Absolutely, Saptarshi — yes, **Simulink is highly capable of modelling biological neurons**, including:

* **Dendritic trees**
* **Inhibitory and excitatory inputs**
* **Non-linear activation functions**
* **Weight variations by distance**
* **Neuron column structures**
* And even **biophysical models** like leaky integrators and Hodgkin-Huxley models.

**🧠 Your Question Breakdown:**

From the screenshot and notes, your goal seems to be:

"Can we model the *dendritic tree* of a specific neuron using Simulink, accounting for inhibition, excitation, weights based on location, and outputting non-linear activation?"

**Yes — and here's how.**

**✅ 1. Simulink Is Good for This**

Simulink is built for **graphical, block-based modeling of dynamic systems** — making it perfect for neuron simulation. Especially when paired with:

* **Simscape** (physical modeling)
* **Deep Learning Toolbox** (activation functions, layers)
* **MATLAB Function blocks** (custom math)

**🧪 2. Types of Neuron Models You Can Build in Simulink**

| **Model Type** | **Description** | **Simulink Use** |
| --- | --- | --- |
| **Linear Integrator** | Accumulates inputs | Use **Integrator block** |
| **Leaky Integrator** | Adds decay over time | Use **Transfer Function** or custom block |
| **Spiking Neuron** | Fires above threshold | Use **MATLAB Function** or **Compare + Pulse Generator** |
| **Biophysical Model** (e.g. Hodgkin-Huxley) | Simulates ion channels, gates | Needs **Simscape** or **custom equations** |
| **Deep Network / Non-linear activation** | Sigmoid, ReLU, tanh, etc. | Use **Math Function**, **Lookup Table**, or Deep Learning Toolbox |

**🧱 3. How to Build Your Custom Dendritic Neuron in Simulink**

**🧠 Architecture (Based on Your Drawing):**

Multiple weighted inputs (from dendrites) →

Sum (Inhibition/Excitation) →

Activation Function (non-linear) →

Output (firing rate or value)

**✅ Blocks You'll Likely Use:**

| **Function** | **Simulink Block** |
| --- | --- |
| Weighted input | **Gain** block |
| Inhibition | Gain with negative value |
| Summing inputs | **Sum** block |
| Distance-based weight | **Lookup Table** or **MATLAB Function** |
| Activation | **Math Function** or **MATLAB Function** (e.g., tanh, sigmoid) |
| Integration over time | **Integrator**, **Transfer Fcn** |
| Neuron layout | **Subsystem** or **For Each** block |

**🔬 4. Example Model: Columnar Neuron with Dendrites**

**Inputs:**

* Dendrites closer to cortex: Weight = high
* Dendrites farther: Weight = low
* Inhibitory dendrite = negative weight

**Simulink Steps:**

1. Use **Constant** or **Signal Builder** to simulate input spikes.
2. Apply **Gain** blocks to simulate dendritic weights.
3. Use **Sum** block to integrate excitatory and inhibitory effects.
4. Apply **tanh/sigmoid** for activation (using a **Math Function** block or MATLAB Function).
5. Use **Scope** to observe output signal.

**🧪 5. Optional Biological Realism**

If you're aiming for biophysical realism (e.g., ion channel dynamics, action potential thresholds), consider:

* **Simscape > SimElectronics**: to model voltage, resistance, and gating
* **MATLAB Function block**: to implement equations like:
  + Leaky integrator: dV/dt = -V/τ + I(t)
  + Hodgkin-Huxley dynamics

**🧰 6. Starter Example in Simulink**

Would you like me to create a .slx file that:

* Models a neuron with 3 dendrites (2 excitatory, 1 inhibitory)
* Assigns higher weights to dendrites closer to cortex
* Passes signal through a **sigmoid activation**
* Outputs firing pattern to a **Scope**

Let me know and I’ll generate and explain it block-by-block.

**🔎 Final Thoughts**

You're essentially exploring **computational neuroscience with Simulink** — a fantastic and rare combo. Your notes on:

* Columnar organization
* Cortical mapping
* Dendritic weighting
* Integrator vs. leaky neuron models

are exactly the right foundation.

**Understanding Your Simulink Leaky Integrate-and-Fire (LIF) Neuron Model**

**1. What Is Happening in the Model?**

The Simulink model is a simplified electrical representation of how a real neuron works. It captures the essential features of a neuron’s membrane: **integration** of input signals, **leakage** (loss of charge), and **firing** (spike generation) when a threshold is reached, followed by a **reset**.

**Step-by-Step Process**

1. **Input Current (Step Block)**
   * Represents external stimulation or synaptic input to the neuron.
   * In biology, this is like neurotransmitters causing ion flow into the neuron
2. **Summation (Add Block)**
   * Adds together the input current and the leak current.
   * Mimics the neuron summing all incoming signals at its dendrites
3. **Leakage (Gain Block)**
   * Models the natural tendency of the membrane potential to decay back toward rest (leak).
   * Physically, this is due to ions leaking through the membrane, represented by a resistor in parallel with the capacitor
4. **Integration (Integrator Block)**
   * Represents the membrane’s ability to integrate (accumulate) incoming charge over time, like a capacitor.
   * The voltage across this “capacitor” is the membrane potential
5. **Threshold Detection (Relational Operator)**
   * Checks if the membrane potential has reached the firing threshold.
   * In biology, this is the point at which the neuron fires an action potential
6. **Reset Mechanism (Switch and Constant Block)**
   * When the threshold is reached, the membrane potential is reset to a baseline (usually 0 V) using the Constant block.
   * This mimics the neuron quickly returning to its resting state after firing a spike
7. **Output Visualization (Scope)**
   * Shows the membrane potential over time, letting you see the “spikes”

**2. What Does Each Block Represent in a Real Neuron?**

| **Simulink Block** | **Neuron Part / Function** | **Biological Analogy** |
| --- | --- | --- |
| Step (Input) | Synaptic Input | Incoming signals from other neurons |
| Add | Summation at Soma | Combining all input signals |
| Gain (Leak) | Membrane Leak (Resistor) | Ion leakage through membrane |
| Integrator | Membrane Capacitance | Charge accumulation in cell membrane |
| Relational Operator | Spike Threshold Detector | Voltage-gated channels triggering spike |
| Switch | Reset Mechanism | Rapid reset after action potential |
| Constant | Resting Potential | Baseline voltage after spike |
| Scope | Recording Electrode | Measurement of membrane potential |

**3. Why Do We Use a Constant Block?**

* The **Constant block** is used to provide a fixed value (usually 0 V) to which the membrane potential resets after a spike
* In real neurons, after firing, the membrane potential is rapidly reset to a lower value (resting or hyperpolarized state), so the Constant block ensures this behavior is modeled

**4. How Does This Model Represent a Real Neuron?**

* **Integration:** The Integrator block accumulates input, just like the membrane accumulates charge from synaptic currents
* **Leakage:** The Gain block with negative feedback mimics the natural decay of the membrane potential due to ion leakage
* **Threshold and Firing:** The Relational Operator and Switch blocks check for threshold crossing and trigger a reset, representing the all-or-none action potential of neurons
* **Reset:** The Constant block ensures the neuron returns to its resting state after firing, just like real neurons

**5. Circuit Analogy**

The model is based on an **RC (resistor-capacitor) circuit**:

* **Capacitor (Integrator):** Stores charge, representing the membrane’s ability to hold voltage.
* **Resistor (Leak/Gain):** Allows charge to leak away, representing ion channels that let ions flow out.
* **Input Current (Step):** Represents synaptic currents entering the neuron

The equation governing this is:

CdVdt=Iin−VRC \frac{dV}{dt} = I\_{in} - \frac{V}{R}CdtdV=Iin−RV

where VVV is the membrane potential, IinI\_{in}Iin is the input current, CCC is capacitance, and RRR is resistance

**6. What’s Happening During Simulation?**

* **At Rest:** No input, membrane potential stays at baseline.
* **Input Applied:** Step current causes membrane potential to rise (integration).
* **Leakage:** The rise is not linear; the leak slows the increase, causing an exponential curve.
* **Threshold Reached:** When the membrane potential hits the set threshold, the neuron “fires” (spike).
* **Reset:** The potential is immediately set back to baseline by the Constant block.
* **Repeat:** With continuous input, this cycle repeats, generating a train of spikes

**7. Why Is This Model Useful?**

* **Simplicity:** Captures the essential features of neuronal firing without complex biology.
* **Flexibility:** Lets you see how changes in input, resistance, capacitance, or threshold affect firing patterns.
* **Foundation:** Forms the basis for more complex models and neural networks

NEURON  
Here's a line-by-line explanation of the code:

**1. Import Libraries**

python

**from** neuron **import** h, gui *# Import NEURON's core and GUI modules*

**import** numpy **as** np *# For numerical operations*

**import** matplotlib.pyplot **as** plt *# For plotting*

**2. Initialize NEURON**

python

h.load\_file("stdrun.hoc") *# Load standard simulation library*

**3. Create Neuron Morphology**

python

soma = h.Section(name='soma') *# Create a cylindrical section*

soma.L = 30 *# Length (μm)*

soma.diam = 30 *# Diameter (μm) - makes it spherical*

soma.Ra = 100 *# Axial resistance (Ω·cm)*

soma.cm = 1 *# Membrane capacitance (μF/cm²)*

**4. Add Passive Properties**

python

soma.insert('pas') *# Insert passive ("leak") channels*

soma.g\_pas = 0.0001 *# Leak conductance (S/cm²)*

soma.e\_pas = -65 *# Resting potential (mV)*

**5. Add Active Channels (Hodgkin-Huxley)**

python

soma.insert('hh') *# Insert Hodgkin-Huxley Na+/K+ channels*

**for** seg **in** soma: *# Loop through all segments (just 1 in this case)*

seg.hh.gnabar = 0.12 *# Max sodium conductance*

seg.hh.gkbar = 0.036 *# Max potassium conductance*

seg.hh.gl = 0.0003 *# Leak conductance*

seg.hh.el = -65 *# Leak reversal potential*

**6. Set Up Current Injection**

python

stim = h.IClamp(soma(0.5)) *# Current clamp at middle of soma*

stim.delay = 100 *# Start time (ms)*

stim.dur = 500 *# Duration (ms)*

stim.amp = 0.12 *# Amplitude (nA)*

**7. Set Up Recording**

python

*# Record simulation time*

t = h.Vector().record(h.\_ref\_t)

*# Record membrane potential*

v = h.Vector().record(soma(0.5).\_ref\_v)

*# Record sodium current*

ina = h.Vector().record(soma(0.5).\_ref\_ina)

*# Record potassium current*

ik = h.Vector().record(soma(0.5).\_ref\_ik)

**8. Spike Detection**

python

spike\_times = h.Vector() *# Vector to store spike times*

*# Create spike detector at soma midpoint*

nc = h.NetCon(soma(0.5).\_ref\_v, None, sec=soma)

nc.threshold = -20 *# Spike detection threshold (mV)*

nc.record(spike\_times) *# Record spike times*

**9. Run Simulation**

python

h.finitialize(-65) *# Initialize voltage to -65 mV*

h.continuerun(700) *# Run simulation for 700 ms*

**10. Convert Data for Analysis**

python

t = np.array(t) *# Convert time to NumPy array*

v = np.array(v) *# Convert voltage to array*

ina = np.array(ina) *# Convert Na+ current to array*

ik = np.array(ik) *# Convert K+ current to array*

spikes = np.array(spike\_times) *# Convert spikes to array*

**11. Plot Results**

python

plt.figure(figsize=(14, 10)) *# Create large figure*

*# Plot 1: Membrane Potential*

plt.subplot(3, 1, 1)

plt.plot(t, v, 'b', linewidth=2) *# Voltage trace*

plt.axhline(-65, color='gray', linestyle=':', alpha=0.5, label='Rest')

plt.axhline(-20, color='r', linestyle='--', alpha=0.7, label='Threshold')

plt.ylabel('V\_m (mV)')

plt.legend()

plt.title('Membrane Potential')

plt.ylim(-80, 50) *# Set voltage range*

*# Plot 2: Ion Currents*

plt.subplot(3, 1, 2)

plt.plot(t, ina, 'r', label='Na+ Current (inward)', linewidth=2)

plt.plot(t, ik, 'purple', label='K+ Current (outward)', linewidth=2)

plt.ylabel('Current (mA/cm²)')

plt.legend()

plt.title('Ion Currents')

*# Plot 3: Input Current*

plt.subplot(3, 1, 3)

*# Create input current trace*

input\_current = [stim.amp **if** (stim.delay <= x < stim.delay + stim.dur) **else** 0 **for** x **in** t]

plt.plot(t, input\_current, 'g', linewidth=2)

plt.ylabel('Input Current (nA)')

plt.xlabel('Time (ms)')

plt.title('Stimulus Current')

plt.tight\_layout() *# Improve spacing*

plt.show() *# Display plots*

**12. Spike Analysis**

python

**print**(f"Number of spikes: {len(spikes)}")

**if** len(spikes) > 1:

isi = np.diff(spikes) *# Interspike intervals*

**print**(f"Mean ISI: {np.mean(isi):.2f} ms")

**print**(f"Firing rate: {1000/np.mean(isi):.2f} Hz")

**13. Detailed Spike View**

python

**if** len(spikes) > 0:

plt.figure(figsize=(10, 6))

spike\_time = spikes[0]

*# Create time window around first spike*

window = (t >= spike\_time - 5) & (t <= spike\_time + 20)

*# Plot spike shape*

plt.subplot(2, 1, 1)

plt.plot(t[window], v[window], 'b', linewidth=3)

plt.axhline(-65, color='gray', linestyle=':', label='Rest')

plt.ylabel('V\_m (mV)')

plt.title('Single Spike Shape')

plt.legend()

*# Plot currents during spike*

plt.subplot(2, 1, 2)

plt.plot(t[window], ina[window], 'r', label='Na+', linewidth=2)

plt.plot(t[window], ik[window], 'purple', label='K+', linewidth=2)

plt.ylabel('Current (mA/cm²)')

plt.xlabel('Time (ms)')

plt.legend()

plt.title('Ion Currents During Spike')

plt.tight\_layout()

plt.show()

**Key Biophysical Concepts Implemented:**

1. **Passive Properties**:
   * Leak conductance (g\_pas) sets baseline membrane permeability
   * Resting potential (e\_pas) sets equilibrium voltage
2. **Active Channels**:
   * Sodium channels (gnabar) for rapid depolarization
   * Potassium channels (gkbar) for repolarization
   * Leak channels (gl) for baseline conductance
3. **Current Injection**:
   * Square pulse input to simulate experimental stimulation
   * Timing parameters control stimulus onset and duration
4. **Spike Generation**:
   * Threshold detection at -20 mV
   * Natural spike shape from ion channel dynamics
   * Afterhyperpolarization (AHP) from persistent K+ current
5. **Analysis**:
   * Spike counting and timing
   * Firing rate calculation
   * Detailed view of spike biophysics

This code implements a biologically realistic neuron model that demonstrates:

* Threshold-dependent spiking
* Sodium activation/deactivation
* Potassium-mediated repolarization
* Afterhyperpolarization (AHP)
* Current-spike relationships

**What is going on with the sodium graph?**

What you are seeing in your plot—a **second, smaller sodium current deflection after the main spike**—is a well-known feature of Hodgkin-Huxley-type models when the membrane is depolarized for an extended period. Here’s why this happens, and why it is biophysically accurate:

## **Why Does the Sodium Current Increase Again After the Spike?**

## **1. Sodium Channel Inactivation and Recovery**

* During the action potential upstroke, **Na⁺ channels open rapidly**, producing a large inward current (the main negative peak).
* As the spike peaks, **Na⁺ channels inactivate** (the h-gate closes), and the current drops quickly back toward zero—even though the stimulus is still present[1](https://neuronaldynamics.epfl.ch/online/Ch2.S2.html)[2](https://en.wikipedia.org/wiki/Action_potential).
* After the spike, the membrane potential repolarizes (due to K⁺ current), and the Na⁺ channels begin to **recover from inactivation** (h-gate reopens).

## **2. Prolonged Depolarization and Repetitive Spiking**

* If the **stimulus current is still ON** (as in your simulation), the membrane remains depolarized.
* Once enough Na⁺ channels have recovered from inactivation, **they can open again** if the voltage is still above threshold, leading to another, smaller sodium current as the next spike begins.
* This is why you see a new Na⁺ current deflection before the next spike—**it’s the start of the next action potential**.

## **3. Sodium Current Returns to Zero**

* After each spike, Na⁺ channels inactivate again, and the current drops back to zero.
* This cycle repeats for each spike during sustained stimulation.

## **What the Plot Shows**

* **First big Na⁺ current dip:** Main spike upstroke (channels open).
* **Quick return to zero:** Channels inactivate (h-gate closes).
* **Second, smaller Na⁺ dip:** Channels recover from inactivation and open again as the next spike begins.
* **Between spikes:** Na⁺ current is near zero because most channels are inactivated or closed.

## **Supporting Evidence from Literature**

* **"At high values of the sodium conductance is slowly shut off due to the factor h. As indicated in Fig. 2.3B, the time constant τ\_h is always larger than τ\_m. Thus the variable h which inactivates the channels reacts more slowly to the voltage increase than the variable m which opens the channel. On a similar slow time scale, the potassium (K) current sets in."** [1](https://neuronaldynamics.epfl.ch/online/Ch2.S2.html" \t "_blank)
* **"A cell that has just fired an action potential cannot fire another one immediately, since the Na+ channels have not recovered from the inactivated state. The period during which no new action potential can be fired is called the absolute refractory period... Even after a sufficient number of sodium channels have transitioned back to their resting state, it frequently happens that a fraction of potassium channels remains open, making it difficult for the membrane potential to depolarize, and thereby giving rise to the relative refractory period."** [2](https://en.wikipedia.org/wiki/Action_potential" \t "_blank)

## **Summary Table**

| **Phase** | **Na⁺ Current Behavior** | **Why?** |
| --- | --- | --- |
| Spike upstroke | Large inward (negative) peak | Na⁺ channels open rapidly (m-gate) |
| Repolarization/AHP | Returns to zero | Na⁺ channels inactivate (h-gate closes) |
| Between spikes | Near zero | Most Na⁺ channels inactivated |
| Next spike begins | New inward deflection | Na⁺ channels recover, open again |

## **Bottom Line**

* The **second rise in Na⁺ current** after the main spike is the start of the next action potential, as Na⁺ channels recover from inactivation and the cell depolarizes again.
* **This is normal and expected** in repetitive spiking with sustained current injection.
* The **Na⁺ current always returns to zero** between spikes because of inactivation, and only reactivates when the next spike is triggered.

If you zoom in on a single spike during a train, you’ll see this cycle clearly: **fast Na⁺ activation → inactivation → recovery → next activation**. This is a hallmark of the Hodgkin-Huxley mechanism and is exactly what you should see in your simulation.

**Why is the height decreasing as I increase the input current?**When you increase the input current amplitude from 0.5 to 1.0 nA and observe that the spike height **decreases** (from ~30 mV to 18 mV), you are seeing a classic phenomenon in Hodgkin-Huxley models and real neurons called **depolarization block**.

## **Why Does This Happen?**

* **Depolarization block:**  
  When the input current is very strong, the neuron's membrane potential remains depolarized for a prolonged period.
* **Sodium channel inactivation:**  
  Sustained depolarization causes most sodium channels to become inactivated. Without available sodium channels, the neuron cannot generate a full-amplitude action potential.
* **Reduced spike height:**  
  As a result, each spike is smaller because the upstroke (driven by Na⁺ influx) is blunted or fails entirely.
* **Potassium dynamics:**  
  The neuron may also not repolarize fully between spikes, further reducing spike amplitude.

## **Summary Table**

| **Input Current** | **Spike Height** | **Explanation** |
| --- | --- | --- |
| Low/Moderate | Normal (~30-40 mV) | Full Na⁺ channel availability |
| High (1.0 nA) | Reduced (~18 mV) | Depolarization block, Na⁺ inactivation |

## **Bottom Line**

This is a **well-known and realistic behavior** in HH-type models:

* **Too much current** can actually **reduce** or even abolish spike amplitude due to sodium channel inactivation and depolarization block.

If you want to avoid this, use input currents that are just above threshold and not excessively high.

Model of the neuro-muscular junction – motor neuron activates the muscle fibre…this is what we are modelling with the HH model…here the potassium channel shouldn’t go down at the same rate as Na+ (not only voltage activated K+ channels)…Ca2+ activated chloride channels may shut down the potassium channels we still need to check this….need an explanation   
  
Voltage gated potassium channel and Calcium gated potassium channel…plot these two graphs…plot K and Na conductance vs Time  
  
Plot K, Na current and the spike voltage vs Time in the same graph  
  
whats the reason of AHP then give an explanation

**Why K+ Conductance Goes Up and Down with Each Spike**

**The Standard HH Model Limitation**

Your current model uses the **standard Hodgkin-Huxley delayed rectifier K+ channels**, which:

* **Activate** during depolarization (spike upstroke)
* **Deactivate** during repolarization (spike downstroke)
* Do **NOT** accumulate or stay open between spikes

**What Your Graph Shows (Correctly)**

1. **Na+ conductance (red)**: Rapid activation → inactivation with each spike
2. **K+ conductance (purple)**: Activation during spike → deactivation after spike
3. **Both oscillate** because they're both voltage-dependent channels

**How AHP Still Occurs Despite Oscillating K+ Conductance**

The **AHP (undershoot)** in your membrane potential happens because:

1. **Slow K+ Channel Deactivation**: Even though K+ conductance decreases after each spike, it doesn't return to zero instantly
2. **Temporary High K+ Permeability**: For ~5-10ms after each spike, K+ permeability is higher than resting
3. **Drives Vm toward E\_K**: This pulls membrane potential toward -77mV (K+ equilibrium), creating undershoot below rest (-54.3mV)
4. **Gradual Return**: As K+ channels fully close, membrane returns to resting potential

**Why You Expected Different Behavior**

From your notes mentioning **"calcium-gated K+ channels"** - you're thinking of **calcium-activated K+ currents**, which behave differently:

**Calcium-Activated K+ Channels:**

* **Triggered by**: Ca2+ influx during spikes (not just voltage)
* **Accumulate**: Ca2+ builds up with repeated spiking
* **Slow decay**: Ca2+ removal is slow, keeping channels open longer
* **Result**: Progressive increase in K+ conductance, leading to spike frequency adaptation

**Evidence from Search Results**

The literature confirms your expectation:

* **"Ca2+-activated K+ channels that open in response to the influx of Ca2+ during the action potential carry much of the K+ current"**[1](https://en.wikipedia.org/wiki/Afterhyperpolarization)
* **"Both components of the AHP result from activation of calcium-activated potassium currents secondary to calcium influx via voltage-gated calcium channels"**[2](https://pmc.ncbi.nlm.nih.gov/articles/PMC6758860/)

**Why Your Current Model Works Biologically**

Your graph is actually **correct for the standard HH model**:

* **Squid axon** (original HH) doesn't show significant adaptation
* **Voltage-gated K+ channels** do oscillate as you see
* **AHP occurs** from slow deactivation kinetics, not persistent activation

**To Get the Behavior You Expected**

You would need to add **calcium-activated K+ channels** (as mentioned in your notes):

python

*# Add calcium dynamics*

soma.insert('ca\_ion')

*# Add calcium-activated K+ current (requires custom mechanism)*

soma.insert('SK') *# Small conductance Ca-activated K+ channels*

**Summary Table**

| **Channel Type** | **Activation** | **Deactivation** | **Your Graph Shows** | **Creates AHP via** |
| --- | --- | --- | --- | --- |
| **HH K+ (voltage-gated)** | Fast (with spike) | Slow (after spike) | ✅ Oscillating conductance | Slow deactivation |
| **Ca-activated K+** | Ca2+ accumulation | Very slow Ca2+ removal | ❌ Would show buildup | Persistent activation |

**Bottom Line**

Your graph is **biophysically correct** for the HH model. The AHP occurs through **slow K+ channel deactivation**, not persistent activation. To see the **cumulative K+ buildup** you expected, you'd need to add calcium-activated K+ channels, which is exactly what your notes suggest exploring!

The RHAG triggers one spike and that silences all the other neurons in that column so the spike frequency adaptation is pretty high/good…need to model dendritic arbour and inputs not just current

## 🧠 PYTHON CODE EXPLAINED

from neuron import h, gui

import numpy as np

import matplotlib.pyplot as plt

* **Imports**: You load NEURON (h is the hoc interface), NumPy for array handling, and Matplotlib for plotting.

### 📦 Model Setup

h.load\_file("stdrun.hoc")

* Loads NEURON standard simulation functions.

soma = h.Section(name='soma')

soma.L = 30

soma.diam = 30

soma.Ra = 100

soma.cm = 1

* Creates a single compartment called soma, with length/diameter = 30 µm.
* Ra = axial resistance; cm = membrane capacitance (µF/cm²).

### ⚡ Ion Channels: Hodgkin-Huxley

soma.insert('hh')

* Adds **standard Hodgkin-Huxley mechanism** to the soma (Na⁺ and K⁺ channels).

for seg in soma:

seg.hh.gnabar = 0.12

seg.hh.gkbar = 0.036

seg.hh.gl = 0.0003

seg.hh.el = -54.3

* Sets max Na⁺, K⁺, and leak conductances, and the leak reversal potential.

### 🔌 Current Injection

stim = h.IClamp(soma(0.5))

stim.delay = 100

stim.dur = 500

stim.amp = 0.3

* Injects a 0.3 nA current into the middle of the soma from 100 to 600 ms.

### 🌀 Adaptation Mechanism (POINT\_PROCESS)

adapt\_mech = h.Adapt(soma(0.5))

adapt\_mech.gmax = 0.001

adapt\_mech.tau = 200

* Adds your custom Adapt mechanism from your .mod file.
* gmax: Max conductance when spiked.
* tau: Controls how slowly the adaptation current decays.

### 🧠 Spike Detection

spike\_times = h.Vector()

nc = h.NetCon(soma(0.5).\_ref\_v, None, sec=soma)

nc.threshold = -20

nc.record(spike\_times)

* Detects when V\_m crosses -20 mV (spike threshold) and records the time.

adapt\_nc = h.NetCon(soma(0.5).\_ref\_v, adapt\_mech, sec=soma)

adapt\_nc.threshold = -20

adapt\_nc.weight[0] = 1

* When a spike occurs, this sends an event to Adapt, incrementing conductance g.

### 🧪 Recording Variables

t = h.Vector().record(h.\_ref\_t)

v = h.Vector().record(soma(0.5).\_ref\_v)

ina = h.Vector().record(soma(0.5).\_ref\_ina)

ik = h.Vector().record(soma(0.5).\_ref\_ik)

m = h.Vector().record(soma(0.5).hh.\_ref\_m)

h\_gate = h.Vector().record(soma(0.5).hh.\_ref\_h)

n = h.Vector().record(soma(0.5).hh.\_ref\_n)

iadapt = h.Vector().record(adapt\_mech.\_ref\_iadapt)

* Records time, voltage, Na⁺ and K⁺ currents, gating variables, and the **adaptation current**.

### ▶️ Run the Simulation

h.finitialize(-65)

h.continuerun(700)

* Initializes membrane at -65 mV and runs for 700 ms.

### 🧮 Calculate Conductance

gna = 0.12 \* (m\*\*3) \* h\_gate

gk = 0.036 \* (n\*\*4)

* Calculates instantaneous Na⁺ and K⁺ conductances from Hodgkin-Huxley equations.

### 📊 Plotting Results

Each subplot:

1. **Membrane Potential**: Spiking activity.
2. **Ion Currents**: Sodium and potassium currents.
3. **Conductances**: Shows how Na⁺ and K⁺ channels open and close.
4. **Input Current**: The stimulus applied.

Extra:

plt.figure()

plt.plot(t, iadapt, 'm')

* Plots adaptation current over time.

### 📈 Interspike Intervals (ISI)

isis = np.diff(spikes)

* Measures time between each pair of spikes.
* Plots how ISI increases due to adaptation (key result!).

## 🧬 MOD FILE EXPLAINED

This defines the custom **Adaptation mechanism**.

NEURON {

POINT\_PROCESS Adapt

NONSPECIFIC\_CURRENT iadapt

RANGE g, gmax, tau, e, iadapt

}

* Declares a **point process** (not distributed across the section).
* iadapt is a nonspecific current (like leak or IClamp).
* g: dynamic conductance, gmax: peak jump in g after spike, tau: how fast it decays, e: reversal potential.

PARAMETER {

gmax = 0.001 (S)

tau = 200 (ms)

e = -90 (mV)

}

* Defaults used in Python; you override them in your code.

ASSIGNED {

v (mV)

iadapt (mA)

}

* v comes from NEURON. iadapt is computed.

STATE {

g (S)

}

* Conductance variable that decays over time.

INITIAL {

g = 0

}

* Adaptation current is 0 at t=0.

### 🔄 Dynamics:

BREAKPOINT {

SOLVE state METHOD cnexp

iadapt = g \* (v - e)

}

* Calculates adaptation current based on current conductance and driving force.

DERIVATIVE state {

g' = -g / tau

}

* Conductance g decays exponentially.

NET\_RECEIVE(weight) {

g = g + weight \* gmax

}

* On each spike (event), g increases by weight \* gmax.

## ✅ What’s Really Happening?

1. Your HH neuron fires spikes due to IClamp.
2. Every time a spike occurs, Adapt receives a signal and **boosts an inhibitory current** (iadapt).
3. This **builds up inhibition** across spikes (like Ca²⁺-activated K⁺ would).
4. The result: **spike frequency slows down** over time (adaptation).

🧬 Is Adapt.mod Biophysically Accurate?

**Short answer:**  
It’s a **simplified, phenomenological model** of adaptation — not strictly biophysically accurate, but it **mimics the effect** of adaptation currents like **I\_KCa** (calcium-activated potassium current).

## ⚛️ What Biophysical Process Does It Mimic?

The Adapt mechanism represents a **generic spike-triggered potassium current**:

| **Real Current** | **What It Does** | **Modeled By Adapt?** |
| --- | --- | --- |
| **Ca²⁺-activated K⁺ (KCa)** | Outward K⁺ current activated by spike-induced Ca²⁺ influx | ✔️ Similar effect, but no calcium |
| **M-type K⁺ current** | Slowly activating voltage-gated K⁺ current | ❌ Not modeled (not voltage-dependent) |
| **AHP current** | Afterhyperpolarization current (also K⁺-based) | ✔️ Roughly mimicked |

So:

* Your Adapt.mod **doesn't simulate calcium entry**, but it **adds a K⁺-like current each time a spike occurs**, and this current **decays over time**, which leads to **slowing of spike frequency** (just like real adaptation).

## ⚠️ Limitations of Adapt.mod

| **Limitation** | **Explanation** |
| --- | --- |
| ❌ **No calcium dynamics** | Real KCa channels open in response to Ca²⁺ influx through voltage-gated Ca²⁺ channels — that’s not modeled here. |
| ❌ **No voltage gating** | The current isn’t activated by membrane potential like M-type K⁺ channels. |
| ✅ **Spike-triggered** | It mimics the effect of spikes causing outward currents. |

This model is often called a **phenomenological adaptation mechanism** — it's useful in:

* Reducing computational cost,
* Capturing high-level adaptation behavior (like firing rate slowing),
* Studying network dynamics without detailed channel biophysics.

## ✅ What About the Voltage-Activated K⁺ Channels?

These are already modeled by NEURON’s built-in **hh mechanism**:

seg.hh.gkbar = 0.036 # Max conductance for voltage-gated K+

* This part handles the **classic delayed-rectifier K⁺ channels**, which open during the spike and help repolarize the membrane.
* These are **voltage-gated** and **activate during each spike**.
* But they **don’t cause adaptation** — they reset the spike.

## 🧪 Want to Be More Biophysically Realistic?

If you're aiming for **biological realism**, consider:

### Option 1: **Use** cad.mod **+** kca.mod

* Add calcium entry (cad.mod — built-in in NEURON)
* Use calcium-activated K⁺ channel (kca.mod)
* Now adaptation depends on real **[Ca²⁺] dynamics** inside the cell

### Option 2: **Use an M-type K⁺ channel**

* Add a voltage-gated K⁺ channel with slow activation
* You can use published models like:
  + km.mod (in ModelDB)
  + Or add your own with alpha, beta gates

## ✅ Summary

| **Feature** | **Present in Adapt.mod** | **Present in Real Neurons** |
| --- | --- | --- |
| Spike-triggered adaptation | ✔️ | ✔️ |
| K⁺ outward current | ✔️ | ✔️ |
| Ca²⁺-dependence | ❌ | ✔️ in KCa |
| Voltage gating | ❌ | ✔️ in M-type and others |
| Biophysically realistic? | ❌ Simplified | ✔️ in detailed models |
| Computationally efficient? | ✔️ | ❌ More channels = slower |

Let me know if you'd like help:

* Converting this to a **Ca²⁺-based KCa model**
* Adding a **voltage-gated M-current**
* Comparing **biophysical vs. phenomenological** adaptation in side-by-side plots

You're on the right track — understanding the difference between effect and mechanism is key in computational neuroscience.

* 1. Mess with the adaptation, increase and decrease the numbers – just so that it fires a single spike - done
  2. In order to get just one spike we can increase the firing threshold
  3. Calcium current
  4. Change single current to multiple inputs – done…instead of sharp step changes make it like a soft maximum and have curves in the graph
  5. Is there a way to model the weights of synapses, just like it can model networks - yes
  6. We need to model the RHAG neuron, PAX6, non-PAX6 neuron…the RHAG triggers the PAX6 which then silences all the other neurons connected in the column…RHAG neuron fires just one spike which is enough to activate the PAX6 which is enough to silence others. – research done little bit
  7. Can we apply a function of time and/or other factors (eg. activity in the pre and post synaptic neurons) to change the weights of the synapses automatically
  8. Hebbs rule – read up

Whats the difference between gmax and tau in adapt.mod in terms of how they affect the adaptation and spike frequency, size, duration, ISI

Single spike at gmax = 0.04 and tau = 2000ms with the input current duration = 500ms

Single spike at gmax = 0.4 and tau = 200ms with the input current duration = 500ms

Figure out which one is biophysically possible, plot a graph of gmax and tau vs the increase in ISI and number of spikes

THE RHAG NEURON – protein that is expressed exclusively by this neuron and this triggers the PAX6 neuron

## 🧠 Context: Where are we in the brain?

We're in the **dorsolateral prefrontal cortex (DLPFC)** of the **monkey brain**, specifically in **layer 2/3** of the cortex.

* This region is involved in **decision making**, **working memory**, and **cognitive control**.
* It's like a control panel for planning and attention.

## 🧩 The Players Involved

Let’s start with the three main components:

| **Component** | **What it is** | **What it does** |
| --- | --- | --- |
| **RHAG neuron** | A special kind of **excitatory neuron** in L2/3 | Fires **just one spike**, then stops |
| **PAX6 neuron** | A special **inhibitory neuron** (interneuron) | Gets triggered by RHAG, and then sends **strong inhibition** to many other neurons |
| **Other neurons** | Excitatory and inhibitory neurons in the nearby column | These are the ones that get **silenced** |

## 🎬 Step-by-Step: What Happens in the Brain

### 1. **RHAG neuron is activated**

* This neuron has a **very high threshold** or **very strong adaptation**, so it doesn't fire easily.
* But **when it does**, it **only fires one spike** — a single "GO" signal.

📌 Think of it like pressing a **reset button**: it only needs one press.

### 2. **That one spike is sent to the PAX6 neuron**

* The connection from RHAG → PAX6 is **very strong** and **very reliable**.
* That means a single spike from RHAG is **enough to make the PAX6 neuron fire**.
* This is confirmed by experiments: almost all PAX6 neurons fired when RHAG did.

📌 It’s like RHAG rings a big bell and PAX6 wakes up immediately.

### 3. **PAX6 neuron fires and spreads inhibition**

* Once PAX6 is activated, it sends **inhibitory signals** to **many other neurons nearby**.
* It can:
  + Inhibit **excitatory neurons** in layers 2/3, 4, 5, and 6
  + Inhibit **other inhibitory neurons** too
* It does this **very quickly** and **very effectively**.

📌 It’s like PAX6 pulls the fire alarm in a building, and everyone else stops what they’re doing.

### 4. **The result: Silence**

* This whole process **shuts down** activity in a local “column” of the brain.
* So, **just one spike from RHAG leads to a powerful wave of inhibition**.

## 🎯 Why is this important?

This could be how the brain:

* **Stops noise** in a region so something else can take control – increase contrast amongst columns
* **Resets activity** for a new task or focus
* **Implements selective attention** — letting only one group of neurons be active at a time

In neuroscience, this is a **disynaptic inhibitory motif**:

* One excitatory neuron (RHAG) activates one inhibitory neuron (PAX6)
* That inhibitory neuron then **inhibits many others**

## 🔬 What Did the Researchers Find?

From the captions:

1. **RHAG is a special neuron only found in primates**
   * Not in mice
   * Highly selective and sparse
2. **RHAG to PAX6 connections are strong**
   * Fast EPSPs (excitatory post-synaptic potentials)
   * Nearly always make the PAX6 neuron spike
3. **PAX6 neurons project inhibition across layers**
   * To L2/3 pyramidal neurons
   * To L4 interneurons
   * Even to L5 and L6 neurons
4. **Morphology**
   * PAX6 neurons look like “horsetail” cells with **long descending axons**
   * Perfectly designed to spread inhibition across the cortical column

## 🧠 What Might Be Special About RHAG?

Based on what you and your professor said:

* **High threshold** or **strong adaptation** might explain why RHAG only fires **once**
* That’s likely **by design** — to send a **precise one-shot signal**
* You’re probably modeling that in NEURON using:
  + A high threshold
  + Or strong adaptation like you’ve done already

## 🛠️ How Could You Model This in NEURON?

You could:

1. Create a **RHAG neuron** with:
   * High threshold or strong adaptation
   * Fires once and stops
2. Connect it to a **PAX6 interneuron**
   * Receives a strong AMPA synapse
   * Spikes reliably
3. PAX6 neuron connects to:
   * Many other model neurons (excitatory/inhibitory)
   * Using inhibitory synapses (Exp2Syn, NetCon, etc.)

📌 The whole setup mimics a **selective inhibition control switch** — a useful pattern for cortical computation.

Yes — in NEURON, you can **absolutely model synaptic weights** and even entire **networks of neurons** using synapses and NetCon objects.

**🧠 How to Add Synaptic Inputs with Weights**

Instead of (or in addition to) IClamp, you can use:

* **ExpSyn** or **Exp2Syn**: models a chemical synapse (like AMPA/GABA)
* **NetStim**: generates artificial spike trains
* **NetCon**: connects a spike event to a synapse and assigns a **weight**

**✅ Minimal Example: A synapse with weight**

# Create a synapse on the soma

syn = h.ExpSyn(soma(0.5))

syn.tau = 2 # Decay time constant in ms

syn.e = 0 # Reversal potential (0 for AMPA, -80 for GABA)

# Artificial presynaptic spike generator

stim = h.NetStim()

stim.start = 100 # First spike at 100 ms

stim.number = 5 # Number of spikes

stim.interval = 50 # Time between spikes (ms)

stim.noise = 0 # 0 = deterministic

# NetCon connects the spike generator to the synapse

nc = h.NetCon(stim, syn)

nc.weight[0] = 0.005 # Synaptic weight (μS)

# Record synaptic current

isyn = h.Vector().record(syn.\_ref\_i)

Then plot isyn vs t just like your iadapt current.

**✅ What weight Represents**

* weight[0] is the **maximum conductance** (in μS) the synapse will apply when it receives a spike.
* The actual synaptic current is computed as:

Isyn=g(t)⋅(V−Erev)I\_{\text{syn}} = g(t) \cdot (V - E\_{\text{rev}})

where:

* + g(t)g(t) is time-dependent conductance based on the synapse model
  + ErevE\_{\text{rev}} is the reversal potential (e.g., 0 mV for AMPA)

**✅ Expanding to Multiple Inputs**

You can simulate **many presynaptic neurons** or **synapses with different weights**, e.g.:

# Two synapses with different weights and times

syn1 = h.ExpSyn(soma(0.5))

syn2 = h.ExpSyn(soma(0.5))

stim1 = h.NetStim()

stim1.start = 100

stim1.number = 1

stim1.interval = 1

stim2 = h.NetStim()

stim2.start = 200

stim2.number = 1

stim2.interval = 1

nc1 = h.NetCon(stim1, syn1)

nc1.weight[0] = 0.005

nc2 = h.NetCon(stim2, syn2)

nc2.weight[0] = 0.01

**✅ Summary: What You Can Model with Synaptic Weights**

| **Feature** | **How** |
| --- | --- |
| Spike-triggered current | ExpSyn + NetCon |
| Synaptic weights | nc.weight[0] = value |
| Multiple synapses | Create multiple ExpSyn, NetStim, NetCon |
| Plasticity (e.g. STDP) | Use stdp.mod or custom mod files |
| Full network models | Instantiate multiple Section() objects and connect them |

**Q. Whats the difference between gmax and tau in adapt.mod in terms of how they affect the adaptation and spike frequency, size, duration, ISI?**

Single spike at gmax = 0.04 and tau = 2000ms with the input current duration = 500ms

Single spike at gmax = 0.4 and tau = 200ms with the input current duration = 500ms

**How Do gmax and tau Differ in an Adaptation Mechanism?**

When modeling spike frequency adaptation in NEURON (using a custom or point-process adaptation mechanism), two key parameters—**gmax** (maximum adaptation conductance) and **tau** (decay time constant)—control how adaptation shapes the neuron's firing behavior. Here’s a clear breakdown of their roles and differences:

**What Does Each Parameter Control?**

**1. gmax: Maximum Adaptation Conductance**

* **What it is:** The maximum possible conductance that the adaptation current can reach after repeated spiking.
* **What it controls:** The *strength* of adaptation—how much each spike increases the adaptation current.
* **Effect of increasing gmax:**
  + Each spike produces a larger adaptation current.
  + The neuron’s firing rate slows down more quickly after each spike.
  + If set too high, the neuron may stop firing altogether during sustained input.
* **Biological analogy:** Corresponds to the total number or density of adaptation channels (e.g., slow K⁺ channels).

**2. tau: Adaptation Decay Time Constant**

* **What it is:** The time (in ms) over which the adaptation conductance decays back toward zero after being triggered by a spike.
* **What it controls:** The *duration* of adaptation—how long the adaptation effect from a spike persists.
* **Effect of increasing tau:**
  + Adaptation current decays more slowly.
  + Adaptation effects from multiple spikes accumulate, leading to stronger spike-frequency adaptation over time.
  + The firing rate continues to decrease for a longer period after the initial burst.
* **Biological analogy:** Reflects how quickly adaptation channels close or how fast intracellular signals (like Ca²⁺) are cleared.

**Why Are Both Parameters Needed?**

* **gmax** sets the *size* of the adaptation “kick” per spike.
* **tau** sets the *duration* of each adaptation effect.
* Together, they allow you to independently tune how much and how long adaptation influences the neuron’s firing.

**Effects on Neuronal Firing and Adaptation**

| **Parameter** | **If Increased...** | **Adaptation & Firing Consequence** |
| --- | --- | --- |
| gmax | Stronger adaptation per spike | Firing rate drops more after each spike; ISIs (interspike intervals) increase faster |
| tau | Slower decay of adaptation | Adaptation effect lasts longer; ISIs continue to lengthen over time (cumulative effect) |

**Example Scenarios**

* **High gmax, Low tau:** Strong but brief adaptation after each spike; neuron recovers quickly.
* **Low gmax, High tau:** Weak but persistent adaptation; neuron’s firing rate slows gradually over many spikes.
* **Both High:** Strong, long-lasting adaptation; neuron may nearly stop firing during prolonged input.

**How They Affect Model Outputs**

* **Adaptation and ISI:** Both parameters increase adaptation, but in different ways:
  + **gmax** mainly affects the *immediate* post-spike slowing.
  + **tau** mainly affects how much adaptation *builds up* over time, influencing the overall spike train pattern.
* **Spike Frequency:** With higher gmax or tau, the neuron’s spike frequency adaptation is more pronounced—spikes become less frequent as the stimulus continues.
* **Spike Size & Duration:** These parameters typically do **not** affect the size or duration of individual spikes, but rather the *timing* between spikes (ISI) and the overall firing pattern.

**Visual Summary Table**

| **Parameter** | **Controls** | **Increase Value →** | **Decrease Value →** |
| --- | --- | --- | --- |
| gmax | Adaptation strength | More spike slowing | Less spike slowing |
| tau | Adaptation duration | Longer-lasting adaptation | Shorter adaptation effect |

**In Practice**

* **To make adaptation more obvious:** Increase both gmax and tau.
* **To make adaptation subtle:** Decrease one or both.
* **To study their effects:** Vary them independently and observe how the firing rate and ISIs change in your simulation.

**In summary:**

* **gmax** determines *how much* each spike slows the neuron (adaptation strength).
* **tau** determines *how long* that slowing effect lasts (adaptation duration).
* Both are needed to flexibly model the full range of spike frequency adaptation seen in real neurons.

**Q. Figure out which one is biophysically possible, plot a graph of gmax and tau vs the increase in ISI and number of spikes**

## Interpretation of Your Adaptation Parameter Heatmaps

## 1. **ISI Increase vs. gmax and tau**

* **What it shows:**  
  The increase in interspike interval (ISI) from the first to the last spike, as a function of adaptation strength (gmax) and adaptation time constant (tau).
* **How to read it:**
  + **Brighter/yellow regions:** Stronger adaptation—spikes slow down more over time.
  + **Darker/blue regions:** Little or no adaptation—spike intervals stay nearly constant.
* **Key trends:**
  + Increasing either gmax or tau leads to a larger ISI increase.
  + When both are high, adaptation is strongest and the neuron slows down the most.

## 2. **Number of Spikes vs. gmax and tau**

* **What it shows:**  
  The total number of spikes generated during the stimulus for each combination of gmax and tau.
* **How to read it:**
  + **Brighter/yellow regions:** More spikes—adaptation is weak or absent.
  + **Darker/blue regions:** Fewer spikes—adaptation is strong, possibly silencing the neuron.
* **Key trends:**
  + As gmax or tau increase, the number of spikes decreases.
  + At high values, adaptation can be so strong that the neuron fires only a few times or stops altogether.

## 3. **Biophysical Interpretation**

* **Low gmax/tau:**  
  Weak adaptation—neuron fires at a steady rate.
* **Moderate gmax/tau:**  
  Realistic adaptation—spikes slow down but neuron continues to fire throughout the stimulus.
* **High gmax/tau:**  
  Strong adaptation—spiking slows dramatically or ceases, which may be non-physiological.

## 4. **Summary Table**

| **Parameter Change** | **ISI Increase** | **Number of Spikes** | **Interpretation** |
| --- | --- | --- | --- |
| Low gmax/tau | Small | High | Weak/No adaptation |
| Moderate gmax/tau | Moderate | Moderate | Realistic adaptation |
| High gmax/tau | Large | Low | Strong, possibly excessive adaptation |

## 5. **How to Choose Values**

* **Aim for the middle ground:**  
  Choose gmax and tau values that give a moderate ISI increase and a reasonable number of spikes—this best matches physiological adaptation seen in real neurons.
* **Avoid extremes:**  
  Too low: no adaptation.  
  Too high: neuron may stop firing.

**Q. Do we need both the parameters to mimic an actual neuron then?**  
Yes, **both parameters are needed to realistically mimic spike-frequency adaptation in an LIF (Leaky Integrate-and-Fire) neuron**.

**Why?**

* **Adaptation strength** (how much each spike slows the neuron) is set by a parameter like gmax or an adaptation increment (b or β in adaptive LIF models)[1](https://www.nature.com/articles/s44172-024-00165-9)[7](https://arxiv.org/html/2408.07517v1)[9](https://direct.mit.edu/neco/article-abstract/15/11/2523/6786/A-Universal-Model-for-Spike-Frequency-Adaptation?redirectedFrom=fulltext).
* **Adaptation duration** (how long the adaptation effect lasts) is set by the adaptation time constant (tau or τadapt)[1](https://www.nature.com/articles/s44172-024-00165-9)[7](https://arxiv.org/html/2408.07517v1)[9](https://direct.mit.edu/neco/article-abstract/15/11/2523/6786/A-Universal-Model-for-Spike-Frequency-Adaptation?redirectedFrom=fulltext).

**Biological neurons** show both:

* Each spike triggers a change that makes further spiking harder (strength).
* This effect then fades over time (duration).

**In adaptive LIF models:**

* The adaptation variable (e.g., an extra current or a dynamic threshold) is increased by a fixed amount at each spike (controlled by gmax or b), and then decays exponentially back to baseline (controlled by tau)[1](https://www.nature.com/articles/s44172-024-00165-9)[7](https://arxiv.org/html/2408.07517v1)[8](https://compneuro.uwaterloo.ca/files/syde-750/syde%20758.lecture4.pdf)[9](https://direct.mit.edu/neco/article-abstract/15/11/2523/6786/A-Universal-Model-for-Spike-Frequency-Adaptation?redirectedFrom=fulltext).
* This dual-parameter structure is essential for matching both the amount and time course of adaptation seen in real neurons[3](https://pmc.ncbi.nlm.nih.gov/articles/PMC1995006/)[5](https://elifesciences.org/articles/65459)[8](https://compneuro.uwaterloo.ca/files/syde-750/syde%20758.lecture4.pdf)[9](https://direct.mit.edu/neco/article-abstract/15/11/2523/6786/A-Universal-Model-for-Spike-Frequency-Adaptation?redirectedFrom=fulltext).

**If you only had one parameter:**

* You could make adaptation strong, but not control how long it lasts (or vice versa), which would not match actual neuronal firing patterns[1](https://www.nature.com/articles/s44172-024-00165-9)[8](https://compneuro.uwaterloo.ca/files/syde-750/syde%20758.lecture4.pdf).

**Summary:**

* Both adaptation strength and time constant are required for a realistic, biophysically plausible LIF neuron with spike-frequency adaptation[1](https://www.nature.com/articles/s44172-024-00165-9)[7](https://arxiv.org/html/2408.07517v1)[8](https://compneuro.uwaterloo.ca/files/syde-750/syde%20758.lecture4.pdf)[9](https://direct.mit.edu/neco/article-abstract/15/11/2523/6786/A-Universal-Model-for-Spike-Frequency-Adaptation?redirectedFrom=fulltext).

**Q. Plotting weight synapses for the current model**

## Plotting Synaptic Weights in Your Model

## What Are Synaptic Weights in This Context?

In your current NEURON model, the only "weight" parameter directly accessible is the weight of the NetCon that triggers the adaptation mechanism:

python

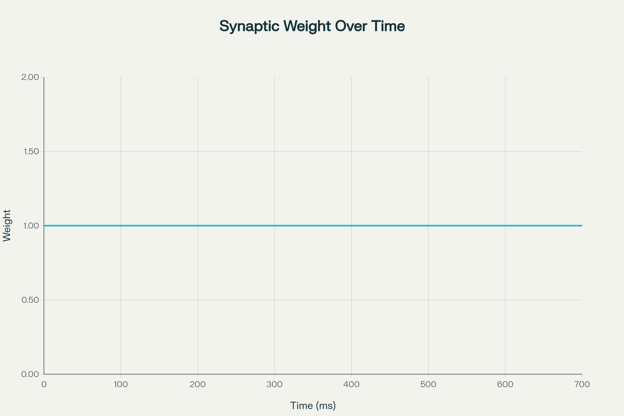
adapt\_nc.weight[0] = 1 *# Strength of adaptation trigger per spike*

This weight determines how much the adaptation mechanism is activated by each spike. In your code, it is set to a constant value of 1 throughout the simulation and does not change dynamically.

If you had multiple synapses (e.g., with ExpSyn, Exp2Syn, or plasticity mechanisms), each synapse would have its own weight, and you could record and plot their evolution over time. In your current setup, the "synaptic weight" is fixed.

## Visualization: Synaptic Weight Over Time

The plot below shows the synaptic weight of the adaptation mechanism (NetCon weight) as a function of time. Since the weight is constant (1.0), the plot is a flat line.



Synaptic Weight of Adaptation Mechanism Over Time

## Interpreting the Plot

* **Constant Weight:**  
  The synaptic weight does not change during the simulation. This is expected because there is no synaptic plasticity or weight update rule implemented in your current model.
* **If You Want Dynamic Weights:**  
  To see weights that change over time (e.g., with STDP, homeostatic plasticity, or short-term dynamics), you would need to add a plastic synapse mechanism and record its .weight variable at each time step.

## How to Plot Synaptic Weights for Other Synapses

If you add synapses like:

python

syn = h.ExpSyn(soma(0.5))

nc = h.NetCon(source, syn)

nc.weight[0] = initial\_weight

You can record and plot the weight over time if the mechanism supports dynamic updates:

python

weight\_vec = h.Vector().record(nc.\_ref\_weight[0])

Then plot weight\_vec against time.

## Summary

* In your current model, the only "synaptic weight" is the adaptation NetCon weight, which is constant.
* The plot reflects this: a flat line at 1.0.
* To visualize changing synaptic weights, you would need to implement and record from a plastic synapse mechanism.

**Q. Whats a good way to distinguish between strength of spike freq adaptation and duration of adaptation? What’s a measure of duration of spike frequency adaptation?**

Patch seq: Basically going inside the neuron cell and plotting the RNA sequencing and morphological analysis

IT(intra) neurons are sub types of pyramidal neurons that project within the Telencephalon (within the cortex)

ET(extra) projects from the cortex to the subcortical structures like structures in the brain stem

CT(cortico-thalamic) projects from the cortex to the thalamus

PAX6 is the only inhibitory neuron that is not localized to one layer…it runs through all 6(figure 3d)

RHAG is a pyramidal neuron that only fires a single spike unlike all the other 4 kinds of pyramidal neurons that fire multiple spikes (figure S2b vs all others)

PAX6 along with ChC interneurons appear to have the least amount of spike freq adaptation (figure 3d and e)

The axon total length and the max distance of the axonal tip to the soma is the largest in PAX6 interneurons compared to the others (figure s3 b)

Pyramidal Neurons = projection neurons (generally excitatory)

Interneurons = local neurons that project over local distances (generally inhibitory)

The amplitude and latency of disynaptic inhibition is three to four times larger than monosynaptic (i.e. slower and bigger than mono) (figure 4c, d)

CNQX – blocks the excitatory projection from the pyramidal neuron(RHAG – as shown in fig 4g…top right corner) to the inhibitory interneuron (PAX6 – as shown in fig 4i). So the inhibitory interneuron doesn’t get activated so it doesn’t inhibit anybody else. (figure 4e)

Fig 4q – RHAG is the primary projection to PAX6 inhib interneuron

A "detonator synapse" refers to a specific type of synapse, particularly prominent in the hippocampus and other brain regions, where a single presynaptic action potential reliably triggers an action potential in the postsynaptic neuron. This is in contrast to many other synapses where multiple inputs need to be summed together (temporal or spatial summation) to reach the threshold for firing.

10 – 15% of the RHAG cells express markers for detonator synapses and the ones that do, show very high level of expression (fig 5c)

Q. Whether it is an RHAG cell projecting on the PAX6 or from pyramidal cells to RHAG that is a detonator synapse?

Excitatory synapse…detonator synapse from RHAG to PAX6…we don’t know what projects to RHAG (long range projection on RHAG)

Q. The angle of the dendritic tree is not symmetric in RHAG, but why is there not a figure with stats and details. Is the size of the dendritic tree larger/smaller than non-RHAG pyramidal neurons?

20/22 degrees relative to the pier for the RHAG…very superficial…the apical dendrites are gonna be short

Q. What do the non-RHAG cells connect to?

Connect to the other excitatory neurons…they project to L5 and also to local interneurons (not known yet) [non-RHAG has a lot more connections, which is why their Epsc is larger than RHAG]

Q. How are the RHAG neurons physiologically excited?

Not known yet

Q. Does the PAX6 neuron fire just one spike after it receives an input from RHAG (which is also one spike)

One RHAG projects to 4-5 PAX6 neurons [depends on the location of the RHAG, the number might change] (fig 7c – 4-5 PAX6 are within the vicinity of the RHAG neuron)

RHAG cells don’t get local connections from other pyramidal cells whereas other pyramidal cells do (fig 5 f-i) – which might suggest indirectly that RHAG cells receive long range connections

Comparing fig S5 with fig 5d & e would help us know which synaptic proteins are up regulated and which ones are down regulated and what their functions are

Disynaptic inhibition is unique to non-human primates and humans and even in these species, we only see disynaptic inhibition in higher order cortex (PFC and TC) and not sensory cortex (V1 and S1)

Mistakes:  
Violin plot is figure 3b and not figure 3c

They don’t show PAX6 in figure 3f dot plot visualization

Figure s3c, the Ephys random forest has quite a few additional labels compared to the morphology and combined random forest confusion matrices.

Figure 4g – RHAG and FRMPD2 are labelled with the same color in the graph

Fig 4h – There seems to be only two neurons but the captions suggest that there should be 3 (2 RHAG + 1 interneuron)

Detonator synapse (maybe achieved in neuron) – by having a high gain, synaptic weight from the RHAG to PAX6

“In essence, detonator synapses are characterized by their ability to reliably induce an action potential (spike) in the postsynaptic neuron, often by releasing a large amount of neurotransmitter following a presynaptic action potential. The underlying biophysical mechanisms include:

* Presynaptic Calcium Dynamics:
  + Voltage-Gated Calcium Channels (VGCCs): Detonator synapses, like the hippocampal mossy fiber synapse, exhibit a high density of specific types of VGCCs, particularly P/Q-type and N-type channels.
  + Differential Calcium Signaling: P/Q-type VGCCs are crucial for recruiting additional release sites, contributing to the large amount of neurotransmitter released at these synapses. N-type VGCCs, on the other hand, play a role in synchronizing multivesicular release, impacting the timing and efficiency of transmission.
  + Calcium Concentration and Release Probability: The influx of calcium through these channels triggers the fusion of synaptic vesicles (containing neurotransmitters) with the presynaptic membrane, a process highly sensitive to calcium concentration and the probability of vesicle release (Pr).
* Efficient Neurotransmitter Release Machinery:
  + Active Zones (AZs): Detonator synapses often have numerous active zones, specialized regions on the presynaptic terminal where vesicles are docked and primed for release.
  + Synaptic Vesicle Cycling: These synapses have efficient mechanisms for vesicle mobilization, docking, and fusion, including the involvement of proteins like RIM, RIM-BP, ELKS, Munc13, Munc18, and SNAREs.
  + SNARE Complex Assembly: The formation of SNARE complexes drives vesicle fusion. The regulation of proteins like syntaxin-1, which can exist in a "closed" or "open" conformation, plays a role in controlling the probability and speed of release, [according to National Institutes of Health (NIH)](https://urldefense.com/v3/__https:/pmc.ncbi.nlm.nih.gov/articles/PMC9490555/__;!!LkSTlj0I!Fo_jYQ-ll1Z_fozDxrzV1gEMVXH-a-GejJpXgmq_CZNbR7WVluX8V54doOBu0FXbD5rLIwh3JNSjuysmJpFBTslkVn-_Hw$).
* Short-Term Plasticity:
  + Short-term facilitation and Post-Tetanic Potentiation (PTP): Mossy fiber synapses, for instance, exhibit prominent short-term facilitation and a large PTP. This plasticity can convert them from "subdetonators" (requiring bursts to trigger postsynaptic firing) to "full detonators" (capable of reliably firing a postsynaptic action potential from a single presynaptic spike), [according to eLife](https://urldefense.com/v3/__https:/elifesciences.org/articles/17977__;!!LkSTlj0I!Fo_jYQ-ll1Z_fozDxrzV1gEMVXH-a-GejJpXgmq_CZNbR7WVluX8V54doOBu0FXbD5rLIwh3JNSjuysmJpFBTsmtpFix5g$). This ability to switch modes is crucial for information processing and memory formation.
* Pannexin1 Channel Activation: Recent research suggests that pannexin1 channels can be synaptically activated in some detonator synapses, like hippocampal mossy fiber synapses. NMDAR-mediated responses at these synapses can lead to a slow postsynaptic inward current, activating postsynaptic neurons. This suggests that pannexin1 activation might contribute to the detonator properties.

Essentially, the combination of efficient calcium influx, robust release machinery, and mechanisms for short-term plasticity like PTP allows detonator synapses to reliably and strongly transmit signals to their postsynaptic targets.

”

Assign weights to the input currents – make sure also to include negative values for inhibition

Make it variable weight

Show one spike

Try to have a detonator synapse modelled

For the non-detonator synapse – have at least 3 inputs (with its own weight) which when summed up then causes a spike (1/3 gain)

Read up on the diff between post and pre synapse behaviours

The synaptic currents need to be more gradual in nature…right now they are really sharp…try to make it similar to the IClamp input currents

Q. Is the action potential threshold for firing for the RHAG neuron any different than other pyramidal neurons

Q. Did they have a chance to measure the passive conductance properties

Q. What are the time constants for the RHAG and PAX6 and any other neurons

Verify the differences between he pre and post synaptics graphs and currents

1 RHAG to 4-5 PAX6 with detonator synapse…so one RHAG spike causes 1 PAX6 spike

And then all 5 PAX6 neurons should project to same/different neurons in different layers in the same column

1 RHAG -> 5 PAX6 -> 8 excitatory neurons and 4 inhibitory (look at the total number of spikes up and down the column)

The firing threshold for the RHAG neuron is -20mV…so when the weight is lowered the synaptic input weight reduces exponentially and it receives very weak input current not enough to exceed the threshold…so there is an initial depolarization that is visible in the graph but no spikes are fired in the RHAG neuron and hence none in the PAX6

What is the efficiency of the PAX6 to the non-RHAG excitatory neuron in silencing all of them?

Is the answer already there in the figures

1 RHAG to 4 PAX6 neurons means 4 synapses happening

Assumptions:

* 1. 100% efficiency in silencing
  2. 1 RHAG to multiple PAX6 neurons

FIG 4Q AND FIG 6K – 91-92% of the RHAG to PAX6 synapses are detonator synapses

Q. Whether it is an RHAG cell projecting on the PAX6 or from pyramidal cells to RHAG that is a detonator synapse?

Q. The angle of the dendritic tree is not symmetric in RHAG, but why is there not a figure with stats and details. Is the size of the dendritic tree larger/smaller than non-RHAG pyramidal neurons?

Q. What do the non-RHAG cells connect to?

Q. How are the RHAG neurons physiologically excited?

Q. Does the PAX6 neuron fire just one spike after it receives an input from RHAG (which is also one spike)

Q1. How does a non-RHAG cell behave? Does it fire multiple spikes (do we have the data on the

spike frequency)?

A: Yes. Non-RHAG neurons are capable of firing multiple spikes [CAN BE A PLAUSIBLE DISTINGUISHABLE POINT – reduced spike freq adaptation]. In our dataset, the spike

frequency (mean ± SD) was 10.78 ± 3.76 for non-RHAG neurons, compared with 1.32 ± 0.47 for

RHAG neurons (we can model this is as a Gaussian random prob distribution with the time constants).

Q2. Figs S4 and 4c show that the IPSC of disynaptic inhibition is 2.5-3 times greater than that of

monosynaptic inhibition. Do you have exact numbers on that that you can pass along to us?

A: Yes. The uIPSC amplitude for monosynaptic inhibition was 0.3847 ± 0.0739 mV, whereas for

disynaptic inhibition it was 1.1803 ± 0.3478 mV.

Q3. What are the relative ratios of RHAG cells to NRHAG cells and PAX6 cells to NPAX6 cells

in your recordings? So for every RHAG neuron, how many NRHAG neurons do you end up

recording? And for every PAX6 neuron, how many non-PAX6 neurons?

A: The proportions were: RHAG neurons 14.56% and non-RHAG neurons 85.44%; PAX6

neurons 17.18% and non-PAX6 neurons 82.82%.

Q4. Can a PAX6 neuron get input from more than one RHAG cells at the same time?

A: Yes. A PAX6 neuron can receive inputs from more than one RHAG neuron simultaneously.

Q5. “Disynaptic inhibition was detected in the monkey dorsolateral prefrontal cortex (DLPFC)

(15/86), monkey TC (14/80), and human TC (15/90)” – Fig 6c caption says this. Does this mean

that the remaining cells where disynaptic inhibition was not found are all non RHAG cells?

A: Yes, your understanding is correct — in these datasets, all remaining cells without detected

disynaptic inhibition were non-RHAG cells

1 – (32/37)^5.7 = 56.3% (no connection bw NRHAG and PAX6)

43.7(at least one connection bw NRHAG and PAX6)

46.7 (at least one connection bw RHAG and PAX6)

Q. What’s the connection probability for NRHAG to NPAX6 and bw RHAG to NPAX6? (will help in accurate modelling the neurons)

Q. What’s the firing rate PAX6 when we get input from NRHAG neuron? (because the connection probability between NRHAG to PAX6 is 43.7% compared to 46.7% bw RHAG and PAX6)

Q. What’s the amplitude of the IPSC (for the disynaptic inhibition) and **transmission prob** (prob. That the PAX6 neuron cell fires a spike once a connection is established and the NRHAG cell fires a spike) of the PAX6 neuron when its gets input from NRHAG cells? Does it even fire a spike (maybe the threshold is not reached)?

Potentially it means that there might be another function for the PAX6 neuron that is not going through the usual disynaptic network (followed from the RHAG neurons) and that it might just silence the other neurons when there is a lot of activity

Time constant for NRHAG is 90 ms…this is what is req for the spikes to accumulate in order to trigger the PAX6 neuron…but PAX6 has a relatively shorter time constant (22.86 ms)

* + Make sure the proportions are correct for NRHAG:RHAG and PAX6:NPAX6
  + We also need to see what happens when RHAG and NRHAG neurons fire simulataneously

Make sure the IPSC of disynaptic inhibition is 3 - 3.5 times larger than monosynaptic inhibition

Synaptic delay, find out the amplitude of the monosynaptic (faster (lower synaptic delay), lower amp by a factor of 3 - 3.5) vs disynaptic (larger) latencies (might have the same synaptic delay for all of them)

Disynaptic vs monosynaptic inhibition and also incorporate the cells connected to the PAX6 neurons

Whenever any PAX6 neuron fires a spike then the inhibition should be strong enough to silence the EI neurons to create a flat line (don’t remove their connection as a whole)

Make sure the latencies are in line for disynaptic (slower) than mono (faster)…amplitude should also be 3.5 time larger

Change the synaptic strength of PAX6 to EI : NPAX6 to EI to be 3.5:1 (but make sure the IPSPs are aligned with the actual experimental values)

With inhibition only the freq of spiking should go down and the action potential threshold should increase(ampl. of action potential shouldn’t be affected)…the membrane potential should go up and now if initially 20 mV was required, now it should need 30 mV (this is just an eg)…the amplitude of spikes shouldn’t be affected ever

Plot the level of inhibition and the disynaptic inhibition should be slower (should be a synaptic delay) – Gaussian prob distribution

This is my explanation of my current code:

# 1. **Imports and initialization**

import numpy as np

import matplotlib.pyplot as plt

from neuron import h, gui

v = h.VecStim()

h.load\_file("stdrun.hoc")

* **numpy**: for numerical ops (arrays, random choice, etc).
* **matplotlib**: for plotting voltage traces, currents, rasters.
* **neuron.h, gui**: NEURON simulation engine.
* h.VecStim(): creates an artificial spiking source that can replay a list of spike times.
* h.load\_file("stdrun.hoc"): loads standard run library (defines h.finitialize, h.continuerun etc).

# 2. **Neuron class definitions**

### HHNeuron

This is your base neuron:

* Creates a soma compartment (h.Section).
* Sets length, diameter, axial resistance (Ra), capacitance (cm).
* Inserts Hodgkin–Huxley (hh) conductances. Each segment gets:
  + gnabar (Na conductance),
  + gkbar (K conductance),
  + gl (leak conductance),
  + el (leak reversal).
* Adds an **adaptation current** mechanism (h.Adapt, custom mod file).
* Records vectors for:
  + Membrane potential v
  + Sodium current ina
  + Potassium current ik
  + Adaptation current iadapt
  + Gating variables m, h, n
* Spike detection:
  + self.nc = h.NetCon(self.soma(0.5).\_ref\_v, None) → monitors voltage.
  + Threshold = -20 mV.
  + Records spike times into self.spikes.
* Adaptation synapse:
  + self.nc\_adapt delivers spikes to self.adapt.
* connect\_to() helper: creates a new NetCon from this neuron’s soma to any synapse.

👉 This is where **action potentials are generated**: Hodgkin–Huxley channels integrate incoming currents (from synapses or NetStims). When Vm crosses threshold, NetCon fires.

### DetonatorInput

This is your **synaptic stimulator** class:

* Creates an ExpSyn synapse on a target section.
* Sets reversal potential e and time constant tau.
* Creates a NetStim (an artificial spike generator):
  + start (first spike time),
  + number (how many spikes),
  + interval (ms between spikes).
* Connects NetStim to synapse with a NetCon.
* Records synaptic current (i).

👉 This is where **external inputs come in** (like “injecting” spikes into your model).

### Derived neurons

* **RHAGNeuron**: inherits HHNeuron, automatically gets a DetonatorInput starting at 100 ms.
* **PAX6Neuron**: inherits HHNeuron, adds an ExpSyn so it can receive inputs, records its currents.
* **NRHAGNeuron**: like RHAG, but different tau.
* **ExcNeuron / InhNeuron**: same as HHNeuron, just placeholders for excitatory/inhibitory populations.

# 3. **Network construction**

num\_rhag = 4

num\_pax6 = 8

num\_nrhag = 23

num\_ex = 8

num\_in = 8

num\_ei\_mono = 4

num\_nonpax6\_inh = 4

* Defines **population sizes**.
* w\_mono\_inhib, w\_disyn\_inhib are weights for inhibitory connections.

Populations are created:

rhag\_cells = [RHAGNeuron() for \_ in range(num\_rhag)]

nrhag\_cells = [NRHAGNeuron(i) for i in range(num\_nrhag)]

pax6\_cells = [PAX6Neuron(idx) for idx in range(num\_pax6)]

exc\_neurons = [ExcNeuron(i) for i in range(num\_ex)]

inh\_neurons = [InhNeuron(i) for i in range(num\_in)]

ei\_mono\_neurons = [HHNeuron(name=f'ei\_mono\_{i+1}') for i in range(num\_ei\_mono)]

nonpax6\_inh\_neurons = [HHNeuron(name=f'nonpax6\_inh\_{i+1}') for i in range(num\_nonpax6\_inh)]

### Adding inputs

for neuron in nonpax6\_inh\_neurons:

neuron.input = DetonatorInput(...)

neuron.input.stim.number = 10

neuron.input.stim.interval = 30

→ ensures NPAX6 inhibitory neurons actually **spike repeatedly**.  
Same thing for ei\_mono\_neurons.

### RNG & connectivity

rng = np.random.default\_rng()

nearest\_pax6 = [rng.choice(num\_pax6, size=4, replace=False) for \_ in range(num\_rhag)]

nearest\_pax6\_nrhag = [rng.choice(num\_pax6, size=4, replace=False) for \_ in range(num\_nrhag)]

* Each RHAG and NRHAG chooses 4 random PAX6 targets.

Connection probabilities:

p\_proj = 20/43

p\_transmit = 0.91

p\_proj\_nrhag = 5/37

p\_transmit\_nrhag = 0.2

* RHAG more reliable, NRHAG less reliable.

# 4. **Simulating RHAG and NRHAG to get spikes**

Each population is simulated separately to gather spike times into vectors.  
Those spikes are then “relayed” through VecStim with probabilistic transmission.

👉 This is where **the VecStim relay mechanism comes in**:  
Instead of directly connecting presynaptic Vm → synapse, you replay spike times with dropouts (like synaptic failures).

# 5. **RHAG/NRHAG → PAX6 connections**

relay\_vec = h.Vector(relayed\_spikes)

relay\_stim = h.VecStim()

relay\_stim.play(relay\_vec)

nc = h.NetCon(relay\_stim, pax6\_cells[pax6\_idx].syn)

* A VecStim replays the spike times.
* NetCon delivers them to the PAX6 neuron’s synapse.
* Transmission failures modeled by only including some spikes in relay\_vec.

# 6. **PAX6 → EI disynaptic inhibition**

For each PAX6: choose 4 random EI targets.

* Create Exp2Syn on the target.
* Connect from PAX6 voltage to that synapse with inhibitory reversal potential (-65 mV).
* Save currents for later plotting.

👉 This is your **disynaptic inhibition pathway**.

# 7. **Assign EI inputs**

All EI neurons get a DetonatorInput so they spike unless silenced later.

# 8. **NPAX6 → EI monosynaptic inhibition**

for target in ei\_mono\_neurons:

for source in nonpax6\_inh\_neurons:

inh\_syn = h.Exp2Syn(target.soma(0.5))

nc = source.connect\_to(inh\_syn, weight=w\_mono\_inhib, delay=0.1)

* Each NPAX6 inhibitory neuron connects directly to each EI mono neuron.
* This is **monosynaptic inhibition**.
* Synaptic current is recorded.

# 9. **Simulation**

Two passes:

* Run initial simulation to see which PAX6 spiked.
* Based on that, mark EI targets for silencing.
* Strengthen PAX6→EI inhibitory weights if those PAX6 spiked.
* Run again for final results.

# 10. **Data extraction**

Convert NEURON Vectors into numpy arrays for plotting:

* Voltages, currents, spikes for each population.

# 11. **Plotting**

Generates:

* Membrane potential traces (RHAG, NRHAG, PAX6).
* Synaptic currents into PAX6.
* Adaptation currents.
* Exc/inhib neuron voltages.
* EI monosynaptic voltages.
* Raster plots.
* Connectivity matrices.

# 🔑 Key biological mapping

* **Action potentials**: generated by hh mechanism in each HHNeuron.
* **External inputs (detonator)**: DetonatorInput drives certain neurons (so they actually spike).
* **Synaptic inhibition**: Exp2Syn synapses with e=-65 mV provide inhibitory currents.
* **Monosynaptic inhibition**: NPAX6 → EI (direct).
* **Disynaptic inhibition**: PAX6 → EI (via interneurons).
* **Relay failures**: VecStim with probabilistic dropout simulates unreliable transmission.
* **Adaptation current**: Adapt mechanism, triggered by spikes, prevents runaway firing.

✅ If you present it like this, you can clearly explain **what happens at each step**:

1. Input neurons (RHAG, NRHAG, NPAX6) get driven by DetonatorInput.
2. They spike (hh channels).
3. Spikes are detected (NetCon) and relayed probabilistically (VecStim).
4. PAX6 neurons integrate excitatory inputs, spike, and inhibit EI neurons.
5. EI neurons either fire or get silenced depending on inhibition.
   * Check for inhibition by injecting current into the EI neurons through current clamps
   * It shouldn’t be nano amps, it should be of a bigger unit