

The NEURON Simulation Environment in Epilepsy Research

Chapter for "Computational Neuroscience in Epilepsy"

N.T. Carnevale

Department of Psychology

Yale University, New Haven, CT

Michael Hines

Departments of Computer Science and Neurobiology

Yale University, New Haven, CT

The first question a busy investigator will probably ask about any potential research tool is "how might this be useful in my work?" If the answer is favorable, the next question is "what special features does it have, that will help me get the job done?" The aim of this chapter is to provide concise answers to both of these questions with regard to the NEURON simulation environment, in the context of epilepsy research. For more detail about the topics discussed here, please see *The NEURON Book* (Carnevale and Hines 2006) and the on-line documentation at <http://www.neuron.yale.edu/>.

How might NEURON be useful in epilepsy research?

Like every other simulator, NEURON has its own "domain of applicability," or range of problems to which it is best suited. The most direct indication of a simulator's domain of applicability may be the kinds of problems that it was designed to solve. Another way to judge a simulator's domain is to find out who has been using it, and what they have been using it for.

NEURON has long been recognized for its facility with mechanistic models of single cells that involve complex anatomical and biophysical properties (Hines and Carnevale 1997). Where models of individual neurons are concerned, NEURON's domain extends "down" to the level of subcellular mechanisms including: ion accumulation, diffusion, and transport; chemical reactions; deterministic and stochastic gating of voltage- and ligand- dependent ion channels. Its domain extends "up" to the level of the extracellular milieu close to the cell membrane, i.e. local concentrations and electrical field, which is sufficient for modeling phenomena such as extracellular ion accumulation and extracellular stimulation and recording.

Networks represent another dimension of model complexity. NEURON was being used for network modeling even in the early 1990s (Destexhe et al. 1993; Destexhe et al. 1994a), but over the past decade there have been two major enhancements to its utility for network modeling. The

first was the advent of an event delivery system for computationally efficient simulation of spike triggered synaptic transmission in networks of biophysical and artificial (integrate and fire) spiking cells (Hines and Carnevale 2003, 2004). The second and more recent enhancement was the ability to execute "distributed simulations," in which models of single cells and/or networks are distributed over multiple processors in a parallel processing environment (standalone multiprocessor workstation, cluster of workstations, or parallel supercomputer (Migliore et al. 2006; Brette et al., submitted for publication)).

As mentioned above, actual usage is another indicator of a simulator's domain. This has the advantage of being a very practical guide, but it is also a "lagging indicator" because there may be a significant interval--months to a year or more--between the announcement of a new feature, and the appearance of research papers that make use of it.

That said, readers may get a quick idea of who is currently using NEURON, and for what purposes, just by browsing the other chapters of this book. A more complete picture can be gleaned from <http://www.neuron.yale.edu/neuron/bib/usednrn.html>, NEURON's online bibliography. As of September 2006, this lists more than 600 publications that report work done with it. Many of these were written by well known neuroscientists engaged in epilepsy research. Some are on mechanisms of action of anticonvulsant drugs (Lytton 1997; Poolos et al. 2002) or address specific types of epilepsy, such as absence or "spike and wave" seizures (Lytton et al. 1997; Destexhe 1998; Destexhe 1999; Destexhe et al. 2001), posttraumatic epilepsy (Bush et al. 1999; Houweling et al. 2005; Santhakumar et al. 2005), or epilepsy associated with channelopathies (Chen et al. 2001; Spampanato et al. 2004a, 2004b; Vitko et al. 2005; Barela et al. 2006). Most of the other papers in the bibliography are about cellular mechanisms or network phenomena that are relevant to epilepsy in one way or another. The full scope of these studies is too broad to be covered in this chapter, but some idea of what has been done may be conveyed

by the following partial list of topics, which includes a few citations that are provided more for illustration than completeness:

- properties of voltage gated currents in central neurons (Baranauskas and Martina 2006; Chen et al. 2006)
- synaptic physiology and integration (Carnevale et al. 1997; Kapur et al. 1997a, 1997b; Jaffe and Carnevale 1999; Magee and Cook 2000; Gullledge and Stuart 2003)
- the relationship between neuronal form and function (Mainen and Sejnowski 1996; Vetter et al. 2001; Migliore et al. 2005a)
- neuronal excitability and bursting (Fleidervish et al. 1996; Colbert and Pan 2002; Carr et al. 2003; Gillies and Willshaw 2006)
- dendritic spikes (Migliore et al. 1999; Day et al. 2005; Jarsky et al. 2005; Kampa and Stuart 2006)
- activity-dependent intra- and extracellular concentration changes, and their effects on neuronal function, e.g. intracellular calcium accumulation (Baccus 1998; Sheasby and Fohlmeister 1999; Sikora et al. 2005; Gillies and Willshaw 2006), second messengers (Destexhe et al. 1994b; Destexhe and Sejnowski 1995), and changes during spreading depression (Kager et al. 2000, 2002; Somjen 2001, 2004)
- networks of inhibitory interneurons (Bartos et al. 2001, 2002; Vida et al. 2006)
- inhibitory interneurons as modulators of neuronal excitability (Pare et al. 1998; Aradi et al. 2002, 2004) and network activity (Aradi and Soltesz 2002)
- network synchrony (Destexhe et al. 1998; Bartos et al. 2002; Sohal and Huguenard 2003) and oscillations (Destexhe et al. 1999; Sohal and Huguenard 2005)

- gap junctions and their effects on neurons (Gibson et al. 2005) and networks (Moortgat et al. 2000; Saraga et al. 2006)
- extracellular fields produced by neuronal activity (Bedard et al. 2004; Moffitt and McIntyre 2005; Gold et al. 2006)
- brain stimulation with extracellular electrical (McIntyre and Grill 1999; McIntyre et al. 2004; Miocinovic and Grill 2004) or magnetic fields (Kamitani et al. 2001)

How NEURON facilitates modeling of neurons and networks

Unlike general purpose programming languages and simulation tools which are intended to be used for a wide range of problems, NEURON is specifically designed to be a simulation environment for computational neuroscience research. Its flexibility and convenience are largely attributable to its interpreter and its graphical user interface (GUI).

NEURON's hoc is a much enhanced version of the interpreter first described by Kernighan and Pike (Kernighan and Pike 1984). Many of its enhancements are specific to the task of representing the properties of biological neurons, while others include extensions for object-oriented programming, customization of the GUI, optimization, and computation in a parallel processing environment. There is also a special language called NMODL for adding compiled functionality to the NEURON executable.

The GUI has powerful tools for setting up and managing the properties of models, running simulations, and plotting variables as functions of time and position. For a wide range of problems, the GUI tools eliminate the need to write any code at all. They are particularly helpful in model prototyping, debugging, and exploratory simulations, and they are often sufficient to do

publication quality work. The most flexible and productive practice is to use the GUI tools in combination with hoc programming in order to exploit the strengths of both.

A full review of NEURON's features that help users deal with common tasks that arise in the course of modeling biological neurons and networks would cover each of these topics:

- model specification and management
- customization of the user interface
- initialization and simulation flow control
- achieving simulation stability, speed and accuracy
- analysis of simulation results

Space limitations constrain this chapter to model specification and management. Most of the other topics are covered in some detail in NEURON's extensive online documentation and tutorials (see <http://www.neuron.yale.edu/neuron/docs/docs.html>, *The NEURON Book* (Carnevale and Hines 2006), and, especially with regard to simulation speed and analysis of simulation results, "Simulation of large networks" by Lytton et al. in this book).

Model specification and management

NEURON's advantages over general purpose programming languages or simulators start to become evident as soon as the conceptual model, which is in the mind of the modeler, is ready to be transformed into a computational implementation. As we point out in chapter 2 of (Carnevale and Hines 2006), a computational model can provide insight to a conceptual model only if there is a close match between the two. Establishing and verifying such a match can be difficult with mechanistic models of biological neurons because of anatomical and biophysical complexities. If

instrumentation effects must be taken into account, it is also necessary to represent the electrical properties of electrodes and circuit components such as amplifiers and voltage or current sources. Network connections add yet another dimension to the problem.

In any case, models that involve biological neurons are described by families of algebraic equations, ordinary differential equations, kinetic schemes, and the cable equation. With a general purpose simulator or programming language, such as Matlab or C++, these equations must be expressed in a form that can be solved by a digital computer. Doing this may require writing hundreds or thousands of lines of code, and the end result will not be easy to understand, let alone debug or maintain. This is much simpler with NEURON because its programming syntax and graphical interface tools have many features that are close counterparts to familiar neuroscience concepts. This reduces the effort required to implement a computational model in the first place, and the resulting model specifications are far more compact and easier to understand and maintain. In the following sections we examine NEURON's special features that facilitate creating models of individual cells, expanding its library of biophysical mechanisms, incorporating instrumentation effects, and building network models.

Modeling individual neurons

In NEURON, models of individual nerve cells can be implemented as mechanistic models that include representations of the anatomical and biophysical properties of real cells, or as artificial spiking cells which are mathematical abstractions (e.g. "integrate and fire cells"). Here we concentrate on mechanistic models; artificial spiking cells are discussed later in the context of network models.

Creating a mechanistic model of a neuron involves specifying its branched topology, the geometry (physical dimensions) and biophysical properties of its branches, and the spatial

discretization (compartmentalization) of the model. In NEURON this can be done by writing hoc code or by using a GUI tool called the CellBuilder. Writing hoc code is an efficient way to specify the properties of small models, or of models that can be constructed algorithmically. The CellBuilder is very convenient for small models, but its greatest strength lies in managing models based on detailed morphometric data, and models with biophysical parameters that are spatially inhomogeneous. Both approaches offer features that help users focus on important biological properties rather than being distracted by computational details. To understand these features and why they are helpful, it helps to know something about how mechanistic models of neurons are specified in NEURON.

Basic anatomical and biophysical properties

The building blocks of mechanistic neuron models are *sections*, which are unbranched cables that correspond to individual neurites (see (Hines and Carnevale 2000), chapter 5 in (Carnevale and Hines 2006)). Here we show how sections allow users to create model specifications while thinking about neuronal architecture and biophysical properties, instead of having to recast everything as cable equations. As a concrete example, we will develop a model that consists of a soma with an axon, both of which have the same biophysical properties as the Hodgkin-Huxley squid axon model. This is most easily done with CellBuilder, a graphical tool for specifying the properties of mechanistic model neurons, whether stylized (stick figure) or anatomically detailed. However, for didactic purposes we will set up our model by writing hoc code.

Each section has its own anatomical and biophysical properties, which may vary along its length (e.g. diameter, specific membrane capacitance, channel and pump densities, membrane potential, current densities, ionic concentrations). These are called "range variables" because their dependence on location is described in terms of arc length or "range", which is the

fractional distance along the length of a section. That is, range is 0 at one end of a section and 1 at the other, so `axon.v(0.5)` refers to membrane potential at the middle of a section called `axon` (see (Hines and Carnevale 1997), chapter 5 in (Carnevale and Hines 2006)).

Sections can be assembled into a tree structure that represents the topology of a cell. For example, this hoc code sets up the topology of a simple model that consists of a soma with an attached axon

```
create soma, axon
connect axon(0), soma(1)
```

by attaching the 0 end of `axon` to the 1 end of `soma`. NEURON automatically takes care of boundary conditions between sections that are connected to each other.

The geometry of a newly created section is a cylinder with diameter 500 μm and length 100 μm , so it is almost always necessary to change these values. To make `axon` 1 μm wide and 1000 μm long, we would use this simple stylized specification of geometry

```
axon.diam = 1
axon.L = 1000
```

Similar statements would set `soma`'s diameter and length to 20 μm . Anatomically detailed models are discussed later in this paper.

A new section has cytoplasmic resistivity and membrane capacitance, but no membrane conductance. This means that the membrane of `axon` has no ion channels yet, so its electrical properties are described by the equation for a cylindrical cable with capacitive insulation

$$C_m \frac{\partial V}{\partial t} = \frac{D}{4R_a} \frac{\partial^2 V}{\partial x^2} \quad \text{Eq. 1}$$

where C_m , D , and R_a are specific membrane capacitance, diameter, and cytoplasmic resistivity (default values of C_m and R_a are $35.4 \text{ } \Omega \text{ cm}$ and $1 \text{ } \mu\text{f/cm}^2$, respectively). In passing, we should mention that the cable properties of any section may be augmented by an "extracellular" mechanism that adds external layers to the cable; this is useful for representing myelination, variations of extracellular resistivity, or for extracellular recording and/or stimulation.

Many sources of electrical or chemical signals, such as voltage-gated ion channels, are distributed over the cell surface or throughout its cytoplasm. In NEURON, these are represented with "density mechanisms" which are quantified with density units such as Siemens per square centimeter. These can be added to a section with the "insert" keyword. For example,

```
axon insert hh
```

puts the Hodgkin-Huxley model's ion channels into axon's membrane, so that this section is now described by the following cable equation

$$C_m \frac{\partial V}{\partial t} + \bar{g}_{Na} m^3 h (V - E_{Na}) + \bar{g}_K n^4 (V - E_K) + \bar{g}_L (V - E_L) = \frac{D}{4R_a} \frac{\partial^2 V}{\partial x^2} \quad \text{Eq. 2}$$

(cf. Eq. 29 in (Hodgkin and Huxley 1952)), where the \bar{g}_X and E_X are the respective conductance densities and equilibrium potentials, and m , h , and n are the familiar sodium and potassium gating variables.

To summarize what we have seen so far, this hoc code

```
create soma, axon
connect axon(0), soma(1)
soma.diam = 20 // 20x20 um cylinder has same surface area
soma.L = 20 // as a 20 um diameter sphere
axon.diam = 1
axon.L = 1000
forall insert hh // this inserts hh into all sections
```

specifies the biological properties of a model cell with a 20 μm diameter soma and a 1 μm diameter, 1000 μm long axon. Note that up to this point, we have concentrated entirely on the model's biological properties, while completely ignoring the issue of spatial discretization.

Anatomically detailed cell models

Stylized geometry specification is satisfactory for stick figure models, but NEURON also offers a 3-D specification method that gives greater control over section geometry, which is particularly useful for models based on morphometric reconstructions. The 3-D specification accepts a sequence of (x, y, z) coordinates plus local diameters to capture and preserve the complexities of neuronal anatomy (see chapter 5 of *The NEURON Book* (Carnevale and Hines 2006)). Creating and managing models based on detailed morphometric data has been greatly simplified by two GUI tools: Import3D and the CellBuilder.

Import3D can read the most common formats (Eutectic, NeuroLucida, SWC), and then export a specification of topology and geometry to the CellBuilder for further management ((Hines and Carnevale 2000), online tutorial at <http://www.neuron.yale.edu/neuron/docs/import3d/main.html>). Import3D preserves 3-D information, unlike some older software that destroyed 3D information and produced stylized

geometry specifications. Furthermore, it automatically diagnoses and fixes many common errors that afflict such data, and helps users identify and repair problems that require informed judgment.

The CellBuilder is a very convenient tool for specifying the biophysical properties and spatial discretization of anatomically detailed cell models (see online tutorial at <http://www.neuron.yale.edu/neuron/docs/cbtut/main.html>). Its Topology page can even be used to perform "cell surgery" through a sequence of click and drag operations that create, copy, delete, or reposition sections or subtrees. While the CellBuilder can be used in conjunction with the Import3D tool, it can also import topology and geometry directly from an anatomically detailed working model, e.g. one of the published models in ModelDB (<http://senselab.med.yale.edu/senselab/modeldb/>).

Import3D and the CellBuilder have made it much easier to reuse morphologies that are becoming available from a wide range of sources, including ModelDB and databases of anatomical data such as NeuroMorpho.org (<http://neuromorpho.org/>). However, one should always be concerned about data quality (Horcholle-Bossavit et al. 2000; Scorcioni et al. 2004), and of course it is always important to cite the source of any data that one reuses.

Painless spatial discretization

Regardless of programming language or simulation environment, mechanistic models generally deal with extended neuronal anatomy by discretizing the cable equation ("breaking the cell into compartments"). The same is true in NEURON, but discretization is handled in such a way that users rarely have to think about compartments.

NEURON discretizes the cable equation according to the central difference approximation, which yields a set of ordinary differential equations (ODEs) that govern membrane potential and

current at one or more points (called "nodes") that are spaced evenly along the interior of a section. Each section has its own discretization parameter called `nseg`, which is a whole number that specifies how many internal nodes it has. The spatial grid in each section therefore has a local resolution given by section length / `nseg`. It may help to think of a section as being cut into `nseg` segments of equal length, with a node at the middle of each segment, and all of the membrane properties of a segment lumped together at the segment's node (Fig. 1). In terms of normalized distance from one end of the section, the internal nodes are located at $(j+0.5)/nseg$ for $j=0$ to $nseg-1$.

<< FIGURE 1 NEAR HERE >>

The membrane potential v_j at the internal node of the j th segment is defined by an ODE of the form

$$c_j \frac{dv_j}{dt} + i_{ion_j}(v_j, t) = \sum_k (v_k - v_j) / r_{jk} \quad \text{Eq. 3}$$

where c_j is the total membrane capacitance of the segment, i_{ion_j} is the net ionic current through the segment's membrane, v_k are the membrane potentials at the nodes of adjacent segments, and r_{jk} are the (cytoplasmic) resistances along the paths between those nodes and the j th node.

Equation 3 is just the current balance equation for segment j : the left hand side is the sum of the membrane capacitive and ionic currents that leave the segment, and the right hand side is the sum of the axial currents that flow into it from its neighbors.

The default value of `nseg` is 1, but larger values may be required for spatial accuracy in sections that are long and narrow, especially if cytoplasmic resistivity and/or specific membrane capacitance is large. A simple assignment statement is all that is necessary to change `nseg`, and NEURON will then automatically update the number of ODEs it must solve, calculating the

values of c_j , r_{jk} , and the terms necessary to compute i_{ion_j} from surface area, ionic conductance densities or permeabilities, gating variables, and equilibrium potentials in each segment of the section. To ensure correct placement of spatially inhomogeneous parameters (e.g. channel densities) and localized signal sources (see **Synapses, clamps, and other localized signal sources** below), it is a good idea to defer specification of density mechanisms, synapses, voltage and current clamps etc., until after `nseg` values have been assigned.

That still leaves the question of how big `nseg` should be. We have generally obtained a good balance of spatial accuracy and computational efficiency by using the `d_lambda` rule with a criterion of $0.1 \lambda_{100\text{Hz}}$ (i.e. make `nseg` the smallest odd number so that segment length is shorter than one tenth of a space constant at 100 Hz ((Hines and Carnevale 2001), chapter 5 of (Carnevale and Hines 2006)). For our model cell, `soma` is electrically compact so we can leave its `nseg` unchanged. However, according to the `d_lambda` rule, `axon` needs a value of 21 for good spatial accuracy. Revising our hoc code to include discretization gives us

```
create soma, axon
connect axon(0), soma(1)
soma.diam = 20 // 20x20 um cylinder has same surface area
soma.L = 20 // as a 20 um diameter sphere
axon.diam = 1
axon.L = 1000
axon.nseg = 21 // assign nseg before inserting mechanisms
forall insert hh // insert hh into all sections
```

which is a complete specification of the model (note that we followed our advice of assigning the value of `nseg` before inserting biophysical mechanisms). Executing these statements with hoc will set up the internal data structures for the family of ODEs that constitute the model's discretized cable equation, and which, along with the ODEs for the gating variables, are

numerically integrated by NEURON in the course of a simulation. The user never has to write a single cable equation, let alone bother with boundary conditions or discretization. The economy and clarity of this model specification contrasts sharply with the complexity that would be needed with any general purpose programming language or generic mathematical simulation environment.

Before leaving this topic, we should mention that spatial discretization is extremely easy with models managed by the CellBuilder, regardless of model complexity. This is so easy that, even though the `d_lambda` rule can be applied with hoc code (see http://www.neuron.yale.edu/neuron/docs/d_lambda/d_lambda.html), we must admit to using the CellBuilder to discover the appropriate value of `nseg` for the simple example shown above. The Geometry page offers a choice of three different discretization methods which can be applied to any individual section or set of sections:

- manual entry: user specifies value of `nseg`
- `d_X` method: user specifies maximum segment length in microns
- `d_lambda` method: user specifies maximum segment length as a fraction of $\lambda_{100\text{Hz}}$

With the `d_X` and `d_lambda` methods, NEURON automatically chooses the smallest odd value of `nseg` that satisfies the discretization criterion (odd to ensure that there will be a node at the midpoint of the section).

Spatially inhomogeneous properties

Sometimes it is known that certain kinds of ion channels are present in one region of a cell, but absent from another, or that there is a systematic variation of channel density with position (e.g. (Hoffman et al. 1997)). Such spatial inhomogeneities can be managed most effectively by

grouping sections into subsets, which greatly simplifies the specification of biophysical properties. This can be done with hoc statements, but it is easiest with the CellBuilder, especially for anatomically complex models. This tool's Subsets page has a canvas that provides immediate visual feedback to guide and verify one's selections, plus a menu of selection choices (individual section, subtree) and logical operations (e.g. union, intersection, invert, subtract) that can reduce the effort of creating subsets to just a few mouse clicks.

The simplest application of subsets is to assert uniform properties over a group of sections, e.g. so that only the basilar dendrites have a particular voltage-gated calcium channel, or to reduce the sodium channel density uniformly over the entire apical dendritic tree. In recent versions of NEURON, subsets can be used to specify that biophysical parameters vary with position according to user-specified rules ((Hines and Carnevale 2000), online tutorial at <http://www.neuron.yale.edu/neuron/docs/cbtut/parameterized/outline.html>). This allows the modeler to stipulate that, for all sections in a particular subset, some parameter (e.g. an ion channel density) is governed by $param = f(p)$ where f can be any function, and p is one of these distance metrics:

- path length from a reference point
- radial distance from a reference point
- distance from a plane ("3D projection onto a line")

Figure 2 shows a CellBuilder in which the density of Hodgkin-Huxley sodium channels in the apical dendritic tree of a model cell decreases linearly with path distance from the soma, from full density (0.12 S/cm^2) at the proximal end of the apical trunk, to zero at the most distal dendritic termination.

<< FIGURE 2 NEAR HERE >>

Synapses, clamps, and other localized signal sources

Earlier we mentioned that density mechanisms are used to represent biophysical properties that are distributed over the surface of a cell or throughout its cytoplasm. However, models often must include signal sources that are very localized, such as synapses and electrodes, which are best described using localized net current in nanoamperes and conductance in microsiemens. In NEURON these are represented by "point processes," which are managed with an object syntax. For example, this code creates a new instance of the `IClamp` class and attaches it to the `soma` of our model cell

```
objref stim
// attach a current clamp to middle of soma
soma stim = new IClamp(0.5)
stim.del = 1 // ms delay
stim.dur = 0.1 // ms duration
stim.amp = 0.5 // nA amplitude
```

Adding new kinds of mechanisms

NEURON has a special programming language called NMODL that can be used to enlarge its library of biophysical mechanisms (active currents, buffers, diffusion, exchange/pumps etc.) ((Hines and Carnevale 2000), chapter 9 in (Carnevale and Hines 2006)) and also to define new types of synapses and artificial spiking cells that can interact with NEURON's event delivery system (see **Modeling networks** below). NMODL offers many examples of how NEURON allows modelers to work with familiar concepts and focus on the biology, while shielding them from computational details. Its notation for specifying dynamics is very similar to kinetic schemes or differential equations, as illustrated by these excerpts from the source code for a calcium pump

```

: k1 & k3 are the forward rate constants
~ cai + pump <-> capump (k1, k2)
~ capump <-> pump + cao (k3, k4)

```

and a Hodgkin-Huxley style voltage-gated sodium conductance

```

m' = (minf - m) / mtau
h' = (hinf - h) / htau

```

Complete specifications of biophysical mechanisms tend to be very compact, with each line equivalent to many statements in C, and the NMODL compiler automatically takes care of such details as generating code that works with each of NEURON's integration methods.

New voltage-and/or ligand-gated channels can also be added with a GUI tool called the Channel Builder ((Hines and Carnevale 2000), online tutorial at <http://www.neuron.yale.edu/neuron/docs/cbtut/main.html>), which borrows much of its design from Robert Cannon's Catacomb simulator (<http://www.compneuro.org/catacomb/>). Channels may be ohmic or described by the Goldman-Hodgkin-Katz equation, with dynamics defined by kinetic schemes or Hodgkin-Huxley-style differential equations (including the Borg-Graham formulation (Borg-Graham 1991)). The Channel Builder has several distinct advantages over NMODL, especially where ease of use is concerned. It presents graphs that describe the voltage dependence of gating variables in terms of rate constants ("alpha and beta") or steady state values and time constants ("inf and tau"; see Fig. 3). Clicking on a button switches these graphs back and forth between "alpha beta" and "inf tau" formats. Furthermore, the individual traces have handles (squares on the plots of infC1C2 and tauC1C2 in Fig. 3) that can be dragged with the mouse in order to quickly explore the effects of shifting or changing the slope of their voltage dependence. Finally, Channel Builder mechanisms actually execute more quickly than equivalent compiled NMODL code, and kinetic scheme state transitions may be deterministic or stochastic.

<< FIGURE 3 NEAR HERE >>

Models that include electronic instrumentation

It is often informative to examine the behavior of electronic circuits, study instrumentation artifacts, or simulate experimental methods such as dynamic clamping. Such tasks are simplified by NEURON's Linear Circuit Builder, a GUI tool that makes it easy to build models that have linear circuit elements and may also include neurons. This tool is a graphical front end for the Linear Mechanism class, which adds linear equations to the matrix of current balance equations that NEURON solves during a simulation. The Linear Mechanism class can even be used to implement gap junction coupling between model cells (see (Migliore et al. 2005b), source code available from ModelDB via accession number 43039). Figure 4 shows a Linear Circuit Builder used to study the effect of electrode capacitance and resistance on stimulation and recording from a patch clamped cell.

<< FIGURE 4 NEAR HERE >>

Modeling networks

A biological neural network consists of neurons and of the connections between them, so what could seem more intuitive than to use computational representations of neurons and their connections as the building blocks of a model network? We already know how to specify the properties of model cells, so what's to stop us from plowing headlong into network modeling?

Nothing, really, except that advance planning might turn up strategies for implementing network models in a way that achieves computational efficiency while preserving conceptual clarity. In the following sections we review how NEURON helps modelers implement efficient

connections and artificial spiking cells, manage large numbers of model cells and connections, and prototype small nets in order to generate "seed" code that can be reused in the algorithmic construction of large nets.

Connections between cells

Communication between cells in biological networks involves some combination of gap junctions, continuous transmitter release from presynaptic terminals, and spike-triggered synaptic transmission. In each case, the effect on the postsynaptic cell is to change some variable in an anatomically localized region, so these three forms of intracellular communication are implemented in NEURON as point processes.

Spike triggered synaptic transmission is most amenable to efficient modeling chiefly because the axon can be approximated by a threshold detector at the proximal end, followed by a delay that represents conduction latency. Eliminating the mechanistic details of spike propagation along the length of the axon reduces the number of equations that must be solved, and results in faster simulations.

In NEURON, presynaptic spike sources are connected to synaptic targets with the NetCon class, which combines the notions of threshold detection, delay, and synaptic weight to separate the notion of "connections" from the notion of "what is connected" (see chapter 10 of *The NEURON Book* (Carnevale and Hines 2006)). A NetCon object monitors some presynaptic variable (e.g. membrane potential or calcium concentration) for a threshold crossing that indicates the occurrence of a spike. After a spike is detected, an interval is allowed to pass, and then an event is delivered that perturbs ("activates") the model synapse.

The model synapse could include an elaborate implementation of mechanisms in the presynaptic terminal, but the most common practice is to increase the NetCon's delay by ~1 ms

(for synaptic latency) and let the event perturb a variable associated with a synaptic conductance whose dynamics are governed by one or two linear differential equations or a simple kinetic scheme. For example, this code

```
objref syn, nc
dend syn = new Exp2Syn(0.5) // synapse is at middle of dend
// monitor v at middle of soma, deliver events to syn
soma nc = new NetCon(&v(0.5), syn)
nc.delay = 5 // ms
nc.weight = 0.001 // microsiemens
```

allows a spike in soma to activate a biexponential synaptic conductance change at the middle of dend. The conductance change starts after a delay of 5 ms and has a peak magnitude of 1 nS.

Even when mechanistic details of axonal spike conduction and transmitter release are ignored, event-driven model synapses can still reproduce many phenomena attributed to presynaptic mechanisms. Such phenomena include fluctuations of axonal conduction latency or frank conduction failure, use-dependent synaptic plasticity (see ModelDB accession numbers 3264 and 3815 for NEURON implementations of short term potentiation and depression described originally by (Varela et al. 1997) and (Tsodyks et al. 1998)), and trial-to-trial variation of quantal number or size (see <http://www.neuron.yale.edu/ftp/ted/neuron/chance.zip>).

Artificial spiking cells

Events are also used to implement artificial spiking cells in NEURON. Artificial spiking cells are highly abstract model neurons, lacking physical extent or membrane properties, that attempt to mimic certain aspects of the function of biological neurons. They have many uses in network modeling, such as prototyping network architectures, and generating afferent spike trains that provide a "synaptic milieu" for mechanistic model neurons. For a more detailed

introduction to artificial spiking cells, see (Hines and Carnevale 2004) or chapter 10 of (Carnevale and Hines 2006); recent advances in making their dynamics more "biological" are presented in the chapter "Simulation of large networks: technique and progress" by Lytton et al. in this book.

The only way an artificial spiking cell can interact with anything is by receiving or sending events, which can be regarded as metaphors for spikes. Arrival of an input event perturbs one or more states of an artificial spiking cell, and when these states satisfy some criterion (e.g. a threshold crossing), an output event is generated. Some kinds of artificial spiking cells have dynamics that allow analytical prediction of the future course of all state variables from the present condition of the cell. Such models are suitable for event-driven simulations, which are very efficient because numerical integration is not necessary to decide if and when a cell will generate an output event. Only algebraic calculations are required, and these are necessary only when a cell receives an input event.

Like synaptic mechanisms and NetCons, NEURON's artificial spiking cells are managed with an object syntax. This

```
objref pre, post, nc
pre = new NetStim() // generates event trains
post = new IntFire1() // leaky integrate and fire
nc = new NetCon(pre, post)
nc.weight = 0.1
```

creates two different kinds of artificial spiking cells and ensures that events generated by `pre` are delivered to `post`. This is an excitatory connection, because the weight of the NetCon that will deliver these events is positive--each input event will make the membrane state variable (analog to membrane potential) jump up by 0.1; an inhibitory connection would have a negative weight.

In NEURON, events generated by artificial spiking cells and mechanistic model cells look the same to the object that receives them--that is, anything that can receive events, can receive events from anything that can send them. This means that a network model may involve any combination of mechanistic model cells or artificial spiking cells, connected in any way imaginable.

An outline for building network models

A network model in NEURON can involve any combination of artificial spiking cells and mechanistic model cells. We already know that biophysical model cells are constructed from sections and density mechanisms. We also know that an event-triggered synapse onto a biophysical model cell is a point process attached to a section, which is capable of responding to an input event by affecting membrane current, concentration of a second messenger, or some other variable at the point of attachment. Finally, we know that NetCons are used to connect event sources, such as artificial spiking cells or a spike trigger zone on a biophysical model cell, to targets such as artificial spiking cells or event-triggered synapses. Now it is time to examine how to put all these pieces together.

In broad outline, building a network model involves these steps:

1. Defining the types of cells that will be used in the net.
2. Creating the cells.
3. Setting up connections between them.

This seems simple enough, but a lot of administrative effort can be required to keep track of all the cells, the synapses attached to them, and the connections between pre- and postsynaptic

elements. Fortunately, NEURON's GUI and hoc programming language can both make this less onerous in several ways.

Object-oriented programming in network modeling

The difficulty of managing network models can be reduced by taking advantage of object-oriented programming in hoc. In particular, the code that defines a biophysical model neuron can be used to define a new object class by wrapping it inside a bit of hoc called a template. Such new object classes can be used like cookie cutters to create multiple instances of various kinds of model cells.

As an almost trivial example, suppose we want to make a network that involves 1000 identical model cells, each of which has a single section plus two synapses. This defines a new cell class called `SmallCell`

```

begintemplate SmallCell
  public soma, exc, inh
  create soma
  objref exc, inh
  proc init() {
    soma {
      insert hh
      exc = new ExpSyn(0.5) // "ampa"
      inh = new Exp2Syn(0.5) // "gaba-a"
      inh.e = -80
    }
  }
endtemplate SmallCell

```

which we can then use to create as many instances of this kind of model cell as we wish

```
objref cells
cells = new List()
for i=1,1000 cells.append(new SmallCell())
```

Lists can also be handy for dealing with collections of artificial spiking cells and NetCons, in part because they lend themselves to programming styles that can deal with any number of objects without requiring the modeler to keep track of "magic numbers" (numeric counts of how many items there are of each particular kind).

Generating reusable code with the Network Builder

A template that defines a new cell class can be created by writing hoc code by hand, but generally it is more convenient to create a new cell classes by using the Network Builder and its associated suite of GUI tools to first build a small prototype net. With these tools, the modeler can specify the properties of classes of artificial and mechanistic model cells (the latter can be imported from a CellBuilder), attach synaptic mechanisms to specific locations on mechanistic model cells, and set up a simple network architecture of interconnected cells (see tutorial at <http://www.neuron.yale.edu/neuron/docs/netbuild/main.html>). The idea is to use the Network Builder to make a simple prototype net which contains one instance of each kind of cell and synaptic connection that is of interest. This can be saved as a hoc file, which may then be mined for code that is reusable for algorithmically building large-scale nets (see chapter 11 in (Carnevale and Hines 2006)).

Network models on parallel computers

NEURON supports parallel simulation of network models in which cells on different processors are coupled by spike events (Migliore et al. 2006). This works for fixed step and variable step integration, including the local variable time step method (Lytton and Hines 2005). Because of cache effects, speedup is superlinear if the number of processors is small, and it

remains at least proportional to the number of processors as long as each one is integrating more than ~100 equations. It is straightforward to write the model specification so that the same code will run on standalone single processor machines or parallel hardware, with setup time scaling properly with the number of processors (e.g. source code for (Migliore et al. 2006) and (Brette et al, submitted for publication), available from ModelDB via accession numbers 64229 and 83319).

For optimum performance, parallel simulations can be carried out in batch mode, saving only a record of spike activity (time of each spike and identity of the spiking cell). The detailed time course of any variable can then be efficiently recreated from the entire network's spike data. This is done by re-simulating any subset (even 1) of neurons with the aid of the GUI, using the PatternStim class to provide as input just those events that would have been generated by the rest of the network. The results for the subnet are quantitatively identical to the full network simulation.

NEURON's parallel network capabilities have been extended to include interprocessor gap junctions and synapses where post-synaptic state is continuously dependent on presynaptic voltage. Communication overhead is greatly increased for such models since voltages must be exchanged at every time step. At present, gap junctions in combination with discrete events can use only the fixed step method but this will be soon extended to global variable step method.

The Model View tool: model analysis at runtime

A common problem in debugging and maintaining one's own models, and an almost universal challenge in dealing with anybody else's models, is the difficulty of understanding just what is in any given model. Code has a tendency to grow like Topsy with little overall structure and few comments to guide the reader. Statements that define model properties are often intermingled with user interface, control, and initialization code. Differences in programming

styles and strategies can make even well-structured code difficult for anyone but the model's author to understand, especially if the model is assembled algorithmically or has a complex initialization.

Model View is a GUI tool that analyzes model properties at runtime and presents the results of its analysis in the form of a compact, browseable outline, with pop-up graphs that display the spatial distribution of inhomogeneous parameters. In the future, Model View will also enable interoperability with other simulators by exporting and importing model specifications in XML.

Summary

In this chapter we have presented an overview of features of NEURON that facilitate the specification and management of models of cells and networks. Extensive documentation of these features and other topics related to the construction and use of models is provided in *The NEURON Book* (Carnevale and Hines 2006) and on-line documentation at <http://www.neuron.yale.edu/>.

Acknowledgments

Supported by NINDS NS11613.

References

Aradi, I., Santhakumar, V., Chen, K., and Soltesz, I. Postsynaptic effects of GABAergic synaptic diversity: regulation of neuronal excitability by changes in IPSC variance. *Neuropharmacology* 43:511-522, 2002.

- Aradi, I., Santhakumar, V., and Soltesz, I. Impact of heterogeneous perisomatic IPSC populations on pyramidal cell firing rates. *J. Neurophysiol.* 91:2849-2858, 2004.
- Aradi, I. and Soltesz, I. Modulation of network behaviour by changes in variance in interneuronal properties. *J. Physiol.* 538:227-251, 2002.
- Baccus, S.A. Synaptic facilitation by reflected action potentials: enhancement of transmission when nerve impulses reverse direction at axon branch points. *Proc. Nat. Acad. Sci.* 95:8345-8350, 1998.
- Baranauskas, G. and Martina, M. Sodium currents activate without a Hodgkin and Huxley-type delay in central mammalian neurons. *J. Neurosci.* 26:671-684, 2006.
- Barela, A.J., Waddy, S.P., Lickfett, J.G., Hunter, J., Anido, A., Helmers, S.L., Goldin, A.L., and Escayg, A. An epilepsy mutation in the sodium channel SCN1A that decreases channel excitability. *J. Neurosci.* 26:2714-2723, 2006.
- Bartos, M., Vida, I., Frotscher, M., Geiger, J.R.P., and Jonas, P. Rapid signaling at inhibitory synapses in a dentate gyrus interneuron network. *J. Neurosci.* 21:2687-2698, 2001.
- Bartos, M., Vida, I., Frotscher, M., Meyer, A., Monyer, H., Geiger, J.R.P., and Jonas, P. Fast synaptic inhibition promotes synchronized gamma oscillations in hippocampal interneuron networks. *Proc. Nat. Acad. Sci.* 99:13222-13227, 2002.
- Bedard, C., Kroger, H., and Destexhe, A. Modeling extracellular field potentials and the frequency-filtering properties of extracellular space. *Biophys. J.* 86:1829-1842, 2004.
- Borg-Graham, L.J. Modeling the nonlinear conductances of excitable membranes. In: *Cellular and Molecular Neurobiology: a Practical Approach*, edited by H. Wheal and J. Chad. New York: Oxford University Press, 1991, p. 247-275.

Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J.M., Diesmann, M., Goodman, P.H., Harris, F.C., Zirpe, M., Natschläger, T., Pecevski, D., Ermentrout, B., Djurfeldt, M., Lansner, A., Rochel, O., Vieville, T., Muller, E., Davison, A., El Boustani, S., and Destexhe, A. Simulation of networks of spiking neurons: a review of tools and strategies, *submitted for publication*.

Bush, P.C., Prince, D.A., and Miller, K.D. Increased pyramidal excitability and NMDA conductance can explain posttraumatic epileptogenesis without disinhibition: a model. *J. Neurophysiol.* 82:1748-1758, 1999.

Carnevale, N.T. and Hines, M.L. *The NEURON Book*. Cambridge, UK: Cambridge University Press, 2006.

Carnevale, N.T., Tsai, K.Y., Claiborne, B.J., and Brown, T.H. Comparative electrotonic analysis of three classes of rat hippocampal neurons. *J. Neurophysiol.* 78:703-720, 1997.

Carr, D.B., Day, M., Cantrell, A.R., Held, J., Scheuer, T., Catterall, W.A., and Surmeier, D.J. Transmitter modulation of slow, activity-dependent alterations in sodium channel availability endows neurons with a novel form of cellular plasticity. *Neuron* 39:793-806, 2003.

Chen, K., Aradi, I., Thon, N., Eghbal-Ahmadi, M., Baram, T.Z., and Soltesz, I. Persistently modified h-channels after complex febrile seizures convert the seizure-induced enhancement of inhibition to hyperexcitability. *Nature Medicine* 7:331-337, 2001.

Chen, Y.A., Yu, F.H., Surmeier, D.J., Scheuer, T., and Catterall, W.A. Neuromodulation of Na⁺ channel slow inactivation via cAMP-dependent protein kinase and protein kinase C. *Neuron* 49:409-420, 2006.

Colbert, C.M. and Pan, E.H. Ion channel properties underlying axonal action potential initiation in pyramidal neurons. *Nature Neuroscience* 5:533-538, 2002.

Day, M., Carr, D.B., Ulrich, S., Ilijic, E., Tkatch, T., and Surmeier, D.J. Dendritic excitability of mouse frontal cortex pyramidal neurons is shaped by the interaction among HCN, Kir2, and k(leak) channels. *J. Neurosci.* 25:8776-8787, 2005.

Destexhe, A. Spike-and-wave oscillations based on the properties of GABA-B receptors. *J. Neurosci.* 18:9099-9111, 1998.

Destexhe, A. Can GABA_A conductances explain the fast oscillation frequency of absence seizures in rodents? *Eur. J. Neurosci.* 11:2175-2181, 1999.

Destexhe, A., Contreras, D., Sejnowski, T.J., and Steriade, M. A model of spindle rhythmicity in the isolated thalamic reticular nucleus. *J. Neurophysiol.* 72:803-818, 1994a.

Destexhe, A., Contreras, D., and Steriade, M. Mechanisms underlying the synchronizing action of corticothalamic feedback through inhibition of thalamic relay cells. *J. Neurophysiol.* 79:999-1016, 1998.

Destexhe, A., Contreras, D., and Steriade, M. Cortically-induced coherence of a thalamic-generated oscillation. *Neurosci.* 92:427-443, 1999.

Destexhe, A., Contreras, D., and Steriade, M. LTS cells in cerebral cortex and their role in generating spike-and-wave oscillations. *Neurocomputing* 38:555-563, 2001.

Destexhe, A., Mainen, Z.F., and Sejnowski, T.J. Synthesis of models for excitable membranes, synaptic transmission, and neuromodulation using a common kinetic formalism. *J. Comput. Neurosci.* 1:195-231, 1994b.

Destexhe, A., McCormick, D.A., and Sejnowski, T.J. A model for 8-10 Hz spindling in interconnected thalamic relay and reticularis neurons. *Biophys. J.* 65:2474-2478, 1993.

Destexhe, A. and Sejnowski, T.J. G-protein activation kinetics and spillover of g-aminobutyric acid may account for differences between inhibitory responses in the hippocampus and thalamus. *Proc. Nat. Acad. Sci.* 92:9515-9519, 1995.

Fleidervish, I.A., Friedman, A., and Gutnick, M.J. Slow inactivation of Na⁺ current and slow cumulative spike adaptation in mouse and guinea-pig neocortical neurones in slices. *J. Physiol.* 493:83-97, 1996.

Gibson, J.R., Beierlein, M., and Connors, B.W. Functional properties of electrical synapses between inhibitory interneurons of neocortical layer 4. *J. Neurophysiol.* 93:467-480, 2005.

Gillies, A. and Willshaw, D. Membrane channel interactions underlying rat subthalamic projection neuron rhythmic and bursting activity. *J. Neurophysiol.* 95:2352-2365, 2006.

Gold, C., Henze, D.A., Koch, C., and Buzsaki, G. On the origin of the extracellular action potential waveform: A modeling study. *J. Neurophysiol.* 95:3113-3128, 2006.

Gulledge, A.T. and Stuart, G.J. Excitatory actions of GABA in the cortex. *Neuron* 37:299-309, 2003.

Hines, M.L. and Carnevale, N.T. The NEURON simulation environment. *Neural Computation* 9:1179-1209, 1997.

Hines, M.L. and Carnevale, N.T. Expanding NEURON's repertoire of mechanisms with NMODL. *Neural Computation* 12:995-1007, 2000.

Hines, M.L. and Carnevale, N.T. NEURON: a tool for neuroscientists. *The Neuroscientist* 7:123-135, 2001.

Hines, M.L. and Carnevale, N.T. NEURON simulation environment. In: *The Handbook of Brain Theory and Neural Networks*, edited by M.A. Arbib. Cambridge, MA: MIT Press, 2003, p. 769-773.

Hines, M.L. and Carnevale, N.T. Discrete event simulation in the NEURON environment. *Neurocomputing* 58-60:1117-1122, 2004.

Hines, M.L. and Carnevale, N.T. Recent developments in NEURON. *Brains, Minds and Media* 1 bmm221 (urn:nbn:de:0009-3-2210):2005.

Hodgkin, A.L. and Huxley, A.F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.* 117:500-544, 1952.

Hoffman, D.A., Magee, J.C., Colbert, C.M., and Johnston, D. K⁺ channel regulation of signal propagation in dendrites of hippocampal pyramidal neurons. *Nature* 387:869-875, 1997.

Horcholle-Bossavit, G., Gogan, P., Ivanov, Y., Korogod, S., and Tyc-Dumont, S. The problem of morphological noise in reconstructed dendritic arborizations. *J. Neurosci. Meth.* 95:83-93, 2000.

Houweling, A.R., Bazhenov, M., Timofeev, I., Steriade, M., and Sejnowski, T.J. Homeostatic synaptic plasticity can explain post-traumatic epileptogenesis in chronically isolated neocortex. *Cerebral Cortex* 15:834-845, 2005.

Jaffe, D.B. and Carnevale, N.T. Passive normalization of synaptic integration influenced by dendritic architecture. *J. Neurophysiol.* 82:3268-3285, 1999.

Jarsky, T., Roxin, A., Kath, W.L., and Spruston, N. Conditional dendritic spike propagation following distal synaptic activation of hippocampal CA1 pyramidal neurons. *Nature Neuroscience* 8:1667-1676, 2005.

- Kager, H., Wadman, W.J., and Somjen, G.G. Simulated seizures and spreading depression in a neuron model incorporating interstitial space and ion concentrations. *J. Neurophysiol.* 84:495-512, 2000.
- Kager, H., Wadman, W.J., and Somjen, G.G. Conditions for the triggering of spreading depression studied with computer simulations. *J. Neurophysiol.* 88:2700-2712, 2002.
- Kamitani, Y., Bhalodia, V.M., Kubota, Y., and Shimojo, S. A model of magnetic stimulation of neocortical neurons. *Neurocomputing* 38:697-703, 2001.
- Kampa, B.M. and Stuart, G.J. Calcium spikes in basal dendrites of layer 5 pyramidal neurons during action potential bursts. *J. Neurosci.* 26:7424-7432, 2006.
- Kapur, A., Lytton, W.W., Ketchum, K.L., and Haberly, L.B. Regulation of the NMDA component of EPSPs by different components of postsynaptic GABAergic inhibition: computer simulation analysis in piriform cortex. *J. Neurophysiol.* 78:2546-2559, 1997a.
- Kapur, A., Pearce, R.A., Lytton, W.W., and Haberly, L.B. GABA_A-mediated IPSCs in piriform cortex have fast and slow components with different properties and locations on pyramidal cells. *J. Neurophysiol.* 78:2531-2545, 1997b.
- Kernighan, B.W. and Pike, R. Appendix 2: Hoc manual. In: *The UNIX Programming Environment*. Englewood Cliffs, NJ: Prentice-Hall, 1984, p. 329-333.
- Lytton, W.W. Computer model of clonazepam's effect in thalamic slice. *NeuroReport* 8:3339-3343, 1997.
- Lytton, W.W., Contreras, D., Destexhe, A., and Steriade, M. Dynamic interactions determine partial thalamic quiescence in a computer network model of spike-and-wave seizures. *J. Neurophysiol.* 77:1679-1696, 1997.

Lytton, W.W. and Hines, M. Independent variable timestep integration of individual neurons for network simulations. *Neural Computation* 17:903-921, 2005.

Lytton, W.W., Stewart, M., and Hines, M.L.. Simulation of large networks: technique and progress, in *Computational Neuroscience in Epilepsy*, eds. Soltesz and Staley, 2007.

Magee, J.C. and Cook, E.P. Somatic EPSP amplitude is independent of synapse location in hippocampal pyramidal neurons. *Nature Neuroscience* 3:895-903, 2000.

Mainen, Z.F. and Sejnowski, T.J. Influence of dendritic structure on firing pattern in model neocortical neurons. *Nature* 382:363-366, 1996.

McIntyre, C.C. and Grill, W.M. Excitation of central nervous system neurons by nonuniform electric fields. *Biophys. J.* 76:878-888, 1999.

McIntyre, C.C., Grill, W.M., Sherman, D.L., and Thakor, N.V. Cellular effects of deep brain stimulation: model-based analysis of activation and inhibition. *J. Neurophysiol.* 91:1457-1469, 2004.

Migliore, M., Cannia, C., Lytton, W.W., Markram, H., and Hines, M.L. Parallel network simulations with NEURON. *J. Comput. Neurosci.* 21:119-129, 2006.

Migliore, M., Ferrante, M., and Ascoli, G.A. Signal propagation in oblique dendrites of CA1 pyramidal cells. *J. Neurophysiol.* 94:4145-4155, 2005a.

Migliore, M., Hines, M.L., and Shepherd, G.M. The role of distal dendritic gap junctions in synchronization of mitral cell axonal output. *J. Comput. Neurosci.* 18:151-161, 2005b.

Migliore, M., Hoffman, D.A., Magee, J.C., and Johnston, D. Role of an A-type K⁺ conductance in the back-propagation of action potentials in the dendrites of hippocampal pyramidal neurons. *J. Comput. Neurosci.* 7:5-15, 1999.

Miocinovic, S. and Grill, W.M. Sensitivity of temporal excitation properties to the neuronal element activated by extracellular stimulation. *J. Neurosci. Meth.* 132:91-99, 2004.

Moffitt, M.A. and McIntyre, C.C. Model-based analysis of cortical recording with silicon microelectrodes. *Clinical Neurophysiology* 116:2240-2250, 2005.

Moortgat, K.T., Bullock, T.H., and Sejnowski, T.J. Gap junction effects on precision and frequency of a model pacemaker network. *J. Neurophysiol.* 83:984-997, 2000.

Pare, D., Lang, E.J., and Destexhe, A. Inhibitory control of somatodendritic interactions underlying action potentials in neocortical pyramidal neurons in vivo: an intracellular and computational study. *Neurosci.* 84:377-402, 1998.

Poolos, N.P., Migliore, M., and Johnston, D. Pharmacological upregulation of h-channels reduces the excitability of pyramidal neuron dendrites. *Nature Neuroscience* 5:767-774, 2002.

Santhakumar, V., Aradi, I., and Soltesz, I. Role of mossy fiber sprouting and mossy cell loss in hyperexcitability: A network model of the dentate gyrus incorporating cell types and axonal topography. *J. Neurophysiol.* 93:437-453, 2005.

Saraga, F., Zhang, X.L., Zhang, L., Carlen, P.L., and Skinner, F.K. Exploring gap junction location and density in electrically coupled hippocampal oriens interneurons. *Neurocomputing* 69:1048-1052, 2006.

Scorcioni, R., Lazarewicz, M.T., and Ascoli, G.A. Quantitative morphometry of hippocampal pyramidal cells: differences between anatomical classes and reconstructing laboratories. *J. Comp. Neurol.* 473:177-193, 2004.

Sheasby, B.W. and Fohlmeister, J.F. Impulse encoding across the dendritic morphologies of retinal ganglion cells. *J. Neurophysiol.* 81:1685-1698, 1999.

Sikora, M.A., Gottesman, J., and Miller, R.F. A computational model of the ribbon synapse. *J. Neurosci. Meth.* 145:47-61, 2005.

Sohal, V.S. and Huguenard, J.R. Inhibitory interconnections control burst pattern and emergent network synchrony in reticular thalamus. *J. Neurosci.* 23:8978-8988, 2003.

Sohal, V.S. and Huguenard, J.R. Inhibitory coupling specifically generates emergent gamma oscillations in diverse cell types. *Proceedings of the National Academy of Sciences of the United States of America* 102:18638-18643, 2005.

Somjen, G.C. *Ions in the Brain*. New York, NY: Oxford University Press, 2004.

Somjen, G.G. Mechanisms of spreading depression and hypoxic spreading depression-like depolarization. *Physiol. Rev.* 81:1065-1096, 2001.

Spampanato, J., Aradi, I., Soltesz, I., and Goldin, A.L. Increased neuronal firing in computer simulations of sodium channel mutations that cause generalized epilepsy with febrile seizures plus. *J. Neurophysiol.* 91:2040-2050, 2004a.

Spampanato, J., Kearney, J.A., de Haan, G., McEwen, D.P., Escayg, A., Aradi, I., MacDonald, B.T., Levin, S.I., Soltesz, I., Benna, P., Montalenti, E., Isom, L.L., Goldin, A.L., and Meisler, M.H. A novel epilepsy mutation in the sodium channel SCN1A identifies a cytoplasmic domain for beta subunit interaction. *J. Neurosci.* 24:10022-10034, 2004b.

Tsodyks, M., Pawelzik, K., and Markram, H. Neural networks with dynamic synapses. *Neural Computation* 10:821-835, 1998.

Varela, J.A., Sen, K., Gibson, J., Fost, J., Abbott, L.F., and Nelson, S.B. A quantitative description of short-term plasticity at excitatory synapses in layer 2/3 of rat primary visual cortex. *J. Neurosci.* 17:7926-7940, 1997.

Vetter, P., Roth, A., and Häusser, M. Propagation of action potentials in dendrites depends on dendritic morphology. *J. Neurophysiol.* 85:926-937, 2001.

Vida, I., Bartos, M., and Jonas, P. Shunting inhibition improves robustness of gamma oscillations in hippocampal interneuron networks by homogenizing firing rates. *Neuron* 49:107-117, 2006.

Vitko, I., Chen, Y.C., Arias, J.M., Shen, Y., Wu, X.R., and Perez-Reyes, E. Functional characterization and neuronal modeling of the effects of childhood absence epilepsy variants of CACNA1H, a T-type calcium channel. *J. Neurosci.* 25:4844-4855, 2005.

Figure Legends

Figure 1. Top: Cartoon of a section. Dashed lines indicate boundaries between segments for $n_{seg} = 3$, and black dots mark the internal nodes (points at which membrane potential and current are computed by numerical integration of the discretized cable equation) which are located at normalized distances of $1/6$, $1/2$, and $5/6$. Bottom: Equivalent circuit of the same section. The resistors represent the cytoplasmic resistance between adjacent nodes, and the capacitors are the total membrane capacitance of each segment. For the sake of clarity, a box is used to represent the parallel combination of each segment's ion channels (or, if you prefer, the total ionic conductances with their associated equilibrium potentials); the current through each box is that segment's net ionic membrane current.

Figure 2. Use of the CellBuilder to specify that g_{na_hh} in the apical dendrites decreases linearly with distance from the origin of the apical tree. In the diagram of the cell, the boundaries between red and black mark the locations that correspond to the displayed value of p . In this figure p is 0.5 , which corresponds to locations that are half way from the origin of the apical tree to the most distal dendritic termination.

Figure 3. A Channel Builder implementation of a three-state kinetic scheme approximation to the Hodgkin-Huxley potassium conductance, with two closed states (C1 and C2) and one open state (O). The small window is actually the Channel Builder's main control panel, where the name of the mechanism and elementary properties such as ion selectivity are specified. The large window is used to specify states, the transitions between them, and the equations and parameters that govern these transitions. Here it displays the voltage dependence of the C1-C2 transition in terms of the steady state $C2/C1$ ratio ($infC1C2$) and time constant for equilibration ($tauC1C2$).

Figure 4. Studying the effect of patch electrode capacitance and resistance on stimulation and recording from a current clamped cell. This example uses a very simple model cell that has just one section called `soma`, with `diam = L = 30 um`, `nseg = 1`, and Hodgkin-Huxley membrane properties. In the Linear Circuit Builder's diagram, the stick figure cartoon labeled `soma(0.5)` is a direct connection to the internal node at the 0.5 location of `soma`. The current clamp is implemented as a current source labeled `Iclamp`, and the electrode's distributed resistance and capacitance are represented by an equivalent T circuit ($Re1 = Re2 = 5$ megohms, $Ce = 0.04$ nF, so total series resistance is 10 megohms and electrode time constant is 0.1 ms). V_m is the membrane potential at the 0.5 location of `soma`. The observed potential is V_e , which shows resistive artifact during the current pulse as well as noticeable delay and attenuation of the observed spike.

Figure 1

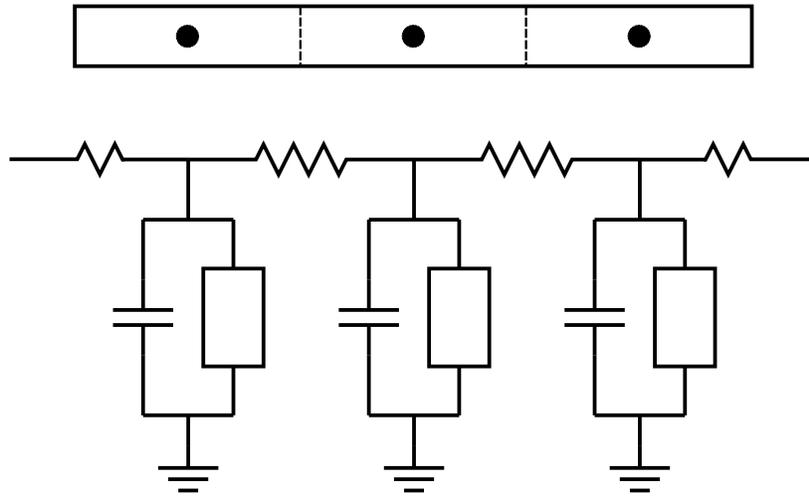


Figure 2

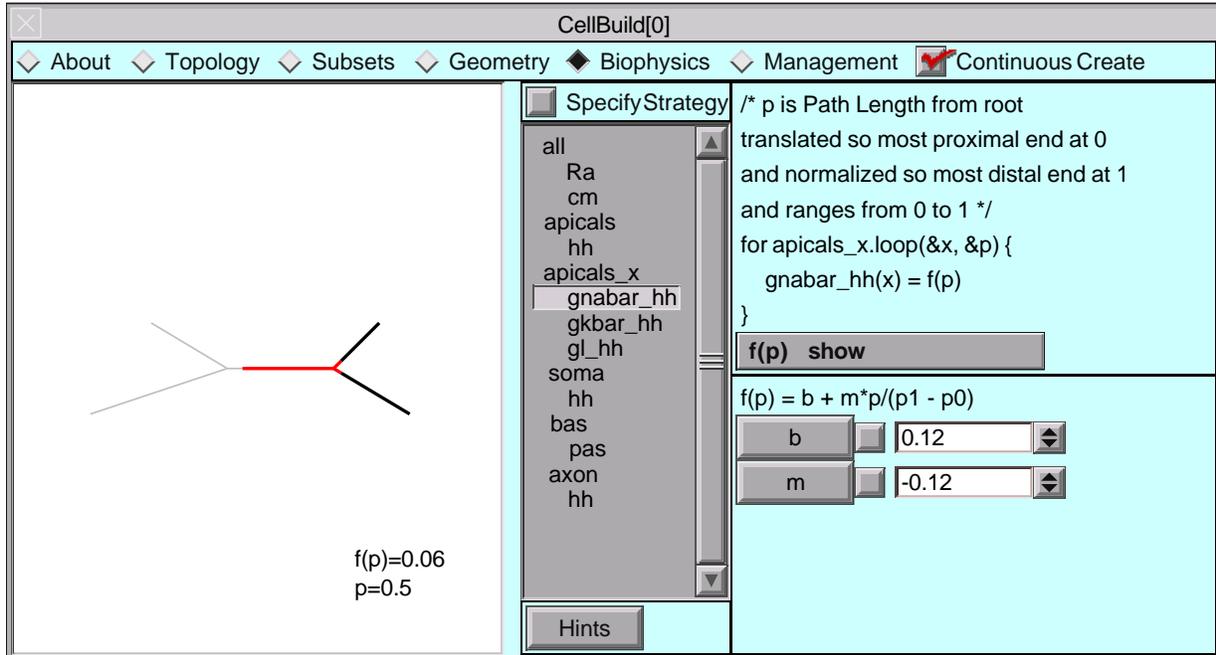


Figure 3

ChannelBuild[0]managedKSChan

Properties

mykhh Density Mechanism
 k ohmic ion current
 $ik \text{ (mA/cm}^2\text{)} = g_{\text{mykhh}} * (v - ek)$
 $g = g_{\text{max}} * O$
 Default gmax = 0.02979 (S/cm2)

O: 3 state, 2 transitions

ChannelBuildGateGUI[0]forChannelBuild[0]

States Transitions Properties

Select hh state or ks transition to change properties

$C1 \xrightarrow{v} C2 \xleftarrow{v} O$

O: 3 state, 2 transitions

Power

Fractional Conductance

C1 fraction

C2 fraction

Adjust Run

infC1C2
tauC1C2

C1 <-> C2 (inf, tau) (KSTrans[0])

Display inf, tau

$infC1C2 = A / (1 + \exp(-k * (d - v)))$

A (1)
 k (/mV)
 d (mV)

$tauC1C2 = A * \exp(k * (v - d))$

A (ms)
 k (/mV)
 d (mV)

EquationType

Figure 4

