Chapter 3
Neural Modeling with GENESIS

JAMES M. BOWER and DAVID BEEMAN

3.1 What is GENESIS?

Now that we have briefly described the numerical basis for the tutorials included in the first part of this book, we are ready to get started with running the tutorials. The tutorials included in this manual are all constructed using GENESIS, the General NEural SImulation System that has been under development in our laboratory at Caltech since 1985. This chapter is intended to introduce each of the tutorials, as well as provide a demonstration as to how to use the GENESIS graphical interface. First, however, we provide some basic information about the GENESIS system on which the tutorials are based.

3.1.1 Why Use a General Simulator?

In principle, each of these tutorials and their graphical interfaces could have been written as a dedicated piece of software not dependent on a general simulation system (cf., Huguenard and McCormick 1994). In this case, each tutorial would have included only the code necessary to run the particular simulation. If the simulation were written well, this would result in the maximum use of the speed and memory of the computer being used. However, as the speed of computers has increased and computer memory has become cheaper and cheaper, the advantage of dedicated simulation code as compared to general simulators has become less and less clear. In addition, a well-designed neural simulator can provide very good performance, even when compared to dedicated code. We have estimated, for example, that the overhead costs associated with GENESIS average to less than a factor of two in simulation speed.

With a usually small compromise in simulation speed, simulation systems like GENESIS can actually provide many important advantages over dedicated code. Many of these advantages result in a dramatic speedup in the process of building and expanding models that usually more than compensates for the slight reduction in simulation speed. Almost always, the time taken to build a model is considerably greater than the time taken to actually simulate it, once built. Furthermore, adding new components to the simulation does not require rewriting all the existing code, as is often the case with dedicated simulations.

A second advantage of general-purpose simulators is that model components can be shared between different simulations. The ability to use previously developed and tested components leads to tremendous speedups in the time necessary to develop new simulations. The second part of this book demonstrates the ease with which GENESIS allows such modifications.

Beyond issues of flexibility and speed of model construction, there are also several technical advantages to general simulation systems. These include some assurance that the software core of the simulations themselves are accurate, more accessibility to new simulation-related technology such as parallel computers (Nelson, Furmanski and Bower 1989), new integration techniques (Hines 1984), or new forms of graphics (Leigh, De Schutter, Lee, Bhalla, Bower and DeFanti 1993), and some form of standardization for model descriptions.

Perhaps most importantly, general simulation systems such as GENESIS can lead to fundamentally new mechanisms of communication between those interested in how the brain computes. For example, by using GENESIS, it is possible to test the modeling results of others by actually running their simulations. The GENESIS users' group, BABEL, maintains a database of published simulations for this purpose. Several of the tutorials in this manual were modified from research simulations. In addition, simulations can serve as a means of communication between different laboratories interested in the same system. We have recently pointed out (Bower 1992) that simulation systems and their component libraries can become a form of neural database that not only represents structural information about the nervous system, but does so in such a way that structural details are functionally organized. Thus, when a general simulation system is properly designed, the more it is used, the more information about the nervous system is included. Finally, a system such as GENESIS allows books like this one to be written, leading, we hope, to a more efficient and interesting way to learn about the organization of the nervous system.

3.1.2 GENESIS Design Features

GENESIS was developed primarily as a means of constructing biologically realistic neuronal simulations. In addition to our interest in supporting the general advantages of simulators just described, three other basic objectives determined our design philosophy: (1) the simulator must be capable of addressing problems at many levels of detail (i.e., parts of neurons to large neural systems); (2) the system should be open-ended, placing few limits on the kinds of problems that can be addressed; and, (3) the system should be user-extensible to allow the incorporation of new modeling efforts. The following specific features of the system have evolved, based on the following criteria.

Modular Design

As a key principle conferring both flexibility and organization on GENESIS, we adopted a highly object-oriented programming approach. Simulations are constructed of "building blocks" or modules, each of which perform well-defined functions and have standardized means for communicating with each other. The level of detail of a simulation is then determined by the nature of the building blocks one chooses. The user has complete freedom to assemble a model with any set of modules. Object-oriented design also enables the user to easily add new modules to extend GENESIS for a particular application.

Flexible Interface

The GENESIS user interface consists of two parts. The underlying level is the Script Language Interpreter, or SLI. This is an interpretive programming language similar to the UNIX shell, with an extensive set of commands related to building simulations. The interpreter can read SLI programs (scripts) either interactively from the keyboard, or from files. The graphical interface is XODUS, the X-windows Output and Display Utility for Simulations. This provides a higher level and user-friendly means for developing simulations and monitoring their execution. XODUS consists of a set of graphical modules that are exactly the same as the computational modules from the user's point of view, except that they perform graphical functions. As with the computational modules, XODUS modules can be set up in any manner that the user chooses to display or enter data. Furthermore, the graphical modules can call functions from the script language, so the full power of the SLI is available through the graphical interface.

Device Independence

In order to make the simulator portable and available to the maximum number of users, the system has been designed to run under UNIX and X-windows and has been compiled and tested on a number of machine architectures. This facilitates communication by enabling groups utilizing different machines to run each other's simulations. A parallellized implementation has also been developed that dramatically extends the range of problems which GENESIS can address. The use of parallel GENESIS is described in Chapter 21.

3.1.3 GENESIS Development

GENESIS was initially developed in our laboratory by Matthew Wilson as an extension of efforts to model the olfactory cortex (Wilson and Bower 1989, 1992). The GENESIS graphical interface, XODUS was subsequently developed primarily by Upinder Bhalla as a general-purpose user interface for the GENESIS system. Since that time, many additional modelers both at Caltech and at other institutions have contributed to its design.

3.2 Introduction to the Tutorials

One of the interesting features of neural modeling is that it is possible to start at almost any scale or level of detail. In the case of realistic modeling, it is usually the case that whatever level one begins with, the results themselves push the modeler to consider other levels. Thus, detailed single cell modeling quickly requires a better understanding of the network in which the cell is embedded. On the other hand, network level modeling often serves to focus attention on the details of single cell physiology.

As we have stated, GENESIS was specifically designed to allow and promote this multi-scale modeling within a single simulation system. At present, it is the only simulator with this capability. The tutorials described in Chapters 4–10 each represent simulations at different levels. The sequence also reflects the way in which large-scale simulations can be built up from more detailed simulations. All of these simulations can be performed without any knowledge of the GENESIS simulation language. The second half of this book retraces this sequence of topics with a series of tutorials and exercises that are designed to teach you how to program your own simulations with GENESIS, emphasizing the process whereby any GENESIS model can be expanded and changed into a new simulation. The tutorials presented in the first part of this book, and introduced briefly below, will first familiarize you with the various modeling levels supported in GENESIS and used by computational neurobiologists to understand neuronal organization.

Chapter 4 describes the experiments and mathematical models used by Hodgkin and Huxley to understand the time- and voltage-dependence of the ionic conductances responsible for the generation of action potentials. In this case, the tutorial simulation represents a single piece of a neuronal axon. However, these Hodgkin-Huxley forms of voltage activated channels are an important ingredient of many more complex neuronal models. Most model tuning involves manipulation of the Hodgkin-Huxley parameters governing ionic channels. The associated equations also usually represent the most computationally intensive components of single cell simulations. In this tutorial, a GENESIS re-creation of the Hodgkin-Huxley model is used to perform current and voltage clamp experiments and to understand the basis of postinhibitory rebound and neuronal refractory periods.

Chapter 5 explores the “cable properties” of a series of passive compartments without active ion channels. In this case the tutorial has linked together several compartments of the type presented in Chapter 4, but without the Hodgkin-Huxley channels. Although such models do not include active channels, which are one of the principal features of the nervous system, there is still a great deal that may be learned from such models. Studies of the passive properties of neurons lay the foundation for much more complex simulations. They can also be used with experimental measurements to determine the morphology of a cell.

Chapter 6 combines elements of Chapters 4 and 5 into a multi-compartmental neuronal model. This tutorial also introduces models of synaptically activated channels. The theory presented, along with the simulation *Neuron*, explores the effects of temporal summation of multiple synaptic inputs. The neuron contains spatially separated dendrite compartments with both excitatory and inhibitory synaptically activated channels, and a soma with Hodgkin-Huxley sodium and potassium channels.

Chapter 7 uses two different simulations to understand how interactions between several different types of voltage-activated channels can lead to more complex firing patterns in single neurons. In one simulation, a model of the soma of a molluscan pacemaker cell, a particular combination of somatic channels generates periodic bursts of action potentials. In a second model of a hippocampal pyramidal cell, bursting action potentials are shown to arise through an interaction between the soma and more distant dendritic regions.

Chapter 8 introduces for the first time networks of neurons. In this case, we explore a simple network taken to represent a central pattern generator, or CPG. Circuits of this sort have been shown to control many types of rhythmic behavior in a wide range of different animal species. As is often the case with multi-neuron network models, the CPG model considered here is more abstract than an actual biological network that might produce similar firing patterns. In this tutorial, we use GENESIS to explore the circuit interactions between four neurons that are similar to the model neuron used in Chapter 6. The behavior of this network is compared to mathematical models of simple coupled oscillators. The network may also be used to replicate the various footfall patterns of a horse.

Chapter 9 incorporates many of the features explored in the previous chapters into a more complex multi-neuronal network model. In this case, the model is intended to allow you to explore several aspects of the dynamical properties of cerebral cortical networks. Unlike the idealized network used in Chapter 8, this simulation attempts to both realistically represent basic features of real neurons, while at the same time linking them together into a sizable network. The result is a fairly complex simulation that also takes longer to execute than any of the other tutorials. The simulation itself is a “user-friendly” version of the research simulation used by Wilson and Bower (1989, 1992) to model the neuronal dynamics of the mammalian olfactory cortex.

We conclude Part I of this book with Chapter 10, returning to the subcellular level of modeling to consider a different type of network — the interacting biochemical signaling pathways that underlie the processes of synaptic tranmission and channel activation. The theory of biochemical computation and the use of the *Kinetikit* graphical interface are discussed in this final chapter.

3.3 Introduction to the GENESIS Graphical Interface

Before we explore the first tutorial, it is first necessary for you to understand the basics of the GENESIS graphical interface, XODUS. By using this interface, it is possible to easily investigate a wide variety of models and to vary their parameters without doing any programming. XODUS allows many different ways of visualizing the state of the many relevant variables that are present in neurobiological simulations. In order to give you a feel for the use of XODUS, we use the tutorial *Neuron*, which is described in Chapter 6. This tutorial is intended for use in understanding the effects of synaptic inputs to a model neuron. However, its user interface is fairly representative of most of the tutorials used in this book.

3.3.1 Starting the Simulation

In this book, it is assumed that GENESIS and its associated scripts for the tutorials have been installed on a workstation that uses the UNIX operating system and some variety of X Windows. Appendix A describes the procedure for acquiring and installing the software if you have not already done so. We recommend that if you are not fully familiar with UNIX and its file system, that installation of GENESIS be carried out by the system administrator or person responsible for maintaining your machine. Typically, the tutorials and other GENESIS scripts will be installed in subdirectories of the */usr/genesis/Scripts* directory. We will refer to this simply as the *Scripts* directory.

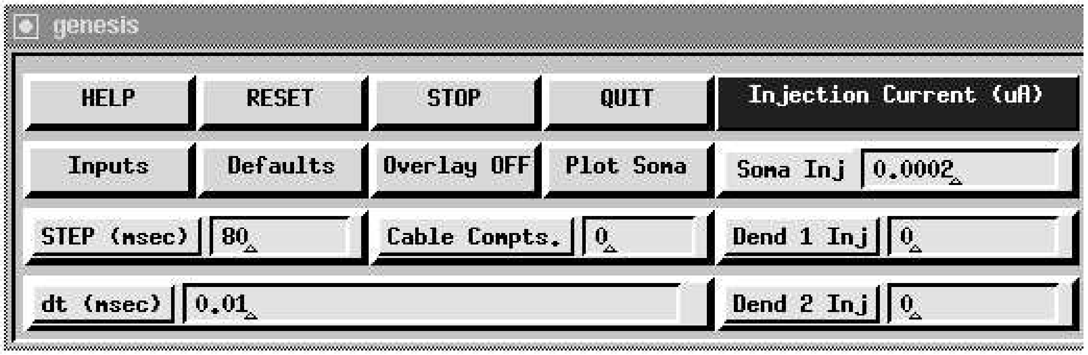
Although you will get a chance to explore this tutorial more thoroughly in Chapter 6, it would be a good idea to run the simulation now, in order to become familiar with the procedures that we will use in the following chapters. In order to use GENESIS, you will obviously have to first login to a UNIX workstation with access to the GENESIS scripts. Follow your local procedures for logging in and creating a “terminal window” on your workstation. So that it will not be completely covered by the simulation display, move the window to the lower left corner of the screen. This is usually accomplished by holding down the left mouse button while the cursor is on the title bar and dragging the mouse to move the window.

The next step in running a GENESIS simulation involves changing your directory to that containing the GENESIS tutorial of interest. In keeping with the modular structure of GENESIS simulations, most tutorials consist of a short main simulation script that invokes several other scripts. This collection of scripts is kept in its own subdirectory of the *Scripts* directory. Although this practice is by no means universal, it is customary to give the main simulation script a name that starts with a capital letter in order to distinguish it from the other associated scripts. The script that runs the *Neuron* simulation used here for illustration is called *Neuron.g*. It is kept in the *Scripts/neuron* directory and is invoked by typing “`genesis Neuron`” after changing to this directory. The command for starting other GENESIS tutorials is of the same form. Thus, instructions for starting tutorials in this book typically start out by suggesting something like “change to the *Scripts/neuron* directory”. This should be interpreted to mean that you should give the UNIX command “`cd /usr/genesis/Scripts/neuron`”. If *Scripts* has been placed elsewhere, you may need to specify a different path.

Once you are in the correct directory, type the command “`genesis Neuron`”. (Don’t forget that UNIX is case-sensitive, so you will have to type the capital N!) If the files have been properly installed and the necessary paths have been set, you should be rewarded with some messages which indicate that GENESIS is loading. Eventually, a number of graphs and a “control panel” will appear on the screen. (If not, you or your system administrator should consult Appendix A, “Acquiring and Installing GENESIS”.)

3.3.2 The Control Panel

Figure 3.1 shows the left-hand portion of the control panel for the tutorial *Neuron*, which should now be present near the bottom of your screen.



Soma Inj 0.0002
STEP (msec) 80
Cable Compts. 0
Dend 1 Inj 0
dt (msec) 0.01
Dend 2 Inj 0

Figure 3.1 Part of the *Neuron* control panel, showing the various types of XODUS widgets that are used to control the simulation.

The control panel and the other windows that appear are examples of GENESIS *forms*. Like the terminal window and the other windows that are used with X Windows, they may be moved and resized by clicking and dragging with the mouse. This is particularly useful if the resolution of your display doesn’t allow everything to be visible on your screen. Try moving the control panel up and down and resizing it. (The procedure will vary depending on the window manager that is used with your system, but forms are normally resized by clicking on the icon at the right end of the title bar and dragging the mouse.) The several rectangular areas in the control panel are XODUS *button*, *label*, *dialog box* and *toggle* widgets. On a color display, these four types of widgets will each have their own distinctive colors. Most simulations have buttons corresponding to the `HELP`, `RESET`, `STOP` and `QUIT` buttons shown here. Clicking the left mouse button on one of these will cause GENESIS to perform the specified action. (Unless otherwise stated, the instructions "click on ... " will mean "click the left mouse button on ....")

The three dialog boxes underneath the `Injection Current` label are used to display information or to enter data from the keyboard. For example, the `Soma Inj` dialog box is used to hold the value of the injection current that is being applied to the soma in this model (see Chapter 6). To change the data field of any particular dialog box, use the mouse to move the cursor into the dialog box. Then use the keyboard right and left arrow keys to move the "`^`" symbol to the right of the place where you wish to make the change. Then use the "Delete" key to back up over anything you wish to change, and type in the changes. To cause the changes to be entered and acted upon, you must then hit "Return" (or 'Enter," on some keyboards) while the cursor is in the data field. (NOTE: forgetting to hit "Return" after changing the contents of a dialog box is a common mistake for novice GENESIS users. If you ever suspect that the old values are being used in the running of a simulation, move the cursor back into the dialog box, hit "Return," and repeat the run.) Clicking the mouse on the label field at the left of a dialog box has the same effect as hitting "Return" in the data field. This is often done when you don't want to change the data in the dialog box, but you would like the simulation to act upon the existing data. For example, the STEP dialog is used to both enter the number of milliseconds of simulation time and to run the simulation for this amount of time. Try clicking on the `STEP` label of the box or hitting "Return" when the cursor is in the data field containing "`80`." This should perform the default simulation, which is to apply a 0.2 *nA* current injection pulse to the soma of the cell. When it has finished, click on `RESET` to reset the simulation to time step 0 and clear the graphs.

The dialog box labeled `dt` gives the step size to be used in the numerical integration of the differential equations that describe the model. Although the default value is usually appropriate, you should pay attention to the considerations mentioned in Sec. 2.4.2 when making significant changes in the simulation parameters.

The rectangular areas labeled `Overlay OFF` and `Plot Soma` are examples of toggle widgets. Clicking on a toggle causes the label and the associated state of the toggle to alternate between two values. For example, after changing to `Overlay ON`, the graphs will not be cleared after clicking on `RESET`. This is useful when you wish to compare the results of two runs using different parameters. Click on `Plot Soma`. What happens?

3.3.3 Using Help Menus

In many cases, clicking on a button causes another window or menu with its own widgets to appear. This is the case with the `HELP` button. Each tutorial has a `HELP` button that may be used to call up a menu of topics. Both scrollable text windows and image windows are available. The latter are used for drawings that illustrate the experimental arrangement of the model being used, or to display typical experimental results scanned from journal articles. Figure 3.2 shows the help menu for this tutorial at the right.

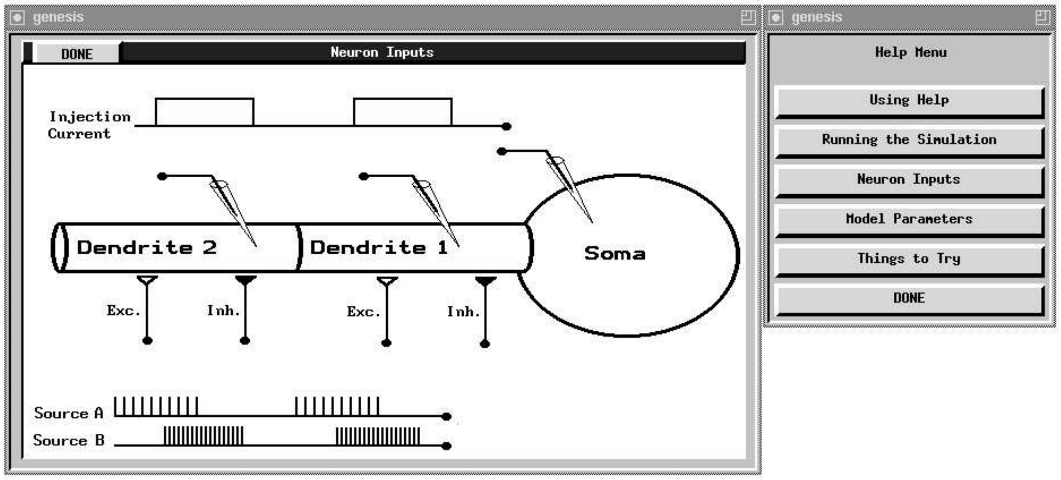


Figure 3.2 The HELP menu on the right has been used to call up the `Neuron Inputs` diagram on the left.

The drawing at the left was brought up by clicking on the `Neuron Inputs` button in the help menu. Many of the tutorials have a help menu selection similar to `Running the Simulation`. This brings up a text window with a description of the model that is depicted in the `Neuron Inputs` diagram, as well as a description of each of the control panel widgets.

In this particular tutorial, we have a soma and two dendrite compartments that are connected by axial resistances, as shown in Fig. 2.3 (page 11). The soma has voltage-activated channels like the ones modeled by Hodgkin and Huxley and described in Chapter 4. The dendrite compartments have both excitatory and inhibitory synaptically activated channels that respond to spikes applied at the synapses. These spike trains represent possible inputs delivered from the axons of other neurons. Chapter 6 uses this simulation to illustrate the properties of these types of channels. The diagram illustrates the different types of stimulation that we can apply. For example, we can inject pulses of current into any of the compartments, or connect spike trains to any of the synapses with a specified synaptic weighting. The `Inputs` button in the control panel brings up a window with dialog boxes to set the timing of the injection pulses and the bursts of spikes from sources A and B.

In any particular model or tutorial, there are typically many parameters of the model that one would like to vary. Other buttons in the control panel (not shown in Fig. 3.1) bring up menus with dialog boxes for changing the values of many of the parameters that characterize the variables in Eq. 2.1. Most of the tutorial simulations have help menu selections similar to `Model Parameters` that describe these options. Usually, there is a selection like `Using Help` that describes the use of the help system and how to move through the text in the text windows. These secondary "pop-up" menus will have a button (usually labeled `DONE`, or `Dismiss`) that is used to hide the window again. When you have finished with the `Neuron Inputs` diagram, click on the `DONE` button. Likewise, clicking on the `DONE` button in the help menu hides it from view.

In some tutorials, including *Neuron*, the interface has been designed to allow the model itself to be altered. In *Neuron*, for example, one can put any number of passive "cable" compartments between the two dendrite compartments. These are similar to the dendrite compartments, but have no variable channel conductances. This will let us see what happens if we have spatially separated inputs to the neuron. Chapter 5 treats the passive propagation of electrical signals in neuronal cables in considerable detail. In the *Neuron* tutorial, cable compartments are added by entering a number in the `Cable Compts` dialog box. Other menus (described in Sec. 6.5) allow you to vary the parameters of these compartments.

3.3.4 Displaying the Simulation Results

GENESIS can use the XODUS graph widgets to plot any quantity of interest. For example, in *Neuron* one can look at the membrane potential or channel conductances of any particular compartment. Figure 3.3 shows the results of performing one of the experiments described in the `Things to Try` help menu selection. This experiment is also described in Chapter 6. A train of spikes at 2 *msec* intervals is delivered to the excitatory synapse in the first dendrite compartment and a burst of spikes to the inhibitory synapse arrives 10 *msec* later. The plots show the channel conductances and membrane potential in this compartment and in the soma.

XODUS allows flexible control of all the graphs it generates. For example, the scales of the graphs can be changed by clicking on the `scale` button, changing the values in the appropriate dialog boxes which appear, and pressing the `DONE` button. For example, you may wish to increase the time scale (`xmax`) if you click on `STEP` more than once without resetting, or you may wish to view a more limited range of data in order to increase the resolution. The existing data will be replotted whenever the scale is changed.

In addition to generating plots like those shown in Fig. 3.3 or presenting images like the diagram in Fig. 3.2, XODUS also has the ability to generate a representation of a cell or a network of cells that can be used to interact with the simulation or to view some quantity of interest throughout the cell or the network. In several of the simulations, color is used to display the propagation of action potentials throughout a multi-compartmental model and to identify the compartments in which they are generated. An example of this can be found in Chapter 7.

Figure 7.3 represents a model of a hippocampal CA3 pyramidal cell. The mouse was used to plant an injection electrode in the soma, and recording electrodes in the soma and a position in the apical dendrite. In this gray-scale rendering of the color display, the light region to the right represents areas in the apical dendrite where calcium channels generate

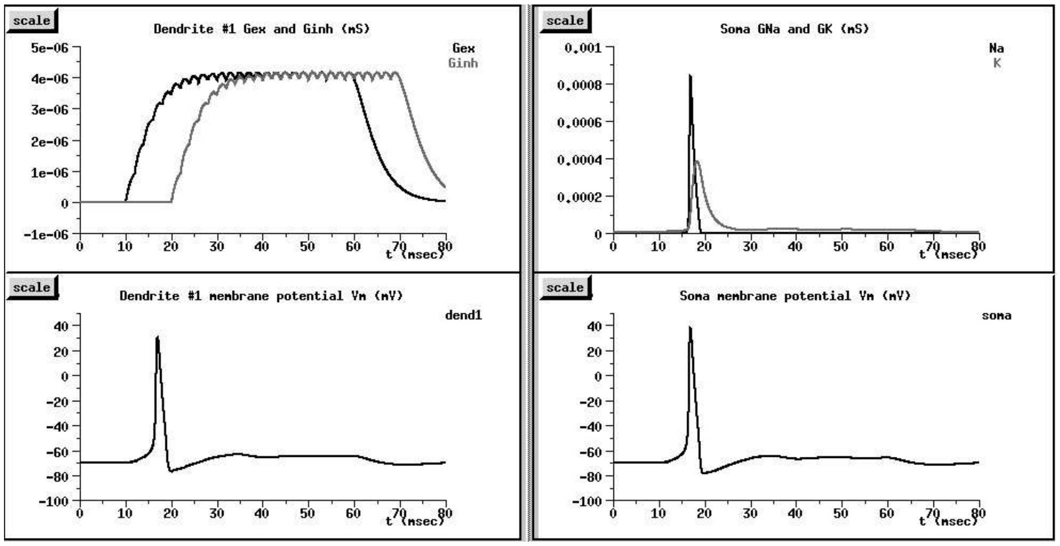


Figure 3.3 Some of the graphs that are produced after providing a combination of excitatory and inhibitory inputs to the model neuron.

action potentials. These are triggered by the propagation of action potentials from sodium channels in the soma. In Chapter 7, we discuss the interesting interactions between different voltage-activated channels that lead to the behavior shown in the figure. However, our first step is to understand how voltage-activated sodium and potassium channels can generate action potentials. This is the subject of the following chapter.