Simulating a Single Compartment Model in NEURON

This tutorial is intended to get you acquainted with the graphical interface capabilities of the NEURON programming environment and to allow you to explore more about the cable properties of neurons with either active (i.e. including voltage-dependent conductances) or passive (including only voltage-independent leak conductances) membranes. For more information about building simulations through code, rather than graphically, see the references to manuals and tutorials listed in the "Getting Help" section of this tutorial.

I. What is NEURON?

NEURON is a programming environment for building biophysical models of neurons and networks. It is a "higher level" program than pure programming environments like C++, MATLAB, or Java in that you do not need to write computational algorithms that tell the computer how to integrate the set of differential equations governing the dynamics of current flows and voltage changes in neurons. Rather, you only need to tell NEURON what the anatomical and electrophysiological properties of the neuron(s) in your network are (and the initial conditions of your simulation) and it takes care of the details of integrating the differential equations that govern the behavior of the modeled cell(s). It also has many nice built-in properties for displaying outputs, including capabilities for producing movies that allow one to visualize voltage changes over time across the entire spatial extent of a neuron.

To specify neuron and network properties, NEURON has both a graphical interface, which allows for quick and easy building of neurons and control of outputs, and also a programming language that allows more flexibility in precisely specifying simulation details. Here we will focus on the graphical user interface, which is best for quickly getting simple simulations running, but those interested in becoming NEURON experts are encouraged to learn how to build neurons through code.

II. Downloading and Launching NEURON

A) Downloading NEURON

To download NEURON, go to the NEURON website: http://www.neuron.yale.edu/neuron/. This website contains a host of useful information about the program, including the Documentation (including additional tutorials and the Programmer's Reference); a Forum where you can post questions & view answers in several categories; and Model DB, an archive of NEURON code posted by investigators (typically from published papers) that provides a great set of code examples that you can play with or use as examples if you ever wish to write your own NEURON code.

Now click on the "Download" tab and select the version corresponding to your operating system from the "Download and install" section. Download the appropriate setup file (I recommend saving it to your Desktop), double-click on it, and follow the ensuing instructions (I recommend accepting the default choices). When the setup has completed, you should see a folder appear on your desktop entitled "NEURON #.#" where "#.#" is the current version number of the program. At this point, you can safely delete the setup file from your Desktop.

B) Launching NEURON
To launch NEURON, open this folder and click on the **nrngui** icon (note: directions might differ slightly for Mac or Linux; this tutorial is written using the PC version of NEURON). This is actually accessing the file located in the NEURON program directory C:\nrn##\lib\hoc (where ## represents the version number). Note that you may also select the NEURON #.# icon through the start menu.

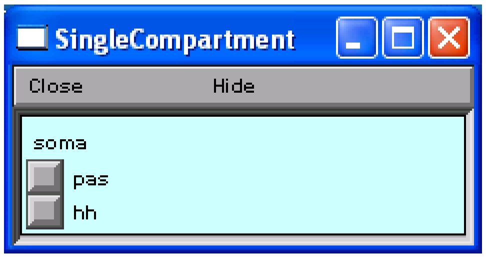
Two windows should open on your screen: the **NEURON Main Menu window** contains menus for controlling the graphical user interface. The **nrngui** (also named "**bash**" or "**nrniv**" in different versions of the program) **window** allows the user to manually type in NEURON commands and code or to print out the values of variables.

III. Building a single-compartment neuron model

Let's jump right in and make a single-compartment neuron. First, we will build a single-compartment neuron with no channels (i.e. a simple lipid bilayer compartment, or capacitor) and record its voltage from various initial conditions. Then we will add leak channels (also known as 'passive' channels because they do not open and close in response to voltage), Hodgkin-Huxley channels (i.e. the combination of Leak, Na^+, and K^+ channels used by Hodgkin and Huxley in their model), and a current source for injecting currents into our cell.

A) Single-compartment neuron with no channels: learning the basic NEURON menus
To build a single-compartment neuron, click on **Build>>single compartment**. A new window should appear entitled "SingleCompartment" (you can resize it if you can't see the full title, and move it if it covers up your NEURON Main Menu window. It is good practice to spread out all of your open windows in an organized manner, like the controls of an airplane's cockpit). *Note: sometimes when a window has previously been covered up by another window, it will develop streaks in it. To get it to look pretty again, just resize it a little. Also, if you ever "Hide" a window, you can bring it up again by clicking on its title in* ***NEURON Main Menu>>Window***.

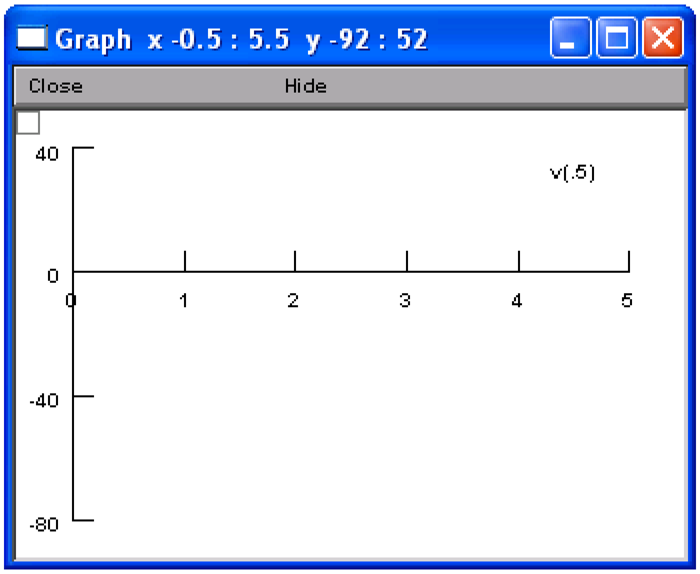
The window should appear as follows (up to cosmetic differences depending on your operating system):



The appearance of this window tells you that a new compartment has been created which is named "soma." There are also two check-boxes which allow you to insert sets of channels called "pas" (for "passive," or what we have called "leak," channels) or "hh" (for the full set of Hodgkin-Huxley channels: Na+, K+, and Leak). For now, let's leave these unchecked and run our "simplest" model of the neuron, i.e. just a lipid bilayer (capacitor) with no channels.

To run this and see the output, we'll need to do two things: First, bring up a voltage vs. time axis to view the results. Second, bring up a "Run control" panel that will let us set parameters like the integration time step dt that is used to integrate the differential equations (in math class, we take the limit that dt goes to zero but in simulations we have to make dt a very small, but nonzero number), the duration of the run, and the initial conditions (i.e. the voltage of the cell at the start of the simulation).

Let's start with bringing up a voltage vs. time plot. To do this, click **Graph>>Voltage axis** in the **NEURON Main Menu** (*note: if the menu stays on the screen, you can get it to hide by clicking on the open gray space to the right of the Iconify button and this usually gets it to disappear [but don't click on Iconify unless you want to turn the window into an icon, which you will then have to click on to make it reappear]*). The following should appear:

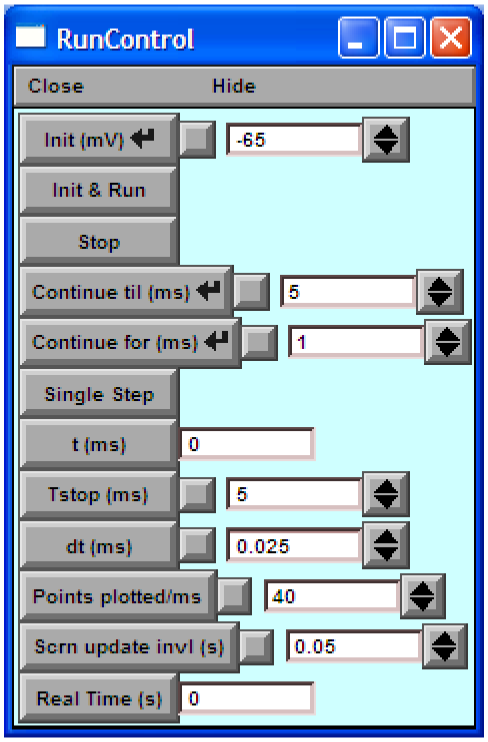


This is a plot of "v(.5)" which is NEURON notation for "the voltage in the middle of the single compartment". NEURON assumes that all compartments are cylindrical in shape so the .5 means we're putting our voltage-measuring electrode in the middle of (i.e. 5/10 of the way down the length of) this cylinder. However, this is really superfluous as the fundamental assumption of a single-compartment model is that the voltage is the same everywhere. This location identification number will become more important later when we deal with extended sections of a neuron like the axon.

Click on the small gray box in the upper left of the white panel. You should see a menu which allows one to specify many plotting options. If you move your cursor onto **View…** there should be many other choices as well (such as zooming in/out). This is where you should go to make your plots look pretty, and we'll play with a few of these menu items later in this tutorial.

Next, we need a panel to let us control our simulation. Bring this up by clicking on **Tools>>RunControl**. A panel entitled "RunControl" should appear with many buttons. We'll cover the most important of them:

The "**Init (mV)**" button sets the initial voltage value for our run. Let's change this: Click in the white box and type "-55" followed by the Enter key (*note: sometimes NEURON will not let you enter a value if your cursor is not located in the box in which you are typing, in case this happens to you)*. A check mark should appear in the neighboring box telling you that you've changed this value. If you click on the check box (do this), the value will return to -65 mV. You can also use the arrow keys to increase or decrease the voltage. By default, the numbers change by 1 mV per click. To change the increment, you can right click on the arrow buttons. A menu should pop up that lets you change the behavior of the arrows button. You can set the arrows button to either multiply the value of Init (mV) by a fixed amount (e.g. *10 would cause each



Init (mV) -65
Continue til (ms) 5
Continue for (ms) 1
t (ms) 0
Tstop (ms) 5
dt (ms) 0.025
Points plotted/ms 40
Scrn update invl (s) 0.05
Real Time (s) 0

click to multiply the value by 10) or add a fixed amount. Let's set this to +10 mV and then using 2 clicks, set the initial voltage for our run to –45 mV (you may need to click the check box to first return the value to -65mV before doing this).

Next, let's run the program for 100 ms. To do this, first reset the duration of the run (as specified in the **"Tstop (ms)"** box) to equal 100 ms. Then click **"Init & Run"** which initializes the voltage to the value specified in Init (mV) and runs the simulation. It also initializes all of the variables governing the opening and closing of channels to their steady-state values at the initial voltage.

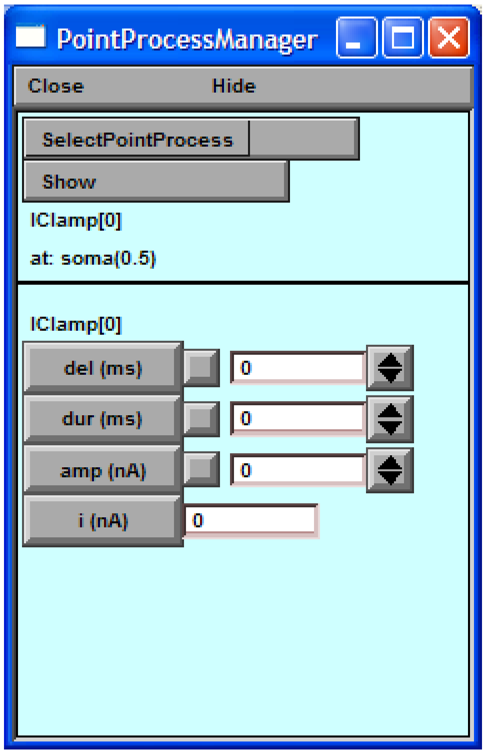
If you look (look very quickly!) at the **"t (ms)"** box, you may see numbers scroll up to the time specified in the **"Tstop (ms)"** box, in this case to 100 ms. Try it again and also observe your V vs. t graph. You should see the voltage start at -45 mV and stay there (as it should, since we have no channels in the membrane so there is nothing to make it change). Notice that **"Real Time (s)"** tells you how long the simulation actually took to run, in real (not simulated) time. This is for the efficiency lovers who want to know how fast their simulation is running.

Some optional and less often used buttons: If for some reason you'd like to stop your run before Tstop is reached, you can click the **"Stop"** button in the middle of the run. You can then use **"Continue til (ms)"** to tell the program how much further to run, i.e. it will continue to run until this time (or continue for a given amount of time if you use **"Continue for (ms)")**. You can also make your code move forward a single time step dt by clicking **"Single step".** These are less used commands but can be useful for debugging (i.e. fixing problems, also known as "bugs", in your simulation). *Alternatively, if you would like to completely terminate a simulation (e.g. because it is frozen), you can go to the nrngui/bash/nrniv window and type CTRL-C.*

The time step **"dt (ms)"** gives the simulation time step. Sometimes you might not want to plot every single time step (because plotting slows down your simulation). If you wish, you can therefore additionally choose to tell the computer how many points to plot/ms using the **"Points plotted/ms"** box. *Technical note: An issue that can arise here is that NEURON can only plot points at times that are integer multiples of the time step dt (because these are the times at which the voltage and other variables are calculated). If for some reason you specify Points plotted/ms to correspond to a spacing between time points that is not equal to an integer multiple of dt, then NEURON will round down your dt value to the nearest value that is commensurate with the Points plotted/ms specification. You will see this update appear in the "dt (ms)" box.*

Now that we understand our basic run and plotting controls, let's add a current injection electrode. NEURON refers to anything that is located at a single point along a neuron as a "Point process" (e.g. an electrode or a synapse). To insert a current injection electrode (i.e. a current source, or also known as a "current clamp"), click on **Tools>>Point**

Processes>>Managers>>Point Manager. Then click on **SelectPointProcess>>IClamp** and the following should appear:



IClamp[0]
at: soma(0.5)

IClamp[0]

del (ms)	0
dur (ms)	0
amp (nA)	0
i (nA)	0

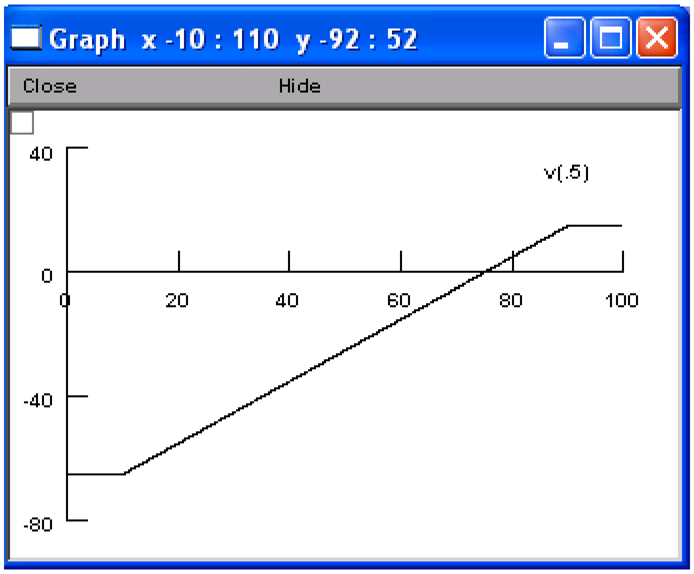
This creates a current clamp (i.e. an electrode that will inject current) named IClamp[0] with location soma(0.5), i.e. in the middle of the soma (*Note: the [0] denotes that this is the "0th" electode, in case you want to have multiple electrodes in the cell (e.g. in different locations in a more complex neuron model). If you were to insert another IClamp in the cell, this second current clamp would be named IClamp[1]. Starting to count from 0 instead of 1 may seem odd, but is common in computer programming*). If you click on **Show>>Shape**, you should see a red line representing the cylindrically shaped soma and a blue dot representing the location of the current clamp electrode.

Go back to the parameters menu by clicking on **Show>>Parameters**. There are 3 boxes for entering data that will describe the properties of the current you would like to inject through this electrode (the 4th box, "i (nA)" will display the value of the injected current during the run):

The **"del (ms)"** is the delay before injecting current. Set this to 10 ms.

The **"dur (ms)"** box is the duration of the injection. Set this to 80 ms. (*Note: you actually do not need to keep typing "Enter" after entering each value. If you click in another white box, NEURON will automatically enter the value you typed in the previous white box.)*

The **"amp (nA)"** box is the amplitude of the current injection. Set this to 0.001 nA.



Now reset the initial voltage of your simulation (in RunControl) back to -65 mV and then run the simulation. The cell should integrate the current pulse (just like our simple "A neuron is a capacitor" model in lab) and the output should look like the following:

For fun, let's make this plot prettier. To have NEURON reset the axes to a range that just fits the simulation, click on the Gray Box in the upper left of the plot and then go to **View…>>View = plot**. If you would like to set your own axis limits, use **View…>>Set View** and type in the range of x-values and then y-values you would like to use. Next, let's make the line a pretty color. Click on the Gray Box and select **"Color/Brush"**. The left column gives color choices and the right column gives choices of line thickness. Choose the red brush and 4th choice from the top to select a thick red line. Then close that window and click on the v(.5) vs. t trace to make it turn thick red (alternatively, you could have clicked on the v(.5) label—either method of selecting the trace will work). Next, to return your cursor to its default behavior in the plotting window click **(Gray Box)>>CrossHair**. If you now click on the voltage trace, the window's menu title should give you the x- and y-coordinates of the point with the cross on it.

Thought question 1: From this graph and the amplitude of your current injection, what is the capacitance of this cell?

Next let's open another plotting window so that we can see the injected current. We can do this by opening another graph, e.g. by clicking **NEURON Main Menu>>Graph>>Current axis**. Unfortunately, unlike opening a voltage axis graph, this does not by default bring up a plot of the injected current vs. time. Rather, it brings up a completely blank graph and we will need to tell NEURON what to plot in it. To do so, click on **(Gray Box)>>Plot what?** A new window should pop up entitled simply "NEURON". To tell NEURON to plot the object known as IClamp[0], click on **Show>>Objects**, then select and *double-click* on IClamp_, then double-click "0." in the next column. The middle column should now be showing you choices for what feature of the IClamp you would like to plot. We would like to plot the injected current, so double-click on "i". The variable name "IClamp[0].i" should now appear in the white text box. Click on "Accept" to finish. Your new graph should now say "IClamp[0].i", specifying what will be plotted in this graph.

Run your simulation and you should now see both the voltage vs. time in one plot and current injected vs. t in the other. What? You don't see the current trace? Ahhh….this is because the value 0.001 is too small to see with the present set of y-axis values. To fix this, click (Gray Box)>>View = plot. Also, make this plot blue (remember how?).

You can also add more plots to a graph and, even though this plot was opened by selecting "Graph>>Current axis", I have absolutely no idea why the button is named "Current axis": it is a misnomer, because we can actually plot anything we want in this graph (e.g. we could add a plot of the voltage to this graph by again using the Plot what? command). We can also delete plots that we no longer want to see plotted by using the **(GrayBox)>>Delete** command (indented under **"Erase"**). Feel free to try adding and deleting plots if you would like.

B) Adding channels to the single compartment neuron

Now let's make our simulation more interesting by adding channels to our membrane. First, let's just add passive (leak) channels:

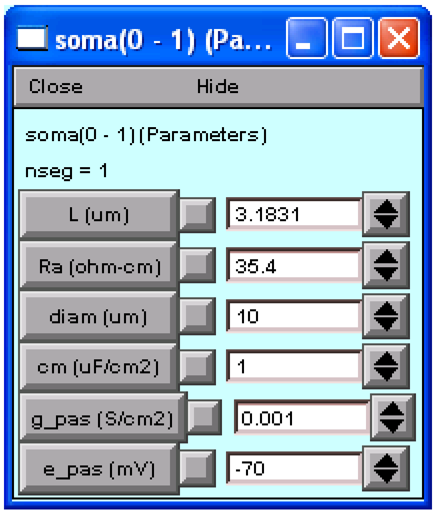
Do this by clicking **"pas"** in the SingleCompartment menu. Also, reset the amplitude of the injected current to zero. Now run your simulation. What happened? To really see it, do a

View=plot to the v(.5) vs. t plot: Apparently, the passive channel has a default equilibrium potential of E_pas = -70 mV.

Thought question 2: From this graph (and using View>>Set View… to set the x axis so you can better see the shape of the graph), what is the time constant of this cell?

You might be wondering, where were the parameters that determined the capacitance and time constant? These were set to default, typical neuronal values by the program so you could quickly get started making simulations. Next we'll see how to access (and optionally change) this information.

Information about neuronal elements that are typically distributed throughout the membrane (like ion channels), rather than being located at a single point, are referred to in NEURON as "Distributed mechanisms". To view the properties of these elements, *as well as other biophysical properties of a given section of the neuron*, click on **Tools>>Distributed Mechanisms>>Viewers>>Shape Name**. A window entitled **"Section Parameters (double click)"** should come up which shows the different sections of the neuron (in our case, we only have a single section called "soma") on the lower right as well as a picture of the neuron morphology (a cylinder in our case) in the left panel. Double click on "soma" to see its parameter values:



soma(0 - 1)(Parameters)
nseg = 1
L (um) 3.1831
Ra (ohm-cm) 35.4
diam (um) 10
cm (uF/cm2) 1
g_pas (S/cm2) 0.001
e_pas (mV) -70

This new menu now tells us that, by default (i.e. by creating a Single Compartment and adding passive channels), we have created a cylindrical compartment with:
Length **L** = 3.1831 μm,
Diameter **diam** = 10 μm,
Axial (i.e. *longitudinal* or *internal*) *resistance* **Ra** = 35.4 Ω-cm (irrelevant in this case because there is only 1 compartment so no longitudinal flow of currents),
Capacitance **cm** = 1 μF/cm^2 = 10 nF/mm^2,
Passive (*leak*) *channel conductance* **g_pas** = 0.001   S/cm^2 = 0.01 mS/mm^2,
Equilibrium potential **e_pas** = -70 mV.

Note here that the capacitance and conductance are given in terms of *densities* (per square cm of membrane) rather than absolute amounts. This is because larger cells have proportionally more capacitance (since capacitance is proportional to surface area) and also more channels (since larger membranes will typically have correspondingly more channels in them).

Thought question 3: From the numbers given, calculate the membrane time constant (and check that your measurement in question 2 was correct!).

Now reduce g_pas by a factor of 10 to 0.0001 S/cm^2 and make e_pas = -80 mV and run this again. Do a View=plot on the voltage graph and you should see a beautiful exponential decay with τ=10 ms down to a steady-state value V_∞ = e_pas = –80 mV.

Now inject 0.01 nA of current and run again. After the initial delay period, the voltage should rise up to about 20 mV. Increase the diameter of the cell to 50 μm and try again. For a bigger cell, the same amount of current injected does not raise the cell's voltage as much!

Finally, let's remove the passive channel (uncheck it in the "SingleCompartment" window) and instead add Hodgkin-Huxley channels. Notice that the properties g_pas and e_pas in your soma window now say "Doesn't exist". To get updated parameter values, let's close this soma(0-1) parameters window and then double-click again on "soma" in the Section Parameters window to bring up the updated values of the soma parameters.

You should now see values for the *maximum conductances* (i.e. the total conductance of a type of channel if *all* of the channels were open, which corresponds to the number of channels in the membrane) of the Hodgkin-Huxley Na^+ and K^+ channels, gnabar_hh and gkbar_hh, the leak conductance gl_hh, and the leak, Na^+, and K^+ equilibrium potentials. Set the current injection amplitude to zero and run your model. The voltage should stay at -65 mV, which is the resting potential for the Hodgkin-Huxley model.

Now inject 0.01 and then 0.1 nA of current. The latter should produce 7 action potentials during the 80 ms of current injection. This is the output of the *real* Hodgkin and Huxley model (no switching channels on and off by hand) and you should feel free to play around with it as you choose.

Wasn't that a nice and easy way to quickly program a Hodgkin-Huxley model? (And a *lot* faster than the mechanical computer Hodgkin & Huxley used to do their calculations!)

IV. Saving and printing

To save this session, with all of the open windows, click **NEURONMainMenu>>File>>save session**. A window entitled "NEURON" should appear with a menu for saving. The item "Filter" in the bottom indicates what types of files will appear as you search through directories: by default, as you search through the directories only files ending in ".ses" (the suffix for NEURON sessions) will appear. To save to your desktop, click Documents and Settings/, then *your_user_or_account_name* (e.g. Administrator if this is your own computer and you never changed from the default account)/, then Desktop (*note: if you wanted to go up to the folder that contains the one you are viewing the contents of, click "../"*). In the white text-entry box **near the top of this window** (**<u>not</u>** the lower white box that specifies the "filter" or type of file—you shouldn't touch this filter box), enter a memorable name for your file ending in ".ses" like "NeuronLab_1CompartHH_*YourFirstAndLastName*.ses" (Compart = "compartment" in this shorthand) by typing this after ".../Desktop/". (Note that you have to explicitly type the ".ses" extension, unlike some MS Windows programs in which the extension is added for you automatically.) The icon for this session should now appear on your desktop.

Now close NEURON by choosing **File>>Quit** from the NEURON Main Menu.

Now re-open NEURON and then use **NEURONMainMenu>>File>>load session** to load back in the session you just saved. You should see the model we had just created. Click Init & Run and you should get back your plot of the neuron firing action potentials in response to injected current.

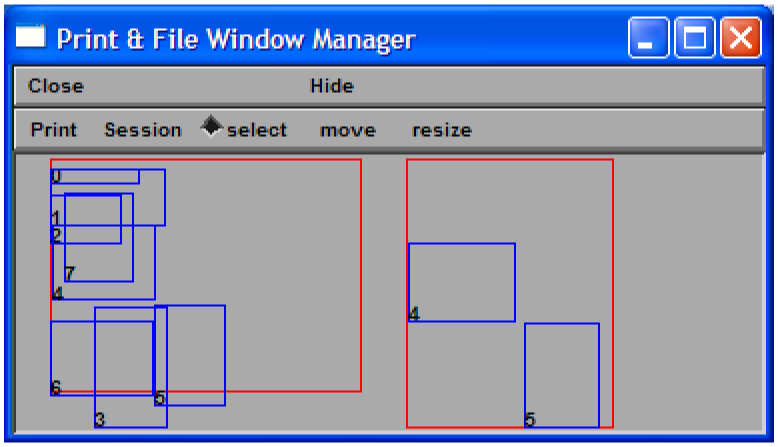
Note that the above way of loading sessions is a bit burdensome in that you had to click your way through Documents and Settings, *your username*, Desktop. If you are always saving your files to the same location (such as the Desktop), there is a shortcut to allow you to navigate straight to this directory. If you click on **File>>working dir** you will get another Directory navigation window, which in this case lets you specify the "working directory" in which you are storing all of your work. Click through this to get to your Desktop (as you did for "save session" above), then click on "**Move To**" at the bottom of the window and Neuron will make this into your default session for the remainder of the session. To test that this has occurred, click again on NEURONMainMenu>>File>>save session and you should notice that you are located in the Desktop (after doing this, you can click Cancel).

Now quit NEURON and re-start it. We again would like to load our previous session, but this time we'll use a shortcut. Click **File>>recent dir** and you should see the directory you previously specified with "working dir" listed (in this case, your Desktop). Click on this and the working directory will be set to your Desktop. You should also note that this change in working directory has been recorded in the nrngui/nrniv/bash window. You should check this window frequently for messages of this sort (***and more importantly, for error messages, which appear here!!!***). Now click on File>>load session and you should see your recently created session file as a choice. Double-click on this to return to your previous session, and click Init & Run to again get back your graph of 7 beautiful action potentials.

If you would like to print your graphs or other windows, there are a few ways of doing this:

Method 1: Select a window you would like to print (e.g. the graph with the voltage trace in it). Click **ALT-PrintScrn** (i.e. click and hold the Alt key while clicking on the PrintScrn button which should be near the top right of your keyboard). This just copied the Graph window into the Windows Clipboard. You can now open Microsoft Word or PowerPoint or your favorite drawing/word processing/presentation program and do Edit>>Paste to paste the figure in. This method actually has nothing to do with the program NEURON – it is a general Windows function and you can do this with any Window on your desktop (not just NEURON windows).

Method 2 (the NEURON way): In the NEURON Main Menu, click **Window>>Print & File Window Manager**. This should give you, in the left panel, a schematic picture of all the NEURON windows you have open. You should also see the **"select"** button marked (if not, click select to mark it). Try moving one of your windows and you'll see the corresponding rectangle move in the Print & File Window Manager. Now click on the rectangle corresponding to any windows you would like to plot (e.g. the RunControl window and the voltage graph). The rectangles corresponding to the selected windows should now appear in the right panel, something like the following (hopefully your windows are less bunched up than those shown below!):



You now have a couple of options.

First, you can save using this window: Under **"Session"** you can **"Save all"** windows, which is equivalent to File>>save session from the NEURON Main Menu. You can also just save the selected windows (i.e. ones in the right panel) by clicking **"Save selected"**. This is useful if you only want to save some, rather than all, windows from your session (equivalently, you could also close the ones you don't want saved and then use File>>save session).

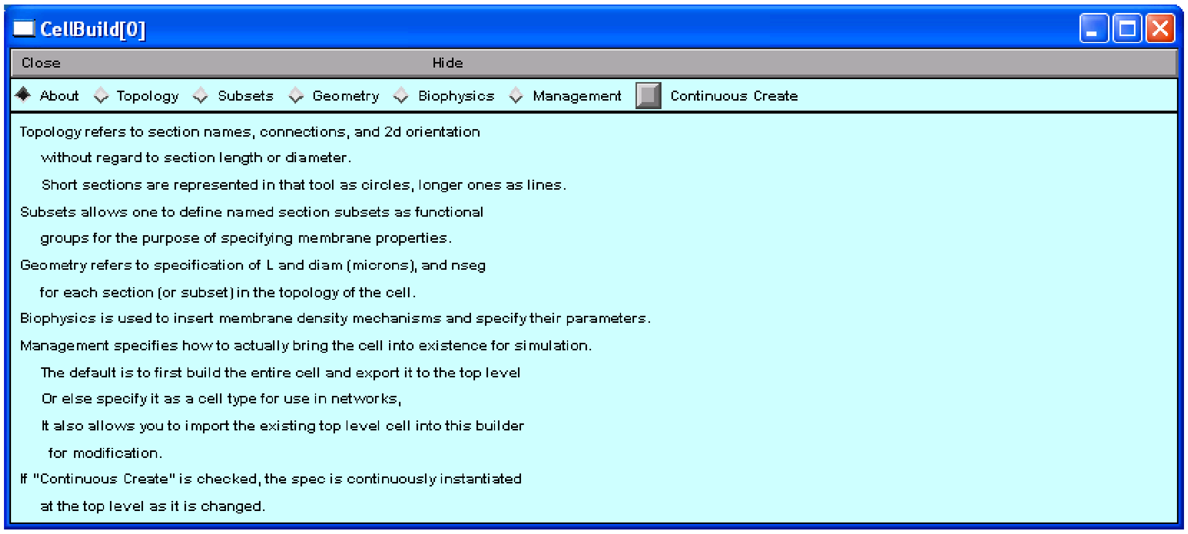
Second, you can print: Under **"Print"** you can click **"To Printer"** and you should be able to choose a printer to send this plot to (there may be an intermediate window the first time you try this which says something like "Windows". Just click "Accept" or the equivalent if this is the case).

Quit this session now, and then re-open NEURON (but do not re-open this session).

V. Building a multi-compartment neuron model

In this portion of the tutorial, we will build a neuron with extended spatial structure using NEURON's Cell Builder. The Cell Builder is a graphical interface for creating cells with multiple compartments (referred to as "segments" in NEURON) and specifying their anatomical and biophysical properties.

Let's first acquaint ourselves with the Cell Builder by skimming through its various menus. First, open the Cell Builder from the **NEURON Main Menu>>Build>>Cell Builder**. The following screen should appear:



Topology refers to section names, connections, and 2d orientation
without regard to section length or diameter.
Short sections are represented in that tool as circles, longer ones as lines.
Subsets allows one to define named section subsets as functional
groups for the purpose of specifying membrane properties.
Geometry refers to specification of L and diam (microns), and nseg
for each section (or subset) in the topology of the cell.
Biophysics is used to insert membrane density mechanisms and specify their parameters.
Management specifies how to actually bring the cell into existence for simulation.
The default is to first build the entire cell and export it to the top level
Or else specify it as a cell type for use in networks,
It also allows you to import the existing top level cell into this builder
for modification.
If "Continuous Create" is checked, the spec is continuously instantiated
at the top level as it is changed.

The checkboxes at the top indicate the steps that need to be taken to define a cell. Skim the information in the About box and then let's click our way through the menus to see what's in the Cell Builder as a default (i.e. before we add anything).

First click "Topology". **Topology** basically lets you draw a picture of your model neuron and label the various sections. Here, "sections" refers to different parts of your neuron with different properties, each of which might themselves have multiple compartments (or "segments" in NEURON's terminology). For example a model might have a soma section, an axon section, and a few separate dendrite sections representing different dendritic branches. By default the neuron is a single section named "soma" (*despite being drawn as a circle, it's actually a short fat cylinder with dimensions we'll soon see*). The menus on the right allow you to add more sections and name them. We'll come back to this shortly.

For now, skip the **Subsets** tab (or just take a quick look), which is for organizing the sections of neurons with more complicated morphologies. Instead click on Geometry.

Geometry allows you to specify the anatomy of each section. You can specify properties that apply to every ("all") sections of the neuron; for example, the axial (also known as internal) resistivity Ra and specific (i.e. per unit area) membrane capacitance, which you will specify under Biophysics below, are typically the same in all sections of a neuron. Alternatively, you can specify properties that apply to only a particular section by clicking on the name of that section (in the default case, the only section is "soma"). *Typically, you will want to first tell NEURON the properties that apply to all sections; then, you can fill in special values that apply to only specific sections. Note that all new specifications over-write previous declarations for the chosen sections.*

Under **Geometry**, click on **"soma"** to specify that we would only like to alter the soma's properties (this is only a pedagogical point for now because right now the only section that exists in this neuron it the soma; however, soon we will have multiple sections and it will be important to specify which one we are defining values for). Then click on **"L"** and **"diam"** under "Constant value over subset" (you won't, and can't, use the "Distinct values over subset" unless you have defined subsets, which we won't be doing for now). Notice that "L, diam" (or "diam, L") now appears next to soma to indicate that you have chosen to set these parameters. This tells NEURON that we would like to specify new (non-default) values for the Length and Diameter of the soma (NEURON calls this process "specifying our strategy" for defining the neuron's parameters, which is why the Specify Strategy box is checked). Now that we have told NEURON what we would like to change, to actually specify the values of L and diam, first uncheck **"Specify Strategy".** The default length and diameter of the soma appear, with text boxes that would allow you to change them (we'll do so shortly, but leave them as is for now and check Specify Strategy again). (Alternatively you can simply specify the "**area"** of the cell and NEURON assumes that the length and diameter are equal.)

Next click on **Biophysics** in order to specify biophysical properties of our neuron such as the capacitance per unit area **cm**, internal resistivity **Ra** (i.e. the resistivity of the intracellular fluid that connects the different compartments of the neuron), and the densities of various ion channel types. Click on **"all"** and then on **Ra** and **cm** and then uncheck Specify Strategy. You can now select whether to edit Ra for all compartments (*warning: this will change the values for every single compartment, so make sure you really want to do this!)* or cm for all compartments. Click on Ra and on cm to see their default values. For now, leave these values unchanged. You also can add passive (**"pas"**) or Hodgkin-Huxley (**"hh"**) channels by returning to Specify Strategy, checking them, and then unchecking Specify Strategy to get the boxes in which you can change their parameters – do not change any of these parameters for now.

Finally, although we won't be using these in this tutorial, under **Management**: **"Cell Type"** is used when creating cells to be used in networks, and **"Export"** allows you to turn the cell you created into lines of computer code (a "*filename*.hoc" file) that you could then manually modify using NEURON's programming language (for more about the NEURON programming language, see the references at the end of this tutorial).

Next, let's build a multi-compartment model…

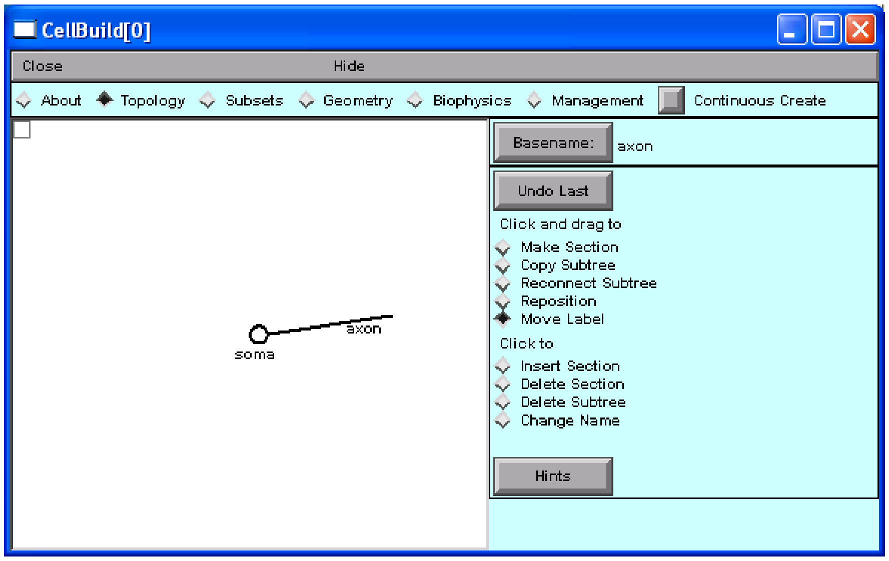
SIMPLE MULTI-COMPARTMENT MODEL: PASSIVE SOMA AND AXON

Here, we will build a neuron with 2 sections: a short, fat soma and a long thin axon, each with passive channels only; in the following Network builder tutorial, we will modify this model to include Hodgkin-Huxley channels. The anatomical and biophysical properties of this neuron can be summarized as:

Cell properties

Section	**L**	**diam**	**Biophysics**
Soma	50 μm	50 μm	pas channels with g_pas = 0.0001 S/cm^2
axon	2000 μm	1 μm	pas channels with g_pas = 0.0001 S/cm^2
All sections			Ra = 100 ohm-cm

To create this, return to the Topology section. We want to add a section named "axon". To set this name, click on **"Basename",** type in "axon" (*note*: your cursor should be in the Section name prefix window when you do this or you may find that you are unable to type anything; this is a convention in many NEURON windows), and Accept this. (*Important note: if you decide to rename NEURON objects such as sections or cells or synapses, do not use spaces or special characters as these can potentially make NEURON crash).* Now notice that "Make section" is selected under "Click and drag to". If we now click on the screen it will draw a section named "axon" and if we hold the button and drag, we can change the size of the section. If you click more times, it will create more sections that connect to the nearest available sections. The new sections should be named axon[1], axon[2], and so on. Try this and then select "Click to…**Delete section"** to get rid of all of them except for the first one (named "axon"). Next reposition the soma section label so that it isn't right on top of the section by selecting **"Move label"** and then dragging the soma label to a more aesthetically pleasing location. You should now have a screen that looks something like the following:



Next, let's specify the Geometry. Click on Geometry and then (after checking Specify Strategy if it isn't already checked), select soma in the left panel and then L and diam (if they aren't already checked). Now uncheck Specify Strategy and assign L=50 μm and diam = 50 μm. Next, similarly specify the axon as being very long and thin by choosing L=2000 μm and diam = 1 μm.

To have all of the changes you have been making get immediately updated in NEURON, click **"Continuous Create"**. As long as this is checked, every change you make to the Cell Builder will be immediately implemented for the next run of your model (if you do not have it checked, NEURON will hold off "creating" the model neuron you have specified until you do check it). To verify that NEURON really knows you have changed these parameters of the neuron, and to see what other default parameters NEURON assumes for your cell, go to the bash/nrngui/nrniv Window and type in **forall psection()** at the "oc>" prompt (first hit Enter if your cursor is not sitting next to this prompt). This command tells you "all" of the parameters ("p") in each "section". You should see a host of information about your cell, telling you that it has sections named "soma" and "axon" and telling you parameters such as the length L, internal/axial resistance Ra, diameter diam, and capacitance cm of each section. *You should get in the habit of typing this often to verify that you have correctly entered the parameter values in your model.*

We next need to tell NEURON how many compartments each section has. Since, in general, each portion of a neuron has a different voltage, as modelers we need to specify how big a portion of the neuron we wish to approximate as having the same voltage. The portion of the neuron that we assume to have the same voltage across its entire extent is called a "compartment" and in the simplest case, the single-compartment neuron, we approximate the whole neuron as having the same voltage. NEURON uses the word "segment" rather than "compartment" and actually specifies how many "segments" each "section" of the cell is broken into. Note that "sections" define different anatomical features of the neuron (e.g. the soma or individual branches of an axon or dendrite) whereas "segments" are purely used for computational purposes and tell NEURON how finely to discretize the sections when computing the voltage of the various portions of the neuron (ideally, we'd like to have a *continuous* spatial measurement of the voltage for every single point along the neuron, but this is computationally impossible as it would require that NEURON keep track of an infinite number of points. This is completely analogous to the way we discretize time into "time segments" of duration dt for computational purposes even though in reality time is a continuous variable).

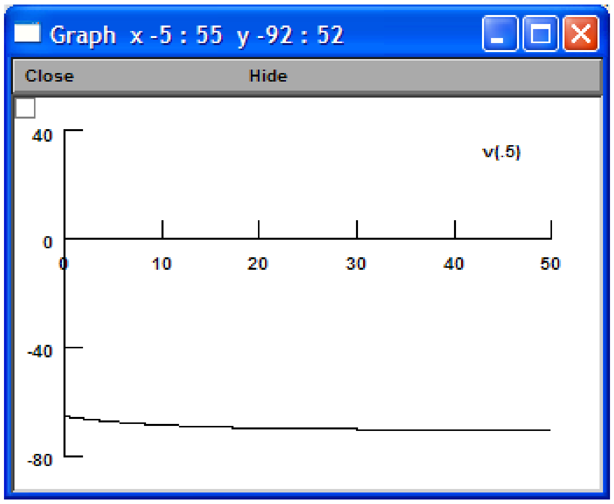
We can manually specify the number of compartments/segments in each section by setting the variable nseg in the Geometry window (If you check this, you'll see that, by default, each section has only 1 segment. Please uncheck the nseg box after doing this check, or you will end up overriding the d_lambda setting we are about to do next). Better than specifying the number of compartments with nseg, NEURON has a nice heuristic rule for calculating how finely to divide up the sections so that neighboring segments do not have very different voltages (e.g. neighboring segments may differ by a fraction of a mV).

To use this heuristic for the whole neuron, select "all" and then click **"d_lambda"** (*technical note: the terminology "lambda" refers to the length constant—basically, NEURON's heuristic is based on the realization that the voltage typically doesn't vary significantly between two points that are much less than one length constant apart. Note that the length constant used is actually defined for oscillating inputs rather than for the constant input typically described in introductory textbooks*). The default of 0.1 (i.e. 1/10 of a length constant) should work fine.

To find out how many segments NEURON decided to divide each section into, you can again type forall psection() at the oc> prompt in the nrniv Window. This command stands for "for all of the sections, give me a list of their parameters." When you do this, you should see the currently specified parameters for the soma and axon sections. Notice that NEURON left the soma as a single compartment but divided the axon into 43 compartments, which makes sense because, unlike the short fat soma, the axon is very thin and long and therefore will likely have quite different voltages across its length.

Next, under Biophysics, specify all sections to have Ra = 100 Ω-cm and verify that cm = 1 μF/cm^2. Because we wish to have *identical* passive properties in both of our sections, we can also use "all" to specify these properties. To do this, re-check Specify Strategy and then check pas for all. Next, assign e_pas = -70 mV (the default) and g_pas = 0.0001 S/cm^2 in these sections.

Now let's bring up a RunControl (reset Tstop to 50 ms) and a Voltage vs. time axis, and run the model. As output, you should see the following:

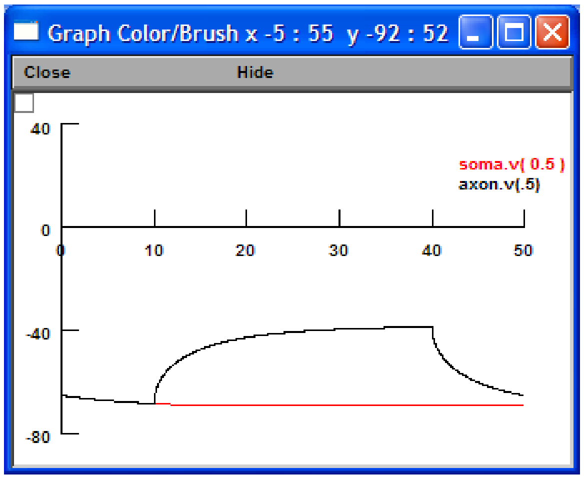


This shows exponential decay to the passive channels' reversal potential of -70 mV. But which section's voltage is being plotted? In this case, the default plot, which is simply called "v(0.5)", refers to the soma. However, to be more specific, you should use Plot what? to explicitly specify which section you would like plotted. Use Plot what>>soma>>v(0.5) to set up a plot of the voltage near the middle of the soma section. Then use Plot what>> axon>>v(0.5) to set up a plot of the voltage near the middle of the axon section. To do the axon specification, you will have to first choose (i.e. double-click) v(0.007…) for the axon segment and then manually change this to v(0.5) by typing in the white text-entry box near the top of the Window. You can always use the text-entry box – the clickable panels below are just (very) helpful tools to allow you to more easily enter information into this white box. If the labels v(.5), soma.v(0.5), etc. on the graph are on top of each other or not spaced as you would like them, you can use **(Gray Box)>>Move text** to rearrange them. Finally, delete the confusingly labeled "v(.5)" plot using **(Gray Box)>>Delete**, followed by clicking on the v(.5) label. Next set up the soma to produce a red trace (you can click on the word/label "soma.v(0.5)" and that will make the corresponding trace red when it plots). Now run again and you should get plots of both the soma and axon…although, for now, they should be identical so you'll only see one trace. Wait just one more moment please…

Next set up a current clamp near the middle of the axon section (e.g. at or near axon(0.5), see italicized note below) using the Point Process Manager. Note that now when you go to Show>>Shape you have a choice as to where to locate the current clamp (try clicking different locations along the picture of the cell) because there are many compartments/segments in which you could place it. Use View=plot if the figure is not centered. (*Note that the locations at which you can locate the current clamp are discretized and correspond to the locations of the centers of the compartments of the section. NEURON assumes that the voltage in a compartment/segment is the same throughout the compartment and, in particular, is equal to its value at the center of the compartment. If, for example when doing "Plot what?" on a graph, you tell NEURON to plot the voltage at a location that is not the center of the compartment/segment, then NEURON will ignore your request and instead (without telling you this!) give you the voltage at the center of the segment. Likewise, current clamps will always be located at the centers of segments. If you change the number of compartments (e.g. with nseg or d_lambda), then it is possible that your current clamp location will move slightly, so beware!!!).*

If you would like a more anatomically true picture of the neuron's morphology, click **(Gray Box)>>Shape style>>Show diam**. You should see that the wider soma is now drawn as a thicker square-like shape, as opposed to the thin line representing the axon (if you can't tell this, then try enlarging your window). Set the current to inject 0.1 nA of current, starting at t=10 ms and stopping at t=40 ms (so what value should go into dur?).

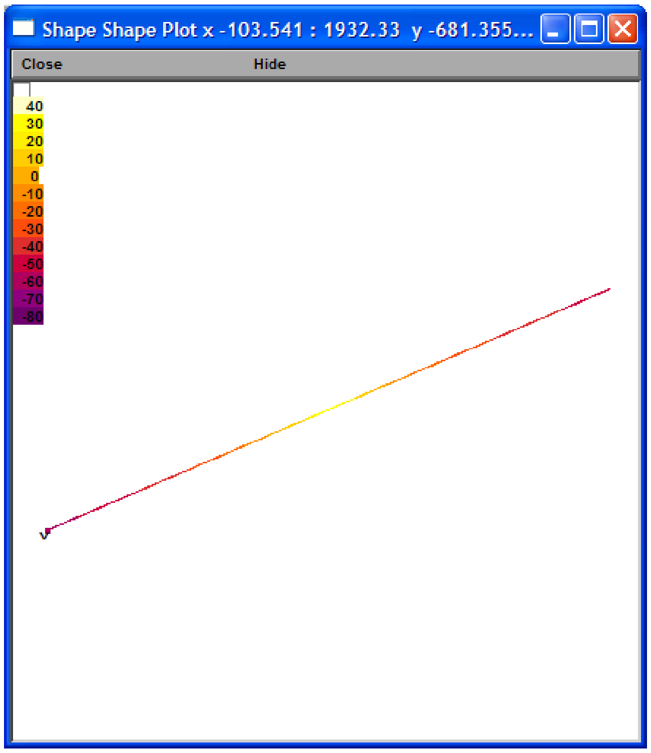
Run the simulation. You should see the following:



Notice that, because the axon is so long and thin, the current injected into the axon barely changes the soma's voltage (to confirm that it changed at all, you can use View = Plot).

We'd really like to see more specifically how the voltage changes over space. To do this, we'll need a new type of plot, the "shape plot". Click on **NEURON Main Menu>>Graph>>Shape plot**. You should see a not very interesting plot of the neuron's anatomy. To make it look a little more interesting and realistic, click (Gray Box)>>Shape style>>show diam and you should notice that the end is a little wider (again, you may have to enlarge the window to see this). Then click **(Gray Box)>>Shape plot** and you should see color(s) corresponding to different voltages (it will likely only be 1 color currently because this reflects the voltage at the end of the last run when the entire neuron was near the resting voltage).

Change the current injection amplitude to 0.3 nA and run again. Look at the shape plot—if you have very fast eyes, you should see the colors change corresponding to the spreading of the voltage away from the point of injection. To slow down the run so you can more easily notice this color change, try reducing dt in the RunControl by a factor of 100 and re-run your simulation (dt tells neuron how often to recomputed the cell's voltage and other dynamical properties). Next change the IClamp delay to 0 ms and duration to 50 ms and run again. The shape plot at the end of this run should give a nice illustration of the steady-state voltage distribution achieved and a feeling for the passive axon's length constant λ, i.e. how quickly the voltage decays over the length of the axon:

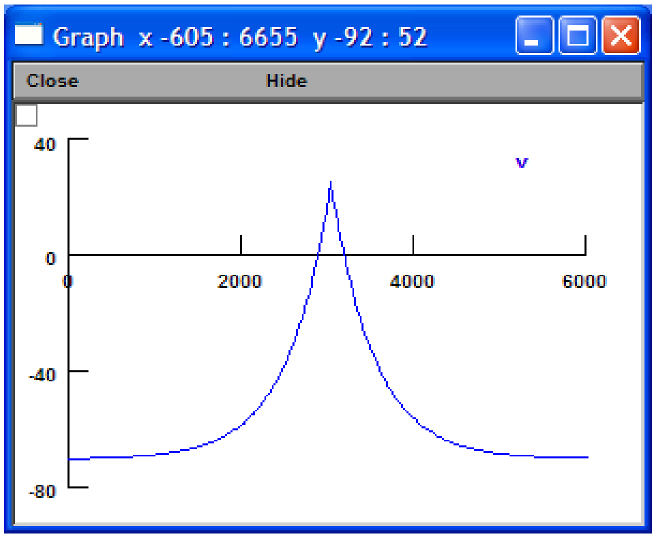


Your simulation may be running somewhat slower at this point (actually probably not perceptibly so…but for more complicated runs in the future this will be an issue). To see how long the last simulation took to run, check the Real Time (s) box in the RunControl window. To speed up the simulation, let's use another feature of NEURON: the variable time step dt. To do this, click on **NEURON Main Menu>>Tools>>VariableStepControl**. A new window entitled "VariableTimeStep" should appear. This menu contains details related to the algorithms used to integrate the differential equations governing our model. The important one we want to use is a variable dt, so check "**Use variable dt**". What this does is allows NEURON's differential equation solver to use different sized time steps depending on whether anything "interesting" is happening, where "interesting" means that some variables in the simulation are changing a lot. When little is changing, NEURON uses a large dt and runs faster. When variables are changing dramatically (e.g. during an action potential) NEURON uses a smaller time step dt. Now run your simulation again and notice how much faster it runs! You can **Hide** the VariableTimeStep window at this time if your screen is starting to overflow with windows. As a more general note, *hiding* windows is always safe as it will never change settings or cause information to be lost; it will just remove it from your view. By contrast, when you **Close** a window in NEURON, the window disappears for good and, in some cases, you may have to redo some of your work to get back what was in the closed window. Therefore, take care in which button you click!

Ok. Let's learn about one more plot type, the "movie" of voltage changing over time known as a "Space plot". On your Shape plot, click **(Gray Box)>>Space plot** (*note*: if you have resized this window, you may need to resize it again to see the Gray Box). The colors should disappear. Now select the entire length of the neuron by clicking your mouse at (or slightly beyond) the far end of it (near the "v", which indicates the default location at which voltage vs. time is plotted if one doesn't use Plot What to specify the exact location in the neuron that you would like to see the voltage) and dragging all the way across it (just past the other far end). A new graph should pop up which shows the voltage as a function of spatial location (measured in μm from the end you first clicked).

Next, turn off (i.e. uncheck) the Use variable dt box (you can bring this window back up through NEURON Main Menu>>Window) and check that your time step dt is still at 0.00025. Then run your simulation again. You should see in the new plot a beautiful movie of the voltage changing over time! The final voltage configuration illustrates nicely the concept of a length constant—it is peaked at the location of current injection and decays exponentially as one moves further away down the length of the axon.

Let's make our axon longer, say L=6000 μm, and then re-select the whole axon to plot in the space plot (you may first want to do a View=plot in the Shape plot picture to recenter it). A new graph should come up. *If it does not, then I recommend going back to the Space plot, turning it into a Shape plot using the gray box, and then turning it back into a Space plot and re-doing the selection of the whole axon; by the way, errors of these types are typically recorded in the nrniv window so when things go wrong or look suspicious I highly recommend looking there for an error message*. Then run your model again. You should see a movie that ends as follows:



This illustrates that the passive length constant λ of this axon is roughly in the range of 0.5mm, a typical value for thin branches of neurons.

At this point, your model parameters (as ascertained by typing forall psection() at the oc> prompt) should be (if you played with other things during the course of the tutorial, it is possible there will be additional lines; just make sure that the parameters below match yours and that you do not have any "insert" lines other than what is given below):

```
oc>forall psection()
soma { nseg=1  L=50  Ra=100
	/*location 0 attached to cell 0*/
	/* First segment only */
	insert morphology { diam=50}
	insert capacitance { cm=1}
```

```
        insert pas { g_pas=0.0001 e_pas=-70}
}
axon { nseg=213  L=6000  Ra=100
        soma connect axon (0), 1
        /* First segment only */
        insert pas { g_pas=0.0001 e_pas=-70}
        insert morphology { diam=1}
        insert capacitance { cm=1}
        }
```

Now save your session to a *different* filename from your last NEURON exercise (such as PassiveCable_*YourNameHere*.ses) by typing this name after the slash (/) in the white box at the top of the save menu (and remember that you can use "recent dir" to do this more easily).

Good work!

VI. Notes on debugging NEURON programs

NEURON can at times be frustrating because the interface is rather "old school" and it is easy to make mistakes when entering information. I strongly recommend the following tips when trying to debug a NEURON program that seems to be performing oddly:

1) Check the bash/nrngui/nrniv window regularly to see if a crash occurred (if so, there will be a message indicating an error and typing Enter will cause NEURON to close).

> *If a crash did occur,* ***do not try to fix*** *this file. You will have to start over from a file that you are sure did not crash. To avoid losing too much information…*

2) Save often, and to a new filename each time (e.g. number your files 1,2,3,…as you progress through a programming assignment) so if a crash does occur, you will not have to start from scratch.

3) I find the following "algorithm" useful for debugging code that is not working:

(a) Do a step of your programming assignment.
(b) Look for a crash.
(c) If there was no crash, save *under a new file name*. If there was no crash, but you are not getting the expected result, type forall psection() at the oc prompt in the nrniv window to check that the listed parameter values are what you expect.
(d) Repeat (a)-(c) above with the next step of the programming assignment.
(For programming of networks, as we will see in the following Network builder tutorial, this will involve having to create the network; uncreate it, save, and quit NEURON; then re-open NEURON in order to be able to add the next step & create the network again).

VII. For those interested…Hoc files

If you are interested in learning what the NEURON code looks like for what you have done, you can, within your CellBuilder, use **Management>>Export>>Export to file** (you can then type the name for your cell, which should be of the format *filename*.hoc) to save your code to a ".hoc" file. The Export method of exporting simply exports your code to a file. If you want to look at the .hoc code generated by either of these exporting methods, I recommend you use a "plain vanilla” text editor like Wordpad rather than Microsoft Word because you don’t want any fancy formatting symbols sneaking into your code – use rich text format (rtf) for the formatting.

VIII. Getting help

The main sources of documentation for more advanced NEURON programming are *The NEURON Book* and the NEURON website and subsidiary information pages:

Carnevale NT & Hines ML (2006) *The NEURON Book*, Cambridge University Press.
NEURON’s website:
http://www.neuron.yale.edu/neuron/
NEURON course information, including exercises and material/slides/tutorials from a multi-day intensive course the authors of the program offer:
http://www.neuron.yale.edu/neuron/courses
NEURON on-line documentation:
http://www.neuron.yale.edu/neuron/docs
This contains many useful tutorials, as well as the NEURON “Programmer’s Reference” manual. For tutorials showing how to build more complex neurons, check out the CellBuilder tutorial.