Chapter 4

Essentials of numerical methods for neural modeling

NEURON uses many strategies to achieve computational accuracy and efficiency, some of which are detailed elsewhere (Hines 1984). It also draws on several numerical methods libraries for vector (Press et al. 1992) and matrix (Stewart and Leyk 1994) methods, solving systems of sparse equations (Sparse 1.3 (Kundert 1986)), and adaptive integration (CVODES (Hindmarsh and Serban 2002) and IDA (Hindmarsh and Taylor 1999)). These all make their own special contributions to NEURON's performance, but they are already well documented elsewhere so this chapter will not discuss them in any detail. Instead, the emphasis will be on how NEURON deals with the fact that neurons are distributed analog systems that are continuous in time and space, but digital computation is inherently discrete. Because of this fundamental disparity, implementing a model of a neuron with a digital computer raises many purely numerical issues that have no relationship to the biological questions that are of primary interest, yet must be addressed if simulations are to be tractable and produce reliable results.

We saw in **Chapter 3** that the principle of conservation of charge can be expressed with a single ordinary differential equation

$$C \frac{dV}{dt} + I_{ion} = I_{inj} \qquad \text{Eq. 4.1}$$

so long as the transmembrane current density is nearly uniform over the surface of a cell. If current density varies too much, the computational representation must consist of two or more coupled compartments. These are described by a set of equations of the form

$$C_j \frac{dv_j}{dt} + I_{ion_j} = \sum_k \frac{v_k - v_j}{r_{jk}} + I_{inj_j} \qquad \text{Eq. 4.2}$$

where the second term on the right hand side is the sum of all axial currents from neighboring compartments. Additional terms and equations are necessary if extracellular fields (the `extracellular` mechanism) or electronic instrumentation (linear circuits) are to be included in the simulation.

Selection of a method for numerical integration of these equations is guided by concerns of stability, accuracy, and efficiency. In this chapter we review these important concepts and explain the rationale for the integrators used in NEURON. We start with a theoretical analysis of the errors that are introduced by discretizing the linear cable equation. Then we move on to a comparative analysis of methods for computing numerical solutions, which is illustrated by a series of case studies that bring up issues related to the practical concerns of empirically-based modeling.

Spatial and temporal error in discretized cable equations

A linear cable with uniform properties is described by the equation

$$c\frac{\partial V}{dt} + gV = \frac{a}{2R_a}\frac{\partial^2 V}{\partial x^2} \qquad \text{Eq. 4.3}$$

where V is the membrane potential in volts, c the specific membrane capacitance in [F/cm^2], g the specific membrane conductance in [S/cm^2], a the radius in [cm], R_a the cytoplasmic resistivity in [Ω cm], and x the distance along the cable in [cm], so that each term in Eq. 4.3 has units of [A/cm^2]. We assume that the cable is L cm long, and that the axial current at each end is zero, i.e. "sealed end" boundary conditions, which implies that $\partial V / \partial x = 0$ at $x = 0$ and $x = L$. The membrane potential is a function of time and location $V(t,x)$, and the initial condition $V(0,x)$ can be any spatial pattern that satisfies the boundary conditions.

Analytic solutions: continuous in time and space

The spatial patterns that preserve their shape, changing only in amplitude, are the Fourier cosine terms $\cos(\pi nx/L)$. From Fourier theory, we know that any spatial pattern can be represented as an infinite sum of such cosine patterns (Strang 1986).

These cosine patterns always satisfy the boundary condition at x=0 because sin(0)=0. Satisfaction of the boundary condition at $x = L$, i.e. $\sin(\pi n) = 0$, requires that n be an integer. The pattern preserves its shape because substituting $V(t,x) = V_n(t)\cos(\pi nx/L)$ into Eq. 4.3 gives

$$c\frac{dV_n(t)}{dt} + gV_n(t) = -\frac{\pi^2 n^2 a}{2R_a L^2}V_n(t) \qquad \text{Eq. 4.4}$$

which has the solution

$$V_n(t) = V_n(0)\,\mathrm{e}^{-k_n t} \qquad \text{Eq. 4.5a}$$

where n is the number of half waves in the cosine pattern, $V_n(0)$ is its initial amplitude, and the rate of decay is

$$k_n = \frac{g}{c} + \frac{\pi^2 n^2 a}{2R_a L^2 c} \qquad \text{Eq. 4.5b}$$

When $n = 0$, voltage is independent of location along the length of the cable and decays with the membrane time constant $\tau_m = c/g$ seconds (top graph in Fig. 4.1). If n is large, i.e. when the spatial frequency of the cosine pattern is high, the second term on the

right hand side of Eq. 4.5b is dominant, so the pattern decays very quickly at a rate that is proportional to the square of the number of half waves on the cable (see Fig. 4.1, especially the bottom graph). In a continuous cable, there is no limit to the spatial frequency, but high spatial frequencies decay extremely quickly.

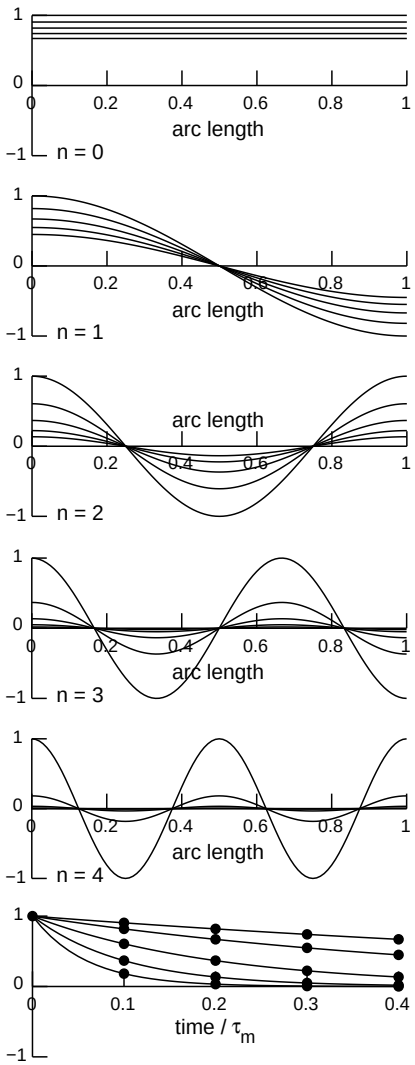

Figure 4.1. Top five graphs: These are the first five spatial patterns of V that preserve their shape along a uniform cylindrical passive cable. V is plotted as a function of normalized distance along the cable for n = 0, 1, 2, 3, and 4 half cycles. The decay of these patterns with time is illustrated by "snapshots" taken at t = 0, 0.1, 0.2, 0.3, and 0.4 times the membrane time constant τ_m. Note that larger n implies faster decay. Bottom graph: Amplitudes of these patterns plotted as functions of normalized time. Starting with the top trace and working down, n = 0, 1, 2, 3, and 4. Dots mark the amplitudes at the times of the snapshots shown in the upper graphs. These amplitudes assume cable length is π times its DC length constant λ, so that n = 1 makes the first and second terms of Eq. 4.5b equal. Shorter cables have bigger k_n, hence decay is more rapid.

Adding a current stimulus to the equations is not difficult, but the detailed derivation is not necessary to our discussion of discretization error. Two points are worth mentioning, however. First, any stimulus can be represented as a Fourier sum. Second, a cosine stimulus with a specific spatial frequency excites a voltage response with the same spatial frequency and an amplitude that follows a single exponential decay, asymptotically approaching a steady state.

Spatial discretization

Now let us compare the continuous cable solution of Eq. 4.5 with the solution of a cable equation that has been discretized in space by replacing $\partial^2 V / \partial x^2$ with the second order correct approximation

$$\frac{\partial^2 V}{\partial x^2} \approx \frac{V(x+\Delta x) - 2V(x) + V(x-\Delta x)}{\Delta x^2} \qquad \text{Eq. 4.6}$$

For concreteness we need to specify precisely which values of x are allowed. The ordinary approach is to suppose m points with the first point at $x = 0$ and the last point at $x = L$, so that $\Delta x = L/(m\text{-}1)$. However, NEURON takes a different approach to discretization, in which there are m intervals of length $\Delta x = L/m$ and the m points are at the centers of these intervals. Thus the centers are at $x = (i + 0.5)L/m$ where $0 \le i < m$.

With either method, m is the number of points in space at which a numerical solution for V is computed, and $m = 1$ corresponds to a spatial frequency of 0, i.e. uniform membrane potential along the entire cable. Furthermore, for either approach the largest number of half waves that can be represented in the discretized system is $n = m\text{-}1$ so the highest spatial frequency is $(m\text{-}1)/2L$ cycles per unit length. This result is related to the Nyquist sampling theorem, which states that at least two samples must be captured per cycle in order to accurately measure the frequency of a signal (Strang 1986).

The ordinary method puts the ith point at $x = iL/(m\text{-}1)$, so $\cos(\pi nx/L) = \cos(\pi(m\text{-}1)iL/(m\text{-}1)L) = \cos(\pi i)$, and the value of V alternates sign at adjacent points. With NEURON's method, the largest n is also m-1 because, at $n = m$, $\cos(\pi nx/L) = \cos(\pi m(m+0.5)L/mL) = \cos(\pi(m+0.5)) = 0$.

With the ordinary method, the second difference at the ith point is most easily computed from the real part of

$$\begin{aligned} & e^{j\pi n(i+1)/(m-1)} - 2e^{j\pi ni/(m-1)} + e^{j\pi n(i-1)/(m-1)} \\ & = \left(e^{j\pi n/(m-1)} - 2 + e^{-j\pi n/(m-1)}\right) e^{j\pi ni/(m-1)} \\ & = 2\left(\cos(\pi n/(m-1)) - 1\right) e^{j\pi ni/(m-1)} \end{aligned} \qquad \text{Eq. 4.7}$$

which is

$$2\left(\cos(\pi n/(m-1)) - 1\right) \cos(\pi ni/(m-1)) \qquad \text{Eq. 4.8}$$

NEURON's method gives

$$2\,(\cos(\pi\, n/m) - 1)\,\cos(\pi\, n(i+0.5)/(m-1)) \qquad \text{Eq. 4.9}$$

Therefore, for either method

$$\frac{d\,V_{nm}}{dt} = -k_{nm}\,V_{nm} \qquad \text{Eq. 4.10}$$

where

$$k_{nm} = \frac{g}{c} + \frac{(1-\cos(\pi\, n\,\Delta\, x/L))\,a}{R_a\, c\,\Delta\, x^2} \qquad \text{Eq. 4.11}$$

The solution of Eq. 4.10 is

$$V_{nm}(t) = V_{nm}(0)\ \mathrm{e}^{-k_{nm}\,t} \qquad \text{Eq. 4.12}$$

Note that k_{nm} approaches k_n (Eq. 4.5b) when $n\Delta x/L$ is << 1 (because $\cos(\phi) \approx 1 - \phi^2/2$ when ϕ is small). This makes sense when one realizes that L/n is half of the wavelength of the spatial pattern, so "$n\Delta x/L$ is small" means that the discretization interval Δx is short compared to the wavelength of the spatial pattern. Thus the discrete system is "sampling" the spatial pattern at an interval that is fine enough to allow a smooth representation of the pattern. Restating this in more formal terms, the discretized system approximates the original continuous system more closely at those spatial frequencies for which the discretization interval Δx is short compared to the spatial wavelength.

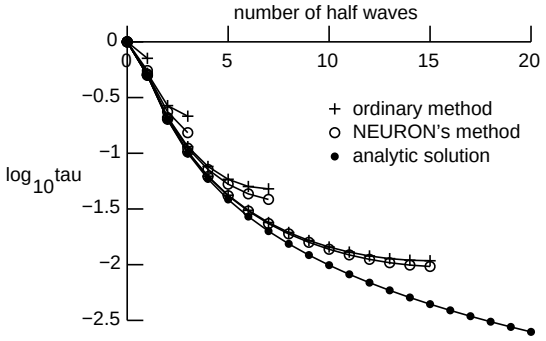

Figure 4.2. Normalized time constant for decay of spatial patterns vs. number of half waves along a uniform passive cylindrical cable (cable parameters as in Fig. 4.1).

Figure 4.2 shows the normalized time constant of decay $\tau = 1/k\tau_m$ as a function of the number of half waves for the continuous cable of Fig. 4.1 as well as for discretized models of this cable with 2, 4, 8, and 16 points. We must point out that, for both discretization methods, doubling the number of points reduces the error in the time constant for a given spatial frequency by a factor of 4. Also note that, for small numbers

of compartments and at the highest spatial frequencies, the spatial error of NEURON's discretization method is significantly less than that of the ordinary method.

Adding temporal discretization

So far we have solved the spatially continuous and spatially discretized cables analytically with respect to time. Now we complete the discretization with respect to time. The numerical integration methods that have seen the widest use in empirically-based neural modeling are forward Euler, backward Euler, and Crank-Nicholson. Later in this chapter we will examine each of these individually and in more detail. For the purpose of our present theoretical analysis, it is better to treat them all at once by introducing a parameter θ so that

$$\frac{dV}{dt} \approx \frac{V(t+\Delta t) - V(t)}{\Delta t} \qquad \text{Eq. 4.13a}$$

is evaluated at $t+\theta\Delta t$ by using V interpolated from its values at t and $t+\theta\Delta t$, i.e.

$$V(t+\theta\Delta t) = (1-\theta)V(t) + \theta V(t+\Delta t) \qquad \text{Eq. 4.13b}$$

Thus Eq. 4.10 becomes

$$\frac{V(t+\Delta t) - V(t)}{\Delta t} = -k_{nm} V(t+\theta\Delta t) \qquad \text{Eq. 4.14}$$

Drawing on Eq. 4.13b, we can write this as

$$\frac{V(t+\Delta t) - V(t)}{\Delta t} = -k_{nm}\left((1-\theta)V(t) + \theta V(t+\Delta t)\right) \qquad \text{Eq. 4.15}$$

When θ = 0 Eq. 4.15 is the forward Euler method, θ = 1 turns it into the backward Euler method, and θ = 0.5 gives us the Crank-Nicholson method.

From Eq. 4.15 we immediately get the iteration equation

$$V_{nm}(t+\Delta t) = \left(\frac{1-(1-\theta)k_{nm}\Delta t}{1+\theta k_{nm}\Delta t}\right) V_{nm}(t) \qquad \text{Eq. 4.16}$$

The first term on the right hand side of this equation is the iteration coefficient; if its magnitude for any spatial frequency is > 1, the iterations will diverge. With the forward Euler method (θ = 0), the iteration coefficient with the largest magnitude is for the spatial frequency at which $n = m$. At this frequency, the $\cos(\pi n\Delta x/L)$ term in Eq. 4.11 is -1, making the decay rate constant

$$k_{mm} = \frac{g}{c} + \frac{2a}{R_a c \Delta x^2} \qquad \text{Eq. 4.17}$$

so we see that the magnitude of the iteration coefficient is > 1 when $k_{mm}\Delta t > 2$. If we want the discretized system to represent high spatial frequencies, Δx must be small, and this makes the second term in k_{mm} dominant. Substituting $\theta = 0$ and $k_{mm} \approx 2a/R_a c\Delta x^2$ into Eq. 4.16 and rearranging, we find that, for the forward Euler method to avoid numerical instability, the combination of Δt and Δx must obey the constraint

$$\frac{\Delta t}{\Delta x^2} < \frac{R_a c}{a} \qquad \text{Eq. 4.18}$$

With the backward Euler method ($\theta = 1$), there is no constraint on Δt because k_{nm} is always positive and so the iteration coefficient is greater than 0 and less than 1. For the Crank-Nicholson method ($\theta = 0.5$), the iteration coefficient never becomes less than -1, so this method is formally stable for all Δt.

Numerical integration methods

Now we continue our comparative analysis of numerical methods for integrating Eq. 4.1 and 4.2 by examining them in the context of practical examples. We start with the simplest approach: explicit or forward Euler, which is *not* used in NEURON for reasons that will become clear. Then we consider the implicit or backward Euler method, Crank-Nicholson, CVODE, and DASPK, which are all available in NEURON.

Forward Euler: simple, inaccurate and unstable

Suppose we are modeling a neuron that has nearly uniform transmembrane current density. For our conceptual model of this cell, we also assume that its resting potential is 0 mV, its membrane conductance g is constant and linear, and that we are not injecting any current into it. The techniques we use to understand and control error in simulations of this linear, passive model are immediately generalizable to active and nonlinear cases.

Conservation of charge in this model is described by Eq. 4.1, which simplifies to

$$\frac{dV}{dt} + kV = 0 \qquad \text{Eq. 4.19}$$

where the rate constant k is the inverse of the membrane time constant $\tau_m = g/c$. The analytic solution of Eq. 4.19 is

$$V(t) = V(0)\mathrm{e}^{-kt} \qquad \text{Eq. 4.20}$$

Let us compare this to a numeric solution computed with the forward Euler method.

The forward Euler method is based on a simple approximation. From the initial conditions we know the starting value of the dependent variable ($V(0)$), and the differential equation that describes the model (Eq. 4.19) gives us the initial slope of the solution ($-kV(0)$). The approximation assumes that the slope of the solution is constant for

a short period of time. Then we can extrapolate from the value of V at time 0 to a new value a brief interval into the future. Now we see why this is called the "forward" Euler method: we are starting from something that is already known and projecting into the future. The forward Euler method is one of many integrators that calculate future values entirely on the basis of present, and possibly also past, values; these are called "explicit" integrators to distinguish them from "implicit" integrators, such as backward Euler and Crank-Nicholson (see below), which involve future values in the calculation.

In general terms, if a system is described by the differential equation

$$\frac{dV}{dt} = f(V, t) \qquad \text{Eq. 4.21}$$

then the forward Euler method approximates a solution by repeatedly applying

$$V(t + \Delta t) = V(t) + f(V(t), t)\,\Delta t \qquad \text{Eq. 4.22}$$

For this example, Eq. 4.22 becomes

$$V(t + \Delta t) = V(t) - k\,V(t)\,\Delta t \qquad \text{Eq. 4.23}$$

(cf. Eq. 4.16 with $\theta = 0$).

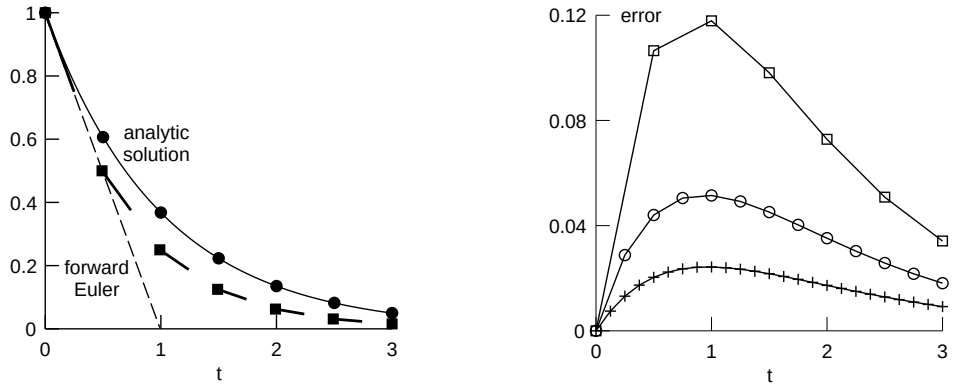

Figure 4.3. Left: analytic solution to Eq. 4.19 (solid line with circles) and results of the forward Euler method (squares) for $V(0) = 1$, $k = 1/s$, and $\Delta t = 0.5$ s (modified from (Hines and Carnevale 1997)). Right: absolute error of the forward Euler method with $\Delta t = 0.5$ (squares), 0.25 (circles), and 0.125 s (+).

The left panel of Fig. 4.3 shows the forward Euler solution obtained for rate parameter $k = 1/s$ (i.e. 1/second), initial condition $V(0) = 1$, and time interval Δt over which we extrapolate, assuming the transmembrane ionic current is constant within each interval. The current that is used for a given interval is found from the value of the voltage at the beginning of the interval (filled squares). This current determines the slope of the line segment that leads to the voltage at the next time step. The dashed line shows the value of the voltage after the first time step as a function of Δt. Corresponding values for the analytic solution (solid line) are indicated by filled circles.

The issue of accuracy in numerical simulation is complex, and we discuss it more thoroughly later in this chapter (see **Error**). For the moment we only mention that the forward Euler method has "first order accuracy," which means that the local error is proportional to Δt. This is demonstrated in the right panel of Fig. 4.3, where the absolute difference between the analytic solution and the results of the forward Euler method is plotted for Δt = 0.5, 0.25, and 0.125 s (squares, circles, and +, respectively). Cutting Δt by a factor of 2 reduced error by very nearly half (Δt was comparable to the model's time constant (1 s) so slight deviations from strict proportionality are to be expected).

Numerical instability

We have already broached this topic from a theoretical standpoint in the setting of a uniform cable model (see **Adding temporal discretization** above), but it is still useful to consider stability of numerical integration in the context of "simpler" compartmental models. What would happen if the forward Euler method were applied to Eq. 4.19 using a very large time step, e.g. Δt = 3 s? The simulation would become numerically unstable, with the first step extrapolating down to V = -2, the second step going to V = -2 + 6 = 4, and each successive step oscillating with geometrically increasing magnitude.

Simulations of the two compartment model on the left of Fig. 4.4 demonstrate an important aspect of instability. Suppose the initial condition is V = 0 in one compartment and V = 2 in the other. According to the analytic solution, at first the potentials in the two compartments converge rapidly toward each other (time constant = 1/41 s), and later they decay slowly toward 0 (time constant = 1 s).

If we use the forward Euler method with Δt = 0.5 s, we realize that there will be a great deal of trouble during the time where the voltages are changing rapidly. We might imagine that we can deal with this by choosing a Δt that will carefully follow the time course of the voltage changes, i.e. let Δt be small when they are changing rapidly, and larger when they are changing slowly.

The results of this strategy are shown on the right of Fig. 4.4. After 0.2 s with Δt = 0.001 s, the two voltages have nearly come into equilibrium. Then we changed Δt to 0.2 s, which is small enough to follow the slow decay closely. Unfortunately, no matter how small the difference between the voltages, the difference grows geometrically at each time step. This happens even if the difference consists only of roundoff error, because the time step used in the forward Euler method must never be more than twice the smallest time constant in the system.

Linear algebra clarifies the notion of "time constant" and its relationship to stability. For a linear system with N compartments, there are exactly N spatial patterns of voltage over all compartments such that only the amplitude of the pattern changes with time, while the shape of the pattern is preserved. The amplitudes of these patterns or "eigenfunctions" are given by $e^{t\lambda_i}$, where λ_i is called the eigenvalue of the ith eigenfunction. The real part of each eigenvalue is the reciprocal of one of the time constants of the solutions to the differential equations that describe the system. The ith

pattern decays exponentially to 0 if the real part of λ_i is negative; if the real part is positive, the amplitude grows catastrophically. If λ_i has an imaginary component, the pattern oscillates with frequency $\omega_i = \mathrm{Im}(\lambda_i)$. In a passive electrical system that contains only resistance and capacitance, all λ_i are real and negative.

Earlier in this chapter (see **Spatial and temporal error in discretized cable equations**) we saw that the eigenfunctions for a uniform cylindrical cable with sealed ends took the form of cosine waves. The decay rates k_n and k_{nm} of that theoretical discussion equal -1 times the corresponding eigenvalues.

Our two compartment model has two such patterns. In one, the voltages in the two compartments are identical; this pattern decays with the time course e^{-t}. The other pattern, in which the voltages in the two compartments are equal but have opposite sign, decays with the much faster time course e^{-41t}.

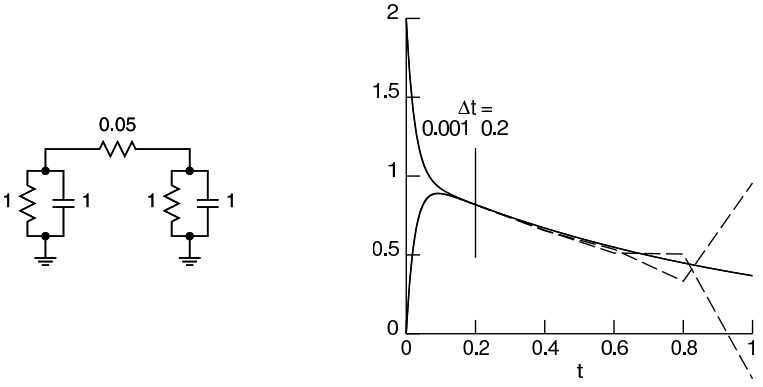

Figure 4.4. Left: The two compartments of this model are connected by a small axial resistance, so the membrane potentials are normally in quasi-equilibrium with each other while at the same time decaying fairly slowly toward 0. Right: The forward Euler method (dashed lines) is numerically unstable whenever Δt is greater than twice the smallest time constant. The analytic solution (solid lines) is the sum of two exponentials with time constants 1 s and 1/41 s. The solution step size was 0.001 s for the first 0.2 s, after which it increased to 0.2 s. Modified from (Hines and Carnevale 1997).

The key idea is that a problem involving N coupled differential equations can always be transformed into a set of N independent equations, each of which is solved separately. Numerical solution of these equations must use a time step Δt that is small enough for the solution of each equation to be stable. This why stability criteria that involve Δt depend on the smallest time constant.

If the ratio between the slowest and fastest time constants is large, the system is said to be stiff. Stiffness is a serious problem because a simulation may have to run for a very long time in order to show changes governed by the slow time constant, yet a small Δt has to be used to follow changes due to the fast time constant.

Signal sources may change the stability properties of a system by altering the time constants that describe it. A current source (perfect current clamp) does not affect stability because it does not affect the time constants. Any other signal source imposes a load on the compartment to which it is attached, changing the time constants and the corresponding eigenfunctions. The more closely it approximates a voltage source (perfect voltage clamp), the greater this effect will be.

Backward Euler: inaccurate but stable

The numerical stability problems of the forward Euler method can be avoided if the equations are evaluated at time $t + \Delta t$, i.e. the approximate solution is found from

$$V(t + \Delta t) = V(t) + f(V(t + \Delta t), t + \Delta t)\, \Delta t \qquad \text{Eq. 4.24}$$

which is called the implicit or backward Euler method. This equation can be derived from Taylor's series truncated at the Δt term but with $t + \Delta t$ in place of t.

For our example with one compartment, the backward Euler method gives

$$V(t + \Delta t) = \frac{V(t)}{1 + k\,\Delta t} \qquad \text{Eq. 4.25}$$

(cf. Eq. 4.16 with $\theta = 1$). Figure 4.5 shows several iterations of Eq. 4.25. Each step moves to a new point $(t_{i+1}, V(t_{i+1}))$ such that the slope there points back to the previous point $(t_i, V(t_i))$. If Δt is very large, the solution does not oscillate with geometrically increasing amplitude like the forward Euler method, but instead converges geometrically toward the steady state.

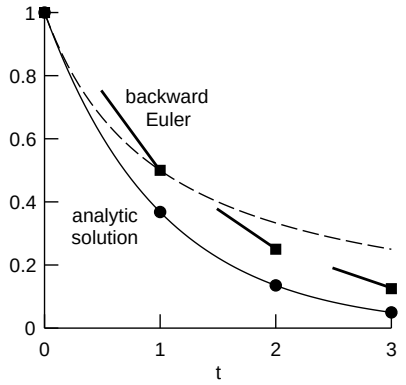

Figure 4.5. Comparison of analytic solution to Eq. 4.19 (solid line with circles) with results from the backward Euler method (Eq. 4.25, squares) for $V(0) = 1$, $k = 1/s$, and $\Delta t = 1$ s. At the end of each step, the slope at the new value (heavy lines) points back to the beginning of the step. The dashed line shows the voltage after the first time step as a function of Δt. Modified from (Hines and Carnevale 1997).

The robust stability of the backward Euler method are readily demonstrated by applying it to the two compartment model (Fig. 4.6). Notice that a large Δt gives a

reasonable qualitative understanding of model behavior, even though the solution does not follow the early rapid voltage changes. Furthermore the step size can be changed according to how quickly the state variables are changing, yet the solution remains stable.

The backward Euler method requires solution of a set of nonlinear simultaneous equations at each step. To compensate for this extra work, the step size needs to be as large as possible while preserving good quantitative accuracy. Like the forward Euler method, backward Euler has first order accuracy (see **Error** below), but it is more practical for initial exploratory simulations since reasonable values of Δt produce fast simulations that are almost always qualitatively correct, and, as we have seen here, tightly coupled compartments do not generate large error oscillations but instead come quickly into equilibrium because of its excellent stability.

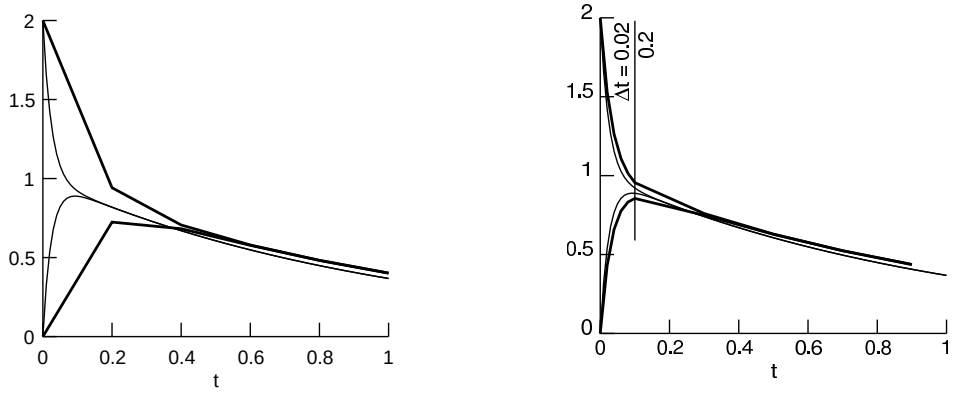

Figure 4.6. Simulation of the two compartment model of Fig. 4.4 using the backward Euler method. Left: Δt = 0.2 s, much larger than the fast time constant. Right: Δt was initially 0.02 s, small enough to follow the first time constant closely. After 0.1 s, Δt increased to 0.2 s but the solution remained stable. Thin lines are analytic solution, thick lines are backward Euler solution. Modified from (Hines and Carnevale 1997).

Crank-Nicholson: stable and more accurate

The central difference or Crank-Nicholson method (Crank and Nicholson 1947) is an implicit integrator that is equivalent to advancing by one half step with backward Euler and then advancing by another half step with forward Euler (Fig. 4.7). The value at the end of each step is along a line determined by the estimated slope at the midpoint of the step. The local error of this method is proportional to the square of the step size, so for a given Δt we can expect a large accuracy increase. In fact, simulation of our one compartment model with Δt = 1 s (Fig. 4.7) is much more accurate than the forward Euler simulation with Δt = 0.5 s (Fig. 4.3).

A most convenient feature of the central difference method is that the amount of computational work for the extra accuracy beyond the backward Euler method is trivial, since after computing $V(t + \Delta t/2)$ with backward Euler, we just have

$$V(t + \Delta t) = 2V(t + \frac{\Delta t}{2}) - V(t) \qquad \text{Eq. 4.26}$$

so the extra accuracy does not cost extra computations of the model functions.

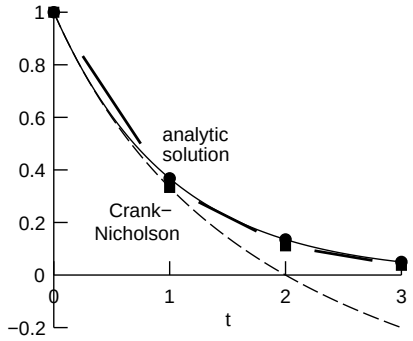

Figure 4.7. Simulations of the one compartment model with the Crank-Nicholson method, which uses the slope at the midpoint of the step (short thick lines) to determine the new value (squares). These are almost indistinguishable from the analytic solution (solid line with circles). The dashed line shows the voltage after the first time step as a function of Δt. Modified from (Hines and Carnevale 1997).

One might well ask what effect the forward Euler half step has on numerical stability. The left panel in Fig. 4.8 shows the solution for the two compartment model of Fig. 4.4 computed using the central difference method with Δt much larger than the fast time constant. The sequence of a backward Euler half step followed by a forward Euler half step approximates an exponential decay by

$$V(t + \Delta t) = V(t)\,\frac{1 - 0.5k\,\Delta t}{1 + 0.5k\,\Delta t} \qquad \text{Eq. 4.27}$$

(cf. Eq. 4.16 with $\theta = 0.5$). As Δt becomes very large, the step multiplier approaches -1 from above, so the solution oscillates with decreasing amplitude. Technically speaking the Crank-Nicholson method is stable because the error oscillations decay with time.

To prevent oscillations in the numeric solution for a model of a cylindrical cable, the normalized increments in time ($\Delta T = \Delta t / \tau_m$) and space ($\Delta X = \Delta x/\lambda$, where Δx is the distance between adjacent nodes and λ is the DC length constant) must satisfy the relationship $\Delta T/\Delta X \leq 1/2$ (see chapter 8 in Crank (1979)).

This example demonstrates that artifactual large amplitude oscillations may result if the time step is too large. Such oscillations can affect simulations of models that involve voltage clamps or in which very small resistances couple adjacent segments. However, in some cases oscillations can be minimized by using small Δt while the solution contains a large amplitude component that is changing rapidly, and increasing Δt after the slower components dominate the solution (Fig. 4.8 right).

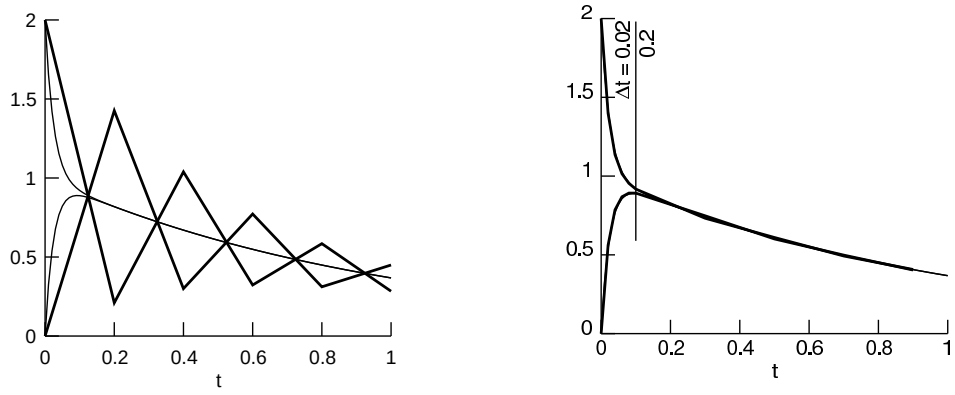

Figure 4.8. Simulations of the two compartment model using the Crank-Nicholson method. Left: Significant error oscillations can appear when the simulation has a large amplitude component with a time constant much smaller than Δt. However, the simulation is numerically stable because the oscillation amplitude decreases at each step. Right: Δt was initially 0.02 s, i.e. smaller than the fastest time constant (~ 0.0244 s), so the simulation followed the rapid collapse of the fast component. After 0.1 s, Δt increased to 0.2 s; this provoked oscillations, but their amplitude is only a small fraction of the total response and decays rapidly, so the trajectories appear smooth. Thin lines are analytic solution, thick lines are Crank-Nicholson solution. Modified from (Hines and Carnevale 1997).

Efficient handling of nonlinearity

Nonlinear equations generally need to be solved iteratively to maintain second order accuracy. However, voltage-dependent membrane properties, which are typically formulated in analogy to Hodgkin-Huxley (HH) type channels, allow the cable equation to be cast in a linear form that can be solved without iterations yet is still second order correct. A direct solution of the voltage equations at each time step $t \rightarrow t + \Delta t$ using the linearized membrane current $I(V,t) = g\,(V - E)$ is sufficient as long as the slope conductance g and the effective reversal potential E are known to second order at time $t + 0.5\Delta t$. HH type channels are easy to solve at $t + 0.5\Delta t$ since the conductance is a function of state variables that can be computed using a separate time step offset by $0.5\Delta t$ with respect to the voltage equation time step. That is, to integrate a state from $t - 0.5\Delta t$ to $t + 0.5\Delta t$ we only require a second order correct value for the voltage-dependent rates at the midpoint time t.

Figures 4.9 and 10 illustrate the differences between the unstaggered and staggered time step approaches. The left panel of Fig. 4.9 shows membrane potential *v* and the gating variable *m* from an action potential simulation computed with the ordinary, i.e. unstaggered, implementation of the Crank-Nicholson method. The superior accuracy achieved with staggered time steps is apparent in Fig. 4.10. The middle panels of these two figures zoom in on the solutions between 2.0 and 2.2 ms to reveal the sequence of calculations. The right panels demonstrate that using staggered time steps turns a system of differential equations with nonlinear coupling into a linear system of decoupled equations, so that second order accuracy is achieved without having to resort to iterations.

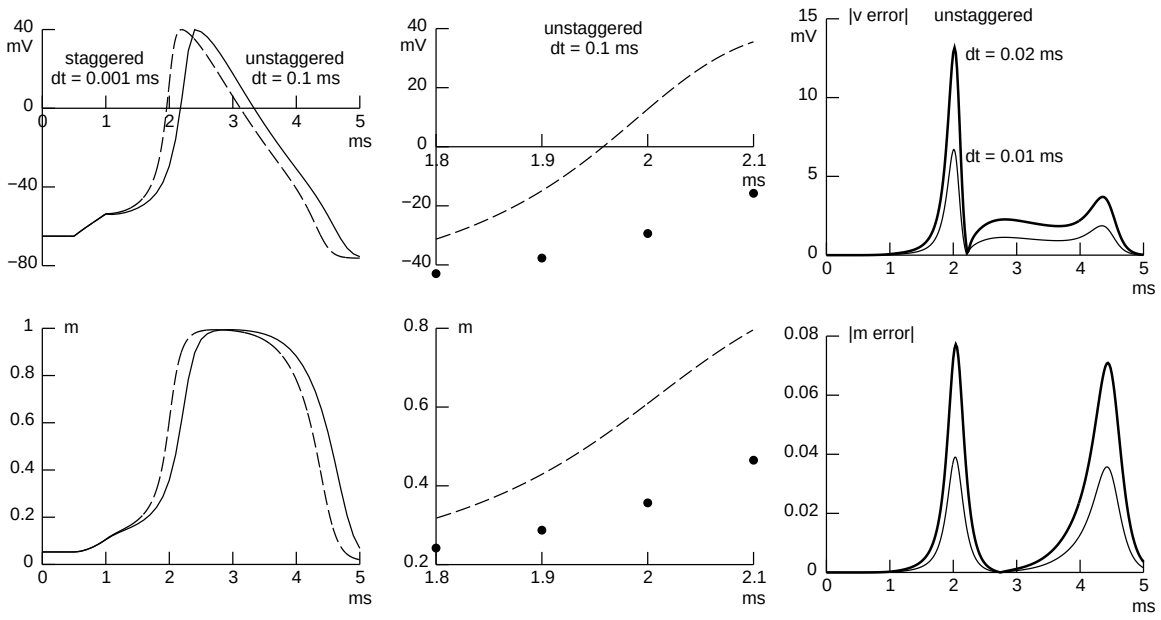

Figure 4.9. Simulated response of a 100 μm^2 patch of membrane with HH channels to a 0.025 nA current lasting 0.5 ms, computed with the ordinary (unstaggered) Crank-Nicholson method using time step Δt = 0.1 ms. Left: The spike was noticeably delayed compared to the standard for accuracy (dashed traces, computed with Crank-Nicholson using staggered time steps and Δt = 0.001 ms). Similar errors were observed in *h* and *n* (traces omitted for clarity). Middle: A magnified view of these solutions from 2.0 to 2.2 ms. Dots mark the individual values computed by the unstaggered Crank-Nicholson method. The unstaggered method advances the solution in two stages. First the new membrane potential $v(t + \Delta t)$ is computed from the values of *v*, *m*, *h*, and *n* at *t*. Then the new values of *m*, *h*, and *n* are computed analytically from their values at *t* and the average of the old and new membrane potentials $(v(t) + v(t+\Delta t)) / 2$. Right: The absolute error of *v* and *m* is proportional to the integration time step Δt, i.e. the solution has only first order accuracy.

For HH equations in a single compartment, using staggered time steps converts four simultaneous nonlinear equations at each step to four independent linear equations that have the same order of accuracy. Since the voltage-dependent rates use the voltage at the midpoint of the integration step, integration of channel states can be done analytically with just a single addition and multiplication and two table lookup operations. While this efficient scheme achieves second order accuracy, the tradeoff is that the tables depend on the value of Δt and must be recomputed whenever Δt changes.

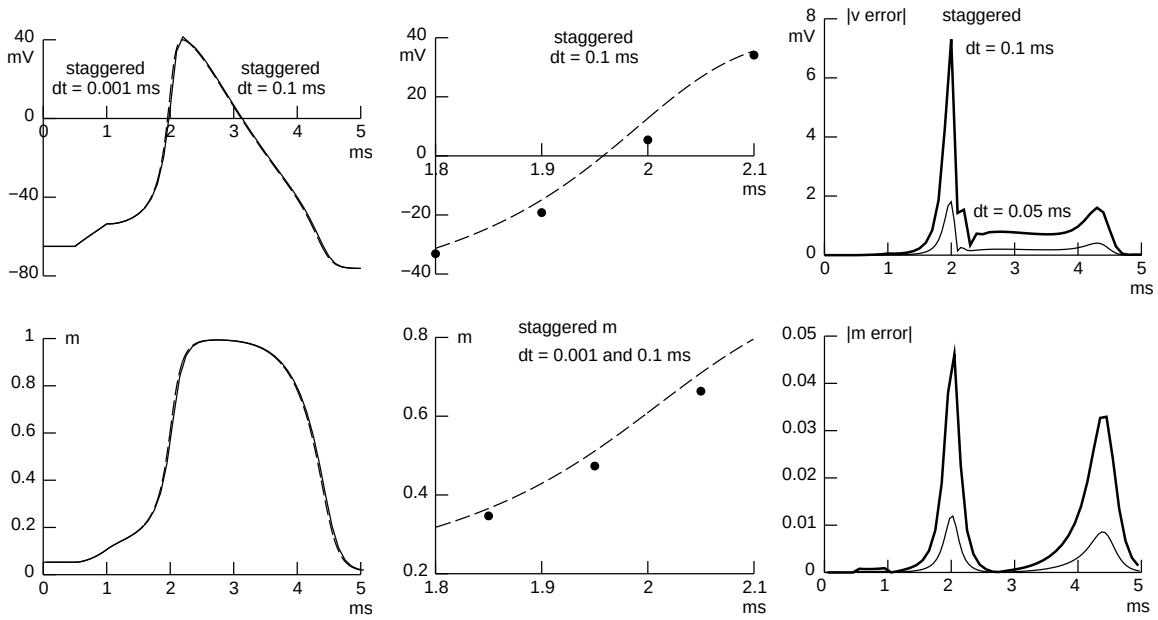

Figure 4.10. Simulated action potential from the same model as in Fig. 4.9, but computed with Crank-Nicholson using staggered time steps. Left: The solution with Δt = 0.1 ms was almost indistinguishable from the standard for accuracy. Similar improvements were observed in *h* and *n*. Right: An expanded view of these solutions, with dots marking the values computed with Δt = 0.1 ms. First the values of *m*, *h*, and *n* at $t + 0.5\Delta t$ are computed analytically from their values at $t - 0.5\Delta t$ and the membrane potential *v* at *t*. Then the values of *m*, *h*, and *n* at $t + 0.5\Delta t$ are used to update *v* from *t* to $t + \Delta t$. Right: Plots of the absolute error of *v* and *m* show that the error is proportional to the square of the integration time step Δt, i.e. using staggered time steps increases solution accuracy to second order.

Adaptive integration: fast or accurate, occasionally both

There is a wide variety of problems for which an adaptive time step method might have much higher performance than a fixed step method, e.g. Δt could grow very large when all states are varying slowly, as during interspike intervals. On the other hand, in problems involving propagating action potentials or networks of cells, it may happen that some state somewhere in the system is always changing quickly. In such cases Δt is always small in order to follow whichever state is varying fastest. Thus it is often not clear in advance that the increased overhead of an adaptive time step method will be repaid by an occasional series of long time steps.

Implementational considerations

The variable order variable time step integrator CVODE was written by Cohen and Hindmarsh (Cohen and Hindmarsh 1994, 1996) to solve ordinary differential equation (ODE) initial value problems of the form

$$\mathbf{y}' = \boldsymbol{f}(\mathbf{y}, t)$$
$$\mathbf{y}(0) = \mathbf{y}_0$$
$$\mathbf{y} \in \mathbb{R}^N$$

Eq. 4.28a-c

where $\mathbf{y}'$ is the first derivative of $\mathbf{y}$ with respect to t, and bold face is used to signify vectors (lower case) and matrices (upper case). Since there are many different adaptive integrators, it is worthwhile to review the reasons why CVODE is particularly relevant to NEURON.

1. CVODE employs Backward Differentiation Formula (BDF) methods suitable for stiff problems, which are common in neuronal modeling.
2. CVODE was easily interfaced to the existing NEURON structure. It would be neither convenient nor efficient to gather *all* of the equations for every compartment and every membrane mechanism into one huge bag and throw it at a solver. The interface between ODE solver and the definition and setup of equations that are already distributed among membrane mechanisms requires a map between the internal NEURON states and the ODE state vector $\mathbf{y}$, as well as a map between the internal computations for $\boldsymbol{f}$ and the ODE state derivative vector $\mathbf{y}'$. Programming an efficient map between the distributed internal Jacobian ($\mathbf{J} = \partial\boldsymbol{f}/\partial\mathbf{y}$) evaluation and a sparse matrix representation is possible but complex. The CVODE solver obviates this problem since it allows programmers to define their own problem-dependent linear solvers. This means NEURON can exploit the existing block structure of the Jacobian matrix and reuse the local block solvers that are already distributed within the membrane mechanism objects.
3. CVODE (and DASPK--see below) allows a sophisticated balance between accuracy of solution of $\mathbf{M}\,\mathbf{y} = \mathbf{b}$ and solution time by supporting the preconditioned iterative Krylov method, which requires one to only supply a solver for $\mathbf{P}\,\mathbf{y} = \mathbf{b}$, where $\mathbf{P}$ is in some sense an approximation to $\mathbf{M}$ such that $\mathbf{P}^{-1}\,\mathbf{M}$ is approximately the identity matrix and is chosen so that computation of the inverse of $\mathbf{P}$ is much faster than computation of the inverse of $\mathbf{M}$. Small off-diagonal elements in the Jacobian are usually ignored for Gaussian elimination efficiency, but can occasionally have an adverse effect on stability and thereby limit the effective time step. It is not yet clear which method is more robust when such off-diagonal terms are ignored in the context of nerve simulations: the Krylov method, or direct use of the approximate Jacobian in CVODE.
4. Finally, CVODE was implemented using encapsulated data structures, so it was conceptually simple to place it in an object-oriented class wrapper for use in implementing a local variable time step method. An important pre-existing feature of CVODE that helped support local variable time steps was the ability to efficiently retreat to any time within the previous integration interval.

Unfortunately, models that contain linear circuits and extracellular fields cannot be expressed, or at least are not easy to express, in the form shown in Eq. 4.28. Such models take the form

$$\mathbf{C}\,\mathbf{y}' = \boldsymbol{f}(\mathbf{y}, t) \qquad \text{Eq. 4.29a-c}$$

$$\mathbf{y}(0) = \mathbf{y}_0$$

$$\mathbf{y} \in \mathbb{R}^N$$

where some rows in the **C** matrix may be 0 (introduction of algebraic equations), and the nonzero rows may have off-diagonal elements (capacitors between nodes). In principle one could use the singular value decomposition of **C** to recast the system as

$$\mathbf{z}' = \boldsymbol{g}(\mathbf{z}, \mathbf{x}, t) \qquad \text{Eq. 4.30a and b}$$

$$\mathbf{0} = \boldsymbol{h}(\mathbf{z}, \mathbf{x}, t)$$

and satisfy the latter constraint directly whenever $\boldsymbol{f}$ is calculated. This is what NEURON does with the zero area nodes at the ends of sections, where membrane potential is governed by an algebraic equation rather than an ODE, without too much trouble and with no loss of efficiency. However, in practice $\boldsymbol{f}(\mathbf{y}, t)$ is given by an algorithm which one cannot multiply by a matrix. Also the sparse structure of $\boldsymbol{f}$ is generally lost in the transformation, making $\boldsymbol{g}$ much more dense and hence less efficient to solve.

For these reasons, when extracellular or linear circuit mechanisms are present and a variable step integration method is requested, the fast CVODE method is replaced by the slower but more robust DASPK method of Brown, Hindmarsh, and Petzold (Brown et al. 1994), which is available from `http://netlib.org`.

The user's perspective

A key feature of using CVODE is that one does not set the integration step size, but instead specifies tolerance criteria for local relative and absolute errors. The solver then adjusts Δt and the local error order of the implicit difference approximation (from first order up to $O(\Delta t^6)$) so that the local error for each state is less than the sum of its relative and absolute error tolerances.

Figure 4.11 illustrates the performance of CVODE in simulations of the two compartment model using two different values for the local absolute error tolerance. CVODE is capable of a high degree of accuracy, but caution must be exercised in setting the error tolerance, and it is a good idea to compare results against fixed time step methods during (and even after) model development.

For a more biologically relevant example of how CVODE can reduce the time necessary to produce accurate simulations, let us compare simulations of a neocortical layer V pyramidal cell model (Mainen and Sejnowski 1996) generated with the Crank-Nicholson and CVODE integrators. The model was subjected to a 900 ms depolarizing current applied to the soma, which evoked two bursts of spikes (Fig. 4.12 top). A series

of simulations was run with the Crank-Nicholson method using progressively smaller Δt until the time at which the last action potential crossed above 0 mV converged to a constant value; this occurred for $\Delta t \leq 0.01$ ms, and a simulation performed with Δt = 0.01 ms took 340 seconds to complete on a 2.2 GHz Pentium 4 PC with 512 K cache. Solutions computed with CVODE converged to the same zero crossing time of the last spike, i.e. same global error, when absolute tolerance was $2.5 \cdot 10^{-3}$ for all states except $[Ca^{2+}]_i$, which had an absolute tolerance of $2.5 \cdot 10^{-7}$; using these tolerances, the solution runtime was 19 seconds. Thus CVODE achieved the same accuracy as the most accurate fixed time step solution, but with a runtime that was almost 20 times faster.

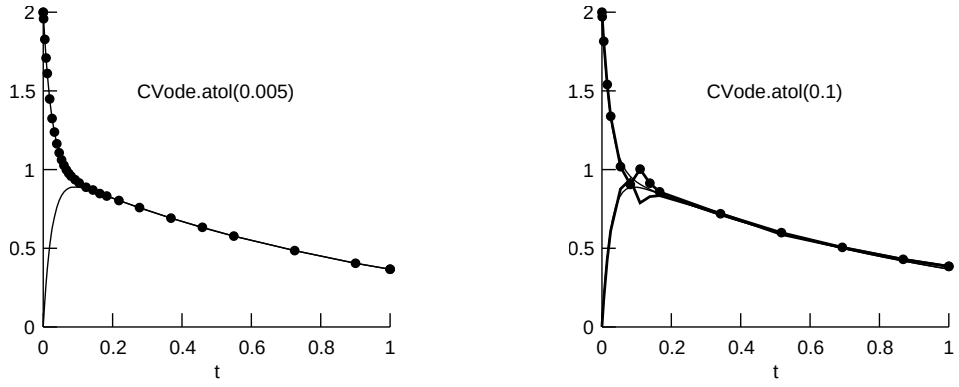

Figure 4.11. Simulations of the two compartment model using CVODE. Left: Filled circles on one of the traces mark the times at which CVODE calculated solutions. When the solution is changing rapidly, Δt is very small, but it grows quite large when the solution changes slowly. If the local absolute error tolerance is sufficiently strict (0.005 for this example), there is no visible difference between the computed and analytic solutions. Right: Thin lines are the analytic solution, thick lines the CVODE solution. Increasing the error tolerance allows CVODE to take larger steps, but spurious transients may occur if the criterion is too lax.

The bottom panel of Fig. 4.12 demonstrates the control that CVODE exerted over Δt throughout the entire simulation. When states were changing most rapidly, Δt fell to values much smaller than 0.01 ms, but during the long interburst interval it increased to a maximum of ~4.4 ms. The smallest steps were restricted to the onset and offset of the injected current (t = 5 and 905 ms) and brief intervals starting just before the threshold and ending shortly after the depolarized peak of each spike, as can be seen in an expanded view of the transition from the interburst interval to the beginning of the second burst (Fig. 4.13). The remarkable speedup by CVODE is due to the fact that Δt was much larger than 0.01 ms for most of the simulation.

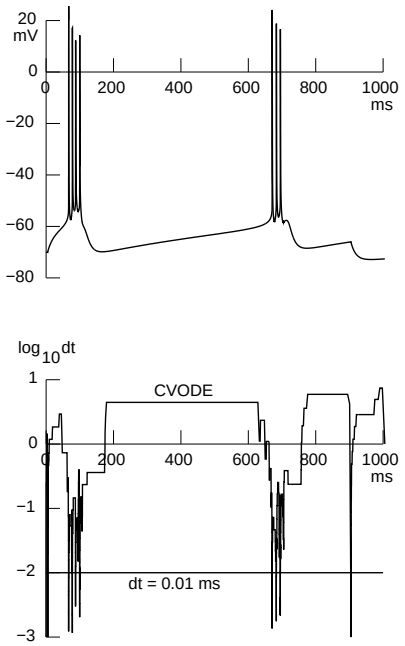

Figure 4.12. Top: CVODE was used to compute somatic membrane potential in a model of a neocortical layer V pyramidal cell subjected to a long depolarizing current pulse; Crank-Nicholson method with Δt = 0.01 ms produced results that are indistinguishable at the scale of this figure. Bottom: For most of the simulation, CVODE used time steps much larger than 0.01 ms. The order of integration (not shown) ranged from 2 to 5, most steps being second or third order. Figure from (Hines and Carnevale 2001).

The only difficulty that CVODE introduced is an excessive literalness required for interpretation of discrete functions. To see what this means, consider this strategy for emulating a "ramp clamp": filling the elements of a `Vector` with a linearly increasing sequence of values and then using the `Vector` class's `play()` method to drive the command potential of a voltage clamp. Figure 4.14 shows this technique applied to a single compartment model with HH currents that was clamped by an `SEClamp` (series resistance `rs` = 10^6 Ω). The elements of a `Vector` were assigned the series of values -65+0.125i for $0 \leq i \leq 401$, i.e. a linear ramp that swept from -65 to -15 mV over the course of 10 ms, assuming Δt = 0.025 ms. A second `Vector` filled with the corresponding times (0.025i) was used to insure that each command potential in the sequence was applied at the proper time.

Simulations of this model using the implicit Euler method with a 0.025 ms time step display smoothly varying membrane potential and clamp current, even when examined at the scale of individual time steps (Fig. 4.14 right). This is because the stream of values delivered by the `Vector` is equivalent to a second order piecewise linear function, i.e. command potential itself varies smoothly with time.

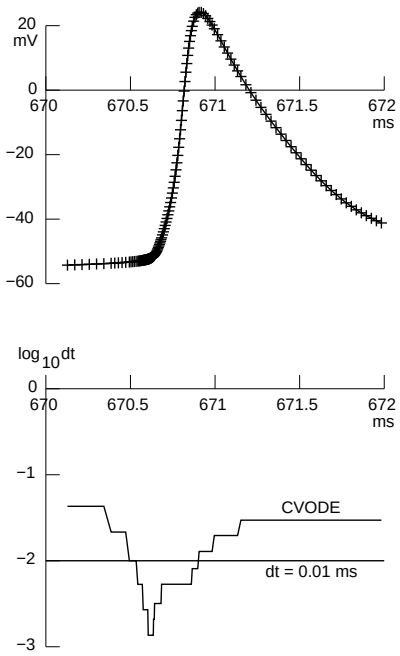

Figure 4.13. Top: An expanded view of the first spike in the second burst from Fig. 4.12. The times of computed solutions are marked by + symbols. Bottom: Δ*t* fell below 0.01 ms from just before the threshold of each spike until shortly after its peak. Figure from (Hines and Carnevale 2001).

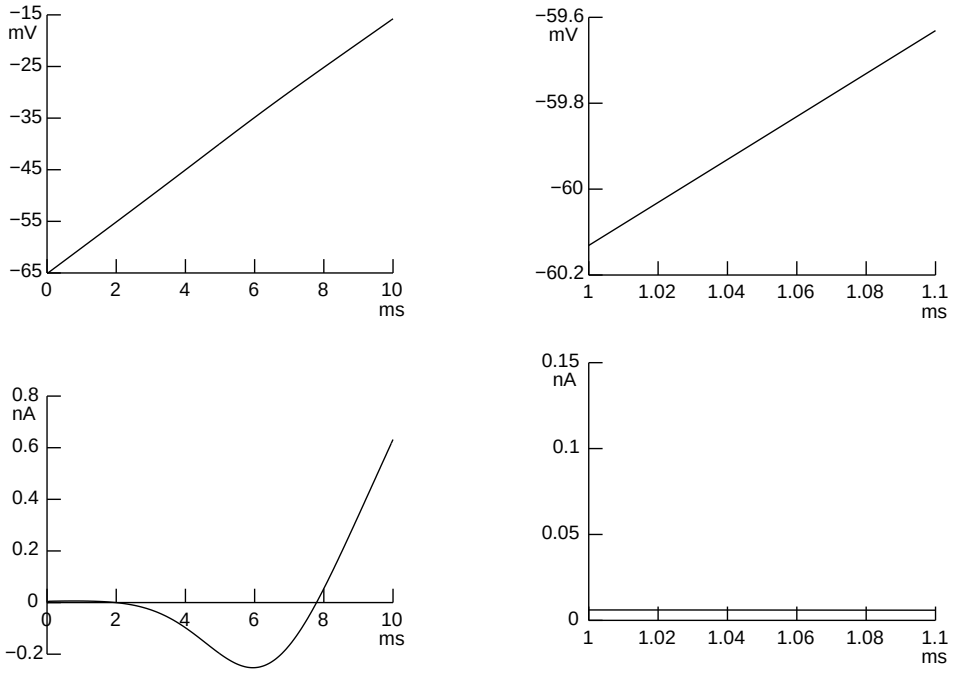

Figure 4.14. Ramp clamp using the `Vector` class's `play()` method works well with fixed Δ*t* integration because command potential is effectively a continuous function of time. Top traces are membrane potential, bottom traces are clamp current.

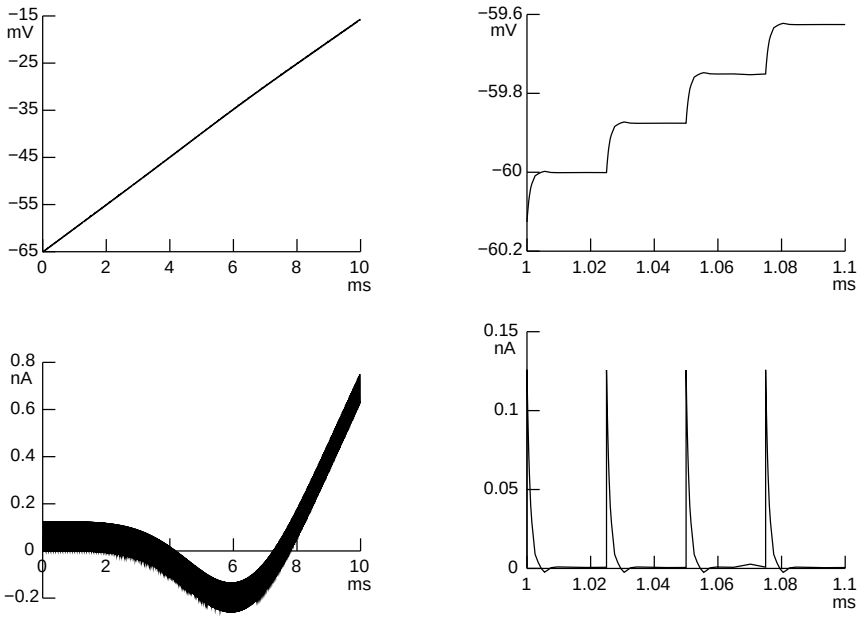

Figure 4.15. Using `Vector.play()` with CVODE produces large capacitive transients in clamp current (bottom traces) because the value sequence in the `Vector` that drives command potential is treated as a first order step function. The local absolute error tolerance parameter `atol` is 0.001 in this simulation and in Fig. 4.16.

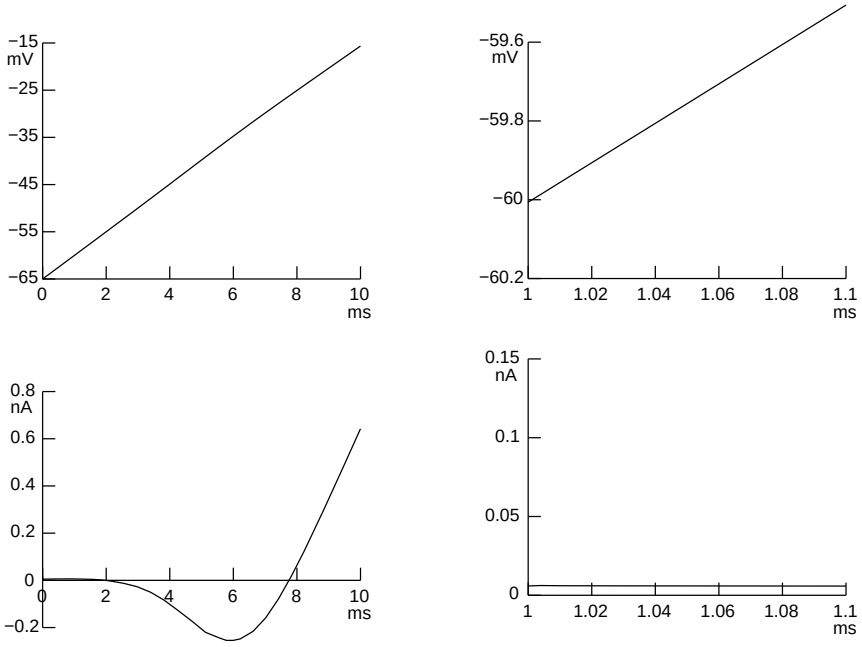

Figure 4.16. `Vector.play()` with interpolation works well with CVODE because the `Vector` that drives command potential is treated as a piecewise linear function. See text for details.

However, under CVODE the stream must be considered a first order equivalent step function. Driving the voltage clamp with this step function makes membrane potential

jump from one level to another and produces substantial capacitance current transients at each step discontinuity (Fig. 4.15).

This problem has been addressed in NEURON 5.4 by adding a linear interpolation option to the `Vector` class's `play()` method. This option, which works both with fixed Δt and CVODE, treats our two vectors as if they defined a piecewise linear function. This means we can represent the ramp command used in this example by a pair of vectors whose elements are -65, -15 and 0, 10, respectively. Simulation results using `Vector.play()` with linear interpolation under CVODE are shown in Fig. 4.16.

In the future, `play()` will be extended to cubic spline and will allow "continuous" play of a smooth function defined by a `Vector`.

Error control

An important issue in adaptive integration is selection of appropriate values for local error control. Variable time steps elevate the issue of "physiological accuracy" (see **Error** below) to a level of high concern. Experience so far suggests that control of local absolute error is much more important than control of local relative error. One *can* specify an error criterion based on local relative error, but in neural modeling there is hardly ever a reason to require increasing absolute accuracy around the 0 value of most states, especially voltage.

The scale of states is often a crucial consideration, in that the maximum absolute error must be consistent with the desired resolution of each state. An extreme example is a calcium pump model with pump density measured in [moles/cm^2]. Here an appropriate value is 10^{-14} [mole/cm^2], and an allowable error of 0.01 is clearly nonsense. For this reason, it is essential that each state that is badly scaled, e.g. $[Ca^{2+}]_i$ measured in [mM], be given its own explicit maximum absolute error. NEURON accommodates this need by allowing the user to set specific error criteria for individual states that take precedence over any global criterion.

NEURON's default error setting for CVODE is 10 μV for membrane potential and 0.1 nM for internal free calcium concentration, so that a simulation of the classical Hodgkin-Huxley action potential at 6.3° C has accuracy comparable to a second order correct simulation with fixed Δt = 25 μs.

Local variable time step method

NEURON provides a network connection (`NetCon`) class for network simulations in which cell to cell communication can be abstractly represented by the (possibly delayed) delivery of logical events, as opposed to graded interaction via gap junctions or electrical synapses (see **Chapter 10**). The notion of a cell driven by discrete input events naturally suggests an expansion of the simulation domain wherein variable time step methods provide substantial performance gains.

It may happen that only a few cells in a network are active at any one time, but with a global time step these active cells govern the time step for all (Fig. 4.17). NEURON's local variable time step method merely uses a separate CVODE solver instance for each

cell, thus integrating that cell's states with time steps governed only by those state dynamics and the discrete input events.

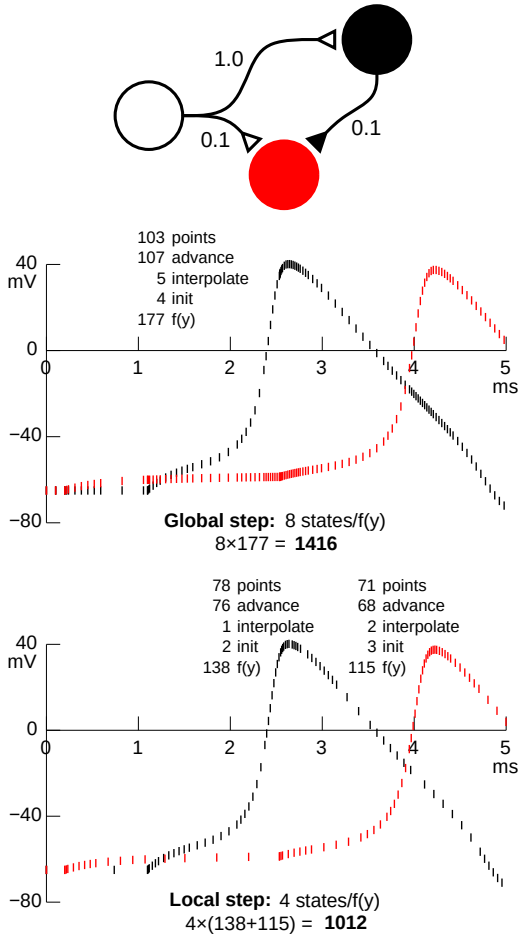

Figure 4.17. Integration with local variable time steps can significantly improve computational efficiency. The top figure shows a simple feedforward network implemented with a `NetStim` artificial spiking cell (white) and a pair of single compartment biophysical model neurons with Hodgkin-Huxley membrane (black and red). All synapses are excitatory, with latencies between presynaptic spike and postsynaptic conductance change shown in ms. The white cell produces a single spike at $t = 0$ ms. This triggers a spike in the black cell, but the red cell requires inputs from both synapses to make it fire. The short vertical lines in the middle and bottom figures mark the times at which solutions are computed using the global (middle) and local (bottom) variable time step methods. Note that, if rapid changes occur in any cell (e.g. onset of an epsp, or the upstroke and peak of a spike), the global method forces extra computations in all cells, even those in which nothing much is happening. This does not occur with the local method. The total computational cost of a simulation depends chiefly on the total number of times that new `STATE`s are calculated. The global method evaluated $\boldsymbol{f}(\mathbf{y})$ (see Eq. 4.29a) 177 times, calculating 8 `STATE`s each time (4 `STATE`s per cell), for a total of 1416; the local method required 253 evaluations of $\boldsymbol{f}(\mathbf{y})$, but these were in individual cells so only 4 `STATE`s were calculated each time, and the local method's total was 1012. Therefore the global method was ~1.4 times more costly than the local method.

All cells are always on a list ordered by their current time and all outstanding events are on a list ordered by their delivery time. These lists are implemented as splay trees to minimize insertion and removal times (proportional to the log of the size of the list), and the least time element can be accessed in constant time. The last fact that prepares our arena for action is that a CVODE instance can, without integrating equations, retreat from its current time to any time back to the beginning of its previous time step.

The network simulation advances in time by checking the cell and event lists to find the least time cell or event, whichever is first. If a cell is first, that cell is integrated according to its current time step, and moved to a location on the cell list appropriate to its new time. If an event is first, it is delivered to the proper cell. That cell retreats to the delivery time and becomes the least time cell, and the event is removed from the list and discarded.

It is easy to devise networks in which the speed improvement of the local time step approach is arbitrarily great. e.g. chains of neurons. However, this method yields no benefit in periods of synchronous activity. If events are extremely numerous, neither the local nor the global variable time step method improves simulation speed. When multiple events per reasonable Δt arrive regularly, fixed time step integration nicely aggregates all events in a step without regard to their temporal microstructure, whereas variable step methods' scrupulous handling of each event is out of all proportion to the conceptual approximation of the network.

The choice of methods is thus dependent on the problem and the user's intent. To encourage the exploration that is necessary to determine which method may be best suited for a particular application, NEURON allows any of its fixed or variable time step methods to be used with no changes to the user-level specification of the problem.

The local variable time step method considerably increases the complexity of the underlying communication between interpreter and solver with respect to recording results. With a global time step, whether fixed or variable, the `fadvance()` function (see **Chapter 7**) has a clear and precise meaning, i.e. the exit time differs from the entry time by the interval Δt. The problem is that, with the local variable time step, each cell has its own time stream, so each recorded variable must be mapped to the appropriate time stream. This problem is solved by the `CVode` class's `record()`, which records both a variable and its associated times into a pair of `Vectors`.

Discrete event simulations

One limiting case of the variable step simulation style is the "event-driven" or discrete event simulation, in which cells jump from event to event. Here a single compartment is used merely as a stage in which the voltage never changes (the natural time step is infinite), and the "cells" are represented by point processes that receive events from, and produce events to, the `NetCon` instances. A large variety of useful artificial spiking cells (e.g. integrate and fire, firing frequency dependent on input), as well as mechanisms of use-dependent synaptic plasticity, are susceptible to discrete event simulation because their equations can be solved analytically, so that "cell" state needs only to be computed at the event. This topic is discussed more thoroughly in **Chapter 10**.

Error

The total or global error in a simulation is a combination of errors from two sources. The local error emerges from the extrapolation process within a time step. For the backward Euler method this is easily analyzed with Taylor's series truncated at the term proportional to Δt.

$$V(t+\Delta t) = V(t) + V'(t+\Delta t)\,\Delta t - V''(t^*)\,\frac{\Delta t^2}{2} \qquad \text{Eq. 4.31}$$

where $t \le t^* \le t+\Delta t$.

The forward and backward Euler methods both ignore second and higher order terms, so the error at each step is proportional to Δt^2. Integrating over a time interval T requires $T/\Delta t$ steps, so the error that accumulates in this interval is on the order of $\Delta t^2 T/\Delta t$, i.e. the local error for the Euler methods is proportional to Δt. Applying a similar analysis to the Crank-Nicholson method finds that its local error is proportional to Δt^2. Therefore we can always decrease the local error of these fixed step methods as much as we like by reducing Δt.

The second contribution to total error comes from the cumulative effect of past errors, which have moved the computed solution away from the trajectory of the analytic solution. Thus, if our computer solution has a nonzero total error at time t_1, then even if we could solve the equations *exactly* from that time forward using the state values at t_1 as our initial condition, the future solution will be inaccurate because we are on a different trajectory. This means that the second component of total error depends on the dynamics of the system itself.

The total error of a simulation is therefore not easy to analyze. For the one and two compartment models we have examined in this chapter, all trajectories end up at the same steady state, so total error tends to decrease with time, but not all systems behave like this. Particularly treacherous are systems with chaotic behavior, in which, once the computed solution diverges even slightly from the proper trajectory, it subsequently moves rapidly away from the original and the time evolution becomes totally different.

Chaos is not the only circumstance that may produce high sensitivity to numerical error. Consider the Hodgkin-Huxley membrane action potentials elicited by two current stimuli, one near threshold and the other twice as strong. The left panel of Fig. 4.18 shows action potentials computed with the backward Euler method using time steps of 25 and 5 μs, the Crank-Nicholson method using Δt = 25 μs, and CVODE using local absolute error tolerance = 0.01. For the strong stimulus, all three integration methods produced nearly identical results. However, the backward Euler method displayed a noticeable error when the 25 μs time step was used to compute the response to the weak stimulus (dashed line). The weak stimulus allowed membrane potential to hover near spike threshold, so that a small error due to the time step could grow into a large error in

the time of occurrence of the action potential. The error was much smaller in the simulation computed with Δt = 5 μs.

However, behavior near threshold is highly sensitive to almost any factor, be it a parameter of the numerical integration method (e.g. Δt or Δx) or a parameter of the model itself. This is seen in the right panel of Fig. 4.18, where all solutions were computed with CVODE (local absolute error tolerance = 0.01) and the sodium channel density $\bar{g}_{Na}$ was varied by only 1%. This small variation of $\bar{g}_{Na}$ did almost nothing to the response to the strong stimulus, but its effect on the latency of the spike elicited by the weak stimulus was comparable to the integration error of the backward Euler method with Δt = 25 μs. This demonstrates that it is important to know the sensitivity of results to every model parameter, and Δt is just one more parameter that is added as a condition of being able to run simulations on a digital computer.

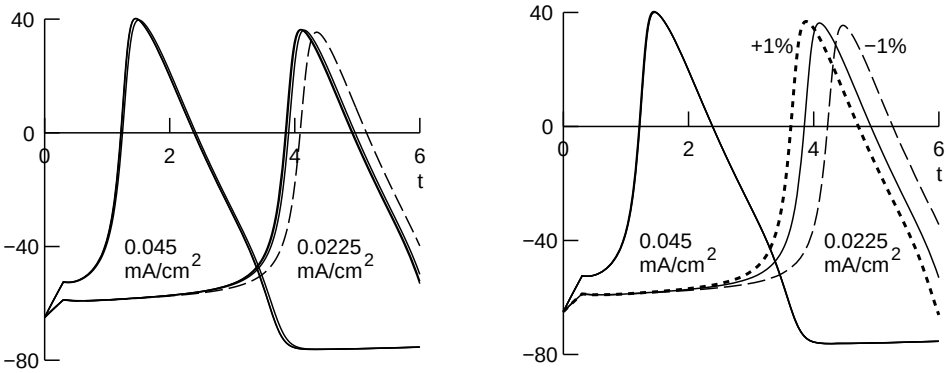

Figure 4.18. Simulations of Hodgkin-Huxley membrane action potentials elicited by 0.3 ms current stimuli with amplitude of 0.0225 or 0.045 mA/cm^2. Left: Sensitivity to integration time step. For each stimulus amplitude, responses were computed using CVODE (local absolute error tolerance = 0.01), Crank-Nicholson (Δt = 25 μs), and backward Euler (Δt = 25 and 5 μs). The backward Euler solution with 25 μs time step showed a noticeable error. Right: Sensitivity to variation in $\bar{g}_{Na}$. All traces were computed with CVODE (local absolute error tolerance = 0.01). Peak sodium conductance was 0.12 S/cm^2 (solid lines) ± 1% (dotted and dashed lines). The three traces elicited with the large stimulus are indistinguishable in this graph.

Using extremely small Δt might seem to be the best way to reduce error. However, computers represent real numbers as floating point numbers with a fixed number of digits, so if you keep adding 10^{-20} to 1 you may always get a value of 1, even after repeating the process 10^{20} times. Operations that involve the difference of similar numbers, as when differences are substituted for derivatives, are especially prone to such roundoff error. Consequently there is a limit to the accuracy improvement that can be achieved by decreasing Δt.

Generally speaking, it would be desirable to use what might be called "physiological" values of Δt, i.e. time steps that give a good representation of the state trajectories without having a numerical accuracy that is orders of magnitude better than the accuracy of our physiological measurements (which is generally not as good as 5%, and seldom better). The question is not so much how large the error of a simulation is relative to the analytic solution, but whether the simulation error leads us to trajectories that are significantly different from the set of trajectories defined by the error in our parameters. Insofar as removal of any source of error has value, there is a temptation to treat the model equations as sacred runes which must be solved to an arbitrarily high precision. Nevertheless, determining the meaning of a simulation run requires judgment. A misplaced emphasis on numerical accuracy should not obscure the fact that qualitative results may be quite sufficient. We agree with John Moore, our mentor and colleague, who is fond of quoting R. Hamming: "The purpose of computing is insight, not numbers" (Hamming 1987).

Summary of NEURON's integration methods

NEURON offers the user a choice of several different integration methods. For any particular problem, the best way to determine which is the method of choice is to run comparison simulations with several values of Δt or local error tolerance to see which executes most quickly while achieving the desired accuracy. In performing such trials, one must remember that the stability properties of a simulation depend on the entire system that is being modeled. Because of interactions between "biological" components and any "nonbiological" elements, such as stimulators or voltage clamps, the time constants of the *entire* system may be different from those of the biological components alone. A current source (perfect current clamp) does not affect stability because it does not change the time constants. Any other signal source imposes a load on the compartment to which it is attached, changing time constants and potentially introducing troublesome stiffness. The more closely a signal source approximates a voltage source (perfect voltage clamp), the greater this effect will be.

Fixed time step integrators

Implicit integrators are used as NEURON's fixed time step methods. This is in part because of their superior stability compared to explicit integrators (Dahlquist and Bjorck 1974).

Default: backward Euler

NEURON's default integration method is backward Euler, a fixed step first order implicit scheme that produces good qualitative results with large time steps when extremely stiff ODEs and even algebraic equations are present in the system, e.g. models that involve voltage clamps. Because of its robust stability, it can be used with extremely large time steps to find the steady state solution for a linear ("passive") system.

Crank-Nicholson

When the global parameter `secondorder` is set to 2, NEURON uses a variant of the Crank-Nicholson method. This has local error proportional to Δt^2 and is therefore particularly accurate for small time steps.

In implicit integration methods, all current balance equations must be solved simultaneously. The backward Euler algorithm does not resort to iteration to deal with nonlinearities, since its numerical error is proportional to Δt anyway. The special feature of the Crank-Nicholson variant is its use of a staggered time step algorithm to avoid iteration of nonlinear equations (see **Efficiency** in the section **Crank-Nicholson: stable and more accurate** above). This converts the current balance part of the problem to one that requires only the solution of simultaneous *linear* equations, making the computational cost per time step almost identical to the backward Euler method.

The second order fixed time step method works with HH-type Ohm's law channels, but its accuracy is really only first order when the instantaneous current-voltage relation of the channels is nonlinear or when channel gating models are expressed with kinetic schemes (the `SOLVE scheme METHOD sparse` statement in NMODL solves kinetic schemes using the fully implicit method). Accuracy is also formally first order for models involving changing ion concentration, though that is a negligible issue when `dt` is small enough to accurately follow voltage changes.

Although the Crank-Nicholson method is formally stable, models with stiff equations require small Δt to avoid numerical oscillations (Fig. 4.8). It is unusable in the presence of voltage clamps, extracellular mechanisms, or linear circuits, since the solution of algebraic equations gives results with large numerical oscillations.

Adaptive integrators

NEURON's adaptive integrators free the user from having to choose an integration step size. Instead, they automatically adjust integration order and Δt so that the solution satisfies a user-specified error criterion. While this may be the most salient feature of these methods, there are several reasons why they may be preferable to fixed step integrators:

- Adaptive integrators usually require less time for a given degree of accuracy.
- They avoid the problem of "empty temporal resolution" (many solution points when nothing is happening) that occurs with fixed time step integration.
- Currents, voltages, and conductances are all known to the same accuracy at the same time, unlike the staggered Crank-Nicholson method.
- Events occur at their actual times instead of being constrained to multiples of Δt. For example, with fixed time steps, current step discontinuities are only first order correct unless they are defined to lie on time step boundaries. Precise timing may be particularly important in network simulations.

Switching between fixed and variable time step methods is as easy as a button press (**NEURON Main Menu / Tools / VariableStepControl / Use variable dt**) and does not affect any GUI tools. Plots of expressions vs. time still look the same, and `Vector` recording of temporal streams still works. There is no need to change model descriptions, or at least to change the statements that define the equations. Ease of switching is crucial since relative performance between high overhead variable step and low overhead fixed step methods ranges widely. For example, simulation of the demonstration models by Mainen and Sejnowski (1996) slowed down by a factor of 2 or sped up by a factor of 7, depending on number of spikes in a simulation run and whether there were long intervals in which *no* state changed rapidly.

CVODE

CVODE handles any kind of model description involving `DERIVATIVE` or `KINETIC` representations of gating states, ion accumulation/diffusion, or nonlinear current-voltage relations. It does not work with models that involve extracellular mechanisms, linear circuits, perfect voltage clamps, or capacitors between nodes. Each cell in a network simulation may have its own local time step, but time steps must be global if there are gap junctions between different cells. Cell mechanisms that have analytical solutions (e.g. integrate and fire artificial spiking cells) can be implemented in a way that allows discrete event simulations.

DASPK

The DASPK method is suitable for models that involve extracellular mechanisms, linear circuits, perfect voltage clamps, or capacitors between nodes. However, there is no local variable step variant of DASPK.

References

Brown, P.N., Hindmarsh, A.C., and Petzold, L.R. Using Krylov methods in the solution of large-scale differential-algebraic systems. *SIAM Journal of Scientific Computing* 15:1467-1488, 1994.

Cohen, S.D. and Hindmarsh, A.C. CVODE User Guide. Livermore, CA: Lawrence Livermore National Laboratory, 1994.

Cohen, S.D. and Hindmarsh, A.C. CVODE, a stiff/nonstiff ODE solver in C. *Computers in Physics* 10:138-143, 1996.

Crank, J. *The Mathematics of Diffusion*. 2 ed. London: Oxford University Press, 1979.

Crank, J. and Nicholson, P. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Proceedings of the Cambridge Philosophical Society* 43:50-67, 1947.

Dahlquist, G. and Bjorck, A. *Numerical Methods*. Englewood Cliffs, New Jersey: Prentice-Hall, 1974.

Hamming, R.W. *Numerical Methods for Scientists and Engineers*. 2 ed: Dover Publications, 1987.

Hindmarsh, A.C. and Serban, R. User documentation for CVODES, an ODE solver with sensitivity analysis capabilities: Lawrence Livermore National Laboratory, 2002.

Hindmarsh, A.C. and Taylor, A.G. User documentation for IDA, a differential-algebraic equation solver for sequential and parallel computers: Lawrence Livermore National Laboratory, 1999.

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

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

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

Kundert, K. Sparse matrix techniques. In: *Circuit Analysis, Simulation and Design*, edited by A. Ruehli: North-Holland, 1986.

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

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P. *Numerical Recipes in C*. 2 ed. Cambridge: Cambridge University Press, 1992.

Stewart, D. and Leyk, Z. *Meschach: Matrix Computations in C*. Proceedings of the Centre for Mathematics and its Applications. Vol. 32. Canberra, Australia: School of Mathematical Sciences, Australian National University, 1994.

Strang, G. *Introduction to Applied Mathematics*. Wellesley, MA: Wellesley-Cambridge Press, 1986.

Chapter 4 Index

C

