Using NEURON to model cells Tutorial

Darshan Mandge
Darshan.mandge@epfl.ch
Blue Brain Project

Tutorial adapted from Oren Amsalem, Yoni Leibner (Idan Segev Lab) and András Ecker

Introduction

The **NEURON Simulation Environment** is designed for modeling individual neurons and networks of neurons.

It is particularly well-suited to explore problems which are closely linked to experimental data.

NEURON was built and is maintained by Ted Carnevale and Michael Hines at Yale. The Blue Brain Project is also heavily involved with the development of the software

Neuron is written in C & C++, the interface with the simulator is with HOC, but a Python wrapper was build and now commonly used instead of HOC (NEURON as a python package). In addition User-defined mechanisms such as voltage- and ligand-gated ion channels are used in order to expand NEURON (NMODL files, need to be compiled).

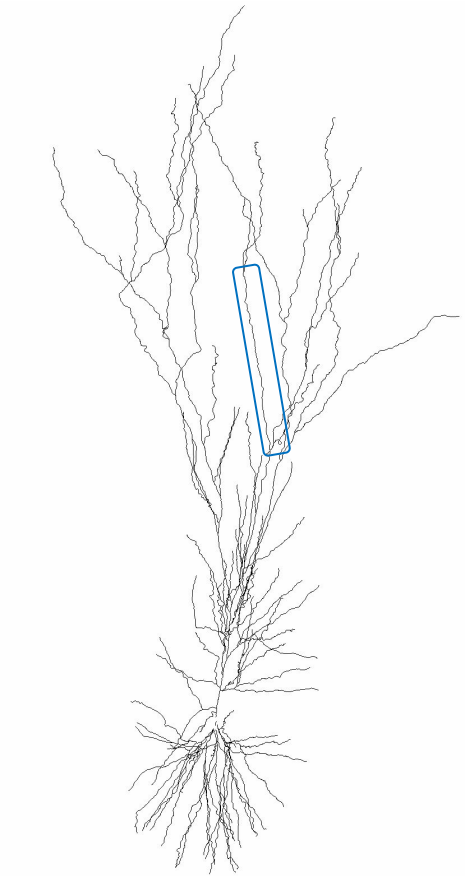
In this presentation we will show the following examples:

- Single compartment model
- Ball and Stick model and the replication of a published result (Gidon et al.)
- Multicompartmental L5 Pyramidal cell and two published results (Hay et al., Doron et al.)

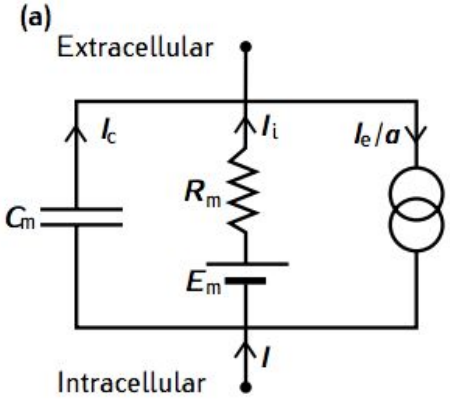
Introduction

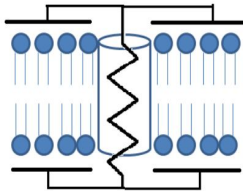
Basic concepts of Neuron models:

- **Model** - a Neuron object, it can have soma, basal dendrites (dend) and apical dendrite (apic).
- **Section** - more of a morphological definition, a section is usually a dendrite between 2 bifurcation points.
- **Segment** - one RC circuit that resemble (approximate) the computation in a piece of membrane.
- **Clamps** - an object that represent an experimental electrode.
- **Synapses** - object that resemble a single synapse (we will see it later, and how it's activated in the simulation).
- **Distributed Process:** generally ion channels spread on the membrane
- **Point Process** - a more general idea, it's an object that seats on the segment (RC circuit), and simulate clamps, synapses, etc.



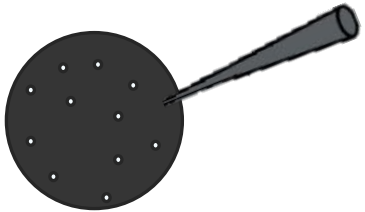
Single compartment neuron - single RC circuit





Neuron Cell Membrane
Equivalent Circuit

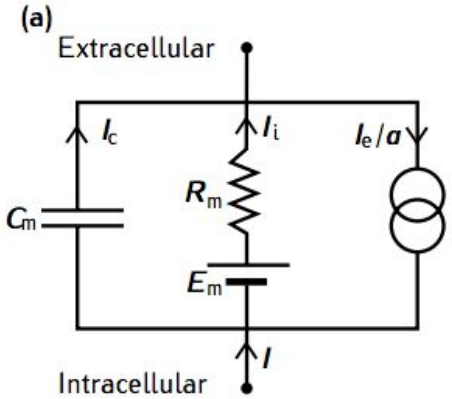
```
from neuron import h, gui
# create model
soma = h.Section(name="soma")
soma.L = 10  # length µm
soma.diam = 10  # diameter µm
soma.insert('pas'). # add passive properties
soma.g_pas = 1/10000  # set the specific membrane
                      resistance to 10000 ohm*cm^2
```



```
# current current clamp
stim = h.IClamp(soma(0.5))
stim.delay = 20. # start of the current
injection (ms)
stim.dur   = 100  # duration (ms)
stim.amp   = 0.01  # amplitude (nA)
```

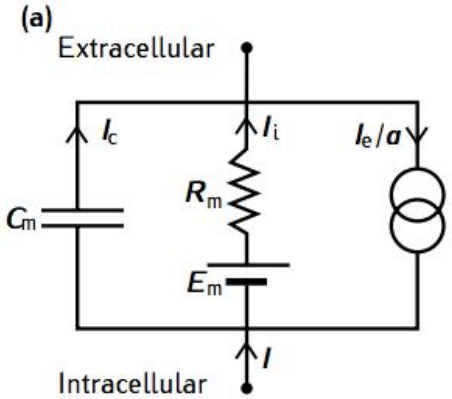
0_soma_passive.ipynb

Single compartment neuron - recording



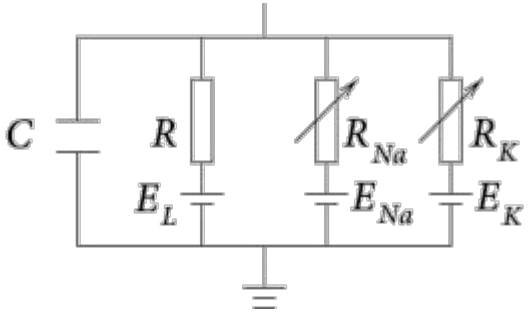

```
# record injected current soma voltage (and time)
soma_v = h.Vector()
soma_v.record(soma(0.5)._ref_v)
stim_current = h.Vector()
stim_current.record(stim._ref_i)
t = h.Vector()
t.record(h._ref_t)

# run simulation
h.tstop = 220  # simulation time (ms)
h.dt = 0.025.
h.v_init = -70. # initial voltage (mV)
h.run()
```

Single compartment neuron - synapse




```
# add a synapse
synapse = h.Exp2Syn(soma(0.5))
synapse.tau1 = 0.3  # rise time constant
synapse.tau2 = 1.8  # decay time constant
stim = h.NetStim()
stim.number = 1
stim.noise = 0  # no noise
stim.interval = 1
net_con = h.NetCon(stim, synapse)
net_con.weight[0]= 0.0004  # the maximal
conductance
                           of the synapse
```

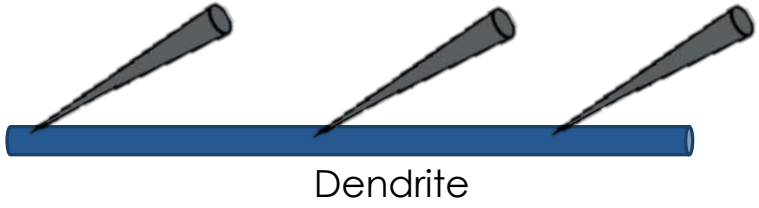
Single compartment neuron – active conductances

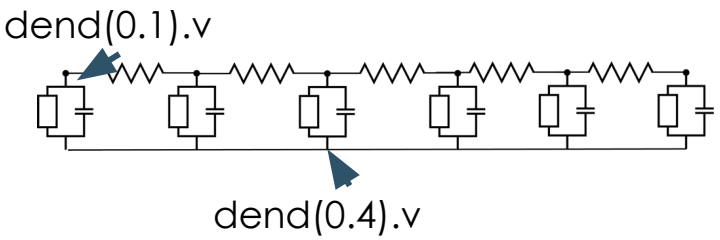


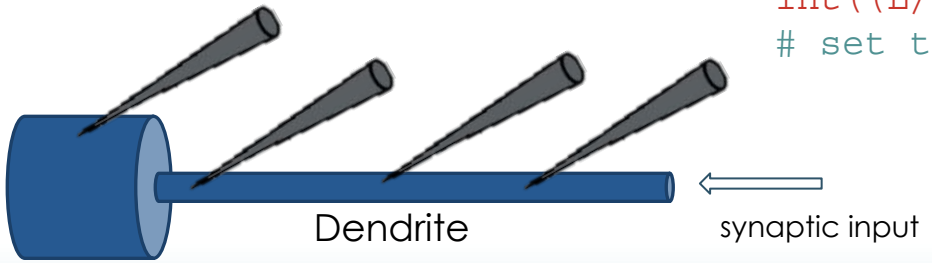

```
from neuron import h, gui
# create model
Soma = h.Section(name="soma")
soma.L = 10  # length µm
soma.diam = 10  # diameter µm
soma.insert("pas"). # add passive properties
soma.g_pas = 1/10000  # set the specific membrane
                      resistance to 10000 ohm*cm^2
soma.insert(”kv")  # add potassium channel
                   (from a mod file)
soma.gbar_kv = 2000 # set the potassium conductance
soma.insert(”na") # add sodium channel
                  (from a mod file)
soma.gbar_na = 8000 # set the sodium conductance
h.celsius = 30. # set temperature
```

1_soma_active.ipynb

Moving towards the cable model





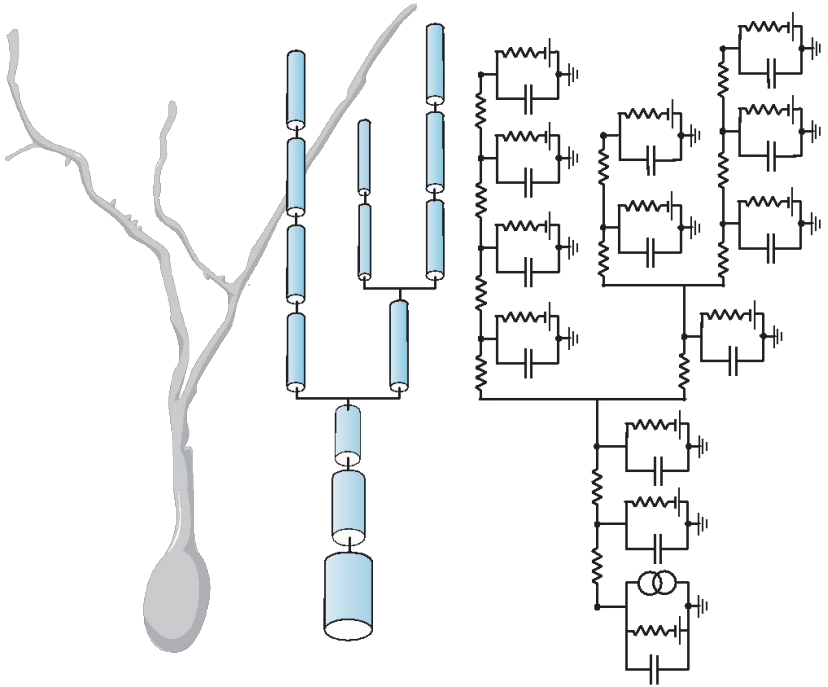


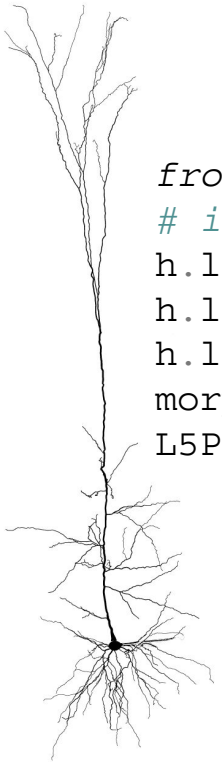

```
# create dendrite
dend = h.Section(name="dend")
dend.L = 500  # um
dend.diam = 1  # um
dend.Ra = 100  # ohm*cm
dend.insert("pas")
dend.g_pas = 1/10000
dend.connect(soma, 1, 0). # connect the end of
the dendrite to the beginning of the soma
h"forall { nseg =
int((L/(0.1*lambda_f(100))+0.9)/2)*2 + 1  }")
# set the number of segments
```

2_soma_dend_step_current.ipynb
3_soma_dend_synapse.ipynb

Note: d_lambda rule – NEURON Book chapter 4

Layer 5b Pyramidal Cell

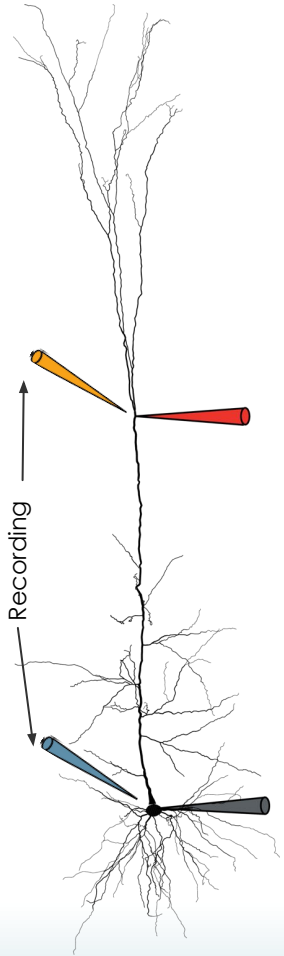


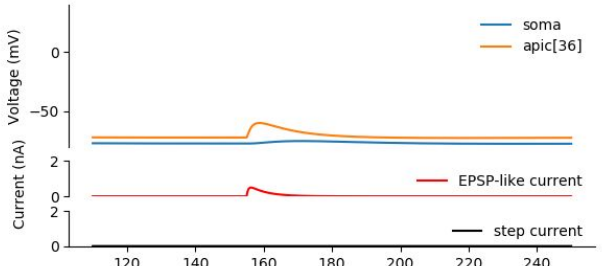


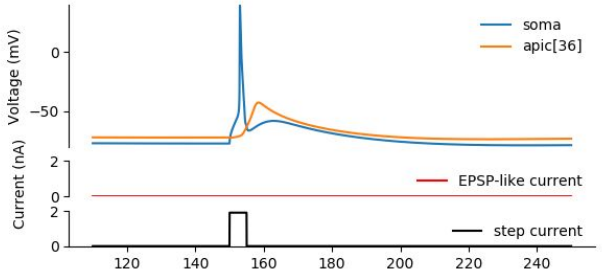
```
from neuron import h, gui
# instantiating cell
h.load_file("import3d.hoc")
h.load_file("models/L5PCbiophys3.hoc")
h.load_file("models/L5PCtemplate.hoc")
morph_fname = "morphologies/cell1.asc"
L5PC = h.L5PCtemplate(morph_fname)
```

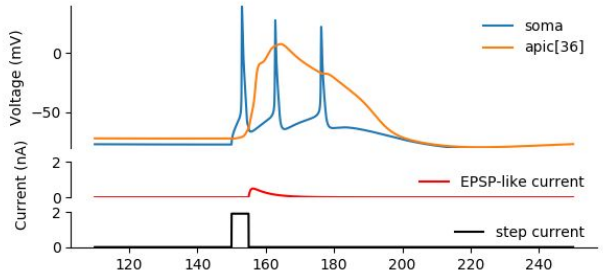
4_L5_PC_Hay_et_al.ipynb

"Models of Neocortical Layer 5b Pyramidal Cells Capturing a Wide Range of Dendritic and Perisomatic Active Properties", Etay Hay , Sean Hill, Felix Schürmann, Henry Markram, Idan Segev. PLOS Computational Biology, 2011










```
# Only EPSP-like current
    syn.imax = 0.5
    stim.amp = 0
    h.run();
    plot_result(...)
```

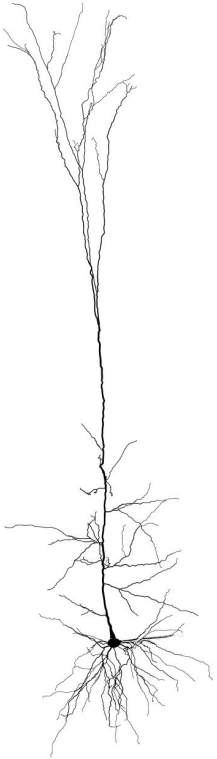
```
# Only small step current
    syn.imax = 0
    stim.amp = 1.9
    h.run();
    plot_result(...)
```

```
# both currents
    syn.imax = 0.5
    stim.amp = 1.9
    h.run()
    plot_result(...);
```

5_calcium_spike_Hay_et_al.ipynb

"Models of Neocortical Layer 5b Pyramidal Cells Capturing a Wide Range of Dendritic and Perisomatic Active Properties", Etay Hay , Sean Hill, Felix Schürmann, Henry Markram, Idan Segev. PLOS Computational Biology, 2011

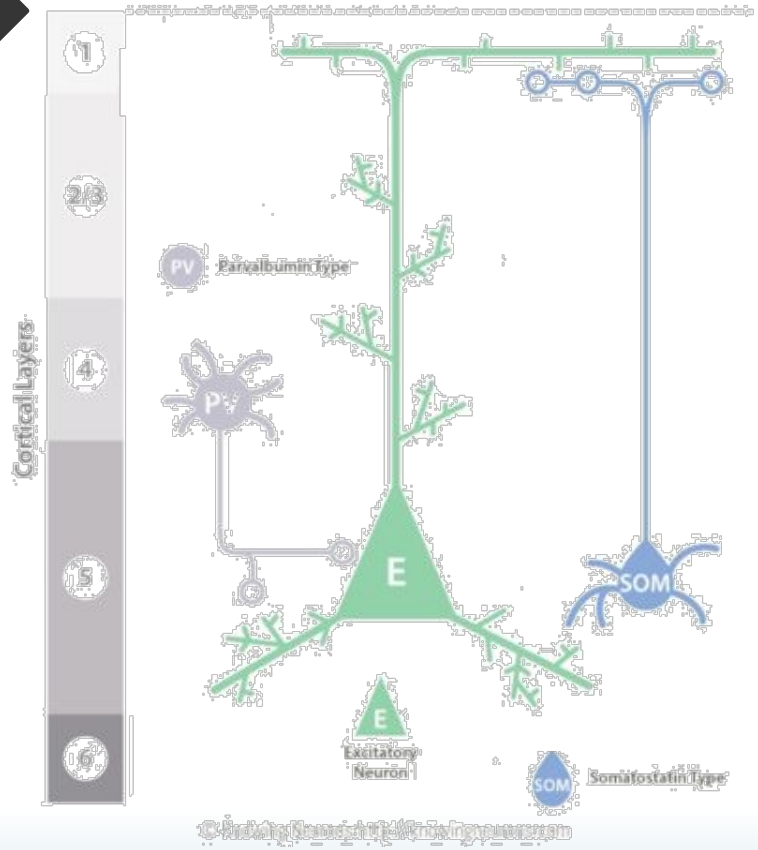
Layer 5b Pyramidal Cell

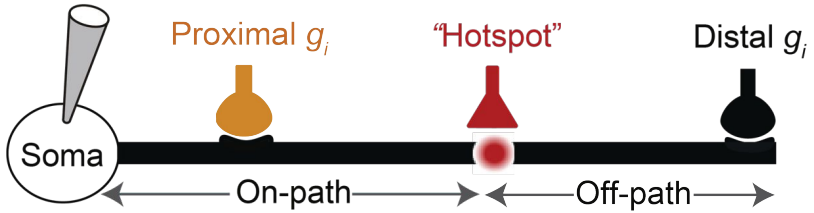


```
# place a single synapse
syn_loc = rng_pos.repick()
apical_exc_syns.append(h.ProbAMPANMDA_EMS(sec(syn_loc)))
# create random stim. times
apical_exc_netstims.append(h.NetStim())
apical_exc_netstims[-1].number = t_stop / 1000 * exc_freq
apical_exc_netstims[-1].interval = 1000 / exc_freq  # ms
apical_exc_netstims[-1].noise = 1 # make it Poisson like
# connect stimulator to the synapse
apical_exc_netconns.append(h.NetCon(apical_exc_netstims[-1]
,
                           apical_exc_syns[-1]))
apical_exc_netconns[-1].weight[0] = gsyn_exc
```

6_random_synapses_Hay.ipynb

Inhibition location

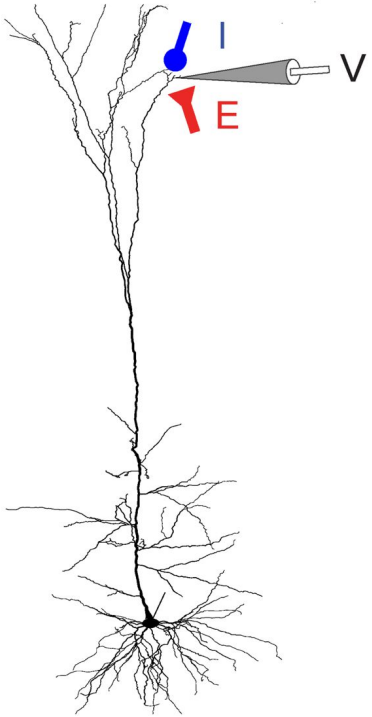


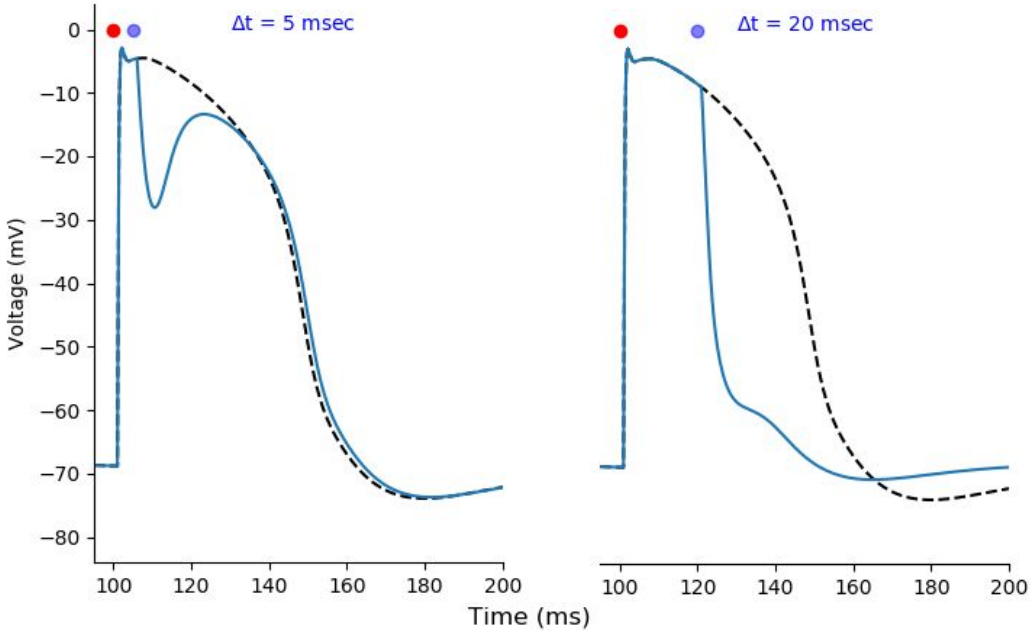


+1_inhibition_location_Gidon_et_al.ipynb

"Principles Governing the Operation of Synaptic Inhibition in Dendrite" Albert Gidon, Idan Segev. Neuron, 2012

Timed inhibition





+2_timed_inhibition_Doron_et_al.ipynb

"Timed Synaptic Inhibition Shapes NMDA Spikes, Influencing Local Dendritic Processing and Global I/O Properties of Cortical Neurons" Michael Doron, Giuseppe Chindemi, Eilif Muller, Henry Markram, Idan Segev. Cell Reports, 2017

Useful links

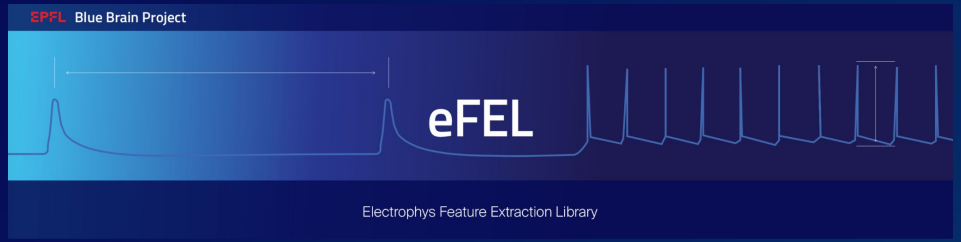
- NEURON: (GitHub)
  - Documentation: Python Doc, HOC Doc, ReadTheDocs
  - More scripting basics, NEURON-Python tutorials, courses and material
  - Queries: FAQs, NEURON Forum and GitHub issues
- Modeling repositories
  - ModelDB (published models)
  - NeuroMoprho (morphologies)
  - Channelpedia and ICGenealogy (ion channels)
  - NeuroElectro (neuron parameters)
- Others
  - Blue Brain Tools (for subcellular, cellular, circuit-level models among others)
  - Allen Institute Data and Tools
  - NetPyNE (modelling networks of neurons)
  - CoreNEURON (faster NEURON execution)
  - Free edx MOOCs: Simulation Neuroscience, Simulating a Hippocampus Microcircuit

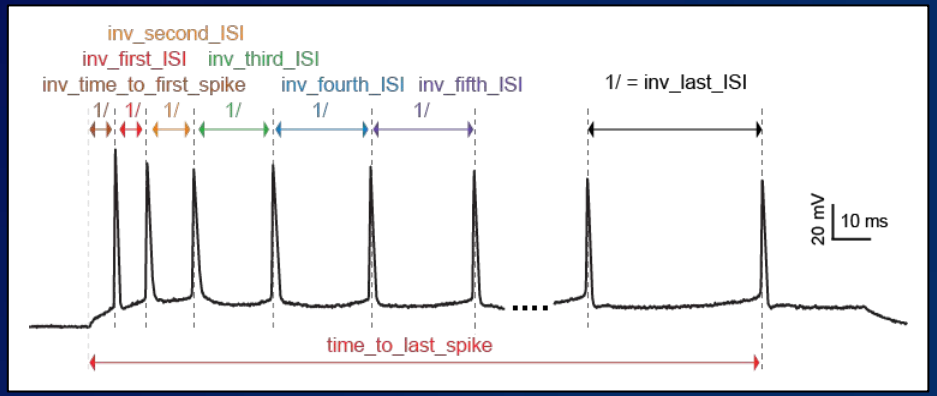
Useful NEURON Tools and Tips

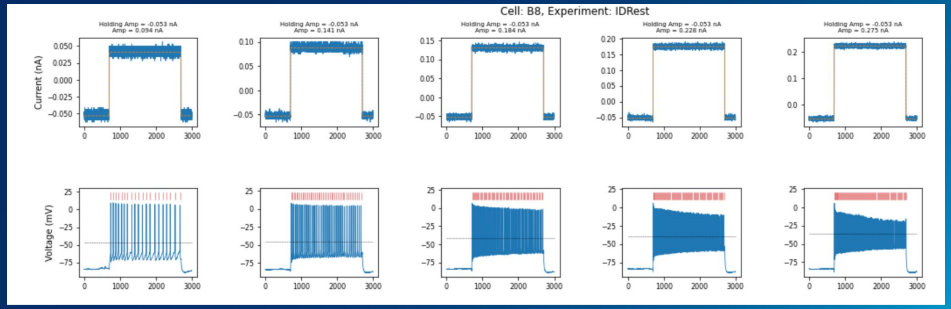
- NEURON-Python scripting Basics
- Units used in NEURON
- NEURON Tips (mainly HOC)
- Specifying temperature in NEURON h.celsius
- Don't forget to compile new mod files before running NEURON
- Randomness in NEURON models
- NEURON Sections overview

Blue Brain Project Software

https://github.com/BlueBrain







github.com/BlueBrain/eFEL

github.com/BlueBrain/BluePyEfe

EPFL Blue Brain Project

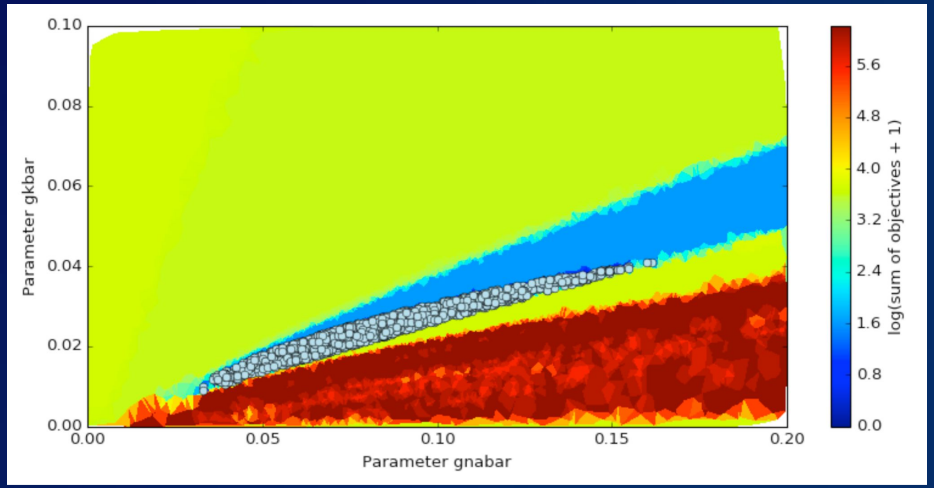
BluePyOpt

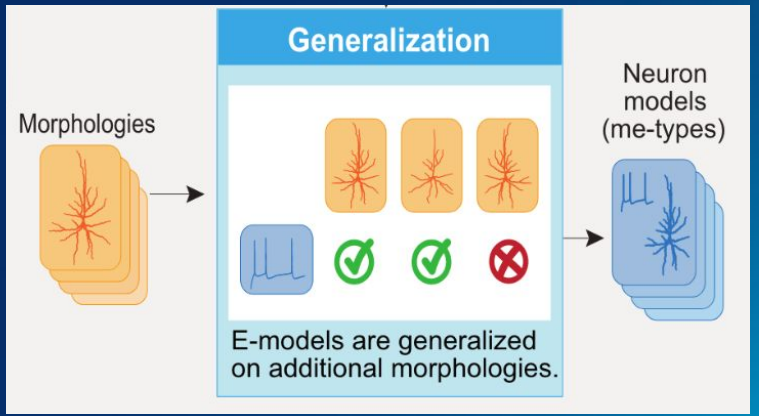
Blue Brain Python Optimisation Library

EPFL Blue Brain Project

BluePyMM

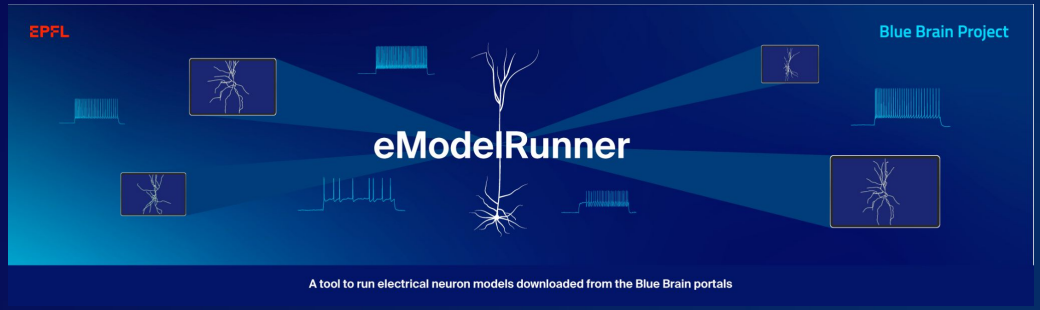
Blue Brain Python Cell Model Management



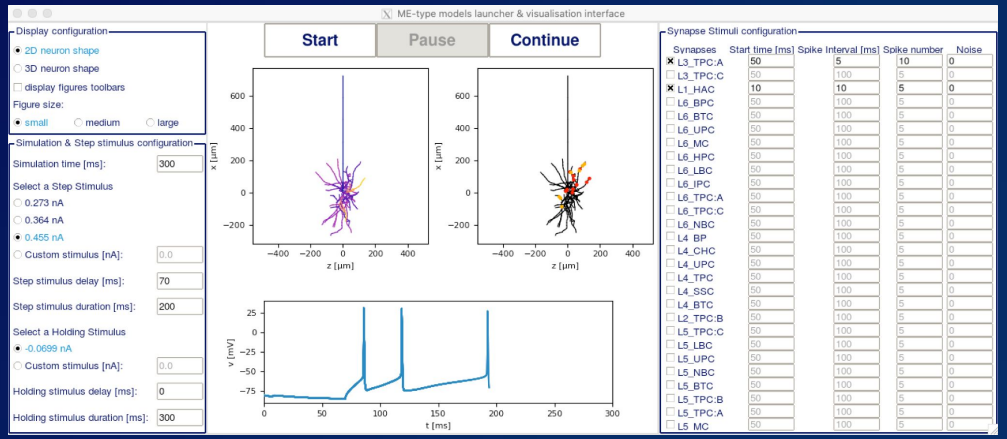


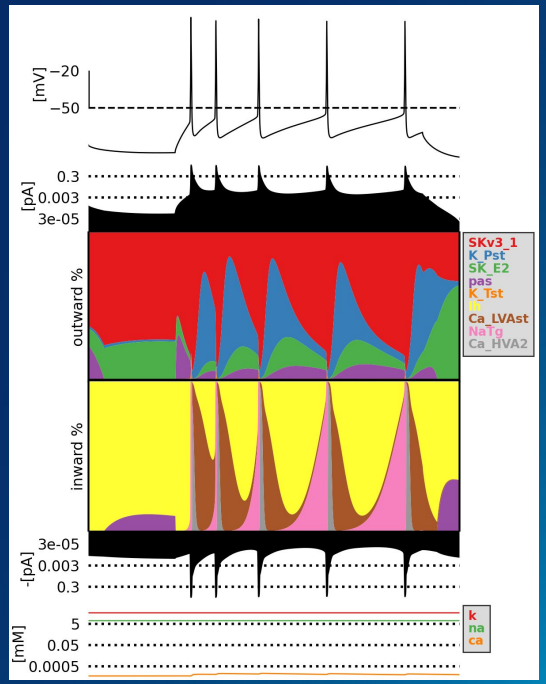
github.com/BlueBrain/BluePyOpt

github.com/BlueBrain/BluePyMM



Currentscape



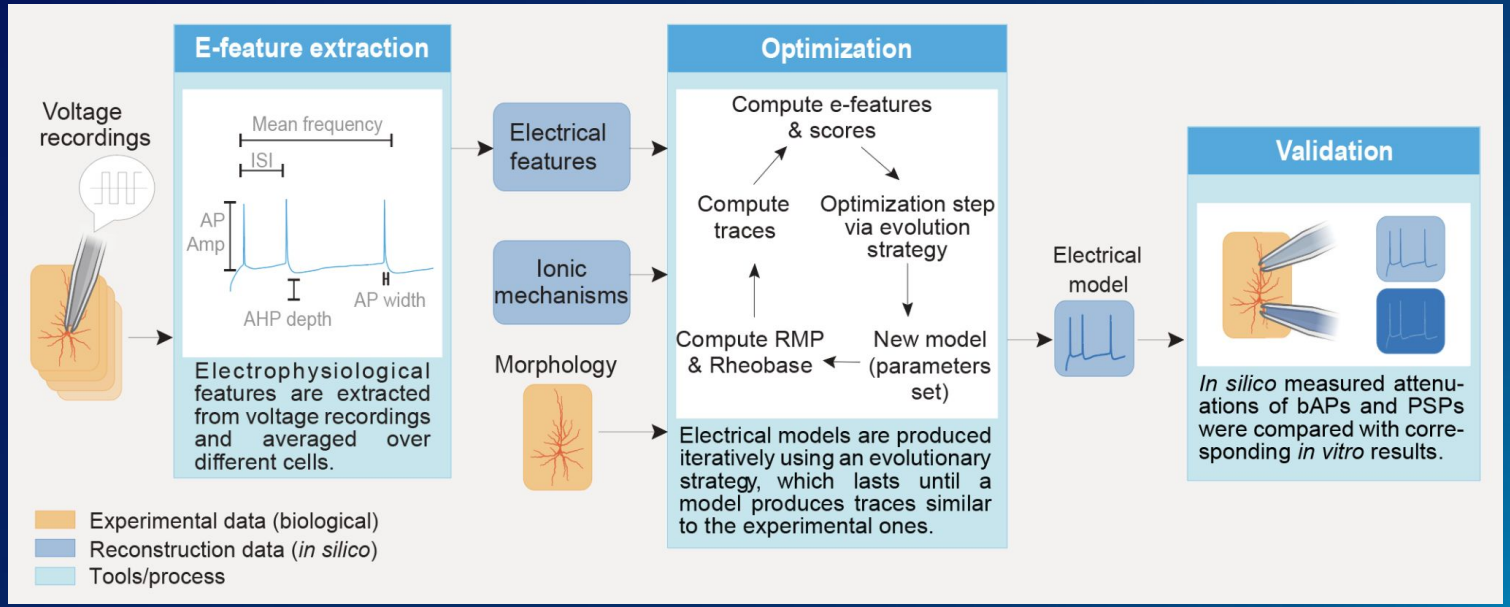


github.com/BlueBrain/EModelRunner

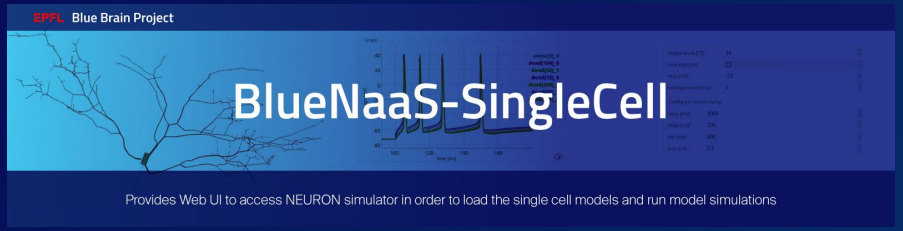
github.com/BlueBrain/Currentscape

BluePyEModel

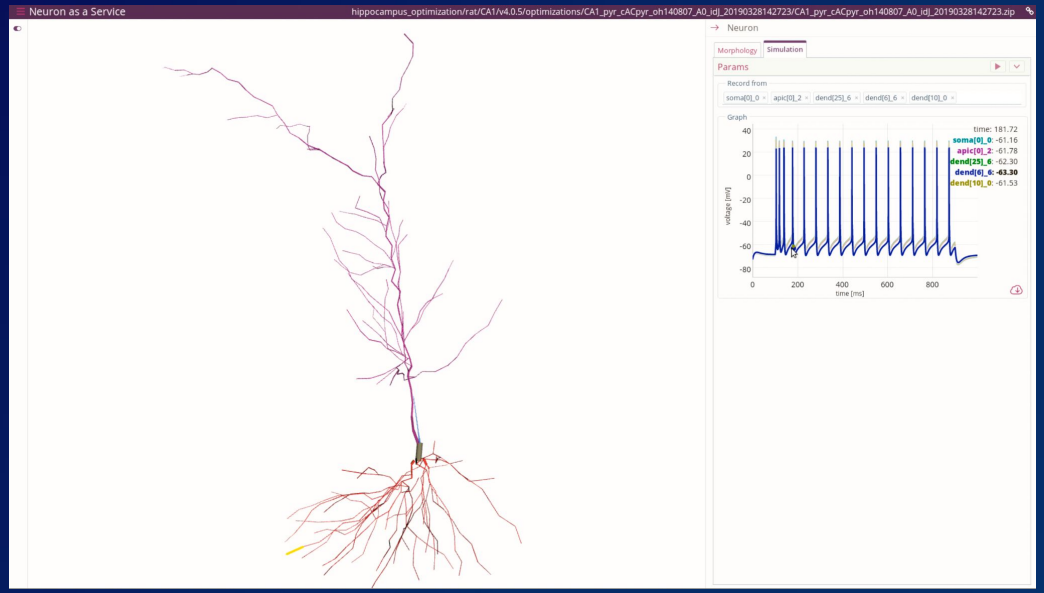
- Unified detailed single cell model building software pipeline
- Uses eFEL, BluePyEfe, BluePyOpt and Currentscape

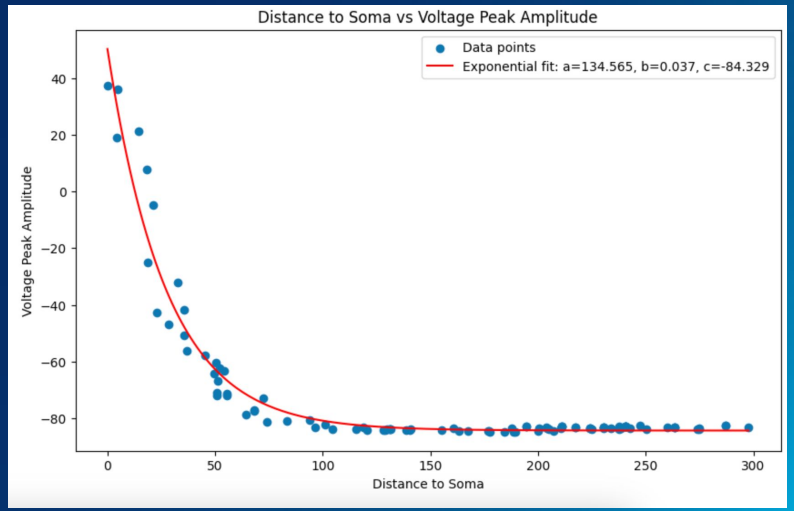


github.com/BlueBrain/BluePyEModel



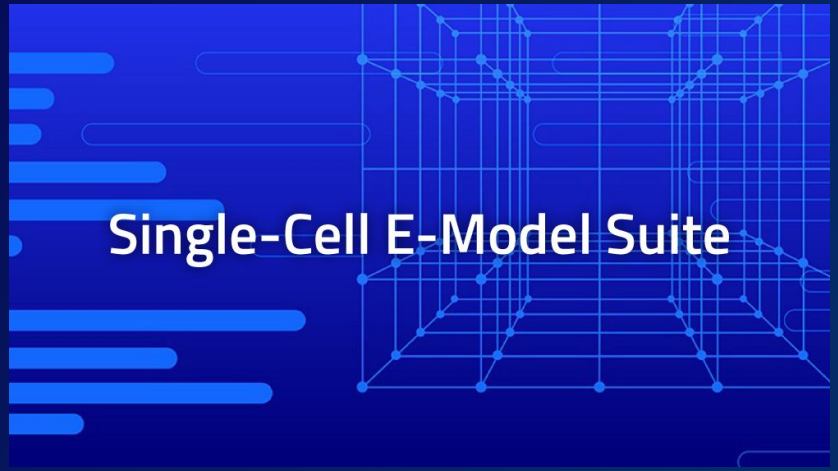
BlueCellulab





github.com/BlueBrain/BlueNaaS-SingleCell

github.com/BlueBrain/BlueCelluLab



- **BluePyEModel** — pipeline for building single cell e-models
- **eFEL** — Electrophys Feature Extraction Library
- **BluePyEfe** — E-feature extraction
- **BluePyOpt** — Parameter optimisation Library
- **BlueCelluLab** — Perform simulations on a single cell or group of cells
- **Currentscape** — Easily plot and visualize currents and ionic concentration in electrical neuron models
- **BluePyMM** — Model management for circuit building
- **EModelRunner** — Runs cells from stand-alone packages
- **BlueNaaS-SingleCell** — Interacts with single cell models through a web application

github.com/BlueBrain/singlecell-emodel-suite

Thank You!