Teaching and Learning Cellular Biophysics: Propagated Action Potential Simulation in Matlab

by

Tommy Ng

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2000

Author .....
Department of Electrical Engineering and Computer Science
May 22, 2000

Certified by .....
Thomas Fischer Weiss
Thomas and Gerd Perkins Professor In Electrical Engineering and Bioengineering
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Teaching and Learning Cellular Biophysics: Propagated Action Potential Simulation in Matlab

by

Tommy Ng

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 2000, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The simulation software was developed to provide an efficient tool for students to study propagated action potentials of unmyelinated nerve fibers. The software allows the user to specify all the parameters of the axon, membrane, bath, stimulating electrodes and recording electrodes. Visual and plotting tools allow the user to view all the membrane variables (such as membrane potential, membrane-current components, etc) either dynamically or statically in three-dimensional plots and in two-dimensional parametric plots. The software also provides the user a choice of numerical methods to solve the Hodgkin-Huxley equations that model the physiological process.

An evaluation of the numerical methods show that there is no single best method for computing a solution. The user may vary the discretization steps, dt and dz, and choose a numerical method that would minimize or maximize the desired criteria (i.e. speed, memory requirement). Tests of the software yielded desired solutions that match the results obtained by previous studies. For example, the software produces an propagated action potential with a conduction velocity of 18.75 m/s using the default parameters of the HH model. This conduction velocity is within 12% of the value calculated by Cooley and Dodge in 1966.

Thesis Supervisor: Thomas Fischer Weiss
Title: Thomas and Gerd Perkins Professor In Electrical Engineering and Bioengineering

Acknowledgments

Without the great attention and invaluable help offered by my thesis advisor, Prof. T. F. Weiss, the development of this software would not have been possible. Prof. Weiss put a great effort to work out the details of the numerical methods that were implemented in the software. Furthermore, he contributed a great amount of time in testing the software. His kindness and attentive attitude have made this a very enjoyable and productive thesis project. Next, I would like to acknowledge all the people who have previously worked in the *Softcell* package. Without many of the previously written codes, it would have been a lot harder and more time consuming to write the *PAP* software. Special thanks goes to T. S. Bhatnagar, who wrote many of the codes adapted in this software.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

An electrically excitable cell produces a difference of potential across its cell membrane, called an action potential. The difference in membrane potential is accompanied by a flow of current longitudinally through the intracellular and extracellular media, as well as current flow across the cell membrane along its entire length. The spatial coupling of current flow through neighboring segments of the membrane results in the propagation of the action potential along the cell surface.

In the 1950s, Hodgkin and Huxley derived a partial differential equation that effectively describes the electrical property of a specific large cell, the giant squid axon (Hodgkin and Huxley, 1952). This partial differential equation consists of the core conductor equations plus a characterization of the electrical properties of the membrane. The cell is assumed to be cylindrical and the inner and outer conductors are assumed to be equipotentials at each longitudinal position along the cell. At the time, the lack of tools to efficiently compute the partial differential equation forced Hodgkin and Huxley to solve the equations by hand calculation. With the introduction of the digital computer in the 1950s, solutions to these equations were computed more rapidly (Fitzhugh and Antosiewicz, 1959). By the 1960's computer speeds had increased to allow the first comprehensive study of the solutions to the partial differential equations (Cooley and Dodge, 1966). Further increases in computer speeds and the development of improved numerical methods have allowed solutions to be obtained on personal computers (Moore et al., 1975; Joyner et al., 1978; Hines, 1984). We have implemented some of these numerical methods along with a full-featured MATLAB

software which allows the user to verify the efficiency and accuracy of each of the numerical methods as well as to understand the Hodgkin-Huxley model of propagated action potential under current-source excitation conditions.

1.2 Overview of the software

This software is a stand-alone graphical application that will be incorporated in a larger package of physiological simulation software developed at MIT, known as the *Softcell* package. The proposed software is an adaptation of the Hodgkin-Huxley single-compartment (non-propagated) action potential software titled simply as *Hodgkin-Huxley Model* in the *Softcell* package. More information about the development history of the *Softcell* package is found in *Cellular Biophysics: Teaching and Learning with Computer Simulations*, authored by Thomas F. Weiss and Tanmaya S. Bhatnagar.

The software is comprised of a number of setup and analysis figures that are launched from a main control figure. Using the setup figures, the user can specify the physical characteristics of the axon (such as length, radius, etc.) and parameters of the membrane and bath (such as membrane capacitance, conductances of membrane ionic channels, kinetic parameters, concentration of all relevant ions, temperature, etc.). A stimulus source can be customized to the user's specifications, and graphic icons of a pair of stimulus electrodes can be visually placed along a schematic representation of an axon. In addition, graphic icons representing a pair of recording electrodes can be placed along the axon schematic to compute and display the membrane potential during a simulation run. When a simulation is started, solutions are calculated using one of four user selectable numerical methods, and the time-evolution of the membrane voltage (action potential) is shown color coded to convey the space-time evolution of potentials on the axon schematic. In other words, the color coding helps the user visualize how an action potential arises and conducts along the axon schematic. After a simulation is completed, the user can study the computed solution variables (such as membrane potential, membrane-current components, membrane-conductance components, and channel activation and inactivation variables) using different interactive analysis figures (such as a figure that displays the space-time evolution of any solution variable, a figure that generates three-dimensional plots to show the dependency on time and space of any solution variable, and a figure that generates comparison plots

between the solution variables in a fixed-time or fixed-space setting).

In brief, with the software, the user can experiment and see the consequences of changing any of the parameters of the Hodgkin-Huxley model by means of color visualization, dynamic 2D-graphs, and 3D-graphs of the computed solution variables.

Chapter 2

Theory

The theory of the propagation of an action potential along an unmyelinated axon was first described for the squid giant axon (Hodgkin and Huxley, 1952). We briefly review this theory here; a more detailed description is available elsewhere (Weiss, 1996).

2.1 Core conductor equations

An incremental portion of the core conductor model of a cylindrical unmyelinated nerve fiber is shown in Figure 2-1. Kirchhoff's laws relate the variables of this incremental model. The core conductor equations result (Weiss, 1996) if the limit of these equations is taken as the incremental length of the fiber goes to zero, $\Delta z \to 0$,

$$\frac{\partial I_i(z,t)}{\partial z} = K_{ei}(z,t) - K_m(z,t), \tag{2.1}$$

$$\frac{\partial I_o(z,t)}{\partial z} = K_m(z,t) - K_{eo}(z,t), \tag{2.2}$$

$$\frac{\partial V_i(z,t)}{\partial z} = -r_i I_i(z,t), \tag{2.3}$$

$$\frac{\partial V_o(z,t)}{\partial z} = -r_o I_o(z,t), \tag{2.4}$$

$$V_m(z,t) = V_i(z,t) - V_o(z,t), \tag{2.5}$$

$$\frac{\partial V_m(z,t)}{\partial z} = -r_i I_i(z,t) + r_o I_o(z,t). \tag{2.6}$$

These equations can be combined to obtain a relation between the membrane potential V_m and the membrane current density J_m at time t and location z along a cylindrical

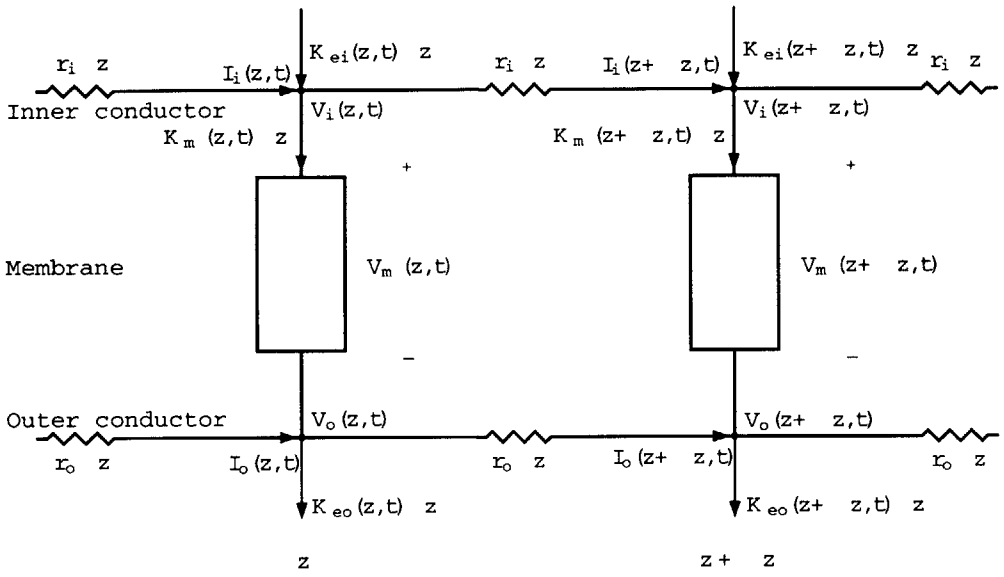


Figure 2-1: Equivalent network model of incremental sections of an unmyelinated nerve fiber showing the reference directions for all the voltage and current variables. The Ks are currents per unit length of the fiber: K_m is the membrane current; K_{eo} and K_{ei} are extrinsic currents applied (by electrodes) externally and internally, respectively. I_i and I_o are the internal and external longitudinal currents. V_m is the potential across the membrane and V_i and V_o are the internal and external potentials, respectively. The internal and external resistance per unit length of the fiber are r_i and r_o.

unmyelinated nerve fiber, often called the *core-conductor equation*

$$\frac{1}{2\pi a(r_o + r_i)} \frac{\partial^2 V_m(z,t)}{\partial z^2} = J_m(z,t) - \frac{1}{2\pi a(r_o + r_i)}(r_o K_{eo}(z,t) + r_i K_{ei}(z,t)),$$

where a is the radius of the cylindrical cell and $K_m = 2\pi a J_m$. In addition, the resistance per unit length of cytoplasm can be expressed in terms of the resistivity as follows

$$r_i = \frac{\rho_i}{4\pi a^2}$$

where ρ_i is the resistivity of cytoplasm.

2.2 The membrane current density

The total membrane current density, $J_m(z,t)$, is the sum of the capacitance current density and the ionic current density

$$J_m(z,t) = J_C(z,t) + J_{ion}(z,t), \tag{2.7}$$

where the capacitance current density is

$$J_C(z,t) = C_m \frac{\partial V_m(z,t)}{\partial t} \tag{2.8}$$

and C_m is the specific membrane capacitance, i.e., the capacitance of a unit area of membrane. The ionic current density can be expressed in terms of its components

$$J_{ion}(z,t) = J_{Na}(z,t) + J_K(z,t) + J_L(z,t), \tag{2.9}$$

These ionic components due to sodium, potassium and leakage are

$$J_{Na}(z,t) = G_{Na}(V_m,t)(V_m - V_{Na}), \tag{2.10}$$

$$J_K(z,t) = G_K(V_m,t)(V_m - V_K), \tag{2.11}$$

$$J_L(z,t) = G_L(V_m,t)(V_m - V_L), \tag{2.12}$$

where V_n is the Nernst equilibrium potential of ion n defined in terms of the concentrations as follows

$$V_n = \frac{RT}{z_n F} \ln\left(\frac{c_n^o}{c_n^i}\right), \tag{2.13}$$

where R is the molar gas constant, T is absolute temperature, F is Faraday's constant, and z_n, c_n^o and c_n^i are the valence, outside, and inside concentrations of ion n, respectively. For a univalent ion, the Nernst equilibrium potential is expressed as

$$V_n = 0.08616\,(T_c + 273.16)\,\ln\left(\frac{c_n^o}{c_n^i}\right)\ (\text{mV}), \tag{2.14}$$

where T_c is the temperature in Centigrade.

For purposes of numerical solution of the core conductor equation with the relation

of the membrane potential to the ionic current density expressed by the Hodgkin-Huxley model, it is helpful to combine the ionic current density with the external currents so that

$$J(z,t) = J_{ion}(z,t) - \frac{1}{2\pi a(r_o + r_i)}(r_o K_{eo}(z,t) + r_i K_{ei}(z,t)), \tag{2.15}$$

so the core conductor equation can be expressed as

$$\frac{1}{2\pi a(r_o + r_i)} \frac{\partial^2 V_m(z,t)}{\partial z^2} = C_m \frac{\partial V_m(z,t)}{\partial t} + J(z,t). \tag{2.16}$$

2.3 The membrane conductances

The sodium and potassium conductances are defined as

$$G_{Na}(V_m,t) = \bar{G}_{Na} m^3(V_m,t) h(V_m,t), \tag{2.17}$$

$$G_K(V_m,t) = \bar{G}_K n^4(V_m,t). \tag{2.18}$$

The first-order kinetic equations for the activation and inactivation factors are written in terms of the rate constants as follows:

$$\frac{dm}{dt} = \alpha_m - m(\alpha_m + \beta_m), \tag{2.19}$$

$$\frac{dh}{dt} = \alpha_h - h(\alpha_h + \beta_h), \tag{2.20}$$

$$\frac{dn}{dt} = \alpha_n - n(\alpha_n + \beta_n), \tag{2.21}$$

where the α's and β's depend upon V_m.

The dependence of the rate constants on membrane potential has been generalized from the original Hodgkin-Huxley model to allow control of the potential dependence of individual rate constants and to represent approximately the effects of changes in calcium concentration and temperature.

The rate constants are

$$\alpha_m = \frac{-0.1(35 + V_m + \Delta V_{Ca} + V_{\alpha m})}{e^{-0.1(35+V_m+\Delta V_{Ca}+V_{\alpha m})} - 1} K_T K_m, \tag{2.22}$$

$$\beta_m = 4e^{-(V_m+\Delta V_{Ca}+V_{\beta m}+60)/18} K_T K_m, \tag{2.23}$$

$$\alpha_h = 0.07e^{-0.05(V_m+\Delta V_{Ca}+V_{\alpha h}+60)} K_T K_h, \tag{2.24}$$

$$\beta_h = \frac{1}{1+e^{-0.1(V_m+\Delta V_{Ca}+V_{\beta h}+30)}} K_T K_h, \tag{2.25}$$

$$\alpha_n = \frac{-0.01(V_m+\Delta V_{Ca}+V_{\alpha n}+50)}{e^{-0.1(V_m+\Delta V_{Ca}+V_{\alpha n}+50)}-1} K_T K_n, \tag{2.26}$$

$$\beta_n = 0.125e^{-0.0125(V_m+\Delta V_{Ca}+V_{\beta n}+60)} K_T K_n, \tag{2.27}$$

where K_T is a temperature factor, that is defined as

$$K_T = 3^{(T_c-6.3)/10}. \tag{2.28}$$

K_T multiplies all the rate constants; this effect approximates the effect of temperature on the electrical properties of the membrane of the squid giant axon (Huxley, 1959). The factors K_m, K_h and K_n have been added to the original Hodgkin-Huxley model to allow changes to be made in the individual rate constants of m, h, and n. The factor ΔV_{Ca} is used to approximate the dependence of rate constants on calcium concentration and is

$$\Delta V_{Ca} = 0.03335\,(T_c + 273.16)\left(\ln\left(\frac{c^o_{Ca}}{c^i_{Ca}}\right) - 12.995\right)\ (\text{mV}). \tag{2.29}$$

At the normal calcium concentration, the potential ΔV_{Ca} has the value 0. For calcium concentration ratios that differ from the normal value, ΔV_{Ca} differs from 0 and the dependence of the rate constants on the membrane potential is shifted by a term that is proportional to the calcium equilibrium potential. This approximates the effect of a change in calcium concentration on the parameters of squid giant axon membrane (Frankenhaeuser and Hodgkin, 1957). The potentials $V_{\alpha m}$, $V_{\beta m}$, $V_{\alpha h}$, $V_{\beta h}$, $V_{\alpha n}$, and $V_{\beta n}$ have the value 0 in the original Hodgkin-Huxley model. These potentials can be used to shift the dependence of individual rate constants on membrane potential.

2.4 Representation of stimulating electrodes

An arrangement of stimulating electrodes is shown in Figure 2-2. From Kirchhoff's current law we have that at any position along the fiber

$$I_i(z,t) + I_o(z,t) + I_s(z,t) = 0. \tag{2.30}$$

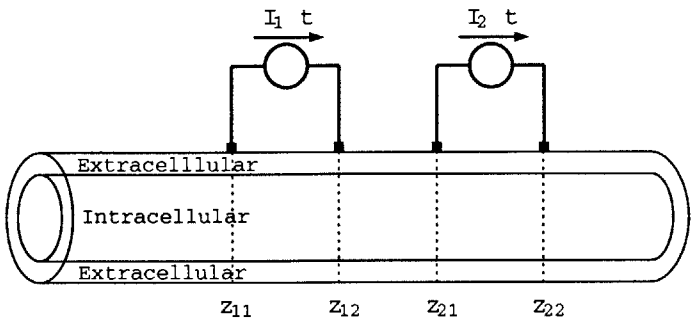


Figure 2-2: The relation of two pairs of stimulus electrodes to a fiber is shown. The stimulus electrodes are shown as extracellular but any of the electrodes can be intracellular or extracellular.

where $I_s(z,t)$ is the stimulus current in the positive z-direction which can be written as

$$I_s(z,t) = I_1(t)(u(z-z_{11}) - u(z-z_{12})) + I_2(t)(u(z-z_{21}) - u(z-z_{22})),$$

where $u(z)$ is the unit step function defined as

$$u(z) = \begin{cases} 1 & \text{if } z > 0, \\ 0 & \text{if } z < 0. \end{cases}$$

The electrode configuration shown in Figure 2-2 also imparts current components to Equations 2.1 and 2.2. With all the electrodes external, the currents per unit length are as follows

$$\begin{aligned} K_{eo}(z,t) &= I_1(t)(\delta(z-z_{11}) - \delta(z-z_{12})) + I_2(t)(\delta(z-z_{21}) - \delta(z-z_{22})), \\ K_{ei}(z,t) &= 0, \end{aligned}$$

where $\delta(z)$ is the unit impulse function.

If all four of the electrodes were intracellular then the currents per unit length would be

$$\begin{aligned} K_{eo}(z,t) &= 0, \\ K_{ei}(z,t) &= -I_1(t)(\delta(z-z_{11}) - \delta(z-z_{12})) - I_2(t)(\delta(z-z_{21}) - \delta(z-z_{22})), \end{aligned}$$

With careful attention to signs, any combination of intracellular and extracellular stimulating electrodes can be accommodated. The algorithm is that extracellular electrodes supply positive impulses at the cathode and negative impulses at the anode; intracellular electrodes supply negative impulses at the cathode and positive impulses at the anode.

2.5 Longitudinal currents

By using the core conductor equations (Equation 2.6) and the KCL equation for the current (Equation 2.30), we can solve for the longitudinal currents as follows

$$\begin{aligned} I_i(z,t) &= -\frac{1}{r_i+r_o}\frac{\partial V_m(z,t)}{\partial z} - \frac{r_o}{r_i+r_o}I_s(z,t), \\ I_o(z,t) &= \frac{1}{r_i+r_o}\frac{\partial V_m(z,t)}{\partial z} - \frac{r_i}{r_i+r_o}I_s(z,t). \end{aligned}$$

2.6 Intracellular and extracellular potential differences

The expressions for the longitudinal currents can be combined with Equations 2.3 and 2.4 and the result integrated on z to yield the potential differences of a recording electrode at z_{r2} minus that at z_{r1} as follows

$$\begin{aligned} V_i(z_{r2},t) - V_i(z_{r1},t) &= \frac{r_i}{r_i+r_o}\left(V_m(z_{r2},t) - V_m(z_{r1},t)\right) \\ &\quad + \frac{r_i r_o}{r_i+r_o}\int_{z_{r1}}^{z_{r2}} I_s(z,t)\,dz, \\ V_o(z_{r2},t) - V_o(z_{r1},t) &= -\frac{r_o}{r_i+r_o}\left(V_m(z_{r2},t) - V_m(z_{r1},t)\right) \\ &\quad + \frac{r_i r_o}{r_i+r_o}\int_{z_{r1}}^{z_{r2}} I_s(z,t)\,dz, \end{aligned}$$

where

$$\begin{aligned} \int_{z_{r1}}^{z_{r2}} I_s(z,t)\,dz &= I_1(t)((z-z_{11})u(z-z_{11}) - (z-z_{12})u(z-z_{12})) \\ &\quad + I_2(t)((z-z_{21})u(z-z_{21}) - (z-z_{22})u(z-z_{22})). \end{aligned}$$

2.7 Default values of parameters

The default numerical values for the Hodgkin-Huxley model are: $\overline{G}_{Na} = 120$, $\overline{G}_K = 36$, and $G_L = 0.3$ mS/cm^2; $C_m = 1$ μF/cm^2; $c^o_{Na} = 491$, $c^i_{Na} = 50$, $c^o_K = 20.11$, $c^i_K = 400$, $c^o_{Ca} = 44$, $c^i_{Ca} = 0.00011$ mmol/L; $V_L = -49$ mV; external resistance per unit length, $r_o = 0$ Ohm/cm; resistivity of axoplasm, $\rho_i = 35.4$ Ohm·cm; axon radius, $a = 238$ μm; temperature is 6.3°C.

Chapter 3

Numerical Method

Several numerical integration methods have been used to solve the core conductor equations in order to determine which is most efficient, in terms of computational time, accuracy and stability (Moore et al., 1975; Joyner et al., 1978; Hines, 1984). The shape of the computed action potential has generally been found to be independent of the method used, but the latency may vary for the different methods (Moore and Ramon, 1974). The latency is the time it takes for an action potential to trigger after a stimulus onset. We have implemented four numerical methods for solving partial differential equations using MATLAB and produce the results using different analysis plots to compare computational efficiencies, stability, and accuracy. The numerical methods that were implemented are: forward Euler, backward Euler, Crank-Nicolson, and a variant of the Crank-Nicolson (Staggered Crank-Nicolson).

We can obtain a numerical solution to the core conductor equation that describes the membrane potential using all four numerical methods provided the temporal and spatial discretization meet certain criteria. These numerical methods employ the finite-difference method which takes the partial differential equation of the core conductor model and discretize it directly by discretizing the membrane potential and membrane current and approximating the spatial and temporal derivatives with finite differences. This reduces the partial differential equation to an algebraic equation (Smith, 1985; Press et al., 1986; Gerald and Wheatley, 1989). Another way to obtain a numerical solution is known as the finite-element method. In the finite-element method, the cylindrical axon is segmented into small cylinders over each of which the membrane potential is assumed to be constant. Ordinary

differential equations (with time as the independent variable) can be written to describe the membrane potential and membrane current for each segment and the set of ordinary differential equations can be solved numerically with appropriate boundary conditions linking the electrical variables in adjoining segments. In general, the finite-element method is less compact and requires the computation of a non-constant number of ordinary differential equations which increases proportionally with the number of discretized segments. Therefore, it is believed to be more efficient to use the finite-difference method which consists of solving one partial differential equation which describes the overall electrical behavior of the entire large cell.

We describe the derivation of the spatial and temporal difference equations from the core conductor equation, and then discuss the numerical methods for solving the equations. The following is adapted from the notes written by T. F. Weiss.

The first step is to discretize the membrane potential and current density variables as follows,

$$\begin{aligned} V_i^n &= V_m(i\Delta z, n\Delta t) = V_m(z,t)|_{z=i\Delta z, t=n\Delta t}, \\ J_i^n &= J(i\Delta z, n\Delta t) = J(z,t)|_{z=i\Delta z, t=n\Delta t}, \end{aligned}$$

where Δz and Δt are the increments in z and t.

The second spatial derivative is approximated by the centered difference (also known as the method of lines),

$$\frac{\partial^2 V_m(z,t)}{\partial z^2} \approx \frac{V_{i+1}^n - 2V_i^n + V_{i-1}^n}{(\Delta z)^2}.$$

All of the numerical methods to be implemented use the above spatial finite-difference approximation. They differ only in the temporal approximation. There are two simple ways to approximate the temporal derivative at time t. In the forward Euler approximation,

$$\frac{\partial V_m(z,t)}{\partial t} \approx \frac{V_i^{n+1} - V_i^n}{\Delta t},$$

whereas in the backward Euler approximation

$$\frac{\partial V_m(z,t)}{\partial t} \approx \frac{V_i^n - V_i^{n-1}}{\Delta t}.$$

The temporal approximation used in the Crank-Nicolson method essentially averages the forward Euler and the backward Euler algorithms. The staggered increment with the Crank-Nicolson method uses a variant (half a step) of the forward Euler approximation.

3.1 Forward Euler method

With the forward Euler approximation, the discretized core conductor equation (Equation 2.16) is

$$\frac{1}{2\pi a(r_o + r_i)}\left(\frac{V_{i+1}^n - 2V_i^n + V_{i-1}^n}{(\Delta z)^2}\right) = C_m\left(\frac{V_i^{n+1} - V_i^n}{\Delta t}\right) + J_i^n. \tag{3.1}$$

By collecting terms, the forward Euler approximation leads to the following difference equations

$$V_i^{n+1} = \eta V_{i+1}^n + (1 - 2\eta)V_i^n + \eta V_{i-1}^n - \xi J_i^n,$$

where

$$\eta = \frac{\Delta t}{2\pi a(r_o + r_i)(\Delta z)^2 C_m} \text{ and } \xi = \frac{\Delta t}{C_m}.$$

Note that the forward Euler approximation results in an *explicit* method of solution since it gives an explicit formula for computing the V_i^{n+1} given both V_i^n and J_i^n, i.e., the values of variables at time $n+1$ are computed directly from their values at time n.

In order to solve the problem numerically, we require initial conditions along the length of the fiber plus boundary conditions at the two ends of the fiber. Initial conditions are simply that the membrane potential is at its resting value and that the membrane current is zero. A variety of boundary conditions are plausible. We shall use the boundary condition that the fiber is closed at both ends so that the longitudinal current is zero at the two ends of the fiber. Equation 3.1 essentially expresses Kirchhoff's current law which can be modified at the closed end to set the appropriate longitudinal current to zero. Using this method for $i = 1$ and setting the longitudinal current from node 0 to node 1 to zero yields

$$\frac{1}{2\pi a(r_o + r_i)}\left(\frac{V_2^n - V_1^n}{(\Delta z)^2}\right) = C_m\left(\frac{V_1^{n+1} - V_1^n}{\Delta t}\right) + J_1^n,$$

which, after collecting terms, results in the difference equation

$$V_1^{n+1} = \eta V_2^n + (1-\eta)V_1^n - \xi J_1^n.$$

A similar equation results at the other end of the fiber model. The above set of equations can be expressed in the following matrix form:

$$\begin{bmatrix} V_1^{n+1} \\ V_2^{n+1} \\ V_3^{n+1} \\ \vdots \\ V_{I-1}^{n+1} \\ V_I^{n+1} \end{bmatrix} = \begin{bmatrix} 1-\eta & \eta & 0 & \cdots & 0 & 0 & 0 \\ \eta & 1-2\eta & \eta & \cdots & 0 & 0 & 0 \\ 0 & \eta & 1-2\eta & \cdots & 0 & 0 & 0 \\ 0 & 0 & \eta & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1-2\eta & \eta & 0 \\ 0 & 0 & 0 & \cdots & \eta & 1-2\eta & \eta \\ 0 & 0 & 0 & \cdots & 0 & \eta & 1-\eta \end{bmatrix} \begin{bmatrix} V_1^n \\ V_2^n \\ V_3^n \\ \vdots \\ V_{I-1}^n \\ V_I^n \end{bmatrix} - \xi \begin{bmatrix} J_1^n \\ J_2^n \\ J_3^n \\ \vdots \\ J_{I-1}^n \\ J_I^n \end{bmatrix}.$$

The matrix equation can be written as

$$\mathbf{V}^{n+1} = \mathbf{A}_{FE}\mathbf{V}^n - \xi \mathbf{J}^n.$$

To obtain a solution, we need to compute J_i^n. This current is linked to the ionic currents (Equation 2.15) which themselves depend upon factors that are solutions to differential equations (Equations 2.19-2.21). Let x stand for m, n, and h then we have

$$\frac{dx}{dt} = \alpha_m - x(\alpha_m + \beta_m),$$

where α and β are instantaneous functions of membrane potential. We discretize this equation with the forward Euler algorithm as follows

$$\frac{x_i^n - x_i^{n-1}}{\Delta t} = \alpha_x\left(V_i^{n-1}\right) - x_i^{n-1}\left(\alpha_x\left(V_i^{n-1}\right) + \beta_x\left(V_i^{n-1}\right)\right),$$

and collect terms to obtain

$$x_i^n = x_i^{n-1}\left(1 - \Delta t\left(\alpha_x\left(V_i^{n-1}\right) + \beta_x\left(V_i^{n-1}\right)\right)\right) + \Delta t \alpha_x\left(V_i^{n-1}\right). \tag{3.2}$$

With the factors m, n, and h substituted for x, the ionic current can be computed as follows

$$(J_{ion})_i^n = \bar{G}_K(n_i^n)^4(V_i^n - V_K) + \bar{G}_{Na}(m_i^n)^3 h_i^n(V_i^n - V_{Na}) + G_L(V_i^n - V_L),$$

where each of the factors m_i^n, n_i^n, and h_i^n are computed according to Equation 3.2. The total current is

$$J_i^n = (J_{ion})_i^n - \frac{r_o(K_{eo})_i^n + r_i(K_{ei})_i^n}{2\pi a(r_o + r_i)}.$$

Although the Forward Euler method is relatively easy to understand, it's numerical properties are undesirable. The forward Euler method is stable provided $\eta \leq 0.5$ which implies that to achieve stable solutions, the discretization in space and in time cannot be chosen independently; specifically, a stable solution requires $\Delta t \leq \pi a(r_o + r_i)(\Delta z)^2 C_m$ (Mascagni and Sherman, 1998). Thus, a choice of Δz limits the choice of Δt required to achieve stable solutions. If Δt is too large, the solution will diverge. Thus, for example, the forward Euler algorithm cannot be used to efficiently compute the response for large values of Δt which would be desirable to determine steady-state responses. Furthermore, the error resulting from temporal and spatial discretization behaves as $O(\Delta t) + O((\Delta z)^2)$. Thus, although the method of solution using the forward Euler algorithm is simple to understand, the numerical properties of the method are not desirable.

3.2 Backward Euler method

With the backward Euler approximation (advanced one point in time), the discretized core conductor equation is

$$\frac{1}{2\pi a(r_o + r_i)}\left(\frac{V_{i+1}^{n+1} - 2V_i^{n+1} + V_{i-1}^{n+1}}{(\Delta z)^2}\right) = C_m\left(\frac{V_i^{n+1} - V_i^n}{\Delta t}\right) + J_i^{n+1}, \qquad (3.3)$$

which, after collecting terms and changing signs, yields the difference equation

$$-\eta V_{i+1}^{n+1} + (1 + 2\eta)V_i^{n+1} - \eta V_{i-1}^{n+1} = V_i^n - \xi J_i^{n+1}.$$

Note that the backward Euler method is an *implicit* method of solution since it gives an implicit formula for computing the V_i^{n+1} given both V_i^n and J_i^n. Thus, the solution can only be obtained by solving a set of coupled algebraic equations.

We use the same initial and boundary conditions as for the forward Euler algorithm. By modifying Equation 3.3 with a closed end condition at $i = 1$ and collecting terms, the difference equation is

$$-\eta V_2^{n+1} + (1+\eta)V_1^{n+1} = V_1^n - \xi J_1^n.$$

A similar equation results at the other end of the fiber model so that the incorporation of the closed-end boundary conditions results in the following matrix equation

$$\begin{bmatrix} 1+\eta & -\eta & 0 & \cdots & 0 & 0 & 0 \\ -\eta & 1+2\eta & -\eta & \cdots & 0 & 0 & 0 \\ 0 & -\eta & 1+2\eta & \cdots & 0 & 0 & 0 \\ 0 & 0 & -\eta & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1+2\eta & -\eta & 0 \\ 0 & 0 & 0 & \cdots & -\eta & 1+2\eta & -\eta \\ 0 & 0 & 0 & \cdots & 0 & -\eta & 1+\eta \end{bmatrix} \begin{bmatrix} V_1^{n+1} \\ V_2^{n+1} \\ V_3^{n+1} \\ \vdots \\ V_{I-1}^{n+1} \\ V_I^{n+1} \end{bmatrix} = \begin{bmatrix} V_1^n \\ V_2^n \\ V_3^n \\ \vdots \\ V_{I-1}^n \\ V_I^n \end{bmatrix} - \xi \begin{bmatrix} J_1^{n+1} \\ J_2^{n+1} \\ J_3^{n+1} \\ \vdots \\ J_{I-1}^{n+1} \\ J_I^{n+1} \end{bmatrix}.$$

The matrix equation can be written as

$$\mathbf{A_{BE}} \cdot \mathbf{V}^{n+1} = \mathbf{V}^n - \xi \mathbf{J}^{n+1},$$

which has the solution

$$\mathbf{V}^{n+1} = \mathbf{A_{BE}}^{-1} \cdot \left(\mathbf{V}^n - \xi \mathbf{J}^{n+1} \right). \tag{3.4}$$

Similar to the procedure with the forward Euler algorithm, we need to compute J_i^{n+1} from the factors that determine the ionic conductances. Once again, we let x stand for m, n, and h, and we discretize this equation, this time with the backward Euler algorithm as follows

$$\frac{x_i^{n+1} - x_i^n}{\Delta t} = \alpha_x \left(V_i^{n+1} \right) - x_i^{n+1} \left(\alpha_x \left(V_i^{n+1} \right) + \beta_x \left(V_i^{n+1} \right) \right),$$

and collect terms to obtain

$$x_i^{n+1} = \frac{x_i^n + \Delta t \alpha_x \left(V_i^{n+1} \right)}{1 + \Delta t \left(\alpha_x \left(V_i^{n+1} \right) + \beta_x \left(V_i^{n+1} \right) \right)}. \tag{3.5}$$

With the factors m, n, and h substituted for x, the ionic current can be computed as follows

$$(J_{ion})_i^{n+1} = \bar{G}_K (n_i^{n+1})^4 (V_i^n - V_K) + \bar{G}_{Na} (m_i^{n+1})^3 h_i^{n+1} (V_i^n - V_{Na}) + G_L (V_i^n - V_L),$$

where each of the factors m_i^{n+1}, n_i^{n+1}, and h_i^{n+1} are computed according to Equation 3.5. The total current is

$$J_i^{n+1} = (J_{ion})_i^{n+1} - \frac{r_o(K_{eo})_i^{n+1} + r_i(K_{ei})_i^{n+1}}{2\pi a(r_o + r_i)}.$$

One difference is that the factor x_i^{n+1} depends upon the potential V_i^{n+1} so that the four equations — Equation 3.4 and three equations (one for each of the factors m, n, and h) of the form of Equation 3.5 — need to be solved simultaneously. To obtain accurate solutions, the set of 4 equations are iterated until the 4 variables V^{n+1}, m_i^{n+1}, n_i^{n+1}, and h_i^{n+1} converge to within some specified limit. For example, Equation 3.4 can be solved for the value of V^{n+1} using the values of the factors at time increment n to compute new values of the factors. These are then used to compute the new value of V^{n+1} which is used to compute new values of the factors, etc.

The backward Euler algorithm is stable for all values of Δt and Δz. Thus Δt and Δz can be chosen independently. Large values of Δt can be used to determine approximate steady-state values of the solution. Just as for the forward Euler algorithm, the error resulting from temporal and spatial discretization using the backward Euler algorithm behaves as $O(\Delta t) + O((\Delta z)^2)$.

3.3 Crank-Nicolson method

In the Crank-Nicolson method, the forward Euler and backward Euler approximations are averaged to yield the following difference equation

$$\frac{1}{4\pi a(r_o + r_i)} \left(\frac{V_{i+1}^{n+1} - 2V_i^{n+1} + V_{i-1}^{n+1}}{(\Delta z)^2} + \frac{V_{i+1}^{n} - 2V_i^{n} + V_{i-1}^{n}}{(\Delta z)^2} \right) = C_m \left(\frac{V_i^{n+1} - V_i^n}{\Delta t} \right) + \frac{1}{2}(J_i^n + J_i^{n+1}). \tag{3.6}$$

Alternatively, this equation can be regarded as satisfying the partial differential equation at the discrete time $n + \frac{1}{2}$ with central differences used for all derivatives. This interpretation is exploited in the next section on the *Staggered increment with the Crank-Nicolson method.*

Equation 3.6 can be rearranged and simplified so that all the voltage variables evaluated

at time step $n+1$ are on the left and all those at time step n are on the right as follows

$$-\frac{\eta}{2}V_{i+1}^{n+1} + (1+\eta)V_i^{n+1} - \frac{\eta}{2}V_{i-1}^{n+1} = \frac{\eta}{2}V_{i+1}^{n} + (1-\eta)V_i^{n} + \frac{\eta}{2}V_{i-1}^{n} - \xi\frac{J_i^{n+1}+J_i^n}{2}.$$

As can be seen from the above equations, the Crank-Nicolson method is also an implicit method.

We use the same initial and boundary conditions as for the forward and backward Euler algorithms. By modifying Equation 3.6 at the closed end at $i = 1$ and collecting terms, the difference equation is

$$-\frac{\eta}{2}V_2^{n+1} + (1+\frac{\eta}{2})V_1^{n+1} = \frac{\eta}{2}V_2^{n} + (1-\frac{\eta}{2})V_1^{n} - \frac{1}{2}\xi(J_1^n + J_1^{n+1}).$$

A similar equation results at the other end of the fiber model so that with the incorporation of the closed-end boundary conditions, the matrix equation for the Crank-Nicolson method is

$$\begin{bmatrix} 1+\frac{\eta}{2} & -\frac{\eta}{2} & 0 & \cdots & 0 & 0 & 0 \\ -\frac{\eta}{2} & 1+\eta & -\frac{\eta}{2} & \cdots & 0 & 0 & 0 \\ 0 & -\frac{\eta}{2} & 1+\eta & \cdots & 0 & 0 & 0 \\ 0 & 0 & -\frac{\eta}{2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1+\eta & -\frac{\eta}{2} & 0 \\ 0 & 0 & 0 & \cdots & -\frac{\eta}{2} & 1+\eta & -\frac{\eta}{2} \\ 0 & 0 & 0 & \cdots & 0 & -\frac{\eta}{2} & 1+\frac{\eta}{2} \end{bmatrix} \begin{bmatrix} V_1^{n+1} \\ V_2^{n+1} \\ V_3^{n+1} \\ \vdots \\ V_{I-1}^{n+1} \\ V_I^{n+1} \end{bmatrix} =$$

$$\begin{bmatrix} 1-\frac{\eta}{2} & \frac{\eta}{2} & 0 & \cdots & 0 & 0 & 0 \\ \frac{\eta}{2} & 1-\eta & \frac{\eta}{2} & \cdots & 0 & 0 & 0 \\ 0 & \frac{\eta}{2} & 1-\eta & \cdots & 0 & 0 & 0 \\ 0 & 0 & \frac{\eta}{2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1-\eta & \frac{\eta}{2} & 0 \\ 0 & 0 & 0 & \cdots & \frac{\eta}{2} & 1-\eta & \frac{\eta}{2} \\ 0 & 0 & 0 & \cdots & 0 & \frac{\eta}{2} & 1-\frac{\eta}{2} \end{bmatrix} \begin{bmatrix} V_1^{n} \\ V_2^{n} \\ V_3^{n} \\ \vdots \\ V_{I-1}^{n} \\ V_I^{n} \end{bmatrix} - \frac{\xi}{2}\left(\begin{bmatrix} J_1^n \\ J_2^n \\ J_3^n \\ \vdots \\ J_{I-1}^n \\ J_I^n \end{bmatrix} + \begin{bmatrix} J_1^{n+1} \\ J_2^{n+1} \\ J_3^{n+1} \\ \vdots \\ J_{I-1}^{n+1} \\ J_I^{n+1} \end{bmatrix} \right).$$

The matrix equation can be written as

$$\mathbf{A_{CN1}} \cdot \mathbf{V}^{n+1} = \mathbf{A_{CN2}} \cdot \mathbf{V}^{n} - \frac{\xi}{2}\left(\mathbf{J}^n + \mathbf{J}^{n+1}\right),$$

which has the solution

$$\mathbf{V}^{n+1} = \mathbf{A_{CN1}}^{-1} \cdot \mathbf{A_{CN2}} \cdot \mathbf{V}^n - \frac{\xi}{2}\mathbf{A_{CN1}}^{-1} \cdot \left(\mathbf{J}^n + \mathbf{J}^{n+1}\right).$$

Similar to the procedure with the forward and backward Euler algorithms, we need to compute J_i^n from the factors that determine the ionic conductances. Once again, we let x stand for m, n, and h, and we discretize this equation, this time with the trapezoidal rule, so as to preserve the $O((\Delta t)^2)$ error in time, as follows

$$\begin{aligned}\frac{x_i^{n+1} - x_i^n}{\Delta t} &= \frac{\alpha_x\left(V_i^{n+\frac{1}{2}}\right) - x_i^{n+1}\left(\alpha_x\left(V_i^{n+\frac{1}{2}}\right) + \beta_x\left(V_i^{n+\frac{1}{2}}\right)\right)}{2} \\ &\quad + \frac{\alpha_x\left(V_i^{n+\frac{1}{2}}\right) - x_i^{n}\left(\alpha_x\left(V_i^{n+\frac{1}{2}}\right) + \beta_x\left(V_i^{n+\frac{1}{2}}\right)\right)}{2},\end{aligned}$$

and collect terms to obtain

$$\begin{aligned}x_i^{n+1} &= \frac{\frac{\Delta t}{2}\alpha_x\left(V_i^{n+\frac{1}{2}}\right)}{1 + \frac{\Delta t}{2}\left(\alpha_x\left(V_i^{n+\frac{1}{2}}\right) + \beta_x\left(V_i^{n+\frac{1}{2}}\right)\right)} \\ &\quad + x_i^n \frac{1 - \frac{\Delta t}{2}\left(\alpha_x\left(V_i^{n+\frac{1}{2}}\right) + \beta_x\left(V_i^{n+\frac{1}{2}}\right)\right)}{1 + \frac{\Delta t}{2}\left(\alpha_x\left(V_i^{n+\frac{1}{2}}\right) + \beta_x\left(V_i^{n+\frac{1}{2}}\right)\right)}\end{aligned} \tag{3.7}$$

With the factors m, n, and h substituted for x, the ionic current can be computed as follows

$$(J_{ion})_i^{n+1} = \bar{G}_K(n_i^{n+1})^4(V_i^n - V_K) + \bar{G}_{Na}(m_i^{n+1})^3 h_i^{n+1}(V_i^n - V_{Na}) + G_L(V_i^n - V_L),$$

where each of the factors m_i^{n+1}, n_i^{n+1}, and h_i^{n+1} are computed according to Equation 3.5. The total current is

$$J_i^{n+1} = (J_{ion})_i^{n+1} - \frac{r_o(K_{eo})_i^{n+1} + r_i(K_{ei})_i^{n+1}}{2\pi a(r_o + r_i)}.$$

As with the backward Euler algorithm, the factor x_i^{n+1} depends upon the potential $V_i^{n+\frac{1}{2}}$ through the parameters α and β so that iteration must be used to arrive at a solution of the 4 simultaneous equations.

For linear, parabolic partial differential equations, the Crank-Nicolson method yields stable solutions although not as stable as the backward Euler method, i.e., spurious bounded amplitude oscillations may appear for large values of increments. In contrast to both the forward and backward Euler algorithms, the error resulting from temporal and spatial discretization for the Crank-Nicolson method behaves as $O((\Delta t)^2) + O((\Delta z)^2)$ provided appropriate boundary conditions (such as those shown above) are used. Thus, with the Crank-Nicolson method it is possible to achieve more accurate errors for large time intervals than for the backward Euler method.

3.4 Staggered increment with the Crank-Nicolson method

A computationally efficient method (Hines, 1984) that avoids iteration of the solution is based on the observation that the Crank-Nicolson differencing scheme satisfies the partial differential equation at the discrete time $n + \frac{1}{2}$. This property can be seen by writing out the expression as follows

$$\frac{1}{2\pi a(r_o + r_i)} \left(\frac{V_{i+1}^{n+\frac{1}{2}} - 2V_i^{n+\frac{1}{2}} + V_{i-1}^{n+\frac{1}{2}}}{(\Delta z)^2} \right) = 2C_m \left(\frac{V_i^{n+\frac{1}{2}} - V_i^n}{\Delta t} \right) + J_i^{n+\frac{1}{2}}, \tag{3.8}$$

which results in the following difference equation

$$-\frac{\eta}{2} V_{i+1}^{n+\frac{1}{2}} + (1+\eta) V_i^{n+\frac{1}{2}} - \frac{\eta}{2} V_{i-1}^{n+\frac{1}{2}} = V_i^n - \frac{\xi}{2} J_i^{n+\frac{1}{2}}.$$

Note that starting with the membrane potential at discrete time n, the difference equation computes (by an implicit method) the potential at $n + \frac{1}{2}$. Then the potential at discrete time $n + 1$ is computed by the explicit method

$$V_i^{n+1} = 2V_i^{n+\frac{1}{2}} - V_i^n. \tag{3.9}$$

We use the same initial and boundary conditions as for the other algorithms. By modifying Equation 3.8 at the closed end at $i = 1$ and collecting terms, the difference equation is

$$-\frac{\eta}{2} V_2^{n+\frac{1}{2}} + (1 + \frac{\eta}{2}) V_1^{n+\frac{1}{2}} = V_1^n - \frac{\xi}{2} (J_1^{n+\frac{1}{2}}.$$

A similar equation results at the other end of the fiber model so that the matrix equation

is

$$\begin{bmatrix} 1+\frac{\eta}{2} & -\frac{\eta}{2} & 0 & \cdots & 0 & 0 & 0 \\ -\frac{\eta}{2} & 1+\eta & -\frac{\eta}{2} & \cdots & 0 & 0 & 0 \\ 0 & -\frac{\eta}{2} & 1+\eta & \cdots & 0 & 0 & 0 \\ 0 & 0 & -\frac{\eta}{2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1+\eta & -\frac{\eta}{2} & 0 \\ 0 & 0 & 0 & \cdots & -\frac{\eta}{2} & 1+\eta & -\frac{\eta}{2} \\ 0 & 0 & 0 & \cdots & 0 & -\frac{\eta}{2} & 1+\frac{\eta}{2} \end{bmatrix} \begin{bmatrix} V_1^{n+\frac{1}{2}} \\ V_2^{n+\frac{1}{2}} \\ V_3^{n+\frac{1}{2}} \\ \vdots \\ V_{I-1}^{n+\frac{1}{2}} \\ V_I^{n+\frac{1}{2}} \end{bmatrix} = \begin{bmatrix} V_1^n \\ V_2^n \\ V_3^n \\ \vdots \\ V_{I-1}^n \\ V_I^n \end{bmatrix} - \frac{\xi}{2} \begin{bmatrix} J_1^{n+\frac{1}{2}} \\ J_2^{n+\frac{1}{2}} \\ J_3^{n+\frac{1}{2}} \\ \vdots \\ J_{I-1}^{n+\frac{1}{2}} \\ J_I^{n+\frac{1}{2}} \end{bmatrix}.$$

The matrix equation can be written as

$$\mathbf{A_{CN}} \cdot \mathbf{V}^{n+\frac{1}{2}} = \mathbf{V}^n - \frac{\xi}{2}\mathbf{J}^{n+\frac{1}{2}},$$

which has the solution

$$\mathbf{V}^{n+\frac{1}{2}} = \mathbf{A_{CN}}^{-1} \cdot \left(\mathbf{V}^n - \frac{\xi}{2}\mathbf{J}^{n+\frac{1}{2}}\right). \tag{3.10}$$

This scheme requires that the current be evaluated at the time $n+\frac{1}{2}$ which can be done by iteration as described previously or by having two computing time grids staggered by $\frac{1}{2}$ units of time. To maintain the error due to time quantization at $O((\Delta t)^2)$, we need to compute the factors with second-order accuracy by a centered difference as follows

$$\frac{x_i^{n+\frac{1}{2}} - x_i^{n-\frac{1}{2}}}{\Delta t} = \alpha_x\left(V_i^n\right) - \left(\alpha_x\left(V_i^n\right) + \beta_x\left(V_i^n\right)\right) \frac{x_i^{n+\frac{1}{2}} + x_i^{n-\frac{1}{2}}}{2}$$

and collect terms to obtain

$$x_i^{n+\frac{1}{2}} = \frac{\Delta t \alpha_x\left(V_i^n\right)}{1 + \frac{\Delta t}{2}\left(\alpha_x\left(V_i^n\right) + \beta_x\left(V_i^n\right)\right)} + x_i^{n-\frac{1}{2}} \frac{1 - \frac{\Delta t}{2}\left(\alpha_x\left(V_i^n\right) + \beta_x\left(V_i^n\right)\right)}{1 + \frac{\Delta t}{2}\left(\alpha_x\left(V_i^n\right) + \beta_x\left(V_i^n\right)\right)} \tag{3.11}$$

With the factors m, n, and h substituted for x, the ionic current can be computed as follows

$$(J_{ion})_i^{n+\frac{1}{2}} = \bar{G}_K (n_i^{n+\frac{1}{2}})^4 (V_i^n - V_K) + \bar{G}_{Na} (m_i^{n+\frac{1}{2}})^3 h_i^{n+\frac{1}{2}} (V_i^n - V_{Na}) + G_L (V_i^n - V_L),$$

where each of the factors $m_i^{n+\frac{1}{2}}$, $n_i^{n+\frac{1}{2}}$, and $h_i^{n+\frac{1}{2}}$ are computed according to Equation 3.11. The total current is

$$J_i^{n+\frac{1}{2}} = (J_{ion})_i^{n+\frac{1}{2}} - \frac{r_o (K_{eo})_i^{n+\frac{1}{2}} + r_i (K_{ei})_i^{n+\frac{1}{2}}}{2\pi a (r_o + r_i)}.$$

The method is outlined as follows:

- Start with the value of V_i^n and the value of $x_i^{n-\frac{1}{2}}$ which allows a computation of $J_i^{n-\frac{1}{2}}$;
- Compute $x_i^{n+\frac{1}{2}}$ using Equation 3.11 which then allows a computation of $J_i^{n+\frac{1}{2}}$;
- Compute $V_i^{n+\frac{1}{2}}$ using Equation 3.10;
- Compute V_i^{n+1} using Equation 3.9;
- The solution has now been advanced to V_i^{n+1} and the value of $x_i^{n+\frac{1}{2}}$.

The process can be initiated by calculating either V_i or x_i for a $\frac{1}{2}$ time step increment to stagger the variables.

Chapter 4

Matlab Software

This software package is part of *Softcell* which includes other simulation packages (such as *Random Walk Model of Diffusion*, *Macroscopic Diffusion Processes*, *Single-compartment Hodgkin-Huxley Model*, etc.). When this software is selected from the *Softcell* menu, two figures are displayed (in addition to MATLAB's command window): *PAP Control*, and *PAP Workspace*; The *PAP Workspace* is distinct from MATLAB's workspace. The *PAP Control* figure allows the user to display other figures for setup and analysis, and to start a simulation. The *PAP Workspace* figure is (as the name suggests) a workspace where a schematic representation of an axon, a stimulus source, and a voltage recorder are displayed. In the *PAP Workspace* figure, the user can choose to display the stimulus setup figure (*PAP Stimulus*), the parameter setup figure (*PAP Parameters*), the voltage recorder figure (*PAP Voltage-Recorder*), and place the electrodes of the stimulus source and recorder using the mouse cursor on the axon schematic. Clicking on the *Start* pushbutton in the *PAP Control* figure results in the computation of the response as a function of time and space. Upon completion of the computation, the user can choose to view the space-time evolution of the membrane potential (which is one of the computed solution variables) by means of a color code along the axon schematic in the *PAP Workspace* figure. The user can also examine the space-time evolution of other response solution-variables to gain insight to the model's dynamic behavior or use 3D and comparison plots for static analyses of the model's behavior.

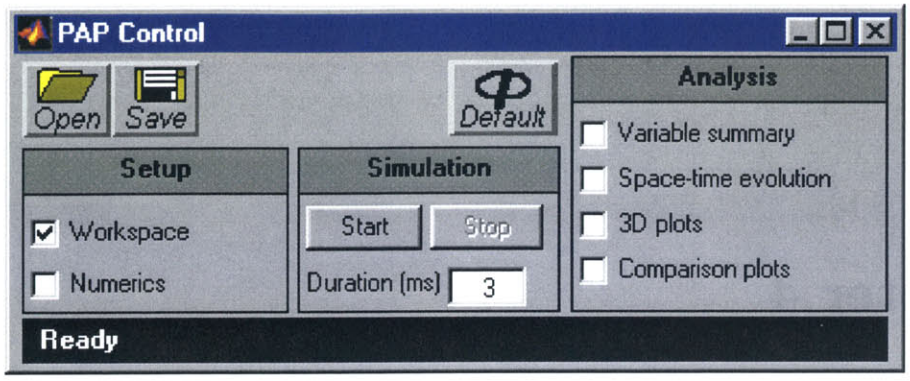


Figure 4-1: The PAP Control figure when the software is initiated.

4.1 PAP Control

There are three panels in the *PAP Control* figure (Figure 4-1). With the left panel, the user can display the *PAP Workspace* figure (See Section 4.2) and the *PAP Numerics* (See Section 4.5). In the center (Simulation) panel, the user can specify the simulation duration and start the simulation by clicking on the *Start* pushbutton. The right (Analysis) panel allows the user to do post simulation analysis of the results. The first (Variable summary) option allows the user to display a numerical summary of all the solution variables. The second (Space-time evolution) option allows the user to view the dynamics of a solution variable (such as membrane potential, membrane current-components, etc.) as a function of space with running time. The third (3D plots) option allows the user to view three-dimensional graphs of any solution variable as a function of time and space. The fourth (Comparison plots) option allows the user to compare multiple solution variables as a function of a solution variable of choice (or as a function of time or space) in two different modes: fixed-time or fixed-space.

Starting, pausing, and stopping a simulation

Clicking on the *Start* pushbutton starts the computation of the solution variables (i.e. a simulation run). While a computation is in progress, the status-bar located along the bottom of the figure shows the name of the numerical method being used and the percentage of completion. The *Start* pushbutton changes to a *Pause* pushbutton. Clicking on the *Pause* pushbutton pauses the computation and causes the status-bar to display *Paused.* When the

simulation is paused, the *Pause* pushbutton changes to a *Continue* pushbutton. Clicking on the *Continue* pushbutton resumes the computation and the two states (pause/continue) toggle with each click of the pushbutton until the computation is completed. Upon completion, the pushbutton changes back to *Start* and the status-bar acknowledges successful completion along with a time stamp. Next to *Start* is the *Stop* pushbutton which allows the user to abort a computation that is currently in progress or paused. Clicking on the *Stop* pushbutton causes the status-bar to display *Simulation Aborted.*

Loading, saving, and re-initializing

The user can save all simulation parameters and results in a file as well as load a previously saved file by clicking on the *Save* and *Open* button, respectively. The data stored includes the parameter values of the axon, the membrane, the bath, the stimulus, the configuration of the electrodes, and solution variables. When a file is loaded, the filename (along with it's directory path) is shown in this figure's status-bar as well as in the status-bar of other setup figures such as *PAP Parameters*, *PAP stimulus*, and *PAP numerics.* This is useful for keeping track of the source of the parameter values used during a session.

When the software is launched, the figure's status-bar displays *Ready*. This indicates that no simulation has been run and that there are no results. This is the initial state of the software and all parameters are set to their default values. All parameter values can be set back to their default values by clicking on the *Default* pushbutton.

4.2 PAP Workspace

Selecting *Workspace* in the *PAP Control* figure displays the *PAP Workspace* figure (Figure 4-2). The *PAP Workspace* figure consists of a window and a control region. The window contains from top to bottom: a ruler, an interactive display area (showing a schematic of an axon, a stimulus-source icon, and a voltage-recorder icon), another ruler, a status-bar, and a horizontal scrollbar. The default length of the axon is three centimeters and fits in the window's display area. The schematic representation of a longer axon model stretches beyond the display area's right edge. The user can use the horizontal scrollbar to scroll to the hidden portion. By default the top ruler is set to show number of discretized segments, and the bottom ruler is set to show distance in centimeters. Clicking on either ruler will

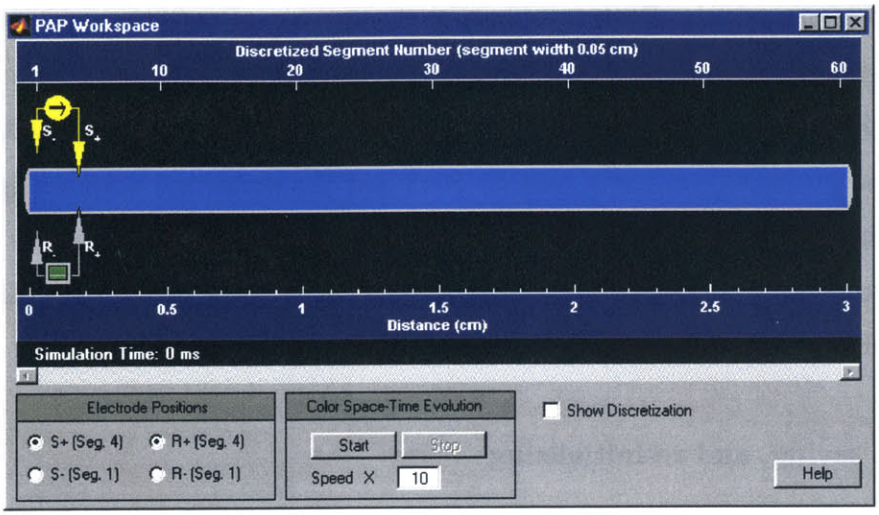


Figure 4-2: The PAP Workspace figure.

toggle its unit of measurement between number of discretized segments and distance. The status-bar shows the simulation time.

Clicking on the *Help* button located in the lower right corner of the *PAP Workspace* figure displays the *PAP Workspace Help* figure. This figure shows a step-by-step guide to setting up a customized simulation as described in Section 4.11.

Positioning electrodes

In the interactive display area, the user can use the mouse cursor to adjust the location of the stimulus electrodes (yellow) located right above the axon schematic. Likewise, the user can adjust the location of the voltage-recorder electrodes (grey) located beneath the axon schematic. To position an electrode, the user must first select the appropriate radiobutton in the *Electrode Positions* panel to control the desired electrode. Note that the positive and negative electrodes of the stimulus ($S+$ and $S-$) are selected independently from the positive and negative electrodes of the recorder ($R+$ and $R-$). After the appropriate radiobutton is selected to control the desired electrode of either the stimulus or recorder, clicking anywhere outside of the axon schematic within the interactive display area (eg. the black region) sets the electrode's longitudinal position to the mouse-click location. Clicking in the black region above the axon schematic moves the stimulus electrodes, and clicking in the black region below the axon schematic moves the recorder electrodes. Clicking on an

electrode will toggle the electrode position between intracellular and extracellular.

Displaying other setup figures

Clicking on the stimulus-source icon (yellow) will bring up the *PAP Stimulus* figure where the user can specify the characteristics of the desired stimulus (See Section 4.4). Clicking on the voltage-recorder icon (green and grey) will display the *PAP Voltage-Recorder* figure where a real-time potential curve is plotted during a simulation run (See Section 4.6). Clicking on the axon schematic will display the *PAP Parameters* figure where the user can change the parameters of the axon, the membrane, and the bath (See Section 4.3).

Showing discretization and color-coded space-time evolution

The option (Show discretization) to the right of the *Color Space-Time Evolution* panel allows the user to toggle on/off the spatial discretization display on the axon schematic. In the *Color Space-Time Evolution* panel, the pushbutton *Start* allows the user to playback the space-time evolution of the membrane potential on the axon schematic by using a color code to convey the range of potential levels. The playback can be stopped using the *Stop* button. During playback, the *Start* becomes a *Pause* button which allows the user to pause the playback. The speed of the playback can be adjusted in the *Speed* field located under the pushbutton. For example, if the simulation data contains membrane potential values for every 0.01 ms, changing the field value to ten will playback simulation data of every 0.1 ms.

Note that a voltage curve is plotted in the *Voltage-Recorder* figure during playback (See Section 4.6). Figure 4-17 at the end of this chapter shows a series of snapshots taken from a time-evolution color code visualization of the membrane potential of a three-centimeters long axon model.

4.3 PAP Parameters

Clicking on the axon schematic in the *PAP Workspace* figure displays the *PAP Parameters* figure (Figure 4-3). The *PAP Parameters* figure is divided into three sections: *Membrane Characteristics*, *Bath Characteristics*, and *Axon Characteristics*. In each section, there are three columns for each parameter: the first column identifies the parameter, the second gives

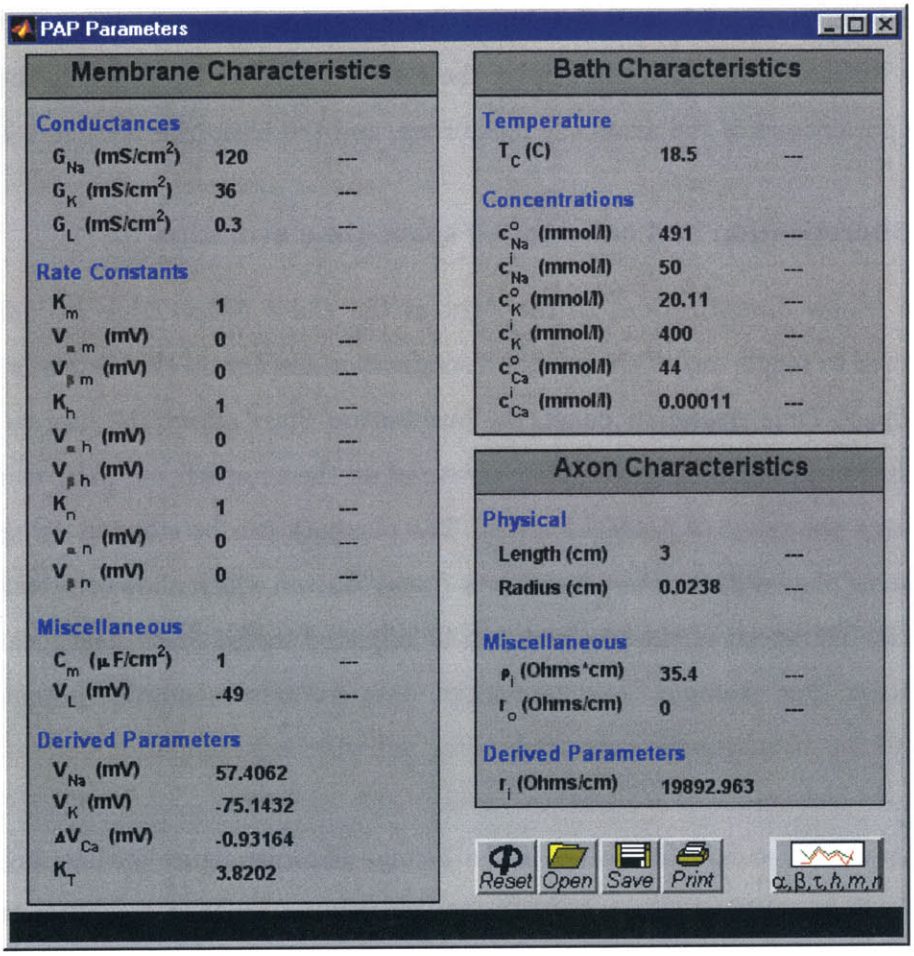


Figure 4-3: The PAP Parameters figure.

its default value, and the third column shows the value of the parameter if it is modified from its default value *(Note that the PAP Parameters figure data entry method is based on that of the HH Parameters figure).*

Changing parameter values

Parameters can be modified by clicking on the entry in the third column of the parameter field. When the parameter is selected, its third column entry is replaced by an edit box. Clicking any mouse button inside the edit box allows the parameter to be changed. The new value of the parameter is entered from the keyboard; < RETURN > terminates the parameter entry. If the new value differs from the default value, then the new value will appear in the third column. With this method of display, the parameter list can be scanned quickly to indicate which parameters differ from their default values.

There are some restrictions on the numerical values of parameters; G's ≥ 0, $C_m \geq 0$, c's > 0, K's > 0, $\rho_i \geq 0$, $r_o \geq 0$ (r_o cannot be zero when $\rho_i = 0$), the axon's radius and length must be greater than zero, and the temperature must be above absolute zero. The derived parameters cannot be changed but are derived from the other parameters and are displayed for the convenience of the user. For example, the sodium equilibrium potential V_{Na} cannot be changed directly by the user, but changes automatically when the sodium concentrations or temperature are changed.

Loading, saving, resetting, and printing

The parameters can be loaded from a file by clicking on *Open* and can be saved by clicking on *Save.* Saving and loading in the *PAP Parameters* figure deals only with the model's axon, membrane and bath parameters; e.g., simulation results are not loaded or saved. When a parameters file is loaded, the status-bar of the *PAP Parameters* figure displays the filename (with the directory path) and the parameter values loaded from the file become the default values. For example, if the user changes the temperature value and clicks the *Reset* pushbutton, the temperature value resets to the value stored in the loaded file. To reset the parameter values to their software-default values, use the *Default* pushbutton in the *PAP Control* figure. If no file is loaded, clicking the *Reset* pushbutton resets the parameter values to the software-default values. The *PAP Parameters* figure can be printed using the *Print* button.

Viewing the voltage dependent parameters

The Hodgkin-Huxley model of a propagated action potential as well as the Hodgkin-Huxley model of a membrane action potential of a space-clamped axon contains several parameters that are instantaneous functions of the membrane potential. Plots of these parameters are available as a function of V_m for the following: $\alpha_m(V_m)$, $\beta_m(V_m)$, $m_\infty(V_m)$, $\tau_m(V_m)$, $\alpha_h(V_m)$, $\beta_h(V_m)$, $h_\infty(V_m)$, $\tau_h(V_m)$, $\alpha_n(V_m)$, $\beta_n(V_m)$, $n_\infty(V_m)$, $\tau_n(V_m)$ (see Section 2.3). These can be accessed by clicking on the button marked $\alpha, \beta, \tau, h, m, n$ which results in the display of the *PAP Parameters vs. Potential* figure (Figure 4-4).

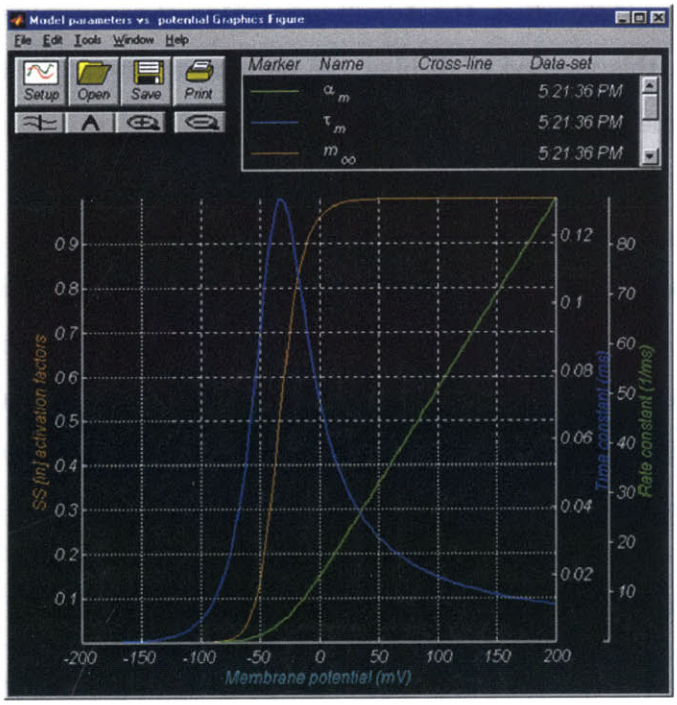


Figure 4-4: The PAP Parameters vs. Potential figure.

Note that the *PAP Parameters vs. Potential* figure is similar to the *PAP Comparison Plots* figure described in Section 4.10. Please refer to that section for more information about the available plot manipulation functions. The *PAP Parameters vs. Potential* figure has its own setup figure which differs from the one of the *PAP Comparison Plots* . Clicking on the *Setup* button displays this setup figure. The user can select one or more variables and click on either *Graph* or *Overlay* to make the plot. The *Graph* and *Overlay* buttons are described in Section 4.10.

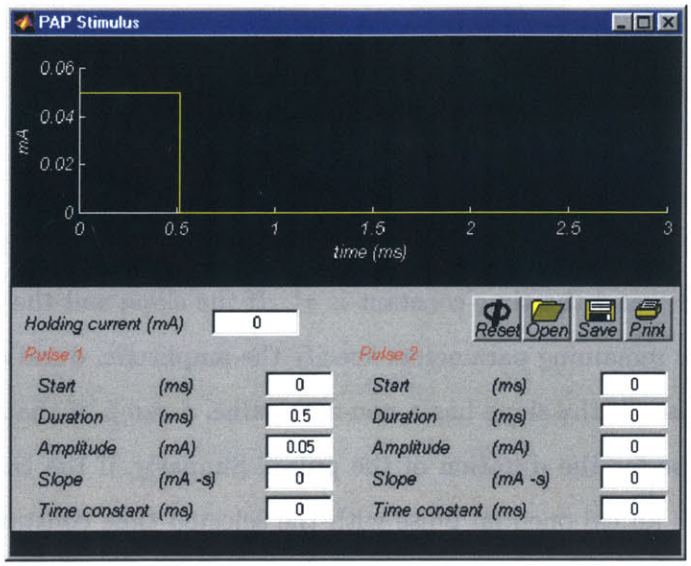


Figure 4-5: The PAP Stimulus figure.

4.4 PAP Stimulus

Clicking on the stimulus-source icon (yellow) in the *PAP Workspace* figure displays the *PAP Stimulus* figure (Figure 4-5). Note that the layout of the *PAP Stimulus* figure is similar to that of the HH Stimulus figure. The upper-half of the figure shows a graph of the stimulus pulse(s) specified in the parameter-entry columns: *Pulse 1* and *Pulse 2*. The duration of the stimulus depends on the duration of the simulation specified in the *PAP Control* figure. For example, if the simulation duration is four milliseconds, then the abscissa of the stimulus graph is also of length four milliseconds.

Specifying a pulse

A maximum of two pulses can be defined using a set of parameters for each pulse. In addition, a holding DC current can be specified in the *Holding current* field. Thus, the overall stimulus is the sum of a constant plus two independently specifiable pulses. In mathematical terms, the stimulus, $s(t)$, is defined as follows

$$s(t) = s^o + s^1(t) + s^2(t), \tag{4.1}$$

where s^o is a constant and $s^1(t)$ and $s^2(t)$ are independently specifiable pulses. Each pulse has the form

$$s^i(t) = \begin{cases} m^i(t - t_o^i) + A^i e^{-(t-t_o^i)/\tau^i} & \text{if } t_o \leq t \leq t_o + t_{dur}, \\ 0 & \text{otherwise,} \end{cases} \tag{4.2}$$

where pulse i ($i = 1$ or 2) starts at t_o and has a duration t_{dur}. The amplitude of the pulse is A^i, the slope is m^i, and the time constant is τ^i. If the slope and the time constant are set to zero, then the remaining parameters specify the amplitude, duration and onset time of a rectangular pulse. If the slope has a non-zero value, a ramp of that slope is added to the rectangular pulse for the duration of the pulse. Similarly, if the time constant has a non-zero value, then an exponential pulse with the selected time constant is added to the waveform. The time constant can be positive or negative.

The variables t_o^i, t_{dur}^i, A^i, m^i, and τ^i that defines a pulse in the form of Equation 4.2 are specified in the *PAP Stimulus* figure by the parameter fields *Start*, *Duration*, *Amplitude*, *Slope*, and *Time constant* in the *Pulse i* column, respectively. By adjusting these parameters, the user can generate rectangular pulses, ramp pulses, exponential pulses or a combination of these. An example is shown in Figure 4-6.

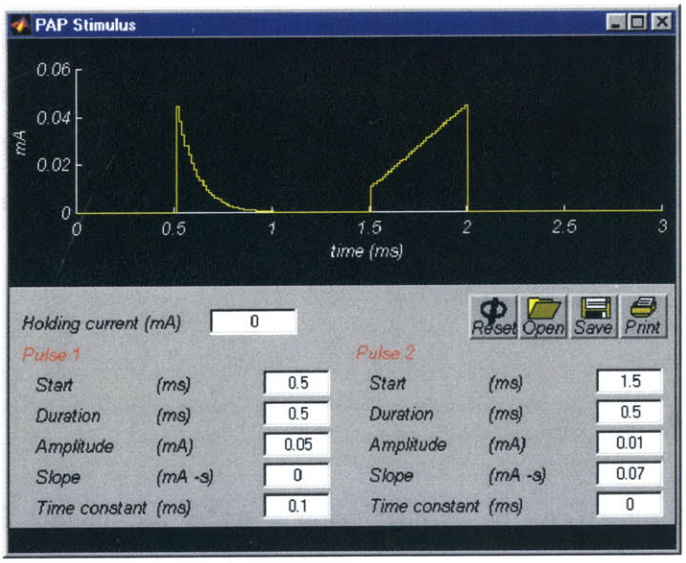


Figure 4-6: The PAP Stimulus figure showing a stimulus that consists of two pulses: the first pulse is an exponential pulse and the second is an ramp plus a rectangular pulse.

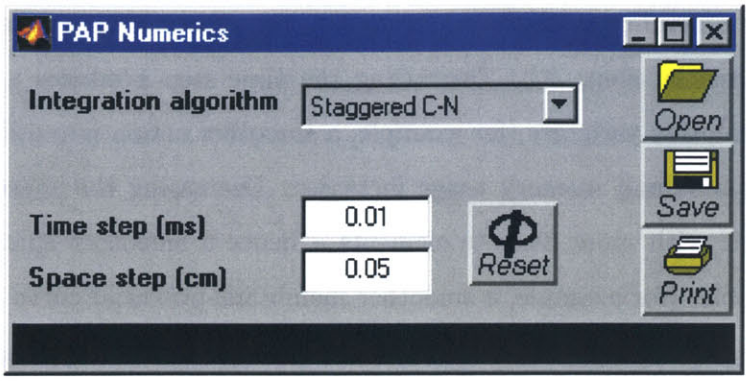


Figure 4-7: The PAP Numerics figure.

Loading, saving, and resetting

The parameters can be loaded from a file by clicking on *Open* and can be saved by clicking on *Save*. Saving and loading in the *PAP Stimulus* figure deals only with the stimulus parameters; e.g., simulation results are not loaded or saved. When a file with stimulus parameters is loaded, the status-bar of the *PAP Stimulus* figure displays the filename (with the directory path) and the parameter values loaded from the file become the default values. For example, if the user changes the holding current value and clicks the *Reset* pushbutton, the holding current value resets to the value stored in the loaded file. To reset the parameter values to their software-default values, use the *Default* pushbutton in the *PAP Control* figure. If no file is loaded, clicking the *Reset* pushbutton resets the parameter values to the software-default values.

4.5 PAP Numerics

Selecting *Numerics* in the *PAP Control* figure displays the *PAP Numerics* figure (Figure 4-7). The *PAP Numerics* figure allows the user to select the numerical method used to solve the partial differential equations described in Chapter 2.

Choosing a numerical method and specifying discretization parameters

The list-box located at the top of the figure allows the user to select one of four available numerical methods: forward Euler, backward Euler, Crank-Nicolson, and staggered Crank-Nicholson (staggered C-N). These numerical methods are described in detail in Chapter 3.

By default, the software uses the staggered C-N numerical method. All numerical methods employ the finite difference method which requires two discretization parameters: a time step, Δt, and a spatial step, Δz. Decreasing the time step produces a smoother time evolution of the solution variables (for example, a smoother action potential propagation), but computation time and memory usage increases. Decreasing the space step produces an axon schematic with more discrete elements – hence a smoother spatial evolution of the solution variables (for example, a smoother membrane-potential curve as a function of space when viewed in a snapshot), but computation time and memory usage increases. In order to obtain a reasonably accurate solution using a particular numerical method, the discretization parameters have to be chosen appropriately. For more information regarding the performance of each numerical method and restrictions on the discretization parameters, refer to Chapter 5.

Loading, saving, resetting, and printing

The parameters can be loaded from a file by clicking on *Open* and can be saved by clicking on *Save*. Saving and loading in the *PAP Numerics* figure deals only with the numerics parameters; e.g., simulation results are not loaded or saved. When a numerics parameters file is loaded, the status-bar of the *PAP Numerics* figure displays the filename (with the directory path) and the parameter values loaded from the file become the default values. For example, if the user changes the time step and clicks the *Reset* pushbutton, the time step resets to the value stored in the loaded file. To reset the parameter values to their software-default values, use the *Default* pushbutton in the *PAP Control* figure. If no file is loaded, clicking the *Reset* pushbutton resets the parameter values to the software-default values. The *PAP Numerics* figure can be printed using the *Print* button.

4.6 PAP Voltage-Recorder

Clicking on the voltage-recorder icon (grey and green) in the *PAP Workspace* figure displays the *PAP Voltage-Recorder* figure (Figure 4-8). The *PAP Voltage-Recorder* figure shows a graphing region where a voltage curve is plotted when the user clicks on the *Start* button located in the *PAP Workspace* figure. When plotting a curve, the graphing area can display only a portion 3 ms long. If the plot extends beyond 3 ms, the scrollbar under the graphing

area becomes active, so the user can scroll to see any section of the curve. Also, a *Show entire plot* option is enabled, so the user can choose to display the whole curve on the graphing area. For example, shrink the abscissa to fit the entire plot in the graphing region.

The voltage plot corresponds to the voltage recorded at the location of the recording electrodes located beneath the axon schematic. The user can re-position the recording electrodes as described in Section 4.2 and click on *Start* to obtain the corresponding voltage plot. Clicking on the *Grid* option displays a grid on the graphing region.

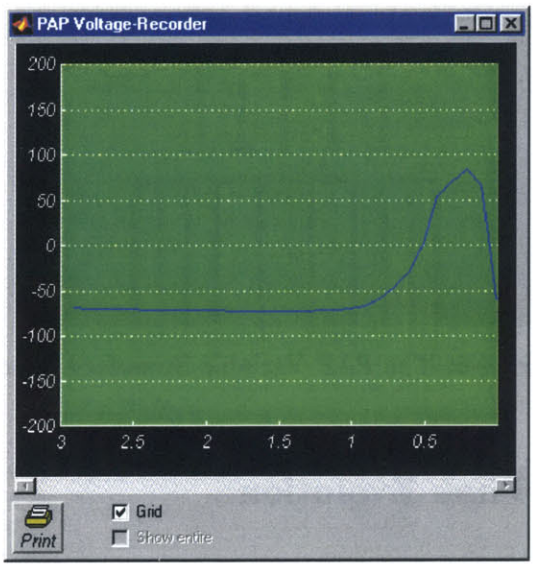
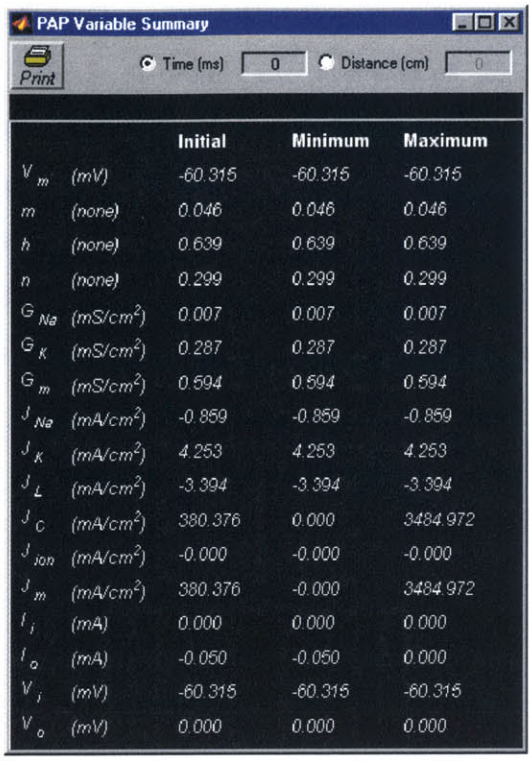


Figure 4-8: The PAP Voltage-Recorder figure showing a potential reading.

4.7 PAP Variable Summary

Selecting *Variable summary* in the Analysis panel of the *PAP Control* figure displays the *PAP Variable Summary* figure as shown in Figure 4-9. This figure allows the user to display the initial, minimum, and maximum value of any solution variable. The user can either select to view these summary values obtained from the spatial dimension at a specified time, or from the temporal dimension at a specified location. The time and location are specified in units of milliseconds and centimeters, respectively. The largest valid time value is equal to the simulation duration, and the largest valid location value is equal to the length of the axon model.



		Initial	Minimum	Maximum
V_m	(mV)	-60.315	-60.315	-60.315
m	(none)	0.046	0.046	0.046
h	(none)	0.639	0.639	0.639
n	(none)	0.299	0.299	0.299
G_{Na}	(mS/cm^2)	0.007	0.007	0.007
G_K	(mS/cm^2)	0.287	0.287	0.287
G_m	(mS/cm^2)	0.594	0.594	0.594
J_{Na}	(mA/cm^2)	-0.859	-0.859	-0.859
J_K	(mA/cm^2)	4.253	4.253	4.253
J_L	(mA/cm^2)	-3.394	-3.394	-3.394
J_C	(mA/cm^2)	380.376	0.000	3484.972
J_{ion}	(mA/cm^2)	-0.000	-0.000	-0.000
J_m	(mA/cm^2)	380.376	-0.000	3484.972
I_i	(mA)	0.000	0.000	0.000
I_o	(mA)	-0.050	-0.050	0.000
V_i	(mV)	-60.315	-60.315	-60.315
V_o	(mV)	0.000	0.000	0.000

Figure 4-9: The PAP Variable Summary figure.

4.8 PAP Space-Time Evolution

Selecting *Space-time evolution* in the Analysis panel of the *PAP Control* figure displays the *PAP Space-Time Evolution* figure as shown in Figure 4-10. This figure only appears if there are simulation results (i.e. after a simulation has been run). The *PAP Space-Time Evolution* figure provides a means of viewing the dynamics of any solution variable along the axon as time changes.

Viewing the space-time evolution of a solution variable

When the figure is first displayed, V_m (membrane potential) is selected in the list-box located at the top of the figure. Use the list-box to view any desired solution variable. After a selection is made, the graph under the list-box shows the selected solution variable as a function of space at time = 0 ms. To view the selected solution variable as a function of space at progressing time starting from 0 ms, click on the *Start* pushbutton. The process can be stopped and paused using the appropriate buttons. The *Start* button changes to

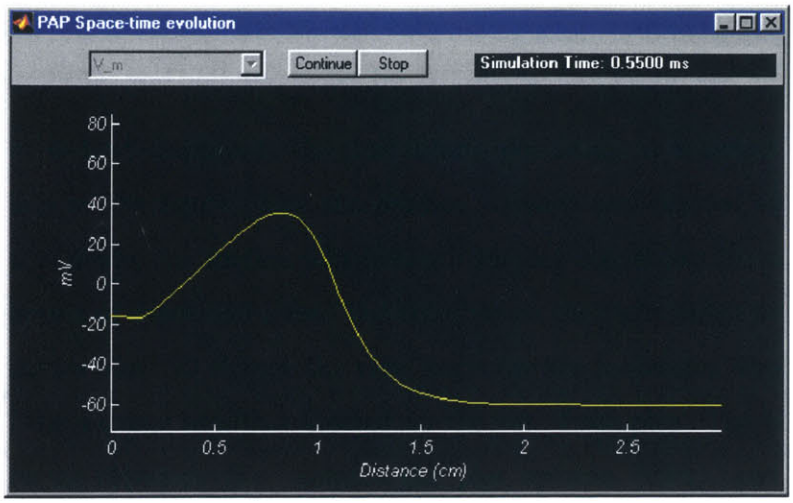


Figure 4-10: The PAP Space-Time Evolution figure showing a snapshot of a propagating action potential.

a *Pause* button when a process is running. Clicking on an axis label displays the *Axis configuration* figure (Figure 4-15) which allows the user to change the properties of the axis.

4.9 PAP 3D Plots

Selecting *3D plots* in the Analysis panel of the *PAP Control* figure displays the *PAP 3D Plots* figure (Figure 4-11). This figure only appears if there are simulation results (i.e. after a simulation has been run). The *PAP 3D Plots* figure provides a means of visualizing the space-time dependence of any solution variable.

Viewing a 3D plot of a solution variable

When the figure is first displayed, V_m (membrane potential) is selected in the list-box located at the top of the figure. Use the list-box to view any desired solution variable. After a selection is made, the surface plot beneath the list-box shows the selected solution variable as a function of space and time. The colorbar on the right indicates the magnitude of the colored surface plot. To change the view angle use the horizontal scrollbar located at the bottom and the vertical scrollbar located on the left of the plot. The vertical scrollbar controls the viewing elevation (i.e. rotates the plot up and down), and the horizontal

scrollbar controls the viewing azimuth (i.e. rotates the plot left and right). In addition, the user can switch between mesh and surface plot by toggling the *mesh* option *on/off* .

Clicking on the *box* option toggles *on/off* an axes-bounding box. Clicking on a solution variable's axis label displays the *Axis configuration* figure (Figure 4-15) which allows the user to change the properties of that axis.

4.10 PAP Comparison Plots

Selecting *Comparison plots* in the Analysis panel of the *PAP Control* figure displays the *PAP Comparison Plots* figure as shown in Figure 4-13. The *PAP Comparison Plots* figure allows the user to plot any solution variable against any other. The figure displays a group of pushbuttons on the upper-left corner, a legend panel on the upper-right corner, and a graphing region.

Using the Comparison Plots Setup figure to generate plots

Clicking on the *Setup* button in the *PAP Comparison Plots* figure displays the *Comparison Plots Setup* figure (See Figure 4-12) where the user can choose: fixed-space plots (Figure 4-13) or fixed-time plots (Figure 4-14). Fixed-space plots require the user to specify a longitudinal location of the axon, z_o, in units of centimeters. The user can use this mode to generate *solution-variable-A vs. solution variable-B at location,* z_o, *plots*. Fixed-time plots require the user to specify a time, t_o, in units of milliseconds. This mode is used to generate *solution-variable-A vs. solution-variable-B at time,* t_o, *plots*.

Follow these steps to make a plot:

- Choose a plotting mode: fixed-space plots or fixed-time plots.
- For fixed-space plots specify the *Distance* parameter: enter a value in units of centimeters. Note that the maximum allowable value is the length of the axon model. Multiple distance values can also be entered in MATLAB array format (i.e. [1.0 1.5 3.5], [1.0 : 0.05 : 2.0]). Please refer to the MATLAB user's manual for more information.
- For fixed-time plots specify the *Time* parameter: enter a value in units of milliseconds. Note that the maximum allowable value is equal to the simulation duration. Multiple time values can also be entered in MATLAB array format.

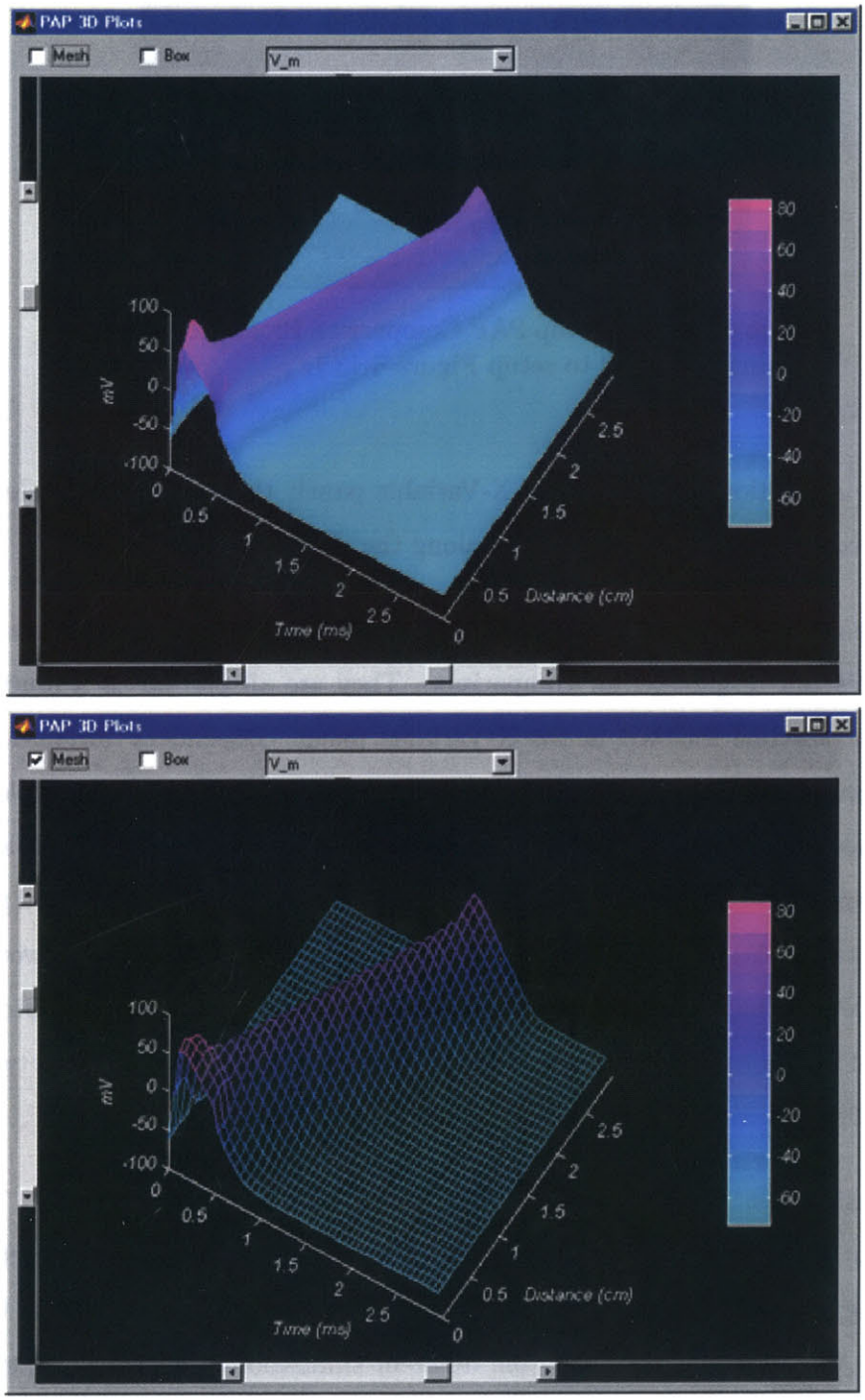


Figure 4-11: The PAP 3D Plots figure showing a propagating action potential. Top: a surface plot. Bottom: a mesh plot.

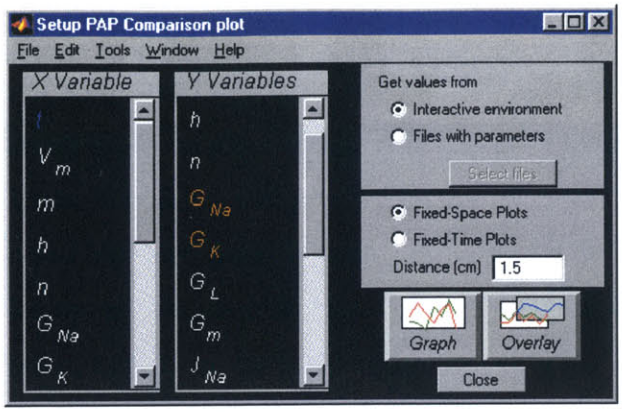


Figure 4-12: The Setup PAP Comparison Plot figure showing the variables used to setup Figure 4-13.).

- Choose a solution variable in the X-Variable panel: this is the independent variable and it corresponds to the the values along the abscissa.

- Choose one or more solution variables in the Y-Variable panel: multiple dependent variables can be plotted for comparison. They are all plotted as a function of the solution variable selected in the X-Variable panel. Dependent variables that have same unit of measurements share one ordinate axis. Both sides of the plotting area may show distinct ordinate axes to accomodate distinct units of the variables selected. For example, the user can choose to plot the membrane potential, V_m, and the sodium conductance, G_{Na}, both as a function of time, t. In this case, the user would select t in the X-Variable panel, and both V_m and G_{Na} in the Y-Variable panel. Since V_m is in units of millivolts and G_{Na} is in units of $\mathrm{mS/cm^2}$, the plot should show two ordinate axes. The abscissa, in this case, should be in units of milliseconds.

- Click on *Graph* to graph the selections on a cleared plotting region. The *Graph* pushbutton clears all previous plots. Click on *Overlay* to graph the selection on the plotting region without clearing any previous plots. This feature allows the user to overlay the solutions obtained from different simulations. For example, suppose the user wants to plot the membrane potential, V_m, as a function of time, t, at the axon longitudinal location, $z_o = 2$ cm, for temperature values of 6.5°, 18°, and 20° Celsius. The user would need to run a simulation with the temperature parameter set to 6.5, plot V_m vs. t at $z_o = 2$ using the *Graph* button. Then do the subsequent simulation

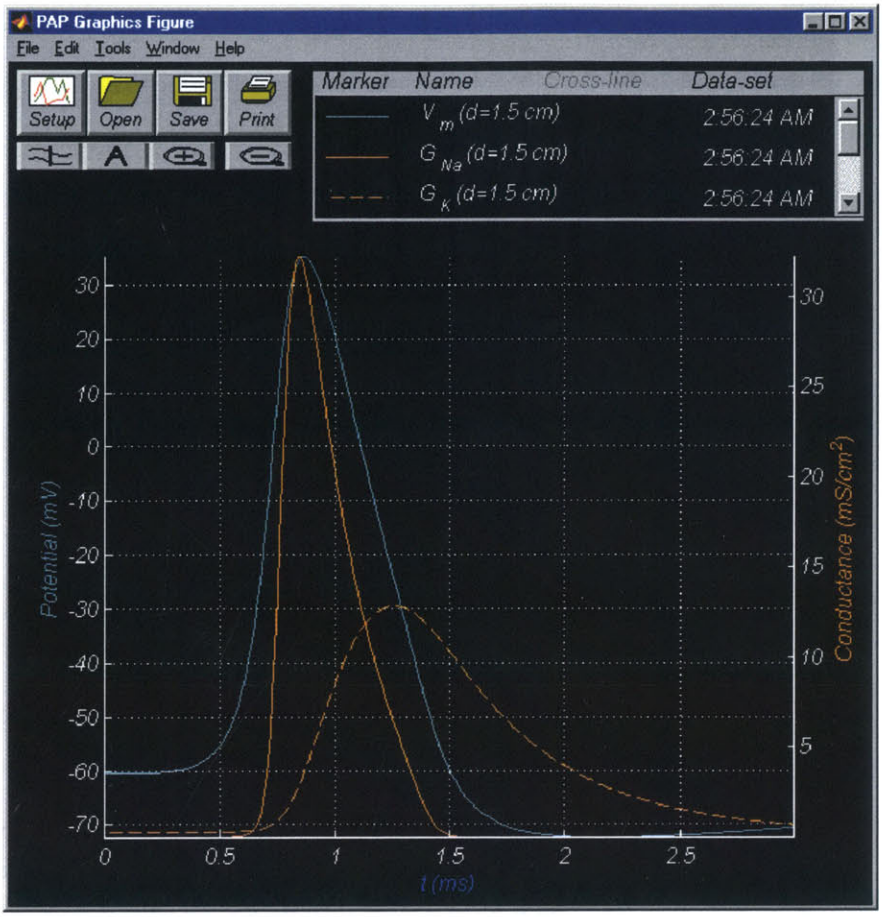


Figure 4-13: The PAP Comparison Plots figure showing a fixed-space plot of the membrane potential, V_m, and the ionic conductances, G_{Na} and G_K, with model parameters and the stimulus defined in Figures 4-3 and 4-5.

runs with the the other two temperature values and plot using the *Overlay* button so that all three plots are retained in the plotting region.

In addition, the user can make plots using data stored in files. To do this, select the *Files with Parameters* option and click on the *Select files* pushbutton. A file-loading dialog box appears which allows the user to select the data file(s). The user can then plot the loaded solution variables.

Annotating, zooming, and other functions

The *PAP Comparison Plots* figure has a few functions that facilitates the analysis of the plots.

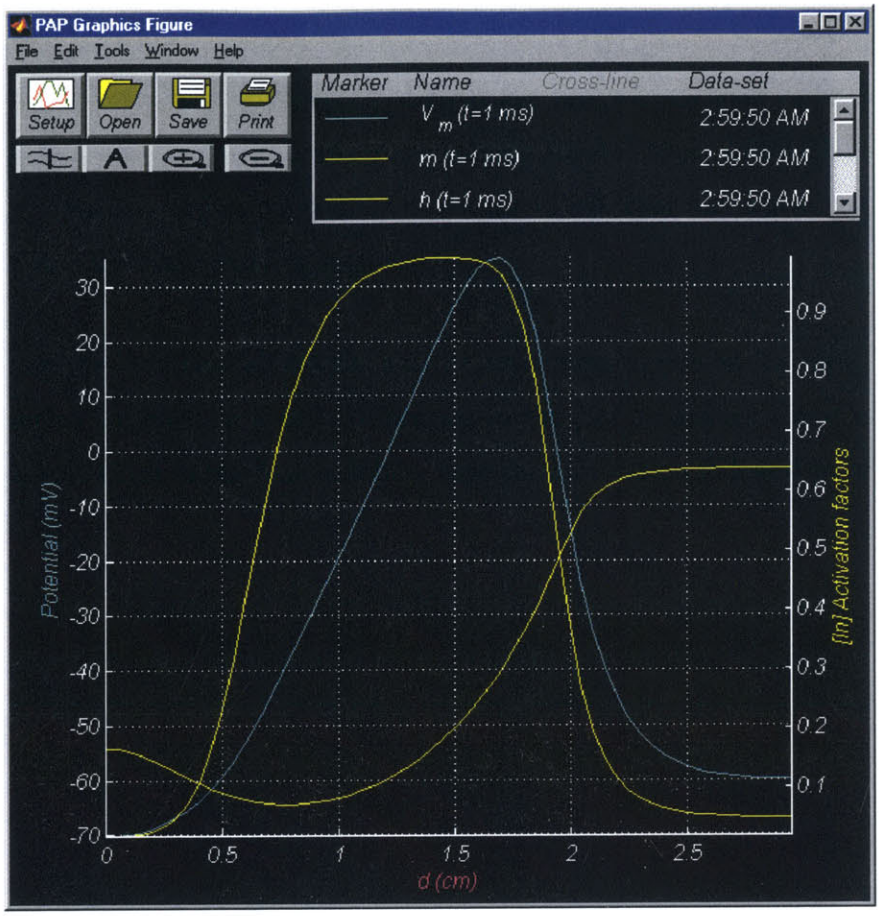


Figure 4-14: The PAP Comparison Plots figure showing a fixed-time plot of the membrane potential, V_m, and the activation factors, m and h, obtained from a simulation with model parameters and the stimulus defined in Figures 4-3 and 4-5.

- Clicking on the *Cross-line* pushbutton displays a vertical line in the plot region at the location of the pointer cursor. The line follows the cursor as it moves across the plot region. The values of all plotted variables at the intersection with the cross-line are displayed in the Legend panel, under the column labeled *Cross-line* . Clicking on the *Cross-line* button again removes the cross-line values from the legend and removes the cross line from the plotting region.

- Clicking on the *A* button allows the user to make annotations on the plot. With this button set to *on*, click on a desired location in the plot to display a text edit box at that location. Click in the text edit box, type the annotation followed by a `<RETURN>`

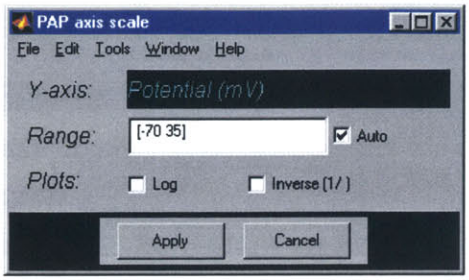


Figure 4-15: The PAP Axis Scale figure.

when the annotation is completed. Clicking on the annotation and dragging the mouse moves the annotation to a desired location in the plot field. The text string can be formatted into mathematical notation by using a LaTeX like notation, e.g., to get m_∞ type `m_{\infty}`.

- Clicking on the + button next to the annotation button allows the user to magnify a region of the plotting region by clicking on it. Clicking again causes further magnification. Clicking on the − button causes the plot to change to its original magnification.

- In the plotting region, clicking on any of the axis labels (i.e. either the abscissa or any of the ordinates) displays an axis scale figure (Figure 4-15) which allows the user to change parameters of the chosen axis. The range of the axis can be specified in MATLAB array format (i.e. [-200 200]) in the *Range* field. The range is set to rescale automatically by default. To inhibit auto-scaling, unselect the *Auto* option. Selecting the *Log* option changes the axis scale to a logarithmic scale (i.e. the logarithm of the magnitude of the variable is plotted). Selecting the *Inverse [1/]* option produces a reciprocal scale axis. (i.e. the reciprocal of the magnitude of the variable is plotted). A combination of the two is allowed.

- In the Legend panel, clicking on any entry under the column labeled *Marker* displays a *modify line properties* figure (Figure 4-16) which allows the user to customize the color, line style, line width, marker type, and marker size of the selected plot curve. The color field is specified in RGB values in a MATLAB array (i.e. to specify the primary color, red, use [1 0 0]).

- In the Legend panel, clicking on any entry under the column labeled *Name* causes

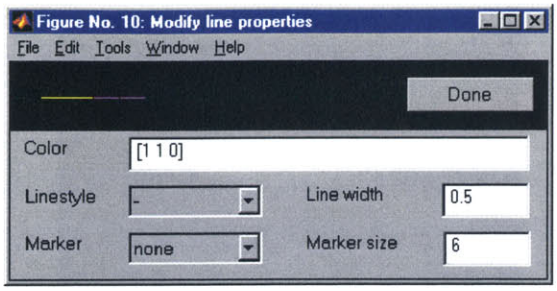


Figure 4-16: The Modify Line Properties figure.

the curve associated with that entry's variable to alternately display and hide. Hiding and displaying curves causes the axes to rescale automatically unless the auto-scaling function of the particular axis is disabled by means of the axis scale figure described above.

Loading, saving, and printing

The user can save the plots by clicking on the *Save* pushbutton. All lines, axis scale and annotations are saved. The *Open* pushbuttons loads a previously saved file. The plots can be printed using the *Print* button.

4.11 Step-by-step Guide To Setting Up A Customized Simulation

Launch the software from the *Softcell* menu. Once the application has started, two figures will be displayed: the *PAP Control* figure, and the *PAP Workspace* figure. All model parameters are started in their default values. To start setting up a simulation with customized parameter values, complete all or any of the following steps:

1. Change the membrane, axon, and bath parameter values: in the *PAP Workspace* figure, click on the axon schematic. The *PAP Parameters* figure is displayed. Adjust any parameter as described in Section 4.2. Clicking on the axon schematic again closes the *PAP Parameters* figure.

2. Change the stimulus-pulse parameter values: in the *PAP Workspace* figure, click on the stimulus-source icon (yellow filled-circle with an arrow inside). The *PAP Stimulus*

figure is displayed. Adjust any parameter as described in Section 4.4. Clicking on the stimulus-source icon again closes the *PAP Stimulus* figure.

3. Position the stimulus electrodes: in the *PAP Workspace* figure, select the appropriate radiobutton in the Electrode Positions panel (i.e. cathode/anode-electrode). To position the selected electrode longitudinally along the axon schematic, click on the black region between the top ruler and the schematic. The desired electrode moves to the position of the mouse cursor. Click on the electrode to toggle its position between intracellular and extracellular. Note that the icon of an intracellular electrode touches the axon schematic, whereas an extracellular electrode does not.

4. Position the voltage-recording electrodes: in the PAP Workspace figure, select the appropriate radiobutton in the Position Electrode panel. Click on the black region between the bottom ruler and the axon schematic to move the desired electrode to the position of the mouse cursor. Click on the electrode to toggle its position between intracellular and extracellular.

5. Change the numerics values: in the *PAP Control* figure, click on the *Numerics* option. The *PAP Numerics* figure is displayed. Adjust any parameter as described in Section 4.5.

6. Change the simulation duration: in the *PAP Control* figure, enter the simulation duration in milliseconds.

7. Start simulation: in the *PAP Control* figure, click on the pushbutton *Start*. Note that when a simulation has started, the *Start* pushbutton changes to a *Pause* pushbutton. Clicking on *Pause* pauses the simulation. Click on *Continue* to resume. Click on the *Stop* pushbutton anytime to abort.

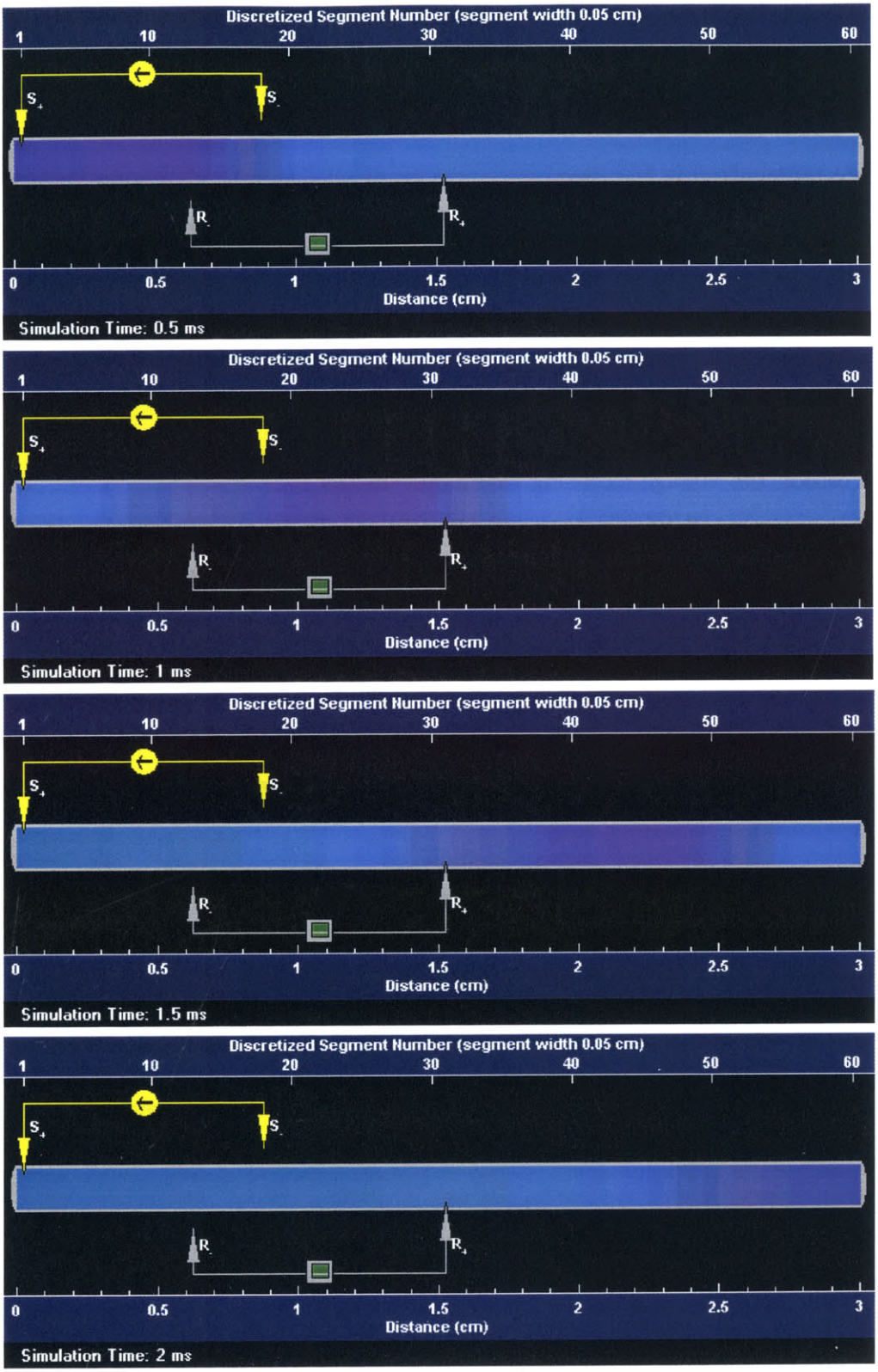


Figure 4-17: Snapshots of a propagating action potential shown using Color Space-Time Evolution in the *PAP Workspace* figure.

Chapter 5

Software Evaluation

5.1 Tests of the sofware

To assess the software, we performed four simulations. The object of these simulations was to determine whether the software produces results in accordance with those obtained previously.

- A basic test is to see how the conduction velocity of the action potential compares with previous studies. As can be seen in Table 5.1, the conduction velocity obtained with the software agrees closely with previous calculations which are within 12% of the measured value.

- Theoretically, when the internal and external resistances, r_i and r_o, are close to zero, the nerve fiber should behave like a small cell and all electrical variables should be independent of position. This condition can be achieved experimentally by placing a highly conductive longitudinal wire inside the nerve fiber and submerging the fiber in a highly conductive medium. Thus, when stimulated, no propagated action potential is seen. Instead, an action potential occurs simultaneously along the entire length of the axon. The software shows this property when the intracellular and extracellular resistances are reduced sufficiently (See Figure 5-1).

- When two action potentials are ellicited separated by a small time lag, the second

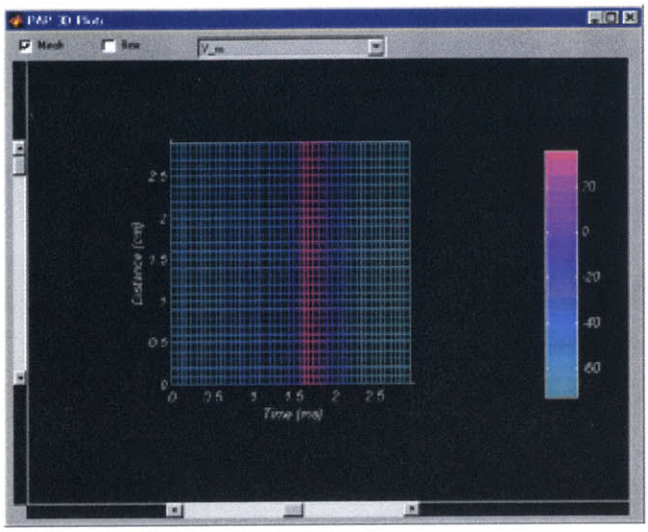


Figure 5-1: An action potential generated with r_o set to 0 Ohms/cm and r_i set to 0.56 Ohms/cm. The action potential is independent of space.

Source		Conduction velocity (m/s)
Hodgkin & Huxley (1950)	measured	21.2
Hodgkin & Huxley (1950)	calculated	18.8
Cooley & Dodge (1966)	calculated	18.7
This study	calculated	18.75

Table 5.1: The conduction velocity obtained in this study is based on a propagated action potential generated using the default values of the HH model and a super-threshold stimulus pulse of amplitude 0.05 mA and of duration 0.5 ms.

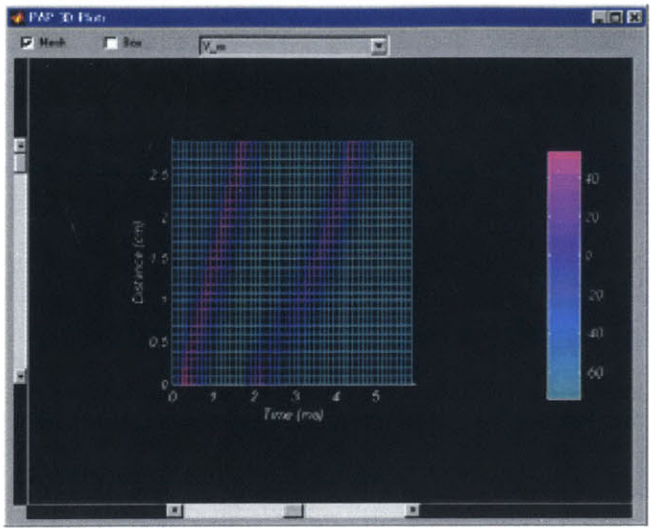


Figure 5-2: Two stimulus pulses of equal strength separated by 3.4 ms ellicited the above action potentials. Note that the second action potential travels at a slower speed.

action potential travels at a slightly slower speed than the first action potential (MacDonald, 1997). Figure 5-2 shows this property.

- Two action potentials colliding cancel each other. We can ellicit two action potentials at the same time using one stimulus source. In section 2.4, it is pointed out that both a negative stimulus electrode placed extracellularly and a positive electrode placed intracellularly depolarize the membrane and can produce action potentials. Thus, if we make the value of the external resistance, r_o, close to the value of the internal resistance, r_i, and if we set the electrodes as indicated, we can ellicit two concurrent propagated action potentials; one at each end of the axon model. When they collide near the mid-point of the axon schematic, the two cancel each other and disappear (See Figure 5-3).

These four examples indicate that the software is capable of producing physiologically relevant results.

5.2 Evaluation of the numerical methods

We look at the computation duration and memory usage of the solutions produced by each numerical method with varying Δt and Δz values.

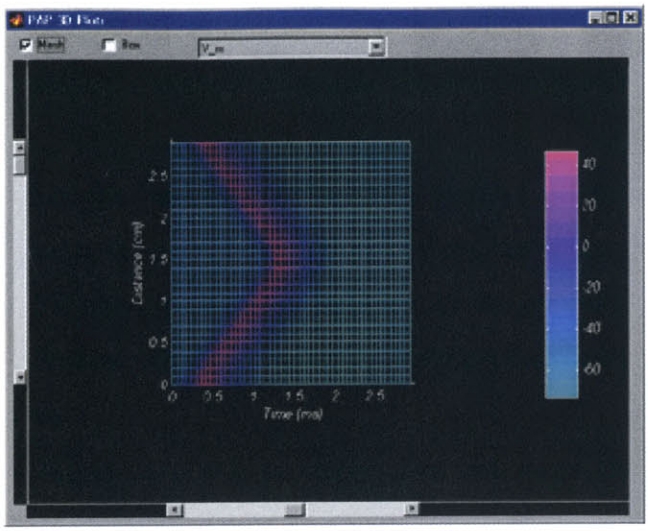


Figure 5-3: Two concurrent action potentials are generated at the ends of a three centimeters long axon model. The figure shows that when the action potentials collide, they cancel each other.

5.2.1 Stability

Not all numerical methods can produce solutions for certain Δt and Δz values. The forward Euler (FE), the Crank-Nicolson (CN), and the staggered Crank-Nicolson (SCN) method may produce a diverging solution if the Δt and Δz values are too large. In the software, if this divergence occurs too quickly and the solution reaches to a value of type NaN before the end of a simulation, the algorithm stops and generates no output. NaN stands for *not a number* in MATLAB and it's assigned to undefined values (i.e. 0/0).

The column labeled *divg* in Tables 5.2-5.4 is assigned a zero value when an algorithm finishes a simulation successfully and produces a solution output for a given Δt value. It is assigned a 1 when no output is generated because the solution has diverged to an infinitely large value (i.e. NaN).

Table 5.3 shows that the backward Euler (BE) method produces outputs for all Δt values between 0.005 and 0.450 ms for simulations of duration 40 ms and Δz set to 0.05 cm. Table 5.2 shows that FE only produces outputs for Δt values less than or equal to 0.003 ms. Likewise, Table 5.4 shows that CN and SCN both produce outputs for Δt values less than or equal to 0.05 ms.

The fact that a solution output is produced does not mean that the solution is not

Δt *(ms)*	*divg*
0.001	0
0.002	0
0.003	0
0.004	1
0.005	1

Table 5.2: Forward Euler produces solutions that diverge to NaN when Δt is greater or equal to 0.004 and Δz is set to 0.05.

diverging – maybe it is diverging at a slower rate. Nonetheless, the results in Tables 5.2-5.4 is a good indication of the relative stabilitly among the four methods. The FE method is the least stable because it requires a relatively small Δt (less than or equal to 0.003 ms) in order for it to produce a solution that does not diverge to NaN before the completion of a 40 ms long simulation. The BE method, on the other hand, is the most stable because it can produce a solution that does not diverge to NaN before the completion of a 40 ms long simulation for all the Δt values.

5.2.2 Computation Efficiency

We compare the time it takes to compute a solution as a measure of computation efficiency among the four numerical methods. We choose Δt values that would allow all numerical methods to produce an output solution for a simulation of 3 ms long and with Δz fixed to 0.05 cm. In the discussion on stability above, we stated that for the FE method to produce an output, a Δt value of less than or equal to 0.003 ms is required when Δz is 0.05 cm. Therefore, we chose to compare the time it takes to compute an output using each method for Δt values of 0.001, 0.002, and 0.003 ms. We use the MATLAB timing functions, *tic* and *toc*, to compute the time needed for the central processing unit (CPU) to complete a simulation. The time measured is in unit of seconds. The results are obtained using a PC with a clock speed of 266MHz.

The results in Figure 5-4 show that for each Δt, the FE and the SCN methods take the least time to compute an output among the four methods. These two methods take about the same time with FE slightly faster. For Δt equals to 0.001 ms, BE is about 2.7 times slower than FE and SCN. The CN method is about 5 times slower. These computation

Δt *(ms)*	*divg*
0.005	0
0.010	0
0.015	0
0.020	0
0.025	0
0.030	0
0.035	0
0.040	0
0.045	0
0.050	0
0.100	0
0.150	0
0.200	0
0.250	0
0.300	0
0.350	0
0.400	0
0.450	0
0.500	0
0.550	0
0.600	0

Table 5.3: Backward Euler produces solutions that do not diverge to NaN before the completion of a 40 ms long simulation for this range of Δt's.

Δt *(ms)*	*divg*
0.005	0
0.010	0
0.015	0
0.020	0
0.025	0
0.030	0
0.035	0
0.040	0
0.045	0
0.050	0
0.100	1
0.150	1
0.200	1
0.250	1
0.300	1
0.350	1
0.400	1
0.450	1

Table 5.4: The Crank-Nicolson and the SCN method both produce solutions that diverge to NaN before the completion of a 40 ms long simulation when Δt is greater or equal to 0.1 ms .

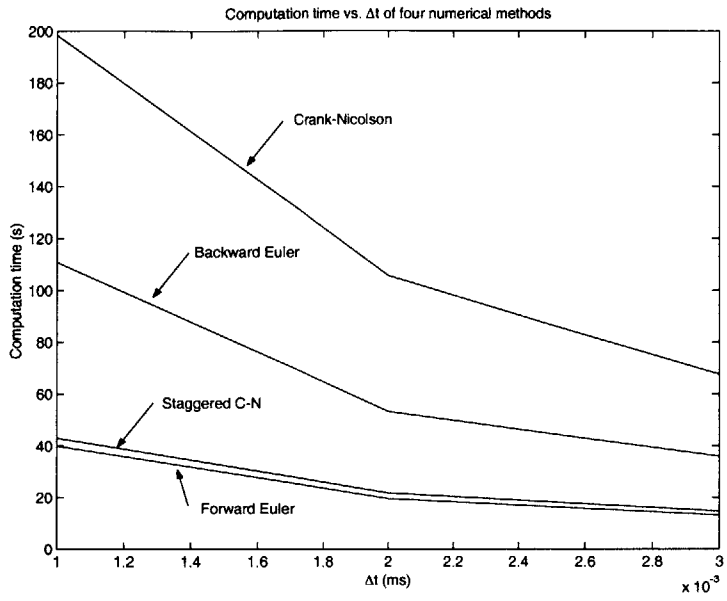


Figure 5-4: Computation times of four numerical methods for Δt values of 0.001, 0.002 and 0.003 ms. The forward Euler and the staggered C-N method are the fastest to produce an output solution.

times are based on simulations of duration 3 ms.

The results in Figure 5-5 again show that SCN is faster than BE and CN. For that range of Δt values, FE fails to produce an output solution. These computation times are based on simulations of 40 ms. The computation time for SCN is approximately proportional to $\frac{1}{\Delta t}$. For BE and CN, the computation time is approximately proportional to $\left(\frac{1}{\Delta t}\right)^{1.12}$, and $\left(\frac{1}{\Delta t}\right)^{1.28}$, respectively. Figure 5-6 shows the effect of increasing Δz on the computation time vs. Δt function and the effect of increasing Δt on the computation time vs. Δz function for the BE and SCN methods.

5.2.3 Memory Usage

The amount of memory required to obtain an output is measured in terms of the number of memory registers required for storing the output data points. The output consists of data points for four solution variables: the membrane potential, and the three activation and inactivation factors m, n, and h. Each data point requires one memory register, and the total number of data points is calculated as follows

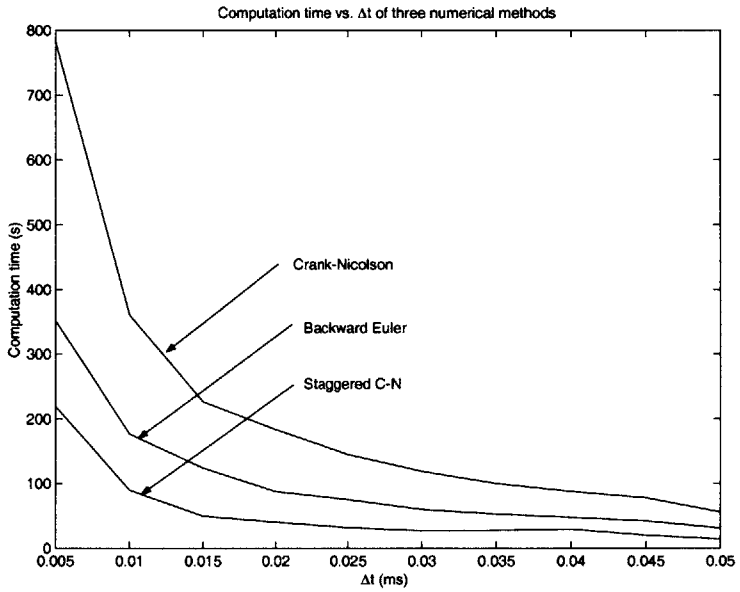


Figure 5-5: Computation times of three numerical methods for Δt ranging from 0.005 ms. to 0.05 ms. In this Δt range, the staggered C-N method is the fastest to produce an output solution, followed by backward Euler and Crank-Nicolson. Forward Euler fails to produce outputs in this range of Δt's.

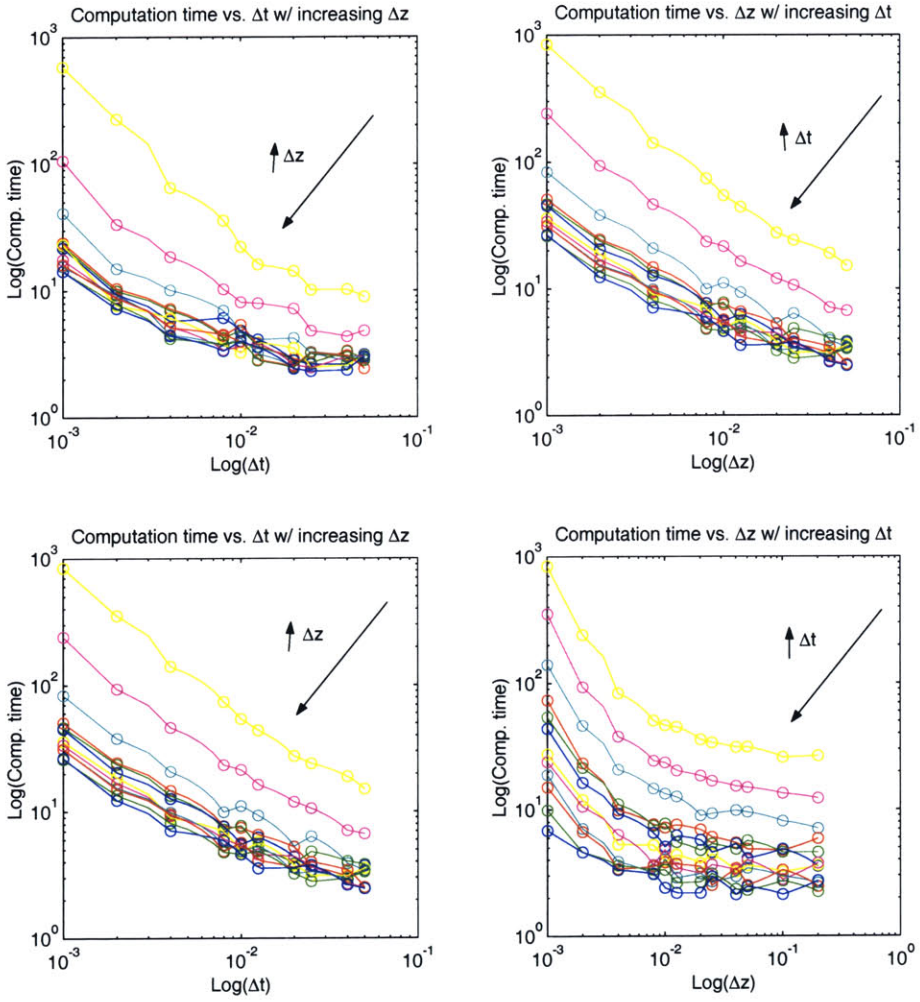


Figure 5-6: Top row: Computation times of staggered C-N with varying Δt and Δz values. Bottom row: Computation times of backward Euler with varying Δt and Δz values.

$$\left(\frac{Simulation\,duration}{\Delta t}\right) \times \left(\frac{Axon\,length}{\Delta z}\right) \tag{5.1}$$

For example, for a simulation of duration = 3 ms, Δt = 0.01 ms, axon length = 3 cm, and Δz = 0.05 cm, the number of data points contained in one solution variable is 18,000. Since there are four solution variables, the software would need to store a total of $4 \times 18,000 = 72,000$ data points; in other words, 72,000 memory registers would be needed.

Memory usage can be reduced by increasing either Δt or Δz, or decreasing either the simulation duration or the length of the axon model.

5.2.4 Accuracy

We use the root mean-squared error (RMSE) to measure the accuracy of an output solution. We first obtain an output solution that serves as the standard solution for all RMSE calculations. For this standard solution to approach to the real solution we use a small Δt and Δz so as to obtain the highest temporal and spatial resolution. We choose a $\Delta t = 0.001$ ms and a $\Delta z = 0.001$ cm. Smaller values are not practical because of long computation time. Also, the number of data points generated with smaller values of Δt and Δz requires more than 64 MB of memory which is usually found in today's personal computers.

Using the above Δt and Δz, we use the staggered Crank-Nicolson method to compute the solution of a 1 ms long simulation. The simulation consists of stimulating a 1 cm long axon at the first segment location with a rectangular pulse of amplitude 0.05 mA, a duration of 0.5 ms, and an onset at 0 ms. The output of this simulation serves as the standard solution to which all subsequent simulations with varying Δt and Δz are compared.

To compute the RMSE of an output, we use the following equation

$$RMSE = \sqrt{\frac{1}{n} \times \sum_i \sum_j (S_{ij} - T_{ij})^2}$$

where S_ij is the output of the standard solution at segment i and time j, T_ij is the output of a particular solution at segment i and time j, and n is the total number of values in the solution matrix. The RMSE for Δt and Δz between 0.001 and 0.05 using the staggered Crank-Nicolson and the backward Euler method is shown in Figure 5-7.

The results show that for both methods, the error in the solution increases as Δt in-

creases for a fixed Δz. Likewise, for a given Δt, the error increases as Δz increases. However, the error increment is higher for increasing Δz. The computed errors are higher than expected. The results clearly reflect flaws in our method of computing the RMSE. The following three points explains some of these flaws:

- The method used to convert the standard matrix solution to the size of a particular solution is a simple decimation algorithm which eliminates intermediate solution values to reduce the size of the matrix. A better method would be to average the intermediate solution values.

- As Δz changes, the location of the stimulus changes. This adds inconsistency to the point of excitation among the simulation results; thus, errors due to inconsistent electrode positioning are introduced as Δz is changed.

- The onset of an action potential may vary depending on Δz. Thus, it is necessary to compute the RMSE due to temporal shift, the RMSE due to spatial shift, and the RMSE due to the discrepancies in the amplitude of the potential at a particular time and space, in order to have a clear picture of the overall RMSE behavior as we change Δt and Δz.

The accuracy of the solutions as Δt and Δz change can be better assessed if the above points are addressed appropriately. Due to time constrain, the proper analysis to assess the accuracy of each method using different combinations of Δt and Δz is beyond the scope of this paper, but the software provides the means to do a proper analysis as suggested above.

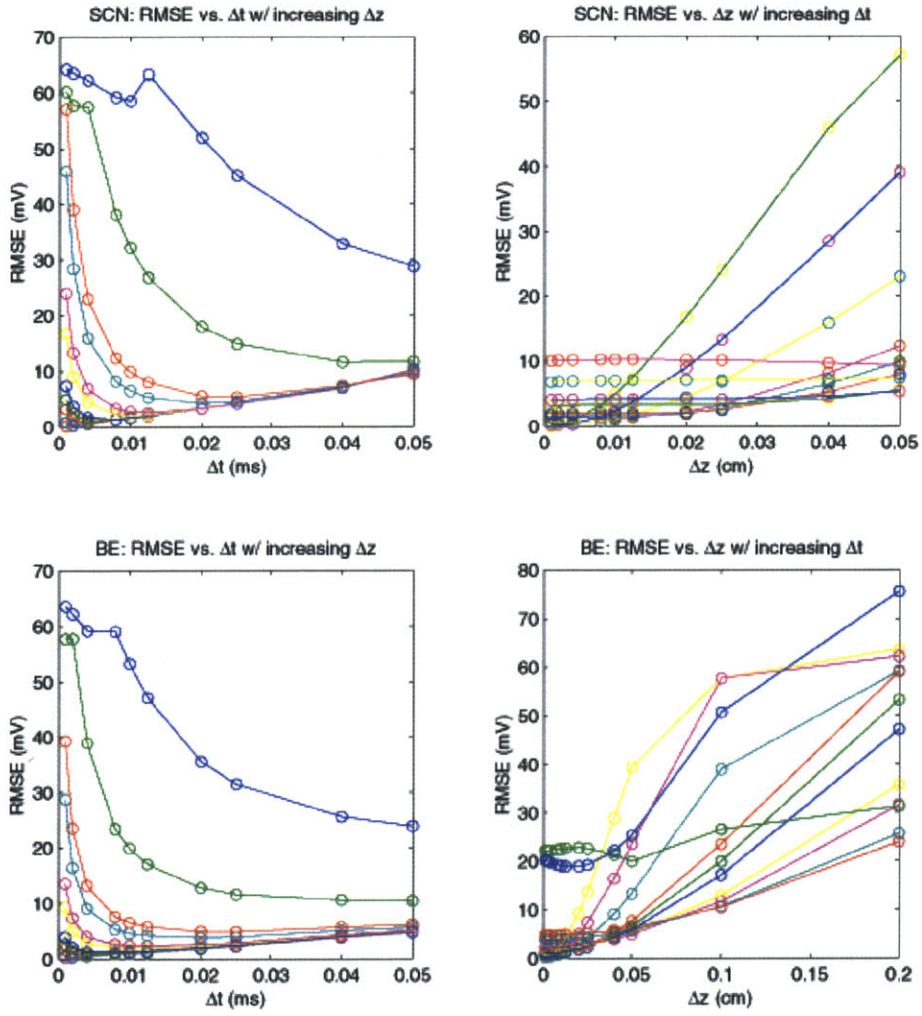


Figure 5-7: Top row: RMSE of staggered C-N with varying Δt and Δz values. Bottom row: RMSE of backward Euler with varying Δt and Δz values.

Chapter 6

Conclusion

The results of the analysis on the performance of the numerical methods presented in this paper is based on the default values of the HH PDE equation. Changing the time constants of the rate functions and other parameters that would affect the conduction velocity of the action potential may affect the performance of each method by a different degree. Thus, we can only conclude that the staggered C-N method has the best performance under the assumption that the model's parameters are close to their default values and the Δt used is within a certain range. The complexity of the non-linear behavior of the ionic components in the partial differential equation is beyond the scope of the analysis presented in this paper. Nonetheless, this complexity along with its implications about which numerical method may be best to use to solve a set of specific parameters can be explored in great depth with the software.

Appendix A

Sample Evaluation Script

The following is a sample MATLAB script used to generate most of the data tables presented in this paper.

```
sim_dur = 40;  %simulation duration (ms)
dt_range = [0.005:0.005:0.6]; %range of delta_t (ms)
algo = [1 2 3 4];  %numerical method: 1-feuler, 2-beuler, 3-cn, 4-scn
more off; feval('apcntrl','dur', sim_dur); fid =
fopen('c:\MATLABR11\work\dump.txt','w');
fprintf(fid,'Computation times for a simulation of
duration=%d ms\n\n',sim_dur);
for num=algo
   switch num
   case 1
      fprintf(fid,'Method: Forward Euler\n');
      feval('apnum','algo','Forward Euler');
   case 2
      fprintf(fid,'Method: Backward Euler\n');
      feval('apnum','algo','Backward Euler');
   case 3
      fprintf(fid,'Method: Crank-Nicolson\n');
      feval('apnum','algo','Crank-Nicolson');
   case 4
```

```
   fprintf(fid,'Method: Staggered C-N\n');
   feval('apnum','algo','Staggered C-N');
end

fprintf(fid,'delta_t (ms)\t comp.time (s)\t error\t
Vmax (mV)\t Vmin (mV)\n');

for dt_val=dt_range
   feval('apnum','dt',dt_val);
   tic;
   err_flag = feval('apcntrl','start');
   tm = toc;

   if err_flag
      fprintf(fid,'%.3f\t\t\t %.3f\t\t %d\t\t
      N/A\t\t N/A\n', [dt_val tm err_flag]);
   else
      vm = feval('apcntrl', 'get', 'V_{ m}', {':',':'});
      vmax = max(max(vm));
      vmin = min(min(vm));

      if vmax>300 & vmin>-300
         fprintf(fid,'%.3f\t\t\t %.3f\t\t %d\t\t >>300\t\t
         %.1f\n', [dt_val tm err_flag vmin]);
      elseif vmax>300 & vmin<-300
         fprintf(fid,'%.3f\t\t\t %.3f\t\t %d\t\t >>300\t\t
         <<-300\n', [dt_val tm err_flag]);
      elseif vmax<300 & vmin<-300
         fprintf(fid,'%.3f\t\t\t %.3f\t\t %d\t\t %.1f\t\t
         <<-300\n', [dt_val tm err_flag vmax]);
      elseif vmax<300 & vmin>-300
         fprintf(fid,'%.3f\t\t\t %.3f\t\t %d\t\t %.1f\t\t
```

```
            %.1f\n', [dt_val tm err_flag vmax vmin]);
         end
      end
      %pause;
   end
   fprintf(fid,'\n\n');
end fclose(fid); disp('Script done');
```

Appendix B

MATLAB code of the staggered Crank-Nicolson method

The following is a sample MATLAB code of one of the four numerical methods implemented for the software. This code pertains to the staggered C-N method.

```
% Staggered C-N Finite-difference method
if counter==2
   sprintf('staggered C-N')
   K = delta_t / (2*pi*a*(r_o+r_i)*Cm*delta_z^2);
   on_diag = (2+(2*K))*ones(z_steps,1);
   off_diag = -K*ones(z_steps-1,1);
   TM_h = diag(off_diag,1);
   TM_h = TM_h + diag(off_diag,-1);
   TM_h = TM_h + diag(on_diag,0);

   % Boundary condition -- use second-order von Neumann
   TM_h(1,2)=2*TM_h(2,1);
   TM_h(z_steps,z_steps-1)=2*TM_h(2,1);
   TM_h = inv(TM_h);
end
```

```
while counter<=t_steps
   t=counter;
   Vm = v(:,t-1);
   [am bm tm minf ah bh th hinf an bn tn ninf] =
   APabtinf(Vm, ab_params);

   %calculate m(t+1/2), n(t+1/2), h(t+1/2)
   if t==2
      m(:,t) = (0.5*delta_t*am) + (1-(0.5*delta_t
      .*(am+bm))).*m(:,t-1);
      n(:,t) = (0.5*delta_t*an) + (1-(0.5*delta_t.*(an+bn)))
      .*n(:,t-1);
      h(:,t) = (0.5*delta_t*ah) + (1-(0.5*delta_t.*(ah+bh)))
      .*h(:,t-1);
   else
      denom = (delta_t^(-1)+0.5.*(am + bm));
      indx = find(denom==0);
      denom(indx) = 1e-10;
      term1 = (am./denom);
      term2 = (delta_t^(-1)-0.5.*(am + bm))./denom;
      m(:,t) = term1 + term2.*m(:,t-1);

      denom = (delta_t^(-1)+0.5.*(an + bn));
      indx = find(denom==0);
      denom(indx) = 1e-10;
      term1 = (an./denom);
      term2 = (delta_t^(-1)-0.5.*(an + bn))./denom;
      n(:,t) = term1 + term2.*n(:,t-1);

      denom = (delta_t^(-1)+0.5.*(ah + bh));
      indx = find(denom==0);
      denom(indx) = 1e-10;
```

```
    term1 = (ah./denom);
    term2 = (delta_t^(-1)-0.5.*(ah + bh))./denom;
    h(:,t) = term1 + term2.*h(:,t-1);
end

GNa(:,t) = gNa.*m(:,t).^3.*h(:,t);
GK(:,t) = gK.*n(:,t).^4;

JNa = GNa(:,t).*(Vm-VNa);
JK  = GK(:,t).*(Vm-VK);
JL  = GL(:,t).*(Vm-VL);
Jion = JNa + JK + JL;
J = Jion - Jext(:,t);
V = (delta_t/Cm)*J;

%v(t + 1/2)
v(:,t) = TM_h*(2.*v(:,t-1) - V);

%v(t + 1) advanced from v(t + 1/2) explicitly
v(:,t) = 2.*v(:,t) - v(:,t-1);

%make dV/dt=0 at boundary
v(1,t)=v(2,t);
v(z_steps,t)=v(z_steps-1,t);

%update counter
counter = counter+1;

%update status
strg = sprintf('Calculating using Staggered C-N method...
 %.0f%% done', (t/t_steps)*100);
set(statusbar, 'string', strg);
```

```
   drawnow;

   %see if diverge
   if any(isnan(v(:,t)))
      set(statusbar,'string', 'Error: Solution diverged
      (Simulation aborted)');
      v=[]; n=[]; m=[]; h=[];
      %state='stop';
      err_flag=1;
      counter=2;
      return;
   end

   %poll to see if pause
   if strcmp(state,'pause')
      set(statusbar, 'string', 'Simulation Paused');
      return;
   end

   %poll to see if stop
   if strcmp(state,'stop')
      counter=2;
      set(statusbar, 'string', 'Ready (Simulation aborted)');
      return;
   end

end

%calculation completed
counter=2;
set(statusbar, 'string', sprintf('Ready (Last simulation
completed at %s)', datestr(now,14)));
```

Bibliography

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

Fitzhugh, R. and Antosiewicz, H. A. (1959). Automatic computation of nerve excitation – detailed corrections and additions. *J. Soc. Indust. Appl. Math.*, 7:447–458.

Frankenhaeuser, B. and Hodgkin, A. L. (1957). The action of calcium on the electrical properties of squid axons. *J. Physiol.*, 137:218–244.

Gerald, C. F. and Wheatley, P. O. (1989). *Applied Numerical Analysis.* Addison-Wesley, Reading, MA.

Hines, M. (1984). Efficient computation of branched nerve equations. *Int. J. Bio-Med. Comput.*, 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.*, 117:500–544.

Huxley, A. F. (1959). Ion movements during nerve activity. *Ann. N.Y. Acad. Sci.*, 81:221–246.

Joyner, R. W., Westerfield, M., Moore, J. W., and Stockbridge, N. (1978). A numerical method to model excitable cells. *Biophys. J.*, 22:155–170.

MacDonald, R. (1997). Velocity of propagated action potentials in the hodgkin-huxley model is not constant. Technical report, MIT.

Mascagni, M. V. and Sherman, A. S. (1998). Numerical methods and neuronal modeling. In Koch, C. and Segev, I., editors, *Methods in Neuronal Modeling*, pages 569–606. MIT Press, Cambridge, MA.

Moore, J. W. and Ramon, F. (1974). On numerical integration of the hodgkin and huxley equations of a membrane action potential. *J. theor. Biol.*, 45:249–273.

Moore, J. W., Ramon, F., and Joyner, R. W. (1975). Axon voltage-clamp simulations. *Biophys. J.*, pages 11–24.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1986). *Numerical Recipes. The Art of Scientific Computing.* Cambridge University Press, Cambridge, Great Britain.

Smith, G. D. (1985). *Numerical Solution of Partial Differential Equations: Finite Difference Methods.* Clarendon Press, Oxford, Great Britain.

Weiss, T. F. (1996). *Cellular Biophysics. Volume 2: Electrical Properties.* MIT Press, Cambridge, MA.