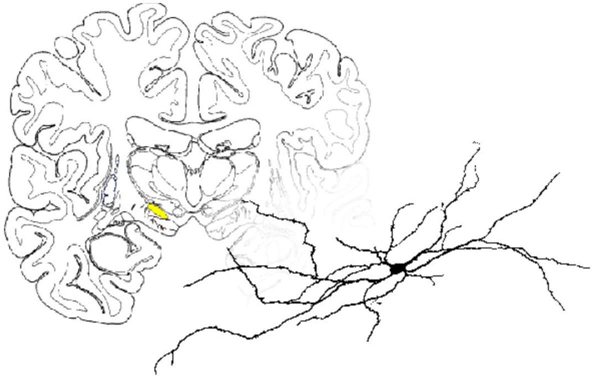
NEURON

Lesson 1

What is NEURON ?

- A neuron modeling and simulation environment that we use in order to model our biologically realistic neurons.
- We can make everything from the very simple to the very complex
- It works with either HOC or python
- It has a lot of useful properties which we will dive into!
- It is our platform of choice due to its compatibility with Sim4Life



NEURON Continued

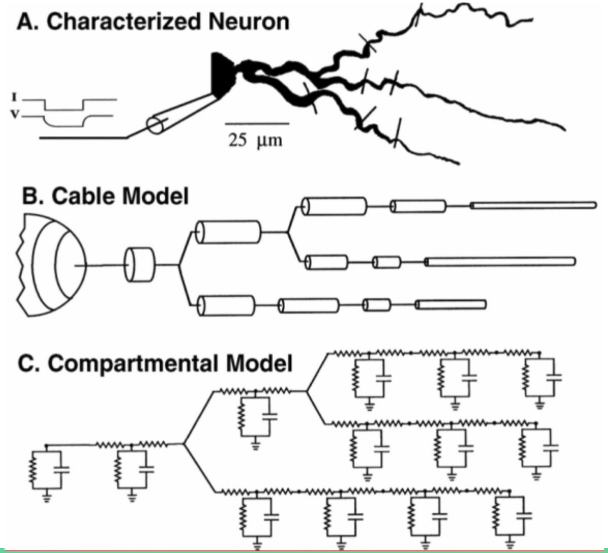
- Hoc is the programming language that we will mostly focus on
- It is based on C
    - This does not mean you need to know C to succeed
- Some of the things that NEURON allows us to do
    - Analyze complex branching morphology, multiple channel types, inhomogeneous channel distribution, ionic diffusion, the effects of second messengers and much more!
- Has a easy to use GUI, and it is object-oriented which allows for more complex analysis and optimization

How do we make these models?

Compartmentalization!

Compartmentalization

- We compartmentalize to simplify the really complex
- The neurons that we model are biologically realistic neurons
- Our goal is to keep the surface area as accurate as possible as this plays a role in EM properties



Things that are considered when modeling

- Time vs accuracy
- Morphology vs electrophysiology
- Large number of simulations vs smaller number of simulations

How do we do this in HOC?

“create soma” is one of the first things you always do

- This soma has a generic list of properties it is given but the main ones we worry about are
  - Nseg, diam, L, Ra
- You use *dot* notation in order to access
  - Ex. soma.diam = 10 /* diameter is now 10 nanometers*/

```
soma.nseg = 1
soma.diam = 18.8
soma.L = 18.8
soma.Ra = 123.0
```

HOC Continued

We can also use all the Hodgkin Huxley properties we discussed last week.

“soma insert hh”

- gnabar_hh: The maximum specific sodium channel conductance [Default value = 0.120 S/cm^2]
- gkbar_hh: The maximum specific potassium channel conductance [Default value = 0.036 S/cm^2]
- gl_hh: The maximum specific leakage conductance [Default value = 0.0003 S/cm^2]
- ena: The reversal potential for the sodium channel [Default value = 50 mV]
- ek: The reversal potential for the potassium channel [Default value = -77 mV]
- el_hh: The reversal potential for the leakage channel [Default value = -54.3 mV]

HOC: Point Processes

- Comes from statistics: A collection of mathematical points randomly located on some underlying mathematical space such as a real line, Cartesian Plane etc...
  - We use it to generate a electrical signal at a specific spot
    - I.e. a synapse or a clamp
- How to do it in HOC

```
stim = new IClamp(0.5)
```

or with the section name:

```
soma stim = new IClamp(0.5)
```

Our limitations so far

We have only discussed material in terms of one dendrite, we are now going to expand to numerous dendrites!

In HOC

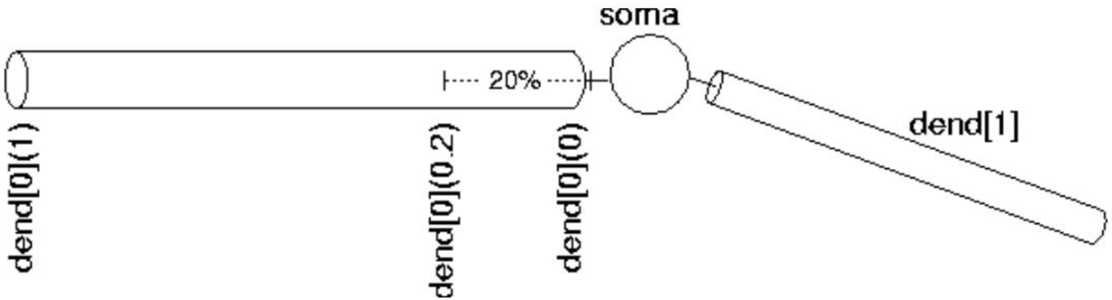
- We are going off of the cable model that was explained earlier
  - A good understanding of arrays will be needed from here on out, any questions?
- Every section of a dendrite that might be defined will have to be given specific properties, as shown below
- Each dendrite also has to be connected back to the home, or to a previous segment

```
dend[0] {
    nseg = 1
    diam = 3.18
    L = 701.9
    Ra = 123
    insert pas
}
```

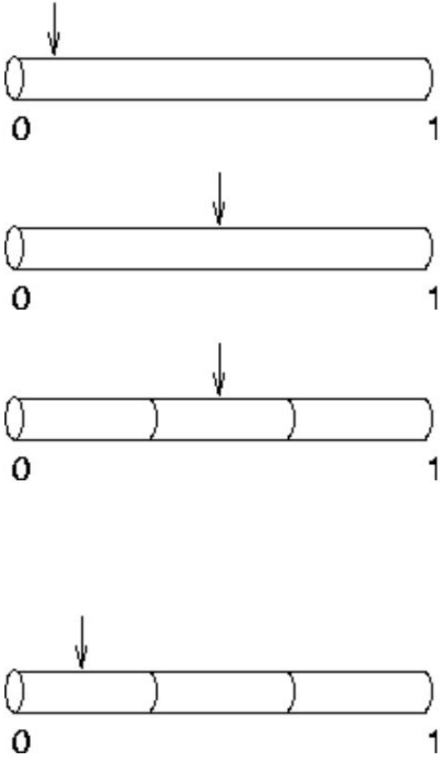
```
ndend = 2
create soma, dend[ndend]
```

Attachments in HOC

- For our purposes we will work on a tree data structure assumption
  - What does that mean?
- As a result we need to take into consideration angles
  - Ex. dend[0](0.2) means a 20% down dendrite 0 as shown below
- ALWAYS FINISH YOUR NSEG BEFORE ATTACHING POINT PROCESSES



What happens to Point Processes?



End

- This is a basic introduction to Computational Neuroscience with HOC and it should set you rolling on your homework!
- “From a small seed a mighty trunk may grow.” -Aeschylus