Software Experiment #1
HOW DO WE MODEL A PASSIVE CELL MEMBRANE?

Content areas: Basic neuroscience covered in biology, physiology, psychology, and engineering courses

Pre-requisite knowledge: Cell biology, human nervous system, first order systems

Learning Objectives: After this lesson, students should be able to:

- understand the structure of a cell membrane
- basics of electrical components such as resistance and capacitance, and Ohm's law.
- develop a model for a first order system, and determine its time constant

Time Required:

Keywords. Cell membrane, resistance, capacitance, time constant

Summary. A cell membrane can be modeled as an electrical circuit with capacitance (bi-lipid layer) and resistance (various channel proteins embedded in it). Application of Ohm's law to this simple circuit yields a first order equation. Building on the concepts in Software Expt. 0, students are introduced to developing models of first order system. Then the passive cell membrane equation (first order system) is 'solved' to determine the evolution of the membrane potential as a function of time with a constant current injection.

SOFTWARE EXPERIMENT #1 ASSIGNMENT

#1. What are the steps in modeling a first order system? How do we model such a system?
We use the term 'model' to represent the set of mathematical equations that predict the evolution of a 'system' (biological, physical, ….), given specific inputs. Think of a 'system' you would like to model, with one input and one output and should be 'dynamic', i.e., should have a first order differential equation representation.
(a) Describe the system mathematically, define the input, output, and parameters
(b) then write the first order equation in the standard form (τ*dy/dt + y = u; where τ is the time constant, y is the output, and u represents all the other terms that do not depend on y, such as input, etc.).
(c) What does your τ depend on, and does that make sense to you?

#2. Modeling the cell membrane as a capacitor using NEURON (see later pages on how to install NEURON)
How would you begin the process of modeling a cell membrane to predict the membrane potential changes?
EXERCISE: Model a passive cell with membrane capacitance and leakage current, and analyze it using the software package NEURON. You can use the code given on page 3, on a computer that has NEURON.
Input: Current injection (constant in Amps, to be set by user); Output: Membrane voltage
Parameters: Membrane maximal conductance=2.0*10^{-5} S/cm2; Area=6.3428*10^{-4} cm^2; Total membrane capacitance Cm=1.4884*10^{-10} Farads; Reversal potential for leak channel, Em= -0.07 V; initVm= -0.06 V.
Note: In NEURON, for 2.0*10-5, you will have to write 2.0e-5.
Analysis: (i) Plot the membrane voltage from 0 to 800 ms, for at least three different current injection values (use no current from 0 to 100 ms, and from 600 to 800 ms, i.e., current pulse is from 100 to 600 ms only). Try current = 1*10^{-12} A and up in multiples of 10; copy the plots into a word document and save them on your flash drive; (ii) determine the time constant from each of the three plots by taking 63.2% of the change as described in class; (iii) without plotting, discuss what happens if the value of G_leak is reduced to 1.0e^{-5} Seimens?

(a) Describe the system mathematically, define the input, output, and parameters
(b) Then write the first order equation in the standard form (τ*dy/dt + y = u; where is the time constant, y is the output, and u represents all the other terms that do not depend on y, such as input, etc.).
Substitute numbers from above and get the first order equation with numbers only, i.e., no symbols.
(c) Does the time constant τ match with the value in the plot? Does τ depend on the input?
(d) What other predictions can the model provide? For instance, what happens when a parameter such as the membrane resistance is doubled? Discuss all such possible cases of using the model to make predictions that might be tested by a biologist in the wet lab.

#3. How does a software package solve a first order differential equation? Complete worksheet on one of the following pages. Remember to complete all parts of the worksheet, and use the reverse side if needed.

#4. Time constants characterize the dynamics of a first order system completely.
Consider any first order linear differential equation, e.g., for a passive membrane. The solution for such an equation is described uniquely by three parameters. For the passive membrane, the solutions can be written in the form V(t) = A+Be$^{-t/\tau}$, where τ is defined as the system time constant, and A and B depend on initial conditions. You are to pick three values each for the pair (A, B), and use constants of 0.1 msec, 100 msec and 3 sec for τ, respectively, i.e., come up with three arbitrary equations with each of the τ values. Then plot them using WolframAlpha (or any plotting package) and verify that the time constants you get by hand calculations using the plots (recall: 63.2% of 'change') match with what you intended it to be.

#5. Exercises on the last page of this handout – using the demo NEURON program provided in the document 'Installing NEURON and getting started' which is included at the end of this document.

<u>NEURON Program for #2</u> <u>(save this as passive.hoc using the Notepad++ editor)</u>

```
// passive.hoc   MODELING A SINGLE CELL WITH PASSIVE MEMBRANE
// AND CURRENT INJECTION

load_file("nrngui.hoc")

create soma
access soma
soma nseg = 1
soma diam = 200                     //um
soma L = 100                        //um

//Note that NEURON assumes cm, and gl_hh as being in units of /cm2
soma cm = 1.4884e-4/6.2832e-4       // uF/cm2

// inserts the passive (leak) channel and assign value.
soma insert hh
gnabar_hh = 0
gkbar_hh = 0
gl_hh = 2.0e-5    //S/cm2
el_hh = -70 //mV

//set initial value of voltage to -60 mV; default is -65 mV
v_init = -60

objectvar stim
//create current clamp in the middle of the soma
soma stim = new IClamp(0.5)

stim.del = 100                // in ms
stim.dur = 500                // in ms
// NEURON assumes injection current to be over the entire area
stim.amp = 1                  // in nA

tstop = 800       // in ms

// use procedure on the previous pages to plot voltage v/s time
```

Worksheet for #3 How do packages perform numerical integration, i.e., solve DEs

Solve the first order differential equation you obtained in part 2b by hand, to reinforce how numerical integration is performed. Recall that you are to use the definition of a derivative to do this. Assume $\Delta t = 1$ ms. Work out by hand the solution from t=0 to t=5 ms. Compare the values of V(t) with that you obtained from WolframAlpha or NEURON by writing them side by side in two columns. Discuss your findings and conclusions clearly. For instance, how would you increase accuracy of numerical solutions. Use the reverse side of the page if needed.

NEURON – A computational modeling package designed for neurobiologists

Software exists presently to model systems in neuroscience at typically only one of the levels, either molecular, cellular, or network/systems level. One reason for this is the large difference in both temporal and spatial complexities between the levels. The reader is referred to [1] for a discussion of software related to molecular level modeling. In this course, we will introduce you to software designed for modeling at the cellular level.

Computational modeling platforms at the cellular and network levels range from general purpose software such as Matlab (http://www.mathworks.com/) which directly model the mathematical equations, to special purpose public domain ones such as GENESIS [2] and NEURON [3] which are being designed for biologists, and require minimal understanding of the underlying mathematics. Figure 2 shows the hierarchical structure used for modeling in GENESIS. The packages can perform simulations of models ranging from single neurons to complex networks representing brain circuits.

Once the mathematical equations/model for the neuron or network are developed (…in the modeling segment of this course), the values of the biophysical parameters have to be determined systematically. In addition to research articles, sources for such information include databases such as CellPropDB, NeuronDB and ModelDB (http://neuron.duke.edu/).

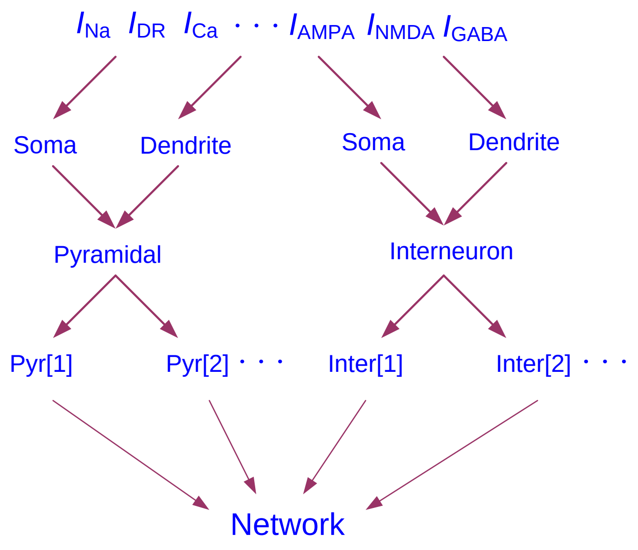


Figure 1. Elements in GENESIS are organized in a tree structure [2]. The symbol 'I' represents current, e.g., I_{Na} is the sodium current. The network comprises cells (pyramidal and interneurons in this schematic), which in turn consist of soma and dendrites populated with the various current channels.

[1]. Buckingham, SD, To build a better model, *Nature Methods,* 2007;4(4):367-369.
[2]. Bower JM, Beeman D, *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System*, Internet Edition, 2003. http://www.genesis-sim.org/GENESIS/.
[3]. Carnevale NT, Hines ML, *The NEURON Book*, Cambridge, UK: Cambridge University Press, 2006.

Downloading and Installing the NEURON Software

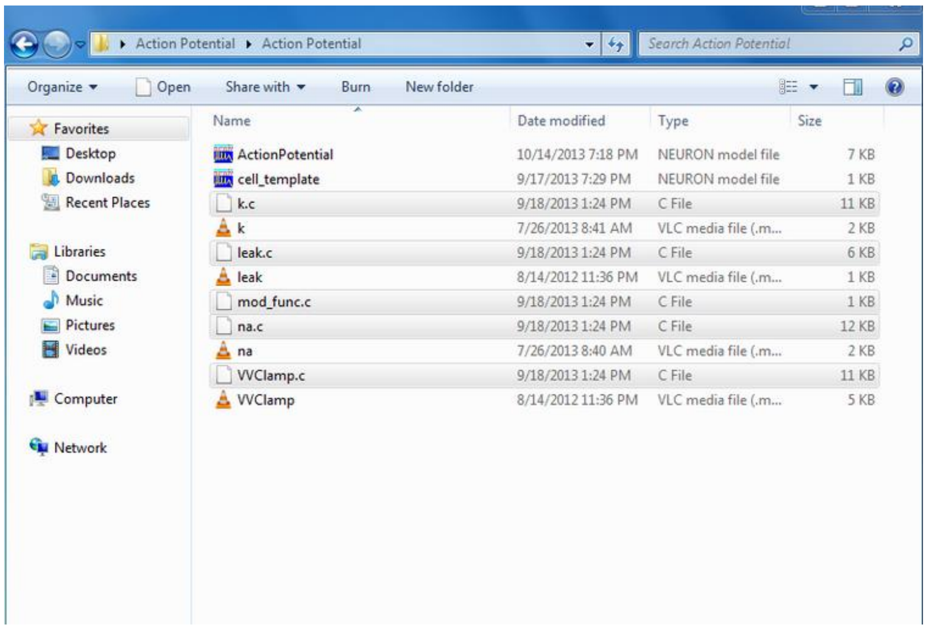
1. Go to the site http://www.neuron.yale.edu/neuron/download

2. Depending on the computer you are using (Mac, MSWin, ..) click on the corresponding button on the left-bottom side. Note: Most MSWin users should use the 64 bit Cygwin-based installer. The MinGW installers are only for those who need to use Enthought Canopy. This will begin the process of download which takes less than 5 minutes.

Getting Started

1. Decide on the directory you will working from, i.e., where you will have your NEURON codes resident. Suggest you open a new directory on C:\ drive (or wherever you would like) and call it NEURON SIMULATIONS. For MAC users put the NEURON file in the Application folder.

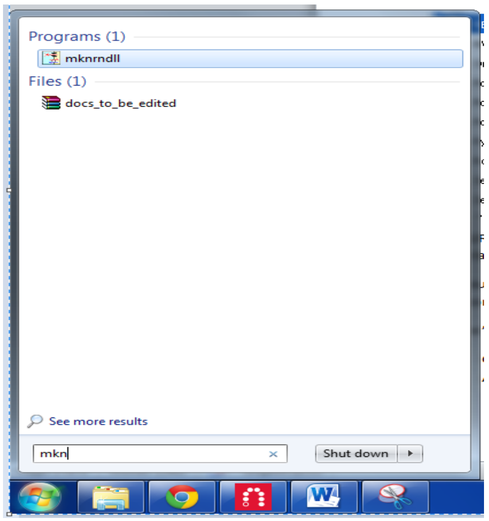
2. If your NEURON codes are in a .zip file, extract the files in the folder.



Name	Date modified	Type	Size
ActionPotential	10/14/2013 7:18 PM	NEURON model file	7 KB
cell_template	9/17/2013 7:29 PM	NEURON model file	1 KB
k.c	9/18/2013 1:24 PM	C File	11 KB
k	7/26/2013 8:41 AM	VLC media file (.m...	2 KB
leak.c	9/18/2013 1:24 PM	C File	6 KB
leak	8/14/2012 11:36 PM	VLC media file (.m...	1 KB
mod_func.c	9/18/2013 1:24 PM	C File	1 KB
na.c	9/18/2013 1:24 PM	C File	12 KB
na	7/26/2013 8:40 AM	VLC media file (.m...	2 KB
VVClamp.c	9/18/2013 1:24 PM	C File	11 KB
VVClamp	8/14/2012 11:36 PM	VLC media file (.m...	5 KB

3. Delete all the .o, .c, and .dll files from the folder if they are present. Note that you can group files into the various types by clicking on ‘Types’ on the right top in the picture above.

2. For Windows users (MAC users skip to step 6): To compile your NEURON files, go to the ‘Start’ menu on your computer.

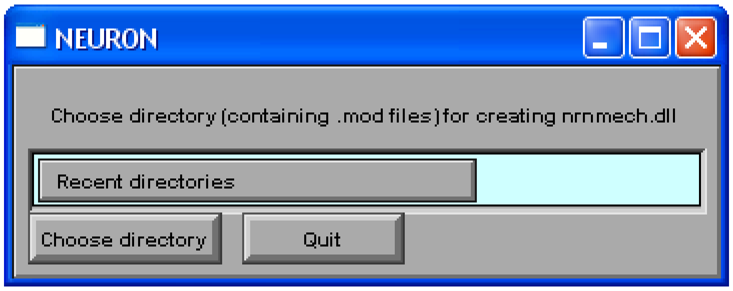


3. NEURON has two main file types, files with extensions '.hoc' and '.mod'. The former can be run directly while the latter has to be compiled. To run a '.hoc' file, just double click it if you are using Windows or Mac.

4. If you have '.mod' files in your directory, or if you have moved all your files to a new directory, you will have to compile them.

   Here's how you compile '.mod' files:

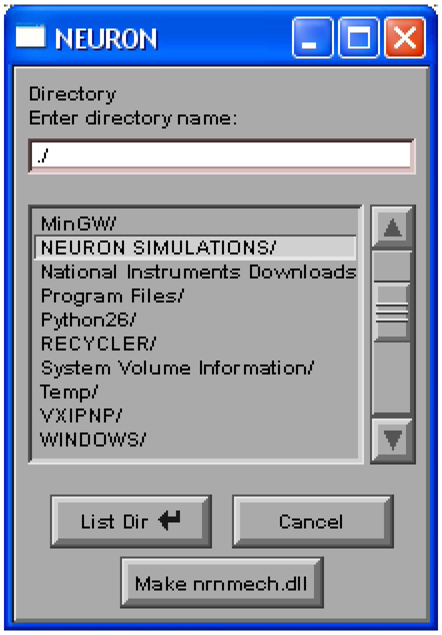
   - Go to Start → Search Bar → mknrndll
   - A window should appear asking you to choose a directory where the '.mod' files are located.

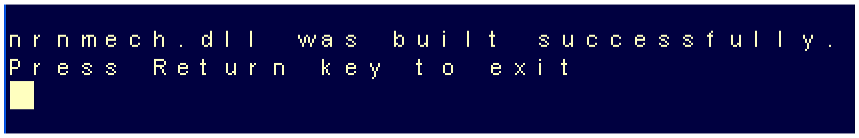
     Choose directory (containing .mod files) for creating nrnmech.dll

     Click 'Choose directory', then change to the directory you have the files in, i.e., NEURON

SIMULATIONS.



- Then click ‘Make nrnmesh.dll’. This compiles the ‘.mod’ files. If the compilation is completed successfully, it will return the following screen, and you press RETURN to exit successfully.



```
nrnmech.dll was built successfully.
Press Return key to exit
```

5. Every time you change of a ‘.mod’ file, or copy it to another location, you have to recompile as described above since only then will NEURON know the path, etc.   Remember to do this! Also, remember to close all files before compiling.

6. FOR MAC USERS: In order to compile .mod files on a mac, you must install the developer tools package. If you are running on OS*X 10.6 or below, you can find these tools on the installation CD that came with your computer. If you are running on 10.7 or above, you can download the XCODE app for free from the app store. IMPORTANT: After installing the XCODE app, you must open the xcode preferences window (found from clicking on the XCODE button next to the apple symbol in the top tab of your mac). From the preferences window, you must go to downloads tab and download the packages (5 or 6, the most important one being ‘Command Line Tools’) that you see there. If the Apple Server is down, then these packages might not show up on the list, and in this case you need to wait till the server if functional.

After installing XCODE, open a terminal window and type *gcc –v*. If you get the error message saying *command not found*, you have not followed the above instructions for installing the necessary XCODE packages. Use the instructions above (found after IMPORTANT) to find the downloads pane in XCODE and install all additional packages.

MAKE SURE NEURON IS INSTALLED IN your "APPLICATIONS" FOLDER (where programs such as Microsoft Word and itunes are installed), ELSE NEURON WILL CRASH WHEN nrngui IS CALLED…

Compiling .mod files on a MAC - Open the NEURON folder (in the Applications folder) and then select your particular program folder (entire folder). Then drag and drop this folder into the "mknrndll" FILE in the NEURON folder. If you have done it correctly, it will give a message saying that the .mod files have been successfully compiled.

Some more information about usage of NEURON on MACs can be found at the site - http://www.neuron.yale.edu/neuron/static/docs/nmodl/macos.html

Also, check out the NEURON discussion board at:
https://www.neuron.yale.edu/phpBB2/viewforum.php?f=4

Sample Tutorials

1. NEURON Tutorial by Gillies and Sterratt http://www.anc.ed.ac.uk/school/neuron/
2. Simple tutorial presented at 1997 SfN –
   http://www.neuron.yale.edu/neuron/static/docs/elementarytools/outline.htm

RUNNING YOUR FIRST MODEL

(adapted from "Tutorial Part A" at the site http://www.anc.ed.ac.uk/school/neuron/)

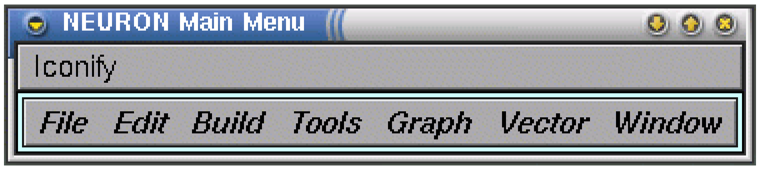
1. Open a text editor such as Notepad or Wordpad and copy the following text into it, and name the file 'Single Cell Model.hoc' (note: the extension has to be '.hoc')

Notice: Make sure the file is saved in the 'text' format (ANSI encoding).

```
load_file("nrngui.hoc")

create soma
access soma
soma nseg = 1
soma diam = 18.8  //Default value is 500 um
soma L = 18.8     //Default value is 100 um
soma Ra = 123     //Default value is 35.4 ohm-cm
soma cm = 1       //Default value is 1 uF/cm2 (specific membrane capacitance)

//The following command inserts the Hodgkin-Huxley Na and K channels, after
which //the channel variables gnabar_hh, etc. are defined......and so can be
accessed.
soma insert hh

gnabar_hh = 0.12  //Default value is 0.120 S/cm2
gkbar_hh = 0.036  //Default value is 0.036 S/cm2
gl_hh = 0.0003    //Default value is 0.0003 S/cm2
ena = 55          //Default value is 55 mV
ek = -77          //Default value is -77 mV
el_hh = -54.3     //Default value is -54.3 mV
```

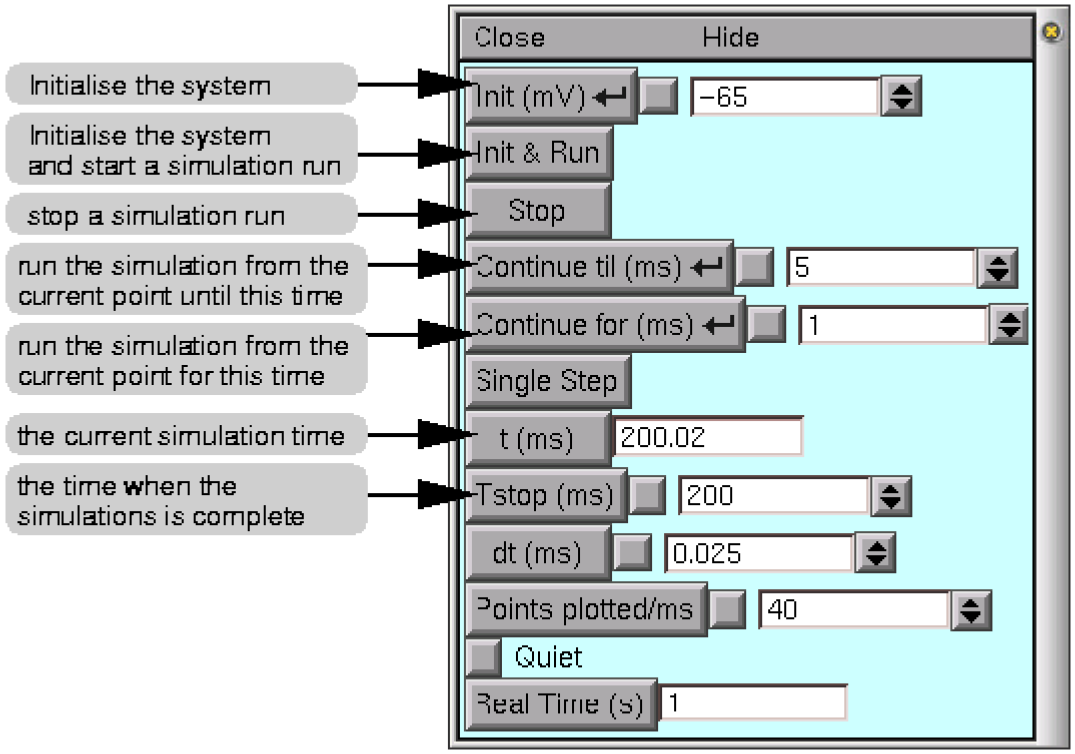
```
objectvar stim
soma stim = new IClamp(0.5)    //creating current clamp in the middle of the
soma

stim.del = 100           // in ms
stim.dur = 100           // in ms
stim.amp = 0.1           // in nA

tstop = 300       // in ms
```
____

2. Double-click on the file and it will run it (if you have NEURON installed). It will show a graphical window as below:

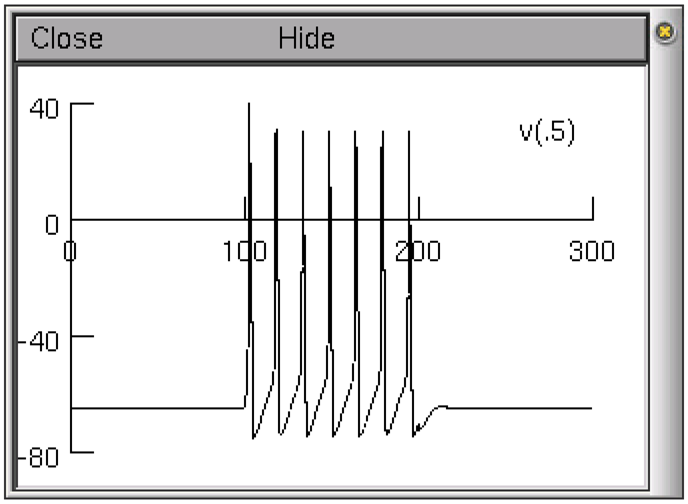


This is the *NEURON* Main Menu toolbar, which can be used to pop up graphical tools for controlling, displaying, and analysing model parameters and simulation results. The window contains menus *File*, *Edit*, *Build*, *Tools*, *Graph*, *Vector* and *Window*. For our purposes here we will explore the GUI methods for running a simple simulation. Under the *Tools* menu select *RunControl*. This generates a window that allows you to control primary parameters for running the simulation, such as the parameter `tstop` described above.

The *RunControl* window, as with many other windows you will encounter, has a common format. There are buttons, such as "*Init & Run*" which will run the simulation, and *field editors* next to some buttons (e.g. next to our `tstop` variable button). Whenever you see this type of box, you can enter a new value by simply placing the mouse pointer into the field editor, clicking the left mouse button, and typing.


Initialise the system
Initialise the system and start a simulation run
stop a simulation run
run the simulation from the current point until this time
run the simulation from the current point for this time
the current simulation time
the time when the simulations is complete
Close
Hide
Init (mV)
-65
Init & Run
Stop
Continue til (ms)
5
Continue for (ms)
1
Single Step
t (ms)
200.02
Tstop (ms)
200
dt (ms)
0.025
Points plotted/ms
40
Quiet
Real Time (s)
1

A vertical bar should appear in the field editor, indicating the point where what you type will be seen. After you have entered the new number (or expression), you need to tell *NEURON* to accept the number (or expression) you just entered. You can do this either by pressing the *Enter* key on your keyboard, or by clicking on any button in the *RunControl* panel. Some labelled buttons have a smaller, unlabelled square button just to their right. This is a *check box*, and a red check mark will appear in it if the value in the adjacent field editor has been changed from its default. Thus, if you change the value of *tstop*, a check mark will appear in the check box signifying that you have changed the value of this field editor. In addition, by pressing the check box, you can toggle between the default value and the changed value.



In addition to controlling the simulation we will want to observe the voltage changes in the soma over the period of simulation (300ms in our current example). Under the *Graph* menu of the main window, select *Voltage axis*. This generates another window that will display voltage. When there is more than one section (e.g. when we have added dendrites - outlined in Tutorial part B) then we must specify which section voltage to plot. In the current example the soma is the default section, so the soma voltage will be plotted by default. You can open as many voltage or other graph windows as you like.

Now, run the simulation (by clicking on the *Init & Run* button in the *RunControl* window), observing the voltage being graphed. You should end up with a voltage graph similar to that illustrated here.

Installing Python + NEURON for Windows (Optional)

Installing Python and NEURON separately will not allow you to program NEURON models in the Python language if you are using the Windows operating system. One must download the Anaconda package of Python for this to work. Anaconda includes all of the normal Python distribution plus packages designed for science and engineering.

Start with neither Python nor NEURON installed on your computer (make sure you uninstall all versions of both before you proceed). Then go to (https://www.continuum.io/downloads) and download the Anaconda package that includes Python 2.7. Follow the on-screen instructions and accept the defaults.

After Anaconda has finished installing, download the NEURON distribution from (https://www.neuron.yale.edu/neuron/download) corresponding to your machine. You will now be able to write NEURON models in the Python language. Note that Python files do not need to be compiled in the way that HOC files do. Note that you run it as a NEURON file, not python file.

EXERCISES:

1. Run the simulation with the parameters provided. Then change it to the default parameters and run it again. Notice how you can change the parameters and alter cell firing characteristics. Use the default parameters for the problems below.
2. Try to block the Na channel and observe what happens. Can you explain it biologically?
3. Now repeat the same with the K channel
4. What is the maximum cell firing frequency using any current injection? What constrains this frequency from a biological viewpoint?
5. Obtain a frequency v/s current injection (FIR) curve for the cell.