Numerical Integration in Computational Neuroscience

The aim of this article is to provide an intuitive introduction to numerical integration in the context of simulating model neurons. It begins with a brief discussion of the nature of the equations that are to be solved, then examines some approaches that have been employed to solve them, and ends with some comments to help users apply numerical integration to their own models.

The Nature of the Problem

Most computational models of biological neurons and networks are used to study the origin, propagation, and interaction of electrical signals. For this reason, the focus here is on numerical solution of initial value problems that involve the cable equation. It has been noted that the deterministic diffusion of chemical signals is described by equations that are formally equivalent to those that govern the spread of electrical signals, and are amenable to similar computational approaches (Carnevale and Hines 2006; Zador and Koch 1994). Readers who wish more detail on these and related topics are referred to chapter 4 in Carnevale and Hines (2006), as well as Iserles (2008), Polyanin et al. (2008), and Wolfram Research (2013) Phenomena that involve small numbers of molecules or ions require methods that are beyond the scope of this chapter.

The spread of electrical signals along an unbranched neurite is governed by the one dimensional cable equation, a partial differential equation (PDE) that is a statement of conservation of charge in an anatomically extended structure. The cable equation is often written as

$$\frac{\partial V}{\partial T} + F(V) = \frac{\partial^2 V}{\partial X^2} \qquad \text{Eq. 1}$$

where T and X are time and anatomical distance scaled by factors related to the physical size of a neurite and the electrical properties of its membrane and cytoplasm, and V and F are continuous functions of T and X (Rall 1977; Jack et al. 1983). Branched cellular architectures can be accommodated by combining multiple instances of the cable equation (one for each unbranched neurite) with appropriate boundary conditions. An analytical solution to the cable equation would express membrane potential as $v(t,x)$, a continuous function of time and position. In special cases it is possible to solve the cable equation analytically, and those solutions have produced many general insights into neuronal function (Rall 1977; Jack et al. 1983; Segev et al. 1995).

However, models that include the anatomical and biophysical complexities of real cells typically produce equations that do not have analytical solutions. The classical example is the Hodgkin-Huxley model of spike propagation along squid axon (Hodgkin and Huxley 1952)

$$C_m \frac{\partial v}{\partial t} + \bar{g}_{na} m^3 h\,(v - e_{na}) + \bar{g}_k n^4 (v - e_k) + g_l\,(v - e_l) = \frac{D}{4 R_a} \frac{\partial^2 v}{\partial x^2} \qquad \text{Eq. 2}$$

In this equation, the gating variables h, m, and n are governed by first order ordinary differential equations (ODEs) whose rate constants depend on local membrane potential v, and the parameters C_m, $\bar{g}_{na}$, $\bar{g}_k$, and g_l (specific membrane capacitance and channel conductance densities) may vary with position; furthermore. Equations with such complexities must be solved numerically.

Discretization of the Cable Equation

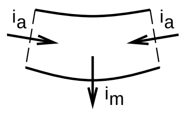
A broad class of strategies for numerical solution of initial value PDEs is based on the notion of discretization. Discretization involves approximating a PDE, whose dependent variables are temporally and spatially continuous, with a set of difference equations that relate the dependent variables to each other at a finite set of points in time and space. This is done by approximating the temporal and spatial derivatives with algebraic difference formulas, and it produces a family of difference equations. The difference equations are then solved algebraically in order to compute the future values of state variables from their present (and possibly past) values. Instead of producing a solution that is a continuous function of time and position, the result is an approximate solution $v_{ij} = v(t_i, x_j)$ over a finite set of points in time and space. The error introduced by discretization depends on the algebraic formulas used to approximate the derivatives, and the size of the temporal and spatial intervals between adjacent solution points.

Here we introduce basic concepts involved in discretization and then explore the stability and precision of various integration methods in the context of some simple models. This discussion is guided by the fact that models of neurons that involve biological anatomical and biophysical details pose a special challenge because the equations that describe them are very stiff, that is, they are characterized by a wide range of time constants. For example, even though neurons typically have membrane time constants on the order of tens of milliseconds, longitudinal redistribution of charge along the length of a neurite has sub-millisecond kinetics, and short neurites or dendritic spines can introduce sub-microsecond time constants. Single compartment model neurons are not immune from stiffness, because some voltage- and ligand-gated ion channels (especially spike sodium channels and AMPAergic synaptic channels) have sub-millisecond time constants. Some voltage-and ligand-gated potassium channels are much slower, on the order of hundreds of milliseconds, while ion accumulation and active transport can produce concentration changes and shifts of membrane potential that evolve over seconds or minutes.

Because of the stiffness of neural models, explicit integration methods (methods that predict future values from present and past values) can require impractically small Δt to avoid instability; this is illustrated below. Also, stability issues aside, the utility of high order methods in fixed time step integration can be limited by the necessity of using short time steps to follow the time course of rapidly changing state variables. Consequently, only a few of the many methods that have been developed for numerical solution of PDEs are suitable for simulating models of biological neurons and networks.

Spatial Discretization

Spatial discretization is the first step in transforming the cable equation to algebraic difference equations. It approximates a spatially continuous physical system by a set of coupled compartments, each of which is small enough that membrane current density can be treated as nearly uniform over its surface. This figure portrays such a region; as indicated by the arrows, axial currents i_a are reckoned as positive if they enter the region, and membrane current i_m is positive when it exits.



If membrane current density varies widely over a model cell, many compartments will be needed to satisfactorily represent this variation. The ODE that describes membrane potential in the *j*th region is

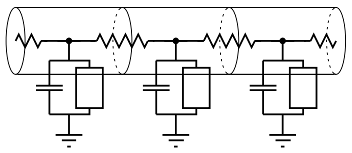
$$c_{m_j} \frac{d v_j}{d t} + i_{ion_j} = \sum_j \frac{v_k - v_j}{r_{jk}} \qquad \text{Eq. 3}$$

where the left hand side is the sum of the capacitive and ionic membrane currents in the *j*th region, and the right hand side is the sum of the axial currents that enter this region from its immediate neighbors.

If the compartments are of equal size and the points at which membrane potential and currents are computed are located at the compartment centers, the right hand side of Eq. 3 is the central difference approximation to the PDE's second spatial derivative

$$\frac{\partial^2 v}{\partial x^2} \approx \frac{v_{j+1} - 2\mathrm{v}_j + v_{j+1}}{\Delta r^2} \qquad \text{Eq. 4}$$

where Δr is the resistance of the cytoplasm between adjacent compartment centers. The equivalent electrical circuit for three adjacent compartments along a neurite that has been discretized with the central difference approximation is



Here each compartment's membrane properties are depicted as the parallel combination of a capacitor and a box that represents the contribution of ion channels to transmembrane current. These are attached to a node in the middle of the compartment, at which membrane potential is to be calculated; each node is separated from adjacent nodes by cytoplasmic resistance.

The accuracy of the central difference approximation is $O(\Delta x^2)$, i.e. the local error introduced by replacing the second partial derivative in space with the central difference approximation is proportional to the square of the distance Δx between adjacent nodes (Iserles 2008). As a consequence, the central difference approximation leads to numerical solutions that can be treated as piecewise linear approximations to the real solution. In other words, if v_j and v_{j+1} are second-order accurate at adjacent nodes x_j and x_{j+1} , linear interpolation can be used to find a order accurate value $v*$ at any point $x*$ that lies between them.

The outcome of spatial discretization can be rearranged and written in vector form as

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{F}(\boldsymbol{y}, t) \qquad \text{Eq. 5}$$

where the elements of $\boldsymbol{y}$ include not only the membrane potentials in each compartment but also state variables associated with other dynamic processes such as ion channel gating and concentrations of ions and second messengers. The right hand side includes t to acknowledge the possible presence of time-varying parameters (e.g. synaptic conductances) or signal sources such as current- or voltage clamps.

Temporal Discretization

Temporal discretization completes the transformation of a PDE to a set of algebraic equations by substituting an algebraic approximation for the temporal derivatives. There is a wide variety of temporal discretization schemes that are characterized by different degrees of stability and orders of accuracy. Those that are most relevant to computational neuroscience are discussed in **Specific Integration Methods** below.

Specific Integration Methods

An ideal numerical integration method would generate highly accurate solutions quickly and robustly regardless of the details of the model. Unfortunately it is not possible to meet all these goals simultaneously. In general, there is a tradeoff between speed, stability, and accuracy.

Explicit Euler method

The explicit Euler method is conceptually simplest and easiest to implement. This is a first-order accurate approach (local error is proportional to Δt) that predicts the solution at the next time t_{i+1} from the values of the state variables and their derivatives at the current time t_i . It approximates the ODE of Eq. 5 with

$$\boldsymbol{y}_{i+1} = \boldsymbol{y}_i + \boldsymbol{F}(\boldsymbol{y}_i, t_i)\,\Delta t \qquad \text{Eq. 6}$$

so advancing a solution from one time to the next requires only calculation of the derivatives followed by some multiplications and additions.

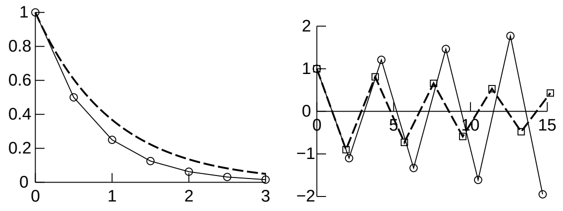
However, the total computational cost of a simulation depends not only on how much effort is needed to advance from one step to the next, but also on the number of advances that must be performed. Explicit Euler's weakness is that Δt must be smaller than the smallest time constant in the system in order to ensure stable solutions that are free of spurious oscillations. For example, consider the ordinary differential equation (ODE)

$$\frac{dy}{dt} = -\frac{y}{\tau} \qquad \text{Eq. 7}$$

According to the explicit Euler method, given y_i at time t_i, the "recurrence equation" that generates y_{i+1} at time t_{i+1} is

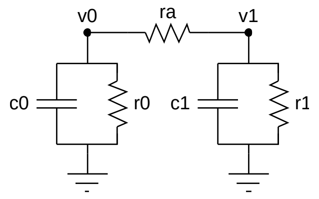
$$y_{i+1} = y_i\,(1 - \Delta t/\tau) \qquad \text{Eq. 8}$$

The following graphs compare the analytical solution for Eq. 7 with $\tau = 1$ and initial condition $y = 1$ (left panel, dashed line) to numerical solutions calculated by the explicit Euler method using different values of Δt. In both graphs the horizontal axis is time in seconds. If $\Delta t < \tau/2$ (left panel, solid line, $\Delta t = 0.5$), the numerical solution evolves smoothly with time and converges toward the analytical solution. Even though there is some error in the numerical solution, the solution is stable because the error decreases at each new time step and approaches 0 in the limit as $t \to \infty$.

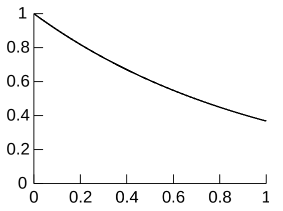


If Δt lies in the interval $(\tau/2, \tau)$ (right panel, dashed line, $\Delta t = 1.9$) the numerical solution shows spurious oscillations. That said, it is still stable because error decreases with time and the solution converges toward the analytical solution. However, if $\Delta t > 2\tau$ the solution becomes unstable: each advance of the solution now amplifies whatever error was already present, so the oscillations grow without limit (right panel, solid line, $\Delta t = 2.1$).

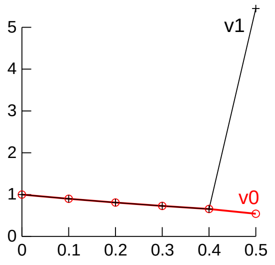
The instability of the explicit Euler method is a particular problem for multicompartmental models because spatial discretization introduces very fast time constants, especially when very small compartments are present. Consider a model cell that has a 10 µm diameter spherical soma to which a cylindrical spine with diameter = length = 1 µm is attached. Let all membrane be passive with resting membrane potential 0 mV, specific membrane conductance and capacitance 0.001 Siemens/cm^2 and 1 µF/cm^2, respectively, and cytoplasmic resistivity 160 Ω cm. The electrical equivalent circuit for this model is



where the soma's parameters are r0 = 3.183e2 megΩ, c0 = 3.142e-3 nF, the spine's parameters are r1 = 3.183e4 megΩ, c1 = 3.142e-5 nF, and the coupling resistance between them is ra = 1.0186 megΩ. This circuit has two time constants: the slower one is 1 ms, and the faster is ~3.2e-5 ms. If membrane potential is 1 mV in the soma and spine when t = 0, v0 and v1 should follow a monoexponential decay described by e^{-t}, i.e. membrane potential in the soma and the spine should both decay toward 0 with a time constant of 1, as shown here



One might think that this model could be simulated using the explicit Euler method with Δt = 0.1 ms. However, the simulation blows up after 0.4 ms, as shown below. This happens because finite precision arithmetic introduces roundoff errors, which are magnified at each step until they completely dominate the simulation.



Implicit Euler method

Like explicit Euler, the implicit Euler method also has first order accuracy. It differs from explicit Euler in that its recurrence equation is based on the derivative at the new time t_{i+1} , i.e.

$$\boldsymbol{y}_{i+1} = \boldsymbol{y}_i + \boldsymbol{F}(\boldsymbol{y}_{i+1}, t_{i+1})\,\Delta t \qquad \text{Eq. 9}$$

Gathering all terms at time t_{i+1} to the left hand side produces a set of algebraic equations

$$\boldsymbol{y}_{i+1} - \boldsymbol{F}(\boldsymbol{y}_{i+1}, t_{i+1})\,\Delta t = \boldsymbol{y}_i \qquad \text{Eq. 10}$$

which must be solved in order to compute the solution at the new time. Now we see why this is called the implicit Euler method: the new value of y is not computed directly from the previous value, but instead is is implicit in a set of equations.

It is clear that implicit Euler requires more computations to advance from one step to the next than explicit Euler does. That said, solving Eq. 10 is easier than it might seem, because the rows on its left hand side that correspond to the nodes of the discretized cable equations are equivalent to a tridiagonal matrix and can be solved very efficiently (Hines 1984), a consequence of the fact that neurons are singly connected branched structures.

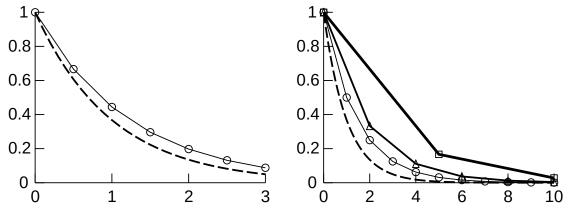
Implicit Euler's extra computational effort is compensated by its greatly improved stability. Applying it to Eq. 7, temporal discretization yields

$$y_{i+1} + \frac{y_{i+1}}{\tau}\Delta t = y_i \qquad \text{Eq. 11}$$

so the recurrence equation is

$$y_{i+1} = \frac{y_i}{1+\frac{\Delta t}{\tau}} \qquad \text{Eq. 12}$$

which is stable for any combination of positive τ and Δt , i.e. any plausible circumstance. These figures illustrate numerical solutions generated from Eq. 12 for $\tau = 1$ and initial condition $y = 1$.



The analytical solution is plotted as a dashed line; the numerical solution in the left panel used a time step Δt of 0.5, and those in the right panel used Δt of 1, 2, and 5, respectively (thin, medium, and thick lines). Note that all numerical solutions converge on the analytical solution, and that there are no spurious oscillations. Implicit Euler also handles stiff models very nicely; the two compartment model described earlier poses no problem for it.

One useful feature of implicit Euler is that, even when Δt is large compared to any time constant in the system, it still tends to produce solutions that are qualitatively correct. For a stable linear system, a single step approaches the steady state solution as $\Delta t \rightarrow \infty$ (chapter 4 in (Carnevale and Hines 2006)).

Crank-Nicholson method

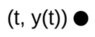
The Crank-Nicholson (C-N) method applies the central difference approximation to both time and space, so it is second order accurate in both (Crank and Nicholson 1947). C-N is conceptually equivalent to using implicit Euler to advance by $\Delta t/2$, then using explicit Euler to advance by another $\Delta t/2$, and involves little more effort than a single implicit Euler advance:

$$\boldsymbol{y}_{i+1} = 2\boldsymbol{y}^* - \boldsymbol{y}_i \qquad \text{Eq. 13}$$

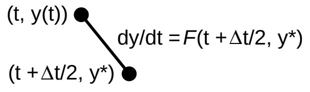
where $\boldsymbol{y}^*$ is the intermediate result produced by the implicit Euler half step advance

$$\boldsymbol{y}^* - \boldsymbol{F}(\boldsymbol{y}^*, t_i + \Delta t/2)\,\Delta t/2 = \boldsymbol{y}_i \qquad \text{Eq. 14}$$

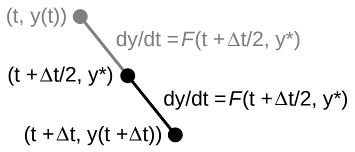
To understand why, consider this example that involves a single ODE. The implicit Euler half step is equivalent to starting at point $(t, y(t))$



and moving to an intermediate point $(t+\Delta t/2,\ y^*)$ where the slope $F(t+\Delta t/2,\ y^*)$ points back to $(t, y(t))$



The explicit Euler half step then uses this same slope $F(t+\Delta t/2,\ y^*)$ to move from the intermediate point $(t+\Delta t/2,\ y^*)$ to the new solution point $(t+\Delta t,\ y(t+\Delta t))$.



Since the slope is the same for both half steps,

$$y(t+\Delta t) = y(t) + 2(y^* - y(t)) = 2y^* - y(t) \qquad \text{Eq. 15}$$

This result readily generalizes to a system described by a family of ODEs.

The advantage of C-N over first order methods for dealing with time are that it offers greater accuracy for nearly the same computational effort when more accuracy is necessary, and it allows use of larger Δt without sacrificing accuracy when speed is important. Also, second order accuracy in time means that linear interpolation produces second order accurate values between adjacent solution times.

However, C-N is prone to spurious oscillations if the system equations are too stiff. This occurs if

$$\frac{\Delta t}{\tau_m} > \frac{2}{1+4\left(\frac{\lambda}{\Delta x}\right)^2} \approx \frac{1}{2}\left(\frac{\Delta x}{\lambda}\right)^2 \qquad \text{Eq. 16}$$

where τ_m is the membrane time constant, λ is the DC length constant, and Δx is compartment length; in other words, when the model's spatial discretization is too fine for the integration time step (Carnevale and Hines 2006). It can also happen if the model includes a voltage clamp, because voltage clamps introduce very fast time constants. Note that even if oscillations do occur, they die out with time, so C-N is stable.

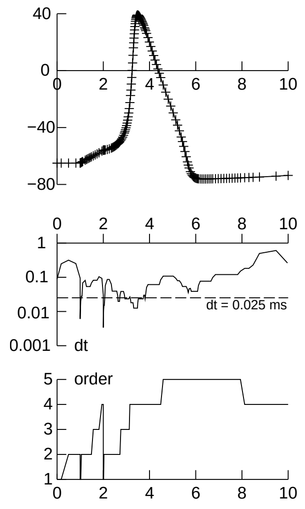
Higher Order Methods and Adaptive Integration

Methods that have a higher order of accuracy are tempting because of the possibility that the extra computational cost of using a high order method will be repaid by the ability to take fewer, longer steps between simulation points without compromising simulation results. However, this advantage comes at the cost of greater computational burden per time step. Also, high order methods tend to be unstable when applied to stiff equations unless Δt is kept small (Iserles 2008).

An alternative strategy for efficiently generating accurate simulations is to use an adaptive integration scheme, in which the computer automatically adjusts integration step size and/or order of integration so that local error lies below a user-specified criterion. Adaptive integration can often reduce total run time, but the computational overhead per advance is larger than with fixed Δt methods and some simulations will actually slow down.

The principal advantage of adaptive integration is improved quality of the numerical results, since local error is controlled and solution points are calculated at shorter time intervals when state variables are changing rapidly, and at longer intervals when not much is going on.

This is illustrated in the following figure, which shows an action potential generated by NEURON (Carnevale and Hines 2006) from a single compartment model with 100 µm^2 surface area and Hodgkin-Huxley membrane properties subjected to a 0.01 nA $\times$ 1 ms stimulus current that started at t = 1 ms. The simulation used the CVODES adaptive integrator of the SUNDIALS suite (Hindmarsh et al. 2005) with absolute error tolerance 0.001. The + marks on the voltage trace indicate the times at which solutions were generated; the other two graphs show how Δt and integration order were varied during the simulation in order to satisfy the error criterion. For most of the simulation, integration order was > 2 and Δt was substantially longer than 0.025 ms, NEURON's default for fixed time step integration (dashed line in middle graph). Note that Δt became very small and order dropped back to 1 at the start and end of the stimulus current; this happened because an abrupt parameter change constitutes a new initial value problem and forces restart of the integrator.



Practical Considerations

How to Select Δt and Δx

The general idea is to make Δt and Δx small enough that the simulation captures important features of model behavior in space and time with reasonable accuracy, but not so small as to needlessly prolong simulation time, increase storage requirements, or make results susceptible to cumulative roundoff error. Even if run time is not important, the tactic of cutting Δt and Δx in order to reduce the error from one time step to the next ("local error") is eventually limited by the accumulation of roundoff error, which increases global error.

That said, certain models and integration methods may limit one's options. As we have seen, the explicit Euler method becomes unstable if Δt is larger than the fastest time constant in the model, and avoidance of oscillations with the Crank-Nicholson method requires use of time steps small enough that temporal error does not outweigh spatial error (Eq. 16). Even with methods for which stability or oscillations are not an issue, it is important to remember that accuracy at short spatial scales may require the use of short time steps, and vice versa (e.g. simulations of the time course of Ca++ concentration in the near vicinity of open channels). Often an empirical approach is necessary, such as making Δt smaller until simulation results are qualitatively unchanged, then doing the same for Δx , and repeating until there is no further qualitative change in results.

Hines and Carnevale (2001) described various rules of thumb for spatial discretization of the cable equation, and introduced a new approach called the "d_lambda rule" which bases spatial discretization on the AC length constant λ_f at a frequency f so high that membrane current is almost exclusively capacitive and ion channel density doesn't matter. The rationale is that good temporal accuracy implies the accurate simulation of signal propagation at "high" frequencies. According to the d_lambda rule, a neurite is represented by N pieces of equal length where N is just large enough to satisfy

$$\frac{\text{neurite length}}{N} < \text{d\_lambda}\ \lambda_f \qquad \text{Eq. 17}$$

where the AC length constant for a cylinder with diameter d µm, cytoplasmic resistivity R_a Ω cm, and specific membrane capacitance C_m µF/cm^2 is

$$\lambda_f \approx \frac{1}{2}\sqrt{\frac{d}{\pi\ f\ R_a\ C_m}} \qquad \text{Eq. 18}$$

Membrane capacitive current far outweighs ionic currents at frequencies at least 5 times faster than the frequency that corresponds to the membrane time constant, i.e. $f > 5/(2\pi\tau_m)$. For most cells, it turns out that 100 Hz is more than fast enough, and d_lambda = 0.1 at 100 Hz is sufficient to produce good results.

The advantages of the d_lambda rule include the facts that it is effective, avoids excessive discretization, and is easily automated. The NEURON Simulation Environment offers the d_lambda rule as a feature of its CellBuilder tool, but in principle it could be easily implemented for any multicompartmental simulator.

Physiological Precision

Any discussion of the accuracy of numerical integration in the context of empirically-based modeling in computational neuroscience would be incomplete without asking the question: how precise must a simulation be in order to be useful? The answer is: it depends.

One concern often raised about empirically-based models is the limited precision of experimental data. In order to be useful for simulation, such data are often fit to functions that may have a sound theoretical basis, yet are only capable of approximating the actual relationships between dependent and independent variables. For parameters that result in a trajectory passing near a separatrix, simulation results may be highly sensitive to variation in any model parameter or simulation parameter (e.g. Δt , Δx , order of integration) (chapter 4 in (Carnevale and Hines 2006)); the same is true for the behavior of a system that lies near a bifurcation.

It is widely acknowledged that results that depend on very finely tuned parameters are biologically implausible because they seem unlikely to be robust. But what degree of accuracy is justified by natural biological variability, let alone the quality of the data from which parameters are estimated? Golowasch et al. (2002) pointed out the wide range of parameter values observed in normally functioning neurons and provided evidence for covariance between parameters. To minimize the effects of biological variability, and to avoid the confounding effects of parameter covariance, it would be best for all parameters to have been acquired in a single set of experiments on a single cell or network. However, this is not possible; modelers must rely on parameters gathered from experiments performed in different labs on different cell types from different species.

Seldom mentioned is another, deeper issue: the results of any model simulation are at least four times removed from the original physical system. The first separation occurs with the formulation of a conceptual model, which of necessity is a simplified representation of physical reality. Indeed, simplification affects all research, even that which does not involve computational modeling, because it is an unavoidable consequence of the way we develop an understanding of complex systems. Two more separations are imposed when the time and space derivatives are replaced by discretization formulas; this produces a new system that is only an approximation to the original physical system. The fourth separation happens when the discretized equations are solved numerically. Each of these separations is a potential source of error that could reduce the value of a simulation as a means for gaining insight into the original physical system. Discretization schemes and numerical integration methods are useful to the extent that the errors they introduce are manageable and allow at least a qualitative understanding of the original conceptual model.

To summarize, numerical integration applied to computational neuroscience models computes an approximate solution to one or more equations that are themselves approximations, and everything rests on partial knowledge inferred from imprecise measurements. In light of this, judgment is required when applying numerical methods so that results have what might be called "physiologically appropriate precision."

References

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

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

Golowasch, J., Goldman, M., Abbott, L. and Marder, E.. Failure of averaging in the construction of a conductance-based neuron model. *Journal of Neurophysiology* 87:1129-1131, 2002.

Hindmarsh, A. C., Brown, P. N., Grant, K. E., Lee, S. L., Serban, R., Shumaker, D. E. and Woodward, C. S.. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)* 31:363-396, 2005.

Hines, M.. Efficient computation of branched nerve equations. *International Journal of Bio-Medical Computation* 15:69-76, 1984.

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

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

Iserles, A.. *A First Course in the Numerical Analysis of Differential Equations*. : Cambridge University Press, 2008.

Jack, J., Noble, D. and Tsien, R.. *Electric Current Flow in Excitable Cells*. London: Oxford University Press, 1983.

Polyanin, A. D., Schiesser, W. E. and Zhurov, A. I.. Partial differential equation. *Scholarpedia* 3:4605, 2008.

Rall, W.. Core conductor theory and cable properties of neurons. In: *Handbook of Physiology, vol. 1, part 1: The Nervous System*, edited by Kandel, E. R.. Bethesda, MD: American Physiological Society, 1977, 39-98.

Wolfram Research, I.. Numerical Solution of Partial Differential Equations, , 2013.

Zador, A. and Koch, C.. Linearized models of calcium dynamics - formal equivalence to the cable equation. *Journal of Neuroscience* 14:4705-4715, 1994.

The Theoretical Foundation of Dendritic Function: Collected Papers of Wilfrid Rall with Commentaries. Segev, I., Rinzel, J. and Shepherd, G. (Ed.). Cambridge, MA: MIT Press, 1995.