Journal of Computational Neuroscience 8, 209–226 (2000)

Additional Efficient Computation of Branched Nerve Equations: Adaptive Time Step and Ideal Voltage Clamp

LYLE J. BORG-GRAHAM
Unité de Neurosciences Integratives et Computationelles, Institut Alfred Fessard - CNRS, 91198 Gif-sur-Yvette, France
lyle@cogni.iaf.cnrs-gif.fr

Received February 16, 1999; Revised May 28, 1999; Accepted June 9, 1999

Action Editor: Alexander Borst

Abstract. Various improvements are described for the simulation of biophysically and anatomically detailed compartmental models of single neurons and networks of neurons. These include adaptive time-step integration and a reordering of the circuit matrix to allow ideal voltage clamp of arbitrary nodes. We demonstrate how the adaptive time-step method can give equivalent accuracy as a fixed time-step method for typical current clamp simulation protocols, with about a 2.5 reduction in runtime. The ideal voltage clamp method is shown to be more stable than the nonideal case, in particular when used with the adaptive time-step method. Simulation results are presented using the Surf-Hippo Neuron Simulation System, a public domain object-oriented simulator written in Lisp.

Keywords: numerical methods, compartmental models, neuron simulation

1. Introduction

Compartmental approximations of biophysically detailed neuron models, where the branching cable structure of the cell approximated by a series of isopotential compartments interconnected with resistors are now a fundamental technique in computational neuroscience (Rall, 1964; Segev et al., 1998).

Many numerical methods spanning a wide range of complexity may be used to solve the resulting systems of partial differential equations (Mascagni and Sherman, 1998). However, the descriptions by Hines (1984) of $O(n)$ integration of tree circuit topologies, as naturally found in individual neurons, and midstep solving of time-dependent nonlinear elements (e.g., voltage-dependent gating particles) have been found to be quite effective despite their relative simplicity (Hines and Carnevale, 1995; implemented in the simulator package NEURON, Hines and Carnevale, 1997). These contributions by Hines have been crucial for the increasing application of biophysically and anatomically detailed compartmental neuron models.

In this article I present two straightforward improvements on the methods introduced by Hines. These include recasting the system equations to allow adaptive time-step integration and a reordering of the circuit matrix to allow ideal voltage clamp of arbitrary nodes.

Since analytical statements concerning the stability of adaptive time-step integration are lacking (Vlach and Singhal, 1983), I present several simulations demonstrating the stability and convergence of the presented methods, using a standard Hodgkin-Huxley axon model. Simulations are presented using the Surf-Hippo neuron simulation system (Borg-Graham, 1998; see Appendix). Example code in the Appendix demonstrates the object-oriented structure of this public-domain Lisp program.

2. Methods

2.1. Adaptive Time Step

We first review the solution of the circuit equation as described by Hines. The method, which uses alternate implicit and explicit integration steps and which is equivalent to the Crank-Nicolson method, at least for linear systems, is second-order correct in time and space and numerically stable.

Given (see Fig. 1):

- t_x, the time grid for the stored node voltages;
- $t_{(n-1)}$, t_n, $t_{(n+1)}$—the last time, current time, and prediction time, respectively;
- $\Delta t_n = t_{(n+1)} - t_n$, the current time step;
- t'_x, the staggered time grid for the midstep evaluation of particle states, inputs, and node voltages, where $t'_{(n+1)} = t_n + \Delta t_n/2$.

In the following discussion, the stored values of state variables at the nth time step will be notated with the subscript n, such as V_n $(= V(t_n))$ for voltages and x_n $(= x(t'_n))$ for gating particles.

Consider a compartmental circuit model with N nodes. The implicit phase of the solution at each time step $t_n \rightarrow t_{(n+1)}$ consists of solving the following matrix equation for the node voltage vector $\mathbf{V}(t'_{(n+1)})$ at the midstep, given the known voltages at the current time step, $\mathbf{V}_n$:

$$\overbrace{\left(\mathcal{G}\left(t'_{(n+1)}\right) - \left(2/\Delta t_n \times \mathbf{C}\mathcal{I}\right)\right)}^{\text{(Almost) Tridiagonal matrix}} \times \overbrace{\mathbf{V}\left(t'_{(n+1)}\right)}^{\text{Solve for}} = -2/\Delta t_n \times \mathbf{C}^T\mathbf{V}_n + \mathbf{GE}\left(t'_{(n+1)}\right) + \mathbf{I}\left(t'_{(n+1)}\right). \quad (1)$$

Where, for the node admittance matrix $\mathcal{G}$ (Desoer and Kuh, 1969),

$$G_{ij} = -g_{ij},\ i \neq j$$
$$= \sum_{k=1}^{N} g_{ik} + \sum g_{elt}\left(t'_{(n+1)}\right),\ i = j, \quad (2)$$

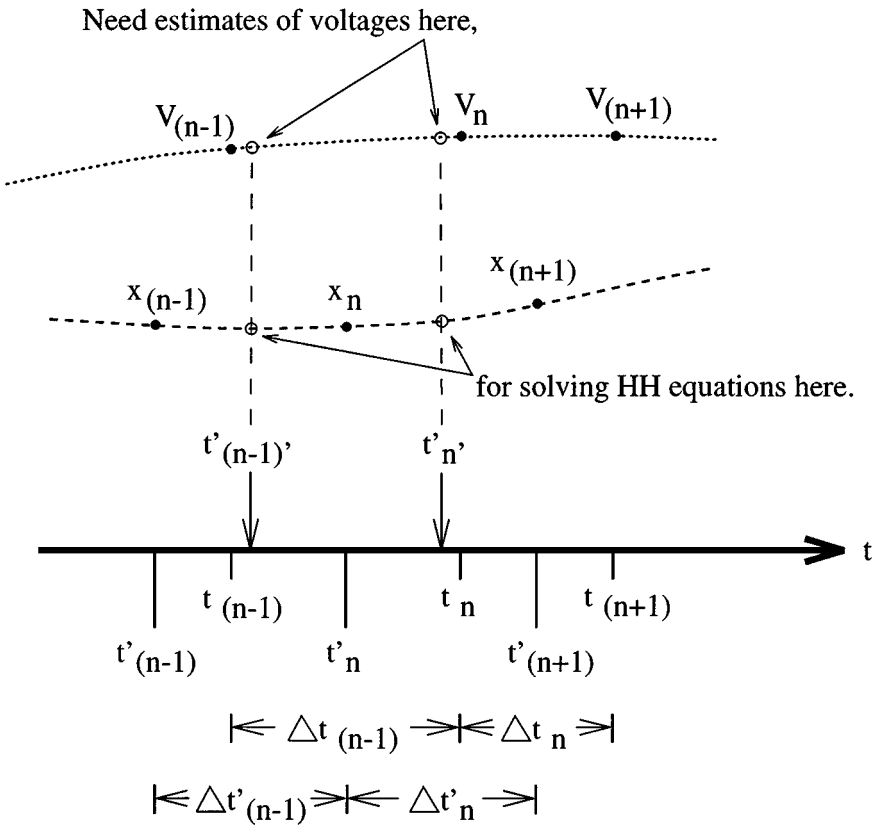

Figure 1. Time grids for evaluation of node voltages, gating particle states, and inputs, as described in the text.

where g_{ij} is the cable compartment axial conductance between nodes i and j. Similarly, for the vector $\mathbf{GE}(t'_{(n+1)})$,

$$GE_i = \sum g_{elt}\left(t'_{(n+1)}\right) \times E_{g_{elt}}\left(t'_{(n+1)}\right). \qquad (3)$$

Thus, g_{elt} refer to the (possibly time-dependent) membrane conductances (leak, channels, synapses) between a given node and ground, and $E_{g_{elt}}$ refer to the appropriate reversal potentials. The sums of g_{elt} in Eqs. (2) and (3) are over all membrane conductance elements connected to node i. Again, note that these circuit elements are evaluated at the staggered grid points $t'_{(n+1)}$. These values require only the previous node voltages $\mathbf{V}_{(n-1)}$ and $\mathbf{V}_n$, as described in Section 2.1.2; therefore, they are available for solving Eq. (1). For the vector $\mathbf{C}$, C_i is the membrane capacitance of node i. $\mathcal{I}$ is the identity matrix. For the vector $\mathbf{I}$, I_i represents any grounded current source (or sum of sources) connected to node i, evaluated at time $t'_{(n+1)}$. Since the compartmental model of either a single neuron or a network of neurons (without gap junctions) describes a tree topology, the solution of Eq. (1) is an $O(N)$ operation (Hines, 1984).

The explicit phase of the solution to finish the time step evaluation is then given by

$$\mathbf{V}_{(n+1)} = 2\mathbf{V}\left(t'_{(n+1)}\right) - \mathbf{V}_n.$$

2.1.1. Solution of Nonlinear Conductances on a Staggered Time Grid. The staggered time-grid evaluation of the Hodgkin-Huxley (HH) gating variables is $O(\Delta t^2)$ accurate without iteration because these variables are not instantaneous functions of voltage (Hines, 1984; Mascagni and Sherman, 1998). This condition holds for ohmic models of pore conduction (such as the classical Hodgkin-Huxley description): permeation models of pore conduction that are based on formulations such as the constant-field equation are another matter since the basic conduction term is an immediate nonlinear function of voltage. For Hodgkin-Huxley ohmic channels, the contribution of a given channel to the circuit matrix is given by the product of the channel's maximum conductance and the ensemble of gating particles. In principle a small-signal linearization of a constant-field permeation channel could be incorporated into the matrix in a similar manner, taking care to adjust the apparent channel "reversal potential" accordingly in the matrix equation. Another approach is to treat the channel as a voltage-dependent current source, adding its contribution to the right-hand side of Eq. (1).

2.1.2. Integration of Particle States. We now consider the solution of Eq. (1) using an adaptive time step. The midstep computation of voltage-dependent state variables may be recast as follows. The problem is to solve the differential equation for a gating particle x:

$$\dot{x} = \frac{x_\infty(V) - x}{\tau_x(V)} \qquad (4)$$

on the staggered time grid t'_n, defined earlier. For second-order accuracy this solution requires the values of $x_\infty(V)$ and $\tau_x(V)$ at the midpoints of the staggered grid. If the time step is fixed, these midpoints map back to the original time grid: thus, solving for $x(t'_{(n+1)})$ references the (known) node voltage V_n. However, for an adaptive time step, the midpoint of the staggered grid in general is not equal to t_n but must be calculated as a function of the present and last time steps of the original grid (Fig. 1). We first derive the time step for the staggered grid:

$$\Delta t'_n = t'_{(n+1)} - t'_n = \frac{\Delta t_n + \Delta t_{(n-1)}}{2}.$$

Letting $n + \frac{1}{2} \equiv n'$, the staggered-grid midpoint time may be expressed as

$$t'_{n'} = t'_n + \frac{\Delta t'_n}{2}.$$

The corresponding voltage can be obtained by simple regression from t'_n:

$$V(t'_{n'}) = V(t'_n) + \dot{V}(t'_n) \times \frac{\Delta t'_n}{2},$$

where

$$\begin{aligned} V(t'_n) &= \frac{V_n + V_{(n-1)}}{2} \\ \dot{V}(t'_n) &= \frac{V_n - V_{(n-1)}}{\Delta t_{(n-1)}}. \end{aligned} \qquad (5)$$

Making the following discretizations:

$$\begin{aligned} x(t'_{n'}) &= \frac{x_{(n+1)} + x_n}{2} \\ \dot{x}(t'_{n'}) &= \frac{x_{(n+1)} - x_n}{\Delta t'_n}, \end{aligned} \qquad (6)$$

we now solve Eq. (4) at $t'_{n'}$. Letting $\tau_x = \tau_x(V(t'_{n'}))$ and $x_\infty = x_\infty(V(t'_{n'}))$, after some algebra we arrive at the following solution for $x_{(n+1)}$:

$$x_{(n+1)} = \frac{[x_\infty \times \Delta t'_n] + x_n[\tau_x - \Delta t'_n/2]}{\tau_x + \Delta t'_n/2}. \quad (7)$$

As pointed out by Hines, the voltage-dependencies of the particle kinetics may be stored in lookup tables for efficiency. In this case, tables for $x_\infty(V)$ and $\tau_x(V)$ are stored. An equivalent formulation to Eq. (7) may be made in terms of the rate constants α_x and β_x associated with Eq. (4) (cf. Eq. (9) in Hines, 1984), but the form given above is perhaps more clear since in the latter case tables for $\alpha_x(V)$ and $\frac{\alpha_x(V)-\beta_x(V)}{2}$ must be generated. In practice, additional computational savings may be made by calculating $V(t'_{n'})$ only for those circuit nodes that have voltage-dependent elements, and only once, at the beginning of each time step, such that the result may be used by multiple elements on a given node.

We may note that Eq. (7) requires three additions, one multiplication and one division. This compares with the single multiplication and addition required in the case of a fixed time step, where the (known) time step may be incorporated in the lookup tables. We will return to this point in the discussion.

2.1.3. Time Step Determined by LTE (Linear Truncation Error) of Node Voltages and Particle States. The time step during the integration is determined according to its relation to the truncation error. Since the Crank-Nicolson method is first order, this error is given by the second term of the Taylor series expansion of the solution and is called the *linear truncation error* (LTE). Thus, after evaluation of the circuit at each time step, estimates of the LTE for the circuit's state variables are determined. For each state variable an estimate of the second derivative at the midpoint between the current time and the two time steps back (note that for a fixed time step this corresponds to the last time step) is made by considering the known values of the first derivatives on the appropriate time grid.

In practice, the maximum of the the estimated errors for, respectively, all the considered node voltages (see below), all channel gating particle states, and all concentration integrator concentrations are examined. These maximum errors are compared with user-specified maximum errors, and if any are exceeded, then the time step is repeated with a smaller step, determined by the maximum LTE at the current step. If all errors are within the user-specified bounds, then the integration moves on, now with the next step determined by the maximum LTE.

Here we first describe the derivation for the LTE of the various state variables, and then we show how these estimates are translated into the next time step.

The LTE for the predicted node voltage $V_{(n+1)}$, LTE_V, is given by

$$\begin{aligned}\text{LTE}_V &= \frac{\ddot{V}(t'_{n'})}{2}\Delta t_n^2 \\ &= \frac{\dot{V}\left(t'_{(n+1)}\right) - \dot{V}(t'_n)}{2\left(t'_{(n+1)} - t'_n\right)}\Delta t_n^2,\end{aligned}$$

where the expression for $\dot{V}(t'_n)$ was given in Eq. (5). Letting the maximum *a priori* allowed node voltage error be given by ϵ^V_{max}, the maximum relative voltage error ϵ^V_{rel} is given by

$$\epsilon^V_{rel} = \frac{\text{LTE}^*_V}{\epsilon^V_{max}},$$

where LTE^*_V is the maximum LTE_V over all considered node voltages.

For concentration integrator systems, the corresponding LTE_C may be handled in an entirely analagous manner as for the node voltages, assuming that the concentrations are evaluated on the same time grid (this is the case for Surf-Hippo, with the default integration method for concentrations being a fully implicit method). In this case, the above equations may be applied as is, with the voltage variables replaced with concentration variables; given a user-specified maximum allowed concentration error ϵ^C_{max}, the important resulting measure is ϵ^C_{rel}.

For the integration of gating particles, the linear truncation error LTE_x for the predicted state value $x_{(n+1)}$ (occuring at time $t'_{(n+1)}$) is derived as follows. Letting t'' be midway between the last two midpoints $t'_{(n-1)'}$ and $t'_{n'}$ of the staggered grid, then

$$\begin{aligned}\text{LTE}_x &= \frac{\ddot{x}(t'')}{2}\Delta t_n'^2 \\ &= \frac{\dot{x}(t'_{n'}) - \dot{x}(t'_{(n-1)'})}{2\left(t'_{n'} - t'_{(n-1)'}\right)}\Delta t_n'^2,\end{aligned}$$

where the expression for $\dot{x}(t'_{n'})$ was given in Eq. (6). Now, letting the maximum allowed particle error be given by ϵ^x_{max}, the maximum relative particle error ϵ^x_{rel}

is given by

$$\epsilon_{rel}^{x} = \frac{\mathrm{LTE}_x^*}{\epsilon_{max}^{x}},$$

where LTE_x^* is the maximum LTE_x over all the gating particles in the circuit.

We now derive the maximum allowed time steps corresponding to the different LTEs. Given the maximum relative voltage error ϵ_{rel}^{V}, the maximum time step allowed by the node voltages, Δt_{max}^{V}, is given by

$$\Delta t_{max}^{V} = \frac{\Delta t_n}{\sqrt{\epsilon_{rel}^{V}}}. \tag{8}$$

A similar parameter, Δt_{max}^{C}, determined by the maximum relative error in the concentrations, ϵ_{rel}^{C}, may be estimated in exactly the same manner.

For the gating particle error, since LTE_x is a function of both Δt_n and $\Delta t_{(n-1)}$ (via $\Delta t_n'$), determining the corresponding maximum allowed time step Δt_{max}^{x} is a bit more complicated than Δt_{max}^{V}. Thus, we obtain

$$\Delta t_{max}^{x} = \max\left(\frac{2\Delta t_n'}{\sqrt{\epsilon_{rel}^{x}}} - \Delta t_{prev},\ \Delta t_{min}^{x}\right), \tag{9}$$

where the first term on the right depends on whether or not the integration step will be repeated:

$$\Delta t_{prev} = \begin{cases} \Delta t_{(n-1)} & \text{Time step repeated} \\ \Delta t_n & \text{Integration advances.} \end{cases}$$

Note, however, that when the time step must be repeated, the first term on the right of Eq. (9) may be less than 0 (when $\sqrt{\epsilon_{rel}^{x}} \geq \frac{\Delta t_n + \Delta t_{(n-1)}}{\Delta t_{(n-1)}}$). Thus, a lower (positive) bound (Δt_{min}^{x}) must be made, as indicated.

Finally, if the largest of the maximum relative errors (ϵ_{rel}^{V}, ϵ_{rel}^{x} and ϵ_{rel}^{C}) is greater than one, then the current time step is repeated; otherwise, the integration moves forward. In either case, the new time step is taken as

$$\Delta t_n = \min\left(\Delta t_{max}^{V},\ \Delta t_{max}^{C},\ \Delta t_{max}^{x}\right). \tag{10}$$

2.1.4. Node Voltages for Consideration of LTE_V. At any given time during the simulation, the node with the largest error must be one that has some input associated with it, such as a source, channel, or synapse. All other nodes are driven by their neighboring nodes, and since the capacitance of each node is nonzero, then the response of a node driven by the voltage of a neighbor will always be slower than the neighbor's voltage. Therefore, for estimating the maximum LTE_V, we need only consider nodes with inputs; at the beginning of the simulation a list of all nodes with active membrane elements may be constructed for this purpose. One exception to this rule is in the case of ideal voltage-clamp simulations, to be described below: here the node with the input is actually removed from the circuit, and copies of the voltage source are transferred to adjacent circuit nodes. Thus, these nodes may now be considered to have "inputs" and are included in the LTE_V calculation.

2.1.5. Time-Step Fudge. Although the determination of the time step as outlined above does the best possible (first-order) job based on the *past* behavior of the circuit, clearly it may underestimate the resulting error for the next time step. This results in a tradeoff between the total number of time points during the integration and any additional iterations that result from underestimating the LTE.

One method for improving the tradeoff is by introducing a fudge factor $\varepsilon_{\Delta t}$, less than or equal to one, which is used as a coefficient in the right-hand side of Eqs. (8) and (9). In general, as $\varepsilon_{\Delta t}$ becomes smaller, the number of time points will increase, and the number of iterations will be reduced. After a certain point for a small enough $\varepsilon_{\Delta t}$, the number of iterations will start to increase; in general, $\varepsilon_{\Delta t}$ should be set to minimize the number of iterations. In Fig. 2, this tradeoff suggests an optimal value for $\varepsilon_{\Delta t}$ to be about 0.8.

2.1.6. Breakpoints. For adaptive time-step integration, breakpoints are time points that the simulation must incorporate in addition to those chosen by the LTE-based algorithm described above. In general, a breakpoint is chosen because there is *a priori* knowledge of some input that begins or changes abruptly at that time. This makes less work for the adaptive time step, since without this information the initial stepping would more than likely encounter the input sometime after its initiation and thus may be forced to back up if the input was too large at the first attempted time point that "saw" the input or the change in the input. The result (in general) with using breakpoints then is fewer overall iterations and a solution that is less likely to have "ringing." Breakpoints also avoid the situation in which an abrupt and short input might be completely

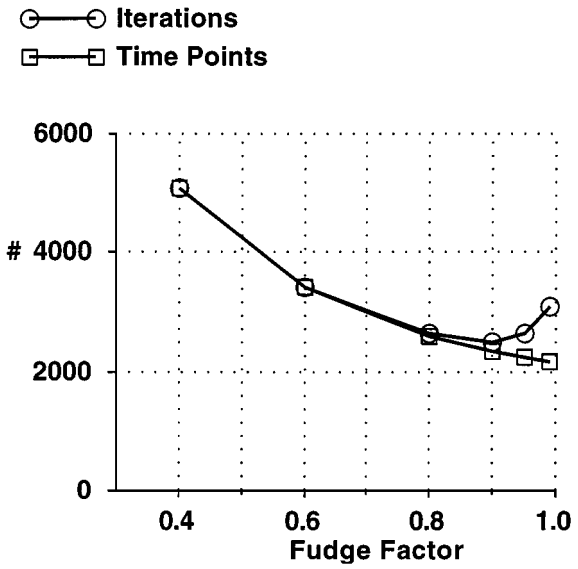

Figure 2. Total iterations and time points versus adaptive-time step fudge factor for simulations of repetitive firing as shown in Fig. 5. In all the adaptive time-step simulations shown here, the fudge factor $\varepsilon_{\Delta t}$ is set to 0.95. Although in general a value about 0.8 is optimal, the higher value is taken to deemphasize this factor in comparing the adaptive versus fixed time-step method.

missed by large time steps determined during a previous period of low activity.

Thus, prior to a simulation, breakpoints are added for any pulse-based source and at the onsets of any autonomous processes, such as autonomous synapses. During a simulation new breakpoints may also be generated with the evaluation of event-based elements, including axons and voltage-dependent synapses. When the simulation is evaluated at a breakpoint, the subsequent time step is taken to be a user-defined minimum step, unless this advance takes the integration beyond the next breakpoint, in which case the next breakpoint determines the time step.

In Surf-Hippo, element type definitions (see Appendix) may optionally include a specification for the inclusion of breakpoints referenced to that element's onset time. For example, a synapse type definition (which includes the time course of the conductance change) may specify that whenever one of its synapses is triggered, then one or more extra breakpoints (referenced to the onset of the synaptic conductance change) are automatically added to the global breakpoint list of the current simulation.

2.1.7. Other Time-Step Constraints. Under some conditions it may be useful to impose an overall maximum time step. For example, a maximum could be imposed during the evaluation of some element types, in particular those that are driven by an *a priori* waveform. In these cases (particularly if the waveform is not well behaved), the global maximum time could be set to a value appropriate for the specific waveform. One may also consider a minimum time step (such as greater than 0), but we have found that in most cases this is not necessary.

2.2. Ideal Voltage Clamp

Ideal voltage clamp means that a given node j (or nodes) in the circuit includes a controlled voltage source, $V_j^s(t)$, connected to ground. However, when the circuit is described in the form given by Eq. (1), this prevents a solution for $\mathbf{V}(t'_{(n+1)})$ since now there are more equations than unknowns (the value of $V_j(t'_{(n+1)})$, where node j is voltage clamped, is known and given by $V_j^s(t'_{(n+1)})$).

One solution is to make the voltage clamp nonideal by adding a series resistance R_{Source} to V_j^s, so that the voltage source and resistor are handled like any other membrane element and associated reversal potential on the right side of Eq. (1). However, although this allows the evaluation of voltage clamp within the original circuit (and simulator) structure, this strategy has two disadvantages. First of all, the smaller the value of R_{Source} the more unstable the integration becomes, neccesitating smaller time steps. This arises because as R_{Source}^{-1} becomes much larger than the g_{elt}s, the stiffness of the resulting system of equations increases. Second, although voltage-clamp circuits in real life are not ideal (for example, they include a nonzero source impedance), it is useful to examine the conceptually simpler ideal case in simulations separately from that of the more realistic case.

These problems may be avoided by transfering copies of the voltage source V_j^s to a grounded membrane branch element of all nodes i connected to j, each of which includes a series conductance given by the axial conductance between i and j, g_{ij} (see Fig. 3). In addition, the original connection between i and j via g_{ij} is eliminated. This procedure results in an invertible circuit matrix that is of order $N-1$. The stability (and stiffness) of the resulting circuit is similar to that of the original circuit, since there are no additional (large) terms to the node admittance matrix, contrary to the case for the nonideal voltage-clamp simulation.

With respect to the topology of the circuit, this procedure may be thought of as cutting the tree into two or more subtrees, depending on how many nodes are

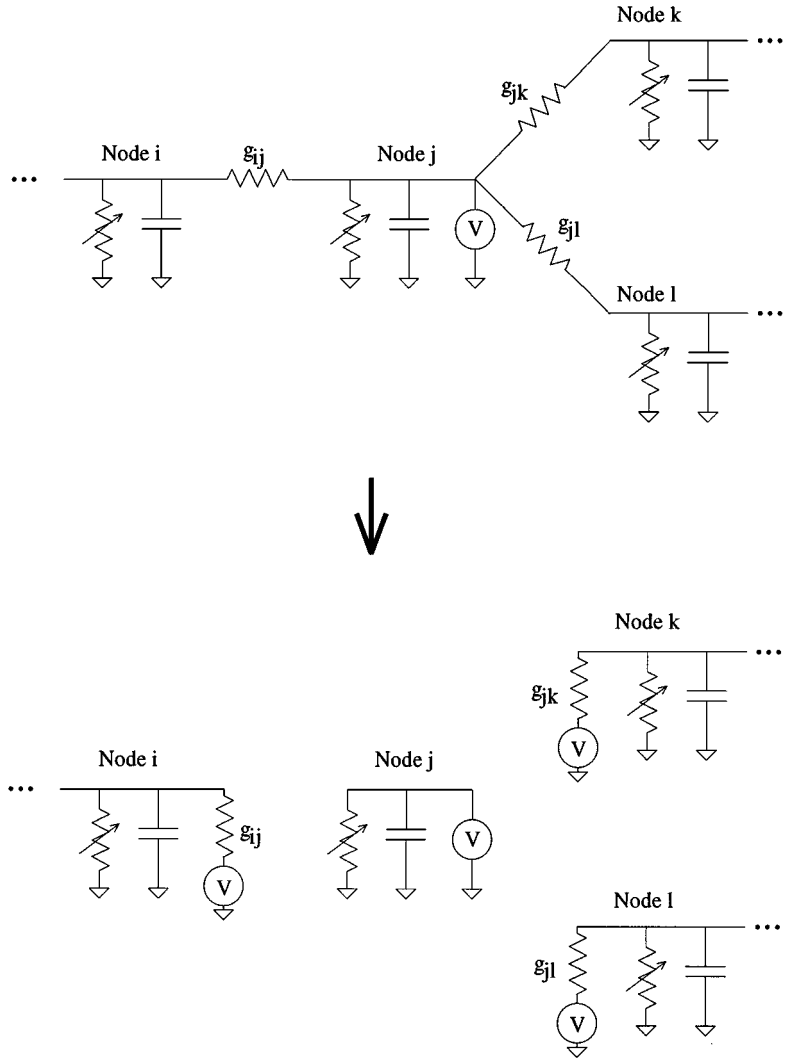

Figure 3. Method for splitting a circuit tree for ideal voltage clamp. In the example shown, the voltage-clamped node j is connected to three other nodes i, k and l, but the method is the same for an arbitrary number of connections. Here the original circuit tree is divided into three new trees (terminated by nodes i, k, and l). Node j is evaluated independently of the new circuit matrix defined by the three trees. The variable resistors represent membrane conduction elements (leak, channels, and synapses) for each node and, implicitly, the associated voltage sources representing the appropriate reversal potentials.

connected with j. Since the removal of any subgraph of a tree topology nevertheless results in one or more tree topologies, the original node ordering and efficient inversions as described by Hines is still applicable, in this case to the new, smaller trees.

The only remaining problem is the calculation of the currents through the membrane elements associated with node j (such as channels, synapses, membrane leak, and capacitance) and the total current through the source V_j^s. In the former case, the circuit elements are evaluated as before, with the node voltage (or derivative, in the case of membrane capacitance) taken from the value of V_j^s at the appropriate time. In the latter case, the total current supplied by V_j^s is simply the sum of all the branch currents from the grounded membrane elements of node j, plus the axial branch currents across the appropriate g_{ij} with a driving force given by the difference between the voltage of the neighboring nodes i and V_j^s. Note that the evaluation of the voltage source current passing through membrane elements of node j is independent of the evalation of the new circuit matrix.

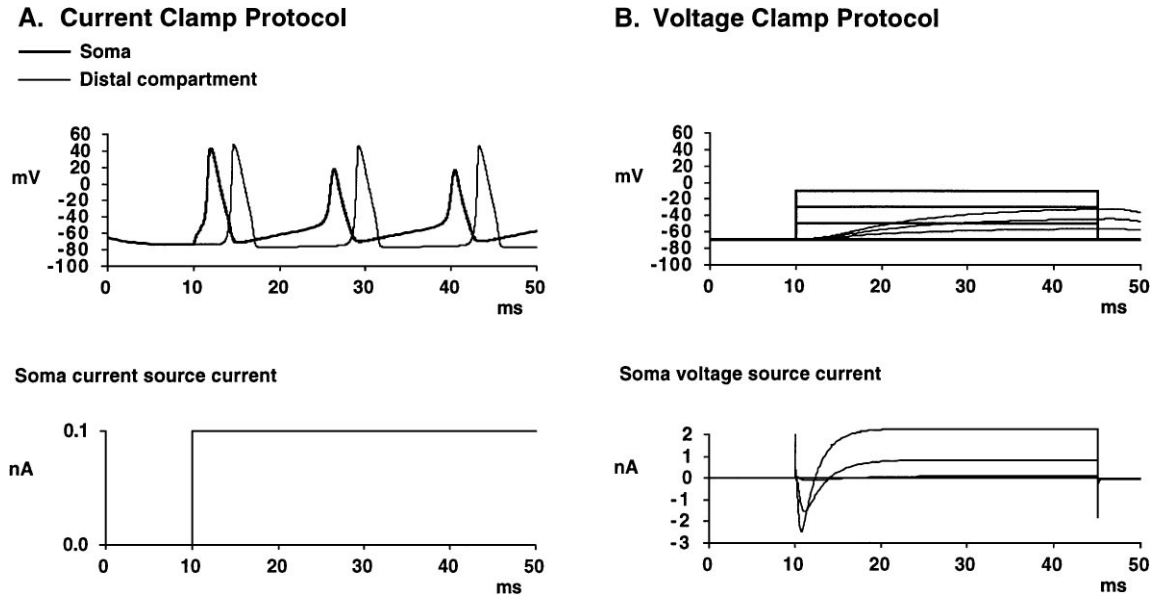

Figure 4. Basic current clamp (**A**) and (ideal) voltage clamp (**B**) protocols for the simulations presented in this paper, run here using an adaptive time step. For the current clamp protocol all compartments have a homogeneous distribution of Hodgkin-Huxley I_{Na} and I_{DR} channels (source code given in Appendix). For the voltage-clamp protocol only somatic channels are enabled.

3. Results

I now present various current-clamp and voltage-clamp simulations to illustrate the methods just described. The cell model is based on the Rallpack 3 specification (Bhalla et al., 1992), which defines a single unbranched 1 micron diameter cable, 1,000 microns long, with a nominally homogenous distribution of the canonical Hodgkin-Huxley I_{Na} and I_{DR} squid axon channels. The version used here is defined with 11 compartments, constructed so that the circuit is symmetrical (see Appendix). The compartment on one end is defined to be the *soma*; the compartment on the other end is referred to as the *distal compartment*. For detailed comparisons all plotted simulation results include every time step; normally plotting every other time step of the integration is sufficient for typical studies.

The current-clamp protocol (Fig. 4A) is the response to a somatic current source of 0.1 nA, applied at 10 milliseconds and lasting until the end of the simulation. The initial holding potential for all compartments is -65 mV; thus, there is an initial transient as the cell moves toward the true resting potential of -72.7 mV. This simulation represents a reasonable test of the adaptive time-step algorithm since the activity in the circuit is somewhat dispersed in both space and time.

For the voltage-clamp protocol (Fig. 4B), the voltage-clamp is applied at the soma and all nonsomatic channels are deactivated for simplicity. A somatic holding potential of -70 mV is used, and the complete protocol consists of four repetitions, each with a 35 millisecond voltage step (to -70, -50, -30, and -10 mV) applied at 10 milliseconds, followed by a return to -70 mV. In all cases the voltage source has a fixed transition slope of 1,000 mV per millisecond.

The fixed time-step simulation of the repetitive firing under current clamp converges as the time step goes from 0.01 (5,000 time steps) to 0.005 milliseconds (10,000 time steps) (Fig. 5A). Similar convergence occurs as the maximum allowed voltage error ϵ^V_{max} goes from 0.05 (1,662 time points, 1,982 total iterations) to 0.005 mV (5,151 time points, 5,408 total iterations) in the adaptive time-step case, without considering the LTE of the particle states (Fig. 5B).

There is an inherent tradeoff between the various LTE criteria, depending on the *a priori* values given for ϵ^V_{max}, ϵ^x_{max} and ϵ^C_{max}. This is illustrated in Figs. 7 through 9 in the case of ϵ^V_{max} and ϵ^x_{max}. These simulations show that there is a smooth and consistent exchange with respect to which type of LTE determines the time steps, as one criteria is made tighter than the other.

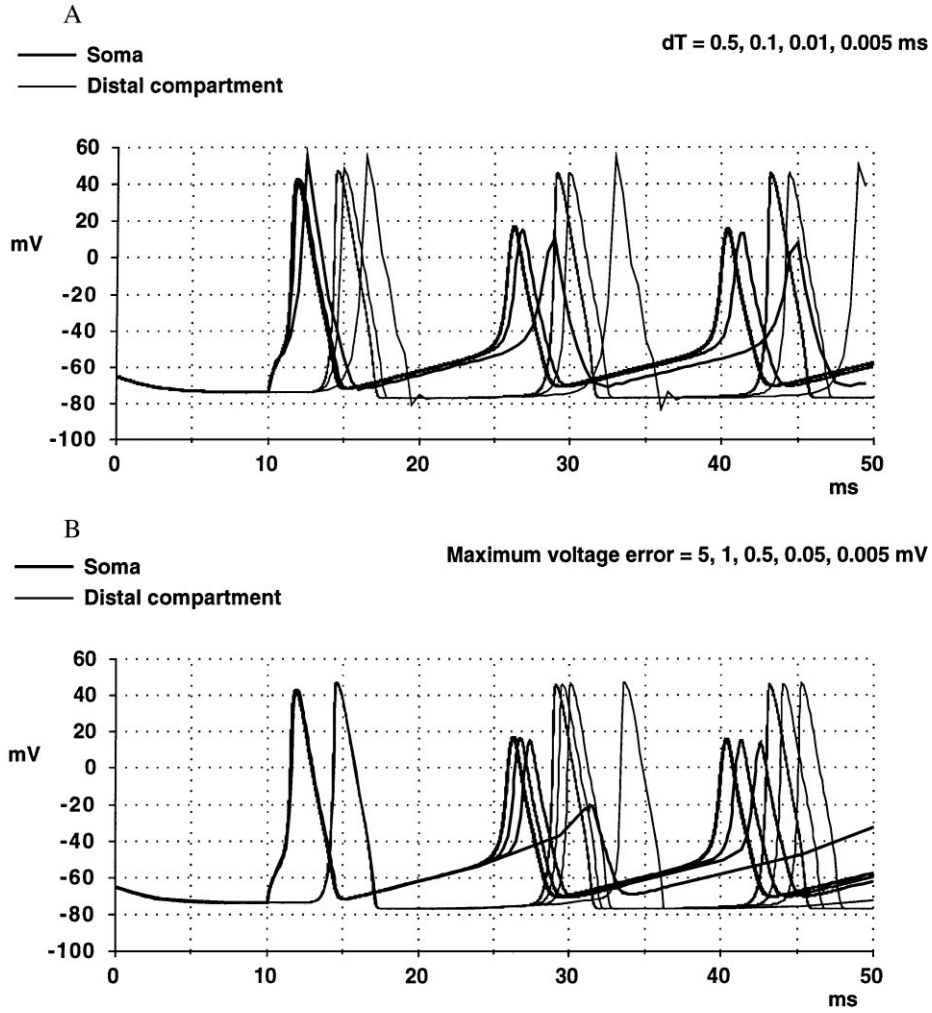

Figure 5. Comparison of convergence properties for repetitive firing during current clamp for different fixed time steps and adaptive time-step error criteria. **A:** Fixed time step simulations show convergence as the time step is reduced from 0.01 milliseconds (5,000 time points) to 0.005 milliseconds (10,000 time points). **B:** Adaptive time-step simulations, with only voltage error considered, show convergence as the maximum allowed voltage LTE ϵ^V_{max} is reduced from 0.05 mV (1,662 time points, 1,982 total iterations) to 0.005 mV (5,151 time points, 5,408 total iterations). Note that for both the fixed and adaptive time steps that the simulations are well behaved (stable) for all parameter values.

Under voltage clamp, the part of the response that is most sensitive to numerical stability and accuracy issues are probably at the pulse transitions, during which time the voltage source current is dominated by the capacitive transient. Under ideal voltage clamp, in Fig. 10 we can see that the adaptive time-step case follows the high-resolution fixed time step case ($\Delta t = 0.001$ ms) quite well, without ringing. In the nonideal voltage-clamp case, there is a basic tradeoff between stability and accuracy as a function of R_{Source}. This is illustrated in Fig. 11, where, under adaptive time step, a very low value of R_{Source} shows oscillations in the voltage source current.

The stability of the source current for nonideal voltage clamp is much more sensitive to R_{Source} when a fixed time step is used, as seen in Fig. 12. When $R_{Source} = 0.01$ MΩ there is a huge oscillation (amplitude on the order of 50 nA) in the voltage-source current during the pulse transitions. When $R_{Source} = 0.1$ MΩ, there is still a significant oscillation (amplitude on the order of 2 nA). As a practical matter, although the voltage error for the nonideal voltage clamp with $R_{Source} = 0.1$ MΩ is not very significant (Fig. 11), these oscillations make it more difficult to extract the correct peak values of the clamp currents.

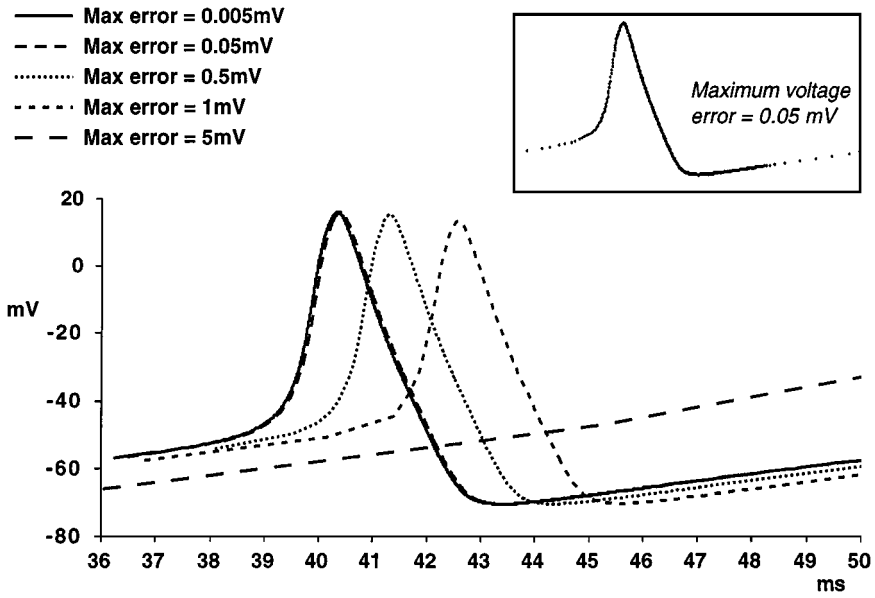

Figure 6. Detailed comparison of third somatic action ptential in repetitive firing simulations shown in Fig. 5, showing the results using various error criteria for the adaptive time step. Inset: Time points for adaptive time step with ϵ^V_{max} set to 0.05 mV.

4. Discussion

The basic performance issue with regards to the adaptive time-step method is at which point the greater number of operations per time step for this method is countered by the smaller number of time steps and iterations for a given accuracy as compared to the fixed-step method. In the current-clamp simulation presented here overall the adaptive time-step case was about 2.5 times faster than the fixed time-step method. For the (ideal) voltage-clamp simulation the difference was about 12 times. While it must be pointed out that there is no formal proof that the tested code is as efficient as possible (for either method), nevertheless this is a strong indication of the advantage of the adaptive method.

In the case of voltage clamp, I have shown how an adaptive time step allows a small enough value of R_{Source} ($= 0.1\,\mathrm{M}\Omega$) for the nonideal voltage clamp so that voltage errors are small and that with a fixed time step the value gives transient oscillations that can complicate data analysis (Fig. 12). Nevertheless, the ideal voltage-clamp formulation is much more robust to this issue, as well as being conceptually simpler. The computational overhead of this method is actually less (though trivially so) than that for the nonideal case, since in the ideal case the dimension of the circuit matrix is reduced by one.

Finally, the programming of both the adaptive time step and ideal voltage clamp algorithms are straightforward and should be easily ported to existing code that use the Hines method (e.g., NEURON, GENESIS) (Bower and Beeman, 1994).

Appendix: Surf-Hippo

In this appendix we illustrate the Surf-Hippo source code used in some of the presented simulations. Surf-Hippo is one of the few complete compartmental modeling packages written in Lisp, as opposed to C. Lisp has the advantage that the user is communicating directly with the Lisp interpreter environment and thus has complete access to all components of a simulation. Simulation scripts, a necessity for serious parameter searching, are also written in Lisp, whose flexible and powerful syntax is arguably more transparent than most other languages. Although Lisp allows "quick and dirty" prototyping, when necessary careful attention to type declarations and other coding details allows numerical performance on par with C.

We now present the Surf-Hippo source code (written in `typewriter style`), which (1) defines the Hodgkin-Huxley I_{Na} and I_{DR} channel types, (2) defines the soma/short-cable Rallpack 3 structure, and (3) sets up and runs the current clamp simulation illustrated on the left in Fig. 4. This code may also be found in the

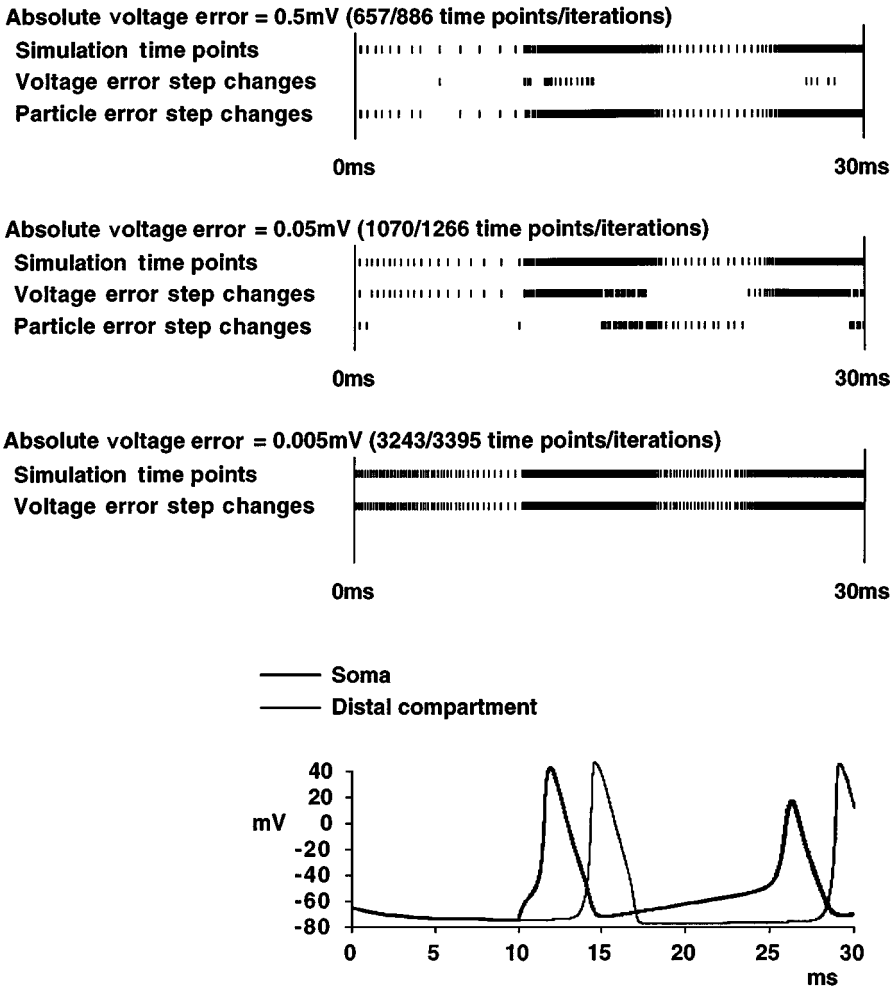

Figure 7. Adaptive time steps as a function of maximum allowed voltage LTE (ϵ^V_{max}), for a fixed maximum allowed particle LTE, ϵ^x_{max} ($= 0.001$), for the first 30 milliseconds of the current-clamp protocol described in Fig. 4. Raster plots in this and the following two figures compare the times of all time steps used in the simulations, with those time steps that are specifically determined by either the voltage error or the gating particle error, as appropriate. In addition, simulation output at the bottom is overlaid for each simulation described in the raster plots; in all cases these plots are indistinguishable.

file "rallpack-3-cc-demo.lisp" in the Surf-Hippo distribution. Note that in Lisp, text that follows a ";" is interpreted as a comment. Symbols (used for names of functions and global variables) are case insensitive. It is useful to point out that although the plethora of parentheses may seem daunting, various text editors (notably EMACS) handle these automatically.

The parameters for the various element types—cell types, channel types, synapse types, particle types, concentration dependent particle types, concentration integrator types, axon types, pump types, and buffer types—are referenced from parameter libraries specific to each type. Adding (and updating) a new entry of a given element type to the appropriate library is done with the Type-def macros, such as `CHANNEL-TYPE-DEF`, `CONC-INT-TYPE-DEF`, and so on. The body of each Type-def macro is a quoted list whose first element is the name (typically a symbol) of an element type, followed by an association list of parameters specific to that sort of element type. The names of the element types given by the Type-defs may then be referenced in the source code.

We start with the definitions for the Hodgkin-Huxley (Hodgkin and Huxley, 1952) I_{Na} and I_{DR} channel types (respectively, `NA-HH` and `DR-HH`). These definitions are made with the `CHANNEL-TYPE-DEF` macro:

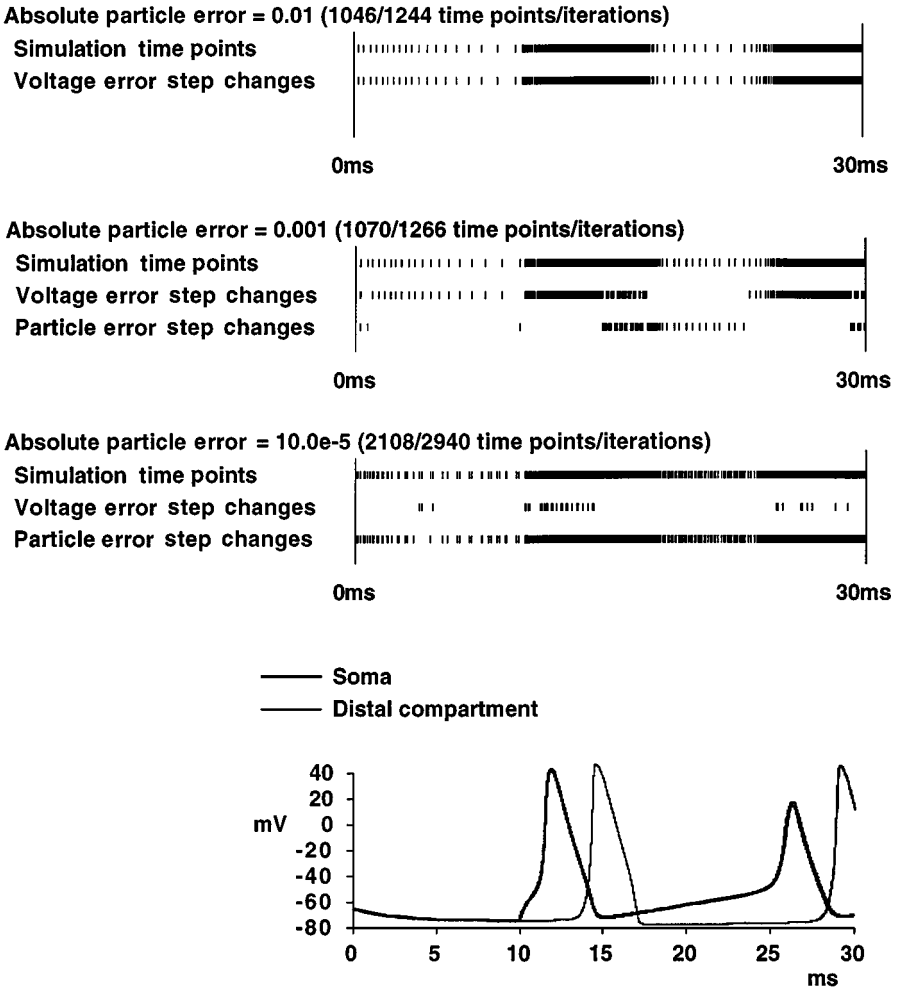

Figure 8. Adaptive time steps as a function of ϵ^x_{max}, for a fixed $\epsilon^V_{max} = 0.05$ mV, for the first 30 milliseconds of the current-clamp protocol described in Fig. 4.

```
(channel-type-def
 '(NA-HH
   (gbar-density . 1200)                  ; pS/um2
   (e-rev . 50)                           ; mV
   (v-particles . ((M-HH 3) (H-HH 1))))) ; There are 3 M-HH particles and 1 H-HH particle.

(channel-type-def
 '(DR-HH
   (gbar-density . 360)                ; pS/um2
   (e-rev . -77)                       ; mV
   (v-particles . ((N-HH 4)))))        ; There are 4 N-HH particles.
```

These channel type definitions reference various particle types, including `M-HH`, `H-HH`, and `N-HH`, whose definitions are made using the `PARTICLE-TYPE-DEF` macro:

```
(particle-type-def
 '(M-HH
   (class . :HH)                        ; This particle is of the canonical HH form.
   (alpha-function . M-HH-ALPHA)        ; The forward rate constant is given by this function.
   (beta-function .  M-HH-BETA)))       ; The backward rate constant is given by this function.

(particle-type-def
 '(H-HH
   (class . :HH)
   (alpha-function . H-HH-ALPHA)
   (beta-function . H-HH-BETA)))

(particle-type-def
 '(N-HH
   (class . :HH)
   (alpha-function . N-HH-ALPHA)
   (beta-function . N-HH-BETA)))
```

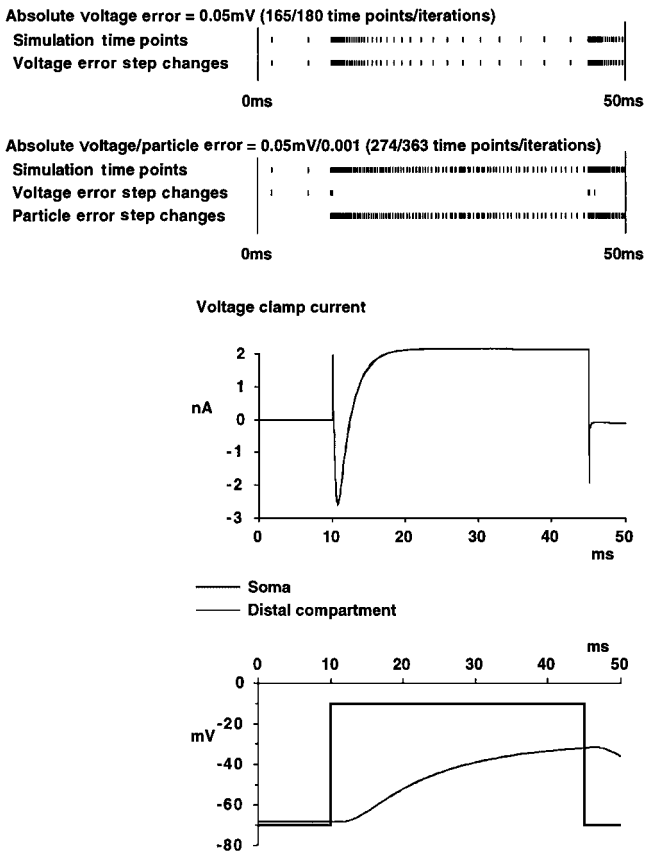

Figure 9. Time steps determined by voltage versus particle LTE for the -70 mV to -10 mV step of the (ideal) voltage-clamp protocol described in Fig. 4. In the top raster plot only voltage LTE $\epsilon^V_{max}(= 0.05$ mV) was used to determine time steps; as described in the text, the appropriate nodes for this calculation included both the clamped (soma) node and the adjacent node. This explains how the voltage LTE may be nonzero after the pulse transitions. In the lower raster plot, gating particle LTE ($\epsilon^x_{max} = 0.001$) were also considered in the determination of the time step. With the (default) values as given, the particle LTE determines all of the time steps subsequent to the pulse transition. Time course for the soma and distal compartment voltages and the clamp current are superimposed in the bottom plots for both error criteria and are essentially identical.

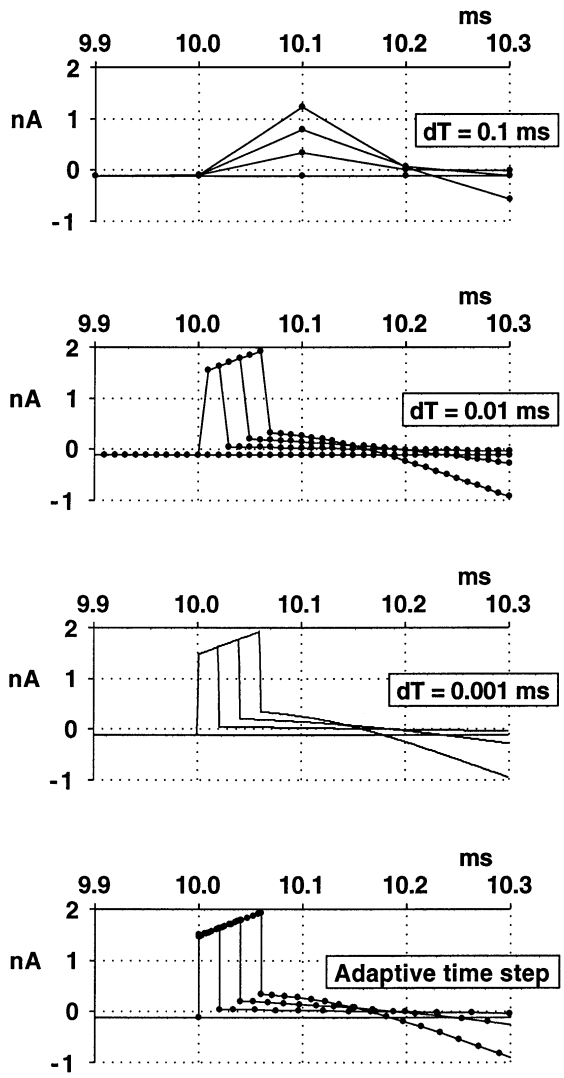

Figure 10. Capacitive transient currents under ideal voltage clamp with fixed and adaptive time step. In the adaptive time-step case, breakpoints are used at the start and stop times of each transition of the voltage source pulse. The actual time steps are shown in each case except for the fixed step of 0.001 milliseconds. Note that the adaptive time step follows very well the transient response for the fixed step of 0.001 milliseconds. The time step prior to 10.0 milliseconds in the adaptive time-step case occurred at 7.0 milliseconds.

The particle type definitions, in turn, reference various forward and backward rate functions (e.g., `M-HH-ALPHA` and `M-HH-BETA`, respectively, for the `M-HH` particle type). In the definitions for these functions, the voltage argument is assumed to be in mV, and the functions return rates in 1/ms:

```
(defun m-hh-alpha (voltage)
  (/ (* -0.1 (- voltage -40))
     (1- (exp (/ (- voltage -40) -10)))))
```

Given the prefix notation of Lisp, this expression is equivalent to

$$\alpha_m(V) = \frac{-0.1(V - -40)}{1 - \exp(\frac{V - -40}{-10})}$$

The remaining rate functions are then:

```
(defun m-hh-beta (voltage)
  (* 4 (exp (/ (- voltage -65) -18))))

(defun h-hh-alpha (voltage)
  (* 0.07 (exp (/ (- voltage -65) -20))))

(defun h-hh-beta (voltage)
  (/ 1.0 (1+ (exp (/ (- voltage -35) -10)))))

(defun n-hh-alpha (voltage)
  (/ (* -0.01 (- voltage -55))
     (1- (exp (/ (- voltage -55) -10)))))

(defun n-hh-beta (voltage)
  (* 0.125 (exp (/ (- voltage -65) -80))))
```

The macro `CELL-TYPE-DEF` defines the parameters for the cell type `HH-AXON`:

```
(cell-type-def
 '(HH-AXON
   (rm . 40000)       ; ohms cm2
   (ri . 100)         ; ohms cm
   (cm . 1)           ; uF/cm2
   (v-leak . -65)))   ; mV
```

Next we define a function that creates a soma/short-cable cell with 11 total compartments (1 soma and 10 segments) and adds I_{Na} and I_{DR} to all compartments.

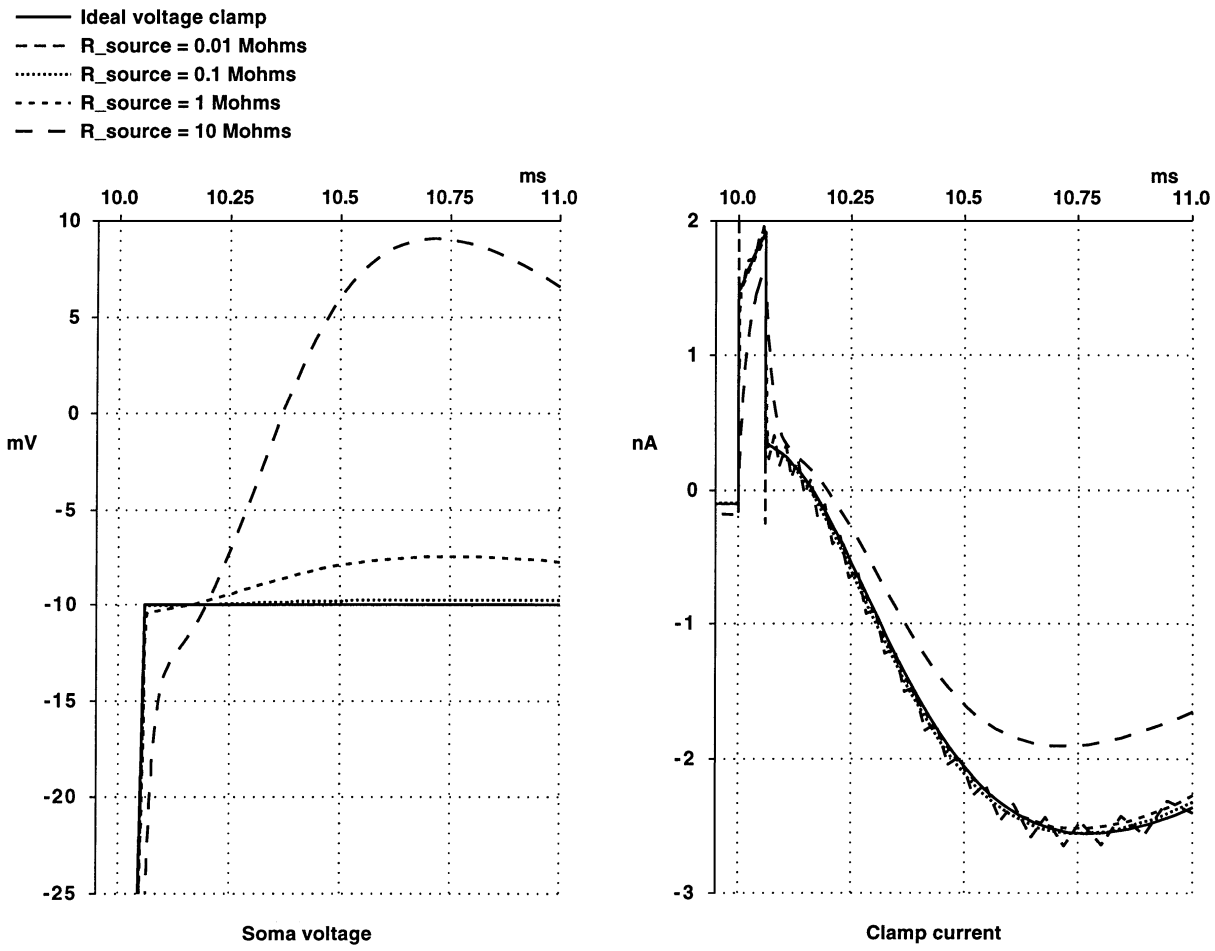

Figure 11. Voltages and currents under ideal and nonideal voltage clamp, for the voltage step for −70 to −10 mV, with an adaptive time step. The somatic voltage shown on the left in the nonideal case for $R_{Source} = 0.01$ MΩ is indistinguishable from the ideal case; however, this small value gives a small oscillation in the current record shown on the right.

In Surf-Hippo the electrical model for a segment is single-ended approximation to the cable section, as follows:

```
          g-axial
Prox o---/\/\/\-----+---o Distal
                    |
                memb-elements
                    |
                   Gnd
```

In addition, the soma circuit is simply a parallel RC. However, the Rallpack 3 specification assumes center-tapped approximation to cable sections for the entire circuit, which implies that the end nodes are half-versions of the remaining nodes. With a total length of 1,000 microns and 9 middle compartments, in the Surf-Hippo model this is accomplished by defining end compartments (soma and distal segment) with an equivalent cylinderical length of 50 microns. To maintain symmetry, we adjust the axial resistance of the distal segment by a factor of 2:

```
(defun rallpack-3-11 ()
  (let ((distal-segment
         (segment-chain  ; Create a chain of segments originating from the soma.
                         ; SEGMENT-CHAIN returns the last segment, which we assign to
                         ; a local variable DISTAL-SEGMENT for use below.
        (create-soma   ; This makes the soma.
         :cell (create-cell 'AXON :cell-type 'HH-AXON)    ; This makes the cell.
         :diameter (sphere-diameter-from-area (* 50 pi))) ; Diameter => area equals 1/2 segment
                                                          ; area.
        NIL            ; Optional base name for segments - not used here.
        10             ; Number of segments.
        100 1)))       ; Default length and diameter of each segment in microns.

 (element-length distal-segment 50)               ; Shorten the distal segment.
 (element-parameter distal-segment 'RI-COEFF 2)   ; Double its effective Ri.

 ;; Add the Hodgkin-Huxley Na and DR channels to the soma and all the segments, as returned
 ;; from the function CELL-ELEMENTS.

 (create-element (cell-elements) 'NA-HH 'DR-HH)))
```

Now load the circuit definition:

```
(topload 'RALLPACK-3-11)
```

We will now set up the details of a current-clamp simulation. First, make sure that the variables defining the integration method and data plot resolution are set (these are the default values):

```
(setq *use-fixed-step* nil ;
        Enable adaptive time step.
      *absolute-voltage-error* 0.05 ; mV
      *absolute-particle-error*  0.001 ; dimensionless
      *user-max-step* 5.0 ; ms
      *user-min-step* 0.0 ; ms
      *pick-time-step-fudge* 0.8 ; dimensionless
      *save-data-step* 2) ; Save every other time point.
```

Note that `*ABSOLUTE-VOLTAGE-ERROR*` and `*ABSOLUTE-PARTICLE-ERROR*` correspond to ϵ^V_{max} and ϵ^x_{max}, respectively, and `*PICK-TIME-STEP-FUDGE*` corresponds to $\varepsilon_{\Delta t}$. Now, add a current source to `*SOMA*` (this global variable references the last created soma in the circuit):

```
(add-isource *SOMA*)
```

The object-oriented nature of this code allows circuit elements to be referenced in a variety of ways. For example, given the above example, we could reference the soma by its name:

```
(add-isource "AXON-soma")
```

We now add a 0.1 nA pulse from 10 to 50 milliseconds to the current source we just created, referenced by the global variable `*ISOURCE*`:

```
(pulse-list *ISOURCE* '(10 50 0.1))
```

Now enable element plotting:

```
(enable-element-plot *ISOURCE*)     ; Default for current sources is the current.
(enable-element-plot *SOMA*)        ; Default for somas is the voltage.
(enable-element-plot (distal-tips)) ; Default for segments is the voltage.
```

The function `DISTAL-TIPS` returns a list of the distal segments in the circuit—in this case this gives the same segment as referenced by the local variable `DISTAL-SEGMENT` in the `RALLPACK-3-11` function definition. Set the simulation time (milliseconds):

```
(setq *USER-STOP-TIME* 50)
```

Finally, run the simulation:

```
(goferit)
```

Note that other than the setup of the cell anatomy, all of the steps in this code example may be done from the menus and point-and-click histology graphics.

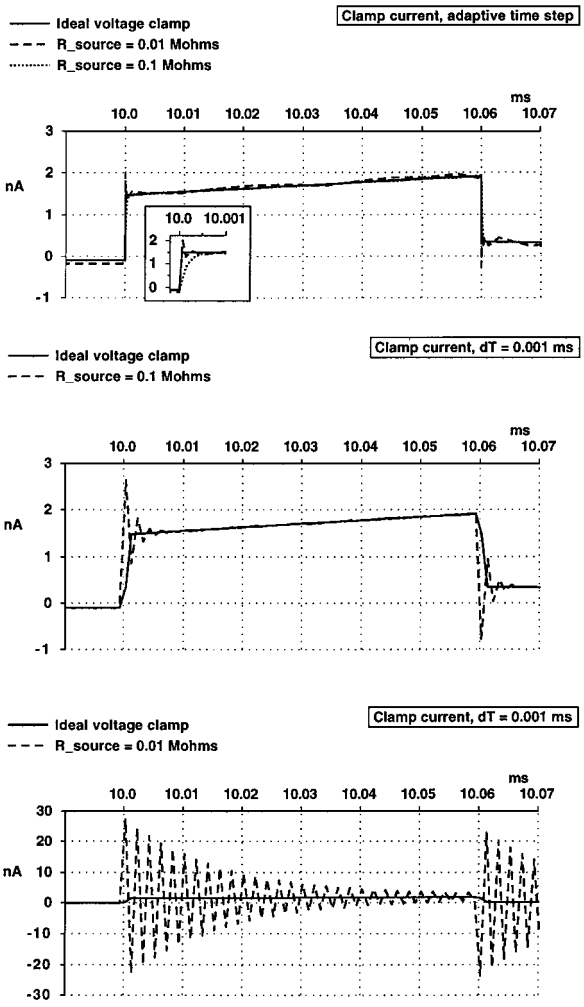

Figure 12. Capacitive transient under ideal and nonideal voltage clamp, both fixed and adaptive time step. The traces at the top are a detailed view of those shown at the right in Fig. 11; note again the oscillations seen when $R_{Source} = 0.01$ MΩ. In the inset at the top it can be seen that the current error for $R_{Source} = 0.1$ MΩ during the transient is small. The sensitivity of the nonideal voltage clamp current to R_{Source} is much more pronounced in the case of fixed time-step integration, as seen in the middle and bottom traces.

Acknowledgments

I would like to acknowledge the helpful comments by Michael Hines and Miki London on the manuscript. This work was supported by HFSP RF0103/1998-B and CNRS.

References

Bhalla US, Bilitch DH, Bower JM (1992) Rallpacks: A set of benchmarks for neuronal simulators. *Trends In Neurosci.* 15(11). Available from ftp://genesis.bbb.caltech.edu/pub/genesis.

Borg-Graham L (1998) The Surf-Hippo neuron simulation system. http://www.cnrs-gif.fr/iaf/iaf9/surf-hippo.html, v2.8.

Bower JM, Beeman D (1994) The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System. TELOS/Springer-Verlag.

Desoer CA, Kuh ES (1969) Basic Circuit Theory. McGraw-Hill.

Hines M (1984) Efficient computation of branched nerve equations. *Int. J. Bio-Med. Comput.* 15:69–76.

Hines M, Carnevale NT (1995) Computer simulation methods for neurons. In: M Arbib, ed. The Handbook of Brain Theory and Neural Networks. MIT Press.

Hines H, Carnevale NT (1997) The NEURON simulation environment. *Neural Comput.* 9:1179–1209.

Hodgkin AL, Huxley AF (1952) A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiol.* 117:500–544.

Mascagni M, Sherman AS (1998) Numerical methods for neuronal modeling. In: C Koch, I Segev, eds. Methods in Neuronal Modeling, Ch. 13. MIT Press/Bradford Books 2nd edition.

Rall W (1964) Theoretical significance of dendritic trees for neuronal input–output relations. In: RF Reiss, eds. Neural Theory and Modelling, Stanford University Press, pp. 73–79.

Segev I, Burke RE, Hines M (1998) Compartmental models of complex neurons. In: C Koch, I Segev, eds. Methods in Neuronal Modeling, MIT Press/Bradford Books, 2nd edition. Ch. 3, pp. 63–96.

Vlach J, Singhal K (1983) Computer Methods for Circuit Analysis and Design. Van Nostrand Reinhold.