CHAPTER 14

Numerical Methods for Neuronal Modeling

Michael V. Mascagni

Center for Computing Sciences, I.D.A.
17100 Science Drive
Bowie, Maryland 20715-4300
Tel:(301) 805-7421
e-mail: na.mascagni@na-net.onrl.gov

Arthur S. Sherman

National Institutes of Health
BSA Building, Suite 350
Bethesda, MD 20892-2690
e-mail: Arthur_Sherman@nih.gov

14.1 Introduction

In this chapter we will discuss some practical and technical aspects of numerical methods that can be used to solve the equations that neuronal modelers frequently encounter. We will consider numerical methods for ordinary differential equations (ODEs) and for partial differential equations (PDEs) through examples. A typical case where ODEs arise in neuronal modeling is when one uses a single lumped-soma compartmental model to describe a neuron. Arguably the most famous PDE system in neuronal modeling is the phenomenological model of the squid giant axon due to Hodgkin and Huxley.

The difference between ODEs and PDEs is that ODEs are equations in which the rate of change of an unknown function of a single variable is prescribed, usually the derivative with respect to time. In contrast, PDEs involve the rates of change of the solution with respect to two or more independent variables, such as time and space. The numerical methods we will discuss for both ODEs and PDEs involve replacing the derivatives in the differential equations with finite difference approximations to these derivatives. This reduces the differential equations to algebraic equations. The two major classes of finite difference methods we will discuss are characterized by whether the resulting algebraic equations explicitly or implicitly define the solution at the new time value. We will see that the method of solution for explicit and implicit methods will vary considerably, as will the properties of the solutions of the resulting finite difference equations.

To simplify our exposition, we will use the Hodgkin-Huxley equations as illustrative examples for the numerical methods we discuss. If one space clamps a section of a squid giant axon the membrane potential will no longer depend on the spatial location within the clamped region. This reduces the original PDE to a system of ODEs, and leads us to model the membrane potential with the following system of ODEs:

$$C\,\frac{dV}{dt} = -\overline{g}_{Na}m^3h(V-E_{Na}) - \overline{g}_K n^4(V-E_K) - \overline{g}_{leak}(V-E_{leak}), \quad (14.1)$$

where

$$\begin{aligned} \frac{dm}{dt} &= (1-m)\alpha_m(V) - m\beta_m(V), \\ \frac{dh}{dt} &= (1-h)\alpha_h(V) - h\beta_h(V), \\ \frac{dn}{dt} &= (1-n)\alpha_n(V) - n\beta_n(V). \end{aligned} \quad (14.2)$$

In addition to this relation for current balance, Hodgkin and Huxley provided expressions for the rate functions $\alpha_m(V)$, $\alpha_h(V)$, $\alpha_n(V)$, $\beta_m(V)$, $\beta_h(V)$, and $\beta_n(V)$. (Hodgkin and Huxley, 1952).

If instead of space clamping the *Loligo* giant axon, one allows the voltage across the membrane of the axon also to vary with longitudinal distance along the axon, x, then the membrane potential satisfies a PDE. This PDE is similar to the space clamped ODE case except that eq. 14.1 is replaced with

$$\begin{aligned} C\,\frac{\partial V}{\partial t} &= \frac{a}{2R}\frac{\partial^2 V}{\partial x^2} - \overline{g}_{Na}m^3h(V-E_{Na}) - \\ &\quad \overline{g}_K n^4(V-E_K) - \overline{g}_{leak}(V-E_{leak}). \end{aligned} \quad (14.3)$$

Below we will consider the complete mathematical description of these two problems related to the squid giant axon and their numerical solution. It is important to note that the Hodgkin-Huxley systems are useful examples for numerical computation in two complementary ways. First, the Hodgkin-Huxley models are very complex, and so provide a realistic and challenging system to test our proposed numerical methods. Numerical methods that work on the Hodgkin-Huxley systems should work equally well on other equations the neuronal modeler may wish to explore. Secondly, the Hodgkin-Huxley equations are basic expressions of current conservation. Thus modification of our formulæ for the numerical solution of the Hodgkin-Huxley system to accommodate other neuronal models is straightforward provided the models are also explicitly based on electrical properties of nerve, and the kinetics associated with the individual ionic currents can be described with first order kinetic equations as in eqs. 14.2.

14.1.1 Numerical Preliminaries

We begin with a discussion of sources of numerical error, both those that affect numerical calculations in general and those that arise specifically in solving differential equations. The fundamental reason for error (loss of *accuracy*) is the finite nature of computers, which limits their ability to represent inherently infinite processes. Irrational numbers, like $\sqrt{2}$, transcendental numbers, like π and e, and repeating decimals, like 1/3, can only be represented to *finite precision.* Even exactly represented numbers are subject to *round-off error* when they are added or multiplied together. For further discussion of these issues consult a general numerical analysis text, such as Golub and Ortega (1992). Numerical methods for ODEs and PDEs involve, in addition, finite approximations of the infinite limiting processes that define derivatives and integrals. These approximations introduce *truncation* or *discretization* error. Even if one solves the discretized problem exactly, the answer is still only an approximation to the original continuous problem.

In order to analyze how the above sources of error are handled by particular numerical methods for ODEs and PDEs, three concepts have been introduced, *stability*, *consistency*, and *convergence.* The most fundamental is convergence, which means that the error between the numerical solution and the exact solution can be made as small as we please. We will be discussing finite difference methods where space and time are discretized with numerical time step, Δt, and a spatial mesh size, Δx. Thus demonstrating convergence for a finite difference method means showing that the numerical solution differs from the exact solution by a term which goes to zero as Δt and Δx go to zero.

In the establishment of general convergence theory, the concepts of stability and consistency of a numerical method also have emerged as fundamental. As the name implies, consistency of a numerical method ensures that the numerical solution solves a discrete problem that is the same as the desired continuous problem. For finite difference methods, this amounts to determining whether the difference equations, when applied to the exact solution to the continuous problem, produce only a small approximation (truncation) error. If this truncation error goes to zero as Δt and Δx go to zero, then the numerical method is consistent. This definition for consistency sounds suspiciously like the definition for convergence. However, a method can be consistent and yet not convergent. This is because consistency only demands that the exact solution satisfy the finite difference equations with a truncation error which formally goes to zero. Convergence demands that the numerical and exact solutions can be made to differ by an arbitrarily small amount at every point in time and space. Convergence is a more restrictive definition.

Stability is the concept that fills the gap between consistency and convergence. We call a finite difference method stable if the solution to the finite difference equations remains bounded as the grid parameters go to zero. It is a notable fact that some consistent finite difference methods are not stable. In these cases the numerical solution may grow without bound even when the analytic solution to the same problem might actually be quite small. A method with this type of behavior is obviously not convergent. We should note that some stable finite difference solutions can exhibit small oscillations about the exact solution and still be of great utility. Therefore we can appreciate the importance of using finite difference methods that give solutions which do not grow without bound.

One of the most remarkable results in the analysis of finite difference methods for differential equations is the Lax Equivalence Theorem (Richtmyer and Morton, 1967), which states that a finite difference method for linear ODEs or PDEs is convergent if and only if it is both consistent and stable. Thus for these linear problems, the two concepts of consistency and stability are complementary. Because of this elegant relationship between these three concepts for linear problems, numerical methods for nonlinear problems also discuss consistency and stability in the context of establishing numerical convergence; however, in these nonlinear cases there is no general Lax Equivalence Theorem. Technical treatments of this theory can be found in Isaacson and Keller (1966), Richtmyer and Morton (1967), and Sod (1985).

14.2 Methods for ODEs

The theory for the numerical solution of ODEs is very well established, and the rigorous analysis of many classes of numerical methods has an extensive literature (see for example Gear, 1971a; Lambert, 1973). We will distinguish numerical methods for ODEs based on a property of ODEs themselves, known as stiffness. Stiffness measures the difficulty of solving an ODE numerically, in much the same way that the condition number of a matrix measures the difficulty of numerically solving the associated system of linear equations (see Appendix A). Stiff systems are characterized by disparate time scales. Non-stiff systems can be solved by *explicit* methods which are relatively simple (*i.e.* can be coded by an amateur), while stiff systems require more complex *implicit* methods (usually from a professionally written package).

Methods can also be classified by how the accuracy depends on the step size, Δt, usually expressed in "big Oh" notation (Lin and Segel, 1988, pp. 112 – 113). For example, a method is $O((\Delta t)^2)$ accurate if the solution differs from the exact solution to the ODE by an amount that goes to zero like $(\Delta t)^2$, as Δt goes to zero. Thus if we halve Δt the error decreases by a factor of $(\frac{1}{2})^2$. Higher order methods are more accurate, but generally are more complicated to implement than lower order methods and require more work per time step. Thus a practical decision must be made weighing the numerical accuracy requirement versus the overall cost of both implementing and using a particular method.

A more precise analysis leads to a distinction between *local* and *global* truncation errors. The local truncation error is the error between the numerical solution and the exact solution after a single time step. Recalling the definition of consistency, it is the local truncation error that must go to zero as Δt goes to zero for a method to be consistent. One can generally compute the local truncation error from the Taylor series expansion of the solution. A method is called p-order accurate if the local truncation error is $O((\Delta t)^p)$. The global truncation error is the difference between the computed solution and the exact solution at a given time, $t = T = n\Delta t$. This is the error that must go to zero as Δt goes to zero for a method to be convergent. In general, one cannot merely use the

Taylor series to calculate the global truncation error explicitly, but it can be shown that if the local truncation error is $O((\Delta t)^p)$ then the global truncation error of a convergent method will also be $O((\Delta t)^p)$ (Stoer and Bulirsch, 1980).

We conclude with the mathematical setting and uniform notation for describing numerical methods. The Hodgkin-Huxley ODE model, eqs. 14.1 and 14.2, is a system of four first-order ODEs. We can define the four dimensional vector of functions, $\vec{U} = (V, m, h, n)$, and rewrite eqs. 14.1 and 14.2 to obtain the single vector differential equation

$$\frac{d\vec{U}}{dt} = \vec{F}(\vec{U},\ t). \tag{14.4}$$

Here $\vec{F}(\vec{U})$ is a vector-valued right hand side corresponding to the right hand sides in eqs. 14.1 and 14.2 with eq. 14.1 rewritten by dividing through by C. In general it is always possible to rewrite any system of ODEs as a single system of first-order ODEs, even when we begin with ODEs which have second or higher order derivatives (Boyce and DiPrima, p. 319 – 320). This is important because almost all numerical methods are designed to handle first-order systems.

Eq. 14.4 only tells us how the unknown functions change with time; we must know the initial values of these functions in order to compute a particular solution. Together, the equations and initial conditions constitute an initial value problem (IVP), where given initial values, the solution is thereafter uniquely determined.

14.2.1 Runge-Kutta Methods

All ODE algorithms begin by discretizing time. Let us denote $t^n = n\Delta t$ and $\vec{U}^n = \vec{U}(n\Delta t)$. Then the simplest method of all, which follows directly from the difference-quotient approximation to the derivative, is the forward Euler method:

$$\frac{\vec{U}^{n+1} - \vec{U}^n}{\Delta t} = \vec{F}(\vec{U}^n,\ t^n). \tag{14.5}$$

An alternative form of the difference quotient gives the backward Euler method:

$$\frac{\vec{U}^{n+1} - \vec{U}^n}{\Delta t} = \vec{F}(\vec{U}^{n+1},\ t^{n+1}). \tag{14.6}$$

Both are examples of first-order Runge-Kutta methods. If we rewrite eq. 14.5 as $\vec{U}^{n+1} = \vec{U}^n + \Delta t\vec{F}(\vec{U}^n,\ t^n)$ we see that the forward Euler method gives us an explicit formula for $\vec{U}^{n+1}$ in terms of the known $\vec{U}^n$. If we rewrite eq. 14.6 with the known quantities on the right and the unknowns on the left we get $\vec{U}^{n+1} - \Delta t\vec{F}(\vec{U}^{n+1},\ t^{n+1}) = \vec{U}^n$. This is not an expression which can be simply evaluated to obtain $\vec{U}^{n+1}$, instead this is an equation whose solution implicitly defines $\vec{U}^{n+1}$. In general some numerical method for solving non-linear algebraic equations, such as Newton's method or functional iteration (Conte and de Boor, 1980), must be used to advance to the next time step. This puts implicit methods at a distinct disadvantage, but, as we will see below, stability considerations often make them the method of choice.

Forward Euler approximates the time derivative with its value at the beginning of the time step, and backward Euler at the end of the time step. By analogy to the trapezoidal rule for numerical quadrature, one can obtain second-order accuracy by using their average:

$$\vec{U}^{n+1} = \vec{U}^n + \frac{\Delta t}{2}\big[\vec{F}(\vec{U}^n,\ t^n) + \vec{F}(\vec{U}^{n+1},\ t^{n+1})\big]. \tag{14.7}$$

This method is also implicit, and one often uses instead the following second-order Runge-Kutta method, also known as Heun's method:

$$\begin{aligned} \vec{U}^{n+1} &= \vec{U}^n + \frac{\Delta t}{2}\big[\vec{k}_1 + \vec{k}_2\big], \text{ where,} \\ \vec{k}_1 &= \vec{F}(\vec{U}^n,\ t^n), \\ \vec{k}_2 &= \vec{F}(\vec{U}^n + \Delta t\vec{k}_1,\ t^{n+1}). \end{aligned} \tag{14.8}$$

Note that second-order accuracy is obtained at the cost of two evaluations of $\vec{F}$ per time step. By going to four function evaluations we can get fourth-order accuracy:

$$\begin{aligned} \vec{U}^{n+1} &= \vec{U}^n + \frac{\Delta t}{6}\big[\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4\big], \text{ where,} \\ \vec{k}_1 &= \vec{F}(\vec{U}^n,\ t^n), \\ \vec{k}_2 &= \vec{F}(\vec{U}^n + \tfrac{1}{2}\Delta t\vec{k}_1,\ t^{n+1/2}), \\ \vec{k}_3 &= \vec{F}(\vec{U}^n + \tfrac{1}{2}\Delta t\vec{k}_2,\ t^{n+1/2}), \\ \vec{k}_4 &= \vec{F}(\vec{U}^n + \Delta t\vec{k}_3,\ t^{n+1}). \end{aligned} \tag{14.9}$$

Heun's method can also be viewed as a *predictor-corrector* version of the trapezoidal rule. That is, one first estimates $\vec{U}^{n+1}$ by taking a forward Euler step. Then the estimate is improved, or corrected, by taking a trapezoidal rule step. Further improvement can be made by correcting again, but it is usually preferable to reduce Δt if more accuracy is desired.

These methods work well when stiffness is not an issue. Therefore, the primary criterion in choosing between them is computational efficiency. There are no hard and fast rules for this, but here are a few rules of thumb. A measure of computational effort is the total number of evaluations of the right hand side functions. Suppose we wish to solve an IVP with initial conditions given at $t = 0$, up to a time $t = T$. If $M = T/\Delta t$ is the number of time steps, and K is the number of function evaluations per time step for a given numerical method, then the most efficient choice of numerical method minimizes the product MK. Higher order methods reduce M but increase K. Unless $\vec{F}$ is very expensive to evaluate, the higher order method generally turns out to be much more efficient. Fifth order Runge-Kutta, however, requires *six* function evaluations, partially explaining the popularity of the fourth-order method. Also, as we shall see, achieving better than second order accuracy is problematic for PDE's, and trapezoidal-rule based methods are the norm.

14.2.2 Multi-Step Methods

The biggest weakness of Runge-Kutta methods is the cost of multiple function evaluations between each time step, whose results are never used again. Multi-step methods attempt to remedy this by approximating the derivative by a combination of the function values at several previous time steps. Defining $\vec{F}^k = \vec{F}(\vec{U}^k, t^k)$, the explicit, 4-step Adams-Bashforth method has the form

$$\vec{U}^{n+1} = \vec{U}^n + \Delta t[c_0\vec{F}^n + c_1\vec{F}^{n-1} + c_2\vec{F}^{n-2} + c_3\vec{F}^{n-3}]. \tag{14.10}$$

while the implicit, 4-step Adams-Moulton method uses the function values at $t^{n+1}, \ldots, t^{n-2}$. The coefficients are chosen so as to interpolate $\vec{F}$ at the several time points with a polynomial (Golub and Ortega, 1992), which is only valid if the solution is smooth.

The Adams-Bashforth and Adams-Moulton methods are often used as a predictor-corrector pair, efficiently attaining fourth-order accuracy with only two function evaluations per time step. One complication, however, is obtaining the initial time points to start the calculation. Generally, a few Runge-Kutta steps are used.

14.2.3 Methods with Adaptive Step Size

The efficiency of both Runge-Kutta and multi-step methods can be enormously enhanced by using adaptive step size control. Adaptive methods exploit known formulæ for the local truncation errors of given methods to estimate the global truncation error. By using this information as a criterion for either increasing or decreasing Δt, a computation can be carried out to within a user specified error tolerance with as large a step size as possible at each time step. In addition to efficiency, this also provides an estimate of the error in the solution.

A useful class of adaptive Runge-Kutta methods is based on the ideas of Fehlberg (Press *et al.*, 1992), who figured out how to combine the results of six function evaluations to obtain two Runge-Kutta methods. One matches the Taylor series of the solution to fourth order, and the other to fifth order. The difference between the two extrapolated solutions is then the fifth term of the Taylor series of the solution, and serves as a good estimate for the error in the fourth order method.

The increased efficiency is of particular value if one is interested in the long time behavior of the system, which requires carrying out calculations until either periodic repetitive firing is observed, or the system reaches a stable steady-state. Also, many neurons are capable of bursting oscillations, characterized by alternating spiking and quiescent periods. An adaptive method will take small steps during the active phase and long steps during the silent phase. As a beneficial side effect, graphical display of the output will be adaptive as well, because many data points are saved when the solution is varying rapidly and few when it is not.

14.2.4 Qualitative Analysis of Stiffness

A system of ODEs, or a single ODE, is said to be stiff if the solution contains a wide range of characteristic time scales. The problem with a wide range of time scales can be appreciated through a simple illustration. Suppose the fastest time scale in an ODE has duration τ, while the slowest has γ. If γ/τ is large, then a numerical time step, Δt, small enough to resolve phenomena on the τ time scale requires γ/τ steps to resolve phenomena on the γ time scale. This is not a problem if the phenomena on the τ time scale is of interest; however, if the τ time scale is of little interest it would seem an obvious strategy to choose $\Delta t \approx \gamma$. For explicit methods, this is a numerical disaster because the inaccuracy in resolving the τ time scale leads to catastrophic instabilities often resulting in wild oscillations in the computed solution.

We can gain insight into this phenomenon by examining the stability properties of some simple methods, applied to the trivial, scalar equation,

$$\frac{dU}{dt} = -kU, \tag{14.11}$$

with $k > 0$. The forward Euler iteration for this equation is

$$U^{n+1} = U^n(1 - k\Delta t). \tag{14.12}$$

In order for the numerical solution to decay, as does the true solution, we must have $-1 < 1 - k\Delta t < 1$. Therefore, Δt must satisfy

$$\Delta t < \frac{2}{k}. \tag{14.13}$$

Monotonic decay requires $1 - k\Delta t > 0$, or $\Delta t < \frac{1}{k}$. That is, Δt must be less than the time constant. For $\frac{1}{k} < \Delta t < \frac{2}{k}$, U^n is multiplied by a negative factor, so the solution undergoes a damped oscillation. These constraints are reasonable but lead to problems when there are multiple time constants. Consider the general linear system of two equations with solution

$$\vec{U}(t) = \vec{A}e^{-100t} + \vec{B}e^{-t}, \tag{14.14}$$

where $\vec{A}$ and $\vec{B}$ are constant vectors that depend on the initial conditions. After a brief time, the first term contributes negligibly to the solution, but the forward Euler method must satisfy the condition of eq. 14.13 for the *fastest* rate constant, or the solution will blow up.

In contrast, the backwards Euler iteration for eq. 14.11,

$$U^{n+1} = U^n \frac{1}{1 + k\Delta t}, \tag{14.15}$$

decays monotonically for all $k > 0$. If Δt is chosen too large, the solution will be inaccurate, but it will not explode. Indeed, the solution will prematurely equilibrate to its steady-state value, 0. This is exactly the behavior we want when solving eq. 14.14. If Δt is chosen appropriate to the slow time scale, the fast component will be solved inaccurately (taken to steady-state), but the total error of the solution will be small.

We can obtain second-order accuracy with the trapezoidal rule, which gives the iteration

$$U^{n+1} = U^n \frac{1 - k\Delta t}{1 + k\Delta t}. \tag{14.16}$$

This is also stable for all Δt, but we pay a price for the increased accuracy: The factor multiplying U^n approaches -1 as $k \to \infty$. For large k, the solution does not decay monotonically, but will exhibit damped *ringing* oscillations.

14.2.5 Methods for Stiff Systems

The space clamped Hodgkin-Huxley ODE model is not particularly stiff. However, it is fairly easy to encounter extremely stiff ODEs that are simply related to eqs. 14.1 and 14.2. If one incorporates the space-clamped Hodgkin-Huxley ODEs into a compartmental model of a neuron, the resulting multicompartment system will generally be stiff, with the stiffness increasing with the number of compartments in the single neuron model. The reason for this is the correspondence between compartmental models of neurons and the so-called method of lines for the numerical solution of PDE models for the same neurons. This relationship between compartmental models and PDE models will be explained in detail in Appendix A along with an explicit calculation of the stiffness of a compartmental model of passive dendritic cable.

Stiffness can also occur in point neurons. A model of the bullfrog sympathetic ganglion cell includes ionic currents with the familiar millisecond time scale along with slow currents with time scales in the hundreds of milliseconds. Some bursting neurons, like R-15, have slow currents and/or other processes, like Ca^{2+} accumulation, that vary on a time scale of seconds.

In general, explicit finite difference methods are susceptible to stiffness, while implicit methods are not. Thus the implicit versions of the ODE methods mentioned above, even backwards Euler, are viable candidates for integrating stiff ODEs. However, one might as well take advantage of the Gear method (Gear, 1971b; Lambert, 1973), which is actually a family of methods of various orders. Both Δt and the order are varied to satisfy the error criteria most efficiently. The methods are implicit multi-step methods like Adams-Moulton, but are based on backward differentiation formulæ. These use old values of $\vec{U}$, rather than $\vec{F}(\vec{U})$ and only evaluate $\vec{F}$ at the future time point. The qth order method has the form

$$\vec{U}^{n+1} - a_0\vec{U}^n - a_1\vec{U}^{n-1} - \ldots - a_{q-1}\vec{U}^{n-(q-1)} = \Delta t b_q \vec{F}^{n+1}. \tag{14.17}$$

The implicit algebraic equations are solved by Newton's method, which requires information about the derivative of $\vec{F}$ (the Jacobian), either supplied by the user or approximated internally by the Gear solver using finite differences.

Note that since these are variable-order methods they do not have the starting problem of the Adams methods. They begin with a one-step, first order method and increase the order as time points are accumulated.

14.2.6 Boundary Value Problems

Another class of mathematical problems that involves ODEs is the boundary value problem (BVP). For example, one can get an idea of spatial effects while avoiding the full complexity of PDEs by letting the PDE settle down to a steady-state, leaving space, x, as the sole independent variable. If we set all the time derivatives in eqs. 14.2 and 14.3 to zero, we obtain the following system of ODEs:

$$\begin{aligned} \frac{a}{2R}\frac{d^2V}{dx^2} &= \overline{g}_{Na} m_\infty(V)^3 h_\infty(V)(V - E_{Na}) + \\ & \quad \overline{g}_K n_\infty(V)^4 (V - E_K) + \overline{g}_{leak}(V - E_{leak}), \\ \text{where} \quad m_\infty(V) &= \alpha_m(V) / \left(\alpha_m(V) + \beta_m(V)\right), \end{aligned} \tag{14.18}$$

and similarly for the other dimensionless variables h_∞ and n_∞. The ∞ subscript indicates that they no longer obey differential equations but are instantaneous functions of membrane potential.

Suppose we want the steady-state solution for a Hodgkin-Huxley axon of length L. Initial conditions are no longer required, but we must now specify boundary conditions. A simple choice is

$$V(0) = V^0, \quad V(L) = V^L, \tag{14.19}$$

corresponding to a two-point voltage clamp, at $x = 0$ and $x = L$.

One approach to solving eqs. 14.18 and 14.19 is the *shooting method*, which converts the BVP to an IVP, with "time" running from 0 to L. Since eq. 14.18 is a second order ODE, proper "initial" conditions are to specify $V(0)$ and $dV(0)/dx$. We know that $V(0) = V^0$, and we a value for $dV(0)/dx$ that will make $V(L) = V^L$. See Conte and de Boor (1980, pp. 412 – 416) for further details.

Although more efficient methods would be used today, it is an interesting historical note that Hodgkin and Huxley solved the problem of determining the wave speed of the axon potential as a shooting problem. For a steadily progressing pulse of unvarying shape, the solution is a function of the variable $z = x - \theta t$, where θ is the wave speed. Hodgkin and Huxley therefore used the substitution $V(x,t) = \phi(z)$ to convert the PDE eq. 14.3 into a second-order ODE in z with boundary conditions $\phi(z) \to 0$ as $z \to \pm\infty$. This leads to an algorithm for determining the wave speed: For θ too small, ϕ diverges to ∞, for θ too large ϕ diverges to $-\infty$. Successive shooting trials allowed Hodgkin and Huxley to bracket the value of θ precisely. This recovery of the speed of the traveling pulse from space- and voltage-clamped currents was the capstone of their achievement that led to the Nobel prize.

More often eqs. 14.18 and 14.19 are solved by the method of finite differences. We introduce a spatial grid with uniform width $\Delta x = L/N$, and with $x_0 = 0$ and $x_N = L$. Defining $V_i = V(i\,\Delta x)$ we use the Taylor series to derive a finite difference approximation to the second spatial derivative:

$$V_{i\pm 1} = V_i \pm \frac{dV}{dx}\Delta x + \frac{d^2V}{dx^2}\frac{(\Delta x)^2}{2} \pm \frac{d^3V}{dx^3}\frac{(\Delta x)^3}{3!} + \frac{d^4V}{dx^4}\frac{(\Delta x)^4}{4!} \pm \ldots \tag{14.20}$$

where all the derivatives are evaluated at x_i. Combining the two equations we solve for $\frac{d^2V}{dx^2}(x_i)$:

$$\frac{d^2V}{dx^2}(x_i) = \frac{V_{i-1} - 2V_i + V_{i+1}}{(\Delta x)^2} + O(\Delta x^2). \tag{14.21}$$

Using this $O((\Delta x)^2)$ accurate approximation we replace eqs. 14.18 and 14.19 with the following nonlinear tridiagonal system of algebraic equations:

$$\begin{aligned} \frac{a}{2R}\frac{V_{i+1} - 2V_i + V_{i-1}}{(\Delta x)^2} &= \overline{g}_{Na} m_\infty(V_i)^3 h_\infty(V_i)(V_i - E_{Na}) + \\ &\quad \overline{g}_K n_\infty(V_i)^4(V_i - E_K) + \overline{g}_{leak}(V_i - E_{leak}), \\ V_0 = V^0, &\quad V_N = V^L. \end{aligned} \tag{14.22}$$

Methods for the solution of difference equations of this form will be treated in the section on PDEs, but note that with the boundary values V_0 and V_N given, eq. 14.22 completely defines $V_1, \ldots, V_{N-1}$.

The general problem of numerically solving the BVP for ODEs has many complications, including singular solutions, unstable solutions, and periodic and translational boundary conditions. A good resource is the monograph by Keller (1976).

14.2.7 Problems with Discontinuities

Discontinuities in solutions or their derivatives pose special challenges for numerical methods. Unfortunately, they occur naturally in simulations of voltage-clamping, applied current pulses, channel noise, and integrate and fire neurons. In the first two cases, the events are often imposed by the user and occur with modest frequency. Naive use of Adams methods can fail here because they try to fit a smooth solution to the discontinuity. The remedy is to stop and restart the calculation at each event, which is clumsy and incurs extra overhead. Robust implementations of the variable-order Gear algorithm can detect discontinuities and restart automatically with a first-order method. It is more efficient, however, to instruct the solver explicitly to restart. Fixed step-size Runge-Kutta methods sail through discontinuities, but accuracy may be reduced unless the jumps are arranged to occur at time-step boundaries.

Channel noise and other stochastic simulations pose a harder problem, because the transitions occur frequently and unpredictably. If events are constrained to time-step boundaries, the simulation can only be first order accurate, forcing use of small steps. Alternatively, one may use knowledge that transitions are, say, exponentially distributed to integrate from event to event (Clay and DeFelice, 1983). However, if the population size is large, this will also result in small steps. One can then reformulate the problem as a continuous diffusion and solve the stochastic differential equations (Fox and Lu, 1994). Special Runge-Kutta methods have been developed to solve these with second or higher order accuracy (Kloeden et al., 1993).

Integrate and fire networks raise similar issues, with the additional complication that each element's firing can influence the other elements. Fixed step-size integration is again only first-order accurate, regardless of the accuracy between events, and can also artifactually synchronize the elements. The safest approach is to determine numerically or analytically when the next unit will fire, integrate up to that time, and calculate the interactions (Tsodyks et al., 1993). As for stochastic problems, one can smooth out the discontinuities for large networks by solving for the distribution of firing events (Kuramoto, 1991).

14.2.8 Guide to Method Selection and Packages

For ODEs stiffness is the critical factor to consider when choosing between alternative numerical methods. In general, a compartmental model of a neuron will yield a rather stiff system of ODEs, so an implicit method should be used to avoid numerical instabilities. Gear's adaptive time step method is particularly attractive. Popular public-domain packages in `FORTRAN` include `DDRIV` and several variants of the subroutine `LSODE`. An updated version of `LSODE`, `CVODE`, is available in `C`. These packages can be downloaded from `netlib`. General commercial libraries, such as IMSL (Visual Numerics, Inc.), and NAGLIB (Numerical Algorithms Group Ltd.) also have `FORTRAN` and `C` routines for ODEs (and PDEs). A single source for information on all of these is GAMS, the Guide to Available Mathematical Software maintained by the National Institutes of Standards and Technology. GAMS includes a problem decision tree and links to online documentation, example driver programs and provider information.

In contrast, small systems of ODEs, or a system of ODEs that does not have direct coupling between its domains may not be very stiff, so it may be computationally cheaper to use an explicit method, such as one of the Runge-Kutta methods. Explicit second or fourth order methods that are easy to write oneself may be adequate. However, adaptive methods usually pay off handsomely, and are well worth the effort to acquire or program one. Public-domain and commercial packages for both fixed and variable-step Runge-Kutta methods can be obtained from the same sources as above. For do-it-yourselfers, Press et al. (1992) give a detailed explanation and code for Runge-Kutta Fehlberg, as well as a number of other methods. Most Gear packages include options for Adams methods and have well-tested heuristics for scaling variables and estimating errors, making them good general purpose solvers. The Adams implementations are also variable order, hence self-starting and robust on discontinuities, and do not need to evaluate the Jacobian, so they can be faster than Gear on non-stiff problems. Some packages (eg. `LSODA`) attempt to automatically evaluate stiffness and switch between Gear and Adams.

If one is unsure about the stiffness of a particular ODE system whose numerical solution is required, it may be worthwhile to begin with a low-order explicit method, like the second order Runge-Kutta method and use it on a test problem where the solution's behavior is known. If the method produces good results, one may not need to look any further for a numerical method. However, if the solution shows unexpected oscillatory behavior or requires a time step much smaller than the time scale of interest in order to avoid such instability, one should suspect stiffness.

Packages for boundary value problems can be obtained from the same sources as above. In addition, the program `AUTO` (Doedel, 1991), designed primarily for bifurcation calculations (the study of how solutions change as parameter are varied) also can be used as a BVP solver.

Complete interactive programs that handle the allocation of memory, data input and output, and graphics are an increasingly popular solution. The user need only define the problem *and* choose an appropriate numerical method. Two programs available for Unix workstations are `dstool`, which has a variable step-size Runge-Kutta solver, and `xpp` which has a Gear solver and allows automatic detection of and restarting at discontinuities. In addition `xpp` can solve BVPs by the shooting method and features an interactive interface to `AUTO`. Both `dstool` and `xpp` are general differential equation solvers, and differ from the compartmental modeling programs described in the PDE section in that they are equation-oriented, not object-oriented. That is, one must specify the problem in terms of equations, rather than in terms of channels and cable properties. DOS PC users can consider `PHASER` (Koçak, 1989) or `PHASEPLANE` (Ermentrout, 1990), a predecessor of `xpp`, and `LOCBIF`. Macintosh users can try `MacMath` (Hubbard and West, 1992), for systems of two or three variables. General interactive mathematics packages such as Maple, Mathematica, Matlab, and Mlab, also have tools for solving differential equations with graphics.

Details on the packages mentioned in this section and information on how to obtain them can be found at http://mrb.niddk.nih.gov/sherman.

We have downplayed the convergence of numerical methods for ODEs, because the convergence theory for ODE methods is relatively straightforward. Even naive methods like forward Euler will converge for well-posed ODE problems provided that Δt is chosen sufficiently small. However, this glibness with convergence will not carry over to our discussion of numerical methods for PDEs.

14.3 Methods for PDEs

In general, numerical methods for PDEs are not as well understood as numerical methods for ODEs, largely because of the greater mathematical complexity of PDEs. In contrast to the numerical methods for ODEs, much of the intuition accumulated for understanding and choosing PDE methods has come from studying methods for solving linear PDEs. For this reason we will begin our discussion not with the example of the Hodgkin-Huxley equations, but with its linear counterpart, the passive cable equation from dendritic modeling. This equation is

$$C\,\frac{\partial V}{\partial t} = \frac{a}{2R}\frac{\partial^2 V}{\partial x^2} - \overline{g}V. \qquad (14.23)$$

Here $\overline{g}$ is the passive membrane conductance per unit area.

14.3.1 Finite Difference Methods

Two popular methods for numerically solving PDEs are finite difference and finite element methods. Finite element methods in many cases reduce to finite difference methods. This is especially so for many neuronal models, where often only one spatial dimension is used in the PDEs. However, in cases where a model results in PDEs with more than one spatial dimension, finite element methods and more advanced finite difference methods should be considered. We describe only finite difference methods in this chapter. As with these methods for ODEs, finite difference methods for PDEs employ finite difference approximations of the derivatives in the PDEs to reduce the differential equations to algebraic equations. The Hodgkin-Huxley system as well as the linear cable equation are PDEs of the parabolic type, and the methods we will discuss will also be generally applicable to other parabolic PDEs, such as the heat and diffusion equations, Douglas (1961).

In treating parabolic PDEs, it is customary to first introduce the discretization of the spatial variables. This is called the method of lines, and reduces the PDEs to a system of coupled ODEs. In Appendix A we will see how conceptually close the method of lines for PDEs is to compartmental modeling. If we replace the continuous variable, x, with a uniformly spaced grid of length $\Delta x = L/N$ with $N+1$ grid points over L spatial units, then the method of lines transforms eq. 14.23 into the following coupled system of ODEs:

$$C\,\frac{dV_i}{dt} = \frac{a}{2R}\frac{V_{i+1} - 2V_i + V_{i-1}}{(\Delta x)^2} - \overline{g}V_i, \quad i = 0, \dots, N. \qquad (14.24)$$

This system is stiff, with the stiffness increasing with N. Eq. 14.24 is the example we will use in Appendix A to explicitly compute the numerical stiffness of a simple compartmental model ODE formulation.

Since eq. 14.24 is stiff, we should use an implicit method to solve it. One could use an ODE package described previously to solve such a system. However, these packages may or may not be able to solve the systems as efficiently as using a simple temporal discretization permits. Thus we will present three finite difference methods for the numerical solution of these parabolic PDEs: forward Euler, backward Euler, and Crank-Nicolson. We will initially present them for the linear cable eqs. 14.23, and then in a later section we discuss the modifications necessary to numerically solve the nonlinear Hodgkin-Huxley PDE system.

All three of these finite difference methods use the $O((\Delta x)^2)$ finite difference approximation to the second spatial derivative that we have previously introduced. The difference between these three methods is only in the manner in which the time derivative on the left hand side of eq. 14.23 is discretized. Let us now define the computational grid which we will use with all the PDE finite difference methods. The time variable will be replaced by a discrete set of time values with a uniform spacing of Δt. ${V_i}^n$ refers to $V(i\Delta x, n\Delta t)$.

The forward Euler method uses the most naive time discretization. The typical form of this discretization is:

$$C\,\frac{V_i^{n+1} - {V_i}^n}{\Delta t} = \frac{a}{2R}\frac{V_{i+1}^n - 2{V_i}^n + V_{i-1}^n}{(\Delta x)^2} - \overline{g}{V_i}^n. \tag{14.25}$$

If we define the constants $\sigma = a\Delta t/2RC(\Delta x)^2$ and $\gamma = \overline{g}\Delta t/C$, then this equation can be rewritten as

$$V_i^{n+1} = \sigma V_{i+1}^n + (1 - 2\sigma - \gamma)V_i^n + \sigma V_{i-1}^n. \tag{14.26}$$

The error in using the forward Euler method is $O(\Delta t) + O((\Delta x)^2)$ (Isaacson and Keller, 1966).

From eq. 14.26 we see that the forward Euler method is an explicit method since V_i^{n+1} is explicitly defined by the right hand side terms, which are known. The forward Euler method is also very easy to implement; however, it has numerical properties which are quite undesirable. The worst of these is that it is numerically unstable when $\sigma > 1/2$, which means that stability requires $\Delta t \leq RC(\Delta x)^2/a$. Thus, if we wish to achieve higher spatial accuracy in our numerical solution by decreasing Δx by a factor Q, we must also then decrease Δt by a factor of Q^2 to assure we maintain numerical stability. Thus the amount of work that must be done to obtain a solution up to a fixed time is multiplied by Q^3 to achieve a factor Q finer spatial grid.

If in eq. 14.25, the left hand side time difference is set equal to the right hand side at time value $n+1$ we obtain the backward Euler method

$$C\,\frac{V_i^{n+1} - V_i^n}{\Delta t} = \frac{a}{2R}\frac{V_{i+1}^{n+1} - 2V_i^{n+1} + V_{i-1}^{n+1}}{(\Delta x)^2} - \overline{g}V_i^{n+1}. \tag{14.27}$$

We can rewrite these equations as

$$-\sigma V_{i+1}^{n+1} + (1 + 2\sigma + \gamma)V_i^{n+1} - \sigma V_{i-1}^{n+1} = V_i^n. \tag{14.28}$$

Unlike eq. 14.26, this equation does not explicitly define the values at that new time step in terms of the values at the old time step. Thus the backward Euler method is an implicit method, and a linear system of equations must be solved at each time step. The type of linear equation that must be solved is called a tridiagonal linear system, since the left hand side of eq. 14.28 involves the unknown voltage and its two nearest neighbors on the grid. In a following section we will discuss the numerical solution of tridiagonal linear systems of equations, with special emphasis on efficiency.

Even though the backward Euler method involves the solution of a tridiagonal linear system at each time step, it is considered superior to the forward Euler method for these types of PDE problems. One reason for this is that one can solve the tridiagonal system which arises at each time step in the backward Euler method in $O(N)$ arithmetic operations, which is the same order of complexity as for the forward Euler method. More importantly, the backward Euler method does not suffer from numerical instability like the forward Euler method. Thus we can choose the grid parameters Δt and Δx independently and not have to worry if some combination violates a stability inequality. Also, if the accuracy in one of the grid parameters is insufficient, we may refine that variable without having to readjust the other grid parameter. One of the advantages of this independence in Δt and Δx for the backward Euler method is the possibility of using a rather large Δt to explore qualitatively the behavior of a system at little computational expense. When interesting behavior is noted, a smaller Δt can be used to reexamine the phenomena of interest with greater numerical accuracy. We note that the backward Euler method has the same numerical accuracy as the forward Euler method, namely $O(\Delta t) + O((\Delta x)^2)$.

The last method we present is called the Crank-Nicolson method (Crank and Nicolson, 1947). This method is related to both the Euler methods as its right hand side is the average of the two Euler right hand sides:

$$\begin{aligned} C\,\frac{V_i{}^{n+1} - V_i{}^n}{\Delta t} &= \frac{1}{2}\Bigg(\frac{a}{2R}\frac{V_{i+1}^{n+1} - 2V_i^{n+1} + V_{i-1}^{n+1}}{(\Delta x)^2} - \overline{g}V_i^{n+1} + \\ &\quad \frac{a}{2R}\frac{V_{i+1}^{n} - 2V_i^{n} + V_{i-1}^{n}}{(\Delta x)^2} - \overline{g}V_i{}^{n}\Bigg). \end{aligned} \tag{14.29}$$

Unlike the Euler methods, the Crank-Nicolson method has numerical accuracy which is $O((\Delta t)^2) + O((\Delta x)^2)$. The rationale for this is the same as that described in the discussion of the trapezoidal rule for ODEs.

Let us rearrange eq. 14.29 by placing the values at the old time level on the right and the new values on the left

$$\begin{aligned} -\frac{\sigma}{2}V_{i+1}^{n+1} + (1 + \sigma + \frac{\gamma}{2})V_i{}^{n+1} - \frac{\sigma}{2}V_{i-1}^{n+1} = \\ \frac{\sigma}{2}V_{i+1}^{n} + (1 - \sigma - \frac{\gamma}{2})V_i{}^{n} + \frac{\sigma}{2}V_{i-1}^{n}. \end{aligned} \tag{14.30}$$

Here σ and γ are as previously defined. As with the backward Euler method, this equation implicitly defines the new values as the solution to a tridiagonal system of linear equations. In addition, this method is unconditionally stable for any choice of the grid parameters Δt and Δx. One subtle difference from the backward Euler method is that in some sense the Crank-Nicolson method is closer to numerical instability. This is observed in Crank-Nicolson numerical solutions where damped oscillations and over/undershoot are seen where none are expected even when large values of σ are used. This ringing is rarely seen with the backward Euler method as it is much more heavily damping than Crank-Nicolson. This behavior is analogous to that described for the corresponding ODE methods described in the previous ODE discussion, see especially eq. 14.16.

An interesting implementational detail exploits a useful relationship between the the backward Euler and the Crank-Nicolson solution. If V_i^{n+1} are the solution voltages to the backward Euler equations (14.28) starting with voltages V_i^n, then $2V_i^{n+1} - V_i^n$ is the solution to the Crank-Nicolson equations starting with voltages V_i^n. Thus it is a trivial task to modify a computer program for the solution of these PDEs from an $O(\Delta t)$ backward Euler solver to an $O((\Delta t)^2)$ Crank-Nicolson solver by modifying a single line of code!

14.3.2 Boundary Conditions

Because PDEs involve derivatives with respect to more than one independent variable, specifying initial conditions is more complicated. Initial values must be given at all values of x. In addition, one must specify boundary conditions at the endpoints of neuronal processes. These types of initial-boundary value problems (IBVPs) have been shown rigorously to be well-posed mathematically, meaning that their solutions are unique, depend continuously on the initial and boundary conditions, and remain bounded above and below for all time (Mascagni, 1989a).

We have already encountered a common boundary condition in our discussion of BVPs for ODEs which arise from steady-state computations for PDE problems. Eq. 14.19 specifies the voltage at two ends of a neural cable. This type of boundary condition, where the solution value is specified at the end points, is called a Dirichlet boundary condition. As we will see below, boundary conditions of the Dirichlet type are quite easy to incorporate into finite difference methods for PDEs.

The second common type of boundary condition is the Neumann boundary condition. Instead of specifying the solution value at the end points, as with the Dirichlet boundary conditions, Neumann boundary conditions specify the first spatial derivative of the solution at the end points. Neumann boundary conditions occur very naturally in neuronal modeling. Since $\partial V/\partial x$ is proportional to the longitudinal current through a cable, specifying a Neumann boundary conditions for neuronal cable models amounts to specifying the longitudinal current values at the end points. For example, the following Neumann boundary conditions for the Hodgkin-Huxley PDE system, eqs. 14.2 and 14.3:

$$\frac{\partial V(0)}{\partial x} = -\frac{RI}{\pi a^2}, \quad \frac{\partial V(L)}{\partial x} = 0, \tag{14.31}$$

biophysically corresponds to injecting I microamps of current at $x = 0$, and demanding that no current pass out the $x = L$ end. The Neumann boundary condition at $x = L$ is commonly called a "no leak" or "sealed end" boundary condition in neuronal modeling (Jack, Nobel, and Tsien, 1983). Incidentally, "open" or "killed" end are other names for zero Dirichlet boundary conditions. Neumann boundary conditions are more complicated than Dirichlet boundary conditions to incorporate into finite difference methods to solve PDE IBVPs, but only slightly so.

Both the Dirichlet and Neumann boundary conditions are linear boundary conditions, in the sense that linear PDEs with these boundary conditions obey a superposition property with respect to the boundary conditions. A more unusual type of linear boundary condition involves a linear combination of the Dirichlet and Neumann boundary conditions as follows:

$$\alpha_0 \frac{\partial V(0)}{\partial x} + V(0) = \beta_0, \quad \alpha_L \frac{\partial V(L)}{\partial x} + V(L) = \beta_L. \tag{14.32}$$

Since we know that Neumann boundary conditions are biophysical statements about the currents at the cable ends, it is clear that these mixed boundary conditions are merely a statement that the voltage at the end points obeys a linear or ohmic current voltage relationship. The difficulty of implementing a finite difference method for a PDE IBVP with this sort of mixed boundary conditions is only slightly greater than handling simply a Neumann boundary condition. Finally we mention that by making the coefficients, α and β from eq. 14.32, nonlinear functions of the end point voltages, we can impose a nonlinear current voltage relationship at the end points. An example of a computation with this type of nonlinear boundary condition is due to Baer and Tier (1986). They considered the effects of a Fitzhugh-Nagumo patch which formed the end of a dendritic cylinder as a model of an active membrane site within a passive dendrite. Li et al. (1995) used a similar approach to study the interaction of intracellular calcium handling with membrane ion channels and pumps, representing the former with a diffusion equation and the latter as a nonlinear boundary condition.

We must comment on how the different BVPs which we have just discussed can be incorporated into the previously discussed finite difference methods. The Dirichlet boundary conditions are the simplest to handle, since they directly prescribe the boundary values. Thus in the forward Euler method, we use the known end point values, V_0 and V_N, in the equations for V_1^{n+1} and V_{N-1}^{n+1}. In the two implicit methods discussed, knowing V_0 and V_N allows us to reduce the number of equations to be solved from $N+1$ to $N-1$. We simply place the terms involving the known boundary values onto the right hand sides of the equations for V_1^{n+1} and V_{N-1}^{n+1}, and solve the resulting tridiagonal system. For example, in the backward Euler case, the equation for V_1^{n+1} is $-\sigma V_2^{n+1} + (2\sigma + \gamma + 1){V_1}^{n+1} - \sigma V_0^{n+1} = V_1^n$. Since V_0^{n+1} is known, we can rewrite this equation as $-\sigma V_2^{n+1} + (2\sigma + \gamma + 1){V_1}^{n+1} = \sigma V_0^{n+1} + {V_1}^n$. We do the same for V_{N-1}^{n+1}. A similar manipulation can be used to incorporate Dirichlet boundary conditions into the Crank-Nicolson method.

It is a only bit more difficult to handle Neumann boundary conditions. We would like the manner in which we incorporate the Neumann boundary conditions to preserve two properties of our numerical methods. First, since all of our numerical methods are $O((\Delta x)^2)$ spatially accurate, we ask that the boundary condition incorporation be at least this accurate. Secondly, we ask that the neat tridiagonal form of the linear equations which arise from the implicit methods be maintained. These two requirements are in a sense competing; however, it is possible to achieve both through the following construction. Consider Neumann boundary conditions from eq. 14.31 for the forward Euler method. A second order accurate finite difference formula for the first derivative that only involves two points is the centered difference approximation (Dahlquist and

Bjorck, 1974). If we wish to approximate the first derivative at $x = 0$ using the centered difference formula, we need to know the value of V at $x = \pm\Delta x$; however, our computational grid does not include the point at $x = -\Delta x$. If we pretend to know V at this point we can achieve both of the previous paragraph's goals (Sod, 1985; Cooley and Dodge, 1966). Knowing both V_1 and V_{-1} gives us the centered difference formula $\frac{V_1 - V_{-1}}{2\Delta x} = \frac{\partial V(0)}{\partial x}$ up to $O((\Delta x)^2)$. Since the Neumann boundary condition specifies the value of $\frac{\partial V(0)}{\partial x}$, this can be used to write the unknown V_{-1} in terms of the given derivative value and V_1. Thus the equation for V_0^{n+1} for the forward Euler method can be rewritten as:

$$V_0^{n+1} = \sigma V_1^n + (1 - 2\sigma - \gamma)V_0^n + \sigma V_{-1}^n = \tag{14.33}$$

$$\sigma V_1^n + (1 - 2\sigma - \gamma)V_0^n + \sigma(V_1^n + 2\Delta x \frac{\partial V(0)}{\partial x}). \tag{14.34}$$

Since $\frac{\partial V(0)}{\partial x}$ is specified in the boundary condition, the right hand side is completely known. One can follow the same procedure for V_N^{n+1}.

Since the implicit methods require the solution of a tridiagonal system of linear equations, we can use this same centered difference approximation of $\frac{\partial V}{\partial x}$ to incorporate Neumann boundary conditions into the tridiagonal systems. Using the same rationale as above, we can solve for V_{-1}^{n+1} and V_{N+1}^{n+1} using the centered difference approximation and then substitute these expressions into the V_0 and V_N equations. Since we use the centered difference approximation to the derivative, this construction maintains the tridiagonal structure of the equations. This is because we incorporate the centered difference approximation only at the two end points. For example, the first equation from the backward Euler system for the IBVP for eq. 14.23 with Neumann boundary conditions, eqs. 14.31, is:

$$(1 + 2\sigma + \gamma)V_0^{n+1} - 2\sigma V_1^{n+1} = V_0^n + \frac{2RI\sigma\Delta x}{\pi a^2}. \tag{14.35}$$

The same problem discretized via Crank-Nicolson yields a first equation of:

$$\left(1 + \sigma + \frac{\gamma}{2}\right)V_0^{n+1} - \sigma V_1^{n+1} = \left(1 - \sigma - \frac{\gamma}{2}\right)V_0^n + \sigma V_1^n + \frac{2RI\sigma\Delta x}{\pi a^2}. \tag{14.36}$$

Knowing how to incorporate both Dirichlet and Neumann boundary conditions into our three PDE methods, it is relatively easy to combine these techniques to allow mixed boundary conditions as in equation (14.32). Using the two extra grid points, V_{-1} and V_{N+1}, we can discretize equation (14.32) up to $O((\Delta x)^2)$ as follows:

$$\alpha_0\left(\frac{V_{-1} - V_1}{2\Delta x}\right) + V_0 = \beta_0, \quad \alpha_L\left(\frac{V_{N-1} - V_{N+1}}{2\Delta x}\right) + V_N = \beta_L \tag{14.37}$$

We notice that both these equations involve the grid's end points and their two neighboring points. As with our treatment of the Neumann boundary conditions, these equations can be solved for the values at V_{-1} and V_{N+1} and the resulting expressions substituted into the equations for V_0 and V_N. This is elementary for the forward Euler method, and for the two implicit methods it is clear that a linear tridiagonal system of equations is again the final result.

14.3.3 Spatial Variation

Recall that the Hodgkin-Huxley PDE models the giant axon of the squid *Loligo*. A basic assumption is that the electrotonic properties of the squid's neuronal membrane are uniform and do not depend on the longitudinal location along the axon. While this is an adequate assumption for that preparation, one should be able to incorporate spatial variation of the neuronal membrane into PDE models. We now discuss how to express longitudinal variation in membrane properties in the linear cable PDE system, as well as how to incorporate this spatial variation into discretizations while preserving $O((\Delta x)^2)$ spatial accuracy.

In the linear cable model PDE, one can incorporate spatial variation in both the ionic conductance, $\overline{g}$, and the membrane capacitance, C, in the most trivial way: make them functions of x. The hard part is what to do when the denritic radius, a, and the specific axoplasmic resistivity, R, are functions of x. One cannot just make them functions of x in eq. 14.23, because that does not preserve the biophysical meaning of this equation. Eq. 14.23 is a statement of the instantaneous conservation of charge along the membrane, and the term $(a/2R)\partial^2 V/\partial x^2$ represents the total membrane current per unit area. This expression was originally obtained by using current conservation for an infinitesimal slice of dendrite and the differential form of Ohm's law. To see how to rewrite this term when a and R depend on x we must go back to this derivation.

The expression for total membrane current per unit are can be written as:

$$\frac{a}{2R}\frac{\partial^2 V}{\partial x^2} = \frac{1}{2\pi a}\frac{\partial}{\partial x}\left(\frac{\pi a^2}{R}\frac{\partial V}{\partial x}\right). \tag{14.38}$$

The term in parentheses is the axial current. Eq. 14.38 restates conservation of charge by stating that the divergence of the axial current equals the total membrane current. The constants in this expression convert the axial current, which is per unit cross sectional area, into the membrane current per unit membrane area. Using eq. 14.38, we can rewrite the linear cable equation, eq. 14.23, to allow spatial variation in all the cable parameters:

$$C(x)\,\frac{\partial V}{\partial t} = \frac{1}{2\pi a(x)}\frac{\partial}{\partial x}\left(\frac{\pi a^2(x)}{R(x)}\frac{\partial V}{\partial x}\right) - \overline{g}(x)V. \tag{14.39}$$

One performs the analogous substitution with the Hodgkin-Huxley PDE model, eqs. 14.2 and 14.3, to incorporate spatial variation in that case. Strictly speaking, one should replace the factor $a(x)$ in the denominator of eq. 14.39 by $a(x)\sqrt{1 + a'(x)^2}$ and make the corresponding change in eq. 14.40 to account for the increased membrane area of a tapered cable. See Jack et al, 1983, p. 150. However, eq. 14.39 is usually quantitatively adequate because $a'(x)^2 \ll 1$.

Now we must consider how to convert these spatially varying continuous models into finite difference equations with $O((\Delta x)^2)$ accuracy. In eq. 14.39, spatial dependence is only problematic when we try to discretize the term for the total membrane current per unit area. This term involves the derivative of a derivative, and so we proceed by discretizing the derivatives one at a time. A first order approximation to the first derivative is $\partial V/\partial x = (V_{i+1} - V_i)/\Delta x$. Applying this expression twice to the total membrane current term in eq. 14.39 gives us

$$\frac{1}{2\pi a(x)}\frac{\partial}{\partial x}\left(\frac{\pi a^2(x)}{R(x)}\frac{\partial V}{\partial x}\right) \approx \frac{1}{2\pi a_i}\frac{\left(\frac{\pi a^2_{i+1/2}}{R_{i+1/2}}\left(\frac{V_{i+1}-V_i}{\Delta x}\right) - \frac{\pi a^2_{i-1/2}}{R_{i-1/2}}\left(\frac{V_i - V_{i-1}}{\Delta x}\right)\right)}{\Delta x}. \tag{14.40}$$

The value at grid point index $i + 1/2$ refers to the numerical value of the indexed function at $x + \Delta x/2$, as $i - 1/2$ refers to $x - \Delta x/2$. Even though the spatial grid does not include these points, it is reasonable to ask for the value of the continuously defined variables $a(x)$ and $R(x)$ at these points. However, if these intermediate values are not available, one can substitute the average of the values at the two flanking grid points in this discretization.

The spatial discretization of the troublesome second derivative term for the total membrane current given in eq. 14.40 can be shown to yield an $O((\Delta x)^2)$ accurate finite difference approximation (Cooley and Dodge, 1966). Thus one can use the discretization in eq. 14.39 to solve a spatially dependent problem with one of the finite difference methods we have already described with the same numerical accuracy. These discretizations lead to the same types of equations as in the constant coefficient case. One can incorporate boundary conditions exactly as in the constant coefficient case, and so we can think of these spatially dependent coefficient problems as being no more complicated than their constant coefficient counterparts. Even so, we have yet to describe the solution of tridiagonal linear systems that arise in all our implicit discretizations.

14.3.4 Solving Tridiagonal Linear Systems

We will discuss only one algorithm for the solution of tridiagonal linear systems of equations, Gaussian elimination. Gaussian elimination is also called LU decomposition, as well as forward elimination with back substitution. Gaussian elimination has many variants which are specialized for efficiency on specific classes of matrices (Golub and Van Loan, 1985). A special variant, called the Thomas algorithm, requires $O(M)$ mathematical operations to solve a tridiagonal system with M unknowns. The Thomas algorithm is numerically stable when the tridiagonal system has the property of diagonal dominance. Since all the tridiagonal systems that arise in finite difference solution of the PDEs we have discussed share are diagonally dominant, we can use the Thomas algorithm, which does not involve pivoting, to solve these systems. To say that a matrix is diagonally dominant means that the magnitude of the diagonal dominates the sum of the magnitudes of the off diagonal elements in each row of the matrix. It is evident from eqs. 14.28 and 14.30 that for the finite difference PDE approximations we have discussed, the tridiagonal systems that arise are diagonally dominant. In general, $O((\Delta x)^2)$ finite difference approximations to well posed parabolic PDEs will be diagonally dominant.

Let us denote our tridiagonal system as

$$L_iV_{i-1} + D_iV_i + U_iV_{i+1} = R_i, \quad 1 \le i \le M, \tag{14.41}$$

with $L_1 = U_M = 0$. The Gaussian elimination algorithm without pivoting proceeds by using adjacent equations in eq. 14.41 to eliminate the subdiagonal unknowns in a step known as forward elimination. One may think of this procedure as sweeping through the equations and redefining the constants L_i, D_i, U_i, and R_i as follows:

$$\begin{aligned}
&D_1 = D_1, U_1 = U_1/D_1, R_1 = R_1/D_1,\\
&D_i = D_i - L_iV_{i-1}, \quad i = 2, 3, \ldots, M,\\
&R_i = (R_i - L_iV_{i-1})/D_i, \quad i = 2, 3, \ldots, M,\\
&U_i = U_i/D_i, \quad i = 2, 3, \ldots, M-1.
\end{aligned} \tag{14.42}$$

This forward elimination procedure succeeds in reducing the original tridiagonal system into an equivalent bidiagonal system. This bidiagonal system can then be solved by a procedure called backwards substitution:

$$V_N = R_N, \tag{14.43}$$

$$V_i = R_i - U_iV_{i+1}, \quad i = M-1, M, \ldots, 1. \tag{14.44}$$

We notice that this procedure requires only five arrays of length M, those for L, D, U, R, and V. Since we overwrite these arrays with the Thomas algorithm, the coefficients for the tridiagonal systems must be recomputed for each time step. In addition, it is obvious from the definition of the algorithm that this procedure requires $O(M)$ arithmetic operations, as it is defined with several "loops", each with a fixed number of arithmetic operations per iteration, and of length no more than M.

14.3.5 Branching

Many of the models we use for individual neurons incorporate branching anatomical data. We consider two numerical procedures for solving PDE neuronal models with branching. Both assume that we employ an implicit finite difference method to the PDEs, and so we will really be considering how to solve the tridiagonal-like linear systems which arise from these discretizations. The first method we consider uses a careful numbering of the unknowns on the branching structure to reduce the resulting linear system to what is essentially a single tridiagonal system. The second method uses the technique of domain decomposition to reduce the solution of the single system of equations on the entire branched structure to the solution of many smaller tridiagonal systems.

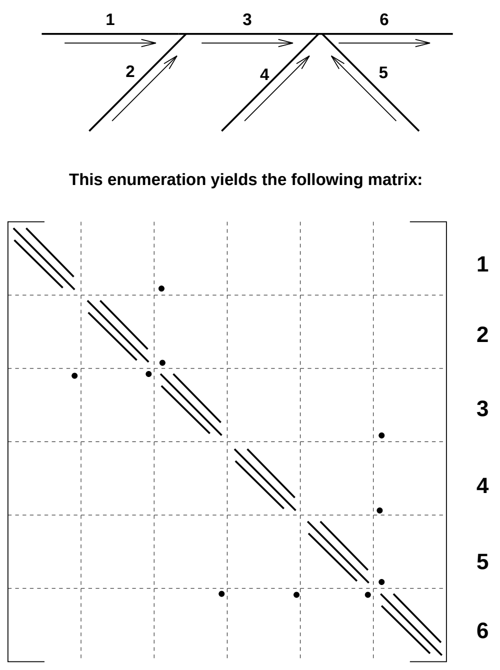

Figure 14.1: Example of branch and grid point numbering of a branched neuronal structure and the resulting linear equation structure which arises with nearest neighbor finite difference discretization. After Hines (1984).

Hines (1984) describes an enumeration of the grid points in a branching cable structure that leads to direct solution of the resulting finite difference equations which is equivalent in complexity to the Thomas algorithm. We illustrate with the example in figure 1, which is a diagram of a neuronal structure with 6 branches. We first choose a branch which is connected at only one end, and designate this as the “trunk”. (A natural choice in a dendritic tree would be the most proximal branch to the soma.) The “trunk” will be the highest numbered branch, in our example branch 6. The grid points in the “trunk” are ordered from the branch point towards the free end. We next number the branches which connect to the “trunk”. Those that are only connected to the “trunk” receive the largest numbers while those with two connections are numbered lowest. The grid points in each branch are ordered towards the “trunk”. Now we designate all branches connected at both ends as new “trunks” and continue the branch and grid point enumeration recursively.

Forward elimination in the branching case is exactly the same as in the nonbranching case except that we must also eliminate all the far off-diagonal elements associated with the several nearest neighbor grid points at a branch point. We proceed in order within the branches. Our chosen enumeration ensures that the only far sub-diagonal element on a branch is associated with the last point. The order of elimination of the branches is also important for assuring that no new off diagonal elements are created. One must carry out forward elimination on all daughter branches before doing so to their parents. We therefore start with the lowest numbered branches (the twigs), and sweep towards the trunk. With the numbering in our example, we can upper-triangularize the branches in order, from 1 to 6, although other orderings, such as 1, 2, 4, 5, 3, 6, are also permissible.

We then proceed to back substitution, marching backwards through the unknowns along each branch. The only far super-diagonal elements are associated with the first point. The order of the branches in back substitution is essentially the reverse of their order in forward elimination: We must process each parent branch before its children. In our example we can simply proceed in reverse order from 6 to 1, although again other orderings are permissible.

Since this method is a mere reworking of Gaussian elimination without pivoting it is obvious that $O(M)$ arithmetic operations and $O(M)$ storage are required to solve a branched structure with M grid points. If the rules above are observed, fill-in can be avoided. On the other hand, if the ordering along branch 1 is reversed, fill-in will result.

The second approach to branching is to solve small tridiagonal systems on the individual branches and form the solution on the entire branching structure out of a linear combination of these local solutions. This technique is known generally as domain decomposition, which is the solution of a problem on one domain by solving smaller problems on subdomains and then building up the whole solution from these smaller parts. Since we are solving a linear equation, we can exploit linear superposition. The simplest example of domain decomposition for a branching one-dimensional structure is illustrated in figure 2. Here we take a single cable and assume that an interior grid point is in fact a branch point. If we assume the value of the solution at the branch point, V_{Br}, is zero, we can solve the tridiagonal systems for the voltages on both branches. We call these solutions V_l^0 and V_r^0 for left and right. In general the value at the branch point will not be zero. To take this into account call V_l^1 and V_r^1 the solutions to the tridiagonal systems on each branch with the right hand side zero except for the contribution from $V_{Br} = 1$. By the principle of superposition, the solution on the left branch is $V_l^0 + V_{Br} V_l^1$, while $V_r^0 + V_{Br} V_r^1$, gives the solution on the right. A single equation for V_{Br} involves the nearest neighbors on the left and right branch. This then gives us the value for V_{Br} which is used to give us the complete solution on each branch via superposition.

In case a branch point is connected to more than two branches, the above procedure still produces the solution on each branch. In this case the solution is a linear combination of three tridiagonal solutions. The solution to the tridiagonal system with normal right hand side but with both branch end points zero; the solution with zero right hand side with one branch point one; and the solution with the other branch point one. Thus we can use this domain decomposition method to solve the equations on any branching structure. Since we only solve several tridiagonal systems to obtain the overall solution, the arithmetic complexity for a structure with M unknowns is still $O(M)$.

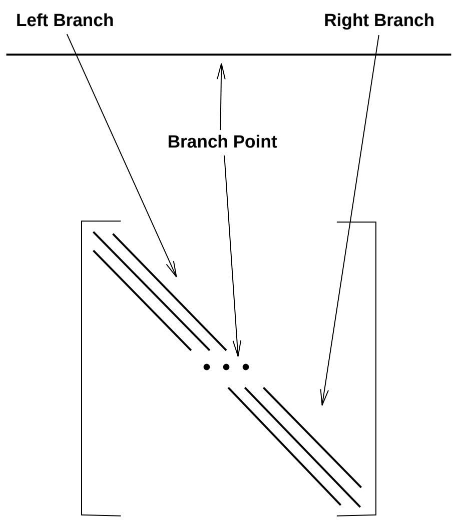

Figure 14.2: An example of domain decomposition for a branching one-dimensional structure using the simplest geometry.

14.3.6 Nonlinear Equations

Up to now we have only considered finite difference methods for linear PDEs. As we know, many of the most interesting PDE models in neuroscience are highly nonlinear, e.g. the Hodgkin and Huxley model. It is a trivial matter to use the forward Euler method on the Hodgkin-Huxley equations. One need only evaluate the nonlinear currents at the old time step value to do so. This involves the computation of the dimensionless variables at the old time value. The forward Euler equation for m_i^{n+1} is simply

$$\begin{aligned} m_i^{n+1} &= m_i^n + \Delta t\big[(1-m_i^n)\alpha_m(V_i^n) - m_i^n\beta_m(V_i^n)\big] \qquad (14.45) \\ &= m_i^n\big[1-\Delta t(\alpha_m(V_i^n)+\beta_m(V_i^n)\big] + \Delta t\alpha_m(V_i^n). \end{aligned}$$

The expressions for h_i^{n+1} and n_i^{n+1} are analogous. The finite difference equation for the voltage via the forward Euler method is then

$$V_i^{n+1} = \sigma(V_{i+1}^n + V_{i-1}^n) + (1-2\sigma-\gamma_i^n)V_i^n + \omega_i^n. \qquad (14.46)$$

Here $\sigma = a\Delta t/2RC(\Delta x)^2$, $\gamma_i^n = (\Delta t/C)\big(\overline{g}_{Na}(m_i^n)^3h_i^n + \overline{g}_K(n_i^n)^4 + \overline{g}_{leak}\big)$, and $\omega_i^n = (\Delta t/C)\big(\overline{g}_{Na}(m_i^n)^3h_i^nE_{Na} + \overline{g}_K(n_i^n)^4E_K + \overline{g}_{leak}E_{leak}\big)$.

Generalizing the two implicit methods we have discussed to nonlinear equations raises more complicated issues. Because implicit methods require the solution of difference equations, when applied to nonlinear PDEs nonlinear difference equations arise. The Hodgkin-Huxley equations have a property that we will call "conditionally linearity." Conditionally linear means that the PDE for the voltage, eq. 14.3, is a linear PDE if the values of the dimensionless variables are known. Similarly, the rate equations for the dimensionless variables, eqs. 14.2, are linear ODEs given values for V.

This is a fairly generic property of nonlinear neuronal models; however, it is by no means universal. The nonlinearities in eq. 14.3 are associated with the nonlinear ionic currents. These nonlinear currents are modeled as the product of a nonlinear ionic conductance and the difference between the membrane potential and the ionic reversal potential. Since the nonlinear ionic conductances are not directly functions of voltage, eq. 14.3 is conditionally linear. Therefore if we can model our nonlinear ionic currents as the products of the difference between the membrane voltage and the ionic reversal potential and nonlinear ionic conductances which are not directly functions of the membrane conductance, we will have nonlinear PDE models of neurons which are conditionally linear.

What are the computational benefits of conditional linearity? Simply stated, the benefits are the separation of the complicated nonlinear problem into two simpler linear problems. This is ramified in different ways for the backward Euler and Crank-Nicolson methods. For the backward Euler method, separability gives us an algorithm for iterative solution of the nonlinear difference equations encountered at each time step via functional or Picard iteration. With Crank-Nicolson, separability reduces the nonlinear equations into a single set of linear equations that can be solved at each time step through a staggered step procedure.

If we use the backward Euler method to discretize the Hodgkin-Huxley PDE, eq. 14.3, we arrive at the following system of nonlinear equations:

$$-\sigma V_{i+1}^{n+1} + (1 + 2\sigma + \gamma_i^{n+1})V_i^{n+1} - \sigma V_{i-1}^{n+1} = V_i^n + \omega_i^{n+1}. \tag{14.47}$$

Here the definition of γ_i and ω_i is as above. These nonlinear difference equations must be solved for the values of V^{n+1}. To make matters worse, if we use the backward Euler discretization on the dimensionless variables as well, then eq. 14.47 must be solved in conjunction with the nonlinear difference equations that arise. The equations for m are:

$$\begin{aligned} m_i^{n+1} &= \frac{m_i^n}{1 + \Delta t[\alpha_m(V_i^{n+1}) + \beta_m(V_i^{n+1})]} + \\ &\quad \frac{\Delta t \alpha_m(V_i^{n+1})}{1 + \Delta t[\alpha_m(V_i^{n+1}) + \beta_m(V_i^{n+1})]}. \end{aligned} \tag{14.48}$$

Equations for h and n are identical in form to eq. 14.48.

The backward Euler method for the Hodgkin-Huxley system requires the simultaneous solution of eq. 14.47 and three equations of the form of eq. 14.48 at each time step. We can hope to solve these nonlinear equations with the following iterative algorithm: **1**. Solve eq. 14.47 with the previously known dimensionless variable values to give new voltage values. **2**. Solve the dimensionless variable equations, eq. 14.48 etc., using the new voltage values to give new dimensionless variable values. **3**. Repeat steps **1** and **2** until voltage and dimensionless variable values have converged. The starting values for this iterative procedure are the values of the unknowns at the previous time step. Since these equations are conditionally linear and hence separable, the equations to be solved in steps 1 and 2 will always be *linear*. The voltage equations become a single linear tridiagonal system, and the equations for the dimensionless variables are reduced to explicit expressions for the new values. This iterative procedure is sometimes called a Picard, fixed-point, or functional iteration. For the backward Euler method applied to the Hodgkin-Huxley equations this iteration method has been proven to converge provided that Δt is chosen sufficiently small (Mascagni, 1987a).

As with the backward Euler method, the Crank-Nicolson method applied to the Hodgkin-Huxley equations leads to nonlinear equations:

$$\begin{aligned}-\frac{\sigma}{2}V_{i+1}^{n+1} + (1+\sigma+\frac{\gamma_i^{n+1}}{2})V_i^{n+1} - \frac{\sigma}{2}V_{i-1}^{n+1} = \\ \frac{\sigma}{2}V_{i+1}^{n} + (1-\sigma-\frac{\gamma_i^{n}}{2})V_i^{n} + \frac{\sigma}{2}V_{i-1}^{n} + \frac{\omega_i^{n+1}+\omega_i^n}{2}.\end{aligned} \tag{14.49}$$

These must be solved concurrently with the difference equations for the dimensionless variables. If we wish to maintain an overall accuracy of $O((\Delta t)^2)$, we must use a method of at least second order accuracy for the rate equations. Of course we know that the trapezoidal rule is second order accurate and is the ODE analog of the Crank-Nicolson method. Applying the trapezoidal rule to the equation for m gives us:

$$\begin{aligned}m_i^{n+1} &= m_i^n \frac{1-\Delta t/2[\alpha_m(V_i^{n+1/2})+\beta_m(V_i^{n+1/2})]}{1+\Delta t/2[\alpha_m(V_i^{n+1/2})+\beta_m(V_i^{n+1/2})]} + \\ &\quad \frac{\Delta t\alpha_m(V_i^{n+1/2})}{1+\Delta t/2[\alpha_m(V_i^{n+1/2})+\beta_m(V_i^{n+1/2})]}.\end{aligned} \tag{14.50}$$

We have used the values $V_i^{n+1/2}$ in the above expression to simplify the form of these difference equations while maintaining the desired second order accuracy. One can use the average of the voltage at the nth and $n+1$st time step to estimate the value at time value $n+1/2$, or one may use another method we discuss below.

Eqs. 14.49 and 14.50 and the analogous equations for h and n are a complicated set of simultaneous nonlinear equations. The fact that these equations are conditionally linear can be exploited to give us an iterative algorithm for their simultaneous solution. There is no rigorous proof for the convergence of this method for the Crank-Nicolson method, but it is believed that the techniques used in the proof of the backward Euler case can be easily extended to this case. We should note that the iterative solution method described for the backward Euler method can be used to solve the nonlinear equations we get with the Crank-Nicolson discretization. In fact, this method was used by Cooley and Dodge (1966), in a predictor-corrector variant of the method we described, to do the first systematic numerical simulation of the Hodgkin-Huxley equations.

Another approach to these equations, which exploits their conditional linearity and results in no iteration, is to use an implementational detail we have previously mentioned and the second order accuracy of the trapezoidal rule. Recall that if V^{n+1} is the backward Euler solution to the linear cable equation, eq. 14.23, then $2V^{n+1} - V^n$ is the Crank-Nicolson solution. If one uses eqs. 14.47 with ω_i^{n+1} replaced with $\omega_i^{n+1/2}$, then $2V^{n+1} - V^n$ gives us a second order accurate solution to the Hodgkin-Huxley equations. Note that the definition of ω_i^n depends on the time level, n, only through the dimensionless variables. Thus we need only compute the values $m_i^{n+1/2}$, $h_i^{n+1/2}$, and $n_i^{n+1/2}$ to obtain $\omega_i^{n+1/2}$. This procedure requires values of the dimensionless variables at the midpoints of the time levels used in the voltage equations. If one were to solve the dimensionless variable equations at only the midpoint values, and the voltage values at the usual values, then they could be combined to give an overall second order method. This staggering of two time grids also simplifies the solution of the difference equations. Since the Hodgkin-Huxley equations are conditionally linear, this grid staggering leads to solving a single tridiagonal system for the voltage equation and evaluating the explicit equations for the dimensionless variables a single time to advance the solution Δt. One complication is the computation of the unknowns on the two time grids staggered by $\Delta t/2$. One can accomplish this by starting either the voltage or dimensionless variable equations with a $\Delta t/2$ sized time step and then proceeding as normal.

With the backward Euler and Crank-Nicolson discretizations of the Hodgkin-Huxley equations we have shown two different ways in which conditional linearity can be exploited to simplify the numerical solution of these nonlinear systems. Similarly, one can use the staggering method we presented for the Crank-Nicolson method with the backward Euler equations. However, since staggering is used to maintain $O((\Delta t)^2)$ accuracy in Crank-Nicolson, no staggering is required with the $O(\Delta t)$ backward Euler method. Thus the minor complication of the staggered grids for Crank-Nicolson disappears if we are happy with using backward Euler. Thus one obtains an $O(\Delta t)$ implicit solution without iteration.

A warning must be made for users of the staggering method with Crank-Nicolson. This method is efficient and produces an $O((\Delta t)^2)$ solution to nonlinear problems without iteration. However, this is true only if the equations are conditionally linear. If one had a cable model that was not conditionally linear, then this method would not be assured of producing an $O((\Delta t)^2)$ solution. For example, if some ionic currents are explicitly nonlinear functions of the voltage, then the model is not conditionally linear. The nonlinear equations arising from the implicit discretization can be solved using a functional (Picard) iteration or Newton's method. Alternatively, one can restore conditional linearity by expanding the nonlinear currents in a Taylor series and evaluating their first derivatives at the half time points along with the conductances. An option based on this idea is included in `Neuron`.

Experience in computing solutions to the Hodgkin-Huxley PDEs has shown that the majority of the effort is spent in the evaluation of the α and β rate functions associated with the dimensionless variables. Because of this fact, it is considered prudent to use a lookup table to speed up the repeated evaluation of these functions. If one examines the equations for the advancement of the dimensionless variables, eqs. 14.45, 14.48, and 14.50, one will notice that they are all of the form $m^{n+1} = m^n K_1(V, \Delta t) + K_2(V, \Delta t)$. Thus we need only evaluate the two function, K_1 and K_2, for each dimensionless variable at each time step. The functions K_1 and K_2, which involve combinations of the α and β functions, can be tabulated for the purpose of speed. If one does not plan to change the time step size, Δt, during a computation, the functions K_1 and K_2 can be tabulated once for all during the initialization of the computation.

It is most convenient to construct the lookup table over a wide range of possible voltage values using a uniform voltage increment. Experience has shown that the voltage range of -100 millivolts to 150 millivolts with a step of 0.1 millivolt will give sufficient accuracy for the Hodgkin-Huxley equations. With piecewise quadratic interpolation, experience has shown that a step as large as 4.0 millivolts can be used with no degradation in the overall accuracy. Interpolation with equally spaced points in a lookup table is handled quite well in Conte and de Boor, (1980).

Unlike linear PDEs, there is no neat Lax Equivalence Theorem for finite difference methods applied to nonlinear PDEs. The analysis of these methods must instead proceed via a case by case approach. Fortunately, convergence of the backward Euler method for the Hodgkin-Huxley equations has been proven (Mascagni, 1987a). In addition, there is a considerable body of results on the convergence of the Crank-Nicolson method for many classes of nonlinear parabolic PDEs (Douglas, 1961; Lees, 1959; Rose, 1956). It appears that for both the backward Euler and Crank-Nicolson methods, no relationship between Δt and Δx need hold; however, it is known for the backward Euler method and likewise conjectured for the Crank-Nicolson method that convergence occurs for all values of Δt smaller than a certain maximal value (Mascagni, 1987a).

14.3.7 Networks

Synthesizing the results of the previous six sections gives us the ability to numerically simulate a single arbitrarily branched neuronal model with spatial variation and nonlinear ionic kinetics. This is already a significant capability; however, with new developments in computer technology, it is possible to solve problems of much greater complexity than merely a single neuron. The new technical capabilities provide us with the opportunity to model several hundred very complicated neurons which are synaptically connected to one another. The neuronal modeler should bear these new developments in mind when considering what types of questions to ask and what type of models to build.

The key to modeling a network of neurons is the selection of a model for synaptic conduction that is both biophysically satisfying and is compatible with the finite difference approach for the numerical solution of the nerve equations. To a large extent this is a matter of personal taste. A general guideline for this selection is that a deterministic model of synaptic conduction is most easily incorporated into a numerical scheme involving finite difference approximations to nerve equations. For example, such a model might be the initiation of a characteristic postsynaptic conduction change in response to the presynaptic arrival of an action potential. This could be further embellished with a stochastic determination of the shape of the elicited postsynaptic conductance change. The types of models of this general form which cause numerical and implementational difficulties are those in which the repertoire of possible postsynaptic conductance time courses is large. In a computation that has many interacting neuronal elements, the accurate determination of the postsynaptic membrane potential requires the summation over the recent synaptic events. If there are potentially many varied forms which these synaptic events effect the postsynaptic potential, a considerable amount of computer memory must be used to implement the complicated bookkeeping task underlying the determination of the postsynaptic membrane potential. In a network of p neurons which are totally interconnected, $O(p^2)$ contributions to the postsynaptic potentials must be calculated each time step. This quadratic scaling with the number of neurons quickly dominates the memory requirements of the computation, and it does so even more quickly when a considerable amount of memory is required to store a single postsynaptic event.

14.3.8 Concluding Remarks and Suggestions for PDEs

In deciding on one of these methods over another, considerations similar to those discussed for choosing finite difference methods for ODEs arise. In general, it takes no more computational work to use an implicit method than an explicit method, so one should use either the backward Euler or Crank-Nicolson method. The backward Euler method with iteration has been used with great success for investigating certain problems (Mascagni, 1987b), and the observed solutions remain qualitatively correct for rather large values of Δt. The Crank-Nicolson method without iteration was first described in its entirety by Hines (1984). This method has also been used with great success for certain computations (Mascagni, 1989b). The authors have observed that with the Crank-Nicolson method one cannot use values of Δt as large as with the backward Euler method due to ringing instabilities. This is true, for example, when modeling a voltage-clamp step, which introduces high-frequency Fourier components into the solution. In circumstances where using a large Δt is necessary and one is willing to sacrifice quantitative accuracy, the backward Euler method is preferable to the Crank-Nicolson method. However, when accuracy is vital it is recommended that the second order accurate method be used, taking care to use Δt small enough to prevent ringing instabilities from contaminating the computation. An alternative way to damp instabilities that avoids the complication of implicit equations is the *exponential Euler* method. However, it can be shown that the method converges only if $\Delta t/(\Delta x)^2 \to 0$, a condition that is far more restrictive than the *stability* condition for forward Euler. Thus, the solutions do not blow up, but they may be highly inaccurate.

One can also use one of the several neural simulation packages that include solvers for compartmental or PDE models. These include `GENESIS` and `Neuron` for Unix workstations and `Nodus` for Macintosh. `Neuron` is also available for Windows. `GENESIS` and `Neuron` include the backward Euler and Crank-Nicolson methods, and so are efficient on stiff problems. The time required to learn these packages may be greater than writing a simple solver for a particular problem. However, they permit the user to define the problem in terms of objects that are natural to neural modeling, such as channels and geometrical features, rather than through equations. Also, once learned, these packages allow much flexibility to change the problem and to perform many numerical experiments. Finally, most of these packages have capabilities for producing highly useful graphical output. As for the ODE packages, further information can be found on the Web at URL: `http://mrb.niddk.nih.gov/sherman`.

There are several other problems that might arise in neuronal modeling that produce PDEs for which we have not discussed methods of solution. In fact, we really have only treated one-dimensional parabolic PDE models. In some cases more than one spatial dimension must be included into a PDE model. Then, one can use finite element methods or extensions of the finite difference methods we have presented here. Whether a finite element or finite difference method is used, the resulting diagonally dominant system that arises from an implicit discretization is pentadiagonal in two dimensions and heptadiagonal in three. The diagonals are not contiguous, however, introducing problems of fill-in that can ruin the sparsity of the matrices. There are special variants of Gaussian elimination for the efficient solution of such problems. However, a particularly efficient method in the finite difference case is the **A**lternating **D**irection **I**mplicit method (Richtmyer and Morton, 1967; Press et al., 1992). The **ADI** method produces an implicit solution to a two- or three-dimensional problem by solving several one-dimensional problems alternately in each of the dimensions.

There are also methods for solving PDEs that offer higher order accuracy in space and/or time. Methods that are higher order in space complicate both the matrix equations that must be solved at each time step and boundary conditions. Thus they are not often used. Methods that are higher order in time can be obtained by using the method of lines with a high-order ODE solver. A version of the Gear algorithm, for example, may be appropriate for problems in which stiffness originates in the kinetics of the problem, rather than from diffusion term of the PDE. However, one must also bear in mind that working hard to refine the temporal discretization is wasted if the spatial accuracy remains at $O((\Delta x)^2)$. Thus it is important to balance spatial and temporal accuracy when solving PDE neuronal models.

14.4 Final Comments

We have surveyed the current wisdom on the best solutions to what might be called the easy problems in neuronal modeling. Small systems of ODEs, even stiff ones, can be solved very efficiently to high accuracy. PDE's are naturally more difficult, but reasonable methods (i.e. second-order accurate, $O(N)$ work) are available for one-dimensional problems, including branched neurons. It is of course not difficult to come up with problems that will confound the best algorithms on the fastest computers, e.g. any problem with stiff kinetics in two or three spatial dimensions. We have consciously avoided venturing into these areas, both because of our own limitations and the limitations of the field as a whole.

In addition to the particular advice we have sprinkled throughout, we conclude here with some general concepts that are relevant to problems on all scales of difficulty. Numerical methods are fallible. Some may have considerable artificial intelligence built into them, but in the end there is no alternative to a deep knowledge of the particular physical problem on the part of the investigator. General dynamical systems theory can be very helpful because it categorizes possible and impossible behaviors.

There is no algorithm that solves all problems, and the user must know enough to adapt the tool to the job. It also pays to solve a problem by more than one method. That means supplementing numerical methods with analytical methods and also using more than one numerical method. In addition to catching routine errors, this may uncover very subtle ones. In one small but illuminating example that we know of, an instability in the dynamical system was sensitive to numerical error introduced by the Gear method, but not Runge-Kutta, leading to discovery of a new class of phenomena (Sherman and Rinzel, 1992). No computer program can be expected to anticipate such cases. Ultimately computational science is isomorphic to all of science, and can no more than all of science ever be complete.

14.5 Appendix A: Stiffness

Let us consider the PDE model of the squid axon, eqs. 14.2 and 14.3, to illustrate the method of lines and their relationship to compartmental models. In eq. 14.3, the right hand side has a term involving a second spatial derivative. If we replace the continuous x-axis with a uniform grid of width Δx, and replace the term $\partial^2 V/\partial x^2$ with the familiar $O((\Delta x)^2)$ finite difference approximation $(V_{i+1} - 2V_i + V_{i-1})/(\Delta x)^2$, the PDE is reduced to a system of coupled ODEs. This is the method of lines, where a single PDE is reduced to a system of ODEs by discretizing all but one of the independent variables in the PDE. The resulting method of lines ODE system for the Hodgkin-Huxley PDE model is stiff. This fact is well known from results for the method of lines for the PDE which describes diffusion, $\partial V/\partial t = \partial^2 V/\partial x^2$, which displays the same qualitative stiffness properties.

A compartmental model which uses individual ODE models for each compartment and where neighboring compartments are resistively coupled is equivalent to a method of lines discretization of some PDE model. This is easily seen by observing that we can rewrite our difference formula for the second derivative as $\big((V_{i+1} - V_i)/\Delta x - (V_i - V_{i-1})/\Delta x\big)/\Delta x$. Now assume that the spatial discretization is no longer uniform. If the distance between grid point $i+1$ and i is denoted by Δx_1, while the distance between grid point i and $i-1$ is called Δx_2, then the second order accurate approximation to the second derivative on this nonuniform grid is given as $\big((V_{i+1} - V_i)/\Delta x_1 - (V_i - V_{i-1})/\Delta x_2\big)/\big((\Delta x_1 + \Delta x_2)/2\big)$. This can be rewritten as $LV_{i-1} + DV_1 + UV_{i+1}$, where L, D, and U are constants. This is exactly the form that one encounters in nearest neighbor resistively coupled compartmental models units, where the ODE at one compartment depends on the voltage values of the central and the two flanking compartments.

We will now perform an elementary calculation that will explicitly compute the stiffness of a linear compartmental cable model. In our above mentioned example on the method of lines, eqs. 14.24, we arrived at a system of coupled linear ODEs to describe the time evolution of a linear compartmental model. These ODEs came from a PDE model of a passive cable, and so we must impose certain boundary conditions to correctly specify the mathematical problem. For simplicity, let us assume that we desire the solution to eq. 14.24 with zero Dirichlet boundary conditions, i.e. $V_0 = V_N = 0$. If we define the vector of voltages on the grid to be $\vec{V} = (V_1, V_2, \ldots, V_{N-1})$, we can rewrite eq. 14.24 as the following linear system of ODEs:

$$\frac{d\vec{V}}{dt} = \mathbf{A}\vec{V}. \tag{14.51}$$

The matrix $\mathbf{A}$ is a tridiagonal matrix with the additional property that $\mathbf{A}$ is Toeplitz. A Toeplitz matrix has constant diagonals. Thus if the elements of $\mathbf{A}$ are denoted as a_{ij}, then $a_{ij} = K_d$ whenever $i - j = d$ for $\mathbf{A}$ Toeplitz. Hence our Toeplitz, tridiagonal $\mathbf{A}$ has $a/2RC(\Delta x)^2$ on both off diagonals and $-(a/RC(\Delta x)^2 + \overline{g}/C)$ along the main diagonal. Since eq. 14.51 is a linear system of ODEs, the general solution to this system can be expressed as a linear combination of exponential functions of the form $e^{\lambda t}$, where λ is an eigenvalue of the matrix $\mathbf{A}$. Since we are interested in computing the stiffness of this system, which we recall is the ratio of the largest to smallest time scale in the problem, we need only compute λ_{N-1}/λ_1, which is the ratio of the largest to the smallest eigenvalue of $\mathbf{A}$. It should be noted that this ratio of eigenvalues is exactly the l_2-norm condition number of this matrix (Stoer and Bulirsch, 1980).

The eigenvalues of the tridiagonal Toeplitz matrix $\mathbf{A}$ are known to be (Isaacson and Keller, 1966):

$$\lambda_k = \frac{2a}{RC(\Delta x)^2}\sin^2\!\left(\frac{k\pi}{2N}\right) + \frac{\overline{g}}{C}, \quad k = 1, \ldots, N-1. \tag{14.52}$$

Thus the numerical stiffness is given by:

$$\frac{\frac{2a}{RC(\Delta x)^2}\sin^2\!\left(\frac{(N-1)\pi}{2N}\right) + \frac{\overline{g}}{C}}{\frac{2a}{RC(\Delta x)^2}\sin^2\!\left(\frac{\pi}{2N}\right) + \frac{\overline{g}}{C}} = \frac{\sin^2\!\left(\frac{(N-1)\pi}{2N}\right) + \rho}{\sin^2\!\left(\frac{\pi}{2N}\right) + \rho}, \tag{14.53}$$

where $\rho = \frac{2\overline{g}R(\Delta x)^2}{a}$. The constant ρ is a parameter which measures the dissipation of this particular cable model. Large values of ρ mean the system relaxes due to this dissipation much faster than due to diffusion.

With this explicit formula for the stiffness, we can ask what happens to the stiffness of this system as we vary certain parameters, namely ρ and N. Since $0 \leq \sin^2(x) \leq 1$, if ρ is large, then eq. 14.53 has a numerical value close to 1, and so the system is in fact not very stiff. Thus, if the system is very dissipative, it is numerically well behaved. On the other hand, if ρ is small, then the stiffness is approximately the ratio of the sine terms in eq. 14.53. It can be shown that $\sin^2(\frac{(N-1)\pi}{2N})/\sin^2(\frac{\pi}{2N}) = O(N^2)$ as N gets large. Thus for small values of ρ, the stiffness gets extremely large as we increase the number of compartments in our cable model. In fact, it grows as the square of the number of compartments in our models. Thus we see that compartmental models can be very stiff when there is little dissipation via the membrane conductance. Paradoxically this stiffness increases as we use smaller and smaller compartments to better resolve spatial details. This relationship of stiffness to the number of compartments is related to the inequality that must be satisfied for numerical stability of the forward Euler method for these PDEs (Lambert, 1973).

Acknowledgments

The authors would like to thank David Golomb, Larry Abbott, Misha Tsodyks, and David Hansel for discussions of integrate and fire problems, and Paul Smolen, Todd Geldon, and Mike Vanier for calculations with the exponential Euler method.

14.6 References

Back, A., Guckenheimer, J., Myers, M., Wicklin, F. and Worfolk, P. (1992) *dstool*: Computer assisted exploration of dynamical systems. *Notices of the Am. Math. Soc.* **39**:303 – 309.

Baer, S. M., and Tier, C. (1986) An analysis of a dendritic neuron model with an active membrane site. *J. Math. Biol.* **23**: 137–161.

Boyce, W. E. and DiPrima, R. C. (1992) *Elementary Differential Equations and Boundary Value Problems*, John Wiley and Sons, Inc., New York.

Clay, J. R., and DeFelice, L. J. (1983) Relationship between membrane excitability and single channel open-close kinetics. *Biophys. J.* **42**: 151–157.

Conte, S. D., and de Boor, C. (1980), *Elementary Numerical Analysis An Algorithmic Approach*, Third Edition, McGraw-Hill Book Company, New York.

Cooley, J.W., Dodge, F.A. (1966) Digital computer solutions for excitation and propagation of the nerve impulse. *Biophys. J.* **6**: 583–599.

Crank, J. and Nicolson, P. (1947) A practical method for numerical evaluation of solutions of partial differential equations of the heat conduction type. *Proc. Camb. Phil. Soc.* **43**: 50–67.

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

Doedel, E. J., Keller, H. B., and Kernévez, J. P. (1991) Numerical analysis and control of bifurcation problems, Part I: Bifurcation in finite dimensions. *Int. J. Bifurcation and Chaos* **1**: 493–520.

Douglas, J. Jr. (1961) A survey of numerical methods for parabolic differential equations. In: *Advances in Computers*, vol. 2, ed. F. Alt, Academic Press, New York, pp. 1–54.

Ermentrout, B. (1990) *PhasePlane: The Dynamical Systems Tool*, Brooks/Cole, Pacific Grove, California.

Fox, R. F., and Lu, Y.-N. (1994) Emergent collective behavior in large numbers of globally coupled independently stochastic ion channels, *Phys. Rev. E* **49**: 3421–3431.

Gear, C. W. (1971a) *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, New Jersey.

Gear, C. W. (1971b) The automatic integration of ordinary differential equations. *Comm. ACM* **14**: 176–179.

Golub, G. H. and Ortega, J. M. (1992) *Scientific Computing and Differential Equations*, Academic Press, Boston.

Golub, G. H, and Van Loan, C. F. (1985) *Matrix Computations*, Johns Hopkins University Press, Baltimore.

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

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

Hubbard, J. H. and West, B. (1992) *MacMath: A Dynamical Systems Software Package for the Macintosh*, Springer-Verlag, New York.

Isaacson, E. and Keller, H. B. (1966) *Analysis of Numerical Methods*, John Wiley and Sons, Inc., New York.

Jack, J.J.B., Noble, D., and Tsien, R.W. (1983) *Electrical Current Flow in Excitable Cells*, Second Edition, Clarendon Press, Oxford.

John, F. (1952) On integration of parabolic differential equations by difference methods. *Comm. Pure Appl. Math.* **5**: 155–211.

John, F. (1982) *Partial Differential Equations*, Fourth Edition, Springer-Verlag, Heidelberg.

Keller, H. B. (1976) *Numerical Solution of Two Point Boundary Problems*, CMBS-NSF Regional Conference Series in Applied Mathematics, number 24, SIAM, Philadelphia.

Kloeden, P., Platen, E., and Schurz, H. (1993) *Numerical Solution of Stochastic Differential Equations Through Computer Experiments*, Springer-Verlag, Heidelberg.

Koçak, H. (1989) *Differential and Difference Equations through Computer Experiments*, Springer-Verlag, New York.

Kuramoto, Y. (1991) Collective synchronization of pulse-coupled oscillators and excitable units. *Physica D* **50**: 15–30.

Lambert, J. D. (1973) *Computational Methods in Ordinary Differential Equations*, John Wiley and Sons, Inc., New York.

Lees, M. (1959) Approximate solution of parabolic equations. *J. SIAM* **7**: 167–183.

Li, Y.-X., Rinzel, J., Vergara, L., and Stojilković, S. S. (1995) Spontaneous electrical and calcium oscillations in unstimulated pituitary gonadotrophs. *Biophys. J.* **69**: 785–795.

Lin, C. C. and Segel, L. A. (1988) *Mathematics Applied to Deterministic Problems in the Natural Sciences*, SIAM, Philadelphia.

Mascagni, M. (1987a) *Negative Feedback in Neural Networks*, Doctoral Dissertation in Mathematics, Courant Institute of Mathematical Sciences, New York University.

Mascagni, M. (1987b) Computer simulation of negative feedback in neurons. *Soc. for Neurosci. Abst.* **13**: 375.4.

Mascagni, M. (1989a) An initial-boundary value problem of physiological significance for equations of nerve conduction, *Comm. Pure Appl. Math.* **42**: 213-227.

Mascagni, M. (1989b) Animation's role in modeling the nervous system, *Iris Universe* **Winter 1989**: 6-18.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P., (1992) *Numerical Recipes: The Art of Scientific Computing*, second edition, Cambridge University Press, Cambridge.

Richtmyer, R. D. and Morton, K. W. (1967) *Difference Methods for Initial-Value Problems*, second edition, Interscience Publishers, division of John Wiley and Sons, Inc., New York.

Rose, M. E. (1956) On the integration of nonlinear parabolic equations by implicit methods. *Quart. Appl. Math.* **15**: 237–248.

Sherman, A. and Rinzel, J. (1992) Rhythmogenic effects of weak electrotonic coupling in neuronal models, *Proc. Natl. Acad. Sci. USA* **89**: 2471–2474.

Sod, G. A. (1985) *Numerical Methods in Fluid Dynamics: Initial and Initial Boundary-Value Problems*, Cambridge University Press, Cambridge.

Stoer, J. and Bulirsch, R. (1980) *Introduction to Numerical Analysis*, Springer Verlag, Heidelberg.

Tsodyks M., Mit'kov, I., and Sompolinsky, H. (1993) Pattern of Synchrony in Inhomogeneous Networks of Oscillators with Pulse Interactions. *Phys. Rev. Lett.* **71**, 1280–1283.