

Outline of this session

- Overview of NEURON
- NEURON concepts and basic scripting
 - Slides
 - Exercises
- Reaction-Diffusion
 - Slides
 - Exercises

Welcome to the community of NEURON users and developers!

The NEURON simulation environment is used in laboratories and classrooms around the world for building and using computational models of neurons and networks of neurons.

Here you will find installers and source code, documentation, tutorials, announcements of courses and conferences, and discussion forums about NEURON in particular and computational neuroscience in general.

Users who have special interests and expertise are invited to participate in the NEURON project by helping to organize future meetings of the NEURON Users Group, and by participating in collaborative development of documentation, tutorials, and software. We also welcome suggestions for ways to make NEURON a more useful tool for research and teaching.

INSTALL NEURON 8.0

On macOS, install via:

```
pip install neuron
```

Installers are also available:

- For machines running on an M1 processor
- For intel-based macs

[Quickstart Guide](#)

[All standard versions](#)

THE NEURON FORUM

The Neuron Forum

- [NEURON Installation](#)
- [Making and using models](#)
- [Programming NEURON with Python](#)
- [NEURON in education](#)
- [Tools of interest](#)
- [Data Sharing](#)
- [Computational neuroscience in general](#)

LATEST NEWS

18 May 2021

[2021 NEURON Webinar Series](#)

3 May 2021

[NEURON 8.0 released](#)

21 December 2020

[NEURON 7.8.2 released](#)

[more news](#)

NEURON Main Menu

Iconify

File Build Tools Graph Vector Window Help

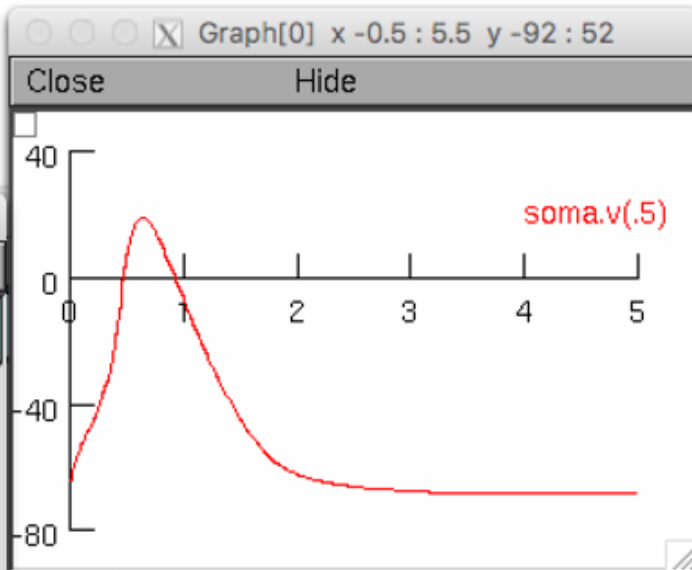
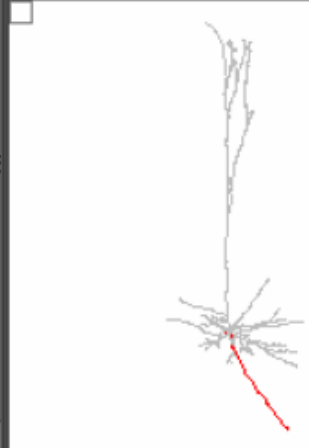
ModelView[0]...

Close Hide Close Hide

File LengthScale

79 sections; 150 segments

- * 1 real cells
 - * root soma
 - 79 sections; 150 segments
 - * 7 distinct values of nseg
 - * 6 inserted mechanisms
 - Ra = 100
 - cm = 1
 - * pas
 - ena = 50
 - ek = -77
 - * hh
 - gnabar_hh = 0.12
 - gkbar_hh = 0.036
 - gl_hh = 0.0003
 - el_hh = -54.3
 - * 3 subsets with constant parameters
 - * 2 Point Processes
- 0 artificial cells
- 0 NetCon objects
- 0 LinearMechanism objects



Absolute Tolerance Scale Factors

Close Hide

Analysis Run Rescale Original

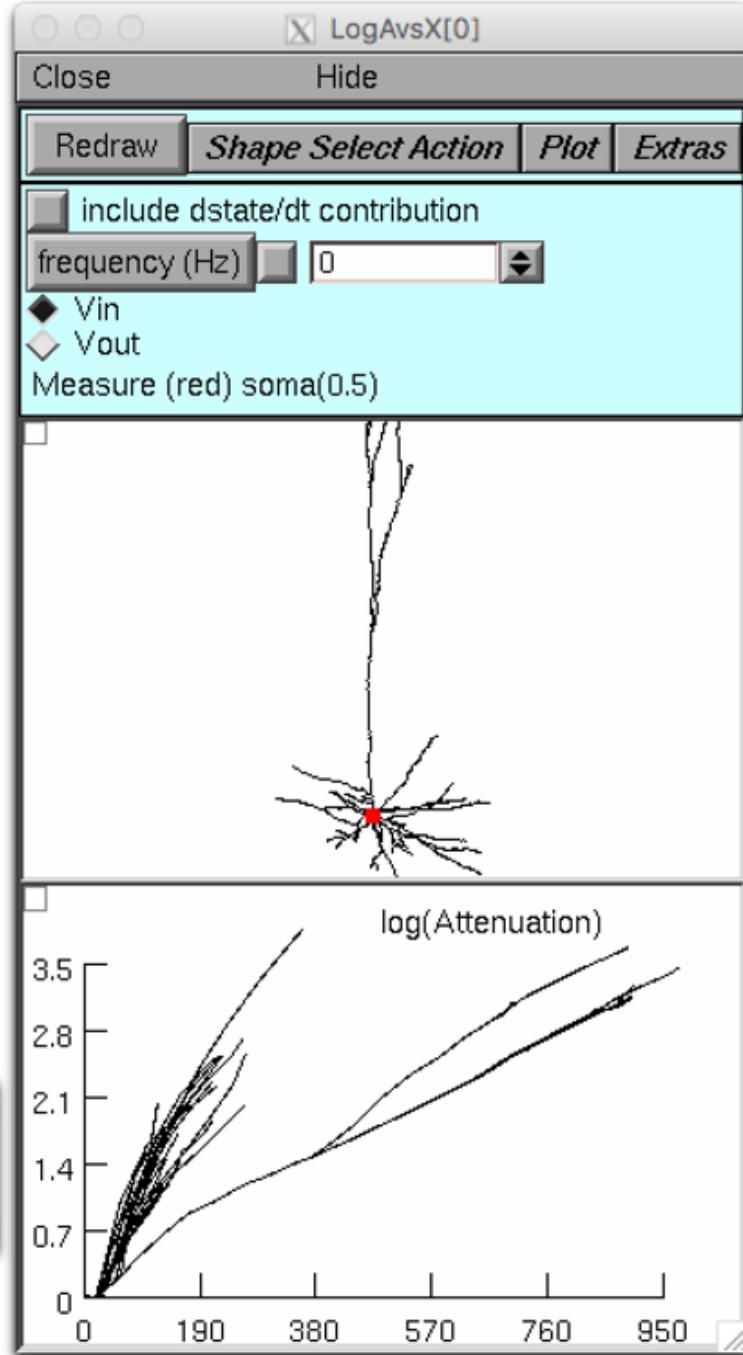
*10 /10 Hints

v	10	76	0.011
m_hh	1	0.99	0.00035
h_hh	0.1	0.6	1.2e-05
n_hh	0.1	0.77	4.4e-06

Temperature

Close Hide

celsius (degC) 15



github.com/mcdougallab/neurongui2

```
Console [1]
File Build Tools Graph Help
Python 3.7.6 (default, Jan 8 2020, 19:59:22)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more
information.
>>>
Type setupSim() or quit() or use the menus or use NEURON as usual
>>> from neuron import h, window
>>> html = """
...     <h1><span style="color:blue">Hello</span> everybody</h1>
...     <input type="number" data-variable="t">
...     <label>time (<i>ms</i></label><br/>
...     The variable s: <span data-variable="s"></span><br/><br/>
...     <button data-onclick="go">Press me</button>
...     """
>>> h('strdef s')
1
>>> h.s = 'Hello'
>>> def go():
...     print(f'You pressed the button! h.t = {h.t}')
...
>>> w = window(html,
...             {'t': h._ref_t, 'go': go, 's': h._ref_s},
...             title='Demo')
>>> h.s = 'Goodbye'
>>> You pressed the button! h.t = 0.0
>>> You pressed the button! h.t = 2.0
>>>
```

Demo

File Build Tools Graph Help

Hello everybody

time (*ms*)

The variable s: Goodbye

Press me

Windows can be defined using HTML.

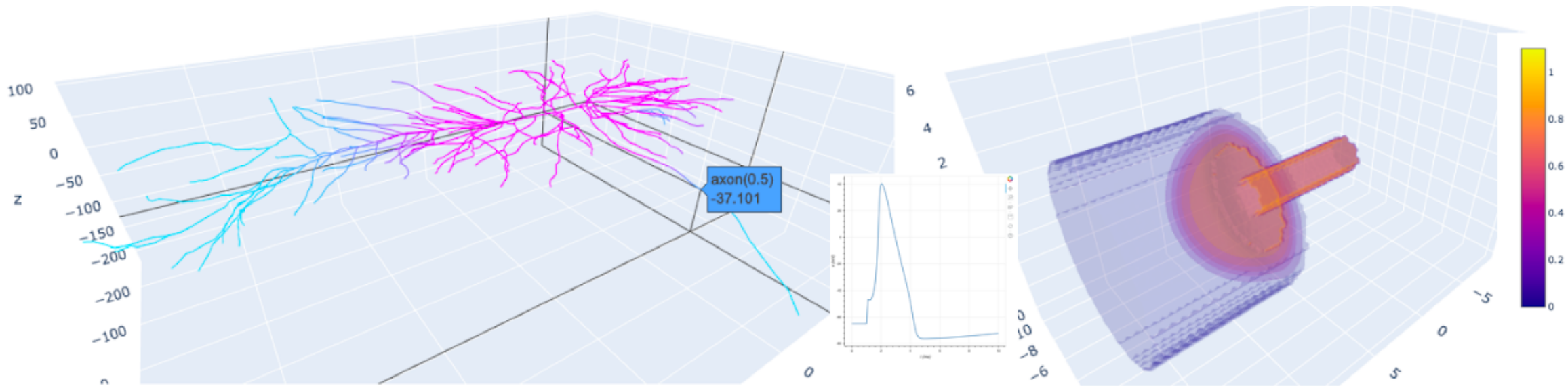
data-* attributes are used to connect to NEURON and are automatically synced and reroutable.

Additional options include specifying menus and size.



NEURON Developer's meetings

- Third Friday of the month, 10am EDT; 16:00 CEST on zoom.
 - Agenda is on the NEURON GitHub wiki:
<https://github.com/neuronsimulator/nrn/wiki>
-



```

from neuron import h
from neuron.units import mV, ms
from matplotlib import cm
import plotly
h.load_file('stdrun.hoc')
h.load_file('c91662.ses')

```

```

h.hh.insert(h.allsec())
ic = h.IClamp(h.soma(0.5))
ic.delay = 1 * ms
ic.dur = 1 * ms
ic.amp = 10

```

```

h.finitialize(-65 * mV)
h.continuerun(2 * ms)

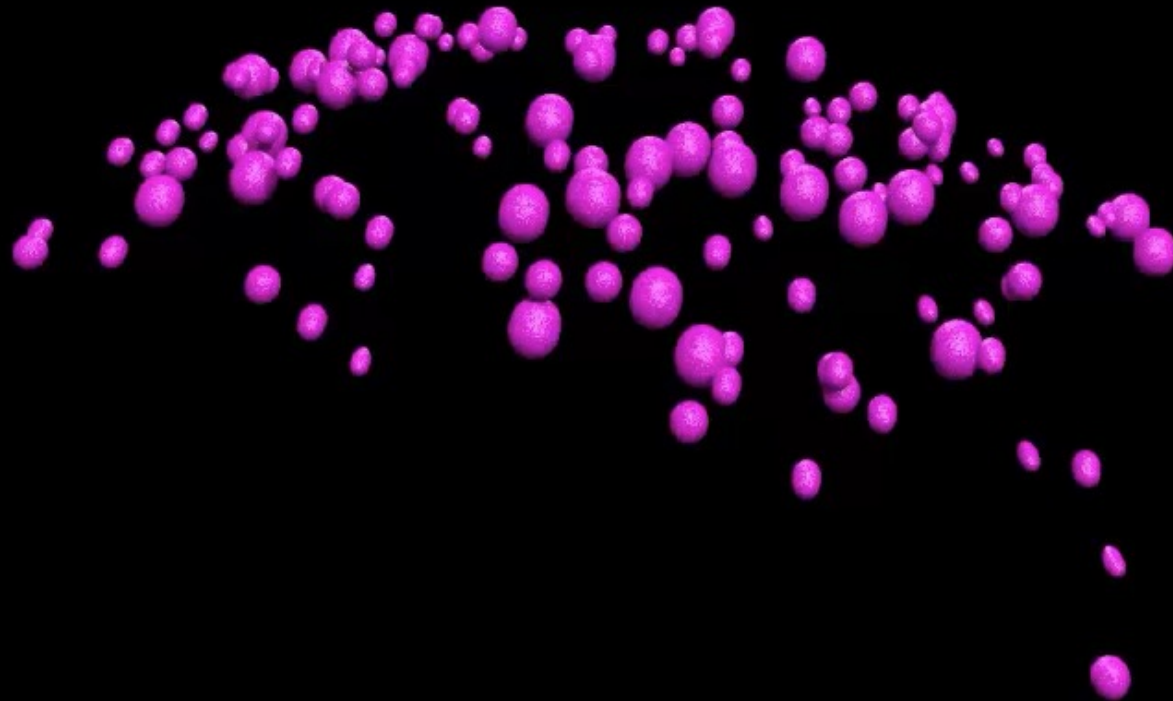
ps = h.PlotShape(False)
ps.variable('v')
ps.plot(plotly, cmap=cm.cool).show()

```

Migliore et al., 2014

doi: [10.3389/fncom.2014.00050](https://doi.org/10.3389/fncom.2014.00050)

modeldb.yale.edu/151681



AVAILABLE TOOLS & SOFTWARE

BluePyOpt, Brian, CARLSim4, DynaSim, EEGLAB, Freesurfer,
Human Neocortical Neurosolver (HNN), Large Scale Neural Simulator, MATLAB, MOOSE, NEST,
NetPyNE, NEURON, Octave, Parameter Search Tool, PGENESIS, PyNN, Python, R, TensorFlow,
Trees/T2N, TVB-Personalized Multimodal Connectome

—UPDATED BASED ON USER NEEDS—

[Register Account »](#)

[Access NSG Portal »](#)

[Join Mailing List »](#)

[Access NSG-R »](#)



Welcome to the community of NEURON users and developers!

The NEURON simulation environment is used in laboratories and classrooms around the world for building and using computational models of neurons and networks of neurons.

Here you will find installers and source code, documentation, tutorials, announcements of courses and conferences, and discussion forums about NEURON in particular and computational neuroscience in general.

Users who have special interests and expertise are invited to participate in the NEURON project by helping to organize future meetings of the NEURON Users Group, and by participating in collaborative development of documentation, tutorials, and software. We also welcome suggestions for ways to make NEURON a more useful tool for research and teaching.

INSTALL NEURON 8.0

On macOS, install via:

```
pip install neuron
```

Installers are also available:

- For machines running on an M1 processor
- For intel-based macs

[Quickstart Guide](#)

[All standard versions](#)

THE NEURON FORUM

The Neuron Forum

- [NEURON Installation](#)
- [Making and using models](#)
- [Programming NEURON with Python](#)
- [NEURON in education](#)
- [Tools of interest](#)
- [Data Sharing](#)
- [Computational neuroscience in general](#)

LATEST NEWS

18 May 2021

2021 NEURON Webinar Series

3 May 2021

NEURON 8.0 released

21 December 2020

NEURON 7.8.2 released

[more news](#)

Welcome to the community of NEURON users and developers!

The NEURON simulation environment is used in laboratories and classrooms around the world for building and using computational models of neurons and networks of neurons.

Here you will find installers and source code, documentation, tutorials, announcements of courses and conferences, and discussion forums about NEURON in particular and computational neuroscience in general.

Users who have special interests and expertise are invited to participate in the NEURON project by helping to organize future meetings of the NEURON Users Group, and by participating in collaborative development of documentation, tutorials, and software. We also welcome suggestions for ways to make NEURON a more useful tool for research and teaching.

INSTALL NEURON 8.0

On macOS, install via:

```
pip install neuron
```

Installers are also available:

- For machines running on an M1 processor
- For intel-based macs

[Quickstart Guide](#)

[All standard versions](#)

THE NEURON FORUM

The Neuron Forum

- [NEURON Installation](#)
- [Making and using models](#)
- [Programming NEURON with Python](#)
- [NEURON in education](#)
- [Tools of interest](#)
- [Data Sharing](#)
- [Computational neuroscience in general](#)

LATEST NEWS

18 May 2021

[2021 NEURON Webinar Series](#)

3 May 2021

[NEURON 8.0 released](#)

21 December 2020

[NEURON 7.8.2 released](#)

[more news](#)

Graph

Grapher

PlotShape

PlotShape Window

RangeVarPlot

Shape

Notification

GUI Look And Feel

MenuExplore

Obsolete Plotting

Analysis

NEURON HOC documentation

Python tutorials

Python RXD tutorials

How to use CoreNEURON

DEVELOPER DOCUMENTATION:

NEURON SCM and Release

NEURON Development topics

C/C++ API

Read the Docs

v: latest

» NEURON Python documentation » Graph

Edit on GitHub Switch to HOC

Graph

[addexpr](#) · [addobject](#) · [addvar](#) · [align](#) · [begin](#) · [beginline](#) · [brush](#) · [color](#) · [crosshair_action](#) · [erase](#) · [erase_all](#) · [exec_menu](#) · [family](#) · [fastflush](#) · [fixed](#) · [flush](#) · [getline](#) · [gif](#) · [glyph](#) · [label](#) · [line](#) · [line_info](#) · [mark](#) · [menu_action](#) · [menu_remove](#) · [menu_tool](#) · [plot](#) · [printfile](#) · [relative](#) · [save_name](#) · [simgraph](#) · [size](#) · [unmap](#) · [vector](#) · [vfixed](#) · [view](#) · [view_count](#) · [view_info](#) · [view_size](#) · [xaxis](#) · [xexpr](#) · [yaxis](#)

Graph

class Graph

Syntax:

```
g = h.Graph()
```

```
g = h.Graph(0)
```

Description:

An instance of the Graph class manages a window on which x-y plots can be drawn by calling various member functions. The first form immediately maps the window to the screen. With a 0 argument the window is not mapped but can be sized and placed with the `view()` function.

Direct documentation link: nrn.readthedocs.io

Use the “Switch to HOC” link in the upper-right corner of every page if you need documentation for HOC, NEURON’s original programming language. HOC may be used in combination with Python: use `h.load_file` to load a HOC library; the functions and classes are then available with an `h.` prefix.



NEURON Simulation Environment

35 subscribers

SUBSCRIBE

HOME

VIDEOS

PLAYLISTS

CHANNELS

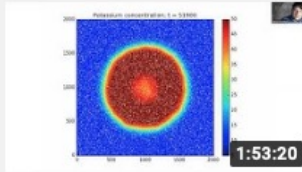
DISCUSSION

ABOUT



Uploads

▶ PLAY ALL



20210701 Python scripting day 3

10 views • 1 day ago



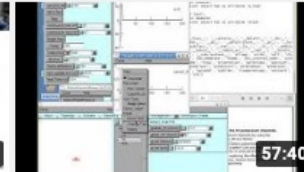
20210629 Python scripting day 2

9 views • 3 days ago



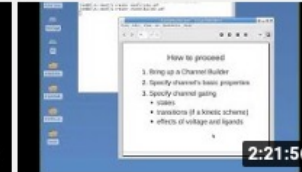
20210624 Python scripting day 1

38 views • 1 week ago



20210622 Adding Ion Channels Q and A

16 views • 1 week ago



20210617 Adding Ion Channels

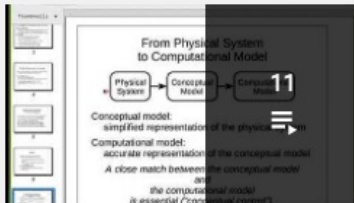
52 views • 2 weeks ago



20210615 Branched Cells Q and A part 2

33 views • 2 weeks ago

Created playlists



2021 NEURON Online Course

NEURON Simulation Environment • Updated yesterday

20210603 Basic Concepts And GUI • 2:16:57

20210608 Basic Concepts And GUI Q and A • 1:22:12

VIEW FULL PLAYLIST

For more...

tinyurl.com/neuron-videos

Installation

- macOS or Linux:

```
pip install neuron
```

or

```
pip install neuron-nightly
```

- Windows:

[DOWNLOAD 7.7.2](#)

[Download Windows installer \(64 bit\)](#)

[Quickstart Guide](#)

[All standard versions](#)

[Alpha versions](#)

[Source on github](#)

+ Python (e.g. Anaconda)

Welcome To Colaboratory - Col X

https://colab.research.google.com/notebooks/intro.ipynb?auth=

Welcome To Colaboratory

File Edit View Insert Runtime Tools Help

Share R

Connect Editing

Table of contents

- Getting started
- Data science
- Machine learning
- More Resources
- Machine Learning Examples
- Section

Examples Recent Google Drive GitHub Upload

Filter notebooks

Title	First opened	Last opened	
Welcome To Colaboratory	Mar 30, 2020	0 minutes ago	
Untitled1.ipynb	Apr 5, 2020	Apr 5, 2020	
Untitled0.ipynb	Apr 5, 2020	Apr 5, 2020	
course-mockup-exercise.ipynb	Mar 30, 2020	Mar 30, 2020	

NEW NOTEBOOK CANCEL

Introduction to Colab to learn more,

book that lets you write and

and prints the result:

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.

Variables that you define in one cell can later be used in other cells:

Connecting to NEURON

- For today's tutorial, we will use Google Colab, but any way you like to use Python is a good way.

- The main NEURON submodule is called `h`, but there are also: `units`, `rxn`, and `gui`.

```
[1] !pip install neuron-nightly

Collecting neuron-nightly
  Downloading https://files.pythonhosted.org/packages/2d/bb/65fa8cba78bfa4d873ae7e... 8.7MB 3.8MB/s
Requirement already satisfied: numpy>=1.9.3 in /usr/local/lib/python3.6/dist-packa...
Installing collected packages: neuron-nightly
Successfully installed neuron-nightly-8.0.dev187

[2] from neuron import h
    from neuron.units import mV, ms
```

Basic unit: h.Section

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

Length: `soma.L`

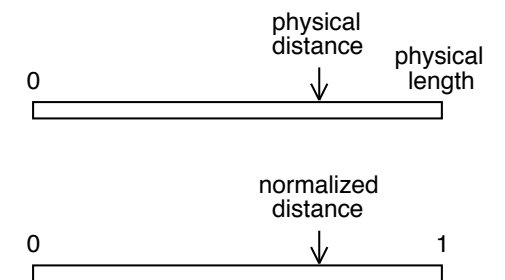
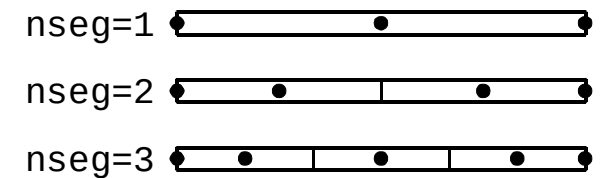
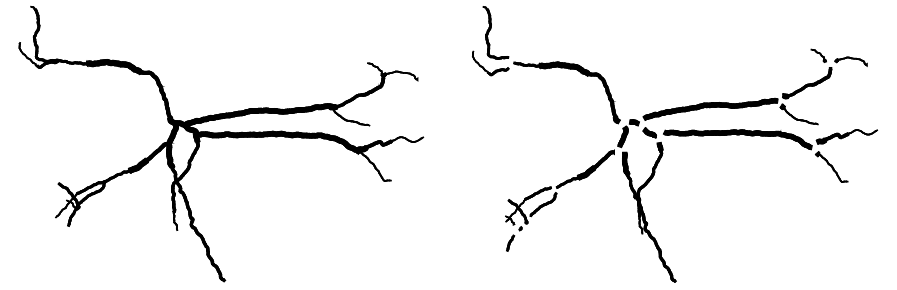
Diameter: `soma.diam`

Discretization: `soma.nseg`

Inside a cell class, specify the cell argument as well:

```
soma = h.Section(  
    name='soma',  
    cell=self)
```

The `connect` method joins Section objects to define arbitrary morphologies.



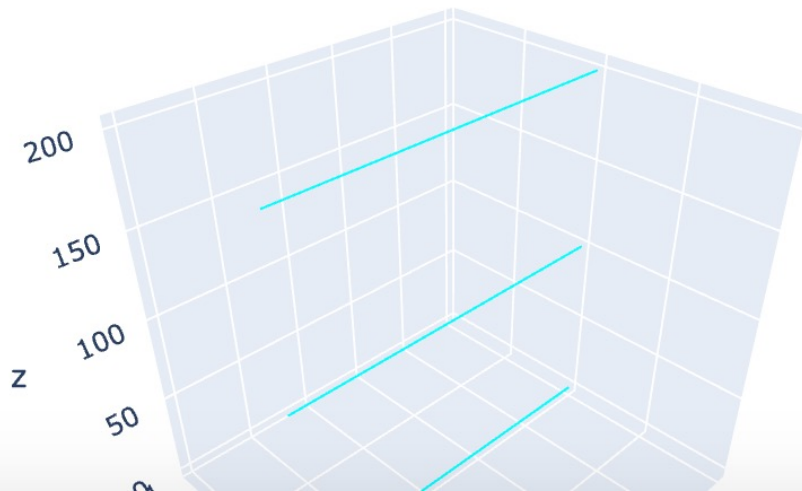

```
[2] from neuron import h
     from neuron.units import mV, ms
     import plotly
```

```
[3] main = h.Section(name='main')
     branch1 = h.Section(name='branch1')
     branch2 = h.Section(name='branch2')
     h.define_shape()
```

☞ 1.0

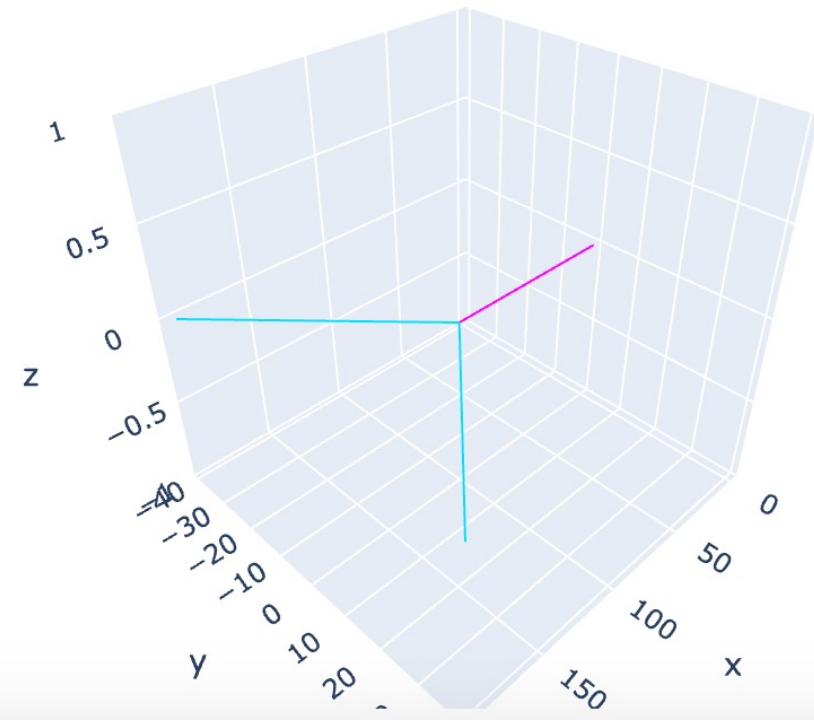
```
[4] h.PlotShape(False).plot(plotly).show()
```

☞



```
[6] branch1.connect(main)
     branch2.connect(main)
     for sec in h.allsec():
         sec.pt3dclear()
     h.define_shape()
     ps = h.PlotShape(False)
     ps.variable('v')
     ps.plot(plotly).show()
```

☞



Viewing voltage, sodium, etc...

Variable	Value	Pointer (e.g. for recording)	With PlotShape or RangeVarPlot
Voltage	seg.v	seg._ref_v	"v"
Na ⁺ (inside membrane)	seg.nai	seg._ref_nai	"nai"
Na ⁺ (outside membrane)	seg.nao	seg._ref_nao	"nao"
Na ⁺ (current)	seg.ina	seg._ref_ina	"ina"
Na ⁺ (reversal potential)	seg.ena	seg._ref_ena	"ena"
d(sodium current)/dv	seg.dina_dv_	seg._ref_dina_dv_	"dina_dv_"

Potassium is the same as for sodium, except with "k" replacing "na"; Chloride is the same except with "cl"; Calcium is the same except with "ca", etc... ions may only be accessed when a mechanism using them is present or when they are explicitly inserted via sec.insert or rxd.

Ion Channels

- Specify using `insert` method.
- Built-in: Hodgkin-Huxley (`h.hh`), passive (`h.pas`)
- Hundreds more on ModelDB (`.mod` files)
- Compile mod files via: `nrnivmodl`

Model

Hodgkin-Huxley cable equations

$$\frac{D}{4R_a} \frac{\partial^2 V}{\partial x^2} = C_m \frac{\partial V}{\partial t}$$

$$+ \bar{g}_m m^3 h \cdot (V - E_{na}) + \bar{g}_k n^4 \cdot (V - E_k) + g_l \cdot (V - E_l)$$

$$\frac{dm}{dt} = -\alpha_m m + \beta_m (1 - m) \quad \alpha_m = \frac{0.1(V+40)}{1 - e^{-0.1(V+40)}} \quad \beta_m = 4e^{-(V+65)/18}$$

$$\frac{dh}{dt} = -\alpha_h h + \beta_h (1 - h) \quad \alpha_h = 0.07e^{-0.05(V+65)} \quad \beta_h = \frac{1}{1 + e^{-0.1(V+35)}}$$

$$\frac{dn}{dt} = -\alpha_n n + \beta_n (1 - n) \quad \alpha_n = \frac{0.01(V+55)}{1 - e^{-0.1(V+55)}} \quad \beta_n = 0.125e^{-(V+65)/80}$$

Simulation

Representation

```
axon = h.Section(name = 'axon')
```

```
axon.L = 2e4 * um
```

```
axon.diam = 100
```

```
axon.nseg = 43
```

```
h.hh.insert(axon)
```

```
TITLE hh.mod  squid sodium, potassium, and leak channels
```

```
COMMENT
```

```
This is the original Hodgkin-Huxley treatment for the set of sodium,  
potassium, and leakage channels found in the squid giant axon membrane.  
("A quantitative description of membrane current and its application  
conduction and excitation in nerve" J.Physiol. (Lond.) 117:500-544 (1952).)  
Membrane voltage is in absolute mV and has been reversed in polarity  
from the original HH convention and shifted to reflect a resting potential  
of -65 mV.  
Remember to set celsius=6.3 (or whatever) in your HOC file.  
See squid.hoc for an example of a simulation using this model.  
SW Jaslove 6 March, 1992
```

```
ENDCOMMENT
```

```
UNITS {  
    (mA) = (milliamp)  
    (mV) = (millivolt)  
    (S) = (siemens)  
}
```

```
? interface
```

```
NEURON {  
    SUFFIX hh  
    REPRESENTS NCIT:C17145 : sodium channel  
    REPRESENTS NCIT:C17008 : potassium channel  
    USEION na READ ena WRITE ina REPRESENTS CHEBI:29101
```

Defining ion channels and synapses

tinyurl.com/hhmodfile

tinyurl.com/expsyn

- ModelDB Help
- User account
 - Login
 - Register
- Find models by
 - Model name
 - First author
 - Each author
 - Region(circuits)
- Find models for
 - Cell type
 - Current
 - Receptor
 - Gene
 - Transmitters
 - Topic
 - Simulators
 - Methods
- Find models of
 - Realistic Networks
 - Neurons
 - Electrical synapses (gap junctions)
 - Chemical synapses
 - Ion channels
 - Neuromuscular junctions

Amyloid beta (IA block) effects on a model CA1 pyramidal cell (Morse et al. 2010)

Download zip file Auto-launch
 Help downloading and running models

Model Information | Model File | Citations | Model Views | Simulation Platform | 3D Print

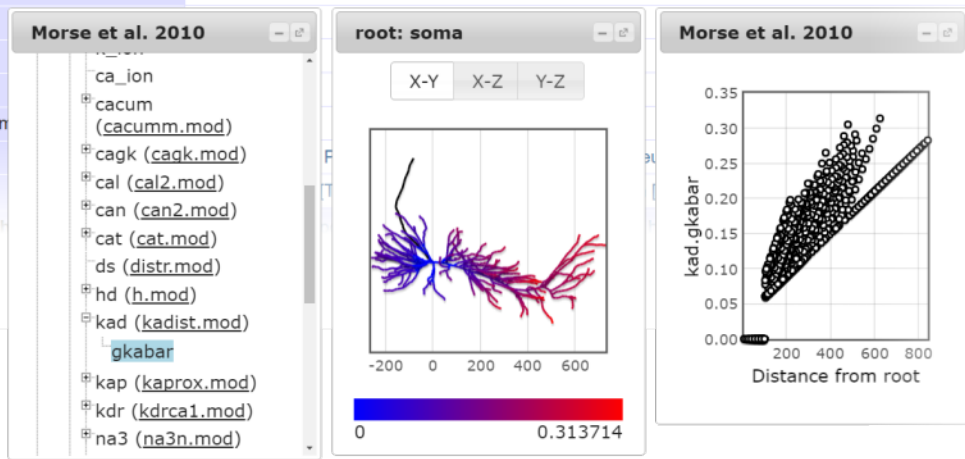
Accession:87284

The model simulations provide evidence oblique dendrites in CA1 pyramidal neurons are susceptible to hyper-excitability by amyloid beta block of channel, IA. See paper for details.

Reference:
 1 . Morse TM, Carnevale NT, Mutalik PG, Migliore M, Shepherd GM (2010) Abnormal excitability of oblique dendrites implicated in early Alzheimer's disease: a computational study *Front. Neural Circuits* 4:16 [PubMed]

Model Information (Click on a link to find other models with that property)

Model Type:	Neuron or other electrically excitable cell;
Brain Region(s)/Organism:	
Cell Type(s):	Hippocampus CA1 pyramidal cell;
Channel(s):	I Na,t; I L high threshold; I N; I T low threshold; I A; I K; I h;
Gap Junctions:	



```

from neuron import h, rxd
import neuron.rxd.node as node
from matplotlib import pyplot
import time

h.load_file('stdrun.hoc')

soma = h.Section()
soma.L = 10
soma.diam = 10
soma.nseg = 11
dend = h.Section()
dend.connect(soma)
dend.L = 50
dend.diam = 2
dend.nseg = 51

def print_nodes():
    print ', '.join(str(v) for v in node._states)

print 'defining rxd'
region = rxd.Region(h.allsec(), nrn_region='i')
ca = rxd.Species(region, name='ca', d=1, charge=2, initial:
reaction = rxd.Rate(ca, -ca * (1 - ca) * (0.3 - ca))

print 'initializing'
h.initialize()
    
```

Morse TM, Carnevale NT, Mutalik PG, Migliore M, Shepherd GM (2010) Abnormal excitability of oblique dendrites implicated in early Alzheimer's: a computational study *Front. Neural Circuits* 4:16 [PubMed]

References and models cited by this paper

- Acker CD, White JA (2007) Roles of I(A) and morphology in action potential propagation in CA1 pyramidal cell dendrites. *J Comput Neurosci* 23(2):201-16 [Journal] [PubMed]
- Roles of I(A) and morphology in AP prop. in CA1 pyramidal cell dendrites (Acker and White 2007) [Model]
- Anderton BH, Callahan L, Coleman P, Davies P, Flood D, Jicha GA, Ohm T, Weaver C (1998) Dendritic changes in Alzheimer's disease and factors that may underlie these changes. *Prog Neurobiol* 55:595-609 [PubMed]
- Andrasfalvy BK, Makara JK, Johnston D, Magee JC (2008) Altered synaptic and non-synaptic properties of

References and models that cite this paper

- Culmone V, Migliore M (2012) Progressive effect of beta amyloid peptides accumulation on CA1 pyramidal neurons: a model study suggesting possible treatments *Front Comput Neurosci* 6:52 [Journal] [PubMed]
- CA1 pyramidal neurons: effects of Alzheimer (Culmone and Migliore 2012) [Model]
- McDougal RA, Morse TM, Hines ML, Shepherd GM (2015) ModelView for ModelDB: online presentation of model structure *Neuroinformatics* 13(4):459-70 [Journal] [PubMed]
- ModelView: online structural analysis of computational models (McDougal et al. 2015) [Model]

modeldb.yale.edu • @senselabproject

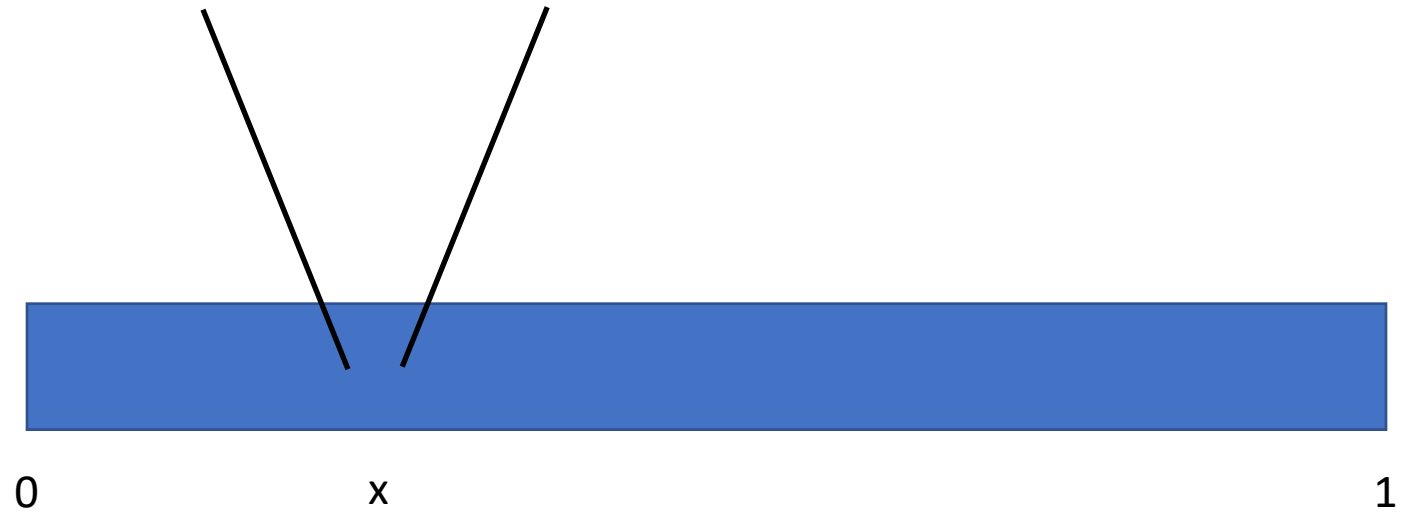
Stimulating a Model

We will inject current using an

`h.IClamp`

This object has several important properties:

- `delay`
- `amp`
- `dur`



```
iclamp = h.IClamp(axon(x))
```

Stimulating a Model II

Set potential

- `soma(0.5).v = 10 * mV`
- Voltage clamp
 - `cl = h.SEClamp(soma(0.5))`
 - `cl.amp1 = -65 * mV`
 - `cl.dur1 = 10 * ms`
 - Similarly for `.amp2`, `.amp3`, `.dur2`, `.dur3`
- Could also:
`vec.play(cl._ref_amp2)`
- SEClamp – single electrode
- VClamp – two electrode

Current Clamp

- `ic = h.IClamp(soma(0.5))`
- `ic.delay = 5 * ms`
- `ic.dur = 0.1 * ms`
- `ic.amp = 1 # nA`

Synaptic input

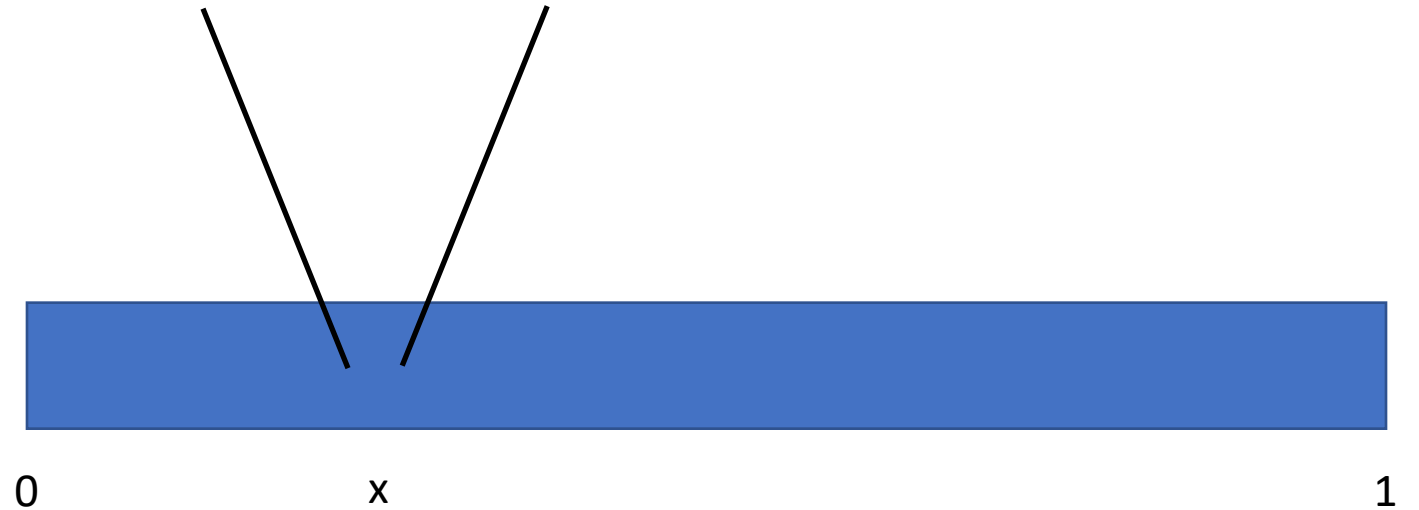
- `ns = h.NetStim()`
 - `ns.number = 1`
 - `ns.start = 5 * ms`
 - `ns.noise = False`
 - `ns.interval = 20 * ms`
 - Only matters for `number > 1`
- `sy = h.ExpSyn(soma(0.5))`
 - `sy.tau = 5 * ms`
 - `sy.e = 0 * mV`
- `nc = h.NetCon(ns, sy)`
 - `nc.weight[0] = 1`

Recording Results

We can read the instantaneous membrane potential at a location via, e.g.

```
axon(0.5).v
```

To record this value over time, we use an `h.Vector` and pass in the pointer (prefixed with `_ref_`) to the `record` method.



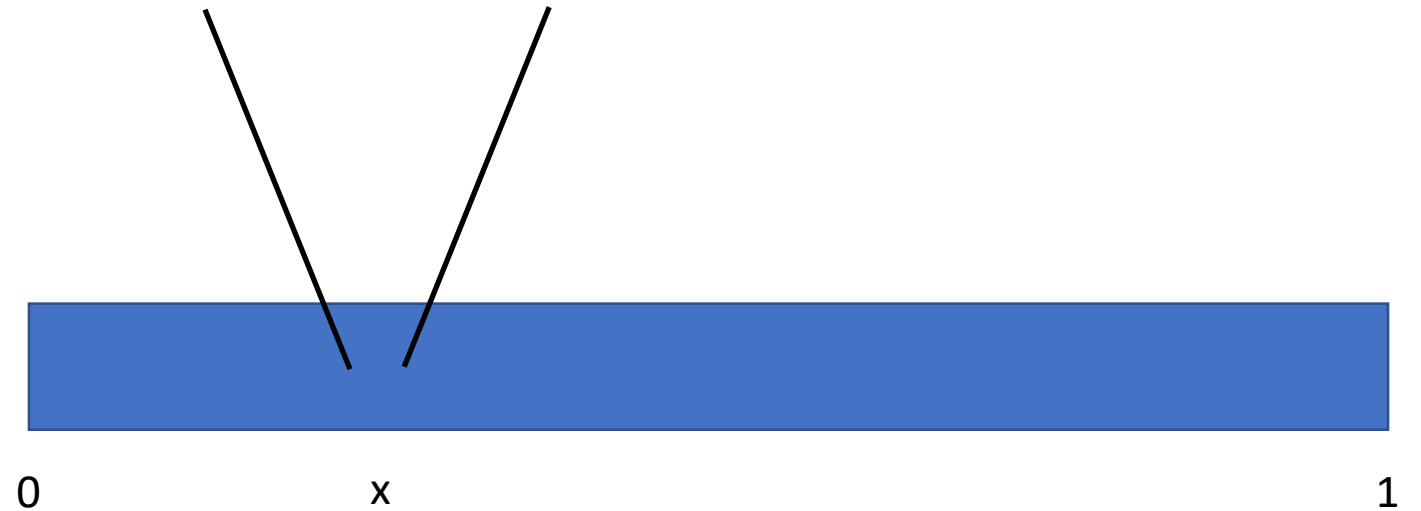
```
v = h.Vector().record(axon(x)._ref_v)
t = h.Vector().record(h._ref_t)
```

eFEL from the Blue Brain Project (`pip install efel`) can be used to identify electrophysiological features (action potential height, half width, etc) from the recorded timeseries.

Recording Results II

NetCon objects can be used as shown to detect the times when a variable crosses a threshold from below.

As the name suggests, a NetCon can be used to *connect* cells together in a *network*. To do this, pass in a synapse as the second argument or use ParallelContext.




```
spike_times = h.Vector()
nc = h.NetCon(axon(0.1)._ref_v, None, sec=axon)
nc.threshold = 0 * mV
nc.record(spike_times)
```

Running the Simulation


For convenience, we use a high-level simulation control functions defined in the `stdrun.hoc` library. Load this via:

```
h.load_file('stdrun.hoc')
```



Initialize to -65 mV:

```
h.finitialize(-65 * mV)
```



Run until time 10 ms:

```
h.continuerun(10 * ms)
```

Caution: Squid

NEURON's defaults are based on the squid giant axon.

sec.diam: 500 μm

sec.Ra: 35.4 $\Omega\text{ cm}$

h.celsius: 6.3 C



Colab time

```
[1] !pip install neuron-nightly
```

```
Requirement already satisfied: neuron-nightly in /usr/local/lib/python3.6/dist-  
Requirement already satisfied: numpy>=1.9.3 in /usr/local/lib/python3.6/dist-pa
```

```
[2] import plotly.graph_objects as go  
from neuron import h  
from neuron.units import mV, ms  
h.load_file('stdrun.hoc')
```

```
1.0
```

```
[3] soma = h.Section(name='soma')  
h.hh.insert(soma)
```

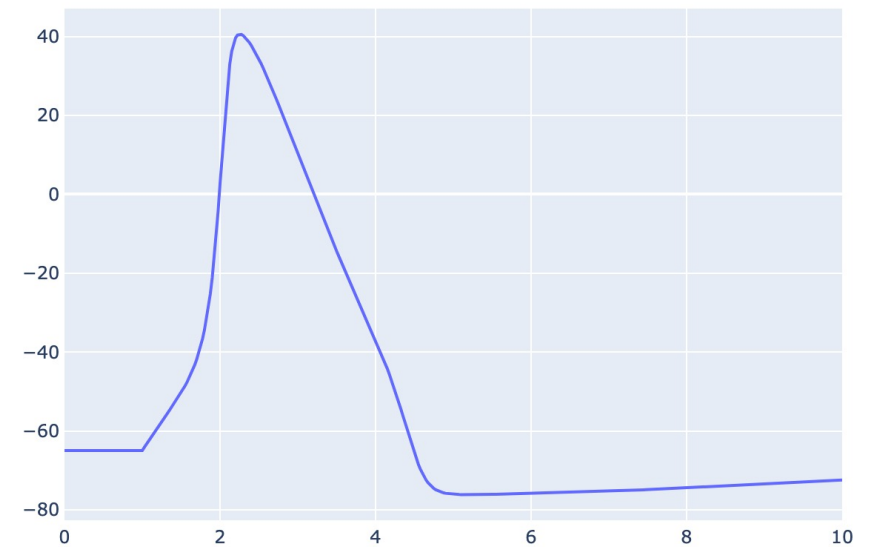
```
[4] ic = h.IClamp(soma(0.5))  
ic.delay = 1  
ic.amp = 50  
ic.dur = 1
```

```
[5] t = h.Vector().record(h._ref_t)  
v = h.Vector().record(soma(0.5)._ref_v)
```

```
[6] h.finitialize(-65 * mV)  
h.continuerun(10 * ms)
```

```
0.0
```

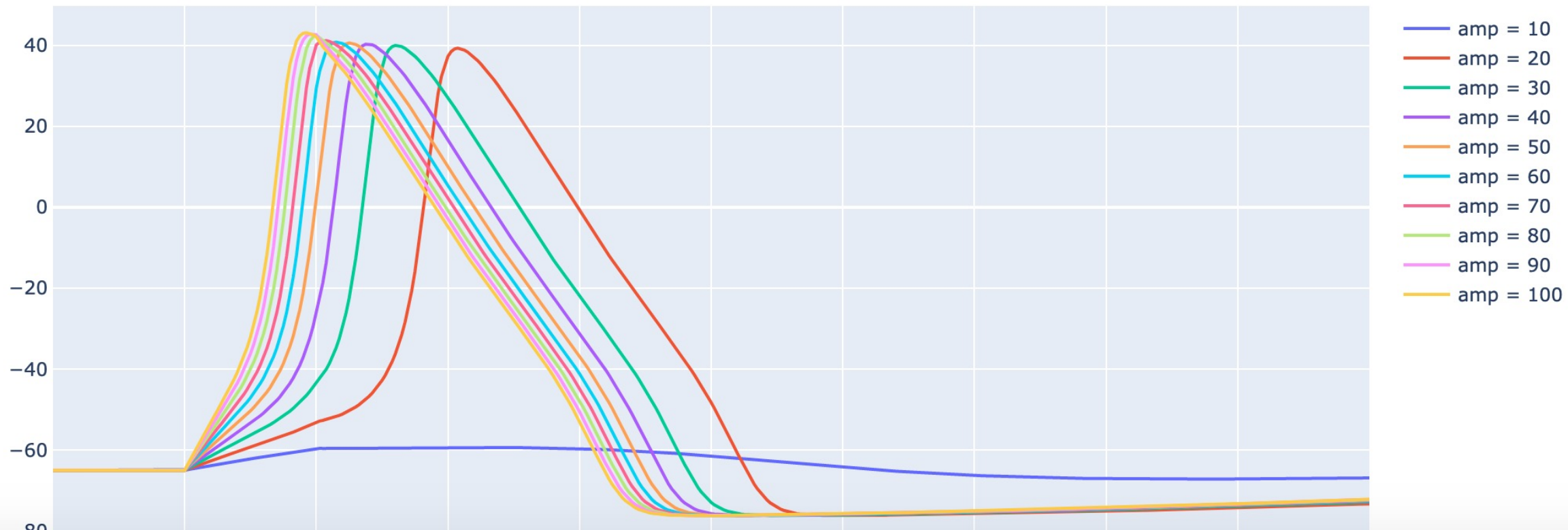
```
[7] go.Figure(data=go.Scatter(x=t, y=v))
```



↑ ↓ ↻ 🗨 ⚙ 🗑 ⋮

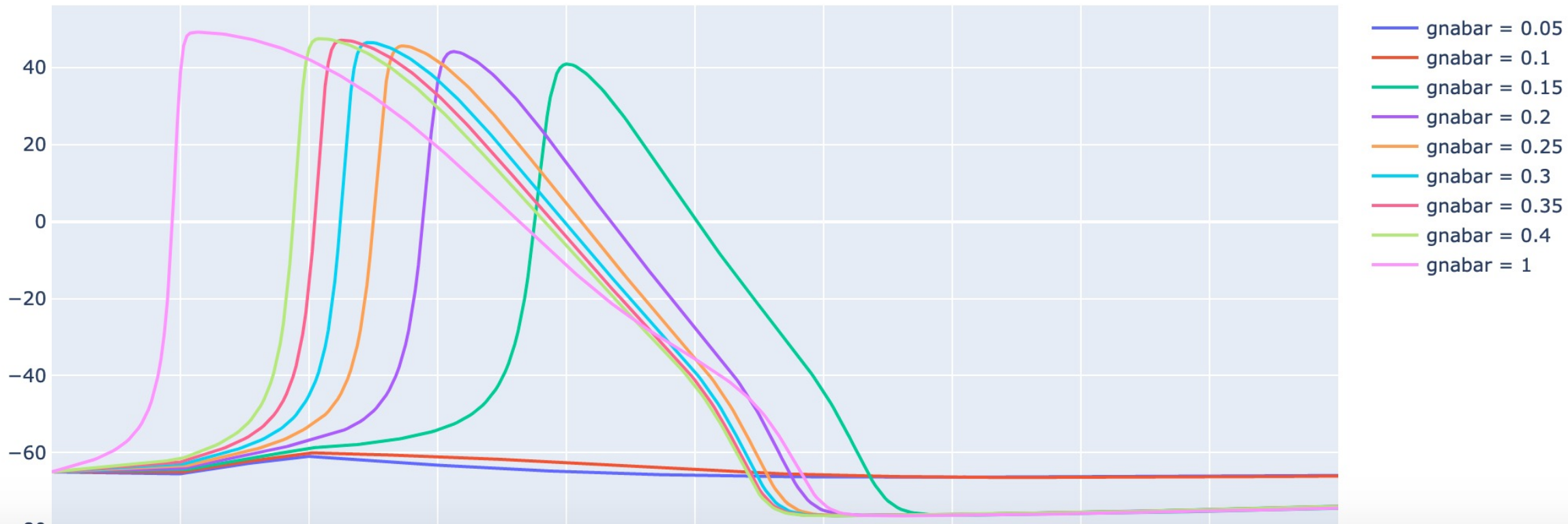
```
[17] fig = go.Figure()
```

```
for amp in [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]:  
    ic.amp = amp  
    h.finitialize(-65 * mV)  
    h.continuerun(10 * ms)  
    fig.add_trace(go.Scatter(x=t, y=v, name=f'amp = {amp}'))  
  
fig.show()
```



```
[24] ic.amp = 10
fig = go.Figure()
for gnabar in [0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 1]:
    soma(0.5).hh.gnabar = gnabar
    h.finitialize(-65 * mV)
    h.continuerun(10 * ms)
    fig.add_trace(go.Scatter(x=t, y=v, name=f'gnabar = {gnabar}'))

fig.show()
```



Increasing accuracy

Increase time resolution (by reducing time steps) via, e.g.

```
h.dt = 0.01
```

Enable variable step (allows error control):

```
h.CVode().active(True)
```

Set the absolute tolerance to e.g. 10^{-5} :

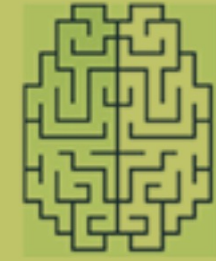
```
h.CVode().atol(1e-5)
```

Increase spatial resolution by e.g. a factor of 3 everywhere:

```
for sec in h.allsec(): sec.nseg *= 3
```



NeuroMorpho.Org



Version 8.1.13 - Released: 2021-07-02 - Content: 150307 neurons

Total number of downloads: 30296545

Total site hits since August 1, 2006: 3173903

[HOME](#)

[BROWSE](#)

[SEARCH](#)

[LITERATURE COVERAGE](#)

[TERMS OF USE](#)

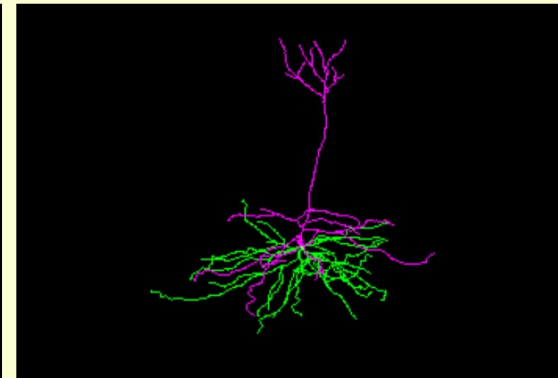
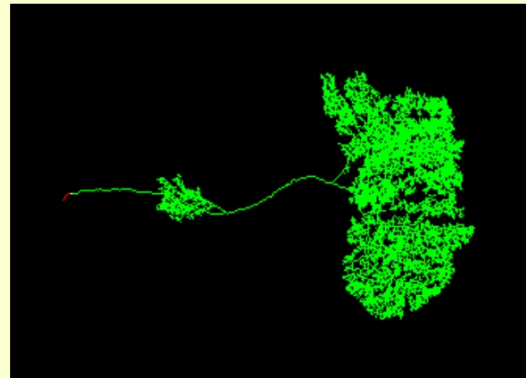
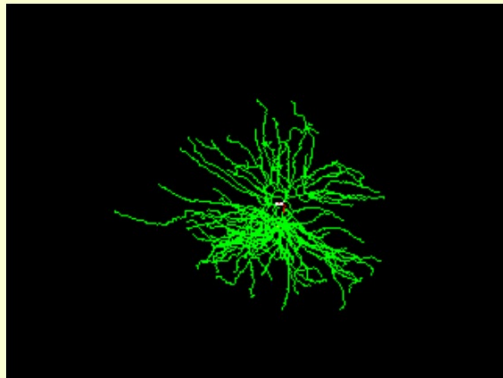
[HELP](#)



[Home > Homepage](#)

Contributions from 785 laboratories

Reconstructions from 953 celltypes



NeuroMorpho.Org is a centrally curated inventory of **digitally reconstructed**

150,307 reconstructions · 953 cell types · 384 brain regions



[Morphology File \(Standardized\)](#)

[Morphology File \(Original\)](#)

[Log File \(Standardized\)](#)

[Log File \(Original\)](#)

Get above files zipped

[3D Neuron Viewer - Java, legacy](#)

[3D Neuron Viewer - WebGL, novel Animation](#)

Standardized:
Always SWC

Original format:
Could be anything

Details about selected neuron

NeuroMorpho.Org ID : NMO_00227

Neuron Name : c91662

Archive Name : Amaral

Species Name : rat

Strain : Sprague-Dawley

Ascoli, G. A., Donohue, D. E., & Halavi, M. (2007). NeuroMorpho. Org: a central resource for neuronal morphologies. *Journal of Neuroscience*, 27(35), 9247-9251.

Metadata

NeuroMorpho.Org ID : NMO_00082
Neuron Name : n401
Archive Name : Turner
Species Name : rat
Strain : Fischer 344
Structural Domains : Dendrites, Soma, No Axon
Physical Integrity : Dendrites Complete
Morphological Attributes : Diameter, 3D, Angles
Min Age : 2.0 months
Max Age : 8.0 months
Gender : Male/Female
Min Weight : 200 grams
Max Weight : 350 grams
Development : young
Primary Brain Region : hippocampus
Secondary Brain Region : CA1
Tertiary Brain Region : Not reported
Primary Cell Class : principal cell
Secondary Cell Class : pyramidal
Tertiary Cell Class : Not reported
Original Format : CVAPP.swc
Experiment Protocol : in vivo
Experimental Condition : Control
Staining Method : biocytin
Slicing Direction : coronal
Slice Thickness : 80.00 µm
Tissue Shrinkage : Reported 25% in xy, 75% in z
Corrected 133% in xy, 400% in z
Objective Type : oil
Magnification : 100x
Reconstruction Method : Neurolucida
Date of Deposition : 2005-12-31
Date of Upload : 2006-08-01

Soma Surface : 903.25 µm²
Number of Stems : 7
Number of Bifurcations : 113
Number of Branches : 233
Overall Width : 363.7 µm
Overall Height : 717.18 µm
Overall Depth : 364.21 µm
Average Diameter : 1.16 µm
Total Length : 22216.3 µm
Total Surface : 84796.1 µm²
Total Volume : 30674.3 µm³
Max Euclidean Distance : 668.56 µm
Max Path Distance : 1893.37 µm
Max Branch Order : 25
Average Contraction : 0.7
Total Fragmentation : 5460
Partition Asymmetry : 0.56
Average Rall's Ratio : 1.78
Average Bifurcation Angle Local : 89.59°
Average Bifurcation Angle Remote : 75.23°
Fractal Dimension : 1.07

THE JOURNAL OF COMPARATIVE NEUROLOGY 391:335-352 (1998)

Dendritic Properties of Hippocampal CA1 Pyramidal Neurons in the Rat: Intracellular Staining In Vivo and In Vitro

G.K. PYAPALI,^{1,2} A. SIK,³ M. PENTTONEN,³ G. BUZSAKI,³ AND D.A. TURNER^{1,2,4*}

¹Department of Neurosurgery, Duke University, Durham, North Carolina 27710

²Durham Veterans Affairs Medical Center, Durham, North Carolina 27710

³Center for Molecular and Behavioral Neuroscience, Rutgers,

The State University of New Jersey, Newark, New Jersey 07102

⁴Department of Neurobiology, Duke University, Durham, North Carolina 27710

Not
everything
was made for
you

Not every morphology was reconstructed with the intent of being in a simulation.

Potential factors affecting the quality of the data:

- histology
 - staining, amputation, shrinkage
- physics
 - diameter
- spines

Before using a morphology found online, always read the associated paper(s) to make sure you understand any limitations of the reconstruction.

For example, why did they make this? Were they studying a disease (e.g. Alzheimer's) that alters morphology?

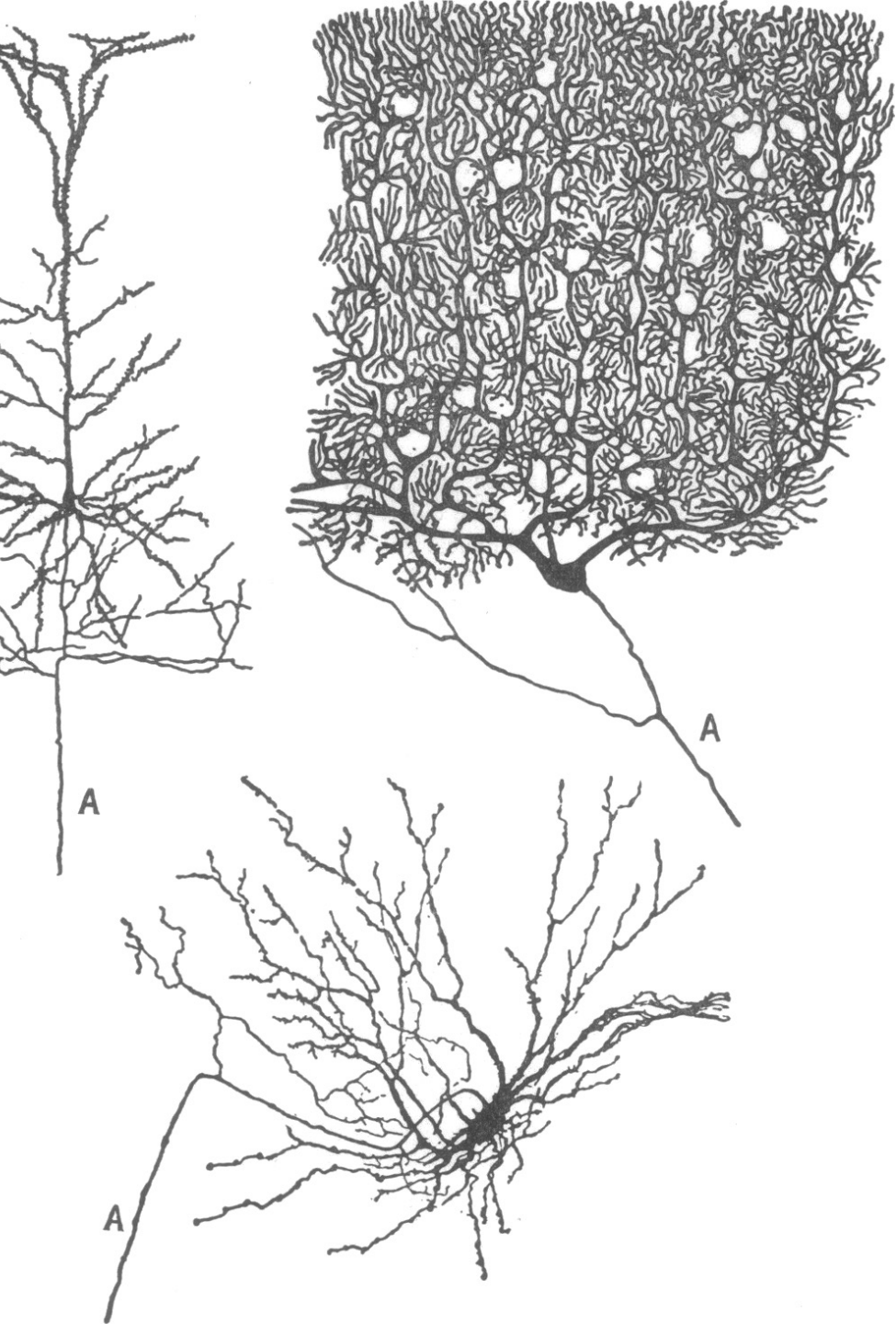
Qualitative tests

Look for orphan sections and bottlenecks.

Insert pas, set Ra and g_pas = pas.g low. Inject large depolarizing current at soma. Examine a PlotShape of v.

Look for z-axis drift and backlash.
Rotate the cell on a PlotShape and look for abrupt jumps.

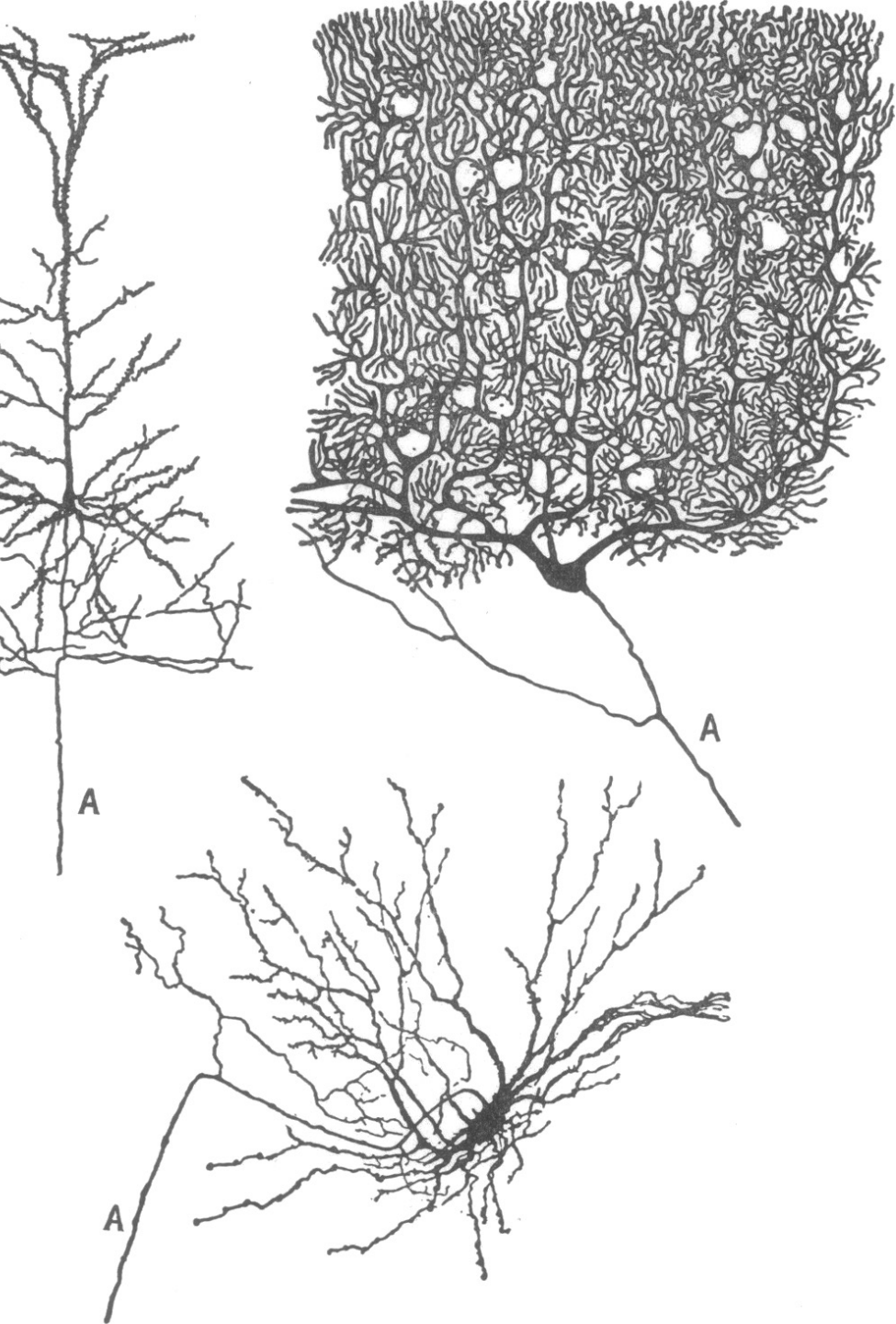
Are diameters constant or varying? Are they reasonable?



Loading Morphologies

```
from neuron import h
h.load_file('import3d.hoc')

cell = h.Import3d_SWC_read()
cell.input('filename.swc')
i3d = h.Import3d_GUI(cell, False)
i3d.instantiate(None) # or i3d.instantiate(self)
```



Plotting Morphologies

```
import plotly
```

```
ps = h.PlotShape(False)
```

```
ps.scale(-80, 40)
```

```
ps.variable('v')
```

```
ps.plot(plotly).show()
```

Does shape or
position
matter?

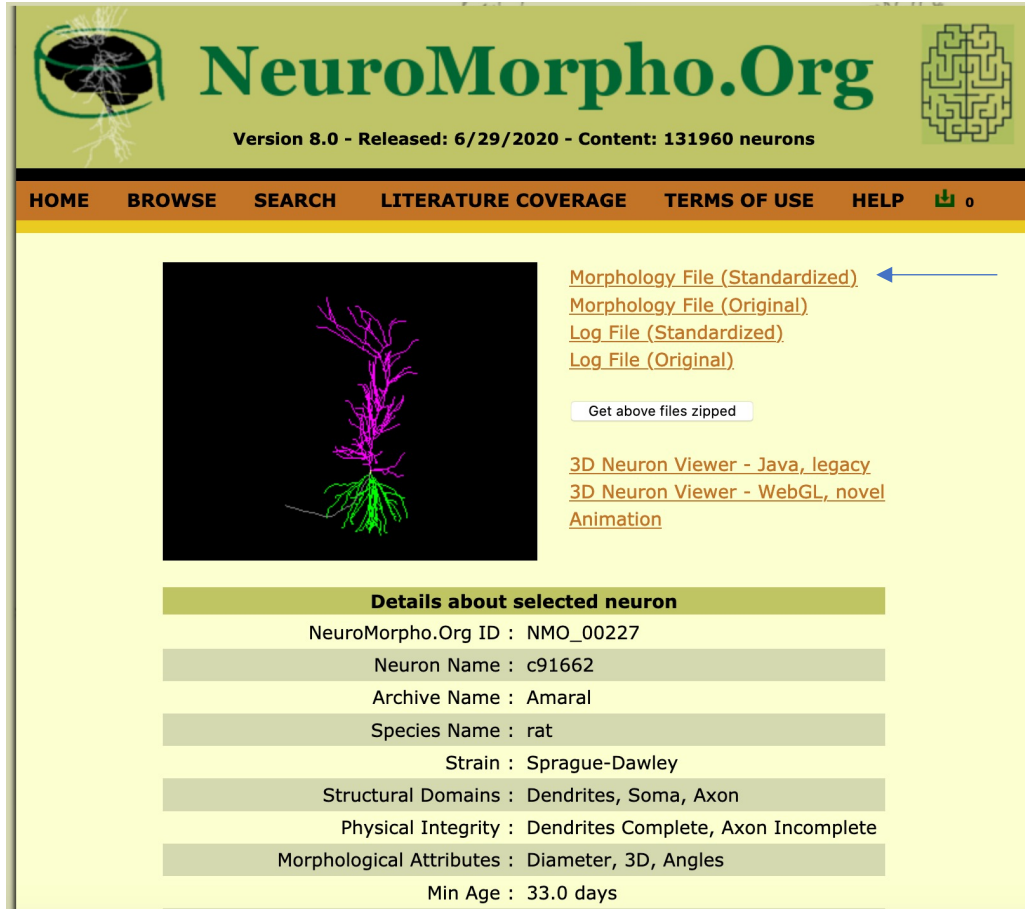
Sometimes.

They matter with:

- Connections based on proximity of axon to dendrite.
- Connections based on cell-to-cell proximity.
- Communicating about your model to other humans.
- Local field potentials.
- Extracellular stimulation.
- Extracellular diffusion.
- 3D intracellular chemical dynamics.

Ishizuka, N., Cowan, W. M., & Amaral, D. G. (1995). A quantitative analysis of the dendritic organization of pyramidal cells in the rat hippocampus. *Journal of Comparative Neurology*, 362(1), 17-45.

Example



The screenshot shows the NeuroMorpho.Org website interface. At the top, there is a logo and the text "NeuroMorpho.Org Version 8.0 - Released: 6/29/2020 - Content: 131960 neurons". Below this is a navigation bar with links for HOME, BROWSE, SEARCH, LITERATURE COVERAGE, TERMS OF USE, and HELP. The main content area features a 3D visualization of a neuron with a purple soma and dendrites, and a green axon. To the right of the visualization are links for "Morphology File (Standardized)", "Morphology File (Original)", "Log File (Standardized)", and "Log File (Original)". Below these links is a button that says "Get above files zipped". Further down are links for "3D Neuron Viewer - Java, legacy", "3D Neuron Viewer - WebGL, novel", and "Animation". A section titled "Details about selected neuron" contains the following information:

- NeuroMorpho.Org ID : NMO_00227
- Neuron Name : c91662
- Archive Name : Amaral
- Species Name : rat
- Strain : Sprague-Dawley
- Structural Domains : Dendrites, Soma, Axon
- Physical Integrity : Dendrites Complete, Axon Incomplete
- Morphological Attributes : Diameter, 3D, Angles
- Min Age : 33.0 days

```
# Original file c91662.swc edited using StdSwc version 1.31 on 11/10/13.
# Irregularities and fixes documented in c91662.swc.std. See StdSwc1.31.doc for more information.
#
# NeuroLucida to SWC conversion from L-Measure. Sridevi Polavaram: spolavar@gmu.edu
# Original fileName:C:\Users\praveen\Desktop\Uzma>ErrorArchives\ToBeProcessed\Amaral\asc\c91662.asc
#The original file has a single soma contour that is averaged into 3 soma points
# NEUROMANTIC V1.6.3 (10/18/2013 6:55:13 PM): Saved to c91662-T1.swc
1 1 0.0 0.0 0.0 8.8677 -1
2 1 1.13 8.71 1.2 8.8677 1
3 1 -1.13 -8.71 -1.2 8.8677 1
4 4 -1.86 11.06 -0.47 1.85 1
5 4 -1.94 19.75 -0.65 1.6 4
6 4 -2.52 31.1 -1.23 1.35 5
7 4 -2.94 39.91 -2.02 1.35 6
8 4 -2.55 49.45 -1.47 1.1 7
9 4 -2.61 56.49 -0.77 1.1 8
10 4 -1.17 70.15 -1.59 1.1 9
11 4 2.04 83.45 -1.43 1.1 10
12 4 1.89 91.65 -1.68 1.1 11
13 4 4.35 106.58 -1.57 1.1 12
14 4 5.09 115.06 -1.02 1.1 13
15 4 7.16 126.11 -1.93 1.1 14
16 4 7.13 129.63 -1.58 1.1 15
17 4 9.19 135.02 -2.02 1.1 16
18 4 11.82 145.77 -1.23 1.1 17
19 4 13.47 151.73 -1.73 0.9 18
20 4 14.65 157.05 -0.86 0.9 19
21 4 15.6 164.42 0.15 0.9 20
22 4 17.22 166.37 -0.76 0.9 21
23 4 17.27 175.11 -1.42 0.8 22
24 4 17.43 180.1 -0.87 0.8 23
25 4 18.41 192.2 -0.94 0.8 24
26 4 20.93 207.62 -0.75 0.8 25
27 4 22.64 214.07 -1.18 0.8 26
28 4 26.23 231.47 2.1 0.8 27
29 4 28.89 246.23 3.3 0.8 28
30 4 31.83 252.62 2.17 0.8 29
31 4 33.06 266.68 2.37 0.8 30
32 4 36.17 276.41 2.67 0.8 31
33 4 38.23 281.8 2.23 0.8 32
34 4 43.26 297.81 3.18 0.8 33
35 4 49.51 314.69 4.04 0.8 34
36 4 51.98 319.51 3.66 0.8 35
37 4 55.37 329.56 5.11 0.8 36
38 4 59.09 339.26 5.05 0.8 37
39 4 63.87 351.94 0.9 0.8 38
40 4 65.01 361.5 0.62 0.8 39
41 4 64.84 372.28 0.1 0.6 40
42 4 63.56 393.06 -1.8 0.6 41
43 4 63.28 401.93 -3.07 0.6 42
44 4 62.98 405.13 -3.87 0.6 43
45 4 61.56 411.24 -2.61 0.6 44
46 4 57.99 423.02 -3.47 0.6 45
```

<http://tinyurl.com/neuromorpho-c91662>

<http://tinyurl.com/neuromorpho-c91662-swc>

Colab time

- Load the cell from tinyurl.com/neuromorpho-c91662-swc, run a simulation, plot with PlotShape midway through a spike.
- Show a pseudo-line-scan path through it (use RangeVarPlot).
- Do the same thing showing a propagating wave at various time points.

```
[3] import plotly
```

```
[4] from neuron import h
     from neuron.units import mV, ms
     h.load_file('stdrun.hoc')
     h.load_file('import3d.hoc')
```

```
↳ 1.0
```

```
[5] cell = h.Import3d_SWC_read()
     cell.input('neuromorpho-c91662-swc')
     i3d = h.Import3d_GUI(cell, False)
     i3d.instantiate(None)
```

```
↳ One point section Import3d_Section[2] ending at line 10 has been removed
   One point section Import3d_Section[1] ending at line 9 has been removed
   0.0
```

```
[6] h.hh.insert(h.allsec())
     for sec in h.allsec():
         sec.nseg = 1 + 2 * int(sec.L / 40)
```

```
[7] ic = h.IClamp(h.soma[0](0.5))
     ic.amp = 10
     ic.delay = 1
     ic.dur = 1
```

```
[8] h.finitialize(-65 * mV)
     h.continuerun(3 * ms)
```

```
[11] fig = rvp.plot(plotly, name='t=3')
      h.continuerun(4 * ms)
      fig = rvp.plot(fig, name='t=4')
      h.continuerun(5 * ms)
      fig = rvp.plot(fig, name='t=5')
      fig.update_layout(
        axis_title='distance from center of soma ( $\mu\text{m}$ )',
        yaxis_title='membrane potential (mV)'
      )
      fig.show()
```

