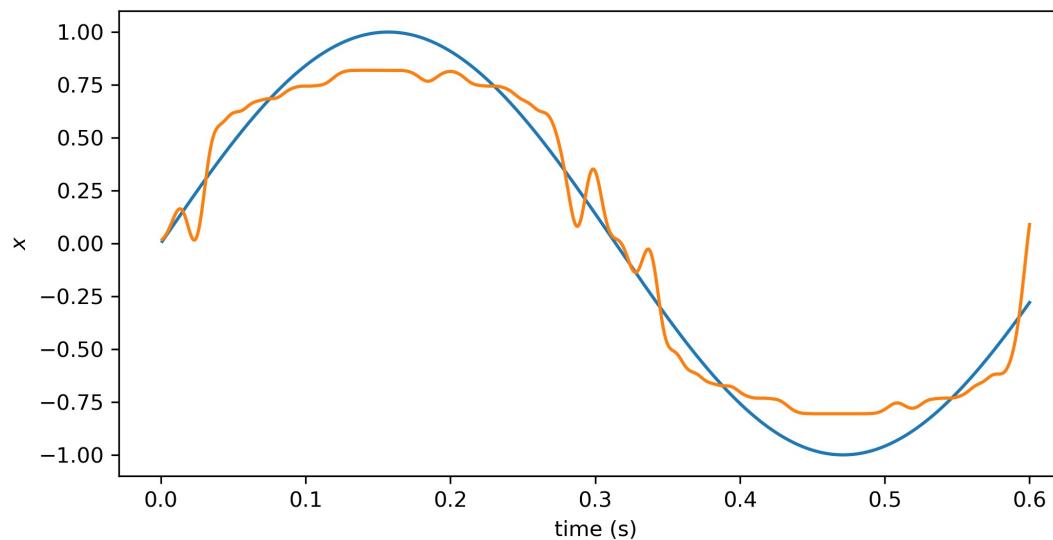
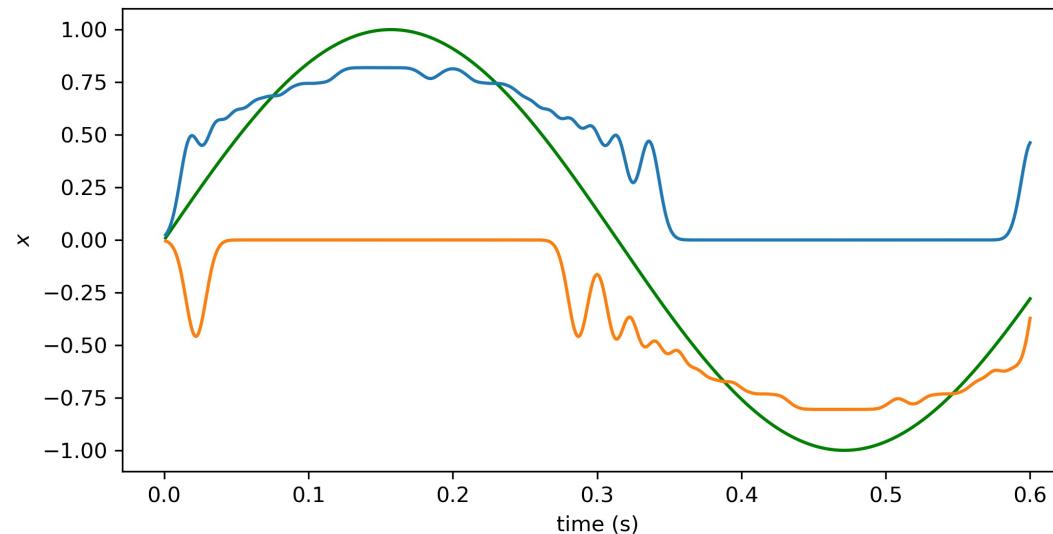
The Neural Engineering Framework (NEF)

Temporal Representation



The Neural Engineering Framework (NEF)

Temporal Representation



The Neural Engineering Framework (NEF)

Temporal Representation

- Can we do better?
- Consider the two neurons case, where $d_1 = -d_2$:

$$\hat{x}(t) = \sum_i a_i(t) * h(t)d_i$$

$$\hat{x}(t) = a_1(t) * h(t)d_1 + a_2(t) * h(t)d_2$$

$$\hat{x}(t) = (a_1(t) - a_2(t)) * h(t)d_1$$

$$\hat{x}(t) = (a_1(t) - a_2(t)) * h(t)$$

$$\hat{x}(t) = r(t) * h(t)$$

- The convolution theorem: convolution turns into multiplication when you do a Fourier transform (https://en.wikipedia.org/wiki/Convolution_theorem):

$$\hat{X}(\omega) = R(\omega)H(\omega)$$

$$E(\omega) = (X(\omega) - R(\omega)H(\omega))^2$$

- Now we can take the derivative and set it equal to zero to do the minimization:

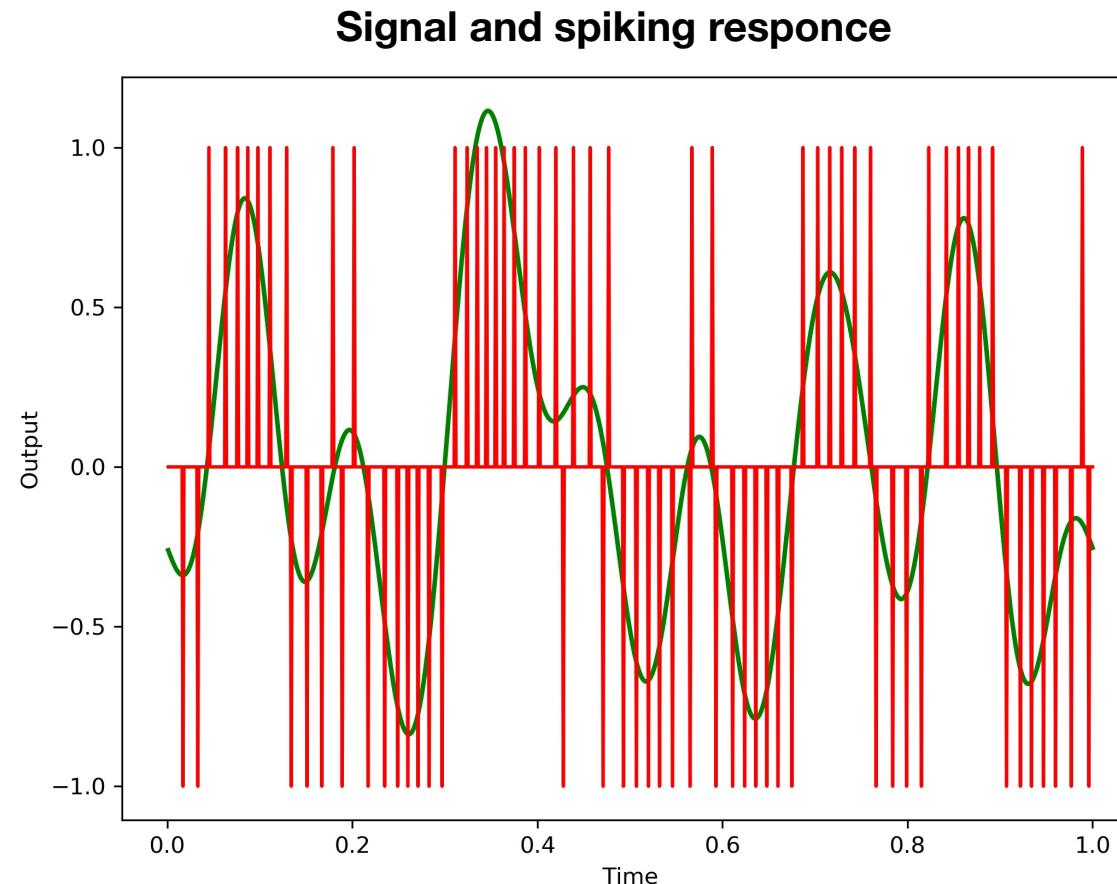
$$H(\omega) = \frac{X(\omega)R^*(\omega)}{|R(\omega)|^2}$$



The Neural Engineering Framework (NEF)

Temporal Representation

- So now we can find $H(\omega)$ given the Fourier transform of our signal $X(\omega)$ and the Fourier transform of the spiking response $R(\omega)$:

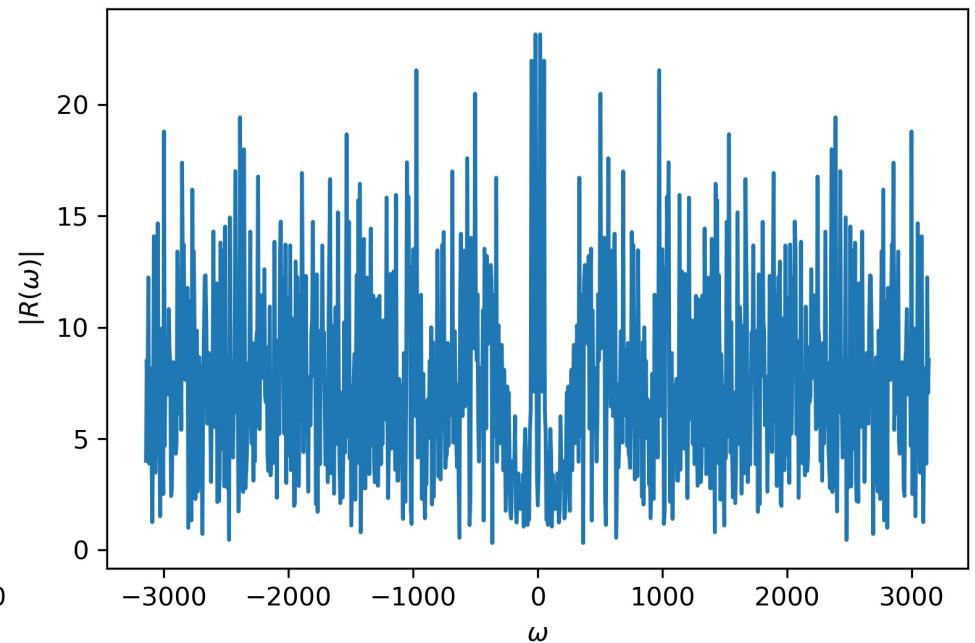
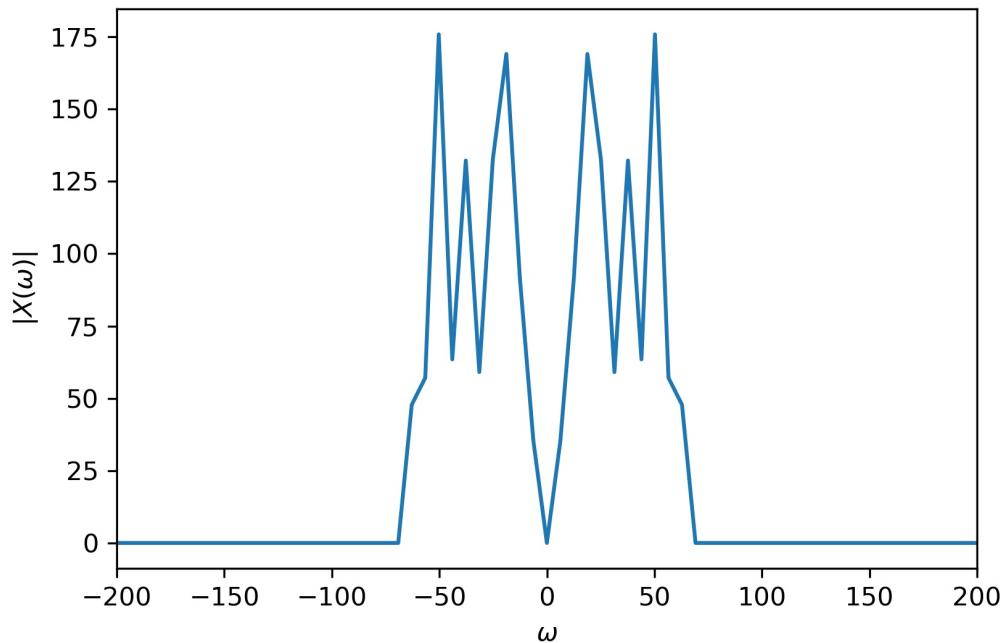


The Neural Engineering Framework (NEF)

Temporal Representation

- So now we can find $H(\omega)$ given the Fourier transform of our signal $X(\omega)$ and the Fourier transform of the spiking response $R(\omega)$:

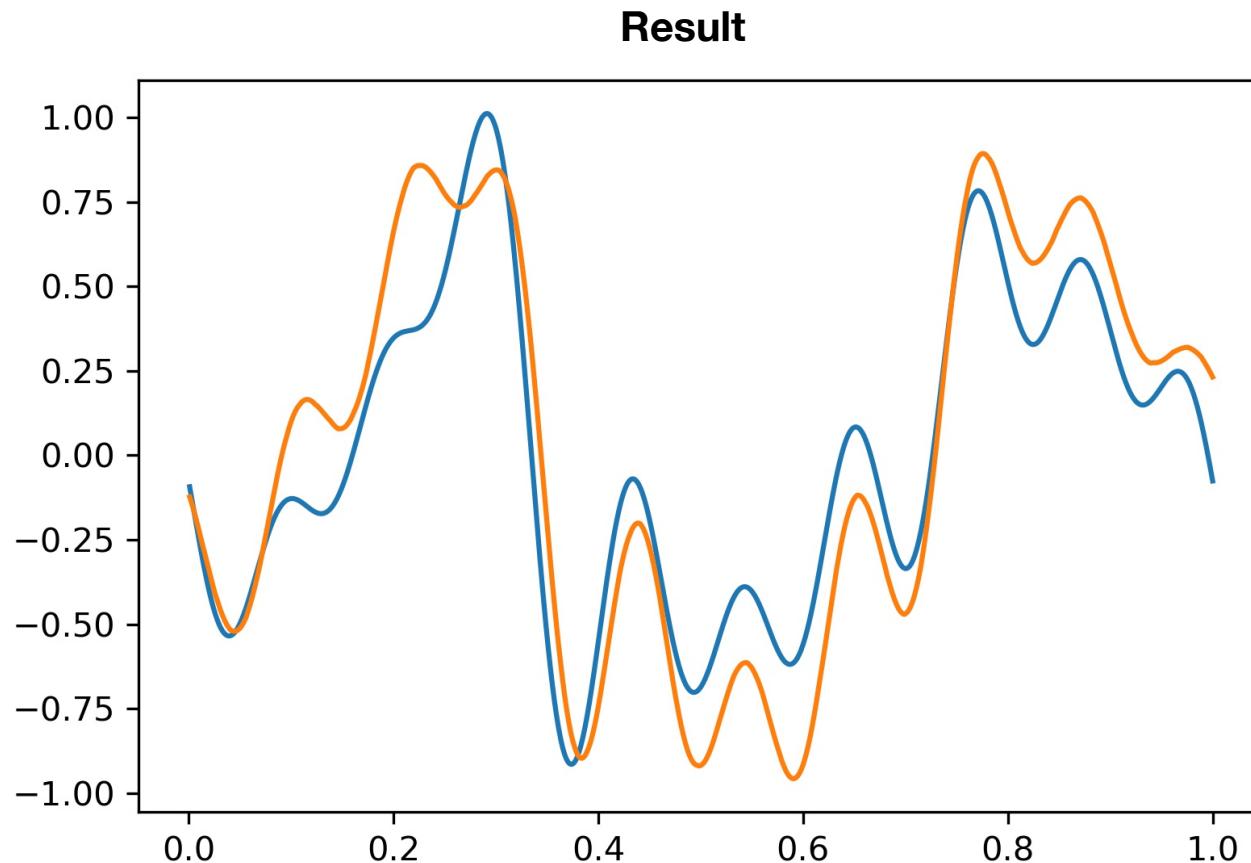
Fourier transform of signal and spiking



The Neural Engineering Framework (NEF)

Temporal Representation

- Can we generalise it?
- Same filter on a different input:



The Neural Engineering Framework (NEF)

Temporal Representation

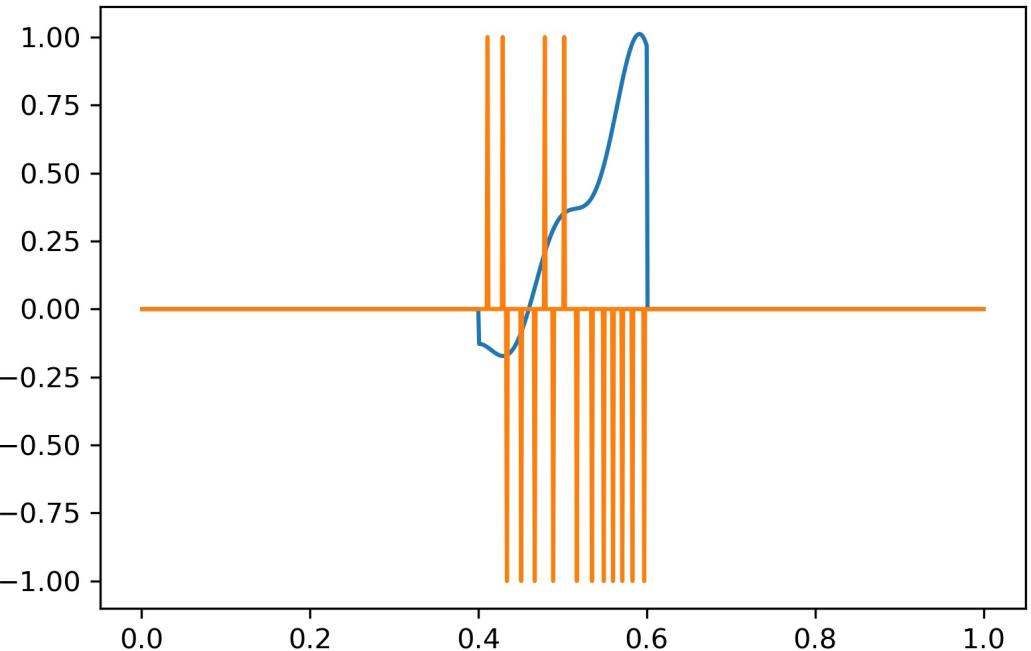
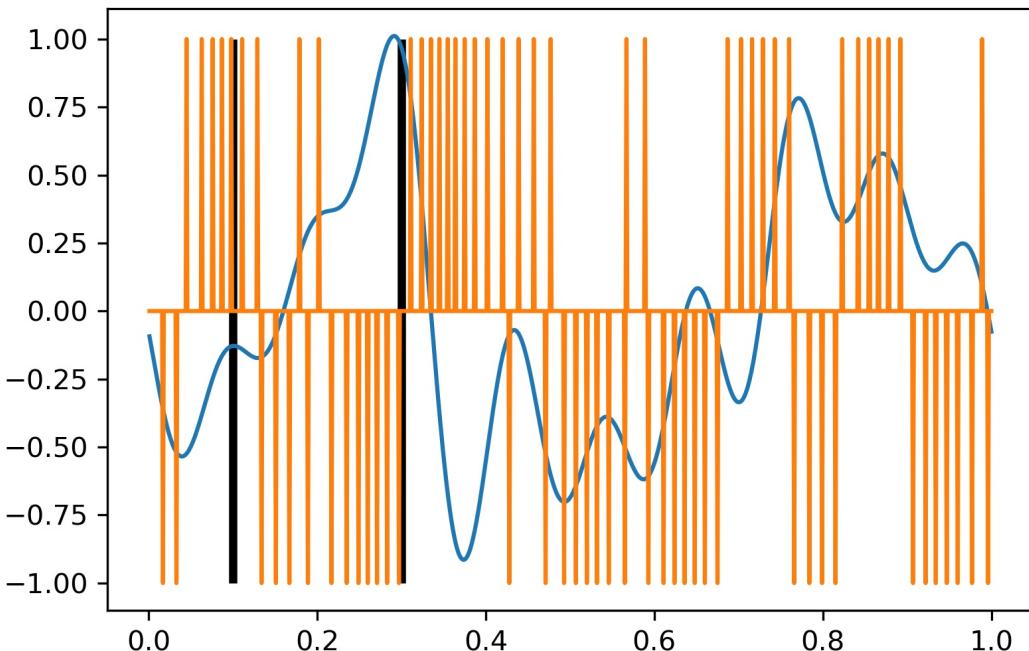
- We're only training on a pretty small set of data
 - So let's increase τ
 - This can help, but can get computationally expensive
- Another approach
 - We've already created a pretty long signal ($T=1.0$ seconds)
 - We know the filter should probably decay towards zero
 - A spike shouldn't affect values far away from when it happens
 - So let's take our one existing signal and chop it up into lots of little signals



The Neural Engineering Framework (NEF)

Temporal Representation

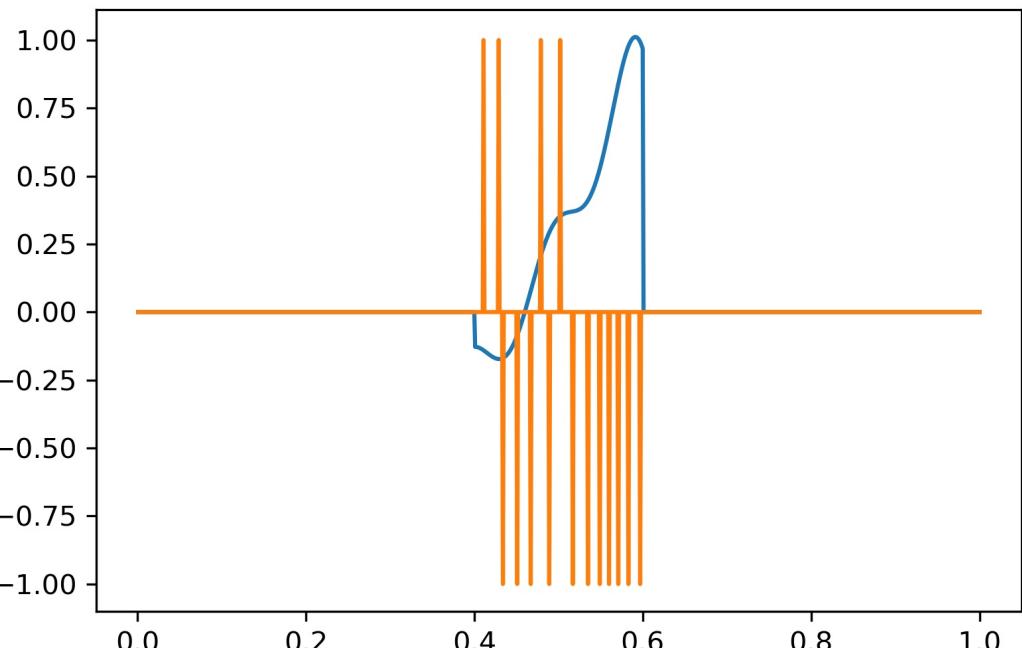
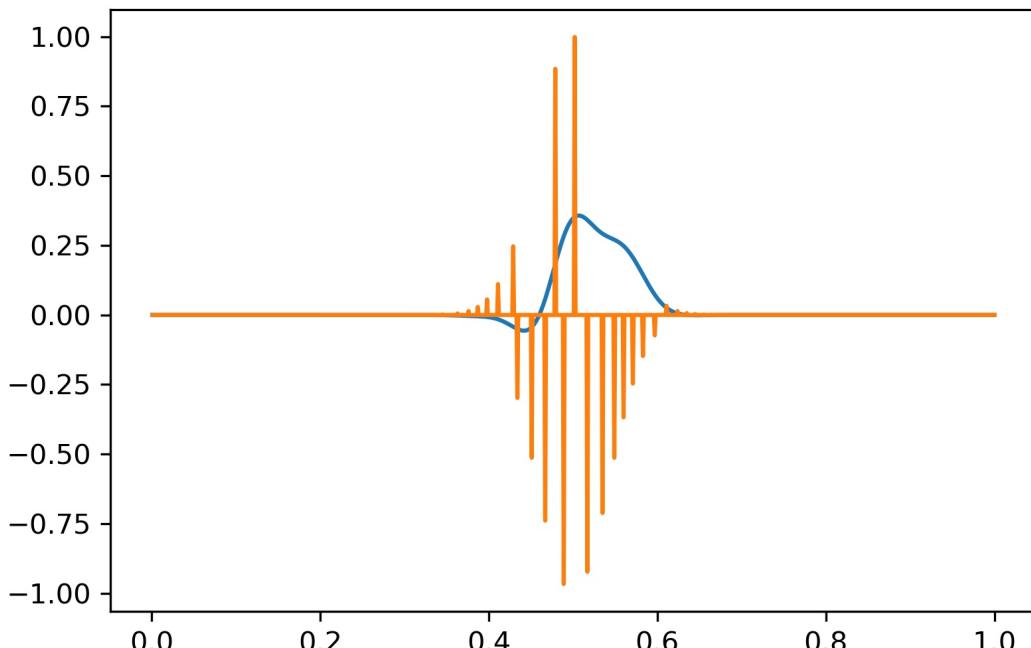
- So let's take our one existing signal and chop it up into lots of little signals



The Neural Engineering Framework (NEF)

Temporal Representation

- This introduces some edge effects at the beginning and ending of the window
- So we can do some sort of smoothing of the data (we use a Gaussian)



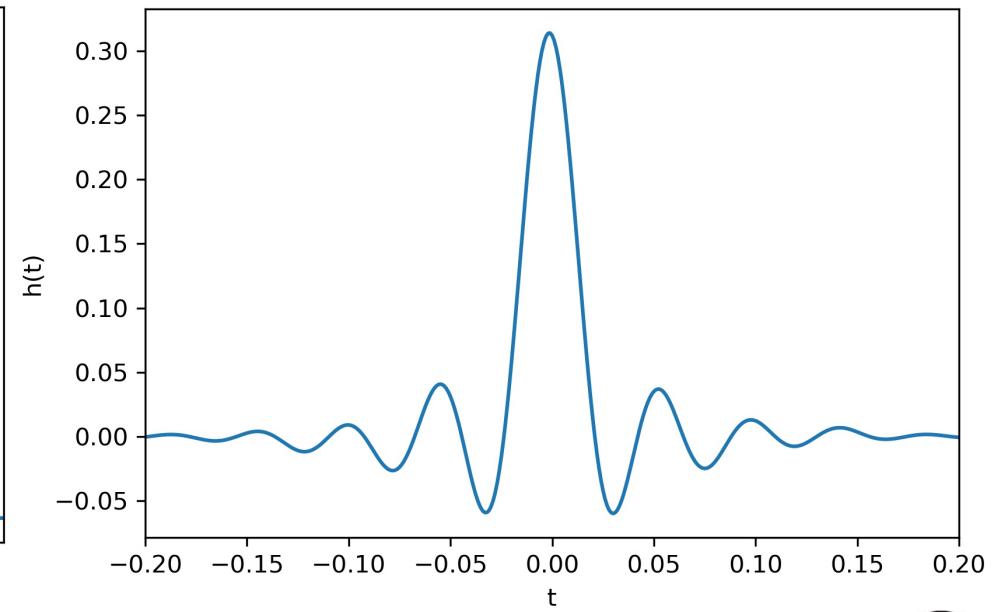
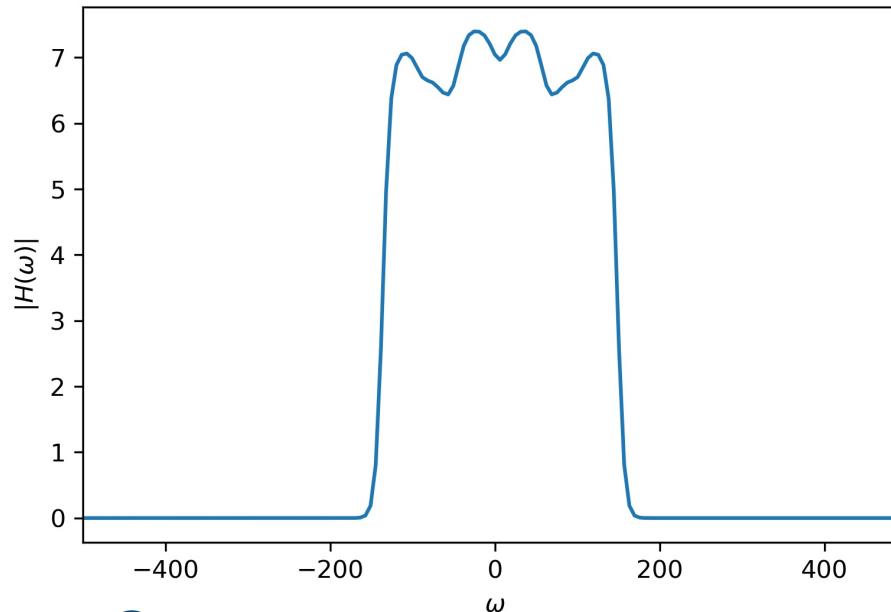
The Neural Engineering Framework (NEF)

Temporal Representation

- Last graph can be thought of as one trial, which we perform at every Δt and then average over:

$$H(\omega) = \frac{(X(\omega)R^*(\omega)) * W(\omega)}{|R(\omega)|^2 * W(\omega)}$$

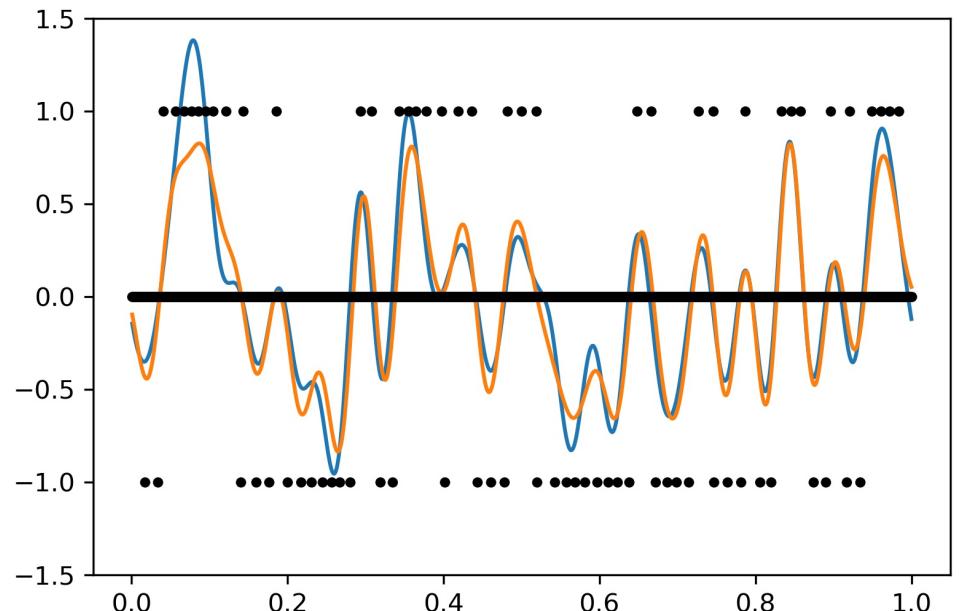
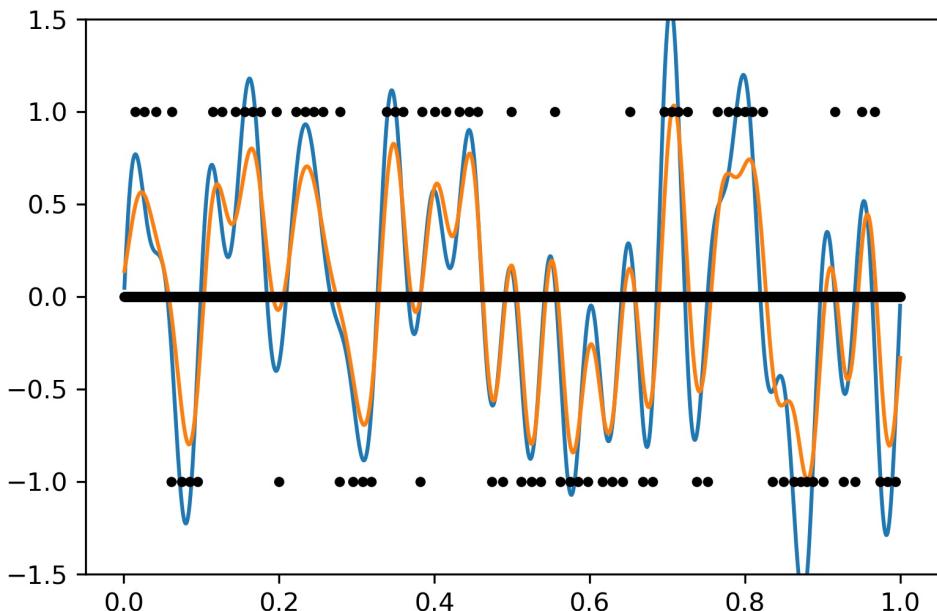
- So we're windowing and averaging in time doing convolution in frequency (it's kind of unusual to see convolution in frequency), creating = an efficient method for estimation and getting:



The Neural Engineering Framework (NEF)

Temporal Representation

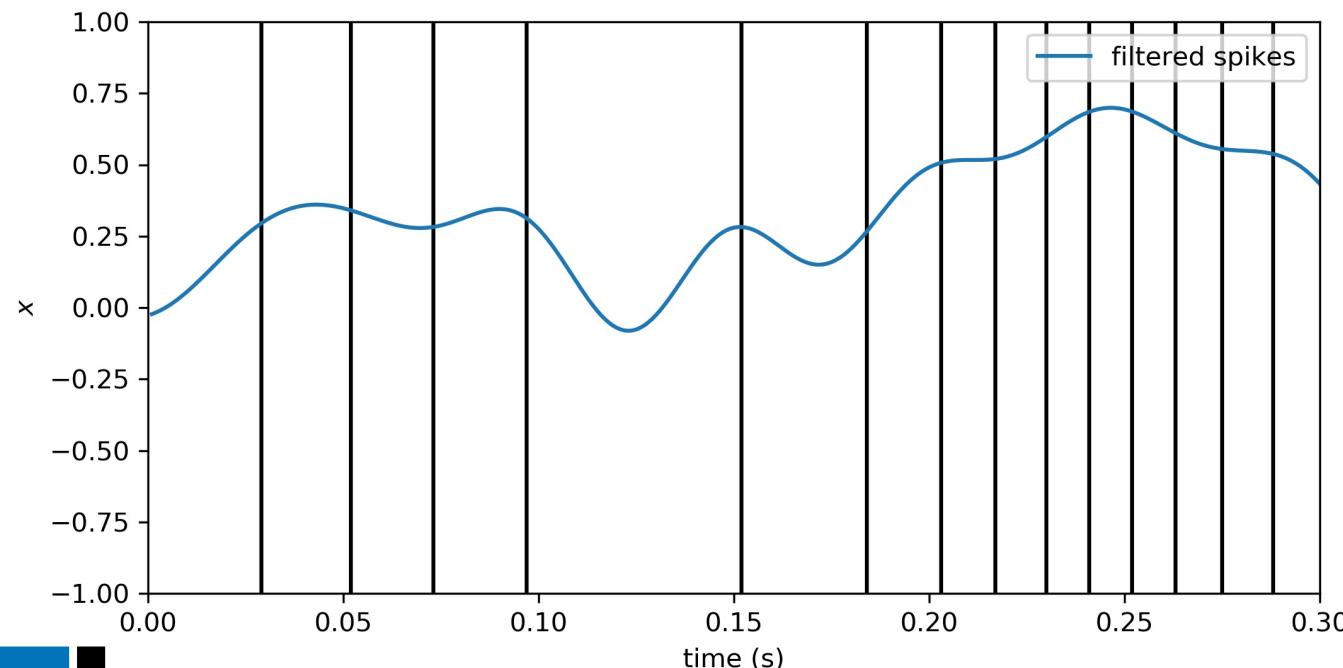
- Results:



The Neural Engineering Framework (NEF)

Temporal Representation **NOTE**

- The decoder works better if it is made for the right range of frequencies
- Here, we've only done this for the case of 2 neurons, but we could use the same filter for many more neurons
- Using our filter, every time a spike occurs, we practically replace the spike with the filter shape shape. For example:



The Neural Engineering Framework (NEF)

Temporal Representation

- If we're analyzing the data after the fact, this is fine
- But what if we're just sitting here observing the neural activity and wanting to know what's being represented right now?
- Notice that the blue line (the filtered value we're using for decoding) starts going up *before* the spike happens
- The filtered value is using information ***from the future***.
- This means that when we look at x this way, we're getting a better estimate of x than it is possible for other neurons to get (assuming neurons aren't psychic)
- What could we do instead?



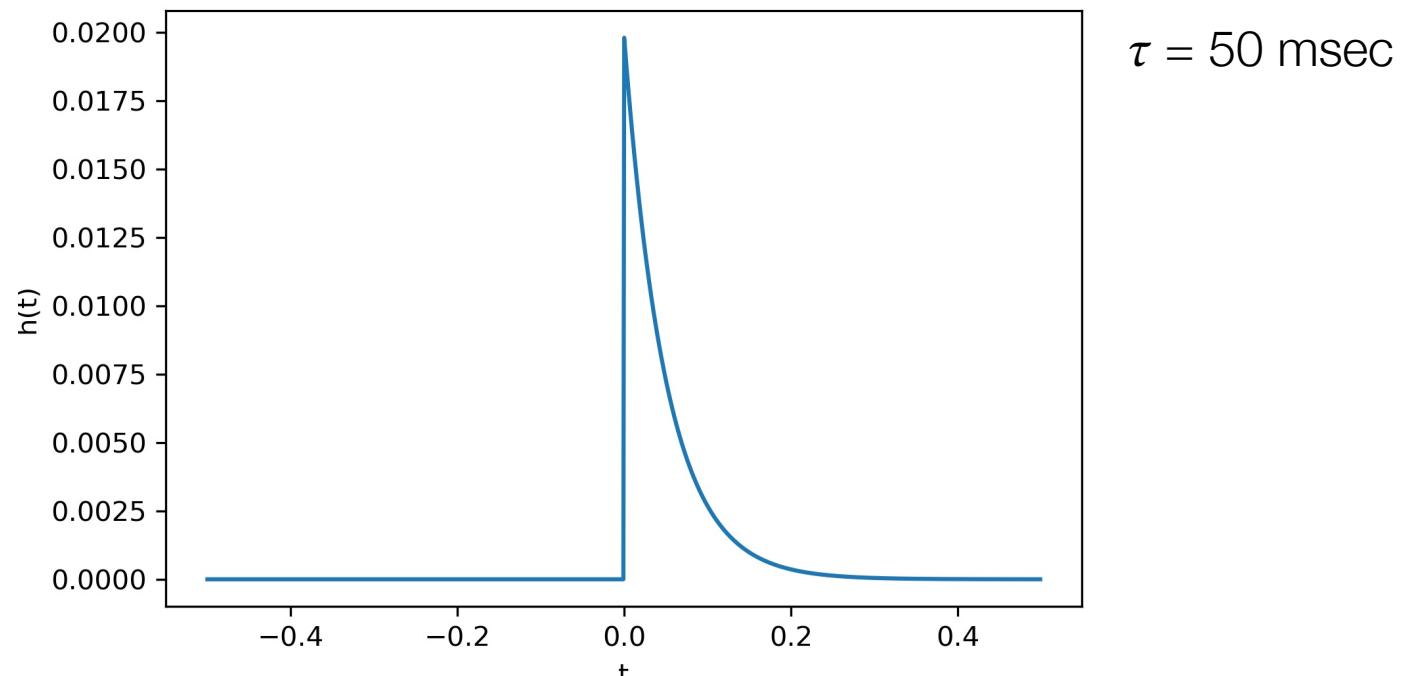
The Neural Engineering Framework (NEF)

Temporal Representation

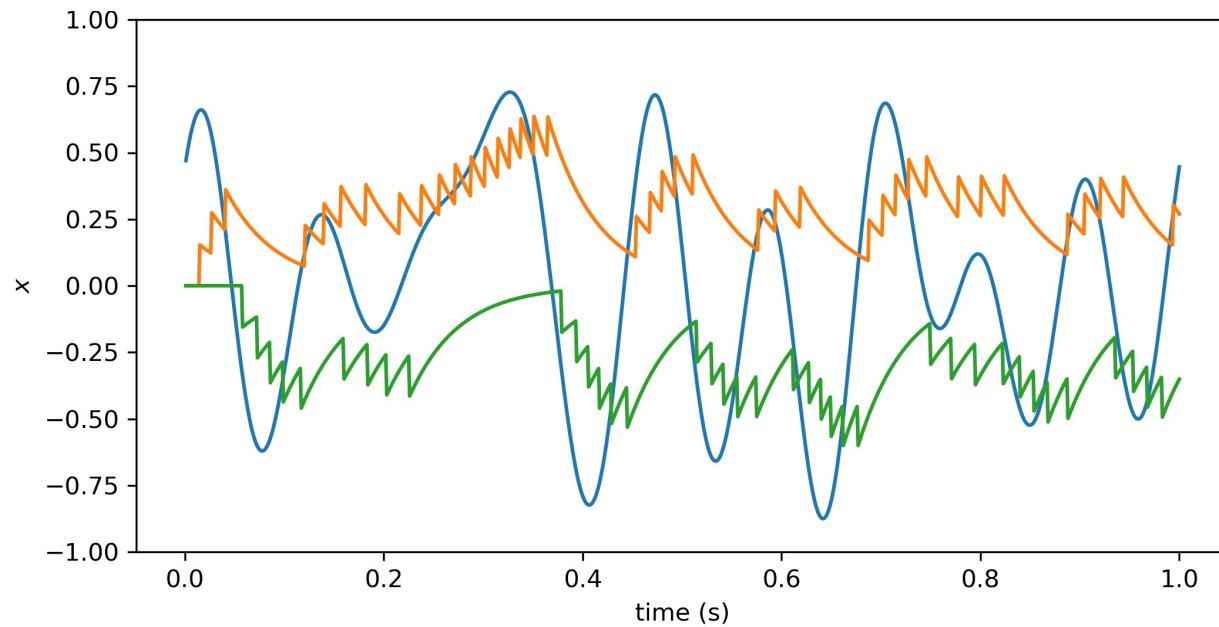
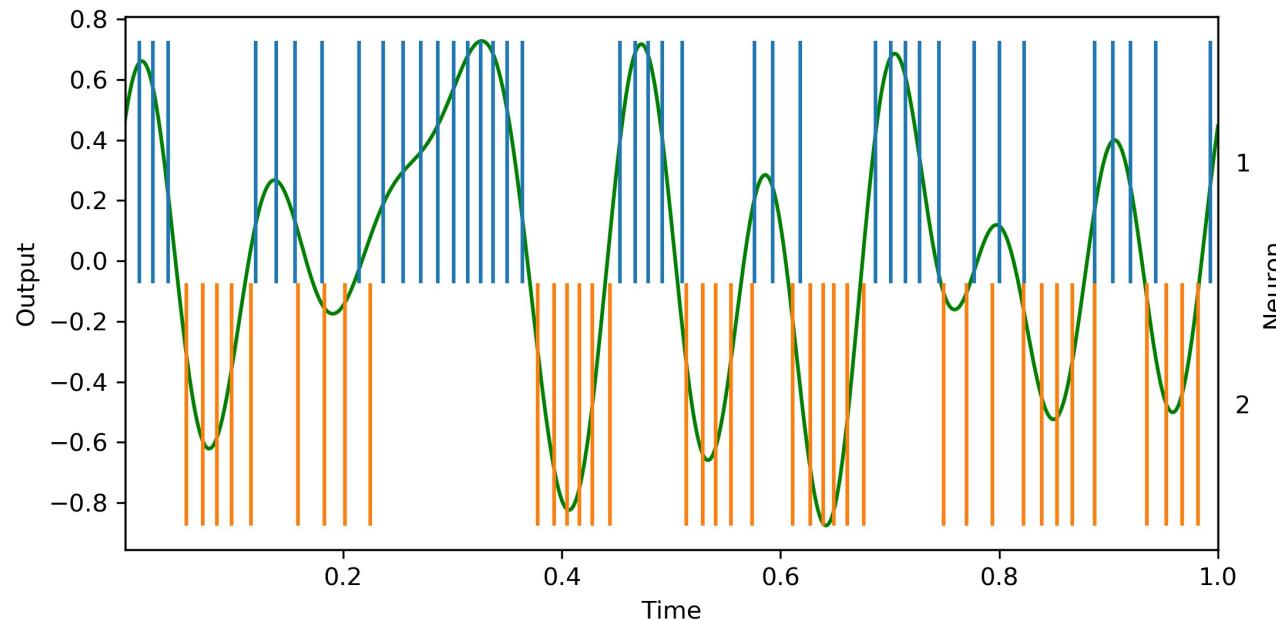
- A biologically plausible model:

$$h(t) = \begin{cases} e^{-t/\tau} & \text{if } t > 0 \\ 0 & \text{otherwise} \end{cases}$$

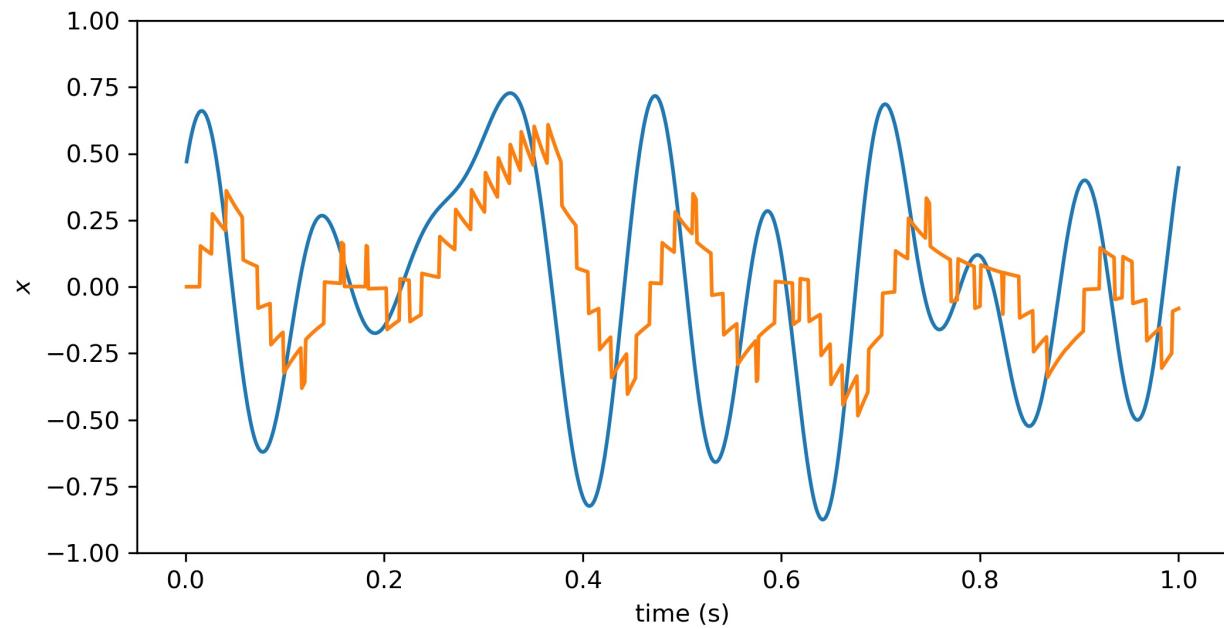
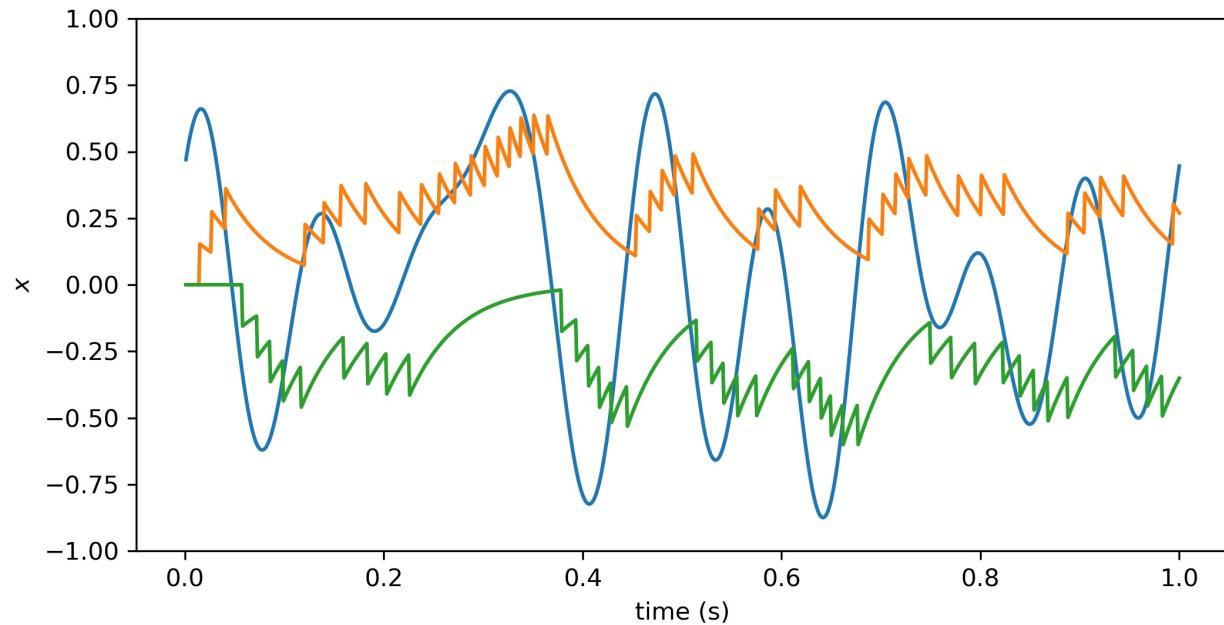
- τ is different for different neurotransmitters and different neurotransmitter receptors (10 - 150 msec)



The Neural Engineering Framework (NEF)



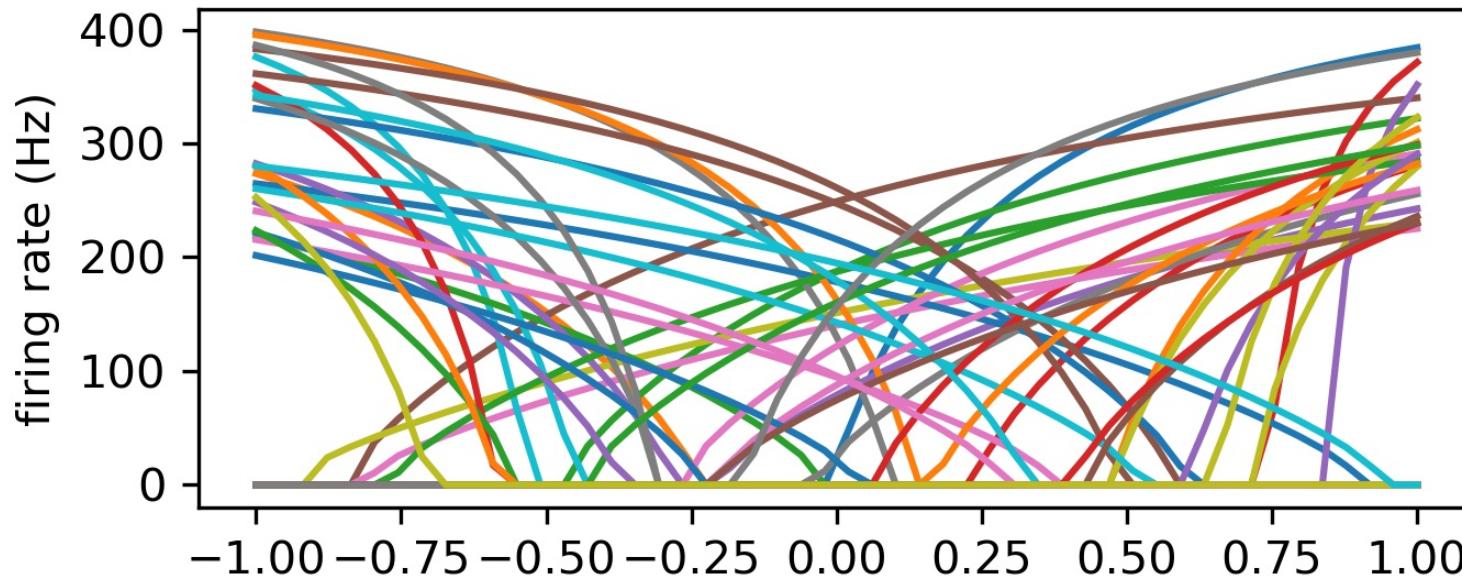
The Neural Engineering Framework (NEF)



The Neural Engineering Framework (NEF)

Representation

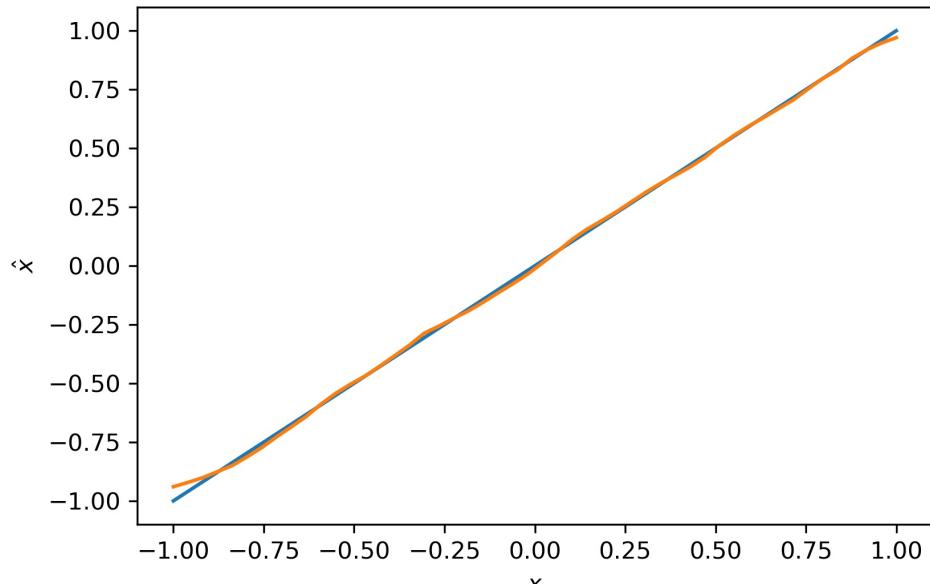
- What functions $f(x)$ can we accurately compute from a group of 50 neurons?



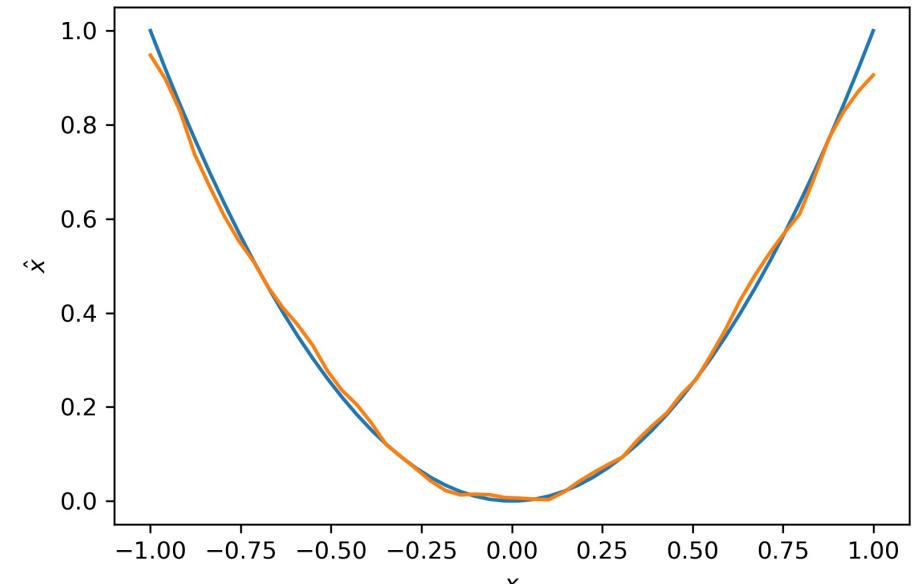
The Neural Engineering Framework (NEF)

Representation

- What functions $f(x)$ can we accurately compute from a group of 50 neurons?



$$f(x) = x$$



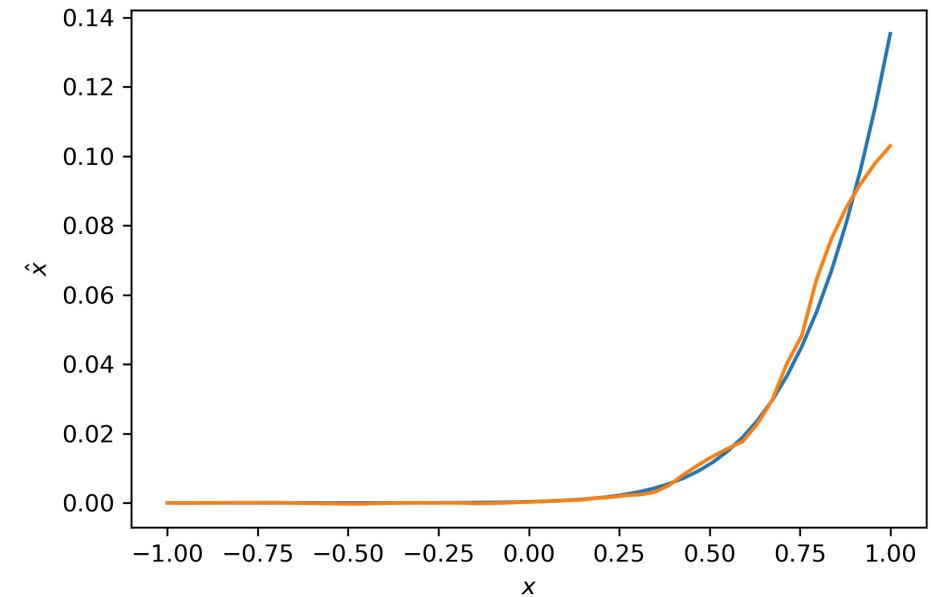
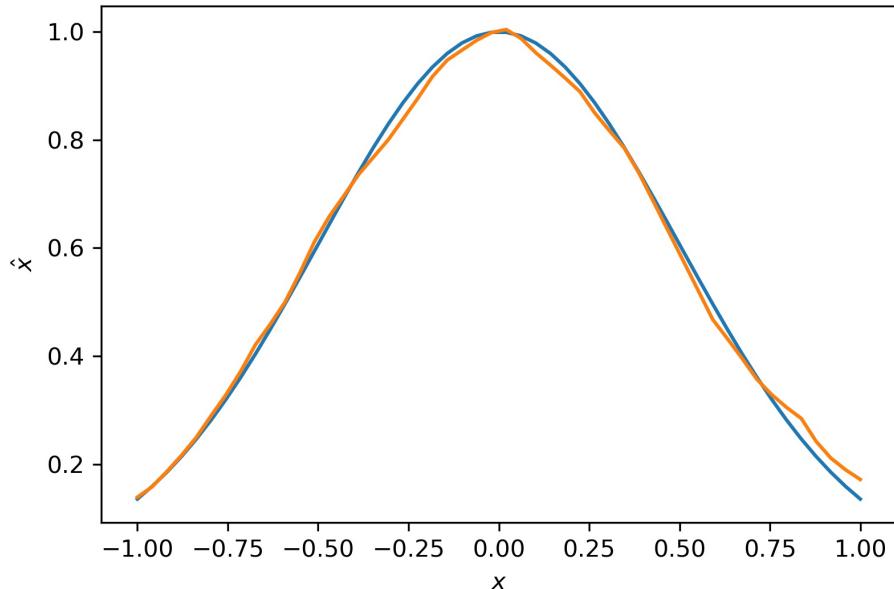
$$f(x) = x^2$$



The Neural Engineering Framework (NEF)

Representation

- What functions $f(x)$ can we accurately compute from a group of 50 neurons?



$$f(x) = e^{-(x-b)^2/(2c^2)}$$



The Neural Engineering Framework (NEF)

Representation

- Because the neural responses are non-independently driven by some variable, only a small part of the subspace of possible neural activity values is ever reached by the population.
- Γ can tell us the correlations between all neurons in the population, providing the information needed to determine that subspace

$$\sum_x a_i x / S = \sum_j \left(\sum_x a_i a_j / S \right) d_j$$

$$\Upsilon = \Gamma d$$

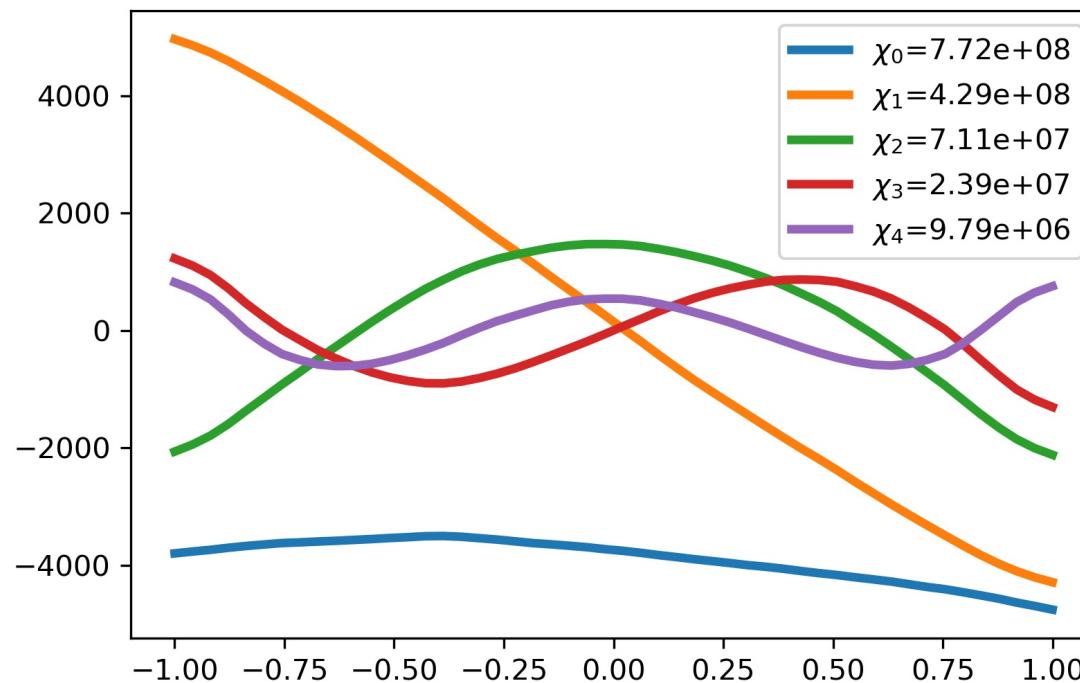
- With SVD (singular value decomposition) **basis functions** for the function space computable from the populations' activity A .
Just as basis vectors define a vector space, so basis functions define a function space.



The Neural Engineering Framework (NEF)

Representation

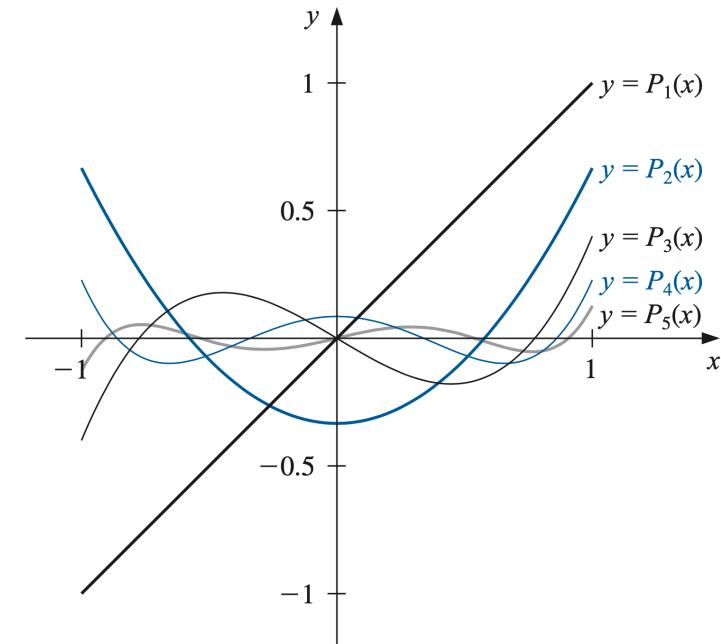
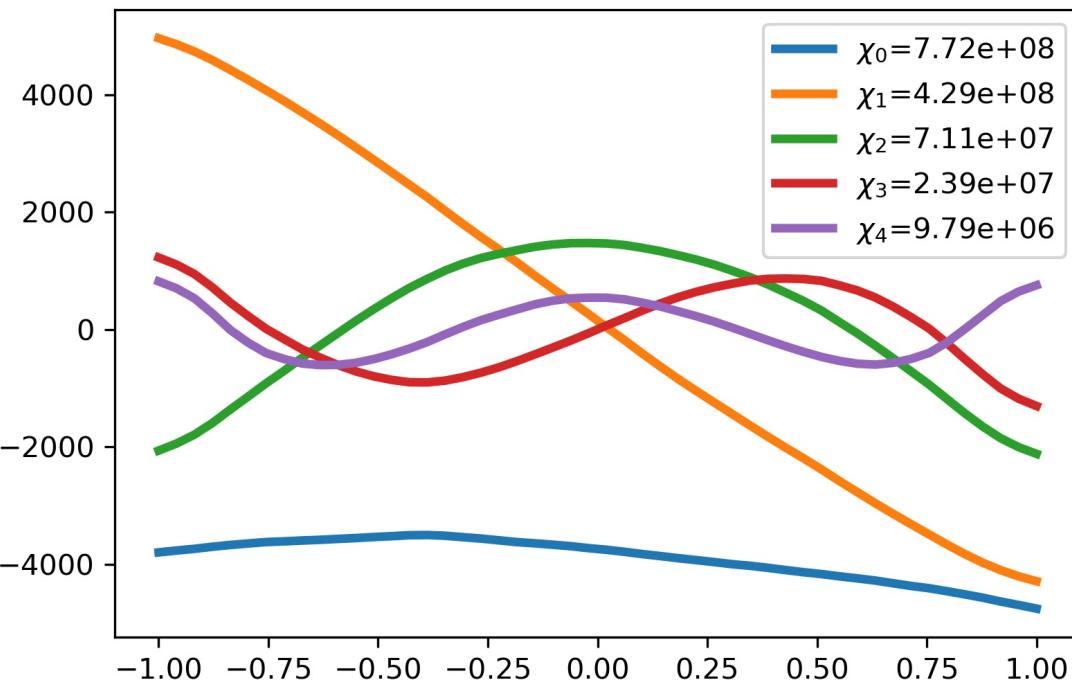
- With neural representation, the basis functions look like:



The Neural Engineering Framework (NEF)

Representation

- With neural representation, the basis functions look like:



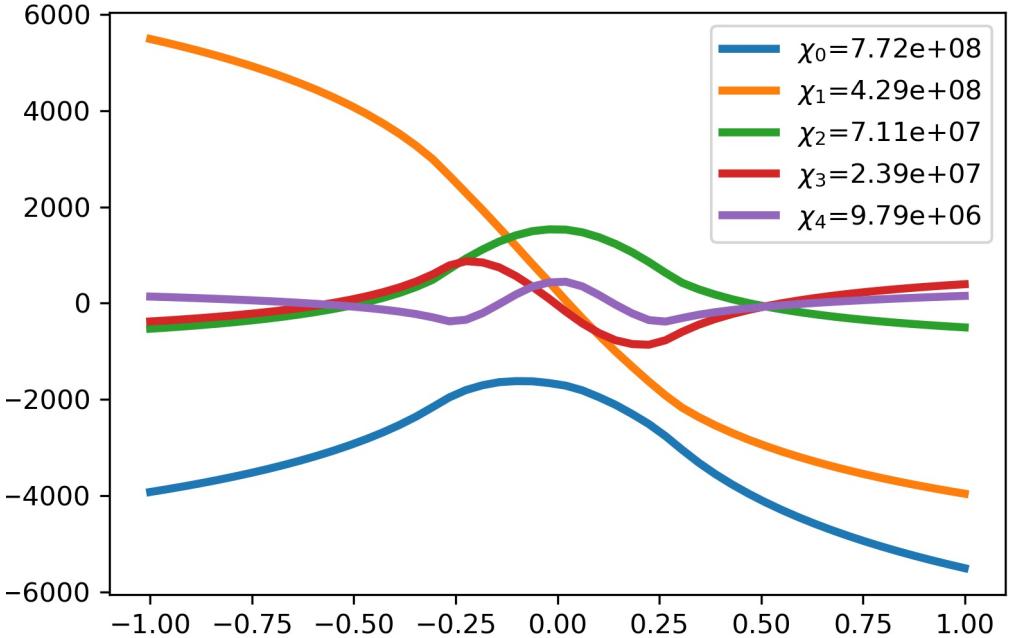
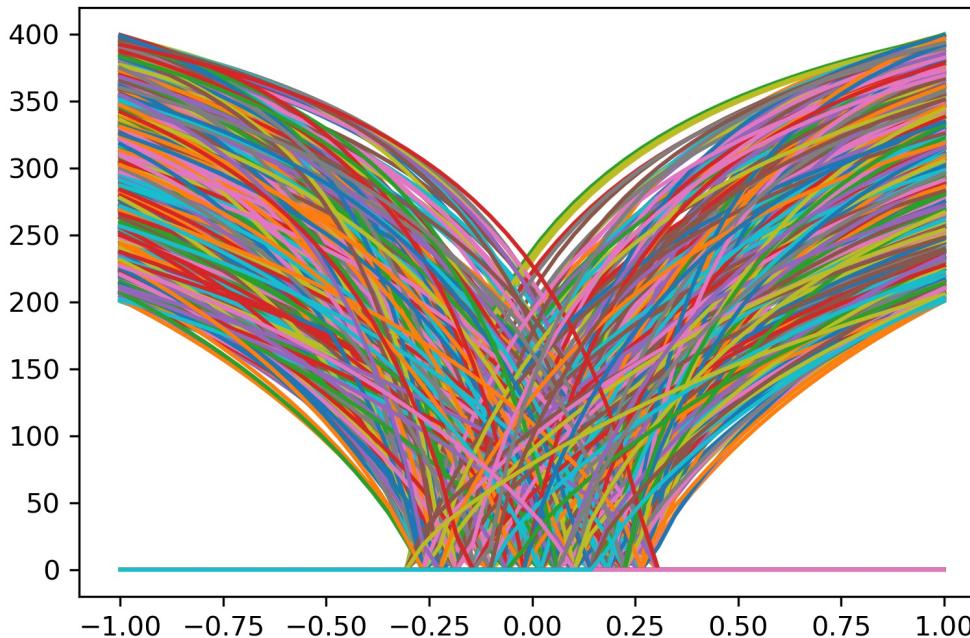
Legendre polynomials

https://en.wikipedia.org/wiki/Legendre_polynomials

The Neural Engineering Framework (NEF)

Representation

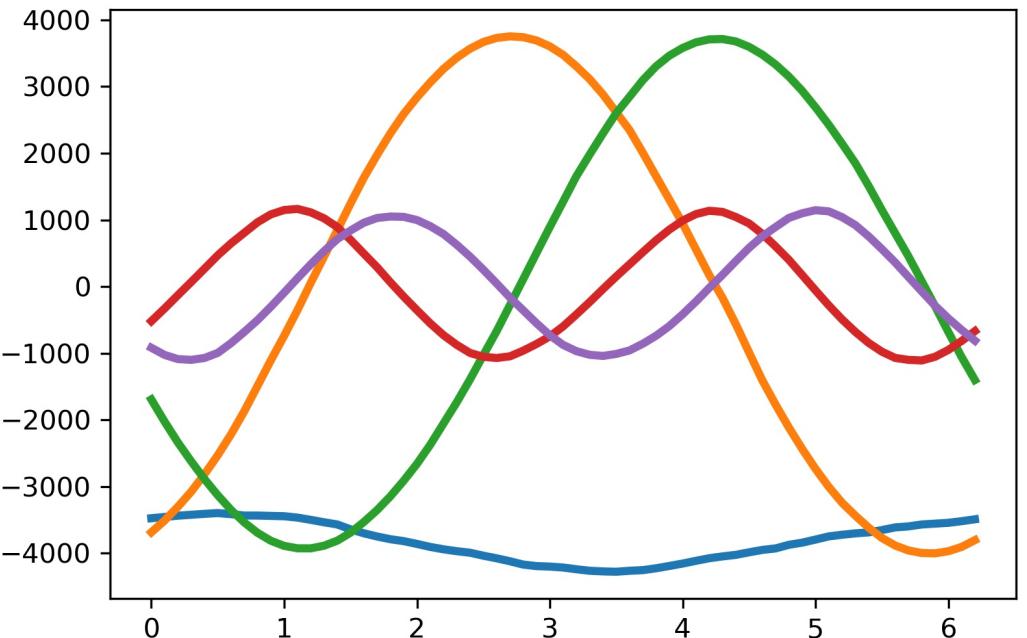
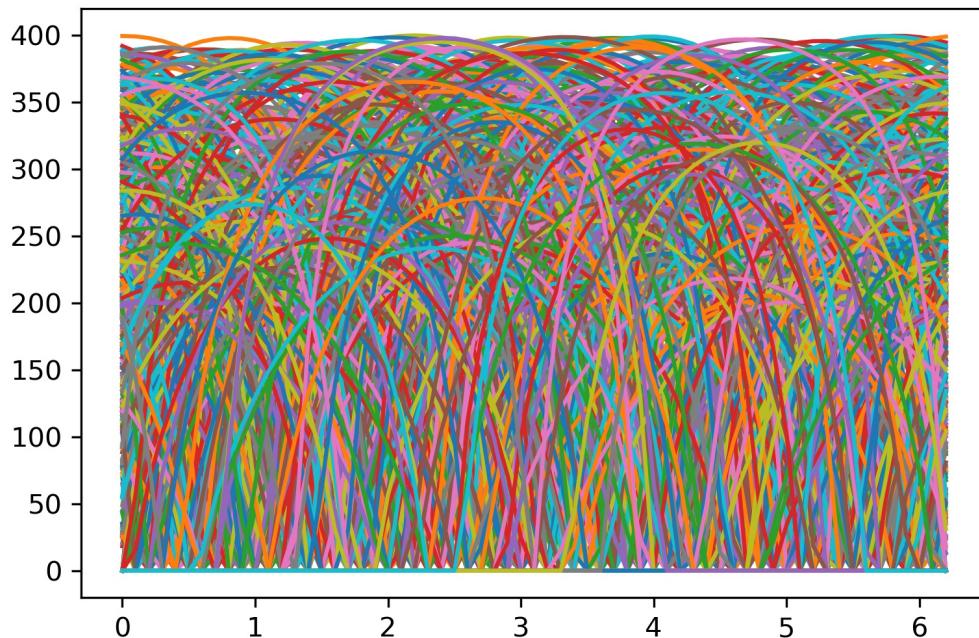
- So we should be able to approximate any function that can be approximated by polynomials
- Changing the tuning curves will change the basis functions. For example:



The Neural Engineering Framework (NEF)

Representation

- So we should be able to approximate any function that can be approximated by polynomials
- Changing the tuning curves will change the basis functions. For example:



Computing with Spiking Neural Networks

Representation

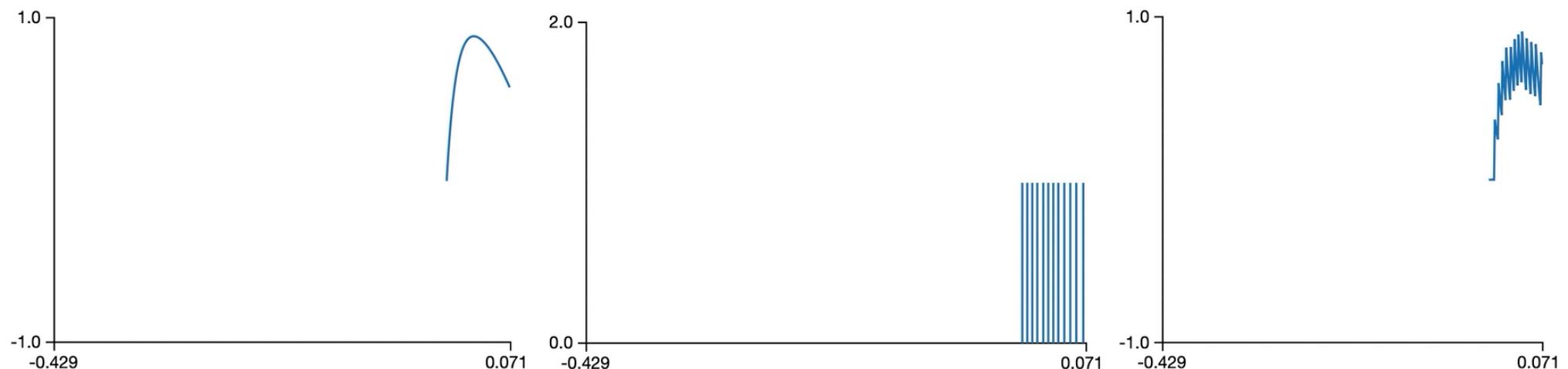
Encoding-Decoding with 1 neuron:



Computing with Spiking Neural Networks

Representation

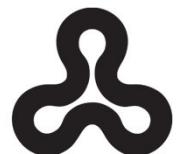
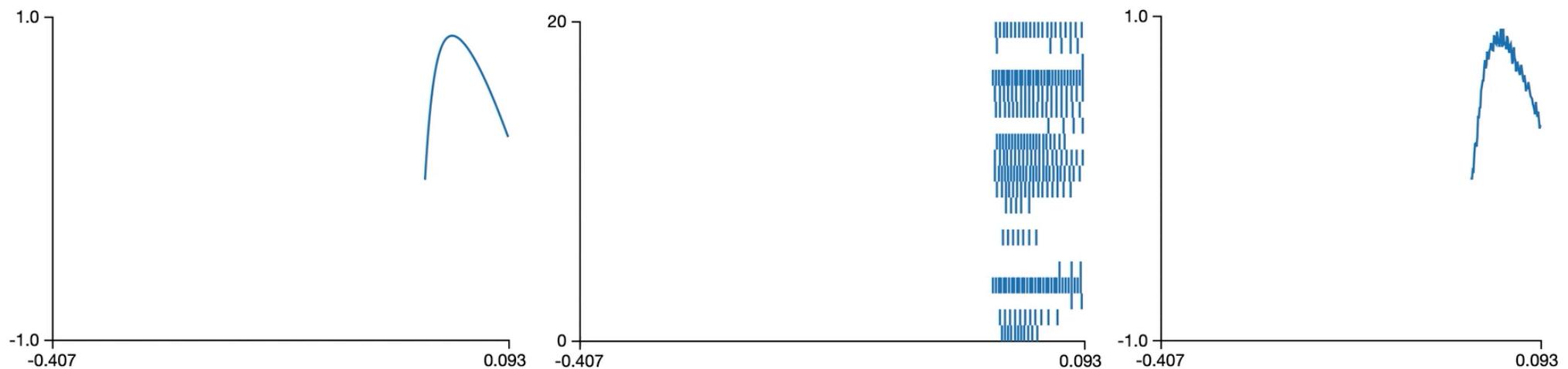
Encoding-Decoding with 2 neurons:



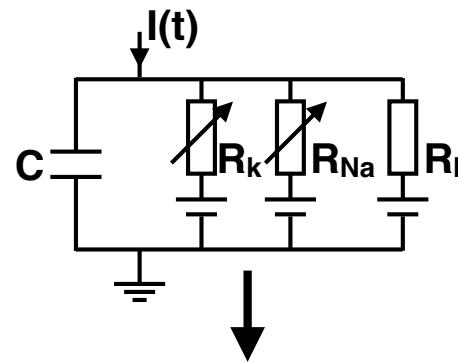
Computing with Spiking Neural Networks

Representation

Encoding-Decoding with 20 neurons:



The Hodgkin-Huxley model

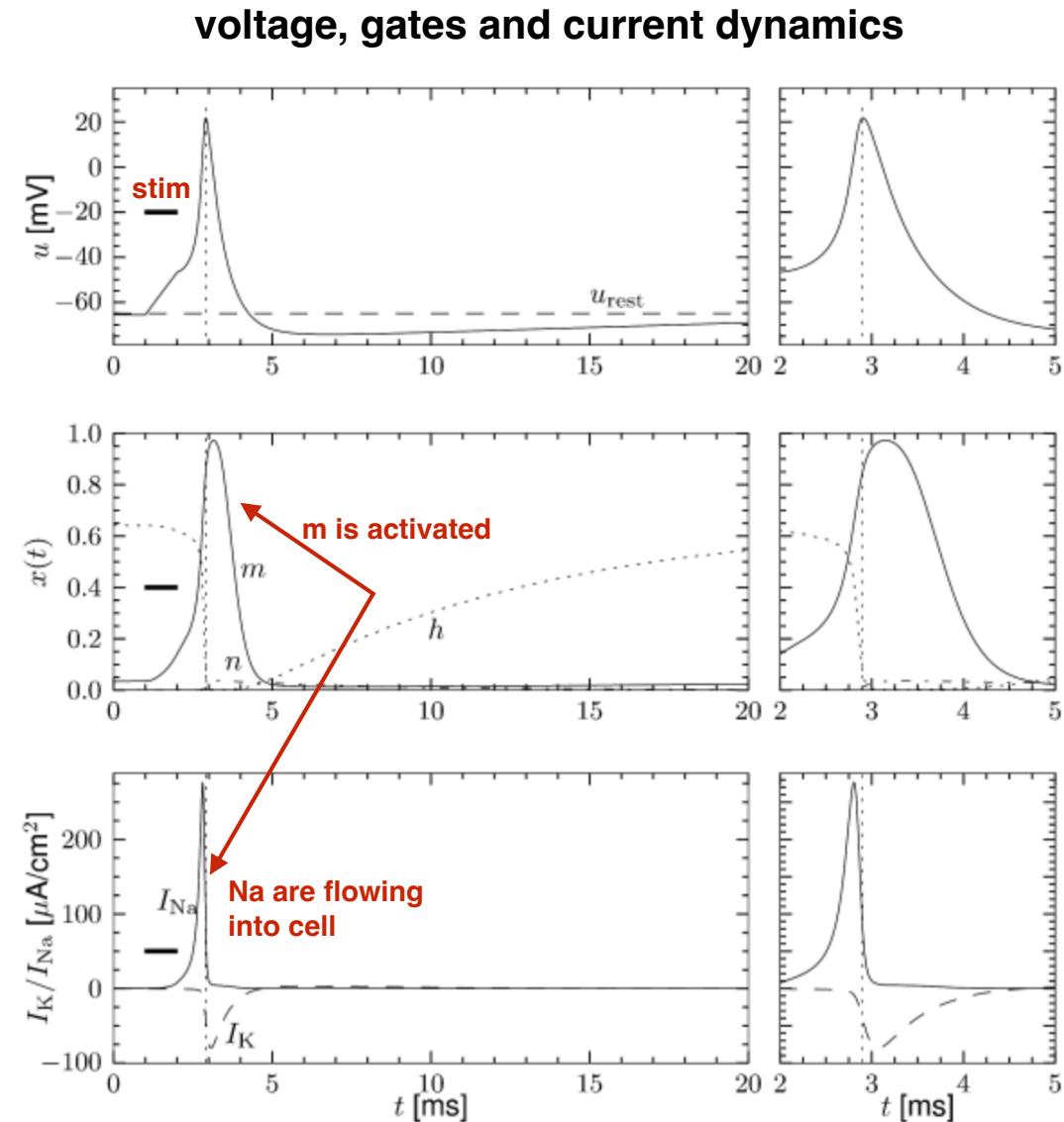


$$C \frac{du}{dt} = -g_{Na} m^3 h (u - E_{Na}) - g_k n^4 (u - E_K) - g_l (u - E_l) - I(t)$$

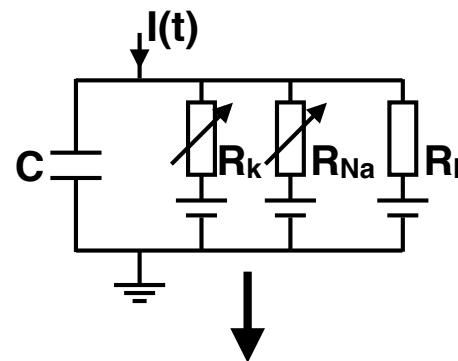
$$\frac{dm}{dt} = -\frac{m - m_0(u)}{\tau_m(u)}$$

$$\frac{dn}{dt} = -\frac{n - n_0(u)}{\tau_m(u)}$$

$$\frac{dh}{dt} = -\frac{h - h_0(u)}{\tau_m(u)}$$



The Hodgkin-Huxley model

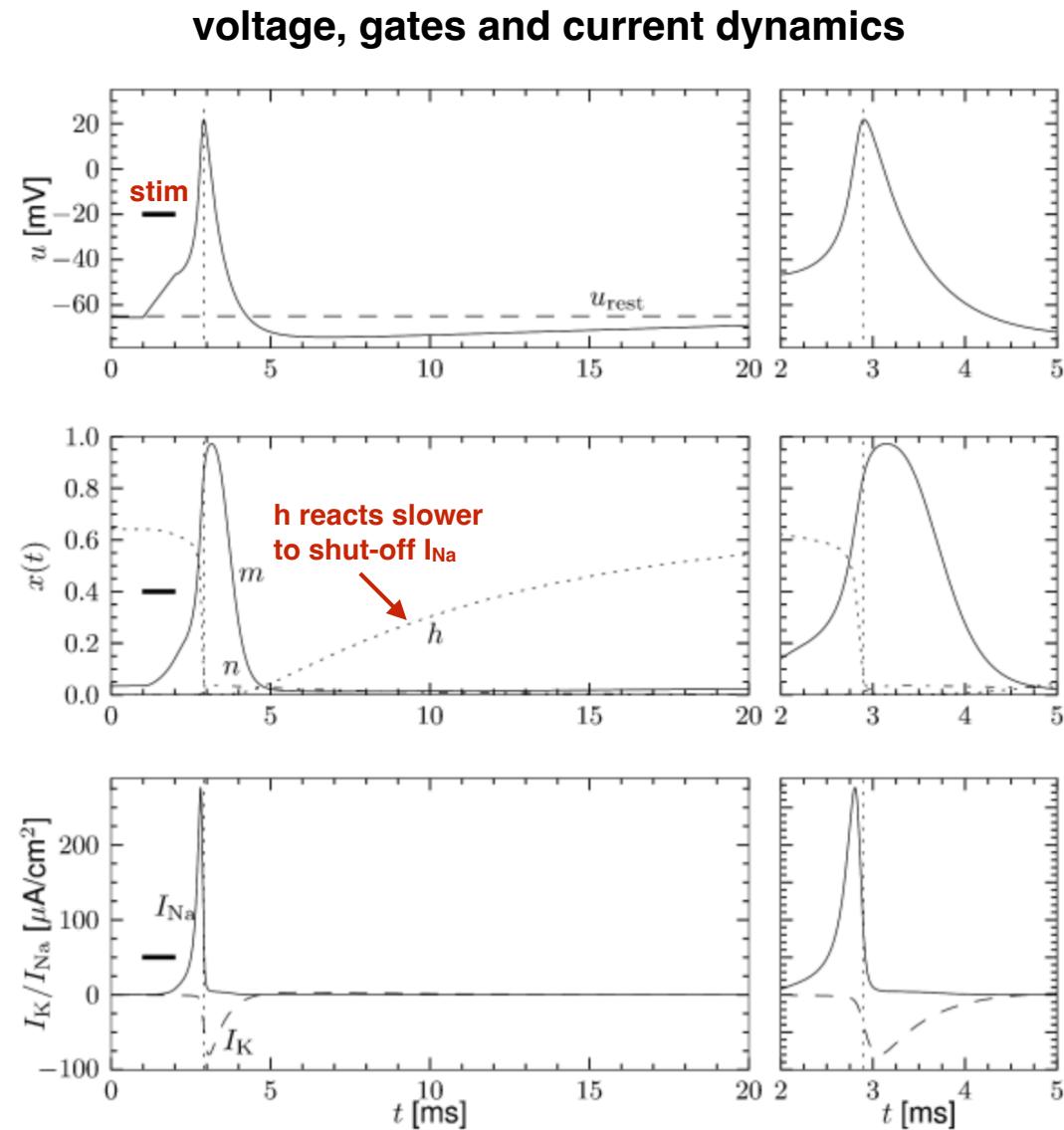


$$C \frac{du}{dt} = -g_{Na} m^3 h (u - E_{Na}) - g_k n^4 (u - E_K) - g_l (u - E_l) - I(t)$$

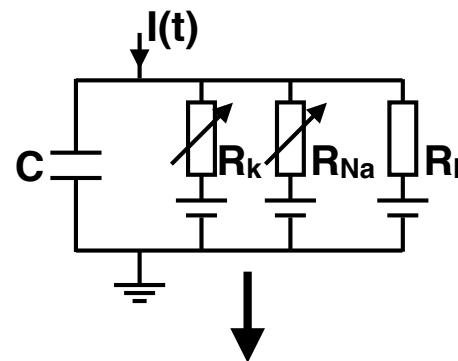
$$\frac{dm}{dt} = -\frac{m - m_0(u)}{\tau_m(u)}$$

$$\frac{dn}{dt} = -\frac{n - n_0(u)}{\tau_m(u)}$$

$$\frac{dh}{dt} = -\frac{h - h_0(u)}{\tau_m(u)}$$



The Hodgkin-Huxley model

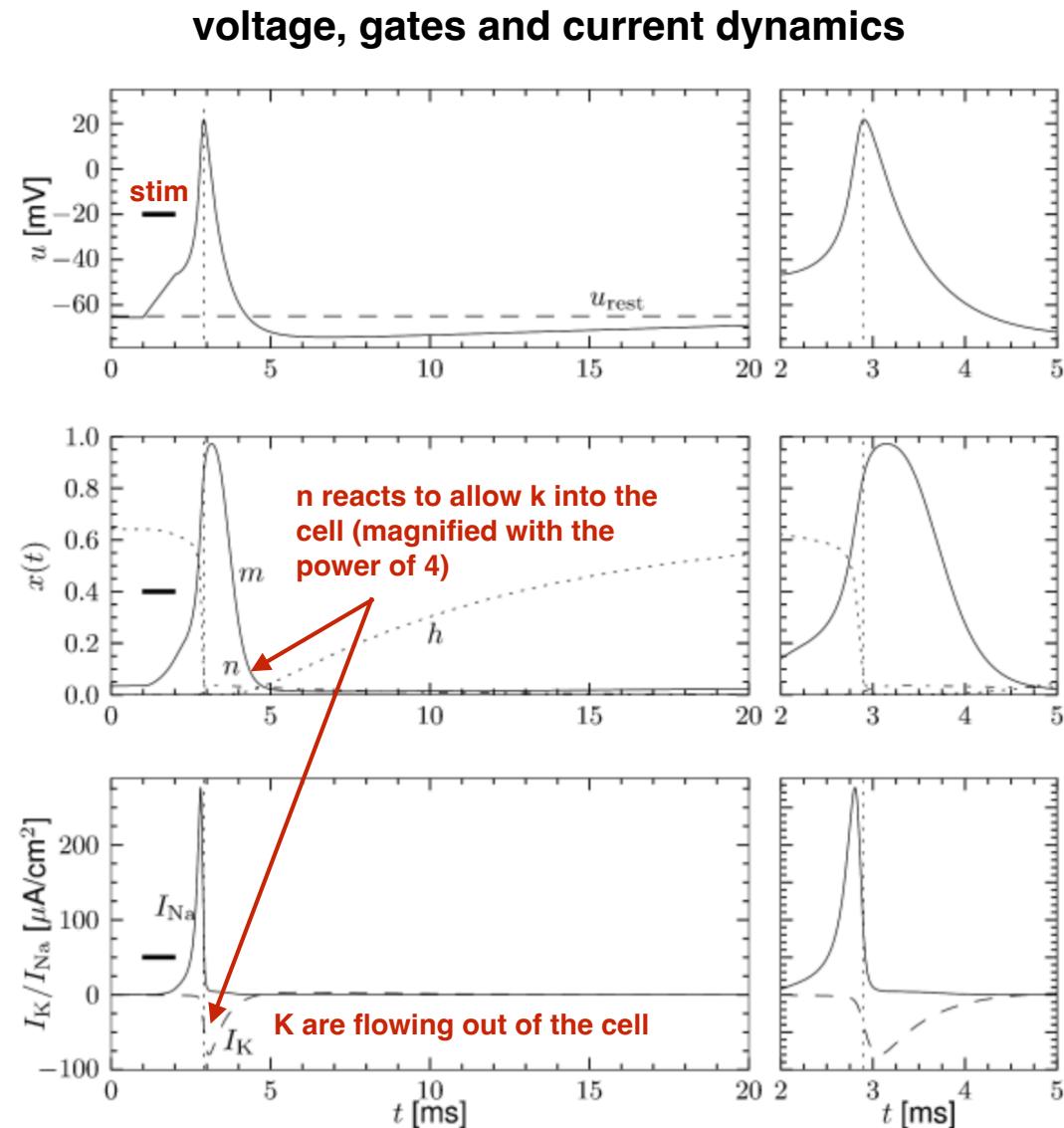


$$C \frac{du}{dt} = -g_{Na} m^3 h (u - E_{Na}) - g_k n^4 (u - E_K) - g_l (u - E_l) - I(t)$$

$$\frac{dm}{dt} = -\frac{m - m_0(u)}{\tau_m(u)}$$

$$\frac{dn}{dt} = -\frac{n - n_0(u)}{\tau_m(u)}$$

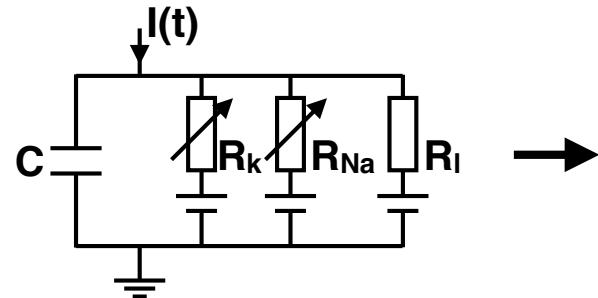
$$\frac{dh}{dt} = -\frac{h - h_0(u)}{\tau_m(u)}$$



Models of Neuronal Dynamics

The Hodgkin-Huxley model

A system of four LDEs



$$\begin{aligned} C \frac{du}{dt} &= -g_{Na}m^3h(u - E_{Na}) - g_kn^4(u - E_K) - g_l(u - E_l) - I(t) \\ \frac{dm}{dt} &= -\frac{m - m_0(u)}{\tau_m(u)} \quad \frac{dn}{dt} = -\frac{n - n_0(u)}{\tau_m(u)} \quad \frac{dh}{dt} = -\frac{h - h_0(u)}{\tau_m(u)} \end{aligned}$$

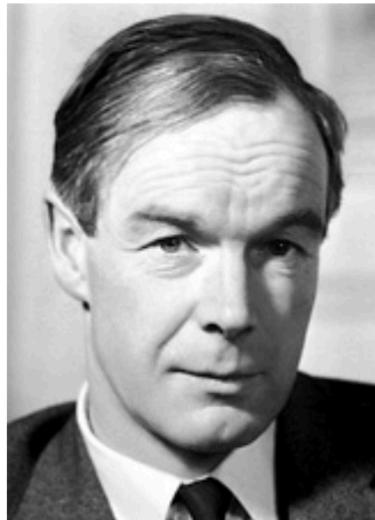
- There is **no strict threshold** - it is coupled with the differential equations
- What we have is a model for a **point neuron** - no spatial structure. Real neurons have dendrites
- **Input injected explicitly** (stimulation current). Real neurons receive spikes, arrived at their synapses.

The Nobel Prize in Physiology or Medicine 1963



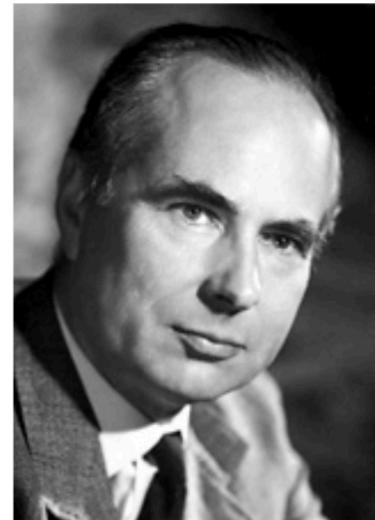
**Sir John Carew
Eccles**

Prize share: 1/3



Alan Lloyd Hodgkin

Prize share: 1/3



**Andrew Fielding
Huxley**

Prize share: 1/3

The Nobel Prize in Physiology or Medicine 1963 was awarded jointly to Sir John Carew Eccles, Alan Lloyd Hodgkin and Andrew Fielding Huxley *"for their discoveries concerning the ionic mechanisms involved in excitation and inhibition in the peripheral and central portions of the nerve cell membrane"*.

