BioSystems 64 (2002) 73–96

www.elsevier.com/locate/biosystems

Real-time computer simulations of excitable media: JAVA as a scientific language and as a wrapper for C and FORTRAN programs

Flavio H. Fenton [a,b,*], Elizabeth M. Cherry [a], Harold M. Hastings [a], Steven J. Evans [b]

[a] *Department of Physics, Hofstra University, Hempstead, NY 11549, USA*
[b] *The Heart Institute, Beth Israel Medical Center, New York, NY 10003, USA*

Received 4 August 2001; accepted 1 September 2001

Abstract

We describe a useful setting for interactive, real-time study of mathematical models of cardiac electrical activity, using implicit and explicit integration schemes implemented in JAVA. These programs are intended as a teaching aid for the study and understanding of general excitable media. Particularly for cardiac cell models and the ionic currents underlying their basic electrical dynamics. Within the programs, excitable media properties such as thresholds and refractoriness and their dependence on parameter values can be analyzed. In addition, the cardiac model applets allow the study of reentrant tachyarrhythmias using premature stimuli and conduction blocks to induce or to terminate reentrant waves of electrical activation in one and two dimensions. The role of some physiological parameters in the transition from tachycardia to fibrillation also can be analyzed by varying the maximum conductances of ion channels associated with a given model in real time during the simulations. These applets are available for download at http://arrhythmia.hofstra.edu or its mirror site http://stardec.ascc.neu.edu/ ~ fenton. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Computer simulations; JAVA; Programs

1. Introduction

Mathematical models of biological and complex systems demonstrate many phenomena found in nature. In particular, models of cardiac electrical activity in cells can be used to illustrate some of the mechanisms underlying cardiac arrhythmias. The continuous increase in computer power over the years has made it possible to solve many of these models using desktop computers. However, systematic study of these models is greatly enhanced by using a graphical user interface (GUI) for variation of parameters, visualization, and the analysis of results.

* Corresponding author. Tel.: + 1-516-463-5586; fax: + 1-516-463-3059.

E-mail addresses: fenton@presto.physics.neu.edu (F.H. Fenton), harold.hastings@hofstra.edu (H.M. Hastings).

The JAVA language is a natural choice for interface programming for several reasons. First, it is straightforward to register on-screen user interactions, such as mouse events, alphanumeric inputs, and click-choice events (e.g. buttons and checkboxes). In addition, it is easy to change parameters in real time and to plot in one, two and three dimensions, allowing easy visualization of results. Furthermore, since executables are platform-independent, programs can be run in different machines and operating systems as well as through the internet. However, using JAVA as a programming language can have some drawbacks. In some cases, unless substantial optimization is performed on a specific algorithm, computational speed can increase unacceptably. In other cases, a well-optimized program may already exist in other language, or the program may use standard subroutines that cannot easily be imported into JAVA. In addition, the program may need to perform external system commands, which JAVA does not allow for security reasons. However, these problems can be overcome by using the JAVA Native Interface (JNI) options, which allows JAVA to interact with applications and libraries written in other (native) languages.

We present in this paper a set of JAVA applets intended for the study of excitable media—in particular, models that describe cardiac electrical activity in single cells, 1- and 2-D tissues. In Section 2, we briefly describe methods and algorithms used in writing the JAVA applets. Section 3 gives an overview of excitable media and provides the basic equations the applets are designed to solve. In Section 4, we discuss several representative applets for single cells, 1-D cables, and 2-D tissues. In Section 5, we present conclusions and discuss future work. In addition, we provide an appendix (Appendix A) with a general example showing how to call C and FORTRAN subroutines from JAVA, which is useful when JAVA is used only as a GUI wrapper.

Other groups have developed computational systems for cardiac modeling and made them available publicly (Barry, 1998; Henriquez et al., 2000; Li, 2000; Joe, 1997; CellML, 2000; NRCAM, 2001). Barry's software requires a fee; other software and our software are available for free for non-commercial purposes. EASIWAV from Henriquez's group simulates cardiac wave propagation over a 2D domain and features nonuniform anisotropy, irregular geometries, multiple membrane models and simulation modules, all with an intuitive GUI interface that runs under UNIX. Li provides a program for modeling the action potential (AP) of a canine ventricular cell. Li's software runs on the net under X-WINDOWS. Joe provides a web-based simulation (using JAVA and JAVASCRIPT) of the heart's specialized conduction system. CELLML is an XML-based language intended to store and exchange computer-based biological models. NRCAM offers a similar approach in their 'virtual cell', a general computational framework for modeling biological processes, at the cellular level.

In contrast, the present programs, and related material on our web site supplementary to this paper, are designed especially for educational use and exploratory research. The advantage of these programs is that they cover a wide range of models and geometries with an easy-to-use interface for simulating complex cardiac electrical dynamics. They are also readily available on the Internet and run under any operating system. In this manuscript we provide a description of both the use of JAVA together with traditional scientific languages and the models themselves.

This paper is dedicated to the memory of Michael Conrad. Professor Conrad worked at the interface between biology and computation, and was in many ways well ahead of the recent thrust in computational biology. His work on computational models of the brain led to many more recent developments. The journal *BioSystems* also reflects his philosophy and guidance as Editor, with its emphasis on theory well grounded in experiment. Finally, Professor Conrad considered teaching an essential part of his career and left a legacy of Ph.D. students. One of us (H.M. Hastings) had the pleasure of knowing Professor Conrad as a colleague and writing two papers with him (Hastings and Conrad, 1979; Conrad and Hastings, 1985). Our present paper on computational models of cardiac dynamics and other excitable media, and its goal of promoting the computer as an experimental and educational tool, follows his tradition. More generally, the continued growth of computational approaches to biology recognizes Professor Conrad's vision and leadership.

2. Systems, methods, and algorithms

The implementation of the programs consists of two major parts, the graphical interface that inputs and outputs data; and the integration of the differential equations of a given mathematical cell model. In the graphical interface, the input of events is handled using the standard methods from the java.awt.*, java.awt.event.* and Java.lang.* classes (SUN, 1997). Basic plotting of graphs and images is relatively simple and many useful examples can be found at (SUN, 1997). However, when a large number of pixels need to be frequently drawn, there are some advanced procedures that are worth mentioning. The simplest way to draw graphs and small images is to use the fillLine($x1$,$y1$,$x2$,$y2$) method, which can create pixels when given identical starting and ending coordinates ($x1 = x2$ and $y1 = y2$) and in general is faster than fillRect(). When generating large images that require a significant set of pixels, this method becomes extremely slow, and the java.awt.image.MemoryImageSource class is a better choice. This class is an implementation of the ImageProducer interface, which uses an array to produce pixel values and can generate images at high speeds, so little time is spent in the plotting procedures. In addition, flickering effects can be avoided by using the createImage() method as a buffer (SUN, 1997).

For the numerical integration component, all applets discussed in this paper except for the rabbit ventricular slice are integrated using time-stepping implicit schemes that are unconditionally stable and have no other restriction on the size of the time step Δt other than accuracy. For the single cell and the 1D applets an implicit Euler scheme is used. For the 2D square tissue applet we used an alternating direction implicit (ADI) scheme (Press et al., 1992). For the 2D rabbit ventricular slice applet we used a phase-field method (Fenton et al., 2001) to handle the irregular boundary conditions combined with an explicit Euler scheme.

All the programs presented in this article were written in JAVA (version 1.1.7B-2) on a Compaq Alpha workstation. The programs have been compiled and run successfully on PCs under WINDOWS using Sun-JAVA 1.3.0 and on PCs under LINUX using IBM-JAVA 1.1.8 and/or kaffe 1.0.6.

3. Models

The mathematical models we implement in our JAVA applets all represent excitable media, a class of nonlinear complex systems of which cardiac electrical dynamics is an example. Excitable media can be defined as systems composed of elementary segments or cells, each of which possesses the following properties:

1. a well-defined rest state,
2. a threshold for excitation, and
3. a diffusive-type coupling to its nearest neighbors.

By a threshold for excitation, we mean that any external stimulus applied to a cell that keeps that cell below this threshold produces a qualitatively different result than a stimulus that raises the cell above threshold. Stimuli below the threshold are damped out and produce no persistent change in the system, which simply returns to the rest state. However, stimuli above the threshold induce the cell to change from its rest state to an excited state. This change produces a pulse in time whose shape and nature are determined by the nonlinear properties of the medium and do not depend on the form of the external excitation. In spatially-extended excitable systems, the excitation pulse may be carried over from one elementary segment to the next by means of a diffusive coupling. The excitability of the tissue and the coupling strength between neighbors determines the minimum size of tissue required for a pulse to expand and propagate as a traveling wave front. The waves in excitable media have the property of propagating without damping.

Cardiac and nerve cells can be considered as elementary segments of excitable media because they exhibit these three characteristics, as we now describe:

1. *Well-defined rest state*. The cell membrane is a bilayer structure composed of molecules called phospholipids, with a polar (charged) side and a nonpolar fatty side. These molecules not only function as a barrier to retain vital cell components but also provide an electrical insulation between the intracellular ionic concentrations and the extracellular ones. Since ions generally are not very soluble in hydrophobic environments such as within the cell membrane, they normally cannot cross the membrane. The membrane thus maintains a concentration gradient and consequently a net membrane potential. That is, the cells have a rest state, which for cardiac muscle cells generally is between -90 and -80 mV, and about -60 mV for nerve cells.
2. *Threshold for excitation*. Cardiac and nerve cells can synthesize linear polymers of amino acids (proteins) that interact with the membrane. Some of these can span the entire membrane to form aqueous pores, which connect the intra- and extracellular media and allow the flow of ions across the membrane. These ion channels have two important properties with respect to ion flux, their ability to be either open or closed and a selective permeability. The opening and closing of the ionic channels is produced by different protein configurations. The probability of a channel being open or closed in general is voltage-dependent, and in many mathematical models the stochastic nature of the channels is replaced by a voltage-dependent probability function that describes what fraction of the total number of channels is likely to be open. The selective permeability allows the flux of only certain kinds of ions through a given type of channel. Many (but not all) ionic channels are closed when the membrane is at the resting potential, but a sufficient change in voltage (the threshold) causes the channels to open. With the channels open, a greater flux of ions is allowed, producing a greater change in voltage and the generation of a superthreshold response, known as an action potential (AP). The ion channels eventually inactivate and the membrane potential returns to the rest state.
3. APs propagate from cell to cell in the heart mostly through connections called gap junctions through which the local internal electrical current flows. The propagation of electrical impulses in one dimension can be modeled in a simple manner by assuming cardiac cells to be cylinders that connect to one another, forming a continuous long cylindrical fiber with internal cytoplasm of uniform resistivity r_a per unit length. The validity of this approach has been supported experimentally (Chapman and Fry, 1978). A further simplification can be obtained by considering the extracellular fluid to be grounded (for the more general case 'bidomain' see Henriquez and Papazoglou (1996)). Then the propagation of an impulse along a one-dimensional cylindrical cable depends on the flow of electric current along the fiber from active to resting regions. As the axial current I_a flows along the fiber, some of it leaks across the surface membrane as membrane current I_m, as shown in Fig. 1.

The amount of membrane current in any region, therefore, must be of equal magnitude and opposite direction to the change in axial current across the regions (charge conservation), thus:

$$I_m 2\pi r l = [I_a(x+l) - I_a(x)]\pi r^2 \approx -\left(\frac{\partial i_a}{\partial x}\right)\pi l r^2 \quad (1)$$

Since we consider that the intracellular medium has a resistivity ρ, the flow of current along the cable is proportional to the voltage gradient by Ohm's law, hence:

$$\left(\frac{\partial V_m}{\partial x}\right) = -\rho i_a \quad (2)$$

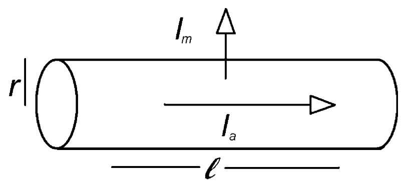


Fig. 1. Cable model composed by multiple cylindrical elements of radius r and length l. The axial current is I_a and the current crossing the membrane is I_m. Any change in I_a is then reflected in the flow of I_m.

The total membrane current per unit length, I_m, includes the currents from the flux of ions through the ion channels plus a capacitive current, I_c, due to the dielectric constant of the cell membrane layer. This defines the membrane current as:

$$I_m = I_c + I_{ion} = C_m\left(\frac{\partial V_m}{\partial t}\right) + I_{ion} \tag{3}$$

By combining Eqs. (1)–(3) we obtain the equation used to describe voltage propagation along a 1D cable:

$$\left(\frac{\partial V_m}{\partial t}\right) = r\left(\frac{\partial^2 V_m/\partial x^2}{2\rho C_m}\right) - \frac{I_{ion}}{C_m} = D\left(\frac{\partial^2 V_m}{\partial x^2}\right) - \frac{I_{ion}}{C_m} \tag{4}$$

Typically C_m is assigned the value 1 μF/cm^2, $\rho = 0.2$ kΩ cm and $2/r$ is the surface-to-volume ratio S_v. The value for the diffusion coefficient D most commonly used in the literature is 0.001 cm^2/ms, which assumes a S_v of 5000/cm (however, this corresponds to a relatively small cell radius of about 4 μm). Cardiac tissue and other excitable media, such as the Belousov–Zhabotinsky chemical reaction (Belmonte et al., 1997), aggregations of *Dictyostelium amoebae* (Loomis, 1982), and cortical neural preparations (Shibata and Bures, 1974), can exhibit many different patterns, including plane and circular (or elliptical) waves in two and three-dimensions. However a more interesting pattern, a spiral wave, can be induced in excitable media when a propagating wave is broken. The site of a wave break forces the wave front and wave back to meet at one point which will have an undefined phase. These phase singularities have zero normal velocity and serve as pivot points about which wave rotates, causing broken waves to evolve into spirals. Spiral waves are of particular interest in cardiology because they can induce continuous electrical activations and consequently mechanical contractions at a much more rapid rate than usual (Gray et al., 1998), resulting in a mechanism for serious arrhythmias, such as tachycardia and fibrillation (Winfree, 1998 and references therein).

4. Applets

The JAVA applets for excitable media, cardiac and nerve cell models presented here, are useful to explain the increasingly complex dynamics arising with the addition of spatial dimensions from single cells to one and two dimensions. The mathematical models implemented use Eq. (4) as their basis and differ only in their definitions of the ionic current sum I_{ion}.

For each applet, we describe some of the experiments that can be performed to help illustrate how the model works.

4.1. Single cell applets

Several different models with different levels of complexity exist to describe cellular dynamics. In the following sections we discuss four of these cell models, the Hodgkin and Huxley model for nerve cells; the FitzHugh–Nagumo model for general excitable media; and the Beeler–Reuter and Luo–Rudy 1 models for cardiac cells. Other cell models available for download from the web site but not described in this paper include the Noble (1962), Karma (1993) and 3v (Fenton and Karma, 1998) models.

4.1.1. The Hodgkin and Huxley (HH) model applet

In the 1950s, Hodgkin and Huxley (1952) introduced the first continuum mathematical model designed to reproduce the membrane AP. In their model, they assumed that when the membrane potential was not equal to the equilibrium potential, there would be a net flow of ions proportional to the difference between the membrane potential and the equilibrium potential. Thus for an ion species labeled y with a corresponding current i_y, Ohm's law gave the equation $i_y = (V_m - V_y)/r_y = g_y(V_m - V_y)$, where V_y is the equilibrium potential, r_y is the resistance to the flow of ions, which can be expressed in terms of a membrane conductance g_y.

The ionic conductance is, in general, a nonlinear function of the membrane voltage and can be represented by several channels. In cells, the ion channels are either open or closed, and in the case of voltage gating, the percentage of channels that are open out of the total number of channels in the cell is a function of the membrane voltage. Therefore, the probability function y (known as a gate variable) can be constructed to define what fraction of the channels are open as a function of voltage and time. Since gate variables range between 0 (all ion channels closed) and 1 (all ion channels open), the total cell or tissue conductance g_{ion} due to a given ion is given by $g_{\text{ion}} = gy(t, V)$, where g is the maximum possible conductance obtained when all channels are open.

Hodgkin and Huxley found that squid axon sodium and potassium conductances, obtained from voltage-clamp techniques, could be reproduced by using gate variables obeying simple first order equations of the form:

$$\frac{dy}{dt} = \alpha_y(V)(1-y) - \beta_y(V)y \tag{5}$$

where $\alpha_y(V)$ and $\beta_y(V)$ can be complex functions of the voltage. These equations describe how the gates open and close at different time rates and as a function of voltage, thereby controlling the kinetics of ionic flow through the channels.

Hodgkin and Huxley reproduced the squid nerve AP by using three different currents: a potassium current I_K, a sodium current I_{Na}, and a leak current later identified as chlorine I_{Cl}. These three currents, whose sum is used in Eq. (4) to define I_{ion}, are formulated as follows:

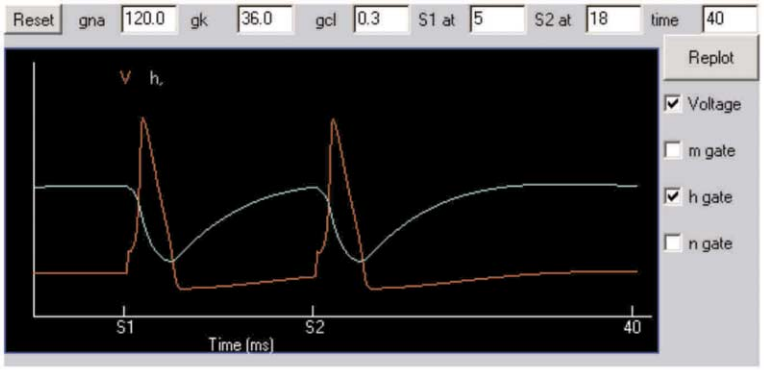
$$\begin{aligned} I_K &= n^4 g_K(V - V_K) \\ I_{Na} &= m^3 h g_{Na}(V - V_{Na}) \\ I_{Cl} &= g_{Cl}(V - V_{Cl}) \end{aligned} \tag{6}$$

where h, m, and n are gate variables. The opening and closing of the potassium ion channels is given by the activation gate n, while the sodium channels are governed by the activation gate m and the inactivation gate h. In this model the sodium activation gate m operates on a time scale several orders of magnitude faster than the other gates. For further information about the model and functions $\alpha_y(V)$ and $\beta_y(V)$ we refer to the original article (Hodgkin and Huxley, 1952).

Fig. 2 shows the Hodgkin–Huxley nerve model applet. The program solves the equations and can plot all four variables of the model simultaneously. The time of integration is initially set to 40 ms but can be varied easily by changing the time value. Other parameters that can be changed are the time at which above-threshold external stimuli S1 and S2 are applied and the maximum conductance of each of the three currents. The gate variables range between 0 and 1 as described above, but they are rescaled in the plot to assist in concurrent visualization of gates and voltage.

When the program is run with the initial settings, it plots two activations produced by the two applied external stimuli, the first of which occurs after 5 ms and the second after 18 ms. When a stimulus is applied in the simulation, the voltage changes from its resting value of -60 to -45 mV. At this voltage the sodium activation gate m quickly opens, allowing the influx of sodium into the cell and thereby depolarizing it. As the membrane potential depolarizes, the slower sodium inactivation gate h closes, and the sodium current terminates (see Eq. (6), where the total current is proportional to the product m^3h). By the time the sodium current has stopped, the much slower n gate has opened completely, generating an efflux of potassium ions that brings the cell membrane back to the rest state. The time course of the opening and closing of the gates as well as their relative speed can be seen when plotted along with the AP.

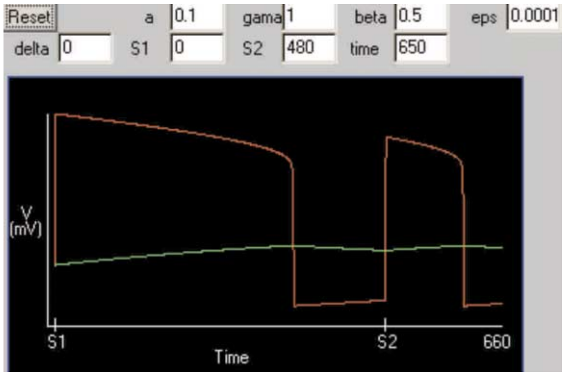
The timing for the stimuli S1 and S2 can be varied to observe how the AP changes. In particular, smaller intervals between S1 and S2 can yield interesting results if the second activation occurs before all the gate variables have time to recover completely. For example, by keeping S1 at 5 ms and changing the timing of S2 to 15 ms, the AP produced is a smaller one. If S2 is set to occur after 14 ms or earlier, no activation is produced. The explanation is simple, if a stimulus is induced very soon after an activation and the ionic gates have not had time to recover completely to their rest states, the application of a stimulus generates much less current than previously, resulting in either a smaller activation or no activation at all. This last effect is a property called refractoriness and is found in many excitable media.



gna 120.0 gk 36.0 gcl 0.3 S1 at 5 S2 at 18 time 40

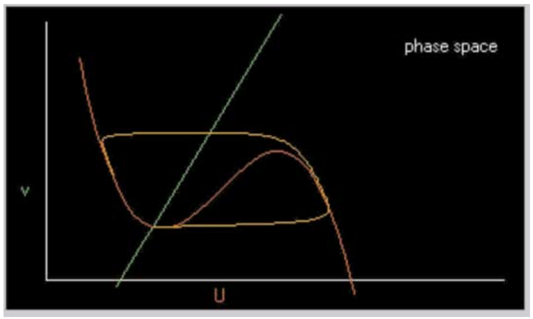
✓ Voltage
☐ m gate
✓ h gate
☐ n gate

Fig. 2

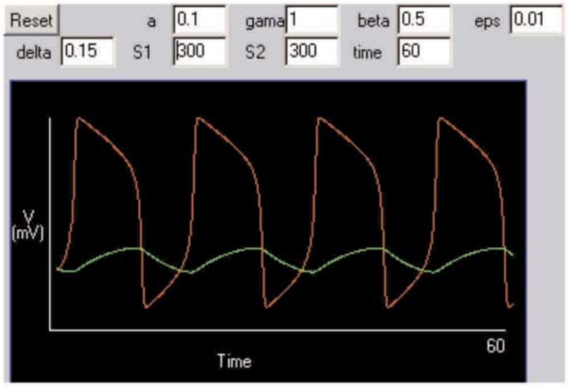


a 0.1 gama 1 beta 0.5 eps 0.0001
delta 0 S1 0 S2 480 time 650

(A)



(B)



a 0.1 gama 1 beta 0.5 eps 0.01
delta 0.15 S1 300 S2 300 time 60

(C)

Fig. 3

Fig. 2. H–H applet showing two consecutive activations produced by two external stimuli (S1 and S2) above threshold. Red corresponds to the membrane AP voltage and blue to the sodium gate h, which closes as the voltage increases and re-opens as the voltage decreases back to the resting potential.

Fig. 3. (a) Effect of excitability on the APD and the restitution relation in the FHN model. The fast variable U is shown in red and the slow variable v is shown in green. The large difference in time scale between the U and V process ($\varepsilon = 0.0001$) lengthens the plateau phase of the activation. Notice that the second activation is smaller in amplitude and much shorter in duration than the first, illustrating the effect of restitution. (b) Phase-plane analysis of the FHN model for 7 time units using the original settings. The almost complete (U,v) trajectory is shown in orange, and the U and v nullclines are shown in red and green, respectively. (c) Auto-oscillatory behavior in the FHN model induced by setting δ to 0.15. Further discussion is given in the text.

With the Hodgkin–Huxley applet, we also can observe how the conductances affect the currents and consequently the AP. For example, decreasing the sodium conductance from 120 to 70 mmho/cm^2 affects the time of rise the sodium current. If the sodium conductance is further decreased to 40 mmho/cm^2, it is no longer possible for the stimuli S1 and S2 to produce APs. We can conclude that a minimum sodium conductance is necessary to produce an activation. On the other hand, increasing the conductance too much can make the system auto-oscillatory. For example, if a single stimulus is applied, either by making S2 larger than the integration time or by setting it equal to S1, we can observe how the resting membrane potential is affected when g_{Na} is increased. At $g_{Na} = 180$ mmho/cm^2 we obtain a faster AP rise, but the resting membrane potential also is increased. When $g_{Na} = 188$ mmho/cm^2, the resting membrane potential is elevated above the threshold and the system becomes auto-oscillatory. In this regime, the larger g_{Na}, the faster the oscillation frequency.

The chlorine conductance helps to bring the membrane potential back to the rest state following an activation. For example, in the previous case where $g_{Na} = 188$ mmho/cm^2 and $g_K = 36$ mmho/cm^2, lowering g_{Cl} from 0.3 to 0.2 mmho/cm^2 suppresses the oscillations. A small negative g_{Cl} such as -0.01 mmho/cm^2 hyperpolarizes the cell by making it more negative during recovery.

4.1.2. The FitzHugh–Nagumo (FHN) model applet

The FitzHugh–Nagumo model (FitzHugh, 1961) is a generic model for excitable media and can be applied to a variety of systems. FitzHugh called his simplified model the Bon Hoeffer–van der Pol model and derived it in the 1960s as a simplification of the Hodgkin–Huxley equations. The model adiabatically eliminates the h and m gates and retains only a slow variable similar to n, denoted here as v. Due to its simple two-variable form and generality, it has been used widely. The model is able to reproduce many qualitative characteristics of electrical impulses along nerve and cardiac fibers, such as the existence of an excitation threshold, relative and absolute refractory periods, and the generation of pulse trains under the action of external currents. We implement the model as described by the following equations:

$$\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2} + (a - U)(U - 1) - v,$$
$$\frac{\partial v}{\partial t} = \varepsilon(\beta U - \gamma v - \delta) \qquad (7)$$

In the model, a represents the threshold for excitation, ε represents the excitability and β, γ and δ are parameters that can change the rest state and dynamics. Since this is a generic model, we keep time in arbitrary units for simplicity.

As in the HH applet, the FHN applet shows the activation produced by two external stimuli. Increasing a from its initial value of 0.1 makes it more difficult for the external stimulus to produce an excitation until at $a = 0.5$, no activation is produced. The reason is that the applet uses 0.5 as the magnitude of the external stimuli. If we instead decrease a to a value such as -0.1, the resting potential becomes unstable. Under these conditions, the system will remain at the resting potential if not perturbed, as can be seen by setting S1 and S2 to times larger than the integration time. However, if an external stimulus is applied by setting S1 to 1, the system will exhibit auto-oscillatory behavior, similar to that seen in the HH model for large values of g_{Na}.

The parameter ε is responsible of the different time scale dynamics between the U and v processes and is some times referred as the abruptness of excitation. In the model, the smaller the value of ε the faster will be the AP rate of rise and the longer the plateau. Since the rate of rise is directly related to the excitability of the system, in this model decreasing ε increases the excitability and the AP duration. These effects can be observed in the applet by setting both S1 and S2 to 1 and by varying ε from 0.01 to 0.0001 and increasing the integration time as needed to see the full activation.

The FHN applet also can demonstrate how the interval of time passing between two APs can affect the second AP. For instance, set ε to 0.0001, S1 to 0 and integration time to 700. Then alter the timing of S2. Notice that a second activation is not produce for S2 below the value 464. When S2 = 464, an activation finally becomes possible, but the duration of the activation is less than half of the previous activation (see Fig. 3a). By increasing time to 2000 and by continuing to increase S2, it can be seen that S2 must be at least 1500 in order for the system to recover its original properties and to produce a second AP as long as the first. The relationship between the duration of an action potential (APD) and the amount of time between the previous activation and the second stimulus (diastolic interval or DI) is known as restitution and is an important characteristic of cardiac tissue. When the heart rate increases, such as during exercise, the lengths of cellular signals that initiate muscular contractions are shortened in a similar way to ensure that filling of the heart chambers and ejection of blood occur efficiently. Without such adaptation, ventricles would not be filled before contracting during faster heart rates. The function that relates APD to DI is known as the restitution curve, and its importance in cardiac dynamics is discussed further in later subsections.

Since the FHN model consist of only two variables, the qualitative properties of the model can be explored by a phase-plane analysis. Therefore, the applet also includes a phase-space plot of the variables U and v, illustrated in Fig. 3b, in which the computed time series $U(t)$ and $v(t)$ are plotted as ordered pairs (U, v) as a function of time. The phase-space view shows how the system starts at a fixed point and, if excited above threshold and allowed to return to rest, follows a closed loop back to the fixed point. The system's nullclines are plotted as well to further explain the dynamics. The nullclines are found by setting $\partial U/\partial t = 0$ and $\partial v/\partial t = 0$ in Eq. (7) and solving both equations in the form $v = f(u)$, resulting for this system in a straight line corresponding to the v nullcline (green) and a cubic curve for the U nullcline (red). The system's fixed point is located at the intersection of the nullclines. Raising the system above threshold corresponds to increasing U beyond the central portion of the cubic nullcline, so that it is in the area under the local maximum. As U and v evolve, they roughly follow the course of the nullclines back to the resting state. Since ε is the time scale difference between the U and v processes, decreasing ε causes the solution to follow more closely the cubic nullcline, since that process is faster. Compare, for example, the phase trajectory produced by using $\varepsilon = 0.01$ (shown in Fig. 3b) and the one produced by using $\varepsilon = 0.1$. The evolution of the phase space trajectory in time can be observed by changing the integration time. Extending the integration time from 1 to 20 in. increments of 1 allows visualization of the different speeds involved in the phase space process. For reference, the time used in Fig. 3b is 7.

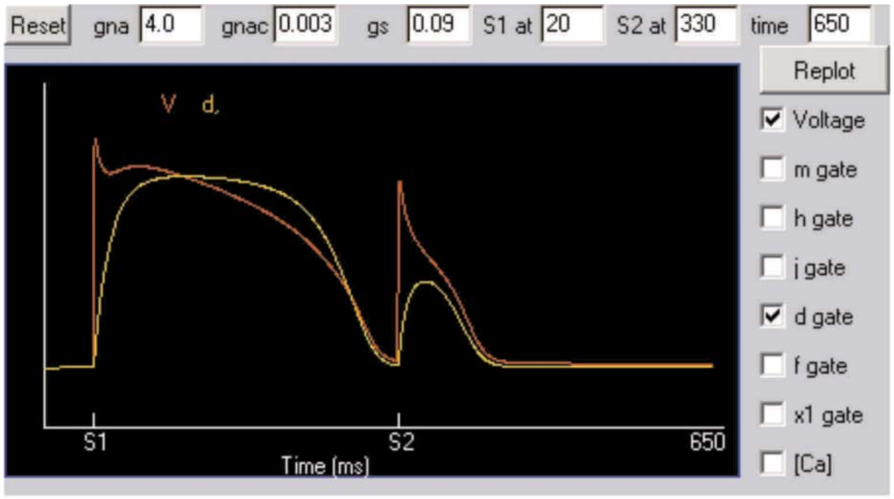
Changing parameter values changes the nullclines, which in turn change the dynamics. For instance, the parameter δ can shift the resting potential from stable to unstable and make the cell auto-oscillatory. Try changing δ from 0 to 0.04 and to 0.15 and observe the oscillatory patterns, shown in Fig. 3c, while also noticing how the v nullcline shifts. The oscillations occur because whenever the fixed point is located along the central region (positive slope) of the U nullcline, the fixed point is unstable, giving rise to oscillations. If δ is increased further to 0.22, the oscillations cease, as the fixed point is once again located in a region of negative slope on the U nullcline. However, the fixed point is now much higher than the initial state, and the system does not exhibit a full cycle. As another example, try changing γ from 1 to 3. This reduces the slope of the v nullcline so that it intersects the U nullcline in three locations, rather than just one. The middle point is unstable, but the other two are both stable. The initial stimulus moves the system to the upper fixed point, where it remains until the application of the second stimulus, which moves the system back to the lower fixed point. For further information on parameters and nullcline analysis we refer to Winfree (1991).

4.1.3. The Beeler–Reuter (BR) model applet

The continuous discovery of new ionic channels in mammalian cardiac cells, as well as the improvements in voltage-clamp techniques and data acquisition over the past 30 years, has led to a parallel increase in the complexity of mathematical models used to describe cardiac cell electrical dynamics. In 1977, Beeler and Reuter (1977) developed a model using four of eight different ionic conductances known at the time in cardiac muscle. They implemented an initial fast inward sodium current I_{Na}, similar to the one used by

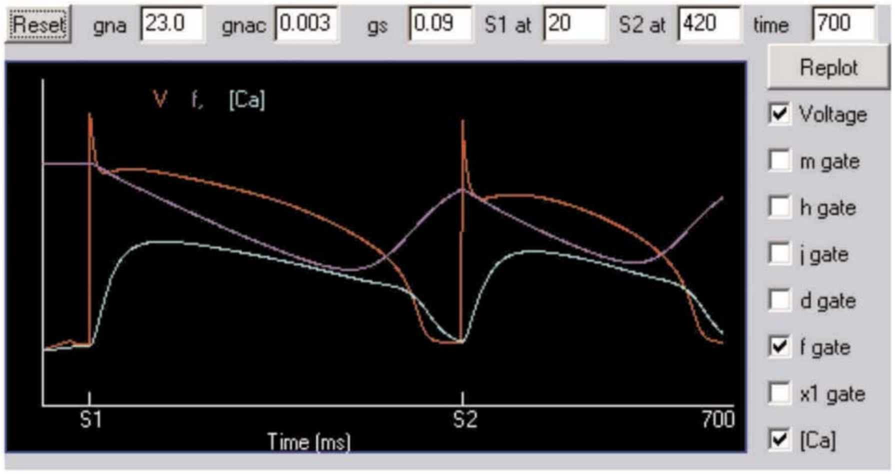
Hodgkin and Huxley, but they added an inactivator slow gate variable denoted by *j*, a time-activated outward current I_{x1}, a time-independent potassium outward current I_{K1}, and a secondary slow inward current I_s carried primarily by calcium ions. Calcium is responsible for the contraction of a cardiac cells and produces a much larger AP plateau in cardiac cells than in nerve cells. The total ionic current in the BR model is given by four currents and uses eight variables: membrane potential, six ionic gates (m, h, j, x_1, f and j) and the intracellular calcium concentration $[Ca]_i$). See Beeler and Reuter (1977) for the full set of equations.

Fig. 4a shows two APs generated by two consecutive stimuli using the BR model. The difference in plateau duration, shape and restitution response can be compared with the HH and FHN models described above. The duration of an AP in Human ventricular cardiac tissue is on the order


gna 4.0 gnac 0.003 gs 0.09 S1 at 20 S2 at 330 time 650

[✓] Voltage
[ ] m gate
[ ] h gate
[ ] j gate
[✓] d gate
[ ] f gate
[ ] x1 gate
[ ] [Ca]

(A)


gna 23.0 gnac 0.003 gs 0.09 S1 at 20 S2 at 420 time 700

[✓] Voltage
[ ] m gate
[ ] h gate
[ ] j gate
[ ] d gate
[✓] f gate
[ ] x1 gate
[✓] [Ca]

(B)

Fig. 4. (a) Activations in the BR model. Voltage (red) and the *d* gate (yellow) are shown. (b) Activations in the LR1 model. Voltage (red), the *f* gate (purple), and the intracellular calcium concentration (blue) are shown.

of 200–300 ms. In the applet, varying the ionic currents can change the duration and APD shape given by the model. For instance, the calcium current is largely responsible for the plateau phase of the AP, so that decreasing the calcium conductance g_s decreases the calcium current and thus the APD. Decreasing g_s from 0.09 to 0.04 mmho/cm^2 shortens the APD by about 50%, and the AP shape becomes more triangular. A further decrease of g_s to 0.01 yields an AP similar to the HH model. On the other hand, increasing g_s to 0.2 mmho/cm^2 prolongs the AP and changes its shape yet again by making the secondary rise in potential more prominent. As in the HH model, the sodium conductance g_{Na} is responsible for the rise of the AP, and changes to this value also can affect the AP shape and duration. Reducing g_{Na} to 1.0 or to 0.9, for example, decreases the maximum depolarization, allowing less time for the calcium to activate and producing a smaller AP. Increasing g_{Na} increases the excitability of the system and makes it easier to induce activations. By changing g_{Na} to 20 mmho/cm^2, for instance, the second activation becomes substantially longer because the increased sodium conductance allows a higher depolarization and, therefore, more time for the calcium current to activate. Under these conditions, it is easier to induce subsequent activations.

In this model the restitution relation is very strong at small DIs, that is, the smaller the difference in time between the end of the first activation and the application of the second stimulus, the smaller the second activation. Fig. 4a shows the small activation that is obtained when S1 is at 20 ms and S2 is at 320 ms. The timing of S2 must be increased much farther to about 500 ms, lengthening the DI to roughly the duration of the first AP, in order to produce a second activation whose duration is almost the same as the initial APD. Second stimulus below about 315 ms do not affect the membrane potential much since a new activation cannot be produced yet, and the system continues to return to the rest state. However, when g_{Na} is larger, a second activation can be produced earlier (e.g. at 312 ms when g_{Na} is increased to 20 mmho/cm^2).

4.1.4. The Luo–Rudy I (LR-I) model applet

In 1991 Luo and Rudy (1991) presented an ionic model (LR-I) for the cardiac AP in guinea pig ventricular cells based on the BR-model, but updated to include more recent experimental results. They reformulated the opening and closing rate coefficients for the sodium current from the BR-model making it a faster process and added three new currents, one plateau potassium current; one background current with constant conductance; and an additional potassium current with a gate variable that can be approximated by its steady-state value due to a small time constant. They retained the BR formulation for the slow inward calcium current as well as the time-dependent potassium current. However, they allowed the possibility of changing the extracellular potassium concentration. In total, phase one of the LR-I describes six different currents and uses nine variables, one of which is approximated by its steady state and is replaced by a function, so that only eight variables are needed in the calculation. Later, Luo and Rudy updated their model further to produce the LR-II model (Luo and Rudy, 1994), since then many more processes including ionic pumps and exchangers have been added and is currently known as the LRd (dynamic) model (see for example Clancy and Rudy 2001, and references therein).

The LR-I, in contrast to the BR model, produces an AP with a faster upstroke more consistent with experimental observations. From our experience with the BR model in the previous section, we expect that because of the faster sodium dynamics, the APD produced by the LR-I model will be longer and with higher depolarization. Therefore, we can predict that this model will be more excitable and will be able to produce excitations at shorter S1–S2 coupling intervals. We can also anticipate that the change in APD size at shorter S1–S2 coupling intervals will be smaller than for the BR model, and indeed this is the case, as shown in Fig. 4b.

4.2. One-dimensional rings of excitable media

Since the early experiments in 1908 by Mayer in jellyfish muscle (Mayer, 1908), it has been known that traveling waves can be self-maintained in rings of excitable tissue. These continuous patterns of re-excitation that circulate around structural or functional obstacles have been accepted as one of the main mechanisms responsible for reentrant cardiac arrhythmias.

Applying a stimulus above threshold to a site in a 1d ring generates a pulse. If all computational nodes (hereafter referred to as cells) in the ring are in the rest state at the time of the stimulus, the pulse propagates in both directions, forming two one-dimensional waves moving in opposite directions. These waves travel along the ring until they collide and annihilate each other. However, if only one of the two waves formed by the initial pulse succeeds in propagating, a continuous self-sustained wave can be initiated. This wave is an example of reentry because without additional stimuli it repeatedly excites the same tissue as it travels.

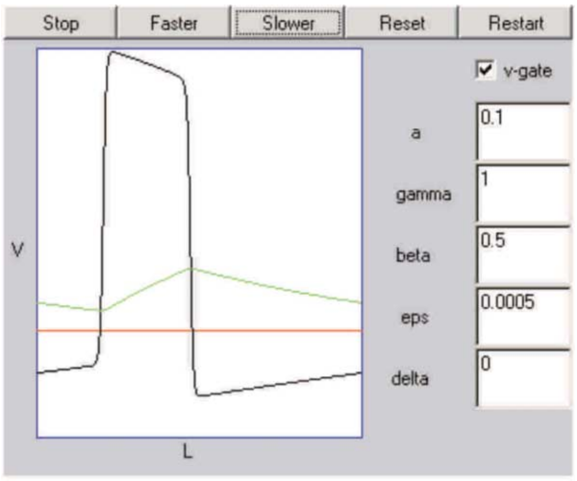
In this section, we describe several applets associated with one-dimensional rings. First, we show a ring where all cells use the FitzHugh–Nagumo model. Second, we discuss the restitution applet, which illustrates how restitution can lead to complex dynamics. Finally, we present an applet using the ring geometry and a 3-variable cell model that further demonstrates the effects of restitution.

4.2.1. The FitzHugh–Nagumo ring applet

The 1D-FHN JAVA applet consists of 200 cell units connected to form a ring. The plot area shows the strip of tissue along the x-axis. The value at the right edge is identical to the value at the left edge due to the periodic boundary conditions, thus forming a loop or a ring. The y-axis indicates the voltage value of each cell, with the red line denoting the initial rest state. When the applet is started, cells 45–50 are excited above threshold in order to produce a pulse. The cells to the right of the excitation are set initially as refractory (by inactivating the slow gate variable), and consequently the pulse only propagates to the left and forms a reentrant wave with a period determined by the length of the ring and the dynamical properties of the medium. The gate variable can be visualized by checking the v-gate choice box. The Reset button sets all cells to the rest state and sets all parameters back to the initial values. The Restart button resets all parameters back to the initial values and re-initiates the unidirectional wave.

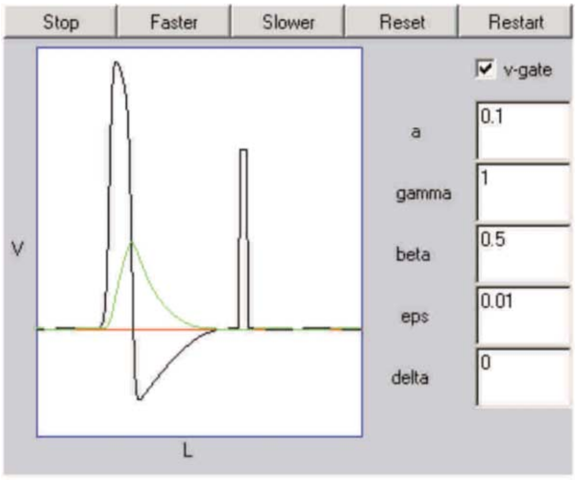
As in the FHN single cell applet, the parameters can be changed in real-time and observe their effects on the propagating wave. For example, varying ε can change the amplitude and speed of the wave, while varying γ and δ can make the system auto-oscillatory as in the single cell case. Fig. 5a shows the left-moving reentrant wave initiated by pressing the Start button, produced by the initial conditions, and subsequently altered by slowly decreasing epsilon from 0.01 to 0.0005 to produce a longer APD. The simulation can be slowed down to allow more time for observation by using the Slower button. Cells along the strip of tissue can be excited above or below threshold, possibly to initiate or to terminate waves, by clicking with the mouse inside the plot area at any time. The stimulus is applied at the cell in the cable closest to the x-position of the mouse, and its strength is proportional to the y-position of the mouse. For example, Fig. 5b shows a wave propagating to the left and a super-threshold stimulus initiated to its right. The effect of small and large perturbations can be studied by varying the threshold for excitation a and the time and size of the external stimuli.

When a stimulus is applied to the back of a wave, one of three scenarios arises. (1) If the stimulus is applied too close to the wave back, all the surrounding tissue is refractory and no excitation is produced. (2) If the stimulus is applied too far from the wave back, all the surrounding tissue has recovered to the rest state, and two new waves moving in opposite directions are produced. The wave moving in the opposite direction of the original wave eventually collides with and annihilates it, leaving again only one wave rotating in the ring. (3) If the stimulus is applied in a region known as the vulnerable window, where the tissue on the side closer to the wave back is refractory but the tissue on the side farther from the wave back is excitable, therefore, only one wave is produced. This wave propagates in the opposite direction of the original wave, and eventually the two waves collide and annihilate each other. The



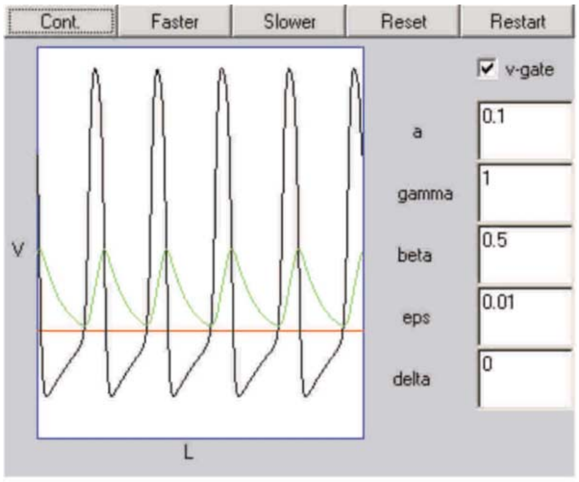
☑ v-gate
a 0.1
gamma 1
beta 0.5
eps 0.0005
delta 0

(A)



☑ v-gate
a 0.1
gamma 1
beta 0.5
eps 0.01
delta 0

(B)



☑ v-gate
a 0.1
gamma 1
beta 0.5
eps 0.01
delta 0

(C)

Fig. 5.

applet is a good tool to try to produce the three scenarios. Slowing the simulation speed may be helpful.

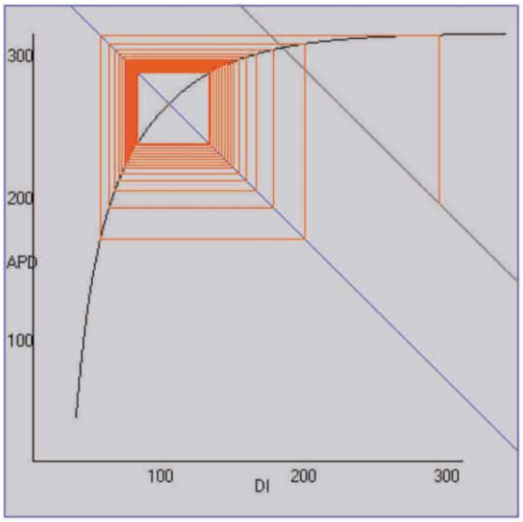
The number of reentrant waves that can fit on the ring depends on the wavelength and on the size of the ring. Using the initial set of values, the ring in the applet can support up to five traveling waves, as shown in Fig. 5c. Reproducing this pattern requires understanding many of the principles of excitable media and reentrant waves and is a recommended exercise.

4.2.2. The restitution applet

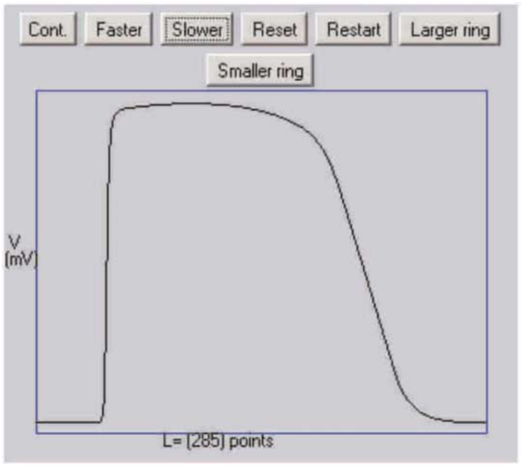
As mentioned earlier, APD restitution curves are functions that relate the duration of an APD to the previous interactivation time, or DI. When APD depends little on the previous DI, the restitution function is almost flat. This is the case of the FHN model when ε is about 0.01. As ε decreases the APD restitution steepness increases. As the restitution function becomes steeper, small changes in DI elicit large changes in APD. In particular, when the slope of the restitution function is greater than one, there is a Hopf bifurcation and oscillations of APD arise. Under such conditions, the dynamics of traveling waves becomes more complex.

Fig. 6a shows an image from the restitution applet illustrating the APD oscillations obtained when the period of a train of consecutive pulses is past the Hopf bifurcation. The period is defined as the sum of the APD and the DI and is denoted by a blue line at $-45°$. The transitions between a steady APD state and APD oscillations can be observed by varying the period up and down with the mouse. For sufficiently small periods, the DI can fall below the minimum DI (that is the DI's for which no AP can be generated), resulting in

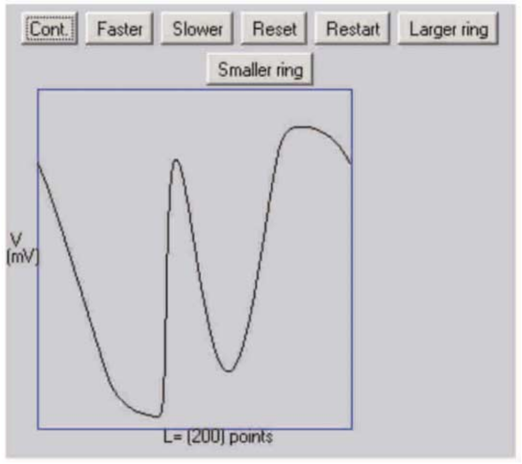
Fig. 5. (a) Instantaneous picture of a one-dimensional wave propagating to the left in a ring geometry of 200 cells using the FHN model. The fast variable U is shown in black and the slow variable v is shown in green. The red line indicates the initial rest state. (b) External excitation produced by clicking with the mouse on the window. The amplitude of the stimulus is proportional to the y-position of the mouse when it was clicked, while the x-position determined the location of the stimulus. (c) Five traveling waves propagating to the left on the ring.



(A)



(B)



(C)

Fig. 6.

what is known as conduction block and failure of wave propagation.

4.2.3. The 3-variable ring applet

The 3V-model 1D applet (Fig. 6b) shows a wave propagating on a ring with up to 300 unit cells using a simplified ionic model for cardiac cells. This model can be fitted to experimental restitution curves and is further described in (Fenton and Karma, 1998). In this case it has been tune as to produce an APD restitution curve with a steep slope, meaning that the slope is greater than one for a range of DI's, as shown in Fig. 6a. Within the applet, the size of the ring can be changed in real time, thus changing directly the period of rotation. In this applet, for single unidirectional waves propagating in rings larger than 230 U, the period of rotation is in the range for which the APD restitution curve has a slope less than one, resulting in stable waves (although some transient APD oscillations may be present if rings sizes are decreased too quickly). As the ring size is decreased below 225 cells and the period reaches the points in the restitution curve with slope greater than one, the system undergoes a Hopf bifurcation and oscillations of APD begin to occur. Fig. 6c shows an example of the complex patterns that result. Notice that although there is only one wave front, its duration in some parts of the tissue is short while in others it is long. This phenomenon is known in cardiology as discordant alternans, and it is believed to be an important factor preceding the initiation of serious arrhythmias such as ventricular fibrillation (Pastore et al., 1999). The alternans disappears as the ring size is increased again above 230 U. For further information on the mechanisms of discordant alter-

Fig. 6. (a) Example of an APD restitution curve with a region having slope greater than one. By sliding the blue line that defines the period of traveling waves, it is possible to observe the Hopf bifurcation and the increase in APD oscillations. (b) Instantaneous picture of a wave propagating on a large ring with a constant period. (c) The same wave as in (b), but on a smaller ring. The shorter period induces oscillations of APD throughout the tissue and produces a complex spatio-temporal pattern.

nans, see Watanabe et al. (2001) and references therein). The minimum ring size for which propagation persists in this system is about 195 cells. This is because at this periods the oscillations will reach DI lower than the minimum DI resulting in conduction block. These conduction blocks produced dynamically are an important mechanism for initiating complex dynamics in 2D, as discussed below.

4.3. Two-dimensional cardiac tissue

In actual cardiac tissue, pre-existing fixed obstacles in the tissue, such as scars, orifices, calcified plaques, and diseased regions with reduced excitability, can block propagating waves and allow spiral waves to form. These spirals can pin to the obstacle, propagating around them in a topologically analogous way to the one-dimensional reentry case, with the period of rotation given by the size of the obstacle and the conduction velocity dynamics along the pathway. Simulations of cardiac models in two dimensions also can support spiral waves. In the following applets, we show examples of spiral waves formed by dynamical functional blocks, which also can cause wave break in cardiac tissue. The first applet uses a square domain, while the second applet incorporates a realistic ventricular geometry so that its effects may be analyzed.

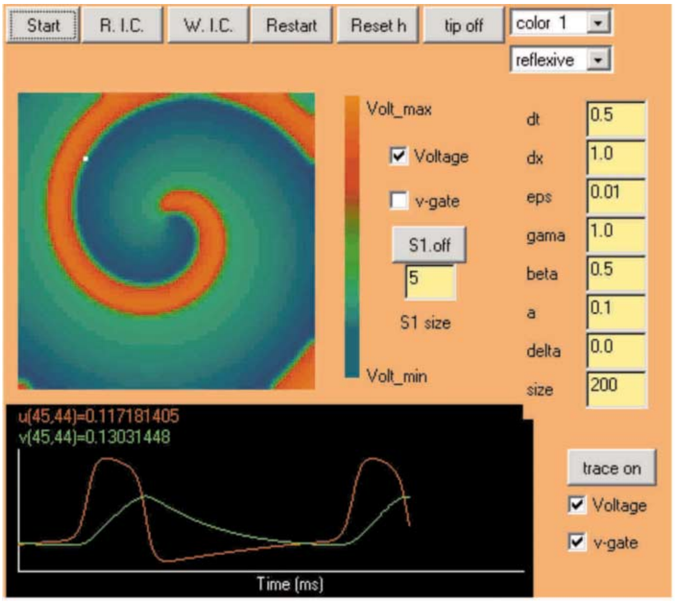
4.3.1. The FitzHugh–Nagumo sheet applet

The 2D FHN applet allows the study of wave propagation in a two-dimensional square of tissue, with 200 cells along each side. The physical size of the tissue can be varied either by resizing the computational grid between 200 and 50 (using the size entry panel) or by changing the spatial resolution Δx (which may require changing the size of the time step Δt to preserve the integration scheme's stability).

The Start button initiates the simulation with the same set of initial conditions as in the one cell and 1D FHN applets. It also initiates a plane wave on the left edge of the tissue, which propagates to the right. The Reset *h* button sets all the upper half of the tissue back to the rest state, so that pressing this button when a wave is in the tissue blocks the upper half and breaks it, forming a spiral wave as previously discussed in Section 3 (see Fig. 7a). The spiral wave dynamics can be affected by changing the parameters of the model (for a survey of spiral wave tip trajectories as function of parameters, see Winfree (1991, 1992), Belmonte et al. (1997)). In the case of very low excitability ($\varepsilon < 0.02$), spiral waves do not form, since wave breaks do not curl but instead retract (Karma, 1991).

The voltage and gate variable traces can be recorded at any time and at any point in the tissue. This is done by activating the trace button, checking the appropriate box (Voltage and/or v-gate), and then clicking at the desired location in the tissue. A white dot marks the recording position in the 2D plot and the graph area displays the trace as function of time. It also displays the coordinates of the recording site in the tissue grid as well as the value, as shown in Fig. 7a. The gate variable can be visualized instead of the voltage by selecting the v-gate choice button. For whichever field is displayed (U or v), three color map choices are available. The Restart button resets the tissue to rest state and reinitiates a wave on the left side.

As in the 1D applets, an external stimulus can be applied to the tissue with the mouse. However, to avoid moving the recording trace site (denoted with the white dot), it is necessary to click on the S1 button to enable the stimulus before clicking in the tissue. A stimulus in the tissue behaves the same way as in the 1D case. If the whole tissue surrounding the excitation is in the rest state, propagation succeeds in all directions and a target pattern (circular wave) forms. If the tissue surrounding the activation is in the refractory state, the excitation dies out. However, if only part of the tissue surrounding the excitation is refractory, propagation succeeds in some directions and fails in others, leading to a broken wave with two ends that become two counter-rotating spirals (Winfree, 1992). The simplest way to visualize this effect is to start the applet and slow down the simulation by decreasing dt. Then apply a stimulus on the back of the plane wave that propagates from the left edge. The plane wave leaves behind a gradient of refractoriness in the medium, so that



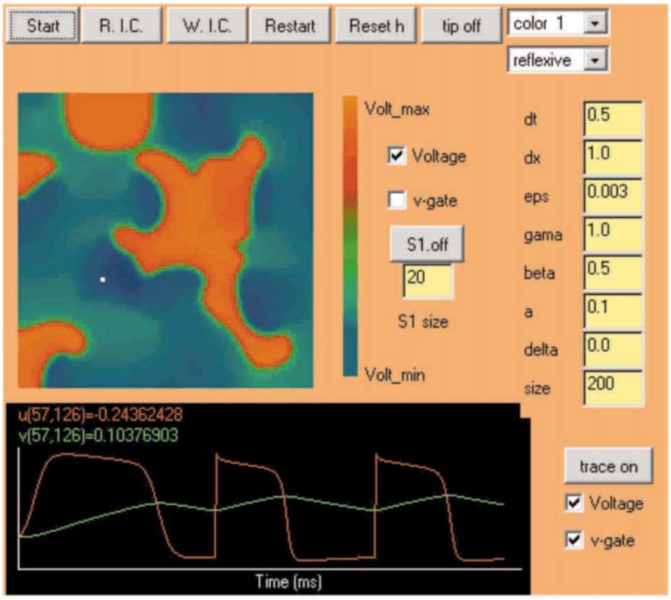
color 1
reflexive
✓ Voltage
☐ v-gate
S1 size: 5

dt	0.5
dx	1.0
eps	0.01
gama	1.0
beta	0.5
a	0.1
delta	0.0
size	200

```
u(45,44)=0.117181405
v(45,44)=0.13031448
```

✓ Voltage
✓ v-gate

(A)



color 1
reflexive
✓ Voltage
☐ v-gate
S1 size: 20

dt	0.5
dx	1.0
eps	0.003
gama	1.0
beta	0.5
a	0.1
delta	0.0
size	200

```
u(57,126)=-0.24362428
v(57,126)=0.10376903
```

✓ Voltage
✓ v-gate

(B)

Fig. 7. (a) Spiral wave induced in the 2d FHN sheet applet. Areas of highest voltage are shown in red and orange, with the lowest voltages in blue. The voltage and v-gate traces at the location of the white dot in the upper panel are shown in the lower panel. (b) Spiral wave breakup induced by the application of external stimuli in the 2d FHN sheet applet.

if the stimulus is applied at the right time behind the wave, it produces two counter-rotating spirals. It is possible to study how easily the spirals can be initiated by varying the size of the stimulus and its location relative to the plane wave. To vary the size of the stimulus, which is always square, type the new size in grid units in the space below the stimulus button. Fig. 7b shows a complex spatio-temporal pattern produced by multiple waves induced by a series of external stimuli.

There are two ways to terminate all the electrical activity in the tissue, one is by using the Restart button, the other one is to reset the whole tissue back to the rest state by stimulating the whole tissue at once (e.g. try a size of 400 U for the S1 stimulus) this is similar to applying a defibrillation shock.

4.3.2. The FitzHugh–Nagumo ventricular slice applet

In cardiac tissue, two other factors can strongly affect wave dynamics. First, propagation is anisotropic, with a wave speed roughly three times faster along the long axis of cells than in other directions. Second, the geometric structure is complex, with curved surfaces, cavities, and varying tissue thickness. Both anisotropy (Panfilov and Keener, 1995; Fenton and Karma, 1998) and anatomic structure (Fenton et al., 2001) can make the wave dynamics more complex. To illustrate some of these effects, we include a 2D applet that can simulate the propagation of waves using the FHN model in a slice of ventricular tissue. We use one 2D plane from a full 3D dissected rabbit heart (Vetter and McCulloch, 1998), with a spatial resolution of 0.025 cm. The slice corresponds to the upper part of the ventricles. The thicker left ventricle is shown on the left and the thinner right ventricle is on the right. The effect of fiber orientation, which is the direction of fastest propagation at each point and varies substantially throughout the tissue, can be included in the simulation by using the *fibers* button. Fig. 8 shows the propagation of a unidirectional wave from the left ventricle to the right ventricle and through the interventricular septum. Fig. 8 also shows how the anatomy of the slice supports reentry. In this example, the tissue contains a single reentrant wave that circulates continuously through the left ventricle and the septum in a manner roughly analogous to the one-dimensional ring geometry. Every time the reentrant wave reaches the junction of the septum and the right ventricle (twice during a given loop, at the lower (Fig. 8a) and upper (Fig. 8b) junctions as they appear in the applet), it generates a transient wave in the right ventricle. The two waves in the right ventricle annihilate each other (their collision course is seen in Fig. 8b), while the wave in the left ventricle continues to circulate.

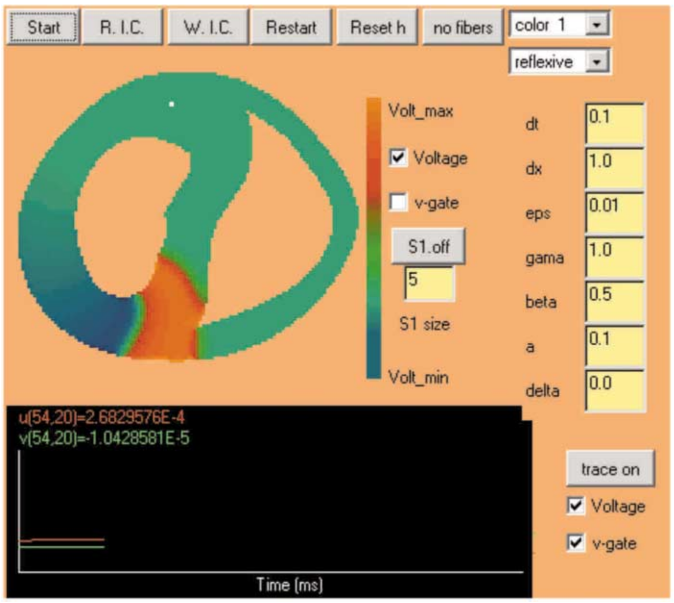
By changing the parameters of the model, it is possible to vary the wavelength of waves produced in the tissue, which can affect their dynamics. For sufficiently small wavelengths, compared with the tissue thickness, is possible to fit a spiral. Therefore, multiple unidirectional waves as well as spiral waves can be initiated.

5. Conclusions

We have described a set of instructive JAVA applets that represent a variety of excitable media, with a particular focus on cardiac cells and tissue. Single cell applets include the Hodgkin–Huxley (neural), FitzHugh–Nagumo (general), Beeler–Reuter (cardiac), and Luo–Rudy I (cardiac) models. The applets also include a restitution applet, one-dimensional rings using the FHN model and a 3-variable model, and a two-dimensional sheet and ventricular slice using the FHN model. Together, these programs can demonstrate many phenomena, including how ionic currents interact during AP generation, how repetitive stimulation can affect tissue dynamics by altering restitution, how blocking of a given ionic current affects membrane potential, how premature stimuli and conduction blocks can be used to generate and to terminate reentrant arrhythmias, and how geometry affects propagation. Thus they can be used not only as introductory illustrations, but also as sophisticated tools to deepen one's understanding of excitable media dynamics.

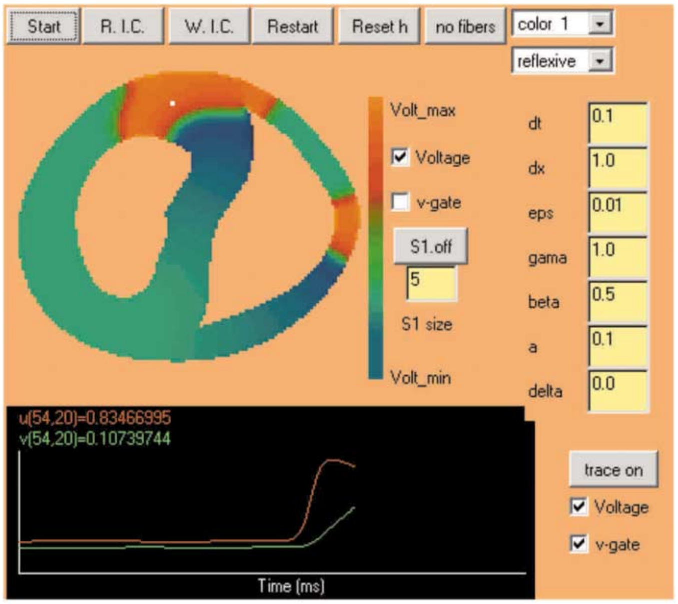
The web site includes additional applets not described in this paper, such as a 2D sheet using the Karma model (Karma, 1993). This applet allows visualization of the breakup of spiral waves due to oscillations of APD when the period of spiral rotation lies in the region for which the APD restitution has a slope greater than one. The oscillations are analogous to the ones produced in the 1D ring with the 3-variable model, but their effect is more complex in two dimensions, since they can generate conduction block and initiate multiple spiral waves.

The set of applets presented here for the study of excitable media and cardiac dynamics are being updated continuously. We will continue to include additional cellular models, in particular second-generation ionic models. These more complex models include more detailed ionic currents along



```
dt     0.1
dx     1.0
eps    0.01
gama   1.0
beta   0.5
a      0.1
delta  0.0
S1 size 5
u(54,20)=2.6829576E-4
v(54,20)=-1.0428581E-5
```

(A)



```
dt     0.1
dx     1.0
eps    0.01
gama   1.0
beta   0.5
a      0.1
delta  0.0
S1 size 5
u(54,20)=0.83466995
v(54,20)=0.10739744
```

(B)

Fig. 8. Reentrant wave circulating through a 2d slice of rabbit ventricles using the FHN model. (a) At this time, the wave is moving from the left ventricle, where it continuously circulates, into the right ventricle and the interventricular septum. (b) After traveling through the septum, the wave splits into two pieces, one entering each ventricle. The new wave in the right ventricle will collide with the existing wave located there, while the new wave in the left ventricle will continue to circulate in the same manner.

with additional physiological mechanisms, such as ionic pumps and exchangers. Examples are the Luo–Rudy phase II model (Luo and Rudy, 1994), the Luo–Rudy dynamic model (Clancy and Rudy 2001, and references therein), and the Winslow et al. model (Winslow et al., 1999) for ventricular cells and the Courtemanche et al. (1998), Nygren et al. (1998) models for atrial cells. In addition, we plan to include other anatomical models, such as the (Nielsen et al., 1991) canine ventricle. Later on we also will include other biological and chemical models for excitable media as well as 3D JAVA interfaces.

Acknowledgements

We acknowledge support from the US National Science Foundation, the Mike and Louise Stein Philantropic Fund, the Rosalyn S. Yalow Foundation for Medical Research, the Medtronic Foundation, the Guidant Foundation, the CR Bard Foundation, J & J Biosense, and NU-ASCC.

Appendix A. Cross-programming with JAVA: calling C and FORTRAN from JAVA

In this section we show how to integrate native programs into JAVA. This is useful when JAVA is intended only as a graphic user interface (GUI) and the rest of the computation is done in another language.

The JNI is a powerful part of the JAVA Developing Kit (JDK) that allows JAVA codes to interact with applications and libraries written in other languages. The JNI provides a framework for many processes and permits native codes to create, access, and update JAVA objects. Therefore, it is easy for JAVA methods to call native methods and vice versa by using the JNI framework. In this Appendix we describe how to integrate an existing C or FORTRAN program into JAVA by showing how scalars and arrays can be passed among C, FORTRAN, and JAVA.

It is worth noting that there are two major issues to consider when integrating native codes into JAVA codes. First, JNI only interfaces with C and C++. Therefore, merging JAVA with any other language such as FORTRAN, PASCAL, DELPHI, ASSEMBLY, etc. needs the use of a small C or C++ interconnecting program. Second, JNI only passes scalars and vectors. Therefore, matrices need to be converted into vectors before they can be shared. This can be accomplished easily through careful indexing. For example, a two-dimensional matrix $u(i,j)$ of size $m \times n$ can be mapped to a vector using the indexing scheme $v(i{*}m+j)$, and a three-dimensional matrix $u(i,j,k)$ of size $m \times n \times p$ can be mapped to a vector using the indexing scheme $v((i{*}m+j){*}n+k)$.

Integrating a native program with JAVA is a multi-step procedure. To illustrate we create a JAVA program that initializes two 2D matrices (5 × 5) (one single precision and the other double precision), which are sent along with two parameters (one integer and one real) to a FORTRAN program. The FORTRAN code updates some of these parameters and arrays and then sends the values back to the JAVA program.

These test codes can also be downloaded from our web site.

The JAVA code: DemoNative.JAVA

```
public class DemoNative
{
  private static final int matrixSize = 5;
    private native void doCalc(double par,
    float[] v, double[] u, int nx);
    static{
      System.loadLibrary("DemoNative");
      }
    public void goForIt()
{
  float v[] = new float[matrixSize * matrixSize];
  double u[] = new double[matrixSize *
  matrixSize];
  double par = 13.1416;
  int nn = 0;
  int nx = 25;
  for ( int i = 0; i < matrixSize; i++ ) {
    for ( int j = 0; j < matrixSize; j++){
      nn = nn + 1;
      v[i*matrixSize + j] = nn;
      u[i*matrixSize + j] = -nn;}
```

```
        }
      System.out.println("Before calling C
      function");
      System.out.print("" + par + "");
      System.out.println("");
      System.out.print("" + nn + "");
      System.out.println("");
    for ( int i = 0; i < matrixSize; i + + ){
      System.out.print("" + v[i*matrixSize] +
      "");
      System.out.print("" + u[i*matrixSize]
      + "");
      System.out.println("");
        }
    System.out.println("Now calling C
    function");
    doCalc(par, v, u, nx);
    System.out.println("After calling C
    function");
    for ( int i = 0; i < matrixSize; i + + ) {
      System.out.print("" + v[i*matrixSize]
      + "");
      System.out.print("" + u[i*matrixSize]
      + "");
      System.out.println("");
        }
      System.out.print("" + par + "");
      System.out.println("");
      System.out.print("" + nn + "");
      System.out.println("");
    }
    public static void main(String args[]){
      DemoNative dn = new DemoNative();
      dn.goForIt();
        }
}
```

The elements of the JAVA code are as follows. The native call is provided by the function 'doCalc()', whose objective is to send a double value 'par', a single precision array 'v', a double precision array 'u' and an integer value 'nx' to the native method and then get them back when the method is completed. To indicate that the function 'doCalc()' is not implemented in JAVA, but in C/C + +, we need to define it as a native procedure, i.e. 'private native void doCalc(double par, float[] v, double[] u, int nx);'. JAVA needs to load this function in a dynamically linked library when the program begins execution. This is indicated in the JAVA code by using 'static{System. loadLibrary("libraryname");}. In the example, DemoNative has been chosen as the name for the dynamic library.

The rest of the JAVA program is defined in the 'goForIt()' function which initializes the arrays, prints some of the values on the screen, executes the native call, and finally prints the new values on the screen for comparison.

The C/C + + connecting code: Csubroutine.c

```
# include <jni.h>
# include "DemoNative.h"
# include <stdlib.h>
# include <sys/types.h>
# include <time.h>
# define matrixSize 5
# ifndef_ALPHA /* necesary only for Dec al-
pha */
extern float "C" void fortransub_();
# else
extern void fortransub_();
# endif
JNIEXPORT void JNICALL Java_
DemoNative_doCalc
(JNIEnv *env, jobject jo, jdouble par, jfloatAr-
ray jv, jdoubleArray ju, jint nx)
{

    int i, j;
    /* Retrieve array length from environment
    */
    jsize len = (*env)- > GetArrayLength(env,
    jv);
    /*Retrieve contents of the array*/
    jfloat * v = (*env)- > GetFloatArrayEle-
    ments(env, jv, 0);
  /*Retrieve contents of the array*/
    jdouble * u = (*env)- > GetDoubleAr-
    rayElements(env, ju, 0);
  fortransub_(v,u,&par,&nx);
    /* releases the computed values back to
    JAVA */
  (*env)- > ReleaseDoubleArrayElements(env,
  jv, v, 0);
  (*env)- > ReleaseDoubleArrayElements(env,
  ju, u, 0);
```

}

The elements of the C/C++ code can be divided into two parts, the JAVA side and the FORTRAN side.

On the JAVA side, the native C/C++ code cannot directly access primitive JAVA arrays or strings. However, the JNI provides functions that can obtain pointers to elements of arrays (see Ref sun for further information on rules for mapping different JAVA types such as Boolean, bytes, characters, etc, with their native equivalents). These functions are incorporated into the header file DemoNative.h, which must be included in the C/C++ code. The header file is created when compiling the DemoNative.JAVA code using the JNI, as we discuss further below. The function that connects with the Java code is given by JNIEXPORT void JNICALL Java_DemoNative_doCalc(JNIEnv *env, jobject jo, jdouble par, jfloatArray jv, jdoubleArray ju, jint nx), where the first two parameters of the function provide information about the JAVA environment to the C/C++ and the rest are the parameters and arrays sent from the JAVA code. Unlike C/C++, arrays in JAVA carry length information, so before mapping types it is necessary to obtain the size array. This is done by jsize len = (*env)- > GetArrayLength(env, jv); Then the conversion from JAVA to C/C++ arrays is done by the functions (*env)- > GetFloatArrayElements(env, jv, 0); and (*env)- > GetDoubleArrayElements(env, ju, 0) for v and u, respectively. Unlike C/C++, JAVA has an automatic garbage collection that reclaims dynamically allocated memory when it determines that there are no longer references to an object and thereby avoids some memory leaks. The garbage collector, therefore, may move JAVA arrays. The JNI either 'pins' down the array or makes a copy in non-movable memory to assure that the result of Get*ArrayElements() points to a non-movable array while the native code executes. For this reason, the C/C++ code must call the function Release*ArrayElements() when it has finished using the array, to allow the JAVA code to free the array and obtain the new values that the native method may have modified.

When a C/C++ code calls a function or a subroutine in another language, the compiler needs to treat the call according to the other language's calling convention. This is not different when calling FORTRAN, where the name for the function or subroutine is system- and compiler-dependent. However, in most UNIX systems, the function name as seen from C/C++ is the original FORTRAN name converted into lower case and with an underscore appended to it (some compilers do not follow this convention, such as the ones for HP and CRAY which have their own conventions).

In this example we use fortransub_(v,u,&par,&nx) as the name for the subroutine. FORTRAN and C/C++ have different parameter passing conventions. FORTRAN parameters are passed by address while C parameters are passed by value. Therefore, when calling FORTRAN subroutines or functions from C/C++ it is necessary to pass explicitly any non-array argument as a pointer by including the character & just in front of the name of the variable passed. Another difference between C/C++ and FORTRAN is in the arrays convention. Indices in C/C++ arrays always start from 0 while in FORTRAN they start at 1 unless otherwise specified. Furthermore, for multidimensional arrays, C/C++ stores each row continuously in memory FORTRAN stores each column continuously. Therefore, indices need to be written in opposite direction when arrays are passed back and forward.

The FORTRAN code: Fsubroutine.f90.

```
  SUBROUTINE fortransub(v,u,par,nx)
implicit real*8(a-h,o-z)
real*4 v
dimension v(0:24),u(0:24)
  WRITE(*,*)'Calculating now in Fortran'
do i = 0,nx-1
  v(i) = v(i) + par
  enddo
END SUBROUTINE fortransub
```

The FORTRAN code is a simple subroutine that adds the value of the parameter 'par' to all the elements of the v array.

Once the three codes have been written they need to be compiled and the order of compilation is important. Some of the compiling flags and libraries for C/C++ and FORTRAN are machinedependent. Here we give the options necessary to compile on a Compaq and PC under WINDOWS. For other operating systems such as Sun or LINUX we refer to (Ziemlinski and Pletzer, 1999).

The DemoNative.java needs to be compiled first:

```
Javac DemoNative.java
```

This produces the executable *DemoNative.class*. The next compile statement,

```
Javah–jni DemoNative,
```

produces the header file *DemoNative.h*, which includes the important prototype information for mapping Java types with their native equivalents and is used by the C/C++ code as described above. Since JAVA is platform-independent, the JAVA compilation is the same in any machine.

The next step is to generate the dynamic library that is loaded by *DemoNative.class* when executed. For this we first create an object file from the FORTRAN code and then link it to the C/C++ when generating the dynamic library.

For Dec alpha:

```
f95 –c Fortransub.f90
cxx -D_ALPHA -shared -pthread -o libDemoNative.so -I/usr/opt/java122/include -I/usr/opt/java122/include/alpha/ csubroutine.c fortransub.o -lfor
```

For Windows using VISUAL C++:

Start a new file, click on the 'Projects' tab in the dialog menu and on 'Win32 Dynamic-Link Library'. Then name the project with the same name as the JAVA class (in this case, DemoNative). Then click on 'empty project' and create the C/C++ program (in this case, paste in the Csubroutine.c code). Add the code to the project (right click on 'DemoNative files' on the left plane, and select 'add files to project'). Add the 'DemoNative.h' header in similar way. Ensure that all files are in the same directory or in the path and click on the 'Build' menu.

This is only to generate the dynamic library from a C/C++ code; to include a fortran code we refer to the VISUAL C++ man pages.

Notice that the name of the dynamic library depends on the operating system. In UNIX the library generated is called lib*libraryname*.so while in windows it is *libraryname*.dll.

Once the dynamic library is generated then the *DemoNative.class* can be ran.

```
JAVA DemoNative
```

This produces the following output:

```
Before calling C function
13.1416
25
1.0-1.0
6.0-6.0
11.0-11.0
16.0-16.0
21.0-21.0
Now calling C function.
Calculating now in FORTRAN.
After calling C function.
14.1416-1.0
19.1416-6.0
24.1416-11.0
29.1416-16.0
34.1416-21.0
13.1416
25
```

The speed gained by using C or FORTRAN, rather than JAVA, to perform the computationally intensive part of an algorithm will depend on many factors such as the machine architecture, the compilers used, the operations performed and the degree of optimization (Haney, 1994).

Nevertheless for completeness and as an example, we show the results obtained when simulating a 2D FHN model on a 650 MHz Compaq Alpha. We simulated 400 iterations of FHN on a 2D grid of 200×200 elements using a pure JAVA (J) code, a JAVA+C (JC) code and a JAVA+C+FORTRAN (JCF) code. The codes were structured in the same way and no optimizations were performed on any of them. The CPU time used was as follow: 143 s for the J code, 48 s for the JC and 19 s for the JCF code. Therefore, compared with J, JC was three times faster, and JCF 7.5 times faster. Passing data from language to language took less than 1% of the total time and, therefore, does not seemed to have much effect on computational speed in this example.

For additional information on JNI, compilation methods and options we refer to SUN (1997), Anderson (1998), Ziemlinski and Pletzer (1999).

References

Anderson, C., 1998. Putting a JAVA Interface on your C, C++, or FORTRAN code. http://www.math.ucla.edu/~anderson/JAVAclass/JavaInterface/JavaInterface.html.

Barry P.H., 1998. Interactive Electrophysiological Software for Research and Teaching, http://www.med.unsw.edu.au/PHBSoft/.

Beeler, G.W., Reuter, H., 1977. Reconstruction of the action potential of ventricular myocardial fibres. J. Physiol. 268, 177–210.

Belmonte, A.L., Ouyang, Q., Flesselles, J.M., 1997. Experimental survey of spiral dynamics in the Belousov–Zhabotinsky reaction. J. Phys. II France 7, 1425–1468.

CellML, 2000. CellML biology. math. data. knowledge., http://www.CellML.org/.

Chapman, R.A., Fry, C.H., 1978. An analysis of the cable properties of frog ventricular myocardium. J. Physiol. 283, 263–282.

Clancy, C., Rudy, Y., 2001. Cellular consequences of HERG mutations in the long QT syndrome: precursors to sudden cardiac death. Cardiovasc. Res. 50, 301–313.

Conrad, M., Hastings, H.M., 1985. Scale change and the emergence of new information processing primitives. J. Theor. Biol. 112, 741–755.

Courtemanche, M., Ramirez, R.J., Nattel, S., 1998. Ionic mechanisms underlying human atrial action potential properties: insights from a mathematical model. Am. J. Physiol. 275, H301–H321.

Fenton, F., Karma, A., 1998. Vortex dynamics in three-dimensional continuous myocardial with fiber rotation: filament instability and fibrillation. Chaos 8, 20–47.

Fenton et al. 2001. To be submitted to prl.

FitzHugh, R.A., 1961. Impulses and physiological states in theoretical models of nerve membrane. Biophys. J. 1, 445–466.

Gray, R.A., Pertsov, A.M., Jalife, J., 1998. Spatial and temporal organization during cardiac fibrillation. Nature 392, 75–78.

Haney, S.W., 1994. Is C++ fast enough for scientific computing? Comp. Phys. 8, 690–694.

Hastings, H.M., Conrad, M., 1979. Length and evolutionary stability of food chains. Nature 282, 838–839 Reviewed in London Times 4 January 1980.

Henriquez, C.S., Papazoglou, A.A., 1996. Using computer models to underestand the roles of tissue structure and membrane dynamics in arrhythmogenesis. Proc. IEEE 84, 334–354.

Henriquez, C.S., Pormann, J., Sokka, T., Lin, R,. Lin, C., Shao, H., 2000. EASIWAVE, http://www.duke.edu/~rcl3/easiwav/.

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

Joe, W., 1997. The Specialized Conduction System of the Heart. http://www.duke.edu/~wj/ecg/.

Karma, A., 1991. Velocity selection in two-dimensional excitable media: from spiral waves to retracting fingers. In: Amar, M.B., et al. (Eds.), Growth and Form. Plenum Press, pp. 271–283.

Karma, A., 1993. Spiral breakup in model equations of action potential propagation in cardiac tissue. Phys. Rev. Lett. 71, 1103–1106.

Li, Z., 2000. Action Potential Model for Canine Ventricular Cell (WRJ3), http://nsr.bioeng.washington.edu/Software/DEMO/CANINE-AP/.

Loomis, W.F., 1982. The Development of *Dictyostelium discoideum*. Academic, New York.

Luo, C.-H., Rudy, Y., 1991. A model of the ventricular cardiac action potential: depolarization, repolarization, and their interaction. Circ. Res. 68, 1501–1526.

Luo, C.-H., Rudy, Y., 1994. A dynamic model of the cardiac ventricular action potential: I. Simulations of ionic currents and concentration changes. Circ. Res. 74, 1071–1096.

Mayer, A.G., 1908. The cause of pulsation. The Popular Science Monthly (December), 481–487.

Nielsen, P.M.F., Le Grice, I.J., Smaill, B.H., Hunter, P.J., 1991. Mathematical model of geometry and fibrous structure of the heart. Am. J. Physiol. 260, H1365–H1378.

Noble, D., 1962. A modification of the Hodgkin–Huxley equations applicable to Purkinje fibre action and pacemaker potentials. J. Physiol. 160, 317–352.

NRCAM, 2001. The Virtual Cell. http://www.nrcam.uchc.edu/index.html.

Nygren, A., Fiset, C., Firek, L., Clark, J.W., Lindblad, D.S., Clark, R.B., Giles, W.R., 1998. Mathematical model of an adult human atrial cell: the role of K^+ currents in repolarization. Circ. Res. 82, 63–81.

Panfilov, A.V., Keener, J.P., 1995. Re-entry in three-dimensional Fitzhugh–Nagumo medium with rotational anisotropy. Physica D 84, 545–552.

Pastore, J.M., Girouard, S.D., Laurita, K.R., Akar, F.G., Rosenbaum, D.S., 1999. Mechanism linking T-wave alternans to the genesis of cardiac fibrillation. Circulation 16 (99(10), 1385–1394.

Press, W.H., Vetterling, W.T., Teukolsky, S.A., Flannery, B.P., 1992. Numerical Recipes. Cambridge University Press.

Shibata, J., Bures, J., 1974. Optimum topographical conditions for reverberating cortical spreading depression in rats. J. Neurobiol. 5, 107–118.

SUN, 1997. Microsystems JAVA page, http://java.sun.com and http://java.sun.com/products/jdk/1.2/docs/guide/jni/spec/jniTOC.doc.html.

Vetter, F.J., McCulloch, A.D., 1998. Three-dimensional analysis of regional cardiac function: a model of rabbit ventricular anatomy. Prog. Biophys. Mol. Biol. 69, 157–183.

Watanabe, M., Fenton, F., Evans, S., Hastings, H., Karma, A., 2001. Mechanisms for discordant alternans. J. Cardiovasc. Electrophysiol. 12, 196–206.

Winfree, A.T., 1991. Varieties of spiral wave behavior: an experimentalist's approach to the theory of excitable media. Chaos 1, 303–334.

Winfree, A.T., 1992. The geometry of excitability. In: Nadel, L., Stein, D. (Eds.), 1992 Lectures in Complex Systems. Addison-Wesley, pp. 207–298.

Winfree, A.T., 1998. Evolving perspectives during 12 years of electrical turbulence. Chaos 8, 1–19.

Winslow, R.L., Rice, J.J., Jafri, M.S., Marban, E., O'Rourke, B., 1999. Mechanisms of altered excitation–contraction coupling in canine tachycardia-induced heart failure: II. Model studies. Circ. Res. 84, 571–586.

Ziemlinski, R., Pletzer, A. 1999. Cross-Platform Computing in a Scientific Environment. http://w3.pppl.gov/~pletzer/programs/fortran-c-c++/multiplatform/index.html.