# Chapter 17
# Building a Cell With Neurokit

DAVID BEEMAN

## 17.1 Introduction and Review

In the first four GENESIS programming tutorials, we covered the features of the GENESIS/XODUS script language needed to construct a simple model neuron. This neuron contains a dendrite compartment with a synaptically activated excitatory channel and a soma with Hodgkin-Huxley sodium and potassium channels. A source of randomly distributed spikes is used to excite the synapse. Action potentials produced in the soma trigger a spike generator that may be used to provide input to a synapse on another cell. In our model, we used a feedback connection to the cell's own synapse that can be toggled on and off. Enough details of XODUS were introduced to create graphs for the membrane potential and channel conductance, along with buttons, toggles and dialog boxes for controlling the simulation.

The previous programming tutorial used the *readcell* function to build the same neuron from a data file. This is the preferred method for the construction of complex cells and the exchange of cell models with other GENESIS users. The tutorial also provided an introduction to the library of prototype cell components that are used with the *Neurokit* cell builder. In this tutorial, we will use the same neuron as an example for the use of *Neurokit* to construct a cell with a minimum of GENESIS programming. Although it is not necessary to have worked through the programming details of the prevous five tutorials before beginning this one, you should read through them in order to understand the model that is being implemented here.

The *Neurokit* cell builder is a GENESIS simulation consisting of a main script *Neurokit.g* and several other included scripts. These make use of the cell reader to create a cell model, just as we did in the previous chapter. However, *Neurokit* also provides a graphical interface for controlling the simulation, modifying the cell model, stimulating the cell with various types of input, and displaying the value of fields in the various compartments.

In order to recreate our simulation with *Neurokit*, we need files for the cell reader similar to the ones that we used in the previous chapter. We will again use a cell descriptor file *cell.p*. We also need the scripts *compartments.g*, *hhchan.g*, *synchans.g* and *protospike.g* in the *neurokit/prototypes* directory. Again, these will be used to create the prototype elements from which the cell will be constructed. As before, we need a script that includes the above scripts and uses them to create the prototype elements under a **neutral** */library* element. In the previous chapter, we chose to call this file *protodefs.g*. *Neurokit* requires that this file be called *userprefs.g*. It performs the functions of our *protodefs.g* file, plus a few others described in the following section.

Before beginning this tutorial, you should also familiarize yourself with *Neurokit* by running one of the *Neurokit*-based simulations described in Chapter 7. Alternatively, you might run the simulation in the *Neurokit* directory (*Scripts/neurokit*), using the *camit.p* cell parameter file which is found there. Try out some of the features that are described in the *README* file, which is available as the on-line help. If you wish to explore the channel editing capabilities of *Neurokit*, you may run the tutorial in *Scripts/channels*. Channel editing with *Neurokit* is also treated in a following chapter, in Sec. 19.2.2. Note that the *Neurokit* `edit channel` menu choice only refers to voltage-dependent ionic channels, and not to synaptically activated channels.

## 17.2 Customizing the *userprefs* File

When *Neurokit* is run, it begins by assigning default values to a number of global variables called *user-variables*. These variables are defined in the *Neurokit* directory file *defaults.g*, and are responsible for the initial values of most of the *Neurokit* dialog boxes, scales for graphs, and a good deal of what you see on the screen. Although you may wish to examine *defaults.g*, it should not be modified. Next, *Neurokit* looks for a file called *userprefs.g*. This file is responsible for creating the prototype elements in the library and for making any desired changes in the user-variables that were set in *defaults.g*. Unless you are running the example *camit* (CA mitral cell) demonstration simulation from the *Neurokit* directory, you will want to run *Neurokit* from another directory that contains your own customized *userprefs* file and your own cell parameter file. Typically, you will have a *.simrc* file that has set a path to the *Scripts/neurokit* and *Scripts/neurokit/prototypes* directories, so that this may be done. If the *userprefs* file is not found in the directory from which *Neurokit* is being run, the default *userprefs.g* in the *Neurokit* directory is used. This file, shown in Fig. 17.1,

may be used as a model for the construction of your own *userprefs* file. To illustrate this customization, we will go through the file, making the changes needed to construct our model neuron. If you are feeling impatient, you may consult the resulting *userprefs.g* listing in Appendix B. However, you will gain the most from this tutorial by developing the file piece by piece as you work through the tutorial.

### 17.2.1 Step 1

The first step is to include the scripts that define the functions to be used to create the prototype elements. In Chapter 19, we will create our own prototype channels. For now, we will build our neuron with channels that we can find in the prototypes directory. The *neurokit/prototypes/LIST* file provides a summary of these.

We need compartments (defined in *compartments.g*), sodium and potassium Hodgkin-Huxley channels (defined in *hhchan.g*), an excitatory synaptic channel (a **synchan** object), and a spike generator like the one we used in the previous tutorial. We can find the latter defined in *protospike.g*. In Fig. 17.1, the file *mitsyn.g* defines functions for creating both an excitatory channel (*make_glu_mit_upi*) and an inhibitory channel (*make_GABA_mit_upi*). Note that, unlike the names in our "generic" *synchans.g* file, the prototype channel names follow the convention recommended in the *neurokit/prototypes/README* file. For example, *glu_mit_upi* is a glutamate-activated mitral cell channel, authored by Upi Bhalla. In general, you should look for a file that defines a prototype closest to the one you need. In some cases, it may be necessary to modify an existing file to produce the desired prototype, or to change the default field values, once the prototype is created. For our model, we can use the file *synchans.g*, which has exactly what we need. Thus, the first section of our *userprefs* file will be fairly similar to that of the default version and to the statements given in Sec. 16.2.2. It will contain the statements:

```
include compartments  /* file for standard compartments */
include hhchan   /* file for Hodgkin-Huxley Squid Na and K channels */
include synchans      /* file for synaptic channels */
include protospike    /* file which makes a spike generator */
```

### 17.2.2 Step 2

*Neurokit* begins by creating a neutral element */library* to contain the prototypes, so our second step should be to change to this element and execute the functions that will create the desired prototype elements. However, we should first consider what we need to do in order to properly set the values of the internal fields of these prototypes. In the case of compartments, all the relevant fields can be set from information contained in the cell parameter file. For channels, the maximum channel conductance is the only field that is determined from the cell parameter file. *Neurokit* allows some other fields to be set from

```
// genesis - default neurokit/userprefs.g file
echo Using default user preferences!

// Step 1 - Including script files for prototype functions
/* file for standard compartments */
include compartments
/* file for Hodgkin-Huxley Squid Na and K channels */
include hhchan
/* file for Upi's mitral cell channels */
include mitchan
/* file for Upi's mitral cell synaptic channels */
include mitsyn

// Step 2 - Invoking functions to make prototypes in the /library element
pushe /library // make all subsequent elements in the library

/* Make the standard types of compartments */
make_cylind_compartment         /* makes "compartment" */
make_sphere_compartment         /* makes "compartment_sphere" */
make_cylind_symcompartment      /* makes "symcompartment" */
make_sphere_symcompartment      /* makes "symcompartment_sphere" */

/* These are some standard channels used in .p files */
make_Na_squid_hh                /* makes "Na_squid_hh" */
make_K_squid_hh                 /* makes "K_squid_hh" */
make_Na_mit_hh                  /* makes "Na_mit_hh" */
make_K_mit_hh                   /* makes "K_mit_hh" */

/* There are some synaptic channels for the mitral cell */
make_glu_mit_upi                /* makes "glu_mit_upi" */
make_GABA_mit_upi               /* makes "GABA_mit_upi" */

pope    /* returning to the root element */

// Step 3 - Setting preferences for user-variables.
user_syntype1 = "glu_mit_upi"
user_syntype2 = "GABA_mit_upi"
```

**Figure 17.1** The *userprefs.g* file for the default *Neurokit* simulation. For brevity, some of the comments have been removed.

dialog boxes, but many would have to be changed, using the GENESIS *setfield* command, if the desired values are not set when the prototypes are created. Thus, we should examine the channel creation functions in order to see what changes we might need to make within the *userprefs* file.

Looking at the listing for *hhchan.g* in Appendix B, we see that it makes use of the global variables *EREST_ACT*, *ENA*, and *EK* to define the resting potential and the Na and K equilibrium potentials. The functions *make_Na_squid_hh* and *make_K_squid_hh* use these values when creating the channels. If they are not the values we want, we should modify them within *userprefs.g*. This should be done at a point *after* the script is included, but *before* the functions are invoked. As the default values had been modified from the original Hodgkin-Huxley squid values for use in a mitral cell simulation, we will need to change them to the proper values, as we did in Chapters 14 through 16. (You may note that *hhchan.g* assigns a value to a fourth global variable, *SOMA_A*. Why may we leave this as is?)

As in the *protodefs.g* file from the previous tutorial, we can use the *make_Ex_channel* function to create the glutamate-activated channel *Ex_channel* with all the proper default field values. Although it isn't used by the cell model that we are going to create, we might as well put a GABA channel *Inh_channel* in the library, in case we want to edit the cell to include it later. Chapter 15 describes how a spike generator linked to the soma may be used to translate somatic action potentials to presynaptic neurotransmitter release. As before, we will use the *make_spike* function in *protospike.g* to create an element named *spike* with a unit amplitude and an absolute refractory period of 10 *msec*. Thus, the second section of *userprefs.g* contains:

```
pushe /library
make_cylind_compartment         /* makes "compartment" */

/* Assign some constants to override those used in hhchan.g */
EREST_ACT = -0.07       // resting membrane potential (volts)
ENA   = 0.045           // sodium equilibrium potential
EK    = -0.082          // potassium equilibrium potential

make_Na_squid_hh   /* makes "Na_squid_hh" */
make_K_squid_hh    /* makes "K_squid_hh" */

make_Ex_channel    /* synchan with Ek = 0.045, tau1 = tau2 = 3 msec */
make_Inh_channe    /* synchan: Ek = -0.082, tau1 = tau2 = 20 msec */

make_spike         /* Make a spike generator element */

pope               /* return to the root element */
```

Of course, one does not have to rely on the functions that are defined in the *neurokit/prototypes* files. *Neurokit* sets the SIMPATH to include "*./prototypes*", so you may

have your own prototypes directory with scripts containing your own function definitions.

### 17.2.3 Step 3

The final step is to make any needed changes in the user-variables from the values given in *defaults.g*. Although most of these may be set from dialog boxes within *Neurokit*, it is far more convenient to have *Neurokit* start up configured for the simulation that you want to run. In this part of the tutorial, we will set some of the more common user-variables.

The *Neurokit* `file` menu contains default values for the name of the cell to be created and the source file name (the cell descriptor file) that will be loaded when you click on `Load from file`. In Chapter 16, we created a cell called */cell* from the file *cell.p*, so we will start with:

```
user_cell = "/cell"
user_pfile = "cell.p"
```

The `run cell` selection brings up a `SIMULATION CONTROL PANEL` form with dialog boxes for many parameters that we will want to customize for our simulation. Figure 17.2 shows the display that is produced with the *camit* simulation. It would be best to run *Neurokit* as you read through this tutorial. If you do this with the *userprefs* file you have created so far, you will be able to experiment by changing the dialog boxes by hand as you edit your *userprefs* file to change the user-variables. The parameters that we are most likely to want to change are the run time, the simulation time step (`clock`), and the number of steps between update of the graphs (`refresh_factor`). In the previous tutorials, we ran the simulation for 100 *msec*, using a time step of 0.05 *msec*. We can save some execution time by plotting the graphs every five simulation steps, instead of plotting at every simulation step. These values may be set with the statements:

```
user_runtime = 0.1
user_dt = 50e-6  // 0.05 msec
user_refresh = 5
```

We will also need to modify some variables that appear in the dialog boxes under the `ELECTROPHYSIOLOGY` heading of the `SIMULATION CONTROL PANEL`. Some of these dialog boxes appear and disappear according to the type of stimulation that is applied to the cell. Table 17.1 lists the types of stimulation, the dialog box labels, and the associated user-variables.

We will experiment with the values of most of these later on in the tutorial. You may assign reasonable initial values to these in the *userprefs* file now, or may do it later after having gained some experience with the simulation. In the earlier tutorials, we found that 0.3 *nA* was a good value to use for the injection current, so you may set *user_inject* to this value now. The two boxes labeled `click_site1` and `click_site2` obtain their values

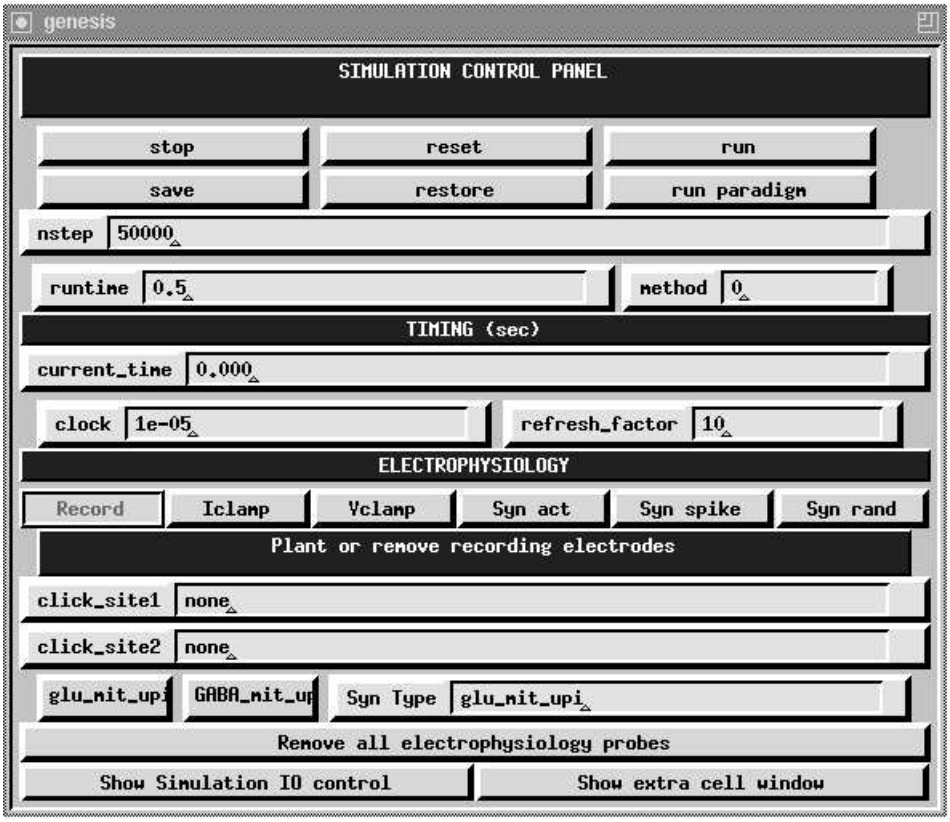


**Figure 17.2** The *Neurokit* SIMULATION CONTROL PANEL form.

when you click on a compartment in one of the two Cell Windows, for either recording or stimulation. Finally, there is a dialog box labeled `Syn Type`. This contains the name of the channel that will be used for one of the three forms of synaptic stimulation. The contents of this box may be set by hand, or by clicking on one of the two buttons to the left. We can set the labels of these buttons to the two available synaptically activated channels by including the statements

```
user_syntype1 = "Ex_channel"
user_syntype2 = "Inh_channel"
```

in the *userprefs* file, as was done in the default *userprefs* file for the *camit* simulation.

There are also one or two Cell Windows that each contain a diagram of the cell and a graph with axes scaled appropriately for the quantity to be plotted. The simulation that we are trying to recreate had a graph to plot the soma membrane potential, and a second graph

| *Stimulation* | *Dialog Box Label* | *User-variable* |
|---|---|---|
| Iclamp | Inject (*nA*) | user_inject |
| Vclamp | Clamp voltage (*mV*) | user_clamp |
| Syn act | Amount of synaptic activation | user_activ |
| Syn spike | Weight of spike | user_spike |
| Syn rand | Spike rate (*Hz*) | user_rate |
| | Spike weight | user_weight |

**Table 17.1** The dialog boxes and user-variables associated with the various types of stimulation that may be applied to the cell.

to plot the conductance of the excitatory channel. We can specify the axis scaling for these two graphs to be the same as we used before, if we add the statements:

```
user_ymin1 = -0.1
user_ymax1 = 0.05
user_xmax1 = 0.1
user_xmax2 = 0.1
user_ymin2 = 0.0
user_ymax2 = 5e-9
```

We also need to use the *user_numxouts* variable to specify that two graphs will be shown, and we need to specify what will be plotted in each graph. The *defaults.g* file specifies values of "`Vm`" and "`.`" for the user-variables *user_field1* and *user_path1*. This means that the *Vm* field will be plotted for the compartment that is selected for recording in the first Cell Window. This is just what we want for the first graph. However, these same defaults are also specified for the second graph. For our second Cell Window, we would like to be able to plant a recording electrode in the dendrite compartment and plot the conductance *Gk* of the excitatory channel *Ex_channel*. We can make these changes with the statements:

```
user_numxouts = 2
user_field2     = "Gk"
user_path2 = "Ex_channel"
```

There are also `scale` buttons that will bring up menus to change the representation of the cell which is shown in each window. The *Neurokit README* and *defaults.g* files provide information on the variables that appear in these menus. Most of these correspond to fields of the **xcell** widget, as described in the GENESIS Reference Manual. For now, we will use the default values.

## 17.3 The Cell Descriptor File

The format of the cell descriptor file was discussed in Chapter 16, so you may wish to review that section of the tutorial. For our simulation under *Neurokit*, we can use the same *cell.p* file.

You should recall that the cell reader uses the "`*set_compt_param`" option to specify its own internal variables *CM*, *RM* and *RA* in the cell descriptor file in order to calculate the fields *Cm*, *Rm* and *Ra* from the compartment dimensions, ignoring any initial values set in the prototypes. The *Em* and *initVm* fields are similarly set from the *EREST_ACT* variable. The *ELEAK* variable may be used if it is necessary to set a different leakage potential for *Em*. Also, the maximum conductance for a channel is set from the *dens* parameter and the dimensions of the parent compartment, overriding any value set in the prototype. However, all other channel parameters remain the same as those in the prototype library. Fortunately, *Neurokit* provides dialog boxes for changing many of them.

## 17.4 Some Experiments Using Neurokit

Now that you have a customized *userprefs.g* file and a *cell.p* file, you are ready to try some experiments with *Neurokit*. You should run GENESIS from the directory that contains these files, and then type "`Neurokit`" to the GENESIS prompt. If all is well, the *Neurokit* title bar should eventually appear at the top of the screen. If GENESIS cannot find *Neurokit*, make sure that your *.simrc* file has properly set the SIMPATH. If you get fatal errors during the loading of *userprefs.g*, look for syntax or typographical errors in this file. Once *Neuorokit* is running, click on the `file` option on the title bar. (As usual, when we say "click on ..." we mean "click the left mouse button on ...," unless another button is specified.) Next, click the `Load from file` button on the `file` menu, and then click `run cell` on the title bar. You should see the `SIMULATION CONTROL PANEL` with the timing parameters you provided and two Cell Windows with properly scaled graphs.

Initially the `ELECTROPHYSIOLOGY` buttons will show that you are ready to plant recording electrodes. Click on the soma in the first Cell Window and on the dendrite in the second Cell Window. The two `click_site` dialog boxes should now show "`/cell/soma`" and "`/cell/dend`".

As a start, let's do a simple current injection experiment to make sure that the soma compartment is working properly. After clicking on `Iclamp`, you should see a dialog box showing that the injection current is 0.3 *nA*. Click on the soma in the first Cell Window to plant an injection electrode. Now click on `run` in the `SIMULATION CONTROL PANEL`. After you have satisfied yourself that this provides the expected action potentials, remove the injection electrode by clicking on the soma with the middle mouse button and click on `reset` to clear the graph.

We are now ready to try applying synaptic activation. We will start by using mouse clicks to deliver spikes to the *Ex_channel* channel. This channel name should be displayed in the `Syn Type` dialog box. If not, the name can be entered in the box "by hand", or by clicking on the button labeled "`Ex_channel`" at the left.

First, click on the `Syn spike` button under the `ELECTROPHYSIOLOGY` heading. Note that the `MOUSE` display at the top of the two Cell Windows now shows that the left button is labeled "`SynSpike`" and that the middle button is labeled "`UnSpike`". Unless you have changed the *user_spike* variable in the *userprefs* file, the `Weight of spike` dialog will show the default value of 1.0. The maximum conductance (*gmax*) of a synaptically activated channel is reached when a spike with a unit amplitude is delivered to the channel, so this is a reasonable amplitude to use.

Now, click on `run` in the `SIMULATION CONTROL PANEL`. After the membrane potential has leveled off to about $-70$ *mV*, click on the dendrite compartment in the second (right) Cell Window. You should see a nearly linear rise of the channel conductance, followed by an exponential decay with a time constant of about 3 *msec*. Try clicking several times rapidly in succession, in order to let the conductance and postsynaptic potential build up to a level sufficient to create an action potential in the soma. In order to check the consistency of these results, we can inspect the channel parameters by selecting `edit cell` from the top menu bar.

Click on the dendrite compartment in the Cell Window (lower left) and then on the *Ex_channel* channel in the Compartment Window (upper right). The Parameter Window should display the dendrite dimensions and the maximum channel conductance in $S/m^2$. Of course, *gmax* can also be found by typing "`showfield /cell/dend/Ex_channel gmax`" to the GENESIS prompt. Verify that this agrees with the maximum conductance provided by a single spike.

In order to deliver spikes at random intervals, select `run cell` again and click on `Syn rand` in the `ELECTROPHYSIOLOGY` menu. Dialog boxes will appear for both `Spike rate` and `Spike weight`. Try a spike rate of 200 *Hz*, and leave the weight at 1.0, as before. The left and middle mouse buttons will now be labeled "`RandSyn`" and "`UnRand`". Click the left button on the dendrite compartment in Cell Window 2, and then run the simulation. Repeated runs (after `reset`) will continue to deliver random spikes until you click the middle button in the Cell Window 2 dendrite compartment. (Note that switching to another form of input to the neuron, such as `Iclamp` or `Syn spike`, will not remove the random spike input unless you first use the middle button to remove the input.)

The `Syn act` button delivers a constant synaptic input. Using the default value of 1000 for the `Amount of synaptic activation`, run the simulation and click the left button in the Cell Window 2 dendrite compartment. Note that the channel conductance levels off at about $4.0 \times 10^{-9}$ siemen. Can you explain why? Be sure to use the middle button to remove the activation when you are through.

Now, how about the feedback connection that we used in Chapters 15 and 16? *Neurokit*

was designed for single cell modeling, without connections between the cells, so we will have to enter some additional GENESIS commands in order to make the connection. As we want to make a synaptic connection between the spike generator and the *Ex_channel* channel, we enter a command at the GENESIS prompt to send a SPIKE message to the channel:

```
addmsg /cell/soma/spike /cell/dend/Ex_channel SPIKE
```

We also need to assign a synaptic weight and a delay for propagation along the axon. In the previous tutorials, we used a weight of 10 synaptic connections and a delay of 5 *msec*. We also need to set some parameters for the spike generator. The cell parameter file only lets you set the threshhold for the spike generator. We also need an amplitude and an absolute refractory period. As with our previous implementations of this cell model, we want to give each spike a unit amplitude and to give the spikes an absolute refractory period of 10 *msec*. Thus, we need to give the additional commands:

```
setfield /cell/dend/Ex_channel synapse[0].weight 10.0 \
    synapse[0].delay 0.005
setfield /cell/soma/spike output_amp 1 abs_refract 0.010
```

By using *showfield* to find the number of synaptic connections to *Ex_channel*, you should verify that the index "0" is the proper one to designate this synaptic connection. After entering these commands, perform the `Syn rand` experiment again. You should notice that after the first action potential occurs, the feedback produces a continous stream of action potentials, as in Chapters 15 and 16. As we did in these tutorials, we can delete the connection with the command

```
deletemsg  /cell/soma/spike  0  -out
```

So far, we have done nothing with the inhibitory synaptic channel that is lying dormant in the library. Use `edit cell` to paste in the *Inh_channel* channel. Use the cell parameters form to set the maximum conductance to the same as that of the glutamate channel. Select this as the `Syn Type` and inject some current to the soma in order to get the cell firing. We would like to deliver some activation to this channel to prevent the cell from firing. However, we would also like to see a plot of the conductance from this channel instead of that from the glutamate channel. This can be changed with the `scale` button at the top of Cell Window 2. Change the `fieldpath` dialog entry to "`Inh_channel`". Now use either `Syn spike` or `Syn rand` to inhibit the firing that is produced by the current injection.

## 17.5 Exercises and Projects

1. In the simulations of Chapters 15 and 16, we had a toggle button to allow us to add and delete the feedback connection between the axon and the excitatory synaptic channel. Here, we had to resort to the rather awkward expedient of entering long GENESIS commands. Will *Neurokit* allow us to extend its capabilities and build in the feedback connection along with a toggle to take it in and out? Can we have dialog boxes to enter the value of the weight and delay parameters for the synaptic connection? Fortunately, the answer is "yes." The `run paradigm` button in the `SIMULATION CONTROL PANEL` will invoke a user-supplied function called *do_paradigm*. The definition of this function and the statements needed to create a form with the needed dialog boxes and toggle can reside in a supplementary script that is brought into the simulation with the *include* statement. This may be done in your *userprefs* file or may be done at the GENESIS prompt after your simulation has been started. The demonstration simulation in *Scripts/vclamp* provides an example of such a paradigm file. For another example, take a look at the script *I_plots.g* from the *Burster* tutorial of Chapter 7. Use this information to write a function that will cause the `run paradigm` button to toggle the feedback connection on and off.

2. There are a number of experiments that one can do with just a "naked" soma in order to study the effects of various types of voltage-activated ionic channels. Modify your *cell.p* file to create just this simple soma with the two Hodgkin-Huxley channels. Also modify your userprefs file to put a non-inactivating muscarinic potassium current (the "M-current") in the library. The conductance may be created with the function *make_KM_bsg_yka* in the *neurokit/prototypes* file *yamadachan.g*. This file contains functions to create several conductances that were used in a model of a bullfrog sympathetic ganglion neuron by Yamada, Koch and Adams (1989).

   Start by calculating and making a plot of the current-frequency (f-I) relationship of a standard Hodgkin-Huxley patch of membrane (i.e., containing only the Na and K channels). That is, inject a constant amount of current for, e.g., 200 *msec*, and plot the resulting frequency of spikes versus the amplitude of the injected current. Reduce the potassium conductance and describe how the f-I curve changes. Next, use `edit cell` to paste in the M-current channel and recompute the f-I curve. How does it change? Note that even if very large input currents are applied, the spiking frequency will not become arbitrarily large. What underlying parameter limits the spiking frequency?

3. In Chapter 7 (Sec. 7.3.2), we briefly mentioned a low threshold calcium current that contributes to burst production in thalamic relay cells. The file *THALMODES.g* in the *neurokit/prototypes* directory implements the channel models that are described

by McCormick et al. (1992). Use these channels to create a single compartment bursting thalamic relay cell model.

4. Try to recreate the results obtained by Connor and Stevens (1971c) for their model of *Anisidoris* gastropod neurons. The channel prototypes are contained in *CSchan.g*.

5. If you have worked through the programming of the tutorial in Chapter 15, you may be interested in doing some "snooping" beneath the surface of *Neurokit*. While the simulation of our simple cell is running with `Syn ran` selected for synaptic input to the *Ex_channel* channel, use the *showfield* and *showmsg* commands to discover how the input is provided to this channel. How does this compare with the way that the random input was implemented in Chapter 15? What connections or messages are used to provide the `Syn act` option?