

NEURON 2021 Online Course

An Introduction to Mechanistic Modeling
with NEURON

Instructors: Ted Carnevale, Robert McDougal,
and Adam Newton

Associate instructors: Mandy Hanes,
Joanna Jędrzejewska-Szmek,
and Kelvin Jones

Supported in part by NIH and NSF

The NEURON Simulation Environment

For models of neurons and neural circuits that are

- closely linked to experimental observations

and involve

- complex anatomical and biophysical properties
- electrical and/or chemical signaling

Features

- Performance
- Ease of use
- Active development
- User support
- Large user base

The NEURON Simulation Environment

Active development

- Open source project directed by Michael Hines at Yale
<https://github.com/neuronsimulator/nrn>
- Supported by NIH and European Human Brain Project

User support

- Downloads, documentation, tutorials at
<https://www.neuron.yale.edu/>
- Individual user support via Forum
<https://www.neuron.yale.edu/phpBB/>,
email, phone / Zoom / Skype etc.
- Courses: meetings (SFN, OCNS, other),
summer course, workshops, webinars

The NEURON User Community

Used by experimentalists, theoreticians, and educators
for neuroscience research and teaching

As of February 2021

- more than 2400 publications
- more than 2000 subscribers to Forum and mailing list
- source code for > 1600 published models
(> 750 using NEURON)
at ModelDB <https://modeldb.yale.edu/>

The What and the Why of Neural Modeling

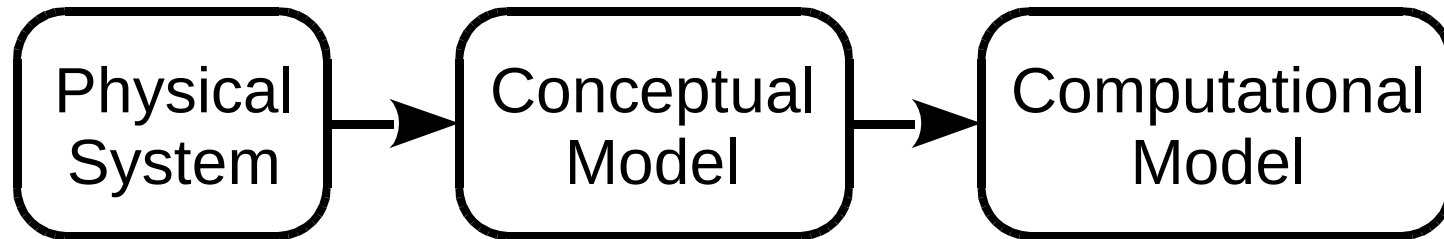
The moment-to-moment processing of information in the nervous system involves the propagation and interaction of electrical and chemical signals that are distributed in space and time.

Empirically-based modeling is needed to test hypotheses about the mechanisms that govern these signals and how nervous system function emerges from the operation of these mechanisms.

Topics

1. How to create and use models of neurons and networks of neurons
 - How to specify anatomical and biophysical properties
 - How to control, display, and analyze models and simulation results
2. How NEURON works
3. How to add user-defined biophysical mechanisms

From Physical System to Computational Model



Conceptual model:

simplified representation of the physical system

Computational model:

accurate representation of the conceptual model

*A close match between the conceptual model
and
the computational model
is essential ("conceptual control").*

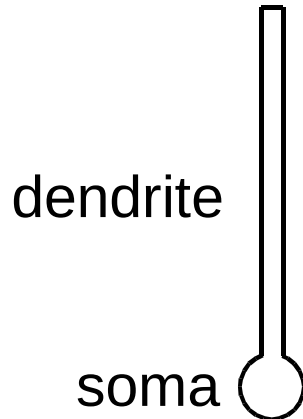
From Physical System to Computational Model

Physical
system



Ca1
pyramidal
cell

Conceptual
model



ball
and
stick

Computational
model

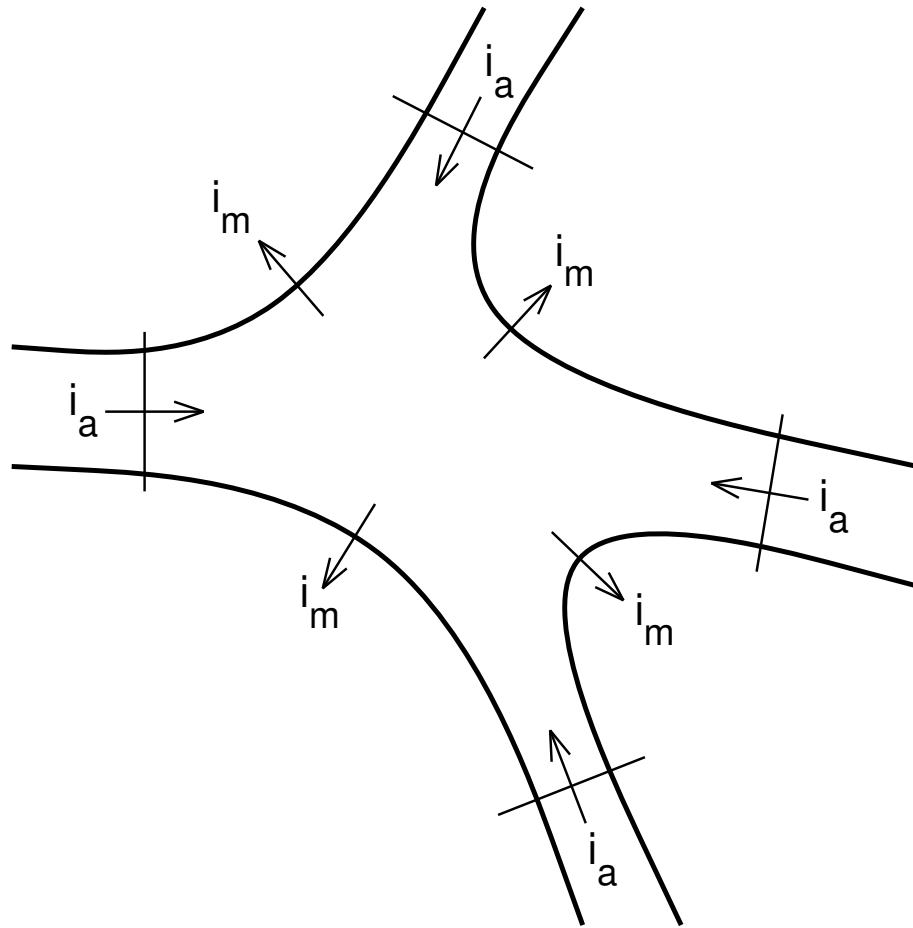
```
# python  
soma = h.Section(name='soma')  
dendrite = h.Section(name='dendrite')  
dendrite.connect(soma(1))
```

```
// hoc  
create soma, dendrite  
connect dendrite(0), soma(1)
```


Fundamental Concepts

Signals	What moves	Driving force	What is conserved
Electrical	charge carriers	voltage gradient	charge
Chemical	solute	concentration gradient	mass

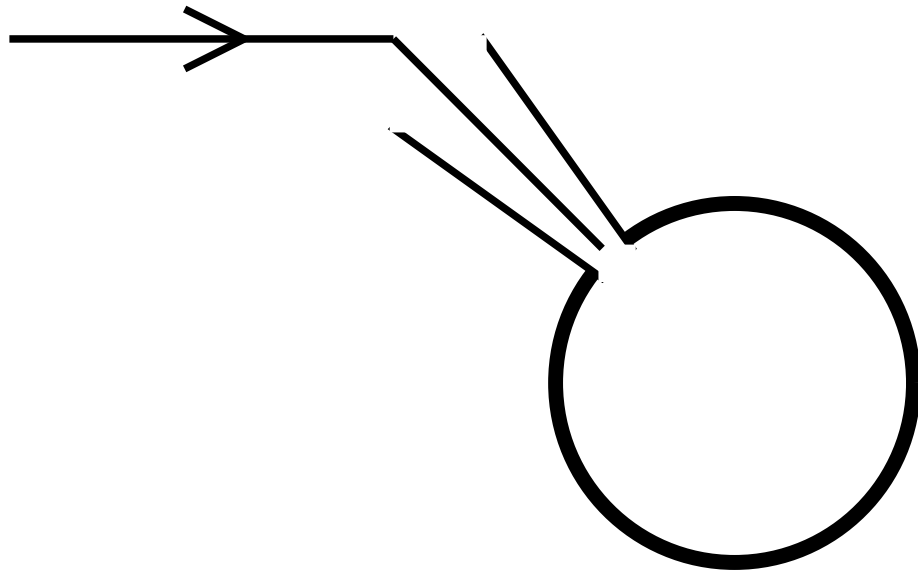
Conservation of Charge



$$C_m \frac{dV_m}{dt} + i_{\text{ion}} = \sum i_a$$

Electrically Compact Model

Injected
current



$$C_m \frac{dV_m}{dt} + i_{\text{ion}} = \sum i_a$$

Is NEURON 8.x installed and working?

In a terminal (user entries **bold**):

```
[ted@blitz Desktop]$ neurondemo
NEURON -- VERSION 8.0a-575-gd9462cc master (d9462cc) 2021-05-27
Duke, Yale, and the BlueBrain Project -- Copyright 1984-2021
See http://neuron.yale.edu/neuron/credits

loading membrane mechanisms from
/home/ted/bin/nrn/share/nrn/demo/release/x86_64/.libs/libnrnmech.
so
Additional mechanisms from files
"cabpump.mod" "cachan1.mod" "camchan.mod" "capump.mod"
"invlfire.mod" "khhchan.mod" "mcna.mod" "nacaex.mod" "nachan.mod"
"release.mod"

oc>>quit()
```

Can Python use NEURON as a module?

... and do you have Python 3.6?

In a terminal (user entries **bold**):

```
[ted@blitz Desktop]$ python3  
Python 3.6.8 (default, Nov 16 2020, 16:55:22)  
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux  
Type "help", "copyright", "credits" or "license" for more  
information.  
>>> from neuron import h, gui  
>>>
```

`import h, gui` should generate no error message,
and you should see the NEURON Main Menu toolbar.

```
>>> quit()
```

Exercise: Single Compartment Model

Start with a lipid bilayer (no channels)

Add linear ion channels (passive leak)

Use the GUI to:

- build a model and an interface for using it
- run simulations and analyze results
- change stimulus params (intensity and duration)
- adjust graphical displays of simulation results
- adjust simulation params dt and Points Plotted / ms

An aside: why the GUI?

Very convenient for interactive tasks--development, debugging, exploratory simulations (how well do you understand your own models?).

Graphical model specifications and presentations of simulation results are intuitive and easily understood.

Adding a custom user interface to your own programs makes it easier to collaborate with non-programmers.

Many GUI tools do things that would be hard to accomplish by writing your own code, e.g. CellBuilder, Channel Builder, Linear Circuit Builder, Impedance tools, VariableStepControl, ModelView, even the lowly Shape Plot.

The GUI always works--no typos, can't violate syntax.

The most powerful approach to using NEURON: combine the GUI and your own code to exploit the strengths of both.

Single Compartment *continued*

Open a terminal window.

In that window enter these commands:

```
cd Desktop  
mkdir simplecell  
cd simplecell
```

Now you have a place to work.

Start Python and tell it to use the neuron module.

```
python3
```

At python's `>>>` prompt enter this command:

```
from neuron import h, gui
```


Single Compartment *continued*

Get a CellBuilder by clicking on

Neuron Main Menu / Build / CellBuilder

Set soma's surface area to 100 μm^2 .

CellBuilder / Geometry tab

Click on **soma**

Click on the **area** checkbox.

Toggle **Specify Strategy** off.

Make sure numeric field next to the
area(μm^2) button says **100**.

Single Compartment *continued*

Make sure that the soma has membrane capacitance, and that $cm = 1 \text{ uF/cm}^2$.

CellBuilder / Biophysics tab

Click on **soma**

Click on the **cm** checkbox to toggle it on.

The other checkboxes (Ra, pas, extracellular, hh) should be off.

Toggle **Specify Strategy** off.

Make sure the numeric field next to the **cm (uF/cm²)** button says 1.

Single Compartment *continued*

Before doing anything else, save your work. Click on

Neuron Main Menu / File / save session

Popup window--enter **simplecell.ses**

Click on **Save** button to write a file called

simplecell.ses in your **Desktop/simplecell** directory. You can use this file to recreate your configured CellBuilder.

Single Compartment *continued*

Next click on the **Continuous Create** button. Your computer's memory now contains a model cell with the properties specified by the CellBuilder.

Let's do some experiments on this model.
You'll need:

a RunControl panel to launch simulations

NMM/Tools/RunControl

a graph of soma membrane potential vs. time

NMM/Graph/Voltage axis

Single Compartment *continued*

a current clamp ("IClamp") that injects a 1 ms x 1 nA current pulse that starts 1 ms into the model cell

**NMM/Tools/Point Processes/Managers/
Point Manager**

Make this be a current clamp

**PointProcessManager/SelectPointProcess/
IClamp**

Show the IClamp's parameters

PPM/Show/Parameters

Set **delay(ms)**, **dur(ms)**, and **amp(nA)** to **1**.

Single Compartment *continued*

Plot stimulus current vs. time.

NMM/Graph/Current axis

Make it plot the IClamp's i variable.

Click on the **Graph's menu button** (left upper corner of its canvas) and select "**Plot what?**"

Click in the variable name browser's edit field and enter

IClamp[0].i

then click

Accept

Single Compartment *continued*

Before you do anything else

1. save your work to a session file. You might call this **cell1rig.ses**
2. on a sheet of paper sketch what you think the plots of somatic membrane potential and IClamp[0].i will look like.

Then launch a simulation by clicking on
RunControl / Init & Run

In the RunControl panel note the simulation parameters **dt**, **Tstop**, and **Points plotted/ms**.
What happens if you change these?

Single Compartment *continued*

Things to do:

Quit Python (enter `quit()` at the `>>>` prompt, or click on **NMM / File / Quit**).

Start Python and load the session file you saved earlier.

```
[ted@blitz simplecell]$ python3
>>> from neuron import h, gui
>>> h.load_file('cell1rig.ses')
>>>
```

The model cell and custom user interface you saved are ready to be used.

Single Compartment *continued*

Things to do:

Use "**View = plot**" to rescale Graph axes.

Maximum soma.v too big? Divide IClamp.amp by 10 and try again.

What IClamp.amp depolarizes the model by 10 mV?

Insert passive channels and see how this affects the model.

In the CellBuilder click on **Biophysics**, then toggle **Specify Strategy** on. Click on the **pas** button so it shows a check mark.

Single Compartment *continued*

What does the pas mechanism do to the model cell's resting potential?

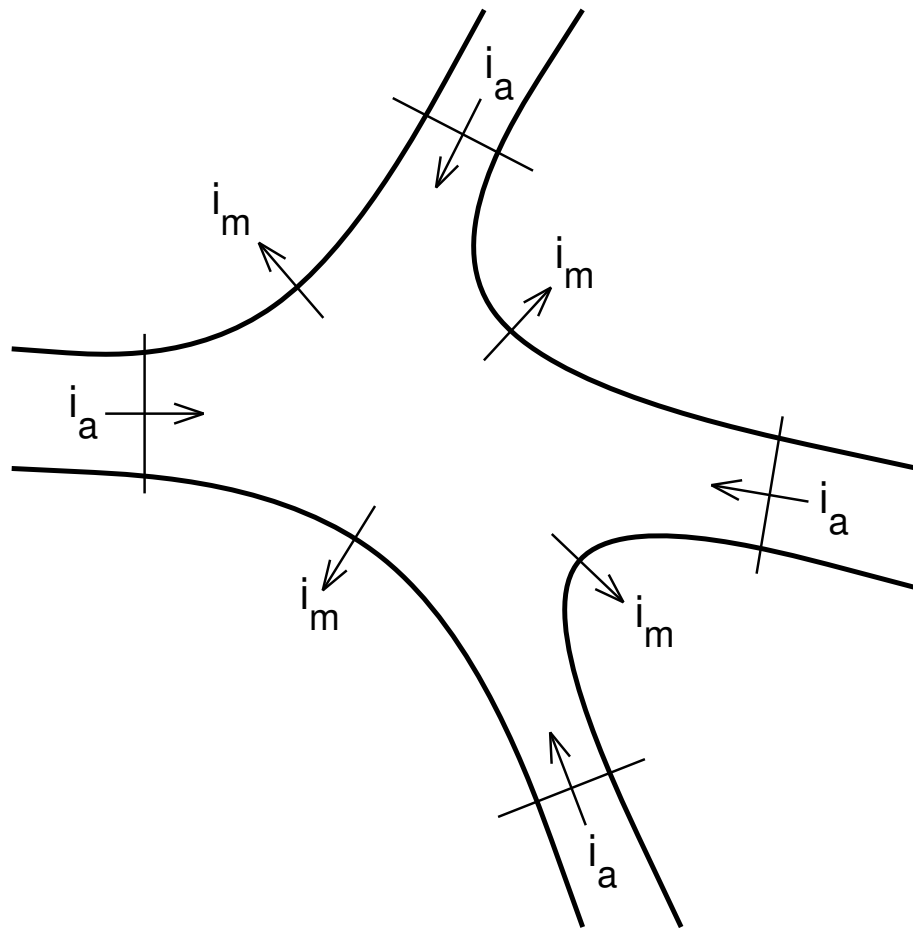
Is it useful to initialize a model to its resting potential?
If yes, how can you make this happen?

Change **IClamp.amp** to 1 A (1e9 nA) and run a simulation. Rescale the graphs if necessary. What would happen to a real cell?

Change **IClamp.amp** to 1 nA and dur to 0.01 ms.
What happens and why?

Change **Points plotted/ms** to 100 and try again.

Conservation of Charge



$$C_m \frac{dV_m}{dt} + i_{\text{ion}} = \sum i_a$$

Models with Significant Electrical Extent

$$c_j \frac{dv_j}{dt} + i_{ion_j} = \sum_k \frac{v_k - v_j}{r_{jk}}$$

v_j membrane potential in compartment j

i_{ion_j} net transmembrane ionic current in compartment j

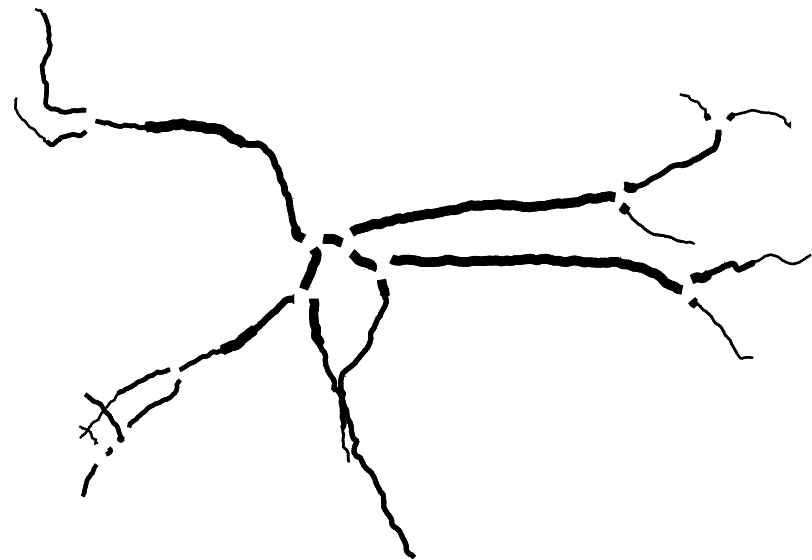
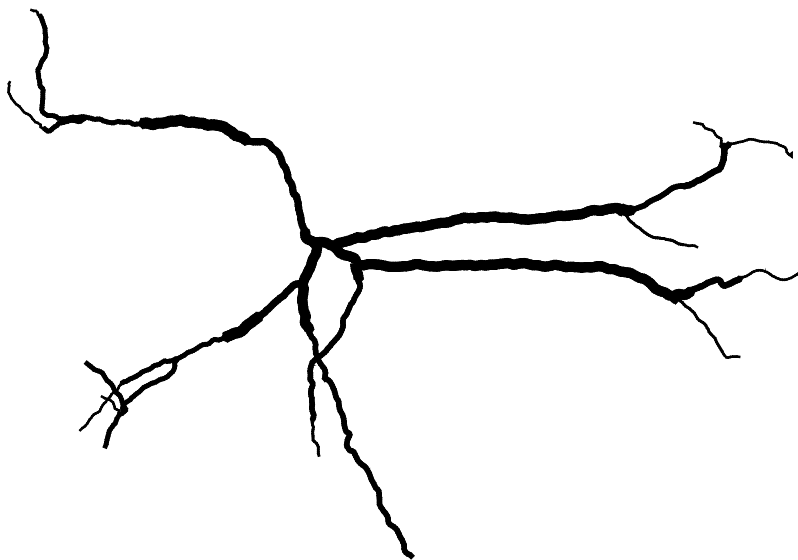
c_j membrane capacitance of compartment j

r_{jk} axial resistance between the centers of
compartment j
and
adjacent compartment k

Separating Anatomy and Biophysics from Purely Numerical Issues

section

a continuous length of unbranched cable



Anatomical data from A.I. Gulyás

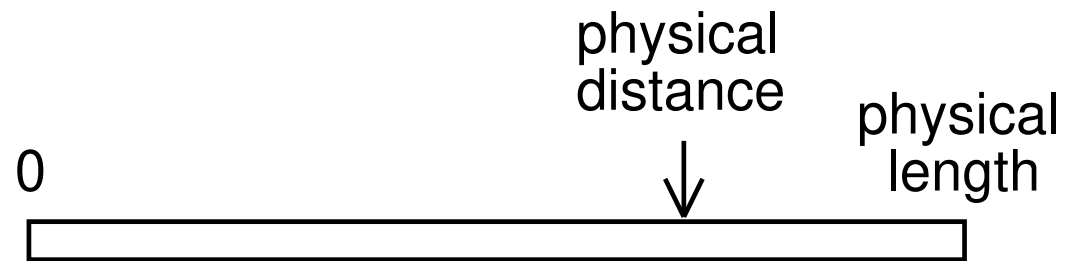
Range Variables

Name	Meaning	Units
diam	diameter	[μm]
cm	specific membrane capacitance	[$\mu\text{f}/\text{cm}^2$]
g_pas (hoc) pas . g (Python)	specific conductance of the pas mechanism	[siemens/ cm^2]
v	membrane potential	[mV]

range

normalized position along the length of a section

$$0 \leq \text{range} \leq 1$$



Syntax:

secname(range).rangevar

Translation: "in *secname*

at the location corresponding to *range*

access the value of *rangevar*"

Examples:

```
# v at middle of dend
```

```
dend(0.5).v # shortcut: dend.v
```

```
# at each point in dend
```

```
# where v is calculated
```

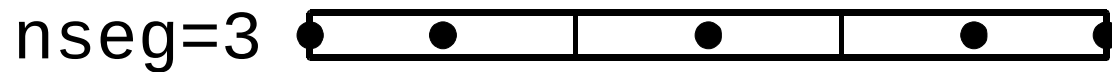
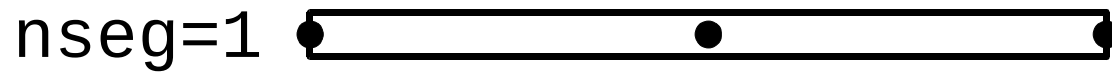
```
# print range, anat distance, and v
```

```
for seg in dend.allseg():
```

```
    print(seg.x, seg.x*dend.L, dend(seg.x).v)
```


nseg

the number of points in a section at which
the discretized cable equation is integrated



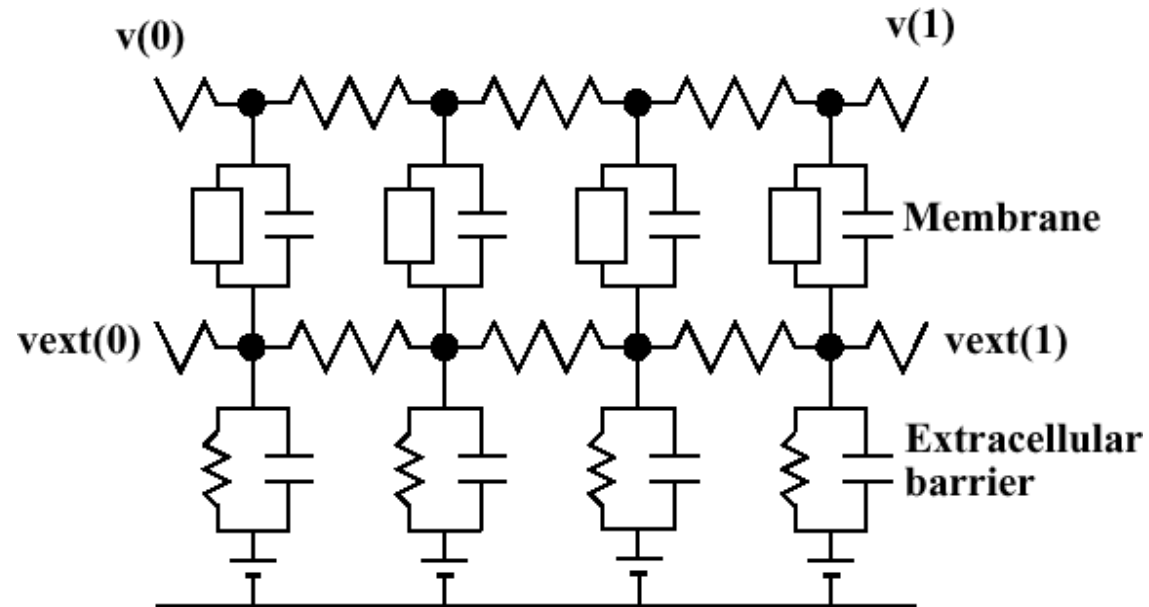
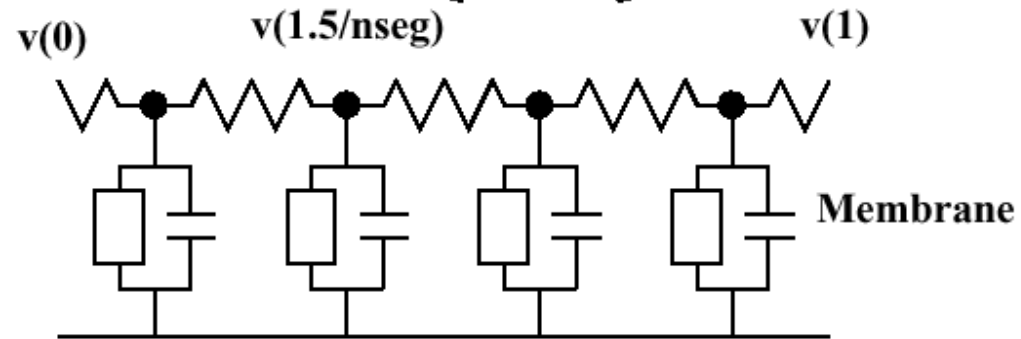
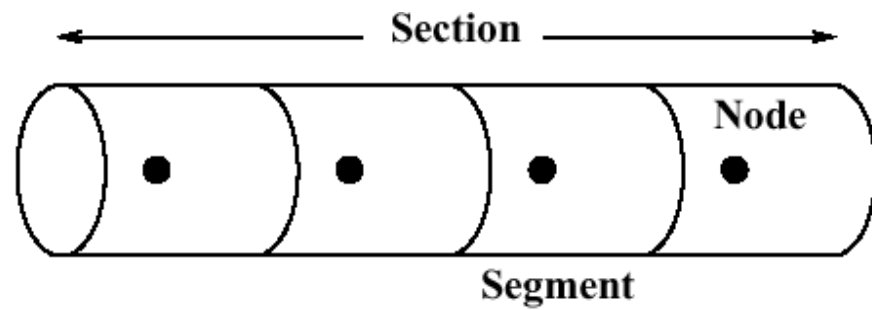
Example: `axon.nseg = 3`

To test spatial resolution

```
for sec in h.allsec():
```

```
    sec.nseg *= 3
```

and repeat the simulation



Category	Variable	Units
Time	t	[ms]
Voltage	v	[mV]
Current		
specific	i	[mA/cm ²] (distributed)
absolute		[nA] (point process)
Capacitance		
specific	cm	[uF/cm ²]
absolute		[nF] (point process)
Length	diam, L	[um]
Conductance		
specific	g	[S/cm ²] (distributed)
absolute		[uS] (point process)
Cytoplasmic resistivity	Ra	[ohm cm]
Resistance	ri()	[10 ⁶ ohm]
Concentration	na _i etc.	[mM]

Physical System



From <http://www.mbl.edu/>

Conceptual Model

Hodgkin-Huxley cable equations

$$\frac{D}{4R_a} \frac{\partial^2 V}{\partial x^2} = C_m \frac{\partial V}{\partial t} + \bar{g} m^3 h \cdot (V - E_{na}) + \bar{g}_k n^4 \cdot (V - E_k) + g_l \cdot (V - E_l)$$

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

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

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

Conceptual Model

Hodgkin-Huxley cable equations

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

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

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

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

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

Computational implementation

Python for NEURON

```
axon = h.Section(name = 'axon')
axon.L = 2.0e4
axon.diam = 100.0
axon.nseg = 43
axon.insert('hh')
```

Conceptual Model

Hodgkin-Huxley cable equations

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

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

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

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

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

Computational implementation

Python for NEURON

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

```
axon.L = 2.0e4
```

```
axon.diam = 100.0
```

```
axon.nseg = 43
```

```
axon.insert('hh')
```

Conceptual Model

Hodgkin-Huxley cable equations

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

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

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

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

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

Computational implementation

Python for NEURON

```
axon = h.Section(name = 'axon')
axon.L = 2.0e4
axon.diam = 100.0
axon.nseg = 43
axon.insert('hh')
```

Conceptual Model

Hodgkin-Huxley cable equations

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

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

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

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

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

Computational implementation

Python for NEURON

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

```
axon.L = 2.0e4
```

```
axon.diam = 100.0
```

```
axon.nseg = 43
```

```
axon.insert('hh')
```


Example: Squid Axon

For this exercise create a new folder called axon.

Use a CellBuilder to create a model with a single section called axon that has these properties:

L 2e4 um

diam 100 um

nseg 43

membrane has the **hh** mechanism

Save configured CellBuilder to a file called **hhaxon.ses**

Squid Axon *continued*

Use the GUI to build an interface that includes:

- a RunControl panel to launch simulations
- a PointProcessManager configured as an IClamp attached to the 0 end of the axon
- a voltage axis graph that shows membrane potential at axon(0.5)
- a space plot that plots v vs. distance along the axon
- a Movie Run tool to launch simulations that show a smooth evolution of the space plot over time

NEURON Main Menu

Iconify

File Build Tools Graph Vector Window Help

RunControl

Close Hide

Init (mV) ← -65

Init & Run

Stop

Continue til (ms) ← 5

Continue for (ms) ← 1

Single Step

t (ms) 5

Tstop (ms) 5

dt (ms) 0.025

Points plotted/ms 40

Scrn update invl (s) 0.05

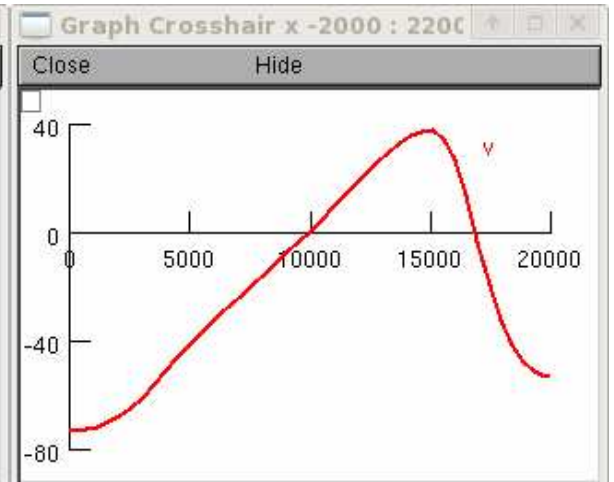
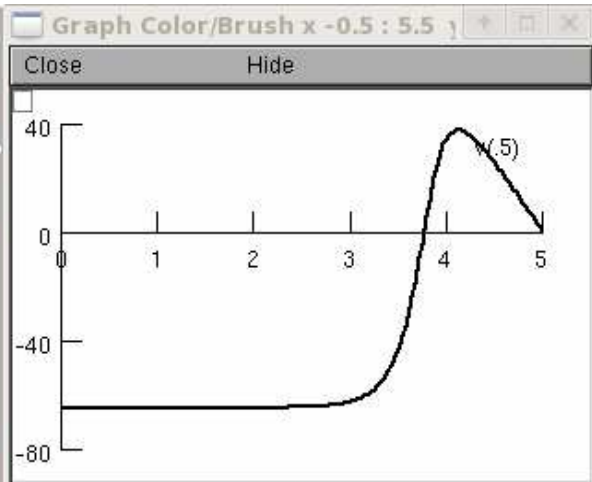
Real Time (s) 0.01

Movie Run

Close Hide

Init & Run

Seconds per step (s) 0.01



PointProcessMa

Close Hide

SelectPointProcess

Show

IClamp[0]
at: axon(0)

IClamp[0]

del (ms) 1

dur (ms) 1

amp (nA) 200

i (nA) 0

Shape Space Plc

Close Hide

v

Squid Axon *continued*

Use a current pulse of 0.1 ms duration to trigger a spike that starts at the 0 end of the axon.

Adjust the pulse amplitude to find the spike threshold to two place precision.

Set the pulse amplitude to $2 \times \text{threshold}$. Measure spike amplitude and half-width at axon(0.5).

Cut sodium channel density in half. What happens to spike amplitude and half-width? Repeat until **gnabar** is so low that you don't get a spike. What is the smallest gnabar needed to produce a spike?

Squid Axon *homework 1*

Use a text editor to create a file called axonrig.py

This file should contain the following commands:

```
from neuron import h, gui  
load_file('hhaxon.ses')
```

Save the file and exit the text editor.

At the system prompt execute the command

```
python3 -i axonrig.py
```

Your model axon and its user interface should be ready for you to use.

Squid Axon *homework 2*

At the system prompt execute the command
`python3 -i axonrig.py`

Attach a second IClamp to the 1 end of the axon.

Set both IClamps to deliver a 2*spike threshold stimulus at 1 ms and run a simulation.

What happens, and why?

Squid Axon *homework 3*

At the system prompt execute the command
`python3 -i axonrig.py`

Use the CellBuilder to change `nseg` to 15 and run a simulation.

What happens and why?

Using the original value of `nseg`, determine conduction velocity over the middle half of the axon.

Squid Axon *homework 4*

At the system prompt execute the command
`python3 -i axonrig.py`

Change `IClamp.dur` to 10 ms and adjust amp to force `axon(0).v` to -80 mV. What happens after the current stops? Why?