NEURON - tutorial C of Gillies & Sterratt (part 2)

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

COSC422 – lecture 7

How to make a more complex dendritic tree

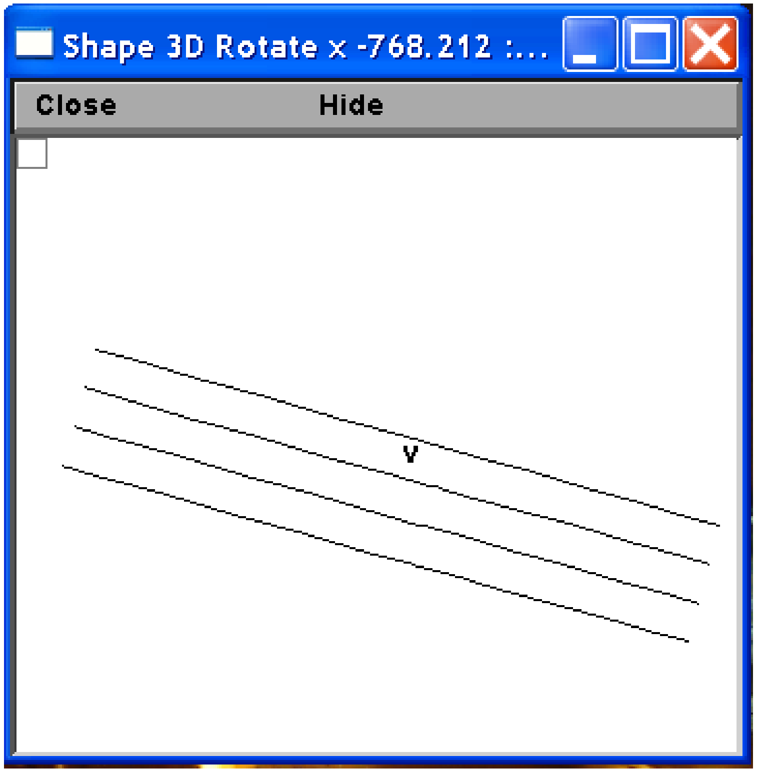
Our model so far **`SthC1.hoc`**

- We have created 4 model neurons positioned in space.

- The model neurons are disconnected.

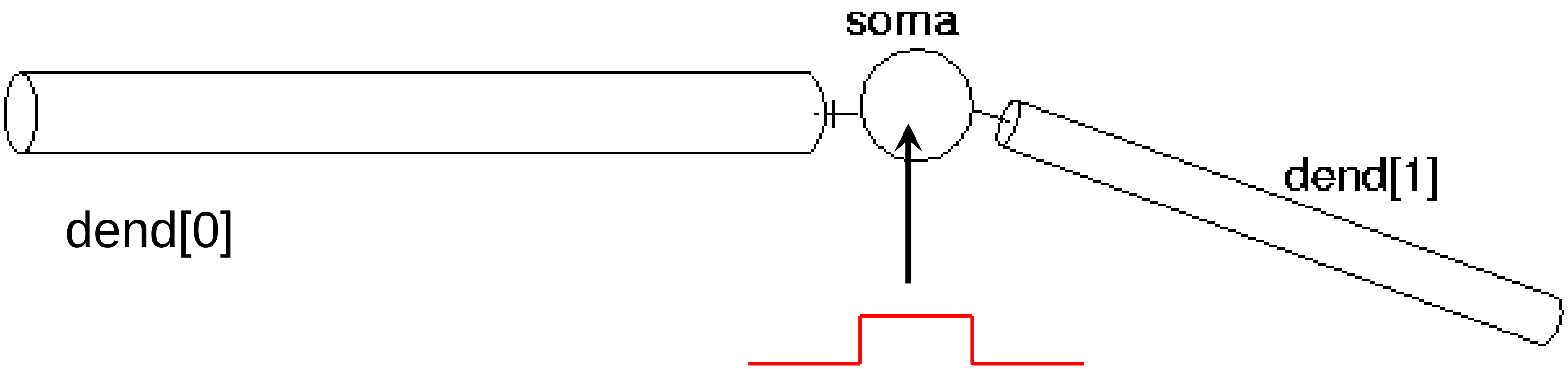
- There is a rectangular pulse of current injected into the soma of all neurons.

- “v” denotes this a default point of measurement for the voltage plot as **`soma[0].v(0.5)`**.



Our model so far **`sthC1.hoc`**

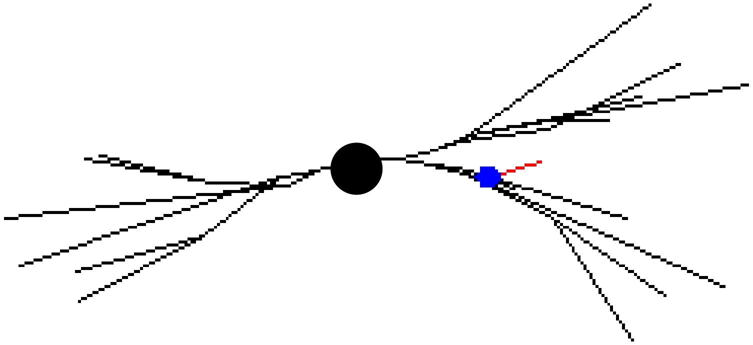
- We have 4 neurons, each having a soma with two dendrites and there is a stimulating electrode in the soma, which injects a rectangular current pulse into the soma lasting 100 ms with the delay of 100 ms.



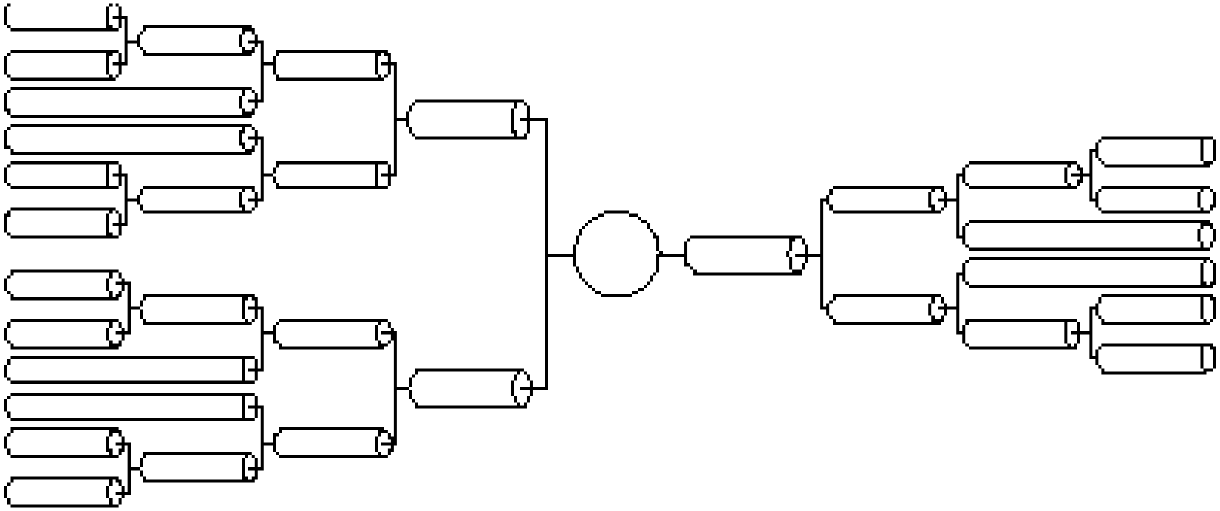
- In this lecture we will create more complex dendritic trees for these 4 model neurons.

More realistic dendritic tree

- The subthalamic neuron has two main dendritic trees:



- We will define them as the following system of 2 dendritic trees:



Tree A
23 sections

Tree B
11 sections

Data file with dendritic tree geometry

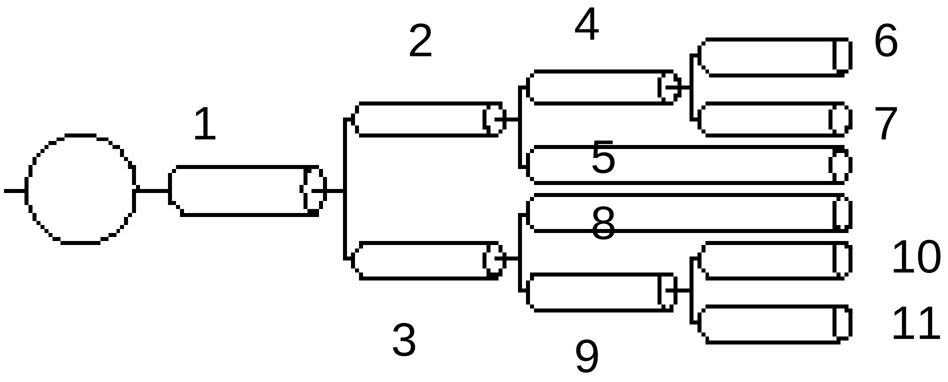
- Geometry of 2 dendritic trees is in the **`.dat`** files. Here's **`treeB.dat`**:

```
11
 1   2   3  2.000  40.000    0.000   0.000   0.000  -45.490  -12.866  -11.489
 2   4   5  1.260  40.000  -45.490 -12.866 -11.489  -84.335  -19.142  -18.677
 4   6   7  0.790 100.000  -84.335 -19.142 -18.677 -162.567  -77.332    3.543
 6   0   0  0.500 150.000 -162.567 -77.332   3.543 -292.617 -115.113   68.037
 7   0   0  0.500 150.000 -162.567 -77.332   3.543 -289.570 -145.223   45.507
 5   0   0  0.790 289.000  -84.335 -19.142 -18.677 -347.759 -109.227  -96.224
 3   8   9  1.260  40.000  -45.490 -12.866 -11.489  -77.557  -26.257  -31.297
 8   0   0  0.790 289.000  -77.557 -26.257 -31.297 -364.171  -61.026  -44.120
 9  10  11  0.790 100.000  -77.557 -26.257 -31.297 -151.728  -23.029  -98.291
10   0   0  0.500 150.000 -151.728 -23.029 -98.291 -266.619    3.635 -190.969
11   0   0  0.500 150.000 -151.728 -23.029 -98.291 -281.148    0.124 -170.503
```

Data `.dat` file with dendritic tree geometry

- First line in treeB.dat is the total number of dendritic sections, i.e. 11.
- First 3 columns: # of a section followed by #'s of section's daughters

```
11
 1   2   3
 2   4   5
 4   6   7
 6   0   0
 7   0   0
 5   0   0
 3   8   9
 8   0   0
 9  10  11
10   0   0
11   0   0
```



Format of the `tree.dat` file

- The first line has the number of sections in the given dendritic tree. Each following line has the following format:

```
branch-num child1 child2 diam L X Y Z X Y Z
```

- where ***`branch-num`*** is the reference number of the branch (starting at 1),
- ***`child1`*** and ***`child2`*** are the daughter branches reference numbers (0 if there is no daughter),
- ***`diam`*** and ***`L`*** are the branch diameter and length respectively, and
- the two sets of 3D coordinates ***`X Y Z`*** are the 3D position of branches (‘0’ and ‘1’ end points of the cylinders in 3D space).

Data file with dendritic tree geometry

- Here’s **`treeB.dat`** again:

```
11
 1   2   3  2.000  40.000    0.000   0.000   0.000  -45.490  -12.866  -11.489
 2   4   5  1.260  40.000  -45.490 -12.866 -11.489  -84.335  -19.142  -18.677
 4   6   7  0.790 100.000  -84.335 -19.142 -18.677 -162.567  -77.332    3.543
 6   0   0  0.500 150.000 -162.567 -77.332   3.543 -292.617 -115.113   68.037
 7   0   0  0.500 150.000 -162.567 -77.332   3.543 -289.570 -145.223   45.507
 5   0   0  0.790 289.000  -84.335 -19.142 -18.677 -347.759 -109.227  -96.224
 3   8   9  1.260  40.000  -45.490 -12.866 -11.489  -77.557  -26.257  -31.297
 8   0   0  0.790 289.000  -77.557 -26.257 -31.297 -364.171  -61.026  -44.120
 9  10  11  0.790 100.000  -77.557 -26.257 -31.297 -151.728  -23.029  -98.291
10   0   0  0.500 150.000 -151.728 -23.029 -98.291 -266.619    3.635 -190.969
11   0   0  0.500 150.000 -151.728 -23.029 -98.291 -281.148    0.124 -170.503
```

b c1 c2 d L X Y Z X Y Z

b’s ‘0’ end **b’s ‘1’ end**

New STh neuron template

```
begintemplate SThCell

public soma,treeA,treeB

create soma,treeA[1],treeB[1]


// object variable for a file

objectvar f


proc init() {

      local i,me,child1,child2

      create soma
```

```
soma {
      nseg = 1
      diam = 18.8
      L = 18.8
      Ra = 123.0
      insert hh
      gnabar_hh = 0.25
      gl_hh = .0001666
      el_hh = -60.0
}

// from now on we will
create treeA and treeB
from their .dat files
```

Notes on a new template SThCell

- We have made the soma, **`treeA`** and **`treeB`** public, so, for example, we could place electrodes anywhere along the dendritic trees.

- We have also created a new **`objectvar f`** used to reference the files.

- Note, we have not yet created our trees. Unlike the previous example, we no longer know the number of sections in the trees as this is now specified in the tree **`.dat`** files (in their first lines).

- Notice we have already created tree section arrays of length one just before the **`init()`** procedure. Each section and object variable that is used in the template must be declared before **`init()`**.

Accessing a file

- We have created a new **`objectvar f`** used to access the files, e.g. files with dendritic tree geometry specification.

- To access a file, we need to create a new file object. This is done in a similar manner to creating other objects (for example the IClamp).

```
f = new File()
f.ropen("treeA.dat")
```

- The first line creates the file object, the second line uses the file object function **`ropen()`** to open the file **`treeA.dat`** to read.

Reading from a file: the function **`scanvar()`**

- We can read the number of sections in the treeA from the 1st line of the treeA.dat file and then use this as a dimension in the create command:

```
ndendA = f.scanvar()
create treeA[ndendA]
```

- Now we can continue to use **`f.scanvar()`** to read the rest of our file. For example, if the next line of our file **`treeA.dat`** there was:

```
1 2 3 3.180 10.000 0.000 0.000 0.000 18.092 -0.346 4.932
```

- Thus, the second call to **`f.scanvar()`** returns the value 1, the third call of **`f.scanvar()`** returns the value 2, the fourth returns 3 and the fifth returns 3.180, sixth call the value 10.000, etc.

Defining the dendritic tree from a file

- We can define our dendritic tree treeA using the following code:

```
ndendA = f.scanvar()
create treeA[ndendA]

for i = 0, ndendA-1 {
     me = f.scanvar() - 1
     child1 = f.scanvar() - 1
     child2 = f.scanvar() - 1
```

- This is a **`for`** loop for creating each section/branch of the tree as defined by the **`.dat`** file.

Defining the dendritic tree from a file

- The local variable **`me`** is the first value read from the file, and is the reference for the parent branch.

- As tree array index starts at 0, but our file references start at 1, so the variable **`me`** is defined as **`f.scanvar() - 1`**.

- Similarly the references to daughter branches **`child1`** and **`child2`** have 1 subtracted to match the array indexing convention.

Continuation of the loop

- We continue defining our dendritic treeA within the above **`for`** loop using the following code:

```
treeA[me] {
     nseg = 1
     diam = f.scanvar()
     L = f.scanvar()
     Ra = 123
     insert pas
     g_pas = .0001666
     e_pas = -60.0
```

- The branch diameter **`diam`** and length **`L`** are directly read from the file.
- Passive conductance and reversal potential are entered based on data.

Continuation of the `for` loop

- Now all the actuall 3D position information is read from the file:

```
// clear and enter new 3D information


pt3dclear() // clearing the default positions
// adding new X Y Z for the section start
pt3dadd(f.scanvar(),f.scanvar(),f.scanvar(),diam)
// adding new X Y Z for the section end
pt3dadd(f.scanvar(),f.scanvar(),f.scanvar(),diam)
```

(Re-)positioning neurons in 3D space

- The first function, **`pt3dclear(),`** will erase any 3D positioning information associated with the section.

- The second, **`pt3dadd(),`** takes four arguments (X, Y, Z, and diam) and will add a new coordinate to the section with diameter = diam.

- We have to give coordinates for each end of the section, which can be set by making two calls to **`pt3dadd()`** – once for the "0" end of the section and once for the "1" end of the section.

- Section positions may be randomly placed, or these coordinates may explicitly follow experimentally derived anatomical measurements.

Finalising the `for` loop

- Finally the branch sections are connected up to form the tree:

```
// connect the children to the parent

if (child1 >= 0) {
      connect treeA[child1](0), 1
      }

if (child2 >= 0) {
      connect treeA[child2](0), 1
      }
} // end of treeA[me]
} // end of the whole loop

f.close  // closing the file
```

Completing the new SThCell template

- The loop repeats the setup for each branch section.
- The second tree (treeB) is done in the same manner.
- To complete the new SThCell template, after both trees have been read in from the files, we must connect the trees to the soma:

```
// Connect things to the soma
connect treeA[0](0), soma(1)
connect treeB[0](0), soma(0)
}
endtemplate SThCell
```

What we've got so far: `sthC2.hoc`

- The final four neurons, each with a full dendritic tree morphology are shown in a shape plot on the left. Next to it is the voltage trace in one of the neurons as a result of the current pulse injection. (Recall the cells are not connected yet.)