Simulating a Simple Network in NEURON

In this tutorial, we will learn to build simple networks using NEURON while modeling a reflex involving two very long neurons and a single synapse. We will learn how to add synapses to neurons built with NEURON and to add spike generators that send signals to the synapses indicating that a presynaptic event has occurred. Another tutorial on building Networks can be found within the NEURON documentation page at http://www.neuron.yale.edu/neuron/docs

I. Background: Propagation down myelinated and unmyelinated cables

Although we may not be consciously aware of this fact, signals in our nervous system travel relatively slowly compared to other signal velocities that we deal with on a day-to-day basis, like the speed of light or the speed of sound. Propagation velocities down axons and dendrites range from tenths of a mm/ms to ~100 mm/ms. In the fastest neurons, the relatively large speeds achieved are a result of two factors:

1) the cables down which the action potentials propagate have a large diameter that reduces resistance to flow along the axis of the cable, and

2) the cables are highly insulated with a fatty glial layer known as myelin.

The second mechanism is utilized primarily by vertebrate animals. Most invertebrates, by contrast, increase propagation speed only by using very large nerve fibers that can have diameters as large as 1 mm! Such large cables occupy correspondingly large volumes and it would be difficult to have many such neurons in an animal without occupying large amounts of space. In fact, vertebrates do have many, many, many more neurons than invertebrates and do not have exceptionally large axons. Instead, they achieve equally fast or even faster propagation velocities by using myelin to decrease the cell's capacitance.

In this tutorial, we will model a monosynaptic reflex arc in which signals generated by stretch of the Achilles tendon ultimately cause the calf (gastrocnemius) muscle to contract. Although the Achilles and calf are quite close to one another, the nerve signal must travel all the way from the calf muscle above the Achilles, where the stretch receptors (muscle spindles) that detect the Achilles tendon strike are located, to the lower spinal cord and back to the calf, traversing a total distance of well over 1 meter in an adult.

We will gather an estimate of how large an unmyelinated axon with Hodgkin-Huxley channels would need to be in order to achieve the observed propagation velocities. If you are interested in continuing this exploration, you could proceed beyond this tutorial by revising the neuron model used here to include alternating sections of passive myelinated cable and small nodes of Ranvier packed with Hodgkin-Huxley channels to see how greatly this enhances the speed of propagation for a fixed cable diameter.

II. Building a simple "network" of stimulator, receptor, and cable

To start building our reflex network, we will first build the following components:

1) A cable representing the axon of a sensory *afferent* nerve fiber (afferent means traveling *from* the periphery. *Efferent* means traveling *towards* the periphery. Thus, the signal

mediating a reflex travels first along the afferent neuron and then back to the periphery along the efferent neuron. If it helps you to remember, “afferent” is also alphabetically before “efferent”.)

2) A receptor on the afferent nerve fiber that will receive the stimulation (muscle stretch) that leads to the initiation of the action potential in the nerve fiber. We will model this receptor as a synapse since, like a synapse, it receives a signal and in response opens conductances (these are mechanoreceptor, rather than synaptic, conductances but functionally they behave similarly; the only practical difference is that they are stimulated by mechanical stretching rather than neurotransmitter molecules).

3) The stimulus representing the stretching of the muscle. We will model this as a spike-like event characterized simply by its time of occurrence.

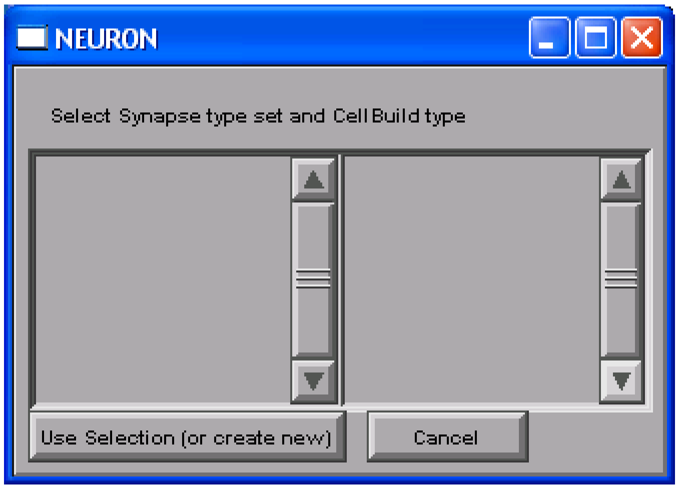
A. Building the afferent fiber model and attaching a receptor

We will first build an afferent fiber model. Realistically, we would like to build a cable that is nearly 1 m in length. However, to simulate such a lengthy cable might require a long simulation time. Thinking ahead, all we really want to know is how long it takes the action potential to propagate a given distance, i.e. the speed of propagation. Therefore, let’s just build a cable of length 100,000 μm = 100 mm to calculate the speed. We can multiply by an appropriate factor later on to calculate how long it would take to propagate down the full-sized cable. We’ll build this cable as a single section, ignoring distinctions between dendrite, soma, and axon here because they together can be considered to act as a single long cable. For other parameters, let’s use:

Ra = 100 Ω-cm
c_m = 1 μF/cm^2 (default value)
Use **HH** channels (default values; really the kinetics of the action-potential-generating channels are a bit different for mammals but the qualitative behavior should be similar)
We’ll play with **diam** to see how large a diameter is required to produce the measured propagation velocity, but for now let’s start with diam = 20μm which is about as large as mammalian fibers get.
For integration, use **d_lambda** to determine the number of compartments and use **variable dt**.

Now open the nrngui and let’s begin!

To build a biophysical cell that is going to be part of a network, we need to use **NEURON Main Menu>>Build>>Network Cell>>From Cell Builder**. This should pop up a window like the following:


NEURON
Select Synapse type set and Cell Build type
Use Selection (or create new)
Cancel

In this window, click on **"Use Selection (or create new)"**. This should result in three windows popping up on your screen:

CellBuild[0] is a CellBuilder window just like we worked with in the Single Neuron tutorial. The [0] indicates that it is the first element of an array (the next elements would be "CellBuild[1]", then "CellBuild[2]")—this is because in a network you could potentially have many different *types* of cells. Note, however, that if you would like to have two or more *identical* cells in your network, then you only need one cell builder.

SynTypeGUI[0] is used to define the parameters of the synapses that will be available to connect onto the cell built with the CellBuild window.

NetReadyCellGUI[0] is used to combine cell information from the CellBuild[er] with synapse information from the SynTypeGUI to create cells that contain synapses ready to receive signals from other cells. These cells-with-synapses can then be used by the Network Builder that we will see shortly. The NetReadyCellGUI is also where we give a name to our cell; the name will be used to identify the cell in the network we will build.

Let's first add to our cell the geometrical and biophysical parameters specified above using CellBuild[0]:

> *Important note:* ***do <u>not</u> check Continuous Create***. We will create this cell later, with a different technique, when we create the network of which it will be a part.
>
> In "Topology", change the Basename to "aff_cable" (short for "the cable of the afferent neuron") and click Accept to accept this change. Then select **"Change Name"** under the **"Click to..."** choices and click on the "soma" label in the diagram. It should change to read "aff_cable". Next, select **Click and drag to..."Move Label"** and drag the name so that it no longer overlaps the circle.
>
> In "Geometry", check d_lambda for "all" sections. Then click "aff_cable" and check **L** and **diam** under **Constant value over subset...**. Uncheck "Specify Strategy" and set

L = 100,000 μm (when you hit return, it should change to "1e+05") and diam = 20 μm.

In "Biophysics", select "aff_cable" and check Ra, cm, and hh. Then uncheck "Specify Strategy" and set Ra = 100 Ω-cm. Leave cm and hh with their default values.

We have now created a cell which is ready to have synapses attached to it.

Notice that the "NetReadyCellGUI[0]" still shows the old picture of the cell named "soma". To update this window, click its **"Refresh"** button now. You should see the picture from the CellBuild[0] topology window. Next, let's give this cell a name that will be used in the network:

Click **"Cell Name"** in the NetReadyCellGUI window and in the window that pops up, type "afferent" and Accept the change. The NetReadyCellGUI window should now say "Cell Name: afferent" (the text might be superscripted somewhat).

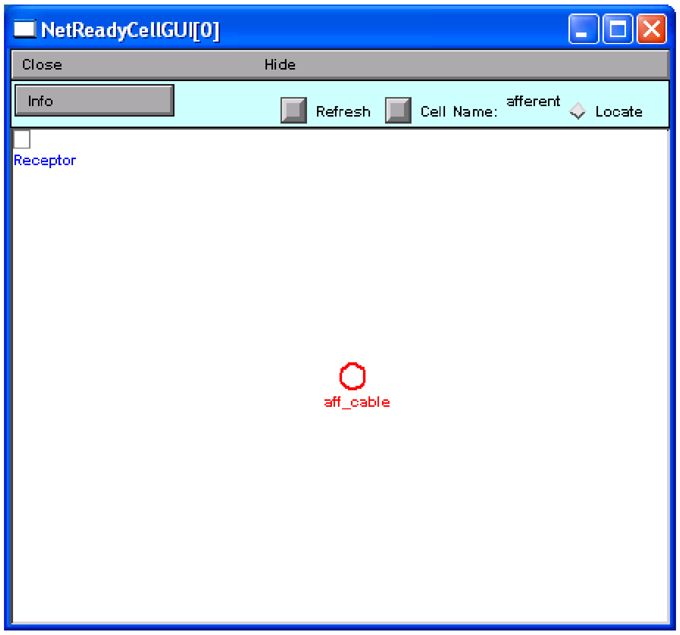
Note 1: the "Cell Name" defined here is different from the name given in the CellBuild[er]. This reflects that the Cell Name here refers collectively to both the neuron constructed in the CellBuilder and the synapses (or, in our case mechanoreceptors) that we will construct and add to it next.

Note 2: Recall that you should not use spaces or special characters within NEURON object names.

Next, let's create a receptor on this cell that will receive the signal that the muscle has been stretched. We'll model this receptor as a synapse with conductances that open upon arrival of the signal sent from the tendon-stretch stimulus (which we'll define below) and that then decay exponentially with a time constant of 3 ms. Let's suppose the receptor channels' equilibrium potential is 0 mV.

To create a synapse, go to the SynTypeGUI window and select **New>>ExpSyn.** Click on **"Rename"** and rename the synapse to "Receptor". Then fill in tau = 3 ms and e = 0 mV. *Note: as for the cell built in the CellBuilder, we have just defined a type of synapse that can be used many times in our network, i.e. if we have many synapses in our network with identical parameters to this one (as is typically the case for complex networks with many synapses), then we do not need to define any new synapse types in the SynTypeGUI.*

Now click on "Refresh" in the NetReadyCellGUI and "Receptor" should appear in blue in the upper left of the window:



Now we are ready to attach the receptor to the neuron in the NetReadyCellGUI. Let's attach it to the "0" (far left) end of the "aff_cable" section of our afferent neuron. To do this:

First, select **"Locate"** in the NetReadyCellGUI.

Then, click and hold on the word "Receptor" in the picture (you have to be careful to click on the correct portion of the word "Receptor" for this to work—clicking in the correct location takes a bit of getting used to and will be important throughout the portions described below). A red copy of the label "Receptor" should appear and the words "Synapse Receptor (Unattached)" should appear in the center of the screen, telling you that the synapse called "Receptor" has not yet been attached to the cell.

Still clicking, drag the red "Receptor" label towards the cell until the words "Synapse Receptor at aff_cable(0)" appear (if you play around, you can see how you could locate the synapse at locations on the cable other than the "0", or left, end). Release your click, and this should attach the synapse onto the cell, as signified by a blue dot and the label "Receptor0" (the "0" indicates that this is the first of this type of synapse attached; the next would be Receptor1, then Receptor2, etc.).

You can (slightly) adjust the location of the receptor label by dragging it (but make sure not to move it to the right too far, or it will no longer be attached to the "0" end of the cable). If you drag it too far away, it will become unattached, *which is how you can remove synapses*.

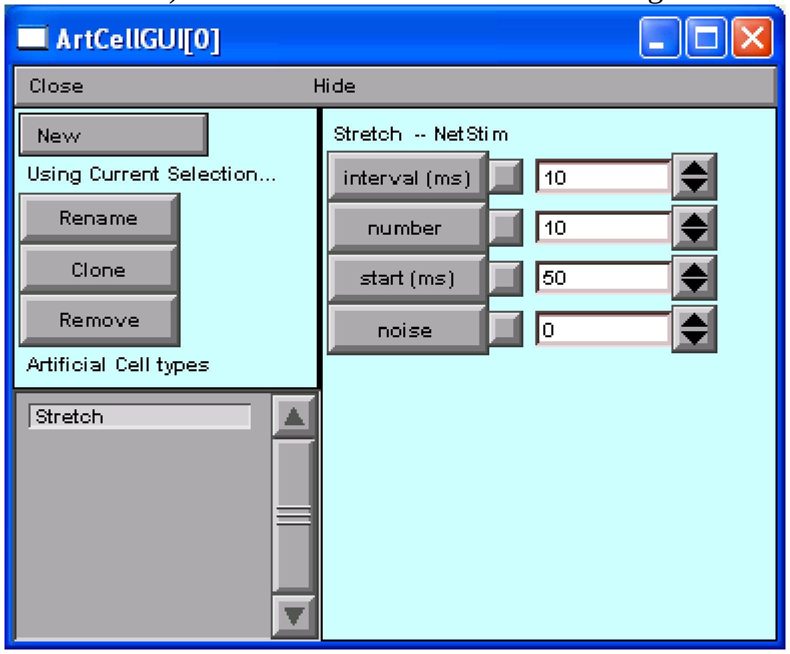
Now close the SynTypeGUI and CellBuildGUI (but **never close the NetReadyGUI or you will lose everything!!!** Similarly, never close the ArtCellGUI we will set up in part B below). If you get a warning message saying "CellBuild[0] has changed since the last save" don't worry (i.e. select "close anyway")—you can get back all of this information by clicking on **"Info>>Cell**

Builder" in the NetReadyCellGUI (or on **"Info>>Synapse Types"** to get back the synapse type information). Try it, then close the SynTypeGUI and CellBuild again.

It is now a good time to save the session, so do so now to your desktop (which you should start by accessing "Documents and Settings/" or wherever you are choosing to save files) with a filename such as "ReflexModel_StepA.ses"

B. Building the stimulator (muscle stretch)
We next want to be able to tell the receptor that it has received a signal. The source of this signal will be an "artifical cell" that creates spikes indicating times at which a signal occurs (usually the signal will be a spike from another cell, but in our case the signal will be the hammer strike on the tendon that stretches the muscle spindle). There are two classes of "artificial cells" in NEURON, spike generators known as "NetStims" (which we'll use) and integrate-and-fire neurons (simple neuron models with Leak channels, plus a voltage threshold at which a spike is fired).

To create our spike generator, click **NEURON Main Menu>>Build>>Network Cell>>Artificial Cell**. This should open a window called **"ArtCellGUI[0]"**. In this window, choose **New>>NetStim** and **Rename** it "Stretch" (to indicate that the event conveyed is the stretching of the muscle, which we will soon use as the signal to open up Receptor conductances). You should now have the following:



Stretch -- NetStim
interval (ms) 10
number 10
start (ms) 50
noise 0

The information in the right side lets you denote the **number** of events to be generated, the **interval** between events, the **start** time of the first event generated, and whether there is any **noise** (i.e. randomness, see the documentation for more about this) in the times at which events are generated.

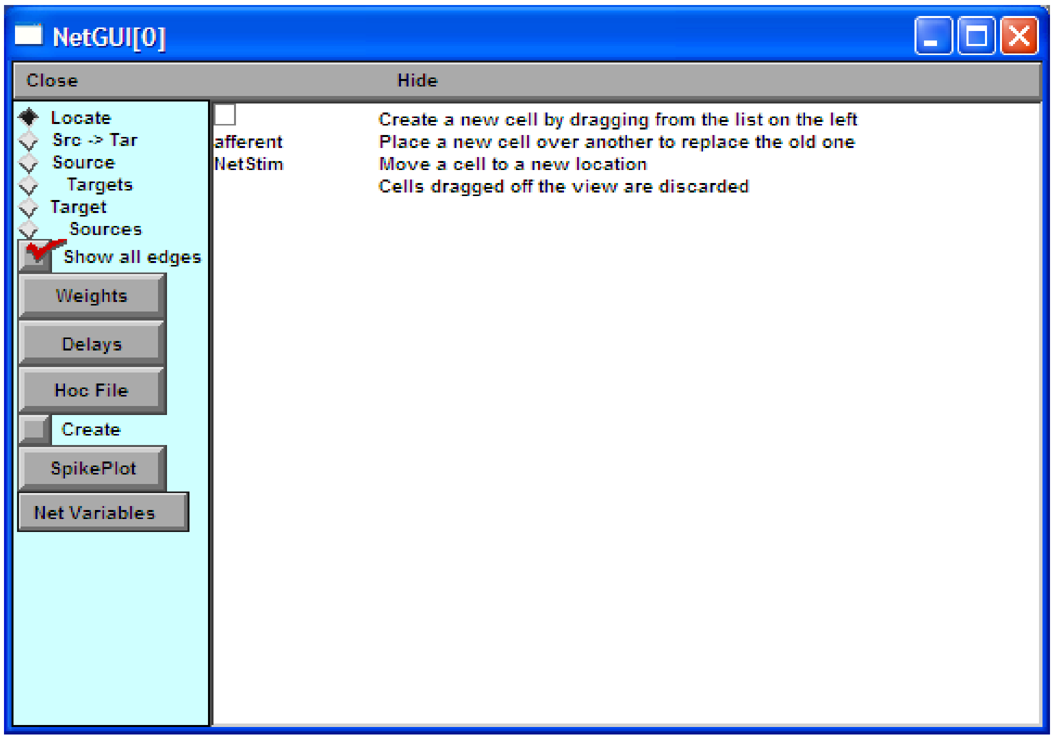
We only want to generate a single event, occurring at start time = 0 ms. Therefore, set number = 1 and start = 0 ms. Interval is irrelevant here because there is only one event.

Save the session with a filename such as "ReflexModel_StepB.ses".

C. Make a network: Connecting the stretch stimulus to the receptor

Next we will attach our "stretch" stimulus to the receptor we have created on the "afferent" cell, thus completing the creation of a very simple network consisting of 1 artificial cell (the muscle stretch "NetStim" stimulus generator) connected to our biophysical cell with a synapse. To do this, we need to create a network that contains the artificial cell "stretch" and the biophysical cell "afferent".

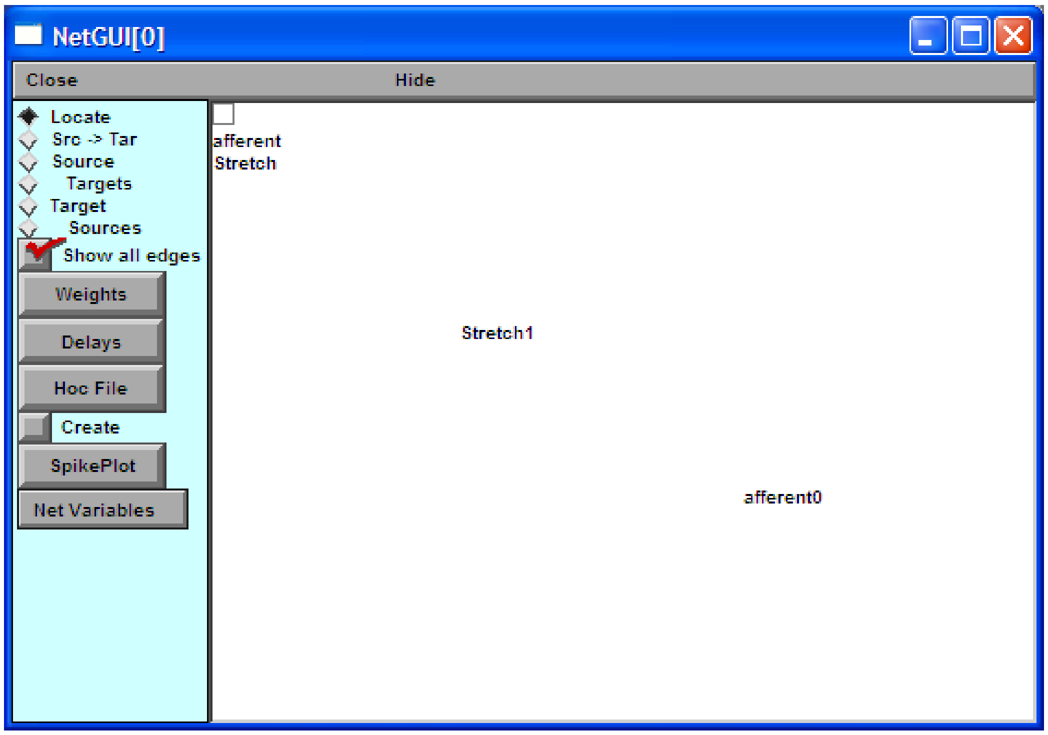
To create the network, click on **NEURON Main Menu>>Build>>Network Builder**. A window called "NetGUI[0]" should appear as follows:



Create a new cell by dragging from the list on the left
Place a new cell over another to replace the old one
Move a cell to a new location
Cells dragged off the view are discarded

In the left panel we see several select diamonds. The most useful are the top two: Selecting **Locate** will allow us to place the cells we have created at a location of our choosing in the panel at right. Clicking **Src->Tar** will allow us to connect a source stimulus (i.e. the producer of spike events) to a target location (the receiver of the spike events).

Let's add an afferent cell and a Stretch stimulus to the network. To do this, with "Locate" checked, you should click on each of these labels and drag them to a convenient location in the window. The cells will be numbered in the order you create them, as follows:



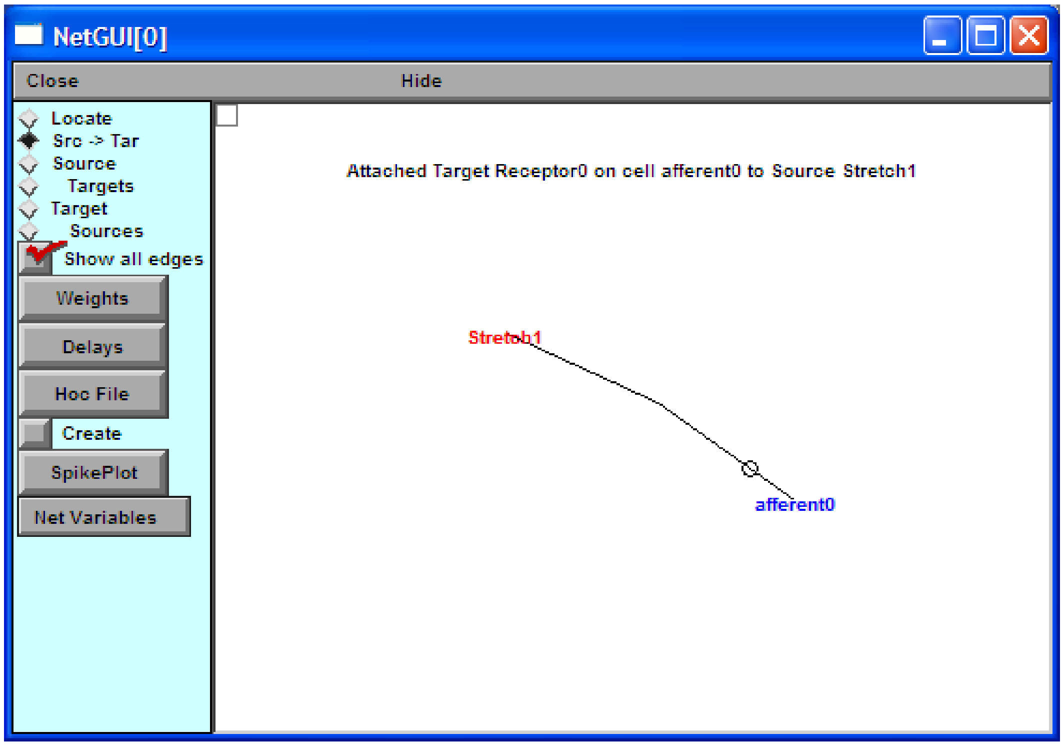
Note that you can actually create as many cells of each type as you like (try it by dragging more cells onto the palette). This is very useful if you want to create a network of many biophysically identical cells that are connected to each other. To get rid of unwanted network cells, just drag them out of the window and they will be gone (they can be a bit tricky to click on – if the text "Nothing selected" appears you have not clicked on a label; when you have clicked on a label the text should read "Move *cellname* to *coordinates_of_location*"). Do this until you are left with only afferent0 (or it may be named "afferent1" if you created it second) and Stretch1 (or it may be named "Stretch0" if you created it first). Also note that you can do a "View = plot" on this or any picture window with a Gray Box in the upper left corner if you want to make the network spread out over the entire available area.

Now we would like to connect Stretch1 (the Source) to afferent0 (the Target). To do this:

First select "Src->Tar" in the left panel. The words "Select Source and drag mouse to Target" should appear.

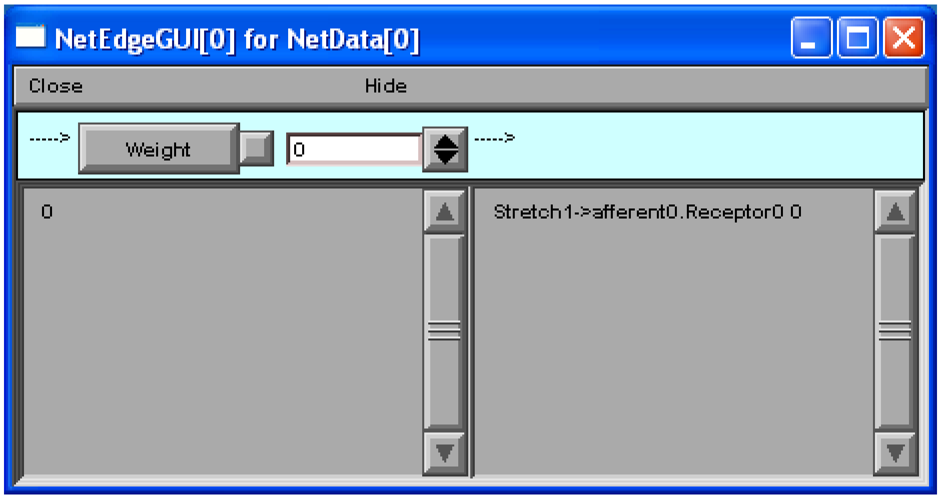
Follow this direction by clicking and holding on Stretch1 (it should turn red and "Source Stretch1 selected" should appear) and dragging onto the afferent0 label. The afferent0 label should turn blue, a picture of the afferent0 cell topology should appear, and the words "No target selected on cell afferent0 from Source Stretch1" should also appear. The latter phrase indicates that we have not yet told NEURON where on the cell we would like to attach the source.

Continue holding the mouse button down. We would like to attach the source onto the Receptor. To do this, with the afferent0 topology visible, drag your mouse onto the **Receptor0 label** (**<u>not</u>** onto the blue dot) which should make the phrase "Attach Target Receptor0 on cell afferent0 to Source Stretch1" appear. Let go of the mouse button and it should be attached, and look like the following:



Attached Target Receptor0 on cell afferent0 to Source Stretch1

We now have a network (yay!!!). However, we still have not defined the strength (or **Weight**) of the “synaptic” (or in our case the mechanoreceptor) connection between Stretch 1 and afferent0. To do this, click on **“Weights”** and the following window should appear:



Stretch1->afferent0.Receptor0 0

Widen this window so that, as shown in the figure above, you can see the full name of the connections and the value of the weight (this value, which is currently zero, is indicated to the right of the connection name “Stretch1->afferent0.Receptor0” in the right-half panel). You can neglect the information in the left-half panel—this is a list of the values of the weights you have previously assigned (currently only the value 0 has been assigned), and is useful for when one would like to assign the same value to each of many synapses in a complex network.

For a synapse (in our case, the receptor) of type ExpSyn, the weight gives the increase in synaptic conductance (often called Δg_{syn} or something similar) created after an event (usually a spike; in our case, the muscle stretch) occurs, i.e. after a spike the original value of g_{syn} is replaced by an updated value g_{syn} + Weight.

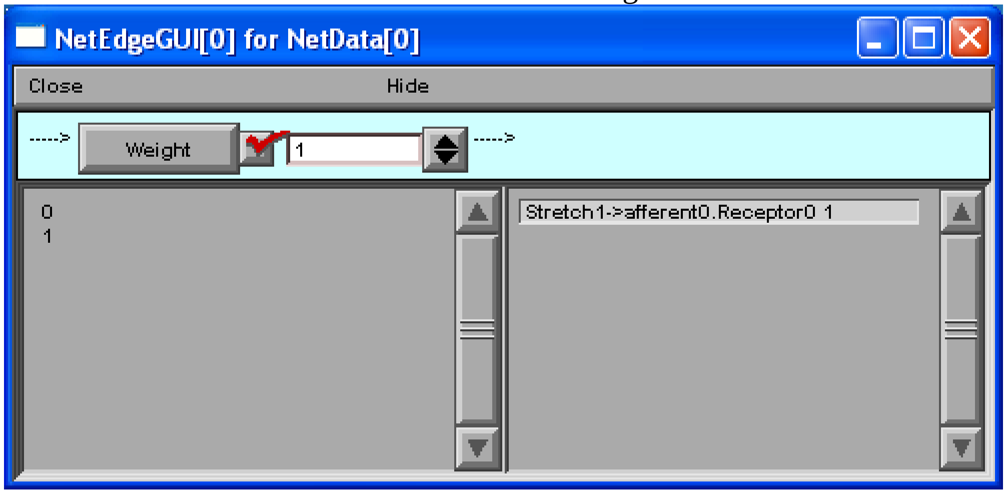
To set this Weight is a three-step process:

1) Type in a value for Weight in the white text box at the top. Let's set Weight = 1 (the units are μS).

2) With the cursor in the white text box, hit the Enter key on your keyboard.

3) Click on the connection you would like this Weight to apply to in the **right panel** of this window. A 1 should appear at the far right of the text in the right panel, indicating that this connection has a connection weight of value 1. (Clicking on a number in the **left panel** puts that number in the white Weight box as in step 1. This is a helpful convenience when building large networks in which many neurons have the same weight values but, for now, I recommend that you ignore the left panel.)

Your window should now look like the following:



Weight: 1

Stretch1->afferent0.Receptor0 1

You can also have the effect of the stimulus event take place only after a delay. If you want to model a delay, you should click **Delays** in the NetGUI window.

Click "Delays" and in the window, set Delay = 1 ms (again, first type this in and hit Enter, then click on the connection in the right panel).

You are done building the network. Close the Weights and Delays windows if they are still open and save the session with a filename such as "ReflexModel_StepC.ses".

D. Running the simulation

We are now ready to run the simulation. To do so, the first thing we need to do is tell NEURON that we have finished creating the network by clicking the **Create** button in the NetGUI window. Do this now.

Once we have created the network, we cannot further update the parameters of the cells in the network during this session. The only thing we are allowed to change at this point are the weights of the connections and the delays (we also might be able to change the connections drawn in the panel, although to be safe, let's not). If we want to go back and change the properties (e.g. the length or capacitance) of any cells in the network we would have to uncheck Create and **quit NEURON before checking Create again**. This, although a bit cumbersome, keeps NEURON from mixing up parameters' values. (Alas, there is no "Continuous Create" for networks, unlike when we are just building a single cell).

If you try unclicking Create (try this!), you will receive a message saying "Create cannot be turned back on without exiting NEURON". Click "Stay on" to keep the Create Button on.

Now let's bring up everything we need to run the model and see a plot of voltage vs. time at the receptor end of our axon and halfway down the axon:

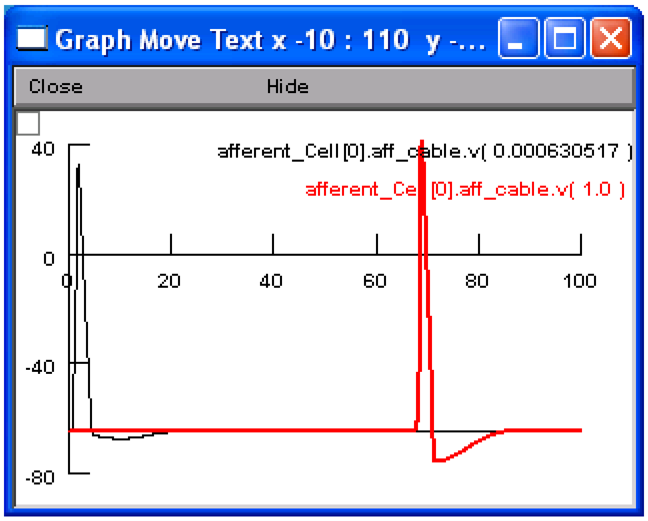
First, bring up a RunControl and set Tstop to 100 ms.

Next, bring up a Voltage axis graph. On this graph, delete v(.5), then using Plot What?, select Show>>Objects, double-click in the left panel on afferent_Cell_(this may be at the bottom of the list), then in the middle panel on "0.", then in the middle panel on "aff_cable.", then in the right panel on v(0.00…). Click Accept. If you were wondering where the "_Cell" came from in "afferent_Cell", you can click on NetGUI>>Net Variables>>Show Map and it will tell you the variable names (right column) that NEURON has given to the various cells and synapses you created (left column). Close the map window when you are done looking at it.

Now use Plot What? again to set up a second variable that will plot the same cell's voltage at the other (1.0) end of the soma. Do this by changing the v(0.00….) part of the label name to v(1.0) in the white text box and click Accept. Finally, move the text of these variable names to be plotted (so you can see them better) and make the v(1.0) plot red.

Finally, bring up a VariableStepControl Window and choose "Use variable dt"

Click on Init&Run and you should get the following output:



Thought question: How fast did the wave travel?

Calculate this by using **Gray Box>>Cross Hair** to find the times of the peaks of the spikes at the beginning and end of our 100 mm length of cable (placing the Cross Hair on your graph gives the x and y coordinates in the title of the window):

This is *way slower* than the actual propagation velocities in large myelinated fibers, which can be many tens of meters per second, even though the diameter of this cable was as big as we ever see in real mammalian neurons! Since the real distance that needs to be traveled is close to 1.5 m, this suggests that if we had unmyelinated axons, this reflex would take a full second to occur!!! (and to receive and respond to any proprioceptive information from our foot would take at least this long since the signal might even have to go all the way up to the brain!).

Next we'll see how big we have to make the axon in order to get a typical propagation velocity of ~40 m/sec seen in very large myelinated fibers. To do this, we'll need to change our cell specifications, which will require (*don't do this just yet—see below*) unclicking Create and quitting and reopening NEURON.

Before doing this, let's save **just the windows related to run control and plotting (and <u>not</u> the VariableTimeStep window).** Why do this instead of saving the entire session? Two reasons:

-First, it's sometimes nice to separate conceptually the specification of the model (as captured by the NetGUI, NetReadyCellGUI, and ArtCellGUI) from the Plots and RunControls.

-Second, there is a more pragmatic reason: When we uncheck the Create box in the NetGUI, it will "uncreate" the network. This means that the variables we are plotting in the Voltage vs. time plot will no longer exist and if we try to load the session it will give us an error (you would find the error output in the bash/nrngui/nrniv window) telling us that we are trying to set up a plot with variables that do not exist.

To save just the RunControl and V vs. t Graph windows of the session, click Window>>Print & File Window Manager. Then click on the panels corresponding to these two windows in the left panel.

Now click **Session>>Save selected** and save them with a name like "ReflexModel_Controls.ses". *Note:* If you are tired of having to click through the menus to finally get to the Desktop or wherever you are saving, you can avoid doing this by telling NEURON what directory you are working in by selecting it through the **NEURON Main Menu>>File>>working dir** menu and then selecting "Move To". Try this! In future NEURON sessions, this directory will be listed in the **NEURON Main Menu>>File>>recent dir** menu so you can select it more quickly this way.

Finally, close the Print & File Window Manager as well as the RunControl, VariableTimeStep, Voltage vs. time Graph, and if you still have them open, the SynTypeGUI and CellBuildGUI. Then **uncheck Create** in the NetGUI window (click "Turn off" in response to the question) and save the remaining windows (using NEURON Main Menu>>File>>save session) to a session file called something like "ReflexModel_StepD.ses".

Quit NEURON.

E. Adjusting the parameters for the unmyelinated neuron.
Now re-open NEURON and load the session "ReflexModel_StepD.ses" that you just saved – try doing this by first changing to the appropriate directory (the desktop or wherever you last saved to) by using the NEURON Main Menu>>File>>recent dir menu you just learned about above. Then use File>>load session to open your session.

Now let's change the cell biophysics by bringing up the CellBuild[0] window through NetReadyCellGUI>>Info>>Cell Builder. If your last simulation didn't take too too long, then change the length of the aff_cable to something more like half the total distance that will need to be traveled in going from the Achilles tendon to the spinal cord and back to the calf muscle (i.e. half the total distance is approximately the length of the afferent nerve, with the other half of the distance being along the efferent nerve that we will not model here). A reasonable length would be 700 mm = 700,000 μm = 7e+05 μm.

Next we will increase the axon size until we get the propagation speed to increase from the value observed in the previous simulation to the experimentally observed propagation velocity. You could try to guess how much bigger the diameter should be, and then test your guess by checking Create on the NetGui and loading your run control & voltage graph. *Note, however, that for each new guess you will have to uncreate the network, quit neuron, and then change the geometry again in the Cell builder before re-creating a network with different cell size.*

To avoid having to keep iteratively trying out guesses, we can use basic results from cable theory to make a reasoned guess as to how big a diameter we'll need in order to increase the propagation speed from the value of ≈1.5 m/s in the previous simulation to the desired propagation velocity of approximately 40 m/s. This requires that we speed things up by a factor of approximately 40/1.5 ≈ 27. From cable theory, the conduction velocity down a long unmyelinated axon scales with the square root of the axonal diameter, i.e. v ~ $\sqrt{diam}$. Thus, to increase v by a factor of approximately 27, we'll need to increase diam by a factor of over 700 (the square root of 700 is approximately 26.5). So, let's try increasing the diameter by a factor of 700, i.e. changing it from diam = 20 μm to diam = 14000 μm = 14 mm = 1.4 cm (!!!).

Now make these changes to the Geometry portion of CellBuild[0], then close the CellBuilder.

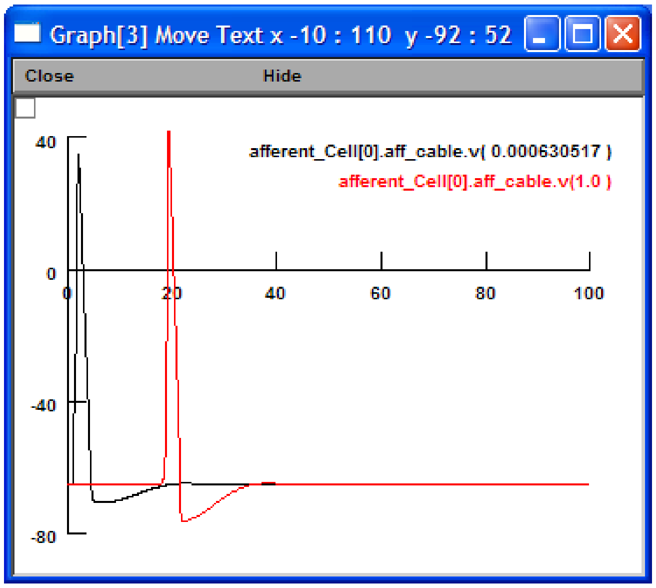
Now click Refresh on NetReadyCellGUI (the cell should turn red; this is actually optional, it only changes the picture; nevertheless, it's a good habit to get into). Then, check **Create** on the NetGUI. Your network should now be updated to contain the new afferent cell parameter values.

Next load your ReflexModel_Controls.ses file. *Note: this must be done after clicking create or the variables in the plots will not exist; go to your bash/nrngui/nrniv window to look for error messages—if you click return and the program quits, there was an error. You should always check the nrniv window in this manner when funny things start happening in NEURON! If this error persists after re-trying this section of the tutorial, you will unfortunately need to re-do the above. However, do not "re-do" bringing up the session controls—instead create a new RunControl panel and Voltage vs. time graph as you did earlier in the tutorial.*

Then open a VariableTimeStep window and check "Use variable dt".

Now run the simulation. What do you see? Nothing! (well, a boring line at -65 mV). *What happened??? Can you think why the receptor opening now creates nothing?*

The problem here is that we have left the receptor weight unchanged while we've increased the cell's size by a tremendous amount. Change the Weight to 10,000 μS (remember to update both in the white text box and by clicking in the right panel) and run again and you should see the following (you may have to resize your window and move the text in order to see the voltage-trace text labels):


Graph[3] Move Text x -10 : 110 y -92 : 52
Close
Hide
afferent_Cell[0].aff_cable.v(0.000630517)
afferent_Cell[0].aff_cable.v(1.0)
40
0
-40
-80
0
20
40
60
80
100

Using Crosshair to locate the peak of the action potentials at each end of the cable, we now find that the 700 mm distance was traveled in about 18 ms for an average velocity of

v = (700 mm)/(18 ms) ≈ 39 m/s,

which is a much more realistic value. Thus, **we conclude that we would need cables of approximately 1.4 cm in diameter to attain the same propagation velocities obtained with myelinated fibers!!!** Score one for the vertebrates!

Finally, let's create a shape plot. Remember how? ... To get the shape plot, it's NEURON Main Menu>>Graph>>Shape plot. Then choose Gray box>>Shape Plot and your window should say "Shape Shape Plot x...". Next (not necessary, but makes the next simulation look prettier), choose Gray Box>>Shape Style>>Show Diam. Your axon should get very thick (reflecting that it is very thick).

Now run your neuron. You should see the voltage propagate down the neuron as a pulse of color. If it is happening too quickly to see well, I recommend turning off "Use variable dt" in your VariableTimeStep window, and slowing down dt to 0.0025. Also, since the pulse has reached the end of the cable by less than 40 ms, change Tstop to 40 ms. Run your neuron again with these settings and you should see a much slower and prettier propagation down the axon.

Now uncheck Create (in case you want to change any other parameter values in the future), close all windows except for the NEURON Main Menu, bash/nrngui/nrniv, ArtCellGUI, NetReadyCellGUI, and NetGUI, and save this session with a file name like "ReflexModel_StepE.ses".

III. Where would you go from here?

You might think we have "cheated" in that our network contained only the afferent cell (plus the stimulus that stimulated the mechanoreceptor on this cell). However, to make a more complicated network requires no new techniques beyond what was needed in creating our "2 cell" (1 artificial cell for the stimulus and 1 biophysical cell) network.

If you wanted to create the efferent cell to bring back the signal from the spinal cord, you would create a new Network Cell (from NEURON Main Menu>>Build) which would bring up a second NetReadyCellGUI and associated SynTypeGUI[1] and CellBuild[1]. You would then specify the topology, geometry, and biophysics of this cell in the new Cell Builder; specify the properties of the synapse onto this cell with the SynTypeGUI, and name the cell and locate the synapse on the cell with the NetReadyCellGUI. Finally, you could incorporate this cell into your previous network and define the synaptic weight between this and the other cell(s) with the NetGUI (the new cell label should appear in NetGUI[0] when you create the new Network Cell. You may have to click on the NetGUI to make it refresh its display.). *Note: if you wanted your efferent cell to be identical to your afferent cell in terms of its geometry, biophysical properties, and synapse types, you would not need to create a separate Network Cell and associated NetReadyCellGUI—instead, you could simply drag out a second "afferent" cell in your NetGUI (although you might want to rename the cell from within the NetReadyCellGUI to be something other than "afferent" since it would now be used as both the afferent and the efferent cell).*

If you have time, try doing this!