ANALYSIS OF CELLULAR CARDIAC BIOELECTRICITY MODELS TOWARD COMPUTATIONALLY EFFICIENT WHOLE-HEART SIMULATION

by

NATHAN ALEXANDER WEDGE

Submitted in partial fulfillment of the requirements

for the degree of Master of Science

Thesis Advisors: Dr. Michael S. Branicky

Dr. M. Cenk Çavuşoğlu

Department of Electrical Engineering and Computer Science

CASE WESTERN RESERVE UNIVERSITY

August, 2004

Contents

List of Figures

Acknowledgements

Throughout my college experience, I have depended upon the support of numerous individuals unlike any other time in my life. The long-lasting effort of compiling and writing my Master's thesis has been no different. However small or large, I have the deepest gratitude to all of these people, and I will not soon forget their contributions to my education.

To all of my friends and brothers, thank you for the unending "How's your thesis going?" and "So when's your defense?" comments. I never tired of hearing them, and it was a comfort that there were so many of you interested in my success. It is my deepest hope that you will all achieve the success you are seeking.

To my advisors, thank you for the mentorship and attention you have provided over the past year. I have benefited greatly from your instruction both in and out of the classroom. There is no way I could have endured a project of this magnitude without the experienced guidance you have provided.

And finally, to my parents, thank you for everything. You have always provided everything I have needed to succeed in life, and I feel incapable of recording words to sufficiently express my gratitude. From the emotional and financial support to the attentive ear to the details of my project and the constant encouragement, I could not ask for anything more.

Analysis of Cellular Cardiac Bioelectricity Models toward Computationally Efficient Whole-Heart Simulation

Abstract

by

NATHAN ALEXANDER WEDGE

This thesis studies the characteristics of excitable cell mathematical models, with the goal of developing new insights and techniques in simulating the electrical behavior of the human heart. While very simple, small-scale models of such behavior can be simulated at real-time or better speeds on powerful computing equipment, the use of realistic cell models or organ-magnitude cell networks make the simulations computationally infeasible. We examine the FitzHugh-Nagumo model using analysis techniques for nonlinear systems and examine the effects of its parameters. Using observations from this analysis and the system's linearization, we develop methods for optimizing calculations in the single-cell model using two local interpolation techniques: nearest neighbor and locally weighted regression. In the multiple-cell setting, we generalize the system's response to stimuli, and building upon these observations, we formulate cell network simulation methods. Finally, we present a method of speeding simulations in multi-cell networks by tracking cellular activations.

Chapter 1

Introduction

1.1 Problem and Motivation

Our purpose in this thesis is to explore the characteristics of cardiac cell models with the goal of developing techniques for a computationally feasible whole-heart model. Such a model could be invaluable in the study of human heart pathology and the development of drugs for the treatment of various disorders. Thus, its potential value is undeniable. A whole-heart model could dramatically expand our presently limited understanding of cardiovascular disease and abnormality while providing a convenient, noninvasive, and inexpensive method of proposing and testing revolutionary drug therapies and other treatment interventions.

Development of techniques for large-scale modeling of systems is a common theme across many fields of research today. Cardiac modeling presents complications not found in many modeling tasks, since excitable cell models are nonlinear in nature, and many ordinary and valuable analysis techniques for differential equations do not apply to nonlinear systems. The simplest cardiac cell models have two variables while realistic models can have dozens, and the total number of cells in the human heart is on the order of 10^{10} (i.e., in the tens of billions). Therefore, extraneous details must

be abstracted away so that only relevant information is actively simulated.

1.2 Cardiac Modeling Background

Researchers have long worked to capture the behavior of the human heart with a realistic mathematical model. In fact, preliminary work can be traced back to Van der Pol and Van der Mark's work in the late 1920's [33] that compared the human heartbeat to a second-order differential equation. Later work over the past seven decades has taken large strides forward to capture the complex voltage signals and ionic flows that mark the operation of the human heart. Successive models have captured specific features not found in their ancestors, leading to recently available realistic cardiac models.

In 1952, Hodgkin and Huxley proposed the first equation model [10] designed to mimic the excitable behavior of an individual cell. Originally developed as a model of the giant squid nerve axon, it was found to provide a useful model of individual cardiac cells as well, due to their similarly excitable nature. This fundamental model is widely considered to be an impressively large initial step into what would later evolve into the field of cardiac modeling. Following their work, other researchers began to develop more complex models to bring additional realism to cardiac modeling.

FitzHugh [7], and soon after, Nagumo [21], were two of the first researchers to investigate the properties of the model put forward by Hodgkin and Huxley years earlier. FitzHugh provided a detailed mathematical analysis of their model, and proposed a model of his own, based on a simplifying assumption that the sum of two of the variables was nearly a constant. Still, FitzHugh's new model retained the essential excitable characteristics of the previous model. By contrast, Nagumo approached the problem from an electrical point of view and created a nonlinear circuit that paralleled the behavior of FitzHugh's simplified model. Credit for this

new model, to which we devote our attention in this thesis, was divided between the two researchers. For a more detailed overview of the history of cardiac model development see [24].

1.3 Current Research

As further research effort increased the complexity and quality of available cardiac models, the new field of computational biology came into being. When FitzHugh published his first paper, the analog computer was a modern computational tool. Even the computational power available today that dwarfs FitzHugh's resources is not capable of providing real-time simulations of the relatively simple dynamics of his model across an organ-magnitude system. Therefore, researchers have begun to turn their attention toward development of novel mathematical and computational methods to make such simulations feasible for modern computing hardware. Still, this effort is recent and has great potential remaining for development.

Historically, two main approaches exist in attempts to build cardiac models: discrete (or network) modeling and continuum modeling. The former, which is the basis for the work in the later portion of this thesis, conceptualizes the heart tissue as a large, interconnected network of cells, each individually described by an instance of some model equation system (e.g., that of FitzHugh and Nagumo). Elements in these networks are subject to an electrical interaction that provides the mechanism for the propagation of potential waves and thus the heartbeat. In such an approach, the individual cells often have a small number of neighbors through which any potential wave must pass to reach them. Simulations of this kind become computationally demanding well before the magnitude scale of true heart tissue.

Modern simulations based on the discrete modeling approach use a cell count that is often well below the actual count of the human heart, to avoid the excessive

demands of such calculations. One of the largest simulations constructed through discrete modeling [20] used an array of over 140,000 circuit elements to simulate a small (0.25 mm by 5.0 mm) section of cardiac tissue. Another used a hybridized method in which tissue was modeled according to continuum modeling theory or discrete modeling theory depending on local conditions. The fully discretized heart contained two million cells [32]. The development of these and other models bore out important qualitative results, but the individual efforts do not attempt to describe the whole-heart in a computationally feasible setting.

In contrast to the discrete modeling idea, other research has followed the continuum modeling theory, in which heart tissue is assumed to be a continuous region of both extracellular and intracellular space rather than a set of distinct cells. Based on the assumption that the features of phenomena to be observed are of large scale compared to that of the individual cell, this approach discards the concept of the discrete cell. The specific equations corresponding to this method of modeling are simulated using the same basic computational ideas as the discrete approach, despite the fundamentally different assumptions of the two approaches.

Continuum modeling suffers from the same restrictions of computational demand as discrete modeling. Recent research suggests that the simulation of a single heartbeat, approximately one half-second of real time, requires "about 50 gigabytes of storage... for a full-scale heart simulation, and many days on a high-performance computer" [13] using this technique. Currently, researchers are inventing new adaptive techniques that regulate the simulation timesteps based on local activity [26, 27] and applying large-scale parallel processing computer systems [36] to deal with the resource demands of heart simulation. Other techniques are adaptive on the spatial scale [11], and actually split discrete cells that show signs of impending electrical activation in order to increase the spatial resolution around wavefronts. For an overview of the approaches to whole-heart modeling (including information about modeling

the mechanical and fluidic aspects of the heart) see [13, 17, 23].

The continued use of the simple FitzHugh-Nagumo model [11, 29] speaks to the lack of real development in the quest to make large-scale simulation feasible. Though models of this level are widely accepted as valid for qualitative analysis of heart phenomena, more modern and complex models like those of Luo and Rudy [16] (with 14 currents and 11 gating variables) or DiFrancesco and Noble [6] (with 12 currents and 7 gating variables) are superior in their ability to represent individual ion channel dynamics and concentration changes. Additionally, their complexity and corresponding computational requirements dwarf the single current and gating variable of the FitzHugh-Nagumo model. Thus, a whole-heart model that provides these details through use of one of the more complex models will be more useful from both a qualitative and quantitative standpoint.

1.4 Thesis Contributions

This thesis explores the structure of the FitzHugh-Nagumo excitable cell model in order to characterize which details can be discarded through abstraction and which are important to the model's overall dynamics. Based upon the concept of electrical wave propagation in the human heart, we examine the model using general analysis techniques for nonlinear systems and give a thorough treatment of the effects of its parameters on waveform shape and periodicity. Using observations from this analysis and an analysis of the system's linearization, we pose a method of optimizing calculations in the single-cell model with two local interpolation techniques: nearest neighbor and locally weighted regression.

To advance to the multiple-cell setting, we establish the system's general response to diffusive stimuli and discuss the effects of several specific types of stimuli. Building upon these observations, we formulate cell network simulations and describe the types

of electrical waves that are possible in such settings. Finally, we use our previous analysis to form the basis for a method of speeding simulations in multi-cell networks by tracking a list of "active" cells.

Portions of this work have been detailed previously in [35].

1.5 Thesis Outline

The thesis is separated into six chapters:

Chapter 1 introduces the theme of cardiac modeling and the associated computational complications, along with progress in the field. It further outlines the motivating problems and contributions of the thesis.

Chapter 2 presents the FitzHugh-Nagumo model, which we use to analyze excitable cardiac cell activity throughout the thesis. It proceeds to characterize the dynamics of the model in a single-cell environment and provides details about the system parameters and its periodic nature.

Chapter 3 studies the degree of nonlinearity in the FitzHugh-Nagumo model by applying a linear approximation for the model. We introduce the idea of capturing the model as a mapping relationship between present and future values. Further, we propose the use of two interpolation algorithms (nearest neighbor and locally weighted regression) to reconstruct this nonlinear mapping by sampling-based methods.

Chapter 4 explains the mathematical concepts that model diffusion and wave propagation through an excitable media. We draw an analogy between the diffusion concept and forcing in ordinary differential equations, by which we analyze the responses of the single-cell FitzHugh-Nagumo model to various stimulus.

Finally, we show that the model possesses a certain "constant shape." That is, there exists a certain excitation shape for which the model outputs the same waveform that is given as the input, albeit with a timeshift.

Chapter 5 progresses toward multi-cell simulations in one and two dimensions by introducing several simulation cases. We outline the waveform classifications that can occur in two-dimensional simulations and apply the previously introduced dynamics of the model to motivate a novel activity list algorithm to speed general simulations.

Chapter 6 concludes the thesis by summarizing the important developments reached and proposes additional possibilities for further research.

1.6 Implementation

Throughout this thesis, the data and plots displayed were generated using MATLAB 6.5.1 on a Pentium IV 2.4 GHz processor PC with 1.5 GB of RAM. Algorithms were implemented using purely built-in MATLAB routines whenever possible. The exception is the numerical integration methods (Euler and Runge-Kutta), which used hand-coded implementations in speed-critical settings, and MATLAB's `ode45` method in analysis. MATLAB code for algorithms described within this thesis can be found in Appendix A. Error plots given reflect absolute error between full-fledged simulations and the output from the appropriate approximation or algorithm. Many preliminary results, upon which this thesis is based, were generated with simulation programs written in C++ using the OpenGL and GLUT graphics libraries. See Appendix B.

Chapter 2

Basic Analysis of the Single-Cell Model

The preliminary step in understanding the evolution of cardiac electrical potentials lies in examining the characteristics of the individual cell. This requires a mathematical model that will realistically approximate these characteristics, to the extent that the model data is relevant. In this exploration, we choose the FitzHugh-Nagumo model as a straightforward but qualitatively representative model with which to develop new insights into model behavior and simulation techniques. This chapter introduces various basic analysis techniques for the model and scrutinizes several features of the model in a single-cell setting.

2.1 FitzHugh-Nagumo Model

In this thesis, we give detailed attention to the *FitzHugh-Nagumo excitable cell model* [7], both in the context of a single-cell environment and of a multi-cellular network. This particular model gives a qualitative representation of cardiac bioelectric phenomena, so its waveforms and time are considered without units. The model is specified as follows.

2.1.1 Equations

The FitzHugh-Nagumo model is governed by two coupled first-order differential equations:

$$\frac{\partial V}{\partial t} = \frac{1}{\epsilon}\left(V - \frac{V^3}{3} - W\right), \tag{2.1}$$

$$\frac{\partial W}{\partial t} = \epsilon\left(V - \gamma W + \beta\right). \tag{2.2}$$

The first variable, V, represents the electrical potential of the cell and is referred to as the fast variable due to its more rapid change compared to the second variable. The slow variable, W, represents a combination of the sodium and potassium gating variable of the cell, and is analogous to the cell's refractory potential. An excitation of a cell's electrical potential is followed by a similar wave pattern in the gating variable, which in turn prevents the cell from being re-excited immediately afterward or sustained in the excited state. Thus, the slow dynamics of the gating variable helps enforce a regularity in electrical excitation of cardiac cells.

2.1.2 Parameters

The FitzHugh-Nagumo system contains three parameters that control its dynamics: ϵ, β, and γ. The value of ϵ, which is typically much less than one, is responsible for the fast movements of V relative to W. The effects of the parameters will be examined in detail later in this section. Unless otherwise stated, typical values applied in this thesis for the aforementioned parameters are as follows:

$$\epsilon = 0.2, \qquad \beta = 0.7, \qquad \gamma = 0.8.$$

2.2 Solution Techniques

In common systems of ordinary differential equations (ODEs), one of several conventional numerical methods is typically applied to solve the system. Two examples of such methods are the Euler method and the Runge-Kutta method (see [9]), both of which can be applied to the FitzHugh-Nagumo system. The Euler method, which predicts the solution values of differential equations for (small) incremental timesteps, is useful in preliminary analysis to qualify the general shapes and characteristics of the FitzHugh-Nagumo curves. However, since it is lower-order and thus prone to numerical errors, more developed analysis often demands application of the Runge-Kutta method to obtain more accurate solutions. Thus, all FitzHugh-Nagumo solutions presented in this thesis are completed using a form of the fourth-order Runge-Kutta method. This Runge-Kutta method (of fourth order) computes a future value by using a weighted average of slopes calculated at the current time and times one-half, one, and two timesteps in the future. Figure 2.1 presents a comparison of solutions reached by the two methods for three different timesteps. The "real" solution is one computed by the Runge-Kutta method with a timestep of 0.0001 (and verified against a similar Euler method calculation), to ensure minimal numerical error. It should be noted that although both Euler and Runge-Kutta method solutions can often remain stable along increasing timesteps in systems like this, numerical errors arise at larger timesteps in the form of time lags that will cause them to diverge from the true solution.

2.3 Phase Plane Analysis

Continuing the application of standard ordinary differential equation analysis methods, this section examines the FitzHugh-Nagumo model in the phase (V-W) plane. Phase plane analysis of the system helps to illustrate the model's long-term behavior,

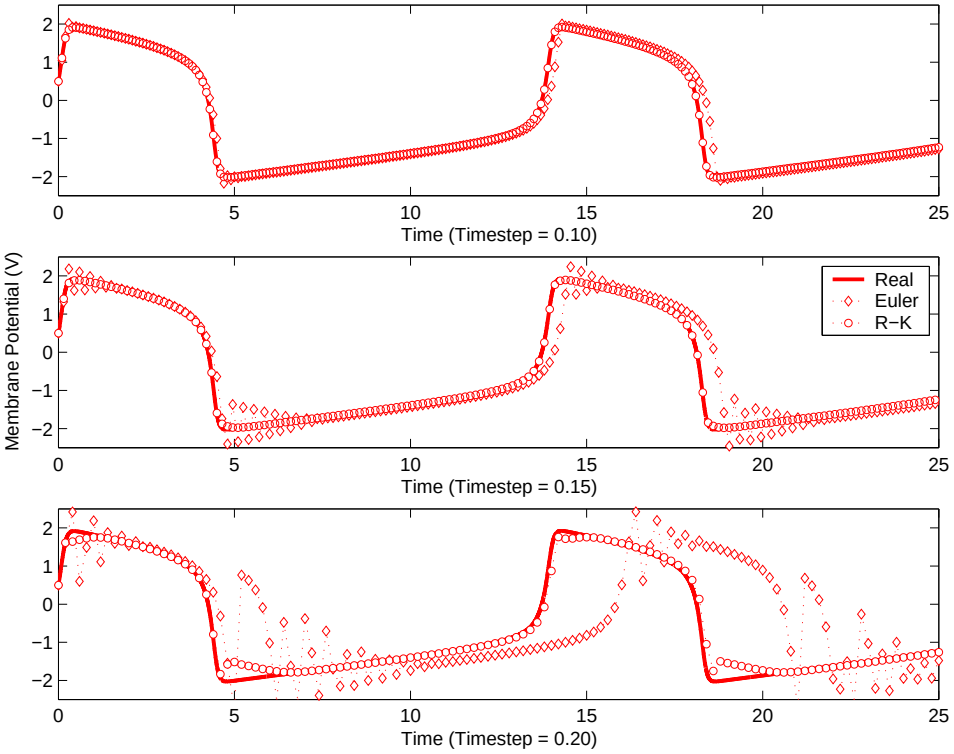


Figure 2.1: Euler vs. Runge-Kutta Comparison

along with its limit cycles and equilibrium points.

2.3.1 Nullclines

The system has two nullclines, whose equations can be derived directly from Equations (2.1) and (2.2) by setting $\frac{dV}{dt}$ and $\frac{dW}{dt}$ to zero. The equations for these nullclines, depicted graphically in Figure 2.2 are as follows:

$$W = V - \frac{V^3}{3}, \tag{2.3}$$

$$W = \frac{V + \beta}{\gamma}. \tag{2.4}$$

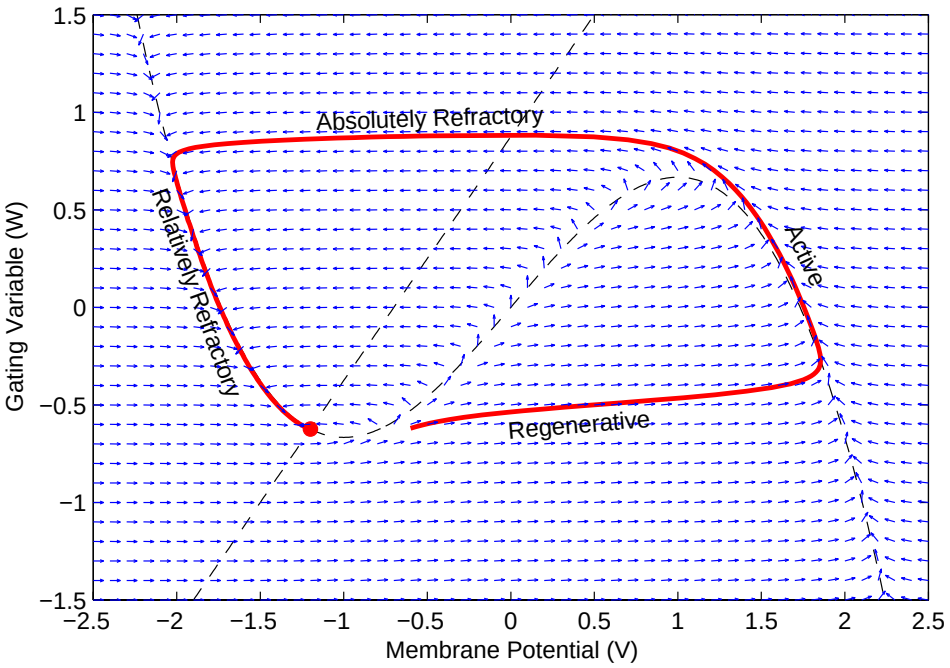


Figure 2.2: Phase Plane with Nullclines and Sample Solution Trajectory. The sample trajectory and nullclines are shown with the system's direction field, and the equilibrium point is signified with a filled point.

2.3.2 Equilibrium Points

The intersections of the two nullclines reveal equilibrium points in up to three locations, given by the roots of a cubic equation. For the system in general, equilibrium points are given by the cubic equation as follows:

$$\frac{V^3}{3} + V\left(\frac{1}{\gamma} - 1\right) + \frac{\beta}{\gamma} = 0. \tag{2.5}$$

For our typical parameter values, two of the roots have imaginary components, leading to a single equilibrium point at $(V, W) \approx (-1.1994, -0.6243)$ in the real numbers.

2.3.3 Cell Phase States

As shown in Figure 2.2, viewing the solutions of the FitzHugh-Nagumo system in the phase plane allows its behavior to be logically separated into four main phases [7]:

regenerative, active, absolutely refractory, and relatively refractory. These phases are listed in the figure along the edges of a sample solution trajectory started with initial values $(V, W) = (-0.6, -0.6243)$. An external electrical stimulus of sufficient magnitude will initiate an excitation cycle, in which the cell progresses through these four states. In the *regenerative phase*, the cell begins a buildup of potential across its membrane. If the perturbation is of insufficient magnitude, the cell will simply relax back to its equilibrium, rather than experiencing a full excitation cycle. The *active phase* represents the period in which the cell's membrane potential is temporarily suspended near a peak value. Entering the *absolutely refractory phase*, the cell's membrane potential drops rapidly toward its equilibrium value. At this point, as shown by the direction field, the gating variable is at a maximum, and the cell is quite resistant to any external stimulus. In the final phase, *relatively refractory*, it is again possible to stimulate the cell into a renewed excitation cycle, though it will display some resistance to such stimulation until it again reaches the equilibrium point.

2.4 Parameter Significance

Examining the FitzHugh-Nagumo model equations under a range of parameters exposes a general similarity in wave shapes across all values of the parameters. The parameter ϵ has the primary role of controlling the relative dynamics between V and W, making for sharp peaks in V at near-zero values and smoother peaks for larger values. Similarly, W curves sharpen at near-zero values of ϵ to the point of being similar to a sawtooth curve and have a smoother pattern for larger values. Contrasts between the variables arise in the magnitude of the slopes. For small values of ϵ, V curves have near-infinite slope values when the electrical potential peaks and relaxes, while W slopes tend to reach a nearly constant value that decreases with ϵ values.

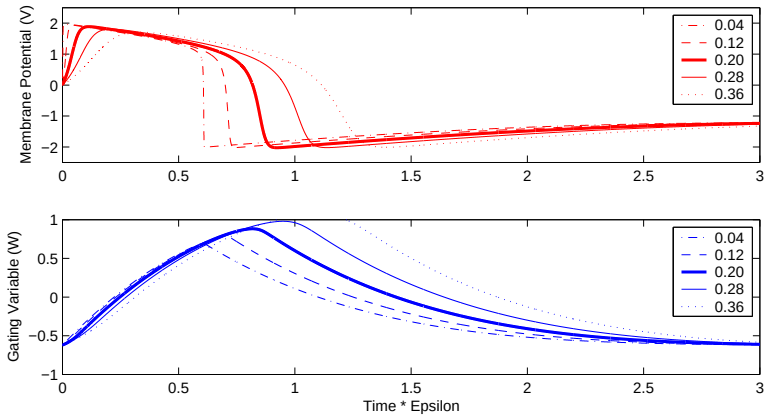


Figure 2.3: Epsilon Parameter Effects (Time Corrected by the Value of ϵ)

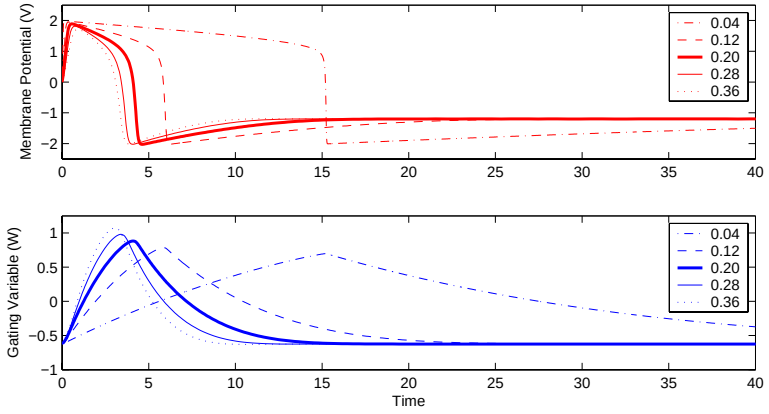


Figure 2.4: Epsilon Parameter Effects (Unscaled Time)

Thus, the W curves take on a triangular shape at smaller values of ϵ. These effects also lead to a time dilation with ϵ that extends excitation cycles in time with smaller parameter values. Figures 2.3 and 2.4 display curves associated with different values of the parameter ϵ with and without time scaled by a factor of ϵ to show these characteristics.

The other two parameters, β and γ, have only subtle effects on the wave shapes of FitzHugh-Nagumo solutions. In fact, the only effect γ has on the wave shape is a negligible deflection at the relaxation of the membrane potential (the so-called relatively refractory phase). The effects of β are quite similar to those of γ, and are still

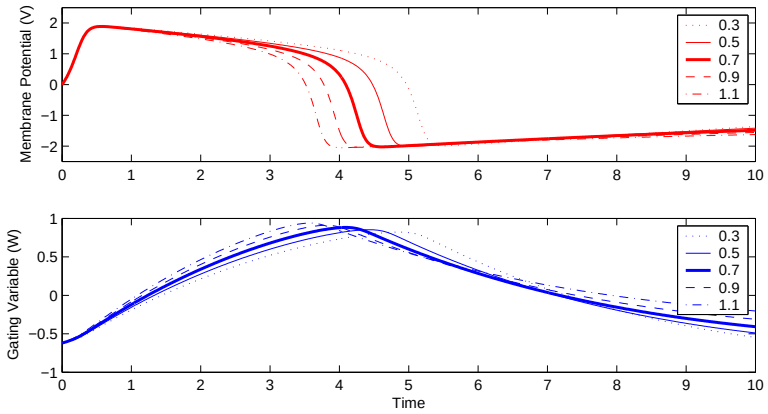


Figure 2.5: Beta Parameter Effects

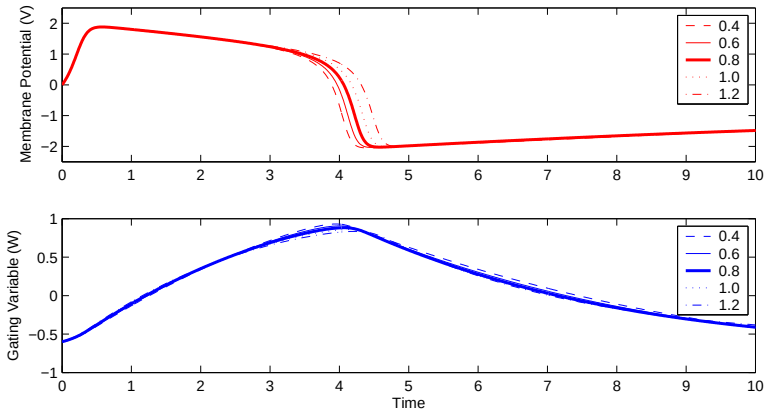


Figure 2.6: Gamma Parameter Effects

centered on the relatively refractory phase of the waveform but are more pronounced. Additionally, different values of β cause a noticeable shift in the time at which the maximum of the gating variable occurs. Figures 2.5 and 2.6 demonstrate the effects of varying these two parameters. Beyond these effects, examining the system's response to different values of β and γ reveals an interesting effect for certain parameter pairs: spontaneous periodicity. We explore this in the next section.

2.5 System Periodicity

There is one special case in the FitzHugh-Nagumo model that brings about a fundamental shift in its time behavior. If the linear nullcline, Equation (2.4), intersects the cubic nullcline, Equation (2.3), in the center region of positive slope ($V \in [-1, 1]$), it creates a Hopf bifurcation. In such a bifurcation, the equilibrium point becomes unstable and a limit cycle of similar shape to the solution curve in Figure 2.2 arises. This creates a natural periodicity in the system, which is not caused by any outside input. The intersection required for this limit cycle behavior is brought about by certain value pairs of β and γ. Since the cubic nullcline is stationary across all values of the system's parameters, it is straightforward to show that the appropriate intersection is bounded by

$$1 - \frac{2\gamma}{3} > \beta, \tag{2.6}$$

with the assumption that $\beta \geq 0$ and $\gamma \geq 0$ (β and γ are both positive parameters). Despite the onset of periodic behavior in this case, the system continues to display strong similarities in waveform shape.

We can characterize the period of the FitzHugh-Nagumo system through the use of the solution's *autocorrelation function*, given by

$$r(t) = \lim_{T \to \infty} \frac{1}{2T} \int_{-T}^{T} f(\tau) f(t + \tau) d\tau. \tag{2.7}$$

Since the autocorrelation function is given by an integral of two time-shifted versions of a solution, it takes up maximum values when the two solutions are synchronized at a multiple of their period. Therefore, the distance between these maximum values is the period of the solution. Figure 2.7 shows a sample autocorrelation function for $\beta = 0.5$ and $\gamma = 0.5$, where the peak-to-peak distance is 13.88, corresponding to the period of the system.

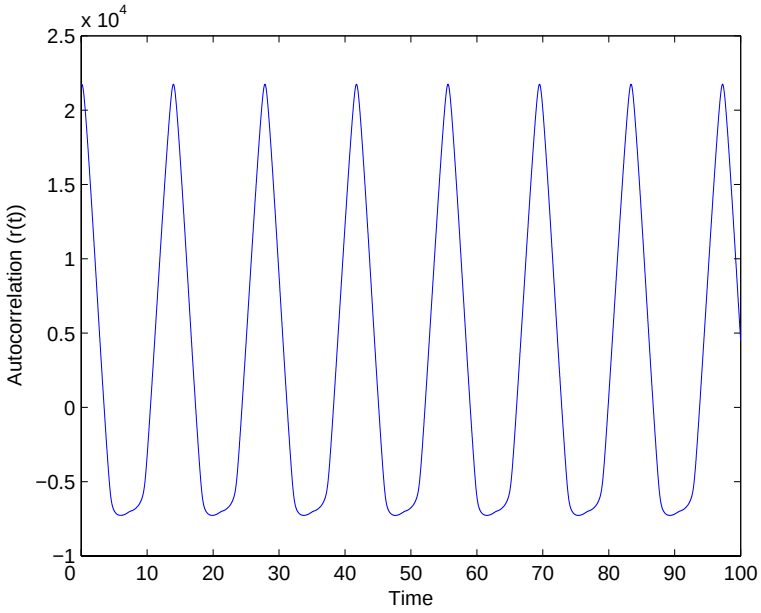
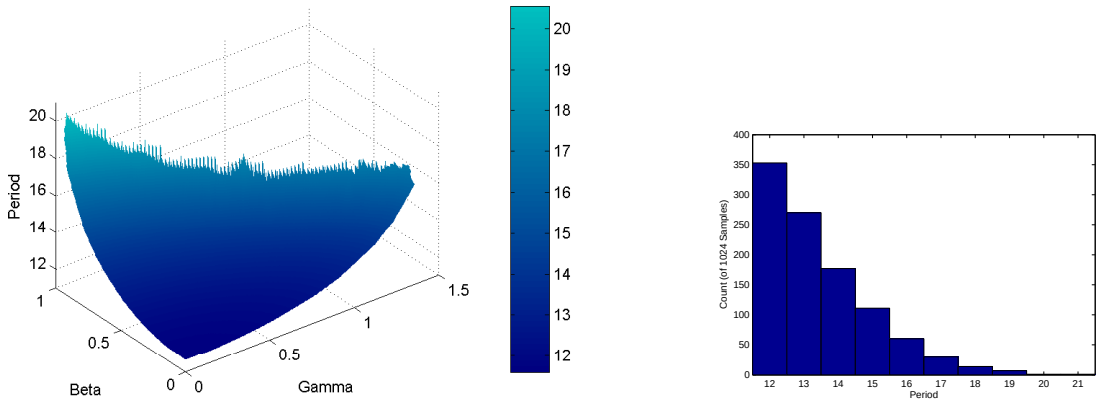


Figure 2.7: Sample Autocorrelation Function (for $\beta = 0.5$, $\gamma = 0.5$)

By sampling within the appropriate values of β and γ and calculating their solution periods, we can represent the period of the system as a smooth function of β and γ. See Figure 2.8(a). Jagged edges in this surface are a consequence of the numerical methods used to produce the plot; they are not a function of the system itself. Figure 2.8(b) shows a histogram plot of the period data, representing the frequency of appearance of the different values of period. In the approximate range of periods ($11.5 \leq T \leq 20.5$), the lower period values ($11.5 \leq T \leq 14.5$, whose solutions are more representative of the characteristic shapes seen in previous figures) dominate.

2.6 Characteristic Wave Shapes

Across all values of the system parameters, the FitzHugh-Nagumo system displays a shape characteristic of the excitation behavior the system seeks to model. Though the shape varies based on the exact values of the parameters and any external stimulus applied, it retains the discernible excitation phase and the four-state motion through



(a) (b)

Figure 2.8: Period Correspondence and Distribution

the phase plane (see Section 2.3.3). Additionally, the excitation shape is very robust to the specific choice of initial conditions; aside from a time shift associated with the regenerative phase of the excitation, waves in the FitzHugh-Nagumo system settle onto the same cycle through the active phase and both refractory phases. This tendency is shown in Figure 2.9 under different initial values of the membrane potential (V) which are capable of expressing an excitation cycle.

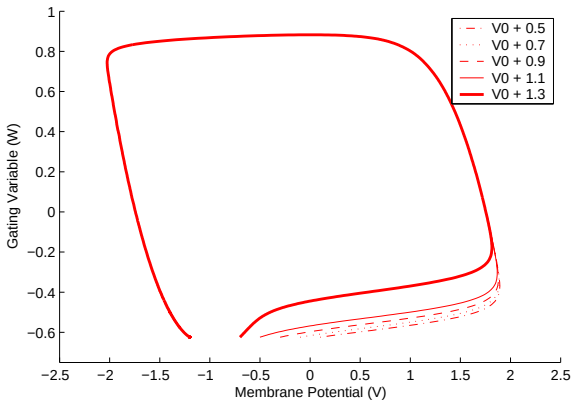


Figure 2.9: Characteristic Wave Shapes

Chapter 3

Simulation Techniques

Having established a number of basic properties of the FitzHugh-Nagumo system in the previous chapter, this chapter outlines methods of approximating the solution of this nonlinear system. We first examine linearization and then move on to establish two computational methods that can be used to determine the system's solutions over large timesteps.

3.1 Linear Approximation

In general, the first step toward analyzing any nonlinear system involves approximating it with a linear system, in the hope that this approximation will yield valuable insights into the nonlinear system's properties. Additionally, if the error associated with the approximation is small, the linear approximation can be used in place of the full-fledged system.

3.1.1 Theory

Applying a concept similar to the approximation of a function using its Taylor Series, we seek to approximate the variables of the FitzHugh-Nagumo system at some future

time as a function of its current values. Taylor's theorem states that a function's value can be approximated by a series involving the derivatives of the function. To extend this to a multi-variable system, we propose that the system can be described by a series of weighted products of current values (where Δx represents a vector of the system's variables):

$$\Delta x^{+} = M\Delta x + \frac{1}{2}H(\Delta x, \Delta x) + \cdots, \tag{3.1}$$

where M and H are the first two terms in this extended Taylor series. Much as M is premultipled to Δx to generate the linear terms, the higher-order terms are composed of two elements: an outer product that generates the individual terms (e.g., $\Delta V \cdot \Delta W$) and a tensor (such as H) whose elements provide their proportionalities. Each tensor has a dimension equal to the order of its term (i.e., H is of order three). The term Δx^{+} denotes the system's variable values at some advanced time τ units later than the original Δx is defined. For the FitzHugh-Nagumo system, $\Delta x = [\Delta V \ \Delta W]^T$. Thus, $\Delta V^{+} = M_{11}\Delta V + M_{12}\Delta W + \frac{1}{2}(H_{111}\Delta V^2 + H_{112}\Delta V\Delta W + H_{121}\Delta W\Delta V + H_{122}\Delta W^2) + \cdots$ would represent a series expansion for the membrane potential value.

Further, we propose that if a linear approximation of the FitzHugh-Nagumo system is valid, any changes (disruptions from equilibrium) to the system can be related to resulting values after a certain passage of time with a single term as:

$$\begin{bmatrix} \Delta V^{+} \\ \Delta W^{+} \end{bmatrix} \approx \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} \Delta V \\ \Delta W \end{bmatrix}. \tag{3.2}$$

This is equivalent to a statement that the system's future values are purely linear functions of its present values, or can be approximated as such. In a multi-variable system, the higher-order terms would include cross terms (i.e. $\Delta V \cdot \Delta W$) and power terms (i.e. $(\Delta V)^2$) to capture the nonlinearity in the response. Therefore, dealing with a higher-order approximation quickly becomes difficult due to the increasing

number of terms and the complexity of available methodology for solving for the constants (i.e., the components of the tensor H).

3.1.2 Implementation

Two methods for determining the components of the matrix M in Equation (3.2) are immediately clear. First, if we analytically compute the system's Jacobian (see [3], Chapter 5), we find that it is

$$A \triangleq \begin{bmatrix} \frac{1}{\epsilon}(1 - V_0^2) & -\frac{1}{\epsilon} \\ \epsilon & -\epsilon\gamma \end{bmatrix}. \tag{3.3}$$

Evaluating it at the equilibrium point $(V_0, W_0) = (-1.1994, -0.6243)$, the matrix A has numerical values and the resulting system is linear (of the form $\dot{x}(t) = Ax(t)$). Therefore, we can use principles of linear system analysis (see [4], Chapter 4) to calculate the matrix M in Equation (3.2) with the quantity $e^{A\tau}$. With $\tau = 0.1$, this results in the linearized system

$$\begin{bmatrix} \Delta V^+ \\ \Delta W^+ \end{bmatrix} = \begin{bmatrix} 0.799 & -0.445 \\ 0.018 & 0.980 \end{bmatrix} \begin{bmatrix} \Delta V \\ \Delta W \end{bmatrix}. \tag{3.4}$$

In place of this method, we can also use a numerical technique to derive the linearized system, as described next.

Conceptualizing the relationship between present values V and W and future values V^+ and W^+ as a pair of surfaces in three dimensions (for convenient visualization) allows us to use the elements M_{ii} as independent slopes of the surfaces. These two surfaces, which map $(V, W) \rightarrow V^+$ and $(V, W) \rightarrow W^+$ with a given timestep τ, can then be approximated by a planar surface tangential to them at the equilibrium point (where $(\Delta V, \Delta W) = (0, 0)$). The elements of the matrix M can then be determined with standard mathematical methods. The most convenient options are to solve the

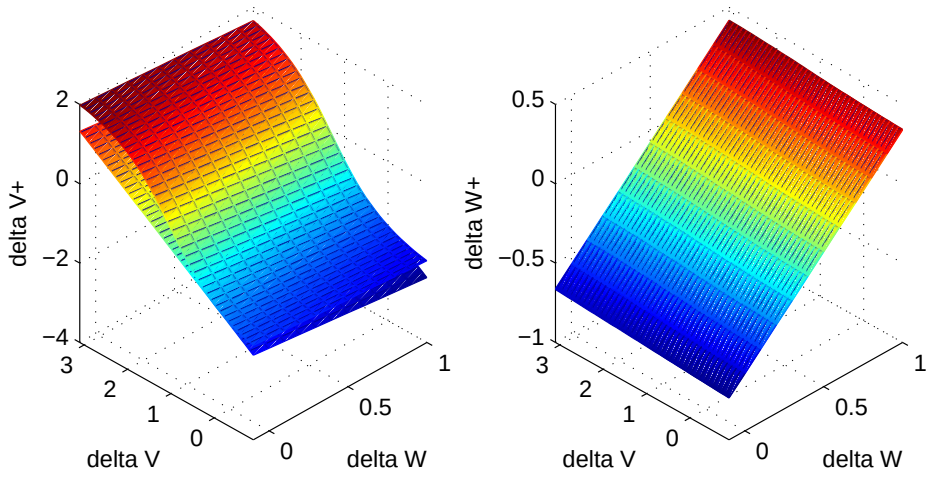


Figure 3.1: Linear Approximation versus Actual Mapping Surfaces for Timestep of $\tau = 0.1$. Though there is significant deviation between the two membrane potential surfaces, the gating variable surfaces are indistinguishable at this small timestep.

plane equation ($ax + by + cz + d = 0$) using the equilibrium point ((0,0) since we are dealing with disturbances from equilibrium) and two other nearby points (for instance, (0,0.01) and (0.01,0)) or to construct parametric equations (of the form $x = x_0 + at$) for the plane using numerical approximations of the derivatives. Either method yields the tangent plane at the equilibrium point, which ensures that the approximation will at least be stable directly on the equilibrium.

Fitting planar surfaces to the transition mapping surfaces results in a matrix M that defines the linear part of the system's response (for a given τ of 0.1):

$$\begin{bmatrix} \Delta V^+ \\ \Delta W^+ \end{bmatrix} = \begin{bmatrix} 0.791 & -0.443 \\ 0.018 & 0.980 \end{bmatrix} \begin{bmatrix} \Delta V \\ \Delta W \end{bmatrix}. \tag{3.5}$$

This results in a similar system to that of Equation (3.4). Figure 3.1 shows the relationship between the mapping surface and the linear approximation. Over the chosen timestep ($\tau = 0.1$), the gating variable (W) is strongly linear, having a sum of squared errors of only 0.030 and a maximum absolute error of only 0.011 between

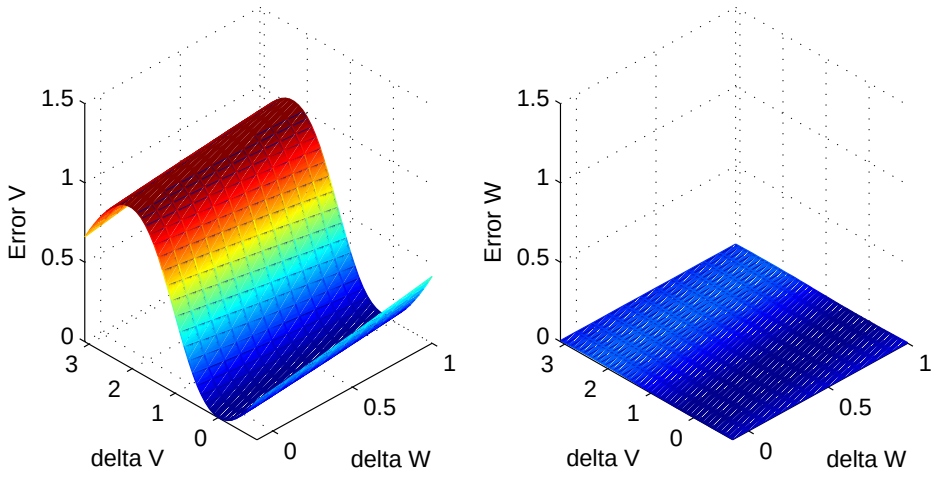


Figure 3.2: Linear Approximation versus Actual Mapping Error Surfaces

the linear approximation and the true surface. In fact, the approximation surface is indistinguishable from the true mapping surface. Figure 3.2, which shows the error between the linear mappings and the true solutions, supports this observation. By contrast, the membrane potential (V) mapping surface displays a lack of linearity away from the equilibrium point, especially at higher disturbances to the value of V, which correspond to the onset of an excitation wave. It has a significantly larger sum of squared errors (200.256) and maximum absolute error (1.030). The excitation wave pattern is the primary factor that produces errors between the true surface and the linear approximation. Such nonlinearity significantly reduces its ability to reproduce the system's response.

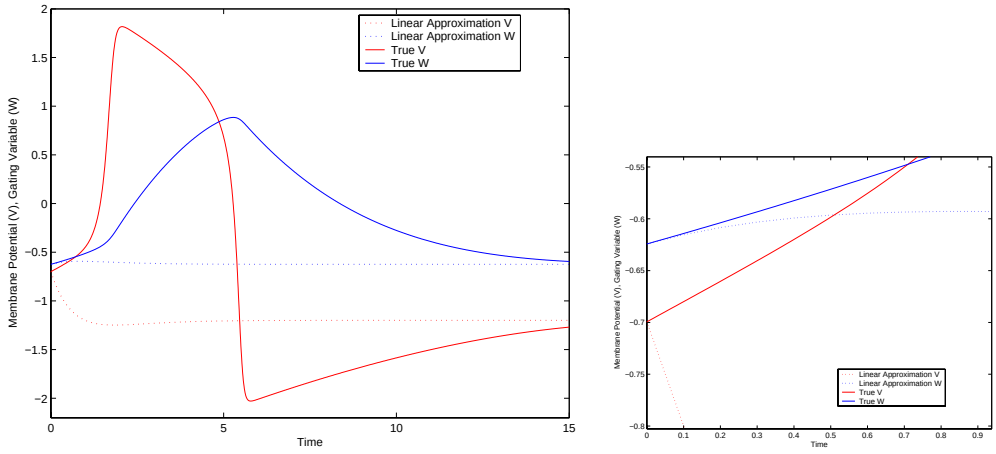
Examining the surfaces in a qualitative light shows us that a significant portion of the deviation between the system and its linearized version occurs in the behavior of the membrane potential (V) and corresponds to the excitation cycle, which is precisely where the form of Equations (2.1) and (2.2) and our previous analysis predict such behavior. Since the gating variable (W) behaves in a quite linear fashion, even during the excitation cycle, the only nonlinear behavior that is realized in the system is that

of the membrane potential during an excitation cycle. Thus, it will be this feature of the model that a successful optimization will capture accurately while still reducing computations.

3.1.3 Issues

Though our method of approximating the FitzHugh-Nagumo system by a linear relationship between its present and future values provides helpful information, it fails to capture the level of accuracy that would be required to substitute it for a full-fledged simulation of the differential equation system. Instead, if the linear approximation replaces the true model equations, the excitation cycles fail to occur. In fact, a calculation of the the eigenvalues of Equation (3.4) suggests that the system will converge quickly to equilibrium since the values (0.873 and 0.905) are less than one. Figure 3.3 compares a FitzHugh-Nagumo simulation to one performed with the linear approximation. The mapping for the gating variable (W) can be replaced in such a way, but this represents an insignificant computational savings, since Equation (2.2) and the W component of Equation (3.2) have similar forms. The membrane potential (V) cannot be captured accurately by this method of linear approximation and deviates immediately from the true curve.

Higher-order terms, such as H in Equation (3.1), can be added to increase the accuracy of the approximation, but in doing so, we add complications. First, a reliable fitting method to solve for the higher-order constants would be required. This fitting method must also provide surfaces that preserve the stability of the equilibrium point, so that the system does not display spontaneous excitations or drift away from the resting state. Second, increasing the count of terms in an approximation such as this reduces the potential gains in speed for which the approximation is developed. Therefore, other methods might be better suited to this problem.



(a) (b) (Magnification)

Figure 3.3: Linear Approximation versus FitzHugh-Nagumo Simulation. The systems are both initialized to the values $(V, W) = (-0.6994, -0.6243)$, which produces an excitation in the normal system.

3.2 Alternative Methods

3.2.1 Theory

Though the full details of the FitzHugh-Nagumo model cannot be captured accurately using linear approximation, we can take another approach to the problem of developing a relationship between the system's current and future values. We know that in the autonomous, single-cell model, variable values at particular times determine the state of the system at all future times. Therefore, we can consider this relationship as a nonlinear mapping between present values and future values, given a certain timestep. This relationship takes the form $(V, W) \mapsto \Phi(\tau, V, W)$. We can take advantage of this mapping relationship to perform high-accuracy calculations of the system's behavior off-line. During a simulation, these pre-created samples can be interpolated to obtain an approximation of the system's future values at timesteps significantly larger than those required for stability and accuracy in methods such as Runge-Kutta.

Rather than approximating the response of the system, we can generate and store

discrete samples of the system's behavior in advance and use these samples to predict arbitrary behaviors during a simulation. Unlike the method of linear approximation, in which the mapping must be nearly planar to be approximated accurately, this approach provides more flexibility in the shape of the predicted mapping surface. It also allows for control of the distribution of sample data, which can be used to increase accuracy in certain regions of the input space.

3.2.2 Methods

We investigate two algorithms that can be used to interpolate the off-line sample data: the nearest neighbor method and locally weighted regression. These approaches have associated storage requirements to preserve the samples, and in the case of locally weighted regression, advanced computational requirements to validate the sample data.

Nearest Neighbor Method

As one of the simplest available interpolation methods, the nearest neighbor method is a straightforward algorithm that predicts an output by proposing that it is equal to the output of the single nearest sample. In its simplest form, the algorithm calculates the distance between the requested query point and each input sample, picking out the closest sample. The distance is defined according to a particular metric (often Euclidean distance) that is chosen based on the problem. The output of this sample is returned as the output of the requested query point. The algorithm is specified in psuedo-code as follows:

```
NearestNeighbor(q,SampleList)
{
   dist_min = distance_metric(q,SampleList[0].x);
   index_min = 0;
```

```
    FOR EACH i ∈ [1,SampleList.length()-1],
    {
        dist = distance_metric(q,SampleList[i].x);

        IF dist < dist_min,
        {
            dist_min = dist;
            index_min = i;
        }
    }

    return SampleList[index_min].x;
}
```

The nearest neighbor method only considers the closest sample point, as shown in Figure 3.4. Therefore, it does not require concentrated data in regions of roughly constant response. Conversely, the quality of its predictions benefits markedly from more concentrated data in regions of sharp change.

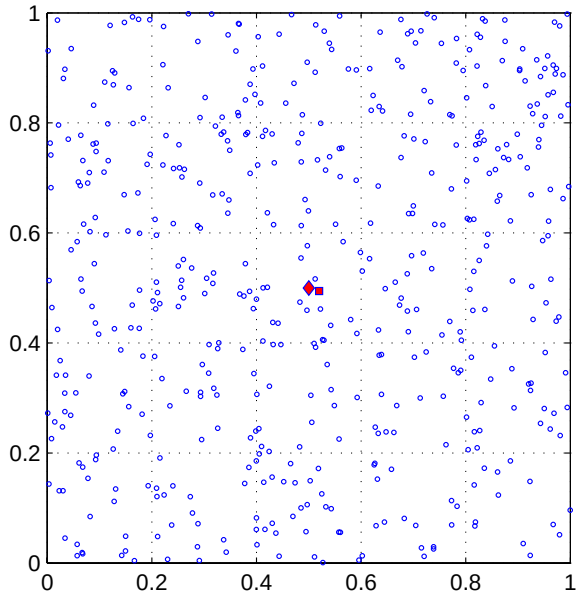


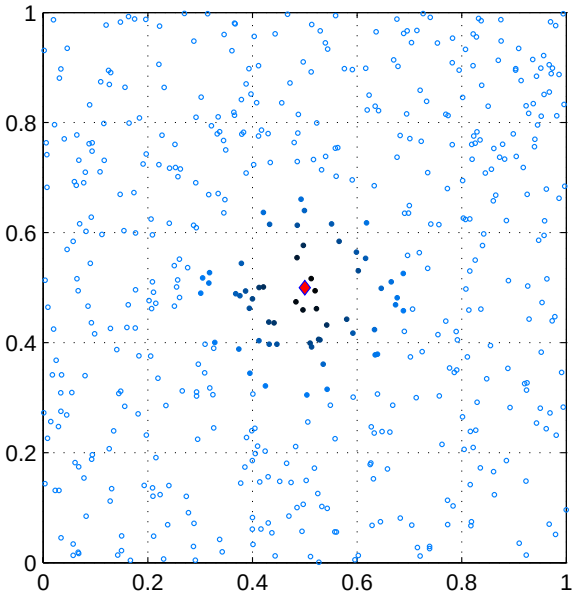
Figure 3.4: Nearest Neighbor Point Selection (query point $(0.5, 0.5)$, marked by the diamond). The nearest neighbor method selects the closest point in space (marked by the filled square) and returns its output as the output of the query point.

Locally Weighted Regression

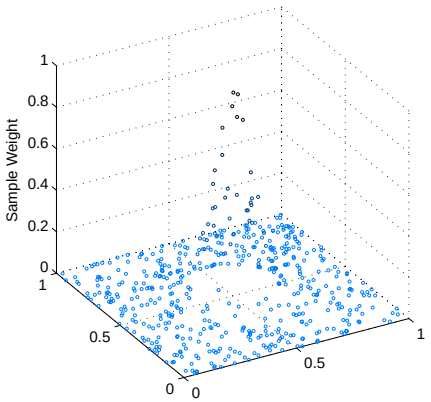
In contrast to the simplicity of the nearest neighbor method, locally weighted regression [30] is a matrix-based method that estimates the output by calculating a weighted average of the outputs of several nearby samples. The sample weights are calculated based on an exponential metric of the form $e^{-\frac{h}{2\sigma^2}d^2}$ (for one dimension) where h is a constant determining the relative size of the neighborhood, σ is the standard deviation of the samples, and d is the distance from the query point to the sample point. The algorithm is specified in psuedo-code as follows (where n is the number of input dimensions; p is the number of sample points; D (n-by-n), W (p-by-p), and X (p-by-$n+1$) are matrices; and D and W are diagonal):

```
LWRegression(h,q,SampleList)          % adapted from [30]
{
   p = SampleList.length();
   D = h · diag(1/σ_1^2, 1/σ_2^2, ..., 1/σ_n^2);

   FOR EACH sample_i ∈ SampleList,
   {
      w_ii = e^{-1/2 (SampleList[i].x - q)^T D (SampleList[i].x - q)};
      xd_i = ((SampleList[i].x - q)^T  1)^T;
   }

   X = (xd_1, xd_2, xd_3, ..., xd_p)^T;
   y = (SampleList[1].y, SampleList[2].y, ..., SampleList[p].y)^T;
   return (X^T W X)^{-1} X^T W y;
}
```

In locally weighted regression, weights are calculated based on the exponential metric to appear as the example weights in Figure 3.5. In the typical case, many points have a weight very near zero, and can be excluded from further calculations, effectively reducing the working sample data set to a fraction of its true size. For evenly-distributed sample data and a smooth, continuous mapping, considering a count of local sample points on the order of 10 will produce an accurate prediction.



(a)



(b)

Figure 3.5: Locally Weighted Regression Point Weighting (query point $(0.5, 0.5)$, marked by the diamond). Locally weighted regression weights surrounding points by an exponential metric with a Gaussian profile. Close by points that have significant weights (e.g., greater than 0.05, marked by filled circles) are considered in the algorithm, whereas others can be ignored. Sample points on any given circle centered at the query point will have equal weights.

The constant h is unique to the sample set of a given problem, and is determined by a process called cross validation. For a given h value, *cross validation* processes the sample data set to determine the error between each sample's output and the locally weighted regression output for that sample's input, ignoring that sample. Essentially, the algorithm gauges the inaccuracy of predicting its own sample data. It then outputs the corresponding sum of squared errors, which can be used in turn to determine the optimal h value for a given data set. The optimal h value gives the smallest sum of squared errors, and correlates to the relative amount of input space considered in interpolating a single output. The cross validation algorithm is specified in psuedo-code as follows:

```
CrossValidate(h,SampleList)          % adapted from [30]
{
   sse = 0;

   FOR EACH sample_i ∈ SampleList,
   {
      sample_temp = sample_i;
      SampleList.remove(i);
      sse += sample_temp.y − LWRegression(h,sample_temp.x, SampleList);
      SampleList.add(sample_temp);
   }

   return sse;
}
```

3.2.3 Progression Shapes

In both of the aforementioned interpolation methods, we can represent the mapping $(V(t), W(t)) \rightarrow (V(t + \triangle t), W(t + \triangle t))$ (given a time passage τ) as a pair of three-dimensional surface that performs the same mapping. See Figure 3.6. In addition to the importance of their data in the interpolation methods, the shapes of these surfaces represent the time progression of the system's variables. Regions of sharp

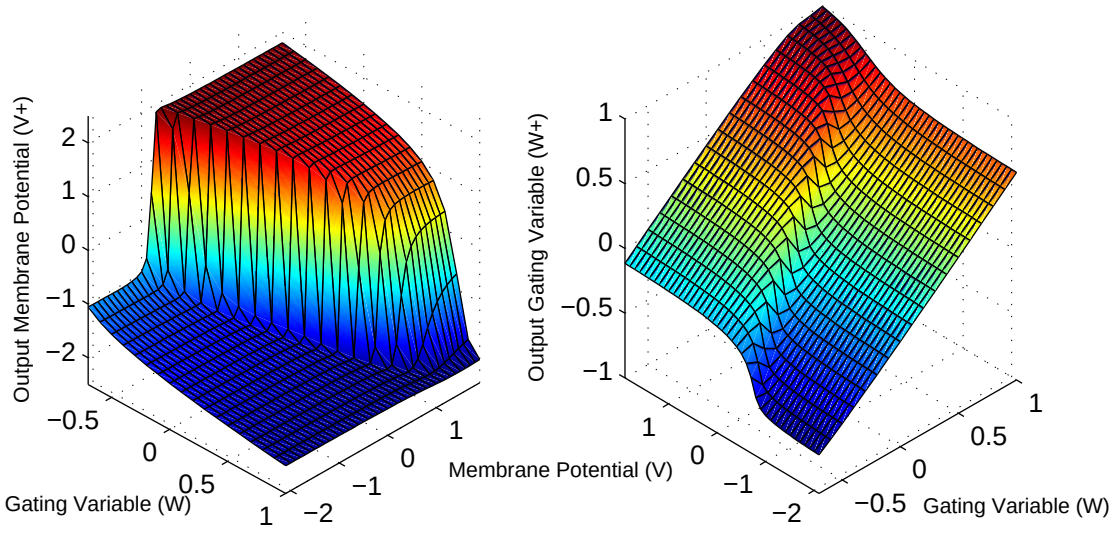


Figure 3.6: Surfaces Mapping Input Values to Output Values ($\tau = 1.0$)

change (notably, on the diagonal of the input plane) denote value pairs that can evolve distinctly differently if subjected to small perturbations. In contrast, there are also regions that display nearly constant or nearly linear output, in which one variable strongly determines a future output value, and the value of the other variable is only weakly significant.

Locally weighted regression and the nearest neighbor method can be used to reconstruct these shapes from sample data, giving an impression of the quality of their output. Figures 3.8 and 3.9 were created by using the nearest neighbor and locally weighted regression algorithms to reconstruct a surface by placing query points on a uniform grid in the input space, similar to the respective mappings in Figure 3.6. Each method analyzed the same set of 1024 sample points, which was created with a uniform random distribution in the input space rectangle $V \in [-2.1, 1.9], W \in [-0.7, 1.0]$. This input space area corresponds to the region though which the typical excitation wave travels. For error analysis, refer to Section 3.2.5.

These methods also preserve the stability of the equilibrium point, a requirement

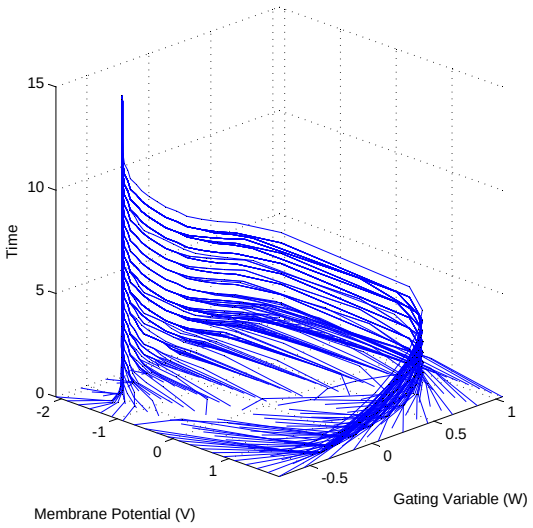


Figure 3.7: Equilibrium Stability of Nearest Neighbor Interpolations. Each line represents a distinct system mapped forward in time ($\tau = 0.5$) by the nearest neighbor method from a uniform grid of initial values.

we noted earlier. Figure 3.7 shows the results of continual mapping through a 4096-sample data set by the nearest neighbor method, with $\tau = 0.5$ and a time length of 15.0. A set of systems initiated ($t = 0.0$) on a grid of reasonable values ($V \in [-2.1, 1.9], W \in [-0.75, 1.05]$) evolves toward equilibrium on a path approximating the one determined by the FitzHugh-Nagumo equations, and settles to the equilibrium value over time. This stability is contingent upon the count of samples used to simulate the system, since a small enough sample data set will clearly introduce errors large enough to spontaneously stimulate excitations from what should be stable equilibrium.

3.2.4 Sampling Distribution

In both the nearest neighbor method and locally weighted regression, the exact distribution and size of the sample data in input space is an important factor in the quality of predictions and the computational speed of the algorithms.

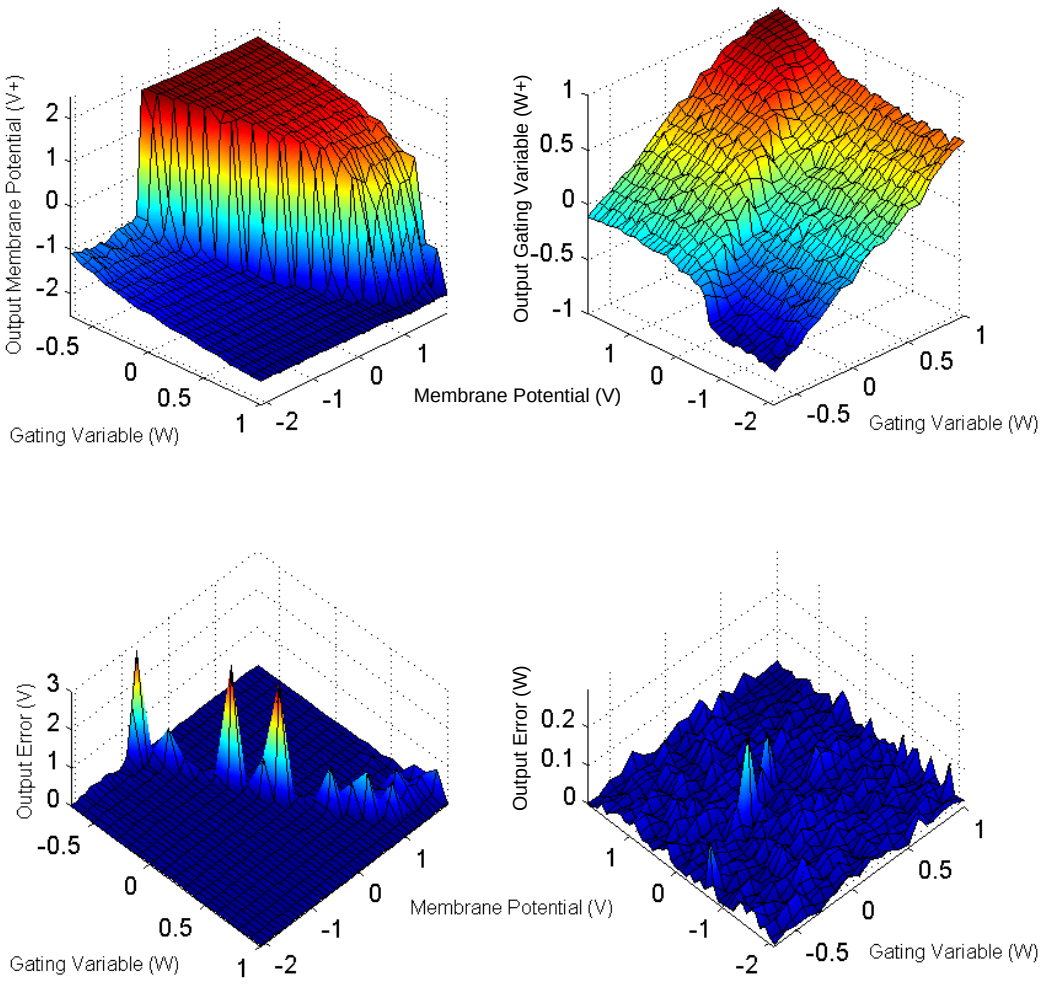


Figure 3.8: Nearest Neighbor Reconstruction of Mapping Surfaces

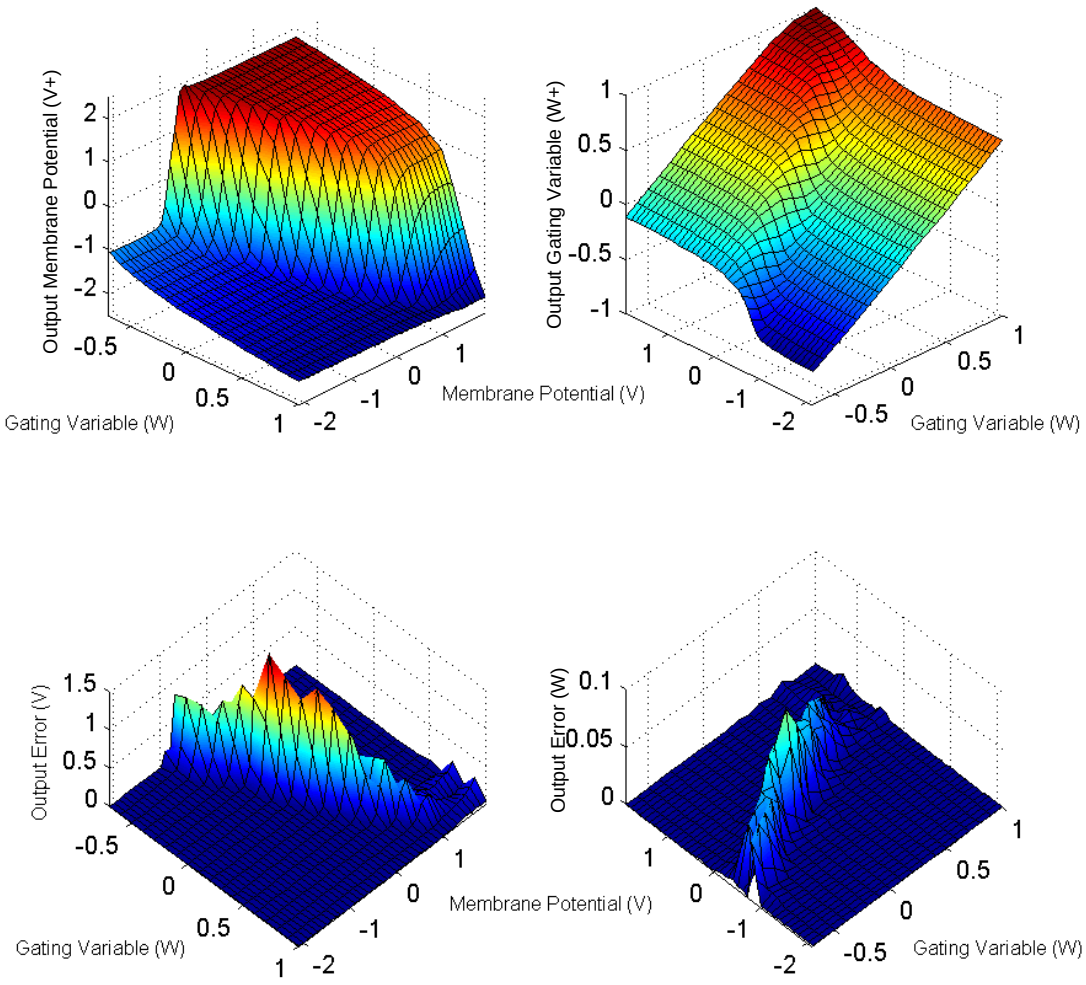


Figure 3.9: Locally Weighted Regression Reconstruction of Mapping Surfaces

Algorithm Speed

In both algorithms, increased concentration of sample data in sharply-changing regions can provide the algorithms with more information to reach their output values. In the nearest neighbor method, highly concentrated data poses no problem for the algorithm, and will reliably produce greater accuracy as the average requested point will have nearer neighbors. However, locally weighted regression does not deal well with concentrated sample data, beyond a certain point. Since the algorithm depends on a tuning process (cross validation) to define a global relative neighborhood size for consideration, unevenly distributed data can cause this neighborhood to include markedly different quantities of sample points, depending on the requested point. For instance, a neighborhood size that includes 20 points in one section of input space may include 100 points in another area of input space with more concentrated sample data. This seriously hinders the algorithm's speed when producing interpolations in these concentrated regions.

Interpolation Accuracy

Beyond the speed factor, the distribution of sample points also has implications in the accuracy of the two algorithms' predictions. As a matrix-based method that relies on a matrix inversion, locally weighted regression begins to suffer from numerical problems caused by nearly singular matrices as samples become concentrated in particular areas of input space. This is due to the difference in total sample weight for a certain neighborhood size between densely-sampled areas and sparsely-sampled ones. A neighborhood size that produces accurate results in densely sampled areas, will make it likely that the largest weight in a sparsely sampled area will be very small, producing a nearly singular matrix and making the algorithm's output invalid. By contrast, the nearest neighbor method does not suffer from such numerical problems.

In Figures 3.8 and 3.9, we can see that both algorithms produce significant error

in the regions where the ratio of the number of sample points to the rates of change of the mapping output is small. In these areas, there is insufficient data representative of the particular output data to produce an accurate output. In simpler terms, the query point is too distant from any sample to provide an accurate output.

One potential solution to address the aforementioned inaccuracies is to use psuedo-random point distributions that ensure a more even distribution of sample points. Rather than using a uniformly random distribution, we can apply Halton or Hammersley points [15] to ensure that no region of input space is far away from any sample point. These points are so-called *low dispersion points*, which indicates that they follow a distribution that avoids spatial regions that can be filled by large spheres that contain no samples. Halton and Hammersley points are given by sequences that follow a deterministic distribution that has lower dispersion than uniform random sampling. Both sequences use representations of the number sequence $i = 0, 1, 2, ...$ in prime bases to distribute sample points. Denoting p_n as the nth prime number, the digits (a_k) of these representations are given as

$$i = a_0 + a_1\ p_n + a_2\ p_n^2 + \cdots \tag{3.6}$$

Then, individual coordinates in the ith sample are given by the digits (a_k) in the base p_n representation of i as

$$r_{p_n}(i) = \frac{a_0}{p_n} + \frac{a_1}{p_n^2} + \frac{a_2}{p_n^3} + \cdots \tag{3.7}$$

Complete sample coordinates in the two sequences are given by a set of d of these numbers $r_{p_n}(i) \in [0, 1]$, so that each sample coordinate belongs in $\Re^d$:

$$\text{Halton: } (r_{p_1}(i), r_{p_2}(i), \cdots, r_{p_d}(i)), \tag{3.8}$$

$$\text{Hammersley: } \left(\frac{i}{N}, r_{p_1}(i), r_{p_2}(i), \cdots, r_{p_{d-1}}(i)\right), \tag{3.9}$$

where d is the dimension of the required sample points. Thus, the Hammersley sequence is a modification of the Halton sequence in which the first coordinate is given by $\frac{i}{N}$, where N is the count of samples. Therefore, the Halton sequence has the advantage over the Hammersley sequence of not requiring an advance definition of the sample count. The Halton sequence requires a calculation of the first d primes, whereas the Hammersley sequence needs the first $d-1$ primes.

Using the example mapping surface $z = \cos(2x + 2y)$, we tested the accuracy of locally weighted regression by reconstructing a rectangular grid from differently-sized sample sets. This procedure is the same as that used previously to reconstruct the FitzHugh-Nagumo mapping surfaces in Figure 3.9. Then, we cross-validated the sample data and compared the sum of squared errors between the true mapping surface and the locally weighted regression mapping surface. The optimal values of the neighborhood sizing parameter h are shown in Figure 3.10, and the corresponding sum of squared errors are shown in Figure 3.11. The tests were run using sample counts of 128, 256, 512, 1024, 2048, 4096, and 8192. Additionally, the number of neighboring sample points considered by the algorithm for the optimal h value in the tests is shown in Figure 3.12.

These tests bear out several interesting conclusions about the pairing between the aforementioned sample generation methods and the locally weighted regression algorithm. As we expect, there is a strong, continuing downward trend in error as the sample size is increased. Similarly, there is a direct relationship between increasing sample count and increasing optimal h value, corresponding to decreasing neighborhood size. As the sample count increases and samples become more concentrated in input space, the neighborhood size required for an accurate output will shrink. The nature of this relationship in our tests is predictable enough to provide a useable h value for other testing, but full cross validation of the particular sample data set is still best for simulation use. As a consequence of the relationship between sample

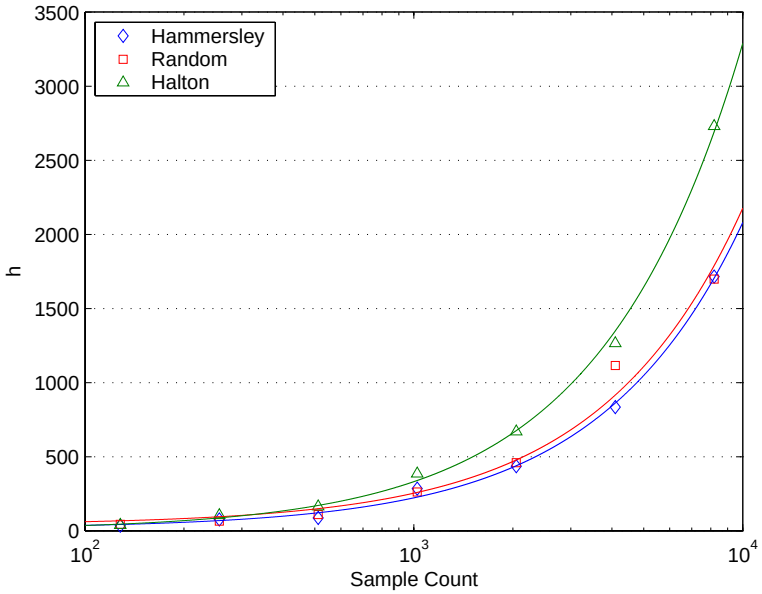


Figure 3.10: Neighborhood Sizing Parameter h versus Sample Generation Method

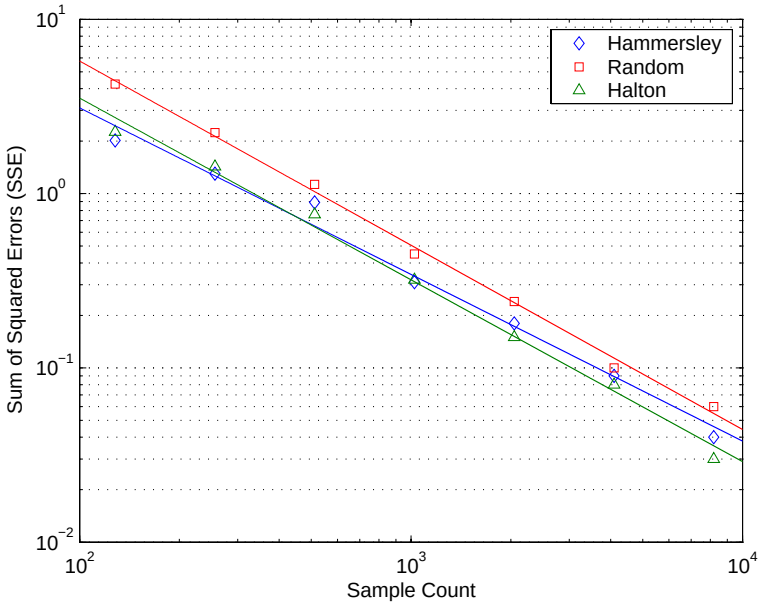


Figure 3.11: Sum of Squared Error (SSE) versus Sample Generation Method

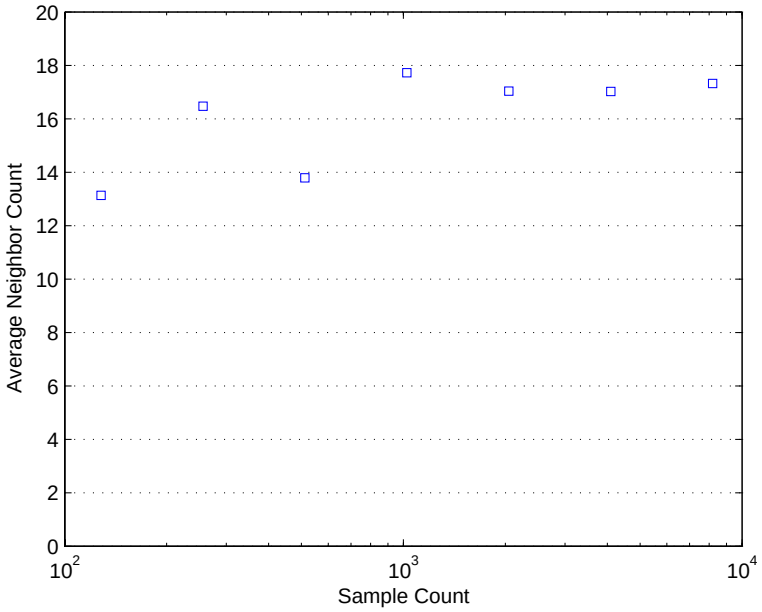


Figure 3.12: Average Neighbor Count for Optimal Parameter h versus Sampling Size

count and h value, the average number of neighbors considered is a roughly constant quantity. Therefore, we can further see the potential repercussions of uneven distribution of samples. Since h is a single constant for each sample data set, query points at different locations in a non-uniform sample data set would have a large variance in the count of their neighbors. The overall conclusion from these tests is that Halton points tend to have a more optimal distribution to produce accurate locally weighted regression results for all but the smallest sample counts. The magnitude of the benefit compared to uniformly random points is small, and Halton points have a greater computational requirement (taking roughly 80 times longer to generate a sample set in our implementation). However, the sampling computations are performed off-line, so Halton points have an advantage over random points, though the latter are useful for testing purposes in which accuracy is less vital.

3.2.5 Benefits and Disadvantages

Both the nearest neighbor method and locally weighted regression are capable of providing significantly more accurate output in capturing the input/output mapping in the FitzHugh-Nagumo system than a linear approximation of these mappings. In both algorithms, the regions of notable error are linked strongly with regions of sharp change in the mappings, which can be identified in advance. Nearest neighbor has a marked advantage in its flexibility to allow non-uniform sampling distributions, because concentrating samples within specific areas can significantly increase the accuracy of the algorithm. Figure 3.13 shows a comparison between the ordinary uniformly-distributed random sampling (right column) and a controlled sampling distribution in which samples are placed with probability scaled to their gradient (left column). The exact distribution of sample points is depicted in the bottom figures. This data was produced by considering the gradients of the mapping $(V, W) \rightarrow V^+$, which does not reduce the errors of the other mapping $(V, W) \rightarrow W^+$. It is straightforward to use separate sample sets to capture the two mappings individually in cases when non-uniform sampling is used, but this requires separate queries to each data set.

When uniform sampling is applied, locally weighted regression is the preferred choice, since it produces a smoother mapping output than the nearest neighbor method. For a particular sample count, locally weighted regression also reproduces mappings with smaller and more predictably localized error magnitudes. Overall, the reproduced mappings in Figures 3.8 and 3.9 have a sum of squared errors of 33.53 for the locally weighted regression surface and 41.41 for the nearest neighbor surface. Additionally, nearest neighbor produced a larger maximum absolute error of 3.10 (locally weighted regression produced no errors greater in magnitude than 1.42). In our implementations, locally weighted regression also displayed a small speed advantage, running 106% faster than the nearest neighbor method at a sample count of 2048.

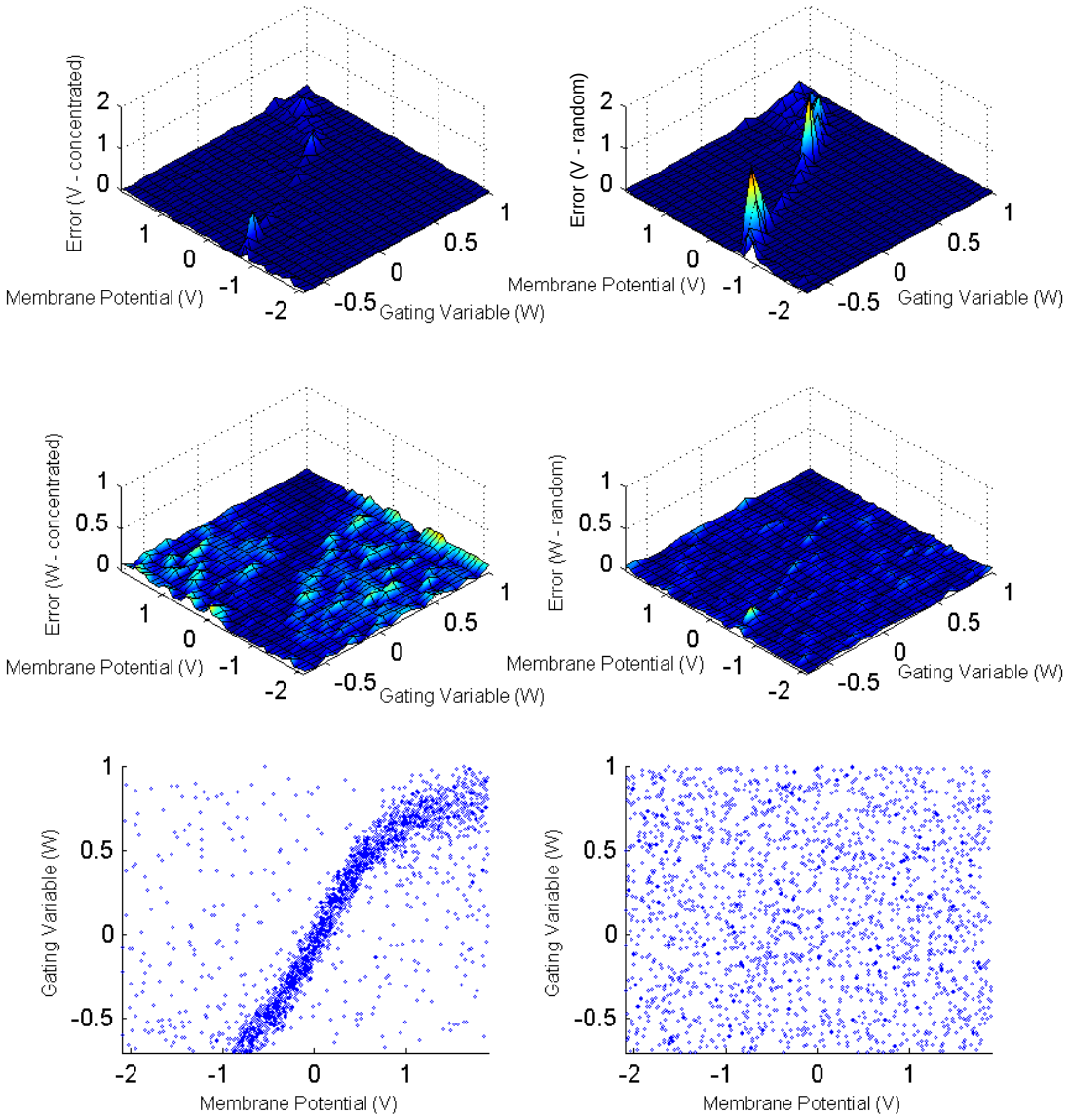


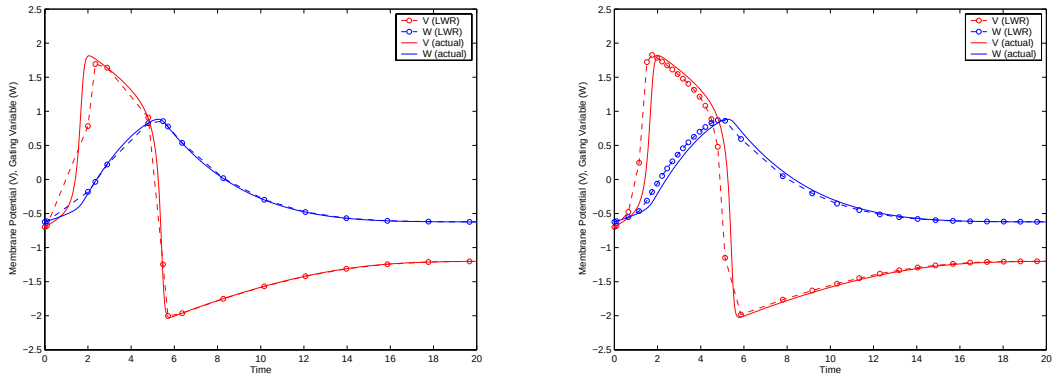
Figure 3.13: Nearest Neighbor Sampling Distribution Comparison (Left - Controlled Sampling Distribution; Right - Uniformly Random Sampling Distribution). The lower scatterplots show the respective distributions of sample data in the input space.

The nearest neighbor method and locally weighted regression also have established methods for optimizing their stored data for efficient retrieval [19, 5]. Previous research [30] shows that such methods can be applied to real-time problems, within the bounds of computing resources and problem scope.

3.3 Time-Flexible Mapping

Up to this point, we have discussed mappings in the FitzHugh-Nagumo system of the form $(V, W) \rightarrow (V^+, W^+)$, given a certain time passage τ. There are two methods of extending this idea. First, we can develop another, higher-dimensional mapping of the form $(V, W, \tau) \rightarrow (V^+, W^+)$. Inserting the time passage τ into the mapping increases the dimension from two to three, but it can then be used to produce output values for arbitrary times. Second, we can utilize one or more mappings with varied values of τ, calling them $F(\tau)$, to selectively advance a simulation of the FitzHugh-Nagumo system by those timesteps. For instance, given initial values of the system, the output after 3.7 time units could be characterized by applying $F(1.0)$ three times and $F(0.7)$ once. In cases where the time passage does not correspond to any stacking of the available mappings, they can be used to advance to a point near the final time, and another method (e.g., Runge-Kutta) can be used to reach the final time.

This development of the mapping idea allows simulations using one of the interpolation techniques to advance at selective timesteps, rather than one predetermined one (i.e., $\tau = 1.0$). Specific time values of the system can thus be provided. This does not come without computational consequences: a higher-dimensional sample set requires more points to achieve the same level of accuracy, increasing both computation times and storage requirements. In this case, the increase is somewhat staggering: to match the accuracy of a two-dimensional sample set of 1024 (32^2) points, we must provide 32,768 (32^3) points.



(a) Inversely Proportional to V Derivative

(b) Inversely Proportional to V Value

Figure 3.14: Time-Flexible Mapping Using Two Timestep Choice Methods. The first chooses timesteps for advancement that are inversely proportional to the derivative of V, whereas the second chooses timesteps that are inversely proportional to the value of V.

To realistically apply this idea, we must have some method of choosing the timestep. In the single-cell setting, a useful way to motivate this choice is by degree of change. In other words, we choose smaller timesteps when the model is changing rapidly (at the initiation and drop of the excitation cycle) and larger ones when it is relatively constant (at or close to equilibrium). Another method is to choose the timestep based directly on the value of one of the variables (i.e., larger timesteps for lower values of V and smaller ones for higher values of V, so the timestep is reduced during excitation cycles). Both methods are similar to variable timestep methods in common differential equation solvers [28], but they retain stability over large timesteps. Figure 3.14 compares these two methods, using a 32,768-point sample set that is randomly-distributed and bounded by $V \in [-2.1, 1.9]$, $W \in [-0.7, 1.0]$, and $\tau \in [0.0, 2.0]$. This sample set covers all realistic values for the system, and allows any timestep less than 2.0 time units in one execution of the interpolation algorithm (locally weighted regression in this case).

Both methods have distinctly different allocations of timesteps. Because of this characteristic, the derivative method required 0.19 seconds to complete the simula-

tion, while the value method required 0.47 seconds. Comparatively, the generation of the sample data required 3.37 minutes. When the timestep is chosen to be inversely proportional to V's derivative, it is overall more accurate than the second method, but it takes some time to "catch up" to the initial rise of the excitation cycle. If the timestep is instead chosen to be inversely proportional to the value of V, the algorithm more correctly reproduces the waveform shape, but the excitation cycle is simulated at an earlier time than is accurate. These two methods are by no means the only ways to pick timesteps and are likely not the most efficient ones.

Chapter 4

Multi–Cell Network Analysis

In the previous chapters, we have examined the FitzHugh-Nagumo model in detail in the context of a single cell. However, in order to progress toward a practical simulation of the heart, techniques to simulate the system in a multiple-cell environment must be available. This chapter establishes terminology and experimental methodology for an examination of the model in a discrete, multi-cell network.

4.1 Diffusion

The important feature of cellular interaction is the propagation of potential waves throughout cardiac tissue. Among other factors, this is made possible by the flow of various ions (sodium, potassium, calcium, etc.) throughout the cardiac tissue. To model the diffusion that allows wave propagation to occur in the system, the term $\frac{\partial^2 V}{\partial x^2}$ must be added to Equation (2.1) given previously, resulting in a modified system:

$$\frac{\partial V}{\partial t} = \frac{\partial^2 V}{\partial x^2} + \frac{1}{\epsilon}\left(V - \frac{V^3}{3} - W\right), \tag{4.1}$$

$$\frac{\partial W}{\partial t} = \epsilon(V - \gamma W + \beta). \tag{4.2}$$

Thus, a sufficient difference in potential between adjacent cells can produce an action

potential that flows across a system. In a system with multiple spatial dimensions (2D or 3D), the $\frac{\partial^2 V}{\partial x^2}$ is replaced with the corresponding $\nabla \cdot (D \cdot \nabla V)$ term, in which D is a matrix representing the diffusion coefficients for the extracellular space.

The diffusion of action potentials, and the associated behaviors of excitation and return to equilibrium of individual cells in a network, is responsible for the normal mechanical contraction of the heart [26]. However, under certain conditions, diffusion can also bring about abnormal, self-sustaining waves which are tied to heart attacks and other ailments. The simplest of these wave types is the spiral wave, which has been shown to be reproducible using the FitzHugh-Nagumo model dynamics [29]. To understand the ways in which these abnormal waves are established, we must first understand how the model responds to diffusive inputs.

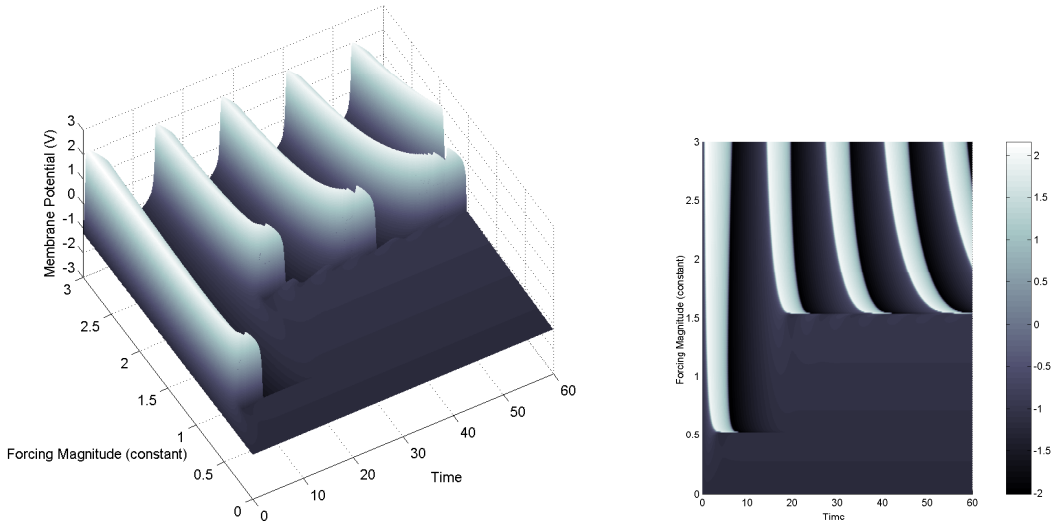
4.2 Forcing Analogue

We begin our analysis by noting that the $\frac{\partial^2 V}{\partial x^2}$ diffusion term can be replaced with any function to give the system's response to arbitrary stimulus. This is analogous to the concept of forcing in the analysis of ordinary differential equations. We can then redefine the model equations in terms of an input/output system (where $f(t)$ is the forcing function and $y(t)$ is the output):

$$\frac{\partial V}{\partial t} = f(t) + \frac{1}{\epsilon}\left(V - \frac{V^3}{3} - W\right), \tag{4.3}$$

$$y(t) = V(t). \tag{4.4}$$

Thus, we can apply standard analysis techniques to capture the system's response to various diffusive inputs. For example, we can use Fourier analysis to show the system's nonlinear frequency response.



(a) (b)

Figure 4.1: Two Views of the System Response to the Forcing Function $f(t) = C$

4.2.1 Applicable Forcing Functions

The simplest forcing functions we can apply to the FitzHugh-Nagumo system are of the form $f(t) = C$ (constant) and $f(t) = \sum_{i=1}^{\infty} A\delta(t - Pi)$ (periodic impulse train). Given sufficient amplitude, both functions will produce periodic excitation behavior in the FitzHugh-Nagumo system. Figure 4.1 shows the system's response to various magnitudes of the constant forcing function.

In the situation of a constant forcing function, we can observe two important threshold values. First, a constant magnitude less than approximately 0.53 is not capable of producing any excitation wave in the FitzHugh-Nagumo system. Second, a constant magnitude of forcing less than approximately 1.54 creates an intermediate state in which the system experiences one initial excitation wave and no further activity. This intermediate state is a consequence of the constant forcing's impact on the gating variable (W), which "holds" the gating variable above its typical equilibrium value and prevents the normal relaxation after an excitation wave. Thus, only the first excitation wave is permitted in this system state. Additionally, we can observe

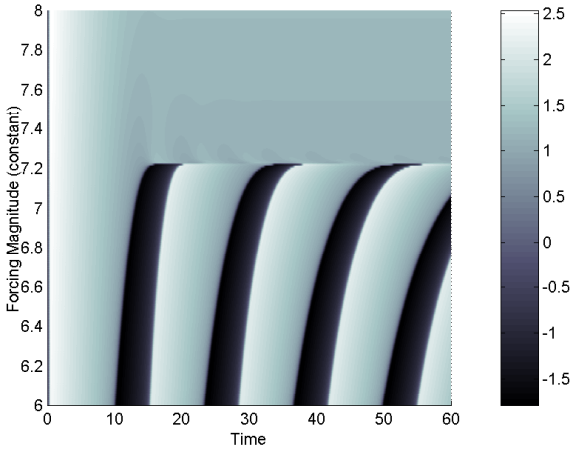
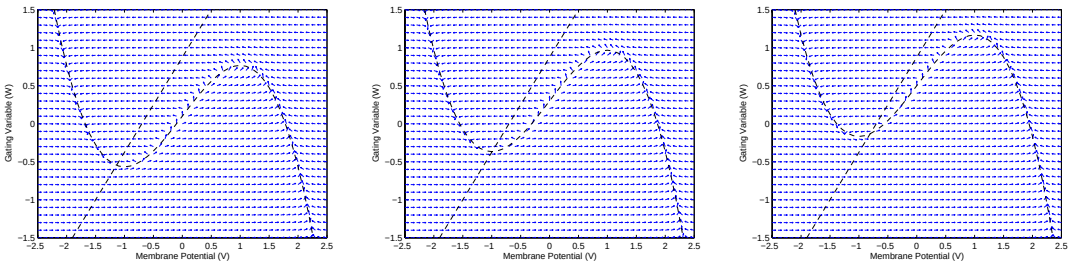


Figure 4.2: System Response to Large-Scale Constant Forcing

a certain limit behavior of the periodicity of the excitations as the magnitude of the constant forcing function increases. This data suggests the existence of a maximum frequency capability of the FitzHugh-Nagumo system to produce excitation waves. In fact, there is an additional threshold in this situation at very high forcing magnitudes. As shown in Figure 4.2, if the forcing magnitude moves beyond approximately 7.23, the system transitions into a state similar to the previous intermediate state (forcing magnitude $\in [0.53, 1.54]$), but the gating variable (W) is driven to an unrealistic positive value. In this state, both the membrane potential (V) and the gating variable are "held" to a new forced equilibrium value. At a forcing magnitude of 7.23, this forced equilibrium is at $(V, W) = (1.03, 2.11)$.

It is interesting to note that the quantitative shifts in Figure 4.1 can be correlated with the changing appearance of the system's phase plane under each magnitude of constant forcing. The overall effect of the constant forcing function is to shift the cubic nullcline along the gating variable axis in the phase plane. By observing snapshots of the phase plane (for $f(t) = 0.5, 1.5, 2.5$, in Figure 4.3), we can see that the movement of the cubic nullcline provides the same conditions for periodicity that we observed in Chapter 2 when the forcing magnitude is approximately equal to 1.5.

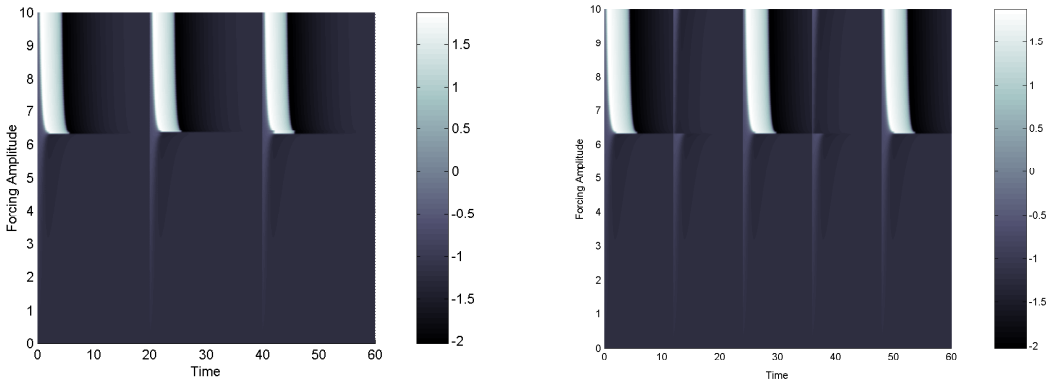


(a) $f(t) = 0.5$ (b) $f(t) = 1.5$ (c) $f(t) = 2.5$

Figure 4.3: Phase Planes under the Forcing Function $f(t) = C$

This corresponds to the threshold between single excitation and periodic excitation in Figure 4.1.

In contrast to the constant forcing function $f(t) = C$, the periodic impulse train $f(t) = \sum_{i=1}^{\infty} A\delta(t - Pi)$ can produce repeated excitation waves without affecting the state of the system in between cycles. Numerically, we realize the δ function by adding the amplitude A of the function at specific timesteps. The period P is set as a multiple of the simulation timestep, so that the occurrences of δ functions coincide with calculated timesteps. Again, we can vary the amplitude A of the forcing function, but we can also vary the periodicity of the function (period $= P$). Figure 4.4 presents a comparison between two periods of impulse-train forcing ($P = 20.0, 12.0$) for varying amplitudes. This reveals another noteworthy aspect of the FitzHugh-Nagumo system: if an excitation wave has not terminated completely, the system is resistant to the onset of another excitation. This is visible in Figure 4.4(b), in which the impulses occur too often for each to cause a distinct excitation cycle. When such impulses occur following the excitation, the gating variable resists a further excitation. As small impulses do not create excitations, the threshold phenomenon is again visible.

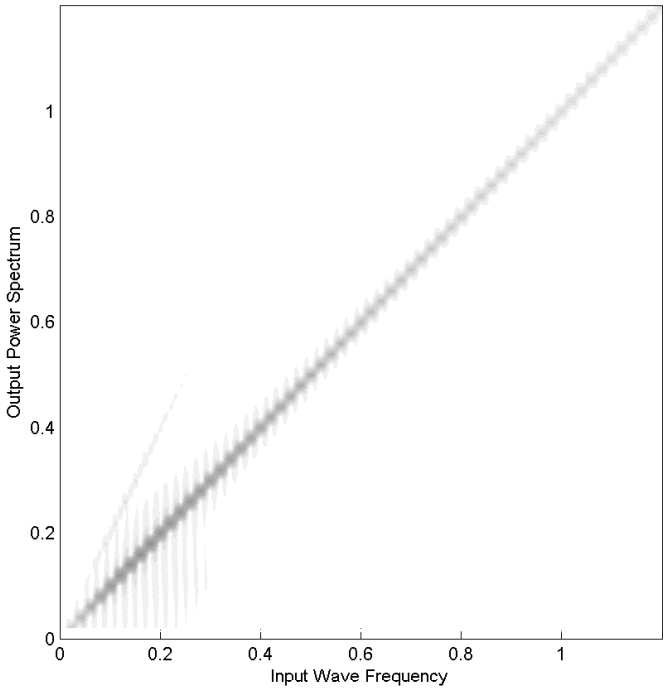


(a) $P = 20.0$ (b) $P = 12.0$

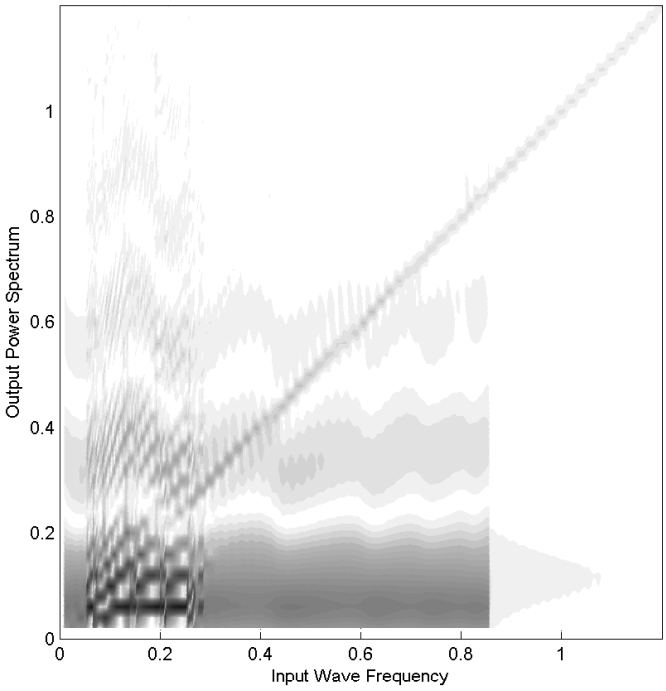
Figure 4.4: System Response to the Forcing Function $f(t) = \sum_{i=1}^{\infty} A\delta(t - Pi)$ (where $A = 2.0$)

4.3 Fourier Analysis

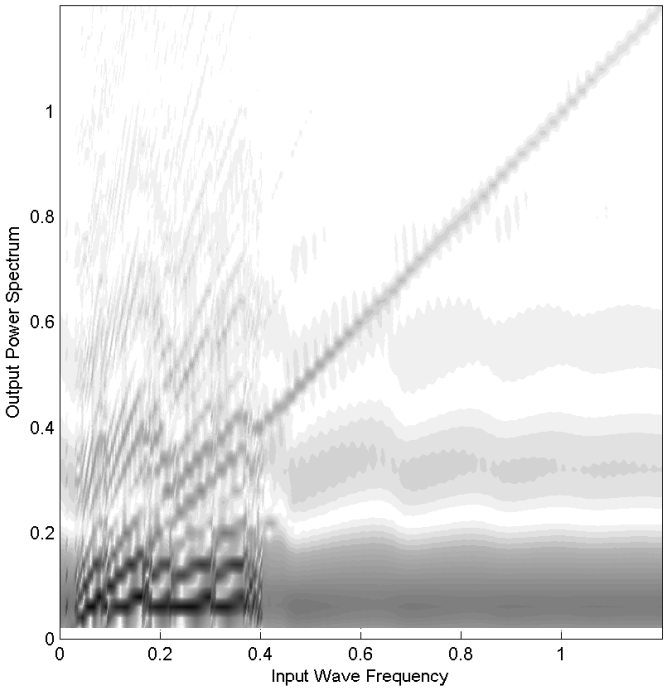
We can further develop the analysis of diffusion in the FitzHugh-Nagumo sytem by using Fourier analysis to characterize its nonlinear frequency response to sinusoidal waveforms. Figure 4.5 provides power spectrum plots generated by MATLAB's `fft` (Fast Fourier Transform) method that relate the frequency of an input sinusoidal diffusion term (of the form $f(t) = A\cos(2\pi f_0 t)$) to the output's frequency components for six different amplitudes $(0.5, 1.0, 1.5, 2.0, 2.5, 3.0)$. The constant component of each output is excluded.


Output Power Spectrum
Input Wave Frequency
0
0.2
0.4
0.6
0.8
1

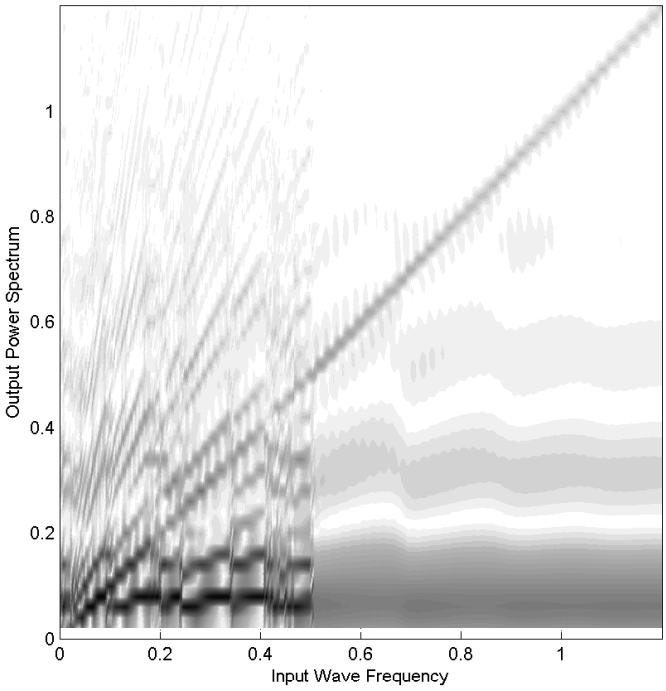
(a) $A = 0.5$


Output Power Spectrum
Input Wave Frequency
0
0.2
0.4
0.6
0.8
1

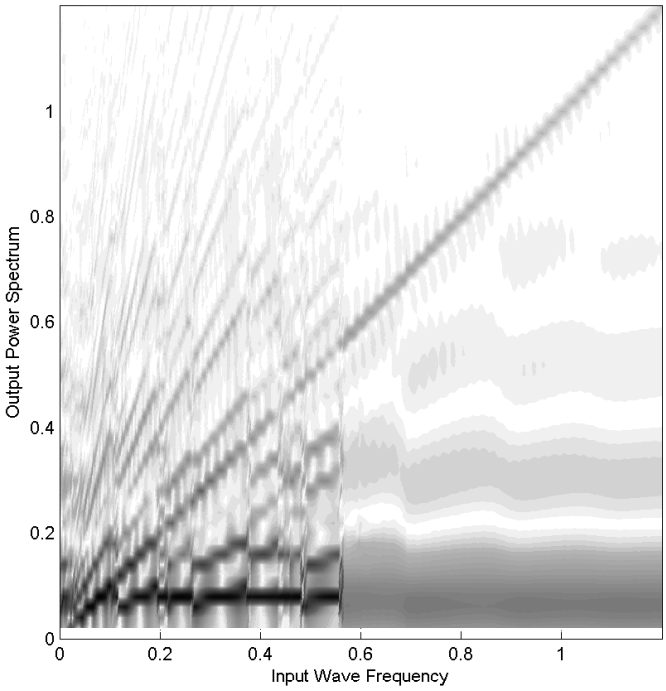
(b) $A = 1.0$


Output Power Spectrum
Input Wave Frequency
0
0.2
0.4
0.6
0.8
1

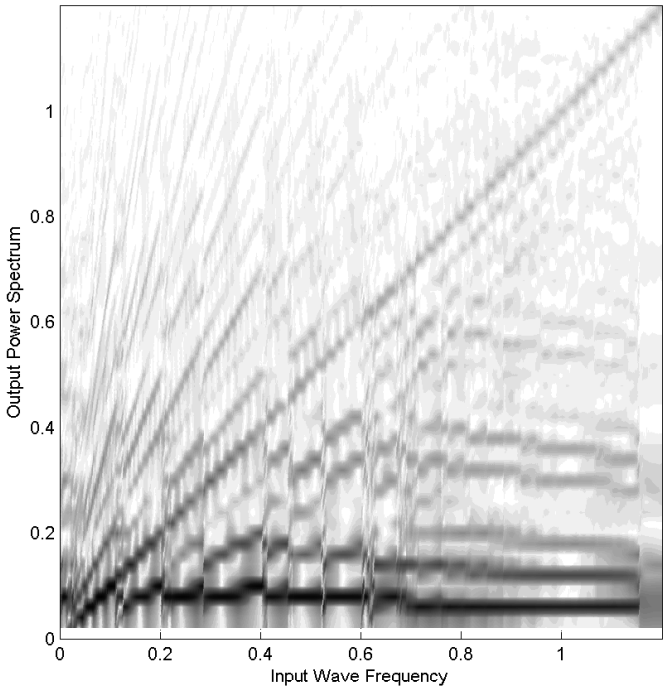
(c) $A = 1.5$


Output Power Spectrum
Input Wave Frequency
0
0.2
0.4
0.6
0.8
1

(d) $A = 2.0$



(e) $A = 2.5$



(f) $A = 3.0$

Figure 4.5: Fourier Analysis of System Response to Sinusoidal Forcing. The frequency components are given in units of inverse time units.

These figures display several interesting patterns. The most striking contrast between the Fourier plots is in the apparent linearity in frequency in Figure 4.5(a). This is further evidence of the threshold phenomena discussed previously. For any signal of small input amplitude (i.e., 0.5 or less), there is no prevailing excitation triggered, resulting in little response. Instead, the system passes through a potential signal of equal frequency and small amplitude. As a consequence, small diffusive signals may be safely passed over as they are incapable of triggering excitations in the system. Additionally, we note that the output of the FitzHugh-Nagumo system tends to have a perceptible direct harmonic of the input across all amplitudes.

Turning our attention to the other plots, we note that for a given amplitude, there is a cutoff input frequency (approximately 0.4 in Figure 4.5(c)) past which the system displays a qualitatively different response. As with the impulse trains in Section 4.2.1, when driven past this frequency, the system undergoes a single excitation cycle in normal fashion. However, following this, the input holds the gating variable at a near-constant value above equilibrium, preventing any further excitations. A second, higher cutoff frequency also exists (approximately 0.85 in Figure 4.5(b), extending beyond the plot ranges in others), above which the single excitation cycle is also inhibited, resulting in an essentially null response. In the remaining frequency regions, the system exhibits a normal periodic excitation caused by the sinusoidal input.

4.4 Forcing by Characteristic Waves

Having examined the FitzHugh-Nagumo system's response to various inputs, we proceed to an issue that will be directly relevant in simulating complex cell networks: how the system responds to inputs of its own characteristic shape. The particular shape used was determined by inserting a finite forcing function (e.g, a single δ impulse) and then passing the output back through the system several times. In applying such

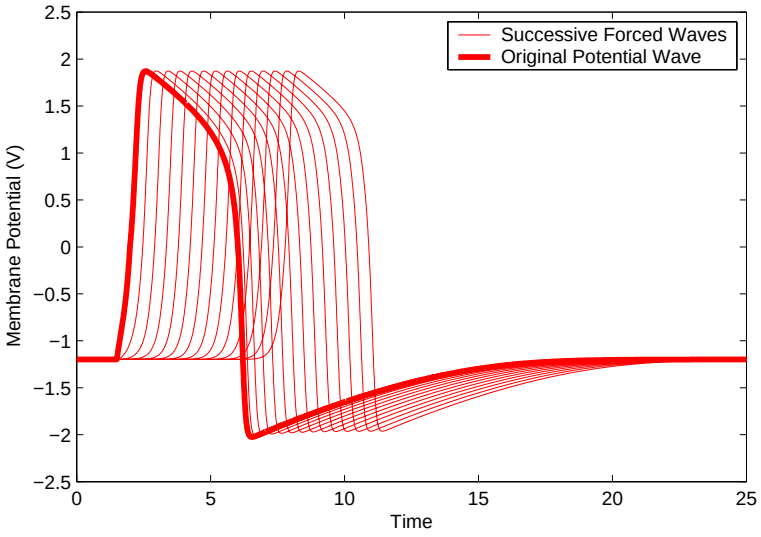


Figure 4.6: System Response to Forcing by Characteristic Wave

potential waves, we must make a minor change in methodology so that the forced cell system is not disturbed when the forcing wave is at equilibrium. This is consistent with the reasonable notion that no net diffusion would occur between adjoining cells at the same potential. Therefore, we consider the forcing function in this case to be $V_{shape} - V_{current}$, in which V_{shape} is a stored FitzHugh-Nagumo excitation wave, and $V_{current}$ is the cell receiving the diffusive input. Figure 4.6 presents a base shape that is used as the input to a second cell system. This pattern continues using the newly generated waveform as the input to the next cell.

The immediately striking feature of this plot is the similarity between successive waveforms. In fact, the excitation shape of the FitzHugh-Nagumo system is an eigenfunction of the system, and a cell that receives a diffusive input in that form will respond with an identical waveform, including a timeshift corresponding to the propagation time. Here, the propagation delay between successive excitation waves is approximately 0.439 time units. In this situation, the forcing shape is a predetermined function; it is not subject to the effects of the cell system being forced. However, in practical multi-cell simulations, this one-sided relationship will not hold,

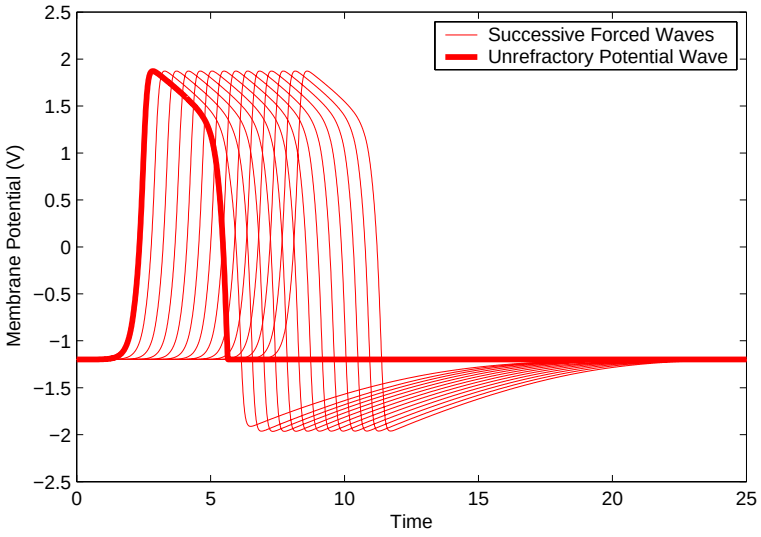


Figure 4.7: System Response to Forcing by Modified Characteristic Wave

and the diffusive connections will be more complex as individual cells have multiple neighbors.

We also note here that the membrane potential through the refractory period has little effect on the action potential of successive cells. If a cell system is forced by a FitzHugh-Nagumo shape as before, but with the membrane potential set to equilibrium for all time after it crosses equilibrium following an excitation, only the first cell will show any perceptible change in response. Furthermore, this small change "filters" as the input is passed through additional cells and is eliminated within two cells of its introduction. Figure 4.7 shows the results of such a process. This result, along with that of the previous characteristic shape forcing, suggests that the FitzHugh-Nagumo excitation shape is extremely robust and will tend to re-establish itself in normal situations.

4.5 Experimental Setups

Through the results of the previous simulations, we can reach a clear conclusion of the relative stability of the FitzHugh-Nagumo system. Its dynamics enforce clearly defined boundaries with respect to frequency of excitation and magnitudes of diffusive inputs, and it responds in a predictable manner to inputs of similar shape to its own excitation waves. Additionally, it has a filtering quality that tends to produce normal excitation cycles from reasonable but abnormal inputs. With a thorough understanding of the responses of the FitzHugh-Nagumo system to various diffusive inputs, we have the tools to establish simulations of multi-cell networks, in which action potentials will propagate through neighboring cells to create electrical waves in the spatial dimensions of the simulation.

Chapter 5

Multi–Cell Simulation Experiments

To progress toward the goal of a full-fledged, computationally feasible simulation of the human heart, it is necessary to construct multi-cell simulations that can be used to examine the interactions between cells. The essential component in such simulations is the diffusion term, for which we approximate $\frac{\partial^2 V}{\partial x^2}$ in Equation (4.1) with the numerical term $(V_{n+1} - V_n) - (V_n - V_{n-1}) = V_{n+1} - 2V_n + V_{n-1}$. Therein, V_n is the membrane potential of a particular cell, and V_{n+1} and V_{n-1} are the membrane potentials of the adjacent cells. From these simulations, we can gain an understanding of the dynamics of the FitzHugh-Nagumo system in an interconnected network environment and examine the types of waveforms that can be produced.

In this chapter, we set up one-dimensional simulations with cells in a line and ring arrangement and a two-dimensional simulation with cells on a uniform grid. Using these simulations, we establish the basic wave patterns and dynamics of the FitzHugh-Nagumo model in the multi-cell environment. Following these simulation studies, we pose an algorithm for speeding their computation times.

5.1 One-Dimensional Simulation Cases

We construct three main types of multi-cell simulations, two of which are one-dimensional and one of which is two-dimensional. Initially, all cells in a simulation are at a resting equilibrium. An action potential is initiated by providing an external stimulus to one or more cells, which in turn propagates throughout the system.

5.1.1 Cell Line

The simplest multi-cell simulation involves a linear connection of cells, which allows for a one-dimensional propagation of electrical potential waves. This simulation is similar in nature to the forcing by characteristic shapes examined in Chapter 4, but we introduce the additional complexity that each cell affects the cells from which it receives input. Based on this addition of feedback between the cells, we expect to see subtly different results in terms of the time shift associated with propagation and the exact waveform shapes that result.

By stimulating the first cell in a line and following the resulting wave, we can characterize the new wave propagation speed by plotting the excitation peak times by the index of the cell. This is shown in Figure 5.1. A linear fit to this data shows that the propagation delay associated with passing an excitation cycle from one cell to the next of a line wave in a multi-cell situation is raised from 0.439 time units (as a forcing function, as in Chapter 4) to 0.591 time units. The waveforms produced in this situation are shown in Figure 5.2. From a qualitative standpoint, the waveforms are quite similar, but the peak and trough in the original, forced cell is noticeably sharper. This is an effect of the interconnection with the previous cell; when that peak time is reached in a cell in the cell line, the previous cell has passed its peak, and is thus exerting a downward pull on the cell. This reduces the magnitude of the peak, and in the same way reduces the sharpness of the trough.

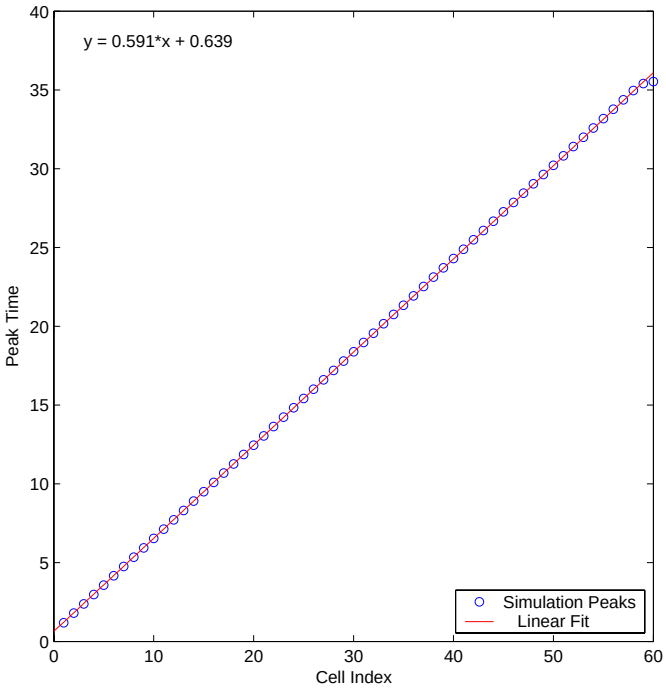


Figure 5.1: Propagation Speed of Waves in the Cell Line

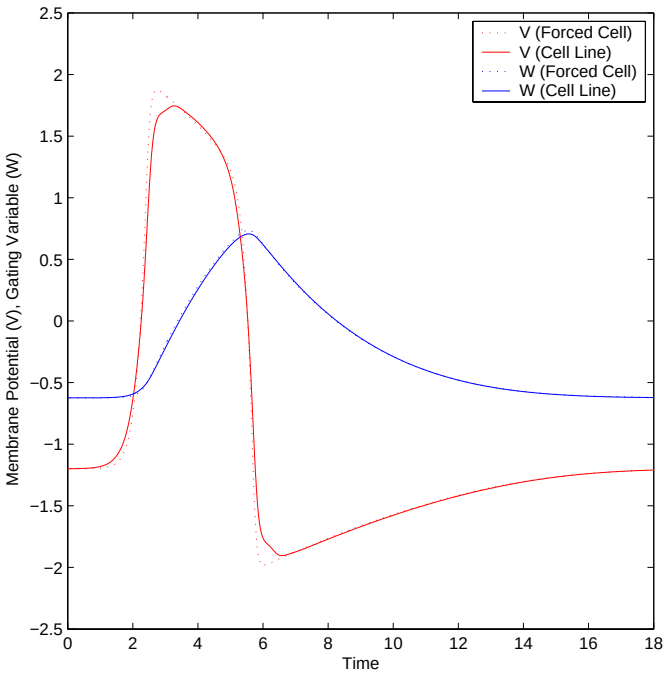


Figure 5.2: Comparison between Forced Cell and Cell Line Waveforms

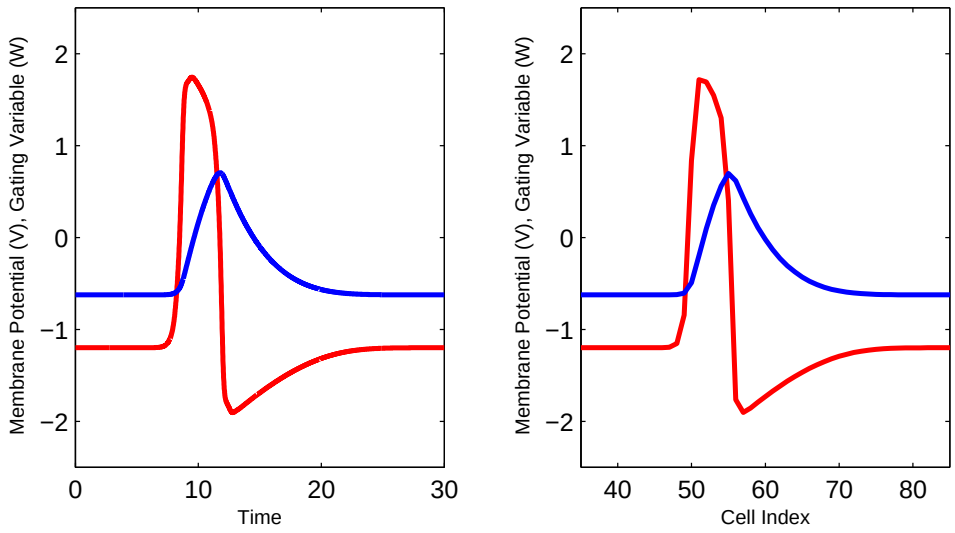


Figure 5.3: Comparison of Individual Cell Time Wave (left) versus Cell Line Spatial Wave (right)

Interestingly, we also observe a striking similarity between the time-varying potential waveform of a single cell and the spatially-varying waveform of the cell line simulation as a whole in Figure 5.3. Aside from the discrete nature of the spatial wave (since it is built up over particular cells in the line), the waveforms are related by the propagation delay. Thus, the networked FitzHugh-Nagumo system produces excitation waves that are related to the excitation waves in particular cells by a factor of 0.591 time units per cell: $V_i(t) = V_{i+1}(t + 0.591)$.

One final aspect of wave propagation can be seen in the cell line setup: model response to wave collisions. As shown in Figure 5.4, mutually incident potential waves annihilate in the FitzHugh-Nagumo model. This is a consequence of the symmetry of the incident waves. To understand this phenomenon, we can consider two approaching waves far away from each other. Due to the dynamics of the FitzHugh-Nagumo system, these waves will quickly assume a standard shape and will be moving with a particular, constant speed. Therefore, there will be some point in space at which these waves will meet, and around which they will have mirror symmetry. Cells on either side of this point will receive no stimulus from each other, as they are always at

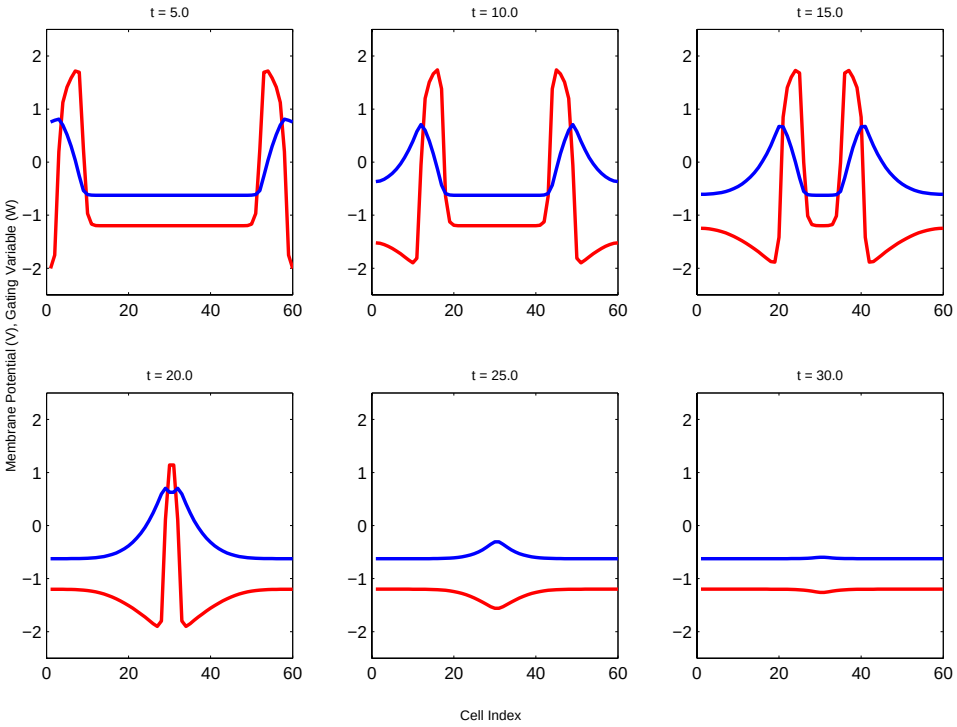


Figure 5.4: Time Snapshots of Colliding Waves in the Cell Line

the same state. Thus, they are unable to pass the excitation wave any further in its current direction of travel, and it dies out at the conclusion of its normal expression.

5.1.2 Cell Ring

We can define a second simulation analogous to the cell line, with the distinction that the first and last cell in the line will be neighbors. This simulation can produce continuing waves around the ring since they can propagate back to the start of the line upon reaching its end. Additional considerations must be made to prevent the initial stimulus that begins the wave propagation from moving in both directions around the ring and annihilating at the opposing end. We prevent this occurrence by selectively severing the first/last connection while initial stimulus is applied and the propagating wave has not completely passed away from the boundary.

There exists a lower bound on the circumference of cells in a ring that will allow a

stable wave to exist. Below this bound, a wave passing one point on the ring will make a complete revolution through the ring to the cell at that point before the cell has finished the complete refractory cycle. The wave will then die off as successive cells are less ready to accept excitations. We found this bound to be at a circumference of 18 cells in our simulations. Considering the previously determined propagation speed of 0.591 time units per cell, the time for a wave to make one revolution of the ring would be 10.64 time units. This number roughly corresponds to the time width of the gating variable (W) excitation wave, which inhibits further excitation waves while above equilibrium. For rings of circumference 18 or more, the travel time of the wave around the ring provides sufficient time for the cells to reach or be sufficiently close to equilibrium to allow re-excitation at the wave's return.

Referring back to the numerical diffusion term $(V_{n+1} - V_n) - (V_n - V_{n-1})$, we note that the two grouped potential differences can be thought of as two currents with an intermediate resistance of unity. These currents flow between each pair of connected cells in the simulation, producing moving potential waves. In the cell ring simulation, there is a clearly visible, perpetual wave motion in one direction only. However, if the current term associated with the opposite direction of wave motion is dropped, there is a significant change in the propagation speed. In actuality, the propagation delay returns to the value of 0.439 time units found previously, since this establishes a chain of characteristic shape forced cells like that investigated in Chapter 4. Figure 5.5 shows the difference in waveforms generated in one cell in a circumference 50 ring, where one simulation uses the normal current terms, and one drops the term corresponding to the opposing direction of wave motion.

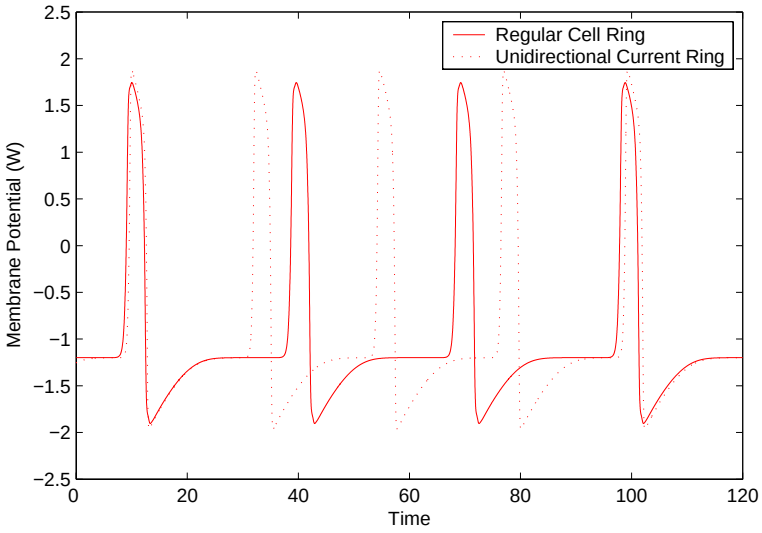


Figure 5.5: Timing Influence of Bi-Directional Current

5.2 Two-Dimensional Simulation Case

When we make the transition from one-dimensional to two-dimensional simulation structures, we inherit an additional complexity in dynamics due to the operation of the gating variable. In previous one-dimensional simulations, we measured the role of the gating variable in regulating the rate of excitation cycles, but this effect was not responsible for any qualitative behavioral shifts. Rather, it provided the difference between a stable, continuous excitation and one of finite duration that would damp out of the system. In two dimensions, however, membrane potential waves can approach a certain cell from multiple directions, creating new and interesting effects like the spiral wave, which will be discussed in the following section.

5.2.1 Wave Types

In addition to the line waves typically generated in the cell line and ring simulations, the cell grid simulation has the unique property of being able to develop expanding circular waves and self-sustaining spiral waves.

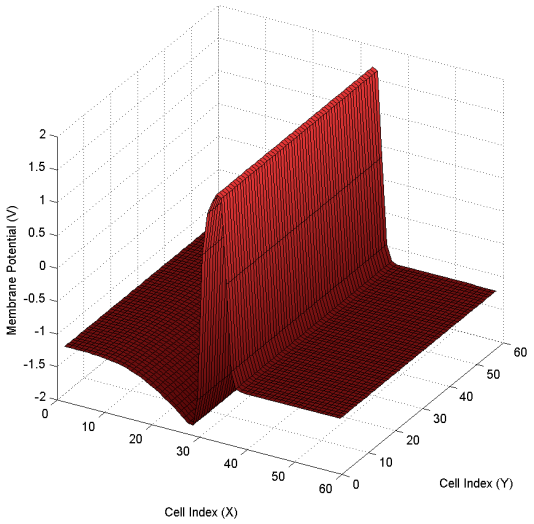


Figure 5.6: Linear Wave Propagation Pattern (at Time $t = 20.0$). The wave is produced by exciting cells along the $x = 1$ line of cells.

Line Wave

As in the line and ring simulations, it is possible to generate a linearly-propagating wave in two dimensions. This situation is analogous to the line simulation case, since such line waves occur when initiated by a uniform stimulus across the length of an edge of the simulation grid. Figure 5.6 illustrates the propagation of a line wave through a square (60-by-60) simulation grid.

Circular Wave

In two dimensions, it is more common for waves to propagate with circular or otherwise curved wavefronts. Due to the constant propagation speed associated with the FitzHugh-Nagumo system, a localized stimulus away from the simulation boundary will trigger an outwardly-moving wave that expands in every direction with the stimulus location as its center. Figure 5.7 shows the typical, regular circular wave. In general, this effect influences most moving wavefronts to create curvature. Figure 5.8 demonstrates the tendency of waves to build curved wavefronts during their motion.

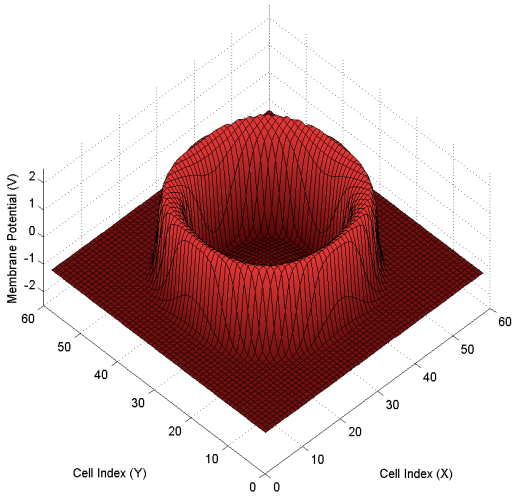


Figure 5.7: Circular Wave Propagation Pattern at $t = 14.0$. The simulation uses a single initial stimulus to a centered cell (29,29) of a 60-by-60-cell grid to initiate the circular wave.

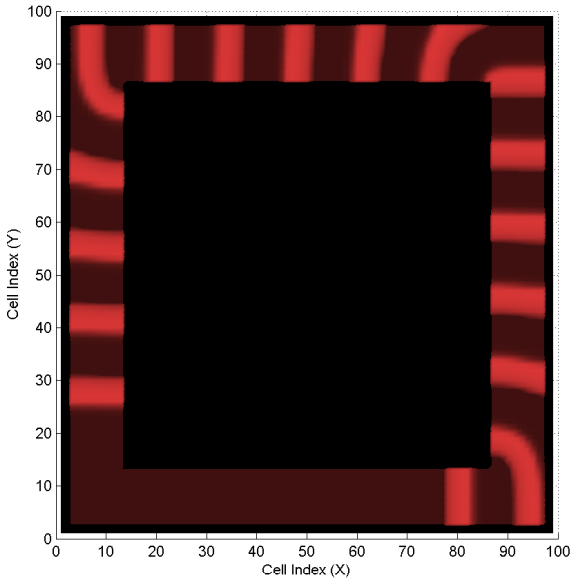


Figure 5.8: Circular Wave Tendency in a Channel Environment. An initial stimulus along the width of the lower channel ($x = 49$) produces two oppositely traveling waves, one of which is eliminated from the system by artificially setting its excited cells to equilibrium. The remaining wave, which is shown at 17 different times, propagates in a counterclockwise direction as a line wave, while in the edges between channels it builds into a circular wave.

Spiral Wave

In select situations, it is possible for heart tissue simulations to form a wave pattern that resembles a spiral and has a self-reinforcing property. The traditional wave patterns mentioned previously tend to propagate through and die out of a system over time (excluding Figure 5.8, in which a channel is specifically designed to hold the wave). By contrast, the spiral wave rotates around a semi-stationary center that moves in a flower-shaped pattern [11, 26] and reinforces itself throughout its motion.

In FitzHugh-Nagumo simulations, spiral waves do not occur in the natural course of stimulus and propagation. Rather, the system must be altered (i.e., by resetting excited sections to equilibrium potential) or repeatedly stimulated (further than just the initial stimulus) during its operation to produce such waves. To produce spiral waves by repeated excitation requires an initial stimulation and a second, later stimulation in the wake of the propagating wave. If the second stimulation happens late enough to create a second excitation, it will initiate a wave that will experience a rotational influence due to the curvature of the previous wave's refractory component. We created such a pairing by initially stimulating a rectangle of cells ($x \in [22, 38], y \in [28, 32]$) and following with a stimulation of a cell line ($x \in [25, 35], y = 37$]) at $t = 11.0$. The stimuli were both rectangular pulses of amplitude 4.0 and duration 2.0. In this situation, a spiral wave pairing like that in Figure 5.9 will arise. It is also possible to initiate a single spiral wave simply by exciting a linear wave and prematurely returning some of its component cells to membrane potential equilibrium. In our case, the spiral wave was generated by initially stimulating one edge (e.g., the $X = 0$ edge) of the simulation and resetting the membrane potentials of half the simulation plane (e.g., the $Y > 30$ half) when the wave is near the center of the simulation grid (e.g., at $t = 18.0$). The asymmetry between membrane potentials in the still-excited cells and the newly-relaxed ones allows the remaining refractory wave to exert a rotational influence as before. Such a lone spiral wave is shown in Figures 5.10 and 5.11.

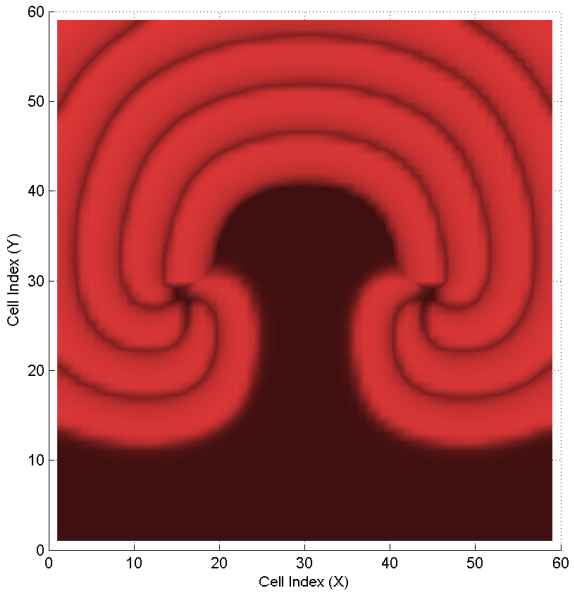


Figure 5.9: Superimposed Rotation Pattern of the Spiral Wave Pairing at Successive Times, Separated by Timesteps of $\Delta t = 8.0$

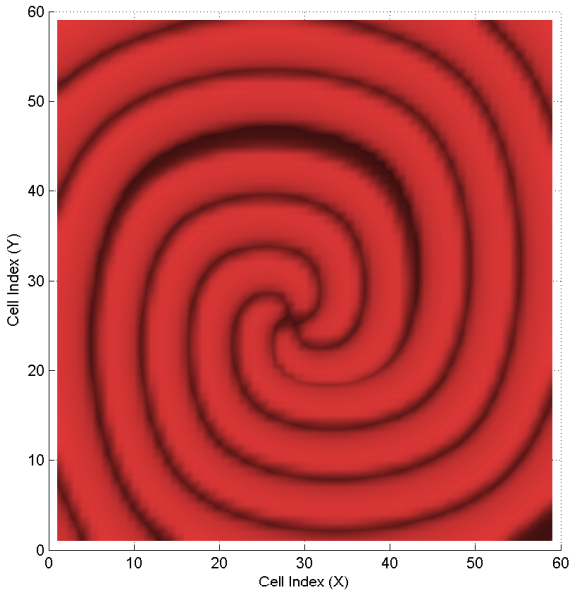


Figure 5.10: Superimposed Rotation Pattern of the Single Spiral Wave at Successive Times, Separated by Timesteps of $\Delta t = 9.0$

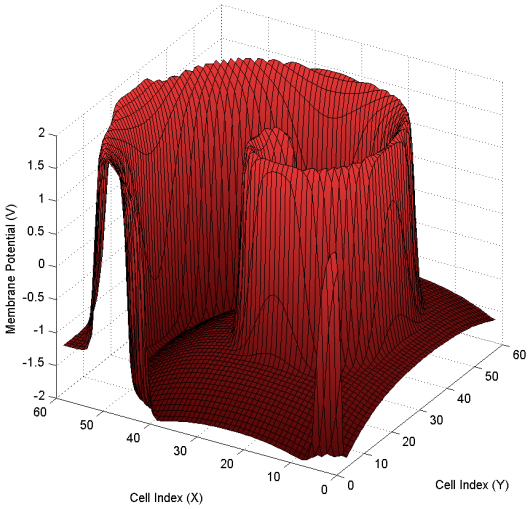


Figure 5.11: The Single Spiral Wave at Time $t = 48.0$

One aspect of note in the dynamics of the spiral wave is the abnormal excitation behavior of the wave rotational center. Though the self-sustaining nature of the wave alone creates an abnormal action potential pattern on the tissue level, the individual cell that coincides with the center of the spiral wave undergoes abnormal excitation cycles. These cycles, which resemble a "stuttering" of the action potential as it tries to spike, last for several periods of normal excitation until the center moves away from the cell. This pattern, shown in Figure 5.12, holds the cell nearby its normal equilibrium, but still prevents normal excitation cycles.

5.3 Activity Tracking

Since the FitzHugh-Nagumo system has a stable, attractive equilibrium point, many cells in a simulation will be implicitly inactive, and hence it is wasteful to calculate their unchanging membrane potentials and small propagation currents. Instead, we can assume that cells near equilibrium (for instance, both V and W are within 0.001 of the equilibrium values $(V_0, W_0) = (-1.1994, -0.6243)$) can be set to equilibrium and

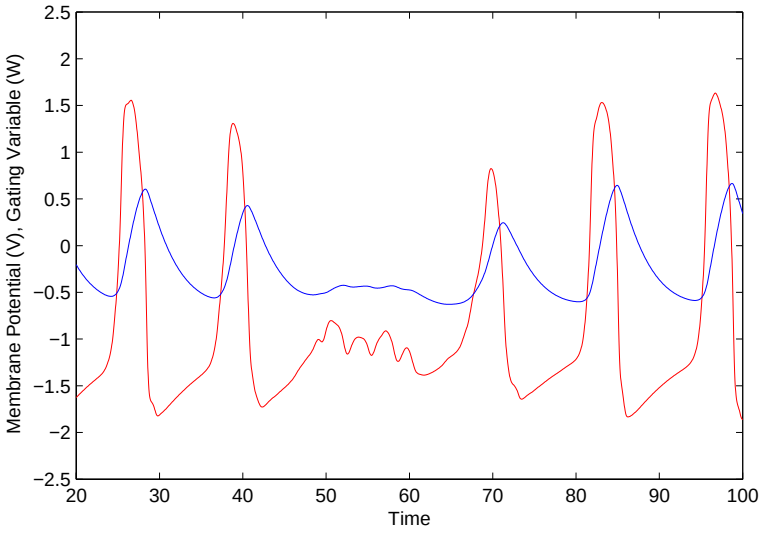


Figure 5.12: Deviated Excitation Behavior Associated with the Spiral Wave Center. The (25,29) cell position is shown.

not updated. It is possible to use a simple list data structure to track the activation of individual cells in order to avoid these calculations. Further, the mechanism of wave propagation ensures that only direct neighbors of activated cells can be activated in the future. Therefore, two lists that track the currently active set of cells and the set of neighbors of these cells will completely describe the subset of a cell network that may be active in the successive timestep. To streamline the process of determining cell neighbors and to prevent cells adjoining multiple active cells from being added to the active list twice, we also retain a matrix structure that marks the active cells in the two-dimensional grid. Provided that the computational overhead of the lists themselves is small, this measure only results in improvements in computation time required for a simulation. The algorithm for tracking cell activity is given as follows:

```
ActivityTrack(ActiveList)
{
   FOR EACH cell_i ∈ ActiveList,
   {
      Model.simulate(cell_i);
```

```
      FOR EACH cell_j ∈ cell_i.Neighbors,
         IF cell_j ∉ NeighborList & cell_j ∉ ActiveList,
            NeighborList.add_cell(cell_j);
   }

   FOR EACH cell_i ∈ ActiveList,
      FOR EACH cell_j ∈ cell_i.Neighbors,
         Model.calculate_diffusion(cell_i,cellj);

   FOR EACH cell_i ∈ NeighborList,
      FOR EACH cell_j ∈ cell_i.Neighbors,
         IF cell_j ∈ ActiveList,
            Model.calculate_diffusion(cell_i,cellj);

   FOR EACH cell_i ∈ ActiveList,
      IF cell_i.equilibrium_state() = TRUE,
         ActiveList.remove_cell(cell_i);

   FOR EACH cell_i ∈ NeighborList,
   {
      IF cell_i.equilibrium_state() = FALSE,
         ActiveList.add_cell(cell_i);
      NeighborList.remove_cell(cell_i);
   }
}
```

To test the viability of this algorithm, we used four benchmarking simulation situations: (1) a 200-cell line, stimulated by the first cell, simulated for 120 time units; (2) a 100-by-100 cell network, stimulated by the center cell, simulated for 30 time units; (3) a 100-by-100 cell network, stimulated at two cells on the diagonal, simulated for 30 time units; and (4) a 60-by-60 cell network, stimulated to produce a spiral wave pairing, simulated for 60 time units. The averaged results of three trials for each situation in both benchmark mode and optimized by the activity list are presented in Table 5.1.

From the time data of simulations run with and without the optimization, it is apparent that the improvement in computation time is dependant on the relative level of activity in the system. This conclusion agrees with the idea of the optimization: when very few cells are active, there is relatively little to calculate, but when a large

portion of cells are active, there is no room to save on calculations. This is apparent from the spiral wave case; most cells are kept active throughout the entire simulation due to the self-reinforcing nature of the wave, so very few cells are skipped by the algorithm. This creates a negligible increase in computation time corresponding to the overhead associated with initializing and maintaining the lists. In the other cases, the activity lists ensure that cells reaching equilibrium after the passage of a wave are again removed from the set of updated cells. This reduces the computation time proportional to the spatial size of the propagating wave.

To further validate this algorithm, we compare the absolute error between the full-blown simulation and the activity list method in both the colliding circular wave case (Figure 5.13) and the dual spiral wave case (Figure 5.14). These plots are time snapshots of the simulations at $t = 40.0$, in order to ensure that any error in propagation speed would be recognizable. Both plots show that the error magnitudes are small, with a maximum error of 0.075 in the membrane potential and 0.008 in the gating variable. The sum of squared errors are similarly small (in the dual circular wave case, 0.53 for the V plot and 0.02 for the W plot). Figure 5.15 shows the change in sum of squared errors over time for the dual circular wave case, which declines from a maximum when the waves begin to vanish past the outside of the simulation grid. Compared with the membrane potential excitation magnitude of approximately 2.0 and the gating variable excitation magnitude of approximately 1.0, all errors are negligible.

Table 5.1: Activity List Optimization Speedups

	Linear	Grid(1)	Grid(2)	Spiral
Full-Blown Simulation	2.94 s	11.89 s	11.87 s	8.75 s
Activity List Method	0.32 s	4.84 s	6.67 s	9.48 s
Time Savings	89.2%	53.9%	43.8%	-8.3%

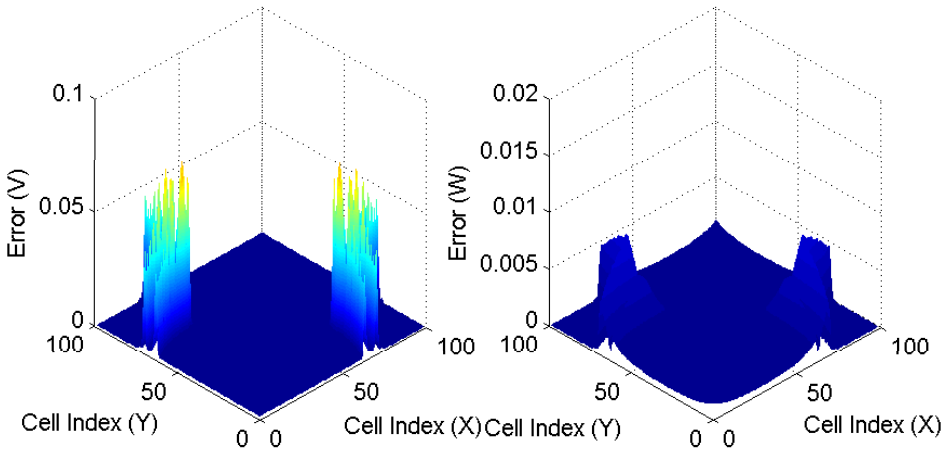


Figure 5.13: Activity List Optimization Error in the Dual Circular Wave Case (for Time $t = 40.0$)

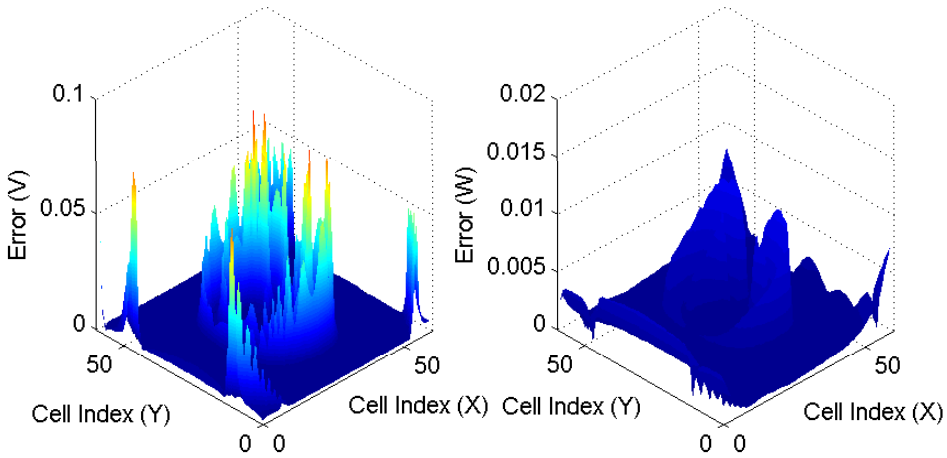


Figure 5.14: Activity List Optimization Error in the Dual Spiral Wave Case (for Time $t = 40.0$)

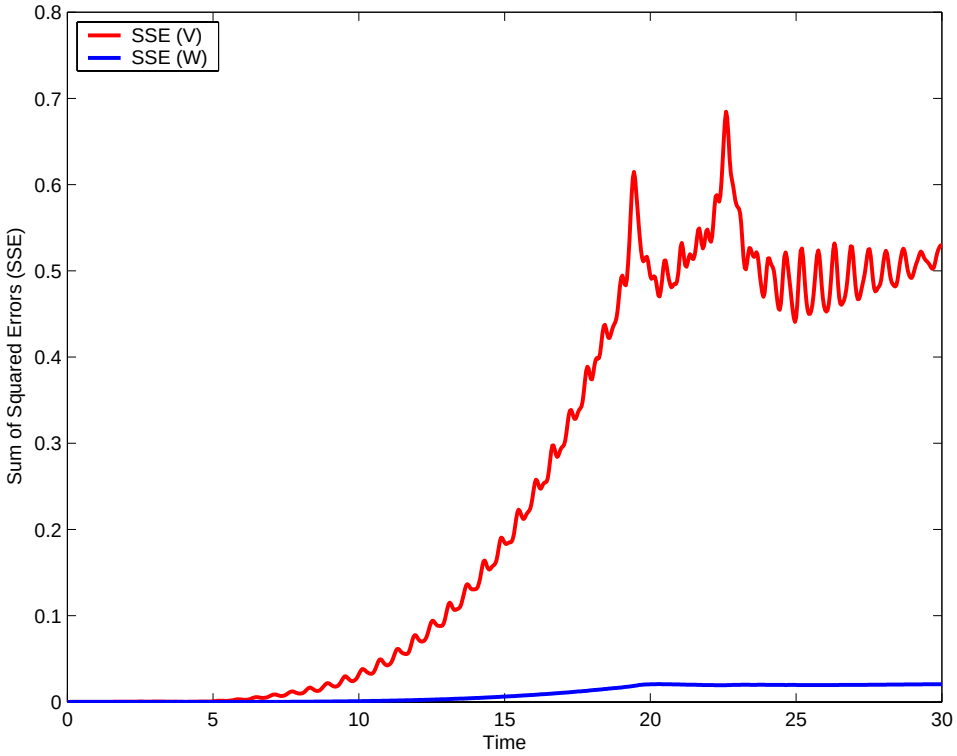


Figure 5.15: Sum of Squared Errors over Time for the Dual Circular Wave Case of the Activity List Optimization

Chapter 6

Conclusion

6.1 Results and Contributions

In this thesis, we have examined the dynamics of the FitzHugh-Nagumo model in both the single-cell and multiple-cell settings. After defining the key aspects of the model's behavior in each setting, we proposed methods of optimizing the simulations associated with cardiac excitation. In the single-cell setting, these methods were in the form of interpolation techniques (nearest neighbor and locally-weighted regression) that could perform input/output mapping operations based on offline sampled data. In the multiple-cell setting, we used the idea that the system often contains a large degree of inactivity to advance a list-based method of optimization.

In **Chapter 2** we gave an in-depth analysis of the properties of the FitzHugh-Nagumo equations, a simple, qualitative system that models the electrical activation of a single cardiac cell. We demonstrated the technique of phase plane analysis and identified the four major states of the excitation cycle. After characterizing the effects of each of the system's three parameters on the shapes and timings of the output waveforms, we described the conditions that create spontaneous periodicity. Finally, we introduced the notion that the system produces characteristic shapes that retain

many common properties across varied initial values and parameters.

Chapter 3 built upon the analysis of the single-cell model by first proposing a method of approximating the system with a linear model. After showing that the linear model was not capable of thoroughly capturing the features of the true FitzHugh-Nagumo system, we advanced the idea of sampling and reconstructing the input/output mapping of the system with two interpolation methods: nearest neighbor and locally weighted regression. We examined the system's nonlinear mapping in the context of these two algorithms and compared them in depth. We also extended the dimensionality of the mapping to three by adding time with the two variables of the system. These ideas allow both large timestep advancements of the system and variable-step calculations, so that it can be "skipped" forward or specified at selective, smaller timesteps.

We introduced the concepts behind diffusion and wave propagation through cardiac tissue in **Chapter 4** and used the idea of forcing from the study of ordinary differential equations to analyze the system's response to stimulus. We provided results detailing the excitation patterns caused by simple stimuli like the constant function and the pulse train and proceeded to show its frequency response using Fourier analysis. To initiate the concept of a tissue simulation composed of multiple, discrete cells, we examined the model's response to stimulus by its own characteristic shapes.

Chapter 5 concludes the development of material by scrutinizing the dynamics of the multiple-cell simulation. We use three simulations (line, ring, and grid) to examine the properties of diffusion, and we proceed to give details about the different types of electrical waves that arise in two-dimensional simulations (linear, circular, and spiral). Finally, we present an algorithm for tracking the activations of individual cells in order to avoid calculations to update inactive portions of the simulation.

6.2 Future Work

The research presented in this thesis investigates the behavior of the FitzHugh-Nagumo model at a basic level. Therefore, there are several potential aspects remaining to be investigated in the quest for a whole-heart model.

6.2.1 Activity List Algorithm Improvements

The activity list algorithm given in this thesis is an idea that can be further developed. One immediate possibility for the algorithm is to include the consideration of "neighbors of neighbors" so that the algorithm would have access to more time-advanced information about possible locations for the wave to propagate. Additionally, it is possible to develop a method for the algorithm to monitor the direction of wave propagation to reduce the number of "wasted" neighbor calculations. The present algorithm is ignorant of the wave propagation direction, so it processes neighbors on both sides of an excitation wave. This results in wasted cycles to track certain neighbors. These optimizations could reduce the computational overhead of the algorithm, while also increasing its accuracy.

6.2.2 Interpolation Methods and Performance

As we noted in Chapter 3, there are significant performance improvements available through the use of innovative data structures and computer methods to improve the speed of the nearest neighbor [5, 19] method and locally-weighted regression [30]. Such improvements are needed before the algorithms can find a place in optimizing practical cell simulations. Also, since both algorithms have identifiable issues with accuracy in the nonlinear maps associated with the FitzHugh-Nagumo model, other algorithms could be investigated to potentially take their place. For example, MATLAB provides several built-in interpolation methods not analyzed here, including

bilinear, cubic spline, and bicubic interpolation. Also, it is qualitatively visible that the FitzHugh-Nagumo system could be approximated by a cubic surface rather than the planar one introduced in Chapter 3. This suggests that such an approximation idea may be a worthwhile exploration for higher-level models.

6.2.3 Complex 3D Cell Networks

Though analogous in nature to two-dimensional simulations, this research has not touched on three-dimensional simulations of multi-cell networks due to complexity and visualization issues. In fact, research has shown that new and qualitatively different wave types can arise in three-dimensional simulations [26] that are not seen in two-dimensional ones. Both increased understanding of cardiac behavior and the goal of whole-heart simulation require three-dimensional multi-cell networks with irregularly-spaced cells.

6.2.4 Model Substitution

The FitzHugh-Nagumo model is one of the simpler available models of excitable cells. More rigorously-developed models provide increased accuracy in certain situations, but also have significantly higher dimensions. The ability to dynamically switch between complex and simple models will be valuable in balancing the real-world accuracy of whole-heart simulation with speed. Fortunately, both the nearest neighbor method and locally weighted regression possess general forms that function for arbitrary input dimension. Additionally, there may even be variables in these higher-order models that can be successfully approximated by a linear or other low-order function to speed calculations. The activity list algorithm will generalize to any model that has a stable equilibrium point. Still, the complexity of such models also ensures that the analysis that must be performed to truly characterize the viability of these techniques will be much more demanding.

6.2.5 Wavelets and Kernel Functions

Due to the existence of characteristic shapes in the FitzHugh-Nagumo model, wavelet theory could be applied to separate the waveforms into components important to wave propagation and those not relevant. In addition, we noted in Chapter 4 that the waveform shape associated with the late refractory period is not important to the prevailing propagation pattern of waves. This further suggests that a decomposition of the waveforms may provide useful information on diffusion phenomena.

Appendix A

MATLAB Code

Nearest Neighbor Code

The nearest neighbor function requires three parameters: `Xq` (the query point), `Xi` (a matrix of sample data), and `Yi` (a matrix of output data associated with the sample data). It returns `Yq`, a copy of the output data associated with the closest sample point in input space.

```
function Yq = GetNN(Xq,Xi,Yi)

[samples dimensions] = size(Xi);

mindist = 100000000;
minindex = 0;

for j=1:samples,
    dist = 0;
    for n=1:dimensions,
        dist = dist + (Xq(n) - Xi(j,n))^2;
    end
    if dist < mindist,
        mindist = dist;
        minindex = j;
    end
end

Yq = Yi(minindex);
```

Locally Weighted Regression Code

The code for locally weighted regression is divided into two parts: the main algorithm (locally weighted regression) and the support algorithm (cross validation).

Locally Weighted Regression

The main locally weighted regression algorithm uses the same parameters for query point and sample data as the nearest neighbor implementation, but it requires several additional parameters. These include `h` (the neighborhood sizing parameter) and `XiVar` (the variance of the sample data in each input dimension). The final parameter, `exclude`, is provided so the algorithm can ignore a particular sample point for compatibility with the cross validation algorithm. The algorithm returns `Yq`, its prediction for the output of the query point.

```
function Yq = GetLWR(h,Xq,Xi,XiVar,Yi,exclude)

[samples dimensions] = size(Xi);

w = zeros(samples,1);
for n=1:dimensions,
    w = w + ((Xi(:,n)-Xq(n)).^2)/XiVar(n);
end

if h == 0,
    k = 1:samples;
else
    k = find(w < (-2/h)*log(0.001));
end

w = exp((-h/2)*w(k));

if exclude ~= 0,
    exclude = find(k == exclude);
    w(exclude) = 0;
end

X = ones(length(k),dimensions+1);
for n=1:dimensions,
    X(:,n) = Xi(k,n)-Xq(n);
end
```

```
W = repmat(w',dimensions+1,1);
Y = Yi(k);

XtW = X'.*W;
B = (XtW*X)\(XtW*Y);
Yq = B(dimensions+1);
```

Cross Validation

The cross validation algorithm cycles through each sample point and computes the error between the sample point's output and the locally weighted regression prediction using a sample set without that sample point. It returns `sserror`, the sum of squared errors for the particular `h` value being tested.

```
function sserror = CrossValidate(h,Xi,XiVar,Yi)

[samples dimensions] = size(Xi);

sserror = 0;
for m=1:samples,
    sserror = sserror + (GetLWR(h,Xi(m,:),Xi,XiVar,Yi,m) - Yi(m))^2;
end
```

Activity List Code

The code below implements the activity list algorithm for the dual circular wave case (see Chapter 5). It produces a pair of three-dimensional matrices containing the complete membrane potential (V) and gating variable (W) data over the simulation time for each cell. Note that the lists (`ActiveXList`, `ActiveYList`, `NeighborXList`, `NeighborYList`) are an approximation of the true idea of a list for the MATLAB environment. Each list is composed of an n-by-1 matrix, where n is the total cell count. The true data length of this matrix/list structure is updated as data is stored to, or removed from, the list. New data is added by writing at the element after the

current length of the list, and data can be removed by overwriting it with the final list element and reducing the list length by one. The matrix (`ActiveNeighborMatrix`) keeps a redundant set of data and is used to facilitate checks of whether a particular cell is already contained within the active or neighbor lists.

```
TimeLength = 30.0;
TimeStep = 0.04;
HalfTimeStep = TimeStep/2;
Time = 0:TimeStep:TimeLength;

DiffusionX = 1;
DiffusionY = 1;

Beta = 0.7;
Epsilon = 0.2;
Gamma = 0.8;

SampleCount = length(Time);
CellWidth = 100;
CellLength = 100;

ActiveXList = zeros(1,CellWidth*CellLength);
ActiveYList = zeros(1,CellWidth*CellLength);
ActiveListLength = 0;
NeighborXList = zeros(1,CellWidth*CellLength);
NeighborYList = zeros(1,CellWidth*CellLength);
NeighborListLength = 0;
ActiveNeighborMatrix = zeros(CellWidth,CellLength);

ForcingPattern = zeros(CellWidth,CellLength);
ForcingPattern(30,30) = 1;
ForcingPattern(71,71) = 1;

ActiveXList(1) = 30;
ActiveYList(1) = 30;
ActiveXList(2) = 71;
ActiveYList(2) = 71;
ActiveListLength = 2;

ActiveNeighborMatrix(30,30) = 2;
ActiveNeighborMatrix(71,71) = 2;

V0 = roots([-1/3 0 1-1/Gamma -Beta/Gamma]);
V0 = V0(3);
W0 = (Beta + V0)/Gamma;

V = V0*ones(CellWidth,CellLength,SampleCount);
W = W0*ones(CellWidth,CellLength,SampleCount);

Threshold = 0.001;
```

```
V0Low = V0 - Threshold;
V0High = V0 + Threshold;
W0Low = W0 - Threshold;
W0High = W0 + Threshold;

for Sample = 2:SampleCount,
    % Fill the neighbor list by the active list and process
    % differential equation for all active cells
    for i = 1:ActiveListLength,
        CellX = ActiveXList(i);
        CellY = ActiveYList(i);

        if CellX ~= 1,
            if ~ActiveNeighborMatrix(CellX-1,CellY),
                NeighborListLength = NeighborListLength + 1;
                NeighborXList(NeighborListLength) = CellX-1;
                NeighborYList(NeighborListLength) = CellY;
                ActiveNeighborMatrix(CellX-1,CellY) = 1;
            end
        end
        if CellX ~= CellWidth,
            if ~ActiveNeighborMatrix(CellX+1,CellY),
                NeighborListLength = NeighborListLength + 1;
                NeighborXList(NeighborListLength) = CellX+1;
                NeighborYList(NeighborListLength) = CellY;
                ActiveNeighborMatrix(CellX+1,CellY) = 1;
           end
        end
        if CellY ~= 1,
            if ~ActiveNeighborMatrix(CellX,CellY-1),
                NeighborListLength = NeighborListLength + 1;
                NeighborXList(NeighborListLength) = CellX;
                NeighborYList(NeighborListLength) = CellY-1;
                ActiveNeighborMatrix(CellX,CellY-1) = 1;
            end
        end
        if CellY ~= CellLength,
            if ~ActiveNeighborMatrix(CellX,CellY+1),
                NeighborListLength = NeighborListLength + 1;
                NeighborXList(NeighborListLength) = CellX;
                NeighborYList(NeighborListLength) = CellY+1;
                ActiveNeighborMatrix(CellX,CellY+1) = 1;
            end
        end

        PreviousV = V(CellX,CellY,Sample-1);
        PreviousW = W(CellX,CellY,Sample-1);
        mV = (PreviousV - (PreviousV^3)/3 - PreviousW)/Epsilon;
        mW = Epsilon*(PreviousV - Gamma*PreviousW + Beta);
        Vtemp = PreviousV + HalfTimeStep*mV;
        Wtemp = PreviousW + HalfTimeStep*mW;
        nV = (Vtemp - (Vtemp^3)/3 - Wtemp)/Epsilon;
        nW = Epsilon*(Vtemp - Gamma*Wtemp + Beta);
        Vtemp = PreviousV + HalfTimeStep*nV;
```

```
    Wtemp = PreviousW + HalfTimeStep*nW;
    qV = (Vtemp - (Vtemp^3)/3 - Wtemp)/Epsilon;
    qW = Epsilon*(Vtemp - Gamma*Wtemp + Beta);
    Vtemp = PreviousV + TimeStep*qV;
    Wtemp = PreviousW + TimeStep*qW;
    pV = (Vtemp - (Vtemp^3)/3 - Wtemp)/Epsilon;
    pW = Epsilon*(Vtemp - Gamma*Wtemp + Beta);

    V(CellX,CellY,Sample) = PreviousV + ...
                                TimeStep*(mV + 2*nV + 2*qV + pV)/6;
    W(CellX,CellY,Sample) = PreviousW + ...
                                TimeStep*(mW + 2*nW + 2*qW + pW)/6;
end

% Process forcing term
for CellX = 1:CellWidth,
    for CellY = 1:CellLength,
        if ForcingPattern(CellX,CellY),
            V(CellX,CellY,Sample) = V(CellX,CellY,Sample) + ...
                           TimeStep*4/(1+exp(16*(Time(Sample)-2)));
        end
    end
end

% Process currents for adjacent cells
I = zeros(CellWidth,CellLength);

for i = 1:ActiveListLength,
    CellX = ActiveXList(i);
    CellY = ActiveYList(i);

    if CellX ~= CellWidth,
        Itemp = DiffusionX*(V(CellX+1,CellY,Sample) - ...
                                    V(CellX,CellY,Sample));
        I(CellX,CellY) = I(CellX,CellY) + Itemp;
        I(CellX+1,CellY) = I(CellX+1,CellY) - Itemp;
    end
    if CellY ~= CellLength,
        Itemp = DiffusionY*(V(CellX,CellY+1,Sample) - ...
                                    V(CellX,CellY,Sample));
        I(CellX,CellY) = I(CellX,CellY) + Itemp;
        I(CellX,CellY+1) = I(CellX,CellY+1) - Itemp;
    end
end
for i = 1:NeighborListLength,
    CellX = NeighborXList(i);
    CellY = NeighborYList(i);

    if CellX ~= CellWidth,
        if ActiveNeighborMatrix(CellX+1,CellY) == 2,
            Itemp = DiffusionX*(V(CellX+1,CellY,Sample) - ...
                                    V(CellX,CellY,Sample));
            I(CellX,CellY) = I(CellX,CellY) + Itemp;
            I(CellX+1,CellY) = I(CellX+1,CellY) - Itemp;
```

```
            end
        end
        if CellY ~= CellLength,
            if ActiveNeighborMatrix(CellX,CellY+1) == 2,
                Itemp = DiffusionY*(V(CellX,CellY+1,Sample) - ...
                                    V(CellX,CellY,Sample));
                I(CellX,CellY) = I(CellX,CellY) + Itemp;
                I(CellX,CellY+1) = I(CellX,CellY+1) - Itemp;
            end
        end
    end

    V(:,:,Sample) = V(:,:,Sample) + TimeStep*I;

    % Move all active cells at equilibrium off the active list
    for i = 1:ActiveListLength,
        CellX = ActiveXList(i);
        CellY = ActiveYList(i);

        if (VOLow < V(CellX,CellY,Sample)) && ...
            (V(CellX,CellY,Sample) < VOHigh) && ...
            (WOLow < W(CellX,CellY,Sample)) && ...
            (W(CellX,CellY,Sample) < WOHigh),
            ActiveNeighborMatrix(CellX,CellY) = 0;
            ActiveXList(i) = ActiveXList(ActiveListLength);
            ActiveYList(i) = ActiveYList(ActiveListLength);
            ActiveListLength = ActiveListLength - 1;
        end
    end
    % Move neighbor cells to the active list or off the neighbor list
    for i = 1:NeighborListLength,
        CellX = NeighborXList(i);
        CellY = NeighborYList(i);

        if (VOLow < V(CellX,CellY,Sample)) && ...
            (V(CellX,CellY,Sample) < VOHigh) && ...
            (WOLow < W(CellX,CellY,Sample)) && ...
            (W(CellX,CellY,Sample) < WOHigh),
            ActiveNeighborMatrix(CellX,CellY) = 0;
        else
            ActiveNeighborMatrix(CellX,CellY) = 2;
            ActiveListLength = ActiveListLength + 1;
            ActiveXList(ActiveListLength) = NeighborXList(i);
            ActiveYList(ActiveListLength) = NeighborYList(i);
        end

        NeighborListLength = 0;
    end
end

clear mV mW nV nW qV qW pV pW Vtemp Wtemp PreviousV PreviousW Itemp ...
                                                    Sample CellX CellY;
```

Appendix B

GUI System

As mentioned in Chapter 1, early results in this thesis were obtained during the design of a GUI system written in C++ using the OpenGL and GLUT libraries. This program contained implementations of a fourth-order Runge-Kutta differential equation solver to simulate the FitzHugh-Nagumo system (in the "FitzHugh-Nagumo" window) given a user-defined parameter set (adjustable using the sliders in the "Control" window). Additionally, it displayed the solutions in the phase plane and provided a sampling-based nearest neighbor and locally weighted regression scheme (in the "Sample" and "Data" windows) to simulate the system as in Chapter 3. Three example screen captures of the GUI system are shown in Figure B.1.

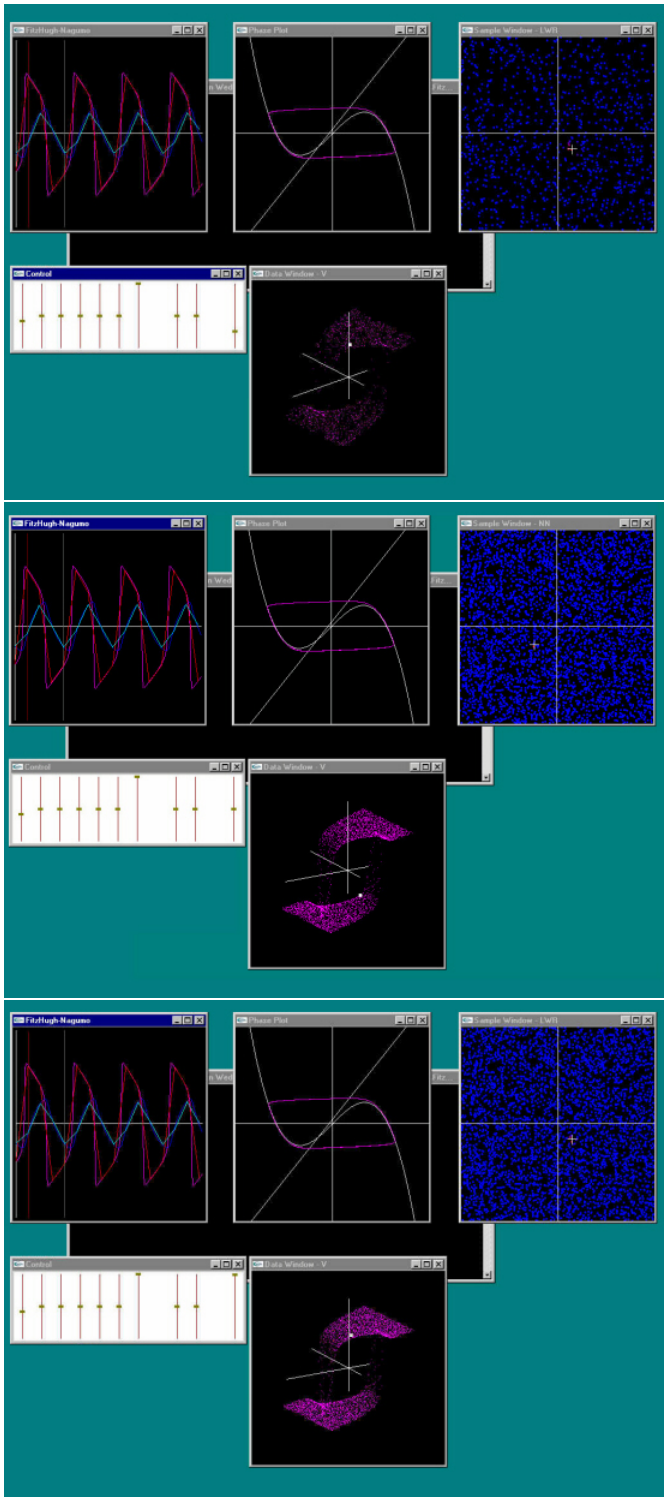


Figure B.1: FitzHugh-Nagumo GUI System

Bibliography

[1] R. Aliev and A. Panfilov. A simple two-variable model of cardiac excitation. *Chaos, Solitons and Fractals*, 7:293–301, 1996.

[2] P. Basser. New currents in electrical stimulation of excitable tissues. *Annual Review of Biomedical Engineering*, 2:377–397, 2000.

[3] P. Blanchard, R. Devaney, and G. Hall. *Differential Equations*. Brooks/Cole, Pacific Grove, CA, 1998.

[4] C. Chen. *Linear System Theory and Design*. Oxford University Press, Oxford, NY, 1999.

[5] M. Curtiss. *Motion Planning and Control using RRTs.* Master's thesis, Department of Electrical Engineering and Computer Science, Case Western Reserve University, Cleveland, OH, USA, 2002.

[6] D. DiFrancesco and D. Noble. A model of cardiac electrical activity incorporating ionic pumps and concentration changes. *Philosophical Transactions: Biological Sciences*, 307:353–398, 1985.

[7] R. FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical Journal*, 1:445–466, 1961.

[8] R. FitzHugh and H. Antosiewicz. Automatic computation of nerve excitation — detailed corrections and additions. *Journal of the Society for Industrial and Applied Mathematics*, 7:447–458, 1959.

[9] R. Hamming. *Numerical Methods for Scientists and Engineers*. Dover Publications, Incorporated, New York, NY, 1986.

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

[11] A. Holden and V. Biktashev. Computational biology of propagation in excitable media models of cardiac tissue. *International Journal of Bioelectromagnetism*, 2, 2000.

[12] P. Hunter, P. Kohl, and D. Noble. Integrative models of the heart: Achievements and limitations. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, 359:1049–1054, 2001.

[13] P. Hunter, A. Pullan, and B. Smaill. Modeling total heart function. *Annual Review of Biomedical Engineering*, 5:147–177, 2003.

[14] J. Keener and J. Sneyd. *Mathematical Physiology*. Springer, New York, NY, 2001.

[15] S. LaValle and M. Branicky. On the relationship between classical grid search and probabilistic roadmaps. *Proceedings of the Workshop on the Algorithmic Foundation of Robotics*, Nice, France, 2002.

[16] C. Luo and Y. Rudy. A dynamic model of the cardiac ventricular action potential: Simulations of ionic currents and concentration changes. *Circulation Research*, 74:1071–1096, 1994.

[17] R. Mazhari. "Are we there yet?!" Cardiac channelopathy and our journey toward computational medicine. *Circulation Research*, 90:842–843, 2002.

[18] R. McAllister, D. Noble, and R. Tsien. Reconstruction of the electrical activity of cardiac Purkinje fibres. *Journal of Physiology*, 251:1–59, 1975.

[19] A. Moore. *Efficient Memory-based Learning for Robot Control*. Ph.D. thesis, University of Cambridge, Cambridge, UK, 1990.

[20] B. Muller-Borer, D. Erdman, and J. Buchanan. Electrical coupling and impulse propagation in anatomically modeled ventricular tissue. *IEEE Transactions on Biomedical Engineering*, 41:445–454, 1994.

[21] J. Nagumo, S. Animoto, and S. Yoshizawa. An active pulse transmission line simulating nerve axon. *Proceedings of the Institute of Radio Engineers*, 50:2061–2070, 1962.

[22] D. Nickerson, N. Smith, and P. Hunter. A model of cardiac cellular electromechanics. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, 359:1159–1172, 2001.

[23] D. Noble. Modeling the heart — from genes to cells to the whole organ. *Science*, 295:1678–1682, 2002.

[24] D. Noble and Y. Rudy. Models of cardiac ventricular action potentials: Iterative interaction between experiment and simulation. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, 359:1127–1142, 2001.

[25] N. Otani. A two-dimensional detailed ion channel model of abnormal cardiac action potential propagation. *Computers in Cardiology*, 25:565–568, 1998.

[26] N. Otani. Computer modeling in cardiac electrophysiology. *Journal of Computational Physics*, 161:21–34, 2000.

[27] N. Otani and D. Allexandre. A fast variable timestep numerical method for excitable systems. Presented at *The 2000 Society for Industrial and Applied Mathematics Annual Meeting*, Rio Grande, Puerto Rico, 2000.

[28] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C*. Cambridge University Press, New York, NY, 1988.

[29] J. Rogers and A. McCulloch. A collocation–Galerkin finite element model of cardiac action potential propagation. *IEEE Transactions on Biomedical Engineering*, 41:743–757, 1994.

[30] S. Schaal, C. Atkeson, and S. Vijayakumar. Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence – Special Issue on Scalable Robotic Applications of Neural Networks*, 17:49–60, 2002.

[31] D. Scollan, A. Holmes, J. Zhang, and R. Winslow. Reconstruction of cardiac ventricular geometry and fiber orientation using magnetic resonance imaging. *Annals of Biomedical Engineering*, 28:934–944, 2000.

[32] K. Simelius, J. Nenonen, R. Hren, and B. Horacek. Anisotropic propagation model of ventricular myocardium. *International Journal of Bioelectromagnetism*, 2, 2000.

[33] B. van der Pol and J. van der Mark. The heartbeat considered as a relaxation oscillator and an electrical model of the heart. *Philosophical Magazine*, 6:763–775, 1928.

[34] Y. Wang, R. Kumar, M. Wagner, R. Wilders, D. Golod, W. Goolsby, and R. Joyner. Electrical interactions between a real ventricular cell and an anisotropic two-dimensional sheet of model cells. *American Journal of Physiology – Heart and Circulatory Physiology*, 278:452–460, 2000.

[35] N. Wedge, M. Branicky, and M. Çavuşoğlu. Computationally efficient cardiac bioelectricity models toward whole-heart simulation. *Proceedings of the IEEE Engineering in Medicine and Biology Society International Conference*, 2004. Submitted.

[36] R. Winslow, D. Scollan, A. Holmes, C. Yung, J. Zhang, and M. Jafri. Electrophysiological modeling of cardiac ventricular function: from cell to organ. *Annual Review of Biomedical Engineering*, 2:119–155, 2000.