A simple program for simulating the responses of neurons with arbitrarily structured and active dendritic trees

Yidao Cai, Edward J. Walsh and JoAnn McGee

Developmental Auditory Physiology Laboratory, Boys Town
National Research Hospital, Omaha, Nebraska 68131, USA

Manuscript # 96 / 3346M
Key words: program, compartment model, dendritic tree, active

Correspondence to:
Yidao Cai
Boys Town National Research Hospital
555 North 30th Street
Omaha, NE 68131
USA
TEL: 402-498-6526
FAX: 402-498-6351
Email: caiy@boystown.org

Abstract

We describe a simple program to simulate neural responses. This program is based on the compartmental approach, in which all compartments of a neuron (i.e., axon, soma, or dendrite) are represented by the same basic electrical structure. A parameter file is used to store the model parameters, including the nonlinear channel characteristics of each compartment. The model is then automatically configured according to the values specified in the parameter file. The computation of the conductance of each active channel over time is handled by a unique subroutine optimized according to the kinetics of each channel. The equations for arbitrarily structured trees are solved implicitly using a simple algorithm similar to that of Hines (1984). The output of the model uses PostScript format. The advantage of this program is that it is small in size, simple to use, efficient, and is platform independent.

Key words: program, compartment model, dendritic tree, active

Introduction

The dendritic tree plays an important role in the information acquisition and processing of a neuron. Rall and co-workers have studied dendrites as cylindrical cables and established the cable theory which formed the basis of our understanding of the computational capacity of neurons (for review, see Rall, 1959, 1989; Jack et al., 1983). During the last few decades, different mathematical methods have been developed to compute the temporal responses generated by an input (e.g. a synaptic input) to the dendritic tree of a neuron. Among them are the compartmental approach, in which each segment of the neural structure is modeled as a compartment (see Segev et al., 1989 for a review); the analytical approach, which usually involves complex time-frequency domain transformations and is aimed at giving analytical solutions (e.g. Horwitz, 1981); the equivalent cylinder approach, which reduces the complex dendritic tree to a single equivalent cylinder (Rall, 1962); and the moment-based approach, which extends the steady-state passive cable theory to the transient case (Agmon-Snir and Segev, 1993; Agmon-Snir, 1995).

Two issues are becoming increasing important in the simulation of the responses of a neuron. The first is the ability to deal with the arbitrariness of the dendritic tree. A dendritic tree can have varying degrees of complexity depending on the function of a neuron. A good simulation program, therefore, must be able to handle arbitrarily structured dendritic trees. The second issue is the ability to handle active dendritic properties. Although traditionally the dendrite was viewed as passive, this view is generally *not* true (see Koch, 1984 for a review). For example, fast, low-threshold Na^+ channels have been found in dendrites of hippocampal neurons (Wong et al., 1979), and dendritic K^+ channels have been shown to underlie the two-state spontaneous membrane potential fluctuations of neostriatal spiny neurons (Wilson, 1995; Wilson and Kawaguchi, 1996). In addition, dendritic calcium dynamics are thought to play an important role in synaptic plasticity and in shaping the electrophysiological properties of neurons (for

review, see Regehr and Tank, 1994). It has been demonstrated through modeling studies that an active dendrite can participate in sub-millisecond coincidence detection (Softky, 1994) and in the dynamic modification of dendritic cable properties and synaptic transmission (Wilson, 1995). The possible functional role of dendritic spines has also received attention (for review, see Wickens, 1988 and Holmes and Rall, 1995). Thus, inclusion of active dendritic properties in neural models would more accurately represent dendritic processing.

Theoretically, the modeling approaches mentioned earlier are able to deal with arbitrarily structured dendritic trees, although they differ widely in efficiency, especially when dealing with complex dendritic trees. However, all except the compartmental approach are based on the assumption that the dendrite is *passive* and, thus, many available models are unable to handle active dendrites. Attempts have been made to include active dendritic properties through linearization (Koch, 1984), but this is obviously not as direct as that in the compartmental approach in which exact kinetics can be included. The compartmental approach also has an obvious advantage over other approaches in that it is easier to model physically realistic neurons. Furthermore, the algorithm for solving branched equations developed by Hines (1984) allows efficient treatment of arbitrarily structured dendritic trees in compartmental modeling. These factors contributed to the widespread use of the compartmental approach in neuronal modeling.

There has been a number of compartment-based software packages available (e.g., GENESIS, NEURON and NODUS, see De Schutter, 1992). These packages, while differing in complexity and functionality, usually require large system resources and can only be run on a specific platform or necessitate platform-dependent installations. In this paper, we describe a simple program which is based on the compartmental approach and is able to deal with arbitrarily structured and active dendritic trees. The program uses a text file to specify the model parameters and automatically configures the model. A simple algorithm, inspired by that of Hines (1984), is used to solve the branched equations

implicitly. The program is small in size (the executive program is under 100 Kb), easy to use, efficient and can be run on a number of platforms.

Theoretical background

A model neuron is schematized in Fig. 1A. This neuron has a soma, an axon, and a simple yet arbitrarily structured dendritic tree. In a compartmental model, the neuron is divided into compartments (also referred to as nodes), each representing one segment of the neural structure. Fig. 1B shows the compartmental representation of the neuron shown in Fig. 1A. The soma and the axon are each represented by, for simplicity, a single compartment, although this may not necessarily be the case. The dendritic tree is represented by multiple compartments. The compartments are interconnected with serial resistors. Each compartment has a structure similar to that shown in Fig. 1C. It contains a branch for membrane capacitance (C_m), a branch with a conductance (g_{leak}) and a battery (E_{leak}), representing the leakage channel. Other branches in Fig. 1C have variable conductances (g_i). They are used to represent active channels (e.g., Na^+ and K^+ channels) and/or excitatory/inhibitory synaptic inputs, and may not be present in all compartments. In the case of active channels, E_i is the equilibrium potential of the particular channel; otherwise E_i is the battery that, together with the membrane voltage, determines the driving force of the excitatory or inhibitory synaptic input. Notice that the compartments, whether they represent the soma, the axon, or the dendrite, all have similar structures. This allows for a uniform treatment of the compartments.

Figure 2 shows the electrical representation of a "typical" compartment and its connections with neighboring compartments. In this representation, the most important parameter describing the status of the compartment (the state variable) is the voltage at each node (i.e. the membrane voltage of the compartment V_m). Apart from the branches representing its own structure, the compartment (labeled "self" in Fig. 2) has several input branches, each with a conductance ($g_{sin\,j}$) connecting it to several input compartments with nodal voltages $V_{in\,j}$; and several output branches, each with a conductance ($g_{sout\,k}$)

connecting it to several output compartments with nodal voltages $V_{out\,k}$. The nodal structure shown in Fig. 2 is a universal representation of the compartments for different parts of a neuron.

In the following, we find the solution for the voltage of a compartment. From basic circuit theory, it is easy to see that

$$I_{in} = I_{self} + I_{out} \qquad (1)$$

where

$$I_{in} = \sum (V_{in\,j} - V_m)\, g_{sin\,j} \qquad (2)$$

$$I_{out} = \sum (V_m - V_{out\,k})\, g_{sout\,k} \qquad (3)$$

$$I_{self} = \sum (V_m - E_i)\, g_i + C_m\, dV_m / dt \qquad (4).$$

I_{in} may also include a term for injected current. These equations can be reduced to

$$C_m\, dV_m / dt + P\, V_m = Q \qquad (5)$$

where

$$P = \sum g_{sin\,j} + \sum g_{sout\,k} + \sum g_i$$

$$Q = \sum V_{in\,j}\, g_{sin\,j} + \sum V_{out\,k}\, g_{sout\,k} + \sum E_i\, g_i\ .$$

Eq. 5 forms the basis of the solution of the voltage response of a compartment.

By writing one equation for each compartment, we obtain a set of nonlinear differential equations. Notice that all the battery-conductance (E_i-g_i) branches in Fig. 2, whether they represent the leakage channel, active channels or synaptic inputs, are treated uniformly in the differential equations. This is possible due to the conditional linearity of these equations, i.e., given the nodal (membrane) voltages, the rate equations used to calculate the channel conductances (e.g., eq. A2) are linear; given the channel conductances, the equations describing the nodal voltages are linear (Mascagni, 1989). The same property also allows us to solve the nonlinear equations as linear equations. Since the number of equations is equal to the number of compartments and thus the number of unknowns (nodal/compartment voltages), and since those equations are independent, we can solve the equations numerically (see Implementation).

Implementation

The model based on this theory was implemented using high-level computer language C. Since we did not use any platform dependent interface, the same program is able to run under a number of platforms (e.g., DOS, unix, VMS) without modification.

Node Identification and Information Storage

A key issue in dealing with an arbitrarily structured dendritic tree is how to efficiently locate and identify a compartment within the tree, for both the user and the computer. This issue becomes more important when the dendritic tree is large and complex: it is almost impossible to locate one compartment in a dendritic tree which involves hundreds or even thousands of compartments. We designed a labeling system to identify each compartment uniquely (cf. Fig. 1B). In this system, each label has one layer number and one or more optional branch numbers, separated by dashes. For example, in a label 3-0-1, the first number (3) indicates the compartment is at layer 3, the second number (0) is the main branch number and the third number (1) is the minor branch number. This labeling system makes it easy to locate a specific compartment, especially when dealing with a large number of compartments.

We designated a "storage structure" to hold the parameters for each compartment. The structures are then linked together using a two-way linked list. The linked list and the labeling system enable us to move within the "dendritic tree" with ease.

The Parameter File

The model parameters are stored in a simple text file, which can be edited using a text editor. The parameter file is used by the program to automatically configure the model. This automatic configuration allows us to simulate neurons with dendritic trees of arbitrary geometry (as specified in the parameter file). It also allows us to simulate

different neurons without changing the program. Thus it is versatile and easy to use by users without knowledge of computer programming.

The basic information about a compartment that is implemented in the model is membrane area (or, alternatively, diameter and length), leakage conductance (g_{leak}) and leakage equilibrium potential (E_{leak}). The membrane capacitance can be calculated as a product of the membrane area and membrane capacitance density (which is usually 1 $\mu F/cm^2$, see Jack et al., 1983, and which can be specified in the parameter file). Both the leakage conductance and the leakage equilibrium potential can be designated for passive compartments, but only one of them is specified in case of active compartments (the other one is determined by the program to maintain a stable resting membrane potential). The serial conductance between the current node and its parent node can be assigned directly or determined by the program if the diameters and lengths of the compartments are provided. A typical passive compartment can be specified in the following way (in this and subsequent examples, anything after the character "#" is comment and is ignored by the program):

```
node 2-1 DENDRITE                    # a dendritic compartment
      surface area          7.6e-6   # cm2
      serial conductance    6.1e-7   # S
      Eleak                 -62.     # mV
      gleak                 7.6e-9   # S
End node
```

For active compartments, one or more types of active (nonlinear) channels must be specified. Since the number of types of active channels is limited, the approach was pre-defining them. This approach has the advantage that the subroutines handling the active channels can be optimized in computation time and accuracy according to the kinetics of each channel. A general mechanism of solving arbitrary functions such as those needed to handle the voltage-dependent channels is no doubt much less efficient. Each nonlinear conductance is handled by one subroutine customized for that channel type. The parameters defining each active channel of each compartment are entered in the parameter file, placed between the "node ..." and "End node" lines. For example, the following

describes a low-threshold, voltage-dependent K^+ channel for the soma compartment, as described in Appendix A.

```
start chan gKLT
      gKLTmax          .0025          # S/cm2
      EKLT             -77.           # mV, reversal potential
      Bsf              10             # mV, voltage shift
      Bfac             .7             # temperature factor
      alpha0_beta0     .03 .042       # coefficients
      Kalpha_Kbeta     7. 15.         # mV, coefficients
end chan gKLT
```

The complexity of the channel description is determined by the flexibility required. In the above example, almost all of the adjustable values required to describe the channel characteristics are included. The parameters of the channels for each compartment are stored in the storage structure designated for that compartment and the values of channel conductance and other variables (e.g., B, α_B and β_B in this example, see Appendix A) are updated at each step.

A synaptic input is in principle treated in the same way as an active channel is treated, i.e., the conductance change due to a synaptic input is updated at each step according to synaptic dynamics. In our preliminary implementation, we only included two types of synaptic inputs, one for excitatory and the other for inhibitory, each having dynamics of an α-function (Jack et al., 1983). Adding synaptic inputs with different dynamics or parameters is trivial, but we found two types enough for our present purpose. We are studying neurons in the auditory cochlear nuclei, which receive their primary inputs from auditory-nerve fibers. The input spike trains (outputs of auditory-nerve fibers) are stored in separate files, which are text files with numbers each representing the time of a spike event. The spike trains are read by the model program at run time and assigned to the synapses, and the synaptic conductances are updated according to the input spike trains and the dynamics of the synapses. Each compartment can accept multiple synaptic inputs (any combination of excitatory or inhibitory inputs).

To simplify the parameter file, by default, a node (compartment) has the same passive parameters as its parent node. For example, the following specifies that node 3-1-0 has the same passive parameters as its parent node (2-1), except for the serial conductance which has been included in the description for this node.

```
node 3-1-0 DENDRITE
      serial conductance    5.0e-7  # S
End node
```

This rule is very useful when the number of compartments becomes large. In that case, the simplicity brought about by this rule and the potential reduction of the size of the parameter file become obvious. Optionally, the description of the active channels for a node can also be "copied" from its parent node.

In some cases, a particular structural pattern (a branch or part of a dendritic tree) which involves several compartments may appear at more than one location in a model, having identical or similar parameters at different locations. In such cases, the parameters for that pattern can be specified only once in the parameter file. In subsequent appearances of the same pattern, the structural pattern and the parameters can then be simply "copied" over. For example, the following copies the structural pattern and the parameters of a branch headed by node 2-1 (cf. Fig. 1B).

```
node 4-0-2        # by default it is a dendritic node
      copytree 2-1
end node
```

The parameters for node 2-1 will be copied to node 4-0-2 (notice that node 4-0-2 and the associated branch are not shown in Fig. 1B). In addition, two new nodes (nodes 5-0-2-0 and 5-0-2-1, which do not need to be explicitly specified in the parameter file) will be created and will be assigned the same parameters as their corresponding nodes (nodes 3-1-0 and 3-1-1) in the branch headed by node 2-1. If the parameters of a node in the new branch are not exactly the same as those of the corresponding node in the old branch,

minor modifications may be made according to the usual rules. Again, one may optionally specify whether to copy parameters of active channels. This feature of copying the structural pattern and the associated parameters further reduces the task for constructing parameter files which have the same patterns, as might occur for a model with dendritic spines.

Numerical solutions

We have implemented several methods for solving the system of equations: the forward Euler, backward Euler, the 2nd-order and 4th-order Runge-Kutta methods (cf. Nakamura, 1991), and a Crank-Nicholson method modified by Hines (1984). In our program, the Runge-Kutta methods were used only for systems with one compartment since they suffer from instability for systems with multiple compartments.

The forward Euler is an explicit method. Its main advantage is simplicity. The difference approximation of eq. 5 using forward Euler is

$$C_m (V_{m\,t+Dt} - V_{m\,t}) / Dt + P (V_{m\,t+Dt} - V_{m\,t}) = Q \qquad (6).$$

At any time, t + Dt, the values of either $V_{in\,j}$ or $V_{out\,k}$ in the expression of Q are known, and the values of the unknown ones at time, t + Dt, can be substituted with their values at time t. But forward Euler is only first-order correct in time. Also, due to constraints on the resolutions in time and in space for forward Euler and the potential instability, it is not possible to increase the temporal accuracy simply by decreasing the step size (Mascagni, 1989).

Implicit methods, however, do not suffer from these constraints. Therefore, we implemented the backward Euler and the modified Crank-Nicholson method (Hines, 1984). The former is first-order correct in time while the latter is second-order correct in time. Both are numerically stable and appropriate for stiff systems.

With an implicit method, the system of equations could not be solved explicitly. For a cable without branching, the system is tridiagonal (cf. Appendix B) and can be solved implicitly with standard forward elimination and backward substitution for tridiagonal systems (e.g., Nakamura, 1991). Hines (1984) developed a method of solving a system of branched equations, which can be used in the case of arbitrarily structured dendritic trees. The method is based on upper triangularization and back substitution, and is, theoretically, as efficient as the method of solving a tridiagonal system for an unbranched cable. One drawback of Hines' method is that it requires non-sequential and different orderings for upper triangularization and back substitution, which creates difficulty for implementation.

Inspired by Hines' method, we developed a simple way of solving equations for arbitrarily structured trees. It is also as efficient as the method of solving a tridiagonal system for an unbranched cable. However, since it uses a simple and natural ordering for both upper triangularization and back substitution processes, it is easier to implement. The details of the algorithm are given in Appendix B.

Currently, we use fixed step sizes to solve the system of equations, mainly because the model runs fast enough for our present applications.

Model output

The output of the model uses PostScript format (*), which can be viewed by PostScript viewers available on almost all platforms or printed on PostScript printers. The graphics (in PostScript format) can also be included into other documents (e.g., Word, LaTex). The program also provides outputs of key variables in text format.

Memory requirement

One of the advantages of our program is that it requires relatively small system resources. The executive program is small (currently 91 Kb for DOS, 82 Kb for HP unix). The basic

memory requirement is 244 bytes per compartment. Extra storage space is allocated if active channels are included for a particular compartment. The storage for active channels depends on the complexity of the description. For example, it is 44 bytes each for the low-threshold K^+ channel described in Appendix A. Most other channel types (except the Na^+ channel) use less storage space. Since the program only stores results that are requested, and it generally resamples the output by a factor of 4 to 16, depending on the step size used in simulation (the resolution after resampling is ordinarily still higher than that of the output device), there is typically not much additional memory requirement for the storage of simulation results. Thus, to simulate a model with 1000 compartments, each with 4 types of active channels, about 520 Kb of memory is enough.

Application

The preliminary model was used to simulate responses of neurons in the auditory cochlear nuclei. Currently we are studying stellate and octopus cells. Octopus cells exhibit transitory or phasic response characteristics; i.e., in response to a stimulus, they generate highly synchronized responses at the stimulus onset (e.g. Feng et al., 1994). Octopus cells have relatively large cell bodies and thick dendrites which taper only slightly (Kane, 1973). Even at the distal end, the dendrite measures about 5 μm in diameter. In our model, the octopus cell has a soma with a diameter of 32 μm and is simulated with one compartment. We included four dendrites, each of which is 200 μm in length and each has a diameter of 7 μm with no tapering. Each dendrite is subdivided into 10 compartments. The axon is simulated as a single compartment. In the axon compartment, we included Hodgkin-Huxley type Na^+ and K^+ channels. For the soma, in addition to the Na^+ and K^+ channels, we also included a Cs^+-sensitive, hyperpolarization-activated inward rectifier (I_h), which is responsible for the low input resistance of this neuronal type, and a 4-AP sensitive, low-threshold K^+ channel (Golding et al., 1995). The two proximal dendritic stages of each dendrite have active channels similar to those in the soma, and the remaining dendritic stages are passive.

With these model parameters, it takes about 9 seconds to simulate 80 ms of response on a 486DX2-50 under DOS, using the modified Crank-Nicholson method and a step size of 20 μs. Work is under way to use this model to simulate responses of neurons with more complex dendritic structures in the cochlear nuclei.

The responses of the model neuron to current injection are shown in Fig. 3. There is one action potential at the onset of the stimulus and then steady-state depolarization during the reminder of the current injection. The soma spike is small, reaching only about -35 mV, whereas the axon spike is large. These characteristics are similar to those observed experimentally (Feng et al., 1994; Golding et al., 1995).

Discussion

Among the different methods developed to simulate neural responses (see Introduction), the compartmental approach is unique in that it places no restriction on the membrane properties of each compartment and is able to handle nonlinear, active elements (Segev et al., 1989). In this paper, we have presented a simple compartment-based program which is able to deal with arbitrarily structured and active dendritic trees. Its implementation is both efficient and flexible.

Simulation programs that are designed for more general purposes (e.g., SPICE) usually require large system resources and are slow and cumbersome in dealing with nonlinear channel conductances (Segev et al., 1989). These programs have gradually phased out. On the other hand, specific, individually tailored implementations are usually fast, but lack flexibility. Most recently developed programs, including ours, fall somewhere in between. They are usually much faster than the general-purpose simulation programs, yet still flexible for general use.

As discussed in the Introduction, the main advantage of our program, as compared to existing simulation packages (e.g., GENESIS, Bower and Beeman, 1994; NEURON,

Hines, 1989, 1993; and NODUS, De Schutter, 1989), is its small size and platform independence. It is not as comprehensive as large packages, and currently can only simulate a single neuron at a time, although it will probably be extended to handle neuronal networks in the future.

In our design, we did not use a platform-specific interface. This not only kept the program small but also allows the program to compile and run on a variety of platforms without modifying the source code. Additionally, the use of predefined channel types allows the user to easily and precisely specify the dynamics of active channels. The specialized subroutines for handling the active channels reduce the overhead in computation and program size, and make the program more efficient. Although it is almost inevitable that the program will be enhanced to accommodate increasing needs (e.g., to support more channel types), it is unlikely that the executive program will double its present size (which is under 100 Kb), nor will the required storage space per compartment increase greatly. One consequence of the small memory requirement is that the program can be used to simulate systems with a relatively large number of compartments on computers with limited resources.

The use of a text file to specify model parameters shifts the burden of defining the model to the user. While parameter input may be accomplished directly through the use of a text editor, users may also write their own programs to automatically generate the parameter file from morphological data. In addition to the ease of simulating different neurons and/or conditions without changing the program, our design also makes it relatively easy to add new types of active channels. However, since adding a new channel requires the addition of subroutines for handling the new channel type in the program, and since the speed and numerical accuracy depend on the implementation of the subroutine, it cannot be done without knowledge of computer programming.

Appendix A Kinetics of the low-threshold K^+ channel

The kinetics of the low-threshold, 4-AP sensitive K^+ channel, modified from Manis and Marx (1991), is described by:

$$gK_{LT}(V, t) = gK_{LTmax} * B(V, t) \quad \text{(A1)}$$

$$\tau_B \, dB(V, t) / dt + B(V, t) = B_\infty \quad \text{(A2)}$$

where

$$\tau_B = B_{fac} / (\alpha_B(V) + \beta_B(V)) \quad \text{(A3)}$$

$$B_\infty = \alpha_B(V) / (\alpha_B(V) + \beta_B(V)) \quad \text{(A4)}$$

and

$$\alpha_B(V) = \alpha_B(0) * \exp((V - V_{1/2} - B_{SF}) / K_\alpha) \quad \text{(A5)}$$

$$\beta_B(V) = \beta_B(0) * \exp(- (V - V_{1/2} - B_{SF}) / K_\beta) \quad \text{(A6)}$$

B_{fac} is the temperature factor for the time constant τ_B and $V_{1/2}$ is the half activation voltage.

Appendix B Algorithm for solving a system of equations for arbitrarily structured trees

Node numbering

A simple branched tree is shown in Figure B1 to help illustrate the algorithm. First, we emphasize that this node numbering scheme is for solving the linear system of equations *only*. It is different from the numbering system which we used to identify nodes (compartments) for both the human and the computer. It is purely internal, i.e., only the computer knows it and uses it, without human intervention. The numbering system is natural. The rules are:

1). The numbering starts from any node, but preferably (for reasons of topological simplicity) a node that is connected to only one other node. We call this node the "root node".

2). When the current node is connected to only one unnumbered node, number that node sequentially.

3). When branching is encountered (i.e. the node is connected to more than one unnumbered node), number all the nodes in one branch first (using rule 4), then all the nodes in the next branch, and so on.

4). For each branch, treat the node from which the branch originates as the "root node" and apply rules 2 to 4 recursively until all nodes are numbered.

Figure B1 illustrates a simple branched tree and the numbering of nodes using the above rules.

The linear system of equations and the matrix form

Obviously, the membrane voltages (V_m) are the state variables and the basic equation governing them is eq. 5. Upon transformation into difference equation using backward approximation, eq. 5 takes the form of

$$... + A_j y_{j+1} + B_j y_j + C_j y_{j-1} + ... = D_j \qquad \text{(B1)}$$

where y_j are the membrane voltages and A_j to D_j are constants resulting from the internodal resistors and the approximation process.

The matrix form of the linear system of equations for the example shown in Fig. B1 is

$$
\begin{array}{llllllllllll}
B_{11} & 0 & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & U_{11} & & & \\
0 & B_{10} & C_{10} & & & & & & & & & \\
 & A_9 & B_9 & 0 & \ldots & \ldots & \ldots & \ldots & U_9 & & & \\
 & & 0 & B_8 & 0 & U_8 & & & & & & \\
 & & & 0 & B_7 & C_7 & & & & & & \\
 & & & L_8 & A_6 & B_6 & C_6 & & & & & \\
 & & & & & A_5 & B_5 & C_5 & & & & \\
 & & & & & & A_4 & B_4 & C_4 & & & \\
L_{11} & \ldots & L_9 & \ldots & \ldots & \ldots & \ldots & A_3 & B_3 & C_3 & & \\
 & & & & & & & & A_2 & B_2 & C_2 & \\
 & & & & & & & & & A_1 & B_1 &
\end{array}
\quad
\begin{array}{c}
y_{11} \\ y_{10} \\ y_9 \\ y_8 \\ y_7 \\ y_6 \\ y_5 \\ y_4 \\ y_3 \\ y_2 \\ y_1
\end{array}
\quad = \quad
\begin{array}{c}
D_{11} \\ D_{10} \\ D_9 \\ D_8 \\ D_7 \\ D_6 \\ D_5 \\ D_4 \\ D_3 \\ D_2 \\ D_1
\end{array}
\; .
$$

Notice that the state variables (y_j) are arranged in reversed order (i.e. from 11 to 1). This is required by the upper triangularization (forward elimination) process, as will be explained later.

Solving the matrix equation

As can be seen, the coefficient matrix is nearly tridiagonal, except for the coefficients, L_j and U_k (L_8, L_9, L_{11} and U_8, U_9, U_{11} in this example). For the sole purpose of solving the matrix equation, we view the coefficients below the diagonal line (formed by B_j) in the matrix as input coefficients (A_j and L_j), and those above the diagonal line as output coefficients (C_k and U_k). We can then follow the rule of solving the usual tridiagonal matrix equations (e.g., Nakamura, 1991) to solve our matrix equation.

In the tridiagonal case, the forward elimination process is carried out by (take into account of our reversed order):

$$R = A_j / B_{j+1}$$

$$B_j = B_j - R*C_{j+1}$$

$$D_j = D_j - R*D_{j+1} \; .$$

These steps eliminate coefficients A_j. The backward substitution process is

$$y_j = (D_j - C_j * y_{j-1}) / B_j .$$

In our case, the coefficients A_j can be eliminated in a manner like that in the standard tridiagonal algorithm during the forward elimination (upper triangularization) process. The difference is with coefficients L_j and U_k. For every $U_k \neq 0$, we have $C_k = 0$ and $A_{k-1} = 0$. Thus coefficient U_k will not be "carried over" to the next step (k -1) after the forward elimination process which eliminates A_k (i.e., it will not create a $U_{k-1} \neq 0$). As a result, no action needs to be taken about U_k in this process. The coefficients, L_j, need some attention, but no more than one or more usual elimination steps, since the only output coefficient of those nodes that provided input coefficients to the node of concern is the node of concern. Take the example of node 3 in Fig. B1, it has two additional inputs from nodes 9 and 11, except for the usual input from node 4. But both node 9 and node 11 output only to node 3 (cf. the matrix). Thus, elimination of coefficients L_9 and L_{11} will not introduce additional coefficients for node 3 in the upper triangle. This would not hold if we arranged the state variables (y_j) in the order of 1 to 11. Thus, the whole forward elimination process does not create any new elements in the upper triangle of the matrix, which is the key of this algorithm. The back substitution process is equally easy, except that U_n and the associated y_m instead of C_j (which is zero) and y_{j-1} must be used for some nodes (e.g., node 9).

Several observations can be made about the algorithm just described. First, the algorithm is basically as efficient as the standard tridiagonal algorithm (which is the most efficient); it requires exactly the same number of multiplications and divisions, plus some extra logical operations. Second, in contrast to Hines' (1984) method, both the forward elimination (upper triangularization) and the back substitution processes are sequential (as in the standard tridiagonal algorithm), thus making implementation simple.

One final observation concerns the storage of the coefficients. Notice that for any U_j, there is a corresponding $C_j = 0$, and for any L_k, there is a corresponding $A_k = 0$. Thus U_j can take the storage of C_j and L_k can take the storage of A_k, and the storage required is

the same as the tridiagonal matrix equations. Furthermore, since $C_j = A_{j-1}$, we can eliminate one more array of storage. Thus we need only 2N immediate storage locations (N is the number of nodes). Of course, information about the connections between nodes must be provided in order to take the above advantages (in our implementation, such information is already available during the simulation).

One limitation of the above algorithm is that, as with Hines' (1984) algorithm, it cannot handle structures involving loops, either global or local. Thus, the input (or output) part in Fig. 2 contains only one branch in the current implementation. Structures involving loops can be handled by a technique called "domain decomposition" (for a brief introduction, see Mascagni, 1989). Simply put, the solution of every non-branching node can be expressed as a linear combination of the nearest branched nodes that it directly or indirectly connects to, thus the process of solving the original matrix equation involves solving a number of small tridiagonal matrix equations of unbranched nodes and one regular matrix equation with a dimension that is equal to the number of branching nodes. Although the theory is simple, the implementation is difficult. For structures with many branching points and no looping, it is not as efficient as the algorithm we have just described.

Acknowledgments

This work was supported by the National Institute on Deafness and Other Communication Disorders, US National Institutes of Health, grant number DC01007. Yidao Cai is supported (in part) by research grant number P60 DC00982-06 from the same institute. The authors wish to thank the reviewer for suggestions on improvement of both the manuscript and the software.

* PostScript is registered by Adobe, Inc. The PostScript header used in our simulation program was copied from a Chinese/Japanese/Korean printing software, CNPRINT (Copyright by Yidao Cai), and the PostScript interface was modified from a two-tone suppression model (Cai and Geisler, 1996).

Figure Captions

Fig. 1 A. A schematic representation of a neuron with an axon, a soma, and a dendritic tree with arbitrary geometry. The small tic marks indicate the boundaries of compartments. B. The compartmental representation of the neuron. The compartments (boxes) are interconnected with serial resistors. C. The structure of a typical compartment (a box in B), which includes the membrane capacitance, a leakage branch and optionally one or more branches with adjustable resistors representing the nonlinear conductances and/or synaptic inputs.

Fig. 2 The electrical representation of a compartment (node) and its associated input/output branches. The $g_{sin\,j}$ and $g_{sout\,k}$ are input and output serial conductances, respectively. $V_{in\,j}$ and $V_{out\,k}$ are the membrane voltages of input and output nodes.

Fig. 3 The responses of a model octopus cell to current injection. The 2.8-nA current was injected at the soma from 10 to 50 ms. Both the soma (low-amplitude) and the axon responses are shown. For clarity, only the first 60 ms of response are shown here (80 ms total).

Fig. B1 A simple branched structure. Each circle represents one compartment.

Reference

Agmon-Snir, H. (1995) A novel theoretical approach to the analysis of dendritic transients, Biophys. J., 69: 1633-1656.

Agmon-Snir, H. and Segev, I. (1993) Signal delay and input synchronization in passive dendritic structures, J. Neurophysiol., 70: 2066-2085.

Bower, J.M. and Beeman, D. (1994) The Book of Genesis: Exploring Realistic Neural Models With the GEneral NEural SImulation System, Springer-Verlag, New York.

Cai, Y. and Geisler, C.D. (1996) Suppression in auditory-nerve fibers of cats using low-side suppressors: III. Model results, Hear. Res., 96: 126-140.

De Schutter, E. (1989) Computer software for development and simulation of compartmental models of neurons. Comput. Biol. Med., 19: 71-81.

De Schutter, E. (1992) A consumer guide to neuronal modeling software. Trends Neurosci., 15: 462-464.

Golding, N.L., Robertson, D. and Oertel, D. (1995) Recordings from slices indicate that octopus cells of the cochlear nucleus detect coincident firing of auditory nerve fibers with temporal precision, J. Neurosci., 15: 3138-3153.

Feng, J.J., Kuwada, S., Ostapoff, E.-M., Batra, R. and Morest, D.K. (1994) A physiological and structural study of neuron types in the cochlear nucleus. I. Intracellular responses to acoustic stimulation and current injection, J. Comp. Neurol., 346: 1-18.

Hines, M. (1984). Efficient computation of branched nerve equations, Int. J. Biomed. Comput., 15: 69-76.

Hines, M. (1989). A program for simulation of nerve equations with branching geometries, Int. J. Biomed. Comput., 24: 55-68.

Hines, M. (1993). NEURON - a program for simulation of nerve equations. In F.H. Eeckman (Ed.), Analysis and Modeling of Neural Systems, V2, Kluwer Academic Publishers, Boston, pp. 127-136.

Holmes, W.R. and Rall, W. (1995) Dendritic spines. In M.A. Arbib (Ed.), The Handbook of Brain Theory and Neural Networks, The MIT Press, Cambridge, pp. 289-292.

Horwitz, B. (1981) An analytical method for investigating transient potentials in neurons with arbitrary dendritic trees, Biophys. J., 36: 155-192.

Jack, J.J.B., Noble, D. and Tsien, R.W. (1983) Electric Current Flow in Excitable Cells, Oxford University Press.

Kane, E.C. (1973). Octopus cells in the cochlear nuclei of the cat: heterotypic synapses upon homeotypic neurons, Int. J. Neurosci., 5: 251-279.

Koch, C. (1984) Cable theory in neurons with active, linearlized membranes, Biol. Cybern., 50: 15-33.

Manis, P.B. and Marx, S.O. (1991) Outward currents in isolated ventral cochlear nucleus neurons, J. Neurosci., 11: 2865-2880.

Mascagni, M.V. (1989) Numerical methods in neuronal modeling. In C. Koch and I. Segev (Eds.), Methods in Neuronal Modeling: From Synapses to Networks, MIT Press, Boston, pp. 439-487.

Nakamura, S. (1991) Applied Numerical Methods with Software, Prentice Hall, New Jersey, Chapters 1, 9 and 10.

Rall, W. (1959) Branching dendritic trees and motoneuron membrane resistivity, Exp. Neurol., 1: 491-527.

Rall, W. (1962) Theory of physiological properties of dendrites, Ann. N.Y. Acad. Sci., 96: 1071-1092.

Rall, W. (1989) Cable theory for dendritic neurons. In C. Koch and I. Segev (Eds.), Methods in Neuronal Modeling: From Synapses to Networks, MIT Press, Boston, pp. 9-62.

Regehr, W.G. and Tank, D.W. (1994) Dendritic calcium dynamics, Current Opinion Neurobiol., 4: 373-382.

Segev, I., Fleshman, J.W. and Burke, R.E. (1989) Compartmental models of complex neurons. In C. Koch and I. Segev (Eds.), Methods in Neuronal Modeling: From Synapses to Networks, MIT Press, Boston, pp. 63-96.

Softky, W. (1994) Sub-millisecond coincidence detection in active dendritic trees, Neurosci., 58: 13-41.

Wickens, J. (1988) Electrically coupled but chemically isolated synapses: Dendritic spines and calcium in a rule for synaptic modification, Prog. Neurobiol., 31: 507-528.

Wilson, C.J. (1995) Dynamic modification of dendritic cable properties and synaptic transmission by voltage-gated potassium channels, J. Comput. Neurosci., 2: 91-115.

Wilson, C.J. and Kawaguchi, Y. (1996) The origins of two-state spontaneous membrane potential fluctuations of neostriatal spiny neurons, J. Neurosci., 16: 2397-2410.

Wong, R.K.S., Prince, D.A. and Basbaum, A.I. (1979) Intradendritic recordings from hyppocampal neurons, Proc. Natl. Acad. Sci. USA, 76: 986-990.