NEURON – tutorial B of Gillies & Sterratt

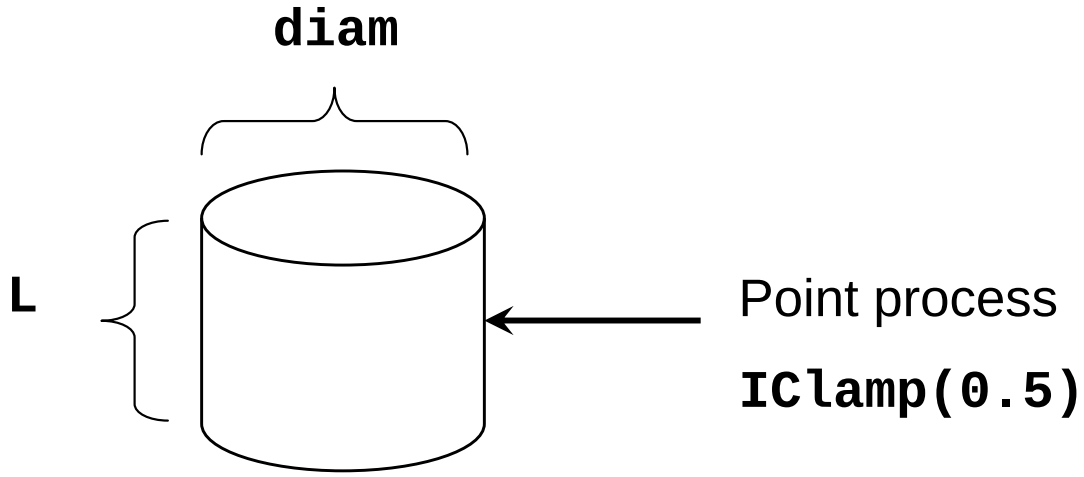
http://www.anc.ed.ac.uk/school/neuron/

Lubica Benuskova

COSC422 – lecture 5

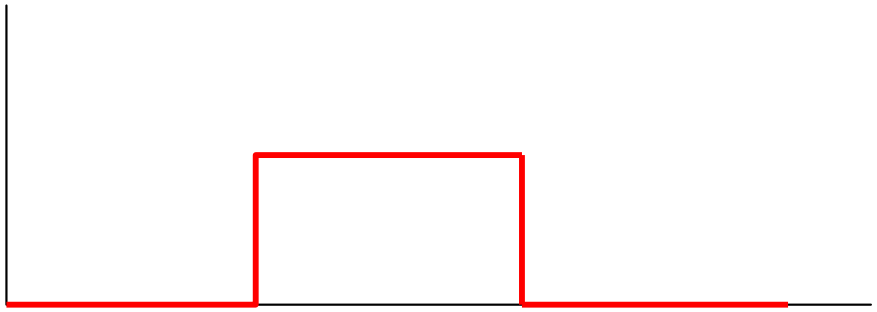
The program `sthA.hoc` so far

```
create soma
access soma

soma {
        nseg = 1
        diam = 18.8
        L = 18.8
        Ra = 123.0
        insert hh
}

objectvar stim
stim = new IClamp(0.5)

stim.del = 100
stim.dur = 100
stim.amp = 0.1

tstop = 300
```

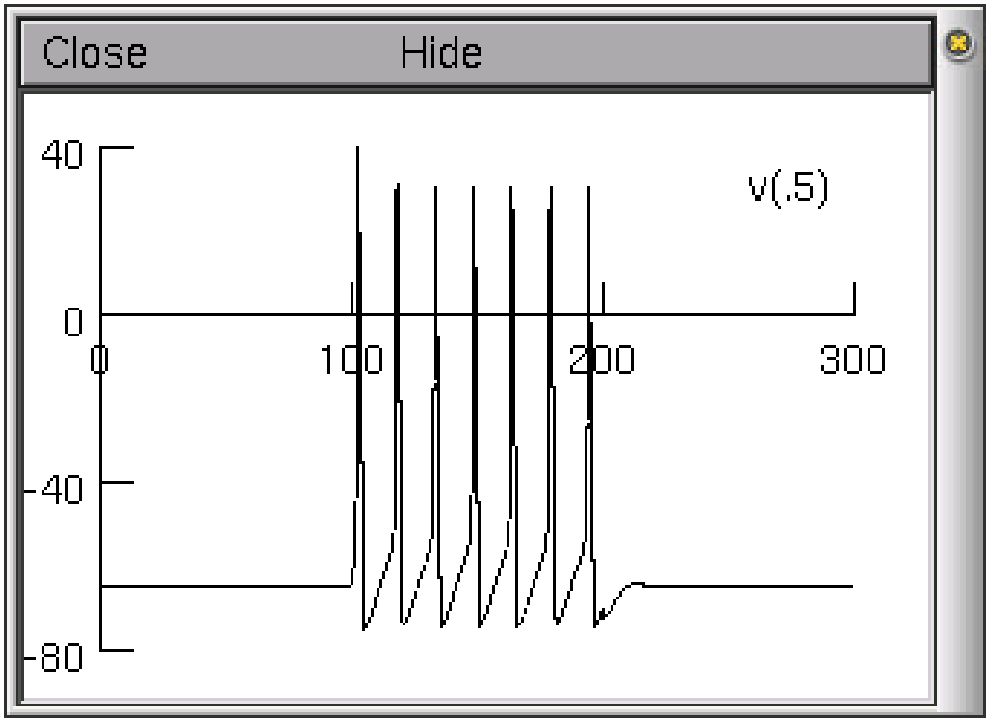


Modelling result so far

- The rectangular current pulse is injected in the middle of the soma section. The current injection starts at 100 ms and lasts 100 ms.



- Injected current causes soma to fire series of 7 consecutive spikes at regular intervals.

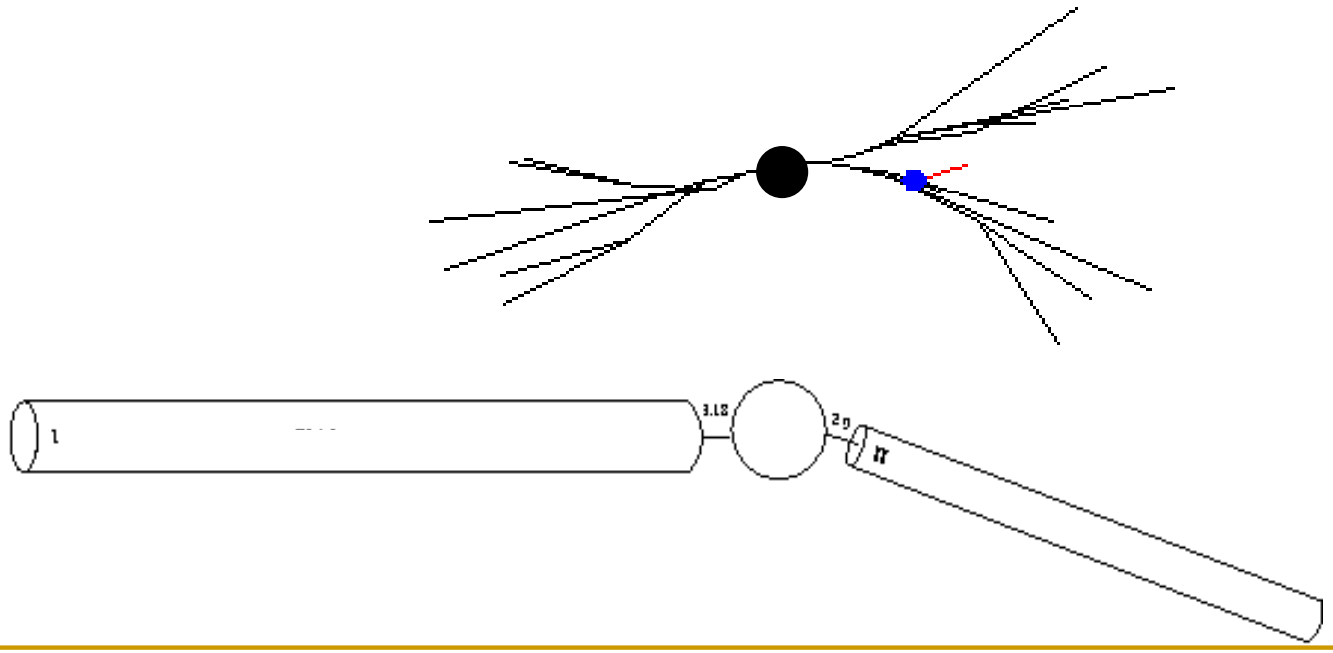


- What will happen when we add dendrites?

Today's plan: adding the dendrites

- Wilfrid Rall showed that it's possible to computationally replace the whole dendritic tree with the so-called equivalent cylinder.

- The subthalamic neuron has two main dendritic trees, so we represent each of them with an equivalent cylinder and add 2 dendrites.



Create dend[number_of_denrites]

- So far, we have only created a single section representing the soma. We are now going to add two more sections representing dendrites. These can be created just as the soma:

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

- Each section listed after the create command is separated by a comma.

- We use an array to represent the dendrites.

- The array indices are numbered from zero, so to access the first element of the array of dendrites, you need to use **dend[0],** and to access the second element, you need to use **dend[1],** etc.

Create dend[number_of_denrites]

- In the future, if we want to re-use this code for another neuron that contains, for example, 4 dendritic trunks, we will have to change all of the 2's to 4's in the code, i.e.:

```
create soma, dend[4]
```

- We can do this more elegantly by using a variable to represent the number of dendrites. Then we just reference the maximum number of dendrites throughout the program through this variable. For example:

```
nden = 4
create soma, dend[nden]
```

- To represent 2 dendrites, we just assign **`nden = 2.`**

Assignment of parameters of dendrites

- The two main dendritic trees are different, so their equivalent cylinders will have different parameter values.

- Let's define parameters of the first dendrite **dend[0]** :

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

`nseg = 1` or more?

- In NEURON, the membrane potential does not change along one segment. What we want to model, however, is how the membrane potential decays along the dendritic tree.
- To achieve decay of potential in the dendrite, we increase the number of segments in the section. But to how many?
- This is very important for accurate simulations. If the section's number of segments is too small, then the spatial resolution is low and you might not be able to see the fine details in the simulation results.
- If the section's number of segments is too large, then you might be unnecessarily extending the simulation time.
- By analysing the effects of different values of **`nseg`** on the results, you can achieve efficient and accurate simulation results.

Increasing the number of segments

- We are interested in seeing the potentials travel along the dendrites, so we want to increase the number of segments. So, we create two dendrites with five segments each:

```
dend[0] {
        nseg = 5
        diam = 3.18
        L = 701.9
        Ra = 123
        insert pas
    }
```

```
dend[1] {
        nseg = 5
        diam = 2
        L = 549.1
        Ra = 123
        insert pas
    }
```

Changing *passive* membrane properties

- When we add a new membrane mechanism (like **`hh`** or **`pas`**) to a section, we add the new parameters and their default values to the section as well.

- For example, if we add passive channels to the section by **`insert pas`**, we introduce two new properties to the section: **`g_pas`** (passive membrane conductance [S/cm^2]) and **`e_pas`** (reversal potential [mV]).

- For our simulation, neither the default Hodgkin-Huxley channels nor the passive properties will accurately model our neuron type.

Changing *passive* membrane properties

- We can modify the passive property parameters to reflect correctly the properties of rat subthalamic nucleus neurons based on experimental data simply by assigning their new values:

```
dend[0] {
            nseg = 5
            diam = 3.18
            L = 701.9
            Ra = 123
            insert pas
            g_pas = .0001667
            e_pas = -60.0

}
```

```
dend[1] {
            nseg = 5
            diam = 2
            L = 549.1
            Ra = 123
            insert pas
            g_pas = .0001667
            e_pas = -60.0

}
```

Changing *passive* membrane properties

- We need to modify the passive membrane parameters **gl_hh** and **el_hh** at the soma as well to model the subthalamic neurons, i.e.:

```
soma {
        nseg = 1
        diam = 18.8
        L = 18.8
        Ra = 123.0
        insert hh
        gl_hh = .0001667
        el_hh = -60.0

}
```

Changing *active* ion channel properties

- To modify the active channel properties we will need to build new active channels using the model description language (NMODL).

- For now we will just modify the maximum sodium conductance:

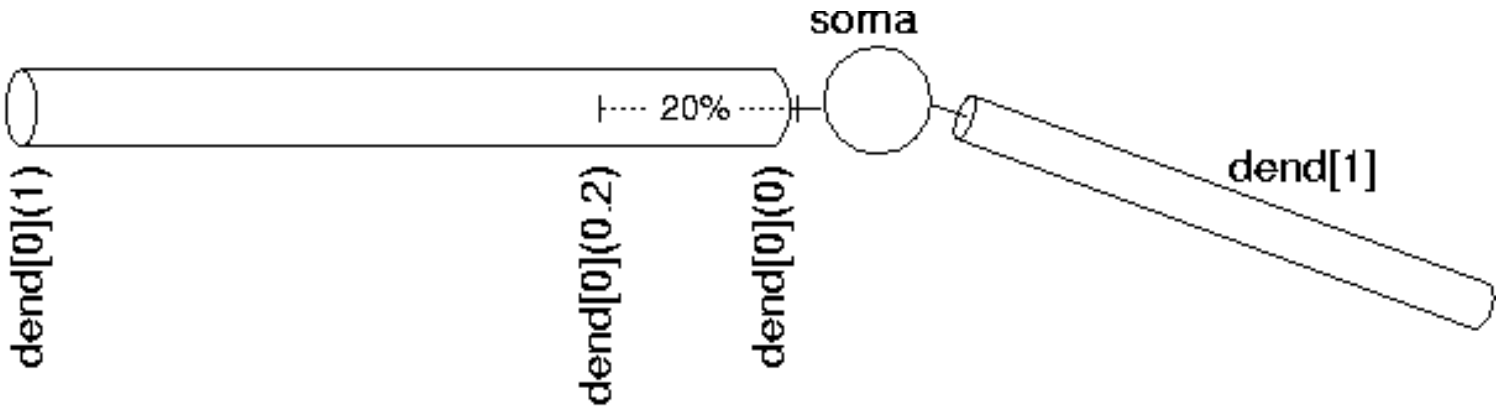
```
soma {
        nseg = 1
        diam = 18.8
        L = 18.8
        Ra = 123.0
        insert hh
        gnabar_hh = 0.25
        gl_hh = .0001667
        el_hh = -60.0

}
```

Connecting the sections

- In order to connect the sections together, we need a way to refer to specific points along the section, where to connect them.
- This is handled by using the section name followed by a number between 0 and 1 in parenthesis, which represents the distance along the section you want to refer to.
- For example, **`dend[0](0)`** refers to a point at the proximal end of dendrite 0, **`dend[0](1)`** refers to a point at the distal end of dendrite 0.

Connecting the sections

- Command **`dend[0](0.2)`** refers to a point 20% down dendrite 0 (going from soma).



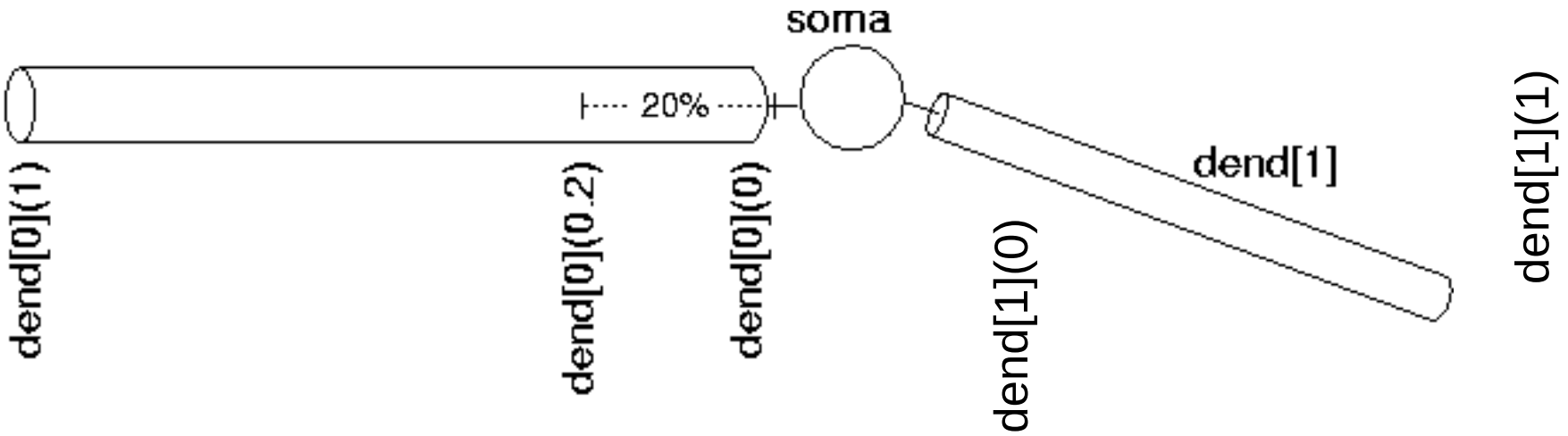
- We use this notation when connecting sections to one another since we need to specify what point of one section connects to what point of another section.

- We also use this notation when we place point processes in a section.

Connecting the sections

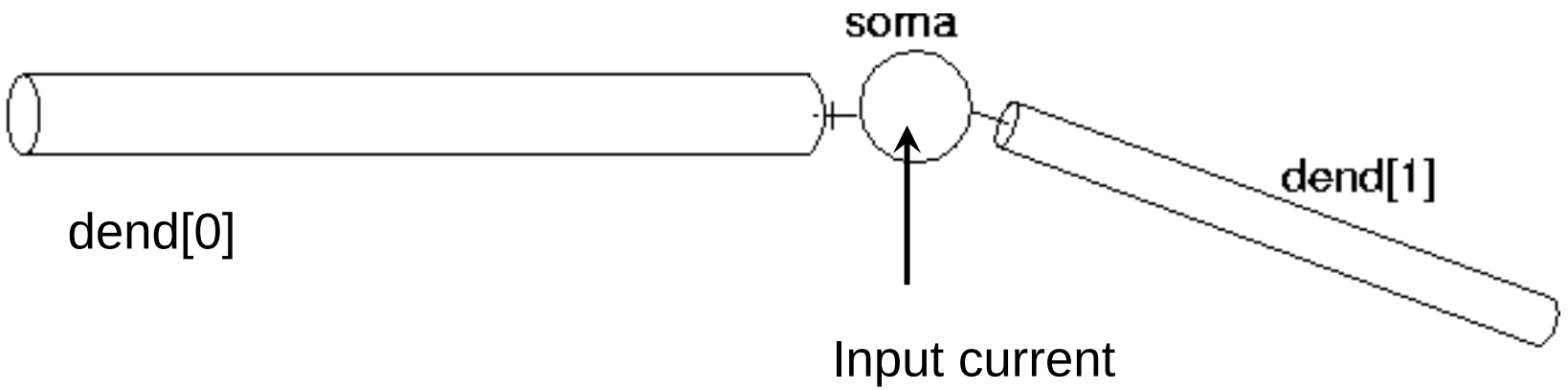
- For our simulation, we are going to connect the soma and the dendrites together. We use the **`connect`** command to do this:

```
connect dend[0](0), soma(0)
connect dend[1](0), soma(1)
```



Our model so far

- We have a soma with two dendrites and there is a stimulating electrode in the soma, which injects a rectangular current pulse into the soma.



- We want to visualise what is happening in this new model neuron, which we'll store in file **`sthB.hoc`**, and how the result differs from the previous model, which had only the soma section.

sthB.hoc

```
load_file("nrngui.hoc")

ndend = 2

create soma, dend[ndend]
access soma

soma {
      nseg = 1
      diam = 18.8
      L = 18.8
      Ra = 123.0
      insert hh
      gnabar_hh = 0.25
      gl_hh = .0001667
      el_hh = -60.0
}
```

```
dend[0] {
            nseg = 5
            diam = 3.18
            L = 701.9
            Ra = 123
            insert pas
            g_pas = .0001667
            e_pas = -60.0
}
dend[1] {
            nseg = 5
            diam = 2
            L = 549.1
            Ra = 123
            insert pas
            g_pas = .0001667
            e_pas = -60.0
}
```

The program `sthB.hoc` continued

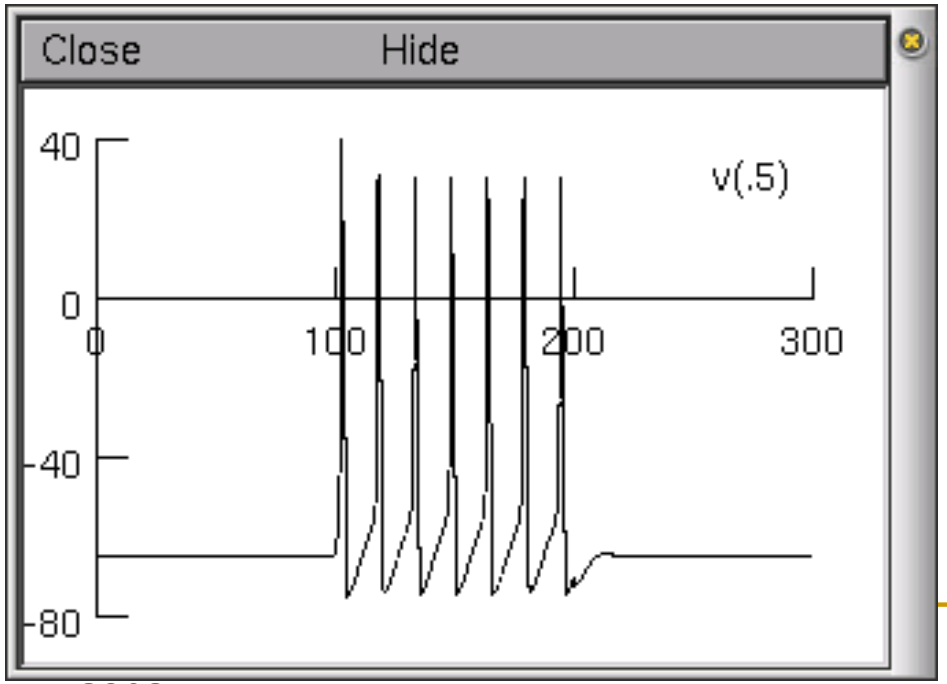
```
// Connect sections together
connect dend[0](0), soma(0)
connect dend[1](0), soma(1)

// Stimulating electrode in the soma

objectvar stim
soma stim = new IClamp(0.5)

stim.del = 100
stim.dur = 100
stim.amp = 0.1

tstop = 300
```

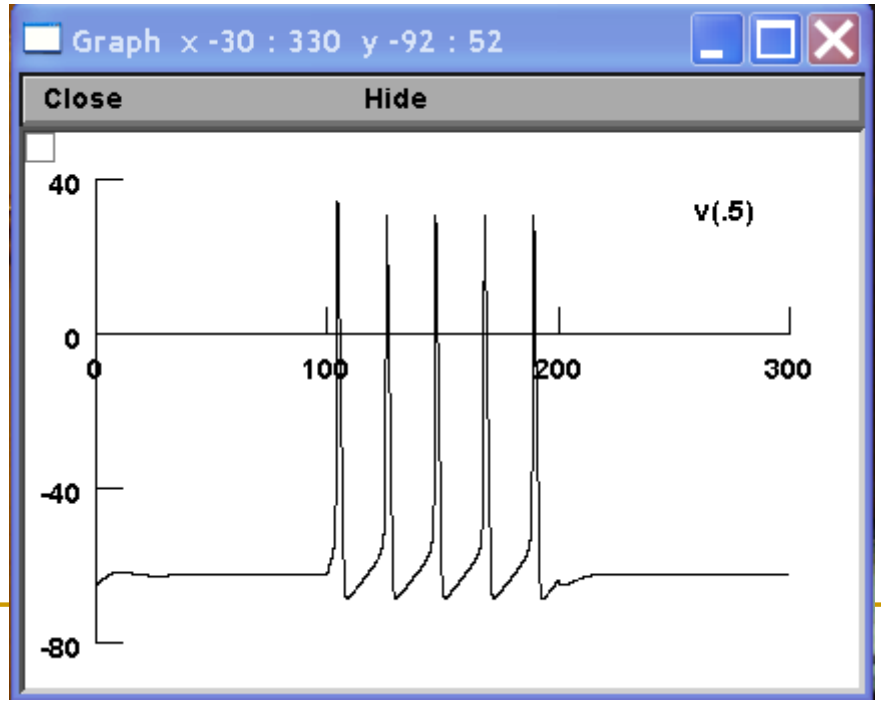
- Notice the use of //. Anything after the double forward slash is considered a comment. It is a useful way of explaining your code.

Running the stored model

- Windows: open **`sthB.hoc`** in NEURON. Open the RunControl window (under Tools) and voltage axis (under Graph) to run and display the simulation, respectively.

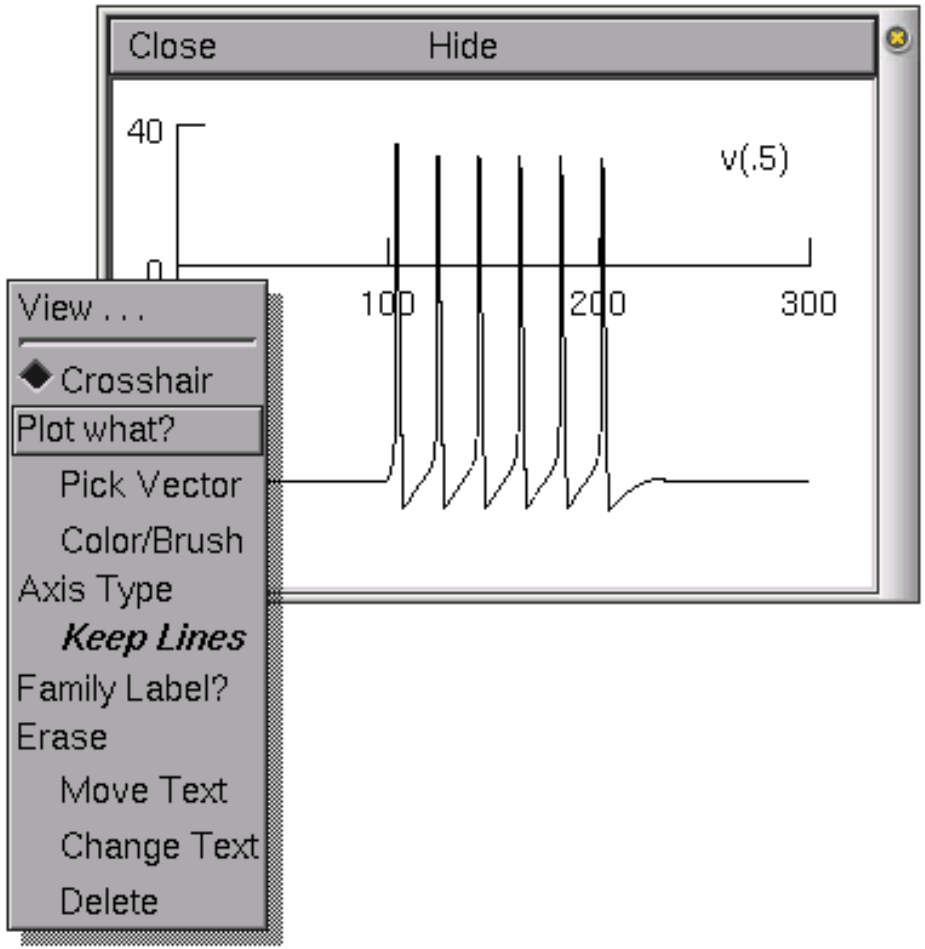
- Soma without dendrites generates 7 spikes whereas soma with the dendrites generates only 5 spikes. Why?





Visualisation of dendritic voltage

- In order to add the dendritic voltage to the graph, right click on the voltage axis and hold.
- This brings up a Graph Properties menu – View. Choose Plot what?. This should bring up a window from which we can choose a variable to plot.
- The variables can be either directly typed in if we know the exact name of the variable we want to plot, or they can be selected from the list of variables in the selection boxes.

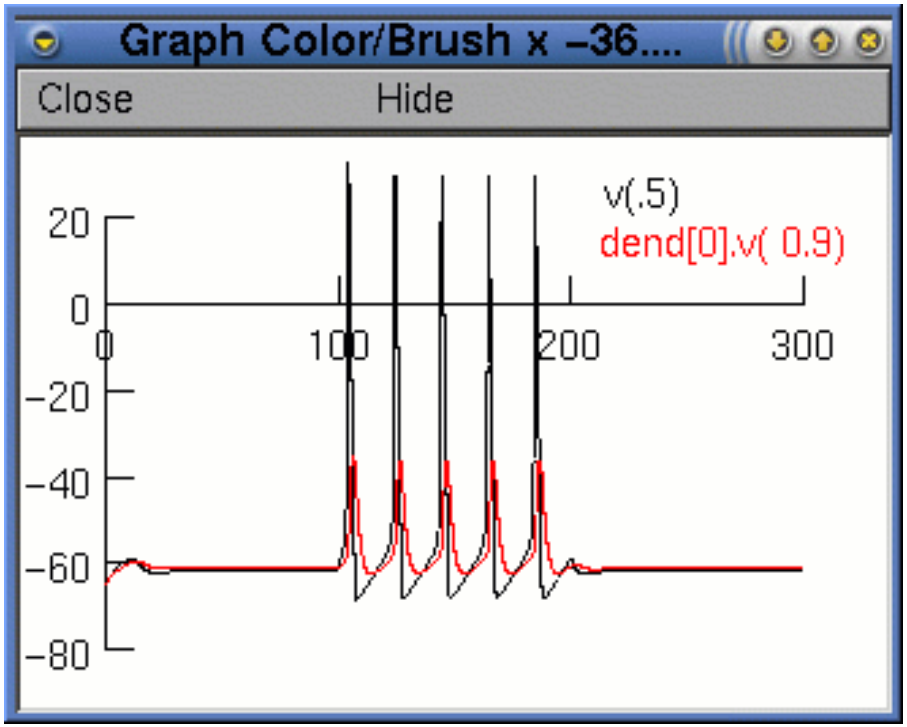


Visualisation of dendritic voltage

- We want to plot the voltage from a distal dendrite, so we need to select dend[0] in the first selection box and press Accept and then edit the text so that it reads dend[0].v(0.9) and press Accept.

- To change the colour and thickness of the lines we can select Colour/brush from the Graph Properties menu (after right clicking on the graph).

- We select colours and/or line types by clicking the button next to the colour or line type and then clicking on the relevant legend, i.e. dend[0].v(0.9).



Now when we run the simulation (hit the Init&Run button in the RunControl window), we will see two traces.

Blocking the ion channels

- Some drugs block ion channels with the consequence the ions cannot flow through them. Insert the following lines after the command **`access soma:`**

```
proc block_sodium() {
    local block_fraction
    block_fraction = $1
    soma gnabar_hh = block_fraction * 0.25
}

proc unblock_sodium() {
    soma gnabar_hh = 0.25
}
```

Simulating TTX action

- To block the active sodium channels use the command (the argument is the fraction of blocked channels) before inputing current:

```
block_sodium()
```

- Insert this command into sthB.hoc and observe what happens with the variables like voltage and currents.

- To unblock the blocked sodium channels we use the command:

```
unblock_sodium()
```

TTX extracted from pufferfish blocks sodium channels. Ingestion can cause death or permanent neurological damage to the brain turning person into a "zombie".