Neural Computation
Practical 1: Getting to grips with NEURON and passive properties

David Sterratt

2003/2004 version by Mark van Rossum

1 Aims

This practical has two aims:

- To acquaint you with the NEURON simulator
- To investigate passive properties of neurons

2 Starting and stopping Neuron

How you run NEURON depends on the operating system you are using.

2.1 DICE network

NEURON is installed in `/usr/i686/bin/`. To start NEURON, type

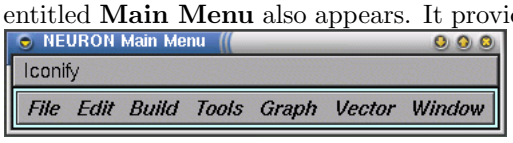
```
/usr/i686/bin/nrngui
```

at the UNIX command line. Or you can add `/usr/i686/bin/` to your PATH and start it with

```
nrngui
```

Some text about the version number of NEURON etc. should scroll past. At the end of the text you should see a prompt like this

```
oc>
```

A new window entitled **Main Menu** also appears. It provides access to the graphical user interface and looks like this: 

This prompt tells you that anything you type at the prompt will be interpreted by NEURON. NEURON understands commands in a programming language called HOC (pronounced *hoak*).

To get out of NEURON, type `quit()` or `Ctrl-D`. You should see the operating system command prompt again.

2.2 Other operating systems

NEURON runs under Windows and MacOS — follow links from `http://www.neuron.yale.edu/` for free downloads.

2.3 Help and more info

Go to `http://www.neuron.yale.edu/` and sublink `http://www.neuron.yale.edu/neuron/docs/docs.html`.

3 Creating a compartment

We will first create the simplest possible model of a patch of membrane: a single passive compartment. Figure 1 shows the equivalent circuit. It comprises a membrane capacitance C_m, a membrane resistance R_m, a leakage reversal potential E_L and a current source I_e. We will imagine our circuit represents a soma, though in fact it misses some crucial features of a real soma, namely active properties.

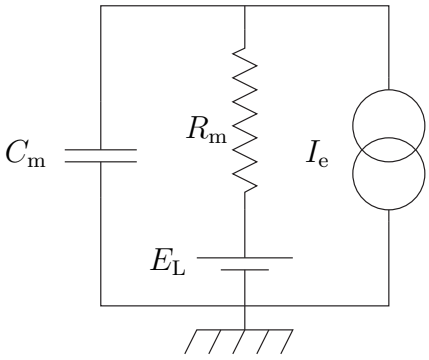


Figure 1: The single compartment

Start NEURON, as explained in the first section. Next type

```
create soma
```

This creates a *section* called `soma`. In NEURON a section is a geometrical unit rather than an electrical one. Each section is divided into one or more *segments*. Segments represent small patches of membrane which we assume to have the same membrane potential — they are another name for the *compartments* of a *compartmental model*.

Next type

```
access soma
```

This means that whenever we refers to properties of a section, NEURON will by default assign them to our `soma` section. In practice, it's better to refer to sections explicitly all the time rather than relying on the default section. However, we need at least one `access` statement for the graphical user interface to work.

Now type

```
soma psection()
```

This prints out the properties of the section we have just created (though in a somewhat strange format; this is because the output is designed so that HOC as well as humans can read it). When NEURON creates a section, it gives it default values, as shown in table 1.

Variable name	Meaning	Value	Units
`nseg`	Number of segments in section	1	
`L`	Length of section l	100	μm
`Ra`	Specific axial resistance R_a	35.4	$\Omega\,$cm
`diam`	Diameter d	500	μm
`cm`	Specific membrane capacitance C_m	1	$\mu\mathrm{F/cm}^2$

Table 1: Default properties of a section.

So far, we have introduced two passive parameters, the specific membrane capacitance C_m and the specific axial resistance R_a. We have still to introduce the *specific leakage conductance* $\overline{g}_\mathrm{L}$ and its associated leakage reversal potential E_L. To do this we must insert channels into the membrane. NEURON has a special channel type called `pas` to simulate the membrane resistance and reversal potential. To insert it into the membrane, type

```
soma insert pas
```

On typing `psection()` you should notice that there are now two extra properties, as shown in table 2.

Variable name	Meaning	Value	Units
`g_pas`	Specific leakage conductance $\overline{g}_{\mathrm{L}}$	0.001	$\mathrm{S/cm^2}$
`E_pas`	Leakage reversal potential E_{L}	-70	mV

Table 2: Default properties of the `pas` channel.

The specific membrane conductance is often given in terms of the *specific membrane resistance* R_{m}. Since conductance is the inverse of resistance the membrane resistance in this case is $r_m = 1/0.001 = 1000\,\Omega\,\mathrm{cm}^2 = 1\,\mathrm{k}\Omega\,\mathrm{cm}^2$.

A more typical value for the specific membrane resistance is $r_m = 10\,\mathrm{k}\Omega\,\mathrm{cm}^2$. This means that the specific membrane conductance should be $0.0001\,\mathrm{S/cm^2}$. To change this, type

```
soma g_pas = 0.0001
```

To confirm that you have changed the properties, type `soma psection()` again.

4 Injecting current into a single compartment

We now have now built a single compartment with passive leakage. To test its properties we will inject some current. To do this, type:

```
objref stim
soma stim = new IClamp(0.5)
```

The first line creates a new reference to an object called `stim`. The second line makes `stim` refer to an `IClamp` object placed halfway down the `soma`.

The job of the `IClamp` object is to inject current into the section. In order to do this, you need to set some properties of the `IClamp` as follows:

```
stim.amp = 1
stim.del = 100
stim.dur = 100
```

This will inject a pulse of current with an amplitude of 1 nA starting 100 ms (`stim.del`) after the simulation has started and lasting for 100 ms (`stim.dur`). Note that we cannot refer to the `IClamp` `stim` in the same way that we can a section. Therefore, we have to refer explicitly to `stim` whenever we want to set its properties using the *dot* notation.

5 Running the simulation from the HOC command line

Now all that remains to do is to run the simulation. [1]

Before we run the simulation, look at the membrane potential of our compartment by typing

```
soma print v
```

Note down the value, and now type

```
run()
```

This runs the simulation for 5 ms of simulated time. Type

[1] We need to load some standard libraries of functions into NEURON by typing `load_file("nrngui.hoc")` In modern versions of NEURON this library is normally loaded on startup, so the line is redundant. However it is needed for portability (Windows users who click on HOC files from Windows Explorer don't get this unless the file contains the statement.

```
soma print v
```

again. You should see that the membrane potential has changed. You can set the period of time the simulation runs for by changing the value of the global variable `tstop` as follows

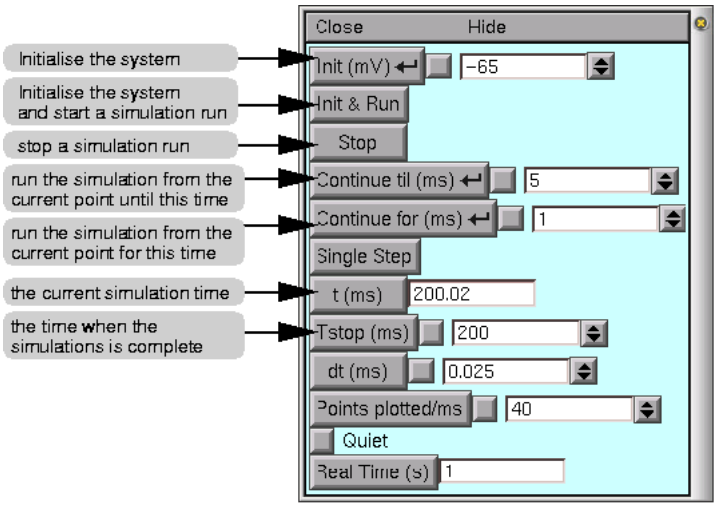
```
tstop = 300
```

When you next type `run()` the simulation will run for 300 ms of simulated time.

6 Running the simulation from the Graphical User Interface

Of course it would be tedious to run a simulation in the way described in the previous section. To see our results plotted in graphs we need to use the graphical **Main Menu** that popped up when NEURON started.

This 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 moving the mouse into the field editor box and start typing. A flashing vertical bar will appear when you start typing and signifies that you are editing the value in the field editor. 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 by either pressing the **Enter** key, by clicking on the button associated with the field editor or, if there is another button (called a check box) between the main button and the field editor box, you can click on that button. The check box is used to toggle between the default value and your changed value. 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. Then, by pressing the check box, you can toggle between the default value and the changed value.

7 Viewing the results graphically

NEURON can produce plots versus time on a number of different types of axes: the **Voltage axis**, the **Current axis** and the **State axis**. In this section we will look at the voltage and current axes. [2]

[2]Each of these require their own graph since the method used to solve the cable equations, developed by Mike Hines, calculates each of these variables at different times. For example, voltages are calculated at each time step, but currents are calculated at the half time step before voltage and the state variables are calculated at the half time step after the voltage. For more information on the numerical method used to solve these equations see **?**.

7.1 Voltage axis

In addition to controlling the simulation we will want to observe the voltage changes in the soma over the period of simulation (300 ms 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) then we must specify which section voltage to plot. In the current example, as we have set the `soma` as the default section in our commands above, 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 (using the **Init & Run** button in the **RunControl** window), observing the voltage being graphed.

To look at the graph in more detail, click the *right* mouse button over the **Voltage Axis**. Select **View...** and then **Zoom in/out** from the menu that pops up. (Throughout the rest of this document I will abbreviate selecting from submenus with an arrow. For example "**View...**→**Zoom in/out**" means "select **View...** and then **Zoom in/out**".)

- Now clicking on the plot with the left mouse button and dragging the pointer *upwards increases* the vertical scale; dragging *downwards decreases* the vertical scale. Similarly dragging right increases the horizontal scale and dragging left decreases it.
- Clicking on the plot with the middle mouse button and dragging moves the trace relative to the axes.

This is easier to do than to describe, so take some time to practise moving the plot around and zooming in and out. Once you have zoomed in, you should be able to see a plot like figure 2.

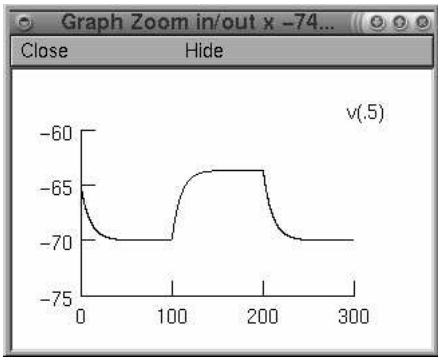


Figure 2: The voltage axis. The trace shows the membrane potential plotted versus time.

7.2 Current axis

It is also useful to plot the different currents flowing through the membrane. To do this we must select **Graph**→**Current axis**. This brings up an empty plot. If you try to **Init & Run** no plot appears as it does in the case of the **Voltage axis**. This is because no quantities are selected by default.

To select some currents to plot, **right click** on the current axis and chose **Plot what?** from the menu. This should bring up a window from which we can choose a variable to plot. Double-click on `soma.` in the first column. This should bring up a list of quantities in the second column. Click on `i_cap(0.5)` ("the capacitive current measured 0.5 of the way down the section"). In the text box at the top of the window[3] you should now see `soma.i_cap(0.5)`. Click on the **Accept** button. You should see the legend `i_cap(0.5)` appear on the current axis.

Now run the simulation using **Init & Run**. Initially it will look as though nothing is appearing on the axes. However, this is because the scale is too small. You could try zooming in as in the previous section. However, a convenient way of setting the scale is to right-click and select **View**→**View = plot**.

[3]The variables can be either directly typed in if we know the exact name of the variable we want to plot.

7.3 Plotting more than one quantity on the same axes

Suppose we would like to plot the leakage current through the cell membrane as well as the current onto the capacitor. To do this, select `soma.i_pas(0.5)` using the same procedure as in the last section.

Now when we run the simulation, we will see two traces. However, it would be nice if they were different colours. To change the colour and thickness of the lines we can right-click on the graph and select **Colour/brush**. This brings up a palette of colours and lines.

We select colours and/or line types by clicking the button next to the colour or line type and then clicking on the relevant legend. After we have run the simulation again, we should see something like figure 3.

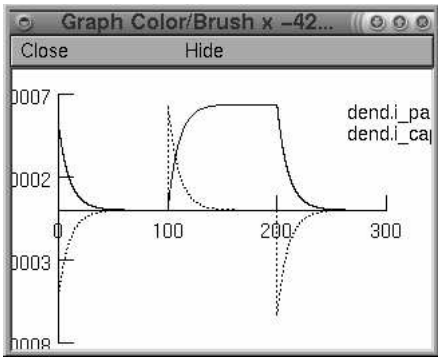


Figure 3: The current axis. The capacitive current and the leakage current are shown.

8 Significance of the results

As in figure 2, you should see that the voltage first settles to $-70\,\mathrm{mV}$ (the leakage reversal potential) from its starting value of $-65\,\mathrm{mV}$ (specified in the **RunControl** window). Then, at 100 ms (the time of the current injection onset), it starts to rise, though not instantaneously. This is because when current is injected it will first mainly charge the membrane capacitance (as shown in figure 3), but as this charges up, more of it will flow through the membrane resistance. It levels out at about $-63.65\,\mathrm{mV}$. In this steady state, all the current flows through the membrane resistance rather than onto the membrane capacitance. At 200 ms (the time of the current offset) the membrane potential starts to decay back to the resting potential. This is due to the capacitance discharging slowly through the resistance. This is a bit like a battery going flat, though on a much faster timescale!

There are some points we should note.

8.1 The input resistance

The *input resistance* R_{in} is a measure of how much current needs to be injected to increase the membrane potential by a certain amount. Electrophysiologists often measure it. To do this the membrane has to have reached the steady state (i.e. no capacitive currents flowing).

In general it is defined by

$$R_{\mathrm{in}} = \frac{\Delta V}{\Delta I_{\mathrm{e}}}$$

where ΔV is a small change in membrane potential due to a small change in injection current ΔI_{e}. In the case of voltage-dependent conductances, the input resistance may vary depending on the holding membrane potential it is measured at. In our case, the conductances are fixed, so the input resistance will not depend on the holding membrane potential and we do not need to make our change in current and membrane potential small. Therefore, the measured input resistance in our case is

$$\begin{aligned} R_{\mathrm{in}} &= \frac{V_0 - E_{\mathrm{L}}}{I_{\mathrm{e}} - 0} \\ &= \frac{(-63.65 - (-70)) \times 10^{-} - 3}{1 \times 10^{-9} - 0} \end{aligned}$$

$$= \quad 6.35 \times 10^6\,\Omega = 6.35M\,\Omega$$

where $V_0 = -63.65\,\text{mV}$ is the limiting maximum potential.

The input resistance is related to the specific membrane resistance (see section 3) and the area A of the membrane according the following formula:

$$R_{\text{in}} = \frac{r_m}{A}$$

In other words, the larger the area of the membrane, the smaller the resistance. This makes sense, because when there is more membrane for current to flow through, it is easier for current to flow.

Since the section we are simulating is a cylinder, the area of membrane is given by

$$A = \text{circumference} \cdot \text{length} = \pi d \cdot l$$

where d is the diameter of the section and l is the length of the section. In our case $l = 100\,\mu\text{m} = 100 \times 10^{-6}\,\text{m} = 100 \times 10^{-4}\,\text{cm}$ and $d = 500\,\mu\text{m} = 500 \times 10^{-4}\,\text{cm}$, so

$$A = \pi \cdot 100 \times 10^{-4} \cdot 500 \times 10^{-4} = 1.57 \times 10^{-3}\,\text{cm}^2$$

As the specific membrane resistance is $r_m = 10\,\text{k}\Omega\,\text{cm}^2$, the calculated input resistance is

$$R_{\text{in}} = \frac{10 \times 10^3}{1.57 \times 10^{-3}} = 6.37 \times 10^6\,\Omega = 6.37M\,\Omega$$

This does indeed match the value we measured.

8.2 The membrane time constant

How fast does the membrane potential reach its limiting value?

Try changing the scale to find out how long it takes from the start of the current pulse for the membrane potential to get two-thirds of the way from the resting potential to the potential it saturates at during the current pulse. (That is, two-thirds of the way between $-70\,\text{mV}$ and $-63.63\,\text{mV}$, which is -65.75 mV.) You should find it takes about 10 ms. You should also find that the time taken for the membrane potential to decay two-thirds of the way towards the resting potential (to $-67.88\,\text{mV}$) at the end of the current pulse is about 10 ms.

This time is called the *membrane time constant* and is represented by the symbol τ. The theoretical value is given by

$$\tau = r_m c_m$$

In our case this means that $\tau = 10 \times 10^3 \cdot 1 \times 10^-6 = 10 \times 10^{-3}\,\text{s} = 10\,\text{ms}$, as should be observed.

The membrane time constant is a measure of how long it takes the membrane to charge or discharge. It tells us how long the neuron "remembers" inputs for. With a large membrane time constant, the compartment has a long memory, and individual inputs will tend to be mixed with others. With a short membrane time constant, the compartment forgets inputs quickly.

8.3 The equations

Assuming that the membrane potential at the start of the current pulse is the leakage reversal potential E_{L}, the membrane potential obeys the following equation during a current pulse starting at $t = 0$ with amplitude I_{e}:

$$V = E_{\text{L}} + R_{\text{in}} I_{\text{e}} \left(1 - e^{-\frac{t}{\tau}}\right)$$

Assuming that the membrane potential starts at V_0, the time course of the membrane potential is as follows when no current is injected:

$$V = E_{\text{L}} + V_0 e^{-\frac{t}{\tau}}$$

Both these equations are the solutions of

$$C\frac{dV}{dt} = -\frac{V - E_{\text{L}}}{R_{\text{in}}} + I_{\text{e}}$$

which is derived from Kirchoff's first law (current conservation).

9 Running a HOC program

So far, we have been typing in commands at the `HOC` command line. It is usually more convenient to save our commands in a program file and then get NEURON to read this file.

This can be done when NEURON starts by giving the name of your program after `nrngui`. For example, if your program is called `practical1.hoc`, at the UNIX command prompt you would type:

```
/usr/i686/bin/nrngui practical1.hoc
```

This will run your program and leaves you at the `oc>` prompt. You can then enter commands just as before, though these commands won't be saved.

It's recommended that you now exit NEURON (by typing `quit()`), create a file called `practical1.hoc` and type the commands into it as you go along. To test what you've done you can always restart NEURON as above.

10 Creating a cable

So far we have looked at the *temporal* passive properties of a single compartment. Now we wish to look at the *spatial* passive properties of a long stretch of axon modelled by a number of compartments connected together as shown in figure 4. This chain of compartments is sometimes called a *cable*, because of its similarity to an electrical cable. To do this, we will first create a new section called `axon`, set is as the default section and insert passive properties.

```
create axon
access axon
axon insert pas
```

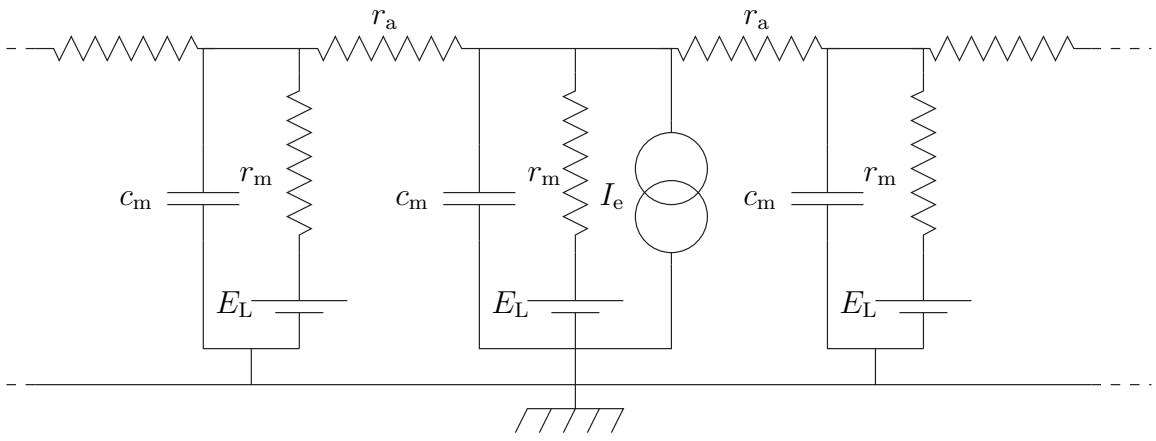


Figure 4: A stretch of axon modelled by single compartments linked together.

To change the specific membrane conductance, diameter, length and number of segments, we will use a new syntax. It means "change the properties of `axon` in the curly brackets without typing `axon` on every line":

```
axon {
  diam = 1
  L = 10000
  nseg = 50
  g_pas = 0.0001
}
```

On the command line it is easier to type

```
axon {  diam = 1   L = 10000   nseg = 50   g_pas = 0.0001   }
```

This code changes the specific membrane conductance to 0.0001 S/cm^2, the diameter to 1 μm and the length to 10000 μm (10 mm). These values are typical for an axon in the mammalian CNS that projects to a different brain area. It also divides the axon into 50 segments. Each segment is the compartment of a compartmental model.

To inject current halfway along the axon, use the following code:

```
objref stim
axon stim = new IClamp(0.5)
stim.amp = 0.1
stim.del = 10
stim.dur = 200
```

11 Viewing the results

In section 6 we looked a plot of the membrane potential as a function of *time*. In this part of the practical, we would like to look at how the membrane potential varies as function of *space*. To see what happens at all points in a section (or a group of sections), you can use a space plot. This dynamic graph plots a variable (for example, membrane potential) against space for each time step in the simulation.

To create a space plot, select **Shape plot** from the **Graph** menu in the **Neuron Main Menu**. This will open a shape plot window in which a schematic representation of the axon will appear. You can rotate the axes by selecting **3D Rotate** from the **Graph Properties** menu. The middle mouse button is used to move the whole representation. (This is more useful for neurons with complex morphology.)

From this window, you can select from the menu (with the right mouse button) a **Space Plot**. To create the space plot graph, you need to select a section of the neuron to include in the space plot by clicking the left mouse button at the beginning of the section (in the schematic picture), dragging the mouse across the section you want to plot (while holding the mouse button down), and releasing the mouse button when you have covered the sections you want to plot. A line will appear from where you first clicked the mouse button to the current location of the pointer. When you release the mouse button, the section you selected will be highlighted in colour, and a new window with the space plot will be opened. You should see that the x-axis of this graph stretches from 0 to 10000 μm, the length of the section we created and that the membrane potential is uniformly -65 mV along the section.

In the **RunControl** window, set `tstop` to 50 ms and then press the **Init & Run** button in the window to see the space plot in action.

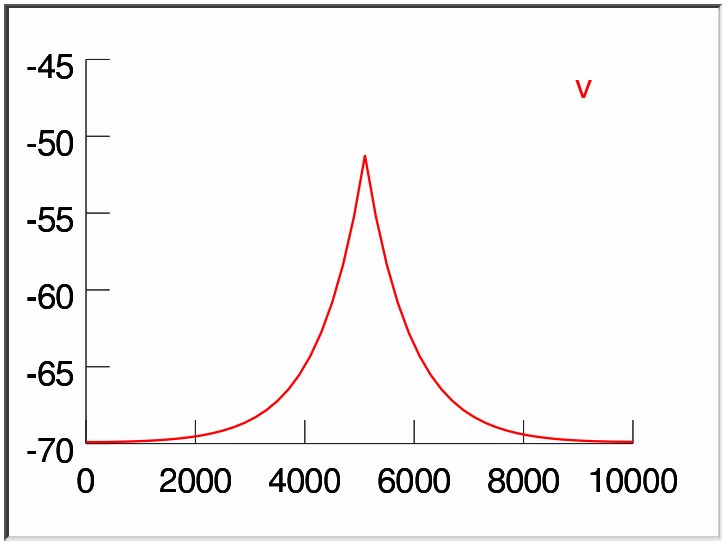


Figure 5: The membrane potential as a function of distance.

You should see that the membrane potential first of all falls gradually towards -70 mV and then, 10 ms into the simulation, develops a peak at 5000 μm that grows towards a maximum height of about -51 mV.

The membrane potential falls away exponentially to either side as shown in figure 5. The distance required for the membrane potential to decay two-thirds of the way towards the resting membrane potential (about $-63.7\,\mathrm{mV}$) should be about $840\,\mu\mathrm{m}$.

12 Significance of the results

The *electrotonic length* (also called *length constant*) λ is defined by the formula

$$\lambda = \sqrt{\frac{r_m d}{4 r_i}}$$

When an infinitely long stretch of neurite is in the steady state with a single point of current injection at x_0, the membrane potential as a function of distance is

$$V(x) = E_\mathrm{L} + V_0 e^{-|x-x_0|/\lambda}$$

where V_0 is the difference between the membrane potential at its maximum and the resting membrane potential. ($V_0 = r_\mathrm{m} I_\mathrm{e}/(2\lambda)$)

In our case,

$$\lambda = \sqrt{\frac{10 \times 10^3 \cdot 1 \times 10^{-4}}{4 \cdot 35.4}} = 0.0840\,\mathrm{cm} = 840\,\mu\mathrm{m}$$

The electrotonic length is a measure of how quickly the membrane potential falls off around the site of current injection. We can see in this case that the membrane potential is virtually zero at the ends of the axon.

Next, shorten the axon to 1000 μm and repeat the experiment. Explain your results.

13 Dendritic filtering

So far we have looked at the temporal properties of a single compartment and the spatial properties of a cable in the steady state. Now we are going to look at the *spatiotemporal* properties of a cable. This is particularly relevant to dendrites, which don't tend to have as many active conductances as axons.

Most of the parameters of our dendritic cable will be the same as the axon we have just investigated. Therefore, we can change the parameters of the axon and imagine that it is a dendrite:

```
axon {
  L = 1000
  nseg = 40
  g_pas = 0.0001
}
```

What we would like to so is measure the membrane potential from the end of the dendrite and inject current at different points along the dendrite.

To measure the membrane potential from the end, bring up a **Voltage axis** and delete any existing plots by right-clicking, selecting **Delete** and clicking on the legend of the trace (i.e. the text `v(0.5)`. Then record from `axon.v(0)` by selecting a new plot from the **Plot what?** dialog box as we did in section 7.2 and editing the text in the text box before clicking **Accept**.

We could create the `IClamps` as we did earlier. However, NEURON provides a convenient way of moving an electrode around. First of all we will delete our old electrode by redeclaring it:

```
objref stim
```

Now, select **Tools→Point Processes→Managers→Electrode** from the main menu. Click on the **IClamp** button and change the amplitude to 1 nA, 10 ms delay and 1 ms duration. Now change the initial membrane potential to $-70\,\mathrm{mV}$ in the **Run Control** window and simulate for about 50 ms.

Change the location of the electrode, click on **Location** in the electrode window and then click on the relevant part of the schematic diagram of the dendrite. You should see the text just above the window change as you click.

The peak of the voltage curve is smeared more as the distance of the injection site increases. This effect is called *dendritic filtering* as the dendrite changes inputs depending where they are on the dendritic tree. The equations for this behaviour are more complex. They are solutions of the full cable equation and require knowledge of boundary conditions to solve.

13.1 Voltage clamp

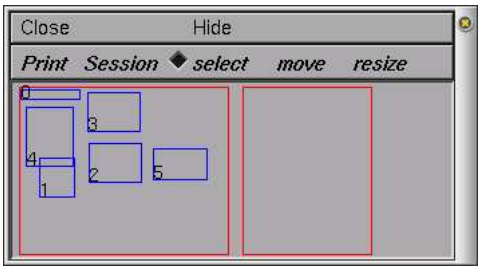
A common experimental configuration is the voltage clamp. In a voltage clamp experiment the voltage stays at a fixed value. This is accomplished by measuring continuously the voltage and injecting compensating current if the voltage starts to deviate. Add an extra electrode to do the voltage clamp (VClamp). Locate it at position 0.0. Clamp continuously at -70mV and plot the current of the voltage clamp (VClamp.i button). This is the current required to maintain the neuron at -70mV.

Move the current injection electrode around and inspect "VClamp.i".

What is the difference in filtering (time to peak and amplitude) compared to the unclamped cable?

14 The Print & File Window Manager

To open the **Print & File Window Manager**, select **Print & File Window Manager** from the **Window** menu on the **Main menu**. The **Print & File Window Manager** window contains several different buttons, menus and two rectangles.



The left most of the two red rectangles in the window represents the entire NEURON display — we will call this the **Manager** rectangle. Each smaller blue rectangle (with a number in it) represents a NEURON window. The second red rectangle represents a sheet of paper — we will call this the **Selection** rectangle. It is used to print selected windows to a file or printer and to save selected windows in a session file. To select one of the windows to print or save, click with right mouse button in one of the smaller blue rectangles in the **Manager** rectangle, and you will see it appear in the **Selection** rectangle. To remove the window from the **Selection** rectangle, simply click with the right mouse button in the small numbered rectangle that you wish to remove. You may also want to move (left mouse button) or resize (middle mouse button) the windows in the **Selection** rectangle.

The **Print** menu gives various options for printing the currently selected windows (i.e., the ones in the **Selection** rectangle) including a POSTSCRIPT printer or a file. If you would like to change the printer that you will print to, you can select the **Select Printer** option in the **Print** menu. This will pop up a window in which you can enter the command to print your file to the new printer. You can also change the layout of the paper to Landscape from Portrait mode (or vice versa) by selecting the **Land/Port** option in the **Print** menu. If you would like to print the PostScript output to a file, then you can select the **PostScript** option under the **Print** menu[4].

Unfortunately there is no direct way of producing encapsulated POSTSCRIPT files that are suitable for inclusion in LaTeX and other documents. To convert the POSTSCRIPT files produced by NEURON to encapsulated POSTSCRIPT a graphics package or command-line utility is required. For example the CONVERT program can be used at the UNIX command line:

```
convert neuron-picture.ps neuron-picture.eps
```

[4]If you don't like the border around the figure, you can get rid of it (under UNIX) by creating a file called `~/.nrn.defaults` that contains the single line `*scene_print_border: 0`

15 Saving and retrieving sessions

After creating several graphs, you may want to save the windows you have created (i.e., graphs and panels) to a file so that you can recall them at a later time. NEURON allows you to save either all or selected windows to a *session* by selecting the **Save selected** or **Save all** option of the **Session** menu in the **Print & File Window Manager**. **Save all** will save the position and contents of all NEURON's windows. **Save selected** will save only those windows that are currently selected in the **Selection** rectangle in **Print & File Window Manager**. Either of these options will pop up a window in which you can enter the filename of your save session.

Try to save all the windows into a session file called `practical1.ses`. (You need to type in the `.ses` suffix explicitly — unlike many modern programs NEURON doesn't add suffixes automatically.)

If we save our session to a file, we can either load the session each time we load our program by selecting the **Retrieve** option of the **Session** menu in the **Print & File Window Manager**, or we can have our program automatically load our session for us. To do this, we need to add the following to our program:

```
xopen("practical1.ses")
```

where `practical1.ses` is the name of the session we saved (**Note**: the name must be in quotes). The next time we start our program, the session with our graphs and menus will automatically be loaded into NEURON.